분산시스템 텀프로젝트 최종보고서

1. 201912335 이홍석

2. 구현 환경

언어: Java, IDE: IntelliJ, 외부 라이브러리: CM

3. 기능별 구현 완료 여부 체크

3.1 클라이언트는 파일 업데이트(생성,삭제,수정)를 자동 또는 수동으로 탐지: 구현 완료(수동)

3.2 클라이언트는 파일 업데이트가 서버의 파일과 충돌하지 않음을 확인 (logical clock 이용): 구현 완료

3.3 클라이언트는 충돌하지 않은 업데이트된 파일을 서버로 전송하여 기존 파일 덮어씀: 구현 완료

3.4 삭제된 파일은 서버에서도 삭제: 구현 완료

4.1 파일 및 공유할 사용자(클라이언트) 선택: 구현 완료

4.2 선택 파일을 (서버를 거쳐) 선택한 클라이언트에게 전송: 구현 완료

4.3 공유된 파일 업데이트되면 서버뿐만 아니라 공유된 클라이언트에게도 전송 하여 업데이트: 구현 완료

----------오늘은 여기까지~~---------

4. 전체 클래스 및 메소드 설계

1. CMServerApp

CMServerApp은 시작과 동시에 CMServerStub 객체를 생성하고, CMServerEventHandler를 등록하여 클라이언트로부터 오는 이벤트를 기다린다.

2. CMClientApp

CMClientApp은 프로세스가 실행되는 동안 사용자에게 반복적으로 input을 받고, 받은 input에 따라 각각 다른 기능을 제공한다.

2.1 Login: userId와 password를 입력받아 서버에 로그인한다.

2.2 Logout: 로그인이 되어 있는 경우, 서버로부터 로그아웃한다.

2.3 SetFilePath: configuration에 기본적으로 설정된 default file path를 원하는 경로로 변경할 수 있다.

2.4 RequestFile: 서버로부터 특정 파일을 전송받는 작업을 요청할 때 사용한다.

2.5 PushFile: 클라이언트에서 서버로 특정 단일 파일을 전송하는 경우 사용한다.

2.6 SendMultipleFiles: 클라이언트에서 서버로 여러 개의 파일을 한 번에 전송하는 경우 사용한다.

2.7 Synchronization: 클라이언트에서 입력받은 경로에 있는 모든 파일이 서버와 비교하여 업데이트가 된 상태인지 확인 후, 업데이트되지 않은 파일을 판단하여 서버로 전송한다. 한편, 해당 파일에 대해 연결 관계를 가지는 다른 클라이언트를 찾아 해당 파일의 동기화 작업을 실시한다.

2.8 ShareMultipleFiles: 클라이언트에서 공유하고자 하는 파일들을 서버를 거쳐 다른 클라이언트에게 전달하고, 해당 파일에 대한 두 클라이언트 간의 연결 관계를 생성한다.

3. CMWinClientApp: ClientApp과 동일한 기능을 하나, 콘솔 출력과 입력이 GUI에서 이뤄진다. 숫자 input을 통해 수행하고자 하는 기능을 선택하고, 수행할 기능은 showComfirmDialog에서 이뤄진다. 한편, File Submit버튼을 누르면 클라이언트에서 해당 경로에 있는 파일들의 목록을 보여준다.

4. CMWinServerApp: ServerApp과 동일한 기능을 하나, 오른쪽에 User이름을 입력하고 Submit버튼을 누르면, 해당 유저의 디렉토리에 해당하는 서버 파일들 목록을 보여준다.

5. CMServerEventHandler: Login, Logout이벤트를 수신했을 경우 작업의 결과를 콘솔에 출력하고, 필요한 경우 적절한 리턴값을 클라이언트에게 다시. 전송한다. 또한, 클라이언트에서 파일을 전송했다는 메시지를 받았을 경우, 파일 전송의 성공 여부를 출력한다.
한편, 동기화 작업 시, 받은 문자열을 공백 기준으로 분할 후, 클라이언트에는 없지만 서버에 남아있는 파일이 있다면 찾아 삭제한다.
한편, 공유 이벤트의 경우는 서버가 중개자 역할을 하기 때문에, 클라이언트로부터 받은 목적지로 해당 파일과 logical clock을 넘겨주는 역할을 한다.

6. CMClientEventHandler: 서버에게 요청한 로그인 작업이 잘 수행되었는지 여부를 이벤트의 리턴으로 받아, 결과를 출력하고, 서버의 데이터베이스에 변화가 생겼을 경우, 이를 감지하여 해당 세션에 누가 새로 참여하거나 떠났는지를 출력한다. 또한, 서버에서 동기화 가능한 파일들의 리스트를 수신 시, 해당 파일들을 다시 서버로 전송하는 역할을 한다.

5. 기능 구현

3. 단순 파일 동기화 기능

3.1 클라이언트는 파일 업데이트(생성, 삭제, 수정)를 자동 또는 수동으로 탐지

클라이언트의 파일 업데이트 탐지 과정은

WinClientApp에서 동기화하고자 하는 디렉토리를 입력

➔ 디렉토리의 파일 중, 마지막 업데이트 테이블에 없는 파일이 새롭게 있다면 새로 생성된 파일로 간주, 마지막 업데이트 테이블에만 있고 현재 디렉토리에 없는 파

일이 있다면 삭제된 파일로 간주, 그 이외의 파일 중 마지막 수정 시간이 마지막 동기화 시간 이후인 파일은 수정된 파일로 간주하여 각각 테이블에 저장

3.2 클라이언트는 파일 업데이트가 서버의 파일과 충돌하지 않음을 확인 (logical clock 이용)

클라이언트는 파일 동기화 과정에서 총 두 번의 DummyEvent를 보내게 된다.

첫 번째로, 전체 파일 리스트를 전송하여, 클라이언트에서 삭제된 파일은 서버에서도 삭제되도록 함

➔ 새롭게 생성된 파일이나, 수정되었다고 판단되는 파일들의 리스트에 logical clock 값을 덧붙여 서버에게 해당 파일들의 동기화 가능 여부를 물음

➔ 서버는 자신의 logical clock값과 비교하여, 서버로부터 동기화가 가능하다고 판단되는 파일들의 목록을 되돌려 받고, 해당 파일들을 fileTransferManager를 이용하여 실제 서버로 전송

3.3 클라이언트는 충돌하지 않은 업데이트된 파일을 서버로 보내어 기존 파일을 덮어씀

동기화 요청 과정에서

logical clock을 비교하여 서버가 동기화 가능한 파일 리스트를 보내 줌과 동시에, 해당 파일을 덮어써야 한다는 사실을 인지하고, 새로 받을 파일을 미리 삭제

➔ 클라이언트는 업데이트 가능한 파일을 서버로 전송

3.4 삭제된 파일은 서버에서도 삭제

3.2 에서,

동기화하고자 하는 폴더의 파일과 서버의 파일을 비교하여, 서버에만 존재하는 파일을 삭제

4. 다른 클라이언트와 파일 단순 공유 기능

4.1 파일 및 공유할 사용자(클라이언트) 선택

클라이언트는 작업 요청 시,

파일명과 공유할 사용자명을 입력

4.2 선택 파일을 (서버를 거쳐) 선택한 클라이언트에게 전송

파일명과 공유할 사용자명을 입력 후,

공유할 파일을 서버에게 선 전송

➔ 서버가 파일을 성공적으로 수신 시, 클라이언트는 바로 이어서 공유할 사용자명과 파일명을 메세지로 전송

➔ 서버는 가지고 있는 파일을 다시 전송받고자 하는 클라이언트에게 전송

4.3 공유된 파일 업데이트되면 서버뿐만 아니라 공유된 클라이언트에게도 전송하여 업데이트

최초로 공유 실행 시

클라이언트의 공유 유저 테이블에 공유한 파일과 공유 중인 사용자가 한 쌍으로 저장

➔ 서버에서 파일 공유 시, 다른 클라이언트에게 메세지를 보내 공유 받는 클라이언트의 공유 사용자 테이블 역시 업데이트

➔ 이후, 어떤 클라이언트에서 동기화를 요청하더라도, 해당 클라이언트의 공유 사용자 테이블을 탐색하여 해당 파일을 공유 중인 사용자에게 파일을 전송

6. 클라이언트와 서버 동작 프로토콜

클라이언트와 서버의 메시지 송, 수신은 CM 라이브러리의 EventHandler를 통하여 이루어지고, 보낸 명령어를 기반으로 메시지를 주고받는 과정은 cmStub객체 내부에서 이루어진다.

파일 동기화 시 프로토콜의 경우, 클라이언트 -> 서버 -> 클라이언트 ->의 과정으로 메세지 전송이 이루어 진다.

1. 클라이언트 -> 서버

```
CMDummyEvent due = new CMDummyEvent();
due.setID(107);
due.setDummyInfo(String.join(" ", strFilesList));
```

여기서, strFilesList는 현재 동기화하고자 하는 폴더 기준 모든 파일을 공백으로 구분하여 저장한 String 배열이다.

서버는 첫 번째 메세지를 수신받아 메세지를 구분자를 기준으로 분할하고, 자신이 가지고 있는 파일 중, 클라이언트에서 삭제된 파일들을 서버에서도 삭제한다.

```
due = new CMDummyEvent();
due.setID(105);
//String[] fullMessage = new String[nFileNum];
List<String> fullMessage = new ArrayList<String>();
for(int i = 0; i < nFileNum; i++)
{
    //System.out.println("isEmpty: " + absFiles[i].equals(" "));
    if(absFiles[i].equals(" "))
        continue;
    String[] tempStrFile = absFiles[i].split("/");
    String strFile = tempStrFile[tempStrFile.length-1];
    //fullMessage[i] = strFile  + ":" +
logicalClock.get(strFiles[i]);
    fullMessage.add(strFile  + ":" +
logicalClock.get(strFiles[i]));
    //System.out.println(fullMessage[i]);
}
//due.setDummyInfo(String.join(" ", strFiles));//동기화시킬
파일들의 목록을 공백으로 구분하여 전송
```

```
due.setDummyInfo(String.join(" ", fullMessage));//동기화시킬
파일들의 목록을 공백으로 구분하여 전송
System.out.println(String.join(" ", fullMessage));
m_clientStub.send(due, m_clientStub.getDefaultServerName());
```

삭제가 완료되면, 클라이언트는 현재 파일들의 이름과 logical clock 값을 "파일 이름:logical clock 값"의 형태를 한 쌍으로 하여, 공백으로 구분 후 서버로 전송한다.
서버는 전송받은 파일 리스트를 구분자를 기준으로 파싱하여, 자신의 logical clock 테이블과 메세지의 logical clock값을 비교하고, 전송받은 logical clock값이 더 클 경우, 동기화 가능한 파일로 보고 리스트에 추가한다.

2. 서버 -> 클라이언트

```
CMDummyEvent newDue = new CMDummyEvent();
newDue.setID(105);
newDue.setDummyInfo(String.join(" ", acceptedFilesList));
```

서버는 자신의 logical clock과 비교하여, 동기화 가능한 파일을 공백으로 구분하여 클라이언트로 재전송한다.
클라이언트는 동기화 가능한 파일들을 구분자 기준으로 파싱 후 FileTransferManager를 통해 하나씩 전송한다.

```
due = new CMDummyEvent();
due.setID(106);

due.setDummyInfo(strFiles[i] + " " + strReceiver);
m_clientStub.send(due, m_clientStub.getDefaultServerName());
```

한편, 동기화 작업 후, 공유중인 사용자 리스트를 찾아서 공유중인 사용자들에게 파일을 공유하는 이벤트까지 실행한다. 이 때 메세지 형식은 "파일명 + " " + 사용자"가 한 쌍으로 이루어져 있다.
서버는 이 메세지를 받아 동기화 된 파일들을 공유중인 클라이언트들에게도 전송한다.

파일 전송 시 프로토콜의 경우, 클라이언트 -> 서버 -> 클라이언트 순으로 메시지 전송이 이루어진다.

1. 클라이언트 -> 서버

```
CMDummyEvent due = new CMDummyEvent();
due.setID(106);
due.setDummyInfo(fileName + " " + strReceiver);
m_clientStub.send(due, m_clientStub.getDefaultServerName());
```

링크 생성을 위해, 클라이언트에서 서버로 링크를 형성하고자 하는 파일과 사용자 이름을 공백으로 구분하여 서버로 보내준다.
이를 전송받은 서버는 임시로 해당 파일명과 사용자를 가지고 있다가, 다시 클라이언트에게 메세지를 보내 링크를 형성하도록 유도한다.

2. 서버 -> 클라이언트

```
CMDummyEvent linkDue = new CMDummyEvent();
linkDue.setID(108);
linkDue.setDummyInfo(fileName + " " + sender);
m_serverStub.send(linkDue, strTarget);
```

서버는 파일 전송을 위한 메세지 전송에 앞서, "파일 이름 + " " + 공유자 값"을 한 쌍으로 하여 메세지를 전송하여, 공유받을 클라이언트도 링크를 형성하도록 한다.
메세지를 전송받은 클라이언트는 자신의 테이블에 해당 파일과 공유중인 사용자를 한 쌍을 추가한다.

```
CMDummyEvent newDue = new CMDummyEvent();
newDue.setID(106);
newDue.setDummyInfo(fileClock);
m_serverStub.send(newDue, strTarget);
```

이어서, 서버는 파일 전송을 위해 "파일명 + ":" + logical clock 값"을 한 쌍으로 하는 메시지를 전송하여, 전송받을 클라이언트가 업데이트된 파일을 미리 지울 수 있게 함과 동시에, 전송받을 클라이언트가 logical clock값을 업데이트하도록 한다.
메세지를 전송받은 클라이언트는 자신이 공유받을 파일의 logical clock값을 업데이트하고, 해당 파일을 전송받을 준비의 일환으로 예전에 받은 파일을 삭제한다.

7. 컴파일 및 실행 방법

1. 전체적으로 IntelliJ환경에서 Build Project를 한다.
2. CMWinServerApp과 CMWinClientApp을 차례대로 Run한다.
3. CMAppClient에서 100을 입력하고 userId, password를 입력하여 서버에 로그인한다.
4. 클라이언트 최하단 input box에 105를 입력하고, 동기화시키고자 하는 파일 디렉토리를 입력한다.
5. 클라이언트 최하단 input box에 106을 입력하고, 전송하고자 하는 파일들의 디렉토리(공백으로 구분)와 파일 갯수, 전송받을 사용자를 입력한다.

서버의 우측 하단 input box에 유저 이름을 입력하면, 해당 유저 폴더에 관리되고 있는 파일들의 목록을 보여주고, 클라이언트의 우측 하단 input box에 디렉토리를 입력하면, 해당 디렉토리가 관리하고 있는 파일들의 목록을 보여준다.

8. 소스코드 포함 프로젝트 GitHub URL

https://github.com/Leehongseok-code/201912335_CM

9. 기능별 시연 영상 youtube 링크//영상촬영만 하면 끝!

https://youtu.be/d5nydvegNWc