Sitos:
Proyecto BitTorrent

75.42 / 95.08 Taller de Programación I
1C 2022 - Ing. Pablo Deymonnaz

¿Quienes Somos?

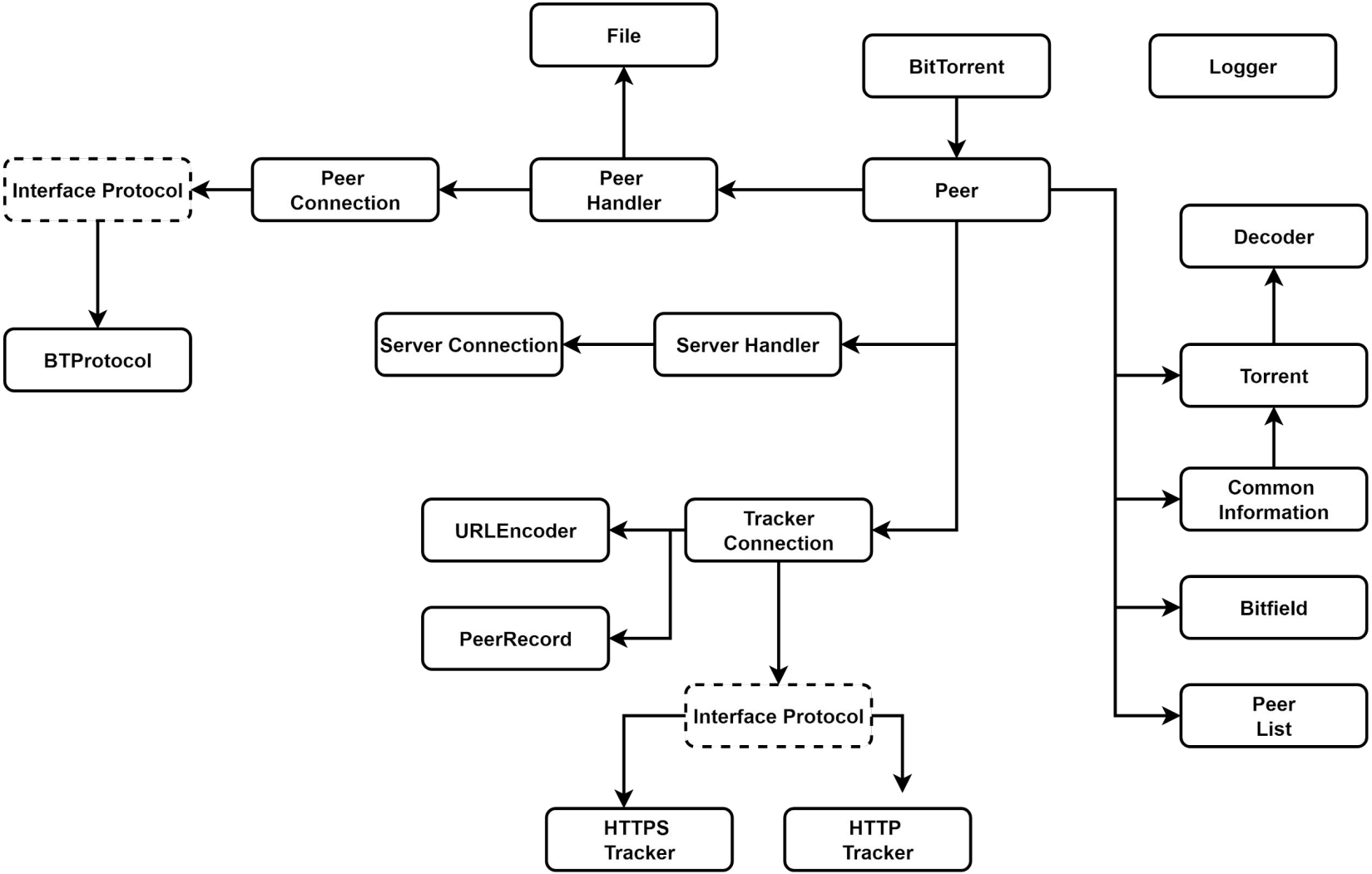
Lautaro Goijman

Sofia Carbon Posse

Martin Veiga

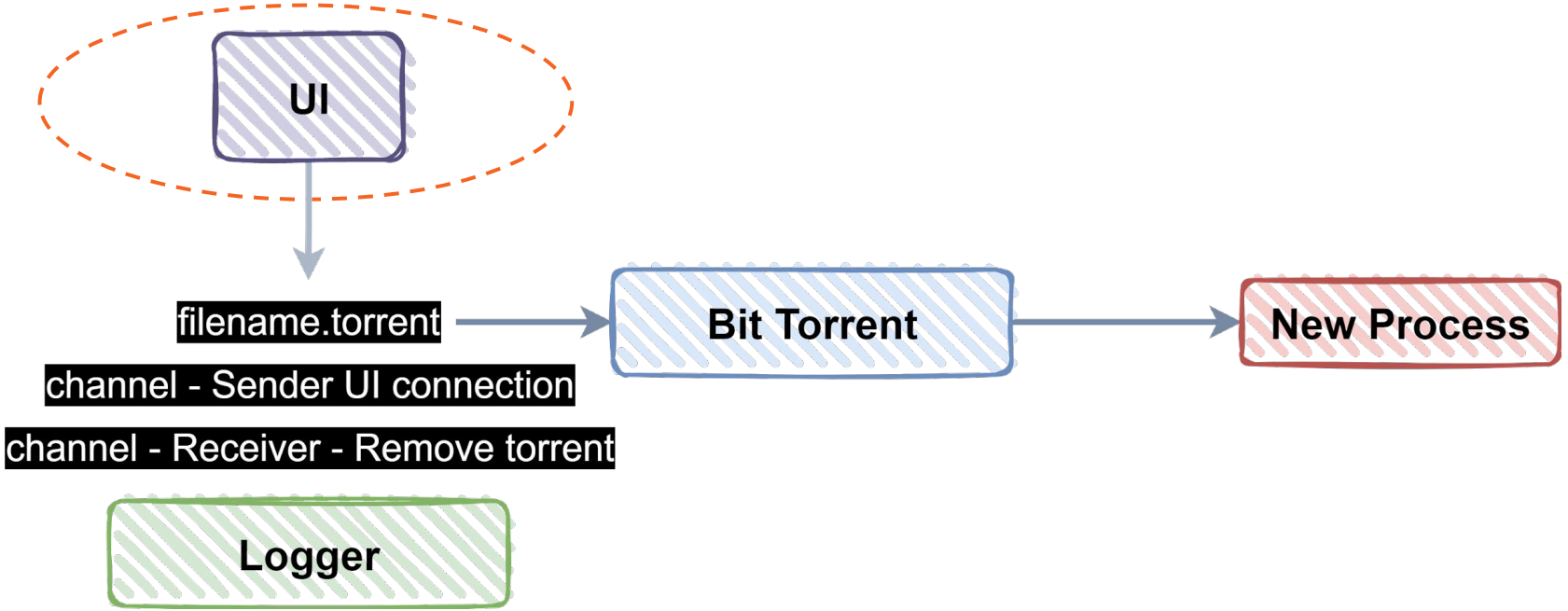
Julian Garcia

Arquitectura del Programa

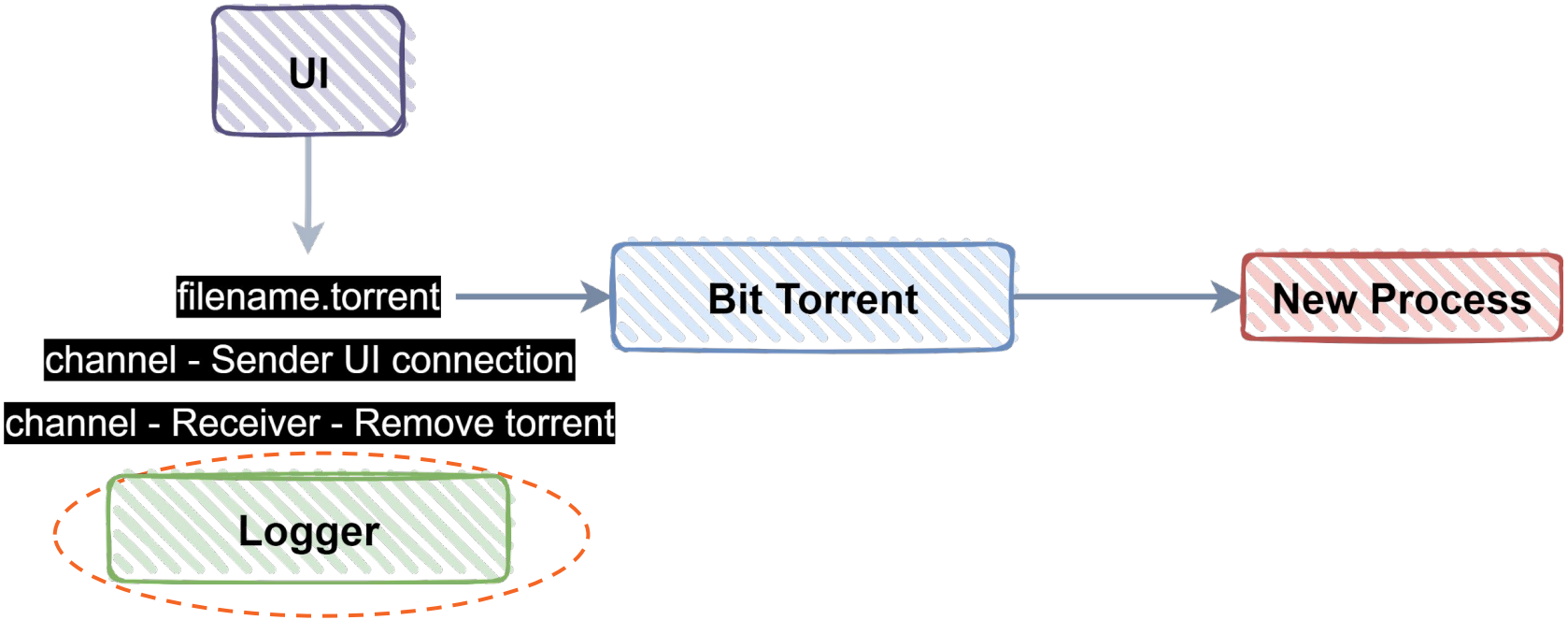

File
BitTorrent
Logger
Interface Protocol
Peer Connection
Peer Handler
Peer
Decoder
BTProtocol
Server Connection
Server Handler
Torrent
URLEncoder
Tracker Connection
Common Information
PeerRecord
Bitfield
Interface Protocol
Peer List
HTTPS Tracker
HTTP Tracker

¿Sobre qué estructuras pasa el flujo de programa cuando queremos descargar un archivo?

Flujo


UI
filename.torrent
Bit Torrent
New Process
channel - Sender UI connection
channel - Receiver - Remove torrent
Logger

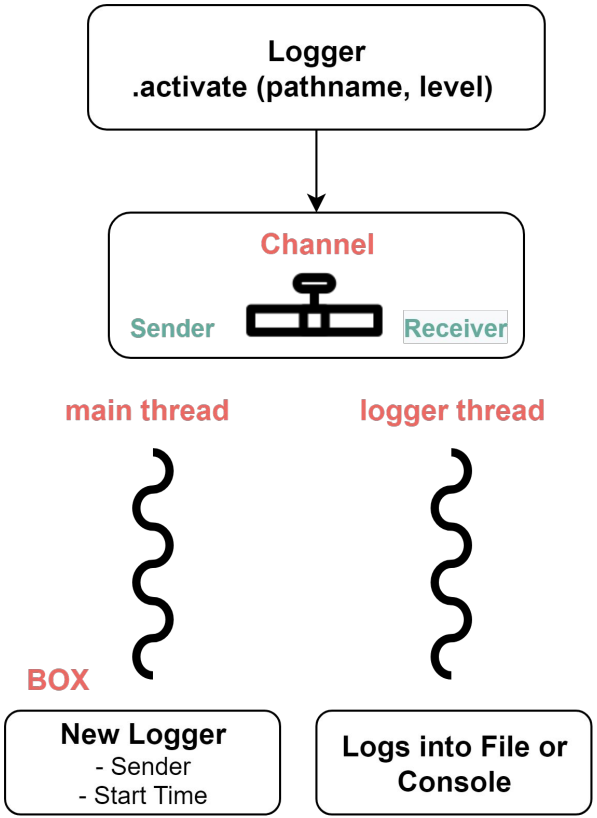
Flujo


UI
filename.torrent
Bit Torrent
New Process
channel - Sender UI connection
channel - Receiver - Remove torrent
Logger

Logger

- Es un **SINGLETON:** una única instancia puede ser creada.
- Loggea por consola o en un archivo específico.
- ***.activate*** (pathname, level)

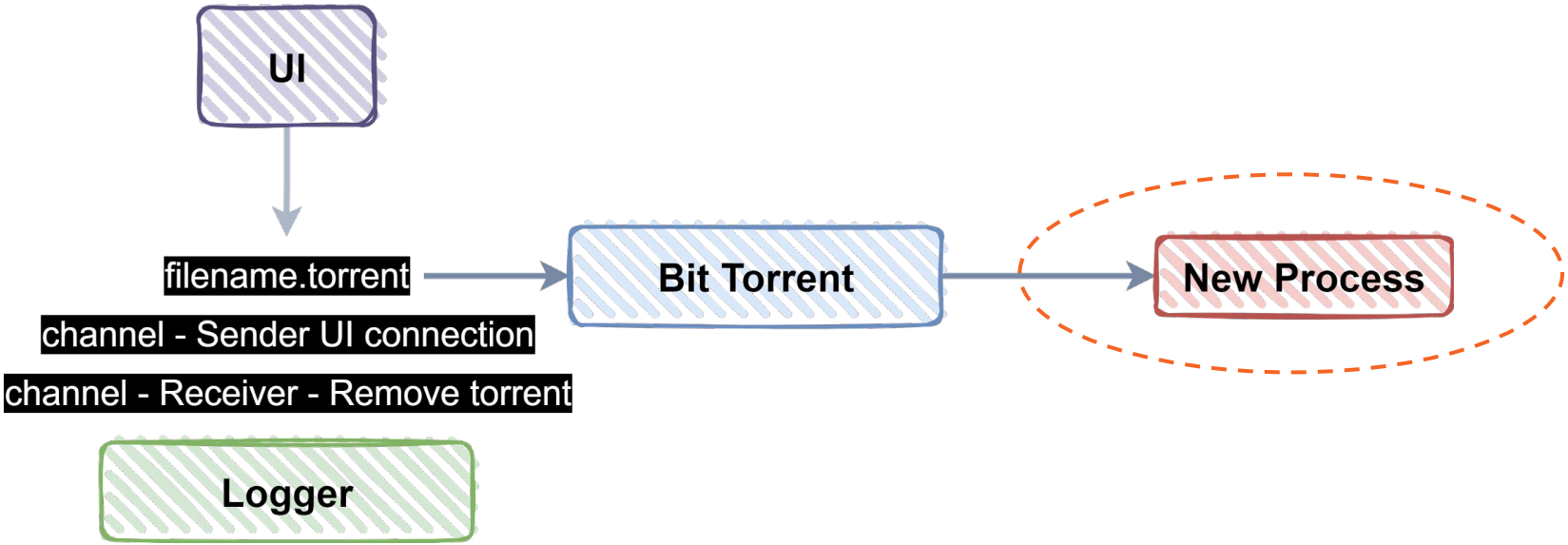
Logger



Formato con el que logea:

```
[00:00:00.000] ERROR  [sitos::logger::index::tests:167] Ups! This is an Error
[00:00:01.000] WARN   [sitos::logger::index::tests:169] This is a Warn
[00:00:02.004] INFO   [sitos::logger::index::tests:171] Info available here
[00:00:03.005] DEBUG  [sitos::logger::index::tests:173] Debug mode
[00:00:04.005] TRACE  [sitos::logger::index::tests:175] Oh no! Trace!
```

Flujo


UI
filename.torrent
channel - Sender UI connection
channel - Receiver - Remove torrent
Logger
Bit Torrent
New Process

– BitTorrent

- Contiene un vector con **JOIN HANDLE** que permite tener control de los threads.
- Pushea al vector un Peer activado.

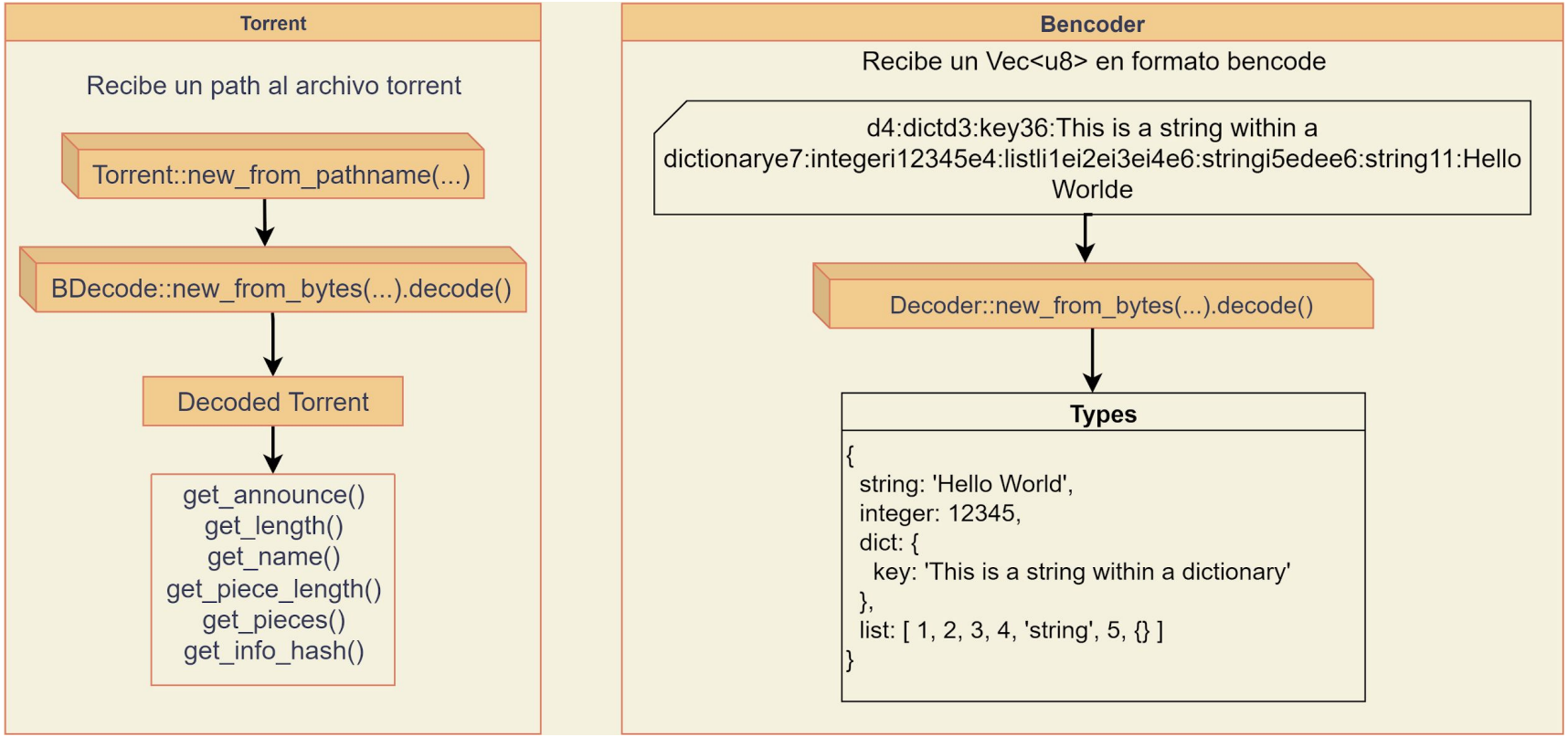
BitTorrent
processes: Vec<JoinHandle<()>>
new_process(torrent_pathname, sender, remove_rx)

Veamos en detalle el Peer...

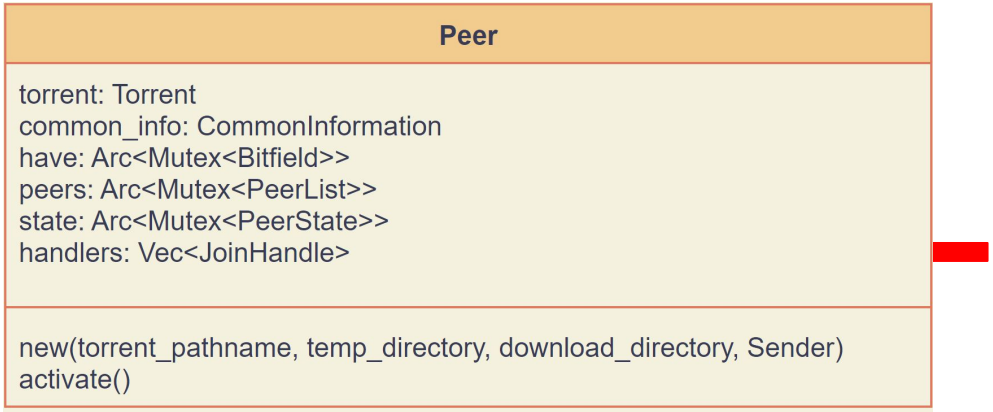
Peer

Peer
torrent: Torrent common_info: CommonInformation have: Arc<Mutex<Bitfield>> peers: Arc<Mutex<PeerList>> state: Arc<Mutex<PeerState>> handlers: Vec<JoinHandle>
new(torrent_pathname, temp_directory, download_directory, Sender) activate()

Torrent

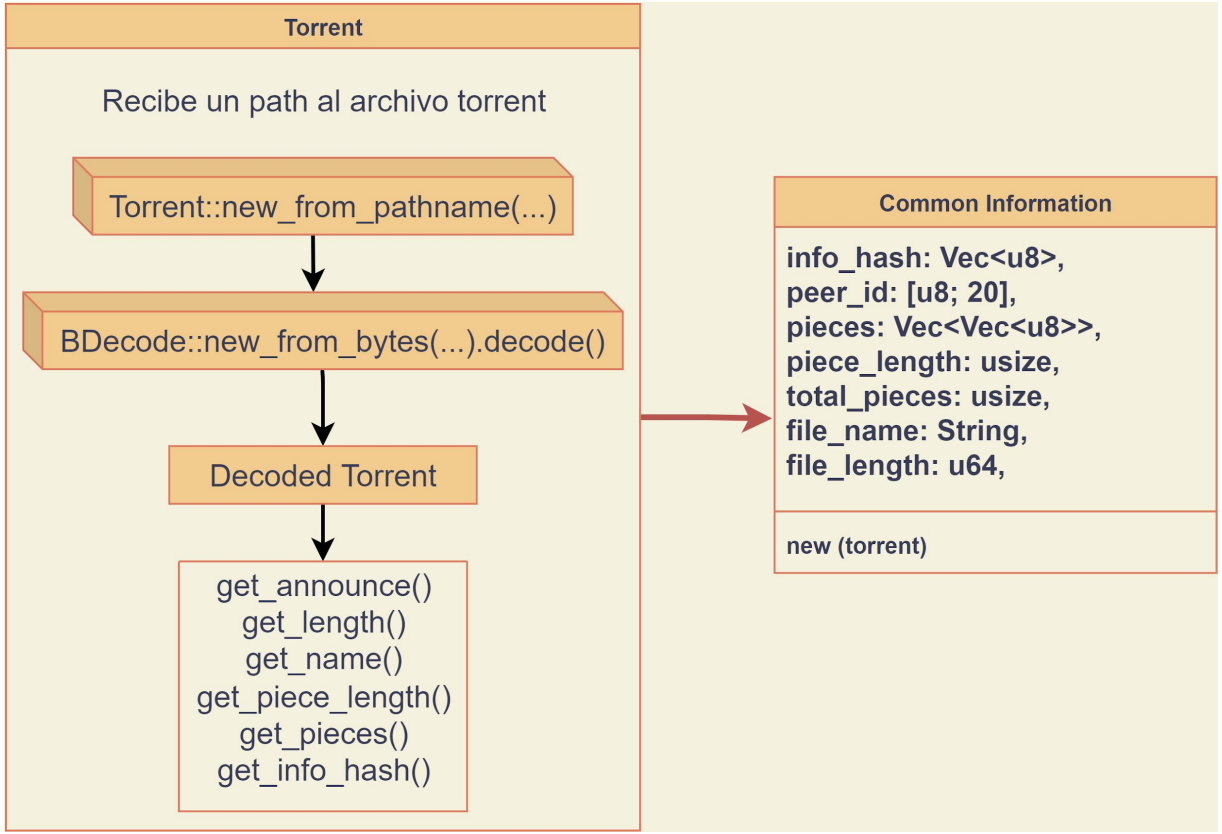

Torrent
Recibe un path al archivo torrent
Torrent::new_from_pathname(...)
BDecode::new_from_bytes(...).decode()
Decoded Torrent
get_announce()
get_length()
get_name()
get_piece_length()
get_pieces()
get_info_hash()
Bencoder
Recibe un Vec<u8> en formato bencode
d4:dictd3:key36:This is a string within a dictionarye7:integeri12345e4:listli1ei2ei3ei4e6:stringi5edee6:string11:Hello Worlde
Decoder::new_from_bytes(...).decode()
Types
{
string: 'Hello World',
integer: 12345,
dict: {
key: 'This is a string within a dictionary'
},
list: [1, 2, 3, 4, 'string', 5, {}]
}

Peer


Peer
torrent: Torrent
common_info: CommonInformation
have: Arc<Mutex<Bitfield>>
peers: Arc<Mutex<PeerList>>
state: Arc<Mutex<PeerState>>
handlers: Vec<JoinHandle>
new(torrent_pathname, temp_directory, download_directory, Sender)
activate()

Common Information

Contenedor de información que se comparte entre todas las partes del proceso.



Peer

Peer
torrent: Torrent common_info: CommonInformation have: Arc<Mutex<Bitfield>> peers: Arc<Mutex<PeerList>> state: Arc<Mutex<PeerState>> handlers: Vec<JoinHandle>
new(torrent_pathname, temp_directory, download_directory, Sender) activate()

– Bitfield

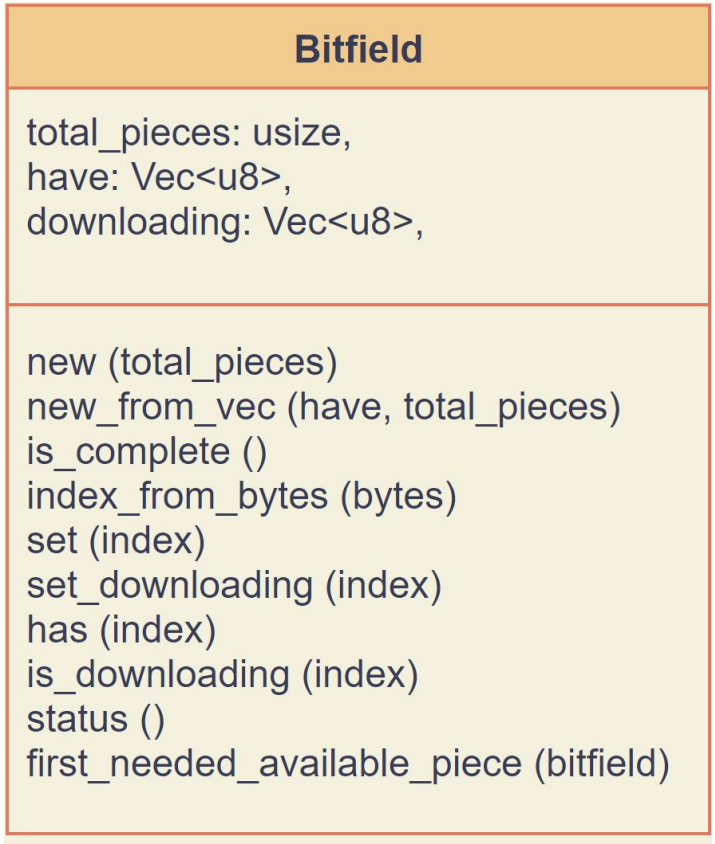
have:

- *1 -> pieces obtenidas*
- *0 -> pieces faltantes*

downloading:

- *1 -> pieces downloading*
- *0 -> pieces not downloading*

*Esta estructura está dentro de un **ARC - MUTEX** debido a que esta información debe ser compartida entre los distintos threads.*



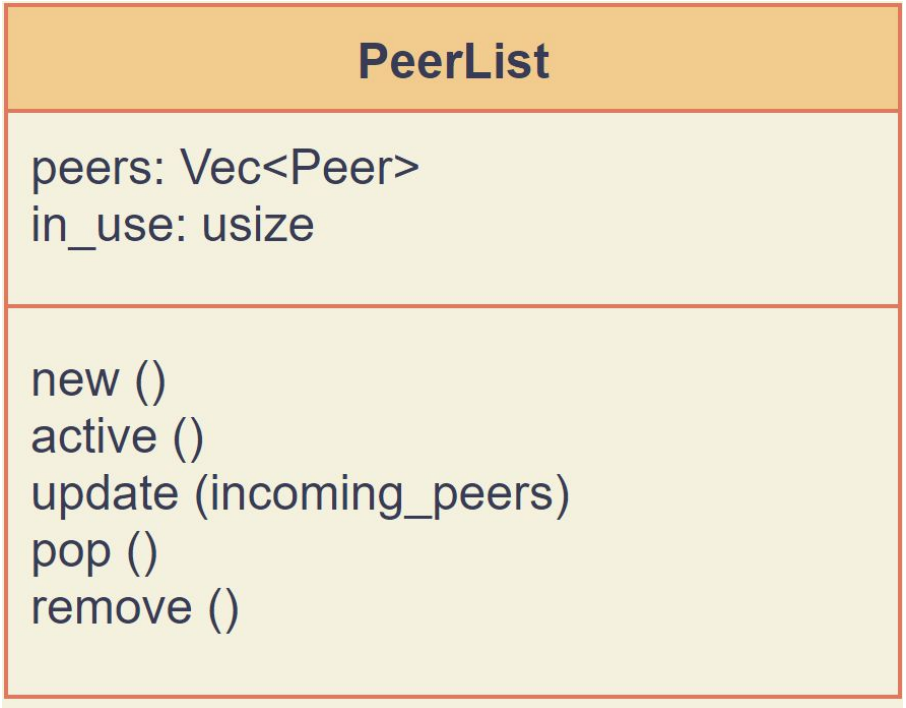
Peer

Peer
torrent: Torrent common_info: CommonInformation have: Arc<Mutex<Bitfield>> peers: Arc<Mutex<PeerList>> state: Arc<Mutex<PeerState>> handlers: Vec<JoinHandle>
new(torrent_pathname, temp_directory, download_directory, Sender) activate()

– Peer List

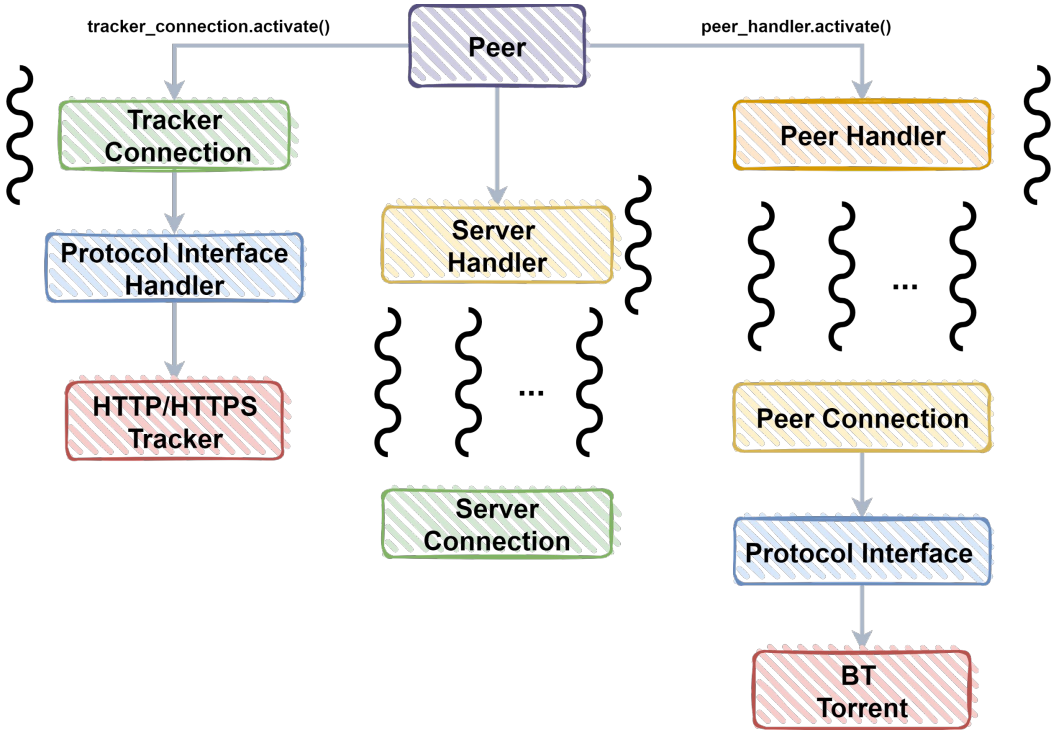
Almacena en un Vector los peers que nos devuelve el Tracker.

*Esta estructura está dentro de un **ARC - MUTEX** debido a que esta información debe ser compartida entre los distintos threads.*



¿Qué sucede cuando activamos el Peer?

Flujo


tracker_connection.activate()
Peer
peer_handler.activate()
Tracker Connection
Protocol Interface Handler
HTTP/HTTPS Tracker
Server Handler
...
Server Connection
Peer Handler
...
Peer Connection
Protocol Interface
BT Torrent

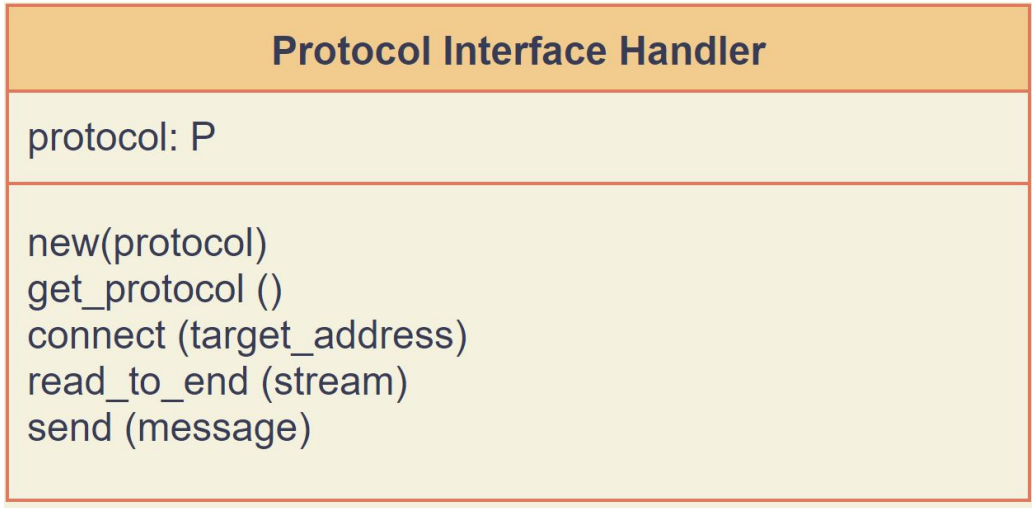
Tracker Connection

Realiza la conexión con el tracker

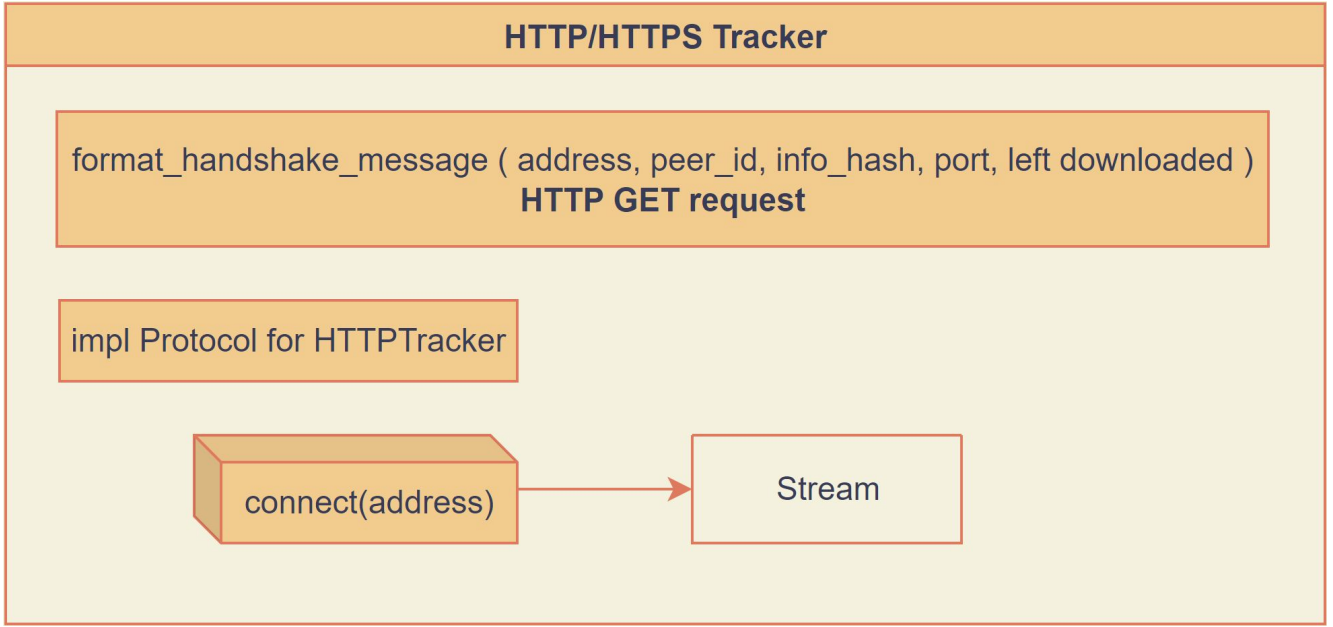
TrackerConnection
protocol_interface: ProtocolInterfaceHandler bitfield: <Arc<Mutex<Bitfield>> peers: <Arc<Mutex<PeerList>> common_info: CommonInformation tracker_address: String sleep: u64 state: Arc<Mutex<PeerState>>
new (announce, bitfield, peers, common_information) activate (listening_port)

– Protocol Interface Handler

Es un enumerate que maneja el request al Tracker dependiendo del protocolo.

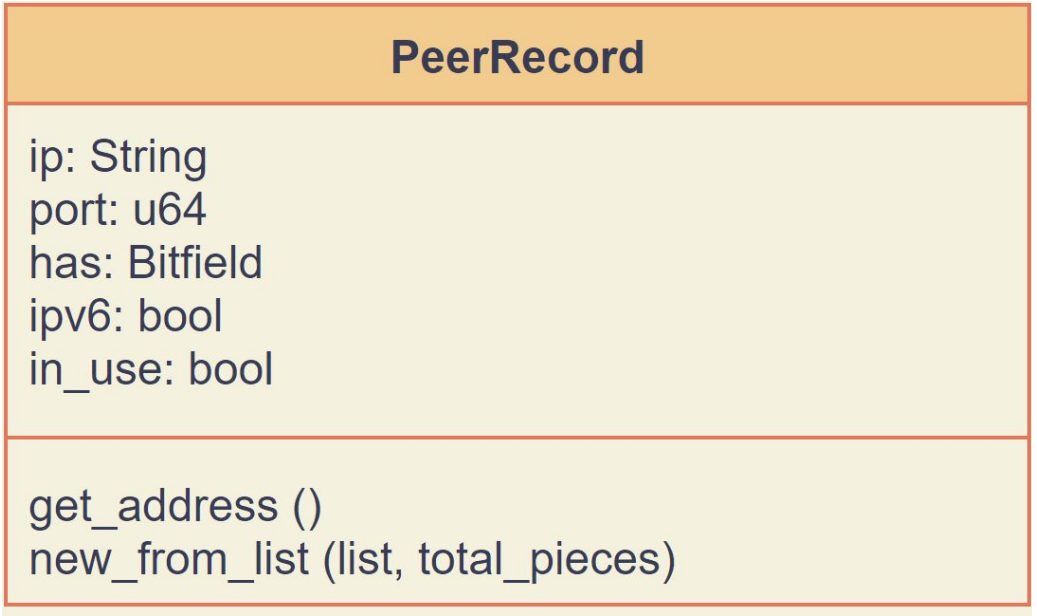


Protocol
HTTP/HTTPS Tracker



Peer Record

Por cada peer obtenido se crea una instancia de *Peer Record* que almacena la información importante de cada peer y actualiza la PeerList.

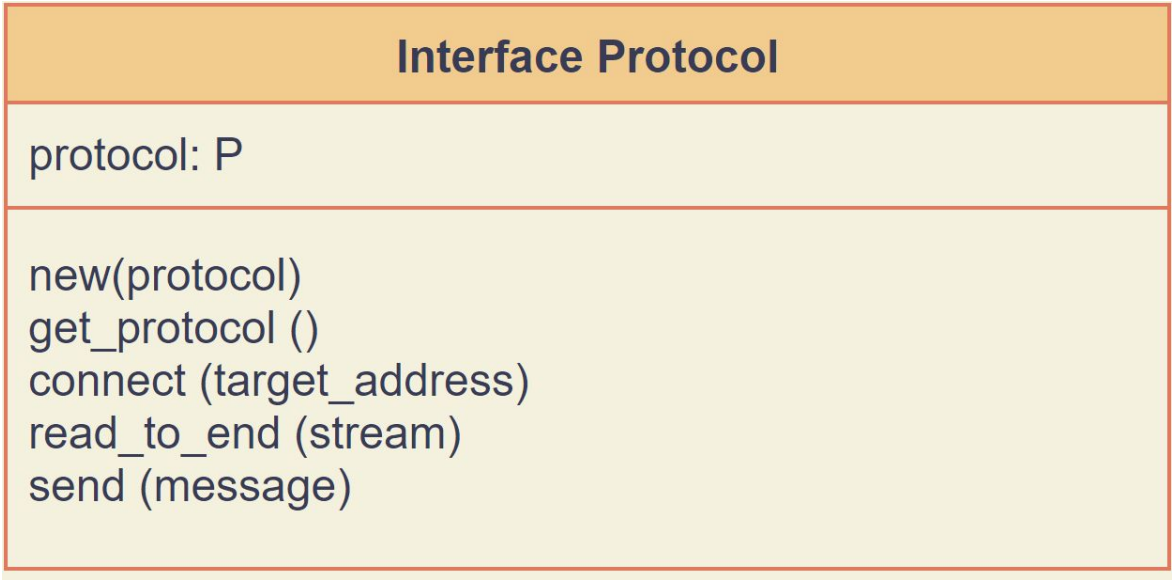


Peer Handler

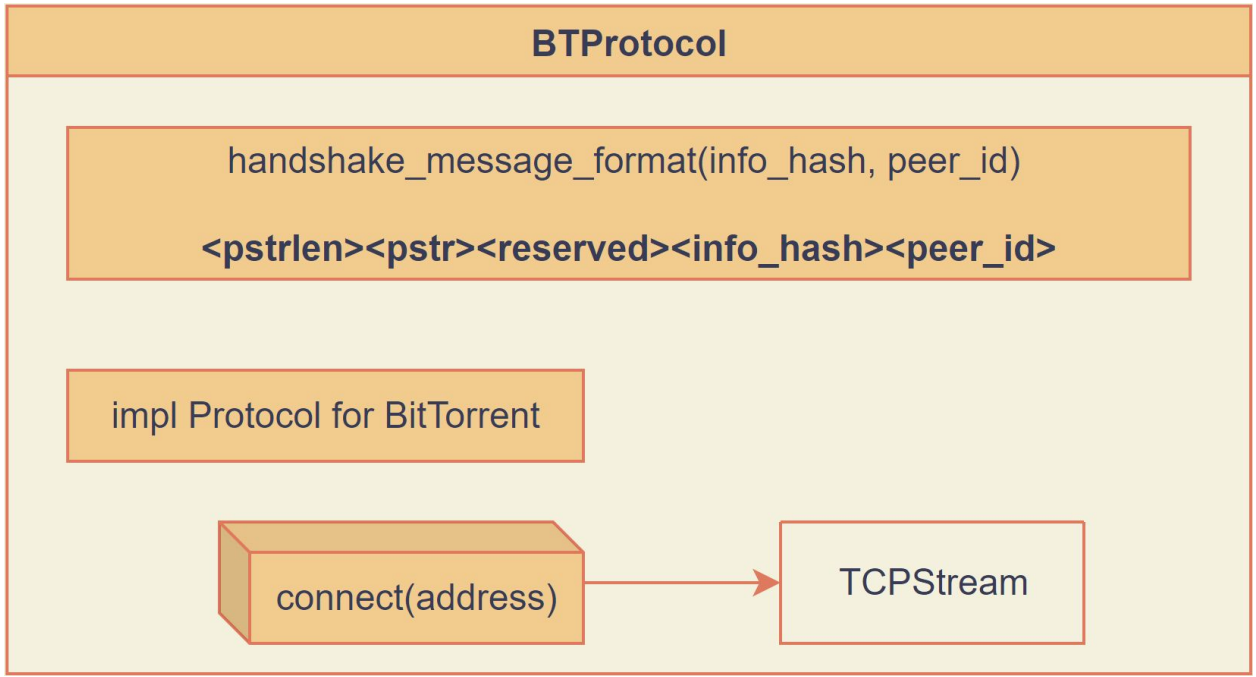
Peer Connection

PeerConnection
peer: PeerRecord protocol_interface: ProtocolInterfaceProtocol<BTProtocol>> bitfield: <Arc<Muex<Bitfiled>> peers: <Arc<Muex<PeerList>> common_information: CommonInformation stream: <BTProtocol as Protocol>::Stream state: State peer_state: <Arc<Muex<PeerState>>
new (bitfield, peers, common_info, peer) active (bitfield, peers, common_information, peer, connections) greet () process_handshake_response () unchoke () download_piece ()

– Protocol Interface



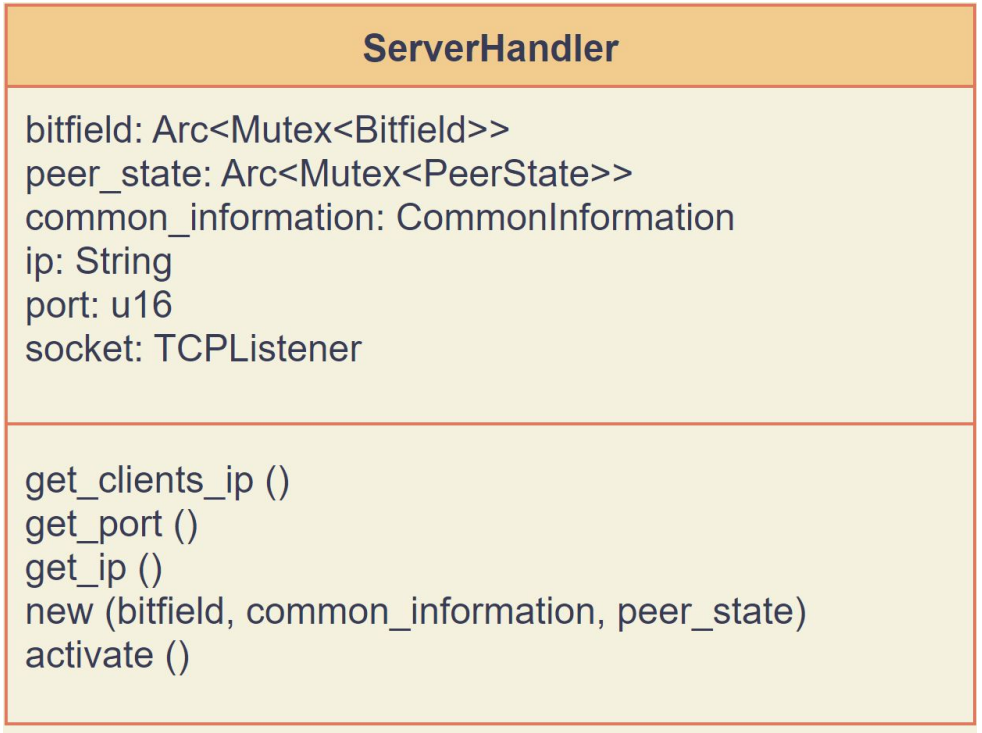
Protocol BitTorrent



File

File
pathname: String handler: std::fs::File (Handler)
open_file (pathname) new (pathname) with_contents (pathname) get_contentes (pathname, contents) get_block (piece_index, piece_size, block_size, block_offset) new_file_from_pieces (piece, pathname) concat (piece) join_pieces (path_from, path_to)

Server Handler



Server Connection

ServerConnection
bitfield: Arc<Mutex<Bitfield>> peer_state: Arc<Mutex<PeerState>> common_information: CommonInformation state: UploadState
new (bitfield, common_information, peer_state) activate (bitfield, common_information, peer_state, stream) serve_file (stream) send_handshake_response (stream) validate_peer (stream, own_peer_id)

¡Ahora sí!
Vamos a la DEMO...

¡GRACIAS!
¿PREGUNTAS?