BitTorrent-like Swarming Protocol - Assignment #2

2018007965 김산

[구현 방식 요약]

이 BitTorrent_Protocol 프로젝트에서는 5개의 peer가 파일을 10KB(10000 Bytes) 청크 단위로 나눠서 서로 송/수신을 하면서 한 명이 온전한 파일을 가지고 있던 seeder의 상태에서 최종적으로 다섯 명의 peer가 모두 온전한 파일을 가지게 된다. 전체적인 Protocol의 진행과정에서 peer 중에 한 명이 하나의 파일 청크라도 가지게 되면 server로서의 역할을 할 수 있게 되며 전체 과정에서 하나의 peer가 파일을 요청해서 가지고 오기도 하고 파일의 요청을 받아 전송을 할 수 있게 됨을 의미한다.

[Optional Implementation]

- 3개의 download neighbor 쓰레드를 생성하여 수행한다.

-> LinkedBlockingQueue의 사이즈를 3으로 지정하여 매 연결 때마다 friend를 3명으로 유지하며 청크를 받아온다.

- 피어(upload neighbor thread)는 3개의 청크를 다운로드 받은 후에 혹은 5초가 지나면 갱신된 bit map을 friend peer에게 보내준다.

-> 구현 못 함.

- 10초 동안 청크 다운로드가 이루어지지 않으면 해당 피어를 포기하고 다른 피어와 friend 관계를 맺어 다운로드 청크 다운로드를 시도한다.

-> FileReceiveJob의 run() 메소드에서 시작 후 청크를 받아오는 데까지 걸리는 시간이 10초가 넘어가면 Thread를 종료한다.

[구현 방법]

1) 폴더 안에 다섯 개의 configuration file이 같은 디렉토리 내에 있게 한다.

2) javac BitTorrent.java를 통해 컴파일 한다.

3) Seed의 역할을 할 포트(8001 ~ 8005번 중 하나)의 번호와 같은 이름의 폴더를 생성하고 그 안에 시드의

역할을 할 파일을 담아 놓는다. ex) BitTorrent₩8001₩a.jpg

3) 5개의 콘솔 창을 열어 각각 **java BitTorrent {포트 이름}.txt {BitTorrent에 사용할 파일}**을 입력하여 BitTorrent를 실행한다. **ex) java BitTorrent 8001.txt a.txt**

4) Ctrl+C를 콘솔 창에 입력하면 해당 Peer가 종료된다.

[클래스 구성]

1. Peer.java

BitTorrent의 Peer 역할을 하기 위한 Java 클래스이다. ipAddress 와 port 번호를 가지고 있으며, 한 콘솔에서 5개의 Peer에 대한 정보를 가지고 있고 그 중 자신의 정보가 어떤 것인지 알기 위해 Boolean 타입의 self 변수를 통해 자신인지 아닌지 파악한다. 파일이 저장될 경로와 프로토콜 상에서 사용될 파일의 이름을 각각 saveFolder와 file로 가지며, 이 파일의 이름을 fileName으로 갖는다. 또한 이 파일의 메타데이터를 담고 있는 fileData 변수도 선언한다. 그리고 마지막으로 이 Peer가 seeder인지 아닌지를 판단하기 위해 Boolean 타입의 seeder 변수를 갖는다.

Peer(String ipAddress, int port)

내 자신이 아닌 Peer에 대한 생성자이다.

Peer(String ipAddress, int port, Boolean self, String filename)

Peer의 기본 생성자로, 나 자신일 경우 현재 위치에 자신의 포트 번호와 동일한 파일 디렉토리를 지정하여, 거기에 그 파일을 담게 된다. 또 만약 해당 파일이 존재하는 파일이라면 자신은 seeder가 된다. 그리고 해당 파일을 인자로 하여 fileData를 생성한다.

2. FileWithMetaData.java

Peer가 어떤 파일을 가지고 있는지를 주변 Peer에게 전달하기 위해 필요한 정보를 담고 있는 자바 클래스이다. file과 그 file의 원래 이름인 originalFileName, 그리고 그 file의 size, 그리고 file의 고유 fileID를 갖는다. 그리고 구분자를 가지며, FileChunk의 정보를 가진 FileChunk 타입의 배열을 갖는다. 그리고 완료의 여부를 파악하는 Boolean 타입의 complete를 가진다.

FileWithMetaData(File file)

FileWithMetaData의 기본 생성자이다. 인자로 받은 file을 바탕으로 originalFileName을 정해주고, file이 존재할 경우, fileSize를 정해준다. 그리고 청크의 개수를 의미하는 fileChunks 배열의 사이즈를 실제 파일의

사이즈를 바탕으로 정해준다. 그리고 랜덤으로 중복되지 않게 파일의 ID를 지정해준다. 그리고 fileChunks의 각 인덱스에서 FileChunk 객체를 position 정보를 담고 있는 객체로 생성한다. 그리고 complete 변수를 true로 바꾼다.

addFileChunk(FileChunk fileChunk)

현재 자신이 가지고 있는 파일 정보에 새로운 FileChunk를 추가하기 위한 메소드이다. 만약 추가하려는 청크가 이미 있는 청크라면 **"FileChunk [chunkID=30] is dumped"**와 같은 메시지를 출력하며 이 파일이 받아지지 않고 버려졌다는 메시지를 출력한다.

3. FileChunk.java

FileChunk를 BitTorrent Protocol 내에서 ID를 부여해서 부족한 FileChunk에 대한 요청을 하고 그 요청을 바탕으로 필요한 FileChunk를 보내기 위해 필요한 자바 클래스이다. 파일의 이름인 fileID를 가지며, 각 chunk의 번호를 나타내는 chunkID를 갖는다. 그리고 파일을 특정 바이트 단위로 쪼개서 담기 위한 byte 배열 chunk를 갖는다. 그리고 특정 위치에 FileChunk를 넣기 위해 position을 갖는다.

FileChunk(String fileID, int chunkID, byte[] chunk)

fileID와 chunkID, chunk를 인자로 받아 FileChunk 객체를 생성해준다.

FileChunk(String fileID, int chunkID, long position)

fileID와 chunkID, position을 인자로 받아 FileChunk 객체를 생성해준다.

4. FileReceiveJob.java

파일을 청크 단위로 받기 위한 자바 클래스이다. 나 자신을 나타내는 self를 가지며, 연결할 friend들을 갖는다.

FileReceiveJob(LinkedBlockingQueue<Socket> friends, Peer self)

FileReceiveJob의 기본 생성자이다.

getEmptyChunkIDs(FileChunk[] origin, FileChunk[] myChunk)

필요한 파일 청크를 받기 위해서 내 파일 청크와 Server의 파일 청크를 비교해서 내가 가지고 있지 않은 파일 청크의 ID가 담긴 배열을 return한다

run()

이 클래스에서 생성된 getEmptyChunkIDs와 receiveChunk를 활용해서 파일을 받는 작업이 시작되면 RandomAccessFile 클래스를 통해서 받아야할 원본파일과 같은 사이즈의 빈 파일을 생성하고 필요한 청크들을 friend에게 요청을 해서 빈 공간의 특정 위치에 파일 청크들을 채워가며 최종적으로 원본 파일을 완성하는 작업을 거친다. **"File Chunk Requests Send[FileChunk [chunkID=33], FileChunk [chunkID=34], FileChunk [chunkID=35]]"**와 같은 형식으로 파일 청크에 대한 요청을 보낸 것을 콘솔창에 출력을 하게 된다.

receiveChunk(Socket clientSocket, FileChunk requestChunk, InputStream is, RandomAccessFile raFile, FileChunk[] origin)

자신에게 필요한 특정 청크를 받아오는 메소드이다. 메소드 내에서 시작 시점과 종료 시점의 시간을 파악하고, 파일이 완성되었는지를 complete 변수를 통해 확인한다. 청크를 하나 받게되면, 자신의 콘솔 창에서 **"FileChunk(30/36) Download (91%)Completed From 8001"** 와 같은 예시처럼 전체 청크 중에 몇번째 청크를 받았고 파일의 완성도는 어느 정도이고 어느 포트로부터 청크를 받아왔는지 확인할 수 있고, 걸린 시간도 파악할 수 있다. 파일이 다 받아졌을 경우, 다 받아졌다는 메시지를 출력하면서 자신이 seeder로서의 역할도 하게 된다.

5. FileServeJob.java

Client Thread에서 특정 청크에 대한 요청을 받았으면 그 요청에 맞춰서 특정 청크를 다시 Client로 내보내는 역할을 하는 자바 클래스이다. 또한, 청크 요청과 응답을 교환하기 전에 자신의 MetaData를 Client에게 제공하는 역할을 한다. 연결에 필요한 socket을 가지며, 송/수신 과정에서 필요한 file, 그리고 그 file의 MetaData인 fileData, 그리고 file의 총 사이즈를 가진다.

FileServeJob(Socket socket, FileWithMetaData fileData)

FileServeJob의 기본 생성자로, fileData의 정보를 바탕으로, file과 fileSize를 정해준다.

run()

실행이 되면 OutputStream을 통해 Client에게 필요한 MetaData를 제공한다. 그리고 그 제공 정보에 맞추어 파일 청크에 대한 요청을 받는다. 요청 값이 null이거나 그 길이가 0이면 실행을 종료한다. 그 외의 경우에는 RandomAccessFile을 만들어서 sendChunkRequest를 통해 요청받은 청크를 내보낸다. **"FileWithMetaData [originalFileName=a.jpg, size=357633, fileID=1ed3b35d-89bb-4a3d-b041-15c66e5f72a9] sent to 60342"**와 같은 형식으로 MetaData를 전송했다는 것을 콘솔 창에 메시지로 출력한다.

sendChunkRequest(FileChunk chunkRequest, RandomAccessFile raFile, OutputStream os)

RandomAccessFile을 읽을 시작점을 요청을 받았던 청크의 ID를 바탕으로 지정한 후 필요한 청크를 읽어온다. 이 읽어온 정보를 byte 배열에 담아 OutputStream을 통해 write한다. 그 이후 다음과 같은

출력을 콘솔 창에 출력한다.

Client: /127.0.0.1:60176 connected.

Progressing: [22/36]10000/357633 Byte(s)

File transfer completed.

연결된 클라이언트의 IP주소와 포트번호, 그리고 10000Byte의 청크가 몇 번째인지의 정보가 출력이 되고 전송이 완료되었다는 메시지가 출력된다.

6. ServerListeningJob.java

Port를 연결 대기 상태를 유지하여 자신이 한 개의 청크라도 가지게 되면 Server Thread를 돌려서, 파일을 전송할 수 있는 역할을 하게 하는 자바 클래스이다.

run()

자신의 포트를 바탕으로 welcomeSocket을 하나 만들고 이 Socket이 5초 안에 연결을 잡지 못한다면 **"Waiting for Request in Port: 8003...."**를 콘솔 창에 출력시키면서 연결이 될 때까지 기다린다. 만약 연결이 성공적으로 되었다면 바로 FileServeJob의 Thread를 가동시켜 자신의 Peer가 Server의 역할을 하도록 한다.

7. IOUtils.java

OutputStream 또는 InputStream을 통해 Object를 쓰고 읽기 위한 메소드를 포함하는 자바 클래스이다.

writeObjectOutputStream(Object obj, OutputStream os)

인자로 받은 Object가 직렬화가 가능하다면 ObjectOutputStream을 통해 Object를 write한다.

readObjectFromInputStream(InputStream is)

필요한 Object 정보를 ObjectInputStream을 통해 읽어와서 그 값을 return값으로 한다.

8. BitTorrent.java

위의 7개의 자바 클래스를 바탕으로 실제 BitTorrent 프로토콜을 실행하여 파일을 청크 단위로 전송하고 받는 역할을 수행하는 메인 메소드가 담겨있는 자바 클래스이다.

main(String[] args)

이 메소드 안에서 IP주소와 Port번호가 담긴 txt파일을 Command-Line에서 불러와서 self라는 변수로 스스로의 피어를 지정한 후 나머지 4개의 피어를 Peer 타입의 ArrayList를 생성하여 그 정보를 담아준다. 그리고 LinkedBlockingQueue를 사이즈 3으로 만들어서 연결을 할 friend를 만들어준다. 그리고 Server 역할을 하는 Thread를 실행시켜주고, 스스로가 seeder가 아닐경우 Client의 역할을 하는 Thread를 만드는 메소드를 호출한다.

createClientThread(Peer self, ArrayList<Peer> neighbor, LinkedBlockingQueue<Socket> friends)

self를 제외한 4개의 peer에서 각각 하나의 Client Thread를 연결해줄 friend를 하나씩 지정해준다. 그 연결이 5초 안에 이루어지지 않을 경우 다른 연결을 시도해서 최종적으로 세개의 FileReceiveJob(Client 역할) Thread가 돌아서 한번에 필요한 청크를 총 3개씩 받아올 수 있도록 한다.

parsingConfig(String configFileName, String fileName, ArrayList<Peer> peer)

이 메소드에서는 Command-Line에서 받았던 IP/Port tuple 정보가 들어있는 txt파일 이름과, 이 BitTorrent에서 사용할 파일의 이름을 인자로 받는다. 그리고 tuple들의 정보를 담아서 관리할 ArrayList까지 인자로 받는다. 첫번째 tuple을 self로 지정해서, BitTorrent에서 사용할 파일의 이름을 포함한 Peer 객체를 하나 만들고, 나머지 4개의 tuple은 Peer 객체를 만들어 ArrayList에 담아주고, self를 return한다.