Quassel IRC - Bug #1151

Quassel crash while afk

04/09/2012 08:45 AM - jotik

Status:	Resolved	**Start date:**	04/09/2012
Priority:	Normal	**Due date:**	
Assignee:	Sputnick	**% Done:**	100%
Category:	Quassel Client	**Estimated time:**	0.00 hour
Target version:	0.9.3		
Version:	0.8.0	**OS:**	Linux

Description

Quassel crash while I was away from the computer, asleep having weird dreams and generally having no good sense of the real world. As far as I know, my KDE session was locked, Quassel was previously minimized to tray and I didn't touch the computer during my sleep. When I returned and unlocked my computer, the KDE crash handler was already shown. After completing the bug-reporting wizard, the following information was shown and I was redirected here to file this bug. It seems to me, that the core itself didn't crash or restart.

From the KDE crash handler:

Application: quasselclient (v0.8.0 (dist-5988f4c))
KDE Platform Version: 4.8.2 (4.8.2) (Compiled from sources)
Qt Version: 4.7.4
Operating System: Linux 3.2.2-hardened-r1-arm x86_64
Distribution: "Gentoo Base System release 2.0.3"

-- Information about the crash:
<In detail, tell us what you were doing when the application crashed.>

-- Backtrace:
Application: Quassel IRC (quasselclient), signal: Aborted
[KCrash Handler]
#6 0x00006576ad6b19c5 in *_GI_raise (sig=6) at ../nptl/sysdeps/unix/sysv/linux/raise.c:64
#7 *0x00006576ad6b2cc5 in* *_GI_abort () at abort.c:92
#8 0x00006576b07429c4 in qt_message_output (msgType=<optimized out>, buf=<optimized out>) at global/qglobal.cpp:2291
#9 0x00006576b0742b91 in qt_message(QtMsgType, const char *, typedef __va_list_tag __va_list_tag *) (msgType=QtFatalMsg, msg=0x6576b08e1158 "ASSERT: \"%s\" in file %s, line %d", ap=0x75514d154e70) at global/qglobal.cpp:2337
#10 0x00006576b0742d65 in qFatal (msg=<optimized out>) at global/qglobal.cpp:2520
#11 0x000008eea2a33d98 in ChannelBufferItem::attachIrcChannel (this=0x8eea34b7a80, ircChannel=0x8eea5a107b0) at /usr/src/debug/net-irc/quassel-0.8.0/quassel-0.8.0/src/client/networkmodel.cpp:532
#12 0x000008eea2a33f65 in NetworkItem::attachIrcChannel (this=0x8eea2ef2b80, ircChannel=0x8eea5a107b0) at /usr/src/debug/net-irc/quassel-0.8.0/quassel-0.8.0/src/client/networkmodel.cpp:161
#13 0x000008eea2a491f7 in NetworkItem::qt_metacall (this=0x8eea2ef2b80, _c=QMetaObject::InvokeMetaMethod, _id=5, _a=0x75514d155170) at /usr/src/debug/net-irc/quassel-0.8.0/quassel-0.8.0_build/src/client/moc_networkmodel.cxx:99
#14 0x00006576b0882cff in QMetaObject::activate (sender=0x8eea35010f0, m=<optimized out>, local_signal_index=<optimized out>, argv=0x75514d155170) at kernel/qobject.cpp:3278
#15 0x000008eea2a978b2 in Network::ircChannelAdded (this=<optimized out>, _t1=0x8eea5a107b0) at /usr/src/debug/net-irc/quassel-0.8.0/quassel-0.8.0_build/src/common/moc_network.cxx:434
#16 0x000008eea2a6ce3d in Network::newIrcChannel (this=0x8eea35010f0, channelname=..., initData=<optimized out>) at /usr/src/debug/net-irc/quassel-0.8.0/quassel-0.8.0/src/common/network.cpp:289
#17 0x000008eea2a97d32 in addIrcChannel (channel=<optimized out>, this=0x8eea35010f0) at /usr/src/debug/net-irc/quassel-0.8.0/quassel-0.8.0/src/common/network.h:251
#18 Network::qt_metacall (this=0x8eea35010f0, _c=QMetaObject::InvokeMetaMethod, _id=39, _a=0x75514d155520) at /usr/src/debug/net-irc/quassel-0.8.0/quassel-0.8.0_build/src/common/moc_network.cxx:264
#19 0x000008eea2a84d31 in SignalProxy::invokeSlot (this=<optimized out>, receiver=0x8eea35010f0, methodId=49, params=..., returnValue=...) at /usr/src/debug/net-irc/quassel-0.8.0/quassel-0.8.0/src/common/signalproxy.cpp:811
#20 0x000008eea2a86580 in SignalProxy::handleSync (this=0x8eea2ecbbd0, sender=0x8eea2fadee0, params=...) at /usr/src/debug/net-irc/quassel-0.8.0/quassel-0.8.0/src/common/signalproxy.cpp:671
#21 0x000008eea2a86b6d in SignalProxy::receivePeerSignal (this=0x8eea2ecbbd0, sender=0x8eea2fadee0, requestType=<optimized out>, params=...) at /usr/src/debug/net-irc/quassel-0.8.0/quassel-0.8.0/src/common/signalproxy.cpp:607
#22 0x000008eea2a871d6 in SignalProxy::receivePackedFunc (this=0x8eea2ecbbd0, sender=0x8eea2fadee0, packedFunc=...) at /usr/src/debug/net-irc/quassel-0.8.0/quassel-0.8.0/src/common/signalproxy.cpp:581
#23 0x000008eea2a874a6 in SignalProxy::dataAvailable (this=0x8eea2ecbbd0) at

/usr/src/debug/net-irc/quassel-0.8.0/quassel-0.8.0/src/common/signalproxy.cpp:832
#24 0x000008eea2a99911 in SignalProxy::qt_metacall (this=0x8eea2ecbbd0, _c=QMetaObject::InvokeMetaMethod, _id=<optimized out>, _a=0x75514d155bb0) at /usr/src/debug/net-irc/quassel-0.8.0/quassel-0.8.0_build/src/common/moc_signalproxy.cxx:113
#25 0x00006576b0882cff in QMetaObject::activate (sender=0x8eea3023420, m=<optimized out>, local_signal_index=<optimized out>, argv=0x0) at kernel/qobject.cpp:3278
#26 0x00006576af63426e in QSslSocketBackendPrivate::transmit (this=0x8eea302dc20) at ssl/qsslsocket_openssl.cpp:1049
#27 0x00006576af62d8f6 in QSslSocket::qt_metacall (this=0x8eea3023420, _c=QMetaObject::InvokeMetaMethod, _id=<optimized out>, _a=0x75514d156ec0) at .moc/debug-shared/moc_qsslsocket.cpp:121
#28 0x00006576b0882cff in QMetaObject::activate (sender=0x8eea2eef650, m=<optimized out>, local_signal_index=<optimized out>, argv=0x0) at kernel/qobject.cpp:3278
#29 0x00006576af60ebbf in QAbstractSocketPrivate::canReadNotification (this=0x8eea33e9520) at socket/qabstractsocket.cpp:643
#30 0x00006576af5f8541 in QReadNotifier::event (this=<optimized out>, e=<optimized out>) at socket/qnativesocketengine.cpp:1103
#31 0x00006576afa691e4 in QApplicationPrivate::notify_helper (this=0x8eea2d8c010, receiver=0x8eea3016e30, e=0x75514d1574b0) at kernel/qapplication.cpp:4481
#32 0x00006576afa6ebf8 in QApplication::notify (this=<optimized out>, receiver=0x8eea3016e30, e=0x75514d1574b0) at kernel/qapplication.cpp:4360
#33 0x00006576ae6cd036 in KApplication::notify (this=0x75514d157950, receiver=0x8eea3016e30, event=0x75514d1574b0) at /usr/src/debug/kde-base/kdelibs-4.8.2/kdelibs-4.8.2/kdeui/kernel/kapplication.cpp:311
#34 0x00006576b0867cab in QCoreApplication::notifyInternal (this=0x75514d157950, receiver=0x8eea3016e30, event=0x75514d1574b0) at kernel/qcoreapplication.cpp:787
#35 0x00006576b089d7a3 in sendEvent (event=0x75514d1574b0, receiver=<optimized out>) at kernel/qcoreapplication.h:215
#36 socketNotifierSourceDispatch (source=0x8eea2d8e9b0) at kernel/qeventdispatcher_glib.cpp:110
#37 0x00006576acf82ada in g_main_dispatch (context=0x8eea2d8e840) at gmain.c:2441
#38 g_main_context_dispatch (context=0x8eea2d8e840) at gmain.c:3011
#39 0x00006576acf83308 in g_main_context_iterate (context=0x8eea2d8e840, block=1, dispatch=1, self=<optimized out>) at gmain.c:3089
#40 0x00006576acf83531 in g_main_context_iteration (context=0x8eea2d8e840, may_block=1) at gmain.c:3152
#41 0x00006576b089ddaf in QEventDispatcherGlib::processEvents (this=0x8eea2d89ca0, flags=<optimized out>) at kernel/qeventdispatcher_glib.cpp:422
#42 0x00006576afb34cb6 in QGuiEventDispatcherGlib::processEvents (this=<optimized out>, flags=<optimized out>) at kernel/qguieventdispatcher_glib.cpp:204
#43 0x00006576b08661a2 in QEventLoop::processEvents (this=<optimized out>, flags=...) at kernel/qeventloop.cpp:149
#44 0x00006576b0866515 in QEventLoop::exec (this=0x75514d157720, flags=...) at kernel/qeventloop.cpp:201
#45 0x00006576b086c9d9 in QCoreApplication::exec () at kernel/qcoreapplication.cpp:1064
#46 0x000008eea28c3d15 in main (argc=1, argv=0x75514d157c68) at /usr/src/debug/net-irc/quassel-0.8.0/quassel-0.8.0/src/common/main.cpp:144

PS: This bugtracker prevents me from setting the Quassel version to 0.8.0.

Related issues:

Has duplicate Quassel IRC - Bug #1152: Assert in NetworkModel was false	**Resolved**	**04/14/2012**
Has duplicate Quassel IRC - Bug #1058: Crash related to getting bounced off f...	**Resolved**	**03/03/2011**
Has duplicate Quassel IRC - Bug #1034: Quassel crashes sometimes when left id...	**Resolved**	**10/21/2010**

Associated revisions

Revision 44212b21 - 03/02/2014 09:19 PM - Manuel Nickschas

Simplify clean-up-on-network-disconnect handling

Previously, disconnecting from a network would trigger lots of unnecessary stuff: because the NetworkModel needs to remove affected IrcChannels and IrcUsers from its model items, we would trigger (on the core side!) an IrcUser::quit() for every known user in that network, which would then remove itself from all channels it's in, triggering the corresponding updates in the related SyncableObjects, which would send lots of signals to the client which would then perform its own cleanups per IrcUser, followed by throwing away all IrcChannels and IrcUsers in that network anyway. By the time we reached Network::removeChansAndUsers(), everything would actually be already all gone triggered through syncobject updates.

Except in some rare cases when there was still something left behind, triggering the dreaded "!_ircChannel && ircChannel" assert, that users have been reporting for years. I still haven't figured out how that could possibly happen.

In any case, the only side effect that explicit call to IrcUser::quit() was supposed to trigger was the removal of the relevant references in NetworkModel's items. So now we just brutally delete all IrcUsers and IrcChannels on disconnect, and have the NetworkModel items listen to the relevant destroyed() signals so

they can do their cleanup. This saves us from sending lots of stuff over the network, and also should fix the assert (which we've replaced by a warning now, just in case).

Fixes #1151 and a bunch of duplicates.

Revision 67074389 - 03/02/2014 09:30 PM - Manuel Nickschas

Simplify clean-up-on-network-disconnect handling

Previously, disconnecting from a network would trigger lots of unnecessary stuff: because the NetworkModel needs to remove affected IrcChannels and IrcUsers from its model items, we would trigger (on the core side!) an IrcUser::quit() for every known user in that network, which would then remove itself from all channels it's in, triggering the corresponding updates in the related SyncableObjects, which would send lots of signals to the client which would then perform its own cleanups per IrcUser, followed by throwing away all IrcChannels and IrcUsers in that network anyway. By the time we reached Network::removeChansAndUsers(), everything would actually be already all gone triggered through syncobject updates.

Except in some rare cases when there was still something left behind, triggering the dreaded "!_ircChannel && ircChannel" assert, that users have been reporting for years. I still haven't figured out how that could possibly happen.

In any case, the only side effect that explicit call to IrcUser::quit() was supposed to trigger was the removal of the relevant references in NetworkModel's items. So now we just brutally delete all IrcUsers and IrcChannels on disconnect, and have the NetworkModel items listen to the relevant destroyed() signals so they can do their cleanup. This saves us from sending lots of stuff over the network, and also should fix the assert (which we've replaced by a warning now, just in case).

Fixes #1151 and a bunch of duplicates.

History

#1 - 03/02/2014 03:21 PM - Sputnick

- Status changed from New to Confirmed

- Assignee set to Sputnick

- Target version set to 0.9.3

- Version changed from 0.8-pre to 0.8.0

This happens rarely if the ircd hickups (has mostly been observed with Freenode). Unfortunately, I didn't manage to reproduce this with a local ircd, so debugging this will be hard.

I have a feeling it's triggered by a reconnect before the clean-up has finished, so I'll dive into the code and see if I can find a possible culprit somewhere.

#2 - 03/02/2014 09:32 PM - Anonymous

- Status changed from Confirmed to Resolved

- % Done changed from 0 to 100

Applied in changeset quassel|commit:670743897bc3b96b17a7fdad3f672637b46ea302.