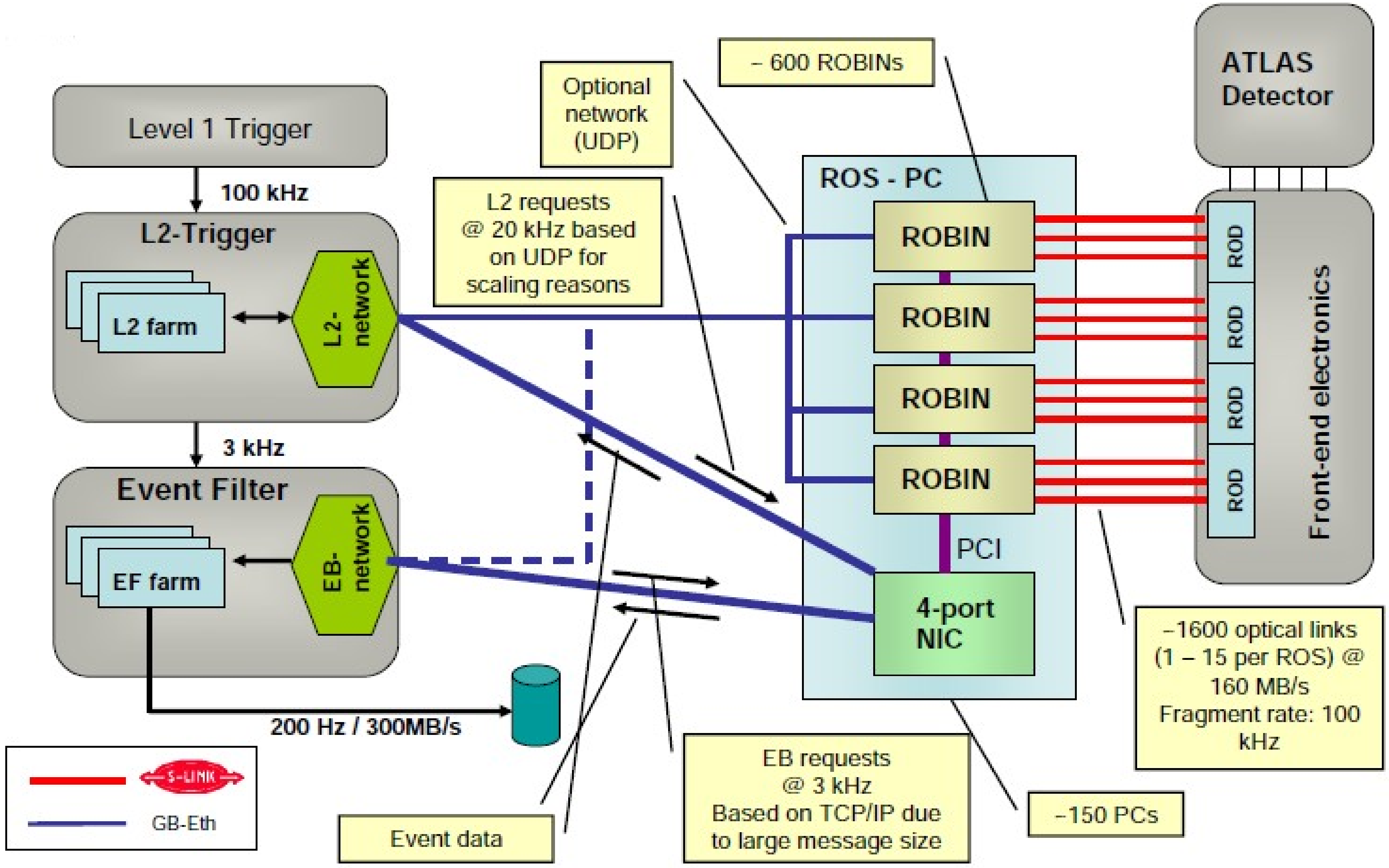
ATLAS DAQ and FTK

Preliminary remarks:
- This is my current personal understanding
- I'm not a ROS expert

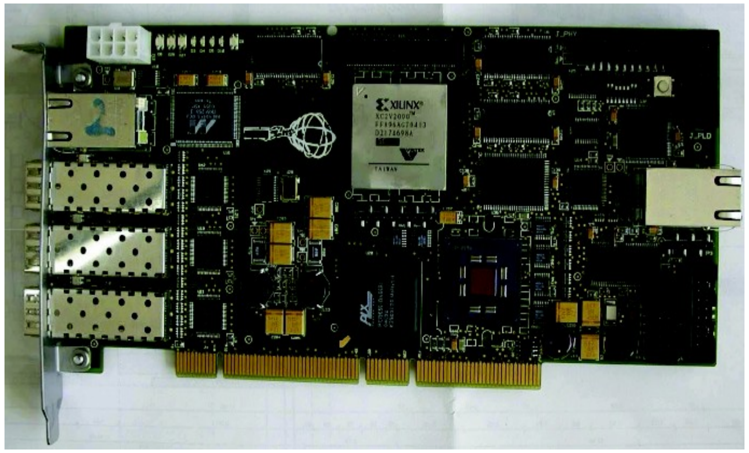
Outline:
- ATLAS T/DAQ overview
- ROS system & FTK
- DF evolution and FTK

ATLAS T/DAQ overview



ROS system (1)

- ROS PC:
  - 4 Robins, each one with 3 input S-links and 3 buffers (ROBs)
  - Max input b/w:
    - ~ 480 MB/s per ROBIN
    - ~ 2 GB/s per ROS
  - Output: 2 Gb links, ~ 240 MB/s

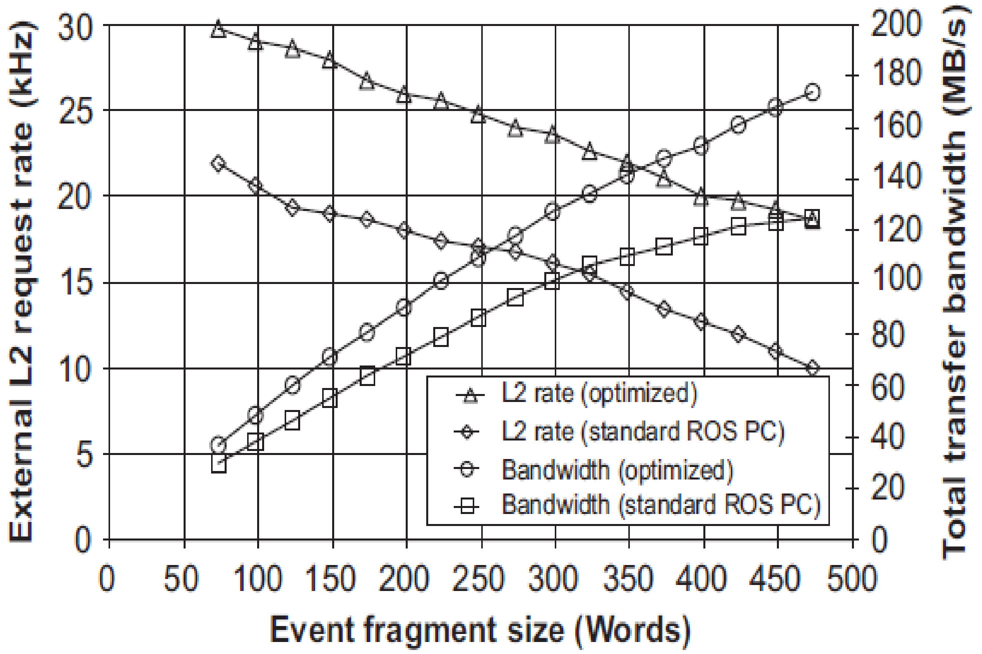


- ROB size: 64 MB, paged
  - The ROB with biggest page size sets upper limit on number of events that can be inside the system (from ROS to EB) at each moment
  - I.e.:
    - If fragment size = 4 kB $\rightarrow$ 16000 events (OK)
    - If fragment size = 16 kB $\rightarrow$ 4000 events (critical)
  - Current Robin PCI based;
    a prototype PCI express exists, but the ROB size is still 64 MB

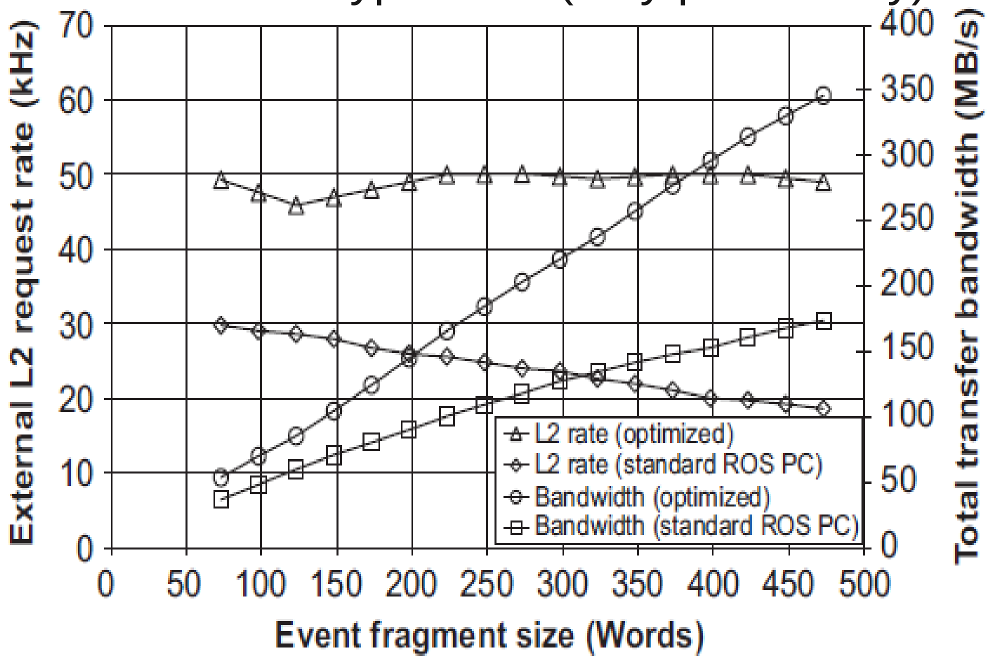
ROS system (2)

- Design L2 request rate: ~ 20 kHz
  - And it does not scale with size

Current H/W

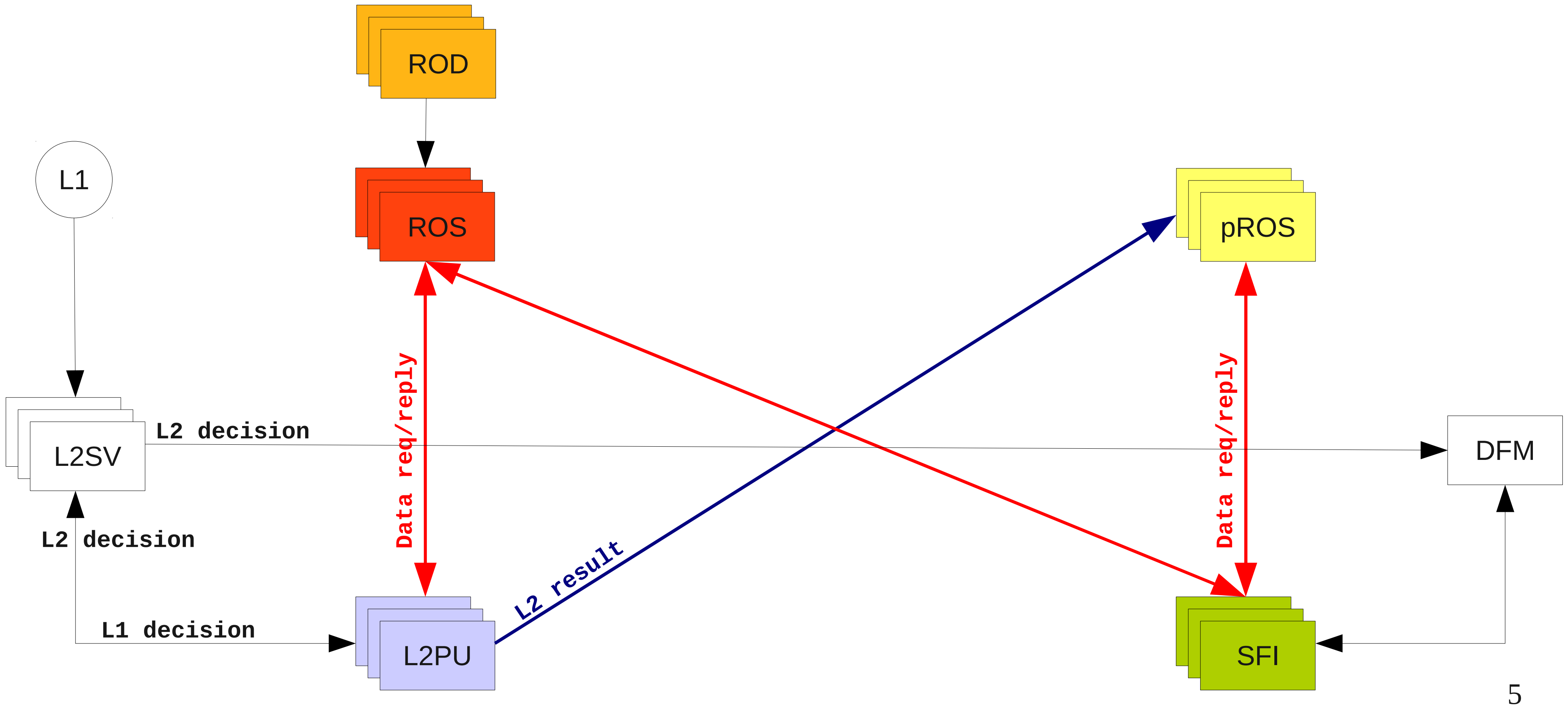


Prototype ROS (very preliminary)



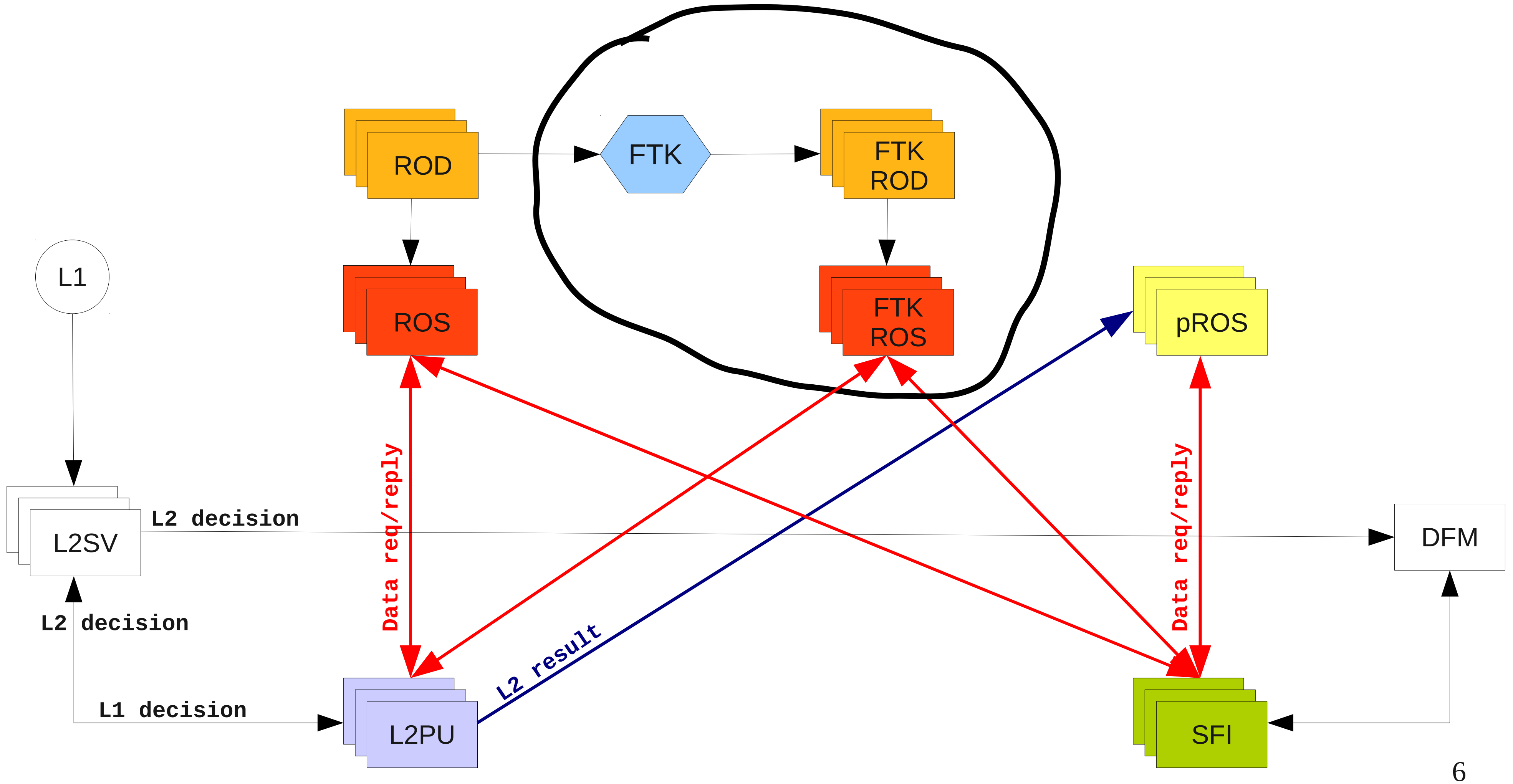
ROS system for FTK (1)

- First approach:
  use “standard” ROD-ROB-ROS chain for buffering FTK output



ROS system for FTK (1)

- First approach:
  use “standard” ROD-ROB-ROS chain for buffering FTK output

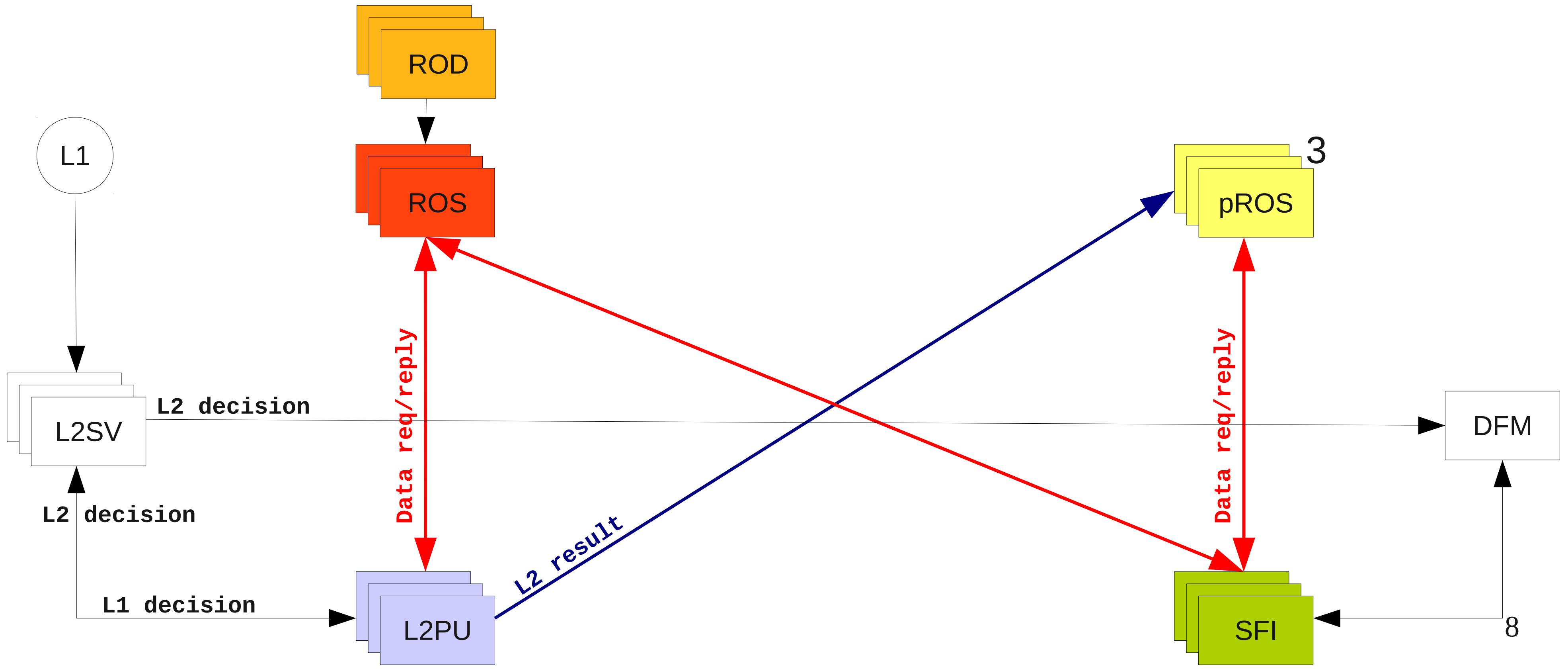


ROS system for FTK (2)

- FTK Event size from 2.5 kB to 15 kB (@75 m-bias events), while the usual ATLAS ROB page size is ~ 1-2 kB
  - N.B.: a page size of 15 kB limits the maximum number of events in the pre-EB system to ~4000 (too little, considering fluctuations)
  - $\Rightarrow$ use a round robin approach to increase the “effective depth”
- The L2 request rate to FTK ROS will be ~ 100 kHz, while the current ROS system was designed for 20 kHz
  - Some prototypes reached 50 kHz
  - $\Rightarrow$ use a round robin approach to reach 100 kHz
- So, use multiple ROSs to be activated according to the L1ID
  - E.g.: ROS-3 used if L1ID%3==0
- If EvSize = 2.5 kB, @ 100 kHz $\rightarrow$ 250 MB/s
  - One ROBIN would be enough (the ROS needs 4 Gb nics)
- If EvSize = 15kB, @ 100 kHz $\rightarrow$ 1.5 GB/s (12 Gb/s)
  - ~ a dozen of Gb/s NICs

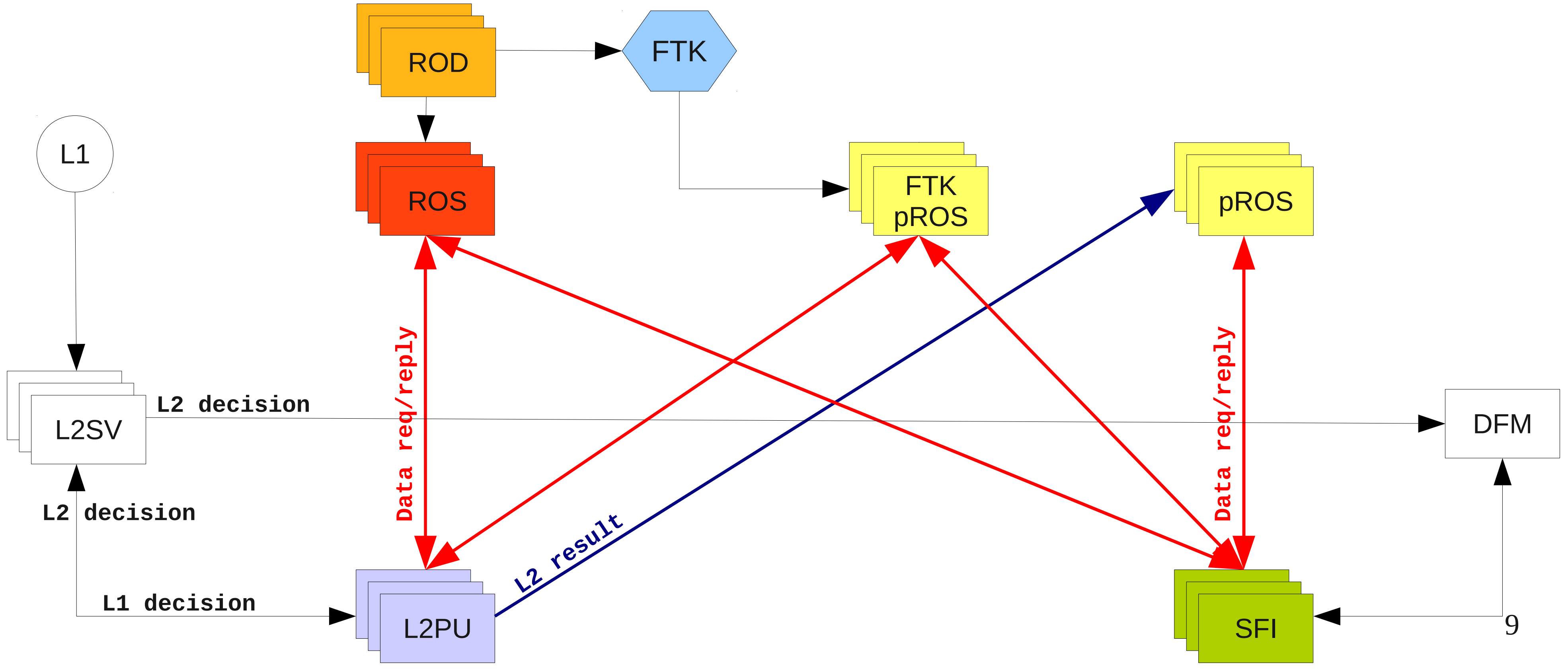
What about pROS (aka L2RH)?

- In the current system L2 results are stored in few “pseudo” ROSs
  - L2 results sent by L2PUs via ethernet
  - EB is collecting fragments from both “real” ROSs and “pseudo” ROSs
- pROS PC: max Rate: ~ 4 kHz
  - It handles ~2000 input UDP connections and ~ 100 output TCP ones
  - Application needs to be optimized/rewritten to reach ~ 20 kHz



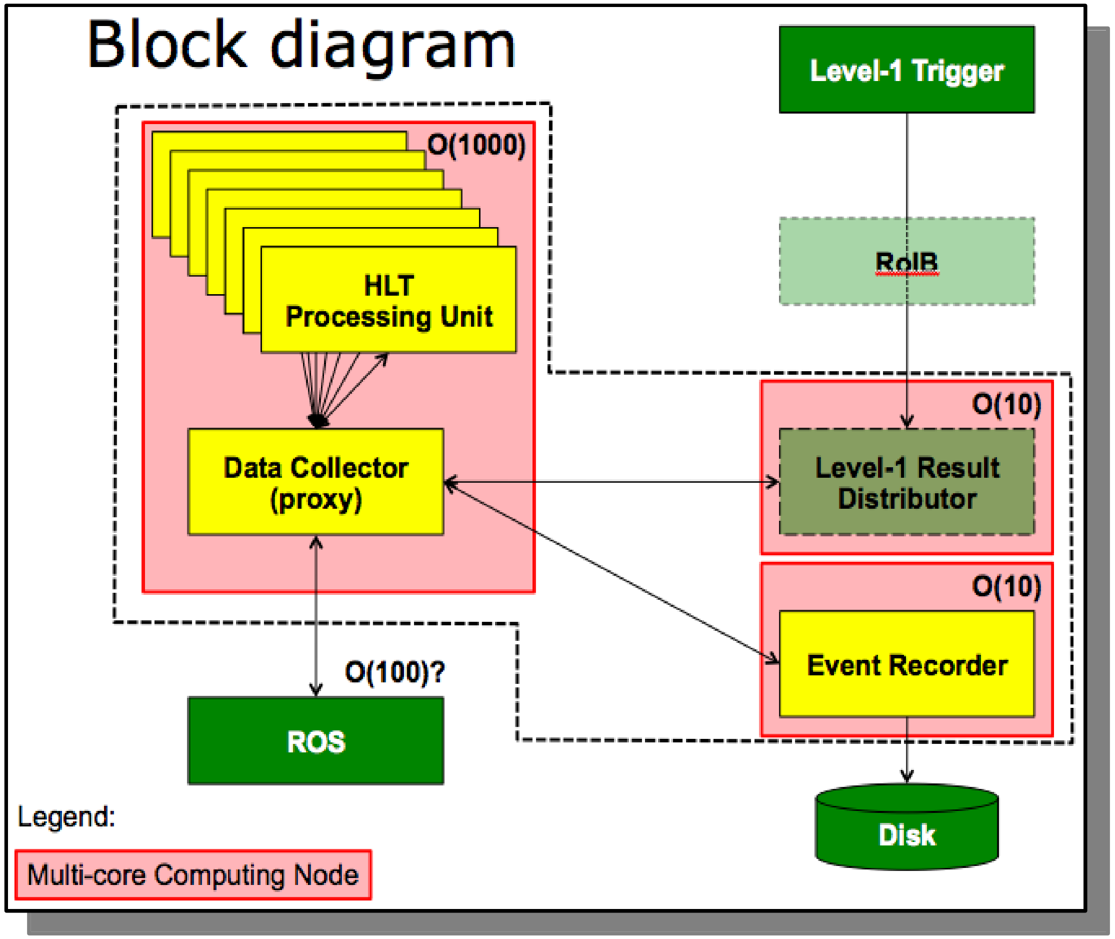
What about pROS?

- The pROS scenario remove buffer size limitations, FTK would
    - Receive data from ID RODs
    - Send output via ethernet (NB: 1.5 GB/s @15 kB event)
    - We just need as much ROS PCs and NICs as the required B/W
- Is it possible to send FTK output over ethernet?
    - Or, use pROS PC with S-LINK cards but w/o ROBINs?



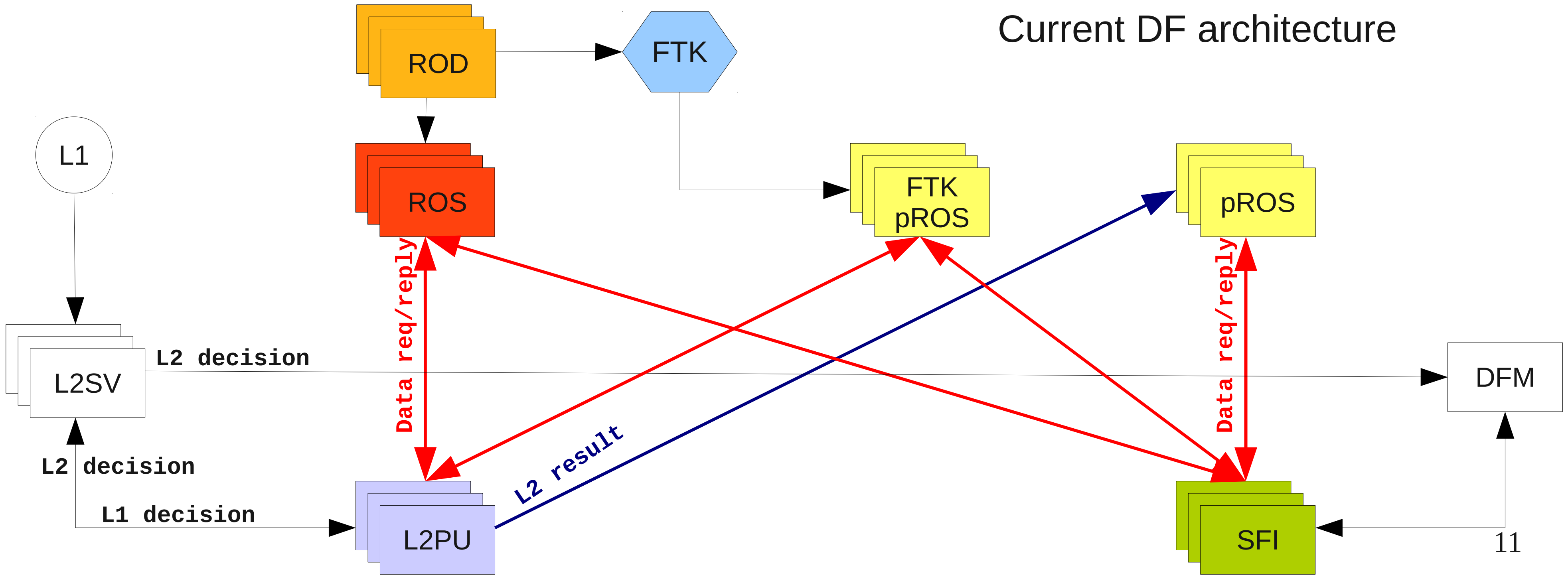
DF evolution

- Timescale: if approved, to be implemented during next shut-down
- A prototype under testing @P1: good results so far



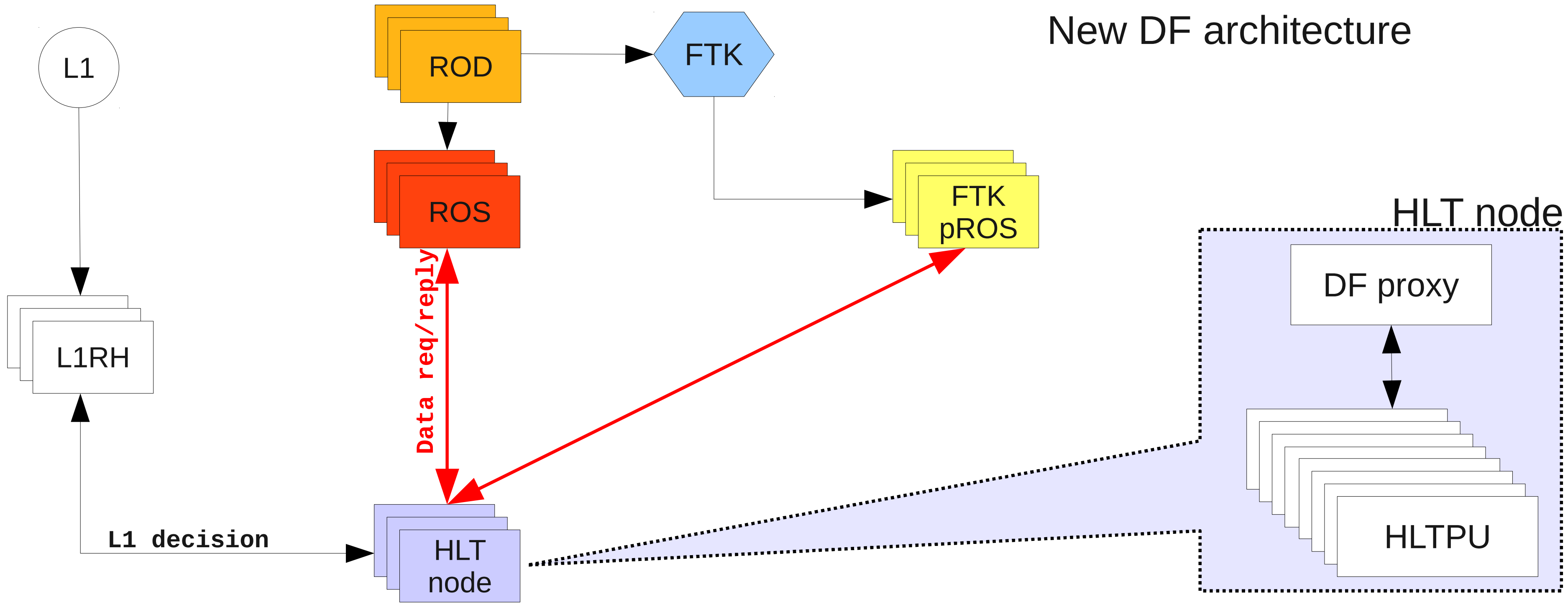
DF evolution

- Design: unify L2, EF and EB farms
  - L2, DC, EB and EF functionalities in each HLT node
  - A single HLT interface
- But keeping RoI concept!
- No changes in the ROS system



DF evolution

- Decoupling data processing from data flow; on each node
  - one process takes care of data movement (DCProxy)
  - multiple processes HLTPU in charge of event selection
- All ROBs only requested once
- Reduced number of network connections (one application per node)
  - But all the HLT nodes will act as L2
- Possibility to implement incremental event building



DF Evolution & FTK

- The new design will not impose limitations to FTK
  - Maybe it could provide additional flexibilities
- The FTK L2 algorithms developed for the current design can be transparently ported to the new one
- FTK fragments requested only once
  - Reduced requests to ROS (~5% for FTK ROS)
- If the FTK fragment is supposed to be accessed for each event (i.e.: @ 100 kHz), one could pre-fetch the FTK data in the DF proxy before waiting for the HLTPU request:
  - Of course, this opportunity depends on the FTK latency
- But the number of events in the DF system <u>before</u> full event building may increase:
  - This would effect the “real” ROS scenario (buffer size and page size), but not the pseudo-ROS one

Conclusions

- The current ROB capacity and the FTK fragment size could reduce the max number of concurrent events inside the DF system
  - Possible solution using redundant ROSs or ad hoc ROBINs
- A “pseudo” ROS approach provides better scalability and avoids the ROB size limitations
  - pROS application shall be improved to support ~ 20 kHz
  - Can FTK send output over ethernet?
- The DF evolution seems to not entail additional limitations to FTK
  - IMO it could offer additional flexibility

Spares

DF prototype and ROB buffering

1. The new design does not entail any additional contribution to the fragment lifetime in ROS queues
   - As in the standard system ROS clears msg are sent (Neglecting for the moment incremental EB strategies)
     - After L2 rejection
     - After full event building (L2 accepted)
   - We could even expect a small reduction of the lifetime

3. In the new design all the cores can act as L2PUs, but this does not increase the number of events in the system: at any moment the number of cores doing L2 task will be roughly the same as now
   - It depends **only** on the average L2 processing time

DF prototype and ROB buffering

Indeed (neglecting EB time) at any moment

- the number of fragments is the RobinQueues
- == number of events in the DC system (i.e. upstream EB)
- == number of cores in the HLT

is a Poissonian distribution with mean

`n = L1Rate x AvgL2ProcTime`

This means that, if AvgL2ProcTime = 40ms (L1Rate@100kH)

- ➔ ~ 4000 fragments in each ROB queues
- ➔ ~ 4000 HLT cores acting as L2PU
  - The other cores are processing events already built