HW2

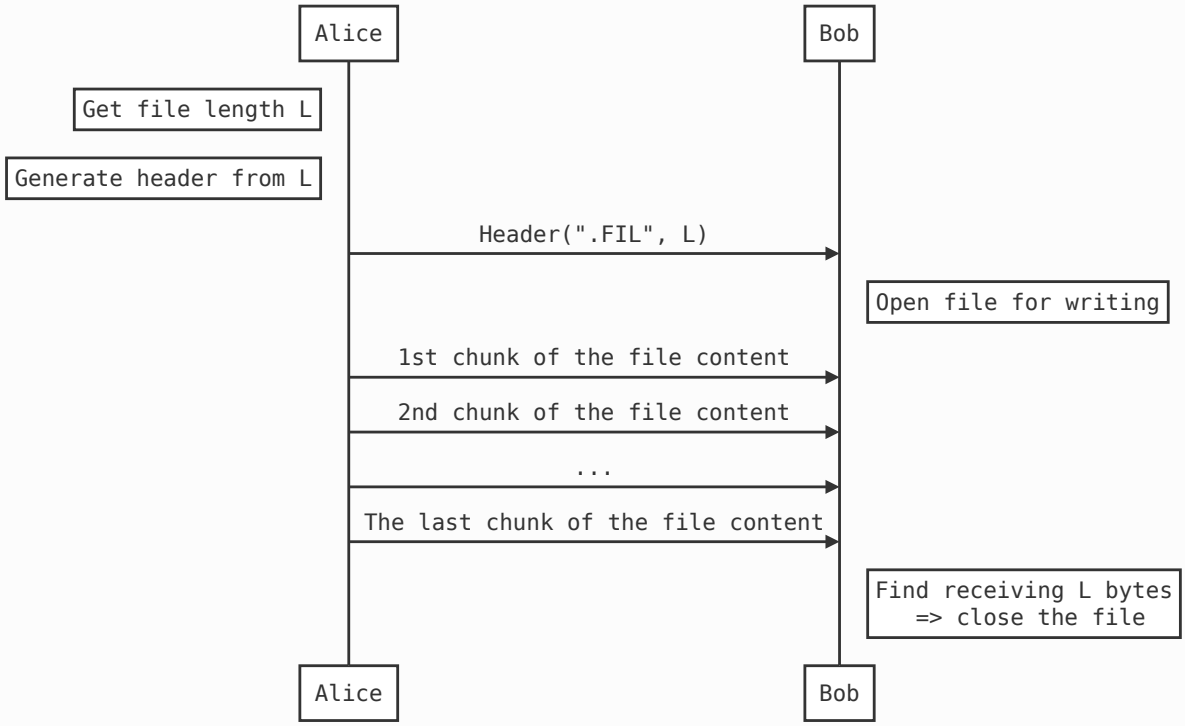
File transferring

For either message or file transferring, I design a package format like below,

```
[TYPE]    (4 bytes)            # Type of package (e.g. ".MSG", ".FIL")
[LENGTH]  (64 bytes)           # Total length of package
[CONTENT]                      # Content
```

[TYPE] and [LENGTH] above are so-called the **header** of the package.

Let Alice be the sender and Bob be the receiver,



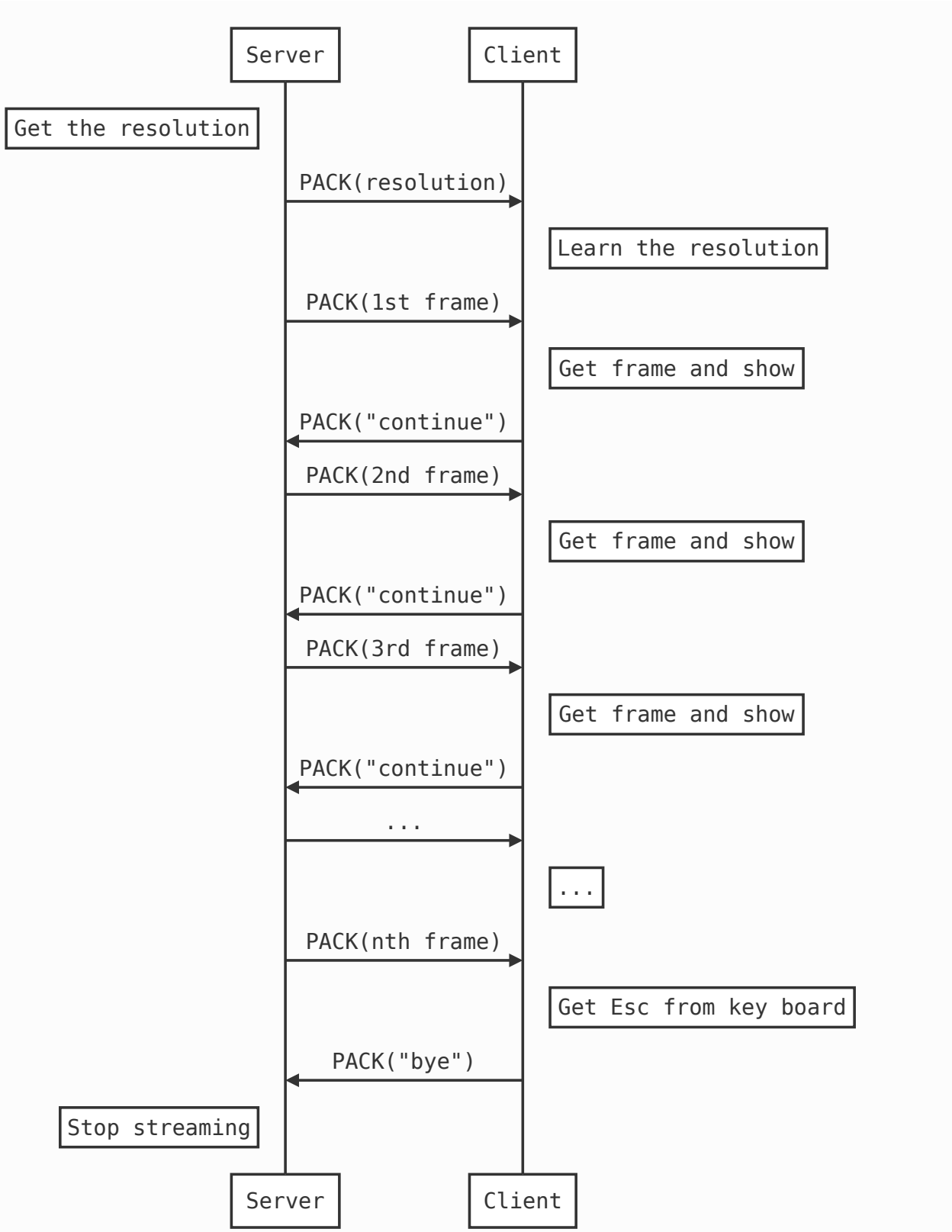
Video streaming

Let `PACK(m)` denote sending a message `m` with header mentioned above.

The server first send the resolution to client.

Then, the server send each frame, to get the next frame, the client will send "continue" (with header) to server.

When receiving "Esc" from keyboard, the client send "bye" instead of "continue" to notify the server.



SIGPIPE

When a process *A* write to a pipe without any process listen on (read) the pipe, the os would send SIGPIPE to process *A*.

It might happen when a server is trying to send bytes to a client, while the client had disconnected.

To handle the SIGPIPE at server part, just add a signal handler to handle the signal (SIGPIPE) and close the socket fd of that client.

Blocking I/O vs Synchronized I/O

Not exactly.

Blocking I/O refer to the I/O operation that will block to wait the data when data isn't prepared.

To understand synchronized I/O, we first explain asynchronized I/O. When using asyncronized I/O, the I/O operation will return immediately even when the data isn't prepared; however, OS will run the I/O operation in the background and send signal to the process when done.

Therefore, synchronized I/O actually includes blocking and nonblocking two types; for blocking, the definition is same as above; for nonblocking, the process only read the data that is ready and return (will not execute in background).