User-defined Transport Protocols for Lustre

Eric Kinzie eric.kinzie@exelisinc.com
Linden Mercer (ARL/PSU)

Work done under contract to the Naval Research Laboratory
Introduction

• Provides a new LND (Iu2kInd)
• Additional layer of abstraction between LNet and network interfaces
• Transport protocol can be implemented outside of kernel
Motivation

• Network connectivity into theater not ideal
  • Latency (satellite links)
  • Packet loss due to congestion and physical errors
  • Limited bandwidth
• Data originates in theater and end-users of processed data also in theater
• Analysts stay clear of the action
• Some existing applications expect to retrieve data from a filesystem
• Protocols of interest implemented outside of the kernel
UDT

- UDT (UDP Data Transport) developed at UIUC’s National Center for Data Mining
- Key characteristics
  - Explicit NACK
  - Available bandwidth discovery + rate shaping
  - RTT measurement (congestion window)
  - Configurable congestion control algorithm
- [http://udt.sourceforge.net](http://udt.sourceforge.net)
Architecture

• A small part of the LND operates in the kernel
• Connection management and transport protocol implemented in daemon process
• Communication between LND and daemon process handled by select()/ioctl()
• Messages are memory mapped
Who keeps track of what?

Kernel
- Peers
- Messages

Daemon
- Peers
- Messages
- Connections
• Connections are unidirectional
• Destination IP address for connection found in NID (same as sockInd, etc.)
• Source network interface and next-hop determined by kernel routing table/policy
• Some additional state is kept when Lustre routing involved
Transmitted Messages

Diagram:
- Kernel
  - TXNOTIFY
  - mmap()
  - TXDONE
  - munmap()
- Daemon
- Network
  - send()
Received Messages

Kernel  ->  Daemon  ->  Network

RXSTART  ->  RXNOTIFY  ->  mmap()  ->  RXDONE  ->  munmap()  <-  LNet message  <-  recv()  <-  LNet header  <-  recv()
Protocol Back-ends

- Designed to allow other transport layers to be glued on
- Kernel half of LND not aware of the selected protocol
- All peers use same protocol
- API with entry points similar to a sockets interface
- Currently have UDT and RDS back-ends
Current Status

- Software working in laboratory setting
- Testing activities focused on validating correct behavior
- Have demonstrated video streaming with 200ms RTT
- Waiting on approval for public release of source code
Next Steps

- Performance tests
- Improved usability
- Feature Creep
  - Connection encryption and/or authentication
  - Load balancing
  - IPv6