使用 UDP 实现可靠文件传输

1711437 任蕾

目录

1 实验目的

熟悉可靠传输的交互过程以及 FTP 协议的基本知识。

2 实验环境

mac 上使用 qt 编写实现。

3 实验要求

1) 下层使用 UDP 协议（即使用数据包套接字完成本次程序）;

2) 完成客户端和服务器端程序;

3) 实现可靠的文件传输：能可靠下载文件，能同时下载文件。

4 设计思路

4.1 主要设计

以流水线的形式发送数据包，定时发送数据包。

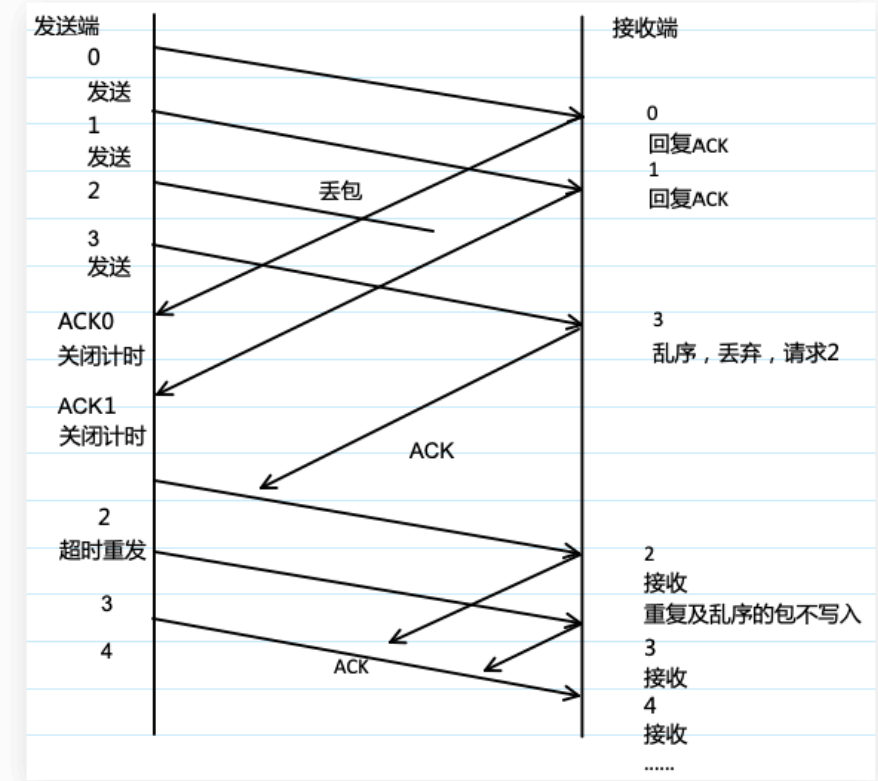
模仿传输层 TCP 的可靠性传输：

1) 添加 seq/ack 机制，确保数据成功发送给对方。因此可以设计发送端发送文件时，将其分为定长数据块（例如每块 512 字节数据），带块号，数据到达接收端后，接收端发回一个对当前收到的块数据的 ack 包，表示已经收到数据;

2) 添加发送和接收缓冲区;

3) 添加超时重传及乱序重传机制。发包后为其设置定时器，在规定时间内收到 ack 确认则停止该报文的计时，否则重传该包数据；若接收到的块号乱序，则接收端向发送端请求重传正确块号以及之后的所有数据，对于发送端送来的乱序包及之后的包和重复的包皆扔掉，但是需向发送端回复 ack 以删除其计时器。

实现形式大致如图：



4.2 各类型数据包格式设计

文件目录请求:

0	通告信息

文件请求回复:

1	文件 id	文件大小

文件下载请求:

2	文件名

文件内容数据:

3	文件 id	块号	数据长度	数据

客户端 ack 回复：

4	文件 id	块号

服务器 ack 回复：

5	-1	op

op 为：0：文件列表请求 ack；1：文件下载请求 ack；2：文件重传请求 ack。

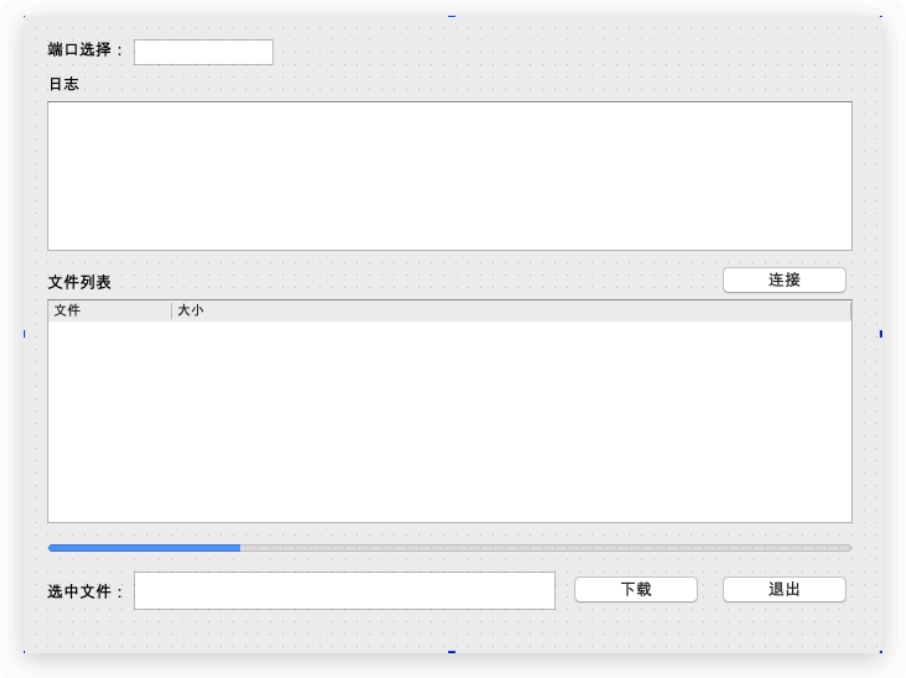
请求重传：

6	文件 id	块号

5 实验内容

5.1 客户端与服务器界面设计

客户端界面，可选择客户端端口，可连接服务器以获取文件列表信息（树形图表示，点击以选中文件），可显示下载日志，可显示下载进度：



服务器界面，可显示下载日志：

5.2 客户端实现

5.2.1 初始化

初始化套接字、定时器、界面等，初始设置数据包发送为广播，端口为服务器默认端口，设置套接字读取以及文件选中的信号槽，定时器因为需要判断相应从属，因此使用匿名函数以传参：

```
//初始化
clientsock = new QUdpSocket(this);
for(int i = 0; i<5; i++)
    curnum[i] = 0;
for(int i = 0; i < 3; i++)
{
    timer[i] = new QTimer;
    timer[i]->setInterval(1000);  //设置时限
    timer[i]->setSingleShot(true);
    timernum[i] = -1;
    connect(timer[i],&QTimer::timeout,this,[=]{resend(i);});
}
curfile = 0;
newt = 0;
filesize = 0;
step = 0;
curstep = 0;
severip = QHostAddress::Broadcast;
severpt = 11005;  //默认服务器端口
ui->logshow->append("****** 客户端就绪 ******");
ui->progressBar->setRange(0, 100);
ui->progressBar->setValue(1);
connect(clientsock,SIGNAL(readyRead()),this,SLOT(getdata()));  //读取数据
connect(ui->filelist,SIGNAL(itemClicked(QTreeWidgetItem*,int)),this,
        SLOT(checkself(QTreeWidgetItem*)));
```

5.2.2 发送请求

1) 文件信息请求：

设置客户端端口号，点击连接按钮，发送文件信息请求数据包并设置计时器：

```
//连接，请求获取文件目录
void Dialog::on_connect_clicked()  //发送自己的端口好并获取文件目录请求
{
    QString port = ui->portselect->text();
    if(port.isEmpty())
    {
        QMessageBox::warning(this, tr("警告"), tr("请选择客户端端口！"));
        return ;
    }
    pt = port.toUShort();
    //绑定到端口，需与服务器端口一致才能保证接收到数据报，ShareAddress为绑定模式
    clientsock->bind(pt,QUdpSocket::ShareAddress);
    QByteArray data = "0\r\nhello server. ";
    //四个参数分别是数据报的内容，数据报的大小，主机地址和端口号
    clientsock->writeDatagram(data.data(),data.size(),QHostAddress::Broadcast,severpt);  //广播
    ui->logshow->append("请求文件列表。");
    timer[0]->start();  //开始计时
}
```

2) 请求下载文件：

获取选中文件的文件信息，设置进度条步长，发送文件下载请求数据包并设置相应定时器：

```
void Dialog::checkself(QTreeWidgetItem *item)
{
    ui->fileselect->clear();  //清空文本
    QStringList filepath;
    QStringList fs;
    QTreeWidgetItem *itemfile = item;  //获取点击选择的文件
    while(itemfile != nullptr)
    {
        filepath<<itemfile->text(0);  //获取点击文件
        fs<<itemfile->text(1);
        itemfile = itemfile->parent();  //父节点
    }
    //其实没用上，因为都是一个目录下的文件
    QString path;
    for(int i = (filepath.size()-1); i >= 0; i--)
    {
        path += filepath.at(i);
        if(i != 0)
            path += "/";
    }
    filesize = fs.at(0).toInt();
    ui->fileselect->insert(path);  //显示选择的文件
}
```

```
//开始下载
void Dialog::on_download_clicked()  //下载文件请求
{
    for(int i = 0; i < 5; i++)
        curnum[i] = 0;
    curstep = 0;
    int stp = filesize%512;
    step = filesize/512;
    if(stp != 0)
        step += 1;
    if(100%step != 0)
        step = 100/step + 1;
    else
        step = 100/step;
    ui->progressBar->reset();  //重置
    //获取文件信息
    QString tmp = ui->fileselect->text();
    if(tmp.isEmpty())
    {
        QMessageBox::warning(this, tr("警告"), tr("请选择下载文件名！"));
        return ;
    }
    QString qs = "2\r\n" + tmp;
    QByteArray data = qs.toLocal8Bit();
    //四个参数分别是数据报的内容，数据报的大小，主机地址和端口号
    clientsock->writeDatagram(data.data(),data.size(),severip,severpt);
    ui->logshow->append("请求下载文件。");
    timer[1]->start();  //可复用,为区分这里不复用
}
```

如果请求消息超时，则重新请求：

```
void Dialog::resend(int i)
{
    ui->logshow->append("请求超时。");
    if(i == 0)
    {
        ui->logshow->append("重新申请文件列表。");
        QByteArray data = "0\r\nhello server. ";
        clientsock->writeDatagram(data.data(),data.size(),severip,severpt);  //广播
        ui->logshow->append("重新请求文件列表。");
        timer[0]->start();  //开始计时
    }
    else if(i == 1)
        ui->logshow->append("请重新下载。");
    else if(i == 2)
        ui->logshow->append("重传请求失败。");
}
```

5.2.3 接收数据处理

统一接收数据，分类进行处理：

```
void Dialog::getdata()  //接收数据统一处理
{
    while(clientsock->hasPendingDatagrams())  //拥有等待的数据
    {
        QByteArray data;  //用于存放接收的数据
        qint64 size = clientsock->pendingDatagramSize();
        data.resize(size);
        clientsock->readDatagram(data.data(),size,&severip,&severpt);
        if(data.at(0) == '1')  //文件信息回复
        {
            fndeal(data);
            ui->connect->setEnabled(false);  //连接按钮设置为不可用
        }
        else if(data.at(0) == '3')  //收到文件数据
            datahandler(data);
        else if(data.at(0) == '5')  //收到ack
            ackdeal(data);
    }
}
```

1) 文件信息回复：

接收到文件信息，回复相应 ack，将接收到的服务器 ip 与端口与套接字绑定，之后就用这个端口进行文件传输。将文件信息用树状图展示出来，并将文件名与 id 号一一对应保存：

```
void Dialog::fndeal(QByteArray data)  //op+fn+sz
{
    QString tmp = data;
    QStringList tp = tmp.split("\r\n");
    add2list(tp.at(1), tp.at(2));
}
void Dialog::add2list(QString fn, QString sz)
{
    sendack(-1,2);
    QTreeWidgetItem *item = new QTreeWidgetItem;
    item->setText(0, fn);
    item->setText(1, sz);
    QPixmap pixmap(":/new/prefix1/apple.png");
    item->setIcon(0, pixmap);
    ui->filelist->addTopLevelItem(item);
    filename[curfile] = fn;
    curfile += 1;
}
```

2) 文件数据处理：

接收到下载数据块，全部回复 ack 以关闭服务器定时器：

```
void Dialog::sendack(int fid, int num)
{
    QString tmp = "4\r\n" + QString::number(fid) + "\r\n" + QString::number(num);
    QByteArray sp = tmp.toLocal8Bit();
    clientsock->writeDatagram(sp.data(),sp.size(),severip,severpt);
}
```

判断块号，如果收到正确顺讯数据块，则将其写入对应文件，可用 QTextStream 读写文件以更好地适应中文，或者使用 QDatastream：

```
void Dialog::datahandler(QByteArray data)  //op+fid+num+len+data
{
    QString dt = data;
    QStringList datalist = dt.split("\r\n");
    int fid = datalist.at(1).toInt();  //文件id
    int num = datalist.at(2).toInt();  //文件块号
    int len = datalist.at(3).toInt();  //数据长度
    sendack(fid, num);  //通告num块已接收以通知服务器关闭计时器，收到多余的也做相同处理
    QString cont = datalist.at(4);  //数据
    for(int i = 5; i < datalist.size(); i++)
        cont += "\r\n" + datalist.at(i);
    if(num == curnum[fid])  //收到正确块
    {
        QString fn = filename[fid];
        ui->logshow->append("成功接收: " + fn + "-" + QString::number(curnum[fid]));
        QString asfn = "/Users/renlei/Qt-workspace/saves/" + QString::number(pt%3)
                + "/" + fn;
        curnum[fid] += 1;
```

文件传输处理：如果数据块的长度小于 512，证明已是最后一块数据，将块号等重置，进度条置满：

```
if(fn.right(3) == "txt")
{
    QFile file(asfn);
    if(file.exists() && num == 0)
        file.remove();
    if(!file.open(QIODevice::ReadWrite))
    {
        ui->logshow->append("文件打开失败! ");
        return ;
    }
    QTextStream out(&file);
    out.seek(file.size());
    //写入文件
    out<<cont.toLocal8Bit();
    curstep += step;
    ui->progressBar->setValue(curstep);
    if(len < 512)  //最后一块
        ui->progressBar->setValue(100);
    file.close();
}
```

如果数据块顺序错误，则丢弃所有不满足当前块号的数据块，并申请重传正确块号以及之后的所有数据，设置申请消息的定时器：

```
else  //乱序
{
    //请求从当前块开始重发之后所有报文
    QString tmp = "6\r\n" + QString::number(fid) + "\r\n" +
            QString::number(curnum[fid]);
    QByteArray pack = tmp.toLocal8Bit();
    clientsock->writeDatagram(data.data(),data.size(),severip,severpt);
    ui->logshow->append("请求重发: " + filename[fid] + "-" +
                        QString::number(curnum[fid]));
    timer[2]->start();
}
```

3)ack 回复处理：

收到 ack 回复，关闭对应定时器：

```
void Dialog::ackdeal(QByteArray data)
{
    QString tmp = data;
    QStringList tp = tmp.split("\r\n");
    if(tp.at(1) == "-1" && tp.at(2) == "0")  //文件列表请求信息ack
    {
        ui->logshow->append("服务器成功接收文件列表请求。");
        timer[0]->stop();  //关闭计时
    }
    else if(tp.at(1) == "-1" && tp.at(2) == "1")  //文件下载请求ack
    {
        ui->logshow->append("服务器成功接收下载请求。");
        timer[1]->stop();  //关闭计时
    }
    else if(tp.at(1) == "-1" && tp.at(2) == "2")  //文件重传请求ack
    {
        ui->logshow->append("服务器成功接收重传请求。");
        timer[2]->stop();  //关闭计时
    }
}
```

5.3 服务器实现

5.3.1 初始化

初始化套接字（同时绑定端口号）、定时器、界面等，设置套接字模式为共享以允许多个客户端连接（主要是 0 号套接字，用于为客户端分配专属套接字，以减小服务器压力），套接字与定时器因为需要判断相应的服务对象，因此使用匿名函数以传参，同时给数据传输定时器的消息响应函数加上 blockSignals() 信号阻塞以允许多用户同时安全下载文件：

```
//初始化
curclient = 0;
for(int i = 0; i < 5; i++)
{
    seversock[i] = new QUdpSocket(this);
    sidfl[i] = -1;
    idpt[i] = 0;
    seversock[i]->bind(11005 + curclient,QUdpSocket::ShareAddress);
    //Lambda匿名函数，传参
    connect(seversock[i],&QUdpSocket::readyRead,this,[=]{getdata(i);});
    curclient += 1;
}
tf = new QTimer;
tf->setInterval(1000);
tf->setSingleShot(true);
connect(tf,&QTimer::timeout,this,[=]{sendfiles(csid);});
for(int i = 0; i < 3; i++)
{
    for(int j = 0; j < 24; j++)
    {
        timer[i][j] = new QTimer;
        timer[i][j]->setInterval(1000);  //设置时限
        timer[i][j]->setSingleShot(true);
        connect(timer[i][j],&QTimer::timeout,this,[=]{resend(i,j);});
    }
}
for(int i = 0; i < 5; i++)
{
    sendtm[i] = new QTimer;
    sendtm[i]->setInterval(1000);  //设置时限
    curnum[i] = 0;
    connect(sendtm[i],&QTimer::timeout,this,[=]{
        sendtm[i]->blockSignals(true);  //阻塞
        timesender(i);
        sendtm[i]->blockSignals(false);});
}
csid = 0;
root = "/Users/renlei/Qt-workspace/shareplace";
ui->logshow->append("****** 服务器就绪 ******");
```

5.3.2 服务器响应处理

每个套接字分别统一接收数据，分类进行处理，记录客户端端口号，如果是新客户端则需要给其分配新的套接字：

```
void Dialog::getdata(int sid)
{
    while(seversock[sid]->hasPendingDatagrams())  //拥有等待的数据
    {
        QByteArray data;  //用于存放接收的数据
        qint64 size=seversock[sid]->pendingDatagramSize();
        data.resize(size);
        seversock[sid]->readDatagram(data.data(),size,&ip,&pt);
        //ui->logshow->append(data);
        int fl = 0;
        for(int i = 0; i < 5; i++)
        {
            if(idpt[i] == pt)
            {
                fl = 1;
                break;
            }
        }
        if(fl == 0)  //新客户端，需分配端口
        {
            csid +=1 ;
            idpt[csid] = pt;
        }
        if(data.at(0) == '0')  //请求共享文件名
            sendfiles(csid);
        else if(data.at(0) == '2')  //收到文件下载请求
            datahandler(sid, data);
        else if(data.at(0) == '4')  //收到ack回复
            ackdeal(sid, data);
        else if(data.at(0) == '6')  //请求重传
            reorder(sid, data);
    }
}
```

1) 文件信息请求：

接收到文件列表请求信息，使用给其分配的专属套接字回复相应 ack，读取文件目录，将文件名与文件 id 对应保存，并发送文件信息到对应客户端：

```
void Dialog::sendack(int sid, int fid, int num)  //op+fid+num
{
    QString tmp = "5\r\n" + QString::number(fid) + "\r\n" + QString::number(num);
    QByteArray sp = tmp.toLocal8Bit();
    seversock[sid]->writeDatagram(sp.data(),sp.size(),ip,idpt[sid]);
}
```

```
void Dialog::sendfiles(int sid)
{
    ui->logshow->append("收到文件列表信息请求。");
    sendack(sid,-1,0);  //回复收到文件列表请求
    QDir dir(root);
    if(!dir.exists())  //文件夹不存在
    {
        ui->logshow->append("文件夹不存在。");
        return ;
    }
    dir.setFilter(QDir::Files);  //除了文件，其他的过滤掉
    QFileInfoList filelist = dir.entryInfoList();  //获取文件信息列表
    int curfile = 0;
    for(int i = 0; i < filelist.size(); i++)
    {
         QFileInfo fileinfo = filelist.at(i);
         QString fn = fileinfo.fileName();
         filename[curfile] = fn;  //文件分配id
         curfile += 1;
    }
    for(int i = 0; i < curfile; i++)
    {
        QFileInfo fileinfo = filelist.at(i);
        QString packet = "1\r\n" + filename[i] + "\r\n" + QString::number(fileinfo.size());
        QByteArray sf = packet.toLocal8Bit();
        seversock[sid]->writeDatagram(sf.data(),sf.size(),ip,idpt[sid]);
    }
    ui->logshow->append("成功发送文件列表信息。");
    tf->start();  //开始计时
}
```

2) 文件下载请求：

接收到下载请求，回复请求的 ack 以关闭客户端定时器，打开文件内容发送定时器以处理发送事件循环，以实现多用户同时安全下载文件：

```
void Dialog::datahandler(int sid, QByteArray data)  //op + filename
{
    ui->logshow->append("收到文件下载请求。");
    sendack(sid,-1,1);
    QString fn = data.right(data.size()-3);
    ui->logshow->append("客服端请求下载文件: " + fn);
    int cf = -1;  //文件id
    for(int i = 0; i < 5; i++)  //查询文件id
    {
        if(filename[i] == fn)
        {
            cf = i;
            break;
        }
    }
    sidfl[sid] = cf;  //当前sid下载的文件id为cf
    //int curnum = 0;  //块号初始化
    if(fn.right(3) == "txt")  //文件下载，陆续发送
        sendtm[sid]->start();
    else if(fn.right(3) == "jpg" || fn.right(3) == "png")
        sendtm[sid]->start();
}
```

打开文件:

```
void Dialog::timesender(int sid)
{
    int fid = sidfl[sid];
    QString asfn = root + "/" + filename[fid];
    QFile file(asfn);
    if(!file.exists())
    {
        ui->logshow->append("文件" + filename[fid] + "不存在! ");
        file.close();
        return ;
    }
    if(!file.open(QIODevice::ReadOnly))
    {
        ui->logshow->append("文件" + filename[fid] + "打开失败! ");
        return ;
    }
```

文件传输，每次读取 512 大小的数据进行发送，并设置每份数据的定时器:

```
if(filename[fid].right(3) == "txt")
{
    QTextStream in(&file);  //文本读取
    in.read(512*curnum[sid]);
    if(!in.atEnd())
    {
        QString buf = in.read(512);  //每次读取大小为512
        if(buf.size() > 0)  //成功读取
        {
            QString tmp = buf;
            QString packet = "3\r\n" + QString::number(fid) + "\r\n" +
                    QString::number(curnum[sid])+ "\r\n" + QString::number(buf.size())
                    + "\r\n" + tmp;
            QByteArray sf = packet.toLocal8Bit();
            seversock[sid]->writeDatagram(sf.data(),sf.size(),ip,idpt[sid]);
            timer[sid][curnum[sid]]->start();
            ui->logshow->append("发送文件: " + filename[fid] + "-" +
                    QString::number(curnum[sid]) + "-" + QString::number(buf.size()));
        }
        curnum[sid] += 1;
    }
    else
    {
        curnum[sid] = 0;
        sendtm[sid]->stop();
    }
    file.close();
}
```

3) 客户端 ack 回复:

接收到客户端 ack 回复，关闭相应定时器:

```
void Dialog::ackdeal(int sid, QByteArray data)  //op+fn+num
{
    QString tmp = data;
    QStringList tp = tmp.split("\r\n");
    int fid = tp.at(1).toInt();
    int num = tp.at(2).toInt();
    if(fid == -1 && num == 2)  //文件列表信息成功被接收
    {
        tf->stop();  //关闭计时
        ui->logshow->append("客户端成功接收文件列表信息。");
    }
    else  //文件下载数据块成功到达客户端
    {
        timer[sid][num]->stop();  //关闭计时
        ui->logshow->append("客户端成功接收: " + filename[fid] + "-" + tp.at(2));
    }
}
```

4) 重传请求:

接收到客户端重传请求，重传对应数据块以及之后的全部数据，大致与发送数据相同处理:

```
void Dialog::reorder(int sid, QByteArray data)  //op+fn+num
{
    ui->logshow->append("收到重传请求。");
    sendack(sid,-1,2);
    QString tmp = data;
    QStringList tp = tmp.split("\r\n");
    QString fn = filename[sidfl[sid]];
    int cnum = tp.at(2).toInt();
    QString asfn = root + "/" + fn;
    int cf = -1;  //文件id
    ui->logshow->append("客户端请求重传: " + fn + "-" + QString::number(cnum));
    QFile file(asfn);
    if(!file.exists())
    {
        ui->logshow->append("文件" + fn + "不存在! ");
        file.close();
        return ;
    }
    if(!file.open(QIODevice::ReadWrite))
    {
        ui->logshow->append("文件" + fn + "打开失败! ");
        return ;
    }
    for(int i = 0; i < 5; i++)  //查询文件id
    {
        if(filename[i] == fn)
        {
            cf = i;
            break;
        }
    }
```

5) 超时重传:

与其他信号槽类似，由初始化时 connect 的槽函数进行处理，使用匿名函数以传参，超时未收到对应 ack，则触发 resend 重传函数重新发送此数据包:

```
connect(timer[i][j],&QTimer::timeout,this,[=]{resend(i,j);});
```

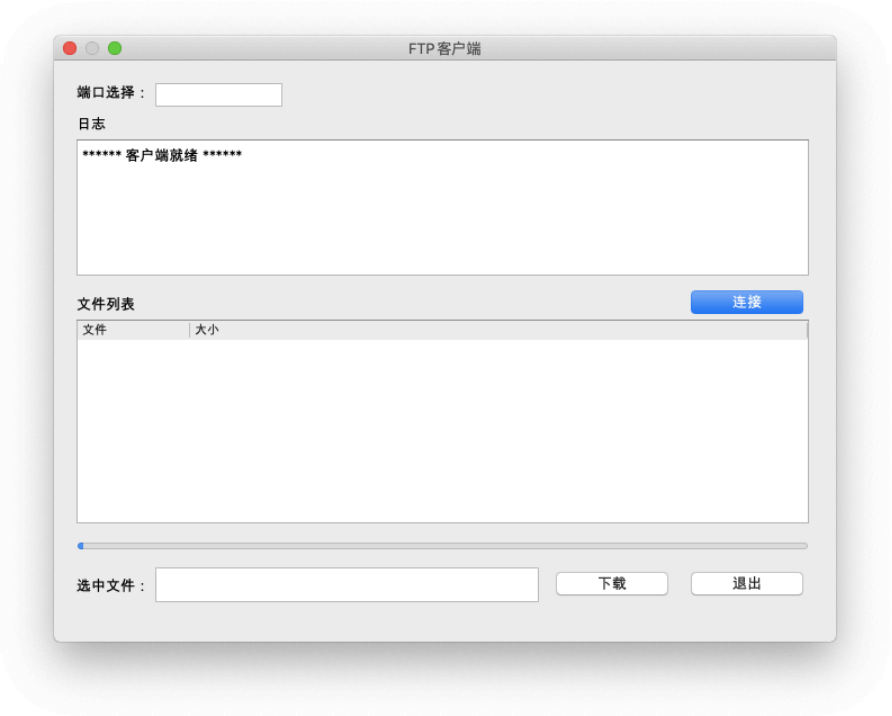
```
void Dialog::resend(int sid, int num)  //超时重传
{
    QString fn = filename[sidfl[sid]];
    int cnum = num;
    QString asfn = root + "/" + fn;
    int cf = -1;  //文件id
    ui->logshow->append(fn + "-" + QString::number(cnum) +"超时重传。");
    QFile file(asfn);
    if(!file.exists())
    {
        ui->logshow->append("文件" + fn + "不存在! ");
        file.close();
        return ;
    }
    if(!file.open(QIODevice::ReadWrite))
    {
        ui->logshow->append("文件" + fn + "打开失败! ");
        return ;
    }
    for(int i = 0; i < 5; i++)  //查询文件id
    {
        if(filename[i] == fn)
        {
            cf = i;
            break;
        }
    }
```

这里只截取了部分代码，大致与发送数据作相同处理。

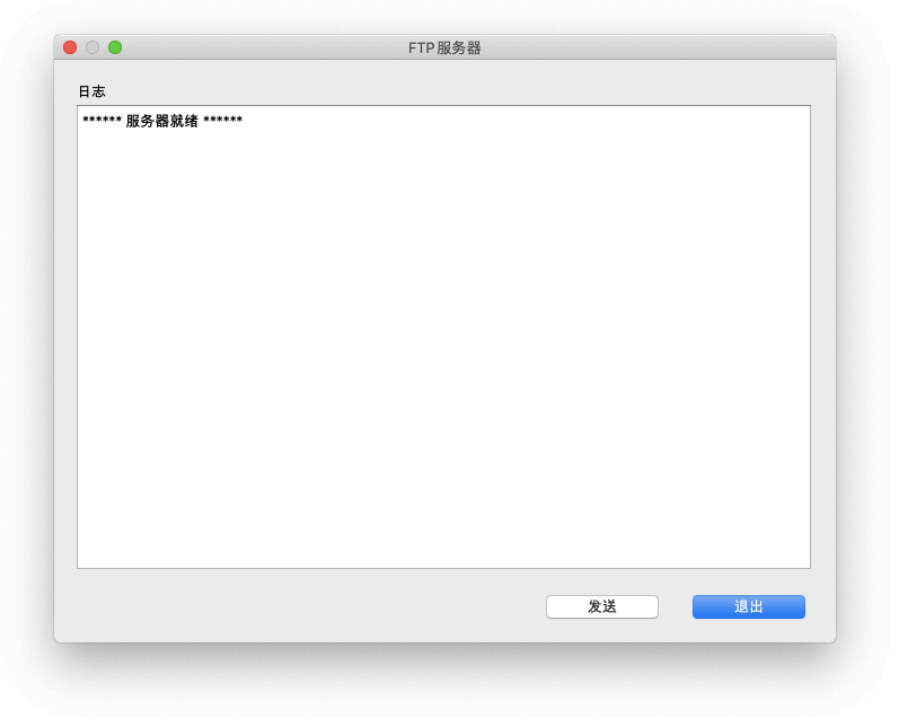
6 实验结果

6.1 初始界面

客户端:

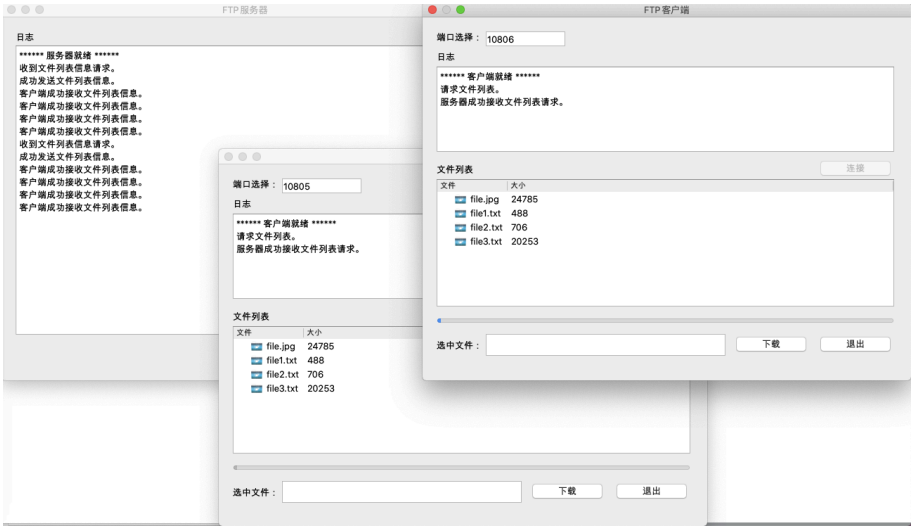


服务器:

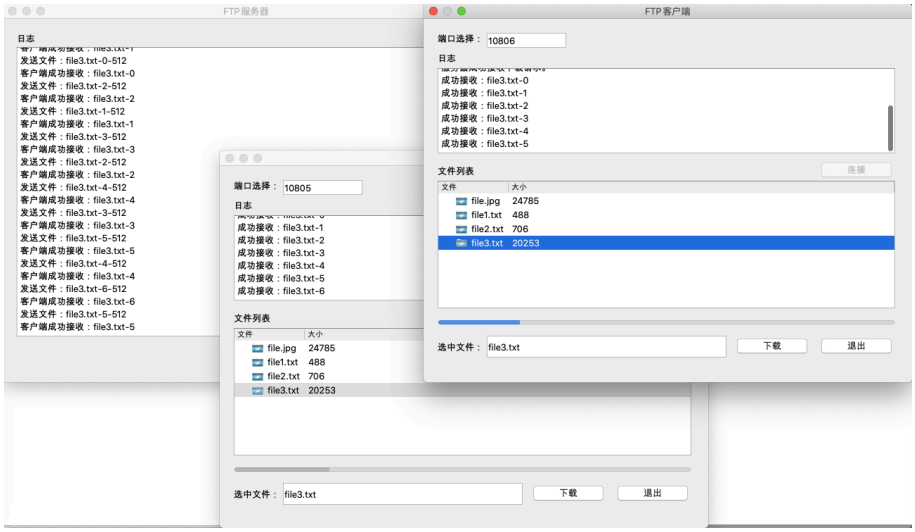


6.2 多用户下载演示

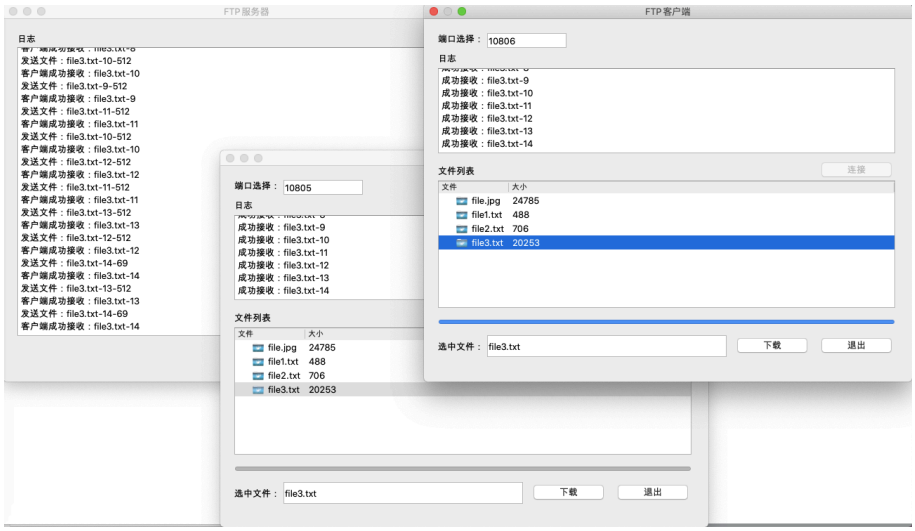
客户端设置端口号，连接服务器获得文件列表：



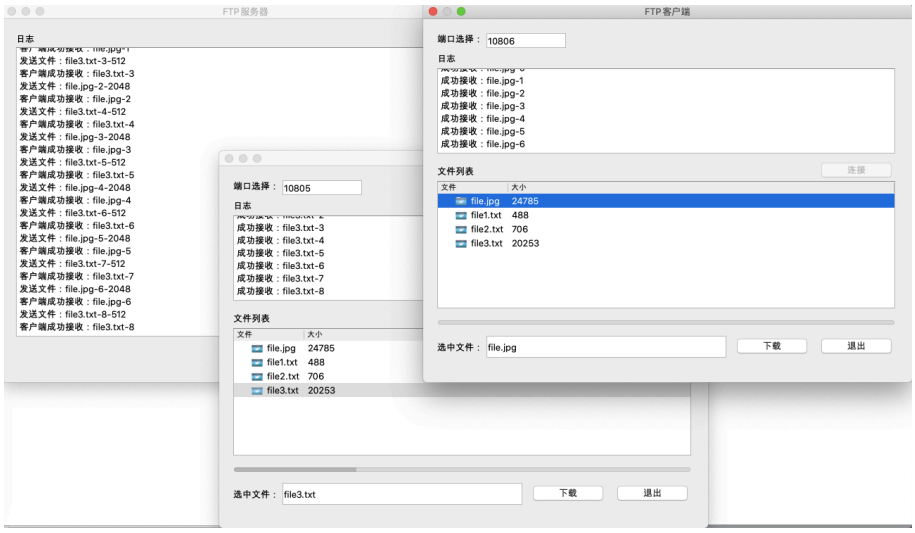
选中同一份文件进行下载：



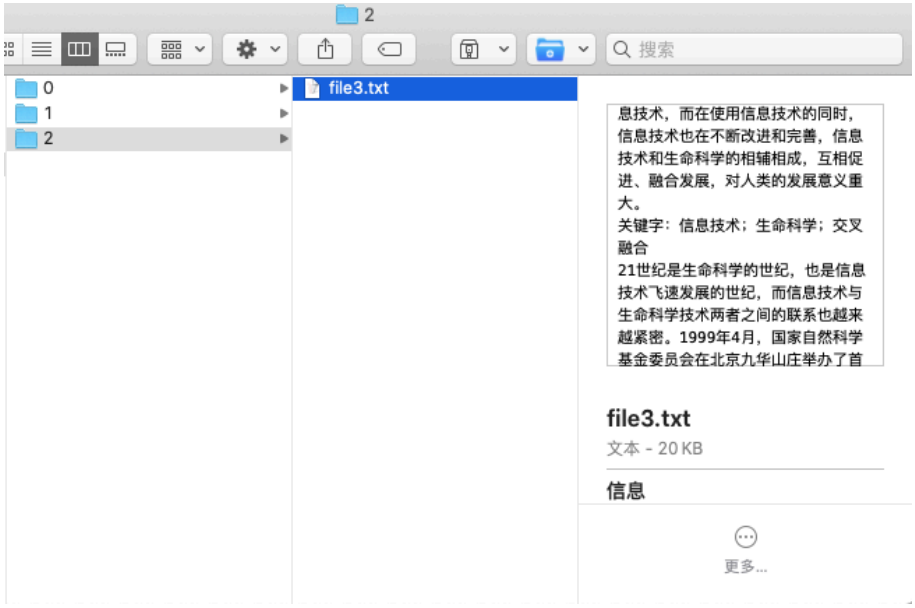
下载完成：

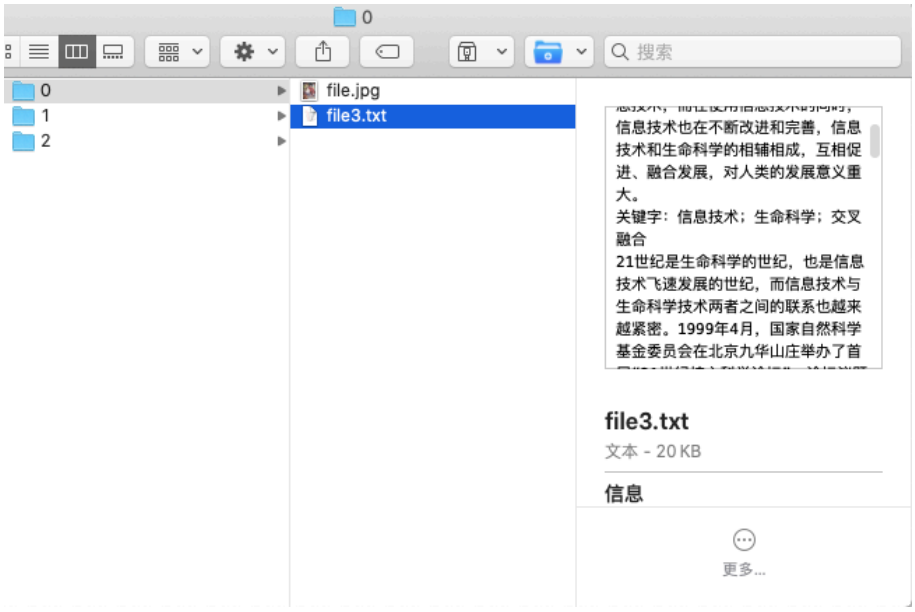


选中不同文件进行下载：


FTP服务器
日志
发送文件：file3.txt-3-512
客户端成功接收：file3.txt-3
发送文件：file.jpg-2-2048
客户端成功接收：file.jpg-2
发送文件：file3.txt-4-512
客户端成功接收：file3.txt-4
发送文件：file.jpg-3-2048
客户端成功接收：file.jpg-3
发送文件：file3.txt-5-512
客户端成功接收：file3.txt-5
发送文件：file.jpg-4-2048
客户端成功接收：file.jpg-4
发送文件：file3.txt-6-512
客户端成功接收：file3.txt-6
发送文件：file.jpg-5-2048
客户端成功接收：file.jpg-5
发送文件：file3.txt-7-512
客户端成功接收：file3.txt-7
发送文件：file.jpg-6-2048
客户端成功接收：file.jpg-6
发送文件：file3.txt-8-512
客户端成功接收：file3.txt-8
端口选择：10805
日志
成功接收：file3.txt-3
成功接收：file3.txt-4
成功接收：file3.txt-5
成功接收：file3.txt-6
成功接收：file3.txt-7
成功接收：file3.txt-8
文件列表
文件 大小
file.jpg 24785
file1.txt 488
file2.txt 706
file3.txt 20253
选中文件：file3.txt
下载
退出
FTP客户端
端口选择：10806
日志
成功接收：file.jpg-1
成功接收：file.jpg-2
成功接收：file.jpg-3
成功接收：file.jpg-4
成功接收：file.jpg-5
成功接收：file.jpg-6
文件列表
连接
文件 大小
file.jpg 24785
file1.txt 488
file2.txt 706
file3.txt 20253
选中文件：file.jpg
下载
退出

下载的文件成功储存：


2
搜索
0
1
2
file3.txt
息技术，而在使用信息技术的同时，信息技术也在不断改进和完善，信息技术和生命科学的相辅相成，互相促进、融合发展，对人类的发展意义重大。
关键字：信息技术；生命科学；交叉融合
21世纪是生命科学的世纪，也是信息技术飞速发展的世纪，而信息技术与生命科学技术两者之间的联系也越来越紧密。1999年4月，国家自然科学基金委员会在北京九华山庄举办了首
file3.txt
文本 - 20 KB
信息
更多...



至此，实验完成。

7 实验总结

7.1 FTP 协议

FTP（File Transfer Protocol，文件传输协议）是 TCP/IP 协议组中的协议之一。其包括两个组成部分，FTP 服务器和 FTP 客户端。其中 FTP 服务器用来存储文件，用户可以使用 FTP 客户端通过 FTP 协议访问位于 FTP 服务器上的资源。

7.2 UDP 套接字编程

数据报套接字使用 UDP 协议进行数据的传输，它提供了一种无连接的面向事务的简单不可靠信息传送服务，可能出现丢包或者数据重复情况，且无法保证顺序地接收到数据，因此要使用 UDP 作可靠传输，需要检测错误并在出错时重传。

相对来说 UDP 套接字对资源的开销比 TCP 小，速度也比 TCP 快，因为不需要维持网络连接，也不用花费时间来建立连接。

UDP 编程步骤：

创建套接字 → 绑定套接字 → 服务器端等待客户的消息 → 客户端发送消息 → 关闭套接字

对于 UDP 来说，真正负责发送和接收的函数都是同步阻塞函数，在执行完这些函数之前无法执行其他函数，直接放到主线程中会导致程序卡死，因此如何避免这些问题是我们设计程序时的重点内容。通过实际操作 UDP 编程，有助于我们更好地掌握相关计算机网络编程的知识与技术。