

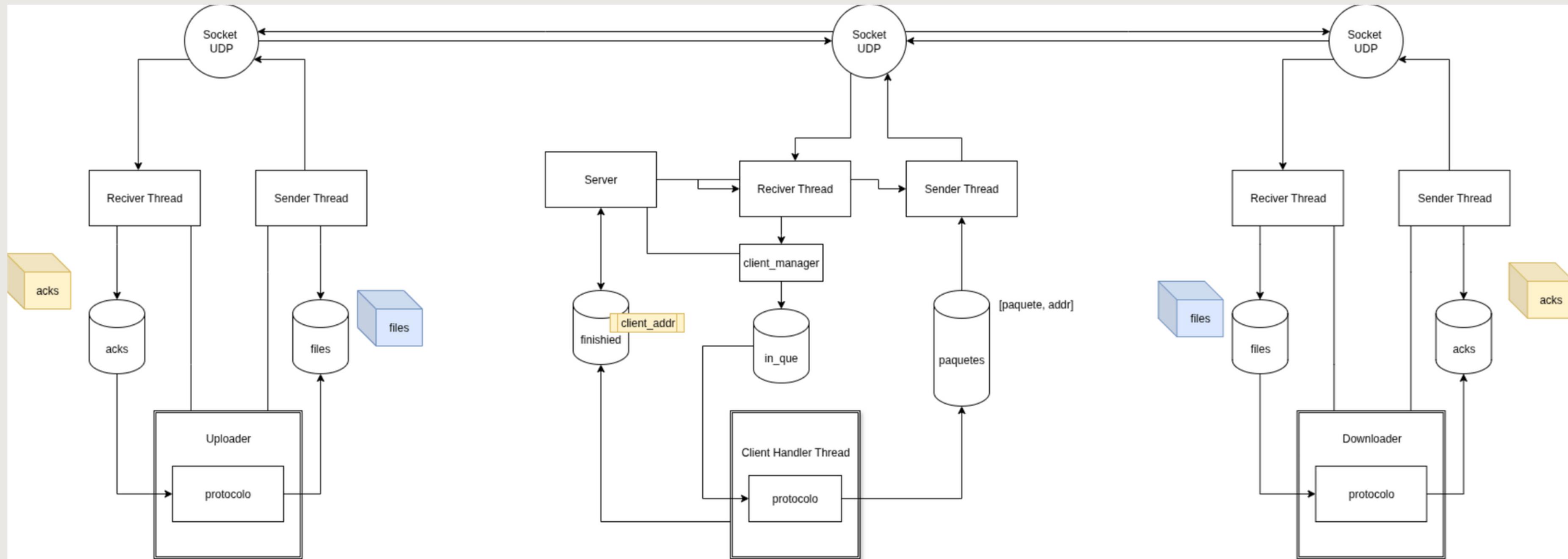
TRABAJO

PRACTICO 1

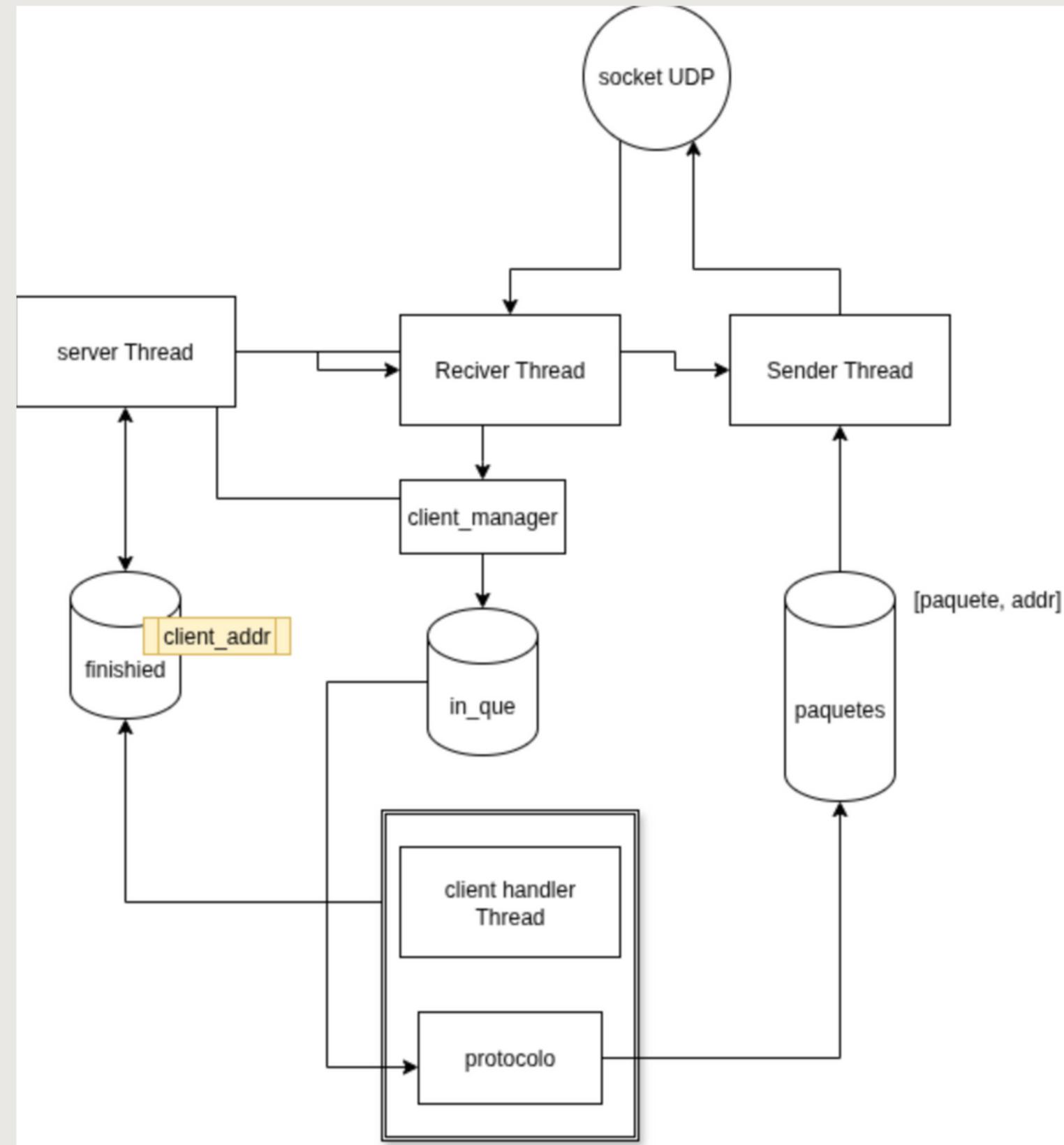
FILE TRANSFER

GRUPO 5

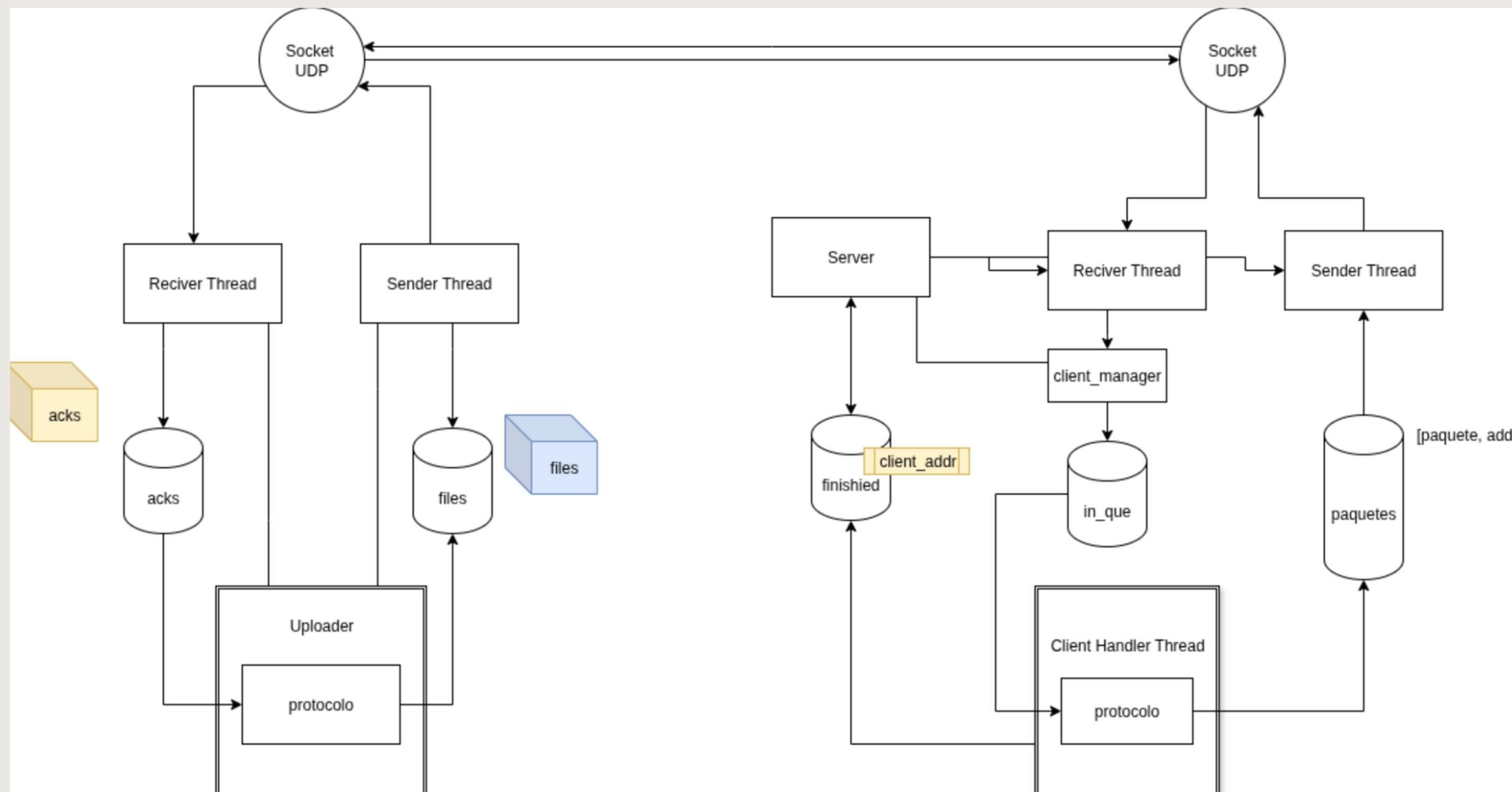
Arquitectura del Sistema



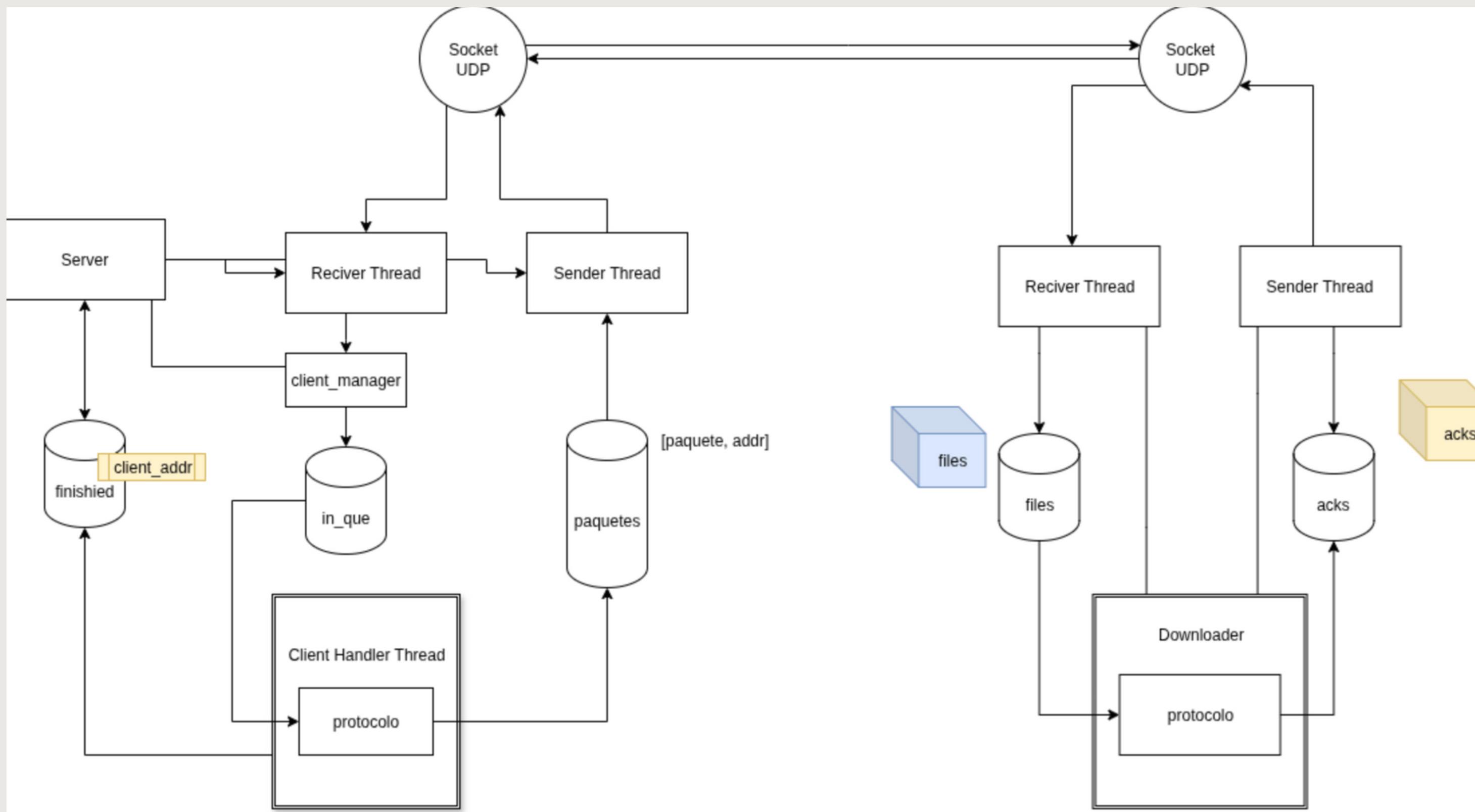
Arquitectura del servidor



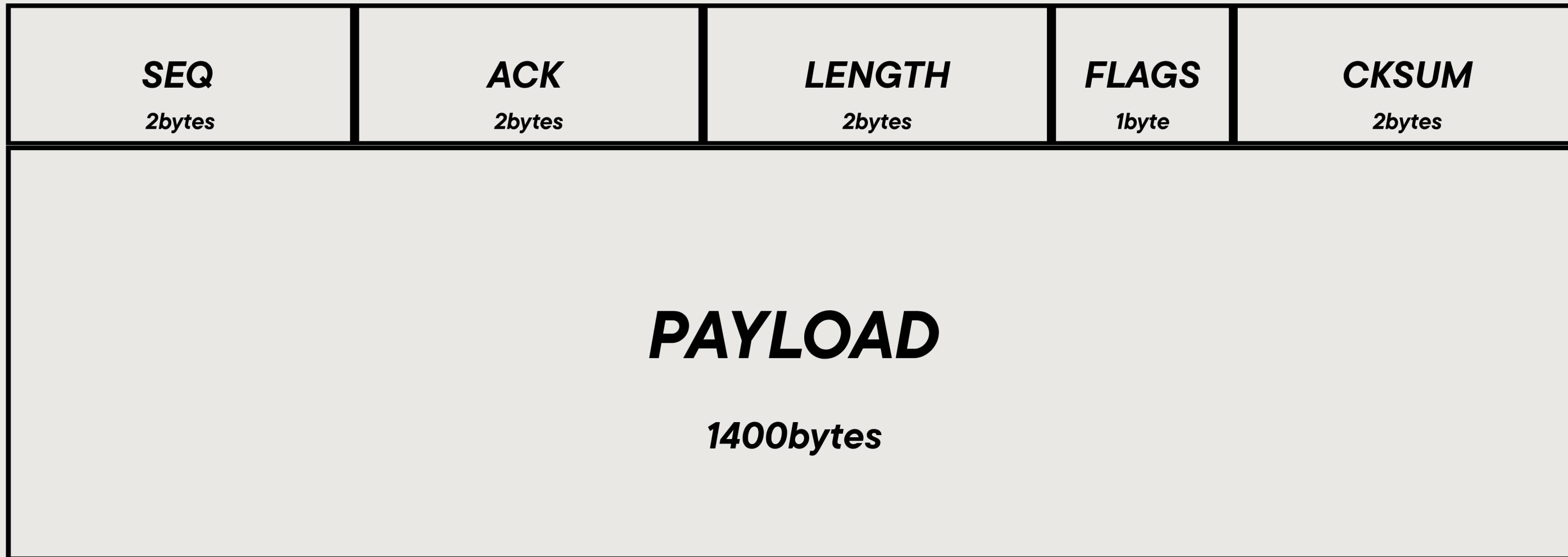
Arquitectura flujo de Upload



Arquitectura flujo de Download



Estructura del paquete



Protocolo Stop & Wait

Este consiste en **chequear**, por **cada paquete enviado**, que el mismo haya sido **recibido correctamente** por el receptor, mediante la recepción de un **acuse de recibo (ACK)**. En caso de **no recibir el ACK** en un tiempo determinado (timeout), se **retransmite el paquete**.

```
1  def send_file(
2      self, filepath, filename, destination_address, in_queue, sender_queue
3  ):
4      seq_num = 0
5      chunk_num = 0
6      chunks = list(read_file_chunks(filepath, filename))
7
8      for chunk_num, chunk in enumerate(chunks):
9          packet = packageLib.create_data_packet(seq_num, chunk)
10         retries = 0
11
12         while retries < MAX_RETRIES:
13             self._send(packet, destination_address, sender_queue)
14             ack_received = self._wait_ack(seq_num, in_queue)
15
16             if not ack_received:
17                 retries += 1
18             else:
19                 break
20
21             seq_num = 1 - seq_num
22
23         fin_packet = packageLib.create_fin_packet()
24         self._send(fin_packet, destination_address, sender_queue)
```

Protocolo Selective Repeat

- **Ventana deslizante:** Mantiene un conjunto de paquetes en tránsito, permitiendo que el emisor envíe múltiples paquetes sin esperar confirmación.
- **Retransmisión selectiva:** Cada paquete tiene su propio temporizador, y solo se retransmiten aquellos que han expirado sin recibir confirmación.
- **Buffers de recepción:** El receptor almacena paquetes recibidos fuera de orden para entregar los datos en secuencia correcta.
- **Timeout adaptativo:** El tiempo de espera para la retransmisión se ajusta dinámicamente según las condiciones de la red.

Protocolo Selective Repeat

```
1 def send_file(
2     self, filepath, filename, destination_address, in_queue, sender_queue
3 ):
4     try:
5         chunks = list(read_file_chunks(filepath, filename))
6         total_chunks = len(chunks)
7
8         while self.base < total_chunks:
9             self._send_window_packets(chunks, destination_address, sender_queue)
10
11            for _ in range(MAX_ACKS_TO_WAIT):
12                self._wait_for_acks(in_queue)
13                if self.base >= total_chunks:
14                    break
15
16                self._check_timeouts(chunks, destination_address, sender_queue)
17
18            fin_packet = packageLib.create_fin_packet()
19            self._send(fin_packet, destination_address, sender_queue)
```