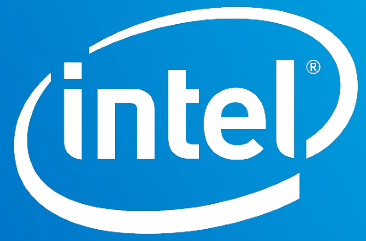


Multi-Rail LNet for Lustre*

Olaf Weber
Senior Software Engineer
SGI Storage Software

Amir Shehata
Lustre Network Engineer
Intel High Performance Data Division

Multi-Rail LNet: What and Why

Multi-Rail LNet

Multi-Rail is a long-standing wish list item known under a variety of names:

- Multi-Rail
- Interface Bonding
- Channel Bonding

The various names do imply some technical differences.

This implementation is a collaboration between SGI® and Intel®.

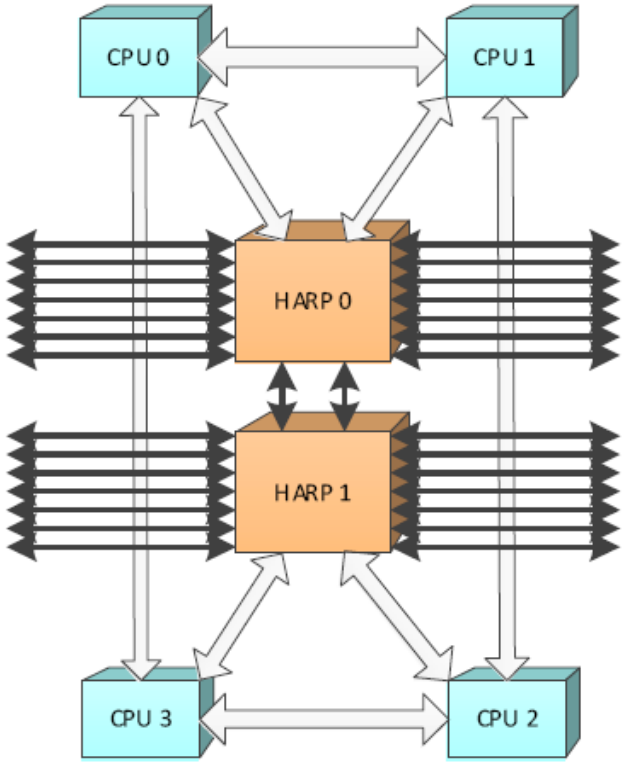
Our goal is to land multi-rail support in Lustre*.

What Is Multi-Rail?

Multi-Rail allows nodes to communicate across multiple interfaces:

- Using multiple interfaces connected to one network
- Using multiple interfaces connected to several networks
- These interfaces are used simultaneously

Why Multi-Rail: Big Clients



We want to support big Lustre nodes.

- SGI UV 300: 32-socket NUMA system
- SGI UV 3000: 256-socket NUMA system

A system with multiple TB of memory needs a lot of bandwidth.

NUMA systems benefit when memory buffers and interfaces are close in the system's topology.

Why Multi-Rail: Increasing Server Bandwidth

In big clusters, bandwidth to the server nodes becomes a bottleneck.

Adding faster interfaces implies replacing much or all of the network.

- Adding faster interfaces only to the servers does not work

Adding more interfaces to the servers increases the bandwidth.

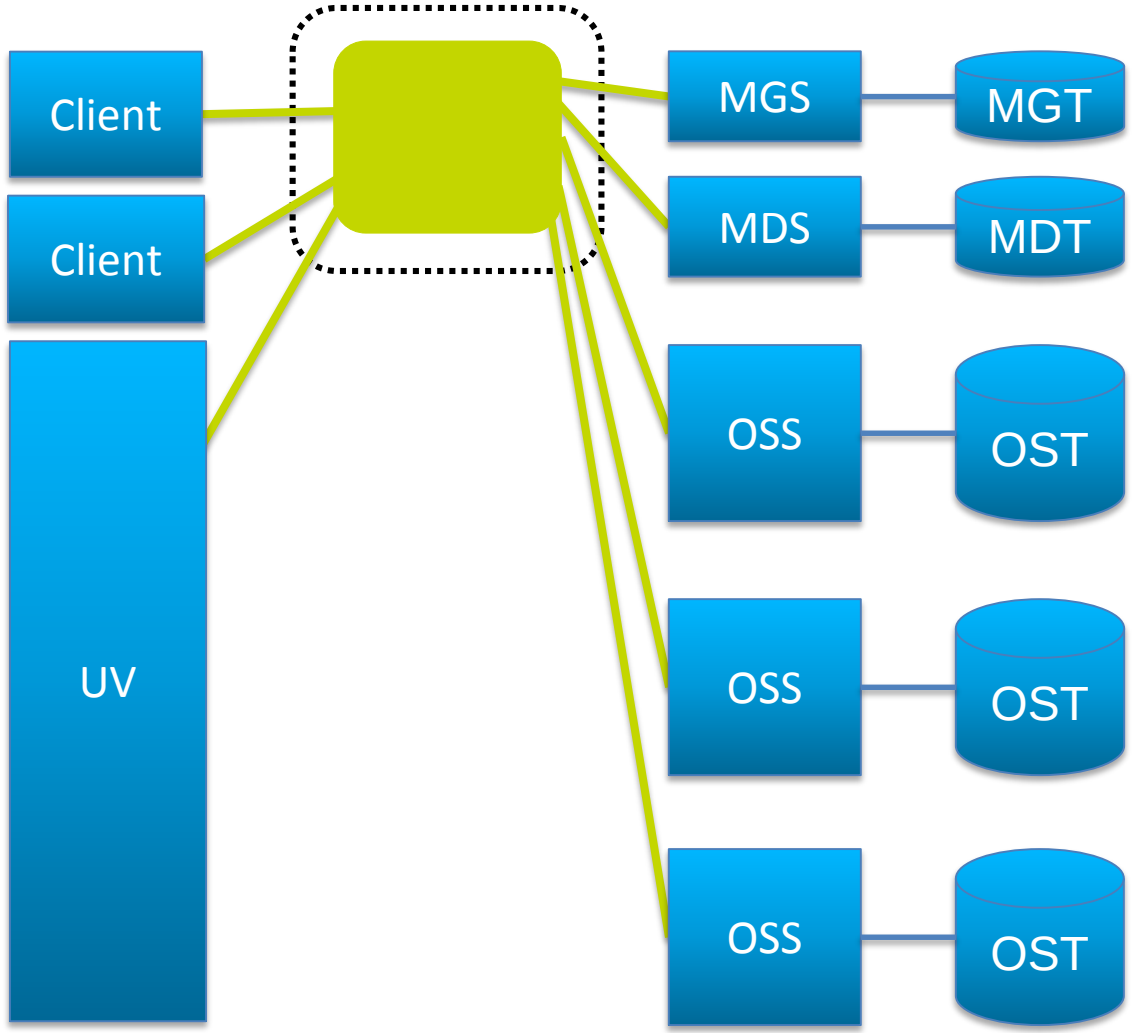
- Using those interfaces requires a redesign of the LNet networks
- Without Multi-Rail each interface connects to a separate LNet network
- Clients must be distributed across these networks

It should be simpler than this.

The Multi-Rail Project

- Add basic multi-rail capability
  - Multiplexing across interfaces, as opposed to striping across them
  - Multiple data streams are therefore needed to profit
- Extend peer discovery to simplify configuration
  - Discover peer interfaces
  - Discover peer multi-rail capability
- Configuration can be changed at runtime
  - This includes adding or removing interfaces
  - *lnetctl* is used for configuration
- Compatible with non-Multi-Rail nodes
- Add resiliency by using alternate paths in case of failures

Single Fabric With One LNet Network

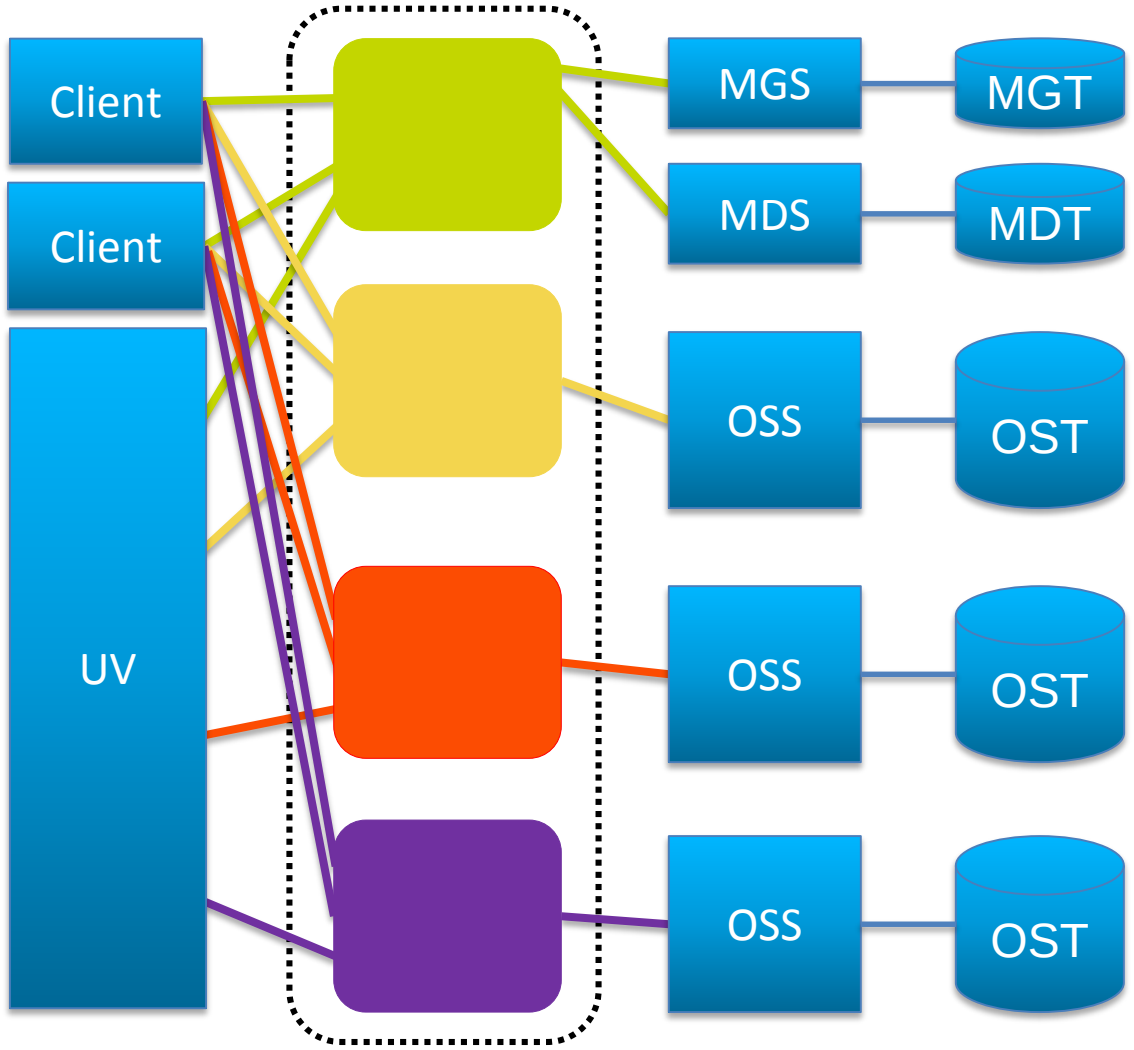


This is a small Lustre cluster with a single big client node.

All nodes are connected to a single fabric (physical network). There is one LNet network connecting the nodes.

The big UV node has a single connection to this network. It has the same network bandwidth available to it as the small clients.

Single Fabric With Multiple LNet Networks



Additional interfaces have been added to the UV to increase its bandwidth.

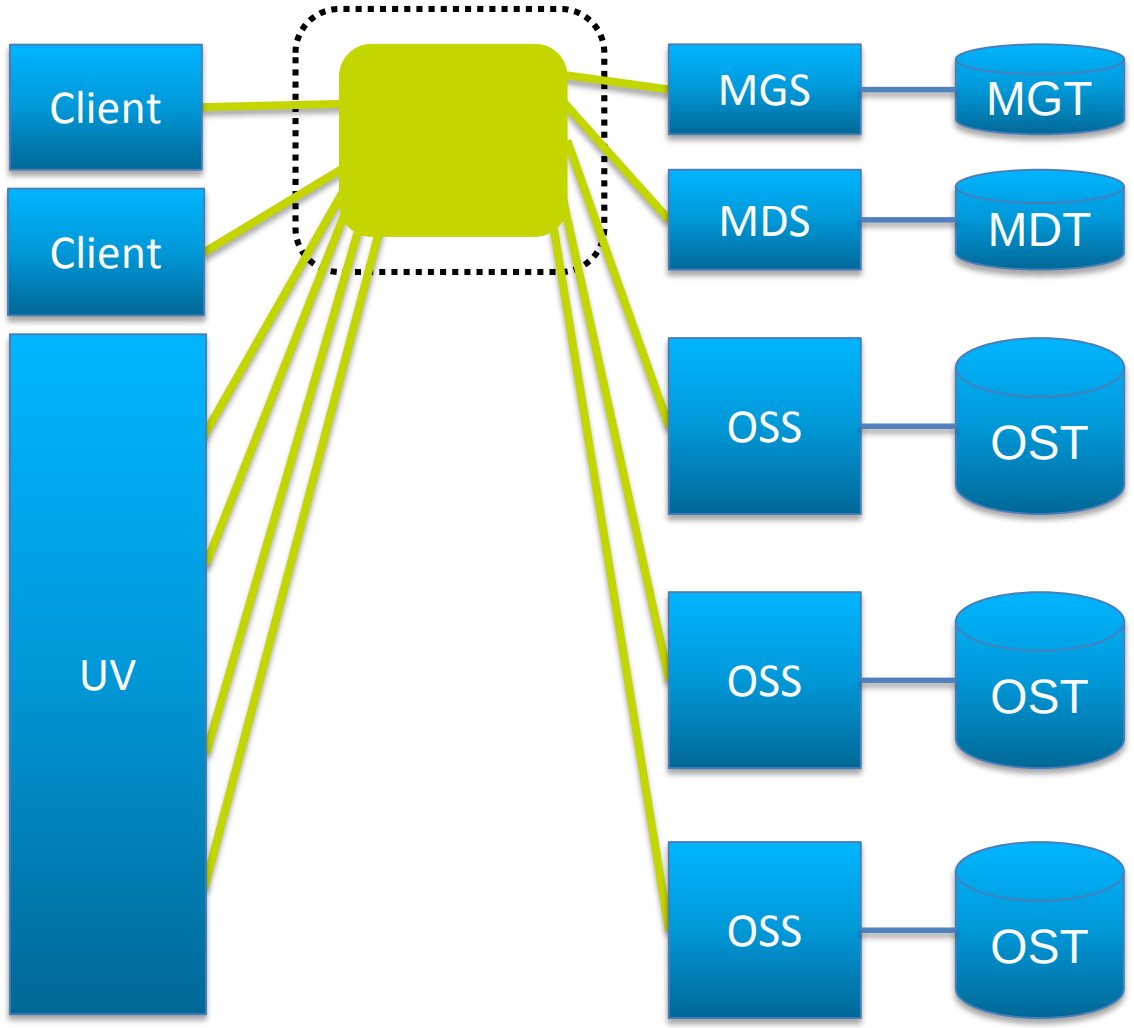
Without Multi-Rail LNet we must configure multiple LNet networks.

Each OSS lives on a separate LNet network, within the single fabric.

Each interface on the UV connects to one of these LNet networks.

On the other client nodes, aliases are used to connect a single interface to multiple LNet networks.

Single Fabric With One Multi-Rail LNet Network



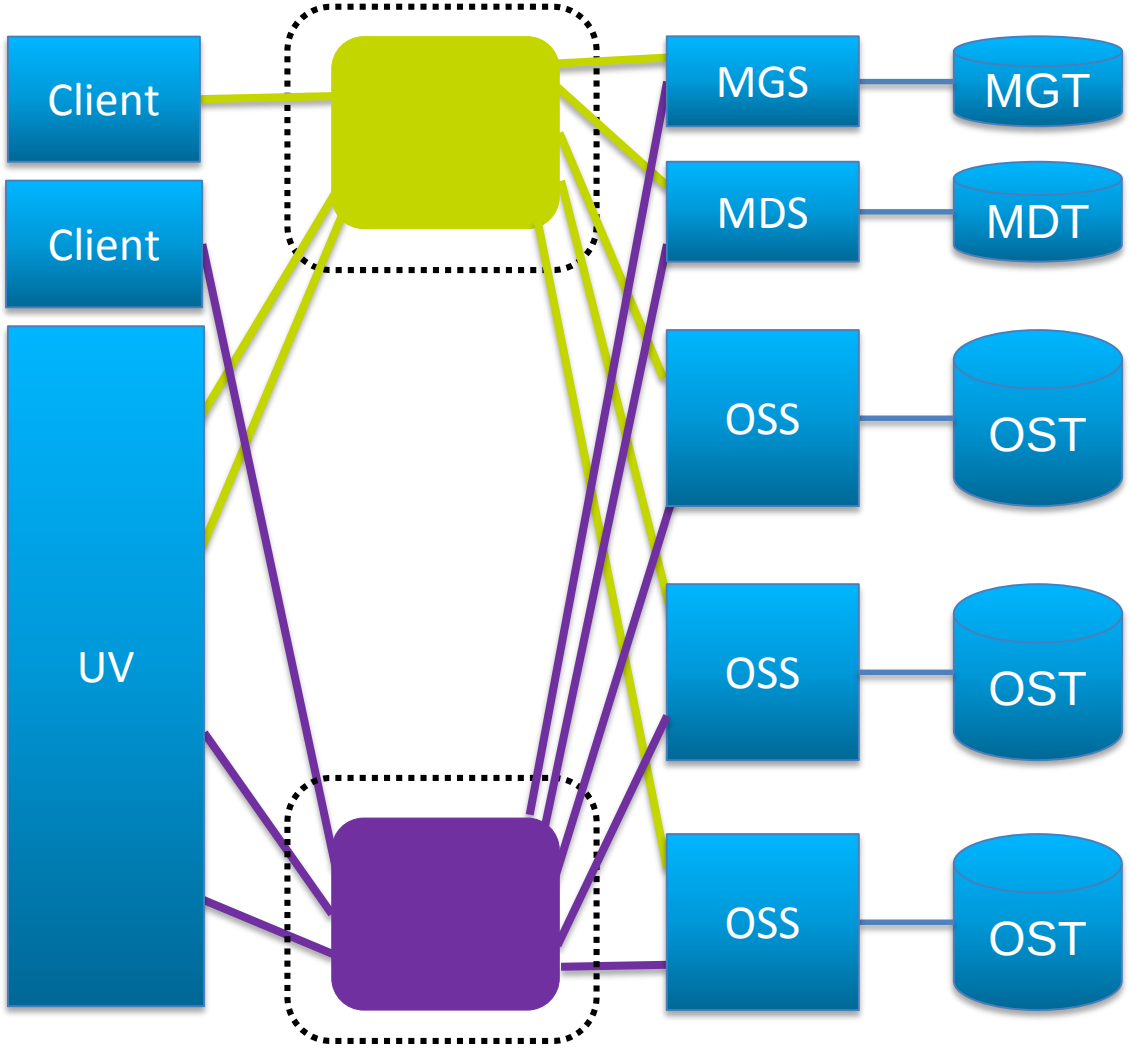
Multi-Rail LNet allows for the LNet network configuration to match the fabric.

The fabric is the same as in the previous slide.

The configuration is much simpler.

The network bandwidth to the UV node is increased to match its size.

Dual Fabric With Dual Multi-Rail LNet Networks



In this example there are two fabrics, each with an LNet network on top.

The server nodes connect to both fabrics.

The UV client connects with multiple interfaces to both fabrics.

The other client nodes connect to only one fabric.

Configuring Multi-Rail LNet

Use Cases

- Improved performance
- Improved resiliency
- Better usage of large clients
  - The Multi-Rail code is NUMA aware
- Fine grained control of traffic
- Simplify multi-network file system access

Two Types of Configuration Methods

Multi-Rail can be configured statically with *lnetctl*.

- The following *must* be configured statically
  - Local network interfaces
    - The network interfaces by which a node sends messages
  - Selection rules
    - The rules which determine the local/remote network interface pair used to communicate between a node and a peer
    - Default is weighted round-robin
- The following *may* be configured statically, but *can* be discovered dynamically
  - Peer network interfaces
    - The remote network interfaces of peer nodes to which a node sends messages

Enable dynamic peer discovery to have LNet configure peers automatically

Static Configuration – Basic Concepts

On a node:

- Configure local network interfaces
  - Example: tcp(eth0,eth1)
    - <eth0 IP>@tcp, <eth1 IP>@tcp
- Configure remote network interfaces
  - Specify the peer's Network Interface IDs (NIDs)
    - <peerX primary NID>, <peerX NID2>, …
- Configure selection rules

Dynamic Configuration – Basic Concepts

LNet can dynamically discover a peer's NIDs.

On a node:

- Local network interfaces must be configured as before
- Selection rules must be configured as before
- Peers are discovered as messages are sent and received
  - An *LNet ping* is used to get a list of the peer's NIDs
  - A feature bit indicates whether the peer supports Multi-Rail
  - The node pushes a list of its NIDs to Multi-Rail peers

Selection Rules – Basic Concepts

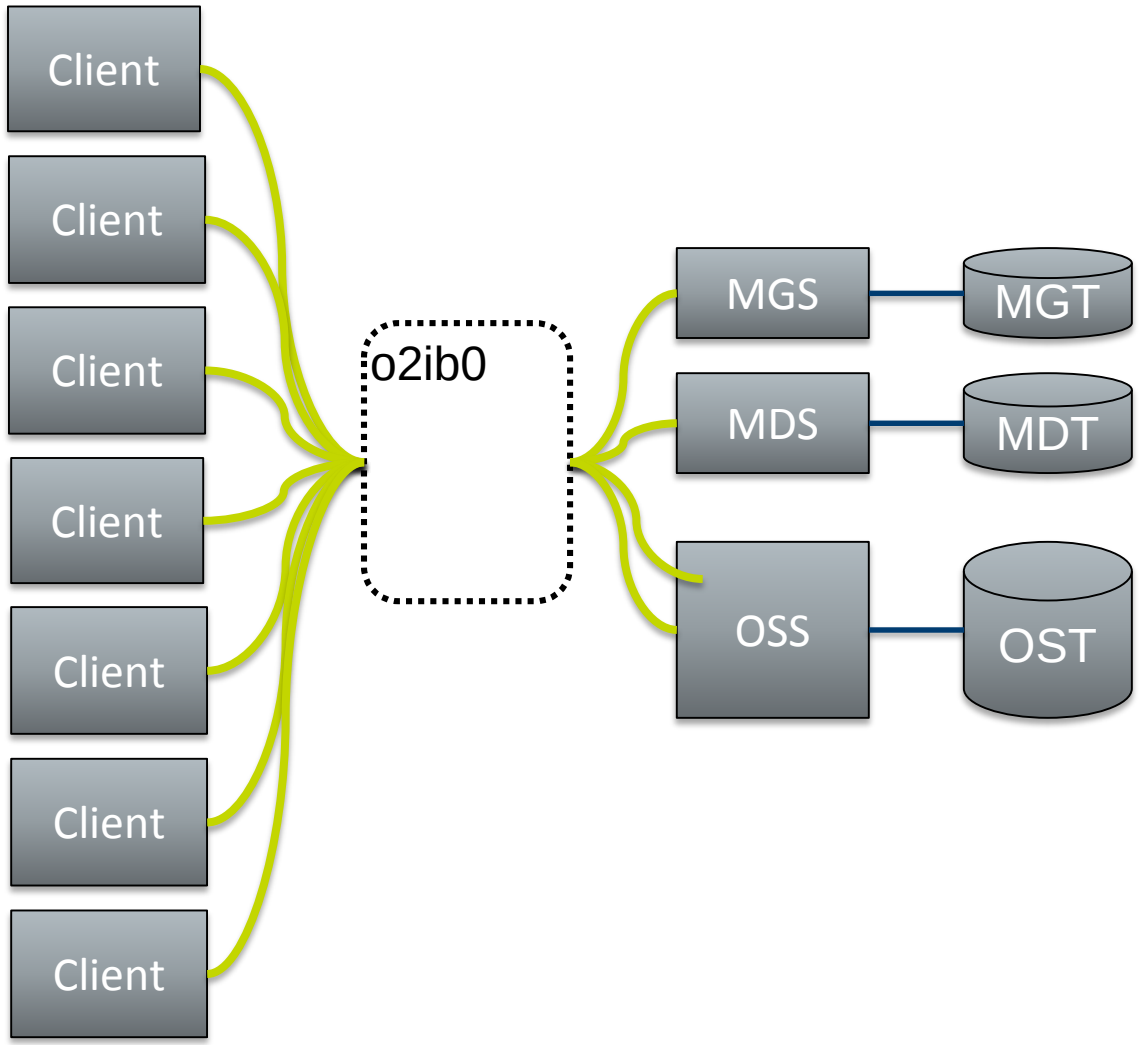
Selection rules work by adding priorities

- Higher priorities are preferred
- The rules specify patterns

The following types of rules are supported

- Network
- Local NID
- Peer NID
- Local NID / peer NID pair
  - These rules have two NID patterns, one for the local NID, one for the peer NID
  - They are used to specify preferred paths

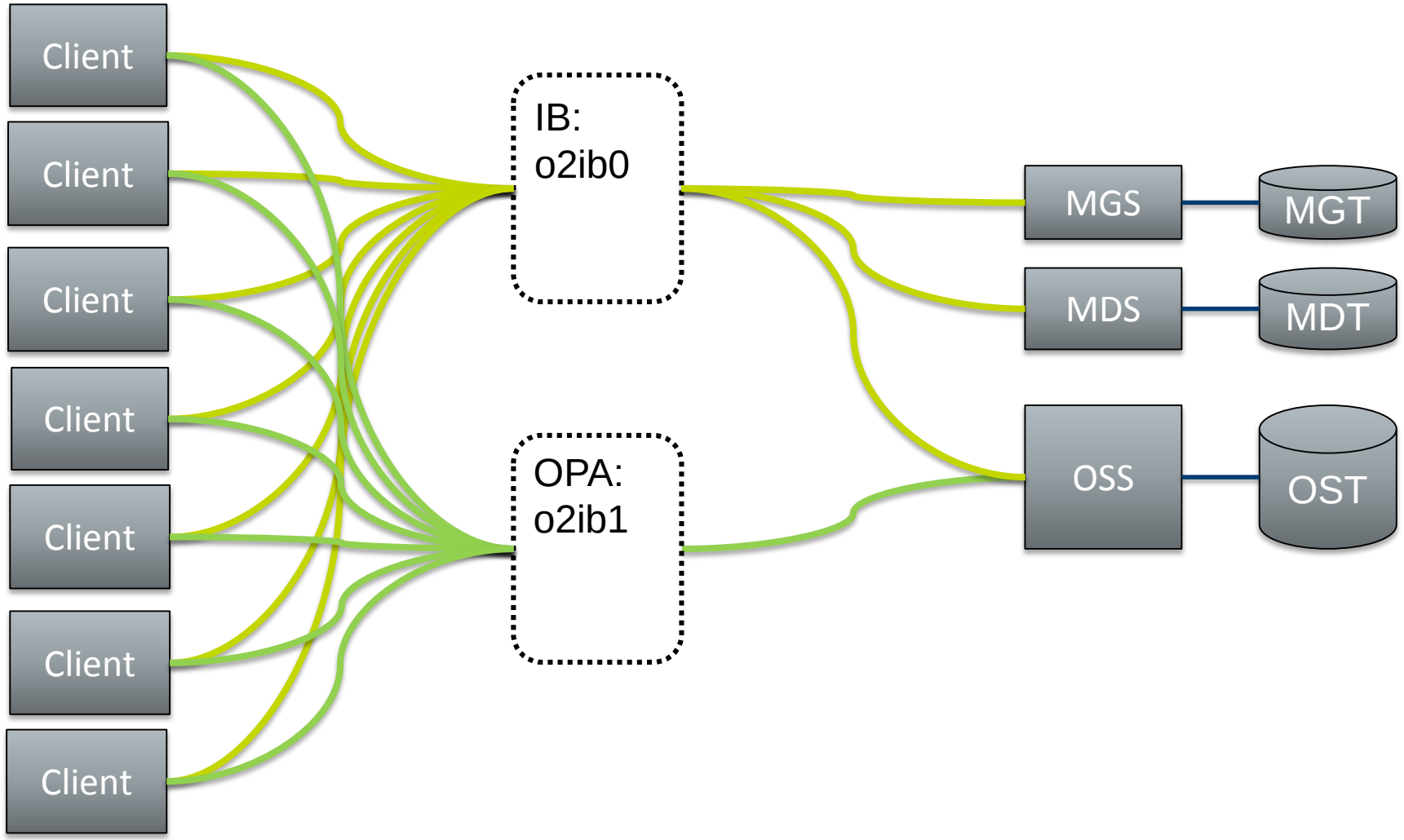
Example: Improved Performance



Configuring the OSS

```
net:
    - net type: o2ib
      local NI(s):
        - interfaces:
              0: ib0
              1: ib1
```

Example: Improved Resiliency



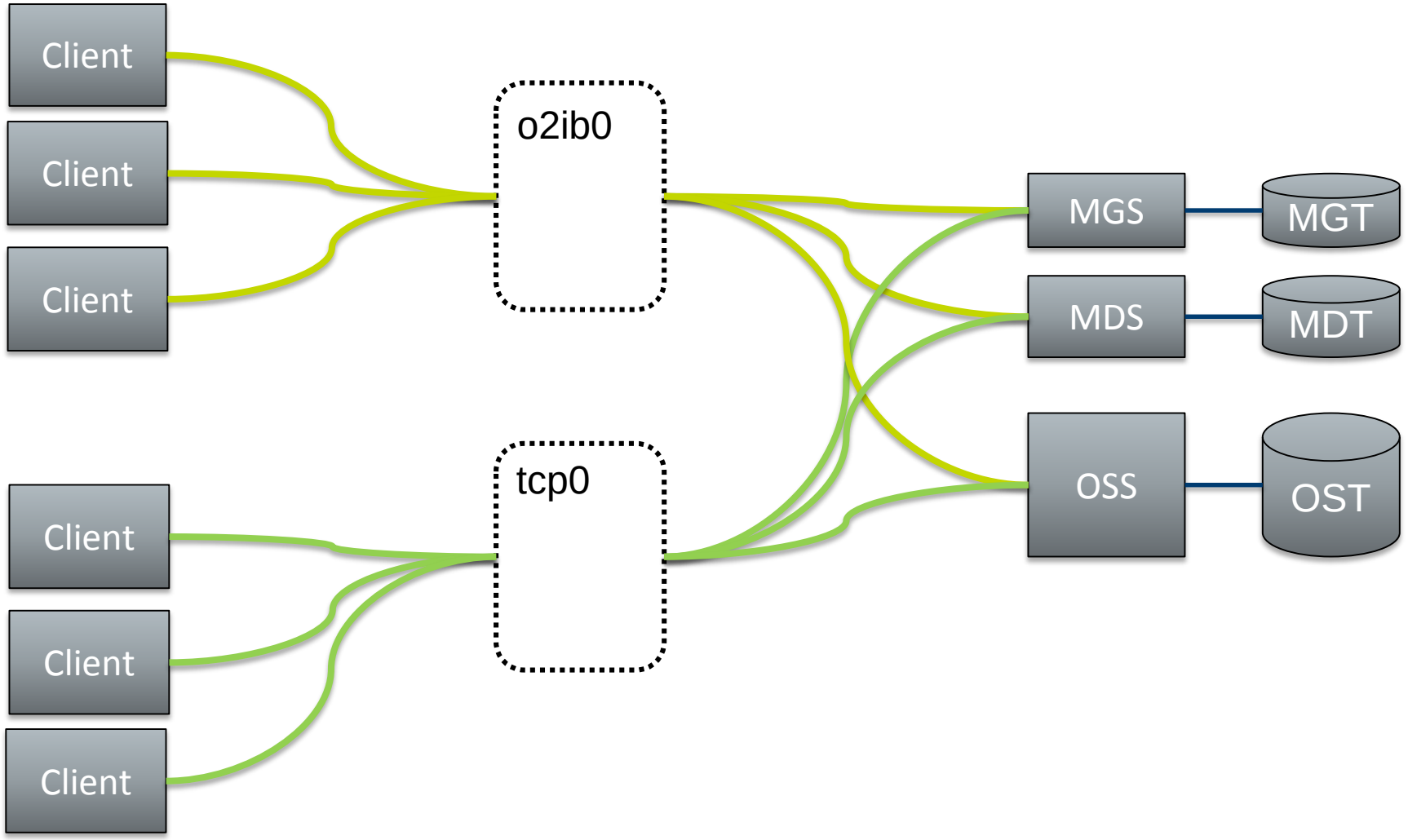
Example: Improved Resiliency

Configuring the Clients and OSS:

```
net:
    - net type: o2ib
      local NI(s):
        - interfaces:
              0: ib0
    - net type: o2ib1
      local NI(s):
        - interfaces:
              0: ib1
```

```
selection:
    - type: net
      net: o2ib
      priority: 0 # highest priority
    - type: net
      net: o2ib1
      priority: 1
```

Example: Multi Network Filesystem Access



Example: Multi Network Filesystem Access

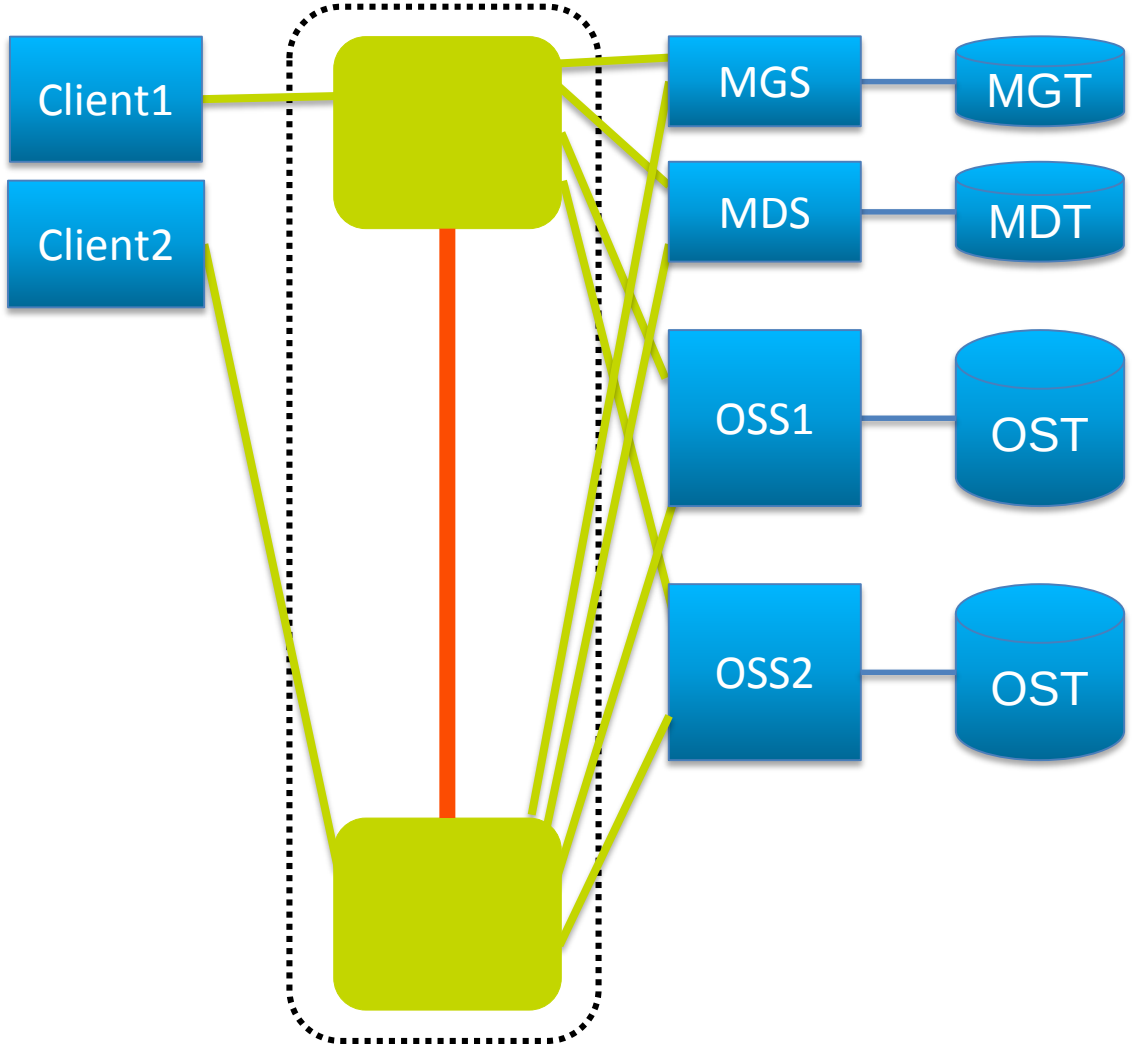
IB Clients

```
net:
    - net type: o2ib
      local NI(s):
        - interfaces:
              0: ib0
peer:
    - primary nid: <mgs-ib-ip>@o2ib0
      peer ni:
         - nid: <mgs-tcp-ip>@tcp0
selection:
    - type: nid
    - nid: *.*.*.*@o2ib*
    - priority: 0 # highest priority
```

TCP Clients

```
net:
    - net type: tcp0
      local NI(s):
        - interfaces:
              0: eth0
peer:
    - primary nid: <mgs-ib-ip>@o2ib0
      peer ni:
         - nid: <mgs-tcp-ip>@tcp0
selection:
    - type: nid
    - nid: *.*.*.*@tcp*
    - priority: 0 # highest priority
```

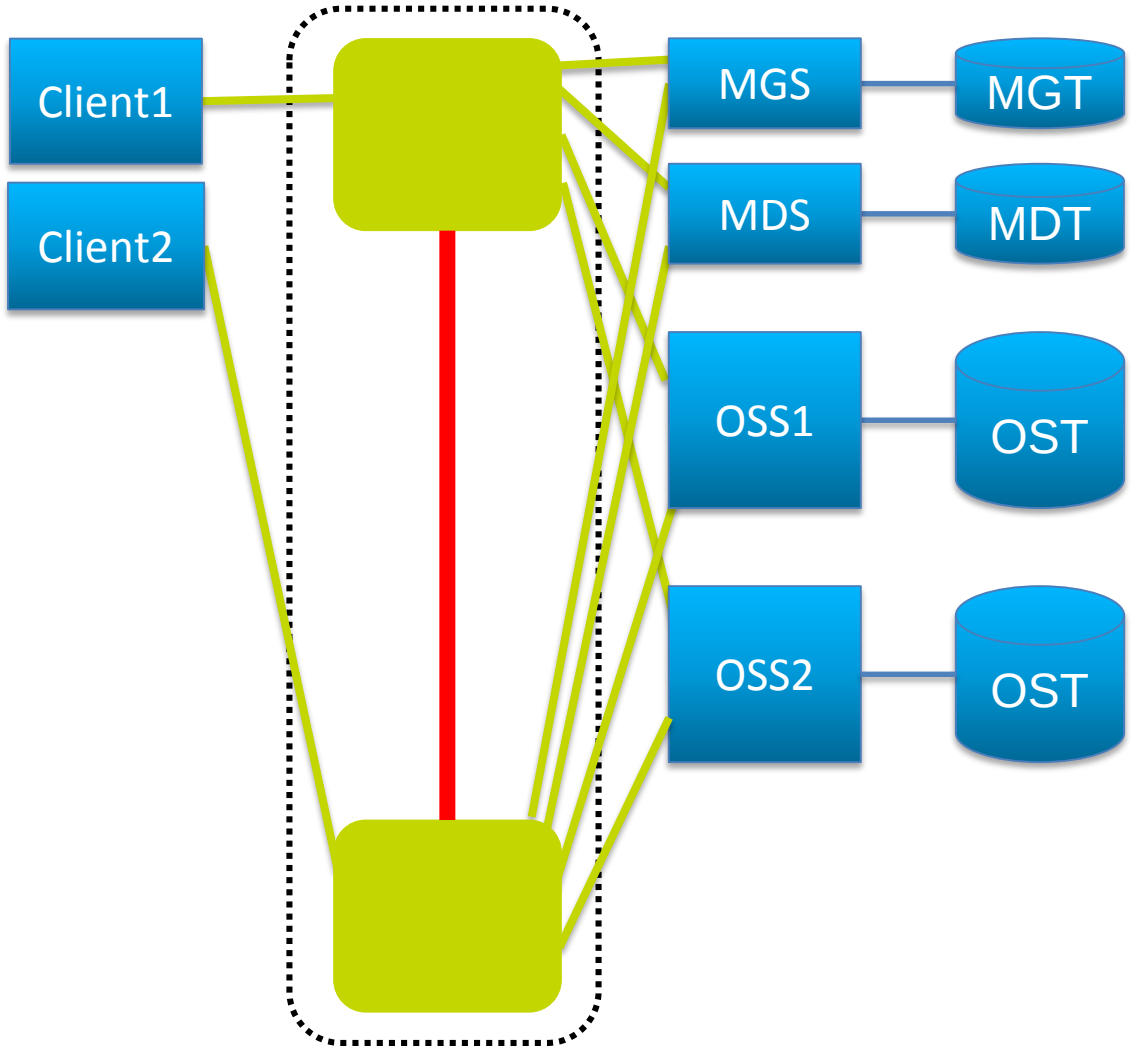
Example: Fine Grained Traffic Control



This is a single fabric with a bottleneck.

Client1: 10.10.10.2@o2ib
Client2: 10.10.10.3@o2ib
MGS-1: 10.10.10.4@o2ib
MGS-2: 10.10.10.5@o2ib
MDS-1: 10.10.10.6@o2ib
MDS-2: 10.10.10.7@o2ib
OSS1-1: 10.10.10.8@o2ib
OSS1-2: 10.10.10.9@o2ib
OSS2-1: 10.10.10.10@o2ib
OSS2-2: 10.10.10.11@o2ib

Example: Fine Grained Traffic Control



This rule makes *Client1* avoid the **red** path:

```
selection:
  - type: peer
    local: 10.10.10.2@o2ib
    remote: 10.10.10.[4-10/2]@o2ib
    priority: 0 # highest priority
```

Client1 will only use the **red** path if there is no other option.

Multi-Rail LNet Project Status

Project Status

Public project wiki page:

- http://wiki.lustre.org/Multi-Rail_LNet

Code development is done on the multi-rail branch of the Lustre master repo.

- Patches to enable static configuration are under review
- Unit testing and system testing underway
- Patches for selection rules are under development
- Patches for dynamic peer discovery are under development
- Estimated project completion time: end of this year
- Master landing date: TBD

Legal Notices and Disclaimers

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at intel.com, or from the OEM or retailer.

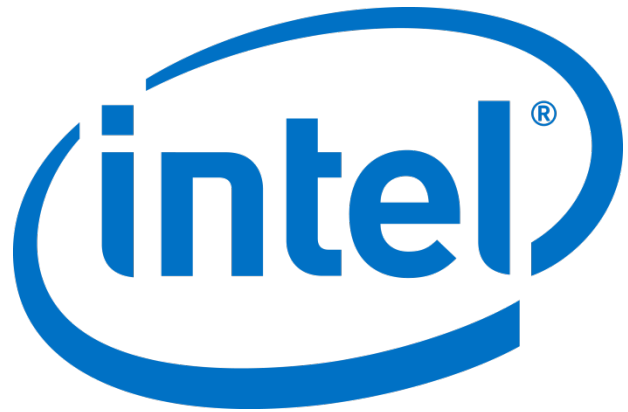
No computer system can be absolutely secure.

Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit **http://www.intel.com/performance**.

Intel disclaims all express and implied warranties, including, without limitation, the implied warranties and merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing or usage in trade.

Intel, the Intel logo and others are trademarks of Intel Corporation in the U.S. and/or other countries. *Other names and brands may be claimed as the property of others.


sgi


intel