ndn-cxx - Bug #3248

Face::Impl::ensureConnected segfault if Face is destroyed without io.poll

10/08/2015 12:47 PM - Junxiao Shi

Status:	Closed	**Start date:**	10/08/2015
Priority:	Normal	**Due date:**	
Assignee:	Junxiao Shi	**% Done:**	100%
Category:	Base	**Estimated time:**	3.00 hours
Target version:	v0.5		

Description

Snippet to reproduce:

```
// g++ -o x -std=c++0x x.cpp $(pkg-config --cflags --libs libndn-cxx)
#include <ndn-cxx/face.hpp>
#include <boost/asio.hpp>

using namespace ndn;

int main()
{
  boost::asio::io_service io;

  auto face = new Face(io);
  delete face;

  io.poll(); // crash

  return 0;
}
```

Stack trace:

```
(gdb) bt
#0  0x00007ffff7884d93 in ndn::Face::Impl::ensureConnected (this=0x0, wantResume=false) at ../src/
detail/face-impl.hpp:126
#1  0x00007ffff78792ba in operator() (this=0x7fffffffe290) at ../src/face.cpp:131
#2  0x00007ffff787f375 in boost::asio::asio_handler_invoke<ndn::Face::construct(std::shared_ptr<nd
n::Transport>, ndn::security::KeyChain&)::<lambda()> >(struct {...}, ...) (function=...) at /usr/i
nclude/boost/asio/handler_invoke_hook.hpp:64
#3  0x00007ffff787eeb4 in boost_asio_handler_invoke_helpers::invoke<ndn::Face::construct(std::shar
ed_ptr<ndn::Transport>, ndn::security::KeyChain&)::<lambda()>, ndn::Face::construct(std::shared_pt
r<ndn::Transport>, ndn::security::KeyChain&)::<lambda()> >(struct {...} &, struct {...} &) (functi
on=..., context=...) at /usr/include/boost/asio/detail/handler_invoke_helpers.hpp:39
#4  0x00007ffff787dad0 in boost::asio::detail::completion_handler<ndn::Face::construct(std::shared
_ptr<ndn::Transport>, ndn::security::KeyChain&)::<lambda()> >::do_complete(boost::asio::detail::io
_service_impl *, boost::asio::detail::operation *, const boost::system::error_code &, std::size_t)
 (owner=0x616780, base=0x62f7b0) at /usr/include/boost/asio/detail/completion_handler.hpp:67
#5  0x000000000040a850 in boost::asio::detail::task_io_service_operation::complete(boost::asio::de
tail::task_io_service&, boost::system::error_code const&, unsigned long) ()
#6  0x000000000040b58b in boost::asio::detail::task_io_service::do_poll_one(boost::asio::detail::s
coped_lock<boost::asio::detail::posix_mutex>&, boost::asio::detail::op_queue<boost::asio::detail::
task_io_service_operation>&, boost::system::error_code const&) ()
#7  0x000000000040b2ba in boost::asio::detail::task_io_service::poll(boost::system::error_code&) (
)
#8  0x000000000040b895 in boost::asio::io_service::poll() ()
#9  0x0000000000409260 in main ()
```

History

#1 - 12/16/2015 10:58 PM - Alex Afanasyev

- Target version set to v0.4

#2 - 01/16/2016 06:47 PM - Davide Pesavento

- Target version changed from v0.4 to v0.5

#3 - 07/19/2016 07:02 AM - Junxiao Shi

Root cause:
Face::construct invokes m_ioService.post([=] { m_impl->ensureConnected(false); });, where m_impl is a member field of unique_ptr<Impl> type.
When the face is deallocated, Impl is deallocated as well, and m_impl variable itself is also deallocated.
When io.poll() or io.run() is invoked the next time, m_impl variable is on freed memory, and everything after that is undefined behavior.

This problem not only occurs in a seemingly corner case "Face destroyed without io.poll()", but also can occur with all other uses of io.post and io.dispatch (in multi-threaded program).

A solution is to change m_impl to shared_ptr<Impl>, and capture a weak_ptr<Impl> in io.post and io.dispatch calls:

```
weak_ptr<Impl> implWeak(m_impl);
m_ioService.post([=] {
  auto impl = implWeak.lock();
  if (impl != nullptr) {
    impl->ensureConnected(false);
  }
});
```

#4 - 07/21/2016 01:48 PM - Junxiao Shi

20160721 call agrees with the analysis in note-3.

Alex believes there are more bugs from the same cause.

#5 - 08/04/2016 06:48 PM - Junxiao Shi

- Status changed from New to In Progress

- Assignee set to Junxiao Shi

- Estimated time set to 3.00 h

#6 - 08/04/2016 08:20 PM - Junxiao Shi

- % Done changed from 0 to 40

https://gerrit.named-data.net/3025 corrects code style in face.* detail/* interest-filter.*.

#7 - 08/08/2016 03:52 PM - Junxiao Shi

- Status changed from In Progress to Code review

- % Done changed from 40 to 100

https://gerrit.named-data.net/3051 has the fix.

In patchset1, I only added one test case for the scenario described in issue description.
Other io.post and io.dispatch calls may have the same issue; I fixed them but didn't add test cases.

#8 - 08/16/2016 07:43 AM - Junxiao Shi

- Status changed from Code review to Closed