Чугунова Д.И.
Научный руководитель: к.т.н., доц. А. А. Колпаков
Муромский институт (филиал) федерального государственного образовательного учреждения высшего образования «Владимирский государственный университет имени Александра Григорьевича и Николая Григорьевича Столетовых»
602264, г. Муром, Владимирская обл., ул. Орловская, 23
dana.chugunova@mail.ru

Разработка систем телекоммуникаций на основе среды Qt

Телекоммуникационная система – упорядоченная совокупность методов, правил, протоколов, технических и программных средств в их взаимосвязи и взаимодействии, обеспечивающих передачу электронного сообщения от источника к получателю по сетям электросвязи [1].

Для разработки телекоммуникационных систем была выбрана среда программирования Qt Creator.

Среда разработки Qt Creator – кроссплатформенный инструментарий разработки ПО на языке программирования C++. Он позволяет запускать написанное с его помощью ПО в большинстве современных операционных системах путём простой компиляции программы для каждой ОС без изменения исходного кода. Включает в себя все основные классы, которые могут потребоваться при разработке прикладного программного обеспечения, начиная от элементов графического интерфейса и заканчивая классами для работы с сетью, базами данных и XML. Qt является полностью объектно-ориентированным, легко расширяемым и поддерживающим технику компонентного программирования.

Qt предоставляет программисту не только удобный набор библиотек классов, но и определённую модель разработки приложений, определённый каркас их структуры. Следование принципам и правилам «хорошего стиля программирования на C++/Qt» существенно снижает частоту таких трудно отлавливаемых ошибок в приложениях, как утечки памяти (memory leaks), необработанные исключения, незакрытые файлы или неосвобождённые дескрипторы ресурсных объектов, чем нередко страдают программы, написанные «на голом C++» без использования [2].

Для того, чтобы облегчить создание сетевых кроссплатформенных приложений, разработчики фреймворка Qt предусмотрели модуль работы с сетью QtNetwork. Модуль QtNetwork содержит как высокоуровневые классы, такие как QHttp или QFtp, так и классы QAbstractSocket, QTcpServer, QUdpSocket, с помощью которых можно работать с сетью на низком уровне.

Сокет – это устройство пересылки данных с одного конца связи на другой. Другой конец может принадлежать процессу, работающему на локальном компьютере, а может располагаться и на удаленном компьютере, подключенному к Интернету и расположенному в другом полушарии Земли. Сокетное соединение – это соединение типа точка-точка (point to point), которое производится между двумя процессами.

Сокеты разделяют на дейтаграммные (datagram) и поточные. Дейтаграммные сокеты осуществляют обмен пакетами данных. Поточные сокеты устанавливают связь и производят потоковый обмен данными через установленную ими связь. На практике, поточные сокеты используются гораздо чаще, чем дейтаграммные из-за того, что они предоставляют дополнительные механизмы, направленные против искажения и потери данных. Поточные сокеты работают в обоих направлениях, то есть то, что один из процессов записывает в поток, может быть считано процессом на другом конце связи, и наоборот [3].

Для дейтаграммных сокетов Qt предоставляет класс QUdpSocket, а для поточных — класс QTcpSocket.

Для телекоммуникационных систем лучше всего использовать дейтаграммные сокеты, то есть класс QUdpSocket в нашем случае, так как благодаря данному классу мы можем предотвратить искажения и потери данных. Протокол UDP обладает такими преимуществами,

как: малое потребление ресурсов и высокая коммуникационная эффективность (передача аудио и видео).

На рисунке 1 показана обобщенная структурная схема телекоммуникационной системы. Ниже (рис. 2) представлен пример простой реализации использования UDPSocket:

Рис. 1 – Обобщенная структурная схема телекоммуникационной системы

Обозначения: ИС-источник сообщений; $ПР_1$ – преобразователь передаваемых сообщений в первичный сигнал (первичный преобразователь); ПРД – передатчик, т.е. комплекс технических средств, предназначенных для согласования передаваемых сигналов и среды распространения; СР-среда распространения; ПРМ – приемник выполняет операции, обратные ПРД; $ПР_2$-преобразователь принятых сигналов в сообщение (обратный преобразователь); ПС – получатель сообщений; a(t)-передаваемое сообщение; U(t) – первичный сигнал электросвязи; S(t) – сигнал, преобразованный к виду, удобному для передачи по данной среде. Канал передачи – это совокупность технических средств и среды распространения, обеспечивающая передачу сигналов электросвязи в определенной полосе частот или с определенной скоростью передачи между оконечными или промежуточными пунктами телекоммуникационной системы или сети. Канал электросвязи – это канал передачи, включающий преобразователи сообщений в первичные сигналы и первичные сигналы в сообщения.

В .pro файл добавляем Qt += network.

Код клиента:

```
#ifndef UCHAT_H
#define UCHAT_H
#include <QMainWindow>
#include <QUdpSocket>
QT_BEGIN_NAMESPACE
namespace Ui { class uchat; }
QT_END_NAMESPACE
class uchat : public QMainWindow
{
   Q_OBJECT
public:
   uchat(QWidget *parent = nullptr);
   ~uchat();
public slots:
   void readPendingDatagrams();
private slots:
   void on_pushButton_clicked();
   void on_pushButton_3_clicked();
private:
```

```
    Ui::uchat *ui;
    QUdpSocket *clientSocketc;
    QUdpSocket * clientSocket;
};
#endif // UCHAT_H
```

В заголовочном файле подключаем библиотеку QUdpSocket и инициализируем переменные и процедуры.

```
uchat::uchat(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::uchat)
{
    ui->setupUi(this);
   clientSocket=new QUdpSocket(this);
   clientSocketc=new QUdpSocket(this);
    clientSocketc->bind(QHostAddress::LocalHost, 7000);
connect(clientSocketc,SIGNAL(readyRead()),this,SLOT(readPendingDatagrams()));
}
uchat::~uchat()
{
    delete ui;
}
void uchat::readPendingDatagrams()
{
while (clientSocketc->hasPendingDatagrams()) {
QTime tm = QTime::currentTime();
QByteArray buffer;
buffer.resize(clientSocketc->pendingDatagramSize());
QHostAddress sender;
quint16 senderPort;
clientSocketc->readDatagram(buffer.data(), buffer.size(),&sender, &senderPort);
ui->textBrowser->append(tm.toString());
ui->textBrowser->append(buffer.data());
}
}
void uchat::on_pushButton_clicked()
{
    QTime tm = QTime::currentTime();
    QString word=ui->lineEdit->text();
    ui->textBrowser->append(tm.toString());
    ui->textBrowser->append(word);
    ui->lineEdit->clear();
    QByteArray buffer;
    buffer.resize(clientSocket->pendingDatagramSize());
    QHostAddress sender;
    buffer=word.toUtf8();
    clientSocket->writeDatagram(buffer.data(), QHostAddress::LocalHost, 8001 );
}
void uchat::on_pushButton_3_clicked()
{
    ui->textBrowser->clear();
}
```

В исходном файле клиента реализуем отправку сообщений на сервер.

Код сервера:

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
```

```
#include <QMainWindow>
#include <QUdpSocket>
#include <QHostAddress>
#include <QTextEdit>
QT_BEGIN_NAMESPACE
namespace Ui { class schat; }
QT_END_NAMESPACE

class schat : public QMainWindow
{
   Q_OBJECT
public:
   schat(QWidget *parent = nullptr);
   ~schat();
public slots:
   void readPendingDatagrams();
private slots:
  void on_pushButton_clicked();
  void on_pushButton_2_clicked();
private:
   Ui::schat *ui;
   QUdpSocket *socketServerc;
   QUdpSocket *socketServer;
};
#endif // MAINWINDOW_H
```

В заголовочном файле проделываем те же манипуляции, что и в заголовочном файле клиента.

В исходном файле сервера реализуем прием, чтение и отправку полученных сообщений клиента.

```
#include "Server.h"
#include "ui_schat.h"
#include <QLineEdit>
#include <QString>
#include <QTime>
#include <QCryptographicHash>
#include <QFileDialog>
#include <QMessageBox>
#include <QTypeInfo>
#include <QFile>
schat::schat(QWidget *parent) : QMainWindow(parent),
ui(new Ui::schat)
{
ui->setupUi(this);
socketServerc=new QUdpSocket(this);
socketServer=new QUdpSocket(this);
socketServer->bind(QHostAddress::LocalHost, 8001);
connect(socketServer,SIGNAL(readyRead()),this,SLOT(readPendingDatagrams()));
}
schat::~schat()
{
delete ui;
}
void schat::readPendingDatagrams()
{
while (socketServer->hasPendingDatagrams()) {
```

```
QTime tm = QTime::currentTime();
QByteArray buffer;
buffer.resize(socketServer->pendingDatagramSize());
QHostAddress Port;
quint16 senderPort;
socketServer->readDatagram(buffer.data(), buffer.size(),&Port, &senderPort);
ui->textBrowser->append("Datagram Received From");
ui->textBrowser->append("Client IP:");
ui->textBrowser->append(Port.toString());
qDebug()<<"Time: " << tm.toString();
qDebug()<<"Client IP" << Port.toString();
qDebug()<<"Client Port Number " << senderPort;
qDebug()<<"\n\n";

QByteArray clientData;
clientData.append( "Datagram received");
socketServer->writeDatagram( clientData, QHostAddress::LocalHost, senderPort );
QUdpSocket* pClientSocket = (QUdpSocket*)sender();
QDataStream in(pClientSocket);
in.setVersion(QDataStream::Qt_5_14);
   QString str;
   in >> str;
   QString strMessage =
      tm.toString() + " " + "Client has sended:" + str;
   ui->textBrowser->append(strMessage);
   ui->textBrowser->append(buffer.data());
     }
 }
void schat::on_pushButton_clicked()
{
   QTime tm = QTime::currentTime();
   QString word=ui->lineEdit->text();
   ui->textBrowser->append(tm.toString());
   ui->textBrowser->append(word);
   ui->lineEdit->clear();
   QByteArray buffer;
   buffer=word.toUtf8();
   QHostAddress sender;
   quint16 senderPort;
   socketServerc->writeDatagram(buffer.data(), QHostAddress::LocalHost, 7000 );
   socketServerc->readDatagram(buffer.data(), buffer.size(), &sender, &senderPort);
}
void schat::on_pushButton_2_clicked()
{
   ui->textBrowser->clear();
}
```

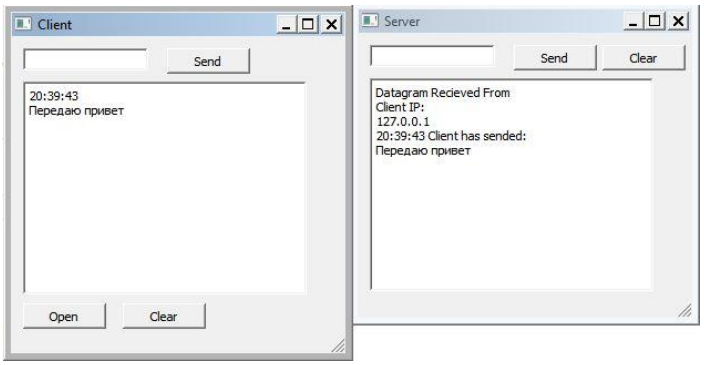

Рис. 2 – Пример простой реализации использования UDPSocket

Главной задачей при построении системы безопасности в телекоммуникациях – это предотвращение утечки информации через отдельные каналы. Поэтому среда разработки Qt Creator, со своим удобным интерфейсом и кроссплатформенностью, и класс QUdpSocket, с возможностью предотвращения потери данных или искажения при передачи, подходят как нельзя лучше для разработки телекоммуникационных систем.

Литература

1. Пескова, С.А. Сети и телекоммуникации: учебник / С.А. Пескова. - М.: Academia, 2017. - 416 с.

2. М. Саммерфилд «Qt. Профессиональное программирование. Разработка кроссплатформенных приложений на C++», Символ-Плюс, 2019 – 560 с.

3. Алексеев Е. Б., Гордиенко В. Н., Крухмалев В. В. и др. «Проектирование и техническая эксплуатация цифровых телекоммуникационных систем и сетей», издательство "Питер", 2017 – 608 с.

4. «RFC 768. Протокол Пользовательских Датаграмм» [электронный ресурс], URL = «https://datatracker.ietf.org/doc/html/rfc768» , режим доступа – свободный. Дата обращения – 06.04.2022 г