

# Networking, testing and Qt

Miroslav Jaroš

Penguins collective

21. 10. 2014

- 1 Network
  - Network basics
  - Network in Qt
  - TCP
  - QTcpSocket
  - UDP

# Contents

## 1 Network

- Network basics
- Network in Qt
- TCP
- QTcpSocket
- UDP

## 2 Testing and Subdirs

# Networking

Aka – A nightmare on Elm street

Network is:

- Unreliable
- Nondeterministic
- Complicated
- Managed by OS
- Basicaly Hell

Remember:

Network is the best way for computer communication. We haven't sill find anything better.

# Networking – basics

What you should already know

- Network is very heterogeneous environment
- Operations over network are divided into 7 abstract layers (ISO/OSI)
- Internet is using TCP/IP protocol suite
- IP – network addressing, defines structure on 3rd "network" layer
- TCP and UDP – works on 4th "transport" layer and are responsible for Data transportation

# Qt networking

What you must have before you start:

- Design of your communication
- TCP or UDP
- Client – Server?
- Client – Client?

## Warning

Your Client2Client communication won't probably work if you'll use TCP

# Qt classes

Qt as framework has whole Network library. To use it you must add:

```
QT += network
```

Then comes the programming part.

- For communication are used Sockets
- Socket is data stream, like file or pipe
- Data are send over network and delivered in same format as they were send.
  - If they are delivered (UDP)
- All sockets has one parent QAbstractSocket

- For communication between two endpoints are mainly two classes
  - QTcpSocket
  - QUdpSocket
- For Client – Server QTcpServer



## Joke

Do you want to hear TCP joke? Yes I do want to hear TCP joke. Ok, I will send you a TCP joke which will be 1024 bytes long and....

TCP is reliable connection, which means data are delivered to receiver complete and in proper order. Although this is nice, there are several disadvantages.

- Only one application can use port at time.
- There is quite a lot of information delivered between peers to ensure reliability
- Every connection is defined as quadruple (Addr1, port1, Addr2, port2)
- Because of these, it's almost impossible to open connection between peers behind NAT (over network)

For TCP connections Qt has two main classes `QTcpSocket` and `QTcpServer`

Their usage is quite obvious. `QTcpSocket` is used for direct communication between peers and `QTcpServer` is used as incoming connections handler.

# QTcpServer

```
class ServerListener : public QTcpServer
{
    Q_OBJECT
public:
    explicit ServerListener(QObject *parent = 0);

    void start();

signals:

public slots:

protected:
    void incomingConnection(qintptr handle);

private:
    SharedList * list;
    SqlConnection * database;
};
```

```
ServerListener::ServerListener(QObject *parent) :
    QTcpServer(parent), list(0)
{
}
```

```
void ServerListener::start()
{
    list = new SharedList;
    database = new SqlConnection;
    if(!this->listen(QHostAddress::Any, 27173))
    {
    }
}
```

```
void ServerListener::incomingConnection(qintptr handle)
{
    ServerThread * thread = new ServerThread(handle,
                                                list, database);

    connect(thread, SIGNAL(finished()),
            thread, SLOT(deleteLater()));

    thread->start();
}
```

# QTcpServer

Class `ServerListener` is derived from `QTcpServer` and overrides its basic method: `void QTcpServer::incomingConnection(qintptr handle)`

Class starts listening after invoking method `void start()` which invokes method `void QTcpServer::listen(QHostAddress, qintptr)`

This method starts infinite loop, in which it waits for connection.

When new connection appears, the `incomingConnection` method is invoked, with `handle` parameter, which is socket descriptor (like file descriptor, number which points to existing socket in kernel space)

## Note

Remember `QTcpServer` is not meant to communicate, it should be running in its very own thread, and do only basic operations, like sending new descriptor to handler thread and so on.

# QTcpSocket

- Is derived from QIODevice and has it's methods
- Sends and recieves data as QDataStream
- When data arrives, the void readyRead() signal is invoked
- You also can read data as stream directly, but this is a bit dangerous
- New socket is able to communicate when:

- Is set it's descriptor:

```
virtual bool setSocketDescriptor(qintptr socketDescriptor,  
    SocketState socketState = ConnectedState,  
    OpenMode openMode = ReadWrite)
```

- Is initialized communication to selected Address and port:  
virtual void connectToHost(const QHostAddress & address,  
 quint16 port, OpenMode openMode = ReadWrite)

# UDP

## Joke

I know a good joke about UDP, but you might not get it.

UDP is another transport protocol.

Main properties:

- Not connected service
- Some times not reliable
- packets can be lost or delivered in another order

But can be really good:

- Faster than TCP
- Not so spamming as TCP
- Doesn't have problem with things like NAT
- You can communitate on whichever port you like ;-)

# QUdpSocket

This is main class for UDP handling in your code. UDP does not use QByteArray, because it's not trying to behave like stream. QUdpSocket can be binded to port and address, so the Socket is capable of listening (obtaining data) from Any peer

```
bool QAbstractSocket::bind(const QHostAddress & address,  
quint16 port = 0, BindMode mode = DefaultForPlatform)
```

Sending data is very simmilar to QTcpSocket

# Qt Subdirs

Let's say we have a complicated project such as QtCreator.  
We'd like to separate it into smaller pieces to do better maintenance:

- Separated plugins
- Shared Libraries
- Tests

That's purpose of Subdirs Projects. The Subdirs projects handles every subproject as separate project which is somehow dependent on other ones. Than it creates main .pro file which is supposed to be main builder.