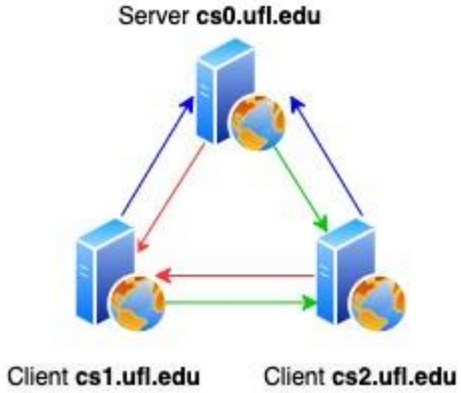
Table of Contents

1. Project Topology and Structure
2. Descriptions of Main Features
   - A. Establishing Peer Nodes
   - B. Communicate with Messages
   - C. Tranfer file and Traffic Controls
   - D. Assemble the File
   - E. Termination

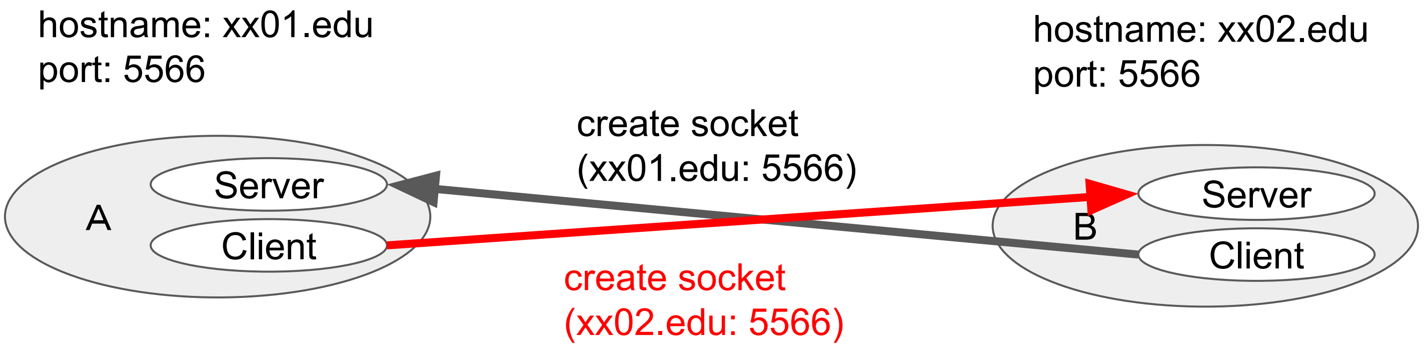
Project Topology and Structure

Project Structure - Topology



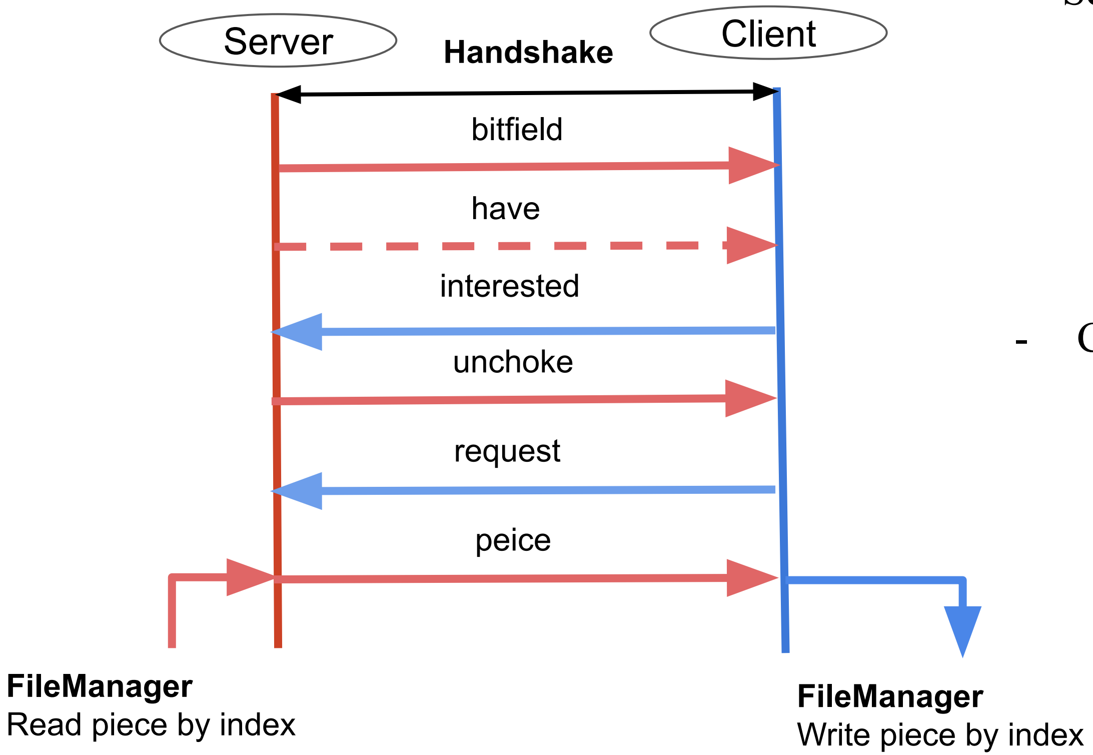
- Arrow shows the streaming direction (Each pear will be both server & client)
- Each nodes will have 2*(N-1) total socket objects
    - **ServerSocket** listening port x which will be connected by N-1 clients connections
    - Also, **Client thread** has to build **Socket** objects to connect with N-1 other Servers

Construct New Peers



1. Peer A Server recieves socket connection request from Peer B Client
2. Peer A Client then (triggered or in a regular interval) sends socket connection request to Peer B Server

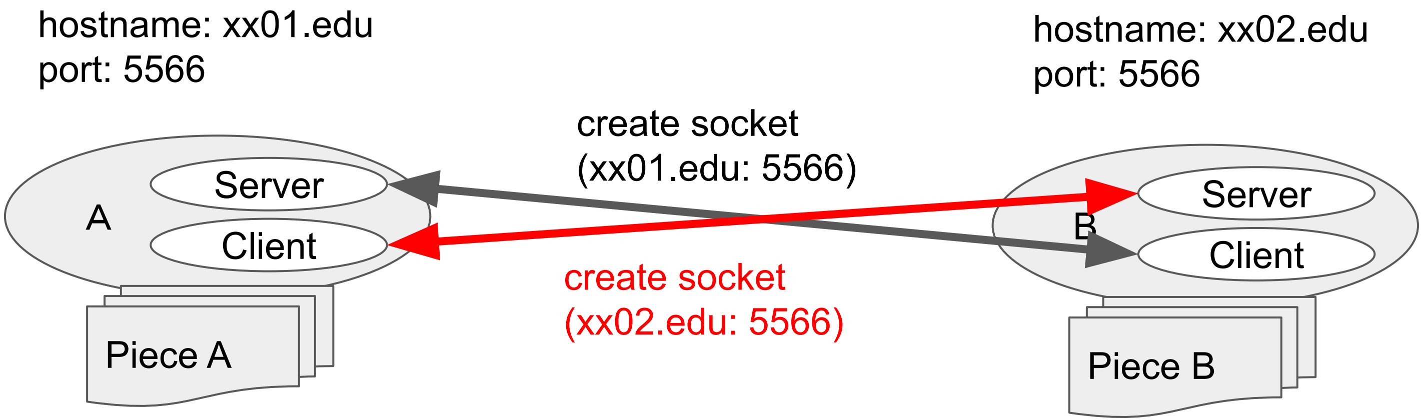
Function Progress



- Server will be the content uploader
  - Sending **bitfield** after **hanshake**
  - Send **have** msg after **peice**
  - Send **piece** msg
  - **unchoke, choke**

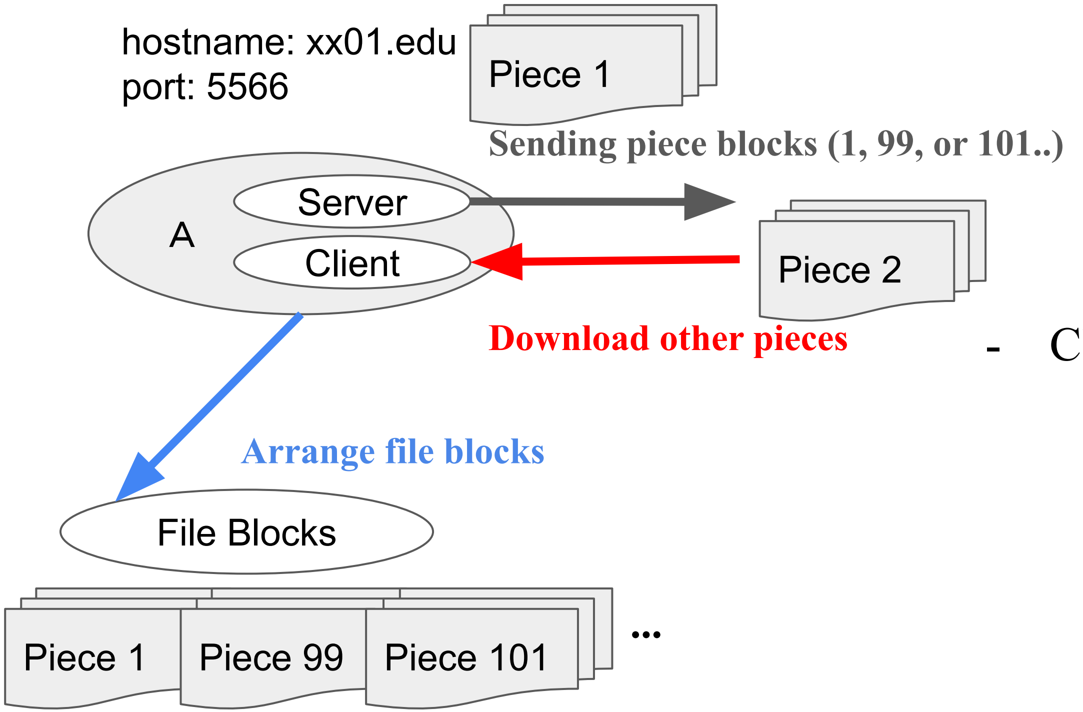
- Client will be a content downloader
  - **interested** or **not interested**
  - return **request** msg
    (after receive unchoke msg)

Handshake and Socket Purpose



- Handshake between Server A & Client B, Server A has build a threading with Client B. Same process between Server B & Client A
- What’s the purpose of this two different socket?
  - Piece A will be pass by the black
  - Piece B will be pass by the red

File Sharing



- **Server** will be the content **uploader**
    - Sending **bitfield** after hanshake
    - Send **have** msg after hanshake
    - Send **piece** msg (Content)
    - **choke** and **unchoke** (max k neighbors)
- Client will be a content **downloader**
    - **interested** or **not interested** (after receive bitfield or have msg, waiting for unchoke msg)
    - return **request** msg (after receive unchoke msg)

Descriptions of Main Features

A. Establishing Peer Nodes (1)

Parsing config files - Common.cfg and PeerInfo.cfg

Commmon.cfg

1. Parse the config file.
2. Using Singleton to store system parameters (FileName, PieceSize...)
3. All threads in the same node reuse the same object to read system parameters

PeerInfo.cfg

1. Parse the config file with the same PeerId with process argvs
2. Create Peer Object with hostname, port, and status flags
3. Status flags - isInterested, isChunked, isComplete

A. Establishing Peer Nodes (2)

Establing TCP connection between peers

Server

1. Listen to port stored in system singleton
2. Create Handler thread if others client binds

Client

1. Create client thread with socket with all neighbors' server.
   (if this Peer has not contain target file)

A. Establishing Peer Nodes (3)

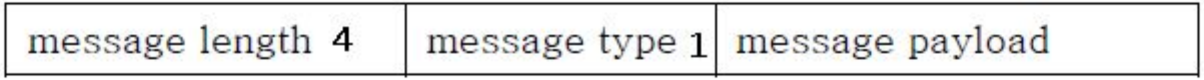
Handshaking

- Sends and receive the Handshake message and checks if it is the correct neighbor.
- Check for the header if it is "P2PFILESHARINGPROJ”.
- Check for the peerID.

Handshake Header	Zero bits	Peer ID

B. Communicate with Messages

Actual Message



message length 4	message type 1	message payload

Four types of message

1. **No payload**: choke, unchoke, interested, notinterested, end(self defined)
2. **4-byte payload**: have, request
3. **variable size payload**: bitfield
4. **4-byte block index + variable size payload**: piece

Decide how to treat incoming message by message type.

C. Tranfer file and Traffic Controls (1)

Bitfield, interested/not-interested

1. Peer Server will notify its current file status to new connect neighbor
2. New neighbor will reply interested in file pieces or not

Request and have

1. Request
   - Peer Client will request for the specific piece, after recieving the “permission”
2. Have
   - Maintain a map will new obtained pieces, broadcast to ever other Peers.

C. Tranfer file and Traffic Controls (2)

Choke/unchoke, optimistic unchoking

1. Choke/unchoke

   i. Random select k nodes if the Peer has the whole file

   ii. Measure own downloading rate from each neighbor, select the k most beneficials

2. Optimistic unchoking

   i. The selected peer has the highest priority

   a. It will not been choke unless it been remove from being the Optimistic

D. Assemble the File (1)

Pieces being generated and transferred

Use RandomAccessFile to read/write a piece

1. file seek to (block_index * block_size) position
2. Read/Write for block_size of bytes

Managed by FileManager class which is a singleton class.
Since this is a singleton class, the seek position will be shared with every threads, thus a lock is used.

D. Assemble the File (2)

Maintain self pieces information

There are three groups of pieces

1. Interested: Pieces this peer doesn’t have
2. Downloading: Pieces currently downloading from other peer
3. Have: Pieces fully downloaded

E. Termination

1. When a Peer gets the whole file
   a. Broadcast to others sending “complete” msg
   b. Receive the return “complete”
   c. Close all their client thread and connections (No receiving or requesting file)
   d. The Server thread will be remain, since others may ask for pieces.

2. System terminates
   a. All neighbors and self peer is complete → Close Server thread
   b. Close the timers and the socket listener