upna
Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

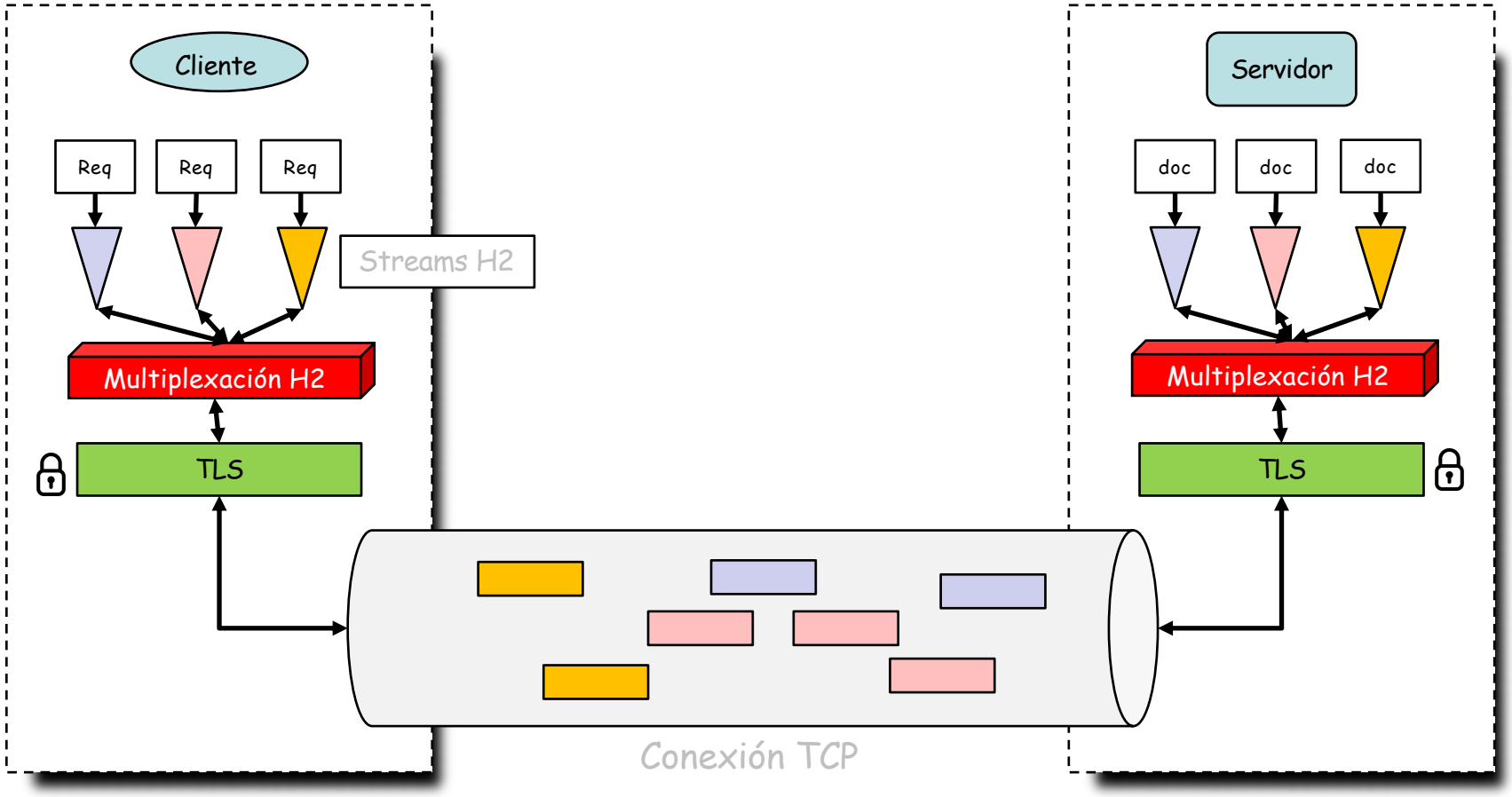
Redes de Nueva Generación
Área de Ingeniería Telemática

HTTP/3

HTTP/3 - Motivación

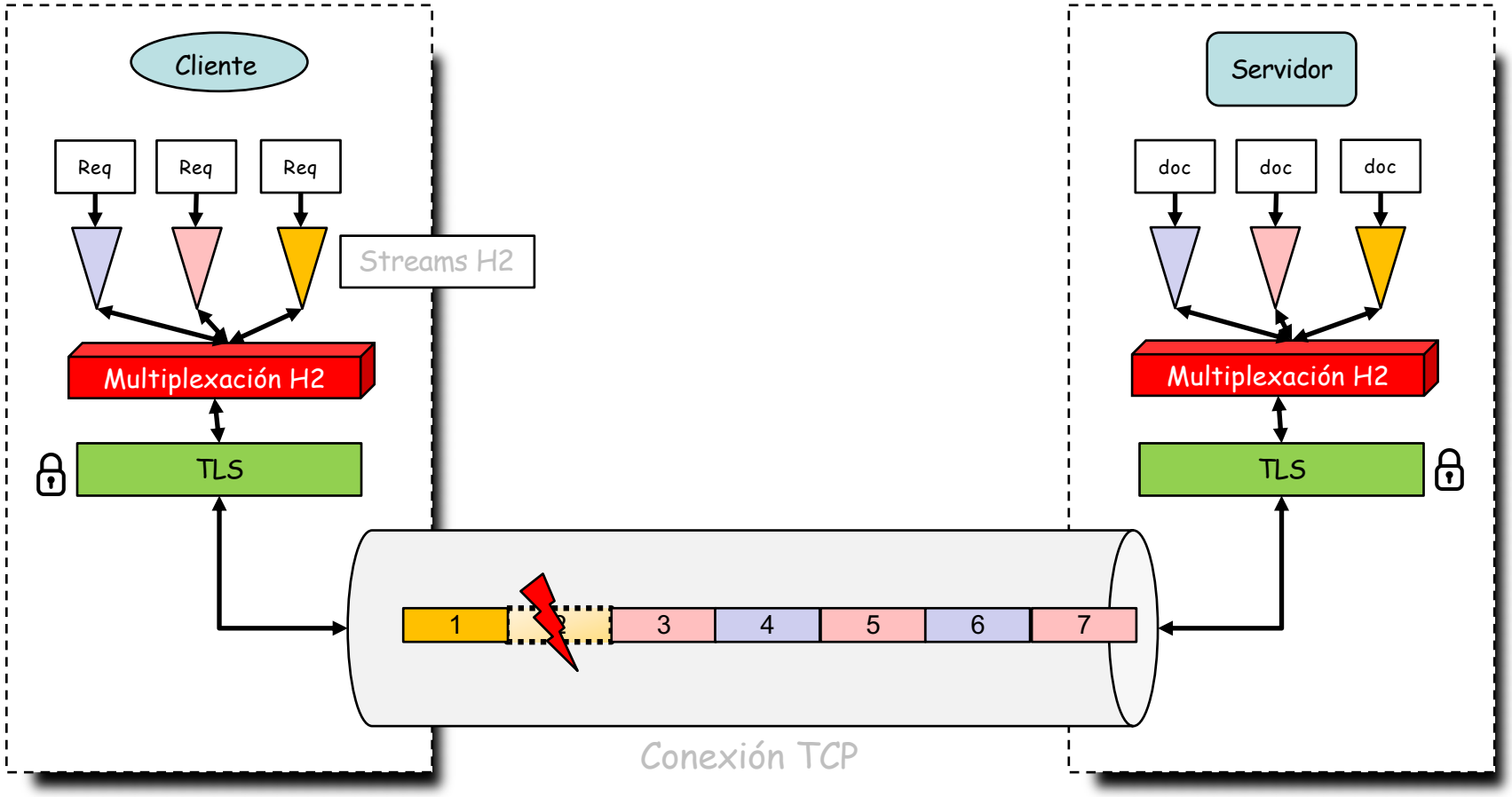
HTTP/2 (H2)

- Multiplexación de streams elimina HOL blocking
- ¿De verdad?



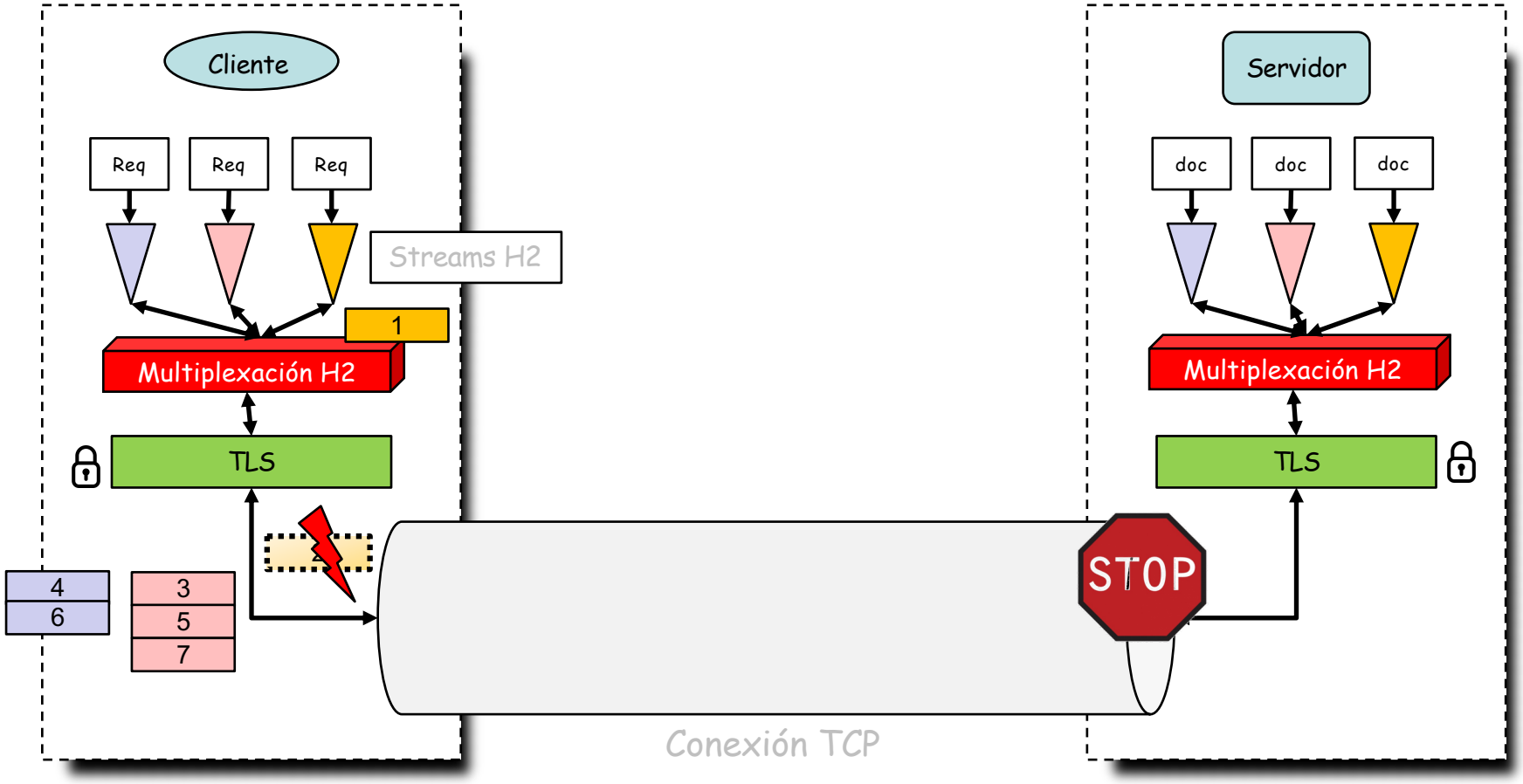
HTTP/2 (H2)

- Emplea una conexión TCP, con 2 streams (uno en cada sentido)
- Cada stream es un flujo bytes ordenados
- ¿Qué sucede si hay una pérdida?
- (...)



HTTP/2 (H2)

- No se entrega a la aplicación el resto de datos hasta “rellenar el hueco”
- La pérdida ha podido afectar solo a paquetes de un stream h2
- Pero afecta a todo el stream TCP y por lo tanto a todos los streams h2
- Además detiene todo el flujo (retransmisiones y control de congestión)
- Resuelto HOL blocking en nivel de aplicación pero no de transporte



Otras limitaciones

- Demasiados RTTs
- Tiempo de establecimiento de la conexión TCP
- Tiempo de establecimiento de la sesión TLS

RTTs con TCP

- 1 RTT establecimiento de la conexión TCP
- Pero tenemos TCP Fast Open (RFC 7413)
  - Permite entregar a la aplicación datos que llegan con el SYN
  - Pero tiene escaso despliegue (modifica TCP y problemas con middleboxes)
- (...)

RTTs con TCP

- 2 RTTs para establecer la sesión TLS
- Pero tenemos
  - Sesiones
  - Tickets
  - 0-RTT con TLS 1.3 (Early data)

Sesiones

No.	Time	Source	Destination	Info
34	24.1546…	192.168.1.101	192.168.1.48	51068 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=166072007
35	24.1546…	192.168.1.48	192.168.1.101	443 → 51068 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=
36	24.1548…	192.168.1.101	192.168.1.48	51068 → 443 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=1660720071 TSecr=
37	24.1557…	192.168.1.101	192.168.1.48	Client Hello
38	24.1557…	192.168.1.48	192.168.1.101	443 → 51068 [ACK] Seq=1 Ack=518 Win=30080 Len=0 TSval=2312132065 TSecr
39	24.1563…	192.168.1.48	192.168.1.101	Server Hello, Change Cipher Spec, Finished
40	24.1564…	192.168.1.101	192.168.1.48	51068 → 443 [ACK] Seq=518 Ack=181 Win=131584 Len=0 TSval=1660720072 TS
41	24.1566…	192.168.1.101	192.168.1.48	Change Cipher Spec, Finished
42	24.1578…	192.168.1.101	192.168.1.48	GET / HTTP/1.1
43	24.1578…	192.168.1.48	192.168.1.101	443 → 51068 [ACK] Seq=181 Ack=1126 Win=31104 Len=0 TSval=2312132067 TS
44	24.1584…	192.168.1.48	192.168.1.101	HTTP/1.1 200 OK (text/html)
45	24.1586…	192.168.1.101	192.168.1.48	51068 → 443 [ACK] Seq=1126 Ack=3077 Win=128640 Len=0 TSval=1660720074
46	24.1586…	192.168.1.101	192.168.1.48	51068 → 443 [ACK] Seq=1126 Ack=3812 Win=127936 Len=0 TSval=1660720074

Early Data

No.	Time	Source	Destination	Info
1	0.000000	10.20.0.19	104.17.143.23	51052 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSv
2	0.015794	104.17.143.23	10.20.0.19	443 → 51052 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1400 SACK
3	0.015811	10.20.0.19	104.17.143.23	51052 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0
4	0.017129	10.20.0.19	104.17.143.23	Client Hello
5	0.017284	10.20.0.19	104.17.143.23	Change Cipher Spec
6	0.017343	10.20.0.19	104.17.143.23	GET /v4/assets/sidebar-lightarrow.svg HTTP/1.1
7	0.033849	104.17.143.23	10.20.0.19	443 → 51052 [ACK] Seq=1 Ack=598 Win=67584 Len=0
8	0.033859	104.17.143.23	10.20.0.19	443 → 51052 [ACK] Seq=1 Ack=604 Win=67584 Len=0
9	0.033861	104.17.143.23	10.20.0.19	443 → 51052 [ACK] Seq=1 Ack=1037 Win=68608 Len=0
10	0.038601	104.17.143.23	10.20.0.19	Server Hello, Change Cipher Spec, Encrypted Extensions, Finished
11	0.038616	10.20.0.19	104.17.143.23	51052 → 443 [ACK] Seq=1037 Ack=689 Win=64128 Len=0
12	0.039258	10.20.0.19	104.17.143.23	End of Early Data, Finished
13	0.053534	104.17.143.23	10.20.0.19	[TLS segment of a reassembled PDU]
14	0.053547	104.17.143.23	10.20.0.19	HTTP/1.1 200 OK

Otras limitaciones

- Demasiados RTTs
- Tiempo de establecimiento de la conexión TCP
- Tiempo de establecimiento de la sesión TLS
- La aplicación no puede sacar provecho a múltiples interfaces en el host

¿Soluciones?

- RFC 4960 Stream Control Transmission Protocol
- Diseñado para el transporte de la señalización de la red telefónica
- Ofrece:
  - Transporte fiable (acknowledged error-free non-duplicated)
  - Multiplexación de sub-streams
  - Transporte ordenado dentro de cada stream
  - Entrega mensajes en lugar de un byte stream
  - Segmentación
  - Multi-homing
  - Congestion avoidance
  - Flow control

Problemas con SCTP

- Implementarlo en los sistemas operativos de los hosts
- Implementarlo en los NATs
- Diferente API

upna
Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

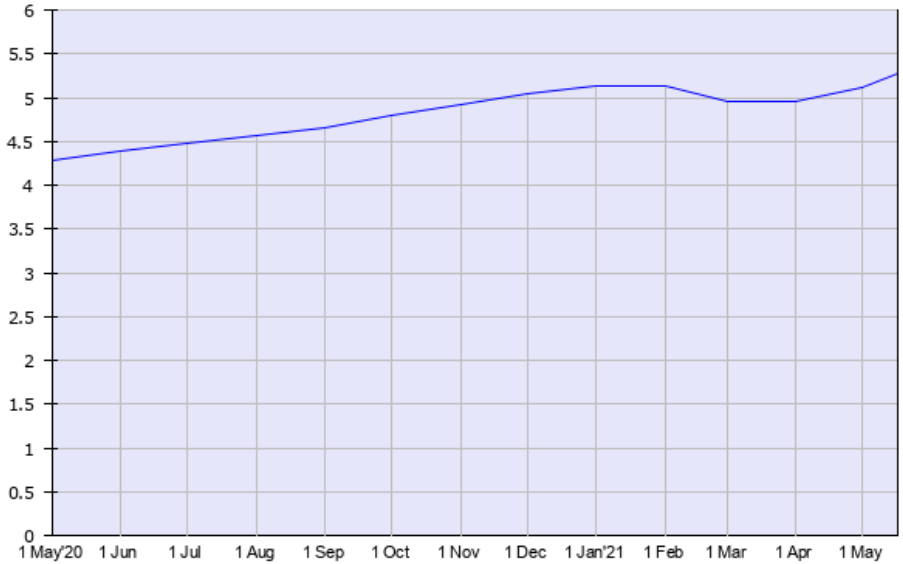
Redes de Nueva Generación
Área de Ingeniería Telemática

QUIC

QUIC - Motivación

¿ QUIC ?

- Quick UDP Internet Connections
- Inicialmente protocolo experimental de Google (2014)
- Desplegado en servicios de Google y Chrome
- En 2017 30% del tráfico que enviaba Google era QUIC (7% del tráfico de Internet[1])
- Es lo que ahora llamamos gQUIC (Google QUIC)
- Akamai, Cloudfare ofrecen soporte de QUIC
- En desarrollo en WG quic del IETF desde octubre de 2016

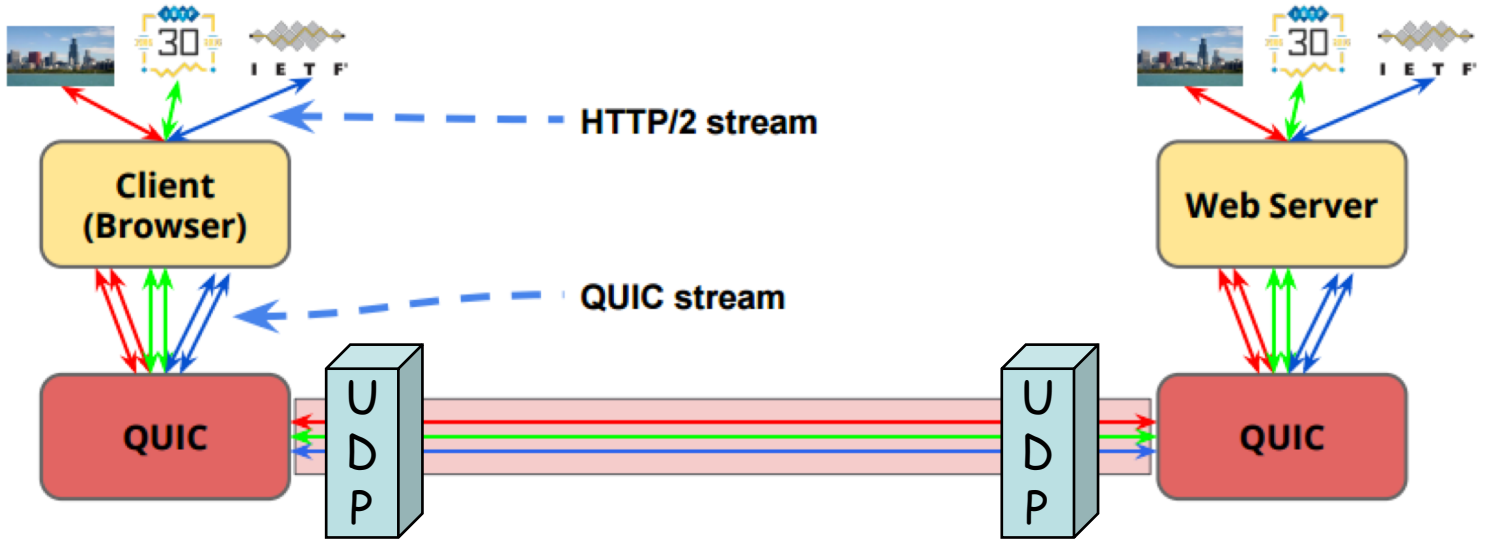


Usage of QUIC for websites, 17 May 2021, W3Techs.com
https://w3techs.com/technologies/details/ce-quic

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

Características de QUIC

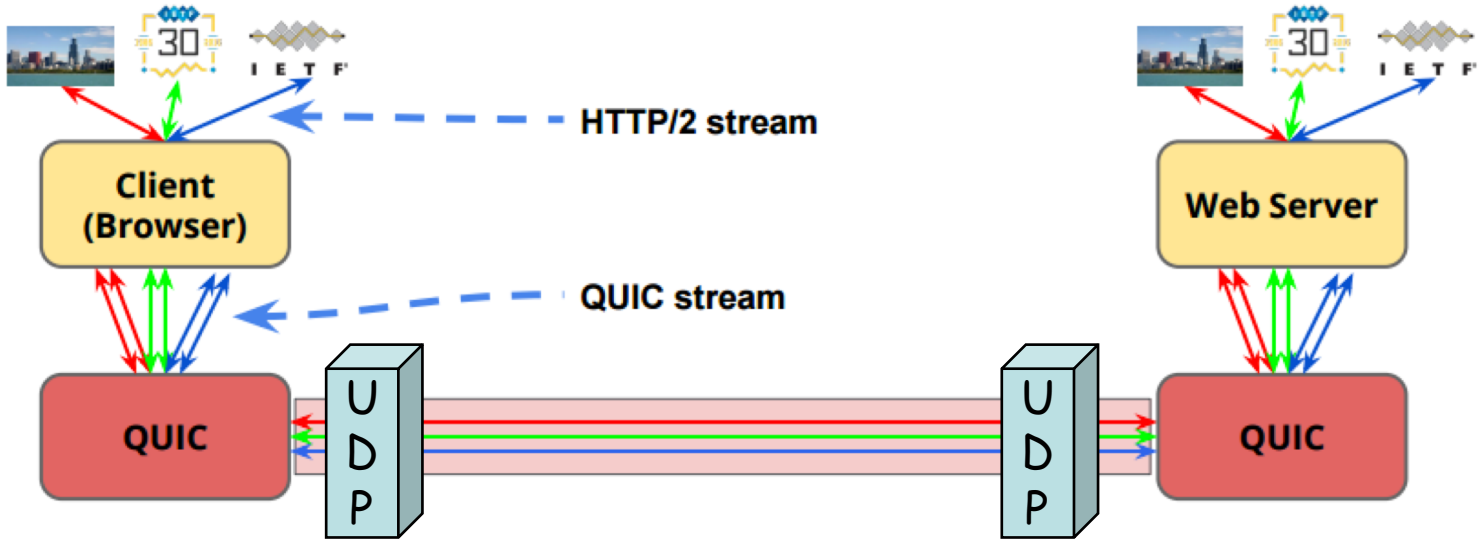
- Diseñado inicialmente para el transporte de HTTP/2
- HTTP/2 + QUIC = HTTP/3
- Implementado sobre UDP
  - Para poder atravesar NATs y otros middleboxes
  - En la aplicación en lugar de ser parte del kernel (despliegue más rápido)
  - En algunas redes puede estar filtrado (fallback a TCP+TLS)
  - Timers en middleboxes menores que para TCP
- Sub-streams (elimina HOL blocking) fiables, ordenados



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

Características de QUIC

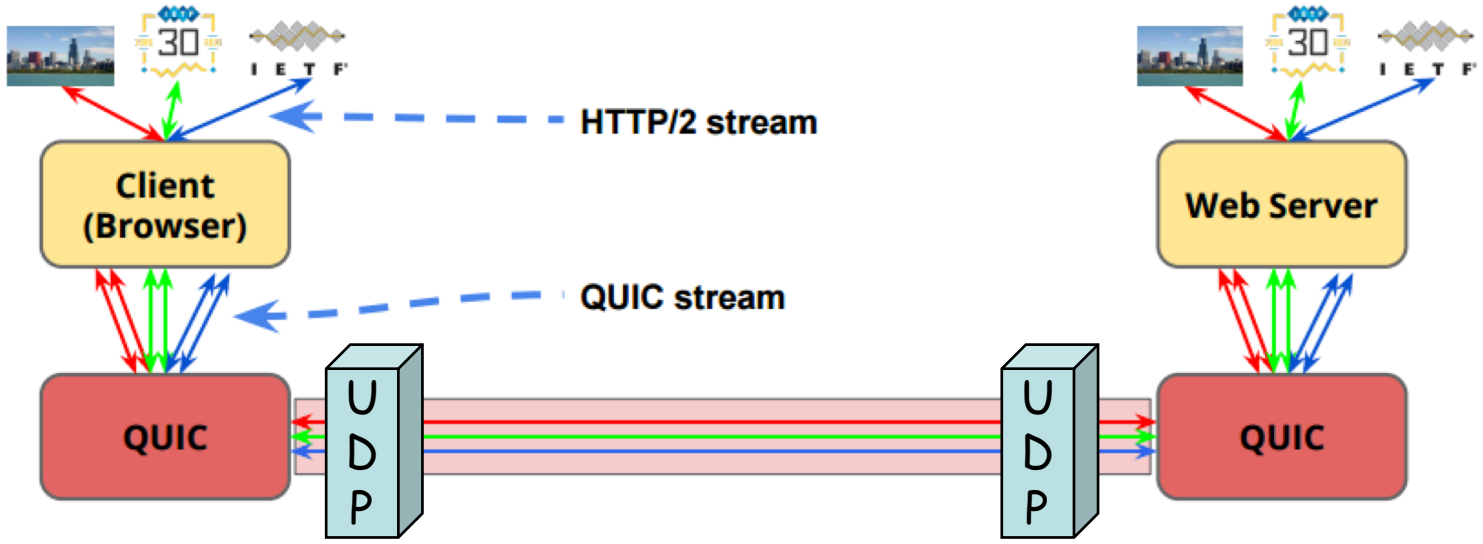
- Puede establecer una conexión en 0 RTTs
  - Si ha establecida una conexión previa que ofrezca las credenciales
  - Manda datos con el paquete para establecer la conexión
  - Si no ha establecido antes una conexión entonces sí gasta 1 RTT
  - 2 RTTs si tiene que negociar versión



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

Características de QUIC

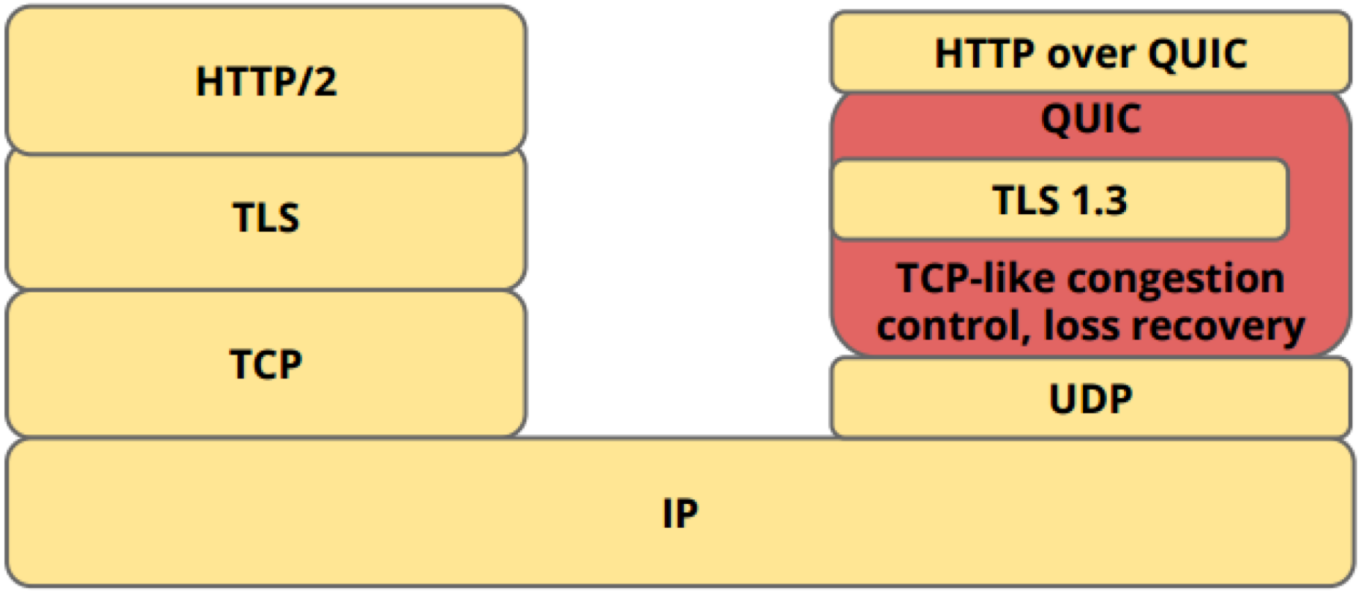
- Mejora la recuperación ante pérdidas
  - No hay ambigüedad en las retransmisiones
  - Da timestamp de llegada de datos en el ACK
  - Permite más rangos confirmados en SACKs
- Más flexible control de congestión
  - En curso, pero parte de más información sobre las pérdidas



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
  - QUIC versión 2 para evitar la ossification
  - Puede ser un problema para ISPs que quieran inspeccionar cabeceras

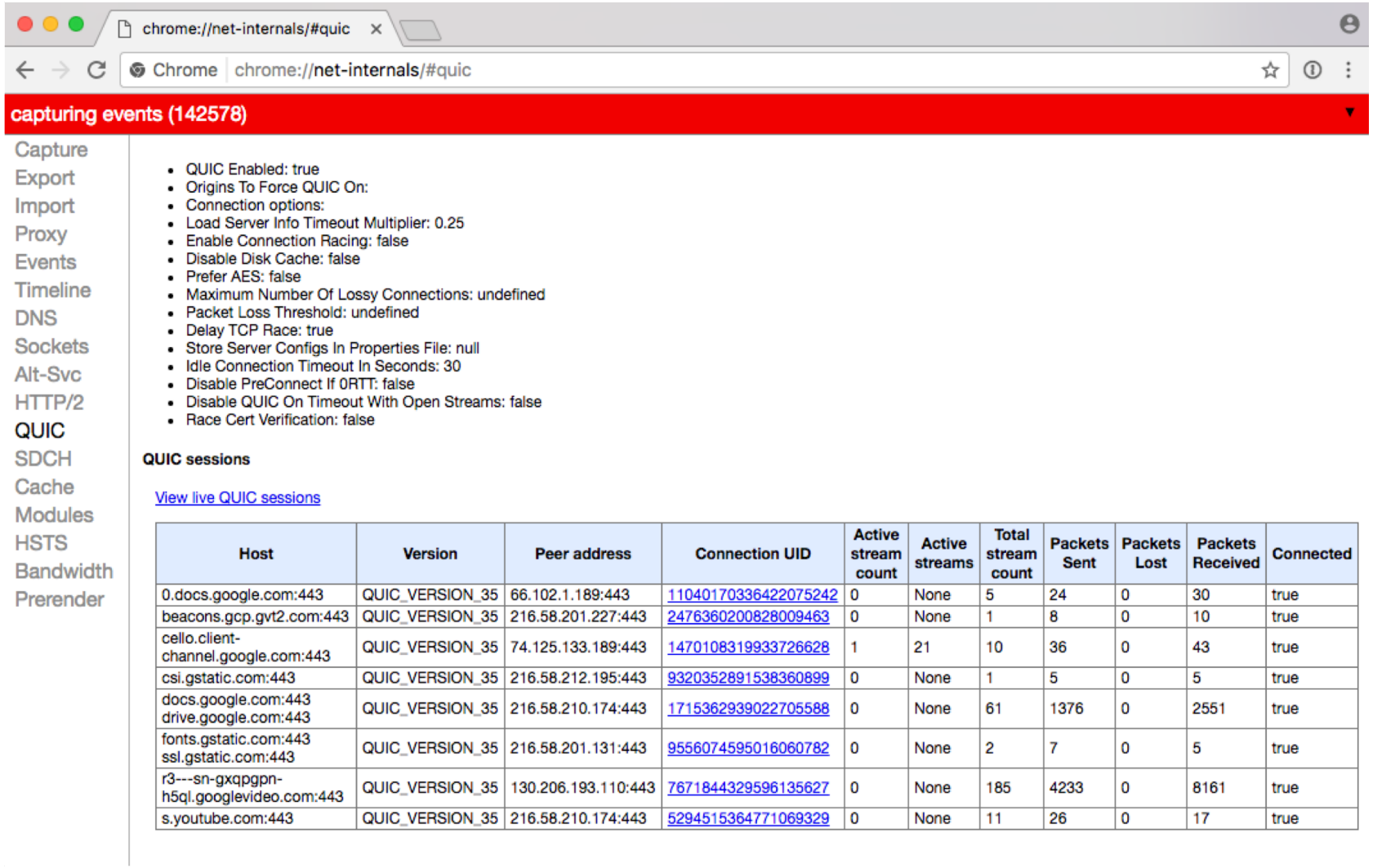


https://www.ietf.org/proceedings/98/slides/slides-98-edu-sessf-quic-tutorial-00.pdf
https://youtu.be/ZF6-KmHz1sU

QUIC hoy

- https://datatracker.ietf.org/wg/quic/
- “QUIC: A UDP-Based Multiplexed and Secure Transport”
  - draft-ietf-quic-transport-34, Enero 2021
  - Fastly, Mozilla
- “QUIC Loss Detection and Congestion Control”
  - draft-ietf-quic-recovery-34, Enero 2021
  - Fastly, Google
- “Using TLS to Secure QUIC”
  - draft-ietf-quic-tls-34, Enero 2021
  - Mozilla, sn3rd
- “Hypertext Transfer Protocol Version 3 (HTTP/3)”
  - draft-shade-quic-http2-mapping-34, Febrero 2021
  - Akamai
- “An Unreliable Datagram Extension to QUIC”
  - draft-ietf-quic-datagram-02, Febrero 2021
  - Apple, Google
- “Multipath Extension for QUIC”
  - Draft-liu-multipath-quic-03, Marzo 2021
  - Alibaba, Private Octopus, ICT-CAS

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

upna
Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Redes de Nueva Generación
Área de Ingeniería Telemática

MPTCP

upna
Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa
Redes de Nueva Generación
Área de Ingeniería Telemática

MPTCP – Situación de partida

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” (v0 RFC 6824, 2013, v1 en RFC 8684)
- MultiPath TCP (MPTCP)

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)

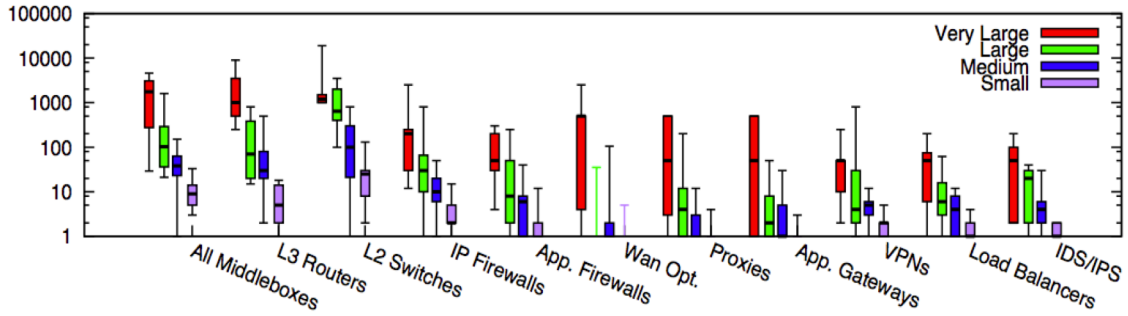


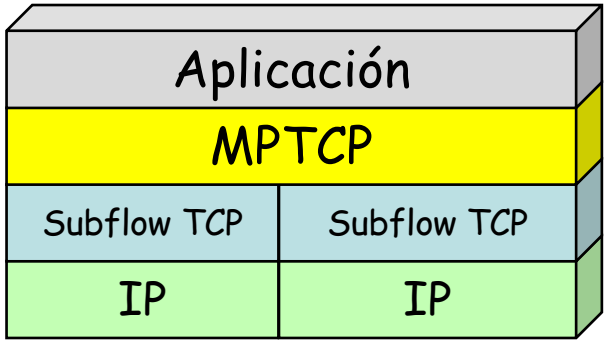
Figure 1: Box plot of middlebox deployments for small (fewer than 1k hosts), medium (1k-10k hosts), large (10k-100k hosts), and very large (more than 100k hosts) enterprise networks. Y-axis is in log scale.

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 ...

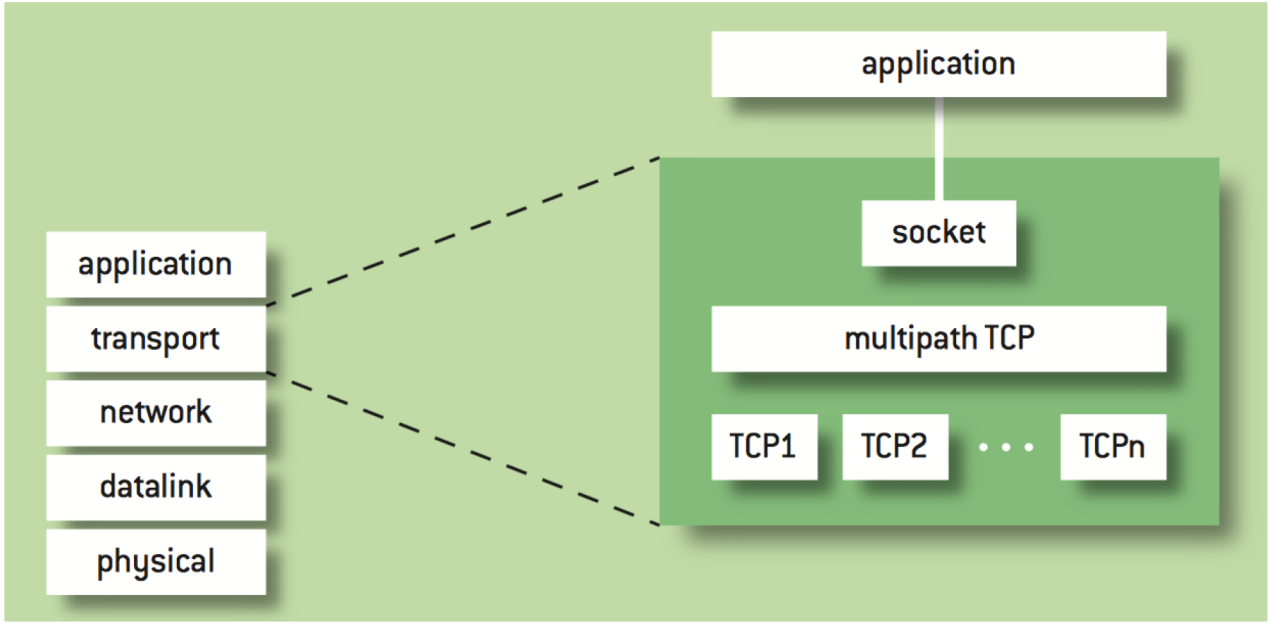
Objetivos de MPTCP

- 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

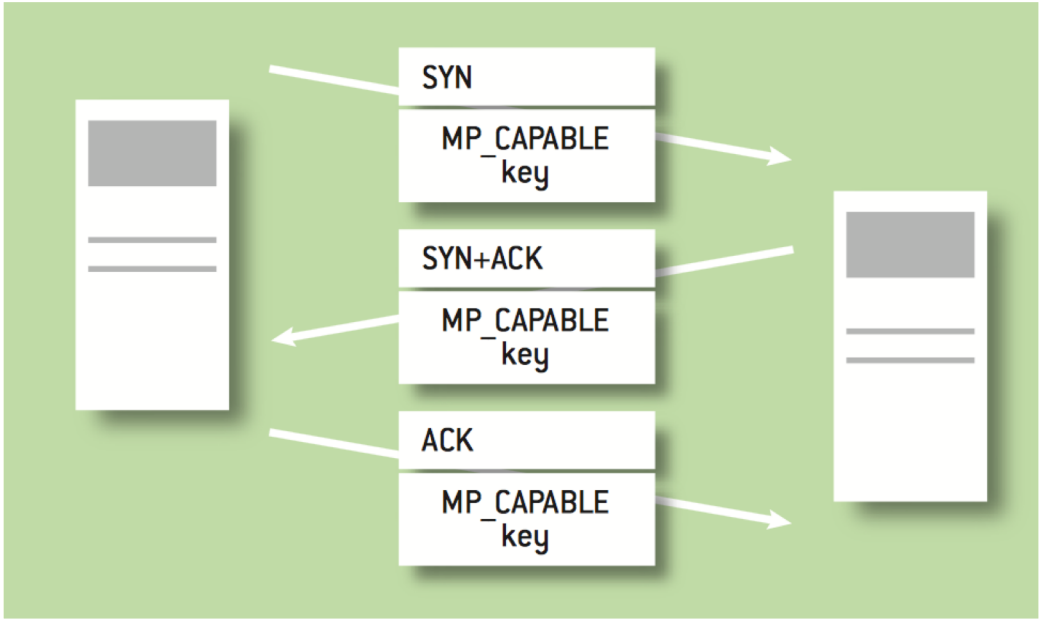
- MPTCP crea subflujos que son conexiones TCP
- Los hosts extremos mantienen estado que une esos flujos
- MPTCP actúa como una capa intermedia entre la aplicación y el nivel de transporte
- La señalización adicional se logra mediante opciones TCP
- Su mayor problema es ser resistente ante los diversos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



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

Negociación

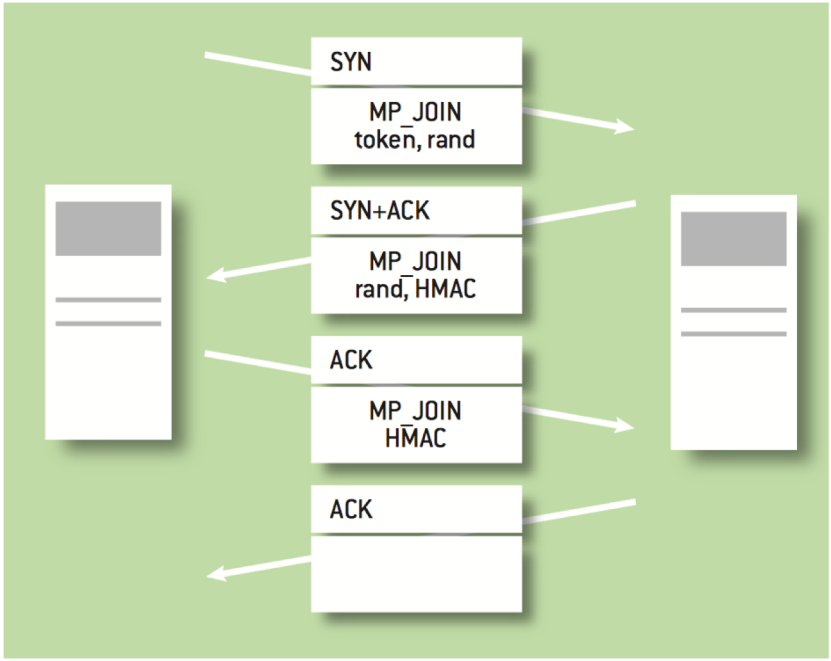
- Mediante una opción TCP nueva (*MP_CAPABLE*) anunciada en el establecimiento de la conexión TCP inicial
- Incluyen una clave
- Middleboxes pueden eliminar la opción (entonces solo TCP)
- Middleboxes pueden descartar los paquetes por no reconocer la opción (entonces reintentar sin ella y solo TCP)



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



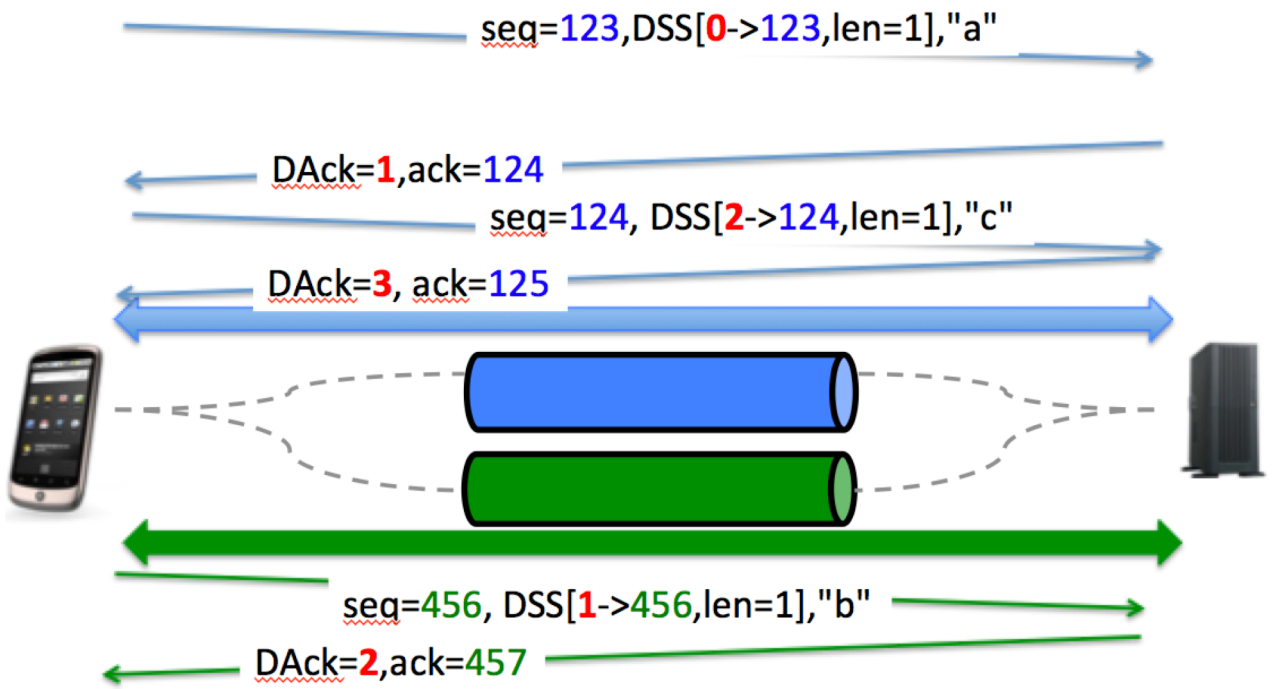
C. Paasch y O. Bonaventure, “Multipath TCP”, ACM Queue, vol. 12, n.2, Marzo 2014

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



http://multipath-tcp.org/data/MultipathTCP-netsys.pdf

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 ;-)

upna
Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Redes de Nueva Generación
Área de Ingeniería Telemática

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