Speeding up your Qt app with new QtNetwork features

Peter Hartmann, BlackBerry

Berlin, 8th 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!

Agenda

Qt Developer Days 2013

outside QtNetwork

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

inside QtNetwork

- DNS
- TCP
- SSL
- HTTP

Agenda

Qt Developer Days 2013

outside QtNetwork

- start event loop early

start event loop early

Qt Developer Days 2013

suboptimal:

```
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();
}
```

start event loop early

optimal:

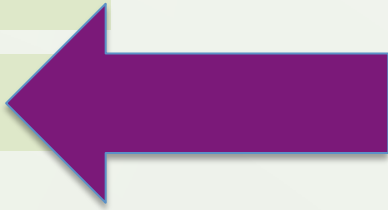
```
MyClass::MyClass() {
    QMetaObject::invokeMethod(this, "init",
        Qt::QueuedConnection);
}

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();
}
```

Agenda

Qt Developer Days 2013

outside QtNetwork

- start event loop early
- start network request before loading QML



start network request before loading QML

Qt Developer Days 2013

suboptimal:

```
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.
}
```

start network request before loading QML

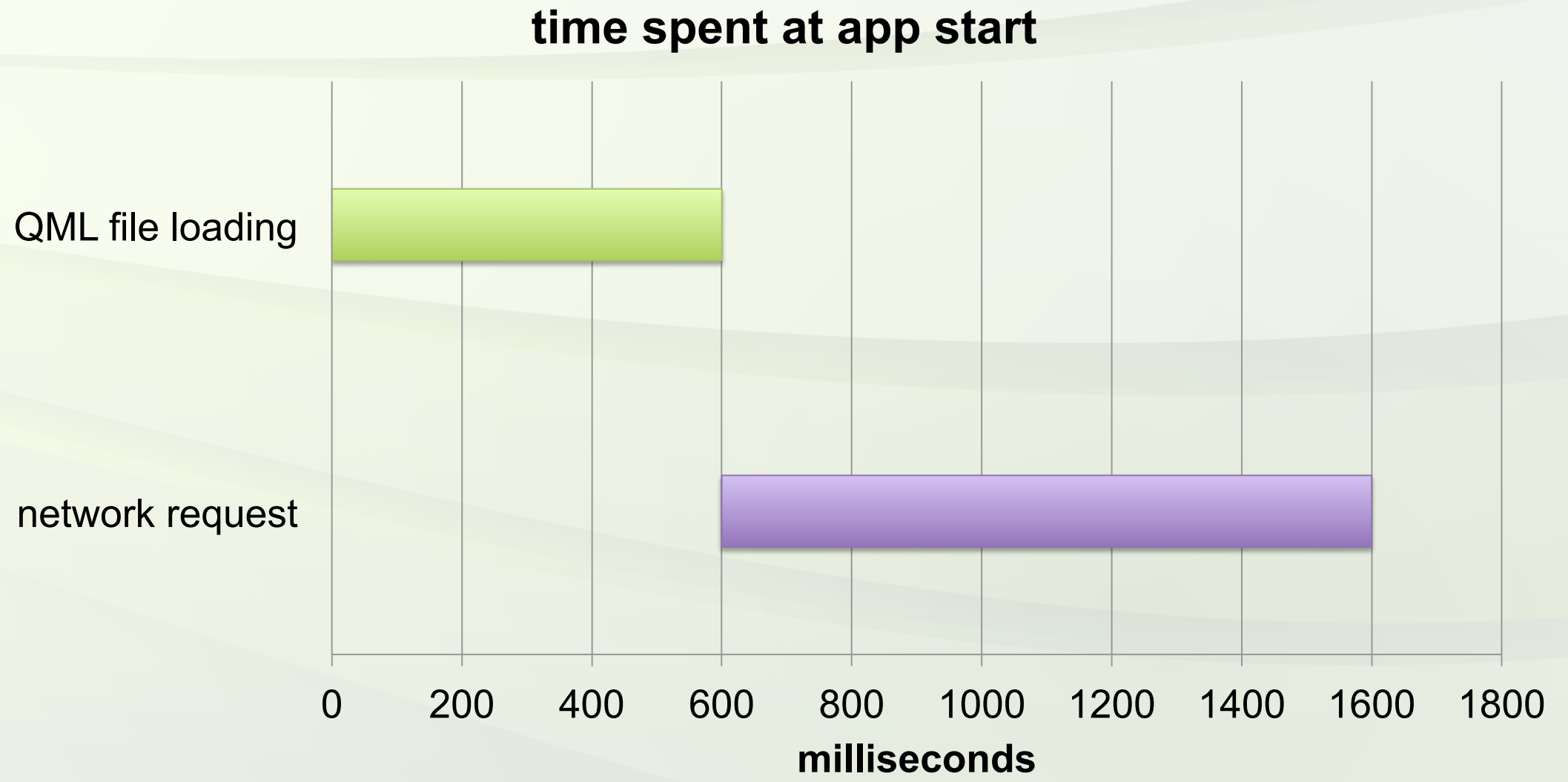
optimal:

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

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

start network request before loading QML

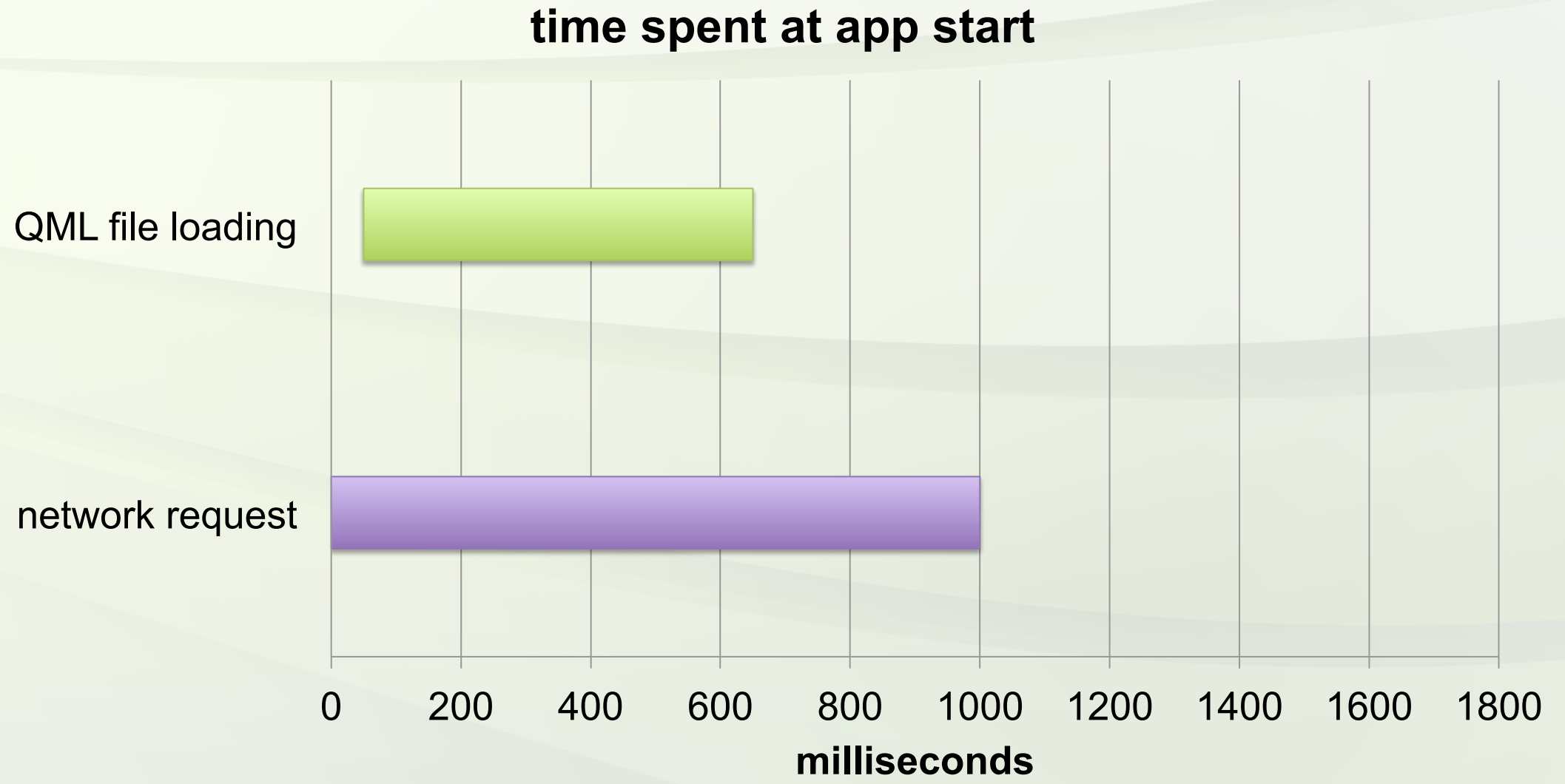
Qt Developer Days 2013

suboptimal:



start network request before loading QML

optimal:

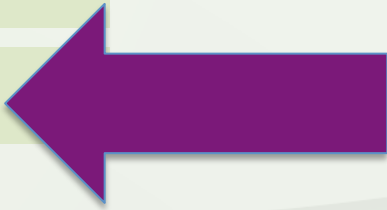


Agenda

Qt Developer Days 2013

outside QtNetwork

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



only use one QNetworkAccessManager per app

suboptimal:

```
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:

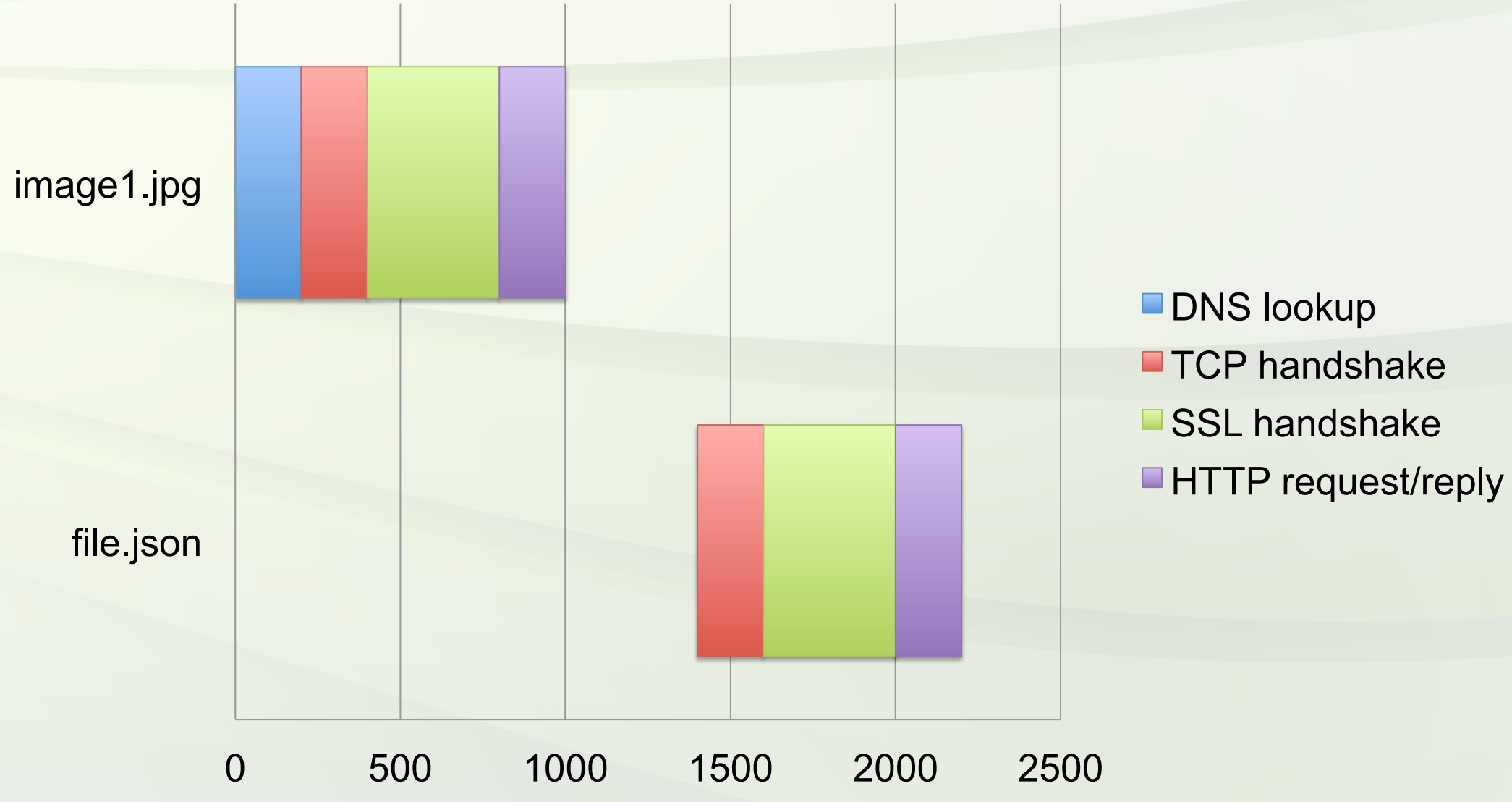
```
QQuickView *view = new QQuickView;
view->setSource(QUrl::fromLocalFile("main.qml"));
view->show();

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

only use one QNetworkAccessManager per app

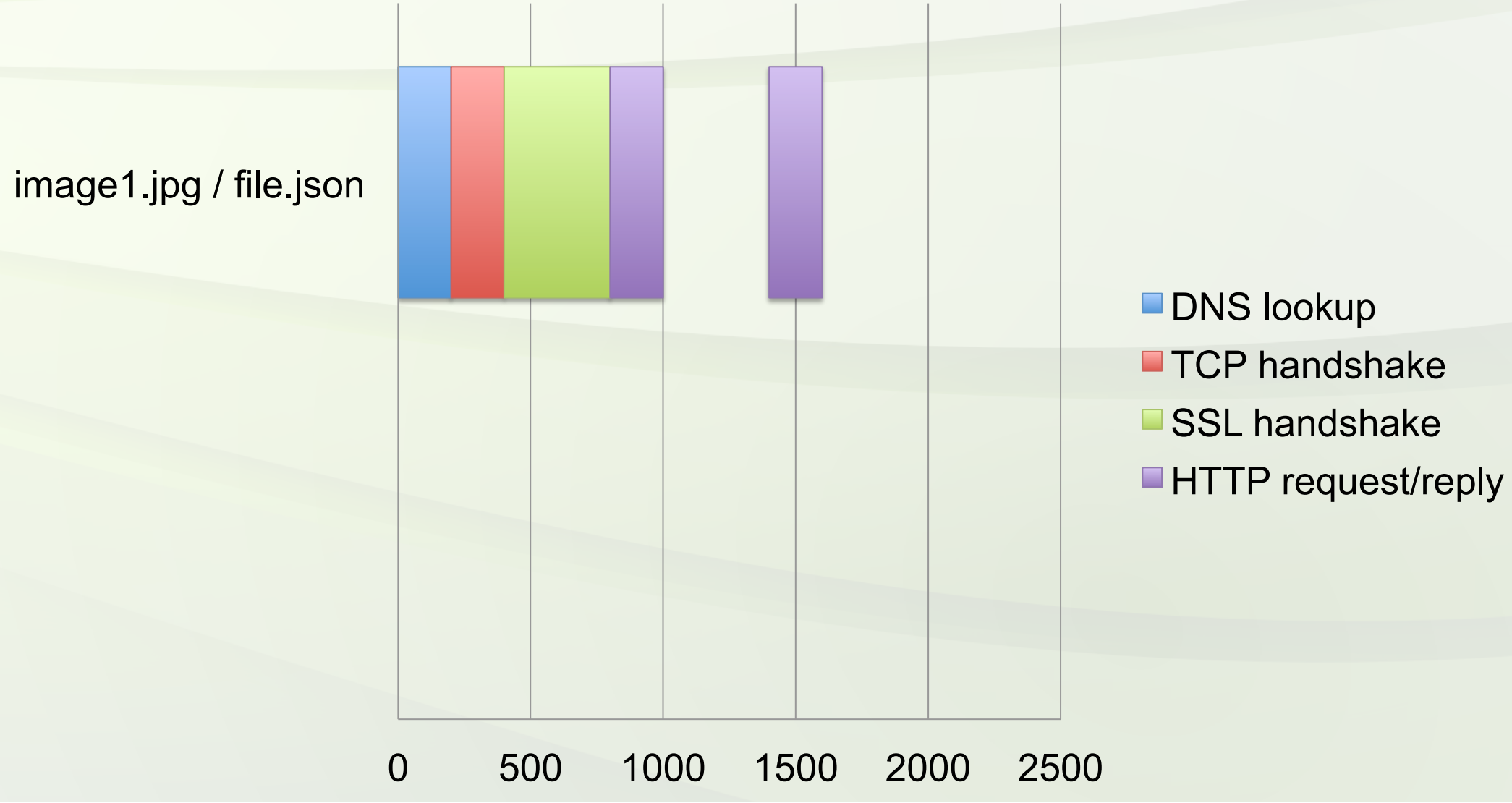
- … matters for sharing of
- SSL sessions
- open sockets:

```
--- 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);
```

using 2 QNetworkAccessManager instances



using 1 QNetworkAccessManager instance



Agenda

Qt Developer Days 2013

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!

optimal:

```
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();
```

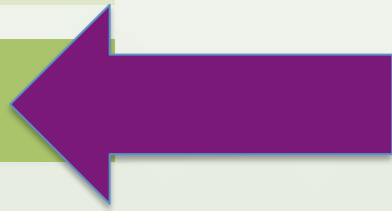
Agenda

Qt Developer Days 2013

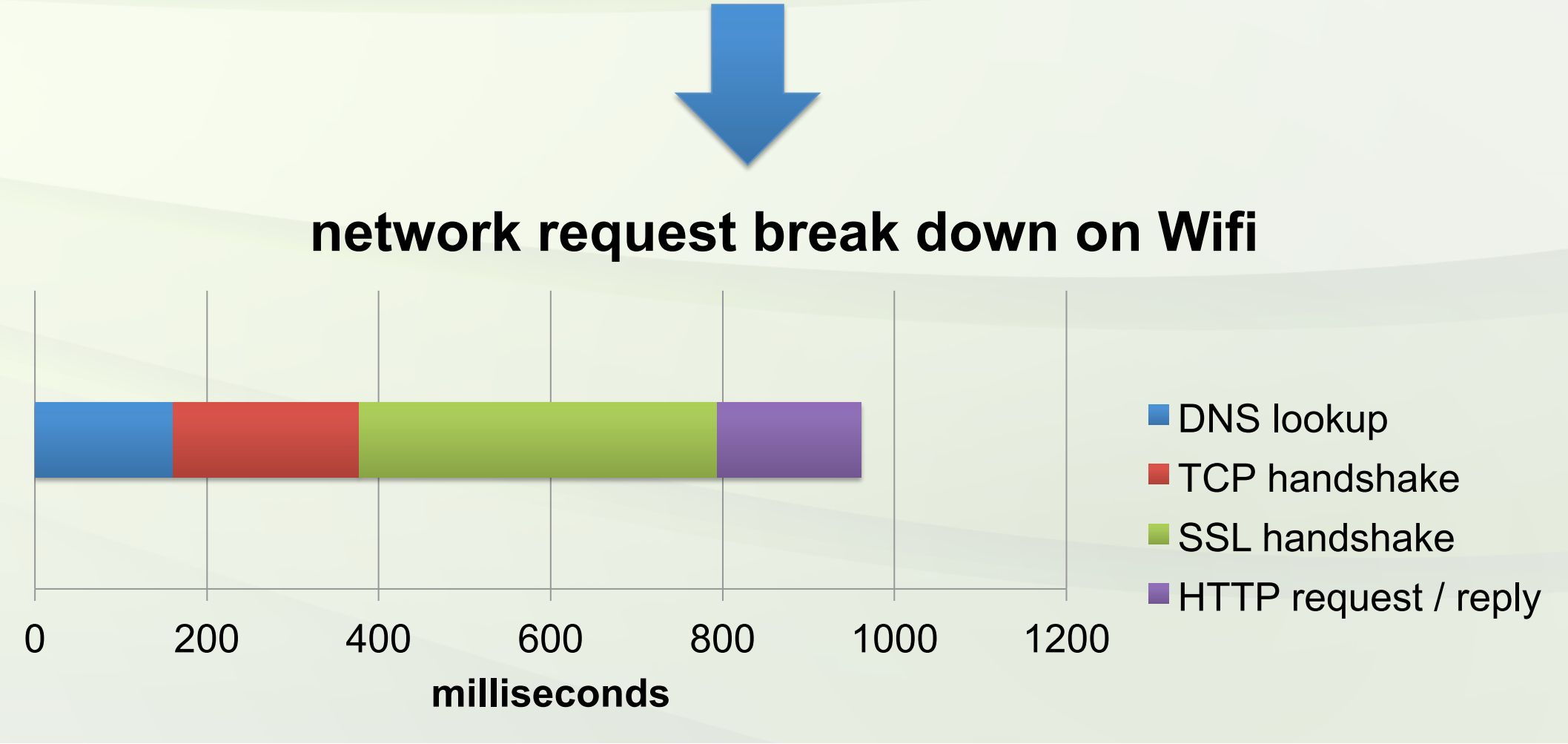
outside QtNetwork

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

inside QtNetwork



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



Agenda

Qt Developer Days 2013

outside QtNetwork

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

inside QtNetwork

- DNS
- TCP
- SSL
- HTTP

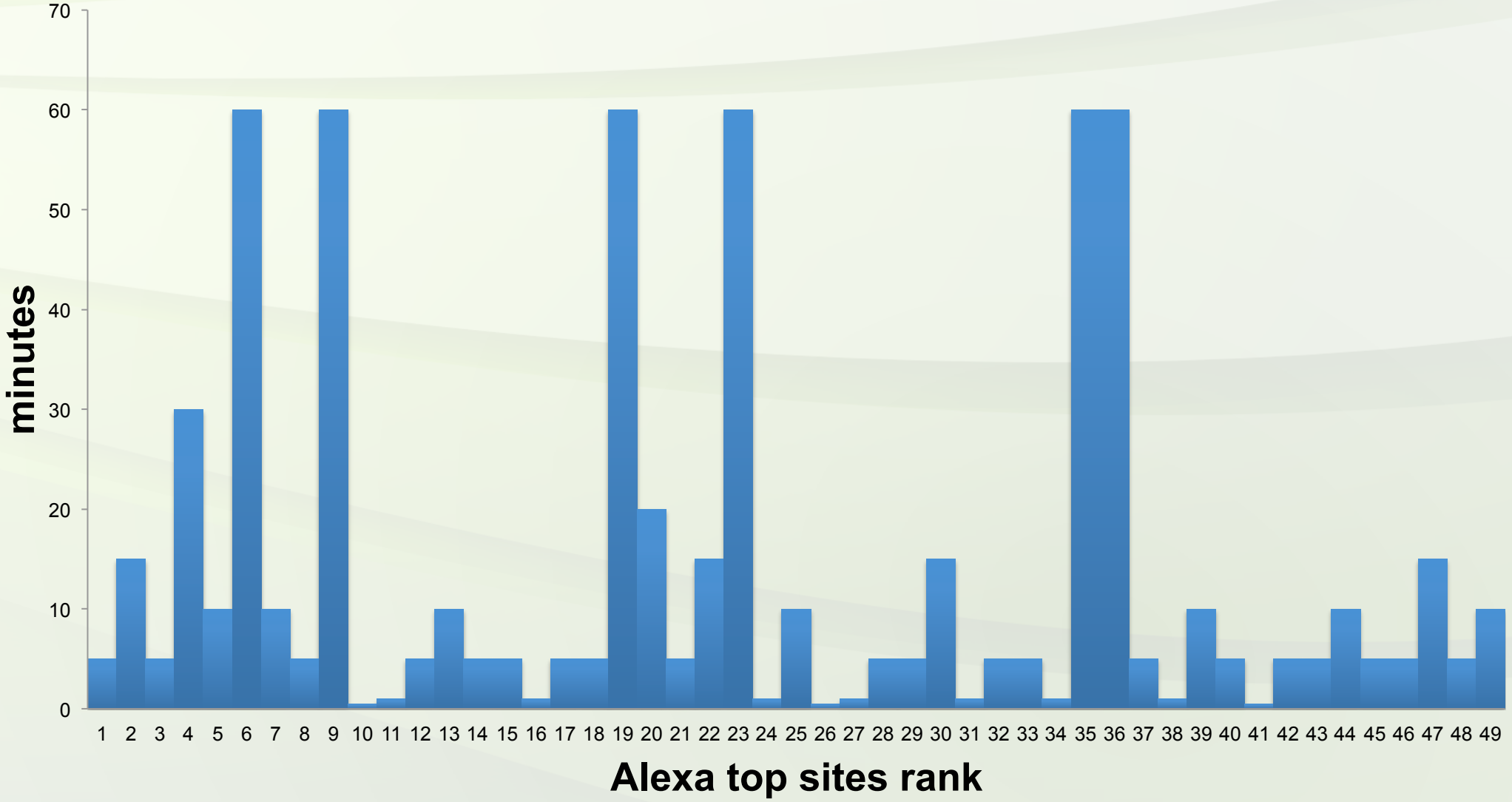
DNS

- 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 record time-to-live



Agenda

Qt Developer Days 2013

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):

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

- ... also works for several sockets:

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

Agenda

Qt Developer Days 2013

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):

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

- ... also works for several sockets:

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

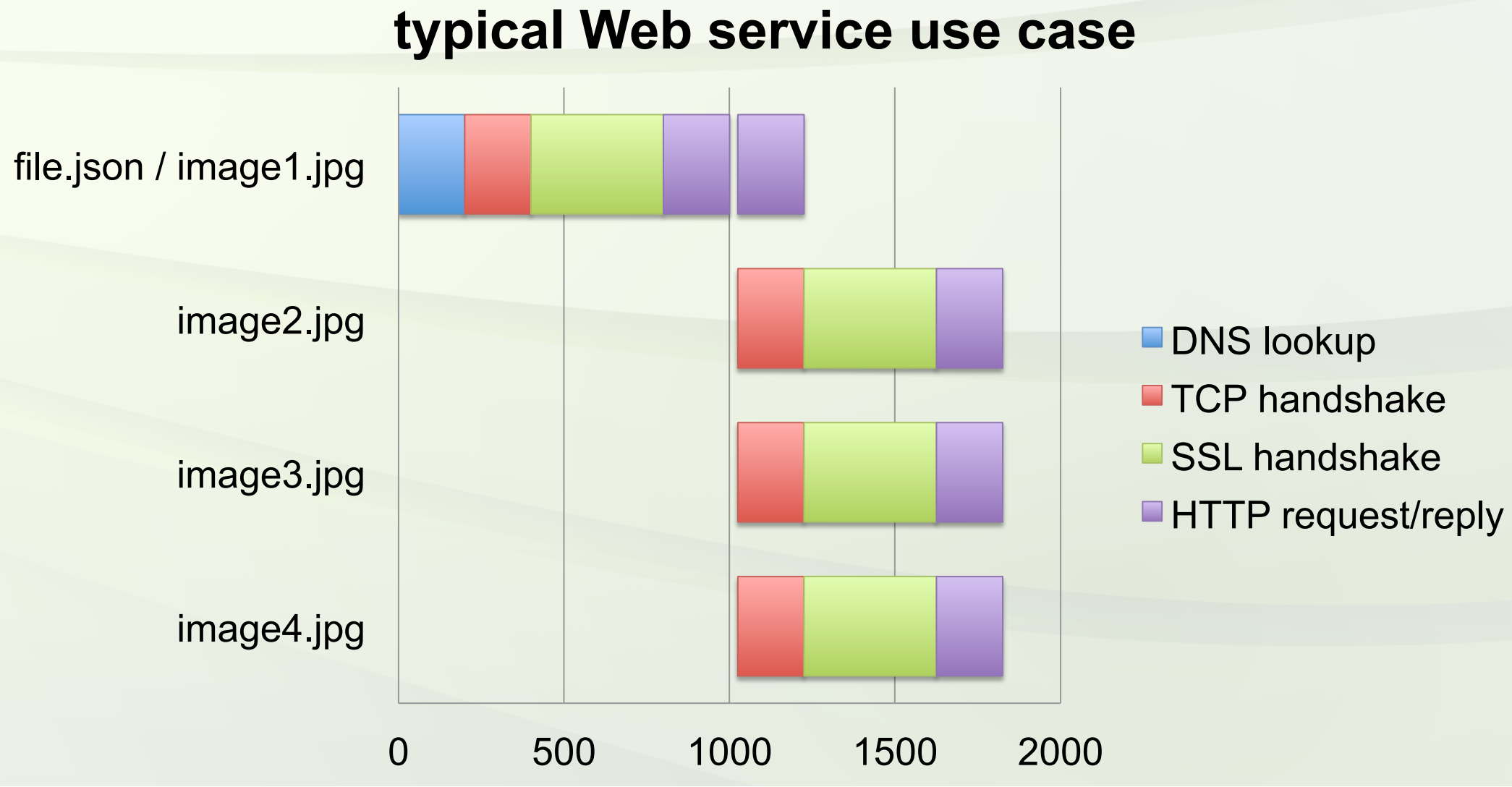
- pre-connect example:

```
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"
      }
   ]
}
```

Qt Developer Days 2013

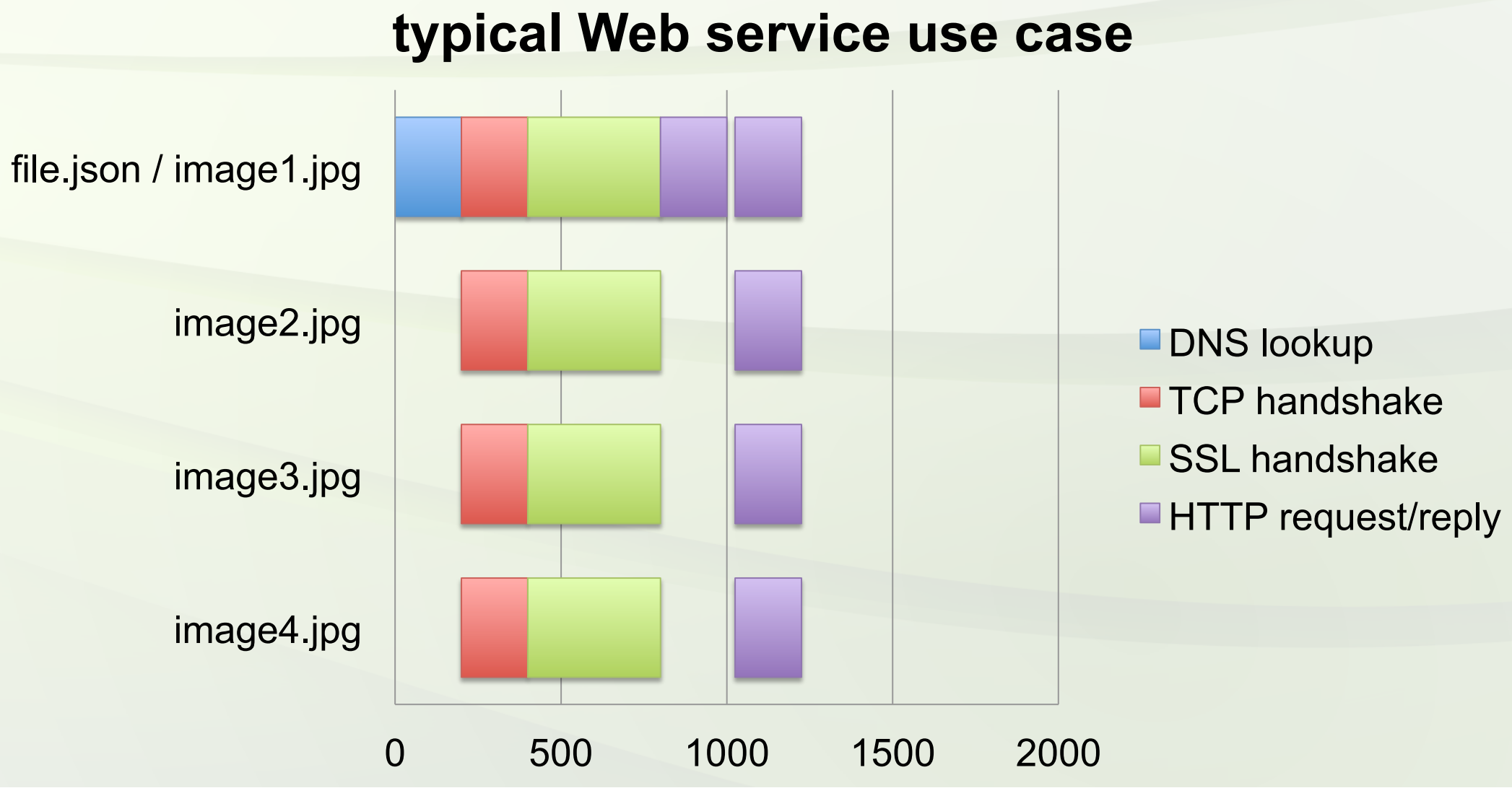
suboptimal:



- pre-connecting 3 sockets:

```
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);
}
```

optimal:

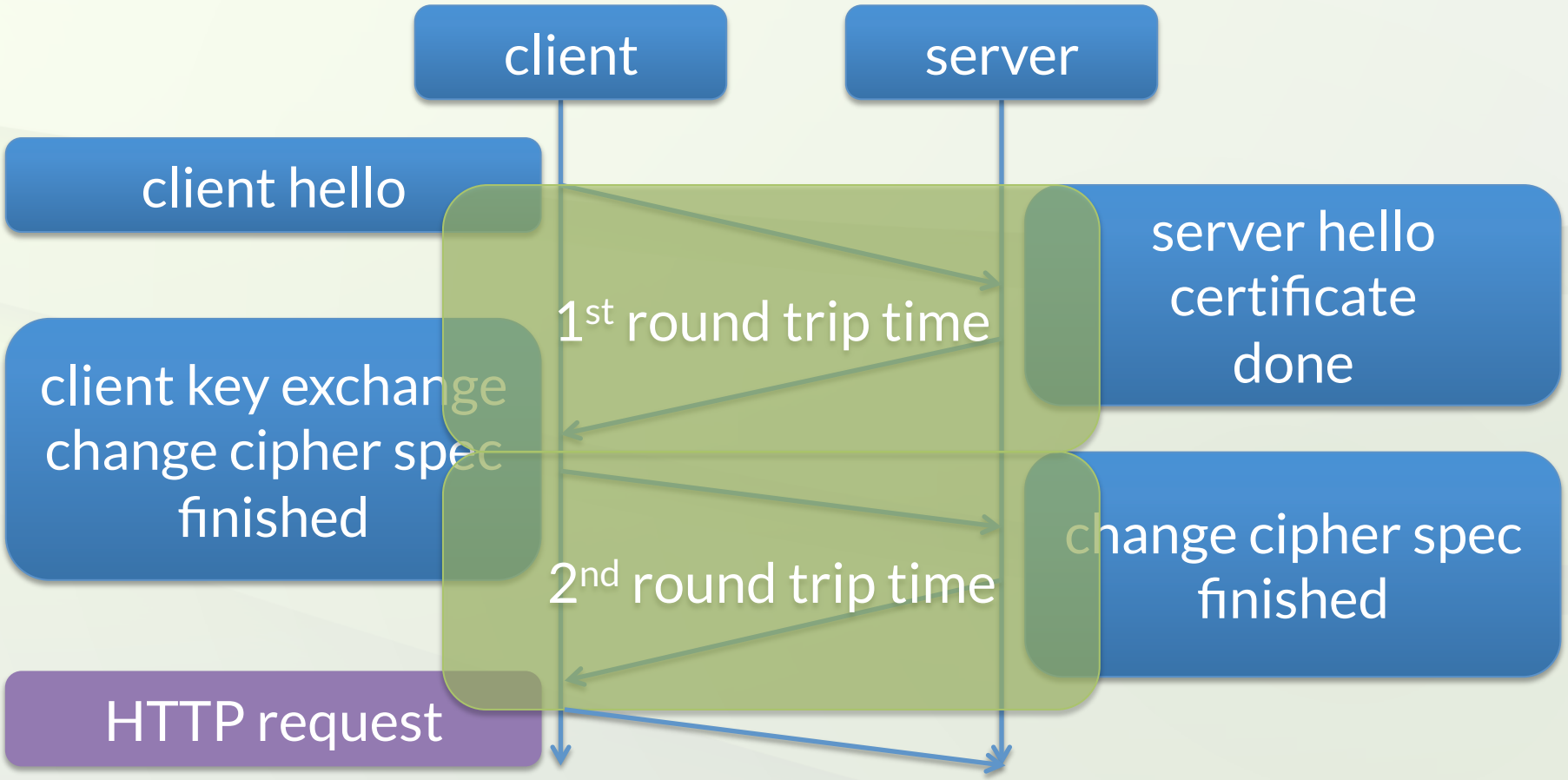

typical Web service use case
file.json / image1.jpg
image2.jpg
image3.jpg
image4.jpg
0
500
1000
1500
2000
DNS lookup
TCP handshake
SSL handshake
HTTP request/reply

full blog post about pre-connecting sockets at

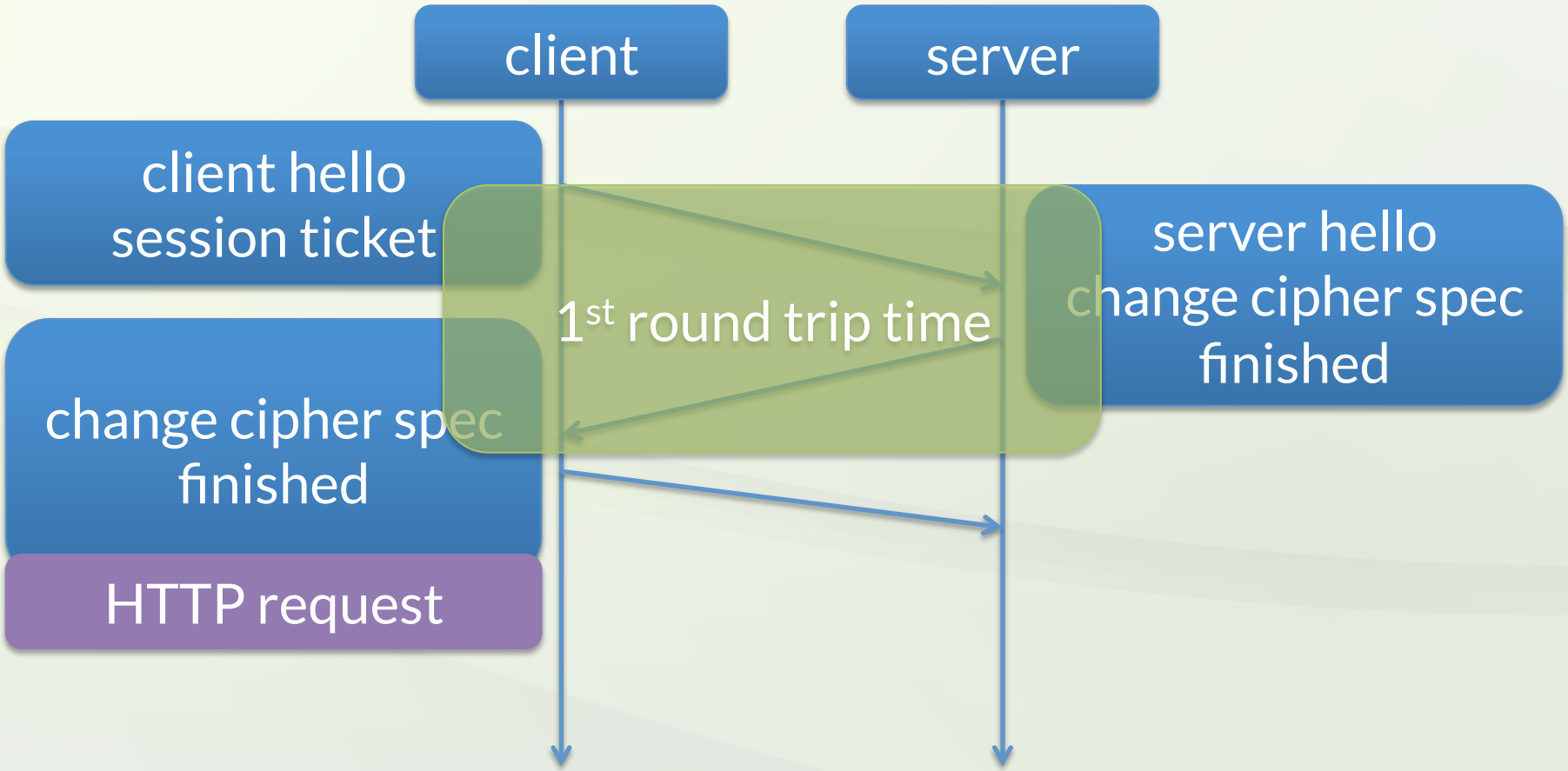
http://ow.ly/pkHXU

store and resume TLS session tickets
(new in Qt 5.2)

full SSL handshake:



resumed SSL handshake:



SSL: store and resume TLS session tickets

in Qt:

store...

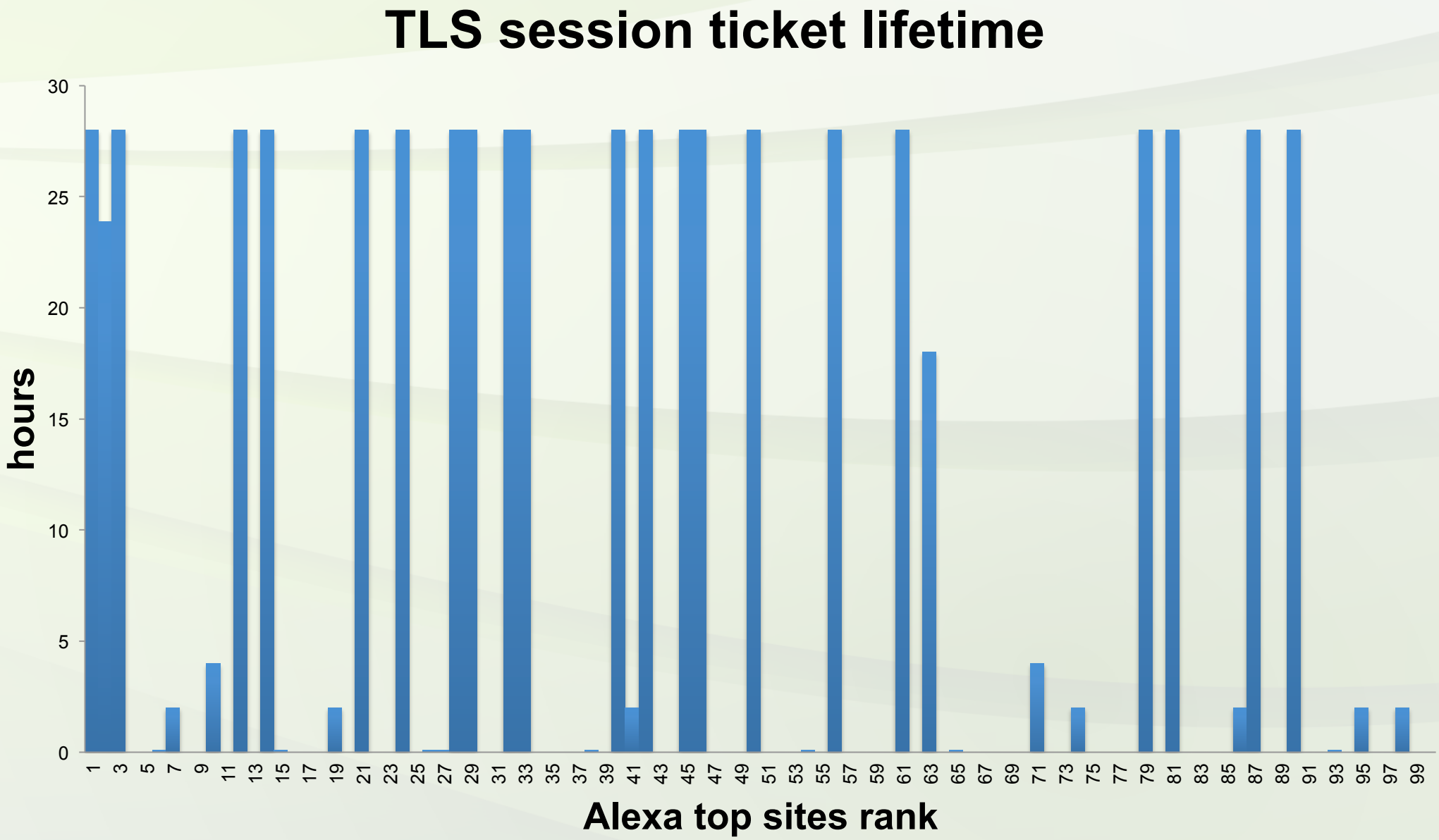
```
void MyClass::replyFinished(QNetworkReply *reply) {
    QByteArray usedSession = reply->sslConfiguration().session();
    // now store usedSession to disk, database etc.
}
```

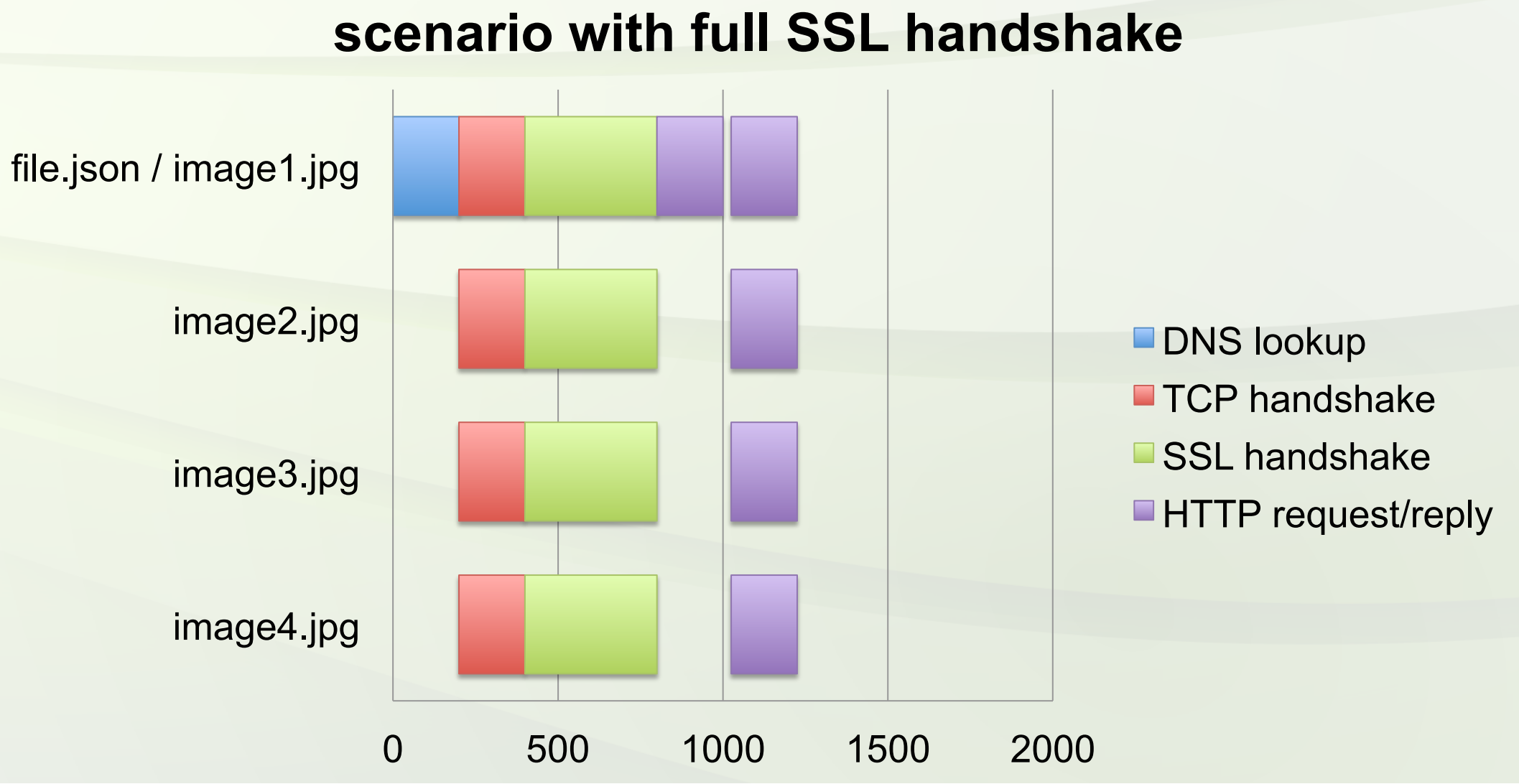
SSL: store and resume TLS session tickets

... and resume:

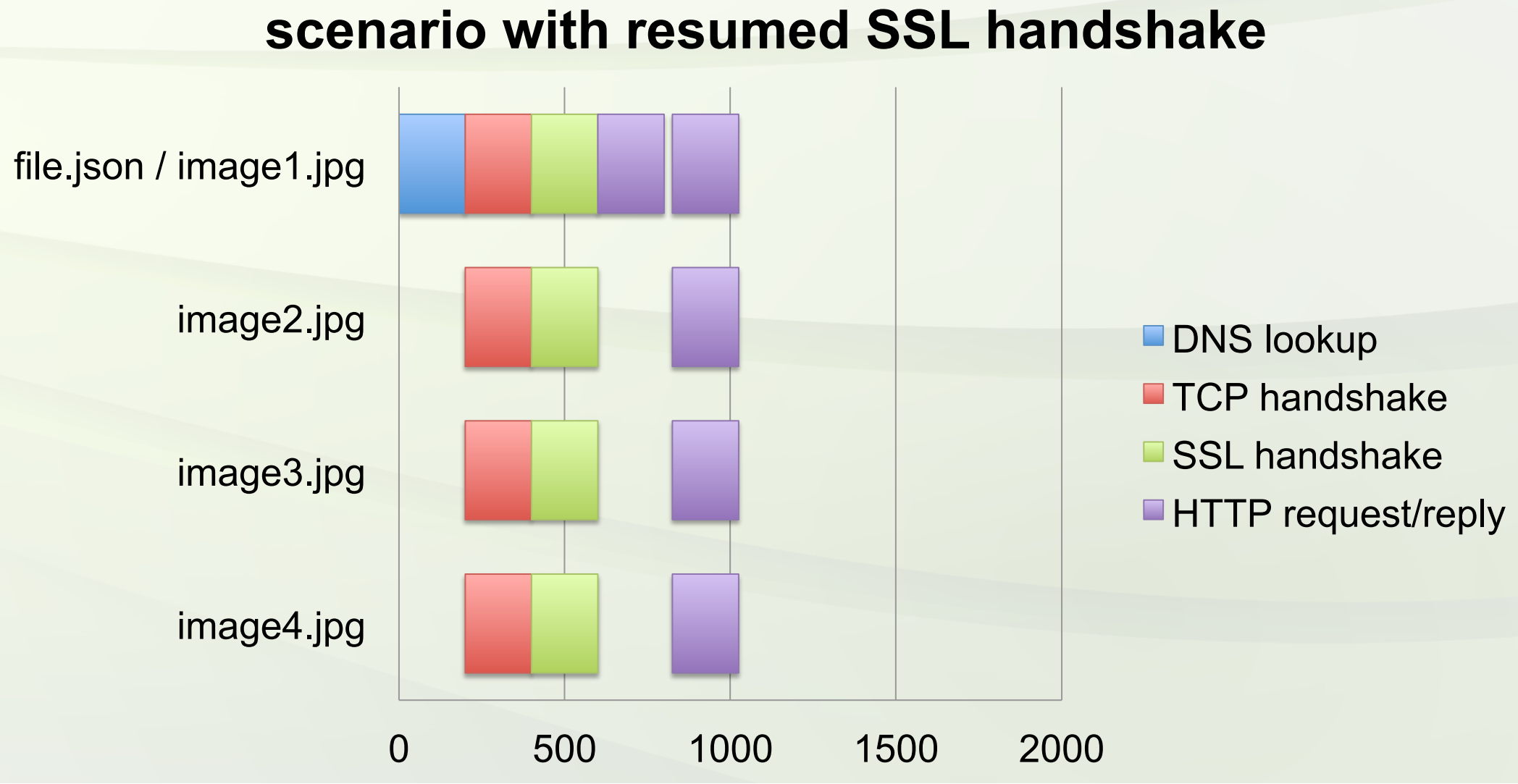
```
QByteArray usedSession = ... // load from disk, database etc.
QNetworkRequest request(url);

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

request.setSslConfiguration(sslConfiguration);
networkAccessManager->get(request);
```

SSL: store and resume TLS session tickets




scenario with full SSL handshake
file.json / image1.jpg
image2.jpg
image3.jpg
image4.jpg
0
500
1000
1500
2000
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
0
500
1000
1500
2000
DNS lookup
TCP handshake
SSL handshake
HTTP request/reply

full blog post about TLS session tickets at

http://ow.ly/pe7Ri

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

- use the cache:

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

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

networkAccessManager->setCache(cache);
```

- … or even prefer the cache

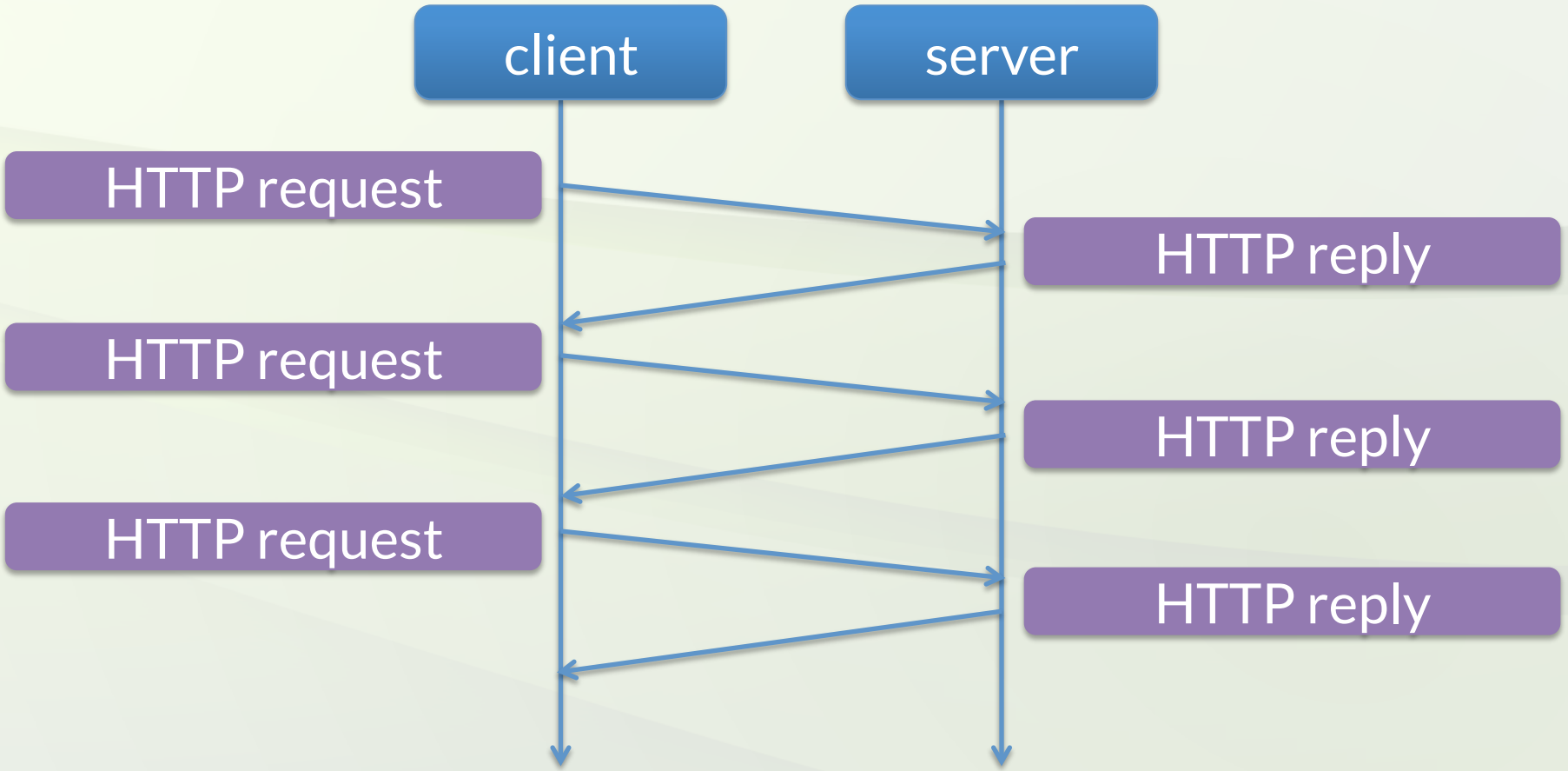
(loads more data from the cache):

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

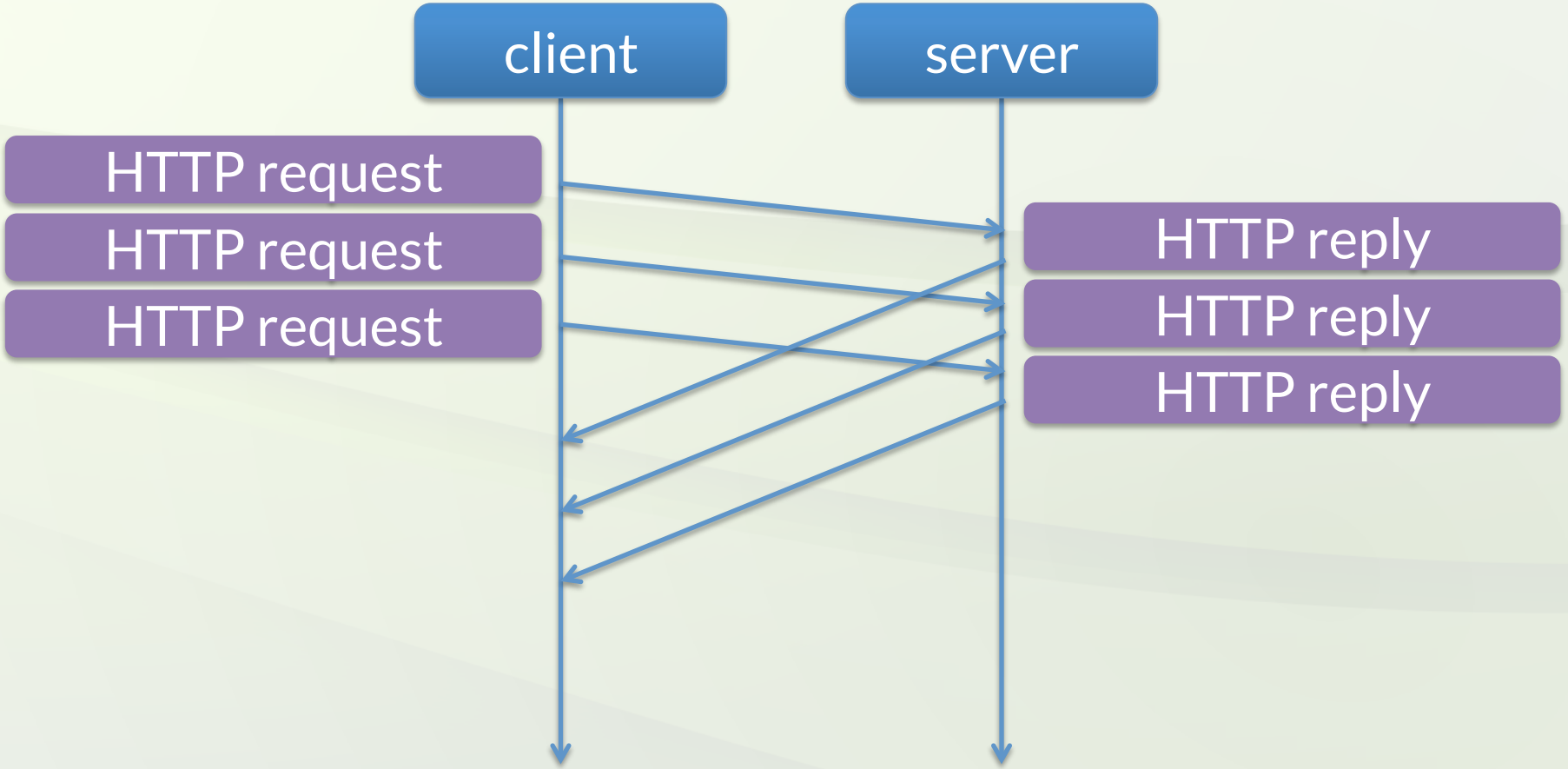
- enable pipelining:

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

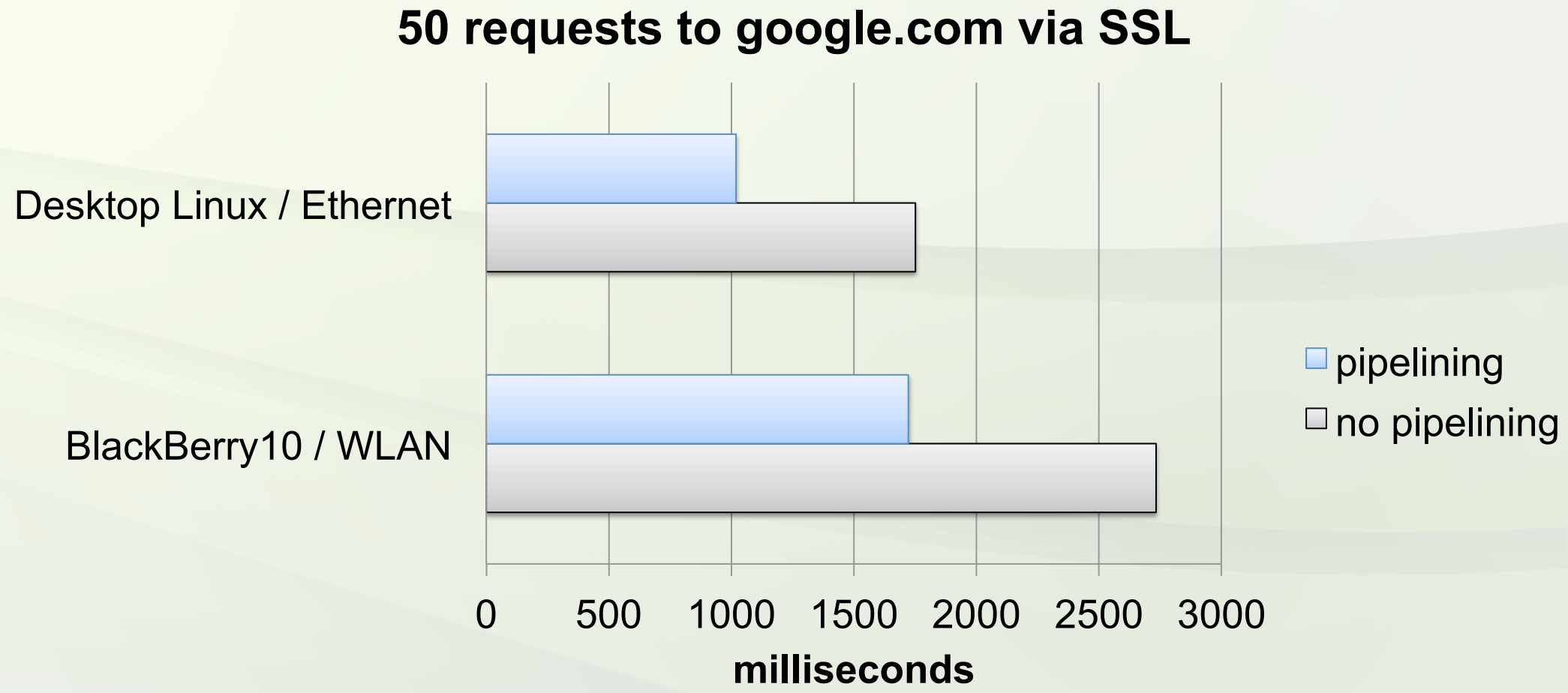
no pipelining:



pipelining:

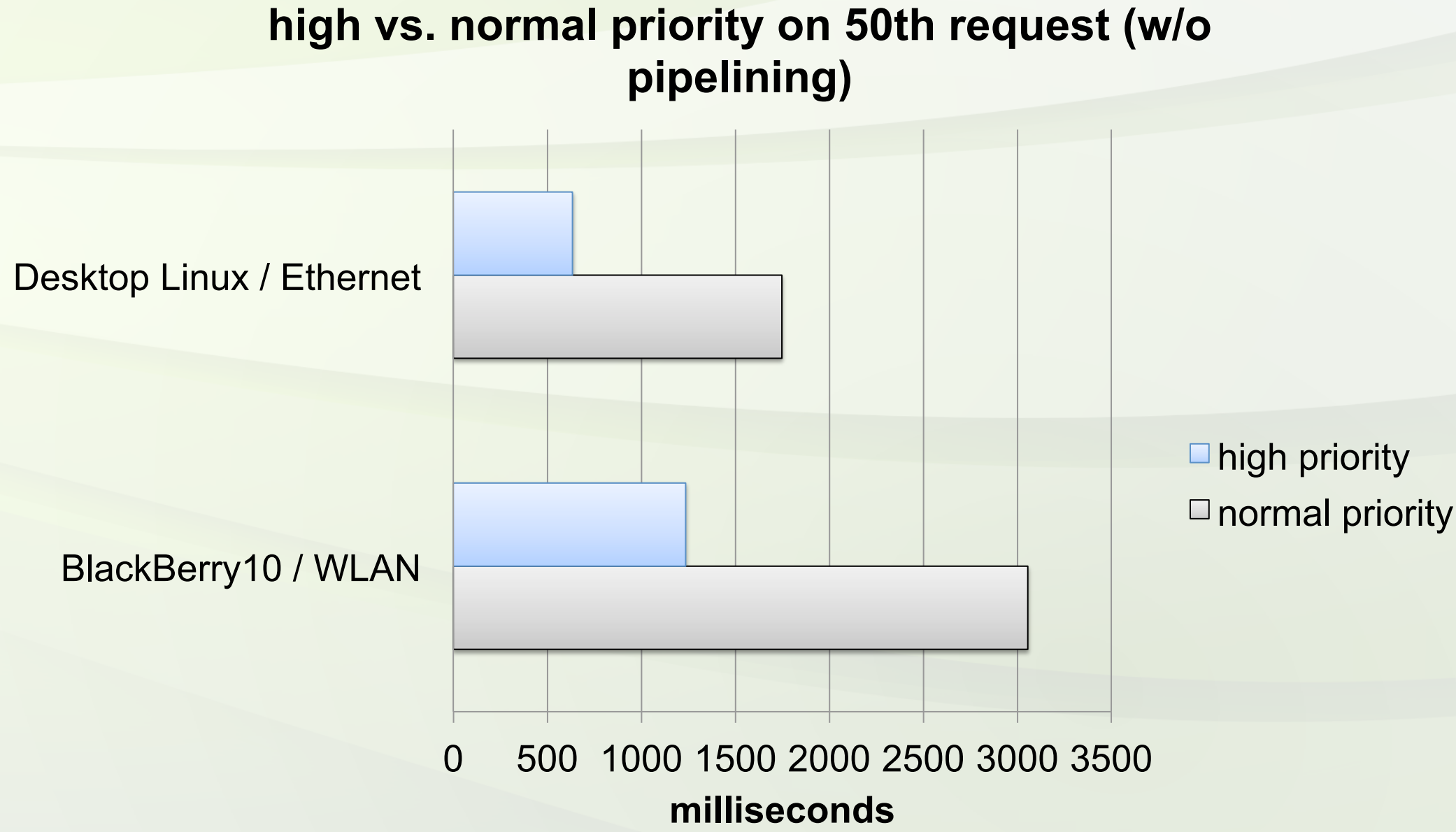

client
server
HTTP request
HTTP request
HTTP request
HTTP reply
HTTP reply
HTTP reply

pipelining helps with many requests to a server:



- set priority:

```
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
high priority
normal priority
0
500
1000
1500
2000
2500
3000
3500
milliseconds

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

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

Outlook

Qt Developer Days 2013

outside QtNetwork

- v4
- QML ahead-of-time-compilation ?

inside QtNetwork

- DNS: asynchronous lookups (c-ARES)
- TCP
- SSL
- HTTP: SPDY (5.3 ?), HTTP/2.0 ?, QUIC ?

Thank you!

Questions?

Qt Developer Days 2013

Feedback?

phartmann@blackberry.com

@peha23


Qt
Developer
Days
2013