题目：RAFT 算法作业说明

姓名：罗晨刚

学号：22151056

作业说明

1.1 Raft 算法简介

Raft 是一种用于管理复制日志的共识算法。在 Raft 算法中，每个节点都处于三种状态中的一种：Follower、Candidate 和 Leader。节点启动时状态为 Follower。如果 Follower 没有收到 Leader 或者 Candidate 的消息，那么他们可以成为 Candidate。

一个服务器节点继续保持着 Follower 状态只要他从 Leader 或者 Candidate 处接收到有效的 RPCs。

要开始选举，Follower 会增加其当前任期并转换到候选状态。然后它为自己投票并向集群中的每个其他服务器并行发出 RequestVote RPC。每个服务器将在给定任期内最多投票给某一个 Candidate，先到先得。

Raft 保证从选举的那一刻起，每个新 Leader 上都存在之前任期内所有已提交的条目，而无需将这些条目转移给 Leader。这意味着日志条目只在一个方向流动，从 Leader 到 Follower，Leader 永远不会覆盖他们日志中的现有条目。

如果服务器出现故障但后来又恢复了，Raft 会负责更新其日志。只要至少大多数服务器还活着并且可以相互通信，Raft 算法就可以继续运行。如果没有这样的多数，Raft 算法将不会取得任何进展，但会在多数服务器可以再次通信时从中断的地方继续达成共识。

1.2 目标

本作业的目标是实现 Raft 算法。本文参考了 2020 年 MIT 6.824 分布式系统课程的 Lab2 要实现的算法。用 Go 语言实现，代码工作量在/src/raft/raft.go 文件中，这里 Raft 算法实现的主要功能包括：

1）leader 选举

2）日志复制

3）保存持久状态，在节点故障后,重新启动的时候读取它

leader 选举部分要实现的主要功能包括选举单个 leader 和心跳机制。选举单

个 leader 后，如果 leader 没有发生故障，leader 仍然保持 leader 身份。

日志复制部分需要实现 leader 通过 AppendEntries RPC 向 follower 附加新的日志条目。

如果 Raft 服务器重新启动，它应该从中断的地方恢复服务。这里实现的 Raft 算法可以应对服务器故障和网络丢失 RPC 请求或回复的问题。

1.3 测试方法和结果

1)在 linux 命令行运行命令

```
go env -w GO111MODULE=off
```

2)在 linux 命令行运行

```
cd src/raft
go test
```

3) 运行结果如下图：

```
ubuntu@VM-0-13-ubuntu:~/2020/6.824/src$ cd raft
ubuntu@VM-0-13-ubuntu:~/2020/6.824/src/raft$ go test
warning: only one CPU, which may conceal locking bugs
Test (2A): initial election ...
  ... Passed --   3.1  3   60   17948    0
Test (2A): election after network failure ...
  ... Passed --   4.5  3  132   25714    0
Test (2B): basic agreement ...
  ... Passed --   0.5  3   16    4706    3
Test (2B): RPC byte count ...
  ... Passed --   1.3  3   48  114958   11
Test (2B): agreement despite follower disconnection ...
  ... Passed --   5.4  3  132   36843    8
Test (2B): no agreement if too many followers disconnect ...
  ... Passed --   3.4  5  240   48900    3
Test (2B): concurrent Start()s ...
  ... Passed --   0.5  3   22    6781    6
Test (2B): rejoin of partitioned leader ...
  ... Passed --   3.8  3  146   34347    4
Test (2B): leader backs up quickly over incorrect follower logs ...
  ... Passed --  16.7  5 2188 1576597  103
Test (2B): RPC counts aren't too high ...
  ... Passed --   2.2  3   66   22336   12
Test (2C): basic persistence ...
  ... Passed --   3.1  3   76   21106    6
Test (2C): more persistence ...
  ... Passed --  18.8  5 1188  245608   16
Test (2C): partitioned leader and one follower crash, leader restarts ...
  ... Passed --   1.7  3   36    9608    4
Test (2C): Figure 8 ...
  ... Passed --  26.9  5 1104  247186   63
Test (2C): unreliable agreement ...
  ... Passed --   5.4  5 1200  407467  246
Test (2C): Figure 8 (unreliable) ...
  ... Passed --  35.2  5 11228 60153098  171
Test (2C): churn ...
  ... Passed --  16.1  5 8072 16870611 1863
Test (2C): unreliable churn ...
  ... Passed --  16.5  5 1768 1604834  129
PASS
ok      _/home/ubuntu/2020/6.824/src/raft       165.150s
ubuntu@VM-0-13-ubuntu:~/2020/6.824/src/raft$
```