ndn-cxx - Feature #3913

Add DummyForwarder to utils

01/07/2017 03:14 PM - Zhiyi Zhang

Status:	Closed	**Start date:**	01/07/2017
Priority:	Normal	**Due date:**	
Assignee:	Zhiyi Zhang	**% Done:**	100%
Category:	Utils	**Estimated time:**	0.00 hour
Target version:	v0.7		

Description

In many test cases, we want to test packet exchange between faces. Though we have DummyClientFace, we do not have a DummyForwarder which can forward packets among faces.
This DummyForwarder was first designed for Name-based Access Control protocol tests: https://gerrit.named-data.net/#/c/2599/12.
We move the DummyForwarder to ndn-cxx.

To add Face to DummyForwarder, the steps is like:

```
DummyForwarder forwarder(io_service);
auto face = make_shared<DummyClientFace>(io_service, keychain, {true, true});
size_t faceID = forwarder.addFace(face);
forwarder.registerPrefix("/face", faceID);
```

To pass the packets, the steps is like:

```
do {
  advanceClocks(time::milliseconds(10), 20);
} while (forwarder.passInterest() || forwarder.passData());
```

History

#1 - 01/07/2017 03:17 PM - Zhiyi Zhang

- Status changed from New to Code review

#2 - 01/07/2017 08:46 PM - Junxiao Shi

> To pass the packets, the steps is like:
>
> ```
> do {
>   advanceClocks(time::milliseconds(10), 20);
> } while (forwarder.passInterest() || forwarder.passData());
> ```

Why not providing a processEvents or passPackets function that takes care of this?

#3 - 07/06/2017 11:53 PM - Junxiao Shi

https://gerrit.named-data.net/3558

#4 - 07/06/2017 11:53 PM - Junxiao Shi

- Category set to Utils

#5 - 12/04/2017 02:48 PM - Zhiyi Zhang

I suggest we abandon the current complex effort and just make a really dummy forwarder. In the latest commit, I ported the DummyForwarder from ChronoShare to ndn-cxx.

#6 - 12/07/2017 09:39 AM - Junxiao Shi

- Target version set to v0.7

As indicated in #3940-18, the dummy forwarder does not belong to src/util/. It should be placed in src/dummyfw/ folder.

> just make a really dummy forwarder

Can you define the goals and non-goals of the "really dummy forwarder"?

#7 - 12/07/2017 10:44 AM - Davide Pesavento

Junxiao Shi wrote:

> As indicated in #3940-18, the dummy forwarder does not belong to src/util/. It should be placed in src/dummyfw/ folder.

I said that when the dummy forwarder implementation consisted of several translation units. It's just one translation unit now, it makes no sense to put it in its own directory.

#8 - 12/07/2017 11:29 AM - Zhiyi Zhang

Junxiao Shi wrote:

> > just make a really dummy forwarder
>
> Can you define the goals and non-goals of the "really dummy forwarder"?

It means the dummy forwarder simply broadcasts all the Interest packets and Data packets from sender face to all the other faces.

#9 - 12/07/2017 11:47 AM - Junxiao Shi

> the dummy forwarder simply broadcasts all the Interest packets and Data packets from sender face to all the other faces.

This isn't a forwarder, dummy or not, and shouldn't be labeled as such.
It's simply a "link".

I suggest this API:

```
void DummyClientFace::linkTo(DummyClientFace&);
```

Any number of dummy client faces can be linked together.

#10 - 12/07/2017 02:19 PM - Zhiyi Zhang

Junxiao Shi wrote:

> I suggest this API:
>
> ```
> void DummyClientFace::linkTo(DummyClientFace&);
> ```
>
> Any number of dummy client faces can be linked together.

I disagree. If we have n faces, we then need to do n(n-1)/2 times linkTo() to build the mesh link.

#11 - 12/07/2017 02:39 PM - Junxiao Shi

If we have n faces, we then need to do n(n-1)/2 times linkTo() to build the mesh link.

Actually not. DummyClientFace::linkTo is the public API. The class has an internal shared_ptr pointing to an object that remembers all the faces that are linked together. There are only n-1 invocations.

#12 - 12/07/2017 03:38 PM - Zhiyi Zhang

Then we need to define that object and we can change the API to:

```
void DummyClientFace::linkTo(Object&);
```

Or we have the current implementation to be that object, but call it DummyBroadcastLink or something. Then it would be:

```
class DummyBroadcastLink
{
public:
  std::vector<shared_ptr<DummyClientFace>> faces;
}

void DummyClientFace::linkTo(DummyBroadcastLink&);
```

#13 - 12/08/2017 06:12 AM - Junxiao Shi

No. It's like this:

```
class DummyClientFace
{
public:
  ~DummyClientFace();

  class AlreadyLinkedException;

  void linkTo(DummyClientFace& other);
  void unlink();

private:
  class BcastLink;

  shared_ptr<BcastLink> m_bcastLink;
};

DummyClientFace::~DummyClientFace()
{
  this->unlink();
}
```

```
void DummyClientFace::linkTo(DummyClientFace& other)
{
  if (m_bcastLink != nullptr && other.m_bcastLink != nullptr) {
    if (m_bcastLink != other.m_bcastLink) {
      // already on different links
      throw AlreadyLinkedException();
    }
    return; // already on same link
  }

  if (m_bcastLink == nullptr && other.m_bcastLink != nullptr) {
    m_bcastLink = other.m_bcastLink;
    m_bcastLink->add(*this);
  }
  else if (m_bcastLink != nullptr && other.m_bcastLink == nullptr) {
    other.m_bcastLink = m_bcastLink;
    m_bcastLink->add(other);
  }
  else {
    m_bcastLink = other.m_bcastLink = make_shared<BcastLink>();
    m_bcastLink->add(*this);
    m_bcastLink->add(other);
  }
}

void DummyClientFace::unlink()
{
  if (m_bcastLink == nullptr) {
    return;
  }
  m_bcastLink->remove(*this);
  m_bcastLink = nullptr;
}
```

DummyClientFace::BcastLink is a nested class. It appears in .cpp only and is not part of public API.

#14 - 12/26/2017 09:23 AM - Davide Pesavento

- Status changed from Code review to Closed

- % Done changed from 0 to 100

Commit 34429cc4f53fef200f63836361468bdc67d788cc added emulation of a broadcast link to DummyClientFace. It's not a dummy forwarder, but I believe it satisfies current needs, hence I'm closing this issue.