Práctica de diseño de protocolos P2P

Redes de Comunicaciones

Mourad Abbou Aazaz G2.1 Pablo José Rocamora Zamora G2.2

Febrero 24, 2017

Contents

Introducción. **3**
Resumen de los paquetes UDP: peer - tracker que usamos: 3
Resumen de los paquetes TCP: peer - peer que usamos: 3

Autómatas de ambos protocolos: **5**
Peer-peer: Autómata unificado 5
Peer-tracker: Cliente y servidor 6

Formato de representación de los mensajes **8**
Peer-peer 8
Peer-tracker 12

Conclusiones **20**

Introducción.

Resumen de los paquetes UDP: peer - tracker que usamos:

Formato del mensaje: CONTROL

Tipos que lo usan:

- Type = 1 (ADD_SEED_ACK)
- Type = 10 (REMOVE_SEED_ACK)

Formato del mensaje: FILEINFO

Tipos que lo usan:

- Type = 2 (ADD_SEED)
- Type = 6 (QUERY_FILES_RESPONSE)

Formato del mensaje: SEEDINFO

Tipos que lo usan:

- Type = 3 (GET_SEEDS)
- Type = 4 (SEED_LIST)

Formato del mensaje: SEEDQUERY

Tipos que lo usan:

- Type = 5 (QUERY_FILES)

Formato del mensaje: CHUNKINFO

Tipos que lo usan:

- Type = 7 (QUERY_CHUNK)
- Type = 8 (QUERY_CHUNK_RESPONSE)

Formato del mensaje: REMOVE

Tipos que lo usan:

- Type = 9 (REMOVE_SEED)

Resumen de los paquetes TCP: peer - peer que usamos:

Formato del mensaje: CHUNKQUERY

Tipos que lo usan:

- Type = 1 (GET_CHUNK)
- Type = 3 (QUERY_CHUNK)

Formato del mensaje: CHUNKQUERYRESPONSE

Tipos que lo usan:

- Type = 2 (GET_CHUNK_RESPONSE)
- Type = 4 (QUERY_CHUNK_RESPONSE)

Autómatas de ambos protocolos:

Peer-peer: Autómata unificado

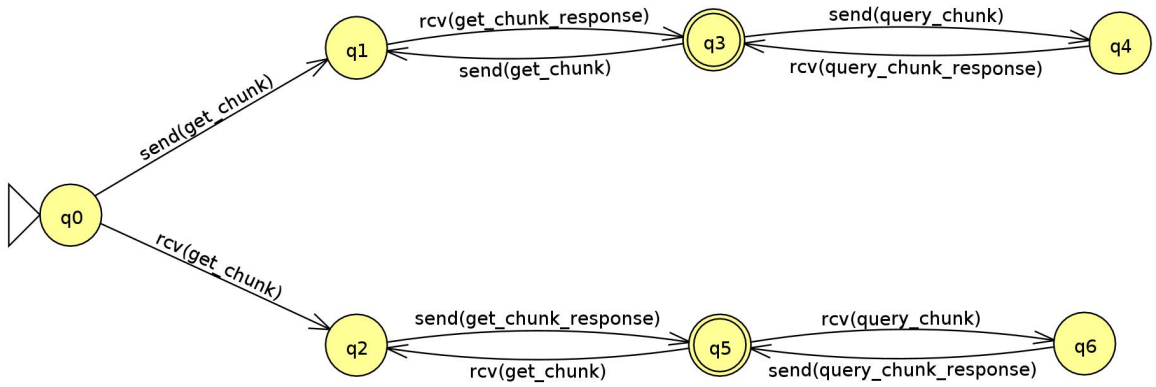

Figure 1: Automata para el peer (unificada) con TCP

Peer-tracker: Cliente y servidor

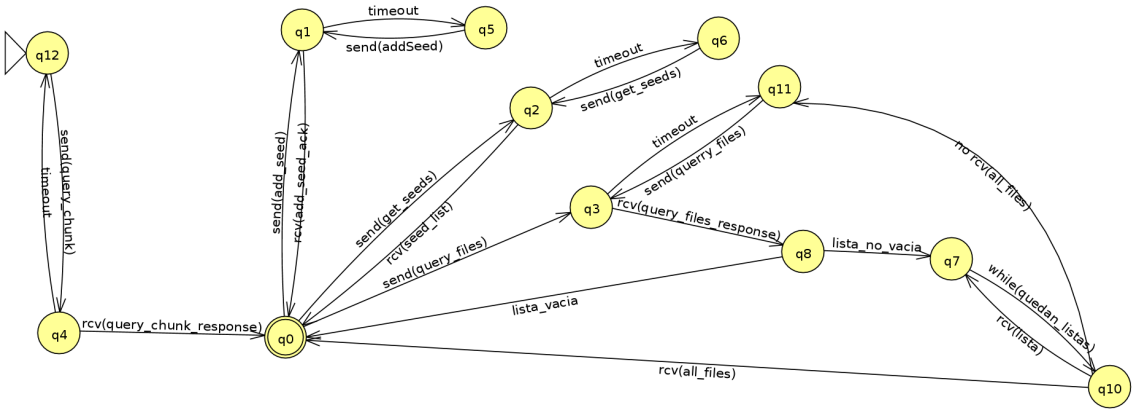

Figure 2: Automata del cliente con UDP

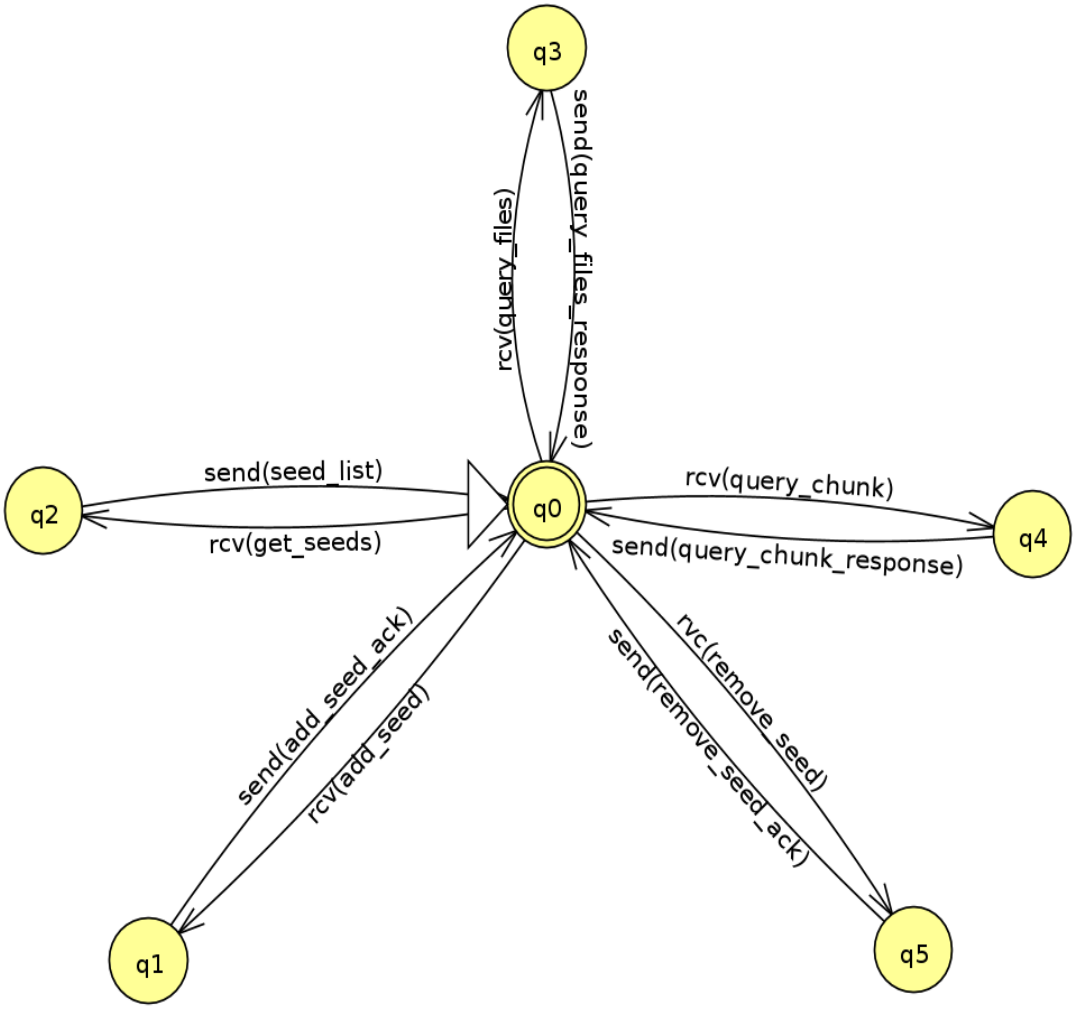

Figure 3: Automata del servidor con UDP

Formato de representación de los mensajes

Peer-peer

Formato del mensaje: CHUNKQUERY para el tipo GET_CHUNK

- Type = 1 (GET_CHUNK)
  - Formato del mensaje: CHUNKQUERY.
  - Un Peer solicita al otro Peer la lista de Chunks que tiene de un determinado fichero, que indica a través de su Hash.

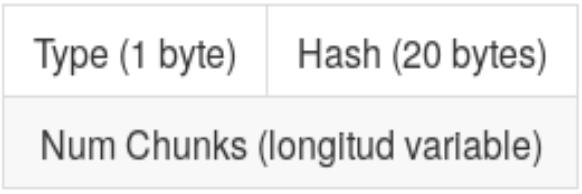

Figure 4: Formato del mensaje: CHUNKQUERY

Información del paquete:

- Type: Siempre sera 1 para indicar que es un GET_CHUNK.
- Hash: Hash del fichero del que deseamos saber los chunks disponibles.
- Num Chunks: No se usa, todo a 0.

Formato del mensaje: CHUNKQUERYRESPONSE para el tipo GET_CHUNK_RESPONSE

- Type = 2 (GET_CHUNK_RESPONSE)
  - Formato del mensaje: CHUNKQUERYRESPONSE.
  - Un Peer informa a otro Peer de la lista de chunks que tiene de un determinado fichero.

Figure 5: Formato del mensaje: CHUNKQUERYRESPONSE

Información del paquete:

- Type: Siempre sera 2 para indicar que es un GET_CHUNK_RESPONSE.
- Hash: Hash del fichero del que nos esta informando.
- Num Chunks: Numero de chunks que tiene para compartir, El tamaño del campo lo obtenemos con la formula: (sale abajo)[1], Si se tienen todos los chunks de un fichero este campo ira todo a 1 para indicarlo.

[1] $\log_2 \frac{TamañoMaximoDeUnfichero=2^{32}}{TamañoDeChunks}$

- Chunk: Chunks que tiene el peer, se repite n veces, siendo n: Num Chunks.

3.1 Ejemplo.

Un peer A solicita al otro peer B la lista de chunks que tiene de un fichero ubuntu14.04.iso (hash b9153318862f0f7b5f82c913ecb2117f97c3153e, tamaño 1.024.572.864 bytes)

1	b9153318862f0f7b5f82c913ecb2117f97c3153e
0000000...	

Figure 6: Ejemplo: GET_CHUNK

Un peer B informa a otro peer A de la lista de chunks que tiene de un determinado fichero listo para compartir ubuntu14.04.iso (hash b9153318862f0f7b5f82c913ecb2117f97c3153e, tamaño 1.024.572.864 bytes)

Le responde que tiene 4 chunks del fichero solicitado (1, 3, 4, 5)

2	b9153318862f0f7b5f82c913ecb2117f97c3153e
4	1
3	
4	
5	

Figure 7: Ejemplo: GET_CHUNK_RESPONSE

Formato del mensaje: CHUNKQUERY para el tipo QUERY_CHUNK

- Type = 3 (QUERY_CHUNK)
  - Formato del mensaje: CHUNKQUERY.
  - Un Peer solicita al otro Peer un Chunk de un fichero especifico indicado a través de su Hash.

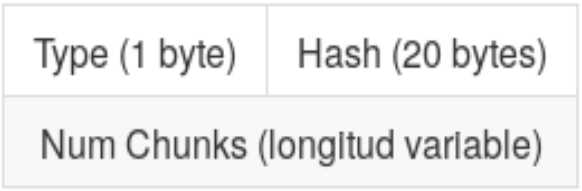

Figure 8: Formato del mensaje: CHUNKQUERY

Información del paquete:

- Type: Siempre sera 3 para indicar que es un QUERY_CHUNK
- Hash: Hash del fichero del que deseamos obtener un chunk.
- Num Chunks: Numero del chunk del fichero que solicita.

Formato del mensaje: CHUNKQUERYRESPONSE para el tipo QUERY_CHUNK_RESPONSE

- Type = 4 (QUERY_CHUNK_RESPONSE)
  - Formato del mensaje: CHUNKQUERYRESPONSE
  - Un Peer manda a otro Peer el chunk que le ha solicitado.

Figure 9: Formato del mensaje: CHUNKQUERYRESPONSE

Información del paquete:

- Type: Siempre sera 4 para indicar que es un QUERY_CHUNK_RESPONSE.
- Hash: Hash del fichero del que procede el chunk.
- Num Chunk: Numero del chunk del que procede el dato.
- Chunk: Dato del chunk, el tamaño se establece acorde al tamaño que nos indica el Tracker.

Un peer A solicita al otro peer B el chunk 30 del fichero ubuntu14.04.iso (hash b9153318862f0f7b5f82c913ecb2117f97c3153e, tamaño 1.024.572.864 bytes)

3	b9153318862f0f7b5f82c913ecb2117f97c3153e
30	

Figure 10: Ejemplo: QUERY_CHUNK

El peer B manda el dato del chunk 30 a el peer A del fichero ubuntu14.04.iso (hash b9153318862f0f7b5f82c913ecb2117f97c3153e, tamaño 1.024.572.864 bytes)

4	b9153318862f0f7b5f82c913ecb2117f97c3153e
30	DATOS

Figure 11: Ejemplo: QUERY_CHUNK_RESPONSE

Peer-tracker

Formato del mensaje: CONTROL para el tipo ADD_SEED_ACK

- Type = 1 (ADD_SEED_ACK)
  - Formato del mensaje: CONTROL.
  - El Tracker confirma con ACK que ha recibido la solicitud ADD_SEED de un Peer para unirse a la red P2P.

```
<message>
    <operation>add_seed_ack</operation>
</message>
```

Información del paquete:

- Type: Siempre sera 1 o add_seed_ack para indicar que es un ADD_SEED_ACK.

Formato del mensaje: FILEINFO para el tipo ADD_SEED

- Type = 2 (ADD_SEED)
  - Formato del mensaje: FILEINFO.
  - Un Peer solicita al Tracker unirse a la red compartiendo una serie de ficheros. Si la lista de ficheros no cabe en un único paquete se partirá en los paquetes que sean necesarios, por cada paquete que se envíe se esperara un ACK y cuando llegue ese ACK se mandara el siguiente paquete. Cambien es usado mas tarde por el Peer para añadir ficheros conforme quiera compartirlos.

```
<message>
    <operation>add_seed</operation>
    <port></port>
    <paquetes></paquetes>
    <file>
        <name></name>
        <size></size>
        <hash></hash>
    </file>
</message>
```

Información del paquete:

- Type: Siempre sera 2 o add_seed para indicar que es un ADD_SEED.
- Port: Indica el puerto por el que escuchara el peer.
- paquetes: Numero de ficheros que mandamos al tracker.
- Filename: nombre del fichero, se repetirá n veces.
- Size: Tamaño del fichero en bytes, se repetirá n veces.
- Hash: Hash del fichero, se repetirá n veces.

Formato del mensaje: SEEDINFO para el tipo GET_SEEDS

- Type = 3 (GET_SEEDS)
  - Formato del mensaje: SEEDINFO.
  - Un Peer solicita al Tracker la lista de semillas para un fichero, para indicar que fichero es usara el Hash del fichero.

```
<message>
    <operation>get_seeds</operation>
    <hash></hash>
    </message>
```

Información del paquete:

- Type: Siempre sera 3 o get_seeds para indicar que es un GET_SEEDS.
- Hash: Hash del fichero del que deseamos obtener los Seeds.

Formato del mensaje: SEEDINFO para el tipo SEED_LIST

- Type = 4 (SEED_LIST)
  - Formato del mensaje: SEEDINFO.
  - Un Tracker informa al Peer de la lista de semillas del fichero solicitado a través de su Hash.

```
<message>
    <operation>seed_list</operation>
    <hash></hash>
    <size></size>
    <seeds>
        <ip></ip>
        <port></port>
    </seeds>
</message>
```

Información del paquete:

- Type: Siempre sera 4 o seed_list para indicar que es un SEED_LIST.
- Hash: Hash del fichero del que obtenemos los Seeds.
- Size: Tamaño del fichero en bytes.
- IP: IP del peer que tiene trozos del fichero, se repetirá n veces.
- Port: Puerto que tiene a la escucha el peer que tiene trozos del fichero, se repetirá n veces.

1.1. Usando mensajes multiformato un peer (155.54.2.3) se agrega como seed en el puerto 4533, y con los ficheros:

- ubuntu14.04.iso (hash b9153318862f0f7b5f82c913ecb2117f97c3153e, tamaño 1.024.572.864 bytes)
- android-studio.zip (hash af09cc0a33340d8daccdf3cbfefdc9ab45b97b5d , tamaño 380.943.097 bytes).

El peer envía al tracker un mensaje FILEINFO tipo ADD_SEED

2	4533	2
15	ubuntu14.04.iso	
1024572864	b9153318862f0f7b5f82c913ecb2117f97c3153e	
18	android-studio.zip	
380943097	af09cc0a33340d8daccdf3cbfefdc9ab45b97b5d	

Figure 12: Ejemplo: ADD_SEED

El tracker responde con un mensaje de CONTROL tipo ADD_SEED_ACK

1

Figure 13: Ejemplo: ADD_SEED_ACK

1.2. Usando un lenguaje de marcas especificar la comunicación del apartado 1.1

El peer envía al tracker un mensaje FILEINFO tipo ADD_SEED

```
<message>
    <operation>add_seed</operation>
    <port>4533</port>
    <paquetes>1</paquetes>
    <file>
        <name>ubuntu14.04.iso</name>
        <size>1024572864</size>
        <hash>b9153318862f0f7b5f82c913ecb2117f97c3153e</hash>
    </file>
    <file>
        <name>android-studio.zip</name>
        <size>380943097</size>
        <hash>af09cc0a33340d8daccdf3cbfefdc9ab45b97b5d</hash>
    </file>
</message>
```

El tracker responde con un mensaje de CONTROL tipo ADD_SEED_ACK

```
<message>
    <operation>add_seed_ack</operation>
</message>
```

Formato del mensaje: SEEDQUERY para el tipo QUERY_FILES

- Type = 5 (QUERY_FILES)
  - Formato del mensaje: SEEDQUERY
  - Un Peer solicita al Tracker la lista de ficheros que coinciden con su patrón de búsqueda. Podemos buscar por tamaño, nombre o ambas, para tamaño tenemos varios operadores aritméticos.

```
<message>
    <operation>query_files</operation>
    <op></op>
    <file>
        <size></size>
        <name></name>
    </file>
</message>
```

Información del paquete:

- Type: Siempre sera 5 o query_files para indicar que es un QUERY_FILES.
- Op: Operador de condición para tamaño: *(podemos usar operadores bash: gt, lt, ge, etc)*
  - 0: >
  - 1: >=
  - 2: <
  - 3: <=
  - 4: Aprox (10Mb) (EXTRA)
- Size: Tamaño del fichero que estamos buscando en bytes, si no lo usamos todo a 0.
- Filename: nombre del fichero que buscamos, si no lo usamos todo a 0.

Formato del mensaje: FILEINFO para el tipo QUERY_FILES_RESPONSE

- Type = 6 (QUERY_FILES_RESPONSE)
  - Formato del mensaje: FILEINFO
  - El tracker lista los ficheros que coinciden con su patrón de búsqueda. Si la lista de ficheros no cabe en un único paquete se partirá en los paquetes que sean necesarios, y se enviaran todos, el Peer tendrá que comprobar si se han recibido todos los paquetes, sino se han recibido volverá a hacer la solicitud.

```
<message>
    <operation>query_files_response</operation>
    <paquetes></paquetes>
    <file>
        <name></name>
        <size></size>
        <hash></hash>
    </file>
</message>
```

Información del paquete:

- Type: Siempre sera 6 o query_files_response para indicar que es un QUERY_FILES_RESPONSE.
- Paquetes: Numero de paquetes que se van a enviar, normalmente sera 1.
- Filename: nombre del fichero, se repetirá n veces.
- Size: Tamaño del fichero en bytes, se repetirá n veces.
- Hash: Hash del fichero, se repetirá n veces.

Formato del mensaje: CHUNKINFO para el tipo QUERY_CHUNK

- Type = 7 (QUERY_CHUNK)
  - Formato del mensaje: CHUNKINFO
  - El Peer pregunta al Tracker cual es el tamaño de los chunks. Este mensaje sera el primero que se mande para iniciar la comunicación con el Tracker.

```
<message>
    <operation>query_chunk</operation>
</message>
```

Información del paquete:

- Type: Siempre sera 7 para indicar que es un QUERY_CHUNK.

Formato del mensaje: CHUNKINFO para el tipo QUERY_CHUNK_RESPONSE

- Type = 8 (QUERY_CHUNK_RESPONSE)
  - Formato del mensaje: CHUNKINFO
  - El Tracker responde al Peer cual es el tamaño de los chunks.

```
<message>
    <operation>query_chunk_response</operation>
    <size></size>
</message>
```

Información del paquete:

- Type: Siempre sera 8 para indicar que es un QUERY_CHUNK_RESPONSE.
- Tamaño Chunk: Indica el tamaño del chunk.

Formato del mensaje: REMOVE para el tipo REMOVE_SEED

- Type = 9 (REMOVE_SEED)
  - Formato del mensaje: REMOVE
  - El Peer informa al Tracker que quiero hacer una solicitud de borrado, hay 2 opciones:
    - Darse de baja como peer: si queremos dejar de ser un Peer activo pondremos el Hash todo a 1.
    - Dar de baja un fichero: Si solo queremos borrar un fichero de la lista de activos pondremos el Hash del fichero.

```
<message>
    <operation>remove_seed</operation>
    <port></port>
    <hash></hash>
</message>
```

Información del paquete:

- Type: Siempre sera 9 o remove_seed para indicar que es un REMOVE_SEED.
- Port: Indica el puerto del peer.
- Hash: Hash del fichero al que damos de baja o todo a 1

Formato del mensaje: CONTROL para el tipo REMOVE_SEED_ACK

- Type = 10 (REMOVE_SEED_ACK)
  - Formato del mensaje: CONTROL
  - El Tracker confirma que ha recibido la solicitud de un Peer para borrar al Peer o a un fichero.

```
<message>
    <operation>remove_seed_ack</operation>
</message>
```

Información del paquete:

- Type: Siempre sera 10 o remove_seed_ack para indicar que es un REMOVE_SEED_ACK.

2.1. Usando mensajes multiformato el peer solicita un QUERY_FILES al tracker y éste responde con la lista de archivos correspondiente.

- ubuntu14.04.iso (hash b9153318862f0f7b5f82c913ecb2117f97c3153e, tamaño 1.024.572.864 bytes)
- android-studio.zip (hash af09cc0a33340d8daccdf3cbfefdc9ab45b97b5d , tamaño 380.943.097 bytes).

El peer envía al track un mensaje SEED_QUERY_FILES tipo QUERY_FILES para buscar los ficheros de tamaño superior o igual a 250000 bytes

5	1	250000
0000000000000000	000000000....	

Figure 14: Ejemplo: QUERY_FILES

El tracker le responde con un mensaje FILEINFO tipo QUERY_FILES_RESPONSE con los ficheros de tamaño superior a 250000 bytes

6	0000000000000000	0000000000000000	2
15	ubuntu14.04.iso		
1024572864	b9153318862f0f7b5f82c913ecb2117f97c3153e		
18	android-studio.zip		
380943097	af09cc0a33340d8daccdf3cbfefdc9ab45b97b5d		

Figure 15: Ejemplo: QUERY_FILES_RESPONSE

2.2. Usando lenguaje de marcas especificar la comunicación del apartado 2.1.

Peer informa a tracker con un mensaje SEEDQUERY:

```
<message>
    <operation>query_files</operation>
    <op>1</op> <!--podemos usar operadores bash: gt, lt, ge, etc-->
    <file>
        <size>250000</size>
        <name></name>
    </file>
</message>
```

El tracker responde con un mensaje FILEINFO:

```
<message>
    <operation>query_files_response</operation>
    <num_seq></num_seq>
    <port></port>
    <file>
        <name>ubuntu14.04.iso</name>
        <size>1024572864</size>
        <hash>b9153318862f0f7b5f82c913ecb2117f97c3153e</hash>
    </file>
    <file>
        <name>android-studio.zip</name>
        <size>380943097</size>
        <hash>af09cc0a33340d8daccdf3cbfefdc9ab45b97b5d</hash>
    </file>
</message>
```

El Peer pregunta al Tracker el tamaño de los chunks

```
<message>
    <operation>query_chunk</operation>
</message>
```

El Tracker responda al peer con el tamaño

```
<message>
    <operation>query_chunk_response</operation>
    <size>1500</size>
</message>
```

El peer elimina un fichero de la lista del tracker

```
<message>
    <operation>remove_seed</operation>
    <port>9541</port>
    <hash>af09cc0a33340d8daccdf3cbfefdc9ab45b97b5d</hash>
</message>
```

El tracker confirma la eliminación

```
<message>
    <operation>remove_seed_ack</operation>
</message>
```

El peer se da de baja como peer activo de la lista del tracker

```
<message>
    <operation>remove_seed</operation>
    <port>9541</port>
    <hash>11111111111111111111111111111111111111111</hash>
</message>
```

El tracker confirma la eliminación

```
<message>
    <operation>remove_seed_ack</operation>
</message>
```

Errores que pueden surgir durante las comunicaciones:

1. Un peer se desconecte sin avisar al tracker, en ese caso el peer que detecte al peer desconectado debera de informar al tracker para que le de de baja.
2. Un peer deje de tener un fichero que estaba compartiendo, en este caso tendra que mandar al tracker un remove del fichero que ya no tiene.

Conclusiones