# o2iblnd connection management

This page documents reference counter strategy used with o2iblnd connections:

- In **kiblnd_create_conn**()
  - atomic_set(&conn->ibc_refcount, 1 + IBLND_RX_MSGS(conn));
    - 1 ref for caller and 1 ref for each rxmsg
    - The 1 ref for caller is decremented when the connection fails:
      - kiblnd_passive_connect() if rdma_accept() fails
      - kiblnd_active_connect() if rdma_connect() fails
      - kiblnd_cm_callback() in most of the failure cases:
        - RDMA_CM_EVENT_UNREACHABLE
        - RDMA_CM_EVENT_CONNECT_ERROR
        - RDMA_CM_EVENT_REJECTED
        - RDMA_CM_EVENT_DISCONNECTED
- In **kiblnd_post_rx()**. When an rx buffer is posted using ib_post_recv()
  - This is needed because another thread could destroy the connection.
  - `kiblnd_conn_decref()` is called at the end of the function.
  - kiblnd_drop_rx() also calls kiblnd_conn_decref() if the ib_post_recv() fails
- In **kiblnd_queue_tx_locked()**: When a new tx is queued on the connection. That means each tx on that connection grabs a refcount.
  - This refcount remains until the call to kiblnd_tx_done() which finalizes the tx.
    - Any txs that are queued but not posted can be removed when the connection is completed/aborted.
      - kiblnd_conn_decref() is called to remove the refcount used by the tx.
    - txs which are sent (ibc_sent > 0) the code waits for those to complete before completing the tx.
      - This means that a connection can hang around for a while longer (not sure how long, maybe indefinitely) if the communication stack has a bug.
- In **kiblnd_launch_tx()**
  - function grabs the refcount to be able to unlock the g_lock and queue the tx using kiblnd_queue_tx()
  - It calls kiblnd_conn_decref() after it's done with the connection.
    - This kiblnd_conn_addref()/kiblnd_conn_decref() are paired in this function.
- In **kiblnd_connreq_done()**
  - When the connection is established and added to the peer's list of connections (peer_niibp_conns)
  - This is decremented in kiblnd_connd().
    - When a connection is being closed (kiblnd_close_conn_locked()) it's added to the kiblnd_data.kib_connd_conns.
    - The kiblnd_connd() wakes up periodically and disconnects the connection, after which it decrements the refcount.
  - Once the connection is added to ibp_conns then it is available to be picked up by scheduler threads or other sending threads. Any further access to the connection needs to grab a refcount. This happens in this function to:
    - kiblnd_check_sends_locked(conn);
    - kiblnd_handle_early_rxs(conn);
- In **kiblnd_check_conns()**
  - When connections timeout or if we need to check that we have to see if there are enough credits to send further txs, then we add those connections to a local closes and checksends lists. These are processed locally in the function. So a refcount is grabbed and released after the connections on the list are processed.
- In **kiblnd_cq_completion()**
  - IB stack calls this function whenever there is a complete even on the completion queue (cq)
  - refcount is acquired when the connection is added on the scheduler list: schedibs_conns
  - The refcount is decremented when that conn is processed by the kiblnd_scheduler() thread. See below.
- In **kiblnd_scheduler()**
  - The cq is polled and if there is >=1 completions two things are done:
    - We add the connection back to the schedibs_conns list and grab a refcount.
    - We also process the completion using kiblnd_complete()
  - kiblnd_conn_decref() is called to release the refcount grabbed in kiblnd_cq_completion()
- In **kiblnd_rx_complete()**
  - This function is called when an rx completion event is processed.
  - If the connection is closing it'll call kiblnd_drop_rx() which decrements the refcount.
    - rdma_disconnect() called from kiblnd_finalise_conn() triggers all posted rxs to be completed. An rx completion event is generated for each.
    - Each one of these events are hanlded by kiblnd_rx_complete() and kiblnd_drop_rx() is called for each therefore taking away the reference counts added for each rx posted when the connection is created.
  - In the normal case, it calls kiblnd_handle_rx()
    - This function re-posts the rx again inline or calls lnet_parse() which eventually calls kiblnd_recv() which then re-posts the rx; thereby maintaining the correct refcount on the connection.