Qt Signal Forwarding

The concept here is simple. You normally connect a signal to a slot on a local instance of an object. In the case of signal forwarding, you connect a signal on an object to a slot on an object that is running in a different process on either the same machine or another node on the network.

The RpcClient API, part of which is show below, gives you an overloaded bind() method. You can inspect the header in rpcclient/rpcclient.h

```
QUuid bind(const QByteArray& className, const QString& hostAddress, qint16 remot
ePort);
bool bind(QObject* client, const QString& hostAddress, qint16 remotePort);
```

The necessity of overloading is to accommodate both use cases. The first bind overload assumes that an object of the type 'className' has been registered at the host address and port supplied as arguments. The corresponding requirement for the server would be a call on the server to registerReceiver, where the receiver is of the same type as className. The client must take the returned QUuid for use in subsequent calls that connect signals with remote slots (below).

```
bool connectSignalToRemoteSlot(QObject *source
                             , const QByteArray& signal
                             , const QByteArray& slot
                             , const QUuid& remoteId
                             , Rpc::BlockingMode mode = Rpc::Asynchronous);
```

As you can see, the QUuid is a required argument. Signal forwarding can be seen in the example testrpcclient/testrpcserver applications in the project. Look at main.cpp in each case to see the calls made that subscribe a local object to a remote object and connects signals on one to slots on the other.

The mode of connection defaults as asynchronous. This tells the client not to wait for an acknowledgement.