

Universidad de los Andes
Departamento de Sistemas y Computación
Infraestructura de Comunicaciones

Laboratorio 5

1. Presente en una tabla la información obtenida. ¿todas las aplicaciones cliente utilizan el mismo puerto local? Explique.

No todas las aplicaciones cliente usan el mismo puerto local, porque para diferenciarse deben utilizar un puerto diferente, por ejemplo, si se corren dos clientes en la misma máquina la forma de diferenciarse es por medio de puertos diferentes. Además, cada thread o ejecución de un nuevo cliente corre en un puerto diferente para la misma máquina. Esto también se puede apreciar en la imagen 2, donde cada archivo muestra un puerto diferente.

2. Presente los resultados obtenidos de las pruebas anteriores en una tabla. Desde Wireshark verifique la información UDP capturada.

Prueba con 1 cliente:

Obj. Enviados	Obj. Faltantes	Tiempo promedio envío de mensajes (ms)
1000	957	365
10000	9703	240
100000	98915	316

Prueba con 3 clientes:

Obj. Recibidos	Obj. Faltantes	Tiempo promedio envío de mensajes (ms)
1000	935	267
1000	755	218
1000	824	329
10000	9453	353
10000	9539	373
10000	9499	364
100000	98874	395
100000	98939	304
100000	99901	328

Se puede ver que, al ser una transmisión de objetos por UDP, se pierde gran porcentaje de los paquetes enviados en los cuales se representan los objetos que se quieren enviar. A medida que se aumenta la cantidad de objetos como de clientes es mayor el porcentaje de pérdida de objetos y funciona correctamente para un número de objetos pequeño (menores a 100). El tiempo promedio tiene un comportamiento similar y parece no verse afectado por el número de objetos enviados, aunque para valores altos se observan promedios un poco mayores que en las demás pruebas.

No.	Time	Source	Destination	Protocol	Length	Info
21	6.455959	192.168.1.10	224.0.0.251	IGMPv2	46	Membership Report group 224.0.0.251
22	6.757362	fe80::e241:36ff:fea0:6...	ff02::1	ICMPv6	94	Router Advertisement from e0:41:36:a0:64:f0
23	6.759112	fe80::e241:36ff:fea0:6...	ff02::1	ICMPv6	94	Router Advertisement from e0:41:36:a0:64:f0
24	6.942898	192.168.1.2	13.59.205.175	UDP	204	63333 → 5000 Len=162
25	6.943602	192.168.1.2	13.59.205.175	UDP	204	63333 → 5000 Len=162
26	6.943603	192.168.1.2	13.59.205.175	UDP	204	63333 → 5000 Len=162
27	6.943603	192.168.1.2	13.59.205.175	UDP	204	63333 → 5000 Len=162
28	6.943702	192.168.1.2	13.59.205.175	UDP	204	63333 → 5000 Len=162
29	6.943829	192.168.1.2	13.59.205.175	UDP	204	63333 → 5000 Len=162
30	6.944031	192.168.1.2	13.59.205.175	UDP	204	63333 → 5000 Len=162
31	6.944182	192.168.1.2	13.59.205.175	UDP	204	63333 → 5000 Len=162
32	6.944306	192.168.1.2	13.59.205.175	UDP	204	63333 → 5000 Len=162
33	6.944455	192.168.1.2	13.59.205.175	UDP	204	63333 → 5000 Len=162
34	6.944615	192.168.1.2	13.59.205.175	UDP	204	63333 → 5000 Len=162
35	6.944786	192.168.1.2	13.59.205.175	UDP	204	63333 → 5000 Len=162
36	6.944873	192.168.1.2	13.59.205.175	UDP	204	63333 → 5000 Len=162
37	6.944961	192.168.1.2	13.59.205.175	UDP	204	63333 → 5000 Len=162
38	6.945074	192.168.1.2	13.59.205.175	UDP	204	63333 → 5000 Len=162
39	6.945190	192.168.1.2	13.59.205.175	UDP	204	63333 → 5000 Len=162
40	6.945281	192.168.1.2	13.59.205.175	UDP	204	63333 → 5000 Len=162
41	6.945350	192.168.1.2	13.59.205.175	UDP	204	63333 → 5000 Len=162
42	6.945426	192.168.1.2	13.59.205.175	UDP	204	63333 → 5000 Len=162
43	6.945550	192.168.1.2	13.59.205.175	UDP	204	63333 → 5000 Len=162
44	6.945669	192.168.1.2	13.59.205.175	UDP	204	63333 → 5000 Len=162
45	6.945787	192.168.1.2	13.59.205.175	UDP	204	63333 → 5000 Len=162
46	6.945896	192.168.1.2	13.59.205.175	UDP	204	63333 → 5000 Len=162
47	6.946007	192.168.1.2	13.59.205.175	UDP	204	63333 → 5000 Len=162
48	6.946140	192.168.1.2	13.59.205.175	UDP	204	63333 → 5000 Len=162
49	6.946268	192.168.1.2	13.59.205.175	UDP	204	63333 → 5000 Len=162

Imagen 1: Captura de objetos UDP

Utilizando Wireshark se pudo comprobar cómo se enviaban los objetos vía UDP, se puede ver que se envían desde el puerto 63333 hacia el 5000 que es el que está recibiendo la información en el servidor. Además, se puede probar que efectivamente el archivo que se crea con la respuesta viene de ese puerto.

```

Número de objetos recibidos: 44
Número de objetos faltantes: 56
Tiempo promedio de envío de mensajes: 252.0ms
[ec2-user@ip-172-31-20-172 ~]$ cd data/
[ec2-user@ip-172-31-20-172 data]$ ls
161.10.9.72:55926.txt 161.10.9.72:60071.txt 161.10.9.72:60175.txt
161.10.9.72:60070.txt 161.10.9.72:60072.txt 161.10.9.72:63333.txt
[ec2-user@ip-172-31-20-172 data]$

```

Imagen 2: Archivos con sus puertos respectivos para UDP con objetos.

- Archivos UDP → Repita las pruebas anteriores modificando el tamaño del buffer de envío con los siguientes valores: mínimo tamaño permitido del buffer, máximo tamaño permitido del buffer, un valor intermedio. Realice la misma prueba varias veces (al menos en 3 ocasiones) para que analice el comportamiento de las aplicaciones. Valide en cada prueba si el archivo fue recibido correctamente.

Tamaño del archivo	Tiempo promedio envío de mensajes (ms)	¿Recibido correctamente?
13KB	6199	Sí
5.4MB	7244	No
52.8MB	5531	No

Para estas pruebas se enviaron los tres tipos de archivo, en el único caso en que fue exitoso fue para los archivos pequeños. Para validar que el archivo llegara correctamente se realizó la implementación de un hash. A partir de esto se pudo ver que, al no tener una transmisión confiable de información, algunos de los paquetes enviados usando el socket de datagrama se pierden y es por esta razón que el hash calculado en ambos lados del proceso es diferente y el archivo no llega correctamente, por lo que para unos casos no puede visualizar. De igual forma,

como no se implementan técnicas de redundancia ni de control de errores al enviar más paquetes es más probable que alguno se pierda por eso únicamente funciona para archivos pequeños. De igual forma, el tiempo promedio aumenta a medida que el tamaño del archivo es mayor, como era de esperarse.

4. Archivos UDP → Verifique la captura realizada en Wireshark para cada uno de los escenarios y cada una de las pruebas, identifique el tamaño de los datagramas UDP y la cantidad de datagramas enviados.

No.	Time	Source	Destination	Protocol	Length	Info
32053	13.775306	192.168.1.2	18.188.139.50	UDP	554	58460 → 5000 Len=512
32054	13.775307	192.168.1.2	18.188.139.50	UDP	554	58460 → 5000 Len=512
32055	13.775307	192.168.1.2	18.188.139.50	UDP	554	58460 → 5000 Len=512
32056	13.775307	192.168.1.2	18.188.139.50	UDP	554	58460 → 5000 Len=512
32057	13.775307	192.168.1.2	18.188.139.50	UDP	554	58460 → 5000 Len=512
32058	13.775308	192.168.1.2	18.188.139.50	UDP	554	58460 → 5000 Len=512
32059	13.775308	192.168.1.2	18.188.139.50	UDP	554	58460 → 5000 Len=512
32060	13.775308	192.168.1.2	18.188.139.50	UDP	554	58460 → 5000 Len=512
32061	13.775308	192.168.1.2	18.188.139.50	UDP	554	58460 → 5000 Len=512
32062	13.775309	192.168.1.2	18.188.139.50	UDP	554	58460 → 5000 Len=512
32063	13.775309	192.168.1.2	18.188.139.50	UDP	554	58460 → 5000 Len=512
32064	13.775310	192.168.1.2	18.188.139.50	UDP	554	58460 → 5000 Len=512
32065	13.775310	192.168.1.2	18.188.139.50	UDP	554	58460 → 5000 Len=512
32066	13.775310	192.168.1.2	18.188.139.50	UDP	554	58460 → 5000 Len=512
32067	13.775311	192.168.1.2	18.188.139.50	UDP	554	58460 → 5000 Len=512
32068	13.775311	192.168.1.2	18.188.139.50	UDP	554	58460 → 5000 Len=512
32069	13.775311	192.168.1.2	18.188.139.50	UDP	554	58460 → 5000 Len=512
32070	13.775311	192.168.1.2	18.188.139.50	UDP	554	58460 → 5000 Len=512
32071	13.775311	192.168.1.2	18.188.139.50	UDP	554	58460 → 5000 Len=512
32072	13.775312	192.168.1.2	18.188.139.50	UDP	554	58460 → 5000 Len=512
32073	13.775312	192.168.1.2	18.188.139.50	UDP	554	58460 → 5000 Len=512
32074	13.775312	192.168.1.2	18.188.139.50	UDP	554	58460 → 5000 Len=512
32075	13.775312	192.168.1.2	18.188.139.50	UDP	554	58460 → 5000 Len=512
32076	13.775313	192.168.1.2	18.188.139.50	UDP	554	58460 → 5000 Len=512
32077	13.775313	192.168.1.2	18.188.139.50	UDP	554	58460 → 5000 Len=512
32078	13.775313	192.168.1.2	18.188.139.50	UDP	554	58460 → 5000 Len=512
32079	13.775313	192.168.1.2	18.188.139.50	UDP	554	58460 → 5000 Len=512
32080	13.784023	192.168.1.2	18.188.139.50	UDP	554	58460 → 5000 Len=512
32081	13.784024	192.168.1.2	18.188.139.50	UDP	554	58460 → 5000 Len=512

Imagen 3: Captura de archivos UDP

A partir de la captura de Wireshark se puede ver que los archivos se envían por el protocolo UDP del puerto local 58460 al puerto 5000 en el servidor de la máquina de AWS. Si se observa al detalle una parte de la captura se puede ver que el tamaño de los datagramas es de 520. Al igual que en la siguiente imagen donde se puede ver claramente su tamaño.

▶	Frame 501: 554 bytes on wire (4432 bits), 554 bytes captured (4432 bits) on interface 0
▶	Ethernet II, Src: Apple_b7:15:b4 (6c:40:08:b7:15:b4), Dst: Mitrasa_a0:64:f0 (e0:41:36:a0:64:f0)
▶	Internet Protocol Version 4, Src: 192.168.1.2, Dst: 18.188.139.50
▼	User Datagram Protocol, Src Port: 58460, Dst Port: 5000
	Source Port: 58460
	Destination Port: 5000
	Length: 520
	Checksum: 0xab23 [unverified]
	[Checksum Status: Unverified]
	[Stream index: 6]
▶	Data (512 bytes)

Imagen 4: Especificación del protocolo UDP con tamaño de datagrama.

5. Bittorrent → Verifique la captura realizada en Wireshark para cada uno de los escenarios y cada una de las pruebas, identifique el tamaño y la cantidad de paquetes intercambiados entre cliente y servidor.

Para las pruebas de este protocolo se utilizó un tracker local asociado a un torrent que ya tenía peers inscritos para que el protocolo se viera mejor en Wireshark. Esto se debe a que cuando se

hizo la prueba con un tracker y archivo local las capturas no mostraban mucho del protocolo Bittorrent. Además, se consideró que al tener el .torrent con previos peers las capturas eran más interesantes y se podía realizar un mejor análisis del protocolo. Al comenzar la descarga se llegan a tener hasta 357 y se realiza la conexión con varios de ellos, que vienen también de diferentes países. Para los clientes se hicieron pruebas con tres diferentes, uno en java con el que se midió el tiempo del paquete pequeño, Deluge y uTorrent para también probar la descarga del pequeño con pocos peers hacer de igual forma la descarga del paquete grande. En la Imagen xxxxxx se puede ver un pedazo de la captura de Wireshark con ese caso de prueba, y filtrando por el protocolo Bittorrent.

Aquí se puede ver como queda el tracker asociado al torrent y se ve cuando comienzan a hacerse descargas:

```

9 [main] INFO com.turn.ttorrent.common.Torrent - Multi-file torrent information:
9 [main] INFO com.turn.ttorrent.common.Torrent - Torrent name: Lettuce Live
9 [main] INFO com.turn.ttorrent.common.Torrent - Announced at:
9 [main] INFO com.turn.ttorrent.common.Torrent - 1. http://192.168.1.2:6969/announce
9 [main] INFO com.turn.ttorrent.common.Torrent - Created on.: Wed Mar 09 15:58:00 COT 2016
9 [main] INFO com.turn.ttorrent.common.Torrent - Found 14 file(s) in multi-file torrent structure.
9 [main] DEBUG com.turn.ttorrent.common.Torrent - 1. Lettuce Live/00_What's-Inside.html (4,563 byte(s))
0 [main] DEBUG com.turn.ttorrent.common.Torrent - 2. Lettuce Live/LETTUCE - Red Rocks 2015 - Chief.mp4 (488,021,576 byte(s))
0 [main] DEBUG com.turn.ttorrent.common.Torrent - 3. Lettuce Live/LETTUCE - Red Rocks 2015 - Get Greasy.mp4 (314,163,688 byte(s))
0 [main] DEBUG com.turn.ttorrent.common.Torrent - 4. Lettuce Live/LETTUCE - Red Rocks 2015 - Makin My Way Back Home w Nigel Hall.mp4 (569,982,160 byte(s))
0 [main] DEBUG com.turn.ttorrent.common.Torrent - 5. Lettuce Live/LETTUCE - Red Rocks 2015 - By Any Shmeeans Necessary _ Colorado Love _ By Any Shmeeans Ne
0 [main] DEBUG com.turn.ttorrent.common.Torrent - 6. Lettuce Live/LETTUCE - Red Rocks 2015 - Phyllis.mp4 (393,738,838 byte(s))
0 [main] DEBUG com.turn.ttorrent.common.Torrent - 7. Lettuce Live/LetUsPlay-TeaserTrailer.mp4 (76,964,330 byte(s))
0 [main] DEBUG com.turn.ttorrent.common.Torrent - 8. Lettuce Live/LetUsPlay-Teaser1.mp4 (52,827,122 byte(s))
0 [main] DEBUG com.turn.ttorrent.common.Torrent - 9. Lettuce Live/LetUsPlay-Teaser2.mp4 (66,409,697 byte(s))
0 [main] DEBUG com.turn.ttorrent.common.Torrent - 10. Lettuce Live/LetUsPlay-Teaser3.mp4 (60,804,720 byte(s))
1 [main] DEBUG com.turn.ttorrent.common.Torrent - 11. Lettuce Live/LetUsPlay-Teaser4.mp4 (40,214,886 byte(s))
1 [main] DEBUG com.turn.ttorrent.common.Torrent - 12. Lettuce Live/Phyllis_Official_Video.mp4 (463,716,304 byte(s))
1 [main] DEBUG com.turn.ttorrent.common.Torrent - 13. Lettuce Live/LetUsPlayPoster_instagram_linked.pdf (5,430,113 byte(s))
1 [main] DEBUG com.turn.ttorrent.common.Torrent - 14. Lettuce Live/Lettuce_crush_linked.pdf (96,265 byte(s))
1 [main] INFO com.turn.ttorrent.common.Torrent - Pieces.....: 681 piece(s) (4194304 byte(s)/piece)
1 [main] INFO com.turn.ttorrent.common.Torrent - Total size...: 2,852,979,320 byte(s)
1 [main] INFO com.turn.ttorrent.tracker.Tracker - Registered new torrent for 'Lettuce Live' with hash c1035a5dcb3a8772af22501ab5103862dd9c6a58.
4 [tracker:6969] INFO com.turn.ttorrent.tracker.Tracker - Starting BitTorrent tracker on http://0.0.0.0:6969/announce...
4 [peer-collector:6969] INFO com.turn.ttorrent.tracker.Tracker - Starting tracker peer collection for tracker at http://0.0.0.0:6969/announce...

```

Imagen 5: Tracker corriendo y su torrent asociado.

Se puede ver que inicialmente se hace un handshake con los peers que se encuentren disponibles, para ello buscan la IP del tracker y que se cumpla el hash, en la siguiente imagen se puede ver cómo cuando se hace el handshake se tiene el hash y el id del nuevo peer.

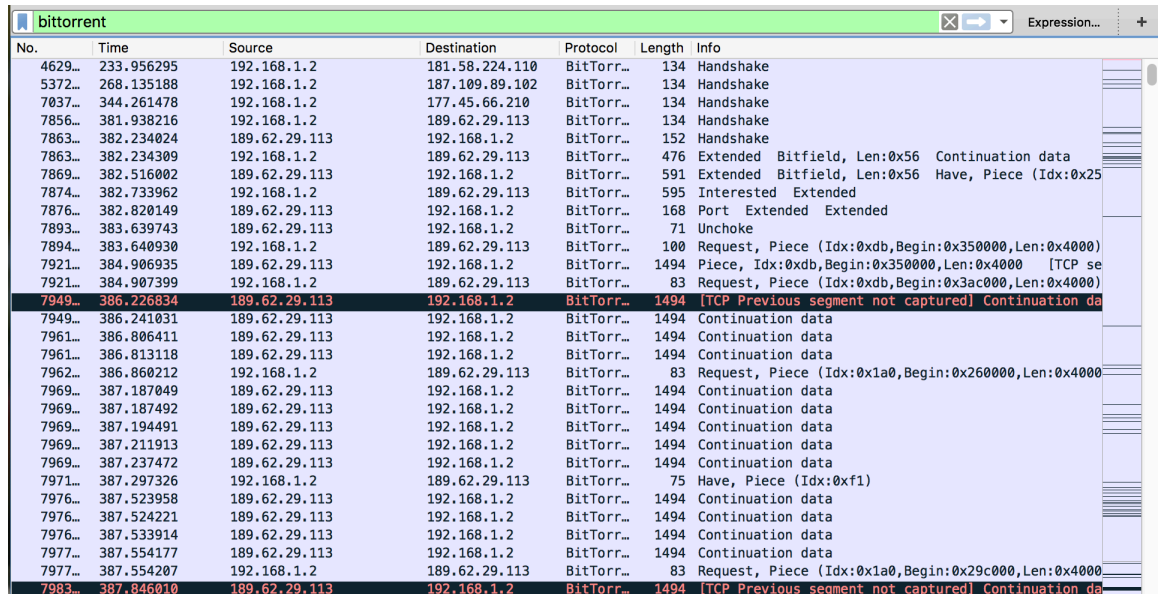
```

▼ BitTorrent
Protocol Name Length: 19
Protocol Name: BitTorrent protocol
Reserved Extension Bytes: 0000000000100005
SHA1 Hash of info dictionary: c1035a5dcb3a8772af22501ab5103862dd9c6a58
Peer ID: 2d5554333533532d46ad4c1147224982d182cb65

```

Imagen 6: Especificación del protocolo bittorrent para el handshake con un peer

Más adelante lo que se hace es mostrar el flujo de paquetes entre peers y además de esto, se ve cómo el cliente pide un pedazo del archivo y lo va recibiendo, dentro de este flujo.



No.	Time	Source	Destination	Protocol	Length	Info
4629...	233.956295	192.168.1.2	181.58.224.110	BitTorr...	134	Handshake
5372...	268.135188	192.168.1.2	187.109.89.102	BitTorr...	134	Handshake
7037...	344.261478	192.168.1.2	177.45.66.210	BitTorr...	134	Handshake
7856...	381.938216	192.168.1.2	189.62.29.113	BitTorr...	134	Handshake
7863...	382.234024	189.62.29.113	192.168.1.2	BitTorr...	152	Handshake
7863...	382.234309	192.168.1.2	189.62.29.113	BitTorr...	476	Extended Bitfield, Len:0x56 Continuation data
7869...	382.516002	189.62.29.113	192.168.1.2	BitTorr...	591	Extended Bitfield, Len:0x56 Have, Piece (Idx:0x25
7874...	382.733962	192.168.1.2	189.62.29.113	BitTorr...	595	Interested Extended
7876...	382.820149	189.62.29.113	192.168.1.2	BitTorr...	168	Port Extended Extended
7893...	383.639743	189.62.29.113	192.168.1.2	BitTorr...	71	Unchoke
7894...	383.640930	192.168.1.2	189.62.29.113	BitTorr...	100	Request, Piece (Idx:0xdb, Begin:0x350000, Len:0x4000)
7921...	384.906935	189.62.29.113	192.168.1.2	BitTorr...	1494	Piece, Idx:0xdb, Begin:0x350000, Len:0x4000 [TCP se
7921...	384.907399	192.168.1.2	189.62.29.113	BitTorr...	83	Request, Piece (Idx:0xdb, Begin:0x3ac000, Len:0x4000)
7949...	386.226834	189.62.29.113	192.168.1.2	BitTorr...	1494	[TCP Previous segment not captured] Continuation da
7949...	386.241031	189.62.29.113	192.168.1.2	BitTorr...	1494	Continuation data
7961...	386.806411	192.168.1.2	192.168.1.2	BitTorr...	1494	Continuation data
7961...	386.811318	189.62.29.113	192.168.1.2	BitTorr...	1494	Continuation data
7962...	386.860212	192.168.1.2	189.62.29.113	BitTorr...	83	Request, Piece (Idx:0x1a0, Begin:0x260000, Len:0x4000
7969...	387.187049	189.62.29.113	192.168.1.2	BitTorr...	1494	Continuation data
7969...	387.187492	189.62.29.113	192.168.1.2	BitTorr...	1494	Continuation data
7969...	387.194491	189.62.29.113	192.168.1.2	BitTorr...	1494	Continuation data
7969...	387.211913	189.62.29.113	192.168.1.2	BitTorr...	1494	Continuation data
7969...	387.237472	189.62.29.113	192.168.1.2	BitTorr...	1494	Continuation data
7971...	387.297326	192.168.1.2	189.62.29.113	BitTorr...	75	Have, Piece (Idx:0xf1)
7976...	387.523958	189.62.29.113	192.168.1.2	BitTorr...	1494	Continuation data
7976...	387.542221	189.62.29.113	192.168.1.2	BitTorr...	1494	Continuation data
7976...	387.533914	189.62.29.113	192.168.1.2	BitTorr...	1494	Continuation data
7977...	387.554177	189.62.29.113	192.168.1.2	BitTorr...	1494	Continuation data
7977...	387.554207	192.168.1.2	189.62.29.113	BitTorr...	83	Request, Piece (Idx:0x1a0, Begin:0x29c000, Len:0x4000
7983...	387.846010	189.62.29.113	192.168.1.2	BitTorr...	1494	[TCP Previous segment not captured] Continuation da

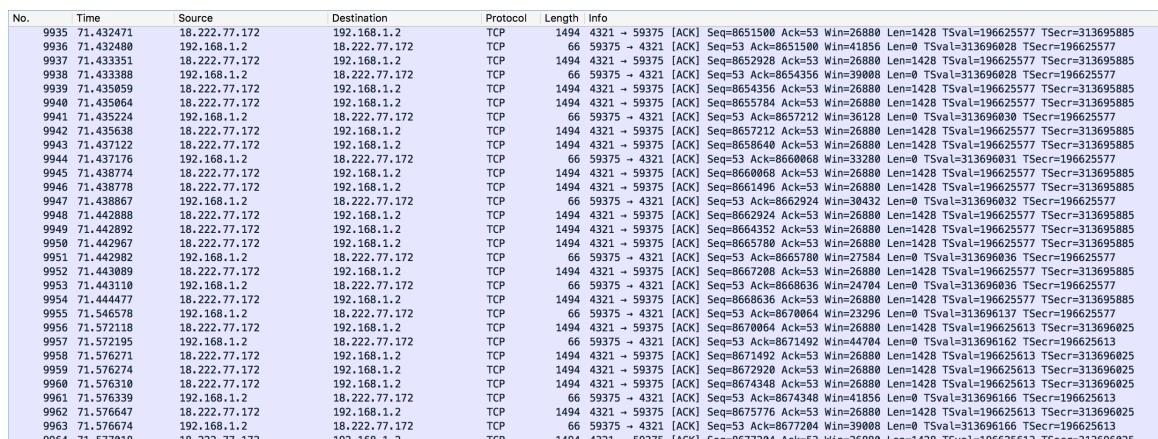
Imagen 7: Captura de Wireshark para un torrent con muchos peers.

Tiempo en descargar Visitor.torrent (48KB) desde un cliente con dos peers: 96ms

Tiempo en descargar Lettuce Live.torrent (2.7GB) desde un cliente con 357 peers: 35min 39 seg

Ambos archivos descargados llegan correctamente al cliente que los descarga, esto se debe a que Bittorrent ofrece el servicio de transporte seguro de información

- TCP → Verifique la captura realizada en Wireshark para cada uno de los escenarios y cada una de las pruebas, identifique el tamaño y la cantidad de paquetes intercambiados entre cliente y servidor.



No.	Time	Source	Destination	Protocol	Length	Info
9935	71.432471	18.222.77.172	192.168.1.2	TCP	1494	4321 → 59375 [ACK] Seq=8651500 Ack=53 Win=26800 Len=1428 TSval=196625577 TSecr=313695885
9936	71.432480	192.168.1.2	18.222.77.172	TCP	66	59375 → 4321 [ACK] Seq=53 Ack=8651500 Win=41856 Len=0 TSval=313696028 TSecr=196625577
9937	71.433351	18.222.77.172	192.168.1.2	TCP	1494	4321 → 59375 [ACK] Seq=8652928 Ack=53 Win=26800 Len=1428 TSval=196625577 TSecr=313695885
9938	71.433388	192.168.1.2	18.222.77.172	TCP	66	59375 → 4321 [ACK] Seq=53 Ack=8654356 Win=39008 Len=0 TSval=313696028 TSecr=196625577
9939	71.435059	18.222.77.172	192.168.1.2	TCP	1494	4321 → 59375 [ACK] Seq=8654356 Ack=53 Win=26800 Len=1428 TSval=196625577 TSecr=313695885
9940	71.435064	18.222.77.172	192.168.1.2	TCP	1494	4321 → 59375 [ACK] Seq=8655784 Ack=53 Win=26800 Len=1428 TSval=196625577 TSecr=313695885
9941	71.435224	192.168.1.2	18.222.77.172	TCP	66	59375 → 4321 [ACK] Seq=53 Ack=8657212 Win=36128 Len=0 TSval=313696030 TSecr=196625577
9942	71.435638	18.222.77.172	192.168.1.2	TCP	1494	4321 → 59375 [ACK] Seq=8657212 Ack=53 Win=26800 Len=1428 TSval=196625577 TSecr=313695885
9943	71.437122	18.222.77.172	192.168.1.2	TCP	1494	4321 → 59375 [ACK] Seq=8658040 Ack=53 Win=26800 Len=1428 TSval=196625577 TSecr=313695885
9944	71.437176	18.222.77.172	192.168.1.2	TCP	66	59375 → 4321 [ACK] Seq=53 Ack=8660068 Win=33200 Len=0 TSval=313696031 TSecr=196625577
9945	71.438774	18.222.77.172	192.168.1.2	TCP	1494	4321 → 59375 [ACK] Seq=8660068 Ack=53 Win=26800 Len=1428 TSval=196625577 TSecr=313695885
9946	71.438778	18.222.77.172	192.168.1.2	TCP	1494	4321 → 59375 [ACK] Seq=8661496 Ack=53 Win=26800 Len=1428 TSval=196625577 TSecr=313695885
9947	71.438867	192.168.1.2	18.222.77.172	TCP	66	59375 → 4321 [ACK] Seq=53 Ack=8662924 Win=30432 Len=0 TSval=313696032 TSecr=196625577
9948	71.442888	18.222.77.172	192.168.1.2	TCP	1494	4321 → 59375 [ACK] Seq=8662924 Ack=53 Win=26800 Len=1428 TSval=196625577 TSecr=313695885
9949	71.442892	18.222.77.172	192.168.1.2	TCP	1494	4321 → 59375 [ACK] Seq=8664352 Ack=53 Win=26800 Len=1428 TSval=196625577 TSecr=313695885
9950	71.442967	18.222.77.172	192.168.1.2	TCP	1494	4321 → 59375 [ACK] Seq=8665780 Ack=53 Win=26800 Len=1428 TSval=196625577 TSecr=313695885
9951	71.442982	192.168.1.2	18.222.77.172	TCP	66	59375 → 4321 [ACK] Seq=53 Ack=8665780 Win=27584 Len=0 TSval=313696036 TSecr=196625577
9952	71.443089	18.222.77.172	192.168.1.2	TCP	1494	4321 → 59375 [ACK] Seq=8667208 Ack=53 Win=26800 Len=1428 TSval=196625577 TSecr=313695885
9953	71.443118	192.168.1.2	18.222.77.172	TCP	66	59375 → 4321 [ACK] Seq=53 Ack=8668636 Win=24704 Len=0 TSval=313696036 TSecr=196625577
9954	71.444477	18.222.77.172	192.168.1.2	TCP	1494	4321 → 59375 [ACK] Seq=8668636 Ack=53 Win=26800 Len=1428 TSval=196625577 TSecr=313695885
9955	71.546578	192.168.1.2	18.222.77.172	TCP	66	59375 → 4321 [ACK] Seq=53 Ack=8670064 Win=23296 Len=0 TSval=313696137 TSecr=196625577
9956	71.572118	18.222.77.172	192.168.1.2	TCP	1494	4321 → 59375 [ACK] Seq=8670064 Ack=53 Win=26800 Len=1428 TSval=196625613 TSecr=313696025
9957	71.572195	192.168.1.2	18.222.77.172	TCP	66	59375 → 4321 [ACK] Seq=53 Ack=8671492 Win=44704 Len=0 TSval=313696162 TSecr=196625613
9958	71.576271	18.222.77.172	192.168.1.2	TCP	1494	4321 → 59375 [ACK] Seq=8671492 Ack=53 Win=26800 Len=1428 TSval=196625613 TSecr=313696025
9959	71.576274	18.222.77.172	192.168.1.2	TCP	1494	4321 → 59375 [ACK] Seq=8672928 Ack=53 Win=26800 Len=1428 TSval=196625613 TSecr=313696025
9960	71.576310	18.222.77.172	192.168.1.2	TCP	1494	4321 → 59375 [ACK] Seq=8674348 Ack=53 Win=26800 Len=1428 TSval=196625613 TSecr=313696025
9961	71.576339	192.168.1.2	18.222.77.172	TCP	66	59375 → 4321 [ACK] Seq=53 Ack=8674348 Win=41856 Len=0 TSval=313696166 TSecr=196625613
9962	71.576647	18.222.77.172	192.168.1.2	TCP	1494	4321 → 59375 [ACK] Seq=8675776 Ack=53 Win=26800 Len=1428 TSval=196625613 TSecr=313696025
9963	71.576674	192.168.1.2	18.222.77.172	TCP	66	59375 → 4321 [ACK] Seq=53 Ack=8677204 Win=39008 Len=0 TSval=313696166 TSecr=196625613

Imagen 8: Captura de Wireshark para TCP.

Se puede observar que al ser hay un intercambio de información en ambos sentidos lo que asegura que todo se transmita de manera confiable, asegurando que todos los bytes enviados lleguen correctamente, así como que haya control de flujo en el canal y no se sobre sature el servidor. Para este caso, todos los archivos enviados llegan correctamente sin fallas como se esperaba.

Tamaño del archivo	Número de paquetes intercambiados	¿Recibido correctamente?
1.7MB	2	Si
7.7MB	4	Si
74.3MB	30167	Si

Preguntas

¿Es posible desarrollar aplicaciones UDP que garanticen la entrega confiable de archivos? Que consideraciones deben tenerse en cuenta para garantizar un servicio de entrega confiable utilizando dicho protocolo. Justifique su respuesta.

Si es posible tener aplicaciones que garanticen la entrega confiable de archivos. Sin embargo, hay que realizar modificaciones en la capa de aplicación para que funcione correctamente. Una opción es tener un hash que se calcula tanto en el servidor como en el cliente con la información del archivo o mensaje que es enviado, de esta manera si el hash calculado corresponde con el hash que se envía del cliente quiere decir que la información llegó completa y se puede confiar en la información recibida. Por otro lado, hay otras técnicas para asegurar que todos los paquetes lleguen como enviar cada uno más de una vez y en el servidor revisar que todos los paquetes lleguen y que el contenido es correcto por la comparación del mismo paquete que es enviado más de una vez.

Presente el análisis de resultados y conclusiones sobre los diferentes escenarios de pruebas realizados, compare los resultados de los 3 protocolos estudiados.

Para terminar, TCP, UDP y Bittorrent son tres protocolos de transmisión de datos los cuales poseen características diferentes, que los hacen más o menos eficientes o adecuados de acuerdo con la necesidad que se necesite satisfacer. Para el caso de archivos grandes que se puedan enviar peer to peer el más adecuado es Bittorrent, para alta confiabilidad en la transmisión de datos el adecuado es TCP y para alta velocidad de transmisión y eficiencia UDP.

Es importante tener en cuenta que, aunque cualquiera de los tres puede ser usado indistintamente, es conveniente tener en cuenta sus fortalezas y debilidades. Especialmente evaluar aspectos como confiabilidad, eficiencia, tamaño del mensaje para decidir cuál de ellos es el que más se adapta a las características de la situación que se necesita resolver.

Lo anterior, pudo ser comprobado a partir de las pruebas realizadas sobre los tres protocolos, donde se confirma la confiabilidad de TCP, la rapidez de UDP pero al mismo tiempo si falla en la confiabilidad, y la eficiencia de Bittorrent al trabajar con varios peers y poder enviar archivos de mayor tamaño.