Hewlett Packard
Enterprise

Lustre Timeout Hierarchy

Chris Horn, Lustre Software Engineer

April, 2025

Agenda

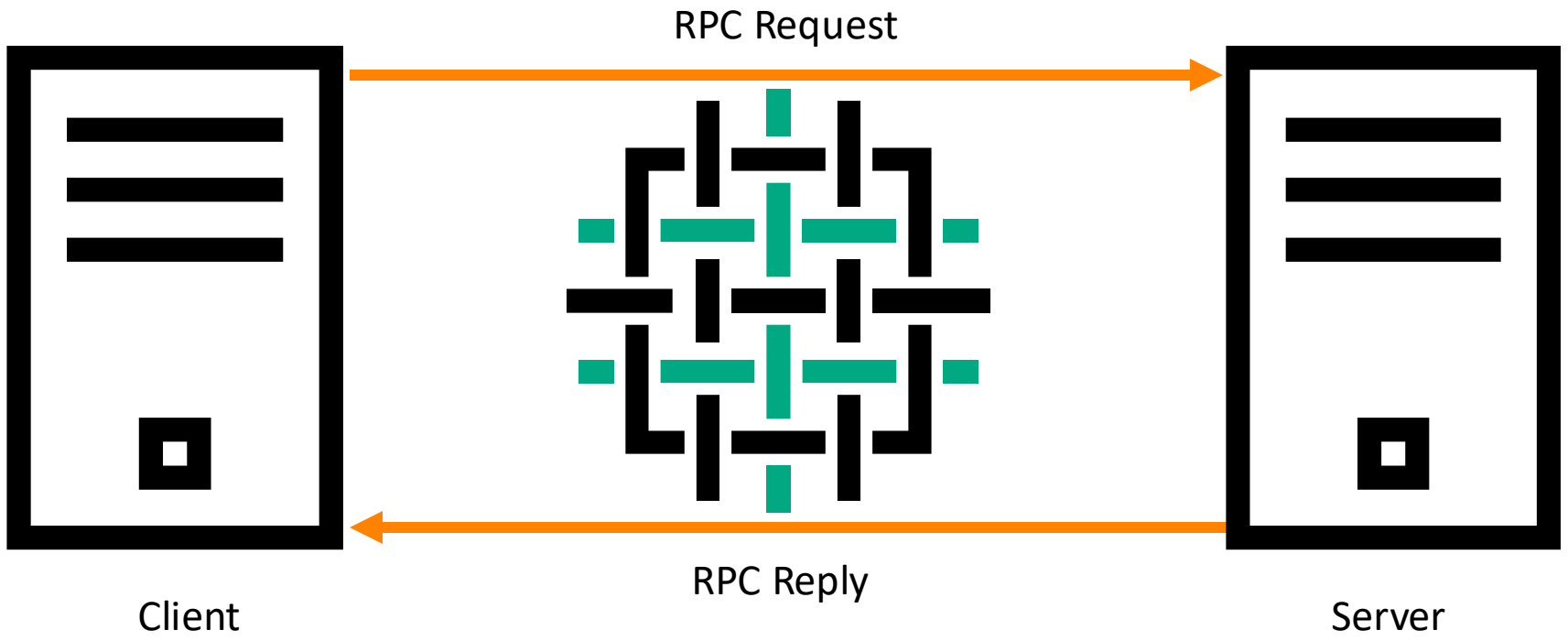
- Introduction
- Tuning Goal
- PtlRPC, LNet, LND Timeouts
- Tuning the Hierarchy
- Dynamically Configured Timeouts, and Future Work
- Q&A

Introduction

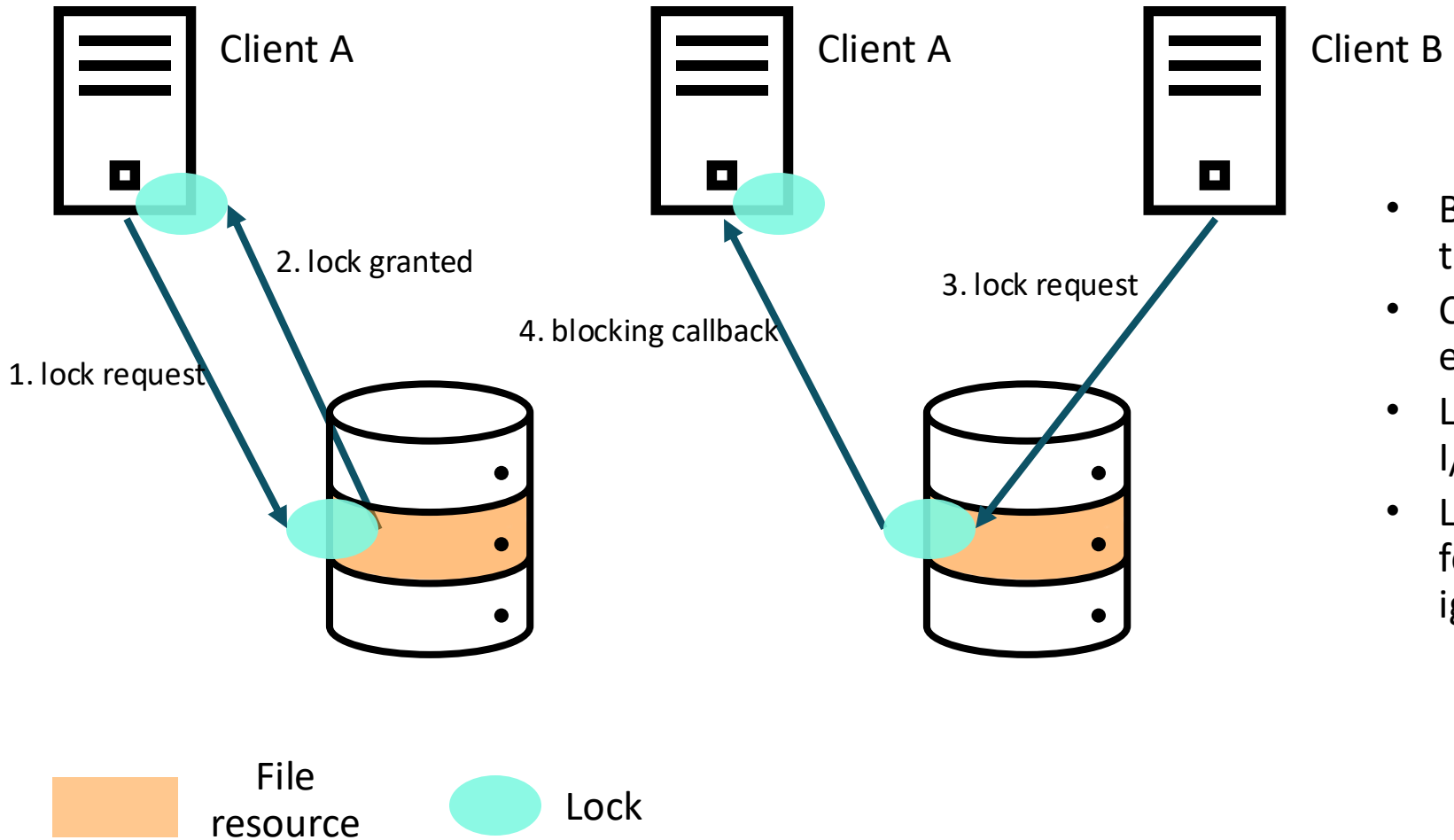
- Message loss resiliency circa 2014-2015 (Lustre 2.7):
  - Closed protocol gap for lock recall
  - Research into tuning timeout hierarchy
- 2015 CUG Paper:
  - https://wiki.lustre.org/Lustre_Resiliency:_Understanding_Lustre_Message_Loss_and_Tuning_for_Resiliency
- Today (Lustre 2.15/2.16):
  - Timeouts at different layers not tied to each other
  - Default tunings not correct for all configurations
  - https://wiki.lustre.org/Lustre_Timeout_Hierarchy

Introduction

- read()/write()/stat()/…
- VFS
- LOV/LMV
- OSC/MDC
- Portal RPC (PtlRPC)
- Lustre Network (LNet)
- Lustre Network Driver (LND)
- kfabric/kVerbs/IP/…



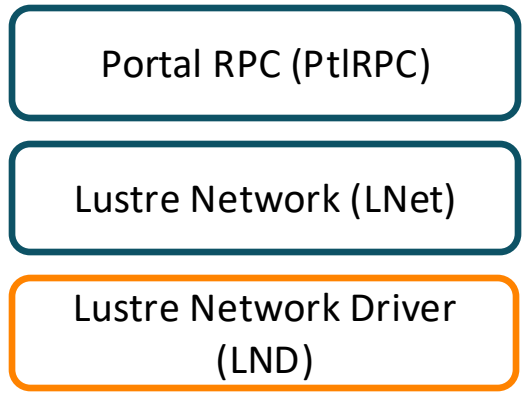
Tuning Goal – Avoid client eviction



- Blocking callback starts lock waiting timeout (LWT)
- Client must cancel lock before LWT expires
- LWT is prolonged as client performs I/O
- LWT ensures clients are not blocked forever if blocking callbacks are ignored

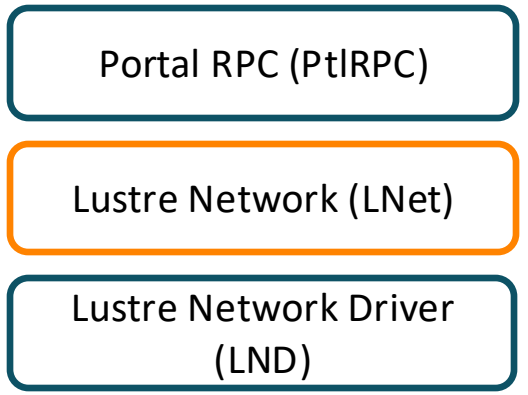
LND Timeouts

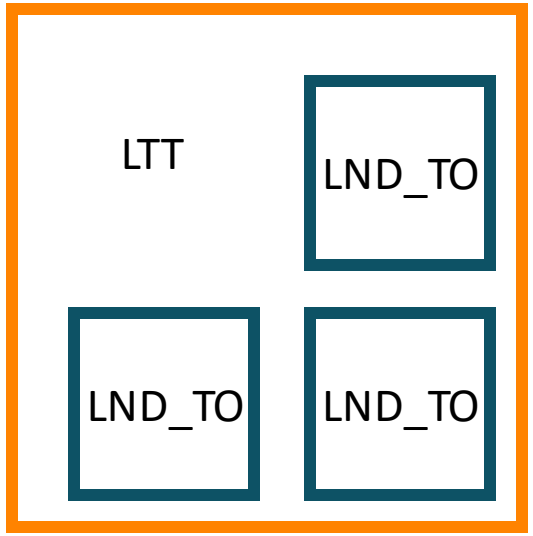
- LND timeout (LND_TO)
  - Common to o2iblnd, socklnd, kfilnd
  - Total time LND will take to complete network transaction (successfully or not!)
  - Accounts for network-specific characteristics
    - Connection vs. connectionless protocols
    - Receiver-not-ready retries
  - May be derived from LNet transaction timeout and LNet retry count parameters (see next slide)
  - May be set explicitly
    - options ko2iblnd timeout=10
    - options ksocklnd sock_timeout=10
    - options kgnilnd timeout=60
    - options kfilnd kfi_timeout=125



LNet Timeouts

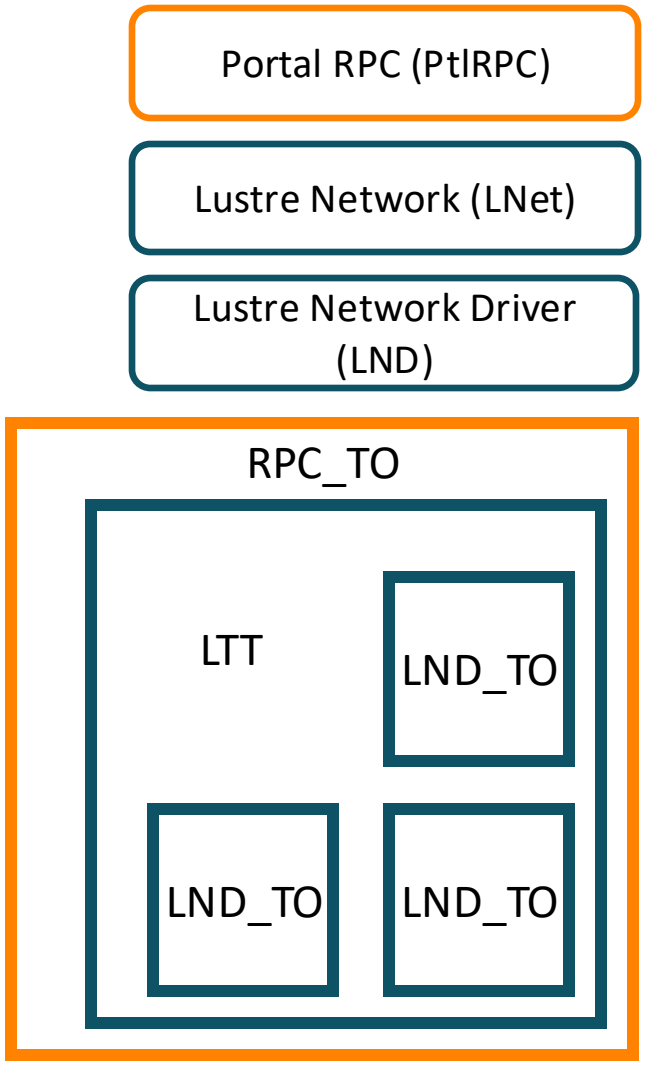
- LNet Transaction Timeout (LTT)
  - Total time LNet will take to complete network transaction (successfully or not!)
  - Used for LNet Multi-Rail features: Retries, Response Tracking, Peer Discovery
  - Accounts for time taken by LNDs (LND timeout) + retries
  - 2.15.6 default 50 seconds
  - 2.16 default 150 seconds
- LNet Retry Count (LRC)
  - Defines the number of times LNet may resend message
  - 2.15.6/2.16 default 2 resends (3 total sends)
- LND_TO = (LTT – 1) / (LRC + 1)
  - 2.15.6 default 16 seconds
  - 2.16 default 49 seconds





Portal RPC (Adaptive) Timeouts

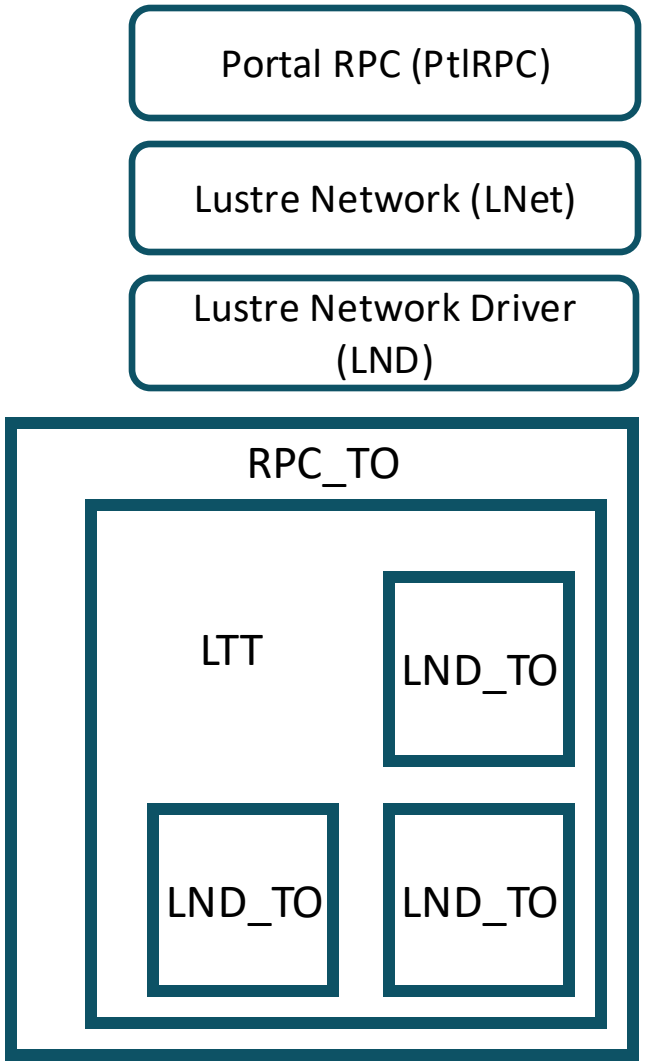
- Lustre uses Adaptive Timeouts to estimate appropriate wait times (RPC_TO)
- AT Components:
  - AT_RPC: Average RPC processing time on server
  - AT_NET: Average network latency time
  - Both bounded by at_min/at_max parameters
  - RPC_TO = AT_RPC + AT_NET >= 2*at_min
- Understanding Lustre Timeouts – LUG 2023
  - Rick Mohr, James Simmons
  - https://wiki.lustre.org/Lustre_User_Group_2023
- Default at_min is 5 seconds, but LTT=150 and LND_TO=49
- RPC timeouts can expire before LNet is finished



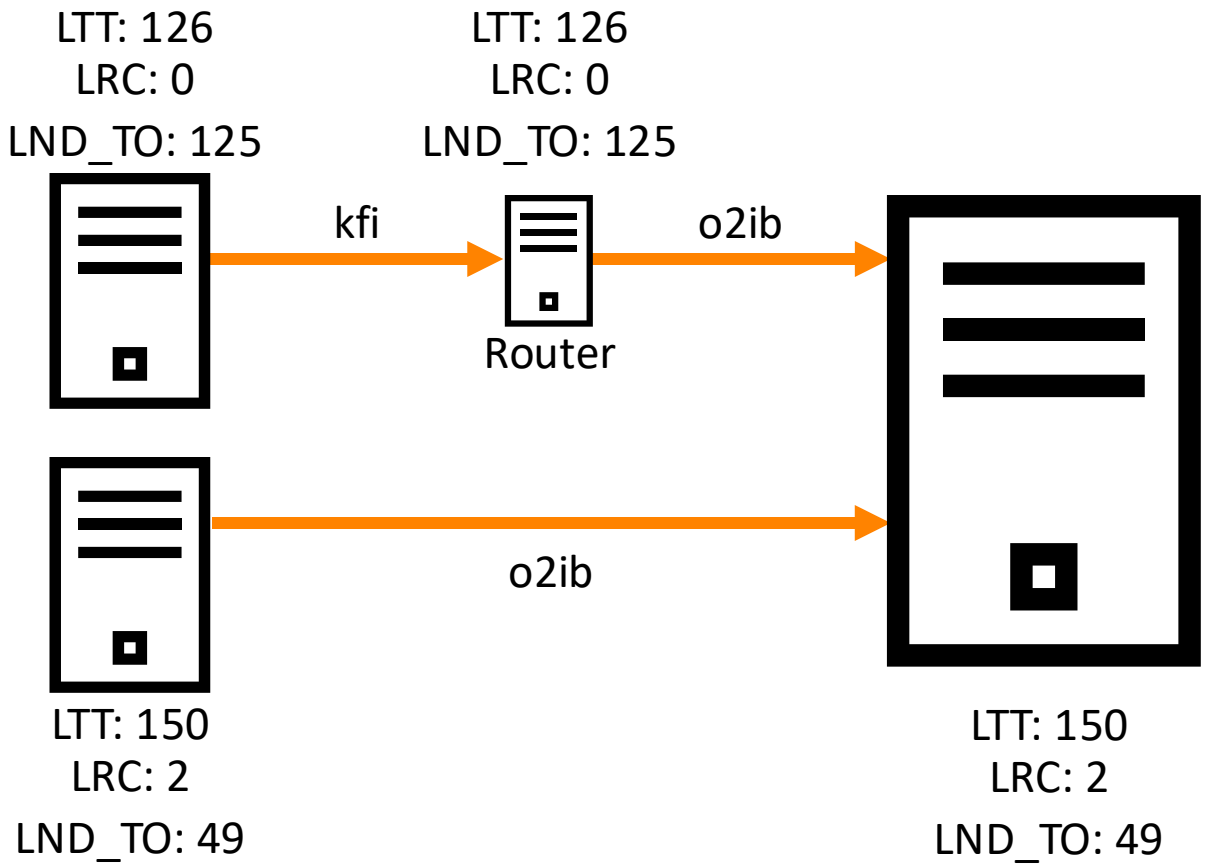
Tuning the Hierarchy – RPC_TO

```
/* We give the server rq_timeout secs to process the req, and
 * add the network latency for our local timeout.
 */
request->rq_deadline = request->rq_sent + request->rq_timeout +
ptlrpc_at_get_net_latency(request);
```

- rq_sent: timestamp when request handed off to LNet
- rq_timeout: estimate based on AT_RPC
  - bounded by at_min/at_max
- ptlrpc_at_get_net_latency() == AT_NET
  - bounded by at_min/at_max
- Goal: RPC should survive single failed network transaction
  - RPC_TO >= 2*at_min
  - LNet may resend successfully within LTT, therefore
  - RPC_TO >= 2*at_min > LTT
  - at_min > LTT / 2

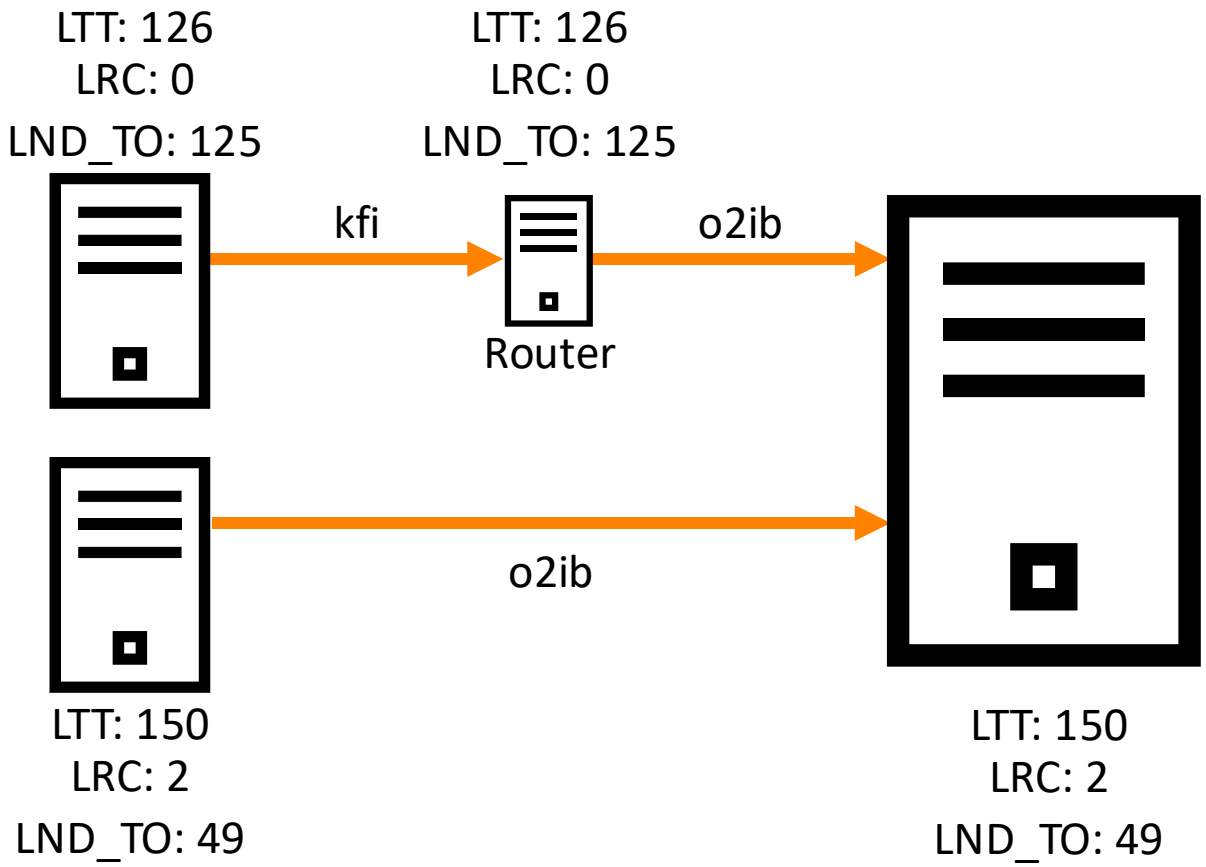


Tuning the Hierarchy – Routers



- LND_TO = (LTT – 1) / (LRC + 1)
- Router config:
  - LND_TO set explicitly
    - options kkfilnd kfi_timeout=125
    - options ko2iblnd timeout=49
  - LTT & LRC = max(client LND_TO, server LND_TO)
- Router checker
  - Uses LNet peer discovery
  - router_ping_timeout = LTT
  - alive_router_check_interval = LTT + 4
  - Future work: Remove router_ping_timeout

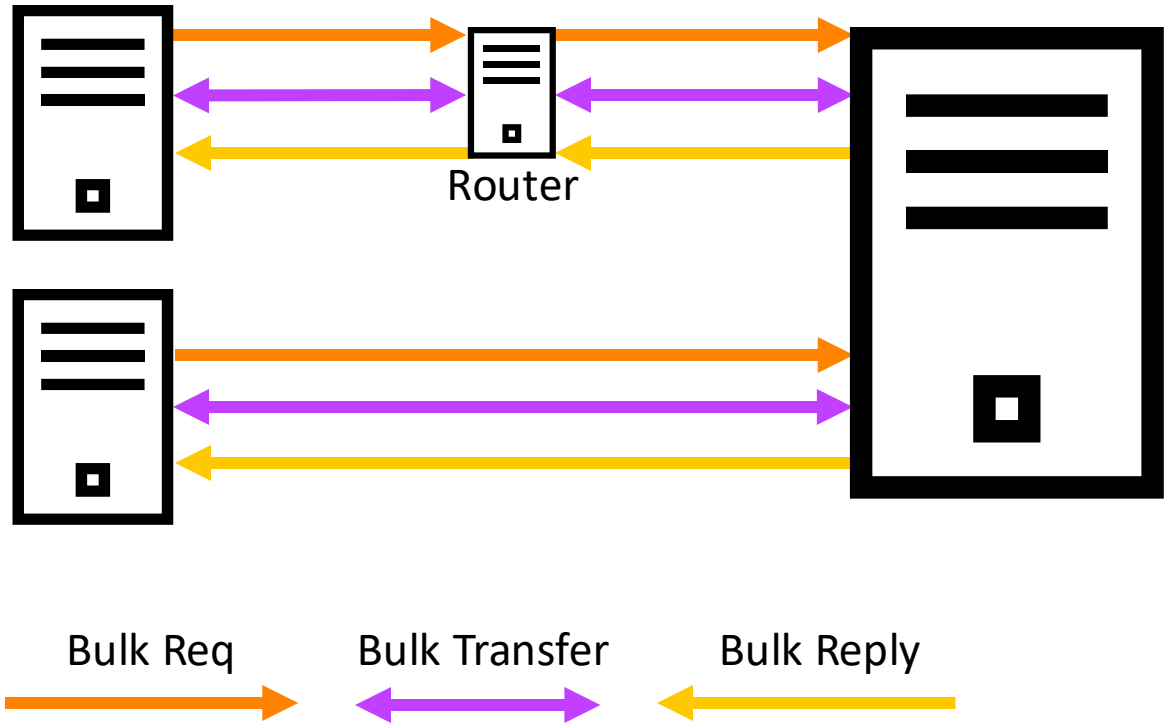
Tuning the Hierarchy – at_min



- at_min > LTT / 2
  - Which LTT?
    - MAX(Server LTT, Client LTT)
  - Router processing time?
    - Call it 5 seconds
  - > ?
    - LTT is just network, but need RPC processing time
    - Call it 5 seconds

at_min = (MAX(Server LTT, Client LTT) +
5 (+ 5 if routed)) / 2 =
(MAX(150, 126) + 5 + 5) / 2 =
(160) / 2 =
80

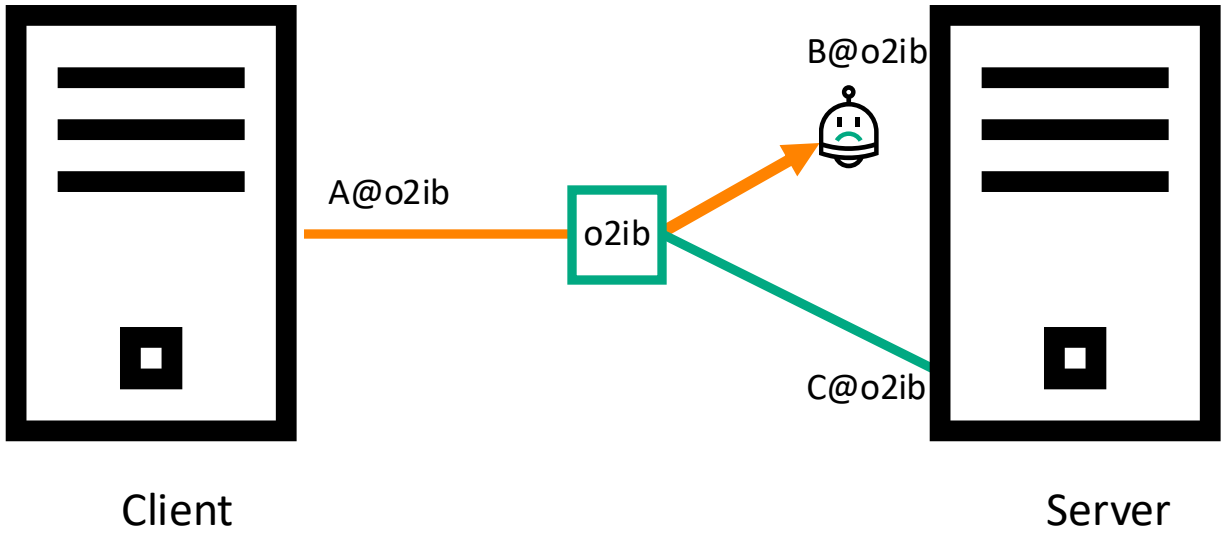
Tuning the Hierarchy – bulk_timeout



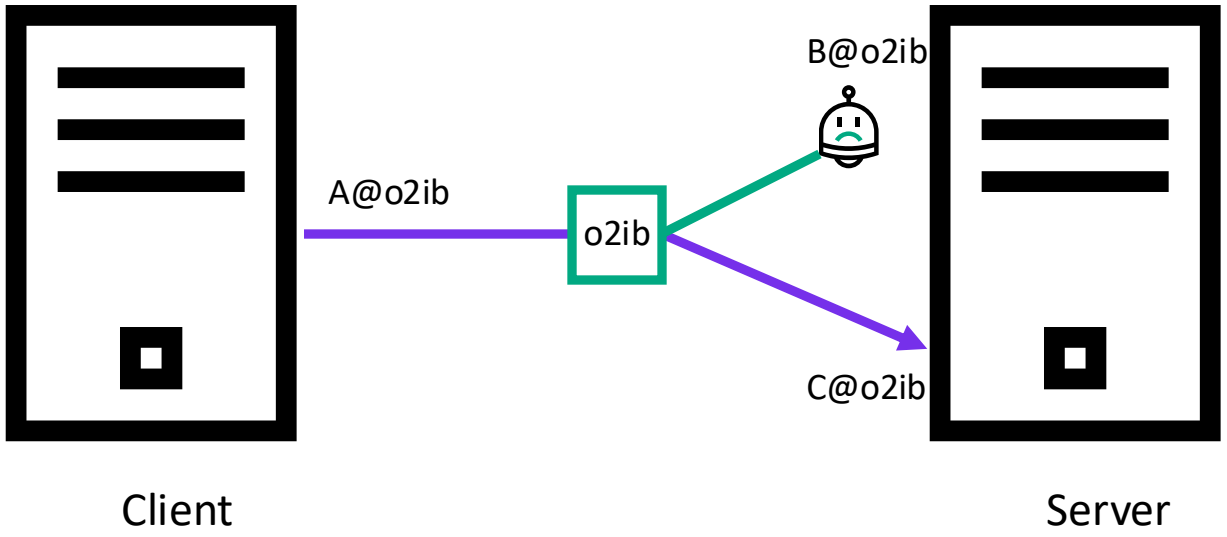
- Covers just bulk transfer
- Server side only
- Default 100s
- bulk_timeout > LTT
- mgs # lctl set_param -P bulk_timeout=<value>

Tuning Goal – Avoid client eviction

- Determine LWT for server NID failure
  - RPC Timeout
    - RPC_TO > 2*at_min

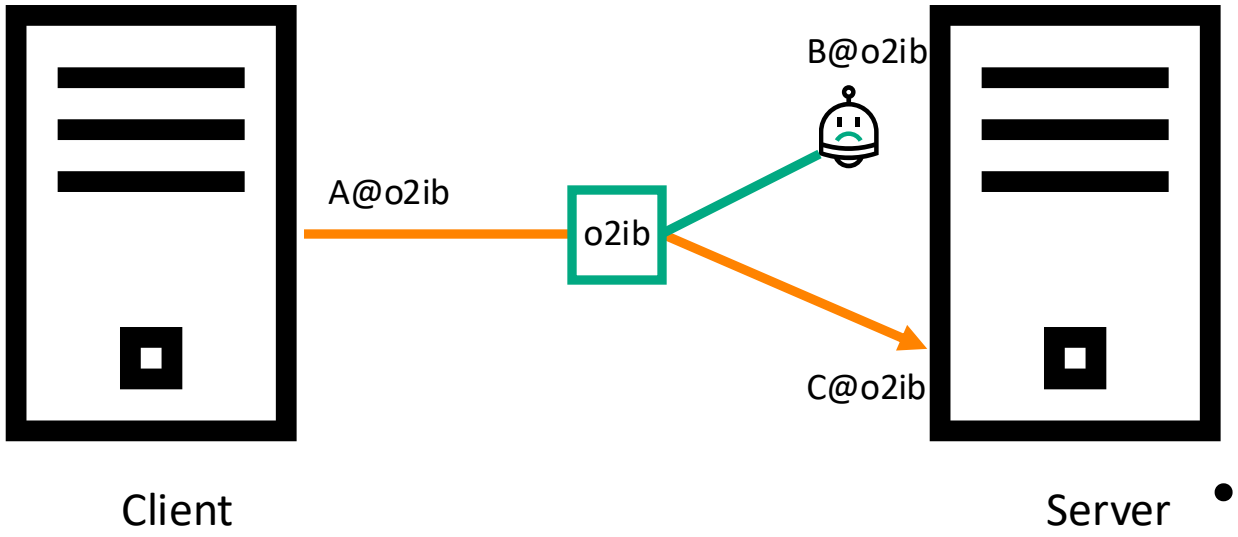


Tuning Goal – Avoid client eviction



- Worst case server NID failure
  - RPC Timeout
    - RPC_TO > 2*at_min
  - PtlRPC Reconnect
    - INITIAL_CONNECT_TIMEOUT (5s) + at_min (AT_NET)

Tuning Goal – Avoid client eviction



- Worst case server NID failure
  - RPC Timeout
    - RPC_TO > 2*at_min
  - PtlRPC Reconnect
    - INITIAL_CONNECT_TIMEOUT (5s) + at_min
  - RPC resend
    - RPC_TO > 2*at_min
  - LWT > 5*at_min + INITIAL_CONNECT_TIMEOUT
- ldlm_enqueue_min parameter for rare cases when LWT may not cover worst case
  - e.g. Aries network could quiesce
  - ldlm_enqueue_min = max(2*net latency, net latency + quiescent time) + 2 * service time

Are the defaults too large?

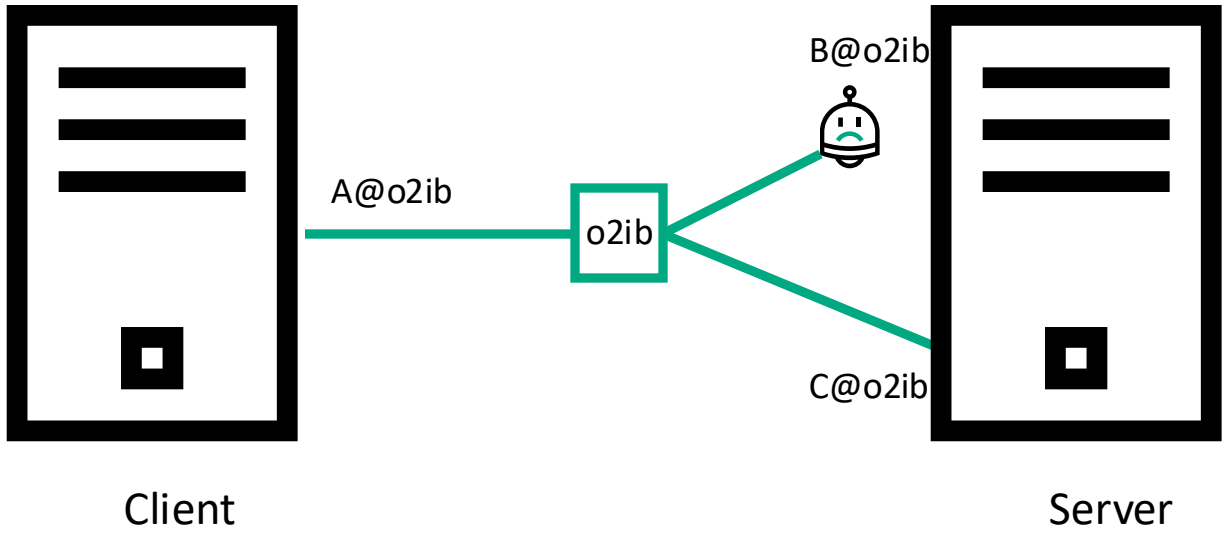
LTT	LRC	LND_TO	at_min	LWT
150	2	49	80	405
100	1	49	50	255
50	0	49	25	130
31	2	10	18	95

Dynamically Configured Timeouts

- LU-18566
  - LTT was historically one-size-fits-all
  - Removes need for defining LTT
  - LNet selects a NI for send and queries LND for timeout
  - LTT = (LND_TO * LRC) + 1
  - By default, LND timeouts are statically defined (again)
  - Landed for 2.17

Future Work

- Extend LWT logic to ping eviction timer (LU-18878)
- LWT > 5*at_min + INITIAL_CONNECT_TIMEOUT
  - Increase probability of 1st re-connect RPC
    - Can we ensure secondary interface is targeted?
  - Split at_min = at_net_min + at_rpc_min
- Derive LRC from RPC_TO
  - PtlRPC provides RPC_TO to LNet
  - LNet determines LND_TO
  - LRC = (RPC_TO / LND_TO)
- Per-LND retry count parameters
- Can we derive all LNet/LND timeouts from RPC_TO, or
- Can we derive upper layer timeouts from LNet?



Sample configurations

- For ko2iblnd and ksocklnd configurations:
  - options ko2iblnd timeout=10
  - options ksocklnd sock_timeout=10
  - options lnet lnet_retry_count=2
  - options lnet lnet_transaction_timeout=31
- For kfilnd configurations:
  - options lnet lnet_retry_count=0
  - options lnet lnet_transaction_timeout=126
  - Server bulk timeout = 127 (mgs # lctl set_param -P bulk_timeout=127)
- For kgnilnd configurations:
  - options kgnilnd timeout=60
  - options lnet lnet_retry_count=0
  - options lnet lnet_transaction_timeout=61

Sample configurations

- Routed examples:
  - Routing ko2iblnd ↔ kfilnd:
    - options lnet lnet_retry_count=0
    - options lnet lnet_transaction_timeout=126
    - options lnet router_ping_timeout=126
    - options lnet alive_router_check_interval=130
  - Routing ko2iblnd ↔ ksocklnd:
    - options lnet lnet_retry_count=2
    - options lnet lnet_transaction_timeout=31
    - options lnet router_ping_timeout=31
    - options lnet alive_router_check_interval=35
  - Routing ko2iblnd/ksocklnd ↔ kgnilnd:
    - options lnet lnet_retry_count=0
    - options lnet lnet_transaction_timeout=61
    - options lnet router_ping_timeout=61
    - options lnet alive_router_check_interval=65

Sample at_min configurations

- ko2iblnd/ksocklnd servers directly connected to ksocklnd/ko2iblnd clients:
  - at_min = (MAX(31, 31) + 5 + 0) / 2 = 18
- kfilnd servers directly connected to kfilnd clients:
  - at_min = (MAX(126, 126) + 5 + 0) / 2 = 66
- ko2iblnd/ksocklnd servers connected via routers to kgnilnd clients:
  - at_min = (MAX(31, 61) + 5 + 5) / 2 = 36
- ko2iblnd/ksocklnd servers connected via routers to kfilnd clients:
  - at_min = (MAX(31, 126) + 5 + 5) / 2 = 68

Thank you

chris.horn@hpe.com