<Programming HW2>

21800147 김유영

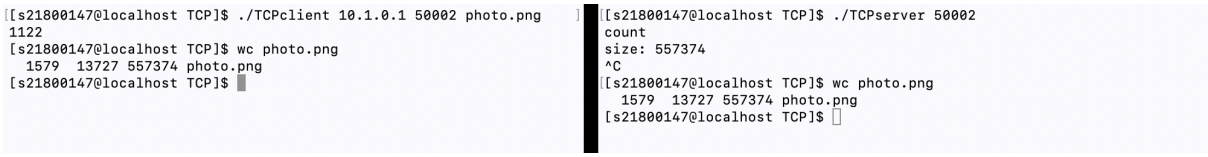
21800268 박민주

A. TCP를 이용한 파일 전송 프로그램

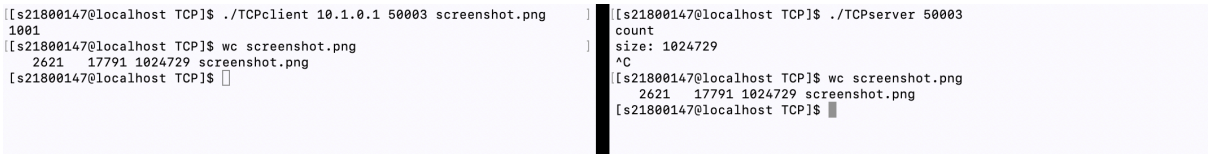
- 실행 결과

- 파일 전송을 마친 다음, server와 client에서 "$wc <file_name>" 의 명령어를 이용하여 파일이 오류없이 전송되었는 가를 확인하여야 한다.

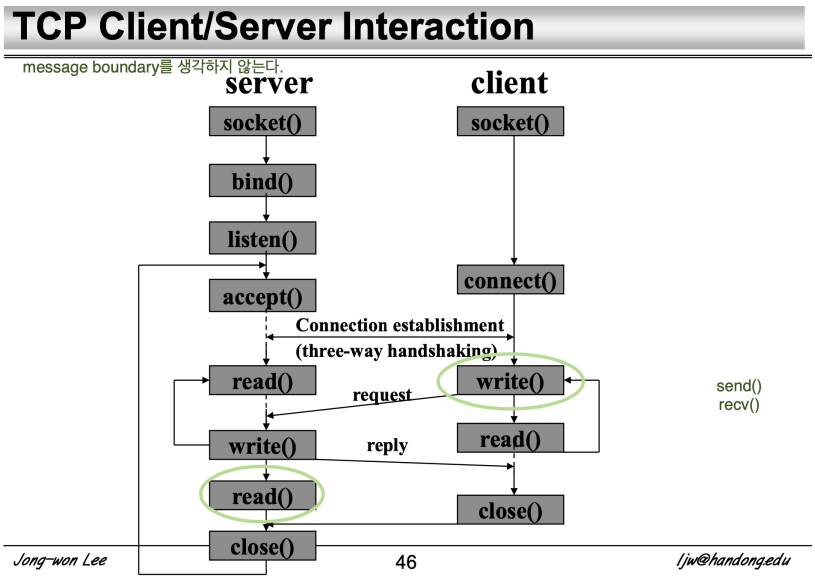
  - 557 KB 이미지 파일 전송 (557374 byte → 557374 byte)



  - 1MB 이미지 파일 전송 (1024729 byte → 1024729 byte)



→ 사진에서 보이다시피 client에서 전송한 파일의 이름과 내용이 정확하게 loss 없이 전달되었음을 알 수 있다. 그러나 전송 속도에 있어서 UDP에 비해 느리다는 단점이 있다는 것을 이번 실험을 통해 경험할 수 있었다.



- 프로그램 시에 유의하여 코딩한 점
  - TCP에서 가장 유의해야할 점은 TCP의 reliable한 특징을 지키는 것이었다. 최대한 loss가 발생하지 않고 전송 받은 파일의 이름, 내용까지 모두 그대로를 저장하려고 노력했다. 그래서 파일의 내용을 받기 전 파일의 전체 크기를 먼저 받아오게 하여 전송 받은 모든 내용을 빠지지 않고 저장될 수 있게 하였다.

- server
  - server는 시작한 후, client의 connection request를 기다린다.
  - 연결이 완료되면 read()함수를 사용하여 client로 부터 파일 이름을 받아 새로운 파일을 생성한다.
  - 그 후, client로부터 전송할 파일의 전체 크기를 받는다. (TCP의 reliable한 특성)
  - recv()함수를 사용하여 해당 파일의 끝에 도달할 때까지 파일을 지정한 버퍼의 크기만큼씩 반복하여 전송받아 저장한다.
  - 현재까지 전송 받은 내용의 크기와 파일 전체 크기가 동일해질 때, 파일과 socket 모두를 close시킨다.

- client
  - send()함수를 이용하여 해당 server에게 입력받은 파일의 이름을 전송한다.
  - server()에게 해당 파일의 전체 크기를 전송한다.
  - 파일의 내용을 전달하기 위해서, 파일의 끝에 도달할 때까지 파일의 내용을 정해 놓은 버퍼 사이즈씩 읽고 send()함수를 이용하여 전송하기를 반복한다.
  - 파일의 내용을 모두 전송하고 나서 파일과 socket 모두를 close한다.

B. UDP를 이용한 파일 전송 프로그램

- 실행 결과

- 파일 전송을 마친 다음, server와 client에서 "$wc <file_name>" 의 명령어를 이용하여 파일 전송의 오류 발생 유무를 확인하여라.

  - 557 KB 이미지 파일 전송 (557374 byte → 489472 byte)

```
[s21800147@localhost UDP]$ ./UDPclient 10.1.0.1 50002 photo.png
file: photo.png
ok file name
done
[s21800147@localhost UDP]$ wc photo.png
  1579  13727 557374 photo.png
[s21800147@localhost UDP]$
```

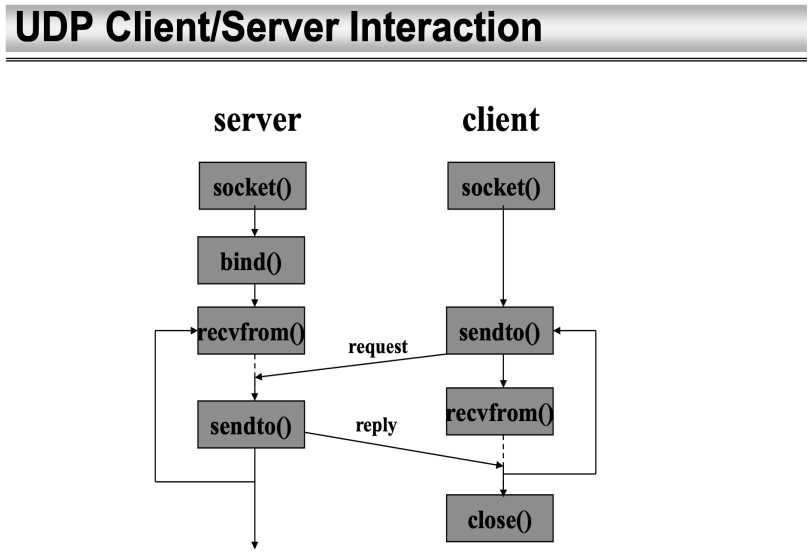
```
[s21800147@localhost UDP]$ ./UDPserver 50002
photo.png make
[s21800147@localhost UDP]$ wc photo.png
  1383  12215 489472 photo.png
[s21800147@localhost UDP]$
```

  - 1MB 이미지 파일 전송 (1024729 byte → 850649 byte)

```
[s21800147@localhost UDP]$ ./UDPclient 10.1.0.1 50002 screenshot.png
file: screenshot.png
ok file name
done
[s21800147@localhost UDP]$ wc screenshot.png
   2621   17791 1024729 screenshot.png
[s21800147@localhost UDP]$
```

```
[s21800147@localhost UDP]$ ./UDPserver 50002
screenshot.png make
[s21800147@localhost UDP]$ wc screenshot.png
  2243  14955 850649 screenshot.png
[s21800147@localhost UDP]$
```

→ 사진에서 보시다시피 일부 내용이 client에서 server로 전송되는 과정에서 유실되었음을 알 수 있다. 그러나 TCP에 비해 전송 속도에 있어서는 더 빠르다는 장점을 가지고 있다는 것을 이번 실험을 통해서 경험할 수 있었다.



-

- 프로그램 시에 유의하여 코딩한 점
  - ○ UDP는 TCP와 달리 패킷의 손실이 발생할 수 있다는 점을 유의하여야 한다. 프로그램의 결과를 보아 평균적으로 15%의 정보가 유실되는 것을 확인하였다. 또한 error recovery가 되지 않는 상황에서 정확하게 종료해야 하는 점을 유의하여 코드를 작성하였다.

- server
  - server는 시작한 후, client의 connection request를 기다린다.
  - 연결이 완료되면 recvfrom()함수를 사용하여 client로 부터 파일 이름을 받는다.
  - 해당 파일명의 파일을 생성하고 이를 다시 client에게로 알린다.
  - recvfrom()함수를 사용하여 해당 파일의 끝에 도달할 때까지 파일을 지정한 버퍼의 크기만큼씩 반복하여 전송받는다.
  - 파일수신이 완료되면 connection을 close()하여 연결이 정확하게 종료하도록 한다.

- client
  - sendto()함수를 이용하여 해당 server에게 입력받은 파일의 이름을 전송한다.
  - server()에게 해당 파일의 이름이 정확하게 전송이 되었는가를 유의하고, 파일 전송을 시작한다.
  - 파일의 내용을 전달하기 위해서 파일의 끝에 도달할 때까지 파일의 내용을 정해 놓은 버퍼 사이즈씩 읽고 sendto()함수를 이용하여 전송하기를 반복한다.
  - 파일의 내용을 모두 전송하고 나서 sleep(5)만큼 기다린 후 빈 문자열을 보내주어 다시 한번 종료됨을 명확하게 알려준 뒤, 파일과 socket 모두를 close한다.