Lecture 9. Implementing Network Communication with Qt

Cross-Platform Application Development

November 24, 2017

Networking Support in Qt

Level	Classes
Lower	`QHostInfo`
	`QUdpSocket`
	`QTcpSocket`
	`QTcpServer`
	...
Higher	`QNetworkAccessManager`
	`QNetworkRequest`
	`QNetworkReply`
	...
Obsolete	`QFtp`
	`QHttp`
	...

Table 1: networking facilities overview

Part of Qt Classes Hierarchy

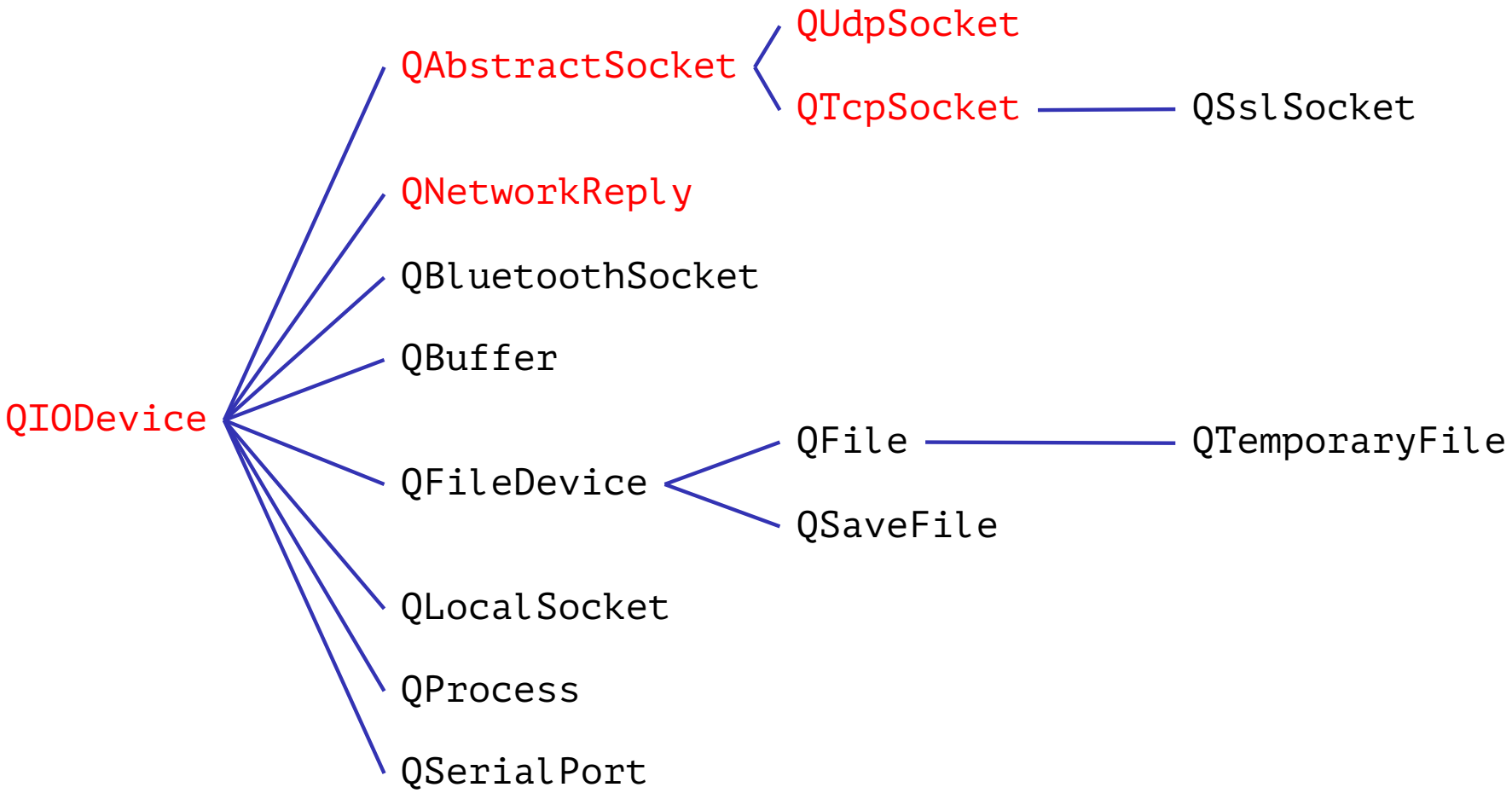

Figure 1: lower-level communications classes hierarchy

QUdpSocket Class

Methods (Reading/Writing)

- `bool hasPendingDatagrams() const;`
- `qint64 pendingDatagramSize() const;`
- `qint64 readDatagram(`
  `char *pchData, qint64 uMaxSize,`
  `QHostAddress *pAddress = Q_NULLPTR, quint16 *puPort = Q_NULLPTR);`
- `qint64 writeDatagram(`
  `const char *pcchData, qint64 uSize,`
  `const QHostAddress &rcAddress, quint16 uPort);`
- `qint64 writeDatagram(`
  `const QByteArray &rcDatagram,`
  `const QHostAddress &rcHost, quint16 uPort);`

QUdpSocket Class (end)

Methods (Multicast)

- **bool** joinMulticastGroup(
  const QHostAddress &rcGroupAddress);
- **bool** joinMulticastGroup(
  const QHostAddress &rcGroupAddress,
  QNetworkInterface &rInterface);
- **bool** leaveMulticastGroup(
  const QHostAddress &rcGroupAddress);
- **bool** leaveMulticastGroup(
  const QHostAddress &rcGroupAddress,
  QNetworkInterface &rInterface);

Example

Example (CMakeLists.txt)

```
find_package(Qt5 REQUIRED Widgets Network)

foreach(EXECUTABLE sender receiver)
  #
  add_executable(
    example-21-${EXECUTABLE}
    example-21-${EXECUTABLE}.cpp
    )
  #
  target_link_libraries(
    example-21-${EXECUTABLE} Qt5::Widgets Qt5::Network)
  #
endforeach()
```

Example (cont.)

Example (example-21-sender.cpp)

```
#include <QApplication>
#include <QUdpSocket>

int main(int nArgC, char *apszArgV[])
{
  QApplication app(nArgC, apszArgV);
  //
  QHostAddress address("127.0.0.1");    // QHostAddress::LocalHost
  QByteArray datagram = "Hello from Sender";
  QUdpSocket socket;
  socket.writeDatagram(
    datagram.data(), datagram.size(), address, 44556);
}    // main()
```

Example (cont.)

Example (example-21-receiver.cpp)

```
#include <QApplication>
#include <QUdpSocket>

#include <iostream>
#include <string>

using namespace std;

int main(int nArgC, char *apszArgV[])
{
  QApplication app(nArgC, apszArgV);
  //
```

Example (cont.)

Example (example-21-receiver.cpp, cont.)

```
QUdpSocket socket;
socket.bind(44556, QUdpSocket::ShareAddress);
//
while (!socket.waitForReadyRead(5000))
{
  string in;
  cout << "Wait more? (y/n) " << flush;
  cin >> in;
  if (in == "n")
    return 1;
}
//
```

Example (cont.)

Example (example-21-receiver.cpp, cont.)

```
while (socket.hasPendingDatagrams())
{
  QByteArray datagram;
  datagram.resize(socket.pendingDatagramSize());
  QHostAddress address;
  quint16 uPort;
  socket.readDatagram(
    datagram.data(), datagram.size(), &address, &uPort);
  //
```

Example (end)

Example (example-21-receiver.cpp, end)

```
    cout <<
      "From: " << address.toString().toStdString() <<
      ":" << uPort <<
      "\nData: " << datagram.data() <<
      endl;
  }    // while (socket.hasPendingDatagrams())
}    // main()
```

Example

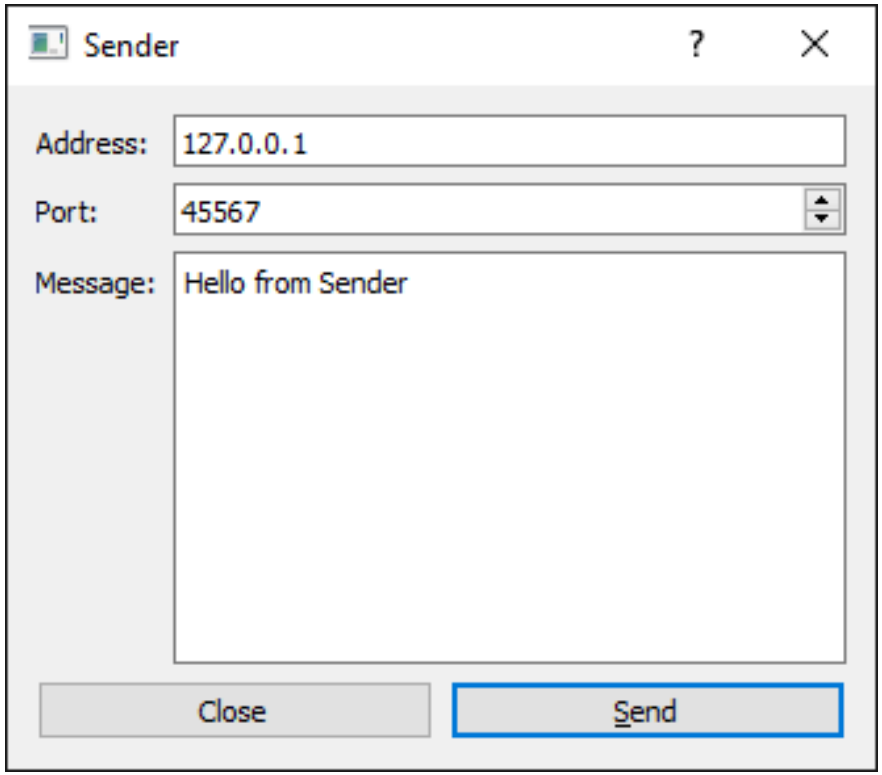

Figure 2: UDP sender program

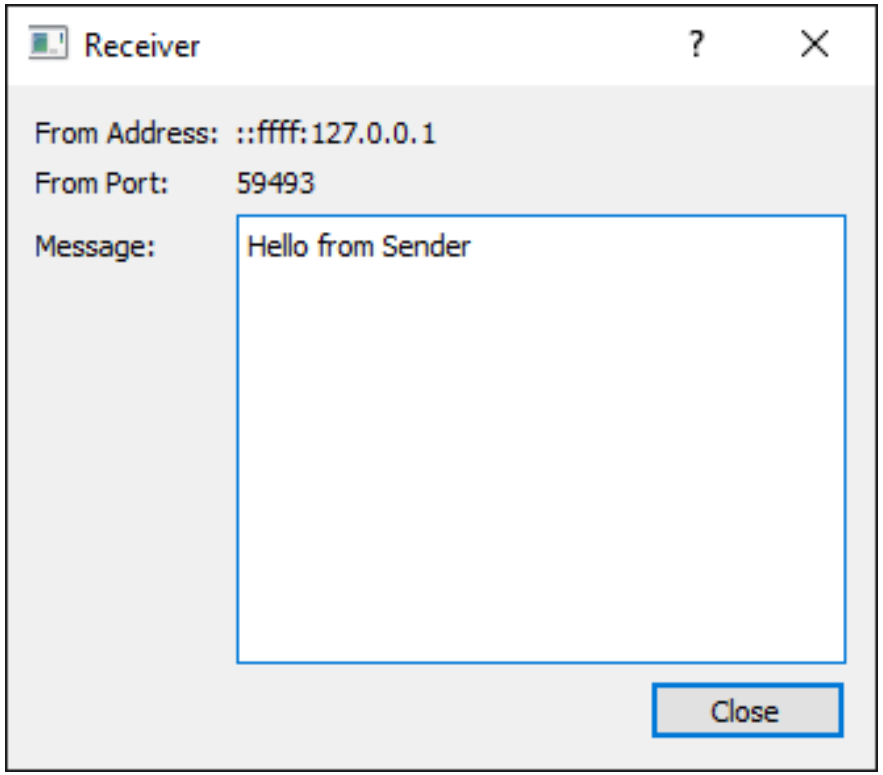

Figure 3: UDP receiver program

Example (cont.)

Example (sender-dlg.cpp)

```
#include "sender-dlg.h"

#include <QUdpSocket>

SenderDialog::SenderDialog()
{
  setupUi(this);
  //
  connect(
    m_pButtonClose, SIGNAL(clicked()), this, SLOT(close()));
  //
  m_pSocket = new QUdpSocket(this);
}
```

Example (cont.)

Example (sender-dlg.cpp, end)

```
void SenderDialog::on_buttonSend_clicked()
{
  QHostAddress address(m_pEditAddress->text());
  quint16 uPort = static_cast <quint16> (m_pSpinboxPort->value());
  QByteArray datagram;
  datagram += m_pTextEditMessage->toPlainText();
  m_pSocket->writeDatagram(
    datagram.data(), datagram.size(), address, uPort);
}
```

Example (cont.)

Example (receiver-dlg.cpp)

```
#include "receiver-dlg.h"

#include <QUdpSocket>

ReceiverDialog::ReceiverDialog()
{
  setupUi(this);
  //
  connect(
    m_pButtonClose, SIGNAL(clicked()), this, SLOT(close()));
  //
```

Example (cont.)

Example (receiver-dlg.cpp, cont.)

```
  m_pSocket = new QUdpSocket(this);
  m_pSocket->bind(45567);    // QAbstractSocket::DefaultForPlatform
  connect(
    m_pSocket, SIGNAL(readyRead()),
    this, SLOT(processPendingDatagrams()));
}    // ReceiverDialog()
```

Example (cont.)

Example (receiver-dlg.cpp, cont.)

```
void ReceiverDialog::processPendingDatagrams()
{
  while (m_pSocket->hasPendingDatagrams())
  {
    QByteArray datagram;
    datagram.resize(m_pSocket->pendingDatagramSize());
    QHostAddress address;
    quint16 uPort;
    m_pSocket->readDatagram(
      datagram.data(), datagram.size(), &address, &uPort);
    //
```

Example (end)

Example (receiver-dlg.cpp, end)

```
    m_pLabelAddress->setText(address.toString());
    m_pLabelPort->setText(QString::number(uPort));
    QString previous = m_pTextEditMessage->toPlainText();
    if (!previous.isEmpty())
      previous += '\n';
    //
    m_pTextEditMessage->setPlainText(
      previous + QString(datagram.toStdString().c_str()));
  }    // while (m_pSocket->hasPendingDatagrams())
}    // processPendingDatagrams()
```

QTcpServer Class

Methods (Basic Operations)

- **bool** listen(
  const QHostAddress &rcAddress = QHostAddress::Any,
  quint16 uPort = 0);
- **virtual** QTcpSocket *nextPendingConnection();
- **bool** waitForNewConnection(
  int nMSec = 0, **bool** *pbTimedOut = Q_NULLPTR);
- **void** pauseAccepting();
- **void** resumeAccepting();

QTcpServer Class (cont.)

Methods (properties)

- `bool isListening() const;`
- `QHostAddress serverAddress() const;`
- `quint16 serverPort() const;`
- `QAbstractSocket::SocketError serverError() const;`
- `QString errorString() const;`
- `int maxPendingConnections() const;/`
  `void setMaxPendingConnections(int nConnections);`
- `QNetworkProxy proxy() const;/`
  `void setProxy(const QNetworkProxy &networkProxy);`

QTcpServer Class (end)

Methods (Signals)

- **void** acceptError(QAbstractSocket::SocketError nSocketError);
- **void** newConnection();

QAbstractSocket Class

Methods (Basic Operations)

- **bool** bind(
  const QHostAddress &rcAddress, quint16 uPort = 0,
  BindMode nMode = DefaultForPlatform);

ShareAddress	DontShareAddress
ReuseAddressHint	DefaultForPlatform

Table 2: flags for `QAbstractSocket::BindMode`

QAbstractSocket Class (cont.)

Methods (Basic Operations)

- **virtual void** connectToHost(
  const QHostAddress &rcAddress, quint16 uPort,
  OpenMode nOpenMode = ReadWrite);

NotOpen	ReadOnly	WriteOnly	ReadWrite
Append	Truncate	Text	

Table 3: flags for QIODevice::OpenMode

QAbstractSocket Class (cont.)

Methods (Basic Operations)

- **virtual void** connectToHost(
  const QString &rcHostName, quint16 uPort,
  OpenMode nOpenMode = ReadWrite,
  NetworkLayerProtocol nProtocol = AnyIPProtocol);

IPv4Protocol	IPv6Protocol
AnyIPProtocol	UnknownNetworkLayerProtocol

Table 4: values for QAbstractSocket::NetworkLayerProtocol

QAbstractSocket Class (cont.)

Methods (Basic Operations)

- `virtual void disconnectFromHost();`
- `virtual bool waitForReadyRead(int nMSecs = 30000);`
- `virtual bool waitForBytesWritten(int nMSecs = 30000);`

QAbstractSocket Class (cont.)

Methods (Properties)

- `bool isValid() const;`
- `SocketState state() const;`
- `QHostAddress localAddress() const;`
- `quint16 localPort() const;`
- `QHostAddress peerAddress() const;`
- `quint16 peerPort() const;`
- `QNetworkProxy proxy() const;/`
  `void setProxy(const QNetworkProxy &rcNetworkProxy);`

UnconnectedState	HostLookupState	ConnectingState
ConnectedState	BoundState	ClosingState

Table 5: values for `QAbstractSocket::SocketState`

QAbstractSocket Class (end)

Methods (Signals)

- `void connected();`
- `void disconnected();`
- `void error(QAbstractSocket::SocketError nSocketError);`
- `void hostFound();`
- `void stateChanged(QAbstractSocket::SocketState nSocketState);`
- `void proxyAuthenticationRequired(const QNetworkProxy &pcProxy, QAuthenticator *pAuthenticator);`

QIODevice Class

Methods (Basic Operations)

- `qint64 read(char *pchData, qint64 nMaxSize);`
- `QByteArray read(qint64 nMaxSize);`
- `QByteArray readAll();`
- `qint64 readLine(char *pchData, qint64 uMaxSize);`
- `QByteArray readLine(qint64 uMaxSize = 0);`
- `qint64 write(const char *pchData, qint64 uMaxSize);`
- `qint64 write(const char *pchData);`
- `qint64 write(const QByteArray &rcByteArray);`

QDataStream Class

Methods (Basic Operations)

- `QDataStream(QIODevice *rDevice);`
- `void startTransaction();`
- `bool commitTransaction();`
- `int readRawData(char *pszData, int nLen);`
- `int writeRawData(const char *pcszData, int nLen);`
- `QDataStream &operator << (/*... */);`
- `QDataStream &operator >> (/*... */);`

Types Supported by QDataStream

bool	qint8	qint16	qint32	qint64	**float**
const char *	quint8	quint16	quint32	quint64	**double**

Table 6: basic types

QDate	QDateTime	QTime
QKeySequence	QRegExp	QRegularExpression
QPair <T1, T2>	QUrl	QVariant

Table 7: utility classes

Types Supported by QDataStream (end)

QBitArray	QByteArray	QString
QList <T>	QLinkedList <T>	QVector <T>
QSet <T>	QMap <Key, T>	QHash <Key, T>

Table 8: containers

QCursor	QIcon	QImage	QPicture	QPixmap
QPoint	QSize	QRect	QMargins	QRegion
QColor	QPen	QBrush	QPalette	QFont
QVector4D	QMatrix4x4	QTransform	QQuaternion	QEasingCurve

Table 9: GUI classes

Example

Example (create-net-session.cpp)

```
#include "create-net-session.h"

#include <QNetworkConfigurationManager>

QSharedPointer <QNetworkSession> createNetworkSession()
{
  QSharedPointer <QNetworkSession> ptr_session;
  QNetworkConfigurationManager manager;
```

Example (cont.)

Example (create-net-session.cpp, end)

```
  if (manager.capabilities() &
    QNetworkConfigurationManager::NetworkSessionRequired)
  {
    QNetworkConfiguration config = manager.defaultConfiguration();
    ptr_session.reset(new QNetworkSession(config));
    ptr_session->open();
    ptr_session->waitForOpened(-1);
  }
  //
  return ptr_session;
}    // createNetworkSession()
```

Example (cont.)

Example (example-23-server.cpp)

```
#include "create-net-session.h"

#include <QApplication>
#include <QTcpServer>
#include <QTcpSocket>
#include <QDataStream>
#include <QTextStream>

#include <cstdio>
```

Example (cont.)

Example (example-23-server.cpp, cont.)

```
int main(int nArgC, char *apszArgV[])
{
  QApplication app(nArgC, apszArgV);
  QTextStream out(stdout), err(stderr);
  //
  // 1. Creating a network session
  //
  QSharedPointer <QNetworkSession> ptr_session(
    createNetworkSession());
```

Example (cont.)

Example (example-23-server.cpp, cont.)

```
//
// 2. Listening
//
out << "Listening..." << endl;
QTcpServer server;
if (!server.listen(QHostAddress::Any, 44967))
{
  err << "Error: " << server.errorString() << endl;
  return -1;
}
```

Example (cont.)

Example (example-23-server.cpp, cont.)

```
//
out <<
  "Server initialized at " <<
  server.serverAddress().toString() <<
  ":" << server.serverPort() << endl;
```

Example (cont.)

Example (example-23-server.cpp, cont.)

```
//
// 3. Connecting to the client
//
out << "Waiting for the client connection..." << endl;
bool bTimedOut;
while (!server.waitForNewConnection(5000, &bTimedOut))
  if (!bTimedOut)
  {
    err << "Error: " << server.errorString() << endl;
    return -1;
  }
```

Example (cont.)

Example (example-23-server.cpp, cont.)

```
//
QScopedPointer <QTcpSocket> ptr_socket(
  server.nextPendingConnection());
out <<
  "Connected to " <<
  ptr_socket->peerAddress().toString() <<
  ':' << ptr_socket->peerPort() << endl;
QDataStream inout(ptr_socket.data());
QString str;
```

Example (cont.)

Example (example-23-server.cpp, cont.)

```
//
// 4. Receiving from the client
//
out << "Receiving data from the client..." << endl;
do
{
  while (!ptr_socket->waitForReadyRead(5000))
    if (ptr_socket->error() != QAbstractSocket::UnknownSocketError)
    {
      err << "Error: " << ptr_socket->errorString() << endl;
      return -1;
    }
```

Example (cont.)

Example (example-23-server.cpp, cont.)

```
    //
    inout.startTransaction();
    inout >> str;
  }
  while (!inout.commitTransaction());
  //
  out << "Reply from the client: " << str << endl;
```

Example (cont.)

Example (example-23-server.cpp, end)

```
  //
  // 5. Sending to the client
  //
  inout << (QString("Hello from Server, ") + str);
  ptr_socket->flush();
  //
  ptr_socket->disconnectFromHost();
  //
  out << "Server finished." << endl;
}    // main()
```

Example (cont.)

Example (example-23-client.cpp)

```
#include "create-net-session.h"

#include <QApplication>
#include <QTcpSocket>
#include <QHostAddress>
#include <QDataStream>
#include <QTextStream>

#include <cstdio>
```

Example (cont.)

Example (example-23-client.cpp, cont.)

```
int main(int nArgC, char *apszArgV[])
{
  QApplication app(nArgC, apszArgV);
  QTextStream out(stdout), err(stderr);
  //
  // 1. Creating a network session
  //
  QSharedPointer <QNetworkSession> ptr_session(
    createNetworkSession());
```

Example (cont.)

Example (example-23-client.cpp, cont.)

```
//
// 2. Connecting to the server
//
QTcpSocket socket;
QDataStream inout(&socket);
socket.connectToHost(QHostAddress::LocalHost, 44967);
out << "Connecting to the server..." << endl;
while (!socket.waitForConnected(5000))
  if (socket.error() != QAbstractSocket::UnknownSocketError)
  {
    err << "Error: " << socket.errorString() << endl;
    return -1;
  }
```

Example (cont.)

Example (example-23-client.cpp, cont.)

```
//
out <<
  "Connected to " << socket.peerAddress().toString() <<
  ':' << socket.peerPort() <<
  "\nLocal: " << socket.localAddress().toString() <<
  ':' << socket.localPort() << endl;
//
// 3. Sending to the server
//
inout << QString("A Client");
socket.flush();
out << "Sent data to the server." << endl;
```

Example (cont.)

Example (example-23-client.cpp, cont.)

```
//
// 4.  Receiving from the server
//
out << "Receiving data from the server..." << endl;
QString str;
```

Example (cont.)

Example (example-23-client.cpp, cont.)

```
do
{
  while (!socket.waitForReadyRead(5000))
    if (socket.error() != QAbstractSocket::UnknownSocketError)
    {
      err << "Error: " << socket.errorString() << endl;
      return -1;
    }
  //
  inout.startTransaction();
  inout >> str;
}
while (!inout.commitTransaction());
```

Example (end)

Example (example-23-client.cpp, end)

```
  //
  out <<
    "Reply from the server: " << str <<
    "\nClient finished." << endl;
}    // main()
```

Example

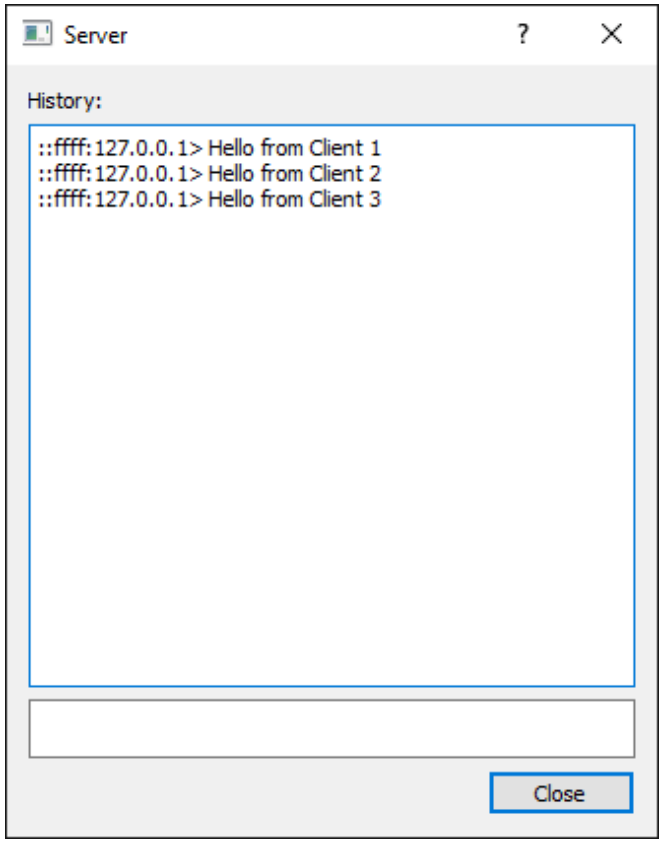

Figure 4: TCP server program

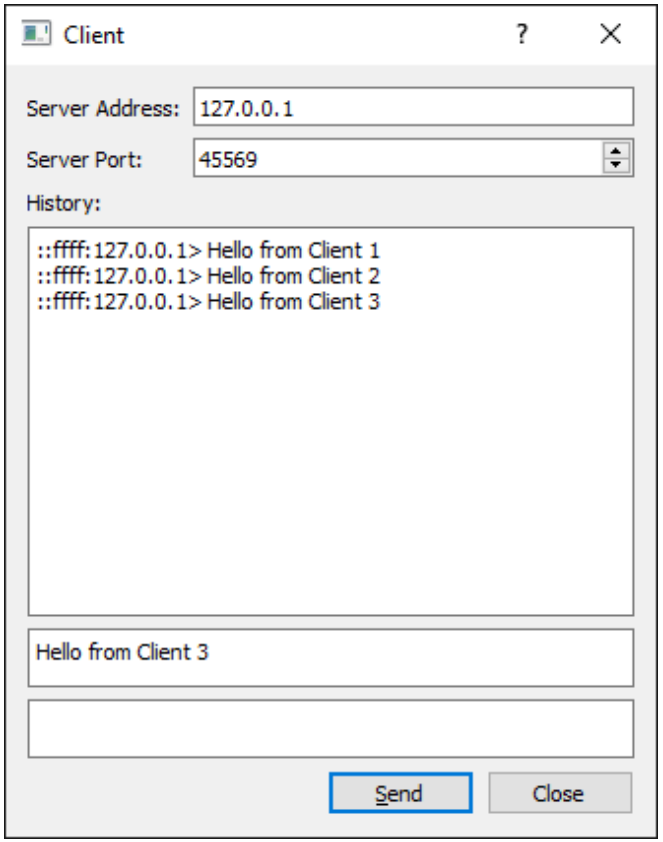

Figure 5: TCP client program

Example (cont.)

Example (create-net-session.cpp)

```
QNetworkSession *createNetworkSession(QObject *pParent)
{
  QNetworkSession *pSession = Q_NULLPTR;
  QNetworkConfigurationManager manager;
  if (manager.capabilities() &
    QNetworkConfigurationManager::NetworkSessionRequired)
  {
    QNetworkConfiguration config = manager.defaultConfiguration();
    pSession = new QNetworkSession(config, pParent);
  }
  //
  return pSession;
}
```

Example (cont.)

Example (server-dlg.cpp)

```
#include "server-dlg.h"

#include "create-net-session.h"

#include <QTcpServer>
#include <QTcpSocket>

ServerDialog::ServerDialog()
{
  setupUi(this);
  //
  connect(
    m_pButtonClose, SIGNAL(clicked()), this, SLOT(close()));
```

Example (cont.)

Example (server-dlg.cpp, cont.)

```
  //
  m_pSession = createNetworkSession(this);
  if (m_pSession)
  {
    connect(
      m_pSession, &QNetworkSession::opened,
      this, &ServerDialog::on_sessionOpened);
    m_pSession->open();
  }
  else
    on_sessionOpened();
}    // ServerDialog()
```

Example (cont.)

Example (server-dlg.cpp, cont.)

```
void ServerDialog::on_sessionOpened()
{
  m_pServer = new QTcpServer(this);
  connect(
    m_pServer, &QTcpServer::newConnection,
    this, &ServerDialog::on_newConnection);
  if (!m_pServer->listen(QHostAddress::Any, 45569))
  {
    QString strErrors = m_pTextEditErrors->toPlainText();
    strErrors += QString("Error: ") + m_pServer->errorString() + '\n';
    //
    m_pTextEditErrors->setPlainText(strErrors);
  }
}
```

Example (cont.)

Example (server-dlg.cpp, cont.)

```
void ServerDialog::on_newConnection()
{
  QTcpSocket *pClientSocket = m_pServer->nextPendingConnection();
  connect(
    pClientSocket, &QAbstractSocket::disconnected,
    pClientSocket, &QObject::deleteLater);
  connect(
    pClientSocket, &QIODevice::readyRead,
    this, &ServerDialog::on_readyRead);
}
```

Example (cont.)

Example (server-dlg.cpp, cont.)

```
void ServerDialog::on_readyRead()
{
  QTcpSocket *pClientSocket =
    qobject_cast <QTcpSocket *> (sender());
  if (!pClientSocket)
    return;
  //
  QDataStream inout(pClientSocket);
  inout.startTransaction();
  QString strChatLine;
  inout >> strChatLine;
  if (!inout.commitTransaction())
    return;
```

Example (cont.)

Example (server-dlg.cpp, end)

```
  //
  QString strChat =
    m_pTextEditChat->toPlainText() +
    pClientSocket->peerAddress().toString() +
    "> " + strChatLine + '\n';
  inout << strChat;
  m_pTextEditChat->setPlainText(strChat);
  //
  pClientSocket->disconnectFromHost();
}    // on_readyRead()
```

Example (cont.)

Example (client-dlg.cpp, cont.)

```
#include "client-dlg.h"

#include "create-net-session.h"

#include <QTcpSocket>

ClientDialog::ClientDialog()
{
  setupUi(this);
  //
  connect(
    m_pButtonClose, SIGNAL(clicked()), this, SLOT(close()));
  //
```

Example (cont.)

Example (client-dlg.cpp, cont.)

```
  m_pSession = createNetworkSession(this);
  if (m_pSession)
  {
    connect(
      m_pSession, &QNetworkSession::opened,
      this, &ClientDialog::on_sessionOpened);
    m_pSession->open();
  }
  else
    on_sessionOpened();
}    // ClientDialog()
```

Example (cont.)

Example (client-dlg.cpp, cont.)

```
void ClientDialog::on_buttonSend_clicked()
{
  m_pSocket->connectToHost(
    m_pTextEditAddress->text(),
    m_pSpinboxPort->value());
}
```

Example (cont.)

Example (client-dlg.cpp, cont.)

```
void ClientDialog::on_sessionOpened()
{
  m_pSocket = new QTcpSocket(this);
  connect(
    m_pSocket, &QAbstractSocket::connected,
    this, &ClientDialog::on_connected);
  connect(
    m_pSocket, &QIODevice::readyRead,
    this, &ClientDialog::on_readyRead);
}
```

Example (cont.)

Example (client-dlg.cpp, cont.)

```
void ClientDialog::on_connected()
{
  QDataStream inout(m_pSocket);
  inout << m_pTextEditLine->toPlainText();
}
```

Example (end)

Example (client-dlg.cpp, end)

```
void ClientDialog::on_readyRead()
{
  QDataStream inout(m_pSocket);
  inout.startTransaction();
  QString strChat;
  inout >> strChat;
  if (!inout.commitTransaction())
    return;
  //
  m_pTextEditChat->setPlainText(strChat);
}    // on_readyRead()
```

QNetworkAccessManager Class

Methods (Setting Connections)

- `void connectToHost(const QString &rcHostName, quint16 uPort = 80);`
- `void connectToHostEncrypted(`
  `const QString &rcHostName, quint16 uPort = 443,`
  `const QSslConfiguration &sslConfiguration =`
  `QSslConfiguration::defaultConfiguration());`

QNetworkAccessManager Class (cont.)

Methods (Sending Requests)

- `QNetworkReply *head(const QNetworkRequest &rcRequest);`
- `QNetworkReply *get(const QNetworkRequest &rcRequest);`
- `QNetworkReply *put(const QNetworkRequest &rcRequest, const QByteArray &rcData);`
  `QHttpMultiPart *`
  `QIODevice *`
- `QNetworkReply *post(const QNetworkRequest &rcRequest, const QByteArray &rcData);`
  `QHttpMultiPart *`
  `QIODevice *`
- `QNetworkReply *sendCustomRequest(const QNetworkRequest &rcRequest, const QByteArray &rcVerb, QIODevice *pData = Q_NULLPTR);`

QNetworkAccessManager Class (cont.)

Methods (Properties)

- `QNetworkConfiguration configuration() const;/`
  `void setConfiguration(const QNetworkConfiguration &rcConfig);`
- `QNetworkProxy proxy() const;/`
  `void setProxy(const QNetworkProxy &rcNetworkProxy);`

QNetworkAccessManager Class (end)

Methods (Signals)

- **void** authenticationRequired(
  QNetworkReply *pReply, QAuthenticator *pAuthenticator);
- **void** proxyAuthenticationRequired(
  const QNetworkProxy &rcProxy, QAuthenticator *pAuthenticator);
- **void** preSharedKeyAuthenticationRequired(
  QNetworkReply *pReply,
  QSslPreSharedKeyAuthenticator *pAuthenticator);
- **void** encrypted(QNetworkReply *pReply);
- **void** finished(QNetworkReply *pReply);
- **void** sslErrors(
  QNetworkReply *pReply, **const** QList <QSslError> &rcErrors);

QNetworkRequest/QNetworkReply Classes

QNetworkRequest Methods (Properties)

- `QVariant header(QNetworkRequest::KnownHeaders header) const;`
- `void setHeader(KnownHeaders nHeader, const QVariant &rcValue);`

QNetworkReply Methods (Properties)

- `QVariant header(QNetworkRequest::KnownHeaders header) const;`

UserAgentHeader	ServerHeader	LastModifiedHeader
CookieHeader	SetCookieHeader	LocationHeader
ContentTypeHeader	ContentDispositionHeader	ContentLengthHeader

Table 10: values for QNetworkRequest::KnownHeaders

Example

Example (example-25.cpp)

```
#include <QApplication>
#include <QLoggingCategory>
#include <QNetworkAccessManager>
#include <QNetworkRequest>
#include <QNetworkReply>
#include <QEventLoop>
#include <QDomDocument>

#include <cstdio>

const char g_cszUrl[] =
  "http://feeds.bbci.co.uk/news/rss.xml";
```

Example (cont.)

Example (example-25.cpp, cont.)

```
void outText(
  QTextStream &rOut, const QDomElement &rcParent,
  const QString &rcChildName, const QString &rcItemTitle);

int main(int nArgC, char *apszArgV[])
{
  QApplication app(nArgC, apszArgV);
  QTextStream out(stdout), err(stderr);
  //
  QLoggingCategory::setFilterRules("qt.network.ssl.warning=false");
  //
```

Example (cont.)

Example (example-25.cpp, cont.)

```
QString hostName { g_cszUrl };
QNetworkAccessManager manager;
manager.connectToHost(hostName);
//
QUrl url { hostName };
QNetworkRequest request(url);
//
QNetworkReply *pReply = manager.get(request);
QEventLoop loop;
QObject::connect(pReply, SIGNAL(finished()), &loop, SLOT(quit()));
loop.exec();
//
```

Example (cont.)

Example (example-25.cpp, cont.)

```
const int cnError = pReply->error();
if (cnError != 0)
{
  err << "HTTP error: " << cnError << endl;
  return -1;
}
//
QByteArray resData = pReply->readAll();
QString strErrorMsg;
int nErrorRow, nErrorCol;
QDomDocument document;
const bool cbSuccess = document.setContent(
  resData, true, &strErrorMsg, &nErrorRow, &nErrorCol);
```

Example (cont.)

Example (example-25.cpp, cont.)

```
if (!cbSuccess)
{
  err <<
    "XML error at " << nErrorRow <<
    " : " << nErrorCol <<
    ": " << strErrorMsg << endl;
  return -1;
}
//
```

Example (cont.)

Example (example-25.cpp, cont.)

```
QDomElement elementRss = document.firstChildElement("rss");
if (!elementRss.isNull())
{
  QDomElement elementChannel = elementRss.firstChildElement(
    "channel");
  while (!elementChannel.isNull())
  {
    outText(out, elementChannel, "title", "Channel title");
    //
    QDomElement elementItem = elementChannel.firstChildElement(
      "item");
```

Example (cont.)

Example (example-25.cpp, cont.)

```
while (!elementItem.isNull())
{
  out << "  Item:" << endl;
  outText(out, elementItem, "title", "    Title");
  outText(out, elementItem, "description", "    Description");
  outText(out, elementItem, "pubDate", "    Date");
  outText(out, elementItem, "link", "    Link");
  //
  elementItem = elementItem.nextSiblingElement(
    "item");
}    // while (!elementItem.isNull())
```

Example (cont.)

Example (example-25.cpp, cont.)

```
      //
      elementChannel = elementChannel.nextSiblingElement(
        "channel");
    }    // while (!elementChannel.isNull())
  }    // if (!elementRss.isNull())
}    // main()
```

Example (cont.)

Example (example-25.cpp, cont.)

```
void outText(
  QTextStream &rOut, const QDomElement &rcParent,
  const QString &rcChildName, const QString &rcItemTitle)
{
  QDomElement element = rcParent.firstChildElement(rcChildName);
  if (element.isNull())
    return;
  //
  QDomNode node = element.firstChild();
  if (!node.isText())
    return;
  //
```

Example (end)

Example (example-25.cpp, end)

```
  QDomText section = node.toText();
  if (section.isNull())
    return;
  //
  rOut << rcItemTitle << ": " << section.data() << endl;
}    // outText()
```