Implementing an LND for the BXIv3 network

Quentin Boyer
30/09/2025

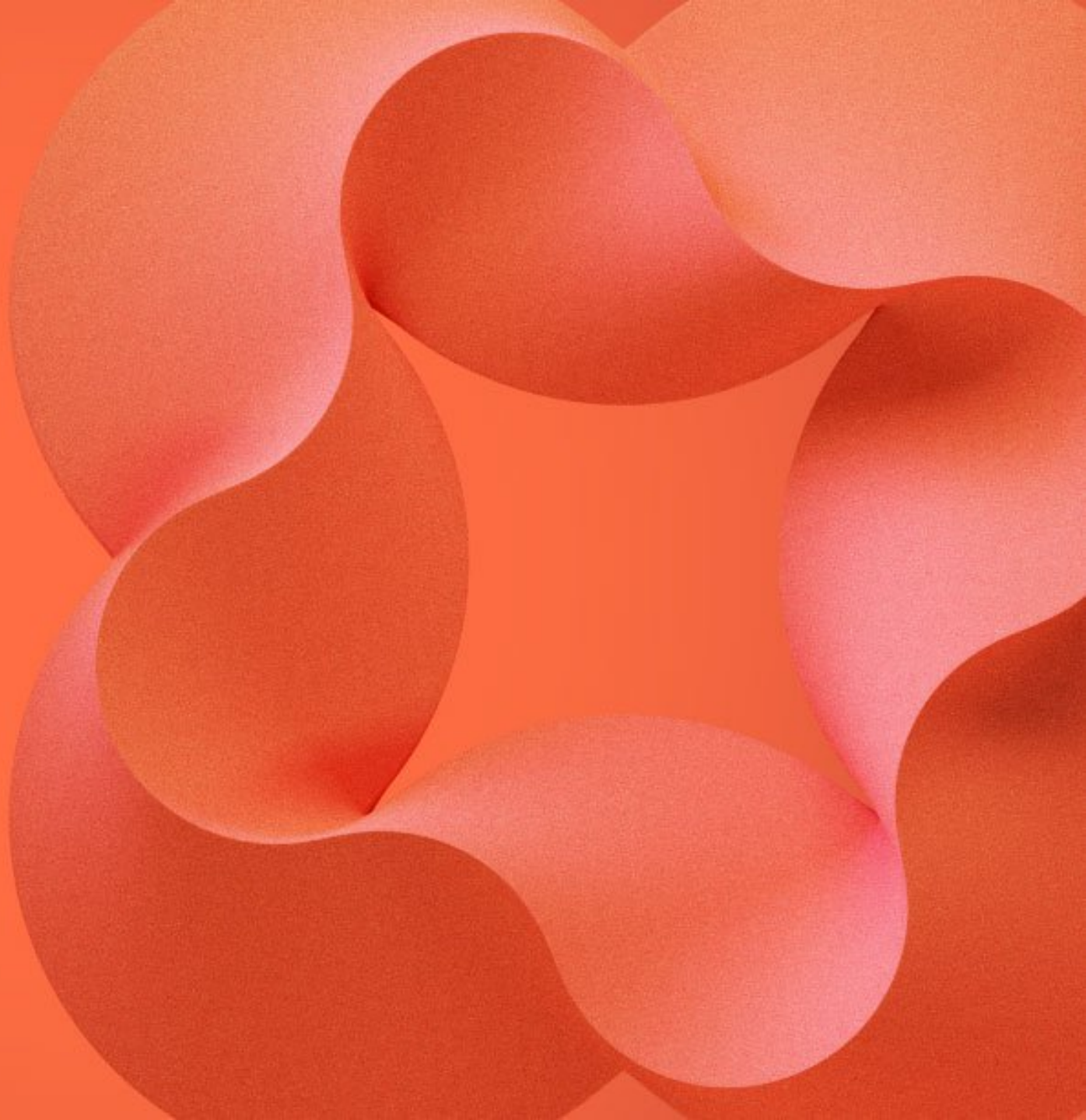
EVIDEN

Content overview

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101175702. The JU receives support from the European Union's Horizon Europe research and innovation programme and France, Greece, Italy, Norway.

EVIDEN

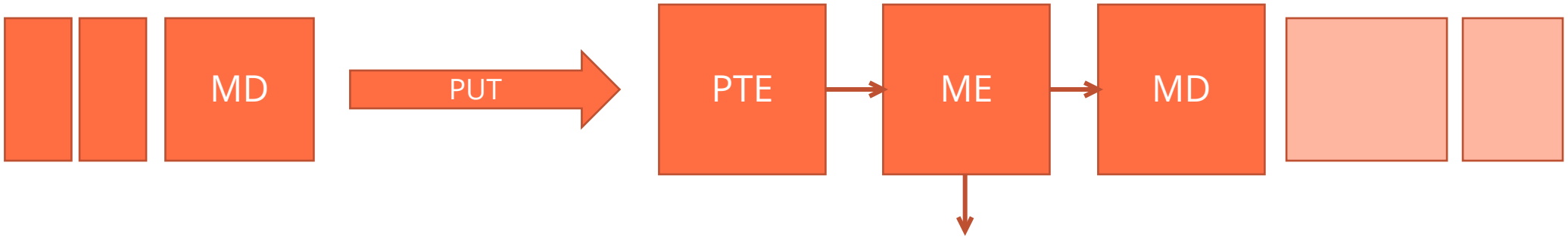
01 LNet and LNDs



LNet semantics

LNet is based on **Portals 3** semantics.

It allows callers to expose memory descriptors (MD) to the network, using match entries (ME) to allow receiving data on them.



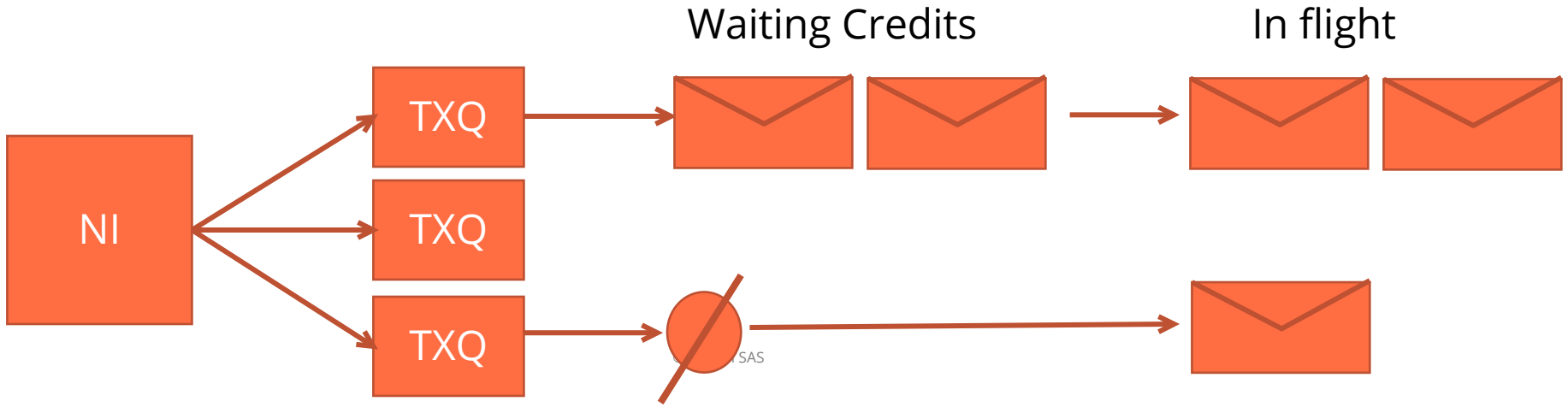
The role of an LND is to implement the data transfer part of a put or get operation. LNet performs the matching of entries and bookeeping of memory descriptors itself.

LNet scalabilty

In order to guarantee scalability there are two important concepts :

- CPT, CPU Partition Table, which allow independent progression of operations on distinct CPUs
- Credit management, which allows to limit the amount of messages sent on the fabric to avoid overwhelming it

In particular, this means that each NI has **n** transmission queues (one for each CPT), each using independent credits.



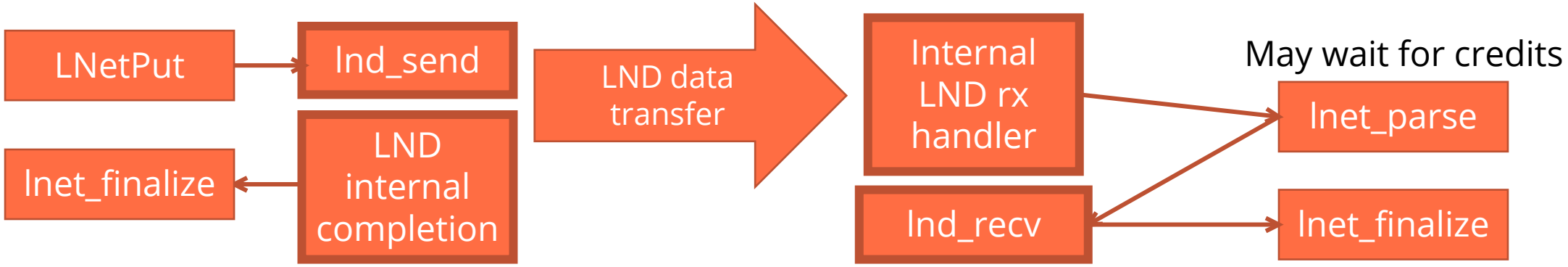
LNet wire protocol

LNet uses 4 kind of messages : Put, Ack, Get, Reply
Messages can have lengths from 0 bytes to 1 MiB
Data payloads may be split into 256 iovec segments

LND Interface

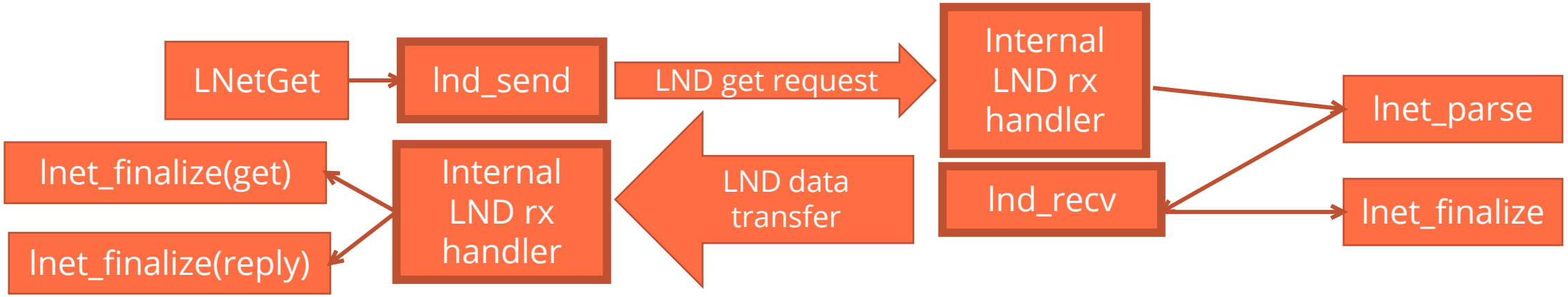
An LND (Lustre Network Driver) has two main callbacks to implement :

- `lnd_send`
- `lnd_recv`



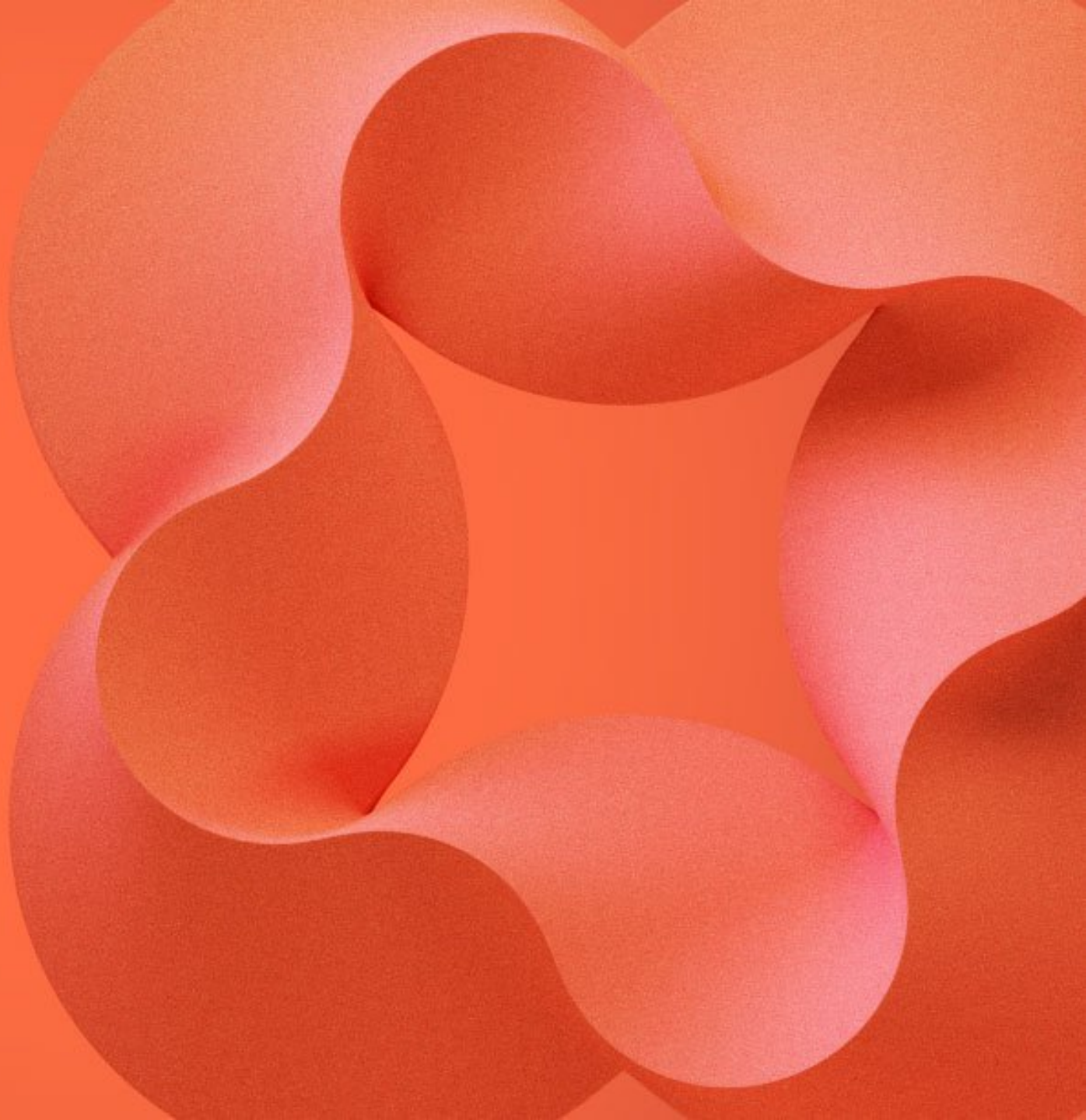
LND Get optimization

LNet allows LND to bypass it to send data from a Get :



EVIDEN

02 BXI Interconnect



BXI Interconnect

Current version : BXIv2
Based on the Portals4 specification, mostly implemented in hardware
Performance : ~90 Gb/s, 20 Million msg/s, 2.5 µs

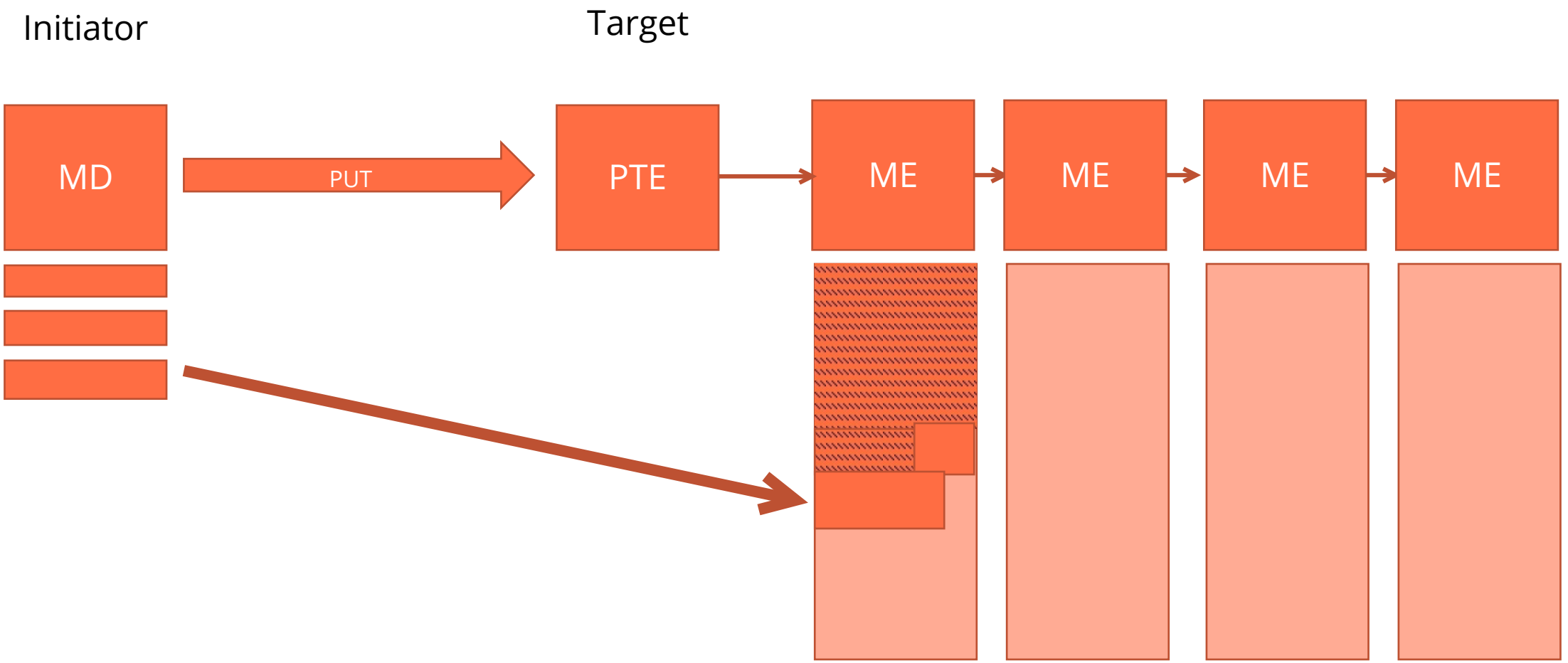
Version in development : BXIv3
Still based on the Portals4 specification
Performance targets : 400 Gb/s, 200 Million msg/s, 1 µs
New features : Congestion management, Increased Parallelism, Increased Asynchrononism

Lustre with BXI

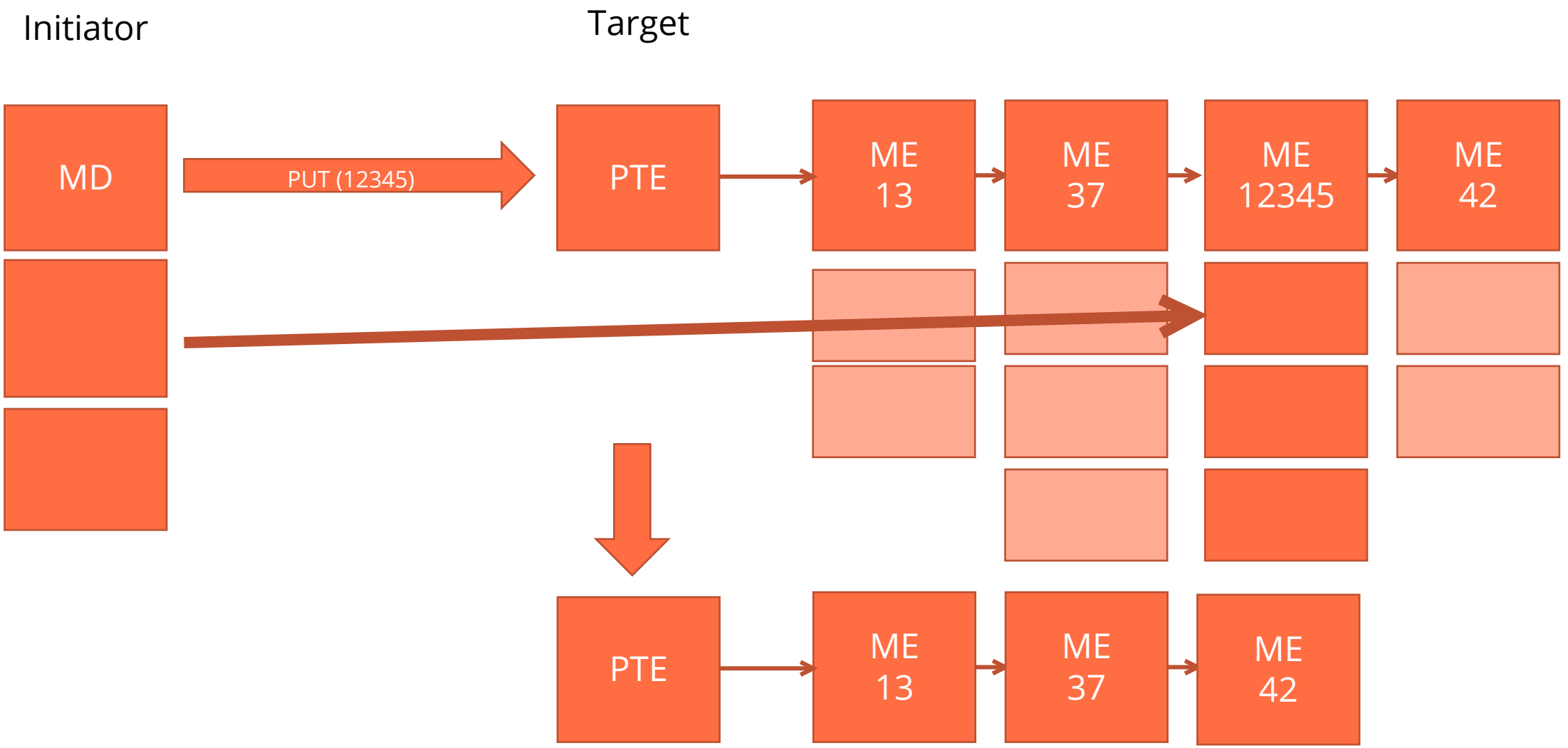
Eviden has developed an LND named **ptl4lnd** for BXIv2.
This LND has not been upstreamed.

In order to take advantage of new features in BXIv3 a new LND is being developed, named **bxi3lnd**.
This LND is meant to be upstreamed once it is sufficiently featureful.

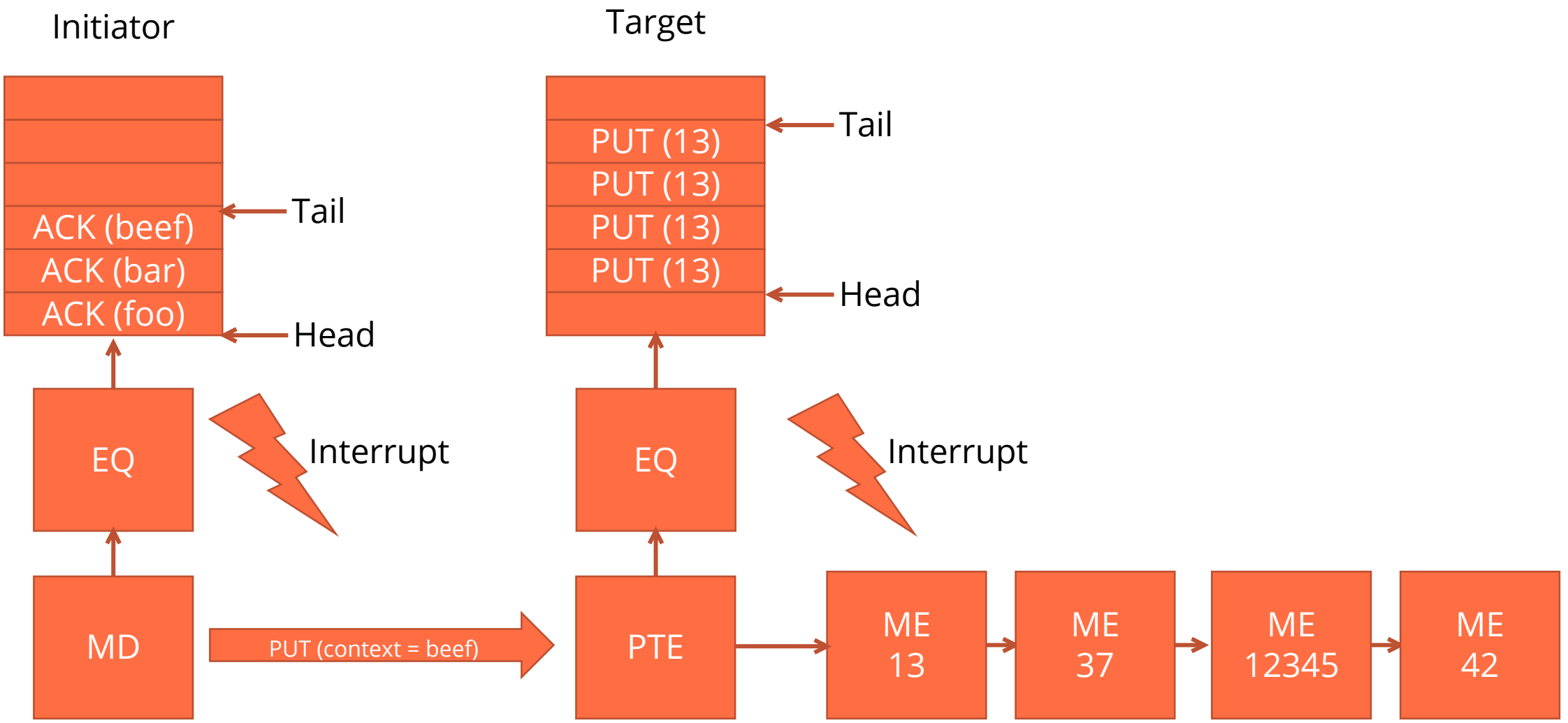
Portals 4.3 Semantics : Locally managed entries



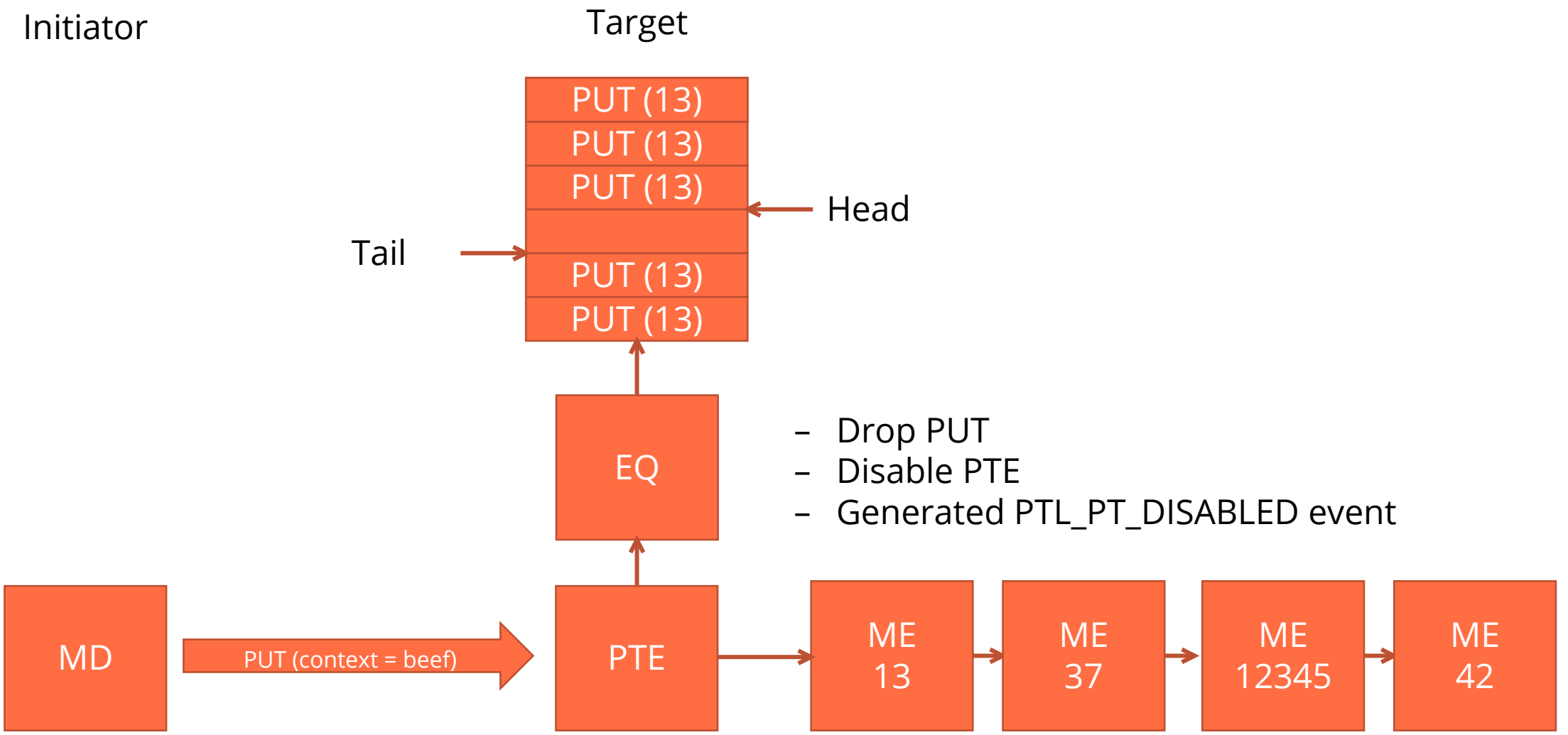
Portals 4.3 Semantics : Use once entries



Portals 4.3 Semantics : Event notification

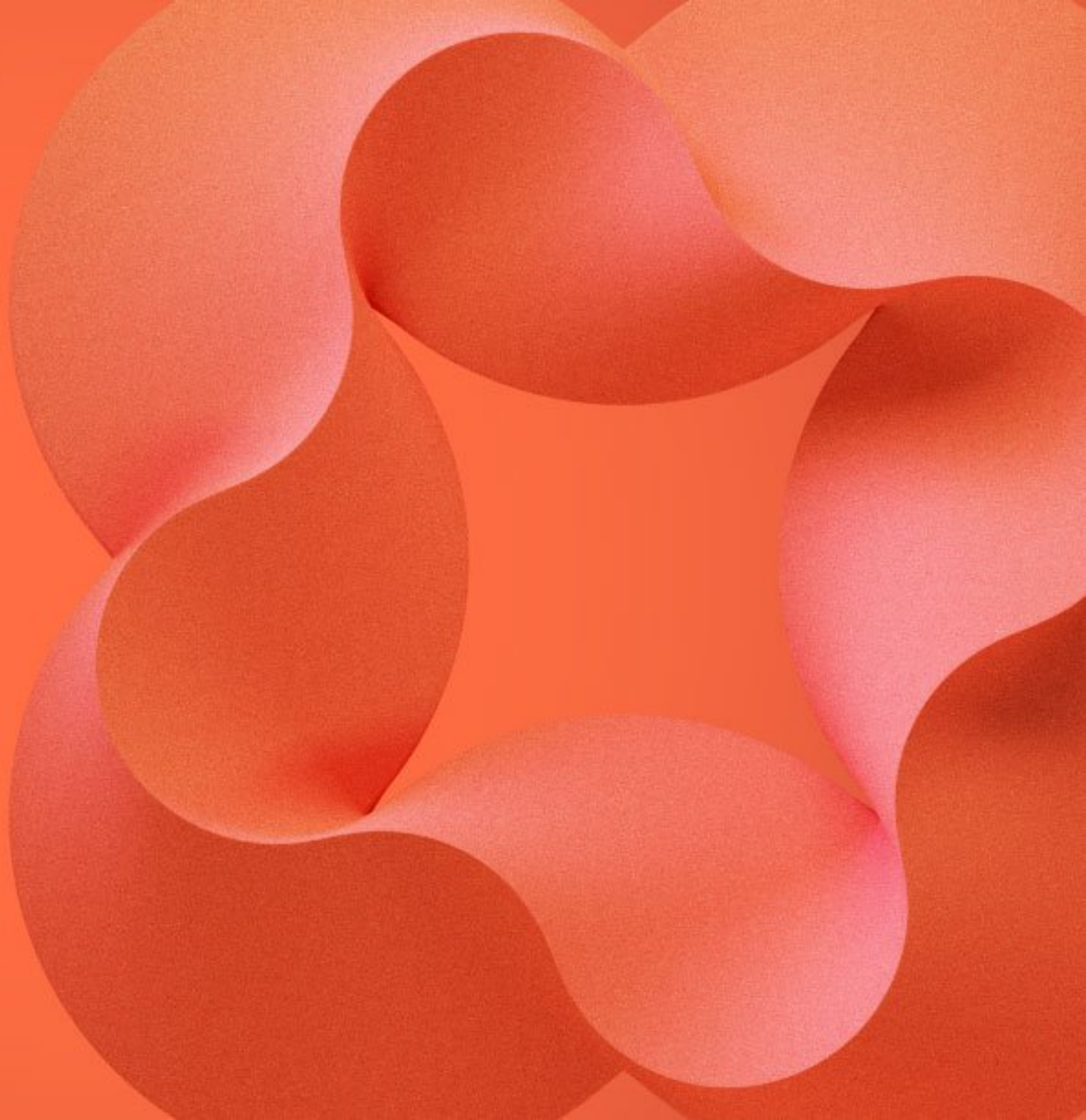


Flow control



EVIDEN

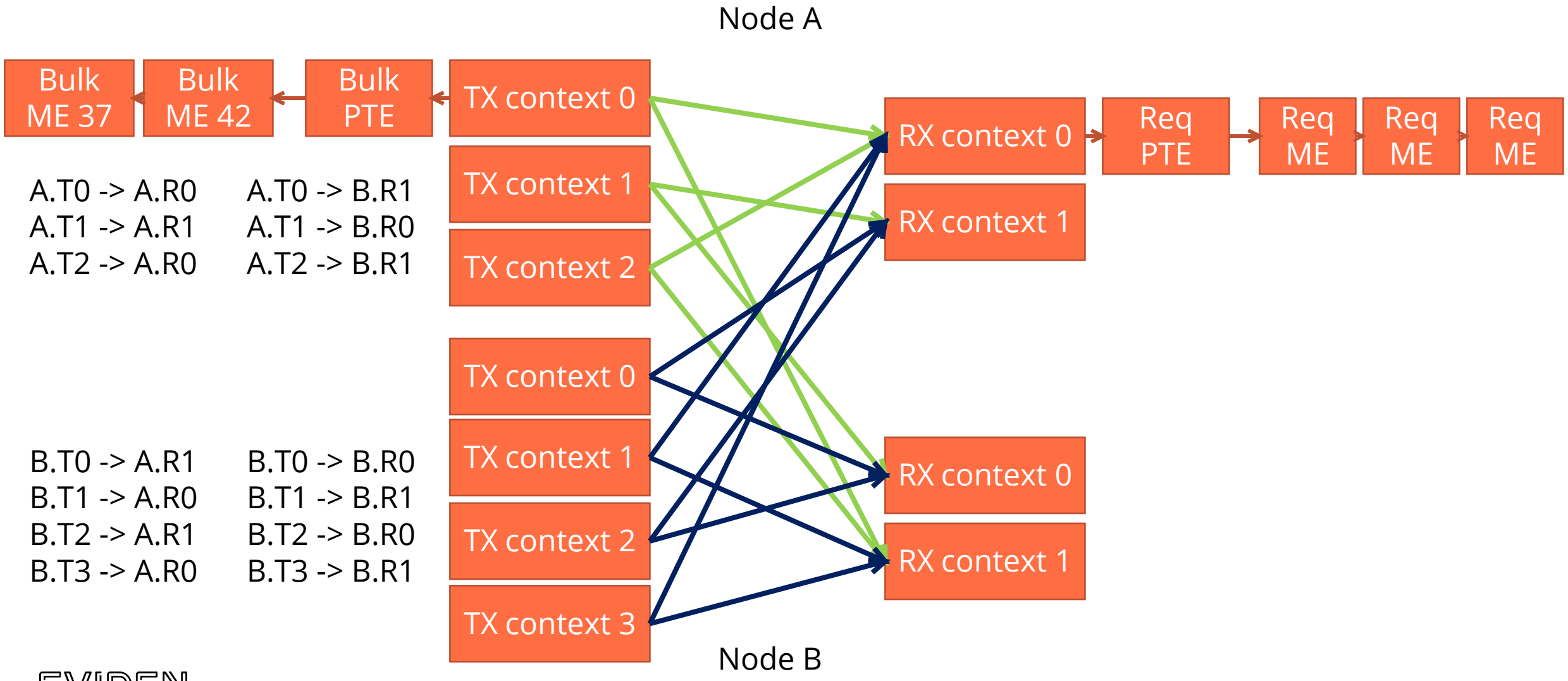
03 Design of bxi3lnd



Design Principles

- Saturate the BXIv3 NIC (for message rate and for bandwith)
- Try to rely as much as possible on Portals features exposed by the NIC
- Use as much parallelism as possible
- Do not maintain a state (by default) for each peer

Global Architecture

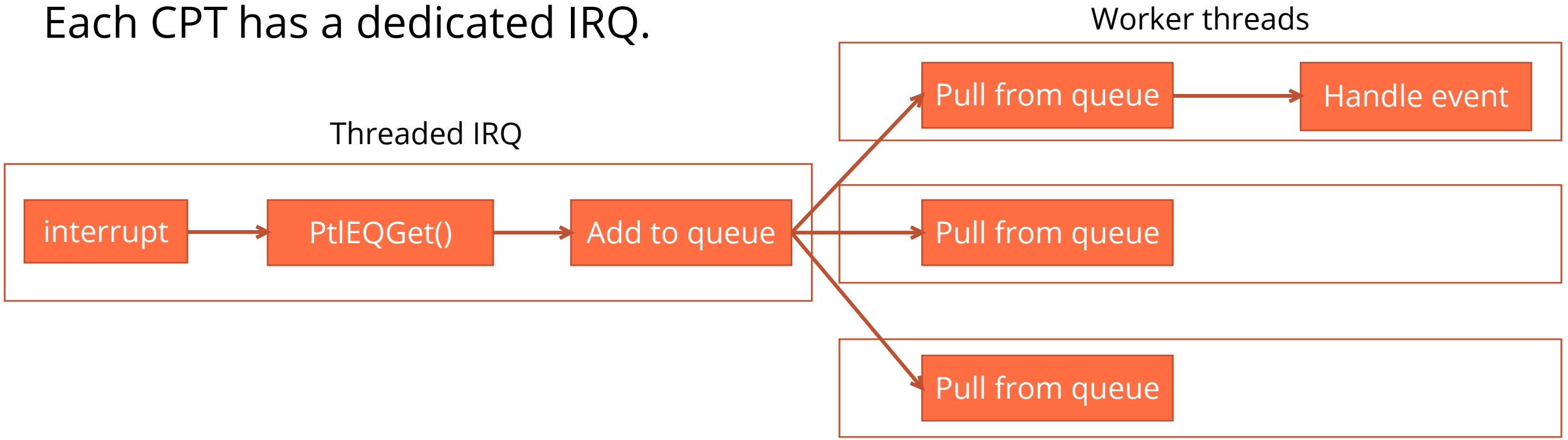


EVIDEN

Handling of events

In order to dispatch events to multiple threads (if a CPT uses them) we have the following architecture.

Each CPT has a dedicated IRQ.



Message types

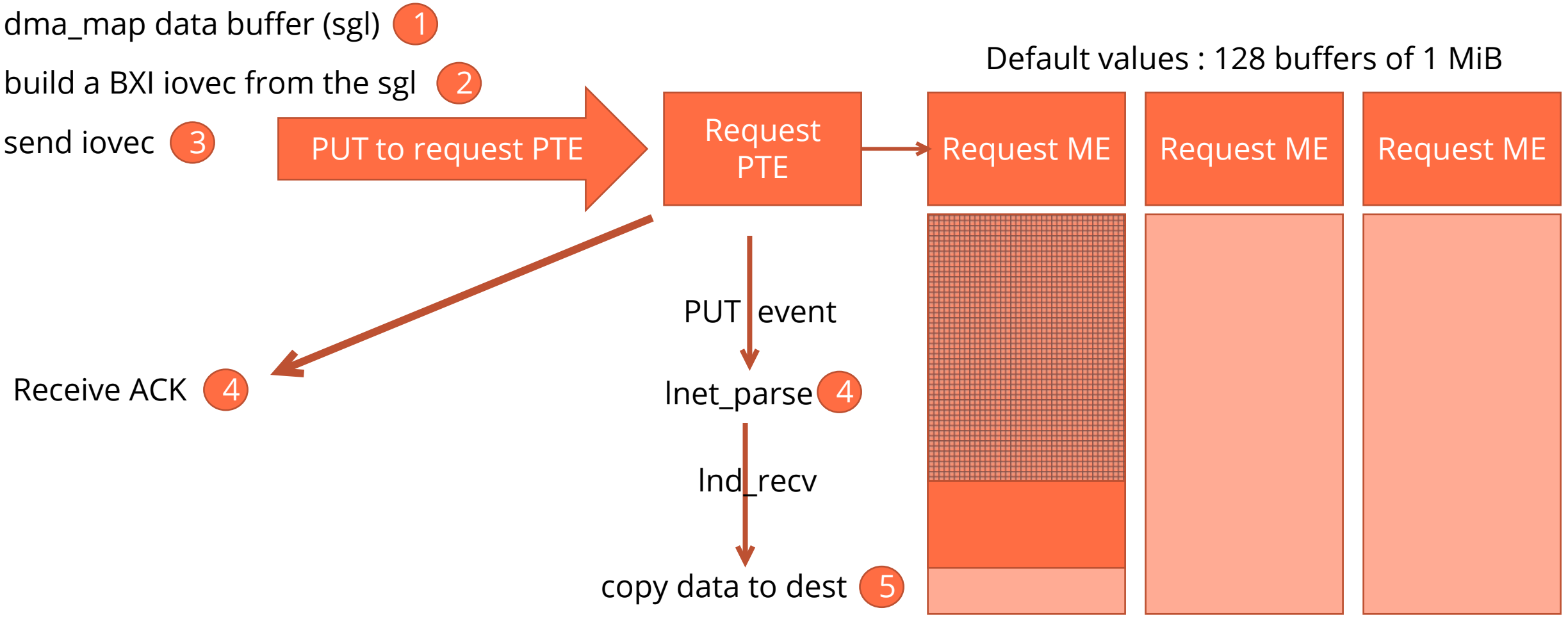
The BXIv3 LND uses (like most LNDs) two kind of messages :

- Immediate (for ACKs, small GET, small REPLY and small PUT)
- Bulk (for large GET, PUT and REPLY)

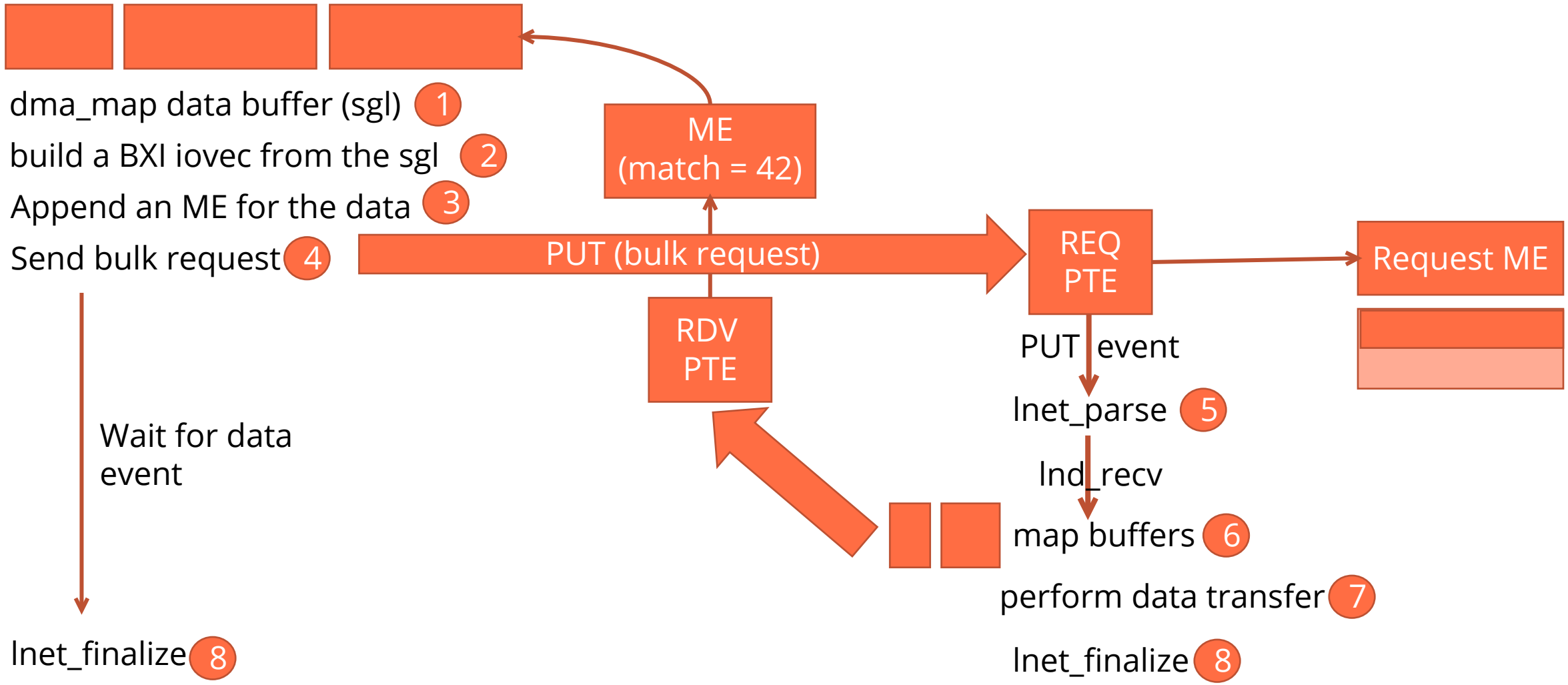
An immediate must fit inside an MTU, meaning it can be at most 9088 bytes total, with 8980 bytes of data payload.

Bulk GET messages can’t target routers, an immediate GET will be performed, resulting in a bulk REPLY.

Immediate messages



Bulk messages



Bulk message timeouts

Immediate messages : Request only

Bulk messages : Request + Reply

→ Issue if reply never comes

Handling the timeout is tricky, because there can be a race between the timeout triggering and the reply arriving on the network

Ressource exhaustion

The LND has several resources that can be exhausted :

- Event queue slots
  - TX EQ → Size = 2 × credits, can’t be overflown by LNet
  - RX EQ → Enable portals flow control on RX PTE
  - Bulk EQ → Queue transmissions while too many are in-flight
- Request buffers on RX contexts
  - Will trigger flow control
- Space in the event workqueue
  - Avoid calling PtlEQGet() → Keep getting interrupted

Flow control recovery procedure

Two possible causes on the target :

- EQ exhaustion → Solved by the time we have found the event
- Buffer exhaustion : Two situations
  - Buffers are being reposted → Nothing to do
  - Too many buffers are still in use due to a lnet_parse call → Allocate new buffers and avoid re-posting the old ones

Inititiator handling : Resend the affected messages

EVIDEN

04 Remaining Work

Implementation status

Not all the described design has been implemented :

- No bulk LNetPut (all other communication are done)
- No router handling / testing
- Only 1 RX context and 1 TX context is used (but all RX and TX contexts are allocated)
- Missing some debugging informations
- Missing some injections for misc error paths (major paths like timeouts are tested)

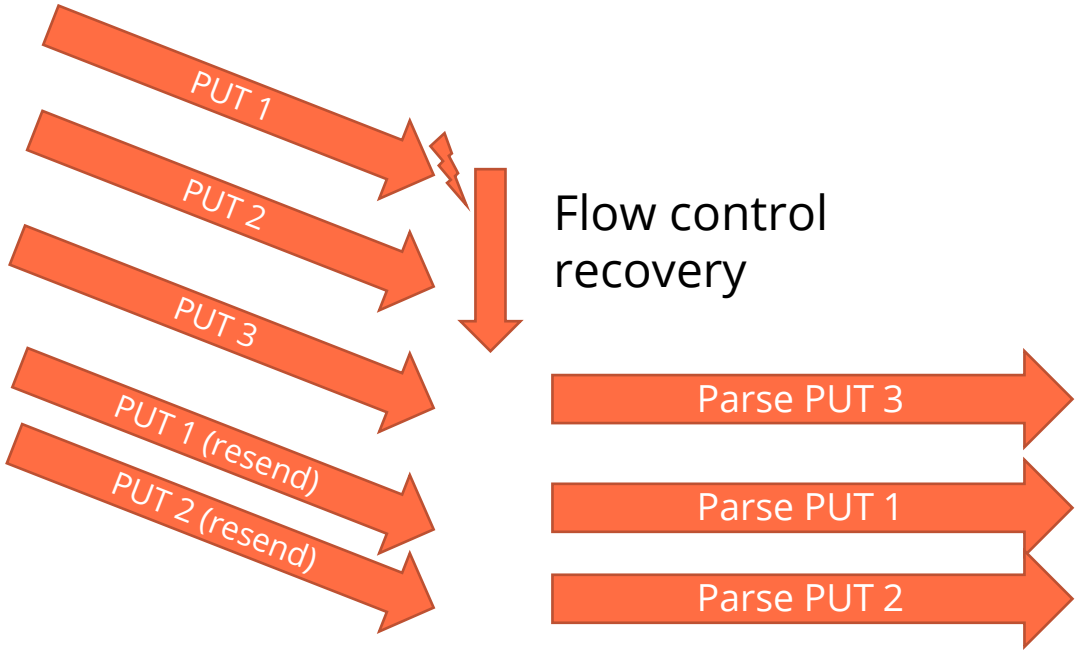
Tuning to be done when hardware is availaible

- How many RX contexts ? 8 for now
- How many RX buffers ? Which size ?
- Optimal credits count ?
- Timeouts ?
  - Request timeout (LNet level)
  - Bulk timeout (LND level)
- Linearization or iovec for immediate messages ?
- Immediate vs Bulk limit ?

Unanswered questions

The presented design is mostly ordered with two main re-ordering points, should the LND internally re-order the messages ? :

- Event re-ordering in the NIC / Races in event queue workers
- Reordering due to flow control recovery



EVIDEN

Questions ?

EVIDEN

Thank you !

For more information please contact :
quentin.boyer@eviden.com

Bulk message timeouts race handling