

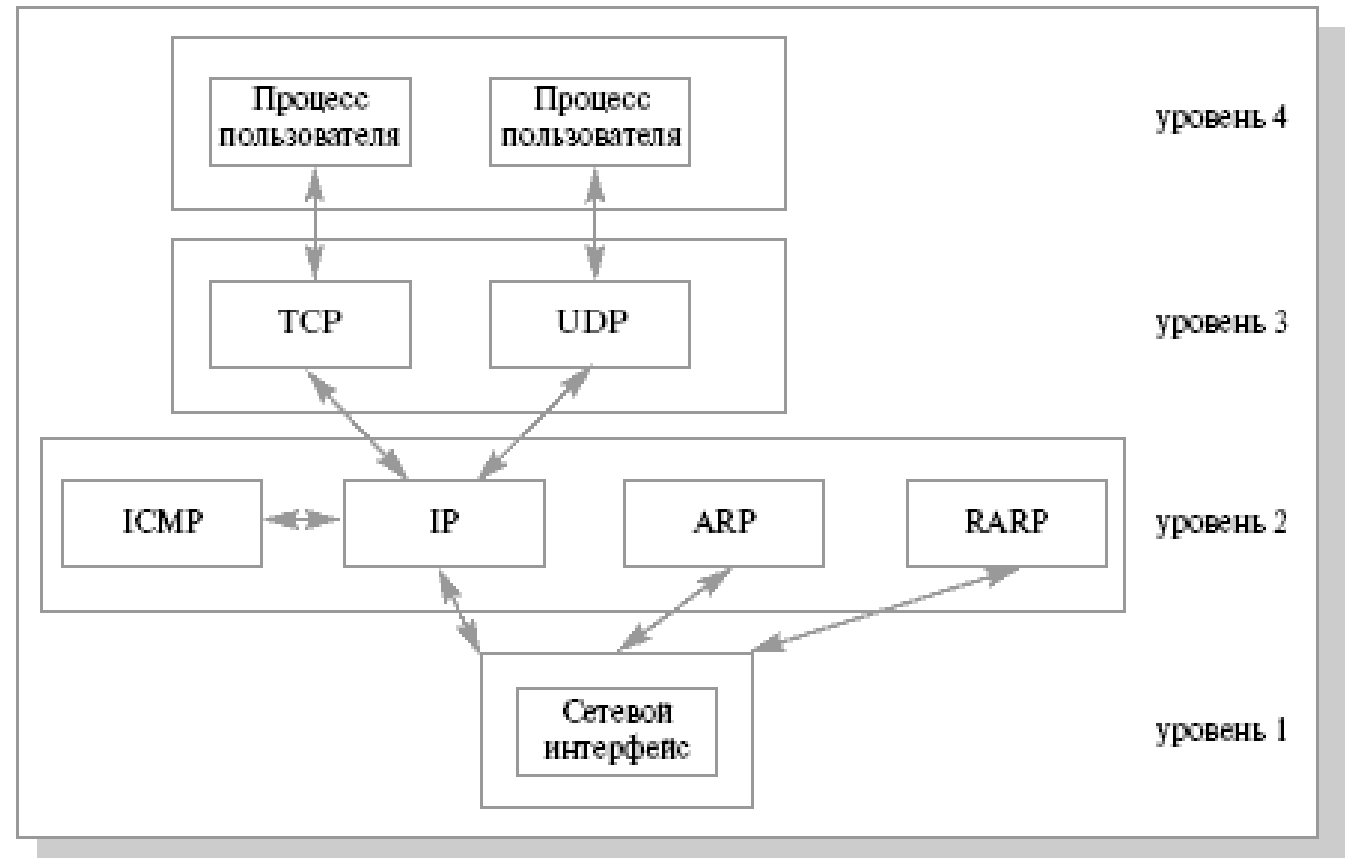
СЕМИНАР 3

Стек протоколов TCP/IP

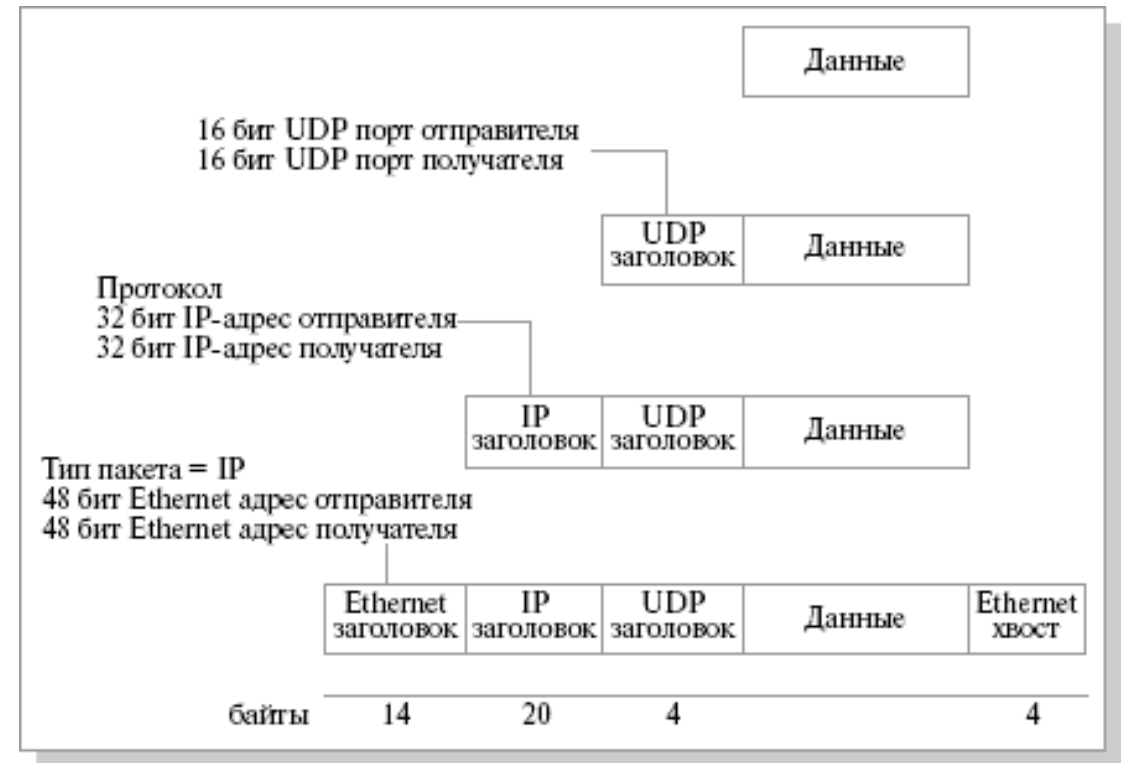
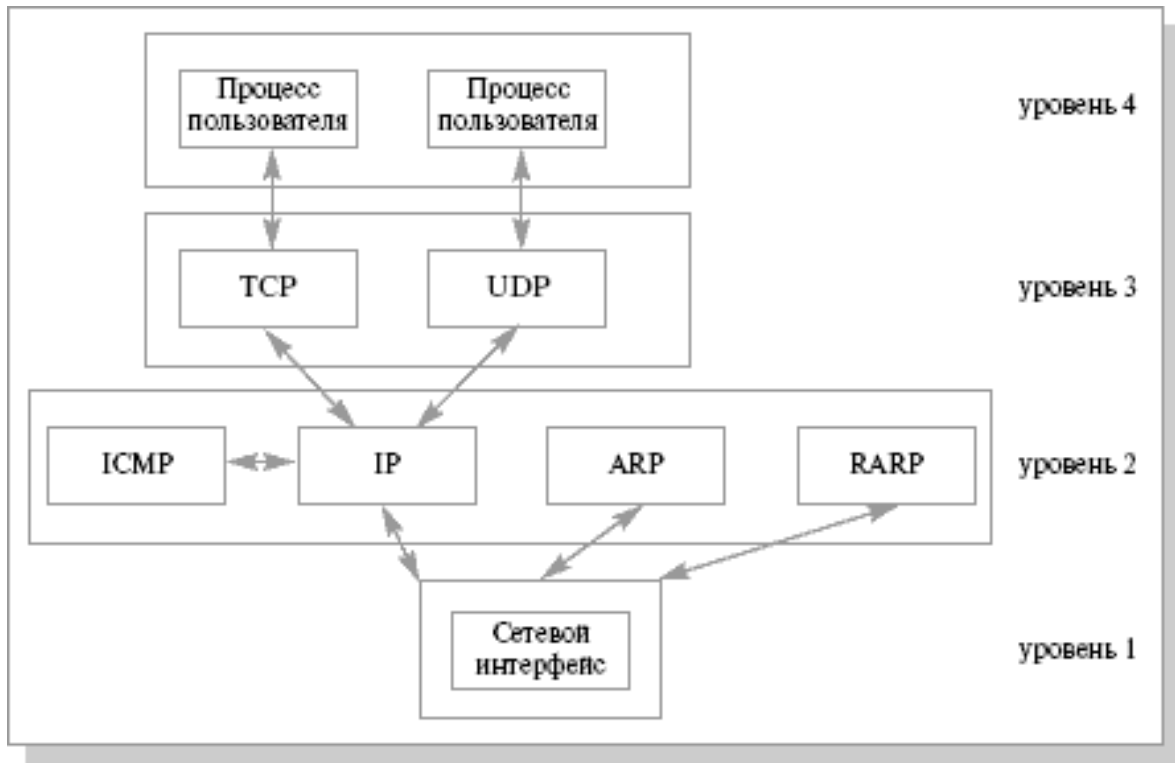
Модель OSI

Работа с сетью в Qt

Сетевая модель



Сетевая модель



UDP (User Datagram Protocol)

- Работает поверх IP
- Не гарантирует доставку данных.
- Не гарантирует порядок доставки пакетов. Вы можете отправить пять пакетов по порядку — 1, 2, 3, 4, 5 — а прийти они могут совершенно в другом порядке — к примеру, 3, 1, 2, 5, 4.
- Прост в реализации

Сокеты Беркли

- Впервые появились в 1989 г. для UNIX в калифорнийском университете Беркли
 - Сокет в UNIX файл специального вида
 - Все, что записывается в файл, передаётся по сети
 - Передача данных по сети скрыта от программиста
- Интерфейс сокетов — это API для сетей TCP/IP
- Поддерживаются не только в UNIX-подобных ОС, но и в Windows, QNX и т.п.

Сокеты Беркли

- Операции
 - Создать новый сокет
 - Bind – связать сокет с IP – адресом и портом
 - Listen – объявить о желании принимать соединения
 - Accept – принять запрос на установку соединения
 - Connect – установить соединение
 - Send – отправить данные по сети
 - Receive – получить данные по сети
 - Close – закрыть соединение

Сокеты Беркли

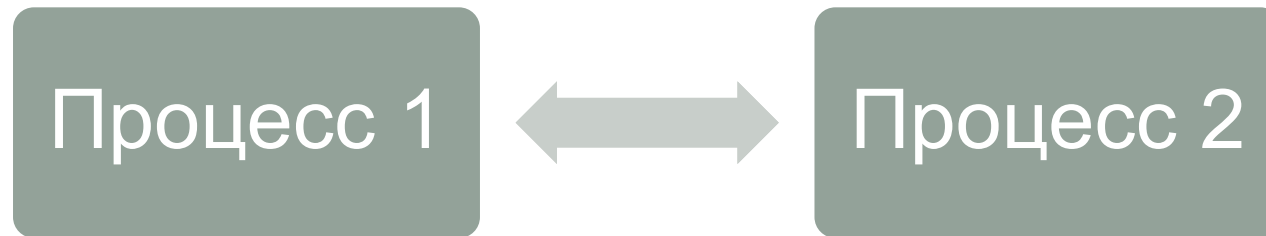
- Взаимодействующие стороны:
 - Сервер
 - Клиент
- Сервер – слушает на определенном IP – адресе и порту
- Клиент – активно устанавливает соединение с сервером на заданном IP и порту

Поддержка сети. QAbstractSocket.

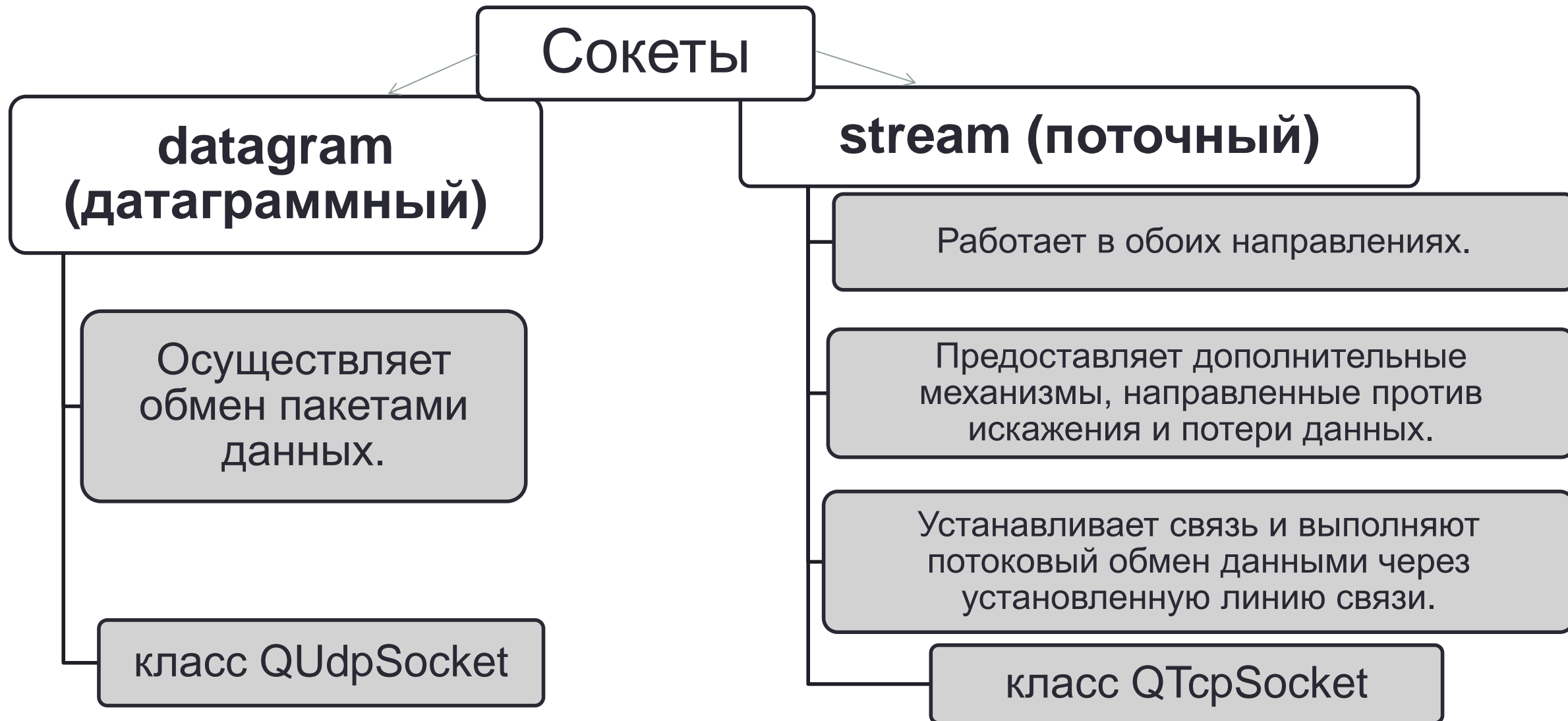
Сокетное соединение.

Сокет – устройство пересылки данных с одного конца линии связи на другой.

Сокетное соединение – соединение типа точка-точка (point to point), которое осуществляется между двумя процессами.



Поддержка сети. QAbstractSocket.



IP адрес

Чтобы обмениваться данными с другими устройствами посредством протокола TCP/IP, устройству необходим IP-адрес.

Представления ip-адреса:

- 4 числа разделенные точкой 1.2.3.4
- 4 байта (unsigned int: 127.0.0.1 = 2130706433)
- 4 числа + маска:
 - cidr 1.2.3.4/24
 - 1.2.3.4/255.255.255.0
 - wildcard 1.2.3.4/0.0.0.255
- Маска нужна для того, чтобы отделить адрес сети от адреса хоста.
- Для работы с IP – адресами в Qt используется класс QHostAddress. (Установка адреса выполняется методом setAddress(""))

**Если устройство имеет IP-адрес и соответствующее программно-аппаратное обеспечение, то оно может отправлять IP-пакеты.*

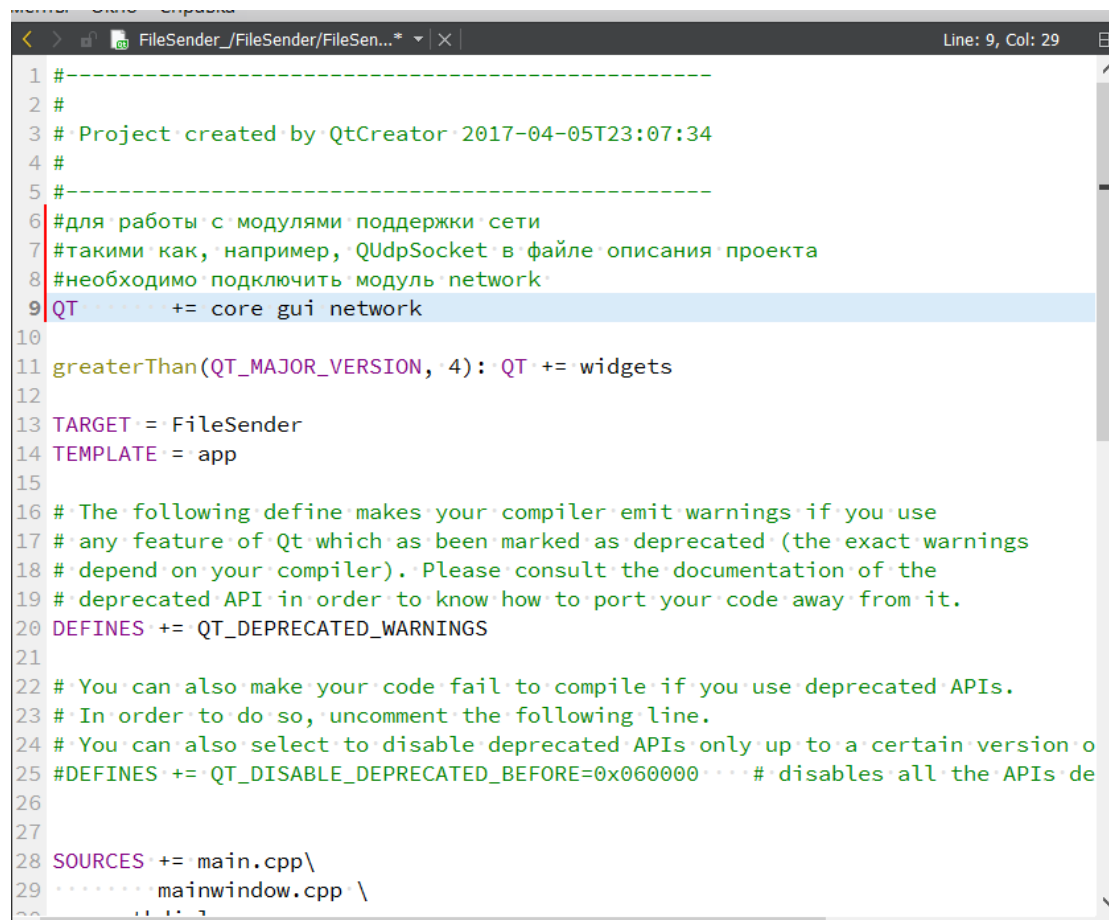
***Любое устройство, которое может отправлять и принимать IP-пакеты, называется IP-узлом (хостом, IP host).*

IP адрес

- Виды адресов:
- Серые / Белые
 - 192.168.0.0/16
 - 10.0.0.0/8
 - 172.16.0.0/12
 - 127.0.0.0/8 - loopback
- Unicast
- Broadcast – последний адрес в сети.
- Multicast

Поддержка сети. QUdpSocket.

Начало работы с классом QUdpSocket. Подключение модуля network.



The screenshot shows a Qt Creator IDE window with a .pro file open. The file contains configuration for a project named 'FileSender'. The configuration includes comments in Russian explaining the need to add the 'network' module for network support. The line 'QT += core gui network' is highlighted in blue. Other configurations include 'greaterThan(QT_MAJOR_VERSION, 4): QT += widgets', 'TARGET = FileSender', 'TEMPLATE = app', and 'DEFINES += QT_DEPRECATED_WARNINGS'. The 'SOURCES' section lists 'main.cpp' and 'mainwindow.cpp'.

```
1 #-----
2 #
3 # Project created by QtCreator 2017-04-05T23:07:34
4 #
5 #-----
6 #для работы с модулями поддержки сети
7 #такими как, например, QUdpSocket в файле описания проекта
8 #необходимо подключить модуль network
9 QT += core gui network
10
11 greaterThan(QT_MAJOR_VERSION, 4): QT += widgets
12
13 TARGET = FileSender
14 TEMPLATE = app
15
16 # The following define makes your compiler emit warnings if you use
17 # any feature of Qt which has been marked as deprecated (the exact warnings
18 # depend on your compiler). Please consult the documentation of the
19 # deprecated API in order to know how to port your code away from it.
20 DEFINES += QT_DEPRECATED_WARNINGS
21
22 # You can also make your code fail to compile if you use deprecated APIs.
23 # In order to do so, uncomment the following line.
24 # You can also select to disable deprecated APIs only up to a certain version of
25 # Qt.
26 #DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000 # disables all the APIs deprecated before Qt 6.0.0
27
28 SOURCES += main.cpp\
29             mainwindow.cpp \
30             ...
```

Поддержка сети. QUdpSocket.

Реализация UDP-сервера и UDP-клиента:

1. Создать объект класса QUdpSocket;
2. Соединить с IP и портом методом bind();
3. Если на порт поступает датаграмма, то происходит отправка сигнала readyRead(), который вы как раз можете связать с вашим слотом приема данных по сети...
4. Считать принятую датаграмму можно методом readDatagram();
5. Записать датаграмму можно методом writeDatagram();

Практическая часть

Дополним ранее написанный проект работы с файлом следующими функциональными возможностями:

1. Передача открытого файла по сети;
2. Приём переданного файла по сети;
3. Чтение инициализационных данных (IP адрес, порт) из конфигурационного файла (для отправителя эта часть кода прописана, для получателя придётся прописывать самим);
4. Вывод диагностической информации пользователю (какой ip/порт используется, сколько байт отправлено/принято)

Практическая часть

Отправитель:

SenderWidget

Открыть Отправить

Patents

MathWorks products are covered by one or more patents.

MATLAB®, MATLAB® Compiler, and other MATLAB family products

U.S. Patents:

6,857,118; 6,973,644; 6,993,772; 7,010,364; 7,051,333; 7,051,338; 7,096,154; 7,139,686; 7,165,253; 7,170,433; 7,181,745; 7,228,239; 7,231,631; 7,237,237; 7,296,054; 7,319,941; 7,340,441; 7,353,502; 7,359,805; 7,365,311; 7,369,070; 7,369,127; 7,400,997; 7,420,486; 7,428,737; 7,454,659; 7,454,746; 7,460,123; 7,500,220; 7,502,031; 7,502,745; 7,523,023; 7,523,440; 7,529,652; 7,542,888; 7,543,270; 7,558,712; 7,584,452; 7,605,814; 7,606,780; 7,606,833; 7,609,192; 7,610,578; 7,613,852; 7,624,372; 7,631,168; 7,634,530; 7,636,887; 7,636,914; 7,640,154; 7,647,213; 7,660,773; 7,664,720; 7,672,792; 7,672,910; 7,673,289; 7,680,637; 7,685,596; 7,690,004; 7,725,679; 7,725,883; 7,725,904; 7,730,166; 7,738,978; 7,739,283; 7,739,676; 7,743,087; 7,752,005; 7,752,138; 7,752,601; 7,752,636; 7,765,561; 7,769,576; 7,802,268; 7,805,466; 7,809,988; 7,823,168; 7,840,938; 7,844,431; 7,849,359; 7,849,470; 7,865,254; 7,873,502; 7,886,307; 7,890,198; 7,904,410; 7,908,313; 7,908,591; 7,921,406; 7,924,294; 7,925,791; 7,933,908; 7,958,490; 7,966,622; 7,975,001; 7,984,416; 7,987,227; 7,996,255; 8,006,313; 8,010,954; 8,032,582; 8,037,481; 8,041,790; 8,046,201; 8,046,736; 8,046,749; 8,046,777; 8,051,105; 8,051,408; 8,055,483; 8,055,598; 8,055,940; 8,056,053; 8,056,055; 8,056,079; 8,060,346; 8,060,421; 8,069,178; 8,108,717; 8,108,845; 8,112,741; 8,127,311; 8,130,239; 8,141,034; 8,156,481; 8,156,493; 8,161,402; 8,165,413; 8,171,027; 8,180,964; 8,185,556; 8,185,868; 8,190,400; 8,200,559; 8,200,560; 8,209,350; 8,209,419; 8,219,378; 8,219,476; 8,225,300; 8,230,424; 8,230,427; 8,234,098; 8,234,623; 8,239,844; 8,239,845; 8,239,846; 8,250,550; 8,255,889; 8,255,890; 8,260,602; 8,260,791; 8,265,916; 8,279,204; 8,280,661; 8,290,892; 8,294,704; 8,296,070; 8,300,060; 8,307,328; 8,312,423; 8,314,813; 8,321,847; 8,321,848; 8,326,750; 8,327,029; 8,346,526; 8,346,793; 8,359,586; 8,365,088; 8,365,144; 8,370,531; 8,380,880; 8,392,467; 8,397,224; 8,402,317; 8,418,158; 8,423,326; 8,433,953; 8,446,425; 8,458,668; 8,458,675; 8,462,146; 8,464,226; 8,467,888; 8,473,901; 8,484,262; 8,489,382; 8,495,114; 8,510,266; 8,514,247; 8,521,438; 8,527,042; 8,527,072; 8,527,402; 8,527,607; 8,528,185; 8,542,939;

D:/patents.txt

Параметры сетевого соединения:

Используемый IP:Port 127.0.0.5:1002

Отправлено (байт) 12328 Получатель: IP:Port 127.0.0.5:1003

Получатель:

Form

Patents

MathWorks products are covered by one or more patents.

MATLAB®, MATLAB® Compiler, and other MATLAB family products

U.S. Patents:

6,857,118; 6,973,644; 6,993,772; 7,010,364; 7,051,333; 7,051,338; 7,096,154; 7,139,686; 7,165,253; 7,170,433; 7,181,745; 7,228,239; 7,231,631; 7,237,237; 7,296,054; 7,319,941; 7,340,441; 7,353,502; 7,359,805; 7,365,311; 7,369,070; 7,369,127; 7,400,997; 7,420,486; 7,428,737; 7,454,659; 7,454,746; 7,460,123; 7,500,220; 7,502,031; 7,502,745; 7,523,023; 7,523,440; 7,529,652; 7,542,888; 7,543,270; 7,558,712; 7,584,452; 7,605,814; 7,606,780; 7,606,833; 7,609,192; 7,610,578; 7,613,852; 7,624,372; 7,631,168; 7,634,530; 7,636,887; 7,636,914; 7,640,154; 7,647,213; 7,660,773; 7,664,720; 7,672,792; 7,672,910; 7,673,289; 7,680,637; 7,685,596; 7,690,004; 7,725,679; 7,725,883; 7,725,904; 7,730,166; 7,738,978; 7,739,283; 7,739,676; 7,743,087; 7,752,005; 7,752,138; 7,752,601; 7,752,636; 7,765,561; 7,769,576; 7,802,268; 7,805,466; 7,809,988; 7,823,168; 7,840,938; 7,844,431; 7,849,359; 7,849,470; 7,865,254; 7,873,502; 7,886,307; 7,890,198; 7,904,410; 7,908,313; 7,908,591; 7,921,406; 7,924,294; 7,925,791; 7,933,908; 7,958,490; 7,966,622; 7,975,001; 7,984,416; 7,987,227; 7,996,255; 8,006,313; 8,010,954; 8,032,582; 8,037,481; 8,041,790; 8,046,201; 8,046,736; 8,046,749; 8,046,777; 8,051,105; 8,051,408; 8,055,483; 8,055,598; 8,055,940; 8,056,053; 8,056,055; 8,056,079; 8,060,346; 8,060,421; 8,069,178; 8,108,717; 8,108,845; 8,112,741; 8,127,311; 8,130,239; 8,141,034; 8,156,481; 8,156,493; 8,161,402; 8,165,413; 8,171,027; 8,180,964; 8,185,556; 8,185,868; 8,190,400; 8,200,559; 8,200,560; 8,209,350; 8,209,419; 8,219,378; 8,219,476; 8,225,300; 8,230,424; 8,230,427; 8,234,098; 8,234,623; 8,239,844; 8,239,845; 8,239,846; 8,250,550; 8,255,889; 8,255,890; 8,260,602; 8,260,791; 8,265,916; 8,279,204; 8,280,661; 8,290,892; 8,294,704; 8,296,070; 8,300,060; 8,307,328; 8,312,423; 8,314,813; 8,321,847; 8,321,848; 8,326,750; 8,327,029; 8,346,526; 8,346,793; 8,359,586; 8,365,088; 8,365,144; 8,370,531; 8,380,880; 8,392,467; 8,397,224; 8,402,317; 8,418,158; 8,423,326; 8,433,953; 8,446,425; 8,458,668; 8,458,675; 8,462,146; 8,464,226; 8,467,888; 8,473,901; 8,484,262; 8,489,382; 8,495,114; 8,510,366; 8,514,247; 8,521,438; 8,527,942; 8,527,973; 8,533,402; 8,533,697; 8,538,185; 8,543,939; 8,548,744; 8,549,096; 8,549,500; 8,561,077; 8,566,375; 8,594,217; 8,595,439; 8,605,109; 8,606,375; 8,612,980; 8,619,090; 8,621,425; 8,627,282; 8,631,392; 8,650,542; 8,654,138; 8,670,001; 8,671,394;

Параметры сетевого соединения:

Используемый IP:Port 127.0.0.5:1003

Принято Received: 12328 Отправитель: IP:Port 127.0.0.5:1002

Практическая часть

В проекте FileSender:

`senderwidget.h` и `senderwidget.cpp`, `senderwidget.ui` – описывают класс формы отправителя файла; (из предыдущего занятия по работе с файлами, но немного надо будет дописать)

`dialog.h`, `dialog.cpp`, `dialog.ui` – описывают класс диалогового окна ввода пути к файлу; (из предыдущего занятия по работе с файлами)

`receiverwidget.h`, `receiverwidget.cpp`, `receiverwidget.ui` – описывают класс формы получателя файла.

Практическая часть

main.cpp

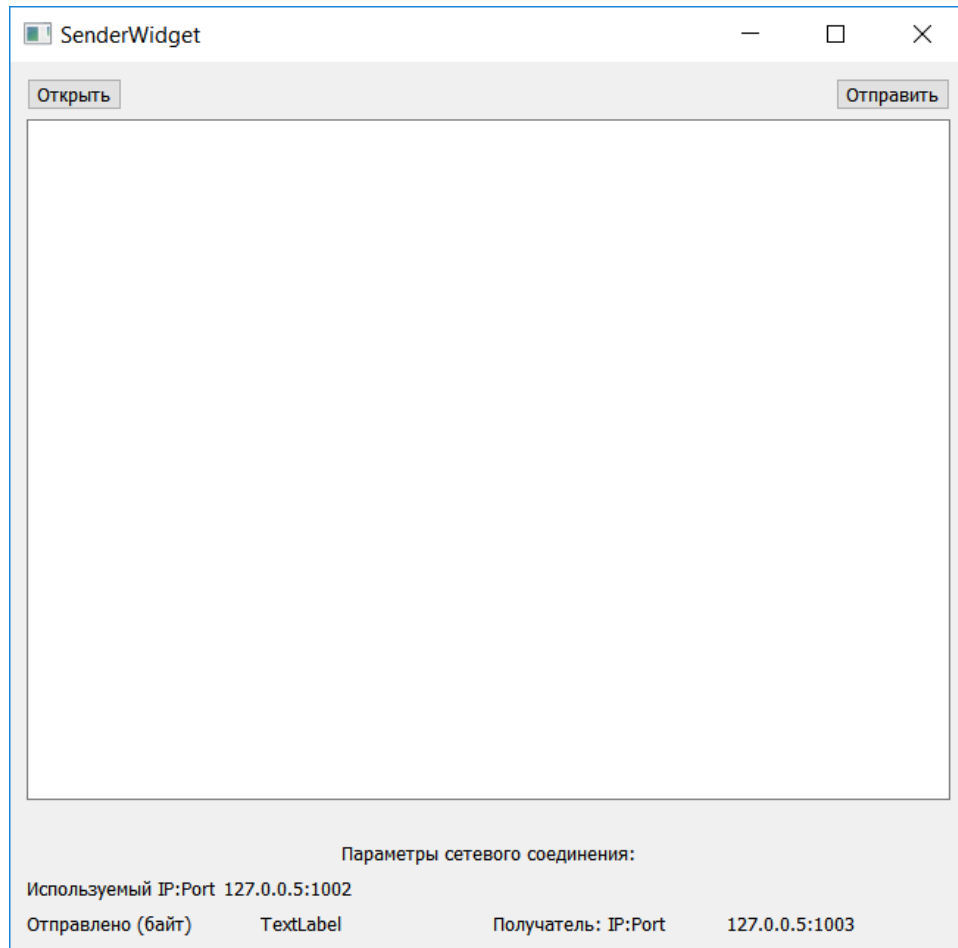
```
1 #include "senderwidget.h"
2 #include "receiverwidget.h"
3 #include <QApplication>
4
5 int main(int argc, char *argv[])
6 {
7     QApplication a(argc, argv);
8     SenderWidget w; //объект класса формы отправителя
9     ReceiverWidget r; //объект класса формы получателя
10    w.show();
11    r.show();
12
13    return a.exec();
14 }
15
```

Поправив main.cpp скомпилируйте проект.

*Для успешной работы проекта необходимо, чтобы файл config.json находился в директории сборки проекта

Практическая часть

Промежуточный результат:



SenderWidget

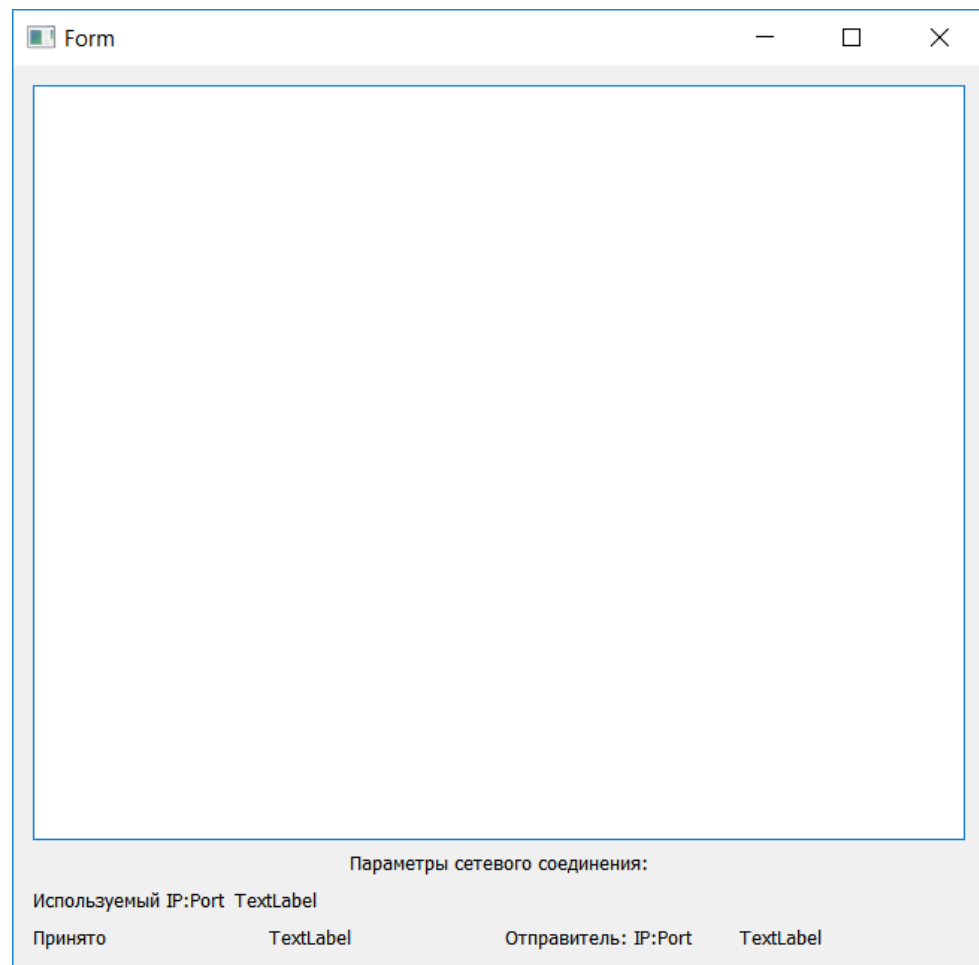
Открыть Отправить

Параметры сетевого соединения:

Используемый IP:Port 127.0.0.5:1002

Отправлено (байт) TextLabel Получатель: IP:Port 127.0.0.5:1003

The SenderWidget window features a title bar with standard window controls. Below the title bar, there are two buttons: 'Открыть' (Open) on the left and 'Отправить' (Send) on the right. The main area of the window is a large, empty rectangular box. At the bottom, a status bar displays network connection parameters. It includes a label 'Параметры сетевого соединения:' followed by 'Используемый IP:Port 127.0.0.5:1002'. Below this, there are four fields: 'Отправлено (байт)' (Sent (bytes)) with a 'TextLabel' placeholder, 'Получатель: IP:Port' (Receiver: IP:Port), and the value '127.0.0.5:1003'.



Form

Параметры сетевого соединения:

Используемый IP:Port TextLabel

Принято TextLabel Отправитель: IP:Port TextLabel

The Form window has a title bar with standard window controls. Below the title bar is a large, empty rectangular box. At the bottom, a status bar displays network connection parameters. It includes a label 'Параметры сетевого соединения:' followed by 'Используемый IP:Port TextLabel'. Below this, there are four fields: 'Принято' (Received) with a 'TextLabel' placeholder, 'Отправитель: IP:Port' (Sender: IP:Port), and another 'TextLabel' placeholder.

Практическая часть

Senderwidget.h

```
senderwidget.h*  ▾ | × | #  🏠 sented: int
1  #ifndef SENDERWIDGET_H
2  #define SENDERWIDGET_H
3
4  #include <QString>
5  #include <QWidget>
6  #include <QUdpSocket>
7  #include "ui_senderwidget.h"
8  #include "dialog.h"
9
10 class SenderWidget : public QWidget, Ui::SenderWidget {
11     ... Q_OBJECT
12 public:
13     ... explicit SenderWidget(QWidget *parent = 0);
14     ... ~SenderWidget();
15 private slots:
16     ... void on_tbtnOpen_clicked();
17     ... void receivePath(const QString &path);
18     ... void on_tbtnSend_clicked(); // слот, который по нажатию кнопки
19     ... // отправит файл получателю
20 private:
21     ... QUdpSocket *udp; // указатель на объект сокета
22     ... QString PATH; // путь к файлу (прошрое занятие)
23     ... QHostAddress ip, receiverIP; // переменные для работы с IP-адресами
24     ... int port, receiverPort; // номера портов отправителя и получателя
25     ... int sented; // переменная для хранения количества отправленных байт
26 };
27 #endif // SENDERWIDGET_H
```

Практическая часть

В приведенной части кода происходит открытие конфигурационного файла и считывания из него IP-адресов и портов отправителя и получателя по сети. Эти данные понадобятся для того, чтобы выполнить метод `bind()` для сокета и отправить файл.

Для парсинга конфигурационного файла используются классы для работы с JSON-протоколом. (см. предыдущее занятие)

```
senderwidget.cpp*  <X> # <Выберите символ>  Line: 6, Col: 1
1 #include "senderwidget.h"
2 #include "ui_senderwidget.h"
3 #include <QFile>
4 #include <QJsonObject>
5 #include <QJsonDocument>
6
7 SenderWidget::SenderWidget(QWidget *parent) :
8     QWidget(parent) {
9     setupUi(this);
10    // прочитаем конфигурационные данные из файла
11    QFile configFile("config.json");
12    QJsonDocument jsonDoc;
13    if (configFile.exists()) { // проверяем существование файла
14        // если файл существует, то открываем его для чтения и работаем
15        configFile.open(QFile::ReadOnly);
16        // считаем конфигурационные данные из файла
17        // используя метод fromJson, который преобразует текст (QByteArray)
18        // в формат JSON
19        jsonDoc = QJsonDocument().fromJson(configFile.readAll());
20        configFile.close();
21        // создадим JSON-объект, в который считаем содержимое jsonDoc
22        QJsonObject obj = jsonDoc.object();
23        // считаем из объекта IP-адрес и порт отправителя файла:
24        ip.setAddress(obj.value("sender.ip").toString());
25        port = obj.value("sender.port").toInt();
26        // считаем IP-адрес и порт получателя файла:
27        receiverIP.setAddress(obj.value("receiver.ip").toString());
28        receiverPort = obj.value("receiver.port").toInt();
29        // выведем полученные значения в окно приложения
30        lblReceiverIPport->setText(receiverIP.toString()+" "+QString::number(receiverPort));
31        lblSenderIPport->setText(ip.toString()+" "+QString::number(port));
32    }
```

senderwidget.cpp

```

SenderWidget.cpp
SenderWidget::on_tbtnSend_clicked(): void

... //создадим UDP-сокет
... udp = new QUdpSocket();

... //забиндим его по адресу и порту, считанным из config-файла
... udp->bind(ip, port);

... PATH="";
... //инициализация переменной, содержащей количество отправленных байт
... sented=0;
}

//синтаксис функции таков, что сигнал clicked() кнопки
//tbtnSend автоматически соединен с нижеследующим слотом
void SenderWidget::on_tbtnSend_clicked() {
... //создадим буфер для хранения пересылаемых по сети данных
... QByteArray baDatagram;
... //создадим объект файла, который хотим переслать
... QFile file(PATH);
... if(!file.exists()) qDebug() << "The file doesn't exists";
... else {
...     file.open(QFile::ReadOnly);
...     //метод readAll() возвращает содержимое файла в формате QString
...     //что мы и запишем в наш буфер
...     baDatagram=file.readAll();
...     //методом writeDatagram отправим считанные значения
...     sented += udp->writeDatagram(baDatagram, receiverIP, receiverPort);
...     qDebug() << "Sended: " << sented << "bytes";
...     lblSended->setText(QString::number(sended));
... }
}

```

Практическая часть.

senderwidget.cpp

```
senderwidget.cpp*  SenderWidget::on_tbtnSend_clicked(): void
... //создадим UDP-сокет
... udp = new QUdpSocket ();

... //забиндим его по адресу и порту, считанным из config-файла
... udp->bind(ip,port);

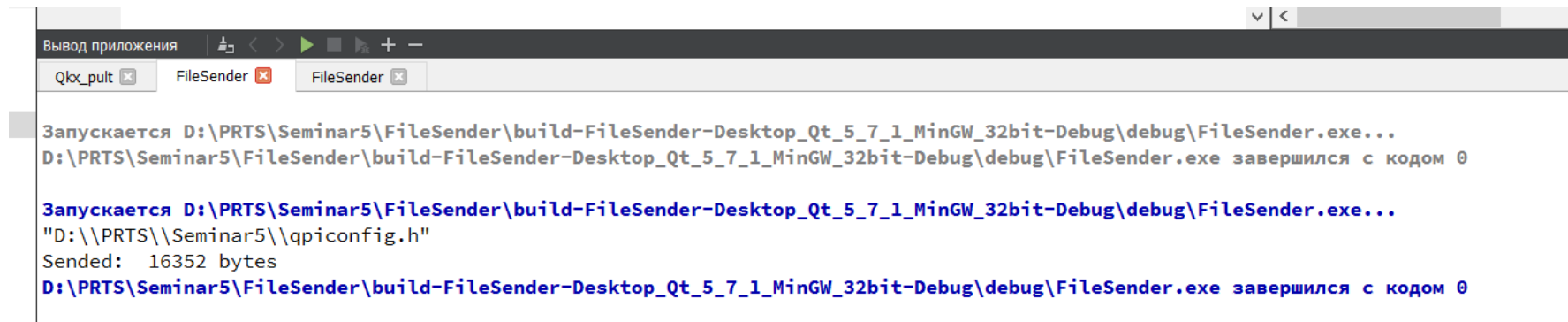
... PATH="";
... //инициализация переменной, содержащей количество отправленных байт
... send=0;
}

//синтаксис функции таков, что сигнал clicked() кнопки
//tbtnSend автоматически соединен с нижеследующим слотом
void SenderWidget::on_tbtnSend_clicked() {
... //создадим буфер для хранения пересылаемых по сети данных
... QByteArray baDatagram;
... //создадим объект файла, который хотим переслать
... QFile file (PATH);
... if (!file.exists()) qDebug() << "The file doesn't exists";
... else {
...     file.open(QFile::ReadOnly);
...     //метод readAll() возвращает содержимое файла в формате QString
...     //что мы и запишем в наш буфер
...     baDatagram=file.readAll();
...     //методом writeDatagram отправим считанные значения
...     send += udp->writeDatagram(baDatagram,receiverIP, receiverPort);
...     qDebug() << "Send: " << send << "bytes";
...     lblSend->setText(QString::number(send));
... }
}
```

Практическая часть

Насладимся промежуточным результатом!)

Отправим файл и посмотрим вывод приложения:



```
Вывод приложения
Qlox_pult FileSender FileSender

Запускается D:\PRTS\Seminar5\FileSender\build-FileSender-Desktop_Qt_5_7_1_MinGW_32bit-Debug\debug\FileSender.exe...
D:\PRTS\Seminar5\FileSender\build-FileSender-Desktop_Qt_5_7_1_MinGW_32bit-Debug\debug\FileSender.exe завершился с кодом 0

Запускается D:\PRTS\Seminar5\FileSender\build-FileSender-Desktop_Qt_5_7_1_MinGW_32bit-Debug\debug\FileSender.exe...
"D:\\PRTS\\Seminar5\\qpiconfig.h"
Sended: 16352 bytes
D:\PRTS\Seminar5\FileSender\build-FileSender-Desktop_Qt_5_7_1_MinGW_32bit-Debug\debug\FileSender.exe завершился с кодом 0
```

Если в графе Sended: записано нужное количество байт, переходим к следующему этапу..

Практическая часть

Receiverwidget.h

```
receiverwidget.h
socketReceived(): void

1  #ifndef RECEIVERWIDGET_H
2  #define RECEIVERWIDGET_H
3
4  #include <QWidget>
5  #include <QHostAddress>
6  #include <QUdpSocket>
7
8  #include "ui_receiverwidget.h"
9
10 class ReceiverWidget : public QWidget, Ui::ReceiverWidget {
11     ... Q_OBJECT
12 public:
13     ... explicit ReceiverWidget(QWidget *parent = 0);
14     ... ~ReceiverWidget();
15 private:
16     ... QUdpSocket *udpSocket; // сокет для приёма данных
17     ... // переменные для работы с IP-адресами
18     ... QHostAddress senderIP, receiverIP;
19     ... int senderPort, receiverPort;
20     ... int received; // переменная для хранения количества отправленных байт
21 private slots:
22     ... void socketReceived(); // слот для обработки принятых данных
23 };
24
25 #endif // RECEIVERWIDGET_H
```


Практическая часть

Receiverwidget.cpp

Алгоритм действий:

1. считать из config.json файла ip адрес и порт, на которых мы будем принимать и обрабатывать датаграммы (это надо сделать самостоятельно по аналогии с senderwidget)
2. создать объект QUdpSocket, который предоставит нам сокет для получения данных по сети.
3. Создать слот для приема приходящих датаграмм socketReceived().

```
receiverwidget.cpp*  ReceiverWidget::ReceiverWidget(QWidget *)  Line: 49, Col: 1
... //создаем объект сокет для нашего клиента
... udpSocket = new QUdpSocket(this);
... //биндим сокет к IP и порту на которых будем "слушать" сообщения
... udpSocket->bind(receiverIP, receiverPort);
... //чтобы осуществить чтение из порта не в цикле ожидания
... //а по мере приема полезных данных... соединим сигнал и слот
... //сигнал - readyRead - отправляется тогда, когда
... //данные готовы для чтения
... connect(udpSocket, SIGNAL(readyRead()),
...         this, SLOT(socketReceived()));
}

void ReceiverWidget::socketReceived(){
    ... QNetworkDatagram networkDatagram;
    ... do{
        ... networkDatagram=udpSocket->receiveDatagram();
        ... }while(udpSocket->hasPendingDatagrams());

    ... txtBrRec->setText(networkDatagram.data());
    ... senderIP=networkDatagram.senderAddress();
    ... senderPort=networkDatagram.senderPort();
    ... lblSenderIPport->setText(senderIP.toString()+":"+QString::number(senderPort));
    ... lblReceived->setText("Received: "+QString::number(networkDatagram.data().size()));
}
```

Практическая часть

- Компилируем и наслаждаемся работающей программой!)