**20160521 이준희 CS443 Lab3 Report**

**코드 설명**

먼저 client.go는 기본적으로 Get/Put/Append 명령을 받으면 리더 서버에 call을 보내는 구조이다. Lab 설명 페이지에 test코드의 시간을 줄이기 위해서 leader를 저장해 놓는 것을 추천하여 exLeader를 만들었다. exLeader를 저장하는 방식은 0부터 시작하여 각각의 operation을 보낼 때 무한 루프 안에서 call을 보내고 이 때 reply의 WrongLeader가 true라면 1씩 추가하는 방식으로 리더를 찾으면 operation의 리턴 값을 받고 return하는 식으로 무한 루프를 탈출하게 하였다. 추가로 unreliable 상황일 때의 중복된 call을 알기 위해 client의 id와 각각 call의 고유 번호를 매겨 server에서 검사하고 루프 탈출에 call 성공 여부도 추가하였다. 따라서, common.go안의 struct에 이에 해당하는 Xid와 ReqNum이 추가되었다.

Server.go는 작동 방식이 client에서 call을 받으면 해당 operation을 raft.Start()로 consensus를 시작하고 끝을 채널 통해 캐치하여 operation에 맞게 key/value DB를 수정하고 이를 다시 채널을 통해 알리는 방식으로 했다. 이 과정에서 KVServer struct에 key/value DB에 해당하는 kvDB, log index를 이용한 각각의 call에 대한 커밋이 완료됨을 알리는 채널인 chanCommits, 중복 검사를 위한 xidDB가 추가 되었다. 이후에 현재 내가 call 보냈던 leader가 죽어 start에 넣었던 operation과 consensus를 통해 commit된 operation이 일치하지 않는 경우가 발생함을 알게 되어 chanCommits를 map[int]chan bool에서 map[int]chan Op으로 변경하였다.