UDP协议的应用

也许有的读者会问，既然UDP是一种不可靠的网络协议，那么还有什么使用价值或必要呢？其实不然，在有些情况下UDP协议可能会变得非常有用。因为UDP具有TCP所望尘莫及的速度优势。虽然TCP协议中植入了各种安全保障功能，但是在实际执行的过程中会占用大量的系统开销，无疑使速度受到严重的影响。反观UDP由于排除了信息可靠传递机制，将安全和排序等功能移交给上层应用来完成，极大降低了执行时间，使速度得到了保证。

**UDP协议的几个特性**

（1） UDP是一个[无连接协议](http://baike.baidu.com/view/732127.htm" \t "_blank)，传输数据之前源端和终端不建立连接，当它想传送时就简单地去抓取来自应用程序的数据，并尽可能快地把它扔到网络上。在发送端，UDP传送数据的速度仅仅是受应用程序生成数据的速度、计算机的能力和传输带宽的限制；在接收端，UDP把每个消息段放在队列中，应用程序每次从队列中读一个消息段。

（2） 由于传输数据不建立连接，因此也就不需要维护连接状态，包括收发状态等，因此一台服务机可同时向多个客户机传输相同的消息。

（3） UDP信息包的标题很短，只有8个字节，相对于TCP的20个字节信息包的额外开销很小。

（4） 吞吐量不受拥挤控制算法的调节，只受[应用软件](http://baike.baidu.com/view/7886.htm" \t "_blank)生成数据的速率、传输带宽、源端和终端主机性能的限制。

（5）UDP使用尽最大努力交付，即不保证可靠交付，因此主机不需要维持复杂的链接状态表（这里面有许多参数）。

（6）UDP是面向报文的。发送方的UDP对应用程序交下来的报文，在添加首部后就向下交付给IP层。既不拆分，也不合并，而是保留这些报文的边界，因此，应用程序需要选择合适的报文大小。

　　虽然UDP是一个不可靠的协议，但它是分发信息的一个理想协议。例如，在屏幕上报告股票市场、在屏幕上显示航空信息等等。UDP也用在[路由信息协议](http://baike.baidu.com/view/149977.htm" \t "_blank)RIP（Routing Information Protocol）中修改路由表。在这些应用场合下，如果有一个消息丢失，在几秒之后另一个新的消息就会替换它。UDP广泛用在多媒体应用中，例如，Progressive Networks公司开发的RealAudio[软件](http://baike.baidu.com/view/37.htm)，它是在[因特网](http://baike.baidu.com/view/1706.htm)上把预先录制的或者现场音乐实时传送给客户机的一种软件，该软件使用的RealAudio audio-on-demand protocol协议就是运行在UDP之上的协议，大多数因特网电话软件产品也都运行在UDP之上。

一下附贴写了个UDP的收发程序基于Qt的

即可以做发送者也可以做接收者

weatherballoon.h

#ifndef WEATHERBALLOON\_H

#define WEATHERBALLOON\_H

#include <QWidget>

#include <QPushButton>

#include <QtNetwork/QUdpSocket>

#include <QTimer>

#include <QDateTime>

namespace Ui {

class weatherBalloon;

}

class weatherBalloon : public QWidget

{

Q\_OBJECT

public:

explicit weatherBalloon(QWidget \*parent = 0);

~*weatherBalloon*();

private slots:

void processPendingDatagrams();

void sendDatagram();

private:

Ui::weatherBalloon \*ui;

QUdpSocket udpSocket;

QTimer timer;

double temperature;

double humidity;

double altitude;

};

#endif // WEATHERBALLOON\_H

main.cpp

#include <QtGui/QApplication>

#include "weatherballoon.h"

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

weatherBalloon w;

w.show();

return a.exec();

}

weatherballoon.cpp

#include "weatherballoon.h"

#include "ui\_weatherballoon.h"

weatherBalloon::weatherBalloon(QWidget \*parent) :

QWidget(parent),

ui(new Ui::weatherBalloon)

{

udpSocket.bind(5824);//绑定端口号

ui->setupUi(this);

connect(ui->pushButton,SIGNAL(clicked()),this,SLOT(close()));

connect(&timer,SIGNAL(timeout()),this,SLOT(sendDatagram()));

connect(&udpSocket,SIGNAL(readyRead()),this,SLOT(processPendingDatagrams()));

timer.start(2\*1000);

temperature = 10.2;

humidity = 5.4;

altitude = 100.0;

setWindowTitle(tr("Weather Balloon"));

}

weatherBalloon::~*weatherBalloon*()

{

delete ui;

}

void weatherBalloon::sendDatagram(){

QByteArray datagram;

QDataStream out(&datagram,QIODevice::WriteOnly);

out.setVersion(QDataStream::Qt\_4\_3);

out<<QDateTime::currentDateTime()<<temperature<<humidity<<altitude;

qDebug()<<QDateTime::currentDateTime();

QHostAddress address;

address.setAddress("192.168.0.46");//发送者要把数据发送到的ip地址

udpSocket.writeDatagram(datagram,address,5824);//发送者把数据发送的端口号，需要接受者绑定该端口号

}

void weatherBalloon::processPendingDatagrams(){

QByteArray datagram;//拥于存放接收的数据报

do{

datagram.resize(udpSocket.pendingDatagramSize());//让datagram的大小为等待处理的数据报的大小，这样才能接收到完整的数据

udpSocket.readDatagram(datagram.data(),datagram.size());//接收数据报，将其存放到datagram中

}while(udpSocket.hasPendingDatagrams());//拥有等待的数据报

QDateTime dateTime;

double temperature;

double humidity;

double altitude;

qDebug()<<"recive date ";

QDataStream in(&datagram,QIODevice::ReadOnly);

in.setVersion(QDataStream::Qt\_4\_3);

in>>dateTime>>temperature>>humidity>>altitude;

ui->dateLineEdit->setText(dateTime.date().toString());

ui->timeLineEdit->setText(dateTime.time().toString());

ui->temperatureLineEdit->setText(tr("%1 °c").arg(temperature));

ui->humidityLineEdit->setText(tr("%1%").arg(humidity));

ui->altiudeLineEdit->setText(tr("%1 m").arg(altitude));

}

一下 是接收者和发送者跟上一程序相互之间发送

dialog.h

#ifndef DIALOG\_H

#define DIALOG\_H

#include <QDialog>

#include <QtNetwork/QUdpSocket>

#include <QTimer>

namespace Ui {

class Dialog;

}

class Dialog : public QDialog

{

Q\_OBJECT

public:

explicit Dialog(QWidget \*parent = 0);

~*Dialog*();

private slots:

void processPendingDatagrams();

void sendDatagram();

private:

Ui::Dialog \*ui;

QUdpSocket udpSocket;

QTimer timer;

double temperature;

double humidity;

double altitude;

};

#endif // DIALOG\_H

dialog.cpp

#include "dialog.h"

#include "ui\_dialog.h"

#include <QByteArray>

#include <QDateTime>

#include <QDataStream>

Dialog::Dialog(QWidget \*parent) :

QDialog(parent),

ui(new Ui::Dialog)

{

udpSocket.bind(5824);

connect(&udpSocket,SIGNAL(readyRead()),this,SLOT(processPendingDatagrams()));

ui->setupUi(this);

timer.start(4\*1000);

temperature = 20.2;

humidity = 15.4;

altitude = 150.0;

connect(&timer,SIGNAL(timeout()),this,SLOT(sendDatagram()));

}

Dialog::~*Dialog*()

{

delete ui;

}

void Dialog::processPendingDatagrams(){

QByteArray datagram;

do{

datagram.resize(udpSocket.pendingDatagramSize());

udpSocket.readDatagram(datagram.data(),datagram.size());

}while(udpSocket.hasPendingDatagrams());

QDateTime dateTime;

double temperature;

double humidity;

double altitude;

QDataStream in(&datagram,QIODevice::ReadOnly);

in.setVersion(QDataStream::Qt\_4\_3);

in>>dateTime>>temperature>>humidity>>altitude;

ui->dateLineEdit->setText(dateTime.date().toString());

ui->timeLineEdit->setText(dateTime.time().toString());

ui->temperatureLineEdit->setText(tr("%1 °c").arg(temperature));

ui->humidityLineEdit->setText(tr("%1%").arg(humidity));

ui->altiudeLineEdit->setText(tr("%1 m").arg(altitude));

}

void Dialog::sendDatagram(){

QByteArray datagram;

QDataStream out(&datagram,QIODevice::WriteOnly);

out.setVersion(QDataStream::Qt\_4\_3);

out<<QDateTime::currentDateTime()<<temperature<<humidity<<altitude;

qDebug()<<QDateTime::currentDateTime();

QHostAddress address;

address.setAddress("192.168.3.217");

udpSocket.writeDatagram(datagram,address,5824);

}

main.c

#include <QtGui/QApplication>

#include "dialog.h"

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

Dialog w;

w.show();

return a.exec();

}

以上两个程序的.pro文件添加QT += network