Based on 6.0.x branch

ATS INTERNALS

Agenda

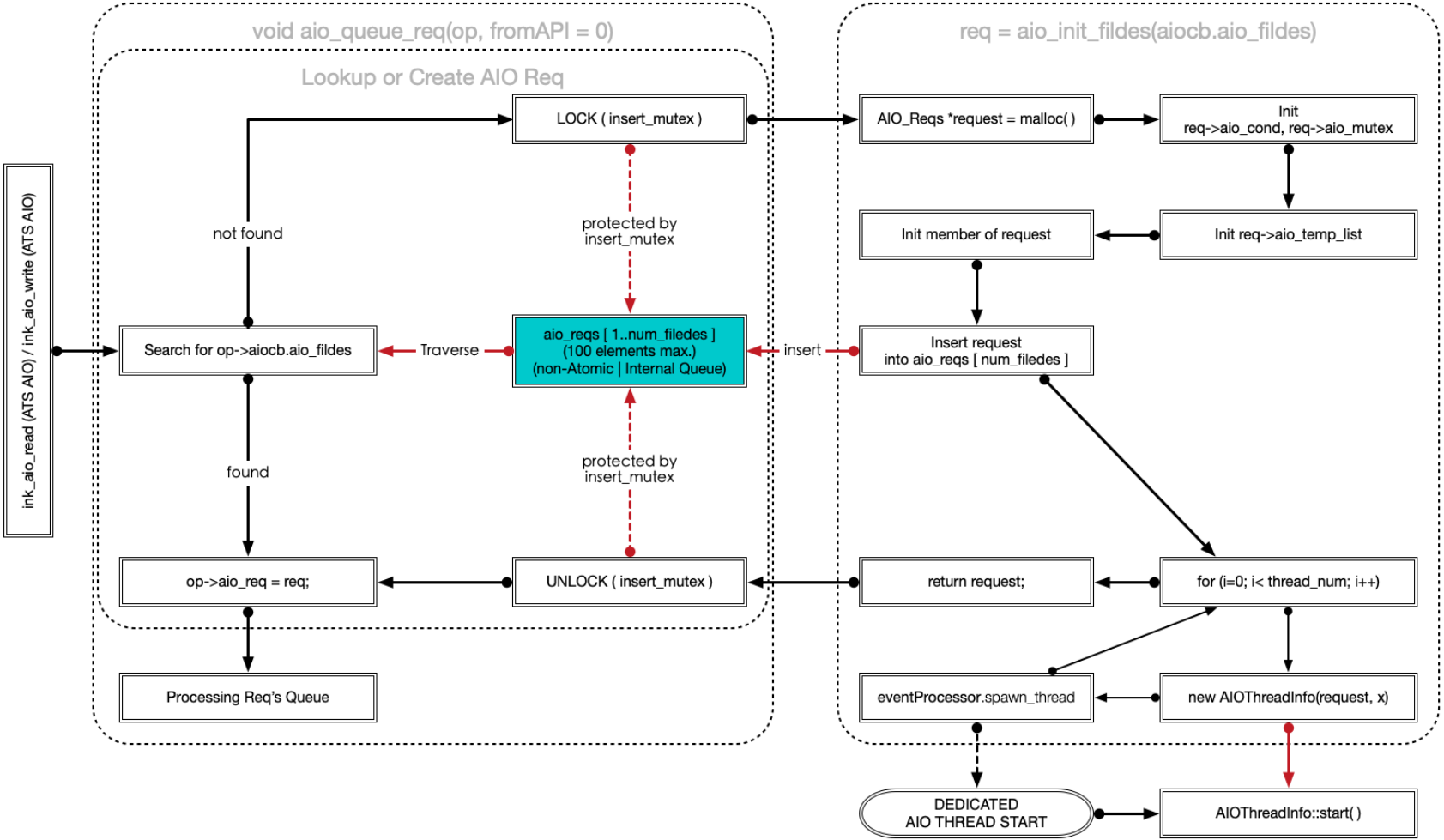
- AIO Sub-system & Native AIO
- DNS Sub-system
- TransformVConnection

AIO Sub-system

AIO Sub-system

- AIO_Reqs per file description
  - AIO_Reqs *aio_reqs[MAX_DISKS_POSSIBLE]
  - MAX_DISKS_POSSIBLE = 100
  - aio_reqs[ ] is protected by `insert_mutex'
- An AIO_Reqs has
  - 8 AIO threads: proxy.config.cache.threads_per_disk
  - An atomic queue: aio_temp_list
  - A sort (by priority) queue: aio_todo, http_aio_todo (p == 0)
    - Protected by aio_mutex and aio_cond

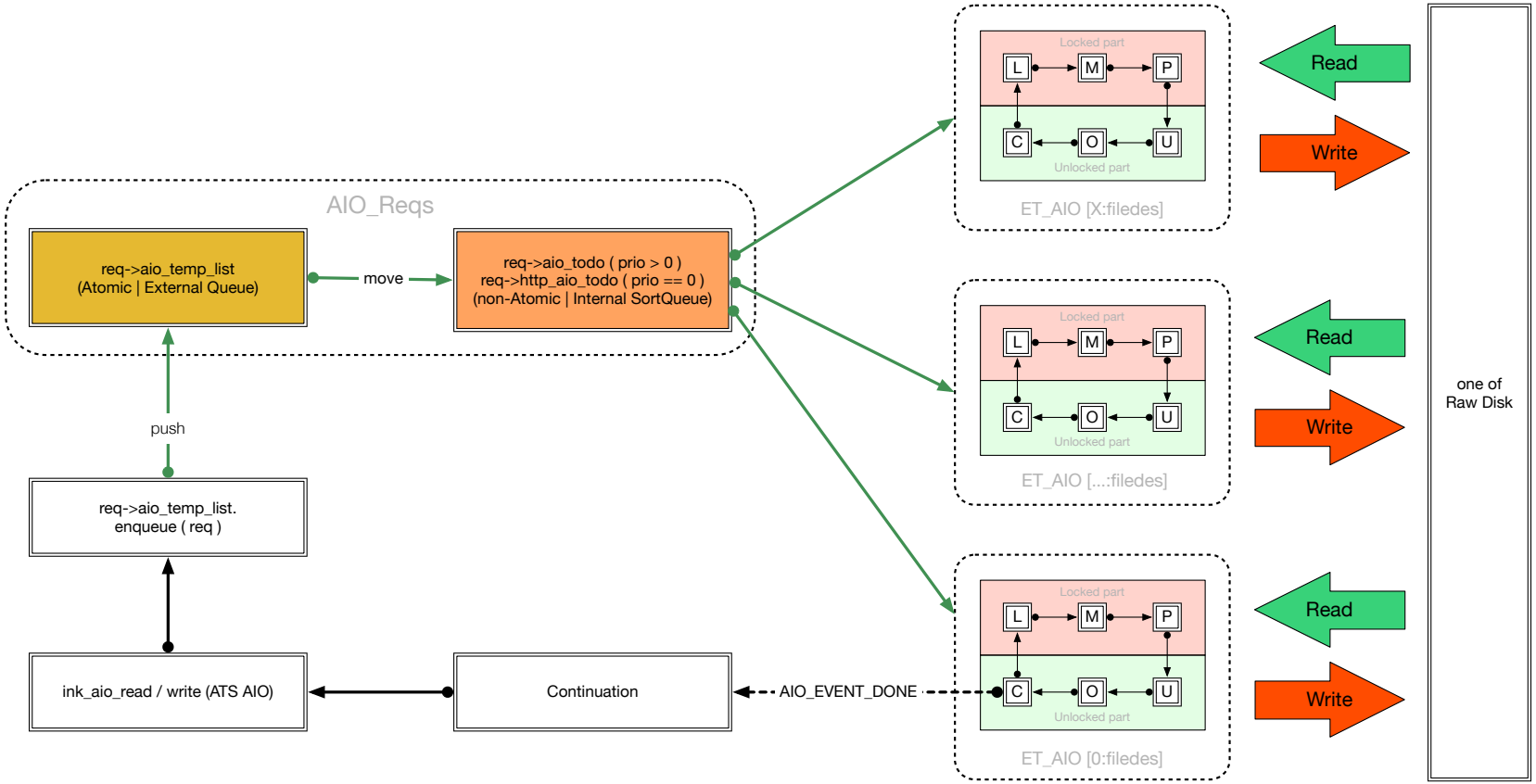
AIO Tasks Queue Lookup, Create and Insert



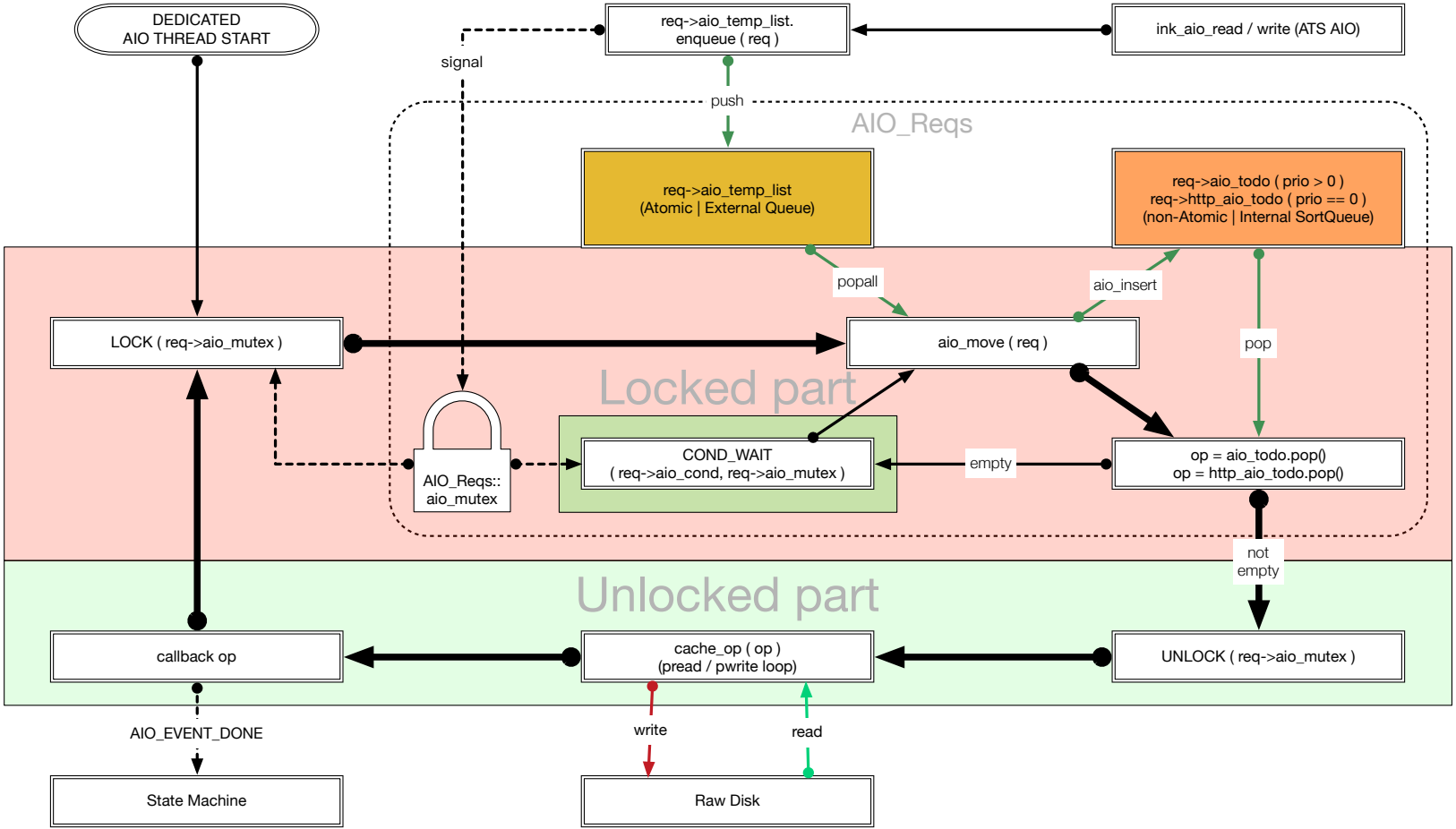
AIO Sub-system (cont.)

- The AIO loop is blocked by one of the following two operations
  - Disk I/O operations
  - Conditional wait
- Therefore, each AIO loop consumes only one I/O task from the queue.
- Create multiple AIO Threads to support concurrent I/O operation on specify file description (block device)

AIO Task Queue and AIO Thread Group



AIO Thread Loops

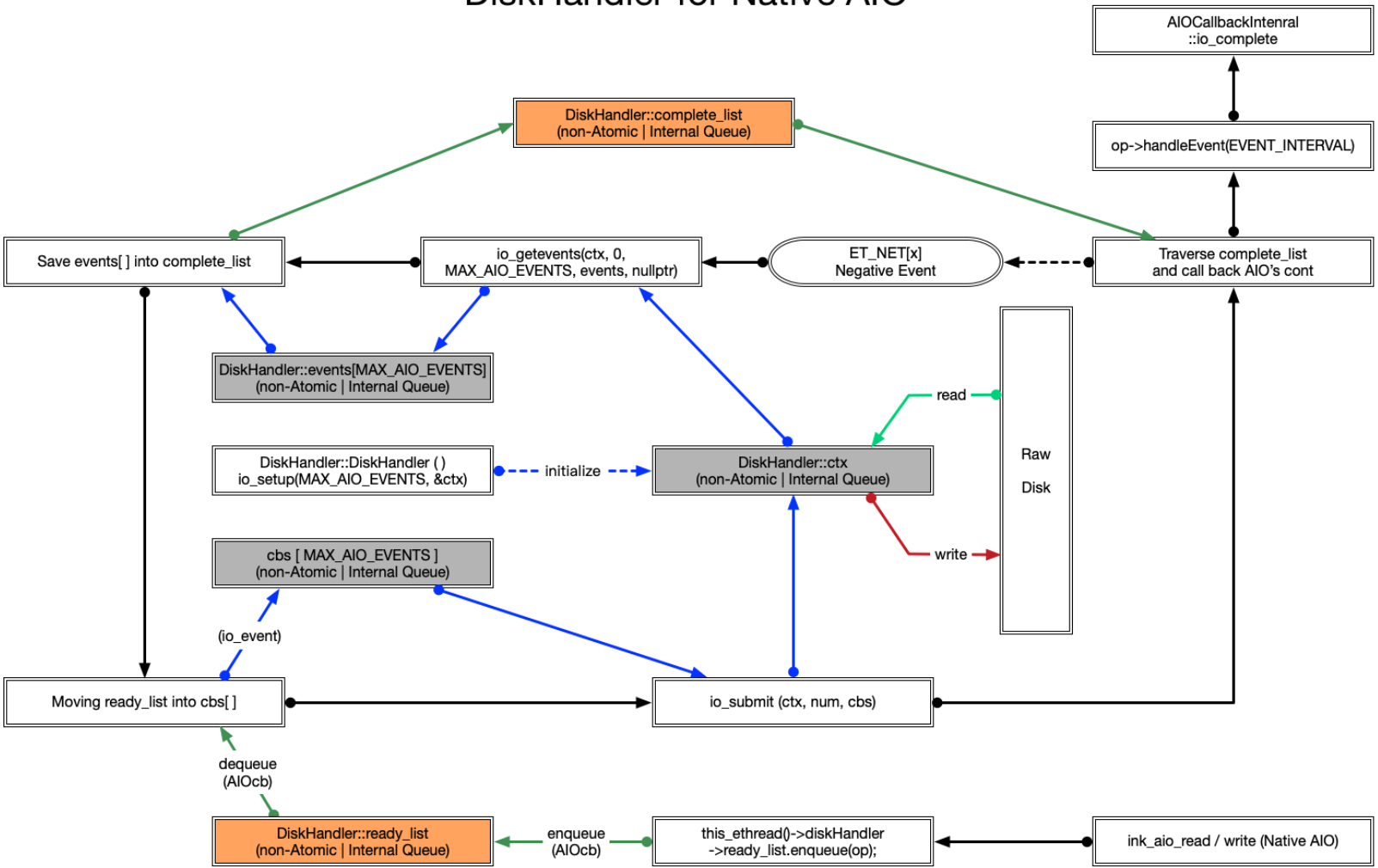


Native AIO

Native AIO

- Native AIO is similar to epoll system
  - io_setup() VS epoll_create()
  - io_getevents() VS epoll_wait()
  - io_submit() VS epoll_ctl()
  - DiskHandler VS NetHandler

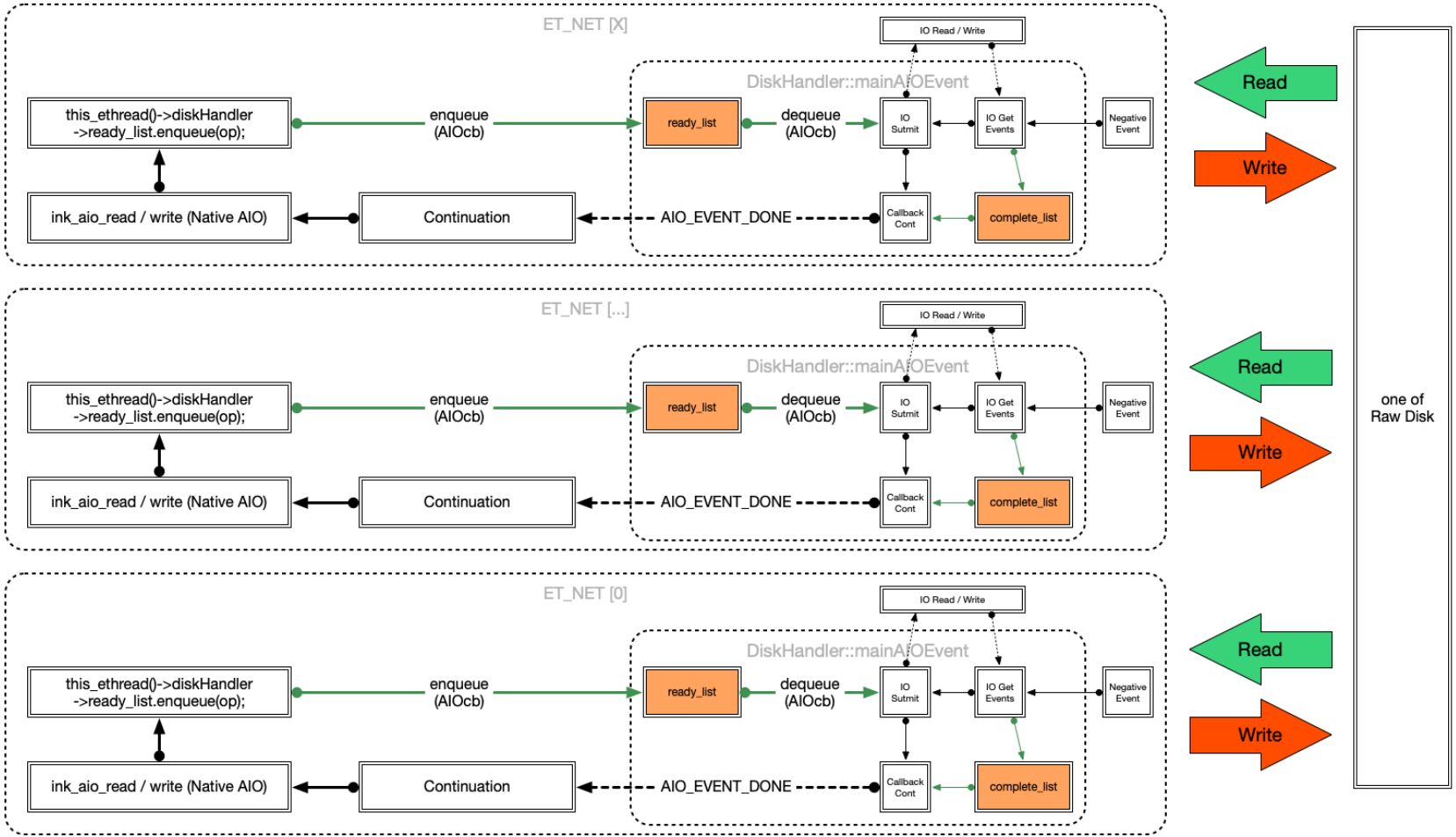
DiskHandler for Native AIO



Native AIO (cont.)

- Similar to NetHandler, DiskHandler is also in every ET_NET threads
  - Different from NetHandler, DiskHandler bundle to its EThread.
  - It shares mutex with its EThread.
- I/O tasks queue : DiskHandler::ready_list
  - EThread local queue
  - Access from current EThread only
- The level of concurrent I/O operations is controlled internally by Native AIO.

Native AIO Task Queue and DiskHandler



DNS Sub-system

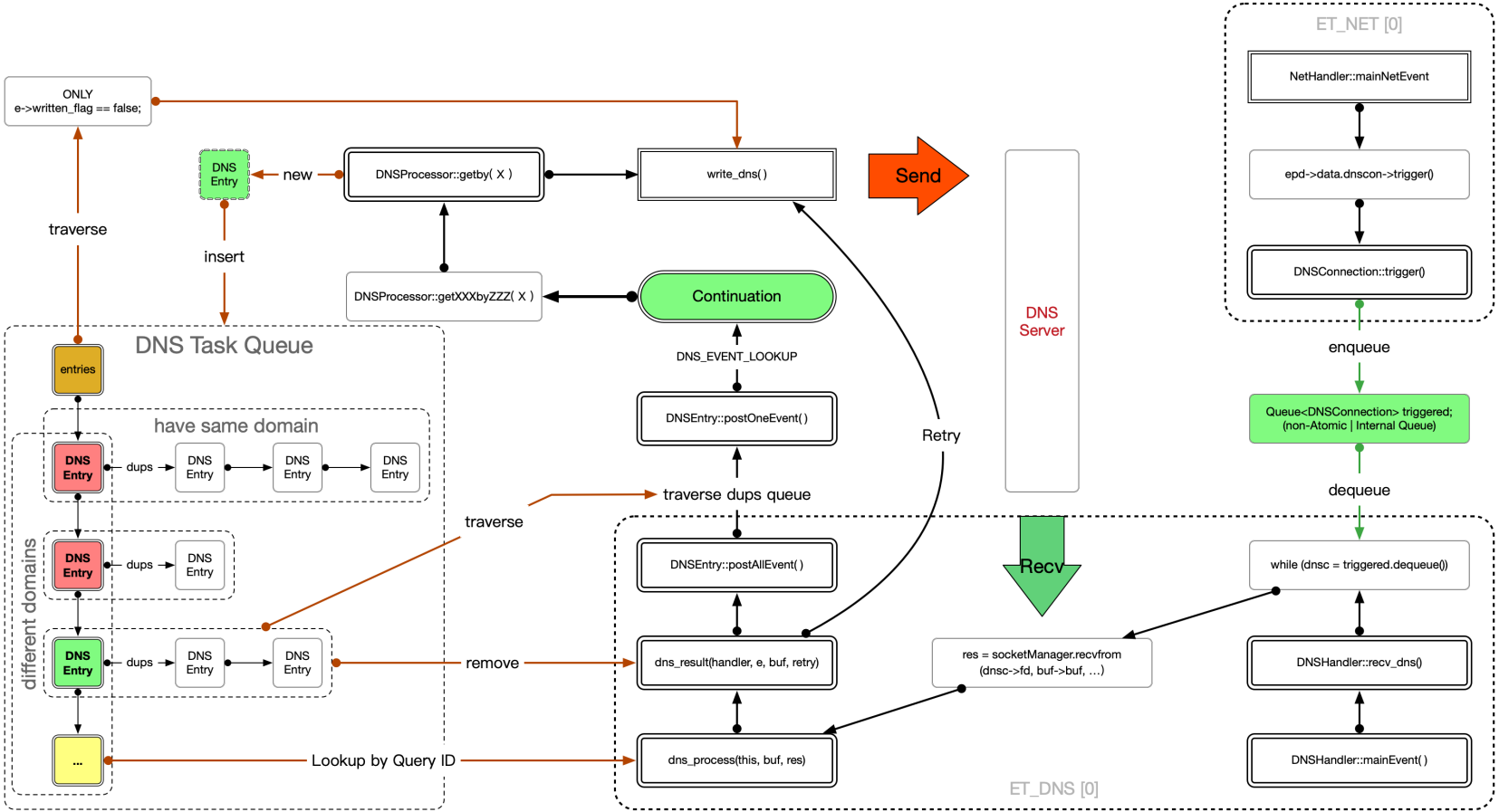
DNS Sub-system

- DNS Task Queue : DNSHandler::entries
  - Pending Entry
    - New queries / retry queries
    - Create an unique Query ID for each DNS request
    - Once DNS requests send out, It will be set to “In flight”
    - Lookup them by domain name
  - In flight Entry
    - DNS requests without response
    - Lookup by domain name / Query ID

DNS Sub-system (cont.)

- Duplicate Queue : DNSEntry::dups
  - Share DNS results for tasks which lookup for the same domain name.
  - Save duplicate tasks.
  - Traverse dups queue and call back continuation one by one.

DNS Task Queue and DNSHandler



DNS Sub-system (cont.)

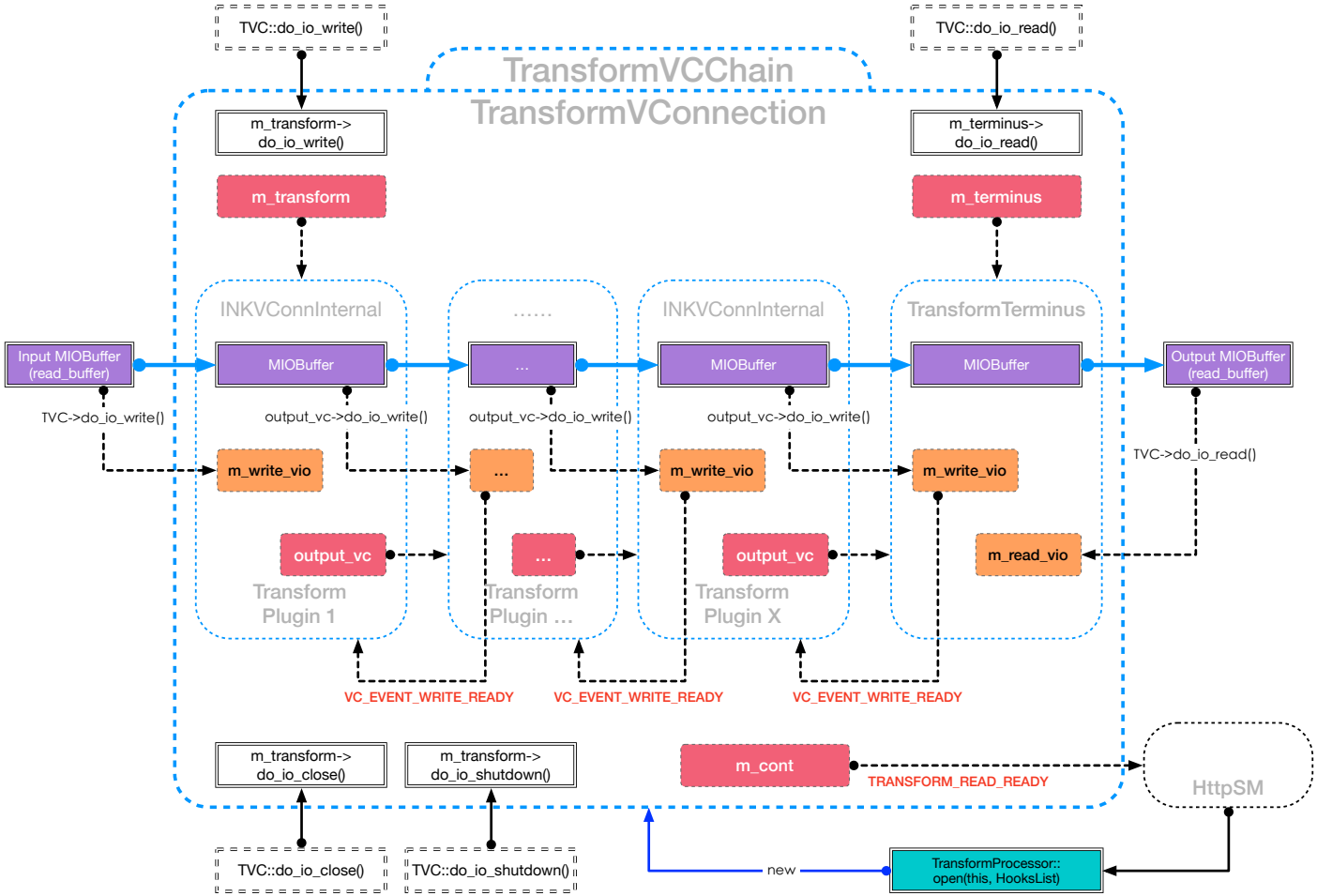
- ET_DNS Thread Group
  - Only one EThread: ET_DNS[0]
  - 2 Key components: DNSHandler and NetHandler
  - If ET_DNS[0] shares EThread with ET_NET[0]
    - There is DNSHandler in ET_NET[0]
  - DNSHandler bundle to ET_DNS[0]
    - It shares mutex with EThread.
- Polling on DNSConnection (UDP socket fd)
  - Only on ET_DNS[0] or ET_NET[0]
  - DNSConnection con[MAX_NAMED], MAX_NAMED = 32

TransformVConnection

TransformVConnection

- What is TransformVConnection ?
  - TVC is a pipe/chain that is connected by one or more INKVConnInternals.
- TVC is a unidirectional pipe
  - The 1st INKVConnInternal is the input
  - The TransformTerminus is always attached to the tail as the output
  - When the TransformTerminus received any data from its upstream, it will send TRANSFORM_READ_READY event to the owner of TVC.

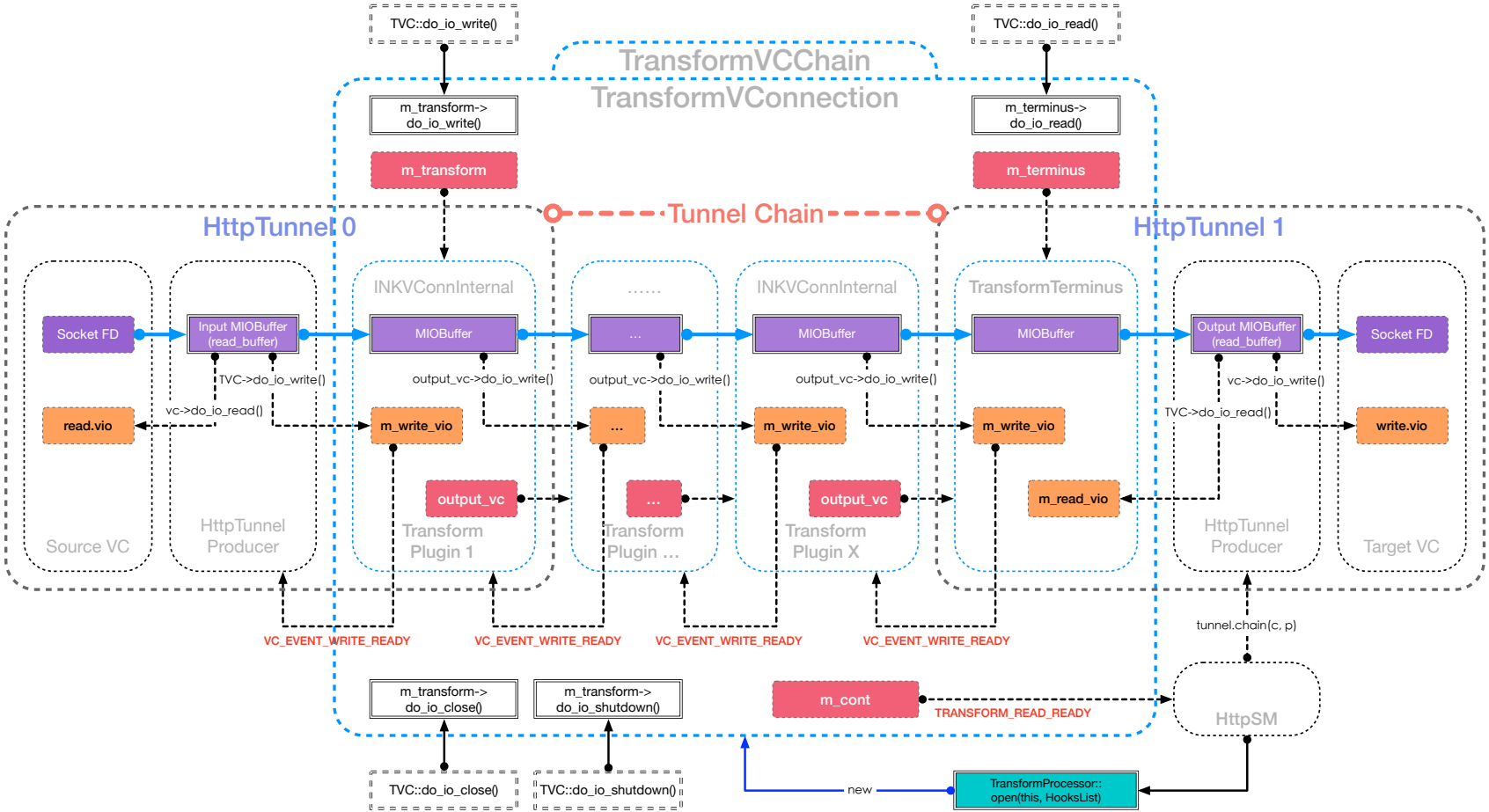
TransformVConnection



TransformVC and Tunnel Chain

- TVC as both a consumer and a producer, connecting two tunnels
- The two tunnels are chained in order to drive the data stream from TVC's input to output
- It is a complete pipe from the source VC to the target VC (for example: from client VC to server VC).

TransformVConnection and Tunnel Chain



Thanks