Testing Tracker

This is a page tracks issues that we run into while testing the Multi-Rail Feature

Problem List

Description	Priority	Reporter	Notes
with 17 interfaces trying to discover on the any of the interface the first time returns an error "no route to host". Ok was able to reproduce. If I follow the steps in UT-DD-EN-0005 exactly, then the first time I try to discover any of the nids it fails steps to reproduce `P1 && P2` `  756  lnetctl lnet configure` `  757  lnetctl net add --net tcp --if eth0,eth1,eth2,eth3,eth4,eth5,eth6,eth7,eth8,eth9,eth10,eth11,eth12,eth13,eth14,eth15,eth16` `P1:` `lnetctl ping --nid 192.168.122.42@tcp` `P2:` `lnetctl ping --nid 192.168.122.23@tcp` `P1:` `[root@MRtest01 Lustre]# lnetctl discover --nid 192.168.122.46@tcp` `manage:` `    - discover:` `          errno: -1` `          descr: failed to discover 192.168.122.46@tcp: No route to host` `Second time it works`	critical	Amir	This appears to be an issue with how the State Machine works. I can reproduce it consistently by first getting the state of the peer to peer state: 1032: 10000001000 LNET_PEER_UNDISCOVERED LNET_PEER_PING_FAILED then after that do another discover, which will cause this problem to occur, basically, the FSM says that ping has previously failed and then says that PING is required and stops there. The way the code works is a little bit odd in this scenario:

```
lnet_discover()
 -> lnet_discover_peer_locked()
   -> clear discovery error
   ->
lnet_peer_queue_for_discovery()
     -> peer gets queued on the
ln_dc_request

lnet_peer_discovery() thread
  -> wakes up
  -> LNET_PEER_NIDS_UPTODATE is
not set
    -> lnet_peer_send_ping()
      -> ping fails
      -> LNetMDUnlink()
        ->
lnet_discovery_event_handler()
          ->
lnet_discovery_event_unlink()
          -> rc ==
LNET_REDISCOVER_PEER
             -> put back on the
ln_dc_request
    -> rc from
lnet_peer_send_ping() is some
error (ex -110)
    -> lnet_peer_discovery_error()
       -> remove
LNET_PEER_DISCOVERING state
    -> if (!(lp->lp_state &
LNET_PEER_DISCOVERING))
       ->
lnet_peer_discovery_complete()
           -> list_del_init(&lp-
>lp_dc_list);

It appears that the event handler
which can run in the context of
another thread, can put back the
peer on the ln_dc_request, while
the discovery thread might want
to remove the peer from all
queues.

Is there a scenario where that
could happen? EX: a ping response
is being processed. And other
ping is sent but that fails
immediately,
peer is removed from the queues,
but then the even handler adds
it back on the queue?
```

Also in the above scenario the peer is left in a LNET_PEER_PING_FAILED | LNET_PEER_UNDISCOVERED state:
or LNET_PEER_UNDISCOVERED | LNET_PEER_PING_REQUIRED

but it's not on the ln_dc_request queue?

Do we want these states set for a peer that's not on the request or working discovery queue?

			Olaf: The intent is that `LNET_PEER_PING_REQUIRED` and `LNET_PEER_PING_FAILED` are used to guide the peer through the state machine, and for that to happen the peer must be queued. If the discovery thread has reason to dequeue a peer, then it should also clear these states at that point. I think clearing `LNET_PEER_PING_REQUIRED` would also include making sure that `LNET_PEER_NIDS_UPTODATE` is cleared, to ensure that discovery will happen next time traffic goes to the peer. Amir: This is still an issue.
With 17 interface discovered "show peer" hangs When the rc from the kernel is not 0. The structure is not copied out of the kernel to user space. The code depends on that in order to pass the new size if the data to be copied out is too big for the buffer passed in by the user. Since that doesn't happen when rc == -E2BIG, user space code gets into an infinite loop sending IOCTLs to the kernel `from libcfs_ioctl()` `145 if (err == 0) {` `146 if (copy_to_user (uparam, hdr, hdr->ioc_len))` `147 err = -EFAULT;` `148 }` `149 break;` `The buffer is only copied to user space if the ioctl handler returns 0.` `Not really sure if it's safe to change that.`		Amir	This has been fixed
"lnetctl discover" command hangs with discovery off. This happened once, so an intermittent issue. Will try to reproduce.	Major	Sonia	not reproducible
"lnetctl discover" discovers the peer even with discovery off	Major	Sonia	discovery can not be turned off now
"lnetctl discover --force" expects a parameter (no parameter should be needed with --force).	Major	Doug	This has been fixed
Doug: I configured a Parallels VM with 16 interfaces (won't let me do 17 as 16 is a limit). When I "lctl network configure" with no YAML or module parameters, I get this error from ksocklnd: "Mar 7 14:01:16 centos-7 kernel: LNet: 5111:0:(socklnd.c:2652: ksocknal_enumerate_interfaces()) Ignoring interface virbr0 (too many interfaces)".	Minor	Doug	When no interfaces are configured, the ksocknal code enumerates all the interfaces and adds them under the same net. That's how socknal tcp bonding works, but then only uses the first interface. Not sure why they do that. The max number of interfaces the the socknal tcp bonding allows is 16. So you probably have 17 interfaces. Look at ksocknal_enumerate_interfaces(). Anyway, this is outside the scope of DD. We can fix it as a separate patch to master.
Doug: When discovering a node with 16 interfaces via "lnetctl discover --nid", it works, but I am seeing this log: `Mar 7 15:48:04 centos-7 kernel: LNetError: 24769:0:(peer.c:1726:lnet_peer_push_event()) Push Put from unknown 0@<0:0> (source 0@<0:0>)`	Minor	Doug	Olaf: I've seen the message before. Not sure how best to get rid of it, but it would be safe to just not emit it for the loopback case.
Doug: Tried the command "lnetctl set discovery" (no args) and got a core dump.	Major	Doug	Olaf: As I recall, the problem is `parse_long()` is passed a NULL pointer for its `number` parameter in this case, and it doesn't check for that. Easiest to fix in `parse_long()` rather than fix all callers. This bug affects all `lnetctl set` commands. Discovery is no longer configurable.
Doug: How do I query the discovery setting? There is no "lnetctl show discovery" (should there be?). I tried "lnetctl set discovery" with no parameters and that core dumped (see previous bullet). From a usability perspective, there is no obvious way to get this information.	Minor	Doug	Doug: was told it is "lnetctl global show". Don't like that (see Usability section) but see this problem as solved. Discovery is no longer configurable

Doug: When I enter "lnetctl set" to see what options I can set, I get this: `[root@centos-7 ~]# lnetctl set` `set {tiny_buffers | small_buffers | large_buffers |` `routing}` It does not mention "discovery" at all.	Major	Doug	Olaf: So the text in `list[]` for the set subcommand needs to be updated. Note that `max_interfaces` nee ds to be added there as well. Discovery is no longer configurable
When you do "lnetctl export" global section doesn't show discovery status	Minor	Amir	Discovery is no longer configurable
Doug: Test: **UT-DD-EN-0002**. The first call to discover P2 fails with this: `[root@centos-7 ~]# lnetctl discover --nid 10.211.55.62` `@tcp` `manage:` `- discover:` `errno: -1` `descr: failed to discover 10.211.55.62@tcp:` `No route to host` The second attempt works as expected. I repeated the test twice and got the same result each time.	Critical	Doug	This is a duplicate of the first entry in this table. Please look above for details.
Doug: Test: **UT-DD-EN-0003**. Same behaviour as above.	Critical	Doug	Duplicate
Amir: There should be a way to turn off Multi-Rail. From the code the `LNET_PING_FEAT_MULTI_RAIL` is not unsettable.	Critical	Amir	We are no longer making multi-rail configurable
Doug: Test: **UT-DD-DIS-0001**. Test passed and worked as described. However, I decided to run lnet-selftest to see how running some traffic after the test goes. The lnet-selftest failed (never stopped running, did not show any stats). I looked at top and can see that the "lnet-discovery" thread is using 70% CPU (it was not using any CPU prior to running lnet-selftest). I suspect it is receiving all in coming traffic so lnet-selftest is not getting anything. Additional note: I just redid the this test but ran lnet-selftest "before" trying to invoke discovery. lnet-discovery thread still takes off and lnet-selftest locks. Seems that turning off discovery causes lnet-discovery thread to misbehave.	Blocker	Doug	So this doesn't have to do specifically with the test case. It's when discovery is off and you run lnet_selftest. From the logs I collected it appears that for each selftest message being sent, the same NID gets queued on the discovery thread. But It doesn't end up doing anything. So in effect it goes into a crazy loop, trying to discover, but because it's off, it doesn't and probably doesn't update the state properly, so the next time to the same peer triggers the nid to be queued on the discovery thread again. Since the discovery thread does pretty heavy locking, it drives the system into a grind. Selftest also reacts poorly and hangs the node. I don't think we should be queuing anything on the discovery thread if discovery is off. This has been fixed
Doug: After the previous point, I tried to kill the lnet-discovery thread. It did not stop. I then tried "kill -9". It still did not stop. I then did a reboot. Node went away and could not complete the reboot because it could not unload LNet. Had to reset the node. We need a way to stop any of our worker threads when things do not go well. Hard resetting a node in the field will be unacceptable to customers.	Blocker	Doug	This has been fixed
DD doesn't handle the case where a Mulit-Rail peer is torn down and then booted with a downrev Lustre (non-mr). This needs to be handled. Both this scenario and turning off the Multi-Rail feature are going to be handled fairly similarly `node gets configured to !MR` `push with flag not set` `peer receives the push tears down the peer and` `recreates` `From now on, the peer is viewed as non-MR.` `Future message exchange will setup the preferred-NID` `in case of a down-rev` `admin will need to trigger an explicit discover` `ping response comes back with feature bit off` `rest is same as above`	Blocker	Amir	Fixed. Setting multi-rail off is a non-issue now.

DD doesn't send a push when an interface is added to an existing network	Blocker	Amir	The functionality to trigger a push when the configuration is updated is missing. Olaf: Adding an interface should cause `lnet_peer_needs_push()` to return true, therefore `lnet_peer_is_uptodate()` to return false, which in turn triggers discovery *when there is traffic to the peer.* Moreover, LNet-internal traffic (traffic to portal 0) will never trigger discovery, see `lnet_msg_discovery()`. Amir: As discussed there is the scenario where triggering discovery on traffic is not sufficient, in case one of the peers changes it's primary interface, or even all of its interfaces. The node initiating traffic will not be able to access it. Fixed
LASSERT hit with the latest timeout patch `<0>LNetError: 4706:0:(peer.c:1704: lnet_peer_discovery_complete()) ASSERTION( lp->lp_state & (1 << 4) ) failed:` `<0>LNetError: 4706:0:(peer.c:1704: lnet_peer_discovery_complete()) LBUG` `<4>Pid: 4706, comm: lnet_discovery` `<4>` `<4>Call Trace:` `<4> [<ffffffffa0c8b885>] libcfs_debug_dumpstack+0x55/0x80 [libcfs]` `<4> [<ffffffffa0c8b9cf>] lbug_with_loc+0x3f/0x90 [libcfs]` `<4> [<ffffffffa0d2e297>] lnet_peer_discovery_complete+0x147/0x150 [lnet]` `<4> [<ffffffffa0d33ffd>] lnet_peer_discovery+0xe5d/0x1440 [lnet]` `<4> [<ffffffff8152a6be>] ? thread_return+0x4e/0x7d0` `<4> [<ffffffff81064bd2>] ? default_wake_function+0x12/0x20` `<4> [<ffffffff8109ebb0>] ? autoremove_wake_function+0x0/0x40` `<4> [<ffffffffa0d331a0>] ? lnet_peer_discovery+0x0/0x1440 [lnet]` `<4> [<ffffffff8109e71e>] kthread+0x9e/0xc0` `<4> [<ffffffff8100c20a>] child_rip+0xa/0x20` `<4> [<ffffffff8109e680>] ? kthread+0x0/0xc0` `<4> [<ffffffff8100c200>] ? child_rip+0x0/0x20` `<4>` `<0>Kernel panic - not syncing: LBUG` `<4>Pid: 4706, comm: lnet_discovery Not tainted 2.6.32.504.16.2.el6_lustre #1` `<4>Call Trace:` `<4> [<ffffffff81529fbc>] ? panic+0xa7/0x16f` `<4> [<ffffffffa0c8b9e6>] ? lbug_with_loc+0x56/0x90 [libcfs]` `<4> [<ffffffffa0d2e297>] ? lnet_peer_discovery_complete+0x147/0x150 [lnet]` `<4> [<ffffffffa0d33ffd>] ? lnet_peer_discovery+0xe5d/0x1440 [lnet]` `<4> [<ffffffff8152a6be>] ? thread_return+0x4e/0x7d0` `<4> [<ffffffff81064bd2>] ? default_wake_function+0x12/0x20` `<4> [<ffffffff8109ebb0>] ? autoremove_wake_function+0x0/0x40` `<4> [<ffffffffa0d331a0>] ? lnet_peer_discovery+0x0/0x1440 [lnet]` `<4> [<ffffffff8109e71e>] ? kthread+0x9e/0xc0` `<4> [<ffffffff8100c20a>] ? child_rip+0xa/0x20` `<4> [<ffffffff8109e680>] ? kthread+0x0/0xc0` `<4> [<ffffffff8100c200>] ? child_rip+0x0/0x20` `(gdb) l* lnet_peer_discovery_complete+0x147` `0x2e2b7 is at /home/ashehata/LustreBuild/mr-dd/lnet/lnet/peer.c:1704.`	Blocker		This seems to be a bit of a race there. I can't reproduce again. Olaf: The most plausible thing I can think of is that `list_for_each_entry_safe()` or `while (! list_empty(...))` should be used here. Otherwise you can hit a use-after-free here, and failing this assert would be possible symptom. Fixed

```
1699    {
1700            CDEBUG(D_NET, "Dequeue peer %s\n",
1701                   libcfs_nid2str(lp-
>lp_primary_nid));
1702
1703            spin_lock(&lp->lp_lock);
1704            LASSERT(lp->lp_state &
LNET_PEER_QUEUED);
1705            lp->lp_state &= ~LNET_PEER_QUEUED;
1706            spin_unlock(&lp->lp_lock);
1707            list_del_init(&lp->lp_dc_list);
1708            wake_up_all(&lp->lp_dc_waitq);
(gdb) l *lnet_peer_discovery+0xe5d
0x3401d is in lnet_peer_discovery (/home/ashehata
/LustreBuild/mr-dd/lnet/lnet/peer.c:3013).
3008            lnet_net_lock(LNET_LOCK_EX);
3009            list_for_each_entry(lp, &the_lnet.
ln_dc_request, lp_dc_list) {
3010                    lnet_peer_discovery_error(lp,
-ESHUTDOWN);
3011                    lnet_peer_discovery_complete
(lp);
3012            }
3013            list_for_each_entry(lp, &the_lnet.
ln_dc_working, lp_dc_list) {
3014                    lnet_peer_discovery_error(lp,
-ESHUTDOWN);
3015                    lnet_peer_discovery_complete
(lp);
3016            }
3017            list_for_each_entry(lp, &the_lnet.
ln_dc_expired, lp_dc_list) {
```

To reproduce

```
Modified the code to not send a REPLY for the GET
initiate discovery from node.
timeout set 180
CTRL-C command
lnetctl lnet unconfigure
**crash
```

Doug: I have a bug which can be reproduced by requires these very specific steps to do so. Start with a node and a peer. I have two interfaces configured for the node and 3 for the peer. I leave discovery on in both the node and peer. In the peer, manually configure the node (as a peer) to have one interface, non-MR: `[root@centos-7 ~]# lnetctl peer add --prim_nid 10.211.55.58@tcp --non_mr` `[root@centos-7 ~]# lnetctl peer show` `peer:` `- primary nid: 10.211.55.58@tcp` `Multi-Rail: True` `peer ni:` `- nid: 10.211.55.58@tcp` `state: NA` Then, trigger discovery on the node:	Critical	Doug	Olaf: The peer created with `-non_mr` is shown with `Multi-Rail: True`? Amir: This the expected behavior. Even though you configure the node on the peer to only have one interface, yet the node itself has two interfaces and is multi-rail capable, so it will round robin over both interfaces. Becuase you hard coded the node to be non-mr on the peer, the peer will continue to view the node as non-mr and thus will see both interfaces as two different peers:

```
[root@centos-7 ~]# lnetctl discover --nid 10.211.55.59
@tcp
discover:
    - primary nid: 10.211.55.59@tcp
      Multi-Rail: True
      peer ni:
        - nid: 10.211.55.59@tcp
        - nid: 10.211.55.62@tcp
        - nid: 10.211.55.63@tcp
[root@centos-7 ~]# lnetctl peer show
peer:
    - primary nid: 10.211.55.59@tcp
      Multi-Rail: True
      peer ni:
        - nid: 10.211.55.59@tcp
          state: NA
        - nid: 10.211.55.62@tcp
          state: NA
        - nid: 10.211.55.63@tcp
          state: NA
```

Ok, that is fine as the node did not have anything configured via MR and was able to discovery the 3 interfaces on the peer.

I then run a write-bulk lnet-selftest from the node to the peer. That works ok. However, when I look at the peers on the peer, I see both interfaces on the node even though MR has configured only one:

```
[root@centos-7 ~]# lnetctl peer show
peer:
    - primary nid: 10.211.55.58@tcp
      Multi-Rail: True
      peer ni:
        - nid: 10.211.55.58@tcp
          state: NA
    - primary nid: 10.211.55.60@tcp
      Multi-Rail: True
      peer ni:
        - nid: 10.211.55.60@tcp
          state: NA
```

Looking at the stats for these two interfaces, I can see they are both used in the test even though they are not bound together:

```
[root@MRtest02 Lustre]# lnetctl
peer show
peer:
    - primary nid: 192.168.122.10
@tcp
      Multi-Rail: False
      peer ni:
        - nid: 192.168.122.10@tcp
          state: NA
    - primary nid: 192.168.122.11
@tcp
      Multi-Rail: True
      peer ni:
        - nid: 192.168.122.11@tcp
          state: NA
```

That explains why you see traffic on both of the node's interfaces. Furthermore, because peer is dynamically discovered as MR on node all of its interfaces are used in round robin.

When you start lnet_selftest in the other direction, where the sender is the peer with the non-mr node configured as peer, then you'll see that only the interface configured for the node is used (mostly)

```
[root@centos-7 ~]# lnetctl peer show -v
peer:
    - primary nid: 10.211.55.58@tcp
      Multi-Rail: True
      peer ni:
        - nid: 10.211.55.58@tcp
          state: NA
          max_ni_tx_credits: 8
          available_tx_credits: 8
          min_tx_credits: 1
          tx_q_num_of_buf: 0
          available_rtr_credits: 8
          min_rtr_credits: 8
          refcount: 1
          statistics:
              send_count: 15692
              recv_count: 15695
              drop_count: 0
    - primary nid: 10.211.55.60@tcp
      Multi-Rail: True
      peer ni:
        - nid: 10.211.55.60@tcp
          state: NA
          max_ni_tx_credits: 8
          available_tx_credits: 8
          min_tx_credits: 2
          tx_q_num_of_buf: 0
          available_rtr_credits: 8
          min_rtr_credits: 8
          refcount: 1
          statistics:
              send_count: 15416
              recv_count: 15418
              drop_count: 0
[root@centos-7 ~]#
```

That was not expected.

```
[root@MRtest01 ~]# lnetctl net
show -v
net:
    - net type: lo
      local NI(s):
        - nid: 0@lo
          status: up
          statistics:
              send_count: 0
              recv_count: 0
              drop_count: 0
          tunables:
              peer_timeout: 0
              peer_credits: 0

peer_buffer_credits: 0
              credits: 0
          lnd tunables:
          tcp bonding: 0
          dev cpt: 0
          CPT: "[0]"
    - net type: tcp
      local NI(s):
        - nid: 192.168.122.10@tcp
          status: up
          interfaces:
              0: eth0
          statistics:
              send_count: 21900
              recv_count: 21900
              drop_count: 0
          tunables:
              peer_timeout: 180
              peer_credits: 8

peer_buffer_credits: 0
              credits: 256
          lnd tunables:
          tcp bonding: 0
          dev cpt: -1
          CPT: "[0]"
        - nid: 192.168.122.11@tcp
          status: up
          interfaces:
              0: eth1
          statistics:
              send_count: 26
              recv_count: 26
              drop_count: 0
          tunables:
              peer_timeout: 180
              peer_credits: 8

peer_buffer_credits: 0
              credits: 256
          lnd tunables:
          tcp bonding: 0
          dev cpt: -1
          CPT: "[0]"
```

So I would consider this standard behavior.

Router Testing Router: Setup two interfaces on a router on two networks say tcp and tcp1 and set routing. Like below • lnetctl net add --net tcp --if eth0 • lnetctl net add --net tcp1 --if eth1 • lnetctl set routing 1 Node: Setup an interface on tcp1. And add route to tcp like • lnetctl route add --net tcp --gateway 10.211.55.23@tcp1 • lctl ping 10.211.55.23@tcp1 12345-0@lo 12345-10.211.55.24@tcp 12345-10.211.55.23@tcp1 • lctl ping 10.211.55.24@tcp results in crash	critical	Sonia	The problem is in this part of the code: ``` 1680 »·······»·······/* best ni is already set if src_nid was provided */ 1681 »·······»·······if (! best_ni) { 1682 »·······»·······»·······/* Get the target peer_ni */ 1683 »·······»·······» ·······peer_net = lnet_peer_get_net_locked(peer, 1684 »·······»·······»·······» ·······»·······»·······» ·······LNET_NIDNET(dst_nid)); 1685 »·······»·······» ·······LASSERT(peer_net != NULL); 1686 »·······»·······» ·······list_for_each_entry(lpni, &peer_net->lpn_peer_nis, 1687 »·······»·······»·······» ·······»······ lpni_peer_nis) { 1688 »·······»·······»·······» ·······if (lpni->lpni_pref_nnids == 0) 1689 »·······»·······»·······» ·······»·······continue; 1690 »·······»·······»·······» ·······LASSERT(lpni- >lpni_pref_nnids == 1); 1691 »·······»·······»·······» ·······best_ni = lnet_nid2ni_locked(1692 »·······»·······»·······» ·······»·······»·······lpni- >lpni_pref.nid, cpt); 1693 »·······»·······»·······» ·······break; 1694 »·······»·······»·······} 1695 »·······»·······} ``` This is going to fail for all non-local networks. Specifically here: ``` 1683 »·······»·······» ·······peer_net = lnet_peer_get_net_locked(peer, 1684 »·······»·······»·······» ·······»·······»·······» ·······LNET_NIDNET(dst_nid)); ``` The code earlier will do the following: • if dst_nid is not on a local network, then try and find a gateway. • If a gateway is found, then this becomes our next hop that we will send to, so find the peer for that gateway. • Now in the code above, we're looking for the network of the dst_nid, the remote network - tcp1 in this case - in the gateway peer, which only has the local network - tcp. This results in peer_net to be NULL, and the assert fires. Fixed

Crash on nodes with routes configured when bringing down LNet (lnetctl lnet unconfigure)

```
<0>LNetError: 3533:0:(router.c:1201:
lnet_router_checker_stop()) ASSERTION( rc == 0 )
failed:
<0>LNetError: 3533:0:(router.c:1201:
lnet_router_checker_stop()) LBUG
<4>Pid: 3533, comm: lctl
<4>
<4>Call Trace:
<4> [<ffffffffa03e1885>] libcfs_debug_dumpstack+0x55
/0x80 [libcfs]
<4> [<ffffffffa03e19cf>] lbug_with_loc+0x3f/0x90
[libcfs]
<4> [<ffffffffa0460f28>] lnet_router_checker_stop+0x98
/0x100 [lnet]
<4> [<ffffffffa04448fa>] LNetNIFini+0x6a/0x110 [lnet]
<4> [<ffffffffa04603cf>] lnet_ioctl+0x27f/0x290 [lnet]
<4> [<ffffffff8152ce66>] ? down_read+0x16/0x30
<4> [<ffffffffa03eb2e3>] libcfs_ioctl+0x113/0x4c0
[libcfs]
<4> [<ffffffffa03e7391>] libcfs_psdev_ioctl+0x51
/0x100 [libcfs]
<4> [<ffffffff811a3ed2>] vfs_ioctl+0x22/0xa0
<4> [<ffffffff811a4074>] do_vfs_ioctl+0x84/0x580
<4> [<ffffffff8119c2d6>] ? final_putname+0x26/0x50
<4> [<ffffffff811a45f1>] sys_ioctl+0x81/0xa0
<4> [<ffffffff810e5f9e>] ? __audit_syscall_exit+0x25e
/0x290
<4> [<ffffffff8100b072>] system_call_fastpath+0x16
/0x1b
<4>
```

Olaf: issue is the return value of LNetEQFree(). From the code this would either be `-ENOENT` (no such queue) or `-EBUSY` (queue not empty).

```
void
lnet_router_checker_stop (void)
{
        int rc;

        if (the_lnet.ln_rc_state
== LNET_RC_STATE_SHUTDOWN)
                return;

        LASSERT (the_lnet.
ln_rc_state ==
LNET_RC_STATE_RUNNING);
        the_lnet.ln_rc_state =
LNET_RC_STATE_STOPPING;
        /* wakeup the RC thread
if it's sleeping */
        wake_up(&the_lnet.
ln_rc_waitq);

        /* block until event
callback signals exit */
        down(&the_lnet.
ln_rc_signal);
        LASSERT(the_lnet.
ln_rc_state ==
LNET_RC_STATE_SHUTDOWN);

        rc = LNetEQFree(the_lnet.
ln_rc_eqh);
        LASSERT(rc == 0);
        return;
}
```

There was a bug in the router code, where the rcd_mdh was being over written with an invalid value. Which caused this crash.

Both fixed

Router Testing Router: Setup two interfaces on a router on two networks say tcp and tcp1 and set routing. Like below • lnetctl net add --net tcp --if eth0 • lnetctl net add --net tcp1 --if eth1 • lnetctl set routing 1 Node 1: Setup an interface on tcp1. And add route to tcp like • lnetctl net add --net tcp1 --if eth0 • lnetctl route add --net tcp --gateway 10.211.55.23@tcp1 • lnetctl net show net: - net type: lo local NI(s): - nid: 0@lo status: up - net type: tcp1 local NI(s): - nid: 10.211.55.20@tcp1 status: up interfaces: 0: eth0 Node 2 : Setup an interface on tcp. And add route to tcp1 like • lnetctl net add --net tcp --if eth0 • lnetctl route add --net tcp1 --gateway 10.211.55.24@tcp • lnetctl ping 10.211.55.20@tcp1 manage: - ping: errno: -1 descr: failed to ping 10.211.55.20@tcp1: Input/output error NOTE: Ping on same network works but for different network always gives above error.		Sonia	The router code was overwriting the rcd_mdh, so we were never sending out the ping for checking the router. Which assumed that the router is down and therefore we never used the router. Fixed.

Usability Issues

Description	Priority	Reporter	Notes
Doug: How do I query the discovery setting? There is no "lnetctl show discovery" (should there be?). I tried "lnetctl set discovery" with no parameters and that core dumped (see previous section). From a usability perspective, there is no obvious way to get this information.		Doug	
Doug: I understand that Chris Morrone pushed for us to have "lnetctl set <key> <value>". That breaks from the original paradigm to follow which was done with the Linux "ip" command. It was designed to be: "ip <object> <action> <optional params>". So, it would be more logical to have: "lnetctl discover set 0/1" than "lnetctl set discovery 0/1". Then starting discovery can be: "lnetctl discover start <nids>". Looking for discovery status can be: "lnetctl discovery show". I found myself guessing at these commands as I have given here and had no idea to look at "set".		Doug	
Doug: You set the max interfaces with "lnetctl set max_interfaces" but is it shown as "max_intf" in the global settings. Should be the same for consistency.		Doug	

Document the behavior of Dynamic Discovery, including what type of traffic triggers discovery. • highlight the use for "lnetctl discover" command.	Critical	Amir	