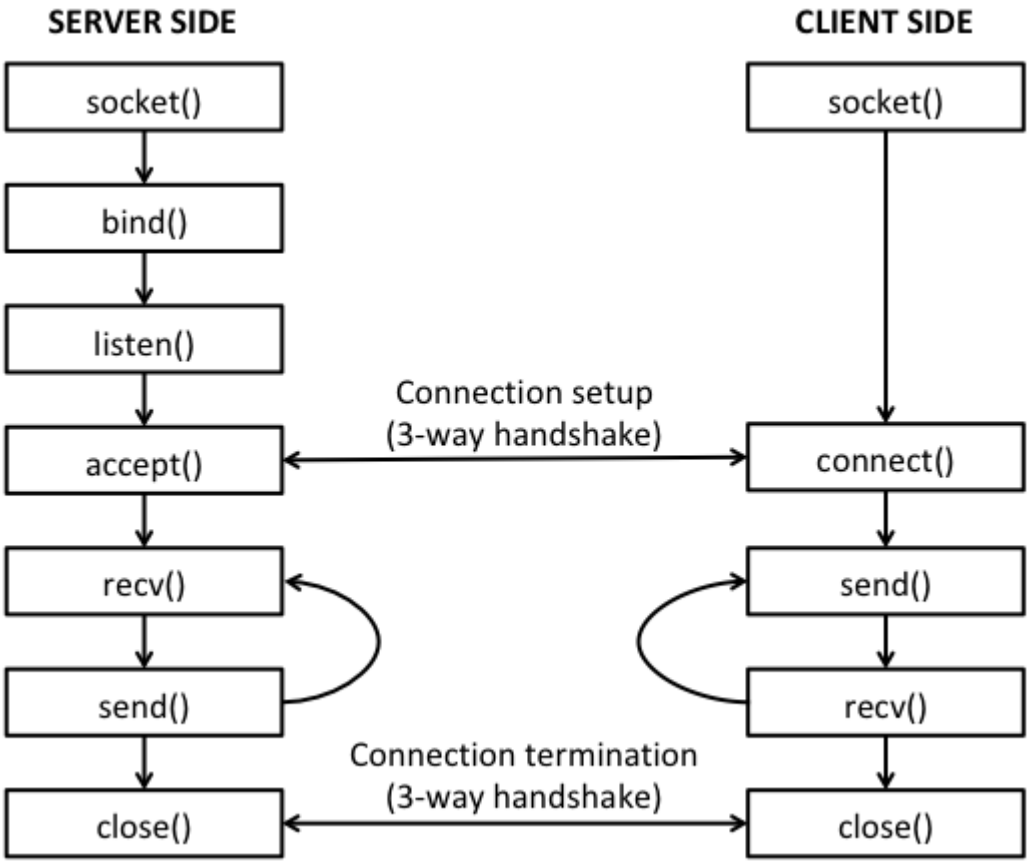
Peer to Peer File Distributed System

setsockopt() earlier when we get seg fault process already have port and IP assigned to it. so we use SO_REUSEADDR and SO_REUSEPORT so we can reuse the port and IP.



We have used TCP and built an application layer on top of it. We have defined chunk size and are prepending the length of data (int value) as a header so that the receiver can decode data size efficiently. Both the client and receiver will know the size of the data. TCP breaks data into more chunks internally, so at the client side, we are using a while loop to gather all the data.

When the client first sends the connection request to the tracker, it sends it on the connection port. All subsequent communication takes place on a dedicated port which is provided by the socket library.

Select and poll system calls are used for asynchronous calls, while send and receive system calls are blocking calls.

We do not have bind() in the TCP socket connection for the client as we don't need a dedicated port for a client.

Any data structure which is manipulated by multiple threads needs a locking mechanism.

Threadpool:

A thread pool is a software design pattern used to manage and efficiently utilize a pool of worker threads to execute tasks concurrently. It helps in reducing the overhead of thread creation and destruction, especially in scenarios where tasks are short-lived and frequent. The main components of a thread pool include:

- A queue of tasks waiting to be executed
- A fixed number of worker threads that continuously pull tasks from the queue
- A mechanism to manage the lifecycle of threads and handle task distribution

Piece Selection algorithm:

The Piece Selection algorithm is a crucial component in peer-to-peer file sharing systems, particularly in BitTorrent-like protocols. It determines which pieces of a file to request from peers in order to optimize download speed and overall network efficiency. Some common piece selection strategies include:

- Rarest First: Prioritizes downloading pieces that are least common among peers, promoting better distribution of pieces across the network.
- Random First Piece: Selects a random piece to download initially, helping to kickstart the download process.
- Strict Priority: Downloads pieces in sequential order, useful for streaming applications.
- End Game Mode: Requests all remaining pieces from multiple peers when nearing completion, to reduce the impact of slow peers.

The choice of piece selection algorithm can significantly impact the performance and efficiency of a peer-to-peer file-sharing system, influencing factors such as download speed, upload contribution, and network congestion.

Peer selection strategies also play a crucial role in optimizing file distribution. These strategies determine which peers to connect to and exchange data with, based on factors such as bandwidth, availability, and network proximity. Effective peer selection can lead to faster downloads, reduced network congestion, and improved overall system performance. Additionally, implementing mechanisms for fairness and incentivizing uploads can help maintain a healthy balance in the peer-to-peer ecosystem.

we also store chunk-wise sha so if any client sends another chunk we can identify it

This allows us to verify the integrity of individual chunks as well as the entire file. If a client sends a chunk that doesn't match the stored chunk-wise SHA, we can immediately identify and reject it. This granular verification is particularly useful in scenarios where chunks are downloaded from multiple peers, ensuring that each piece of the file is authentic and uncorrupted.