课程报告

学号：MF1733071 姓名：严德美

一、 背景简述

许多分布式系统是基于进程间的显式消息交换的，然而，消息的发送(send)和接收(receive)过程无法隐藏通信的存在，而通信的隐藏对于在分布式系统中实现访问透明性是即为重要的。远程过程调用能有效实现这一功能。

二、 系统分析与设计

首先定义 server，定义生成时间方法 ShowTime，提供客户端调用的方法，定义服务标识，生成验证信息，验证请求调用的客户端,server 监听 8080 端口,server 注册 ShowTime 方法并处理 HTTP 请求，使用 tcp 网络传输协议，紧接着定义 client 使用 tcp 网络传输协议，请求 server 调用生成时间的方法，返回给客户端 server 端的时间，计算传输过程中的时延 elapsed，除以 2 大概估计了传输过程中的所用时延，加上从 server 得到的时间最后展示出来，大概和 server 端的时间同步。

三、 实现演示

server 代码:

```
package main
import (
	"net/rpc"
	"time"
	"net"
	"log"
	"net/http"
	"flag"
	"fmt"
	"errors"
)
type DispalyTime int     //定义服务标识
var password string = "root"
func (c *DispalyTime)ShowTime(key *string, destTime *time.Time) error {//生成时间并返回的
方法
	if password == *key{
		*destTime = time.Now()
	}else{
		return errors.New("key error   ! ")//
	}
	return nil
}

func main(){
	port := flag.String("port","8080","Port")
	displayTime := new(DispalyTime)
	rpc.Register(displayTime)           //注册生成时间方法
	rpc.HandleHTTP()         //处理 HTTP 请求
	flag.Parse()
	listener, err := net.Listen("tcp",":"+*port)          //监听 8080 端口

	if err != nil{
		log.Fatal("listen err",err)
	}
	//输出本机 ip
	addrs, err := net.InterfaceAddrs()
	if err !=nil{
		log.Fatal("ip err",err)
	}
	go    http.Serve(listener,nil)         //并发
	for _,address := range addrs{
		if ipnet, ok := address.(*net.IPNet); ok && !ipnet.IP.IsLoopback() {
			if ipnet.IP.To4() != nil {
				for ;true;{
```

```
                        fmt.Println("listen ",ipnet.IP.String(),listener.Addr().String()[4:9])
                    }

                }
            }
        }
        fmt.Print("------------Server Run--------------")
}
```

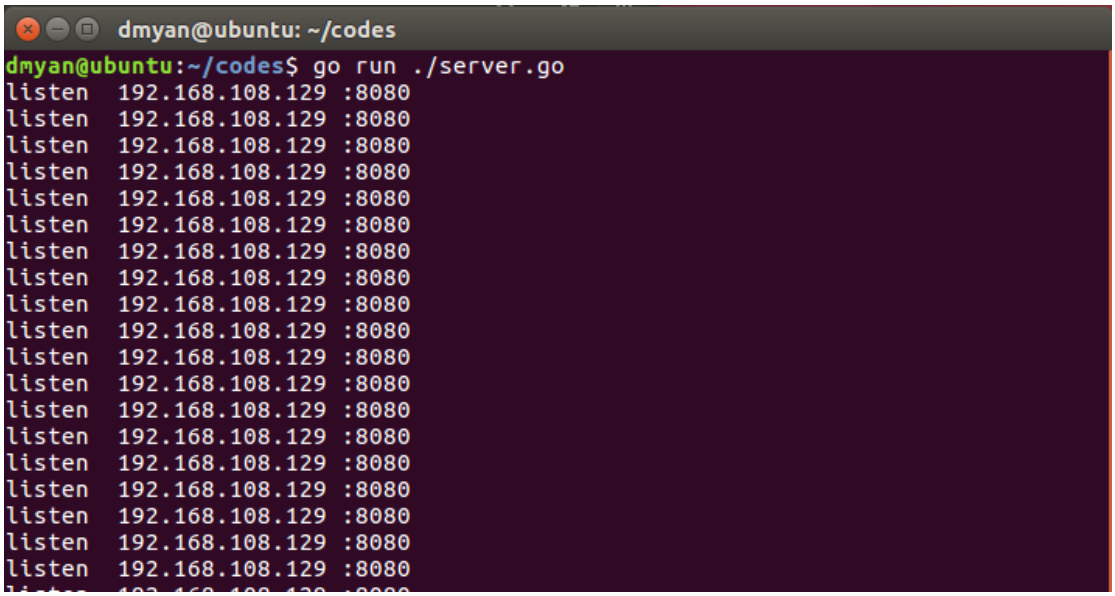
client 代码:

```
package main
import(
        "net/rpc"
        "flag"
        "log"
        "time"
        "fmt"
)
func main(){
        ip := flag.String("ip","192.168.108.129","IP Adress")
        port := flag.String("port","8080","Port")
        flag.Parse()
        client, err := rpc.DialHTTP("tcp",*ip+":"+*port)          //定义 client
        if err !=nil{
                log.Fatal("dialing error:",err)
        }
        var key string ="root"
        var displayTime time.Time
        time1 := time.Now()
        err = client.Call("DispalyTime.ShowTime",&key,&displayTime)              //远程过程调用
        time2 := time.Now()

        if err != nil{
                log.Fatal("showTime error:",err)
        }
        datetime :=displayTime.Format("2006-01-02 15:04:05")
        fmt.Println(displayTime)
        fmt.Println(displayTime.Add(time2.Sub(time1)/2)," elapsed =",time2.Sub(time1)/2)
        //计算时间差
        fmt.Println(datetime)
}
```

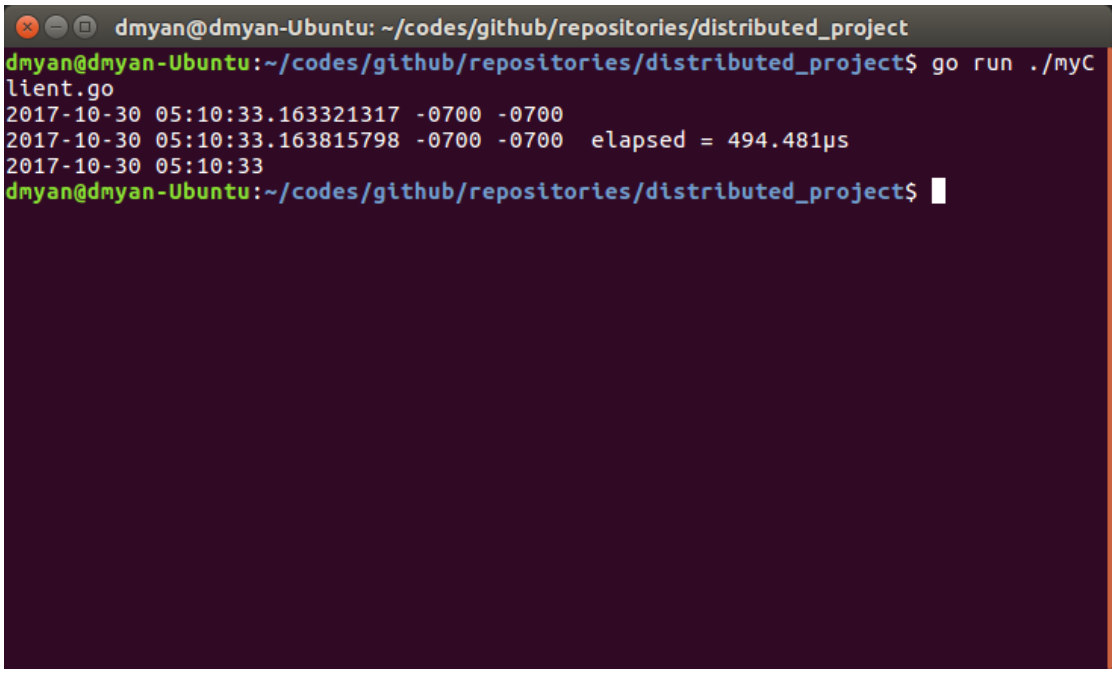
server 运行在一台 linux 虚拟终端上，两个 client 分别运行在一台虚拟 linux 虚拟终端和一台 win7 终端上。

server:



监听 8080 端口

linux client:



显示了 server 端的时间

windows client:

key 错误,server 验证不通过。

四、 总结

通过本次实验对远程过程调用(rpc)有了一个初步的了解，理解了分布式系统中进程间的消息交换，对进一步的理解分布式系统大有益处，使用 go 语言提供的 net/rpc 包来编写 rpc 程序方便了很多，go 语言以并发程序设计著称，动手编写了第一个 go 语言的并发程序。