

# *Protocolos de Transporte y de Ruteo Externo*

Comunicación de datos (86.12)  
Primer cuatrimestre de 2022

Becker, Gonzalo Agustín  
Padrón: 104291  
gbecker@fi.uba.ar

## Indice

<b>1. Análisis de capturas de TCP</b>	<b>2</b>
1.1. Intentos de conexión TCP . . . . .	2
1.2. Origen y destino del intento de conexión . . . . .	2
1.3. Conexiones exitosas . . . . .	2
1.4. Análisis de la primer conexión . . . . .	3
1.4.1. Cliente y servidor . . . . .	3
1.4.2. Números de secuencia iniciales (ISN) . . . . .	4
1.4.3. Opciones intercambiadas en el Three Way Handshake . . . . .	4
1.4.4. Primer segmento con datos . . . . .	5
1.5. Control de flujo . . . . .	6
1.6. Control de congestión . . . . .	7
1.7. Round Trip Time (RTT) . . . . .	8
<b>2. Ruteo externo</b>	<b>10</b>
2.1. Cliente y servidor . . . . .	10
2.2. Descripción de la captura . . . . .	10
2.3. Utilización de direcciones privadas . . . . .	11
2.4. Análisis de la conexión BGP . . . . .	11
2.5. KEEPALIVEs . . . . .	12
2.6. Esquema de conexiones propuesto . . . . .	13

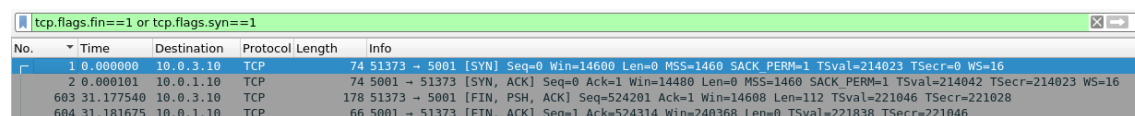
## 1. Análisis de capturas de TCP

TCP es un protocolo de transporte orientado a conexión (a diferencia de UDP), que encapsula los datos intercambiados entre aplicaciones. A su vez, cada segmento TCP es encapsulado dentro de datagramas IP en el nivel de red. Una gran ventaja de que TCP sea un protocolo orientado a conexión es la posibilidad de implementar controles de flujo y congestión. Esto se debe a que la constante comunicación bidireccional entre dos estaciones le permite a cada una estimar el estado de congestión en la red como también conocer la capacidad de recepción de datos en la estación destino.

### 1.1. Intentos de conexión TCP

En la captura bajo análisis, se distingue solamente una sola conexión TCP. Esto puede verificarse mediante la utilización de filtros en wireshark, como se observa en la Figura 1, con la finalidad de encontrar todos los segmentos TCP que han sido transmitidos con la flag SYN para iniciar una conexión, o bien con FIN para finalizarla.

Figura 1: Cantidad total de conexiones en la captura



tcp.flags.fin==1 or tcp.flags.syn==1

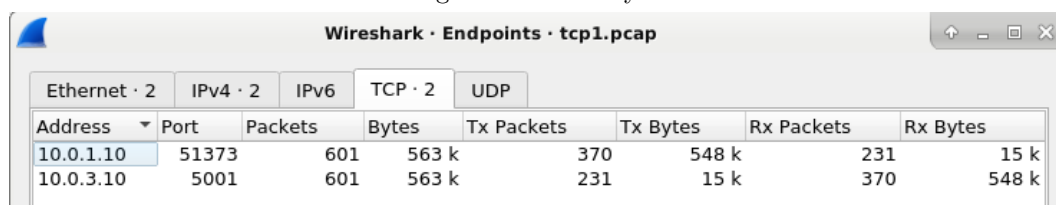
No.	Time	Destination	Protocol	Length	Info
1	0.000000	10.0.3.10	TCP	74	51373 → 5001 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=214023 TSecr=0 WS=16
2	0.000101	10.0.1.10	TCP	74	5001 → 51373 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK_PERM=1 TSval=214042 TSecr=214023 WS=16
603	31.177540	10.0.3.10	TCP	178	51373 → 5001 [FIN, PSH, ACK] Seq=524201 Ack=1 Win=14608 Len=112 TSval=221046 TSecr=221028
604	31.181675	10.0.1.10	TCP	66	5001 → 51373 [FIN, ACK] Seq=1 Ack=524314 Win=240368 Len=0 TSval=221838 TSecr=221046

Se evidencia entonces que existen dos intentos de conexión a lo largo de la captura, uno en el sentido 10.0.1.10→10.0.3.10, y otro en el sentido inverso.

### 1.2. Origen y destino del intento de conexión

En la Figura 2 se pueden observar las direcciones IP de las dos estaciones involucradas, cómo también los puertos utilizados por cada una. Se concluye entonces que, para el primer segmento (No. = 1 en la Figura 1), el origen es la IP 10.0.1.10 con el puerto 51373, mientras que el destino es la IP 10.0.3.10 con el puerto 5001. En cambio, en la segunda operación (No. = 2) el segmento va en el sentido inverso: 10.0.3.10:5001 → 10.0.1.10:51373 (siguiendo el formato IP:Port)

Figura 2: Puertos y IP



Wireshark · Endpoints · tcp1.pcap

Ethernet · 2		IPv4 · 2		IPv6		TCP · 2		UDP	
Address	Port	Packets	Bytes	Tx Packets	Tx Bytes	Rx Packets	Rx Bytes		
10.0.1.10	51373	601	563 k	370	548 k	231	15 k		
10.0.3.10	5001	601	563 k	231	15 k	370	548 k		

### 1.3. Conexiones exitosas

En TCP, una conexión exitosa comienza con el "Three Way Handshake", que se caracteriza por el intercambio de tres segmentos:

- El primero involucra la transmisión de un segmento con el flag SYN desde el cliente hasta el servidor. Es decir, mediante este segmento el cliente solicita al servidor el establecimiento de una conexión.
- El segundo segmento lo transmite el servidor al cliente, y contiene tanto las flags de SYN como ACK. El ACK confirma el establecimiento de la conexión en el sentido cliente→servidor, mientras que el SYN representa la solicitud del servidor de establecer una conexión en el sentido servidor→cliente.
- El tercer segmento es la confirmación del cliente, mediante un ACK, del establecimiento de la conexión en el sentido servidor→cliente.

Es evidente entonces que la conexión (en ambos sentidos, cliente ↔ servidor) que se estudia en esta captura se corresponde a una conexión exitosa, pues ambos segmentos con la flag SYN son confirmados con sus respectivos ACK, como se observa en la figura 3.

Figura 3: Three Way Handshake

No.	Time	Destination	Protocol	Length	Info
1	0.000000	10.0.3.10	TCP	74	51373 → 5001 [SYN] Seq=0 Win=14608 Len=0 MSS=1460 SACK_PERM=1 TSval=214023 TSecr=0 WS=16
2	0.000101	10.0.1.10	TCP	74	5001 → 51373 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK_PERM=1 TSval=214042 TSecr=214023 WS=16
3	0.234495	10.0.3.10	TCP	66	51373 → 5001 [ACK] Seq=1 Ack=1 Win=14608 Len=0 TSval=214082 TSecr=214042

Window size value: 14600
[Calculated window size: 14600]
[Checksum: 0x2409 (unverified)]
[Checksum Status: Unverified]
Urgent pointer: 0
Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale
TCP Option - Maximum segment size: 1460 bytes
Kind: Maximum Segment Size (2)
Length: 4
MSS Value: 1460
TCP Option - SACK permitted
Kind: SACK Permitted (4)
Length: 2
TCP Option - Timestamps: TSval 214023, TSecr 0
Kind: Time Stamp Option (8)
Length: 10
Timestamp value: 214023
Timestamp echo reply: 0
TCP Option - No-Operation (NOP)
Kind: No-Operation (1)
TCP Option - Window scale: 4 (multiply by 16)
Kind: Window Scale (3)
Length: 3
Shift count: 4

Hacia el final de la captura, se observa un mecanismo similar para finalizar la conexión. El cliente envía un segmento con el flag FIN, el servidor responde con un segmento con las flags FIN y ACK, y el cliente responde finalmente con un ACK, como se puede ver en la Figura 4.

Figura 4: Fin de la conexión

No.	Source	Destination	time	Length	Protocol	Info
601	10.0.1.10	10.0.3.10	31.171779	1514	TCP	51373 → complex-link(5001) [ACK] Seq=522753 Ack=1 Win=14608 Len=1448 TSval=221046 TSecr=221028
603	10.0.1.10	10.0.3.10	31.177540	178	TCP	51373 → complex-link(5001) [FIN, PSH, ACK] Seq=524201 Ack=1 Win=14608 Len=112 TSval=221046 TSecr=221028
605	10.0.1.10	10.0.3.10	31.333956	66	TCP	51373 → complex-link(5001) [ACK] Seq=524314 Ack=2 Win=14608 Len=0 TSval=221857 TSecr=221838

## 1.4. Análisis de la primer conexión

### 1.4.1. Cliente y servidor

En la Figura 3 se observa el Three Way Handshake, que exitosamente permite establecer la conexión entre las dos estaciones. Considerando que la estación que solicita la primer conexión a través del primer segmento con la flag SYN es 10.0.1.10 a través del port 51373, se puede concluir que esta es el cliente, mientras que la 10.0.3.10:5001 es el servidor.

### 1.4.2. Números de secuencia iniciales (ISN)

Si se modifica la configuración de Wireshark para mostrar los números de secuencia iniciales (ISN) no relativos, se obtienen los valores observados en la Figura 5. Es decir, el cliente indica un valor de 3436242234 mientras que el servidor un valor de 2219352712. Estos números son un indicador del primer byte en cada segmento transmitido. Más aún, los segmentos de ACK utilizan el número de Ack para indicar cuál es el próximo número de secuencia que la estación espera recibir en el siguiente segmento.

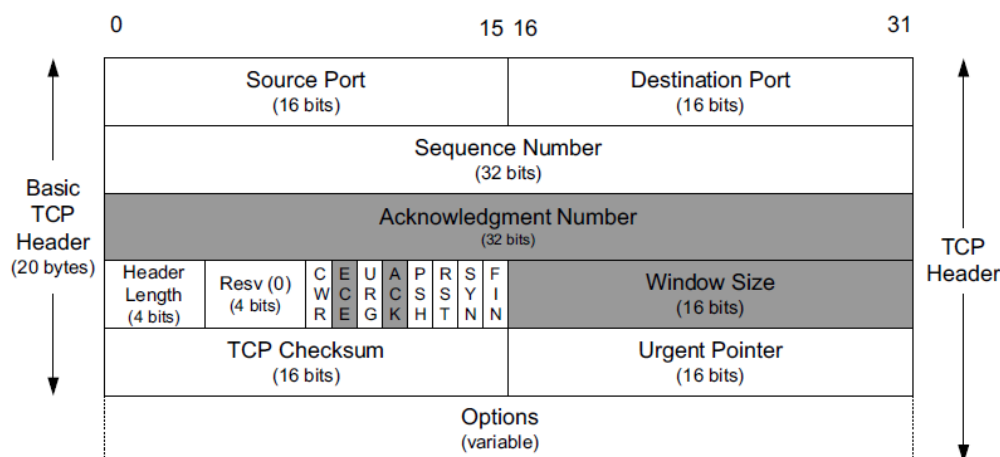
Figura 5: ISN no relativos

No.	time	Source	Destination	Source	Length	Protocol	Info
1	0.000000	10.0.1.10	10.0.3.10	51373	74	TCP	51373 → complex-link(5001) [SYN] Seq=3436242234 Win=14608 Len=0 MSS=1460 SACK_PERM=1 TSval=21402
2	0.000101	10.0.3.10	10.0.1.10	complex-link	74	TCP	complex-link(5001) → 51373 [SYN, ACK] Seq=2219352712 Ack=3436242235 Win=14480 Len=0 MSS=1460 SACK
3	0.234495	10.0.1.10	10.0.3.10	51373	66	TCP	51373 → complex-link(5001) [ACK] Seq=3436242235 Ack=2219352713 Win=14608 Len=0 TSval=214082 TSec

### 1.4.3. Opciones intercambiadas en el Three Way Handshake

En la Figura 6 se encuentra la estructura de un segmento TCP. Hasta ahora ya hemos definido y estudiado los campos de puerto origen y destino, número de secuencia y número de acknowledgment. A continuación se detallan algunos de los campos más comunes en el campo de opciones.

Figura 6: Estructura TCP



En el Three Way Handshake, las opciones que se negocian o establecen entre las estaciones en la captura (ver Figura 3) son las siguientes:

- MSS (Maximum segment size): Refiere a la máxima extensión que puede tener el campo de datos en un segmento. El default es de 536 bytes, pero en este caso el valor se establece en el three way handshake como 1460 bytes (ambas estaciones indican el mismo valor).
- Escalaje de la ventana: Como el largo de la ventana tiene un largo fijo, se utiliza la opción de Window scaling para incrementar el máximo largo de la ventana de 65535 bytes a aproximadamente 1 Gbyte. Esta opción sólo se configura durante el Three Way Handshake, y multiplica al largo de la ventana por  $2^{Window-scaling}$ . En este caso en particular multiplica al ancho de la ventana del servidor por  $2^4 = 16$ , de modo que la ventana pasa a tomar un

valor inicial de  $905 \cdot 16 = 14480$ . Por supuesto, el valor de esta ventana varía a lo largo de la captura, pues posee un comportamiento dinámico

- **SACK permitted:** mediante esta opción, el cliente y servidor informan si permiten la utilización de los selective acknowledge (SACK). El reconocimiento selectivo de TCP (TCP SACK), definido en la RFC 2018, tiene el objetivo de solucionar los problemas ocasionados por la congestión y la pérdida de segmentos, especialmente en las aplicaciones que utilizan ventanas de TCP grandes a través de enlaces vía satélite o enlaces transcontinentales. Esta opción permite retransmitir únicamente los segmentos que aún no fueron enviados correctamente, evitando así una retransmisión completa de todos los segmentos desde el primer segmento faltante.
- **Timestamps:** De acuerdo a la RFC 7323, los timestamps permiten el cálculo del RTT para cada segmento confirmado. Funcionan a partir de dos números: TSval y TSech. En general, si un segmento tiene un determinado valor de TSval, la confirmación (ACK) de este segmento replicará ese valor en la variable TSech.

#### 1.4.4. Primer segmento con datos

En la Figura 7 se observa el primer segmento que contiene datos. Este tiene lugar en la cuarta línea, es un segmento enviado desde el cliente hacia el servidor y su número de secuencia real (no relativo) es 3436242235 (se incrementó en 1 respecto al primer segmento transmitido, en la primera fila).

Figura 7: ISN no relativos

No.	time	Sou	Des	Source	Length	Protocol	Info
1	0.000000	1...	1...	51373	74	TCP	51373 → complex-link(5001) [SYN] Seq=0 Win=14608 Len=0 MSS=1460 SACK_PERM=1 TSval=214023 TSecr=0 WS=16
2	0.000101	1...	1...	complex-link	74	TCP	complex-link(5001) → 51373 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK_PERM=1 TSval=214042 TSecr=214023 ...
3	0.234495	1...	1...	51373	66	TCP	51373 → complex-link(5001) [ACK] Seq=1 Ack=1 Win=14608 Len=0 TSval=214082 TSecr=214042
4	0.234507	1...	1...	51373	90	TCP	51373 → complex-link(5001) [PSH, ACK] Seq=1 Ack=1 Win=14608 Len=24 TSval=214082 TSecr=214042
5	0.234601	1...	1...	complex-link	66	TCP	complex-link(5001) → 51373 [ACK] Seq=1 Ack=25 Win=14480 Len=0 TSval=214101 TSecr=214082

La confirmación de recepción de este segmento se encuentra en la quinta fila. Esto se puede verificar de dos formas:

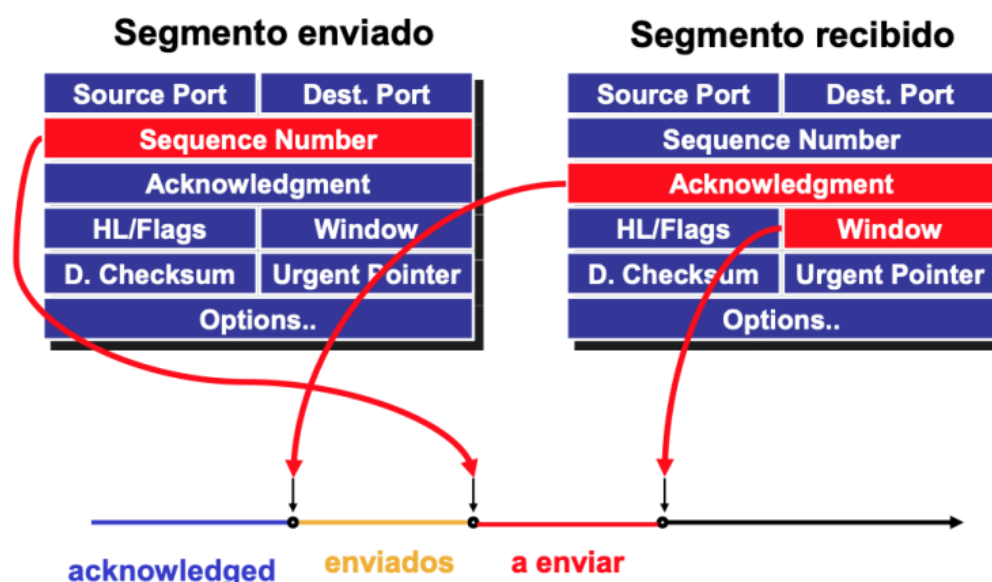
- **Mediante el valor de Ack y de Seq:** El número de Ack indica cuál es el próximo número de secuencia que se espera recibir. De este modo, si una estación recibe un ACK con un cierto número Ack, se confirman todos los segmentos enviados con un número de secuencia menor a Ack. En este caso el Ack vale 25, y es mayor al Seq anterior, que valía 1, por lo que efectivamente el segmento de la quinta fila confirma al segmento en la cuarta fila (y también al de la tercera).
- **Mediante los timestamps:** Los ACK utilizan el valor de TSecr para confirmar a los segmentos con un determinado valor de TSval. En este caso, el TSecr del ACK en la quinta fila vale 214082, al igual que el TSval en la cuarta fila. De este modo, se verifica nuevamente que la quinta fila es la confirmación de la cuarta.

Considerando que el valor de Ack se incrementa en 24 respecto al número de secuencia, se deduce que el ACK confirma los 24 bytes de datos del segmento.

## 1.5. Control de flujo

El control de flujo permite sincronizar el envío con la recepción, y de ser necesario frenar al emisor para evitar colapsar el buffer del receptor. Esto se logra a través de una ventana (rwnd) que cada estación anuncia en cada segmento, y que indica cual es la cantidad de bytes del buffer que se encuentran libres para recibir nuevos segmentos. Luego, se aplica el mecanismo de ventana deslizante, en donde cada estación realiza constantemente cálculos para decidir cuantos bytes de información puede enviar, teniendo en cuenta los segmentos que fueron ya enviados pero todavía no confirmados y el espacio disponible en el buffer del receptor. Esto se evidencia en la Figura 8.

Figura 8: Ventana deslizante



En la Figura 9 se observa la única ocurrencia de control de flujo en la captura. El control de flujo se da entre las filas 280 y 285, inicia a los 12.16 segundos y finaliza a los 22.31 segundos, y tiene una duración de 10.15 segundos. Notar que, como en la línea 279 la ventana del receptor toma un valor de 1024 bytes, en la fila 280 el cliente solamente puede mandar 1024 bytes de datos, más 66 del header, totalizando 1090 bytes. Por lo tanto, podría decirse que a partir del número de secuencia 254873 (fila 280) se activa el control de flujo, pues rwnd limita el flujo de datos a partir de dicho punto. A partir de la línea 281 el servidor envía un segmento con ventana nula, indicando que ya no tiene capacidad de recibir nuevos segmentos. El control de flujo se desactiva en la línea 286, cuando el servidor envía una ventana no nula, de valor 83984. También es interesante destacar los segmentos keep-alive, que le permiten al cliente verificar si la conexión todavía está operativa, aunque no se transmiten datos en estos segmentos.

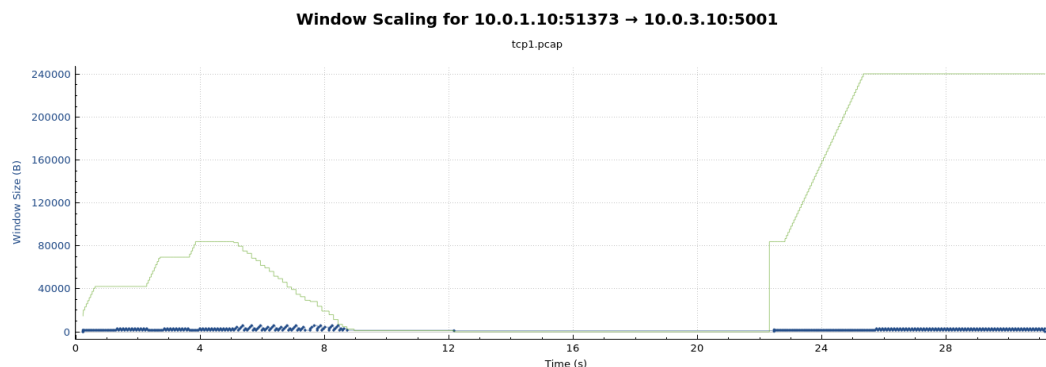
Figura 9: Control de flujo

280	12.168621	1..	1..	51373	1090	TCP	[TCP Window Full] 51373 → complex-link(5001) [PSH, ACK] Seq=254873 Ack=1 Win=14608 Len=1024 TSval=217066 TSecr=217066
281	12.168664	1..	1..	comm..	66	TCP	[TCP ZeroWindow] complex-link(5001) → 51373 [ACK] Seq=1 Ack=255897 Win=0 Len=0 TSval=217084 TSecr=217066
282	15.336591	1..	1..	51373	66	TCP	[TCP Keep-Alive] 51373 → complex-link(5001) [ACK] Seq=255896 Ack=1 Win=14608 Len=0 TSval=217088 TSecr=217084
283	15.336634	1..	1..	comm..	66	TCP	[TCP ZeroWindow] complex-link(5001) → 51373 [ACK] Seq=1 Ack=255897 Win=0 Len=0 TSval=217076 TSecr=217066
284	21.521222	1..	1..	51373	66	TCP	[TCP Keep-Alive] 51373 → complex-link(5001) [ACK] Seq=255896 Ack=1 Win=14608 Len=0 TSval=219404 TSecr=217876
285	21.521376	1..	1..	comm..	66	TCP	[TCP ZeroWindow] complex-link(5001) → 51373 [ACK] Seq=1 Ack=255897 Win=0 Len=0 TSval=219423 TSecr=217066
286	22.310994	1..	1..	comm..	66	TCP	[TCP Window Update] complex-link(5001) → 51373 [ACK] Seq=1 Ack=255897 Win=83984 Len=0 TSval=219620 TSecr=217066

En la Figura 10 se encuentra un gráfico que expone el comportamiento dinámico de la ventana de recepción (rwnd) del servidor. Notar que los instantes en los que parece crecer linealmente, se corresponden a segmentos con el flag PSH (push) de parte del cliente, que obligan al servidor a

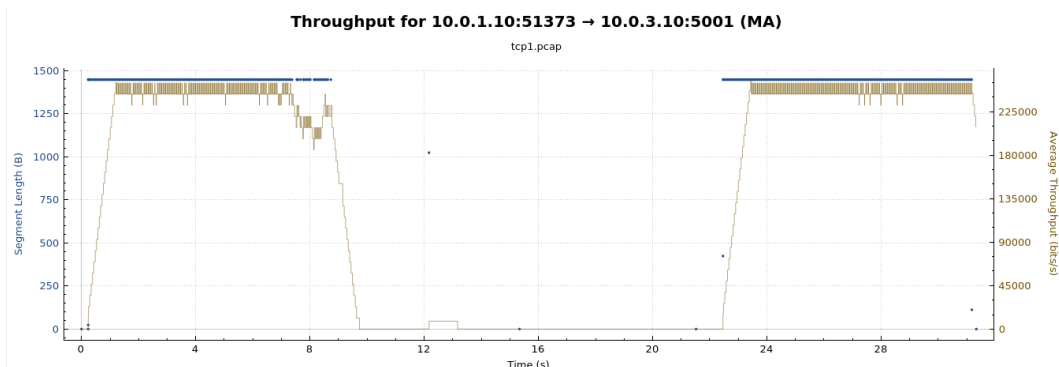
transmitir los datos en su buffer hacia su correspondiente aplicación. Es lógico que *rwnd* incremente luego de esa acción, ya que el servidor tiene más espacio en su buffer para recibir nuevos segmentos.

Figura 10: Ventana de recepción del servidor



En la figura 11 se encuentra un gráfico que expone el throughput desde el cliente hacia el servidor. Como es esperable, la velocidad de transmisión (en bits/segundo) es nula en el momento en que se instaura el control de flujo.

Figura 11: Throughput



## 1.6. Control de congestión

El control de congestión implica aproximar constantemente cual es el nivel de congestión en la red. Para ello, TCP hace uso de una ventana de congestión (*cwnd*), que es calculada dinámicamente por el emisor. Dicha ventana suele comenzar en un valor reducido, y, a medida que los segmentos son confirmados por la estación receptora, es incrementada para permitir un mayor flujo de datos. No obstante, si el emisor de los segmentos no recibe una confirmación (ACK) dentro de un cierto tiempo (Retransmission Time Out, RTO), o bien recibe un ACK duplicado (DUP ACK), entonces la ventana será reducida.

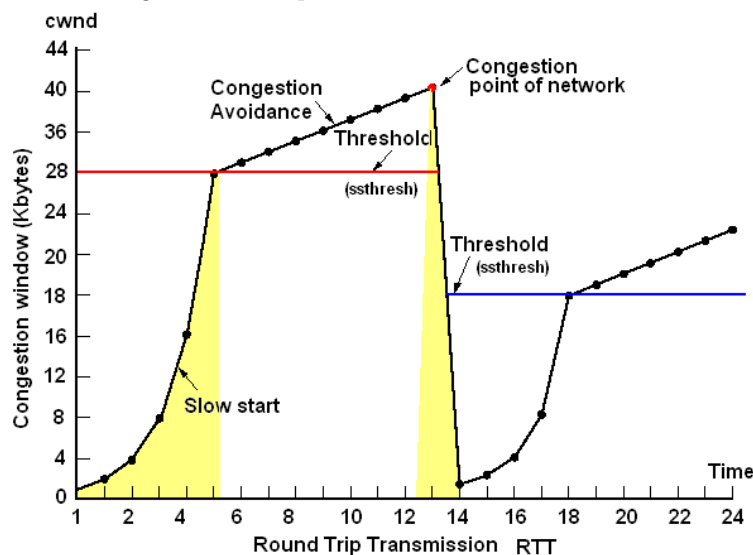
En una red, tanto el control de flujo como el control de congestión pueden ocurrir. Así, el valor de ventana que efectivamente utilizará una estación es  $W = \min(cwnd, rwnd)$ . En estado estático, el comportamiento del control de congestión de TCP tenderá a lograr la conservación de la cantidad de segmentos en la red.

El algoritmo de slow start es implementado cuando se establece una nueva conexión TCP o cuando se detectaron pérdidas de segmentos. Slow start presenta un crecimiento exponencial de



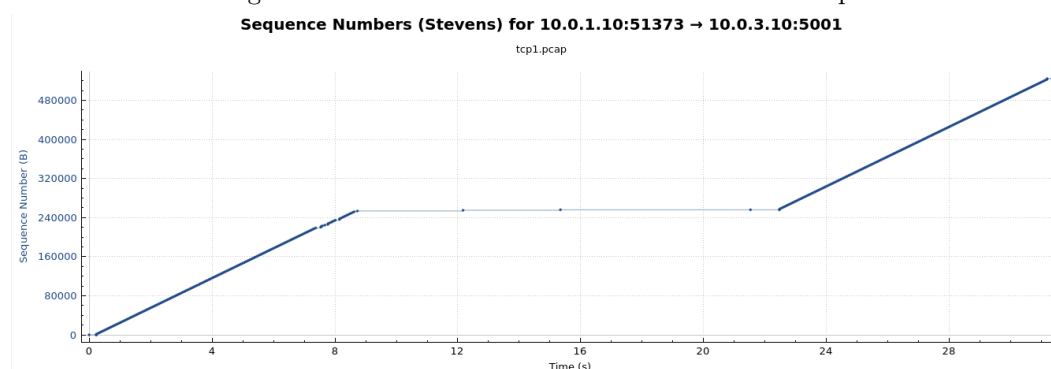
la ventana de congestión (cwnd). Una vez que se detectan pérdidas de segmentos, el algoritmo cambia a congestion avoidance, en donde se observa un crecimiento aproximadamente lineal de la ventana. Esto se evidencia en la Figura 12.

Figura 12: Comportamiento dinámico de cwnd



En la figura 13 se encuentra el gráfico exportado desde wireshark que indica los números de secuencia de los segmentos en función del tiempo. Siendo que dicha curva es siempre creciente, se concluye que no existen retransmisiones. Otra forma de verificarlo es usando el filtro tcp.analysis.retransmission. Por lo tanto, en ningún instante de la captura se llega a detectar congestión, y tampoco parecieran observarse etapas como slow start o congestion avoidance.

Figura 13: Números de secuencia en función del tiempo



## 1.7. Round Trip Time (RTT)

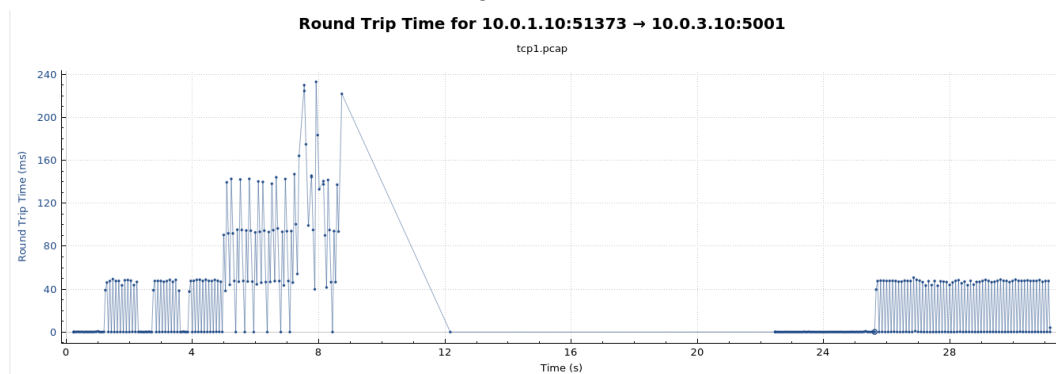
En la figura 14 se encuentra representado un gráfico del RTT. Este valor puede calcularse a partir de la siguiente fórmula:

$$RTT = t_{ACK} - t_{datos}$$

En la captura bajo estudio, pareciera que los segmentos son enviados desde el cliente cada

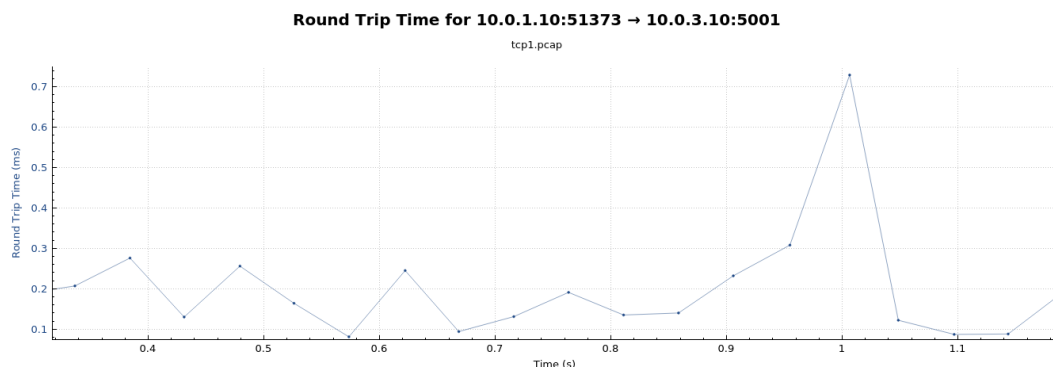
aproximadamente 50 ms. No obstante, hay casos en los que múltiples segmentos son confirmados por un mismo ACK del servidor. Esto genera que en los cálculos del RTT que realiza el cliente se vea un incremento igual al retardo que existe entre la transmisión de segmentos consecutivos por parte del cliente (de 50 ms). Por eso, en el gráfico pareciera que el RTT oscila en valores múltiplos de 50 ms, según la cantidad de segmentos que un mismo ACK confirme. El RRT real es mucho menor a 50 ms, y coincide con los valores que se encuentran más cercanos a 0 en el gráfico.

Figura 14: RTT



En la Figura 15 se observa el RRT para los casos en los que no existen retardos adicionales causados por la confirmación simultánea de múltiples segmentos. Se observa que el RRT propio de la red ronda entre los 0.1 y los 0.7ms.

Figura 15: RTT: valores reales



## 2. Ruteo externo

### 2.1. Cliente y servidor

En la Figura 16 se encuentra la captura que se analizará. En las primeras tres líneas se observa el Three Way Handshake, a partir del cual se puede concluir que el cliente es la IP 192.168.12.1 a través del puerto 37019, mientras que el servidor es la IP 192.168.12.2 a través del puerto 179 (bgp).

Figura 16: Captura bajo análisis

time	Source	Destination	Source Length	Protocol	Info
1 0.000000	192.168.12.1	192.168.12.2	37019	60 TCP	37019 → bgp(179) [SYN] Seq=0 Win=16384 Len=0 MSS=1460
2 0.006619	192.168.12.2	192.168.12.1	bgp	58 TCP	bgp(179) → 37019 [SYN, ACK] Seq=0 Ack=1 Win=16384 Len=0 MSS=1460
3 0.009725	192.168.12.1	192.168.12.2	37019	60 TCP	37019 → bgp(179) [ACK] Seq=1 Ack=1 Win=16384 Len=0
4 0.020192	192.168.12.1	192.168.12.2	37019	111 BGP	OPEN Message
5 0.028136	192.168.12.2	192.168.12.1	bgp	54 TCP	bgp(179) → 37019 [ACK] Seq=1 Ack=58 Win=16327 Len=0
6 0.032618	192.168.12.2	192.168.12.1	bgp	111 BGP	OPEN Message
7 0.035073	192.168.12.2	192.168.12.1	bgp	73 BGP	KEEPALIVE Message
8 0.038837	192.168.12.1	192.168.12.2	37019	60 TCP	37019 → bgp(179) [ACK] Seq=58 Ack=77 Win=16308 Len=0
9 0.042187	192.168.12.1	192.168.12.2	37019	73 BGP	KEEPALIVE Message
10 0.049259	192.168.12.1	192.168.12.2	37019	73 BGP	KEEPALIVE Message
11 0.051291	192.168.12.1	192.168.12.2	37019	131 BGP	UPDATE Message, UPDATE Message
12 0.054598	192.168.12.2	192.168.12.1	bgp	73 BGP	KEEPALIVE Message
13 0.057318	192.168.12.2	192.168.12.1	bgp	131 BGP	UPDATE Message, UPDATE Message
14 0.062042	192.168.12.1	192.168.12.2	37019	60 TCP	37019 → bgp(179) [ACK] Seq=173 Ack=173 Win=16212 Len=0
15 0.062547	192.168.12.2	192.168.12.1	bgp	54 TCP	bgp(179) → 37019 [ACK] Seq=173 Ack=173 Win=16212 Len=0

### 2.2. Descripción de la captura

La captura trata de la visualización de los mensajes TCP y BGP transmitidos en una conexión entre dos routers de borde. Esto se evidencia a partir de los mensajes OPEN intercambiados, como se observa en las figuras 17 y 18. Allí se observa que los campos "My AS", que hacen referencia la sistema autónomo, son diferentes entre los dos extremos. En consecuencia, los routers involucrados están en SAs distintos.

Figura 17: Mensaje OPEN: Cliente a servidor

No.	time	Source	Destination	Source	Length	Protocol	Info
4	0.020192	192.168.12.1	192.168.12.2	37019	111 BGP	OPEN Message	
8	0.038837	192.168.12.1	192.168.12.2	37019	60 TCP	37019 → bgp(179) [ACK] Seq=58 Ack=77 Win=16308 Len=0	
<div> <div>Border Gateway Protocol - OPEN Message</div> <div> <div>Marker: ffffffffffffffffffffffffffffffff</div> <div>Length: 57</div> <div>Type: OPEN Message (1)</div> <div>Version: 4</div> <div>My AS: 1</div> <div>Hold Time: 180</div> <div>BGP Identifier: 1.1.1.1</div> <div>Optional Parameters Length: 28</div> </div> </div>							

Figura 18: Mensaje OPEN: Servidor a cliente

time	Source	Destination	Source	Length	Protocol	Info
6 0.032618	192.168.12.2	192.168.12.1	bgp	111 BGP	BGP	OPEN Message
7 0.035073	192.168.12.2	192.168.12.1	bgp	73 BGP	BGP	KEEPALIVE Message

▼ Border Gateway Protocol - OPEN Message
Marker: ffffffffffffffffffffffffffffffffff
Length: 57
Type: OPEN Message (1)
Version: 4
My AS: 2
Hold Time: 180
BGP Identifier: 2.2.2.2
Optional Parameters Length: 28

### 2.3. Utilización de direcciones privadas

Dado que las direcciones públicas escasean y son costosas, es una práctica común utilizar direcciones privadas dentro de una misma organización. Generalmente, una organización cuenta con unas pocas direcciones IP públicas asignadas por un proveedor de servicios de internet (ISP), las cuales son compartidas por múltiples hosts que se encuentran interconectados entre sí con redes privadas. Esto se logra a través de un procedimiento de traducción de direcciones de red (NAT, Network Address translation) que realizan los routers intermediarios entre la red de la organización y la IP pública conectada a Internet. En el caso de la captura estudiada, si se asume que los SAs pertenecen a una misma organización, es lógico que se utilicen direcciones privadas entre los dos routers de borde, para así evitar el uso innecesario de IPs públicas.

Otro motivo por el que pueden estar utilizándose IPs privadas es una cuestión de seguridad, ya que la utilización de IPs privadas conlleva varios riesgos, como la exposición de la ubicación física de la IP, riesgo de ataques de hackers y pérdida de privacidad, entre otros.

### 2.4. Análisis de la conexión BGP

De acuerdo a la RFC 4271, una vez que la conexión TCP es establecida, el primer mensaje enviado por cada extremo es un mensaje OPEN, y si dicho OPEN es aceptable, un mensaje KEEPALIVE es enviado por el otro router para confirmar su recepción. Así, se observa que el KEEPALIVE en la línea 7 confirma al OPEN de la línea 4, y que el KEEPALIVE en la línea 9 confirma al OPEN de la línea 6. Luego, la conexión BGP se establece con éxito.

Los mensaje UPDATE tienen la función de informar nuevas redes a las que el router tiene acceso. En la Figura 19 se puede visualizar el envío del mensaje UPDATE desde la IP 192.168.12.1, en donde se observa lo siguiente:

- El campo Withdrawn Routes Length tiene un valor de 0, por lo que no hay rutas que serán retiradas.
- El atributo ORIGIN, que es obligatorio y bien conocido, indica que el NLRI se aprendió a través de IGP en el SA de origen.
- El atributo AS\_PATH indica que el NLRI proviene del SA 1. Este atributo es útil para evitar lazos y elegir rutas óptimas. El Segment type es AS\_SEQUENCE, indicando que la lista de SAs recorridos está ordenada (aunque hay un sólo SA en dicha lista).

- El atributo NEXT\_HOP es la IP 192.168.12.1. Esto significa que el router se ha nombrado a si mismo como NEXT\_HOP para llegar a la ruta indicada en el NLRI. De acuerdo a la RFC 4271, sección 5.1.3, esto se corresponde a un UPDATE enviado hacia un router externo.
- De acuerdo al NLRI, la red que se anuncia es la 1.1.1.0/24.

El otro mensaje UPDATE (ver Figura 20), enviado desde la IP 192.168.12.2, se puede analizar de forma analoga. Allí, se anuncia la red 2.2.2.0/24 al router de borde en el SA 1.

Figura 19: Mensaje UPDATE

No.	time	Source	Destination	Source	Length	Protocol	Info
11	0.051291	192.168.12.1	192.168.12.2	37019	131	BGP	UPDATE Message, UPDATE Message
12	0.054598	192.168.12.2	192.168.12.1	bgp	73	BGP	KEEPALIVE Message

Border Gateway Protocol - UPDATE Message

Marker: ffffffffffffffffffffffffffffffff

Length: 54

Type: UPDATE Message (2)

Withdrawn Routes Length: 0

Total Path Attribute Length: 27

Path attributes

Path Attribute - ORIGIN: IGP

Path Attribute - AS\_PATH: 1

Flags: 0x40, Transitive, Well-known, Complete

Type Code: AS\_PATH (2)

Length: 6

AS Path segment: 1

Segment type: AS\_SEQUENCE (2)

Segment length (number of ASN): 1

AS4: 1

Path Attribute - NEXT\_HOP: 192.168.12.1

Path Attribute - MULTI\_EXIT\_DISC: 0

Network Layer Reachability Information (NLRI)

1.1.1.0/24

Figura 20: Mensaje UPDATE

13	0.057318	192.168.12.2	192.168.12.1	bgp	131	BGP	UPDATE Message, UPDATE Message
----	----------	--------------	--------------	-----	-----	-----	--------------------------------

Border Gateway Protocol - UPDATE Message

Marker: ffffffffffffffffffffffffffffffff

Length: 54

Type: UPDATE Message (2)

Withdrawn Routes Length: 0

Total Path Attribute Length: 27

Path attributes

Path Attribute - ORIGIN: IGP

Path Attribute - AS\_PATH: 2

Path Attribute - NEXT\_HOP: 192.168.12.2

Path Attribute - MULTI\_EXIT\_DISC: 0

Network Layer Reachability Information (NLRI)

2.2.2.0/24

## 2.5. KEEPALIVES

De acuerdo a la RFC 4271, se recomienda que el tiempo entre la transmisión de KEEPALIVES consecutivos sea como máximo un tercio del hold timer. En este caso, sería un tercio de 180 segundos, es decir, 60 segundos como máximo entre sucesivos KEEPALIVES. Además, la RFC 4271 indica que obligatoriamente no deben enviarse KEEPALIVES con una frecuencia mayor a uno por segundo. Teniendo esto en consideración, se concluye que el ritmo en el que se envían los KEEPALIVES excede lo permitido por la RFC 4271. En efecto, en las líneas 9 y 10 se observa que dos KEEPALIVES consecutivos con el mismo origen se transmiten con 7 ms de diferencia.

## 2.6. Esquema de conexiones propuesto

De acuerdo a la información en la captura y a partir de los análisis aquí realizados, se deduce que el esquema de conexiones podría ser el expuesto en la Figura 21.

Figura 21: Esquema de conexiones

