FTP Server Programming Assignment 2

2018068622 임문경

Server argument : control port number / data port number (optional)

ex) java FTPServer.java 20 21

default argument : 2020 / 2121

Client argument : server ip / control port number / data port number (optional)

ex) java FTPClient.java 192.168.0.2 20 21

default argument : 127.0.0.0 2020 2121

SR Protocol 구현을 위한 추가사항

1. "TimerTask_" class 구현

기존의 TimerTask를 상속받아 생성된 TimerTask는 Chunk, read size, next point, output stream을 인자로 입력받아 각 chunk가 전송될때 마다 timerTask class내부에 해당 변수들을 저장하여 Timer가 만료될시 해당 chunk를 재전송하도록 설계하였습니다.

2. "DROP", "TIMEOUT", "BITERROR" commend 추가

위 commend들을 추가하였고 그에 따른 boolean변수들을 각각 설정하여 다음 "GET"또는 "PUT" 명령어 실행시 ERROR발생여부를 확인할 수 있도록 하였습니다.

3. 각종 변수 추가선언

-sender쪽 추가된 변수

1. int seqNum : 설정된 sequence number의 개수가 저장되어 있는 변수입니다.

2. int windowSize : 설정된 window size 값이 저장되어 있는 변수입니다.

3. int currentPoint : oldest unACKed packet의 sequence number값이 저장되어 있는 base pointer입니다.

4. int nextPoint : 다음으로 전송될 sequence number값이 저장되어 있는 next sequence pointer 입니다.

5. Timer[] timer : unACKed packet들 각각에 대한 timer가 저장되어 있는 배열입니다. 최대 windowSize만큼의 공간이 필요하지만 구현의 편의를 위해서 seqNum만큼의 공간을 할당하였습니다.

6. TimerTask_[] task : 각 timer에 대한 task가 저장되어 있는 변수입니다. timer 배열과 마찬가지로 구현의 편의를 위해서 seqNum만큼의 공간을 할당하였습니다.

7. boolean[] ACKed : out-of-order packet에 대해서 ACK가 들어왔을 경우 체크를 해 후에 in-order packet이 들어왔을 경우 올바른 current point로 이동할 수 있도록 도와주는 변수입니다.

8. Timer timeoutTimer : TIMEOUT시에 사용되는 timer변수입니다.

9. TimerTask_ timeoutTask : TIMEOUT시에 사용되는 task변수입니다.

10. int ackRangeCheck : 전송받은 ACK number가 올바른 window범위인지 체크해주는 변수입니다

- receiver쪽 추가된 변수

1. int expectedNum : 다음으로 들어올 seqNum값을 가지고 있는 변수입니다. 이 변수를 통해 in-order packet인지 out-of-order packet인지 여부를 확인할 수 있습니다.

2. int seqNum : 설정된 sequence number의 개수가 저장되어 있는 변수입니다.

3. int windowSize : 설정된 window size 값이 저장되어 있는 변수입니다.

4. byte[][] buffer : out-of-packet일 경우 값을 저장해놓을 buffer배열입니다. 구현의 편의를 위해 seqNum만큼의 공간을 할당하였습니다.

10. int seqRangeCheck : 전송받은 ACK number가 올바른 window범위인지 체크해주는 변수입니다.

4. TimerTask_ Class

기존에 java에서 지원하는 TimerTask Class를 상속받아 SR Protocol에서 Timer의 기능과 unACKed chunk를 임시로 저장하는 기능을 통합하도록 구현하였습니다.

TimerTask_ class의 인자로 저장할 chunk, 해당 chunk에 저장되어 있는 데이터양, 전송할 outputStream을 입력받아 schedule method로 지정된 시간이 지나면 입력받은 outputStream으로 저장된 chunk를 전송하도록 설계하였습니다.

SR Protocol을 위해 변경된 알고리즘

기존의 Assignment1와의 내용의 중복을 피하고 내용의 간소화를 위해서 Assignment1에서 추가되거나 변경된 사항들만을 기록하려고 하였습니다.

또한 client -> server 방향 뿐만아니라 server -> client 방향도 SR Protocol을 구현하였기 때문에 "Sender algorithm"과 "Receiver algorithm"으로 분류하여서 sender algorithm은 "GET"명령어에서 server측의 동작방식을, "PUT"명령어에서 client측의 동작방식을 한번에 설명드리고자 하였으며, Receiver algorithm에서는 "GET"명령어에서 client, "PUT"명령어에서 server측의 동작방식을 설명드리고자 하였습니다.

1. DROP, TIMEOUT, BITERROR Commend

"commend [seqNum]" 명령어를 통해서 에러를 발생시킬 sequence number를 지정할 수 있습니다.

"PUT"과 "GET"명령어 모두 에러 모의를 진행시키기 위해서 receiver와 sender측 모두 ERROR여부를 표시할 수 있는 boolean type의 변수가 선언되었으며 잘못된 명령어에 에러가 발생하는 것을 방지하기 위해서 각각 "PUT"또는 'GET"명령어가 실행될때마다 값을 FALSE로 초기화하는 작업을 하였습니다.

ERROR명령어를 입력받으면 각 console에 해당 에러모의에 대한 내용을 출력한 후 에러를 발생시킬 sequence number를 "errorChoNo"변수에 저장합니다.

2. Sender Algorithm

0. File Class에 이상이 없어서 파일 전송이 실행된 경우만 고려

1. 데이터 전송 최초에는 sequence number 0부터 window size가 모두 찰때까지 데이터를 한번에 전송합니다. 이를 구현하기 위해서 nextPoint와 currentPoint의 차이가 windowSize보다 작을 때까지 chunk를 전송하도록 while문을 구현하였습니다.

2. 최초에 window size만큼 데이터를 전송한 후 receiver에게 ACK를 입력받은 후 console로 입력받은 ACK number를 출력한다.

3. 입력받은 ACK를 현재 currentPoint와 비교하여 out-of-order ACK인지 아닌지를 판단하여 실행할 내용을 결정한다.

> 3-1) ACK number의 범위가 [currentPoint, currentPoint + windowSize)일 경우 unACKed chunk이므로 해당 chunk의 타이머를 종료하고, ACKed 체크합니다.
>
> 이때에 ACKed 배열에 "true"값을 저장하여 후에 in-order chunk가 전송되었을 때 올바른

currentPoint를 찾아갈 수 있도록 합니다.

또한 ACK number의 범위를 체크할 때 주의할 점은 currentPoint와 ACK number는 0 ~ sequence number까지 반복해서 할당되는 구조이므로 currentPoint + windowSize값이 sequence number값을 초과할 경우 올바른 ACK number 범위 역시 바뀌어야 하므로 각 상황에 맞는 올바른 범위인지를 나누어서 검사해야 합니다. 해당 검사값을 "ackRangeCheck"변수에 저장합니다.

3-2) ACK number의 범위가 올바른 범위가 아닌 경우에는 해당 ACK를 무시하고 다시 ACK를 수신합니다.

3-3) currentPoint와 입력받은 ACK number가 같은 경우 currentPoint를 한칸 이동시킵니다. 이때에 "ACKed"배열을 체크하여 currentPoint보다 먼저 들어온 ACK값들에 대해서도 역시 currentPoint를 이동시킵니다.

4. ACK를 수신받을 때 마다 현재 window에 여유공간이 있는지 검사하여 여유공간이 있을 시에 chunk를 전송합니다. chunk를 전송한 후에는 nextPoint값을 1증가시킵니다.

window의 여유공간 체크는 기본적으로 currentPoint와 nextPoint의 차와 windowSize를 비교하여 체크하지만 ACK number의 범위를 체크할때와 마찬가지로 값이 0 ~ sequence number까지 순환되는 구조이므로 각 상황에 맞게 값의 범위를 체크할 필요가 있었습니다.

5. chunk를 전송할 때마다 nextPoint와 errorChkNo가 같은지 체크합니다. 만약 nextPoint와 errorChkNo가 같다면 해당 chunk에 오류를 모의한다는 의미입니다. 그 후에 미리 정의해둔 "DROP", "BITERROR", "TIMEOUT" boolean값을 체크하여 어떠한 오류를 모의하는지 체크합니다.

5-1) DROP ERROR인 경우, 해당 chunk의 타이머가 그 전에 설정되어 있으므로 단순히 chunk를 전송하지 않고 다음 ACK를 전송받습니다.

5-2) TIMEOUT ERROR인 경우, 미리 설정한 chunk의 타이머 외에 하나의 timer를 더 설정합니다. 이는 앞전에 미리 정의해 둔 timeoutTimer와 timeroutTask 변수를 이용하며 이때에 설정될 timer의 시간은 기존의 재전송 시간의 2배인 2sec으로 설정했습니다.

5-3) BIT ERROR인 경우, chunk의 checksum값을 0xFFFF로 변경한 후 chunk를 전송합니다.

ERROR를 설정한 후 ERROR설정 변수들을 모두 false로 초기화하고 errorChkNo역시 -1로 초기화합니다.

3. Receiver Algorithm

0. File Class에 이상이 없어서 파일 전송이 실행된 경우만 고려

1. sender로부터 chunk를 입력받아 chunksum부터 체크한다.

    1-1) chunk의 checksum이 0xFFFF라면 BIT ERROR 메세지를 console로 출력한 후, ACK 메세지를 보내지 않고 다시 chunk를 입력받는다.

    1-2) chunk의 checksum이 0x0000라면 chunk의 sequence number를 전송할 ack의 ack number로 설정한 후 console에 전달받은 sequence number를 출력하고 sender에게 ack를 전송한다.

2. 전송받은 sequence number의 유효범위를 체크하여 각 상황에 맞는 동작을 실행한다. 이때에도 역시 순환되는 값 때문에 sequence number값을 두가지 범위로 나누어서 체크하여야 했습니다. 해당 검사값을 "seqRangeCheck"변수에 저장합니다.

    2-1) expectedNum과 sequence number값이 같은 경우 expectedNum을 1증가시킨 후 바로 FileOutputStream으로 전송한다. 그리고 expectedNum보다 먼저 들어와 버퍼에 저장되어 있는 값이 있는지 여부를 체크하고 버퍼값들을 순서대로 FileOutPutStream으로 전송하며 expectedNum값을 증가시킨다.

    2-2) expectedNum과 sequence number값이 다르지만 out-of-order chunk일 경우 buffer에 값을 저장한다.

    2-3) window의 유효범위 밖의 chunk일 경우 해당 값을 무시한다.

BIT ERROR인 경우를 제외하고 out-of-order chunk나 window의 유효범위 밖의 chunk들을 포함한 모든 chunk에 대해서 ACK를 전송합니다.

또한 receiver의 window는 sender쪽에서 조건문을 통해서 제어되고 있기 때문에 별도의 window를 구현하지 않았습니다.

programme 실행 방식

1. 실행환경 windows 10

windows PowerShell / Eclipse

java version "15.0.1"

Java(TM) SE Runtime Environment (build 15.0.1+9-18)

2. 실행방법

(ㄱ) "java FTPServer.java"명령어로 server를 구동시킨다. 만약 필요하다면 인자를 입력해 port번호를 지정할 수 있다.

(ㄴ) "java FTPClient.java"명령어로 client를 구동시켜 FTP server에 연결시킨다. 만약 필요하다면 인자를 입력해 server IP와 port번호를 지정할 수 있다.

3. 실행예시

실행화면은 잘 안보일 수 있어서 압축폴더에 별도로 압축해서 제출드리도록겠습니다.

[server]

```
Request:cd ../../desktop/ftp
Response: 200 Moved to C:\Users\moon\Desktop\ftp

request :TIMEOUT 13
response : PACKET TIMEOUT SET

rcv ack0
rcv ack1
rcv ack2
rcv ack3
rcv ack4
rcv ack5
rcv ack6
rcv ack7
rcv ack8
rcv ack9
rcv ack10
rcv ack11
rcv ack12
rcv ack14
rcv ack15
rcv ack0
rcv ack1
resend pkt13
rcv ack13
rcv ack2
rcv ack3
rcv ack4
rcv ack5
rcv ack6
rcv ack7

Requst: get secretDoc.docx
Response: 200 Containing 23599 bytes in total

request :BITERROR 5
response : PACKET BITERROR SET

seq 0
seq 1
seq 2
seq 3
seq 4
5 BIT_ERROR
seq 6
seq 7
seq 8
seq 9
seq 5
seq 10
seq 11
seq 12
seq 13
seq 14
seq 15
seq 0
seq 1
seq 2
seq 3
seq 4
seq 5
seq 6
seq 7
Request: secretDoc.docx
Request: 47
Response: 200 Ready to receive
```

[client]

```
###Success connection###
cd ../../desktop/ftp
C:\Users\moon\Desktop\ftp
TIMEOUT 13
request :TIMEOUT 13
response : PACKET TIMEOUT SET
get secretDoc.docx
###Open Data Link###
Received secretDoc.docx/ 23599bytes
seq 0
seq 1
seq 2
seq 3
seq 4
seq 5
seq 6
seq 7
seq 8
seq 9
seq 10
seq 11
seq 12
seq 14
seq 15
seq 0
seq 1
seq 13
seq 2
seq 3
seq 4
seq 5
seq 6
seq 7
Completed...
BITERROR 5
request :BITERROR 5
response : PACKET BITERROR SET
put secretDoc.docx
###Open Data Link###
secretDoc.docx transferred  / 23599 bytes
rcv ack0
rcv ack1
rcv ack2
rcv ack3
rcv ack4
rcv ack6
rcv ack7
rcv ack8
rcv ack9
resend pkt5
rcv ack5
rcv ack10
rcv ack11
rcv ack12
rcv ack13
rcv ack14
rcv ack15
rcv ack0
rcv ack1
rcv ack2
rcv ack3
rcv ack4
rcv ack5
rcv ack6
rcv ack7
Completed...
quit
```