NFD - Bug #3642

Duplicate Nonce is used when forwarding a consumer retransmission

06/02/2016 03:20 PM - Junxiao Shi

Status:	Closed	**Start date:**	06/02/2016
Priority:	Normal	**Due date:**	
Assignee:	Junxiao Shi	**% Done:**	100%
Category:	Forwarding	**Estimated time:**	6.00 hours
Target version:	v0.5		

Description

Topology:

```
A--\
    \
B----R---P
    /
C--/
```

P publishes a live video stream with ndncon. Its prefix is statically registered on R.

A,B,C access this live stream simultaneously, and they start at roughly same time.

Expected: Most Interests are answered.

Actual: Many consumers are unable to receive Data.

History

#1 - 06/02/2016 03:52 PM - Junxiao Shi

- Assignee set to Junxiao Shi

This behavior was reported by Peter.
From the tcpdump trace and other diagnostic information provided, I can see that prefix registrations are successful, and congestion or packet loss is not a factor.

Part of ndndump log is interesting:
(host 2 is the router, host 3 is the producer, and other hosts are consumers)

```
vagrant@m0212:~$ ndndump -r dump.pcap | grep low/key/32/data/%00%00 | sed -e 's/From: 10.10.12.//' -e 's/, To:
 10.10.12./->/' -e 's/, Tunnel Type: UDP, / /' -e 's|/ndn/edu/ucla/remap/ndnrtc/user/client1/streams/camera/lo
w/key/32/data/%00%00|()|'
1464720224.570010 7->2 INTEREST: ()?ndn.MustBeFresh=1&ndn.InterestLifetime=3000&ndn.Nonce=2090128233
1464720224.570288 2->3 INTEREST: ()?ndn.MustBeFresh=1&ndn.InterestLifetime=3000&ndn.Nonce=2090128233
1464720224.586433 5->2 INTEREST: ()?ndn.MustBeFresh=1&ndn.InterestLifetime=3000&ndn.Nonce=2159426678
1464720224.586565 2->3 INTEREST: ()?ndn.MustBeFresh=1&ndn.InterestLifetime=3000&ndn.Nonce=2090128233
1464720224.587221 3->2 NACK: Duplicate, ()?ndn.MustBeFresh=1&ndn.InterestLifetime=3000&ndn.Nonce=2090128233
1464720224.587402 2->7 NACK: Duplicate, ()?ndn.MustBeFresh=1&ndn.InterestLifetime=3000&ndn.Nonce=2090128233
1464720224.587426 2->5 NACK: Duplicate, ()?ndn.MustBeFresh=1&ndn.InterestLifetime=3000&ndn.Nonce=2159426678
1464720224.612528 8->2 INTEREST: ()?ndn.MustBeFresh=1&ndn.InterestLifetime=3000&ndn.Nonce=2239689571
1464720224.615932 2->3 INTEREST: ()?ndn.MustBeFresh=1&ndn.InterestLifetime=3000&ndn.Nonce=2239689571
1464720224.623036 11->2 INTEREST: ()?ndn.MustBeFresh=1&ndn.InterestLifetime=3000&ndn.Nonce=3014351709
1464720224.628697 10->2 INTEREST: ()?ndn.MustBeFresh=1&ndn.InterestLifetime=3000&ndn.Nonce=3854802010
1464720224.662820 9->2 INTEREST: ()?ndn.MustBeFresh=1&ndn.InterestLifetime=3000&ndn.Nonce=3854802010
1464720224.662932 2->9 NACK: Duplicate, ()?ndn.MustBeFresh=1&ndn.InterestLifetime=3000&ndn.Nonce=3854802010
1464720224.676057 12->2 INTEREST: ()?ndn.MustBeFresh=1&ndn.InterestLifetime=3000&ndn.Nonce=17098276
1464720224.676168 2->3 INTEREST: ()?ndn.MustBeFresh=1&ndn.InterestLifetime=3000&ndn.Nonce=2239689571
1464720224.676734 3->2 NACK: Duplicate, ()?ndn.MustBeFresh=1&ndn.InterestLifetime=3000&ndn.Nonce=2239689571
1464720224.676910 2->8 NACK: Duplicate, ()?ndn.MustBeFresh=1&ndn.InterestLifetime=3000&ndn.Nonce=2239689571
1464720224.677082 2->10 NACK: Duplicate, ()?ndn.MustBeFresh=1&ndn.InterestLifetime=3000&ndn.Nonce=3854802010
1464720224.677111 2->11 NACK: Duplicate, ()?ndn.MustBeFresh=1&ndn.InterestLifetime=3000&ndn.Nonce=3014351709
1464720224.677129 2->12 NACK: Duplicate, ()?ndn.MustBeFresh=1&ndn.InterestLifetime=3000&ndn.Nonce=17098276
1464720224.698429 4->2 INTEREST: ()?ndn.MustBeFresh=1&ndn.InterestLifetime=2000&ndn.Nonce=3322721076
1464720225.538034 3->2 DATA: ()/16/960/928/4/33985/sl/low/32
1464720225.538206 2->4 DATA: ()/16/960/928/4/33985/sl/low/32
```

In the first 7 lines:

1. Consumer 7 expresses an Interest with Nonce 2090128233. Router forwards this Interest to producer.
2. 16ms later, consumer 5 expresses the same Interest with Nonce 2159426678. Router forwards this Interest to producer, with Nonce 2090128233.
3. Producer returns a Nack to router because it sees a duplicate Nonce.
4. Router has no alternate path to retry the Interest, so it returns the Nack to consumer 7 and 5.

In step2, the router should have forwarded the Interest with Nonce 2159426678.
There's no reason to use the Nonce from an older Interest.

This is the primary cause of this bug.
I'll review relevant code in forwarding pipelines to find out where's the problem.

The ndndump log also exposes some secondary issues:

- Consumer 10 and 9 are expressing the Interest with the same Nonce. This is probably an issue in random number generator seeding in ndn-cpp or ndncon.
- None of the consumers re-express the Interest after getting a Nack.

#2 - 06/02/2016 03:59 PM - Peter Gusev

> None of the consumers re-express the Interest after getting a Nack.

NACKs are not currently supported in NDN-RTC. Retransmissions follow other logic (playback deadlines). Is it expected that the App should always re-express after receiving NACK?

#3 - 06/03/2016 07:58 AM - Vince Lehman

- File ndndump-duplicate-nonce.txt added

Based on the description, I believe we may have previously seen this issue in our hyperbolic experiments. I've included a description of the problem and have attached an ndndump file:

> While running some experiments with link-state routing, I noticed some odd behavior in what I presume is the strategy. I was hoping you might be able to explain the behavior or point me in the right direction.
>
> In the experiment, the logs show: Node A sends an Interest with nonce=x; the strategy on each node covering the Interest's name is the Multicast strategy. Node A multicasts this Interest to its neighbor B; B multicasts the Interest to its neighbors, one of which is C; C multicasts the Interest to its neighbors, one of which is D; D multicasts the Interest to its neighbors, one of which is E; E does not have any neighbors other than D and so does not send an Interest. Six seconds later, node D again multicasts the Interest with nonce=x to its neighbors including E. Node D did not receive any additional Interest with nonce=x prior to performing the multicast again. In my understanding, the Multicast strategy should not perform router retransmission, but this is what the logs seem to show.

#4 - 06/05/2016 04:36 PM - Junxiao Shi

- Status changed from New to In Progress

- % Done changed from 0 to 10

The bug is in **outgoing Interest pipeline**.

https://github.com/named-data/NFD/blob/f3410d8c59b49eff99048e7df8a1599e4eed59f6/daemon/fw/forwarder.cpp#L283

According to design, the outgoing Interest is the last incoming Interest (that does not come from the outgoing face).
This is implemented as an std::max_element call with compare_pickInterest function as the comparison function.

STL algorithms expect the comparison function to return true when the first argument is less than the second.
In order for std::max_element to return the last incoming Interest, compare_pickInterest should return true if the first Interest arrives earlier than the second Interest.

However, the code is doing the opposite return a.getLastRenewed() > b.getLastRenewed();, so that the oldest incoming Interest gets picked instead.

This analysis is consistent with note-1 log, and the fix should be easy.

compare_pickInterest function was written pre-C++11, and is used only once.
As part of the patch, I'll rewrite it as a lambda to eliminate a bind call.

#5 - 06/05/2016 05:02 PM - Junxiao Shi

20160602 conference call discussed note-2 question.

I made the suggestion for consumer to retransmit the Interest after receiving a Nack because I have designed some strategies that rely on this assumption.
In particular, in those strategies, a consumer retransmission indicates there's possibly a problem with the current preferred path, so that the strategy should consider trying a different path.

Beichuan points out that consumer retransmission does not necessarily indicate a problem between current node and content source; instead, the Nack itself indicates this problem.
Thus, strategies should not assume the consumer would retransmit after receiving a Nack, and the consumer is free to decide whether to retransmit.

Furthermore, as indicated in note-1 log, ndn-rtc never retransmits the same Interest (same Name+Selectors) even after a timeout, but instead could send different Interests (eg. different Exclude selector).
This tells us that strategies should not rely on consumer retransmission as a signal of packet loss between current node and content source.

#6 - 06/05/2016 05:20 PM - Junxiao Shi

- % Done changed from 10 to 20

I notice many test cases in Fw/TestForwarder test suite is still using the old LimitedIo and those usages are unnecessary.

I'm converting them to UnitTestTimeFixture in http://gerrit.named-data.net/2875.

#7 - 06/05/2016 05:53 PM - Junxiao Shi

- % Done changed from 20 to 60

http://gerrit.named-data.net/2876 patchset1 has the test case.

#8 - 06/05/2016 08:36 PM - Junxiao Shi

- Status changed from In Progress to Code review

- % Done changed from 60 to 100

http://gerrit.named-data.net/2876 patchset2 has the patch.

#9 - 06/06/2016 04:01 PM - Junxiao Shi

@Peter, can you test whether http://gerrit.named-data.net/#/c/2876/2 solves the problem?

#10 - 06/19/2016 10:00 PM - Junxiao Shi

- Status changed from Code review to Resolved

Changes are merged. Peter, can you test with master branch (or commit:891f47bcfad45885408d1956657be5612620a12d)?

#11 - 06/21/2016 05:04 PM - Junxiao Shi

- Status changed from Resolved to Closed

Peter confirms the problem wasn't reproduced on the latest commit from the master branch for 1-to-10 scenario.

#12 - 06/26/2016 05:08 PM - Junxiao Shi

- File capture.pcap.zip.aa added

- File capture.pcap.zip.ab added

- File capture.pcap.zip.ac added

- File capture.pcap.zip.ad added

- File capture.pcap.zip.ae added

- File capture.pcap.zip.af added

Files

ndndump-duplicate-nonce.txt	614 KB	06/03/2016	Vince Lehman
capture.pcap.zip.aa	4 MB	06/27/2016	Junxiao Shi
capture.pcap.zip.ab	4 MB	06/27/2016	Junxiao Shi
capture.pcap.zip.ac	4 MB	06/27/2016	Junxiao Shi

capture.pcap.zip.ad	4 MB	06/27/2016	Junxiao Shi
capture.pcap.zip.ae	4 MB	06/27/2016	Junxiao Shi
capture.pcap.zip.af	3.64 MB	06/27/2016	Junxiao Shi