Nuevos protocolos

Area de Ingeniería Telemática
http://www.tlm.unavarra.es

HTTP/2

HTTP1.1

- Algunas mejoras en HTTP1.1
  - Conexiones persistentes
  - Entrega de rangos de bytes
  - Pipelining

- Y algunos “hacks” de navegadores y desarrolladores
  - Conexiones en paralelo (limitadas a 5-6 para un servidor)
  - *Domain sharding* (recursos en diferentes dominios)
  - Unión de ficheros pequeños (CSS, Javascript, imágenes)
  - Inline de ficheros con el HTML

HTTP1.1

- Problemas
  - Optimizado para grandes transferencias pero una web contiene muchos recursos pequeños
  - Para mejorar el tiempo de carga hay que reducir el RTT (CDNs), pero esto tiene un límite (“c”)
  - Pipelining no suele emplearse (problemas con proxies y algunos servidores)
  - Aún con pipelining tiene HOL blocking
  - Conexiones en paralelo y domain sharding crean mayor número de sockets, más conexiones en NATs, más DNS
  - Concatenación e inlining rompe la modularidad y entorpece las caches

HTTP/2 (H2)

- RFC 7540 “Hypertext Transfer Protocol Version 2 (HTTP/2)” (BitGo, Google, Mozilla, 2015)
- (Antes SPDY, de Google)
- Binario
  - Ya no es texto, más compacto, más eficiente al procesarlo
  - Se pueden comprimir las cabeceras (evita tener que enviarlas en cada petición o respuesta)
  - Permite multiplexación
- Multiplexación
  - Peticiones y respuestas pueden estar multiplexadas
  - Eso quiere decir por ejemplo que no es necesario completar una respuesta para empezar a enviar otra
  - Trabaja con *streams* dentro de una sola conexión
  - El navegador puede asignarles prioridades en las peticiones
  - Esto elimina el HOL blocking y la necesidad de conexiones en paralelo
- Server Push
  - El servidor puede entregar recursos al cliente aunque no los haya pedido
  - Ahorra la petición (esto se estaba haciendo con el *inlining*)

HTTP/2

- Flow Control
  - Para cada *stream*
  - HTTP/2 no fuerza a un algoritmo para el control de flujo, solo lo soporta
- Negociación
  - Mediante mensajes se puede subir una conexión de HTTP1.1 a HTTP/2

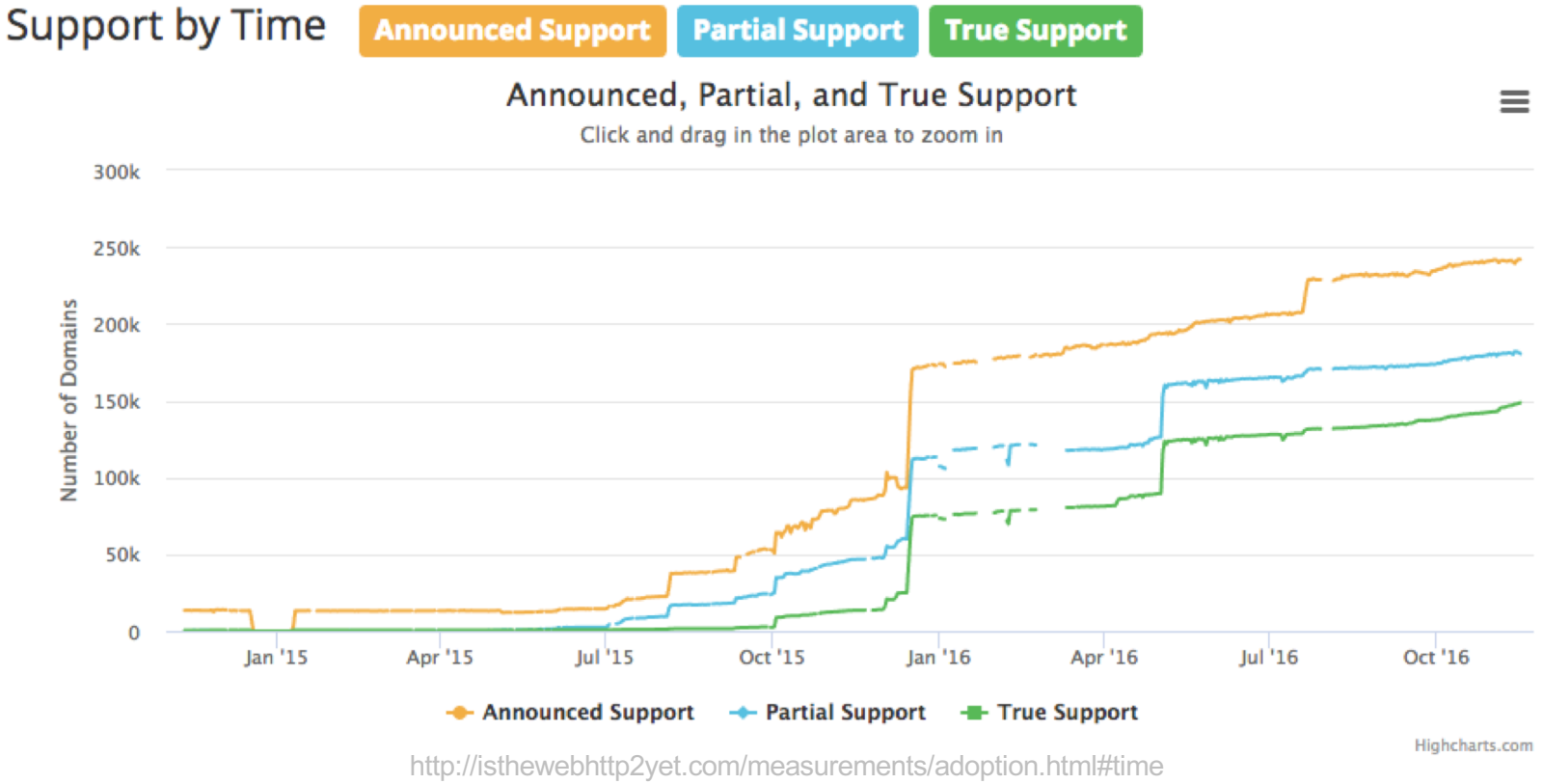
```
GET /page HTTP/1.1
Host: server.example.com
Connection: Upgrade, HTTP2-Settings
Upgrade: HTTP/2.0
HTTP2-Settings: (SETTINGS payload)
```

```
HTTP/1.1 200 OK
Content-length: 243
Content-type: text/html
(... HTTP 1.1 response ...)

 (or)
HTTP/1.1 101 Switching Protocols
Connection: Upgrade
Upgrade: HTTP/2.0
(... HTTP 2.0 response ...)
```

HTTP/2: Status

- RFC aprobada
- Múltiples implementaciones, en navegadores y servidores
- Soportado en IE11, Firefox51, Chrome49, Safari10, Opera43, etc
- Soportado ya por muchas CDNs y hostings (ej: Akamai, Azure CDN, MaxCDN, KeyCDN, CacheFly, CloudFlare, Hawk host, etc)
- Estimaciones de que es empleado en el 25% de websites[1]



http://isthewebhttp2yet.com/measurements/adoption.html#time

[1] https://w3techs.com/technologies/details/ce-http2/all/all (Abril 2018)

MPTCP

Problemas de partida

- La separación entre TCP e IP no es completa
- Una conexión TCP viene asociada a la 5-tupla, lo cual implica estar asociada a las direcciones IP
- Una conexión TCP no puede mantenerse ante el cambio de las direcciones de nivel de red
- Soluciones en capa 3
  - Mobile IP (RFC 5944), HIP (Host Identity Protocol, RFC 4423), Shim6 (Site Multihoming by IPv6 Intermediation, RFC 5533)
  - Ocultan a TCP el cambio de dirección, con lo que ocultan el cambio de camino al control de congestión
- SCTP (Stream Control Transmission Protocol, RFC 4960)
  - Protocolo de nivel de transporte que soporta múltiples direcciones IP por conexión de transporte
  - Despliegue imposible por falta de soporte en NATs (SCTP over UDP?)
  - API diferente para las aplicaciones
- “TCP Extensions for Multipath Operation with Multiple Addresses” (Multipath TCP, RFC 6824, 2013)

Middleboxes

- Descartan paquetes de otros protocolos de transporte
- Modifican cabeceras IP y TCP (NAT, proxy transparente)
- Modifican ventana de control de flujo (control de BW, escalado)
- Modifican números de secuencia (ej: ISN aleatorio)
- Eliminan opciones que no conocen
- Abortan conexiones con opciones que no conocen
- Descartan paquetes si no han visto el inicio de la conexión
- Hacen coalescencia o segmentación de paquetes
- Modifican el flujo de datos (ALGs)

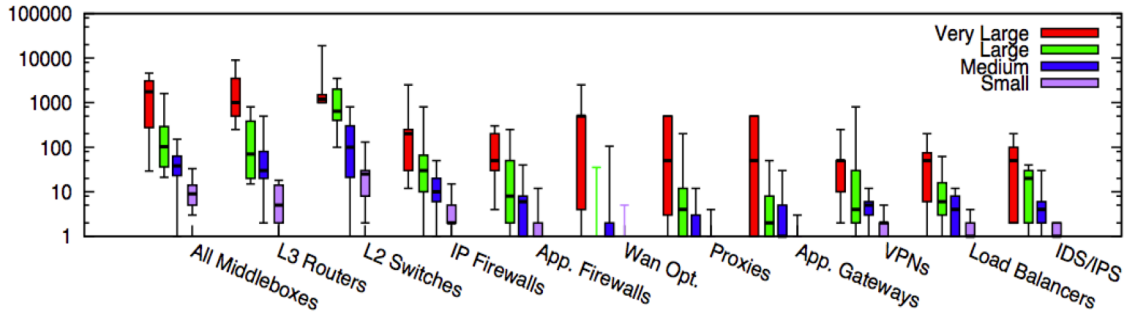


Figure 1: Box plot of middlebox deployments for small (fewer than 1k hosts), medium (1k-10k hosts), large (10k-100k hosts), and very large (more than 100k hosts) enterprise networks. Y-axis is in log scale.

Sherry, Justine, et al. "*Making middleboxes someone else's problem: Network processing as a cloud service.*" Proceedings of the ACM SIGCOMM 2012 conference. ACM, 2012.

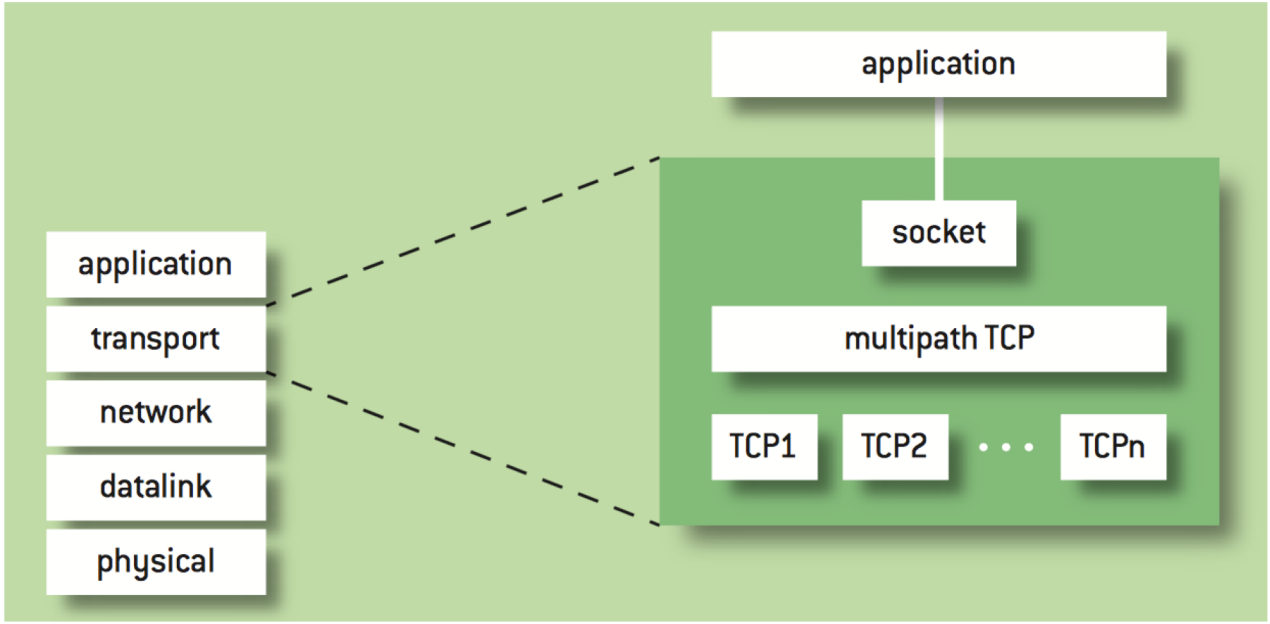
- Hay más middleboxes que routers: Firewalls, balanceadores, VPN concentrator, SSL terminador, IP telephony router ...

Objetivos de MPTCP

- Emplear múltiples caminos en paralelo para una misma conexión
  - WiFi y Ethernet
  - Múltiples interfaces Ethernet en servidores
  - Múltiples caminos en el datacenter
  - Failover
- Emplearlos tan bien como TCP, siendo TCP-friendly
  - Que no ahogue a otros flujos TCP
- Usable como TCP tradicional (API)
- Si TCP funciona en un camino entonces habilitar MPTCP no debe impedir la comunicación

Arquitectura

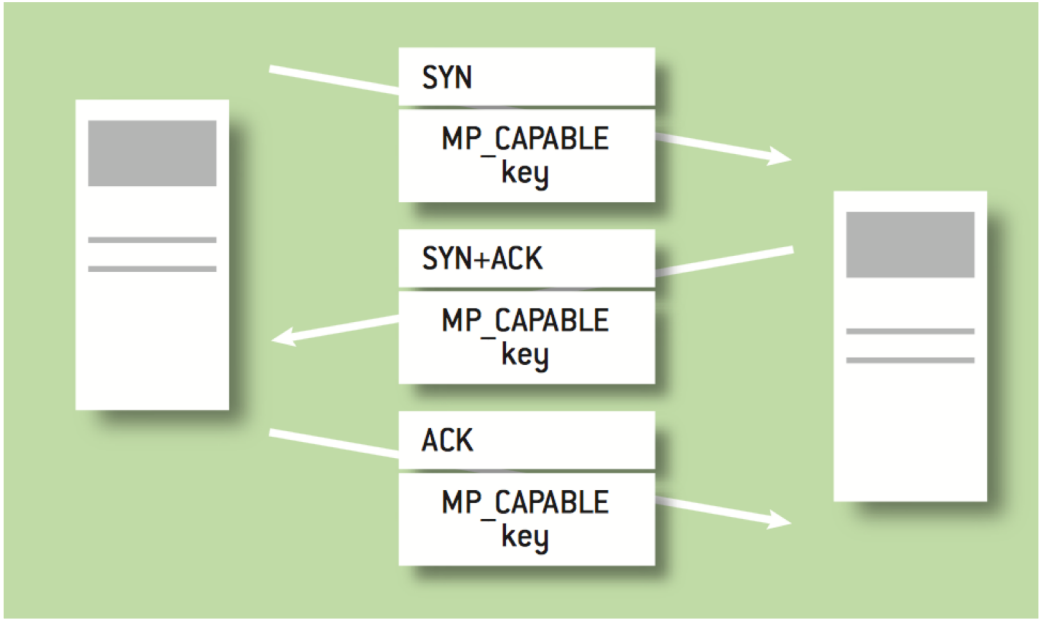
- MPTCP crea subflujos que son conexiones TCP
- Los hosts extremos mantienen estado que une esos flujos
- MPTCP actúa como una capa intermedia entre la aplicación y el nivel de transporte
- La señalización adicional se logra mediante opciones TCP
- Su mayor problema es ser resistente ante los diversos comportamientos de middleboxes



C. Paasch y O. Bonaventure, “Multipath TCP”, ACM Queue, vol. 12, n.2, Marzo 2014

Negociación

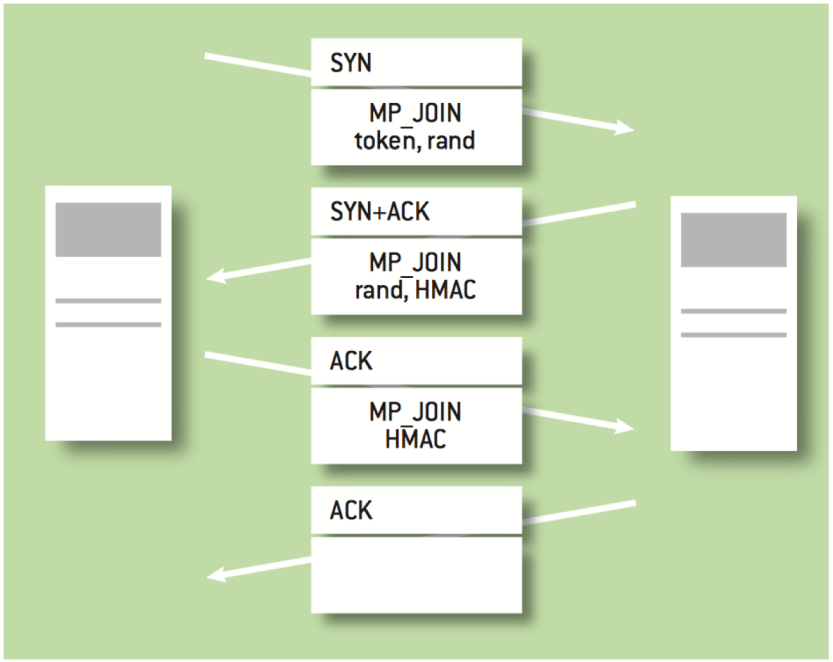
- Mediante una opción TCP nueva (*MP_CAPABLE*) anunciada en el establecimiento de la conexión TCP inicial
- Incluyen una clave
- Middleboxes pueden eliminar la opción (entonces solo TCP)
- Middleboxes pueden descartar los paquetes por no reconocer la opción (entonces reintentar sin ella y solo TCP)



C. Paasch y O. Bonaventure, "Multipath TCP", ACM Queue, vol. 12, n.2, Marzo 2014

Añadir subflujos

- Una nueva conexión se añade como subflujo a la primera
- Puede estar iniciada desde otro interfaz
- No se pueden identificar las conexiones mediante la 4-tupla por la posible presencia de NATs
- Se identifica la conexión MPTCP a la que unirse mediante la opción MP_JOIN con un token generado a partir de la clave



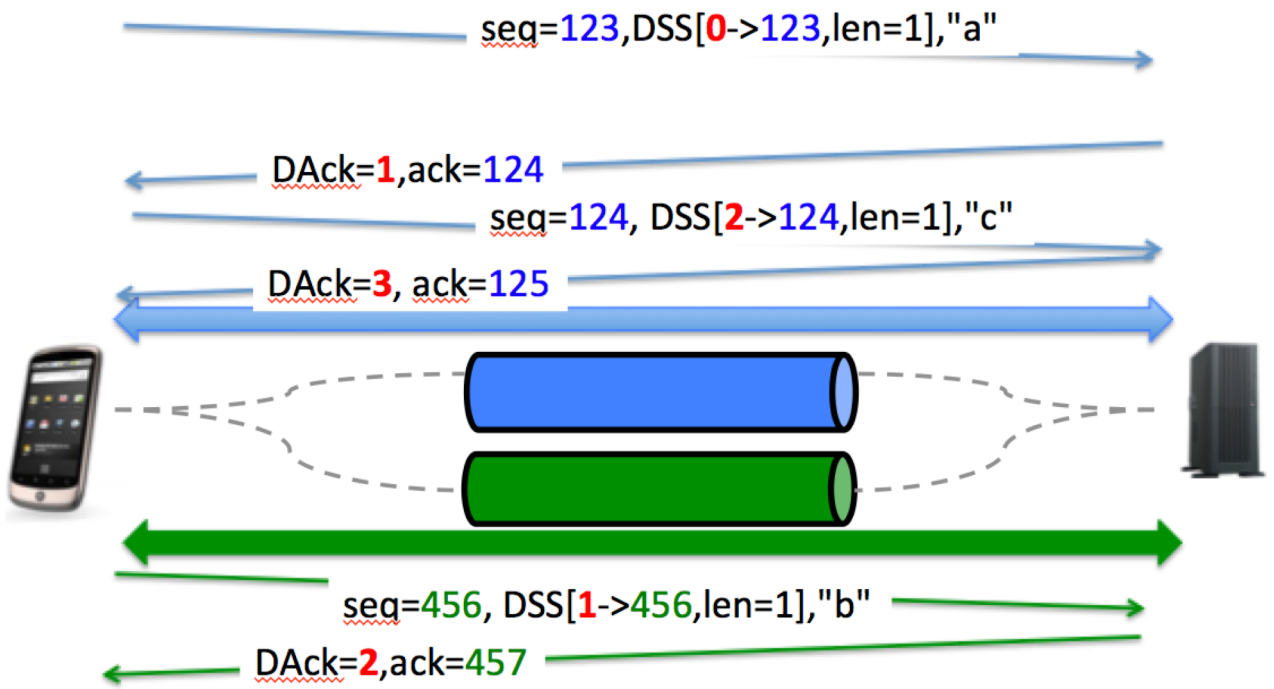
C. Paasch y O. Bonaventure, "Multipath TCP", ACM Queue, vol. 12, n.2, Marzo 2014

Secuencia

- Se puede enviar los datos de la aplicación por cualquiera de los subflujos
- Para hacer una entrega en orden hace falta numerar los datos independientemente del subflujo
- ¿Números de secuencia de segmentos TCP sean del flujo MPTCP?
  - Habría huecos en la secuencia de un subflujo: algunos middleboxes (DPI) no lo soportan (tal vez aborten la conexión con un RST generado por ellos)
  - Hay middleboxes que cambian el ISN en las conexiones TCP
  - Hay middleboxes que dividen los segmentos (ej: NICs con TSO)

Secuencia

- Se mantienen los números de secuencia independientes para cada subflujo
- Secuencia global de MPCTP de 64 bits
- Relación con secuencia global va en una opción TCP (DSS)
- En la opción va un offset respecto al inicio del subflujo
- Un checksum permite detectar que un middlebox haya introducido cambios en los datos (ALG) y se termina el subflujo



Confirmaciones y flujo

- En cada subflujo hay confirmaciones y retransmisiones mediante TCP (SACK, fast retransmit, RTO, etc)
- Se debe retx por el subflujo donde se produjo la pérdida
- Se puede enviar la retransmisión también por el otro subflujo
- Confirmación de la secuencia global (ACK acumulado) mediante la opción DSS (Data Sequence Signal)
- Los datos enviados no pueden liberarse hasta ser confirmados tanto en el DSS como por los ACK del subflujo
- ACK global sirve como inicio de la ventana de control de flujo
- La ventana de control de flujo es compartida entre los subflujos
- Es decir, la ventana anunciada en un subflujo es la global
- Emisor emplea la mayor de las anunciadas por los subflujos (un middlebox podría estar cambiando una de ellas)

Control de congestión

- Cada subflujo TCP tiene su cwnd
- Se debe acoplar el control de flujo de los subflujos o si no habrá reparto injusto con TCP tradicional
- Ahora un flujo de la aplicación son varios de nivel de transporte y el reparto en los cuellos de botella suele ser por esos flujos
- Se desea que si múltiples subflujos pasan por el mismo cuello de botella empleen tanta capacidad como un solo flujo TCP
- En la RFC queda abierto
- Una propuesta: RFC 6356 (Experimental) “Coupled Congestion Control for Multipath Transport Protocols”

Implementaciones

- Linux kernel
- FreeBSD
- iOS
- MacOS
- Solaris
- Algunos middleboxes ;-)

QUIC

¿ QUIC ?

- Quick UDP Internet Connections
- Protocolo experimental de Google (2014)
- Desplegado en servicios de Google y Chrome
- Mejora la latencia de carga de páginas web (inicialmente para HTTP)
- 30% del tráfico que envía Google es QUIC (7% del tráfico de Internet[1])
- Akamai ofrece soporte de QUIC
- En desarrollo en WG quic del IETF desde octubre de 2016

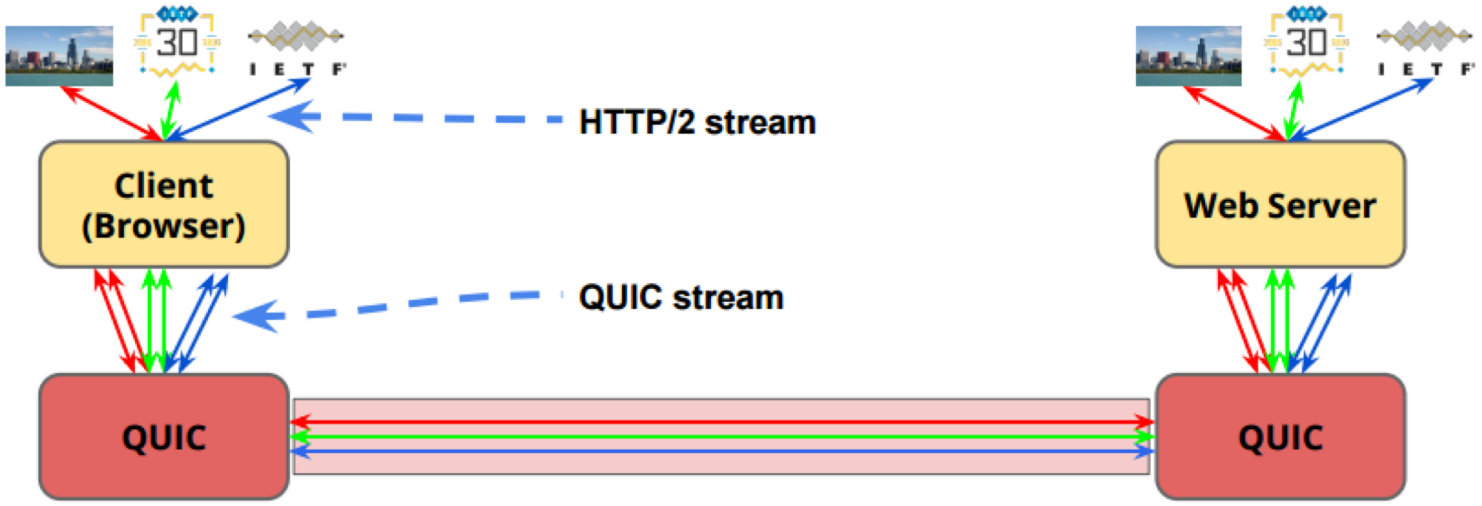
[1]Adam Langley et.al., “The QUIC Transport Protocol: Design and Internet-Scale Deployment”, SIGCOMM’17

¿H2 over TCP es lento?

- 1 RTT establecimiento de la conexión TCP
- 2 RTTs para establecer la sesión TLS
- Pero tenemos TCP Fast Open (RFC 7412)
  - Ya permite establecimiento en 0 RTTs
  - Pero tiene escaso despliegue (modifica TCP y problemas con middleboxes)
- TCP tampoco sabe que hay múltiples flujos, la multiplexación es en la aplicación
- Entonces no sabe que no necesita mantener el orden total, solo de cada flujo
- T/TCP, SCTP, también mejoran tiempos, multiplexación, pero los middleboxes no permiten desplegarlos

Características de QUIC

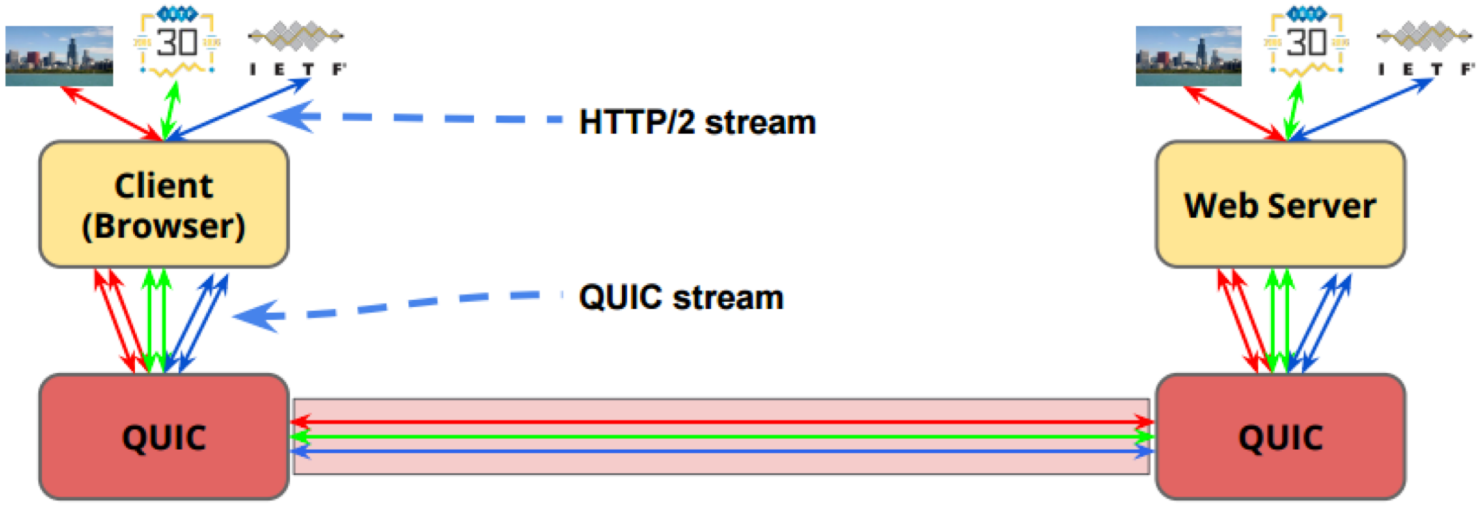
- Puede establecer una conexión en 0 RTTs
  - Manda datos con el paquete para establecer la conexión
  - Si no ha establecido antes una conexión entonces sí gasta 1 RTT
  - 2 RTTs si tiene que negociar versión
- Implementado sobre UDP
  - Para poder atravesar NATs y otros middleboxes
  - En la aplicación en lugar de ser parte del kernel (despliegue más rápido)



https://www.ietf.org/proceedings/98/slides/slides-98-edu-sessf-quic-tutorial-00.pdf
https://youtu.be/ZF6-KmHz1sU

Características de QUIC

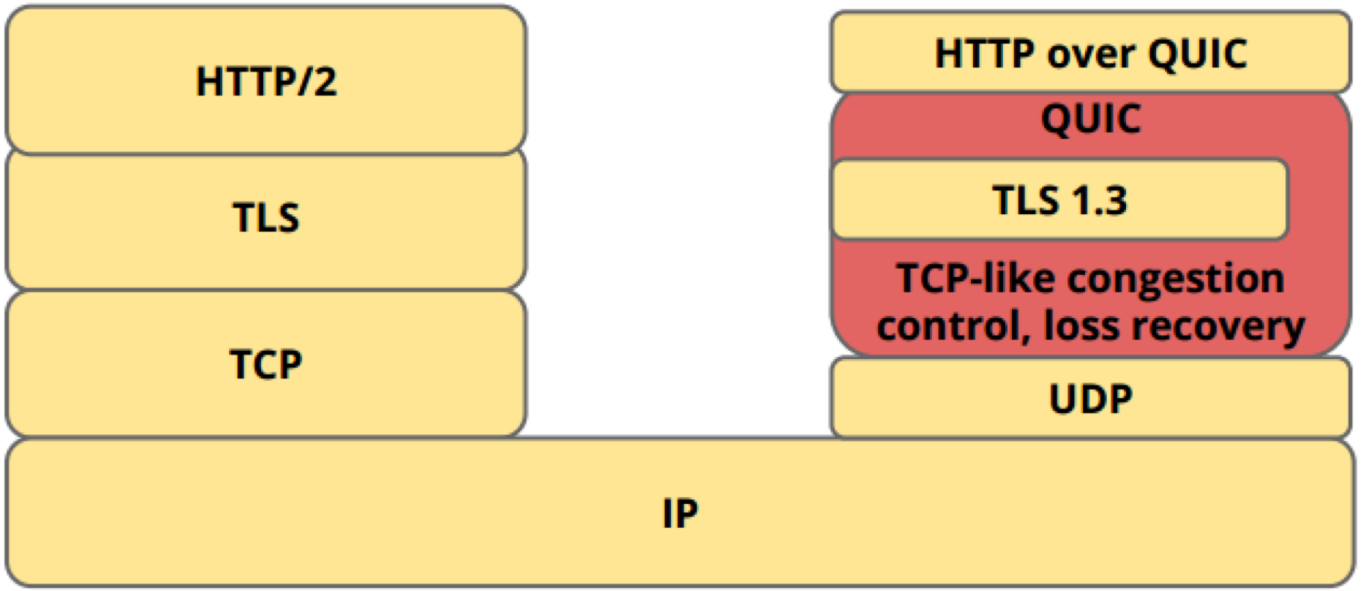
- Mejora la recuperación ante pérdidas
  - No hay ambigüedad en las retransmisiones
  - Da timestamp de llegada de datos en el ACK
  - Permite más rangos confirmados en SACKs
- Más flexible control de congestión
  - En curso, pero parte de más información sobre las pérdidas



https://www.ietf.org/proceedings/98/slides/slides-98-edu-sessf-quic-tutorial-00.pdf
https://youtu.be/ZF6-KmHz1sU

QUIC en IETF

- Ligeramente diferente a la versión de Google
- TLS 1.3 ofrece el handshake en 0 RTTs
- La encriptación oculta QUIC a los equipos de red y lo limita a los extremos
  - Middleboxes no podrán basarse en ello (proxy transparente)
  - Impide la solidificación del protocolo debido a middleboxes
  - Puede ser un problema para ISPs que quieran inspeccionar cabeceras

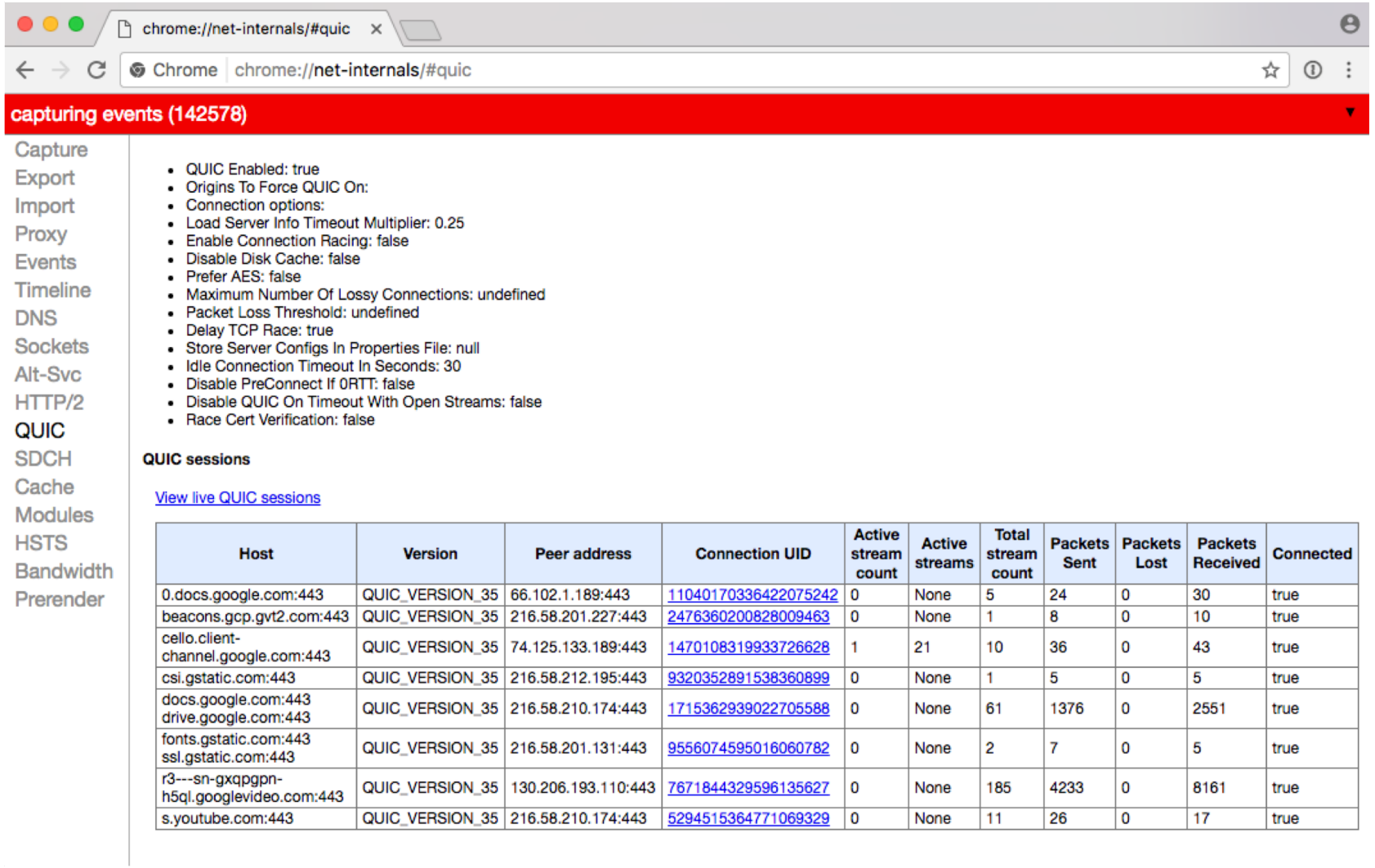


https://www.ietf.org/proceedings/98/slides/slides-98-edu-sessf-quic-tutorial-00.pdf
https://youtu.be/ZF6-KmHz1sU

QUIC hoy

- https://datatracker.ietf.org/wg/quic/
- "QUIC: A UDP-Based Multiplexed and Secure Transport"
  - draft-ietf-quic-transport-25, Enero 2020
  - Fastly, Mozilla
- "QUIC Loss Detection and Congestion Control"
  - draft-ietf-quic-recovery-26, Enero 2020
  - Fastly, Google
- "Using Transport Layer Security (TLS) to Secure QUIC"
  - draft-ietf-quic-tls-25, Enero 2020
  - Mozilla, sn3rd
- "Hypertext Transfer Protocol Version 3 (HTTP/3)"
  - draft-shade-quic-http2-mapping-25, Enero 2020
  - Akamai
- "Network Address Translation Support for QUIC"
  - draft-duke-quic-natsupp-00, Noviembre 2019
  - F5 Networks

QUIC en Chrome



Host	Version	Peer address	Connection UID	Active stream count	Active streams	Total stream count	Packets Sent	Packets Lost	Packets Received	Connected
0.docs.google.com:443	QUIC_VERSION_35	66.102.1.189:443	11040170336422075242	0	None	5	24	0	30	true
beacons.gcp.gvt2.com:443	QUIC_VERSION_35	216.58.201.227:443	2476360200828009463	0	None	1	8	0	10	true
cello.client-channel.google.com:443	QUIC_VERSION_35	74.125.133.189:443	1470108319933726628	1	21	10	36	0	43	true
csi.gstatic.com:443	QUIC_VERSION_35	216.58.212.195:443	9320352891538360899	0	None	1	5	0	5	true
docs.google.com:443 drive.google.com:443	QUIC_VERSION_35	216.58.210.174:443	1715362939022705588	0	None	61	1376	0	2551	true
fonts.gstatic.com:443 ssl.gstatic.com:443	QUIC_VERSION_35	216.58.201.131:443	9556074595016060782	0	None	2	7	0	5	true
r3---sn-gxqpgpn-h5ql.googlevideo.com:443	QUIC_VERSION_35	130.206.193.110:443	7671844329596135627	0	None	185	4233	0	8161	true
s.youtube.com:443	QUIC_VERSION_35	216.58.210.174:443	5294515364771069329	0	None	11	26	0	17	true

New DNS

DNS

DNS over TLS

- RFC 7858: “Specification for DNS over Transport Layer Security (TLS)”
- USC/ISI, ICANN, Mayo 2016
- TCP, puerto 853

DNS over HTTPS

- RFC 8484: 2DNS Queries over HTTPS (DoH)
- ICANN, Mozilla, Octubre 2018
- Petición DNS va en petición HTTP (GET o POST)
- El servidor DoH define el URI y el template para las peticiones
- Respuesta de tipo application/dns-message