Speeding up your Qt app with new QtNetwork features

Peter Hartmann, BlackBerry

Berlin, 8\textsuperscript{th} of October 2013
Who am I?

- software engineer at BlackBerry
- previously working for Trolltech and Nokia
- main focus: QtNetwork
What is the goal of this presentation?

Make your app faster!
outside QtNetwork

- start event loop early
- start network request before loading QML
- only use one QNetworkAccessManager per app

inside QtNetwork

- DNS
- TCP
- SSL
- HTTP
outside QtNetwork

start event loop early
MyClass::MyClass() {
    // load QML files
    // make network request
    // setup translations etc.
}

int main(int argc, char **argv) {
    QGuiApplication app(argc, argv);
    MyClass c;
    return app.exec();
}
MyClass::MyClass() {
    QMetaObject::invokeMethod(this, "init",
        Qt::QueuedConnection);
}

void MyClass::init() {
    // load QML files
    // make network request
    // setup translations etc.
}

int main(int argc, char **argv) {
    QGuiApplication app(argc, argv);
    MyClass c;
    return app.exec();
}
outside QtNetwork

- start event loop early
- start network request before loading QML
void MyClass::init() {
    QQuickView *view = new QQuickView;
    view->setSource(QUrl::fromLocalFile("main.qml"));
    view->show();

    QNetworkRequest request(QUrl("https://api.twitter.com"));
    QNetworkReply *reply = networkAccessManager->get(request);
    // here connect signals etc.
}
void MyClass::init() {
    QNetworkRequest request(QUrl("https://api.twitter.com"));
    QNetworkReply *reply = networkAccessManager-&gt;get(request);
    // here connect signals etc.

    QQuickView *view = new QQuickView;
    view-&gt;setSource(QUrl::fromLocalFile("main.qml"));
    view-&gt;show();
}

optimal:
start network request before loading QML

**Suboptimal:**

<table>
<thead>
<tr>
<th>Time Spent at App Start</th>
<th>QML file loading</th>
<th>Network Request</th>
</tr>
</thead>
<tbody>
<tr>
<td>milliseconds</td>
<td>600</td>
<td>1600</td>
</tr>
</tbody>
</table>

Qt Days 2013
start network request before loading QML

optimal:

time spent at app start

QML file loading

network request

milliseconds

0 200 400 600 800 1000 1200 1400 1600 1800
outside QtNetwork

- start event loop early
- start network request before loading QML
- only use one QNetworkAccessManager per app
QQuickView *view = new QQuickView;
view->setSource(QUrl::fromLocalFile("main.qml"));
view->show();

networkAccessManager = new QNetworkAccessManager;
QNetworkRequest request(QUrl("https://api.twitter.com"));
QNetworkReply *reply = networkAccessManager->get(request);
only use one QNetworkAccessManager per app

better:

QQuickView *view = new QQuickView;
view->setSource(QUrl::fromLocalFile("main.qml"));
view->show();

networkAccessManager = view->engine()->networkAccessManager();
QNetworkRequest request(QUrl("https://api.twitter.com"));
QNetworkReply *reply = networkAccessManager->get(request);
only use one QNetworkAccessManager per app

- ... matters for sharing of
- SSL sessions
- open sockets:

--- main.qml ---

Image {
  source: "https://www.server.com/image1.jpg"
}

--- MyClass.cpp ---

QNetworkRequest request(
    QUrl("https://www.server.com/file.json"));
QNetworkReply *reply = networkAccessManager->get(request);
only use one QNetworkAccessManager per app

using 2 QNetworkAccessManager instances

image1.jpg

file.json

- DNS lookup
- TCP handshake
- SSL handshake
- HTTP request/reply
only use one QNetworkAccessManager per app

using 1 QNetworkAccessManager instance

image1.jpg / file.json

DNS lookup
TCP handshake
SSL handshake
HTTP request/reply
outside QtNetwork

- start event loop early
- start network request before loading QML
- only use one QNetworkAccessManager per app

inside QtNetwork
start network requests as early as possible!

QQuickView *view = new QQuickView;

networkAccessManager = view->engine()->networkAccessManager();
QNetworkRequest request(QUrl("https://api.twitter.com"));
QNetworkReply *reply = networkAccessManager->get(request);

view->setSource(QUrl::fromLocalFile("main.qml"));
view->show();
outside QtNetwork

- start event loop early
- start network request before loading QML
- only use one QNetworkAccessManager per app

inside QtNetwork
inside QtNetwork

QNetworkRequest request(QUrl("https://api.twitter.com"));
QNetworkReply *reply = networkAccessManager->get(request);

network request break down on Wifi

- DNS lookup
- TCP handshake
- SSL handshake
- HTTP request / reply
outside QtNetwork

- start event loop early
- start network request before loading QML
- only use one QNetworkAccessManager per app

inside QtNetwork

- DNS
- TCP
- SSL
- HTTP
• make pre-DNS lookup (new in Qt 5.1):

QHostInfo::lookupHost(QStringLiteral("api.twitter.com"), 0, 0);

• use system-wide DNS cache (outside of Qt)
DNS lookup

DNS record time-to-live

minutes

Alexa top sites rank
### Agenda

#### outside QtNetwork
- Start event loop early
- Start network request before loading QML
- Only use one QNetworkAccessManager per app

#### inside QtNetwork

- DNS
- TCP
- SSL
- HTTP
• pre-TCP handshake (new in Qt 5.2):

```cpp
networkAccessManager = ...
networkAccessManager->connectToHost(
    QStringLiteral("api.twitter.com"), 80);
```

• ... also works for several sockets:

```cpp
networkAccessManager = ...
for (int a = 0; a < 6; ++a) {
    networkAccessManager->connectToHost(
        QStringLiteral("api.twitter.com"), 80);
}
Agenda

outside QtNetwork
- start event loop early
- start network request before loading QML
- only use one QNetworkAccessManager per app

inside QtNetwork
- DNS
- TCP
- SSL
- HTTP
• pre-TCP-and-SSL handshake (new in Qt 5.2):

```cpp
networkAccessManager = ...
networkAccessManager->connectToHostEncrypted(
    QStringLiteral("api.twitter.com"), 443);
```

• ... also works for several sockets:

```cpp
networkAccessManager = ...
for (int a = 0; a < 6; ++a) {
    networkAccessManager->connectToHostEncrypted(
        QStringLiteral("api.twitter.com"), 443);
}
```
- **pre-connect example:**

```cpp
QNetworkRequest request(
    QUrl("https://www.server.com/file.json"));
QNetworkReply *reply = networkAccessManager->get(request);

--- file.json ---
{
    "images": [
    {
        "url": "https://www.server.com/image1.jpg",
        "url": "https://www.server.com/image2.jpg",
        "url": "https://www.server.com/image3.jpg",
        "url": "https://www.server.com/image4.jpg"
    }
    ]
}
```
SSL

suboptimal:

typical Web service use case

file.json / image1.jpg
image2.jpg
image3.jpg
image4.jpg

DNS lookup
TCP handshake
SSL handshake
HTTP request/reply
pre-connecting 3 sockets:

```cpp
QNetworkRequest request(QUrl("https://www.server.com/file.json"));
QNetworkReply *reply = networkAccessManager->get(request);

for (int a = 0; a < 3; ++a) {
    networkAccessManager->connectToHostEncrypted(
        QStringLiteral("www.server.com"), 443);
}
```
SSL

optimal:

typical Web service use case

file.json / image1.jpg

image2.jpg

image3.jpg

image4.jpg

0  500  1000  1500  2000

DNS lookup
TCP handshake
SSL handshake
HTTP request/reply
full blog post about pre-connecting sockets at

http://ow.ly/pkHXU
store and resume TLS session tickets (new in Qt 5.2)
full SSL handshake:
resumed SSL handshake:

- Client sends a client hello message.
- Server sends a server hello message.
- Client sends a change cipher spec message.
- Server sends a change cipher spec message.
- Client sends a finished message.
- Server sends a finished message.

The first round trip time is indicated between the client and server.
in Qt:

```cpp
void MyClass::replyFinished(QNetworkReply *reply) {
    QByteArray usedSession = reply->sslConfiguration().session();
    // now store usedSession to disk, database etc.
}
```
... and resume:

QByteArray usedSession = ... // load from disk, database etc.
QNetworkRequest request(url);

sslConfiguration.setSslOption
    (QSsl::SslOptionDisableSessionPersistence, false);
sslConfiguration.setSession(usedSession);

request.setSslConfiguration(sslConfiguration);
networkAccessManager->get(request);
SSL: store and resume TLS session tickets

TLS session ticket lifetime

Alexa top sites rank

hours
scenario with full SSL handshake

- file.json / image1.jpg
- image2.jpg
- image3.jpg
- image4.jpg

- DNS lookup
- TCP handshake
- SSL handshake
- HTTP request/reply
optimal:

scenario with resumed SSL handshake

- file.json / image1.jpg
- image2.jpg
- image3.jpg
- image4.jpg

- DNS lookup
- TCP handshake
- SSL handshake
- HTTP request/reply
full blog post about TLS session tickets at

http://ow.ly/pe7Ri
outside QtNetwork

- start event loop early
- start network request before loading QML
- only use one QNetworkAccessManager per app

inside QtNetwork

- DNS
- TCP
- SSL
- HTTP
• use the cache:

```cpp
networkAccessManager = view->engine()->networkAccessManager();

QNetworkDiskCache *cache = new QNetworkDiskCache;
cache->setCacheDirectory(QStringLiteral("cacheDir"));

networkAccessManager->setCache(cache);
```
... or even prefer the cache (loads more data from the cache):

```cpp
QNetworkRequest request(QUrl("https://api.twitter.com"));
request.setAttribute(QNetworkRequest::CacheLoadControlAttribute,
                     QNetworkRequest::PreferCache);
QNetworkReply *reply = networkAccessManager->get(request);
```
• enable pipelining:

QNetworkRequest request(QUrl("https://api.twitter.com"));
request.setAttribute(
    QNetworkRequest::HttpPipeliningAllowedAttribute, true);
QNetworkReply *reply = networkAccessManager->get(request);
HTTP

no pipelining:

Client

HTTP request
HTTP request
HTTP request

Server

HTTP reply
HTTP reply
HTTP reply
HTTP reply
HTTP

pipelining:

client

HTTP request
HTTP request
HTTP request

server

HTTP reply
HTTP reply
HTTP reply
pipelining helps with many requests to a server:

50 requests to google.com via SSL

- Desktop Linux / Ethernet
- BlackBerry10 / WLAN

- pipelining
- no pipelining
HTTP

- set priority:

```cpp
QNetworkRequest request(QUrl("https://api.twitter.com"));
request.setPriority(QNetworkRequest::HighPriority);
QNetworkReply *reply = networkAccessManager->get(request);
```
high vs. normal priority on 50th request (w/o pipelining)

Desktop Linux / Ethernet

BlackBerry10 / WLAN

milliseconds

high priority

normal priority
<table>
<thead>
<tr>
<th><strong>outside QtNetwork</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>start event loop early</td>
</tr>
<tr>
<td>start network request before loading QML</td>
</tr>
<tr>
<td>only use one QNetworkAccessManager per app</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>inside QtNetwork</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS</td>
</tr>
<tr>
<td>TCP</td>
</tr>
<tr>
<td>SSL</td>
</tr>
<tr>
<td>HTTP</td>
</tr>
</tbody>
</table>
Summary:

- start network requests as early as possible
- open more required sockets as early as possible
- cut down latency by caching as much as possible (DNS, TLS tickets, HTTP replies)
(subjective) priority list to speed up your app

1. start network request before loading QML
2. pre-connect sockets
3. persist TLS session tickets
4. use network disk cache
5. system-wide DNS cache
6. only use one QNetworkAccessManager per app
7. set request priority
8. start event loop early
<table>
<thead>
<tr>
<th>outside QtNetwork</th>
</tr>
</thead>
<tbody>
<tr>
<td>v4</td>
</tr>
<tr>
<td>QML ahead-of-time-compilation?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>inside QtNetwork</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS: asynchronous lookups (c-ARES)</td>
</tr>
<tr>
<td>TCP</td>
</tr>
<tr>
<td>SSL</td>
</tr>
<tr>
<td>HTTP: SPDY (5.3 ?), HTTP/2.0 ?, QUIC ?</td>
</tr>
</tbody>
</table>
Thank you!

Questions?
Feedback?

phartmann@blackberry.com

@peha23