Quassel IRC - Bug #358

quassel core crashes when the user runs out of disk space

10/18/2008 07:22 PM - hades

Status:	Resolved	**Start date:**	
Priority:	High	**Due date:**	
Assignee:	EgS	**% Done:**	0%
Category:	Quassel Core	**Estimated time:**	0.00 hour
Target version:			
Version:	0.13.1	**OS:**	Any

Description

When the user running quassel core runs out of disk space, the core crashes.

2008-10-18 21:02:33 Error: unhandled Error in QSqlQuery!
2008-10-18 21:02:33 Error: last Query:
INSERT INTO backlog (time, bufferid, type, flags, senderid, message)
VALUES (:time, :bufferid, :type, :flags, (SELECT senderid FROM sender WHERE sender = :sender), :message)
2008-10-18 21:02:33 Error: executed Query:
INSERT INTO backlog (time, bufferid, type, flags, senderid, message)
VALUES (?, ?, ?, ?, (SELECT senderid FROM sender WHERE sender = ?), ?)
2008-10-18 21:02:33 Error: bound Values:
2008-10-18 21:02:33 Error: 0 : 17
2008-10-18 21:02:33 Error: 1 : 0
2008-10-18 21:02:33 Error: 2 : PReDiToR, what would I need other than the driver?
2008-10-18 21:02:33 Error: 3 : fuxxy!n=Josh@c-76-31-189-187.hsd1.tx.comcast.net
2008-10-18 21:02:33 Error: 4 : 1224349353
2008-10-18 21:02:33 Error: 5 : 1
2008-10-18 21:02:33 Error: Error Number: 13
2008-10-18 21:02:33 Error: Error Message: database or disk is full Unable to fetch row
2008-10-18 21:02:33 Error: Driver Message: Unable to fetch row
2008-10-18 21:02:33 Error: DB Message: database or disk is full
ASSERT: "msgId.isValid()" in file /var/abs/local/yaourtbuild/quassel-git/src/quassel-build/src/core/sqlitestorage.cpp, line 672

1. 0 quasselcore 0x081023f2 Quassel::handleCrash()
2. 1 quasselcore 0x0810302c Quassel::handleSignal(int)
3. 2 0xffffffffb80a8400 __kernel_sigreturn
4. 3 0xffffffffb80a8424 __kernel_vsyscall
5. 4 libc.so.6 0xffffffffb7ac9720 gsignal
6. 5 libc.so.6 0xffffffffb7acb058 abort
7. 6 libQtCore.so.4 0xffffffffb7f08815 qt_message_output(QtMsgType, char const*)
8. 7 libQtCore.so.4 0xffffffffb7f088c6 qFatal(char const*, ...)
9. 8 libQtCore.so.4 0xffffffffb7f08955 qt_assert(char const*, char const*, int)
10. 9 quasselcore 0x0808c75c SqliteStorage::logMessage(Message)
11. 10 quasselcore 0x08079c9c Core::storeMessage(Message const&)
12. 11 quasselcore 0x080a1499 CoreSession::recvMessageFromServer(Message::Type, BufferInfo::Type, QString, QString, QString, QFlags<Message::Flag>)
13. 12 quasselcore 0x080bac26 CoreSession::qt_metacall(QMetaObject::Call, int, void**)
14. 13 libQtCore.so.4 0xffffffffb7ffb89b QMetaObject::activate(QObject*, int, int, void**)
15. 14 libQtCore.so.4 0xffffffffb7ffbb70 QMetaObject::activate(QObject*, QMetaObject const*, int, int, void**)
16. 15 quasselcore 0x080bb6e5 NetworkConnection::displayMsg(Message::Type, BufferInfo::Type, QString, QString, QString, QFlags<Message::Flag>)
17. 16 quasselcore 0x080bbdc4 NetworkConnection::qt_metacall(QMetaObject::Call, int, void**)
18. 17 libQtCore.so.4 0xffffffffb7ffb89b QMetaObject::activate(QObject*, int, int, void**)
19. 18 libQtCore.so.4 0xffffffffb7ffbb70 QMetaObject::activate(QObject*, QMetaObject const*, int, int, void**)
20. 19 quasselcore 0x080b95f5 BasicHandler::displayMsg(Message::Type, BufferInfo::Type, QString, QString, QString, QFlags<Message::Flag>)
21. 20 quasselcore 0x080bd453 BasicHandler::displayMsg(Message::Type, QString, QString, QString, QFlags<Message::Flag>)
22. 21 quasselcore 0x080c680c CtcpHandler::parse(Message::Type, QString const&, QString const&, QByteArray const&)
23. 22 quasselcore 0x080d6927 IrcServerHandler::handlePrivmsg(QString const&, QList<QByteArray> const&)
24. 23 quasselcore 0x080dc80d IrcServerHandler::qt_metacall(QMetaObject::Call, int, void**)
25. 24 quasselcore 0x080bd87f BasicHandler::handle(QString const&, QGenericArgument, QGenericArgument,

QGenericArgument, QGenericArgument, QGenericArgument, QGenericArgument, QGenericArgument, QGenericArgument, QGenericArgument)
26. 25 quasselcore 0x080ce6e1 IrcServerHandler::handleServerMsg(QByteArray)
27. 26 quasselcore 0x080a7b17 NetworkConnection::socketHasData()
28. 27 quasselcore 0x080bb893 NetworkConnection::qt_metacall(QMetaObject::Call, int, void**)
29. 28 libQtCore.so.4 0xffffffffb7ffb89b QMetaObject::activate(QObject*, int, int, void**)
30. 29 libQtCore.so.4 0xffffffffb7ffbdf2 QMetaObject::activate(QObject*, QMetaObject const*, int, void**)
31. 30 libQtCore.so.4 0xffffffffb8030097 QIODevice::readyRead()
32. 31 libQtNetwork.so.4 0xffffffffb7e8cf27 0x0000000
33. 32 libQtNetwork.so.4 0xffffffffb7e9060e QSslSocket::qt_metacall(QMetaObject::Call, int, void**)
34. 33 libQtCore.so.4 0xffffffffb7ffb89b QMetaObject::activate(QObject*, int, int, void**)
35. 34 libQtCore.so.4 0xffffffffb7ffbdf2 QMetaObject::activate(QObject*, QMetaObject const*, int, void**)
36. 35 libQtCore.so.4 0xffffffffb8030097 QIODevice::readyRead()
37. 36 libQtNetwork.so.4 0xffffffffb7e791c2 0x0000000
38. 37 libQtNetwork.so.4 0xffffffffb7e6a72b 0x0000000
39. 38 libQtNetwork.so.4 0xffffffffb7e6b926 0x0000000
40. 39 libQtCore.so.4 0xffffffffb7fe93a3 QCoreApplicationPrivate::notify_helper(QObject*, QEvent*)
41. 40 libQtCore.so.4 0xffffffffb7fe9a43 QCoreApplication::notify(QObject*, QEvent*)
42. 41 libQtCore.so.4 0xffffffffb7fea041 QCoreApplication::notifyInternal(QObject*, QEvent*)
43. 42 libQtCore.so.4 0xffffffffb8010957 0x0000000
44. 43 libglib-2.0.so.0 0xffffffffb79fd2b1 g_main_context_dispatch
45. 44 libglib-2.0.so.0 0xffffffffb7a00943 0x0000000
46. 45 libglib-2.0.so.0 0xffffffffb7a00b01 g_main_context_iteration
47. 46 libQtCore.so.4 0xffffffffb80106c8 QEventDispatcherGlib::processEvents(QFlags<QEventLoop::ProcessEventsFlag>)
48. 47 libQtCore.so.4 0xffffffffb7fe87aa QEventLoop::processEvents(QFlags<QEventLoop::ProcessEventsFlag>)
49. 48 libQtCore.so.4 0xffffffffb7fe896a QEventLoop::exec(QFlags<QEventLoop::ProcessEventsFlag>)
50. 49 libQtCore.so.4 0xffffffffb7f0cab3 QThread::exec()
51. 50 quasselcore 0x08082db2 SessionThread::run()
52. 51 libQtCore.so.4 0xffffffffb7f0f910 0x0000000
53. 52 libpthread.so.0 0xffffffffb79b2145 0x0000000
54. 53 libc.so.6 0xffffffffb7b6c63e clone

History

#1 - 10/18/2008 08:18 PM - EgS

In contrast to other IRC clients, the logging of messages is a crucial part of Quassel. The (Quassel)Client can't work correctly if the core sends a Message without a MessageId. Those MessageIds are generated in the storage backend (currently only the Sqlite DB).
To make it short: Quassel won't ever work without free HDD space, so we cannot do much more than raising some fatal error.