

# Testing Tracker

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

## Problem List

Description	Priority	Reporter	Notes
<p>with 17 interfaces trying to discover on the any of the interface the first time returns an error "no route to host".</p> <p>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</p> <p>steps to reproduce</p> <pre>P1 &amp;&amp; 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</pre>	critical	Amir	<p>This appears to be an issue with how the State Machine works.</p> <p>I can reproduce it consistently by first getting the state of the peer to</p> <pre>peer state: 1032: 10000001000 LNET_PEER_UNDISCOVERED LNET_PEER_PING_FAILED</pre> <p>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.</p> <p>The way the code works is a little bit odd in this scenario:</p>

```

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?

			<p>Olaf: The intent is that <code>LNET_PEER_PING_REQUIRED</code> and <code>LNET_PEER_PING_FAILED</code> 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 <code>LNET_PEER_PING_REQUIRED</code> would also include making sure that <code>LNET_PEER_NIDS_UPTODATE</code> is cleared, to ensure that discovery will happen next time traffic goes to the peer.</p> <p>Amir: This is still an issue.</p>
<p>With 17 interface discovered "show peer" hangs</p> <p>When the <code>rc</code> 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 <code>rc == -E2BIG</code>, user space code gets into an infinite loop sending IOCTLs to the kernel</p> <pre> from libcfs_ioctl() 145             if (err == 0) { 146                 if (copy_to_user (uparam, hdr, hdr-&gt;ioc_len)) 147                     err = - EFAULT; 148             } 149             break; </pre> <p>The buffer is only copied to user space if the ioctl handler returns 0. Not really sure if it's safe to change that.</p>		Amir	This has been fixed
"Inetctl discover" command hangs with discovery off. This happened once, so an intermittent issue. Will try to reproduce.	Major	Sonia	not reproducible
"Inetctl discover" discovers the peer even with discovery off	Major	Sonia	discovery can not be turned off now
"Inetctl 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 <code>ksocknalnd</code> : "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	<p>When no interfaces are configured, the <code>ksocknal</code> code enumerates all the interfaces and adds them under the same net. That's how <code>socknal tcp bonding</code> works, but then only uses the first interface. Not sure why they do that. The max number of interfaces the <code>socknal tcp bonding</code> allows is 16.</p> <p>So you probably have 17 interfaces. Look at <code>ksocknal_enumerate_interfaces()</code>. Anyway, this is outside the scope of DD. We can fix it as a separate patch to master.</p>
Doug: When discovering a node with 16 interfaces via "Inetctl discover --nid", it works, but I am seeing this log:	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.
<pre> Mar 7 15:48:04 centos-7 kernel: LNetError: 24769:0: (peer.c:1726:lnet_peer_push_event()) Push Put from unknown 0@&lt;0:0&gt; (source 0@&lt;0:0&gt;) </pre>			
Doug: Tried the command "Inetctl set discovery" (no args) and got a core dump.	Major	Doug	<p>Olaf: As I recall, the problem is <code>parse_long()</code> is passed a NULL pointer for its <code>number</code> parameter in this case, and it doesn't check for that. Easiest to fix in <code>parse_long()</code> rather than fix all callers. This bug affects all <code>lnetctl set</code> commands.</p> <p>Discovery is no longer configurable.</p>
Doug: How do I query the discovery setting? There is no "Inetctl show discovery" (should there be?). I tried "Inetctl 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	<p>Doug: was told it is "Inetctl global show". Don't like that (see Usability section) but see this problem as solved.</p> <p>Discovery is no longer configurable</p>

<p>Doug: When I enter "lnetctl set" to see what options I can set, I get this:</p> <pre>[root@centos-7 ~]# lnetctl set set {tiny_buffers   small_buffers   large_buffers   routing}</pre> <p>It does not mention "discovery" at all.</p>	Major	Doug	<p>Olaf: So the text in <code>list[]</code> for the set subcommand needs to be updated. Note that <code>max_interfaces</code> needs to be added there as well.</p> <p>Discovery is no longer configurable</p>
<p>When you do "lnetctl export" global section doesn't show discovery status</p>	Minor	Amir	Discovery is no longer configurable
<p>Doug: Test: <b>UT-DD-EN-0002</b>. The first call to discover P2 fails with this:</p> <pre>[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</pre> <p>The second attempt works as expected. I repeated the test twice and got the same result each time.</p>	Critical	Doug	This is a duplicate of the first entry in this table. Please look above for details.
<p>Doug: Test: <b>UT-DD-EN-0003</b>. Same behaviour as above.</p>	Critical	Doug	Duplicate
<p>Amir: There should be a way to turn off Multi-Rail. From the code the <code>LNET_PING_FEAT_MULTI_RAIL</code> is not unsettable.</p>	Critical	Amir	We are no longer making multi-rail configurable
<p>Doug: Test: <b>UT-DD-DIS-0001</b>. Test passed and worked as described. However, I decided to run <code>lnet-selftest</code> to see how running some traffic after the test goes. The <code>lnet-selftest</code> 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 <code>lnet-selftest</code>). I suspect it is receiving all in coming traffic so <code>lnet-selftest</code> is not getting anything. Additional note: I just redid the this test but ran <code>lnet-selftest</code> "before" trying to invoke discovery. <code>lnet-discovery</code> thread still takes off and <code>lnet-selftest</code> locks. Seems that turning off discovery causes <code>lnet-discovery</code> thread to misbehave.</p>	Blocker	Doug	<p>So this doesn't have to do specifically with the test case. It's when discovery is off and you run <code>lnet_selftest</code>. 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.</p> <p>I don't think we should be queuing anything on the discovery thread if discovery is off.</p> <p>This has been fixed</p>
<p>Doug: After the previous point, I tried to kill the <code>lnet-discovery</code> 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.</p>	Blocker	Doug	This has been fixed
<p>DD doesn't handle the case where a Multi-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</p> <pre>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</pre>	Blocker	Amir	Fixed. Setting multi-rail off is a non-issue now.

<p>DD doesn't send a push when an interface is added to an existing network</p>	<p>Blocker</p>	<p>Amir</p>	<p>The functionality to trigger a push when the configuration is updated is missing.</p> <p>Olaf: Adding an interface should cause <code>lnet_peer_needs_push()</code> to return true, therefore <code>lnet_peer_is_updated()</code> to return false, which in turn triggers discovery <i>when there is traffic to the peer</i>. Moreover, LNet-internal traffic (traffic to portal 0) will never trigger discovery, see <code>lnet_msg_discovery()</code>.</p> <p>Amir: As discussed there is the scenario where triggering discovery on traffic is not sufficient, in case one of the peers changes its primary interface, or even all of its interfaces. The node initiating traffic will not be able to access it.</p> <p>Fixed</p>
<p>LASSERT hit with the latest timeout patch</p> <pre> &lt;0&gt;LNetError: 4706:0:(peer.c:1704: lnet_peer_discovery_complete()) ASSERTION( lp_ &gt;lp_state &amp; (1 &lt;&lt; 4) ) failed: &lt;0&gt;LNetError: 4706:0:(peer.c:1704: lnet_peer_discovery_complete()) LBUG &lt;4&gt;Pid: 4706, comm: lnet_discovery &lt;4&gt; &lt;4&gt;Call Trace: &lt;4&gt; [<fffffffffa0c8b885&gt;] #1="" &lt;0&gt;kernel="" &lt;4&gt;="" &lt;4&gt;call="" &lt;4&gt;pid:="" (gdb)="" -="" 0x1440="" 0x150="" 0x16f="" 0x20="" 0x2e2b7="" 0x40="" 0x7d0="" 0x80="" 0x90="" 0xc0="" 2.6.32.504.16.2.el6_lustre="" 4706,="" <="" ?="" [<fffffffff8100c200&gt;]="" [<fffffffff8100c20a&gt;]="" [<fffffffff81064bd2&gt;]="" [<fffffffff8109e680&gt;]="" [<fffffffff8109e71e&gt;]="" [<fffffffff8109ebb0&gt;]="" [<fffffffff81529fbc&gt;]="" [<fffffffff8152a6be&gt;]="" [<fffffffffa0c8b9cf&gt;]="" [<fffffffffa0c8b9e6&gt;]="" [<fffffffffa0d2e297&gt;]="" [<fffffffffa0d331a0&gt;]="" [<fffffffffa0d33ffd&gt;]="" [libcfs]="" [lnet]="" ashehata="" at="" autoremove_wake_function+0x0="" child_rip+0x0="" child_rip+0xa="" comm:="" default_wake_function+0x12="" home="" is="" kthread+0x0="" kthread+0x9e="" l*="" lbug="" lbug_with_loc+0x3f="" lbug_with_loc+0x56="" libcfs_debug_dumpstack+0x55="" lnet="" lnet_discovery="" lnet_peer_discovery+0x0="" lnet_peer_discovery+0xe5d="" lnet_peer_discovery_complete+0x147="" lustrebuild="" mr-dd="" not="" panic="" panic+0xa7="" peer.c:1704.="" pre="" syncing:="" tainted="" thread_return+0x4e="" trace:=""> </fffffffffa0c8b885&gt;]></pre>	<p>Blocker</p>	<p>Amir</p>	<p>This seems to be a bit of a race there. I can't reproduce again.</p> <p>Olaf: The most plausible thing I can think of is that <code>list_for_each_entry_safe()</code> or <code>while (!list_empty(...))</code> should be used here. Otherwise you can hit a use-after-free here, and failing this assert would be possible symptom.</p> <p>Fixed</p>

```

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. Because 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

- `inetctl net add --net tcp --if eth0`
- `inetctl net add --net tcp1 --if eth1`
- `inetctl set routing 1`

Node: Setup an interface on tcp1. And add route to tcp like

- `inetctl 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> [
```

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

<p><b>Router Testing</b></p> <p>Router: Setup two interfaces on a router on two networks say tcp and tcp1 and set routing. Like below</p> <ul style="list-style-type: none"> <li>• Inetctl net add --net tcp --if eth0</li> <li>• Inetctl net add --net tcp1 --if eth1</li> <li>• Inetctl set routing 1</li> </ul> <p>Node 1: Setup an interface on tcp1. And add route to tcp like</p> <ul style="list-style-type: none"> <li>• Inetctl net add --net tcp1 --if eth0</li> <li>• Inetctl route add --net tcp --gateway 10.211.55.23@tcp1</li> <li>• Inetctl net show</li> </ul> <pre>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</pre> <p>Node 2 : Setup an interface on tcp. And add route to tcp1 like</p> <ul style="list-style-type: none"> <li>• Inetctl net add --net tcp --if eth0</li> <li>• Inetctl route add --net tcp1 --gateway 10.211.55.24@tcp</li> <li>• Inetctl ping 10.211.55.20@tcp1</li> </ul> <pre>manage: - ping:   errno: -1   descr: failed to ping 10.211.55.20@tcp1: Input/output error</pre> <p>NOTE: Ping on same network works but for different network always gives above error.</p>	<p>Sonia</p>	<p>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.</p> <p>Fixed.</p>
--	--------------	--

## Usability Issues

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

Document the behavior of Dynamic Discovery, including what type of traffic triggers discovery.

- highlight the use for "Inetctl discover" command.

Critical

Amir

--