南开大学
Nankai University

南　开　大　学

计 算 机 学 院

计算机网络实验报告

实验一 ———— 简单客户端服务器实现

学号：1712991

姓名：周辰霏

年级：2017 级

专业：计算机科学与技术

2019 年 10 月 27 日

摘要

借助 QT 提供的 QUdpSocket 实现简单的客户端服务器程序实现双向请求和响应等, 辅以提示信息。在实现的过程中更深入的体会 UDP 协议的特性以及应用的方法。

关键字： UDP 协议；客户端；服务器；QT

目录

一、 实验目的

- 利用 UDP 协议进行 Socket(套接字) 编程
- 服务器可以根据客户端发送的”date“、”time“请求, 分别使用本地日期和时间进行响应。

二、 实验环境

- 系统环境:MacOS 10.15
- IDE:Qt Creator 4.9
- 编程语言:C++

三、 UDP 工作流程 & 程序结构

(一) 工作流程

如下图为 UDP 服务器及客户端的简单工作流程, 首先创建套接字 Socket, 之后实例化初始化, 借助 Socket 发送或接收信息, 最后还要记得关闭 Socket

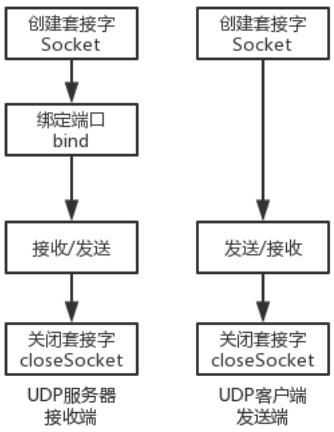

图 1: UDP 工作流程

(二) 程序结构

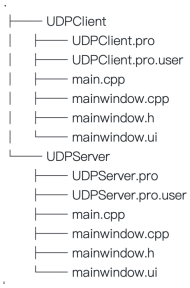

图 2: 代码结构

ui 文件包含了程序的 GUI 界面;pro 文件描述了整个 QT 程序, 在其中加入 network 组件以便处理网络编程;main.cpp 中将 mainwindow 呈现出来;mainwindow.h 和 mainwindow.cpp 包括了本次实验的核心代码, 实现了可以双向通信的服务器以及客户端。

1. **QUdpSocket** 创建与初始化 在 mainwindow.h 以及 mainwindow.cpp 中

Initial

```
#include <QUdpSocket>                    //mainwindow.h
#include <QHostAddress>
QHostAddress ip;//目标host
int port;//目标port
QUdpSocket *udpSocket;//创建UdpSocket对象
udpSocket = new QUdpSocket(this);              //mainwindow.cpp初始化
udpSocket->bind(port);                         //将Socket绑定到指定IP与端口
connect(udpSocket,SIGNAL(readyRead()),this,SLOT(messageGet()));
//将udpsocket接收到可以读取data的信号同自定义获取message的slot绑定在一起
```

2. 接收并处理请求 在 Client 和 Server 中通过 messageGet 和 dataRecieved 的 SLOT 来响应接收到的可以 Read 的数据报, 这其中使用了 readdatagram 接收请求

GetMessage——Client

```
void MainWindow::messageGet()                  //Client
{//只要有一个等待被read的数据报就进行GetMessage
    while(udpSocket->hasPendingDatagrams())
    {
    QByteArray datagram;            //二进制字节数组数据报
    QHostAddress ip;//ip
    quint16 port;//端口号
    //格式调整为待解析的数据报格式
    datagram.resize(udpSocket->pendingDatagramSize());
    //通过readDatagram函数解析接收的数据报,获取(请求内容),数据包大小,ip及端口
    udpSocket->readDatagram(datagram.data(), datagram.size(),&ip,&port);
    QString msg = datagram.data();     //转换为字符串便于显示在屏幕上
    QString str=QString::fromLocal8Bit(datagram);//解决中文乱码问题
    ui->lineEdit_4->setText(str);
    }
}
```

GetMessage——Server

```
void MainWindow::dataReceived()          //Server
{
    while(udpSocket->hasPendingDatagrams())
    {
        QByteArray datagram;
        QHostAddress ip;//ip
        quint16 port;//端口号
        QDateTime datetime=QDateTime::currentDateTime();//获取当前时间
        datagram.resize(udpSocket->pendingDatagramSize());
```

```
10
11          udpSocket->readDatagram(datagram.data(), datagram.size(),&ip,&port);
12          QString msg = datagram.data();
13          if(msg.toUpper()=="DATE") // 请求为 date
14          {// 在日志 textedit 中插入日志信息
15              ui->textEdit->insertPlainText\
16              (datetime.toString("yy/MM/dd hh:mm:ss:")+"收到IP="+\
17              ip.toString().mid(7,ip.toString().size())+":"+\
18              QString::number(port)+"请求【"+msg+"】, 响应【"+\
19              datetime.toString("yy/MM/dd")+"】\n");
20              // 利用 sendmessage SLOT向客户端返回响应内容
21              this->sendMessage(datetime.toString("yy/MM/dd"),ip,port);
22          }
23          else if(msg.toUpper()=="TIME") // 请求为 time
24          {
25              ui->textEdit->insertPlainText\
26              (datetime.toString("yy/MM/dd hh:mm:ss:")+"收到IP="+\
27              ip.toString().mid(7,ip.toString().size())+":"+\
28              QString::number(port)+"请求【"+msg+"】, 响应【"+\
29              datetime.toString("hh:mm:ss")+"】\n");
30              this->sendMessage(datetime.toString("hh:mm:ss"),ip,port);
31          }
32          else //错误请求
33          {
34              ui->textEdit->insertPlainText\
35              (datetime.toString("yy/MM/dd hh:mm:ss:")+"收到IP="+\
36              ip.toString().mid(7,ip.toString().size())+":"+\
37              QString::number(port)+"请求【"+msg+"】, 响应【错误请求】\n");
38              this->sendMessage("错误请求                ",ip,port);
39          }
40      }
41  }
```

3. 发送请求 在 Client 和 Server 中通过 messageSend 的 SLOT 来发送数据报, 这其中使用了 writedatagram 发送请求

SendMessage——Server

```
1  void MainWindow::sendMessage(QString msg,QHostAddress ip,int port)
2  {// 不使用 tolatin1 是为了解决中文乱码
3      udpSocket->writeDatagram(msg.toLocal8Bit(),msg.length(),ip,port);
4  }
```

SendMessage——Client

```
1  void MainWindow::messageSend()
2  {
3      QString strIPAddress = ip.toString();
4      QString msg = ui->lineEdit_2->text(); // 获取输入的 ip
```

```
    int length = 0;
    if( msg == "")
    {
        return;
    }
    if(strIPAddress=="255.255.255.0")
    {//如果datagram转换出错不执行
     //255.255.255.0可以开启broadcast连接询问是否加入网络,通过网络访问服务器
        if((length = udpSocket->writeDatagram(msg.toLatin1(), \
        msg.length(), QHostAddress::Broadcast,port )) != msg.length())
       {
            return;
       }
    }
    //直接访问
    else if((length = udpSocket->writeDatagram(msg.toLatin1(), \
    msg.length(), ip,port )) != msg.length())
    {
        return;
    }
    else return;
}
```

4. 其它 之后再进行一些程序的完善即可,如在服务器端加入连接失败的错误提示框、客户端输入 IP 地址错误的提示框、一开始服务器默认绑定的端口、按下客户端" 发送 "按钮以后的动作响应等即完成了该实验。

四、 程序演示

- 工作界面: 实现了实验要求的全部界面以及功能

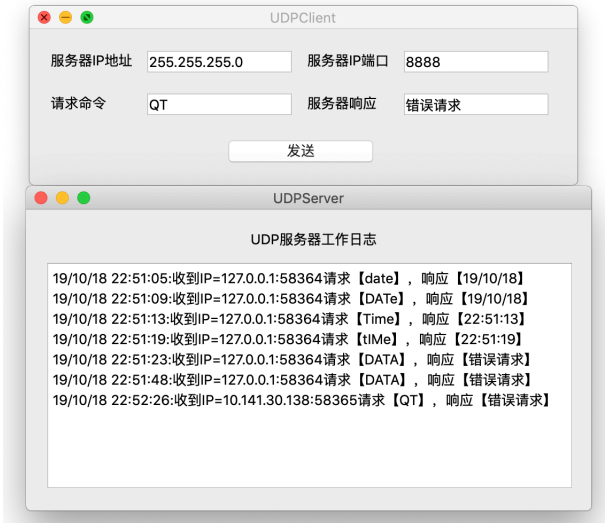

图 3: 工作界面

- 服务器连接失败报错提示: 增加了服务器连接不成功,Socket 创建失败的提示

图 4: Server 报错

- 客户端 IP 输入错误报错: 增加了 IP 输入错误的提示

图 5: Client 报错

五、 实验遇到问题及解决

- 前期没有加入服务器端 Socket 是否创建连接成功提示的时候经常没有响应, 导致 debug 的时候不清楚是 datagram 传输的问题还是 socket 连接的问题, 之后通过加入提示解决了这一问题
- 一开始的 IP 输入“127.0.0.1”是没有响应的, 但是测试的时候在 Client 中直接使用 QHostAdress::Localhost 连接却是没有任何问题的, 在将 QHostAdress::Localhost 转成 QString 格式与输入对比后将输入的空格去掉解决这一问题
- 服务器返回“错误请求”的时候由于是中文所以会出现乱码问题, 使用网上推荐较多的 textcodec 的编码格式转换 UTF-8 没有任何效果, 最终在发送数据报的 writedatagram 时, 使用 $msg.toLocal8Bit$ 而不是简单的 $msg.toLatin1$ 解决这一问题, 但是由于长度限制, 所以返回的错误请求后面追加了很多空格, 防止只显示“错”字

六、 结论

这是我第一次将学到的网络知识的微末应用于实践实验编程, 从一开始的 Socket 流程都是极为陌生的, 在许多次的尝试和失败并更正后, 终于通过 QT 封装好的 QUdpSocket 实现了一个简单的服务器和客户端程序, 他们之间可以完成简单的请求和响应工作, 这之中还牵扯到了中文转码以及 qdebug 的一些知识的习得和复习, 收获满满, 加深了对 UDP 协议的理解。这也是在学习完计算机网络应用层后的第一次实践, 虽然过程不乏困难重重, 但是在学习体会上确是比单纯上课的效果要好得多。

最后还有一小点点算是这次作业的建议吧, 这次的作业很多人上一周就已经检查完了, 最早布置作业的时候并没有说明有附加分的事情, 而作业要求直到这一周才发放, 恳请助教学长学姐早些发放实验要求。

此外, 附加分里面的可靠传输机制属于运输层的内容, 可响应传输文件用到的 ftp 协议也没有讲到, 而且后面的作业对于这两项或多或少都会涉及, 其实没有必要留这么一次作业, 尤其是如果早一些知道作业一的附加分要求, 很多人可能一开始作业一和作业三就一起完成了。

以上仅为我个人的意见反馈, 如有不当之处, 还请见谅。