컴퓨터네트워크 assignment#2

2015002606

황재준

* 프로그램 실행 방법

1. C:\myTorrent\Seeder\origin\_file 까지 폴더를 생성해주세요
2. origin\_file 폴더 내부에 옮길 파일을 넣어주세요. (한 개만 넣어야 함)
3. myTorrent 폴더 내에 configuration.txt를 넣어주세요

(configuration 파일은 한 줄 단위로 [ip addr] [port num]가 적혀 있습니다.)

1. Seeder.java를 실행해주세요.
2. leech1~4.java를 실행시켜주세요. (순서 상관없음)
3. 전송된 파일은 각 Leech\_[n]의 폴더의 incoming에 들어있습니다.

* 프로그램 작동 방식

1. Server는 먼저 파일을 10KB 단위의 chunk로 나누어, Seeder/chunks의 위치에 저장합니다. chunk는 [number].part 파일로 저장됩니다.
2. 각 Seeder와 Leecher는 1개의 Server Thread, 1개의 ClientSet Thread (Client Thread를 관리함), 3개의 Client Thread를 가집니다.
3. ClientSet은 각 Client Thread를 시작하고, 종료되었을 때 chunk를 합치고 종료합니다.
4. Server와 Client는 chunk를 최대 3회 반복해서 전송합니다. 더 이상 전송할 chunk가 없거나, 문제가 생긴 경우엔 3회 미만 일 수 있습니다.
5. Server는 Client에게 자신의 Chunkmap을 보내주고, Client는 이 Chunkmap을 자신의 Chunkmap과 비교해서 자신이 받을 Chunk가 있는지 확인하고, 이에 대한 flag를 세팅해서 Server로 보내줍니다.
6. 있을 경우 Client는 필요한 Chunk의 이름을 송신합니다. 또한 자신의 Chunkmap을 업데이트 합니다.
7. Server는 받은 문자열(필요로 하는 Chunk 이름)과 자신이 가지고 있는 파일들과 비교하여 요청받은 청크의 File 객체를 생성합니다.
8. Server는 7번에서 생성한 File 객체를 받아 ChunkfileObj 라는 오브젝트를 생성합니다.

*\* ChunkfileObj는 클래스를 소켓 통신하기 위해 만들어진 객체로 Serialized 되었습니다. 청크파일의 이름, 데이터, 사이즈의 정보를 담고 있습니다. 위 정보들은*

*get, set의 method로 구현되어 있습니다. \**

1. Server는 ChunkfileObj를 client에 전송합니다.
2. Client는 받은 ChunkfileObj를 이용해 /chunks의 위치에 저장하고 1초가 휴식합니다. (영문명세에 작성됨: console msg를 천천히 확인하기 위함)
3. 3회의 반복이 끝나거나, 특정 이유로 탈출해서 과정이 끝난 경우, 소켓 연결을 끊고, Client는 새로운 Server를 받습니다. (랜덤하게)
4. Chunkcount(청크 개수)와 자신의 Chunkmap의 크기가 동일하고, Chunkcount가 0이 아니라면(처음에 Chunk가 아무것도 없는 Leecher는 파일의 Chunkcount를 모르기 때문에 0으로 초기화 되어있으며, Chunk를 하나라도 받게되면 원본 파일의 총 Chunk 개수를 알 수 있음) Client Thread를 종료하고, Chunk들을 합쳐 원본 파일과 동일한 파일을 생성합니다.
5. 텍스트이(가) 표시된 사진

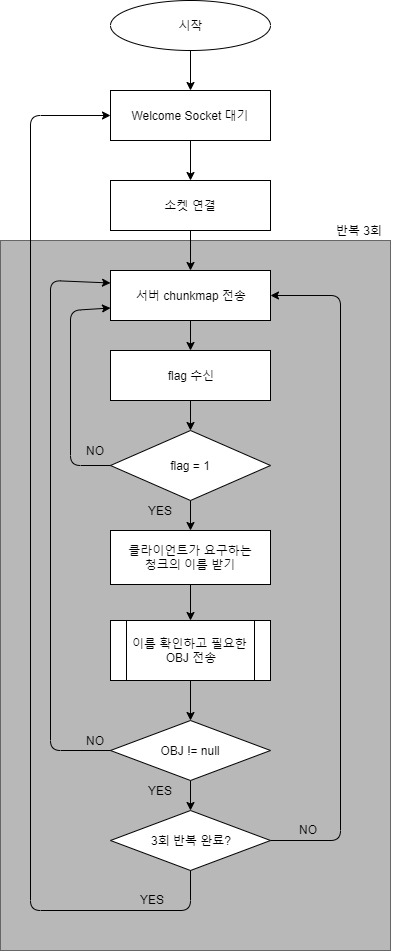
   자동 생성된 설명위 과정이 끝나면, ClientSet Thread도 작동을 멈추고, 오직 Server Thread만 작동합니다. (영문 명세)

**↑텍스트이(가) 표시된 사진

자동 생성된 설명 다운로드 후 Thread 상태 (0번이 Server)**

**← 다운로드 중 Thread 상태 (2,3,4가 Client)**

**(1번은 ClientSet Thread)**

* 서버 작동 방식

서버 쓰레드는 클라이언트와 연결하면 최대 3개의 청크를 보내주는 반복을 실행합니다.

서버는 먼저 자신의 chunkmap을 보내주게 됩니다.

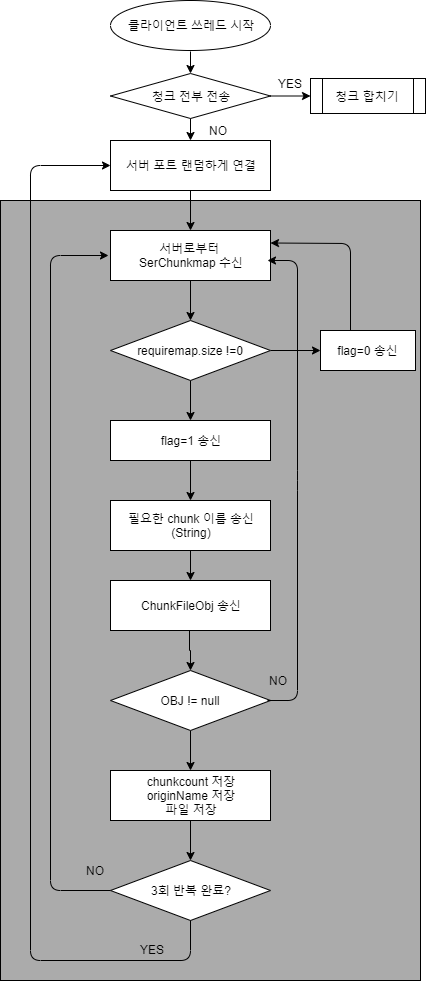
클라이언트가 서버의 chunkmap을 확인하고, 필요한 청크가 있다면, flag=1로 보내주게 됩니다.

그렇지 않은 경우에는 flag=0 됩니다. flag가 0인 경우에는 반복문을 탈출합니다.

flag가 1인 경우에, 서버는 클라이언트가 보내주는 특정 String을 받습니다. 이 String은 chunk의 이름과 동일하고, 서버를 이를 자신의 chunk와 비교하여 필요한 chunk를 ChunkfileObj라고 하는 Serialized된 object를 보내줍니다.

서버는 보낸 obj 파일이 null인 경우 반복문을 탈출합니다. (클라이언트의 멀티쓰레드 시 발생하는 문제 해결)

위 과정인 3회 반복된 경우 (for문 탈출도 포함), 서버는 클라이언트와 연결을 종료하고, 새로운 연결을 기다립니다.

* 클라이언트 작동 방식

이 flow chart는 하나의 클라이언트의 쓰레드에 대해서 설명하고 있습니다.

클라이언트 쓰레드는 연결할 포트를 랜덤하게 선택하고, 그 서버와 연결합니다.

연결이 성공하면, 아래 과정을 3회 반복합니다.

먼저 서버로부터 서버의 chunkmap을 수신합니다.

서버의 chunkmap과 자신의 chunkmap을 비교해서, 자신이 필요한 chunk의 이름을 set으로 가지는 requiremap을 생성합니다.

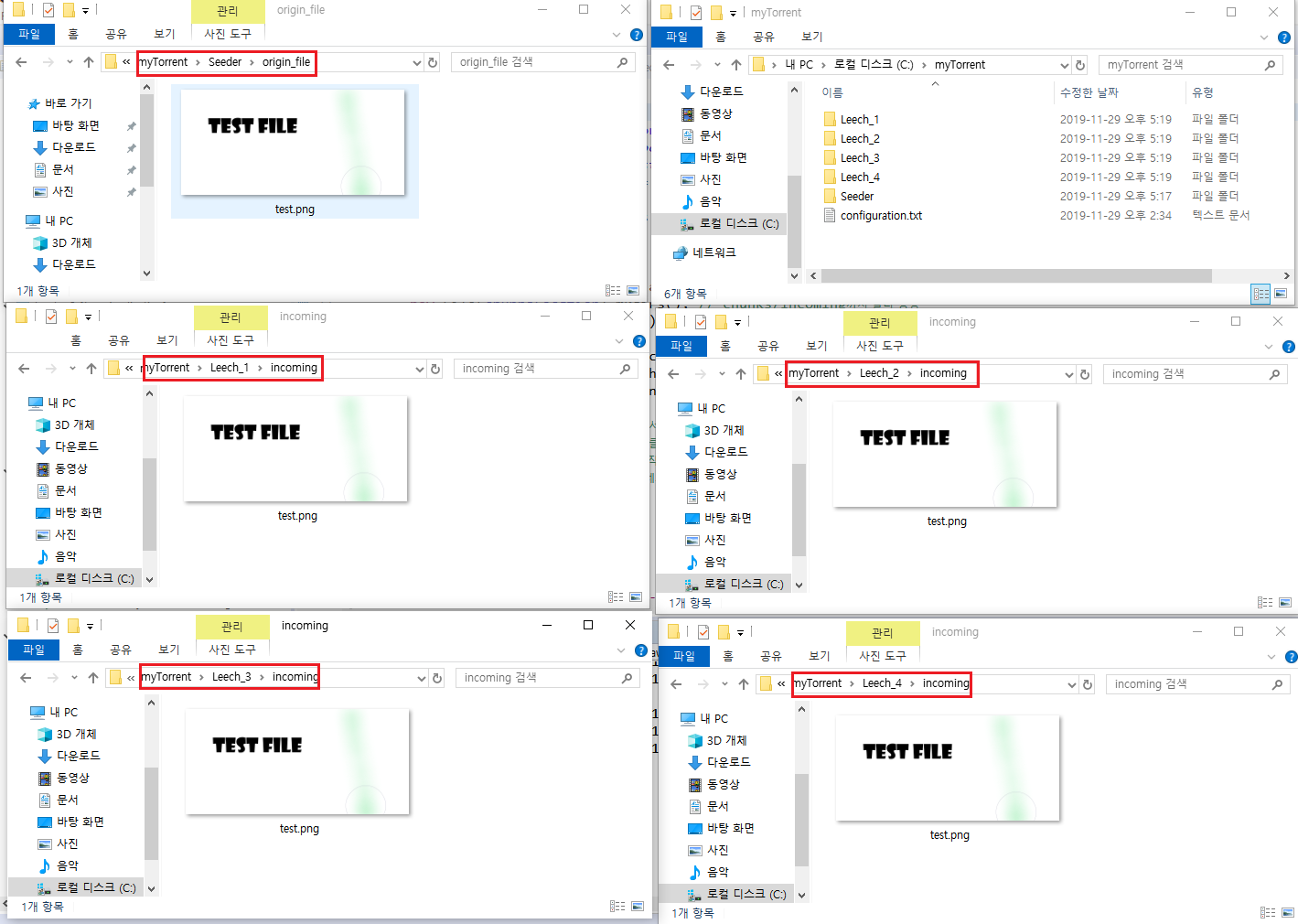
requiremap의 크기가 0이라면 서버에게 flag=0를 송신하고 반복문을 탈출하고, 그렇지 않다면 다음 flag=1을 송신합니다.

flag=1인 경우 랜덤하게, 필요한 chunk를 하나 요청합니다.

도착한 chunk가 null이 아니라면 chunk의 정보들을 추출하고 (chunkcount: chunk의 전체 개수, originName: 파일 원본명) 파일을 저장합니다.

과정이 완료되면, 새로운 서버와 연결하고, 모든 chunk가 도착하면 종료하고 chunk를 합칩니다.

* 스크린샷, 실내, 컴퓨터, 테이블이(가) 표시된 사진

  자동 생성된 설명결과 화면 (Chunk)
* 결과 화면 (Origin File)
* 프로그램 구조

1. **개요**

Seeder는 Leecher와 몇 가지 다른 부분이 있습니다. Seeder는 원본 파일을 가지고 있으며, 이를 Chunk 단위로 분할합니다. 그 외 동작과 구조는 Leecher와 동일합니다.

하나의 Server Thread, 3개의 Client Thread를 관리하는 ClientSet Thread로 구성되어 있습니다.

IP address와 Port number는 configuration.txt 파일에서 읽습니다. 이 파일은 myTorrent 폴더 안에 위치해 있고, 한 줄씩 [IP addr] [Port num]으로 되어있습니다.

java 파일 코드에는 전체 4가지의 class가 있습니다. Seeder, Server, ClientSet, Client.

Chunk는 ChunkFileObject라고 하는 Serializable한 객체로 전송됩니다. 이 객체는 chunk의 이름, 원본 파일명, 파일 데이터, 전체 chunk 개수, chunk의 크기를 가지고 있으며, set과 get이 가능합니다.

1. **Seeder class, Leecher class**

이 class에는 Server와 Client가 사용하는 자신의 Chunkmap이 있습니다. 이 Chunkmap은 HashSet으로 선언되었으며, synchronized되어 Client Thread들끼리와 Server Thread 간에 발생할 수 있는, 동시성 문제를 해결하였습니다. (이후 세부 설명)

클립아트이(가) 표시된 사진

자동 생성된 설명

1. **Server Thread (serThread)**

Server Thread는 자신의 파일을 모두 다운받아도 Welcome Socket을 닫지 않습니다. 고로 다른 Peer에게 지속적으로 파일을 전송해줄 수 있습니다.

앞서 서술한대로, flag를 세팅하여 Client가 원하는 Chunk가 있을지 없을지 확인합니다.

위 기능은 **requiredChunk** method로 구현되어 있습니다. 자신의 Chunk 폴더의 파일들을 확인하고, 이름을 대조하여 일치하는 파일을 Serialized object에 담아 전송합니다.

1. **ClientSet Thread**

이 Thread는 Client Thread를 start하고 전체 쓰레드가 종료되면, chunk를 합쳐서 하나의 파일로 만듭니다. 쓰레드를 종료하기 위해서 join() method를 사용하였습니다.

1. Client Thread (cliThread)

Client는 자신의 Chunkmap과 Server의 Chunkmap을 비교하여, 자신이 필요한 Chunk의 이름을 뽑아냅니다.

스크린샷이(가) 표시된 사진

자동 생성된 설명이 기능은 **filerequire** method로 구현되어 있습니다.

 Thread끼리는 동일한 Chunk를 받게 되는 상황이 발생할 수 있습니다. 그렇기 때문에 Server의 Chunkmap을 받은 직후에 이 method를 실행합니다.

자신의 Chunkmap(mymap)을 확인하고, Server의 Chunkmap (sermap)이 빈 경우에는 null 을 return하고 그렇지 않은 경우에는 두 가지 case로 나눕니다.

mymap이 비어있다면, sermap의 아무것이나 요청합니다.

mymap이 비어있지 않다면, mymap에 있는 string을 제외하고, 요청합니다. 위 기능을 위 해서 requiremap이라는 sermap을 복사한 set을 생성합니다.

위 method는 synchronized 되어있기 때문에, 실행 중에는 다른 Thread는 기다려야 합니 다. 이 함수 내에서 자신의 Chunkmap을 업데이트하기 때문에, 다음 Thread는 중복되는 Chunk를 요청하지 않게 됩니다. 이 결과는 같이 첨부한 leecher의 console msg를 통해서 알 수 있습니다. 파일 수신은 오직 chunk의 개수만큼만 이루어집니다.



실내이(가) 표시된 사진

자동 생성된 설명 클라이언트는 timeout으로 10초를 설정하여, 10초간 기다리고, 접속이 되지 않으면 다른 Server로 연결을 시도합니다. ConnectException, SocketException에도 동일하게, 새로운 접속을 시도합니다. 소켓 연결 부분은 while문으로 감싸져 있기 때문에 새로운 Server로 접속이 가능합니다.