NFD - Bug #1511

EthernetFace cannot receive on Ubuntu 14.04

04/20/2014 09:23 PM - Junxiao Shi

Status:	Closed	**Start date:**	04/20/2014
Priority:	Normal	**Due date:**	
Assignee:	Davide Pesavento	**% Done:**	100%
Category:	Faces	**Estimated time:**	0.00 hour
Target version:	v0.2		

Description

Ubuntu 14.04 Server 64-bit

Topology: A-B

Steps to reproduce:

1. start NFD and NRD on A and B
2. on A, start ndnpingserver /A
3. on B, run nfd-status to observe FaceId of the Ethernet multicast face connected on A-B link
4. on B, invoke nfdc add-nexthop / <etherFaceId>
5. on B, start ndnping /A

Expected: ndnping displays "Content from /A"

Actual: ndnping displays "Timeout from /A", host A's NFD complains "1398054254.409812 WARNING: [EthernetFace] [id:7,endpoint:eth2] pcap_next_ex() timed out"

Note: on both A and B, sudo tcpdump -p -i ifname shows Ethernet frames from B to multicast group

Related issues:

Related to NFD - Feature #3131: EthernetTransport: re-enable TPACKET_V3 for c...	**Rejected**

History

#1 - 04/21/2014 06:17 AM - Davide Pesavento

Junxiao Shi wrote:

> "1398054254.409812 WARNING: [EthernetFace] [id:7,endpoint:eth2] pcap_next_ex() timed out"
> Note: on both A and B, sudo tcpdump -p -i eth1 shows Ethernet frames between two hosts

So you're saying that the EthernetFace endpoint is eth2 but tcpdump sees packets on eth1?

#2 - 04/21/2014 07:19 AM - Junxiao Shi

- Description updated

tcpdump sees Ethernet frames on A:eth2 and B:eth1. These two are on same subnet.

#3 - 04/23/2014 10:08 AM - Davide Pesavento

I think I figured out what's going on here. It's a case of bad interaction between boost::asio, libpcap, and the kernel (though IMHO most of the fault lies in the kernel).

The problem is triggered by the usage of the new TPACKET_V3 kernel code for AF_PACKET sockets and its associated semantics. TPACKET_V3 was introduced in Linux 3.2 but not used until libpcap 1.5.0 (this explains why prior versions of ubuntu are unaffected).

A bit of background... In TPACKET_V3, the RX ring buffer consists of multiple "slots", each of which is marked to belong either to the kernel or userland. As packets arrive, the kernel starts filling the first non-full slot marked for kernel usage, until there's no room left in the slot. At that point, the slot is handed to userland, and the kernel starts filling the next available slot in the buffer. If there are no buffer slots available (i.e. all slots are marked as belonging to userland), incoming packets are dropped. In addition to handing over slots to userland when they're full, the kernel also marks a slot for userland when an internal timeout expires, even if the current slot is empty. The default timeout is 8ms, but usually it's dynamically adjusted depending on line speed. However, it appears that the user process waiting on the packet socket fd is woken up *only* when a packet is put in the buffer *or* when a packet is dropped due to the buffer being full.

Therefore what happens here is that boost::asio event loop enters a wait (epoll_wait on linux, according to strace) on the EthernetFace's fd (dup'ed from libpcap's fd), triggered by an earlier async_read_some call. There are no incoming packets yet, so the kernel's TPACKET_V3 code repeatedly times out *without filling any buffer slots*, and eventually ends up marking all slots as belonging to userland, thus *running out of buffer space*. Under this condition, arriving packets must be dropped, and userland notified. So only at this point, when the ping interest arrives and is dropped by the kernel, nfd wakes up and EthernetFace::handleRead is invoked. The handler calls pcap_next_ex which however finds that the buffer slots are empty and returns 0, causing the warning print. Finally nfd goes back to sleep waiting on the socket, but is bound to fail again when the next packet arrives if the incoming rate is sufficiently low.

I'm open to suggestions for an efficient solution to this problem, in a way that avoids polling but which also doesn't add noticeable latency to the packet receive path.

#4 - 04/23/2014 11:08 AM - Alex Afanasyev

How does pcap itself (tcpdump) solves this problem? Do they use polling?

#5 - 04/23/2014 02:34 PM - Davide Pesavento

libpcap itself worked around the issue essentially with polling, i.e. by default poll() is called with a timeout of 1ms and you cannot "wait forever" with TPACKET_V3 (see https://github.com/the-tcpdump-group/libpcap/commit/ee4085152260466ea845d9e9109a251a39ded93b). This workaround doesn't work for us because we're not using libpcap's routines for the event loop.

Other network monitoring or packet capturing apps such as tcpdump, wireshark and dumpcap have a radically different purpose and requirements, and I wouldn't be surprised if they raised the timeout to a few tens or hundreds of milliseconds in order to be more efficient. Maybe they already did that before the behavior changed. However introducing such a long delay at the beginning of the packet processing path is IMO unacceptable for an application like nfd.

#6 - 04/24/2014 10:12 AM - Alex Afanasyev

Stupid question. Is it possible to use an old libpcap on this ubuntu?

#7 - 04/25/2014 06:21 AM - Davide Pesavento

I suppose so. You'll have to download an older tarball (say 1.4.0) and build it on 14.04... In that case pcap will use TPACKET_V2.

#8 - 04/29/2014 02:32 PM - Junxiao Shi

- Target version changed from v0.1 to v0.2

20140425 conference call agrees to defer this bug to next version.

RELEASE NOTES mentions this limitation.

#9 - 05/05/2014 07:21 AM - Davide Pesavento

Davide Pesavento wrote:

> Other network monitoring or packet capturing apps such as tcpdump, wireshark and dumpcap have a radically different purpose and requirements, and I wouldn't be surprised if they raised the timeout to a few tens or hundreds of milliseconds in order to be more efficient.

For the record, I checked tcpdump, it uses a timeout of 1 second.

#10 - 05/05/2014 10:31 AM - Davide Pesavento

- Status changed from New to Code review

- % Done changed from 0 to 100

I submitted http://gerrit.named-data.net/#/c/800/ which in practice forces libpcap to use the old TPACKET_V2 memory-mapped interface on Linux, thus solving this issue. Unfortunately pcap_set_immediate_mode() is new in libpcap 1.5.0 and is almost undocumented (there's a man page for the function saying that it enables "immediate mode" but it doesn't explain what "immediate mode" actually means), so I had to look at the sources to figure out what really happens under the hood.

#11 - 05/06/2014 10:21 PM - Anonymous

- Status changed from Code review to Closed

Applied in changeset commit:nfd|c6fcd5ea2d5b9cefd18f4b6ec9543c1db3597cf8.

#12 - 01/28/2015 05:50 PM - Davide Pesavento

For the record, the kernel behavior with TPACKET_V3 has recently been fixed: http://www.spinics.net/lists/netdev/msg309291.html

#13 - 08/25/2015 08:57 AM - Davide Pesavento

- Related to Feature #3131: EthernetTransport: re-enable TPACKET_V3 for capture if kernel and libpcap are recent enough added