Assignment 1

MG1833074
王易木

1. 如何达到目标

Raft 算法主要是 Follower、Candidate 和 Leader 之间的转换，以及其中的通信。

在具体实现中，利用 go 语言的并发性和管道特性，使得角色转换和心跳发送可以同时进行，完成了这一功能。

2. 分析与设计

首先定义好 raft。

```
type Raft struct {
    mu        sync.Mutex
    peers     []*labrpc.ClientEnd
    persister *Persister
    me        int // index into peers[]

    // Your data here.
    // Look at the paper's Figure 2 for a description of what
    // State a Raft server must maintain.
    State           string

    CurrentTerm     int
    VotedFor        int
    log []LogEntry

    CommitIndex int
    LastApplied int

    NextIndex []int
    MatchIndex []int

    lastMessageTime int64//判断什么时候开始去看能不能竞选
    startHeatBeat       chan bool//是否开始发送心跳
}

type LogEntry struct {
    Command interface{}
    Term int
}
```

其中 lastMessageTIme 和 startHeatBeat 是两个自己定义的变量。

以及三个类型的转换：

```go
func (rf *Raft) becomeCandidate() {
    rf.State = "Candidate"
    rf.CurrentTerm = rf.CurrentTerm + 1
    rf.VotedFor = rf.me
}

func (rf *Raft) becomeLeader() {
    rf.State = "Leader"
}

func (rf *Raft) becomeFollower(term int, candidate int) {
    rf.State = "Follower"
    rf.CurrentTerm = term
    rf.VotedFor = candidate
    rf.lastMessageTime = getPresentMileTime()
}
```

在 make 函数重定义好整体。

```go
func Make(peers []*labrpc.ClientEnd, me int,
    persister *Persister, applyCh chan ApplyMsg) *Raft {
    rf := &Raft{}
    rf.peers = peers
    rf.persister = persister
    rf.me = me

    // Your initialization code here.
    rf.CurrentTerm = 0
    rf.VotedFor = -1
    rf.State = "Follower"
    rf.startHeatBeat = make(chan bool)

    go rf.election()
    go rf.sendLeaderHeartBeat()

    // initialize from State persisted before a crash
    rf.readPersist(persister.ReadRaftState())


    return rf
}
```

使用 GO 语法同时运行 election 和 sendleaderheartbeat 两个函数。分别进行选举和发送心跳。

首先是简单的发送心跳。当且仅当是 leader 的时候才可以发送心跳。于是判定条件为管道接收到这个信号。

```go
func (rf *Raft) sendLeaderHeartBeat() {
    timeout := 20
    for {
        select {
        case <-rf.startHeatBeat:
            rf.sendAppendEntriesImpl()
        case <-time.After(time.Duration(timeout) * time.Millisecond):
            rf.sendAppendEntriesImpl()
        }
    }
}
```

具体的实现部分就比较简单，一个个发送，如果没有超市并且接收到回复为成功的话那么就计数，当数量超过一半的服务器接收到即视为发送成功。

```go
func (rf *Raft) sendHeartBeat() int {
    timeout := 20
    var success_count int
    success_count = 0
    // 发送
    for i := 0; i < len(rf.peers); i++ {
        if i != rf.me {
            c := make(chan bool, 1)

            var reply AppendEntriesReply
            var args AppendEntriesArgs
            args.LeaderId = rf.me
            args.Term = rf.CurrentTerm
            go func() { c <- rf.peers[i].Call("Raft.AppendEntries", args, &reply) }()
            select {
            case ok := <- c:
                if ok && reply.Success {
                    success_count++
                }
            case <-time.After(time.Duration(timeout) * time.Millisecond):
                continue
            }
        }
    }
    return success_count
}

// 发送心跳
func (rf *Raft) sendAppendEntriesImpl() {
    //如果我是leader
    if rf.State == "Leader" {
        // 发送
        c := make(chan int, 1)
        go func() { c <- rf.sendHeartBeat() }()
        select {
        case count := <- c:
                // 如果发送成功 一半以上收到
            if count >= len(rf.peers)/2 {
                rf.mu.Lock()
                rf.lastMessageTime = getPresentMileTime()
                rf.mu.Unlock()
            }
        }
    }
}
```

选举部分比较复杂。
首先判断自己是否可以去选举（超时并且没人去选举）。

```go
func (rf *Raft) election() {
    var result bool
    for {
        timeout := rand.Int63n(100) + 50
        rf.lastMessageTime = getPresentMileTime()
        for rf.lastMessageTime+timeout > getPresentMileTime() {
            select {
            // 超时了开始判定
            case <-time.After(time.Duration(timeout) * time.Millisecond):
            //如果没人竞选
                if rf.lastMessageTime+timeout <= getPresentMileTime() {
                    break
                } else {
                    //别人去竞选了
                    rf.lastMessageTime = getPresentMileTime()
                    timeout = rand.Int63n(100) + 50
                    continue
                }
            }
        }

        // 竞选到成功
        result = false
        for !result {
            result = rf.election_one_round()
        }
    }
}
```

然后开始进行拉票。拉票的算法和发送心跳类似，当对方在规定时间内（没有超时）的情况下回复了给我投票的时候，计数。当计数超过一半成功竞选为 leader。

```go
func (rf *Raft) election_one_round() bool {
    var timeout int64
    timeout = 100
    last := getPresentMileTime()
    success := false
    rf.mu.Lock()
    rf.becomeCandidate()
    rf.mu.Unlock()
    done := 0
    for {
        // 找大家求选票
        c := make(chan int, 1)
        go func() { c <- rf.sendRequestVoteAndCount() }()
        select {
        case ok := <- c:
                // 如果发送成功 一半以上收到
            done = ok
            success = (done >= len(rf.peers)/2) && (rf.VotedFor == rf.me)
            if success {
                rf.mu.Lock()
                rf.becomeLeader()
                rf.mu.Unlock()
                rf.startHeatBeat <- true
            }
        }
        if (timeout+last < getPresentMileTime()) || (done >= len(rf.peers)/2) {
            return success
        }
    }
}
```

Definition:
raft.go:75

```go
// 选举部分
func (rf *Raft) sendRequestVoteAndCount() int {
    timeout := 20
    var done int
    done = 0
    for i := 0; i < len(rf.peers); i++ {
        if i != rf.me {
            var args RequestVoteArgs
            args.Term = rf.CurrentTerm
            args.CandidateId = rf.me
            var reply RequestVoteReply

            c := make(chan bool, 1)
            go func(){c <- rf.sendRequestVote(i, args, &reply)}()
            select {
            case <- c:
                if reply.VoteGranted {
                    done++
                }
            case <-time.After(time.Duration(timeout) * time.Millisecond):
                continue
            }
        }
    }
    return done
}
```

主要的函数便是这样。

3. 实现演示

进入主目录然后加入到 gopath 后便可直接运行。

```
[NJU-DisSys-2018 ) export GOPATH=$PWD
[NJU-DisSys-2018 ) cd src/raft/
[raft ) go test -run Election
Test: initial election ...
  ... Passed
Test: election after network failure ...
  ... Passed
PASS
ok      raft    7.011s
```

4. 总结

熟悉了 Raft 算法中三个角色的转换，以及 GO 语言中的一些语法运用。
Go 语言的 Channel 像管道一样使用起来方便，结合 select 可以直接设置定时/响应函数。