컴퓨터네트워크

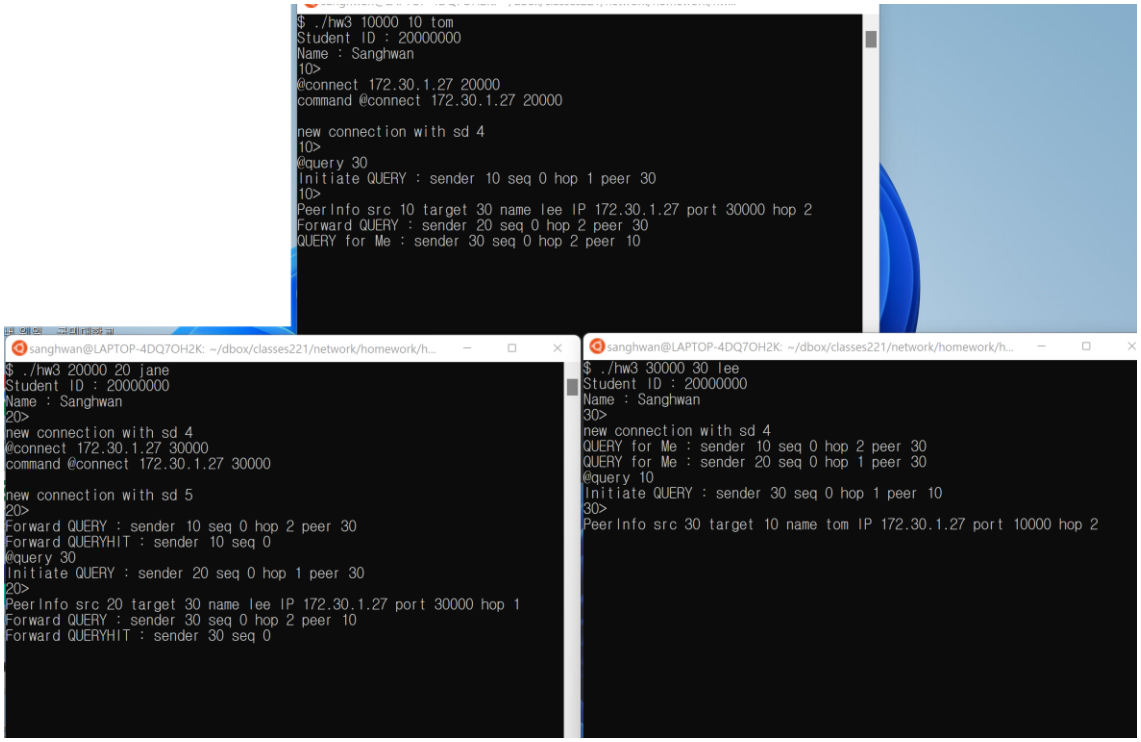
HW #3 (30 Points)

Due date : 2022/5/18 (eCampus)

제출물 : HW3_학번.c, HW3_학번.cpp, HW3_학번.java, HW3_학번.py (python3 사용)
하나의 파일에 모든 것을 다 구현함

이 과제는 간단한 p2p 사용자 검색 프로그램을 구현하는 것이다.

아래는 실행 예이다. 소스를 컴파일해서 hw3라는 실행파일을 만들어 냈다.
그리고 "hw3"를 실행하는데, 파라미터로 포트번호, 사용자 아이디, 사용자 이름을 제공한다.
프로그램은 학번과 이름을 출력한다. 그리고 프롬프트가 출력된다.



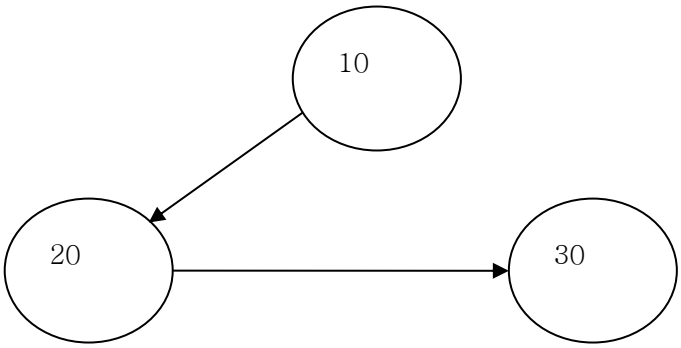
실행을 위한 문법은 아래와 같다.

$ ./hw3 tcpport userid username

- tcpport : 연결을 받기 위한 tcp 포트번호

- userid : 사용자 아이디 (정수이어야 함)
- username: 사용자 이름 (임의의 문자열)

위 그림에서는 위쪽 사용자가 왼쪽의 사용자에게 연결하고, 왼쪽 사용자는 오른쪽 사용자에게 연결한 것이다. 이에 따라 아래와 같은 그래프 형태의 연결 관계가 형성된다.



프로그램의 프롬프트에서는 3가지 명령을 실행할 수 있다.
첫번째 명령은 @connect 이다.

@connect hostname tcpport

- hostname : 연결할 프로그램의 호스트 이름 (도메인 이름이나 IP 주소 모두 가능함)
- tcpport : 연결 대상 프로그램의 포트번호

위 예에서는 사용자 10이 "@connect 172.30.1.27 20000" 명령을 통해 사용자 20에 연결하고, 사용자 20은 "@connect 172.30.1.27 30000" 명령을 통해 사용자 30에 연결한 것이다.

두번째 명령은 @query 이다.
연결설정이 끝나고 나면, 각 사용자는 @query 명령을 통해 특정 사용자의 IP 주소와 TCP 포트 번호 및 사용자 이름을 검색하게 된다.
@query 명령의 문법은 아래와 같다.

@query peerid

- peerid : 검색을 하고자 하는 사용자 아이디.

위 예에서는, 사용자 10이 "@query 30" 명령으로 사용자 30의 정보를 찾는다. 정보를 찾는 방법은 아래에서 설명한다.

마지막으로 @quit 명령을 통해 프로그램을 끝낸다.

세부 구현 내용:

1) “@connect” 명령

파라미터로 주어진 호스트와 포트번호를 이용하여 TCP 연결을 설정한다. 각 사용자는 **자신과 연결된 TCP 연결에 대한 정보를 잘 저장**하고 있어야 한다.

2) “@query”

이 명령어는 질의 메시지를 생성한다. 그 메시지의 이름이나 포맷은 적절하게 결정하면 된다. 당연히 찾고자 하는 아이디 정보는 포함되어야 한다. 위 예제에서는 QUERY라는 이름을 가지는 메시지를 생성하였고, 그 포맷은 그냥 아래와 같은 1줄짜리 문자열이었다.

QUERY qs qseq hop pid
qs: 보낸 사람 아이디
qseq: 보낸 사람이 생성한 질의 번호
hop: forwarding 한 회수
pid: 찾고자 하는 사용자 아이디

어쨌든 생성된 메시지는 해당 사용자와 연결된 **다른 모든** 사용자 프로그램으로 전달된다. 이러한 방식을 flooding이라고 한다.

이 질의 메시지를 받은 사용자 프로그램은 이 메시지를 분석하여, 우선 이 메시지가 전에도 들어왔는지를 검사하고, 이전에 들어온 메시지라면 무시한다.
처음 받은 질의 메시지라면 아래와 같이 처리한다.

1) 자신의 아이디와 같은 아이디를 찾는 경우: 바로 응답 메시지를 생성해서 보낸다. 위 예에서는 QUERYHIT라는 응답 메시지를 생성해서 보냈다.
2) 주어진 질의 메시지의 내용이 자신을 찾는 것이 아닌 경우: 메시지가 들어온 연결을 제외한 **다른 모든** 연결로 메시지를 복사해서 전달한다. 메시지를 전달할 때마다 몇 번의 사용자를 거쳤는지에 대한 정보를 저장하게 하여 이를 메시지에 포함시켜야 한다. 간단하게 QUERY 메시지 내에 counter 정보를 하나 추가해서 다른 사용자에게 forwarding 될 때마다 counter를 1씩 증가시키면 된다.
3) 최종적으로 질의 메시지가 찾고자 하는 사용자에게 도착하면 이 사용자는 응답 메시지를 생성해서 최초 질의한 사용자에게 전송한다. **이 때 이 응답 메시지는 해당 질의 메시지가 도달한 경로를 통해서 최초 질의자에게 도착하도록 해야 한다.** 절대로 flooding 방식을 사용하면 안 된다.
4) 메시지를 받을 때에는 어느 연결을 통해 받았는지를 저장하고 있어서 응답 메시지를 정상적으로 최초 질의자에게 전달할 수 있다.

최초 질의자가 응답 메시지를 받으면 아래와 같은 형식으로 그 정보를 출력한다.

PeerInfo src <질의한 아이디> target <사용자 아이디> name <사용자 이름> IP <IP 주소> port <포트 번호> hop <hop 수>

위 메시지를 이용하여 프로그램의 정확성을 검증하기 때문에 위 포맷대로 출력하면 된다. 따라서 **응답 메시지는 위 정보를 출력할 수 있는 내용을 포함하고 있어야 한다.** 중간에 디버깅을 위해서 출력하는 메시지는 원하는대로 출력해서 사용하면 된다.

아래의 예에서는 30번 아이디를 가진 사용자의 이름이 lee이라는 것을 알 수 있다. 그리고 IP 주소와 포트번호도 알 수 있다. 마지막 숫자 2는 대상 사용자가 질의한 사용자 10으로부터 2 hop 떨어져 있다는 것을 알 수 있다.

PeerInfo src 10 target 30 name lee IP 172.30.1.27 port 30000 hop 2

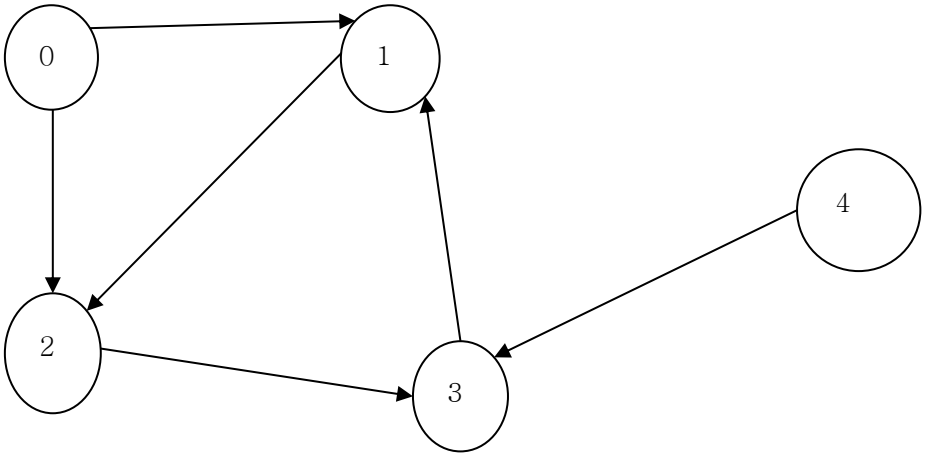
테스트:

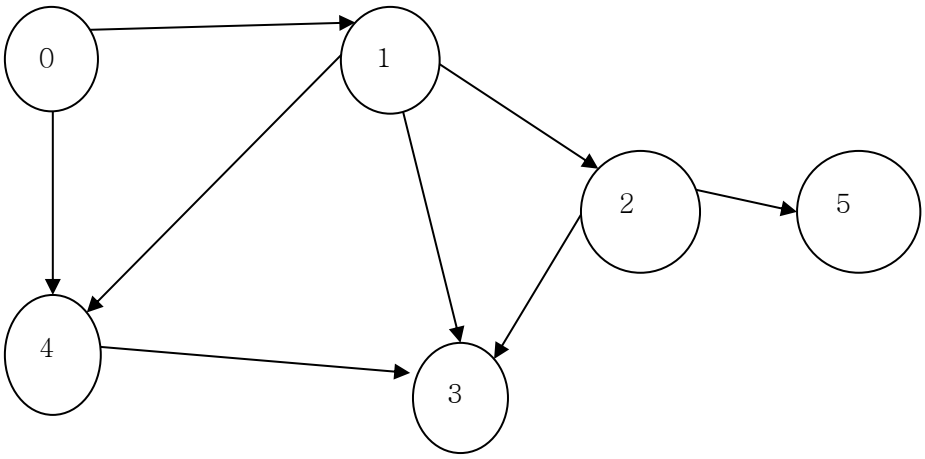
예를 들어 아래와 같은 좀 더 복잡한 토폴로지를 생성해서 테스트해보기 바람.

0번 사용자가 “@query 4” 실행

4번 사용자가 ”@query 0” 실행

…





테스트 자동화

첨부한 labtest.sh 파일을 소스코드가 있는 곳에 복사한다.

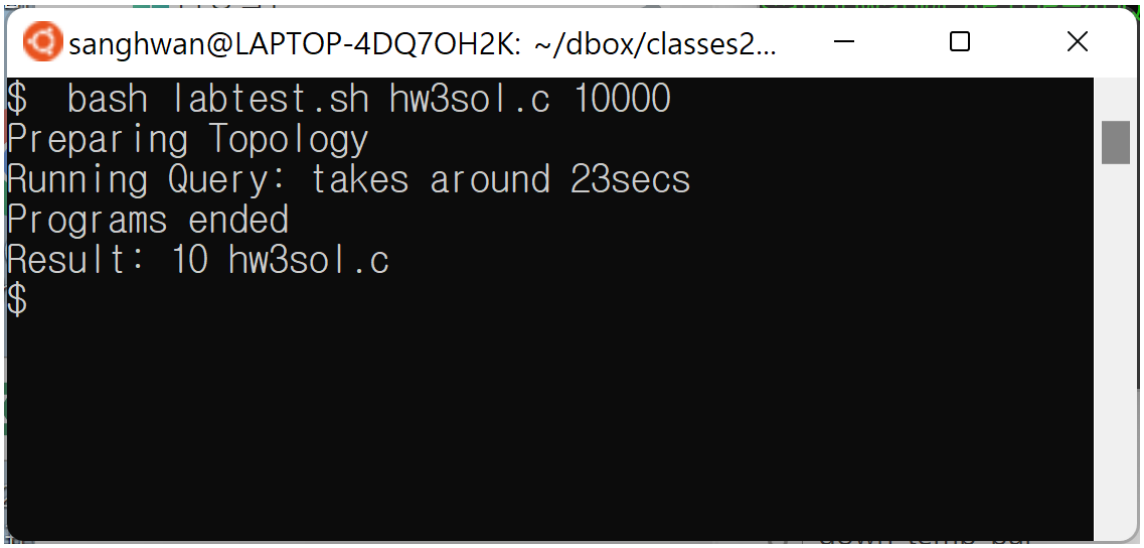
그리고 구현한 파일 확장자에 따라 다음 명령 중 하나를 실행. 아래 그림 참조.

$ bash labtest.sh hw3sol.c 10000

$ bash labtest.sh hw3sol.java 10000

$ bash labtest.sh hw3sol.py 10000

$ bash labtest.sh hw3sol.cpp 10000



최종 라인에 “Result: 10”이 나와야 한다.

한 머신에서 테스트하기 때문에 프로세스간의 경쟁에 의해 혹시 9가 나오는 경우도 있으니 다시 한번 실행하면 10이 나올 수 있다.

그러나 8 이하라면 코드에 문제가 있다고 생각하고 좀 더 디버깅을 하기 바람.