**컴퓨터 네트워크**

* Final Assignment -

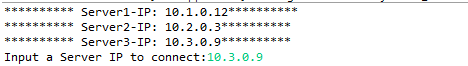
**공과대학교**

**소프트웨어학과**

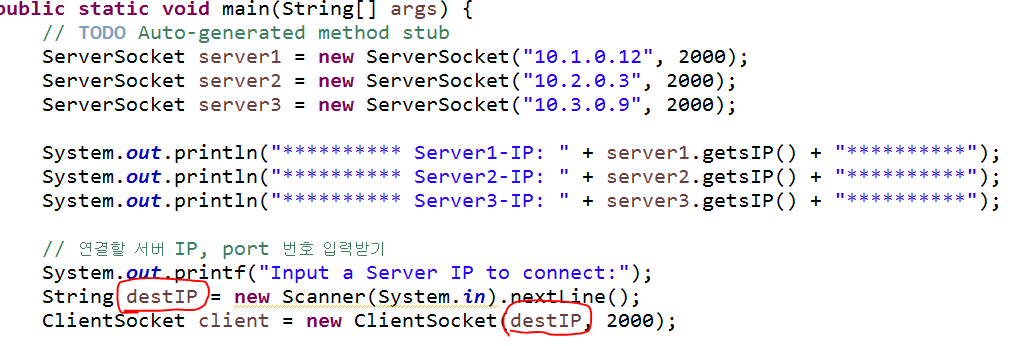
**201611244**

**한상규**

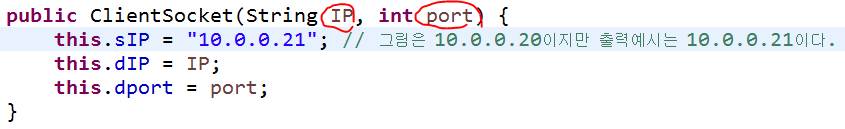
1. **연결할 서버의 IP주소 입력하기**



위와 같은 결과를 출력하기 위해서는

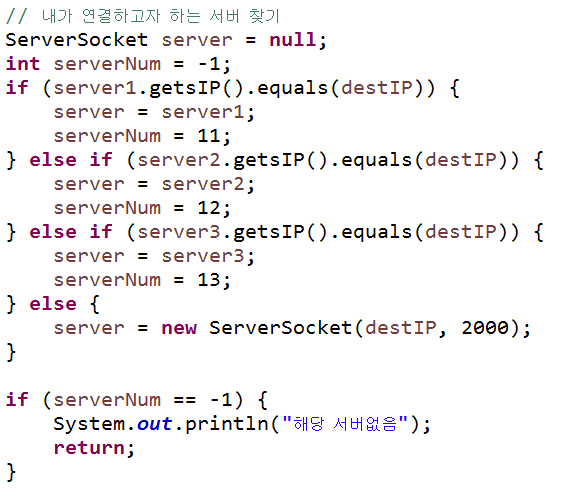


destIP에 연결할 서버의 IP 주소를 입력한다.



이렇게 하면 clientSocket은 목적지의 ip와 port를 초기화 할 수 있다.

1. **IP로 연결할 서버 찾기**

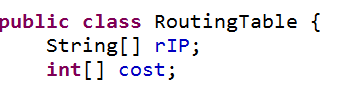


destIP에는 연결할 서버의 IP가 저장돼있다. 생성한 server들을 하나씩 살피면서 destIP와 같은 IP를 갖은 서버를 찾는다. serverNum은 다이젝트라에서 도착지 라우터를 설정해주기 위한 변수이다.

1. **라우터와 라우팅테이블 만들기**

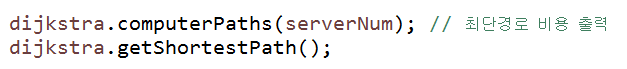
|  |
| --- |
| // 라우터A 만들기  RoutingTable rt = **new** RoutingTable(4);  rt.add(0, "10.0.0.3", 2);  rt.add(1, "10.1.0.1", 5);  rt.add(2, "10.0.0.2", 1);  rt.add(3, client.getsIP(), 0); // 클라이언트 IP  Router A = **new** Router("10.0.0.1", rt);  // 라우터B 만들기  rt = **new** RoutingTable(3);  rt.add(0, "10.0.0.1", 1);  rt.add(1, "10.1.0.1", 5);  rt.add(2, "10.2.0.1", 10);  Router B = **new** Router("10.0.0.2", rt);  // 라우터C 만들기  rt = **new** RoutingTable(2);  rt.add(0, "10.0.0.1", 2);  rt.add(1, "10.1.0.2", 5);  Router C = **new** Router("10.0.0.3", rt);  // 라우터D 만들기  rt = **new** RoutingTable(3);  rt.add(0, "10.0.0.1", 5);  rt.add(1, "10.1.0.2", 5);  rt.add(2, "10.0.0.2", 1);  Router D = **new** Router("10.1.0.1", rt);  // 라우터E 만들기  rt = **new** RoutingTable(4);  rt.add(0, "10.0.0.3", 5);  rt.add(1, "10.1.0.1", 1);  rt.add(2, "10.1.1.2", 3);  rt.add(3, server1.getsIP(), 0); // 서버 IP  Router E = **new** Router("10.1.0.2", rt);  // 라우터F 만들기  rt = **new** RoutingTable(3);  rt.add(0, "10.1.0.2", 3);  rt.add(1, "10.2.0.3", 5);  rt.add(2, "10.3.0.1", 8);  Router F = **new** Router("10.1.1.2", rt);  // 라우터G 만들기  rt = **new** RoutingTable(4);  rt.add(0, "10.0.0.2", 10);  rt.add(1, "10.2.0.3", 2);  rt.add(2, "10.3.1.1", 5);  rt.add(3, server2.getsIP(), 0); // 서버 IP  Router G = **new** Router("10.2.0.1", rt);  // 라우터H 만들기  rt = **new** RoutingTable(2);  rt.add(0, "10.1.1.2", 5);  rt.add(1, "10.2.0.1", 2);  Router H = **new** Router("10.2.0.3", rt);  // 라우터I 만들기  rt = **new** RoutingTable(3);  rt.add(0, "10.1.1.2", 8);  rt.add(1, "10.3.1.1", 3);  rt.add(2, server3.getsIP(), 0); // 서버 IP  Router I = **new** Router("10.3.0.1", rt);  // 라우터J 만들기  rt = **new** RoutingTable(2);  rt.add(0, "10.2.0.1", 5);  rt.add(1, "10.3.0.1", 3);  Router J = **new** Router("10.3.1.1", rt);  // 경로비용에 대한 정보 넣어주기  Dijkstra dijkstra = **new** Dijkstra(13);  dijkstra.input(1, 2, 1);  dijkstra.input(1, 3, 2);  dijkstra.input(1, 4, 5);  dijkstra.input(2, 4, 5);  dijkstra.input(2, 7, 10);  dijkstra.input(3, 5, 5);  dijkstra.input(4, 5, 1);  dijkstra.input(5, 6, 3);  dijkstra.input(5, 11, 0);  dijkstra.input(6, 8, 5);  dijkstra.input(6, 9, 8);  dijkstra.input(7, 10, 5);  dijkstra.input(7, 12, 0);  dijkstra.input(9, 13, 0);  dijkstra.input(9, 10, 3); |

모든 라우터들은 라우팅테이블을 가지고 있다. 라우팅 테이블에는 이웃 라우터의 IP 주소와 이웃 라우터까지 cost가 저장돼있다.

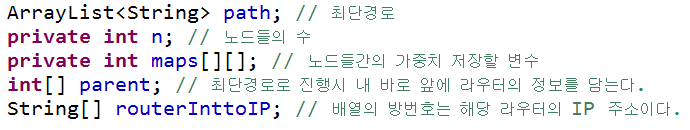


라우팅 테이블은 동적으로 배열을 만들어서 IP와 cost를 저장할 수 있게 했다.

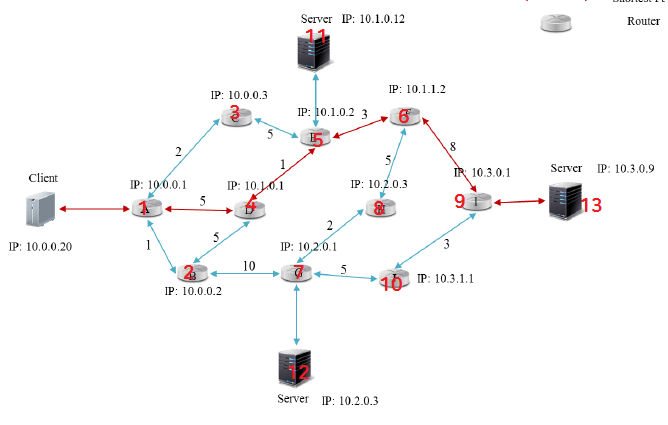
1. **다이젝스트라로 최단 경로 계산**



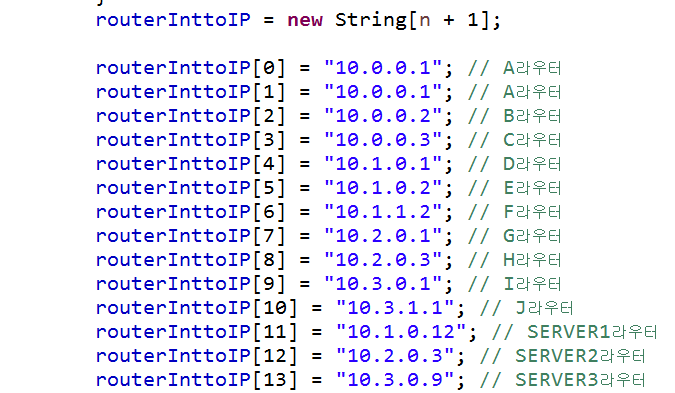
다음은 다이젝트스라 알고리즘을 통해 서버까지 최단 경로를 구해보자



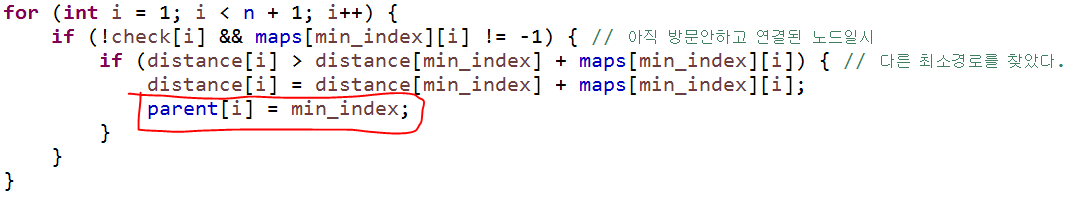
**class** Dijkstra 의 필드들이다. 다이젝트라 알고리즘을 구현하는데 2차원 배열을 사용했다. for문을 돌면서 아직 방문하지 않은 라우터의 최소비용을 계속해서 갱신하였다.

****

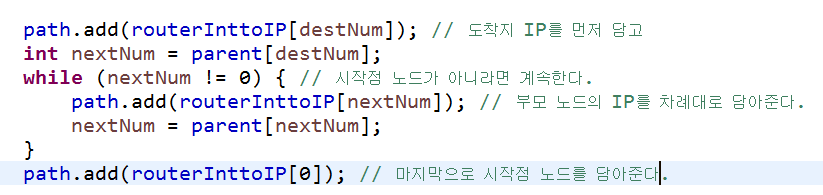
이차원 배열을 사용하기 때문에 라우터마다 번호를 지정해줘야 했다.



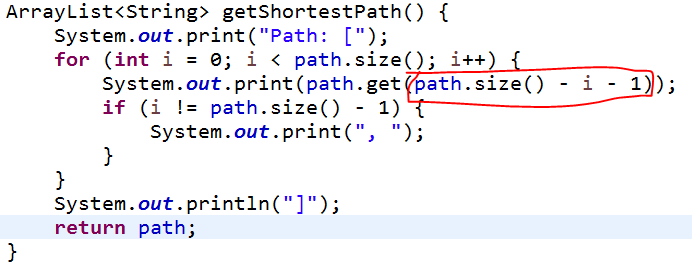
모든 라우터의 번호와 IP주소가 연결되게 따로 배열을 만들어줬다.



최단경로가 갱신될 때마다 parent배열에 이전 노드의 번호를 담아준다. 이렇게 하면 최단경로로 진행했을 때 꼭 거쳐야 하는 라우터를 명시해줄 수 있다.



Path에 destIP에 도달할 때까지 거쳐야할 라우터의 IP를 저장한다.



경로를 출력해줄 때는 담아준 역순으로 출력해준다. 그 이유는 담아줄 때 목적지의 IP부터 반대로 진해하면서 담아줬기 때문이다.

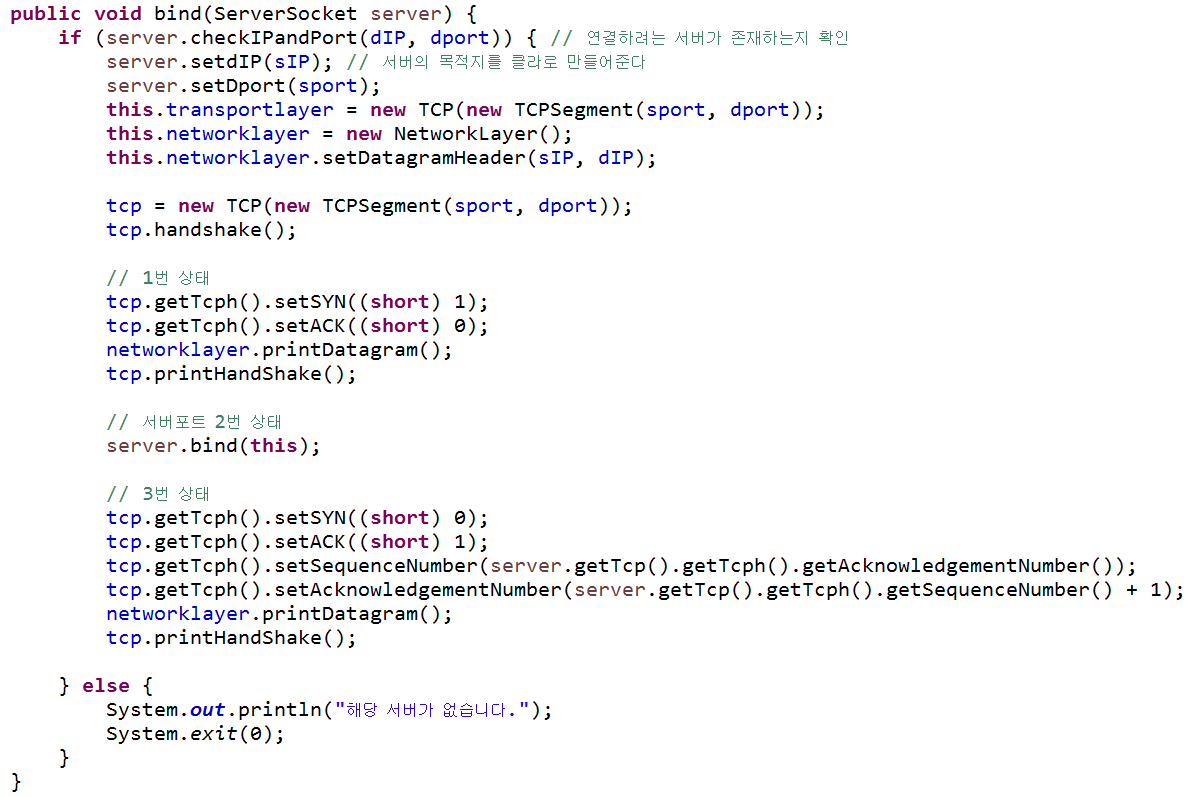


출력결과

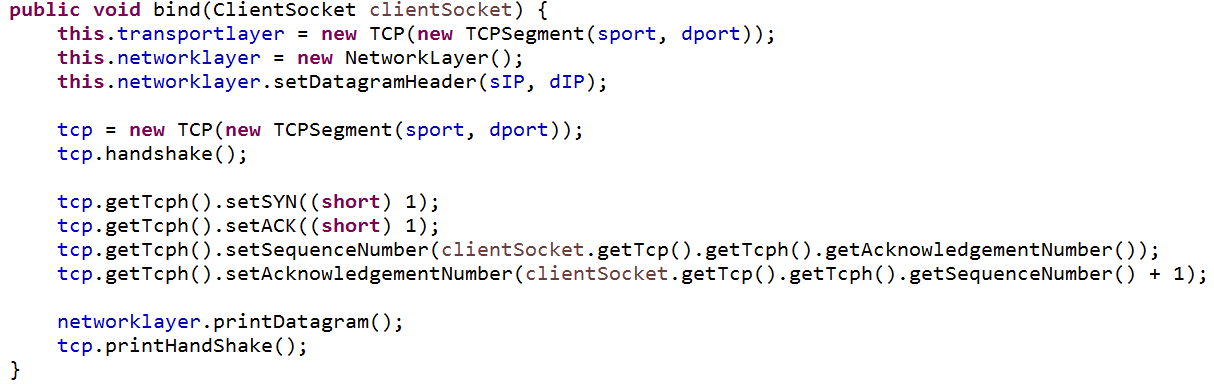
1. **클라이언트와 서버 핸드쉐이크**



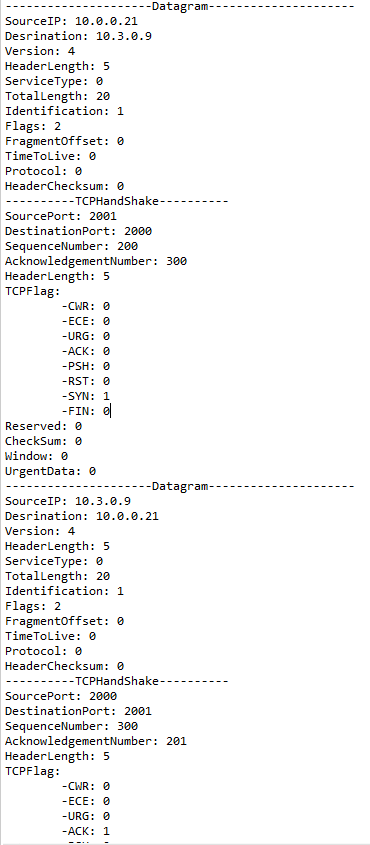
클라이언트와 서버의 핸드쉐이크는 bind()에서 모두 처리해줬다.

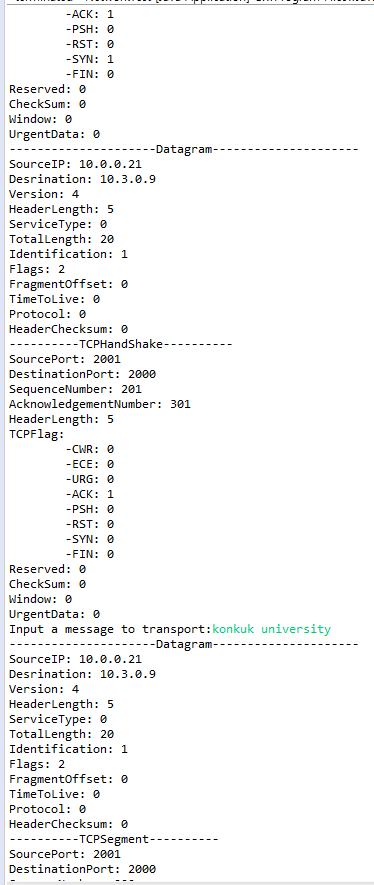


Syn와 ack를 바꿔주면서 datagramheader와 TCPheader 상태 정보를 출력해줬다.



serverSocket에서도 bind()라는 함수를 통해 핸드쉐이크가 처리되게 했다.





1. **데이터그램 만들어서 보내주기**



데이트를 5글자 단위로 잘라서 TCPSegment를 출력해준다.

