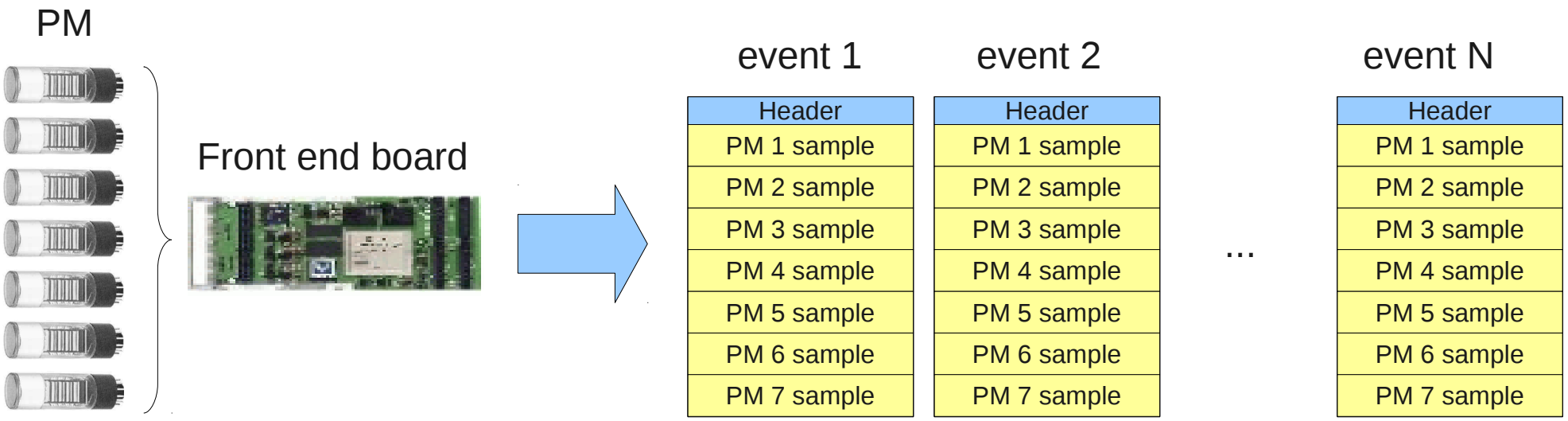
A data acquisition system for the Cerenkov Telescope Array

Julien HOULES, Dirk HOFFMANN, Romain DE LUCA
CPPM/IN2P3/CNRS
houles@cppm.in2p3.fr

And the CPPM CTA group

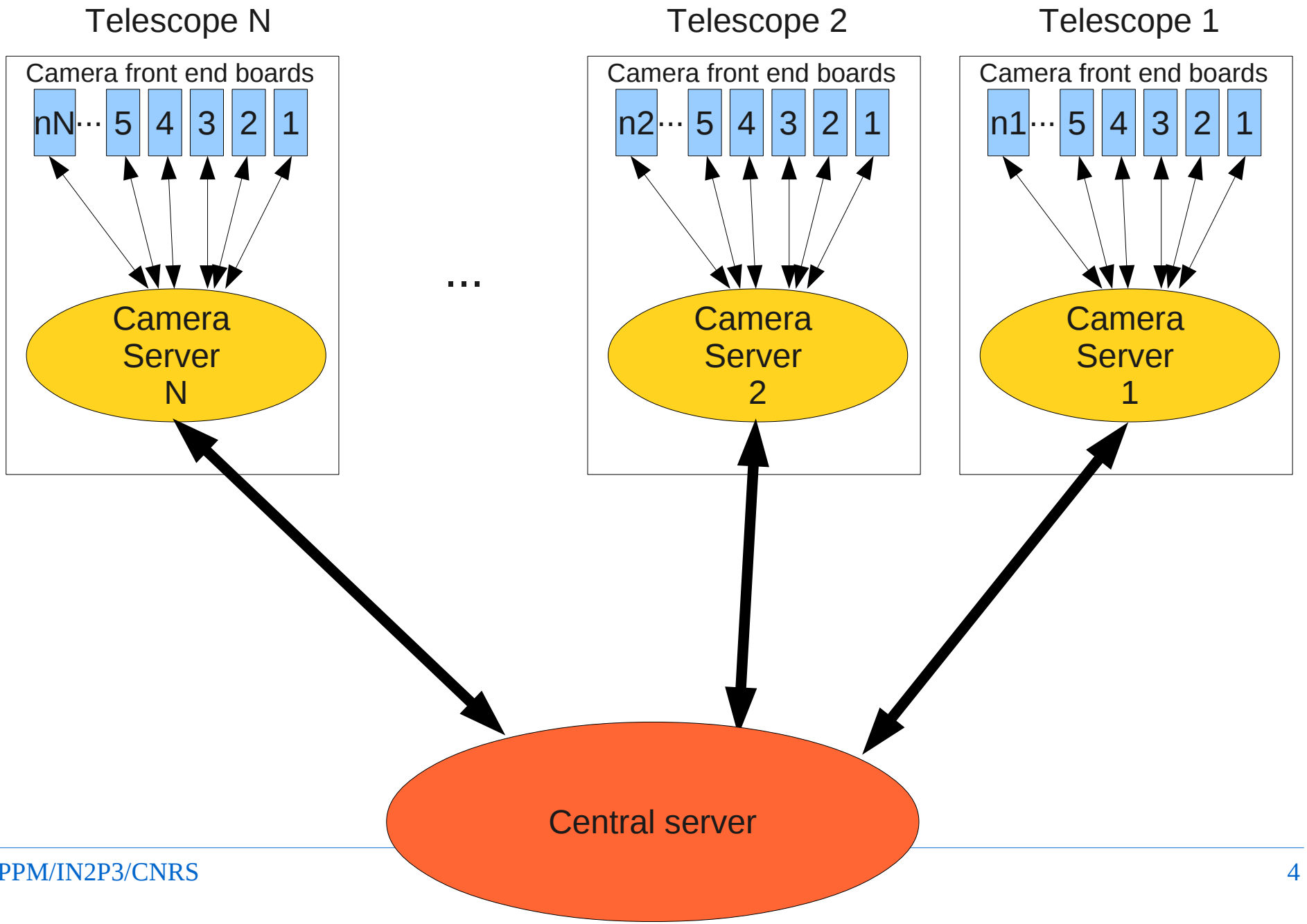
Camera server

Camera data flow



Whole Camera ~ 2000 PM -> 300 front end boards

Global architecture



Camera server

- L1 trigger on front end boards
- Event building
- L2 trigger on camera server :
  - CPU (SSE, AVX...)
  - GPU
- Compression ?
- Send data to central server (array level)

Data flow hypothesis

7 detectors for each front end board

- ~ 2000 pixels camera

→ 300 boards needed

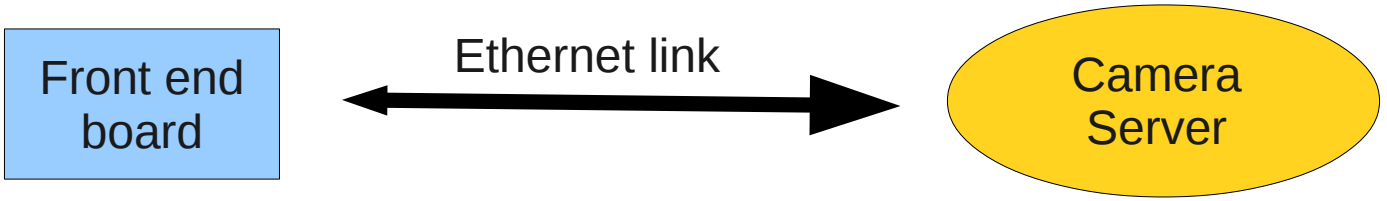
- L1 trigger rate : 10 KHz
- Size of a sample : 144 bytes (16 bit * 72)
- No data loss (all the L1 events are sent)

→ Max theorical bandwidth = 10000 * 2000 * 144 = 2.88 GB/s

Each board generates a flow of 2880/300 = 9.6 MB/s

https://martwiki.in2p3.fr/twiki/bin/view/CTA/DataAcquisition

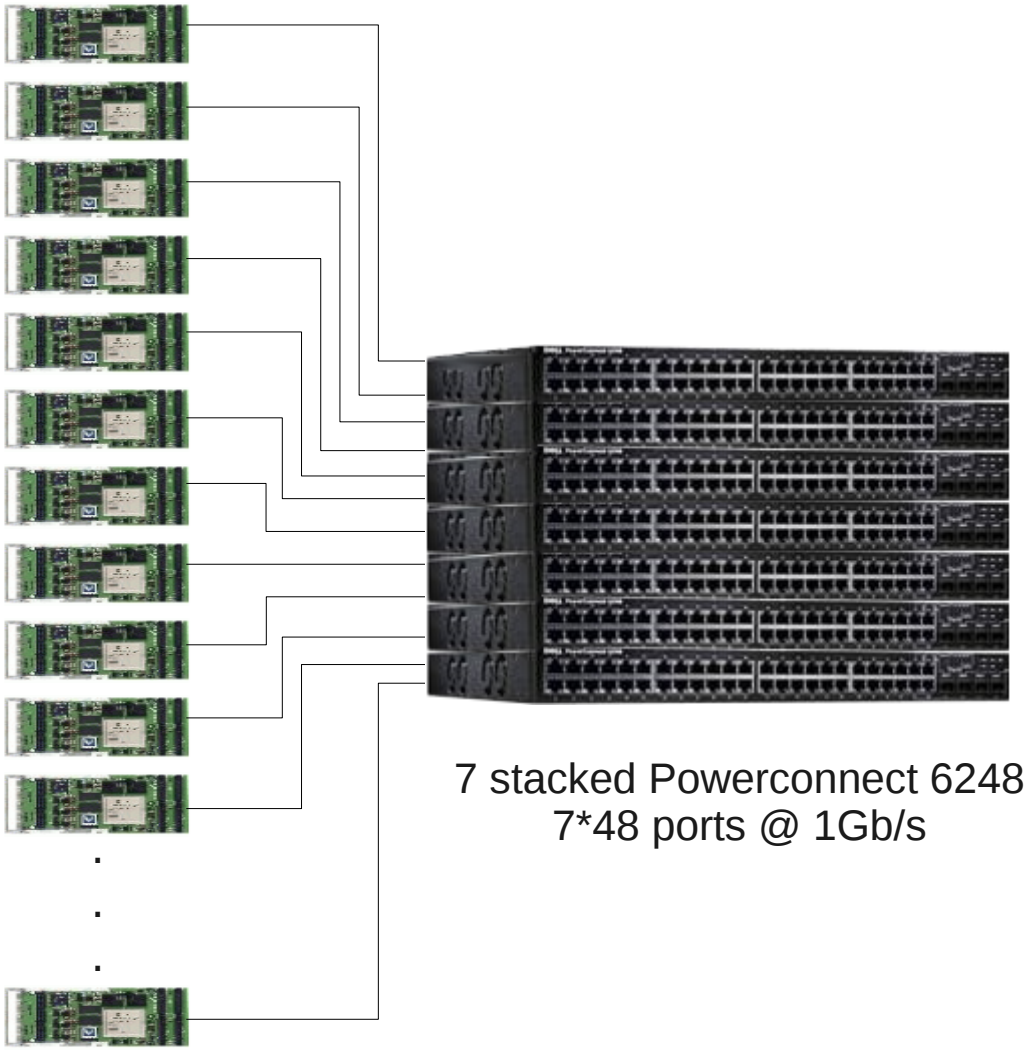
Global architecture



Camera infrastructure

300 * 1Gb/s
Ethernet links

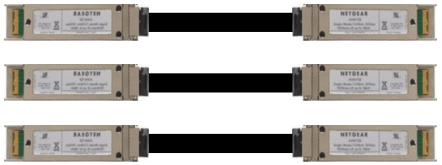
Front end boards



7 stacked Powerconnect 6248
7*48 ports @ 1Gb/s

To central server

3 * 10Gb/s
Ethernet links
SFP+



One or several
Dell T7500 workstations

Dell Precision T7500

- Two Intel **Xeon X5650** (2.66GHz,6.4GT/s,12MB,6Cores)
- Memory : 24GB (6x4GB) 1333MHz

- **Intel X520** DA2 10GbE Dual Port SFP+ Server Adapter, PCIe x8

- Triple channel (maximum speed reached)
- QPI at 6.4 GT/s (maximum speed on the market)
- Memory at 1333 Mhz
- 2 full speed full duplex 10 Gb/s links (PCIe x8 Gen 2)
- 1 PCIe x16 slot free (->GPU)
  and 1 PCIe x8 free (-> one more 10 Gbps adapter)
- SFP+ -> Copper or Optical link

~ 3500 euros

Dell Powerconnect 6248

- 48 * 1 Gb/s ports
- Backplane 184 Gb/s
- 2 * 10 Gb/s SFP+ ports included
  2 more 10 Gb/s optional ports
- Up to 12 switches stackable
  -> 576 ports

~ 1500 euros (with 2 * 10 Gb/s)

Event builder

Why a prototype ?

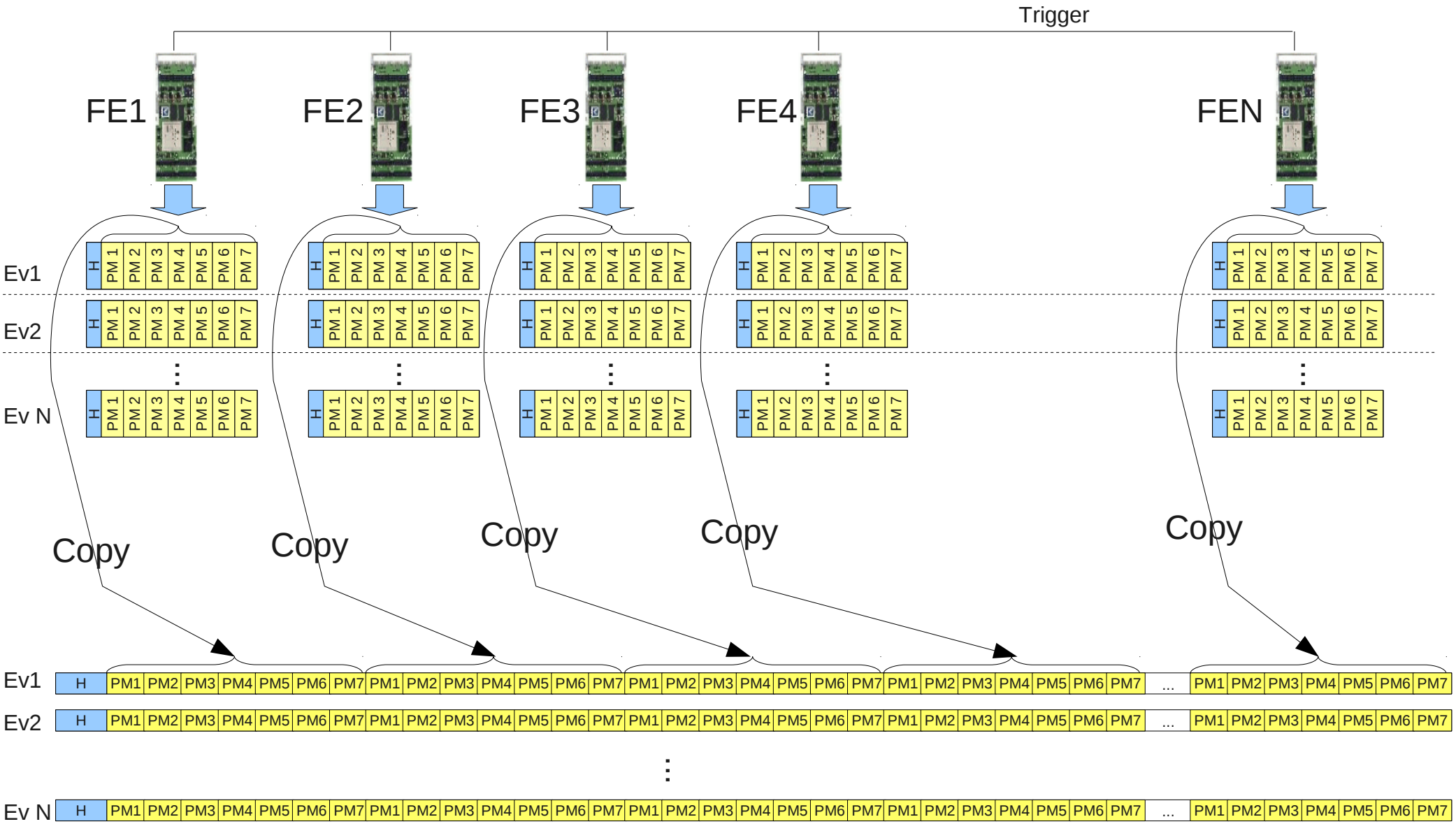
We need a prototype :

- To evaluate the maximum speed reachable
- To test several technologies
- To validate different approaches of the data processing
- To adapt our needs to what we can do

Our first approach

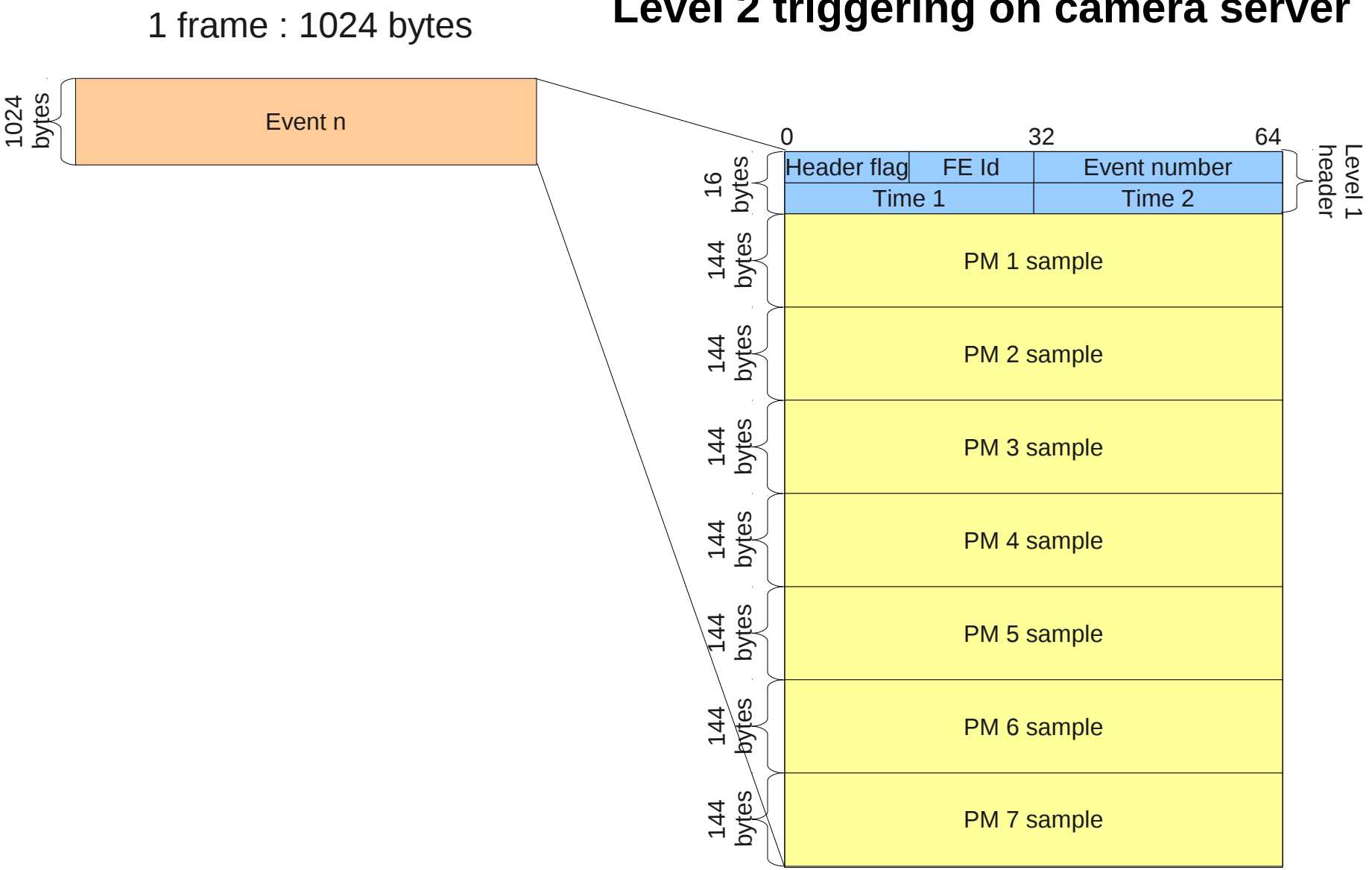
- High modularity to make adaptation to different front end electronics easier
- Multitask approach to divide the flow processing if needed
- Use of a standard Linux distribution but take control on scheduling and memory allocation
- Constrained electronics to reach the best performances (in a first time)

Event builder

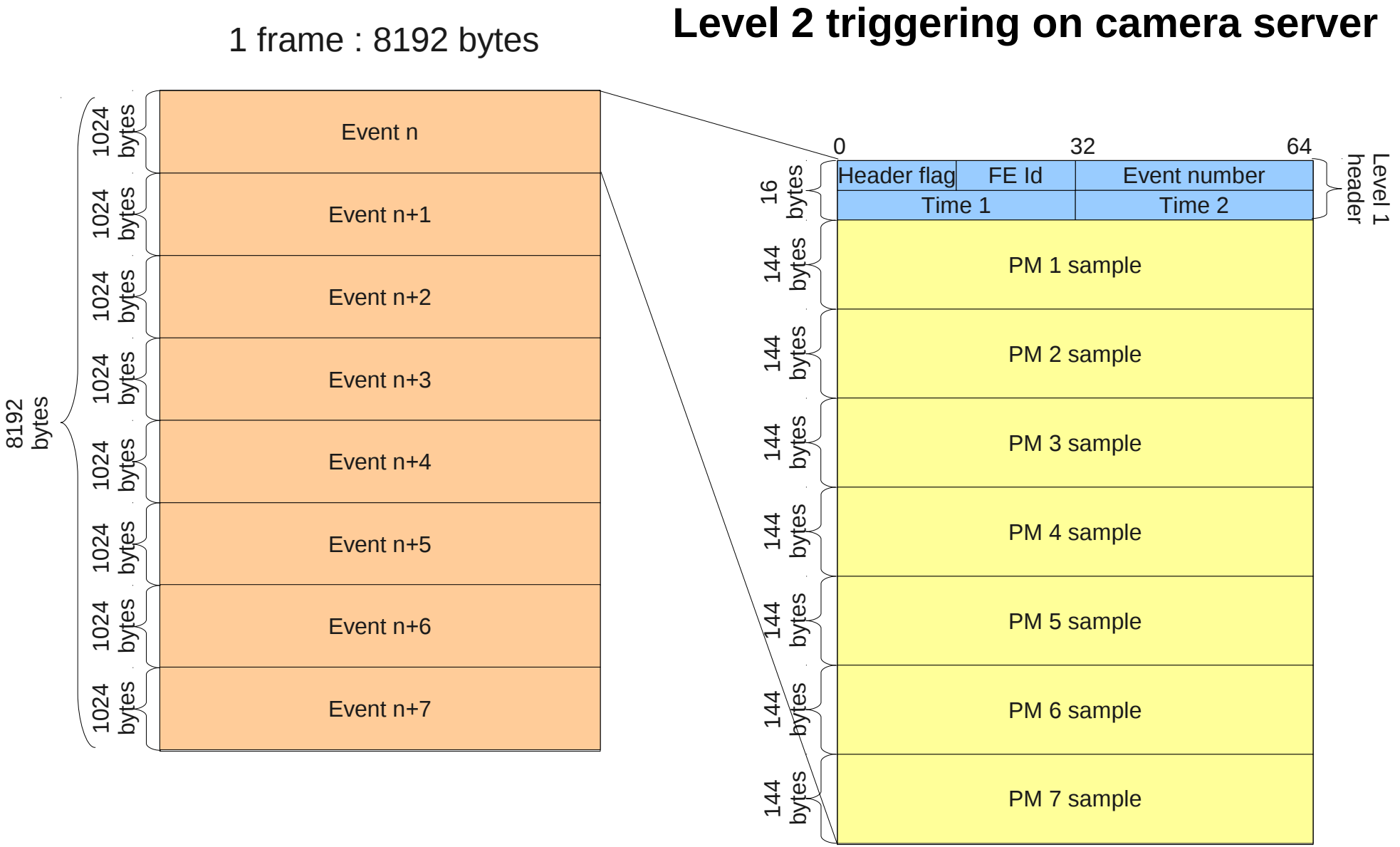


Data format : standard frame

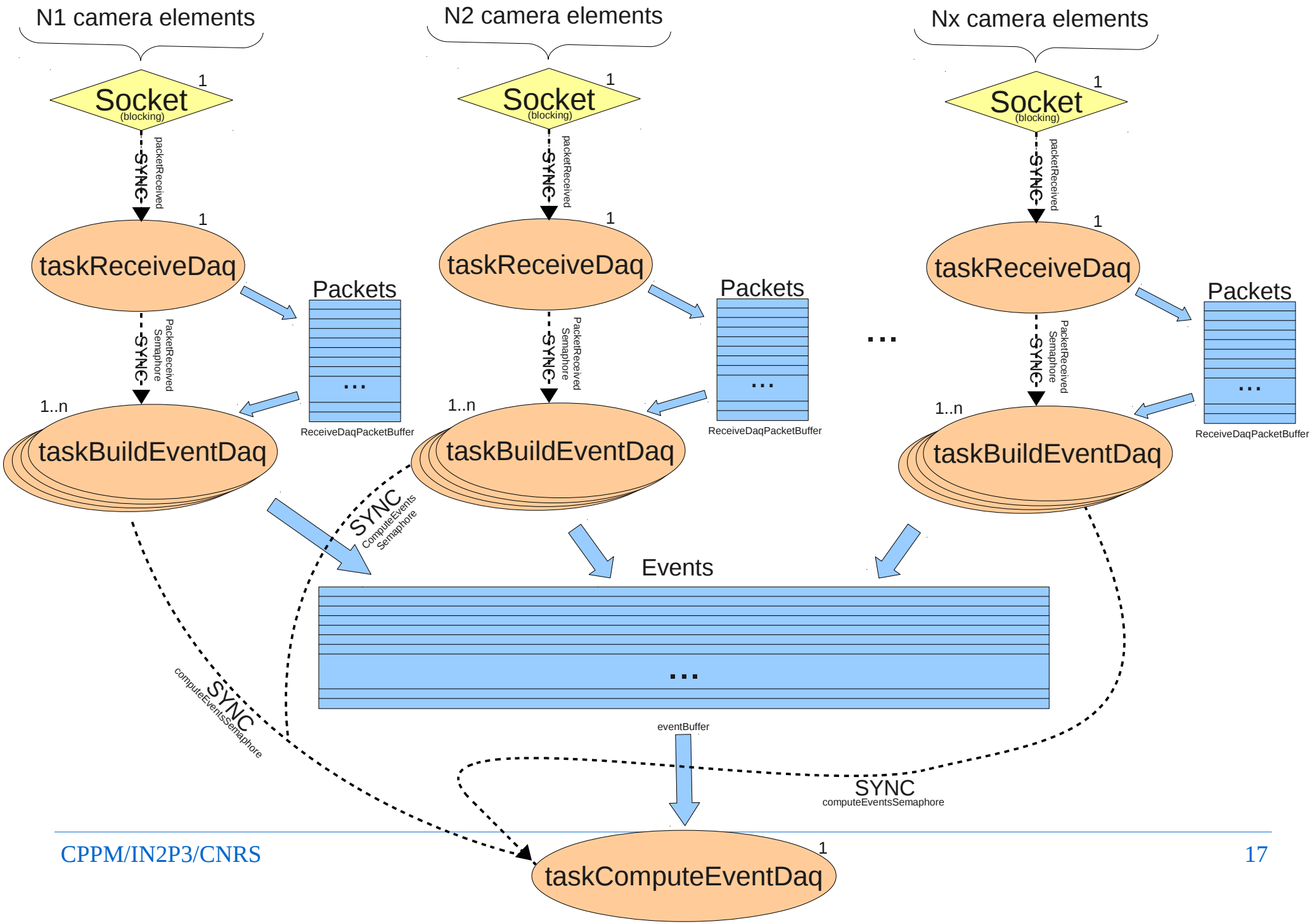
Level 2 triggering on camera server



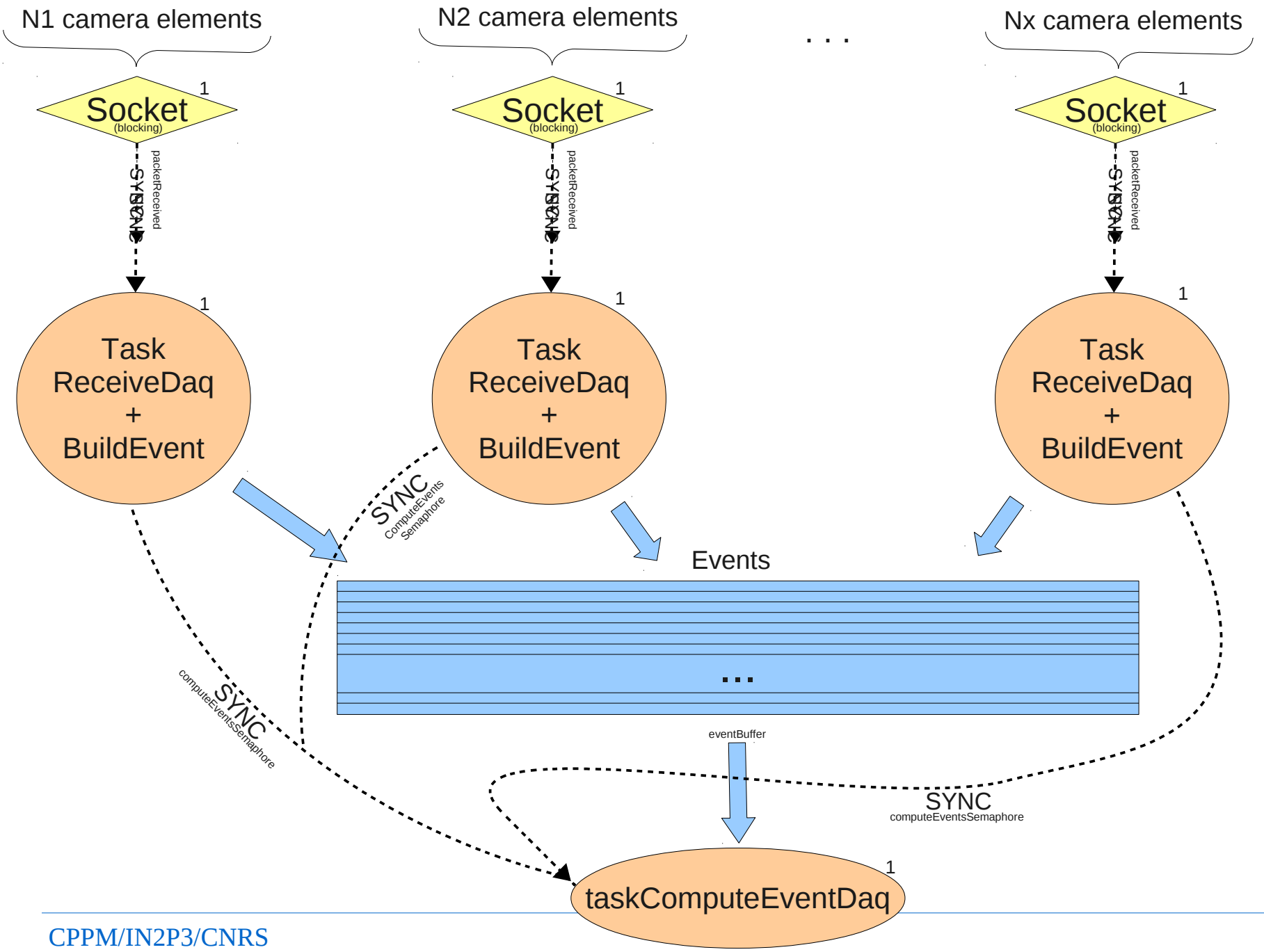
Data format : jumbo frame



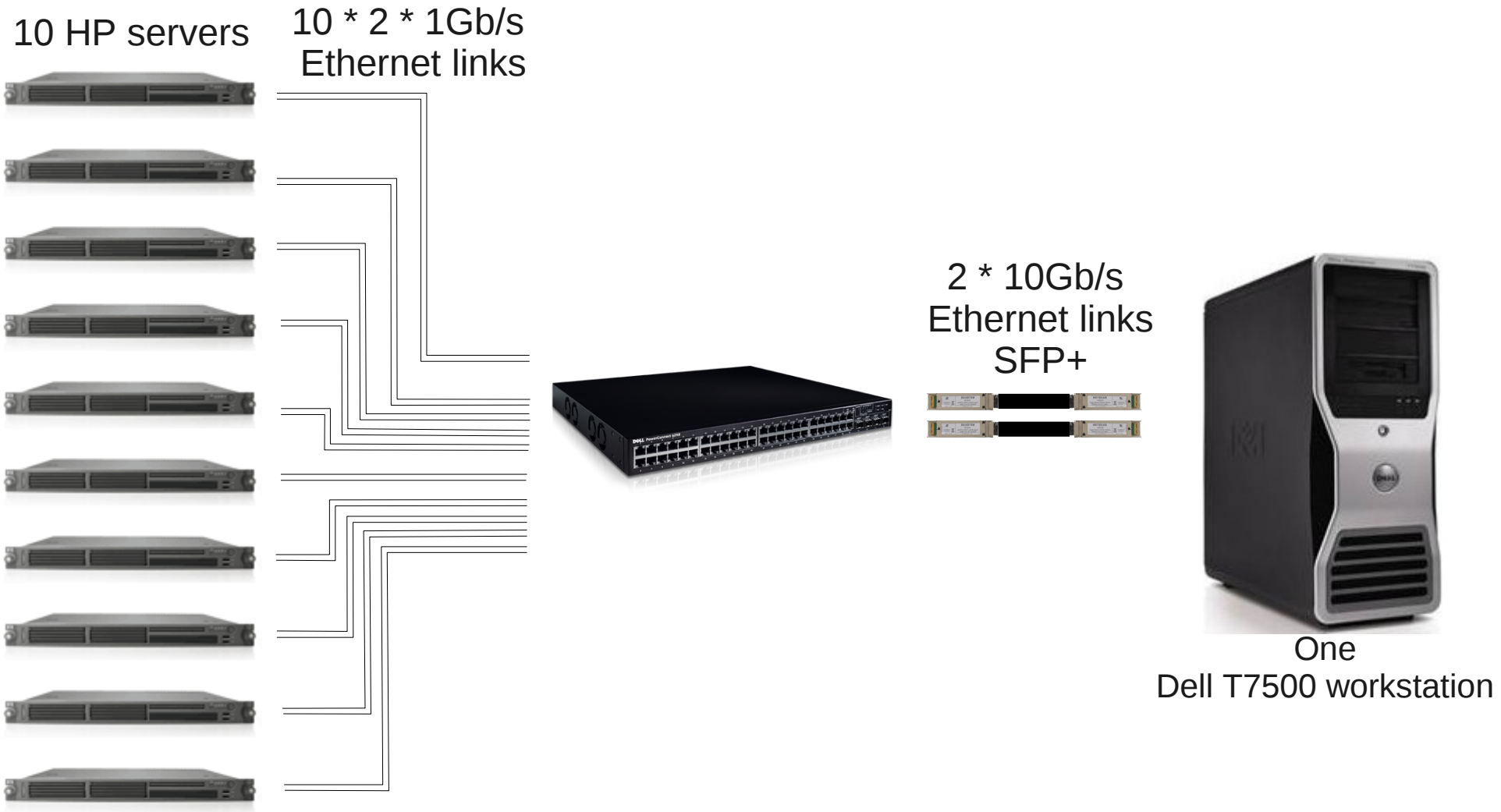
Software overview : 1st architecture



Software overview : 2nd architecture



Stimulation configuration



First results

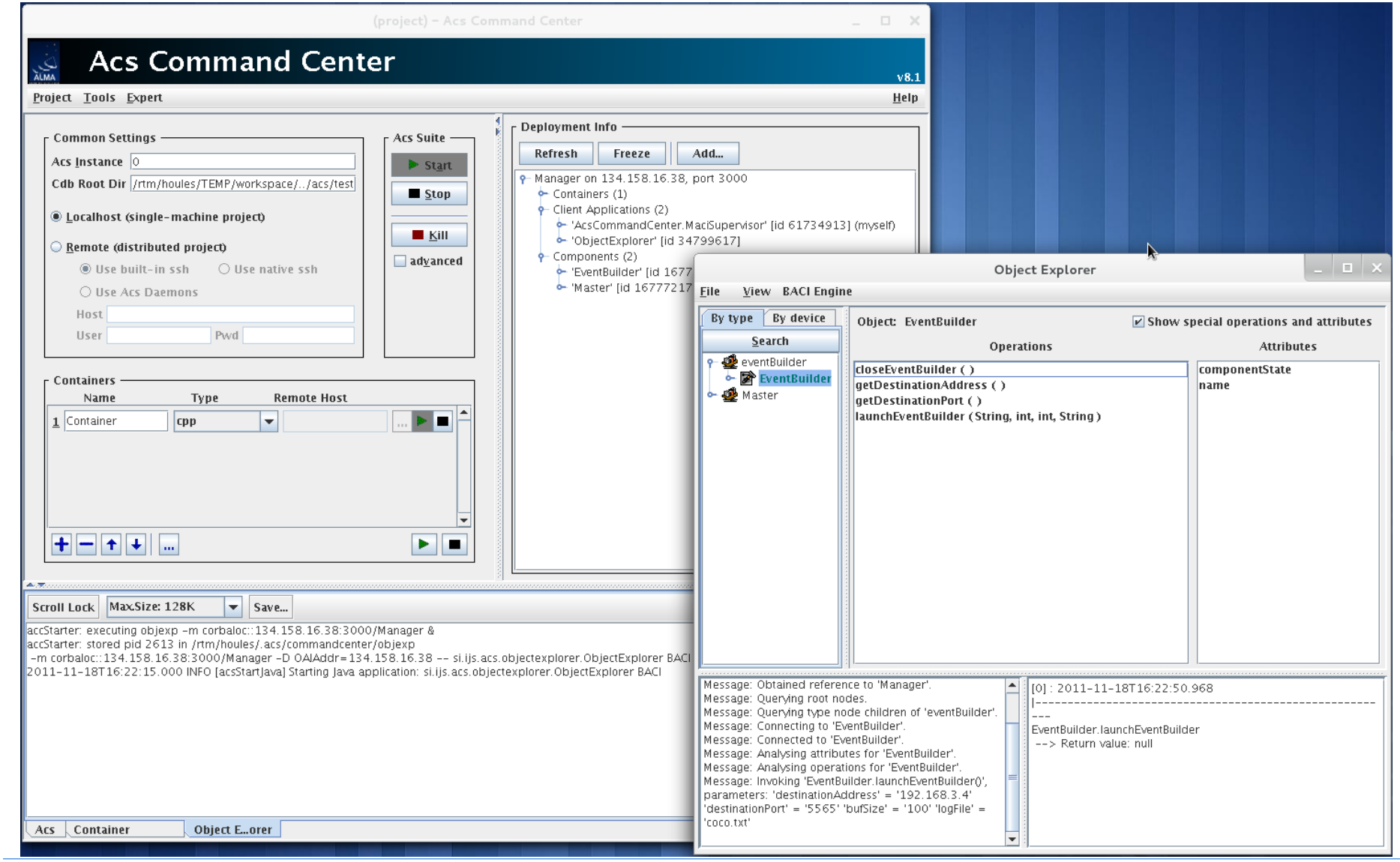
50 nodes (5 per HP server) sending data to interface 1
50 nodes (5 per HP server) sending data to interface 2
} 100 nodes

1st Achitecture	2nd Achitecture
Jumbo frames (8192 bytes) : 19,2 Gb/s (2,4 GB/s) with no loss CPU usage : 300 % (3 cores/12)	Jumbo frames (8192 bytes) : 19,2 Gb/s (2,4 GB/s) with no loss CPU usage : 180 % (1.8 cores/12)
Standard frames (1024 bytes) : 6,5 Gb/s (0,82 GB/s) with no loss CPU usage : 300 % (3 cores/12)	Standard frames (1024 bytes) : 8 Gb/s (1 GB/s) with no loss CPU usage : 190 % (1.9 cores/12)

→ Test with 300 nodes

Integration in ACS

The basic functions of the Event Builder are available from the ACS interface



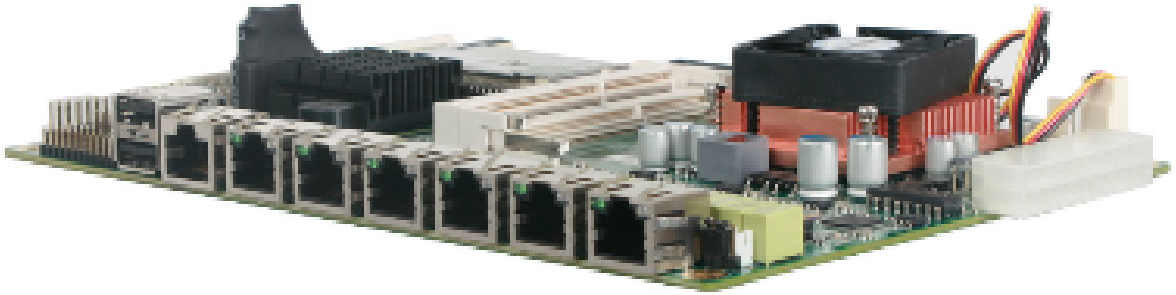
Stimulator

Need for a real stimulator

Need a stimulator to make :

- timing measurements on software
- real time validation
- algorithms validation
- trigger validation
- latency measurements on network
- front end boards and stimulator mix
- validate the complete acquisition chain

EVOC NET-1820



Intel Atom D525 dual core processor 1.8GHz
4.0 GB RAM
6 x Intel 82574L Giga LAN
(supports 9K frames and boot on LAN)
8-bit Digital I/O interface
1 x Parallel port, Serial port

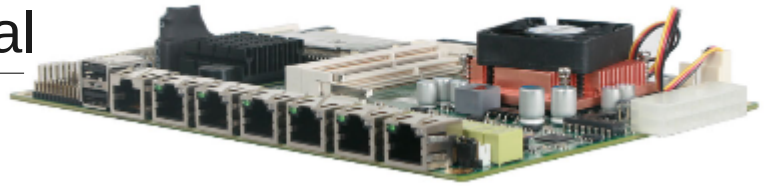
~ 300 euros each

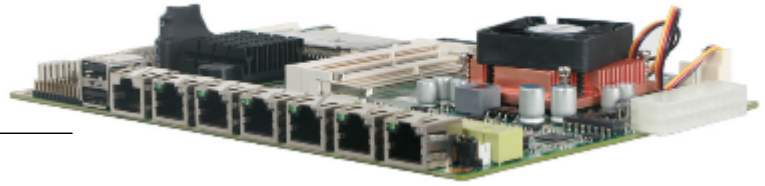
Measured throughput : ~ 2,4 Gb/s (400 Mb/s for each port)

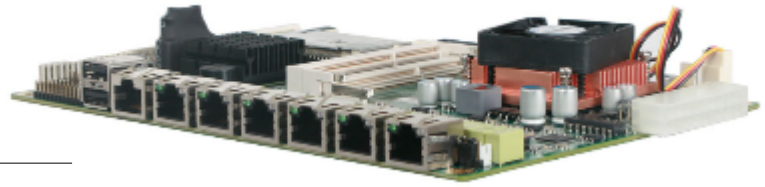
Testing configuration

Sync Signal

EVOC NET-1820

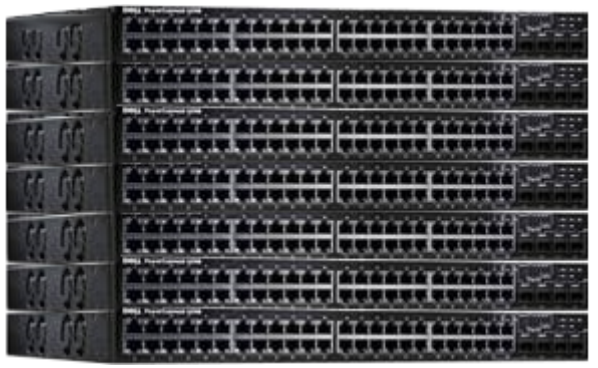






1Gb/s links

Powerconnect 6248 stack



10 Gb/s links

To the camera server

Future

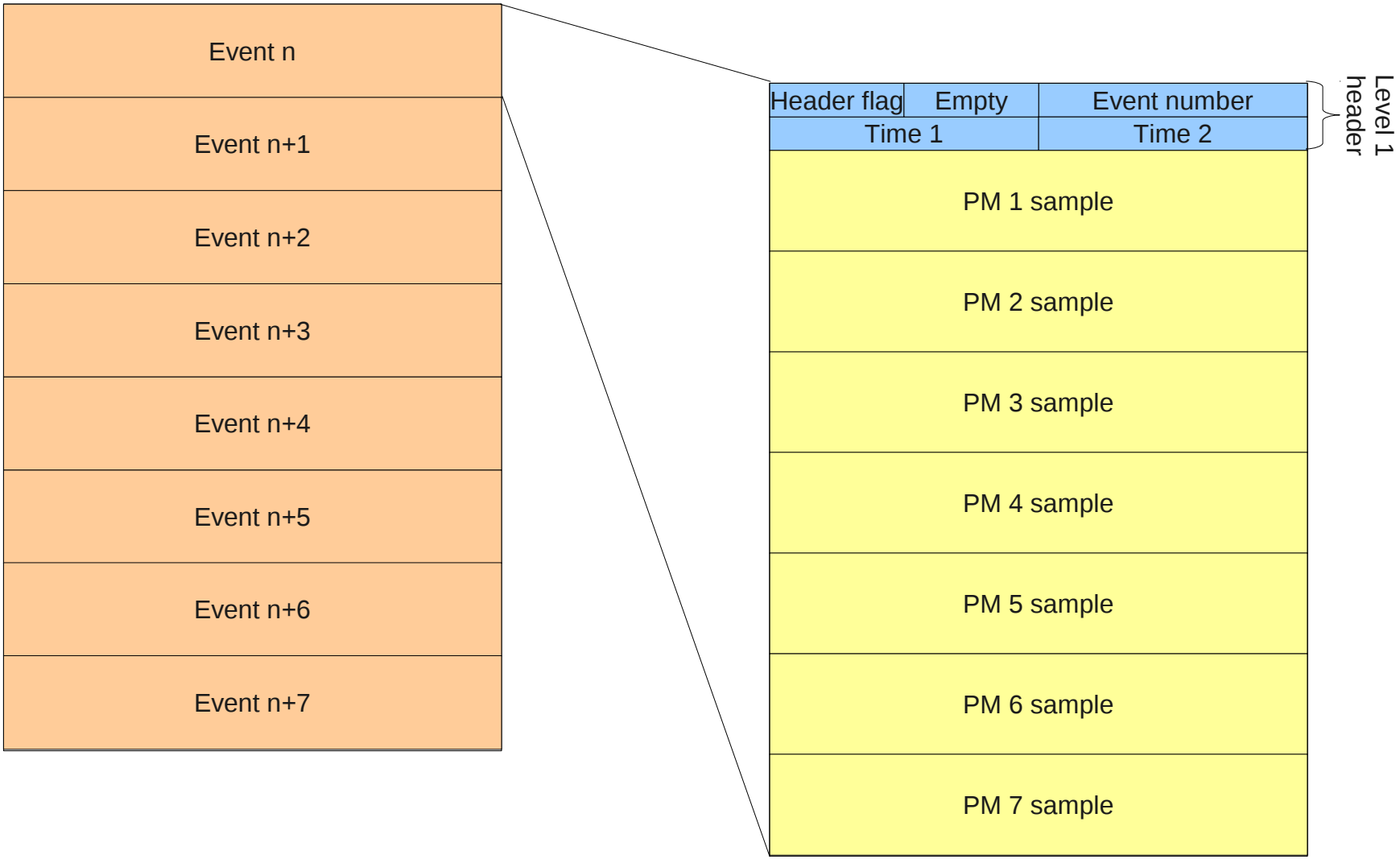
Future work

- Improve the event builder
- Try zero copy solutions
- Design a L2 trigger (CPU ? GPU ?)
- Make the software reliable enough for production stage
- Build a full-size stimulator

Interface definition

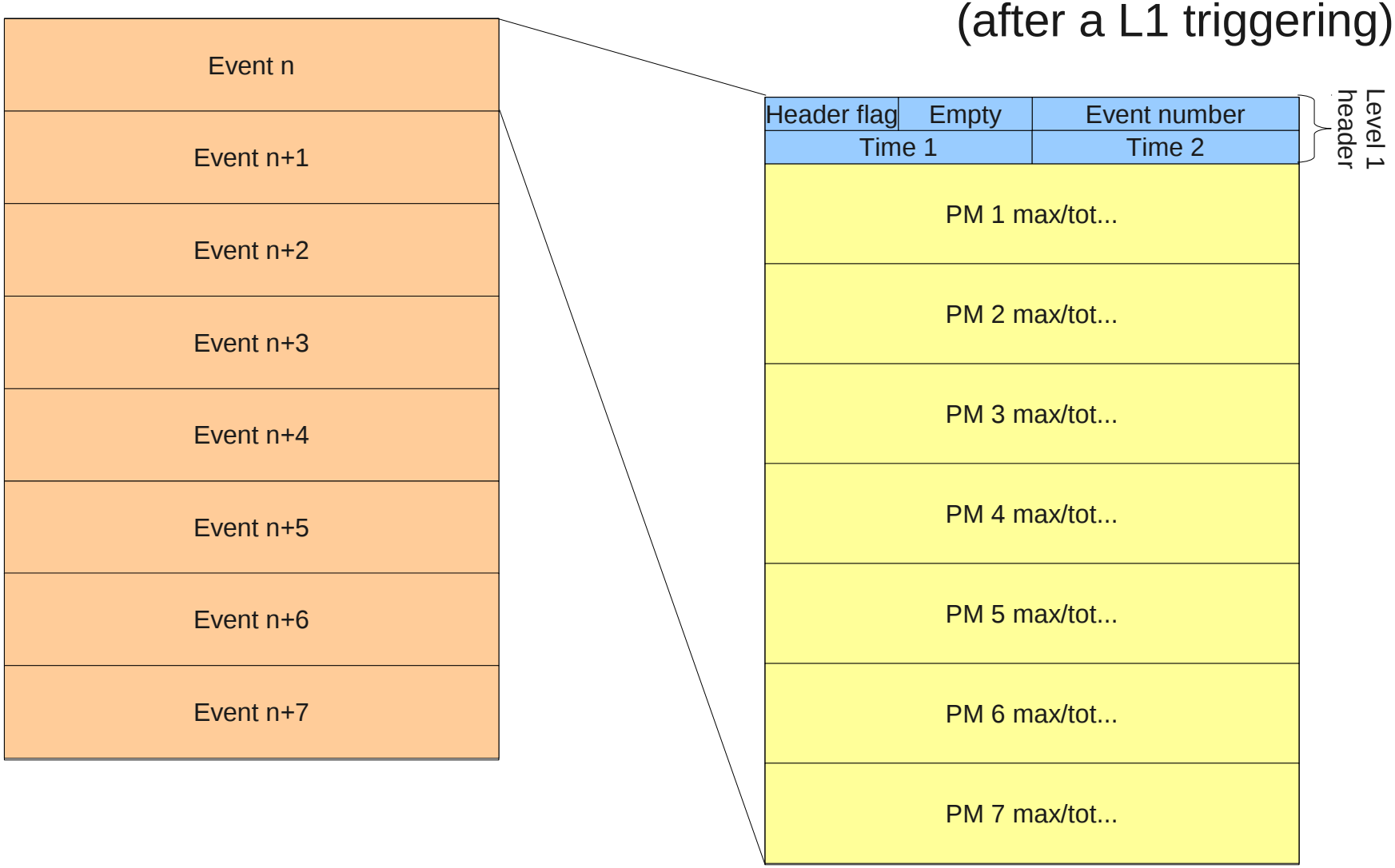
Data format : type 1.0

The front end electronics transmit all events (after a L1 triggering)



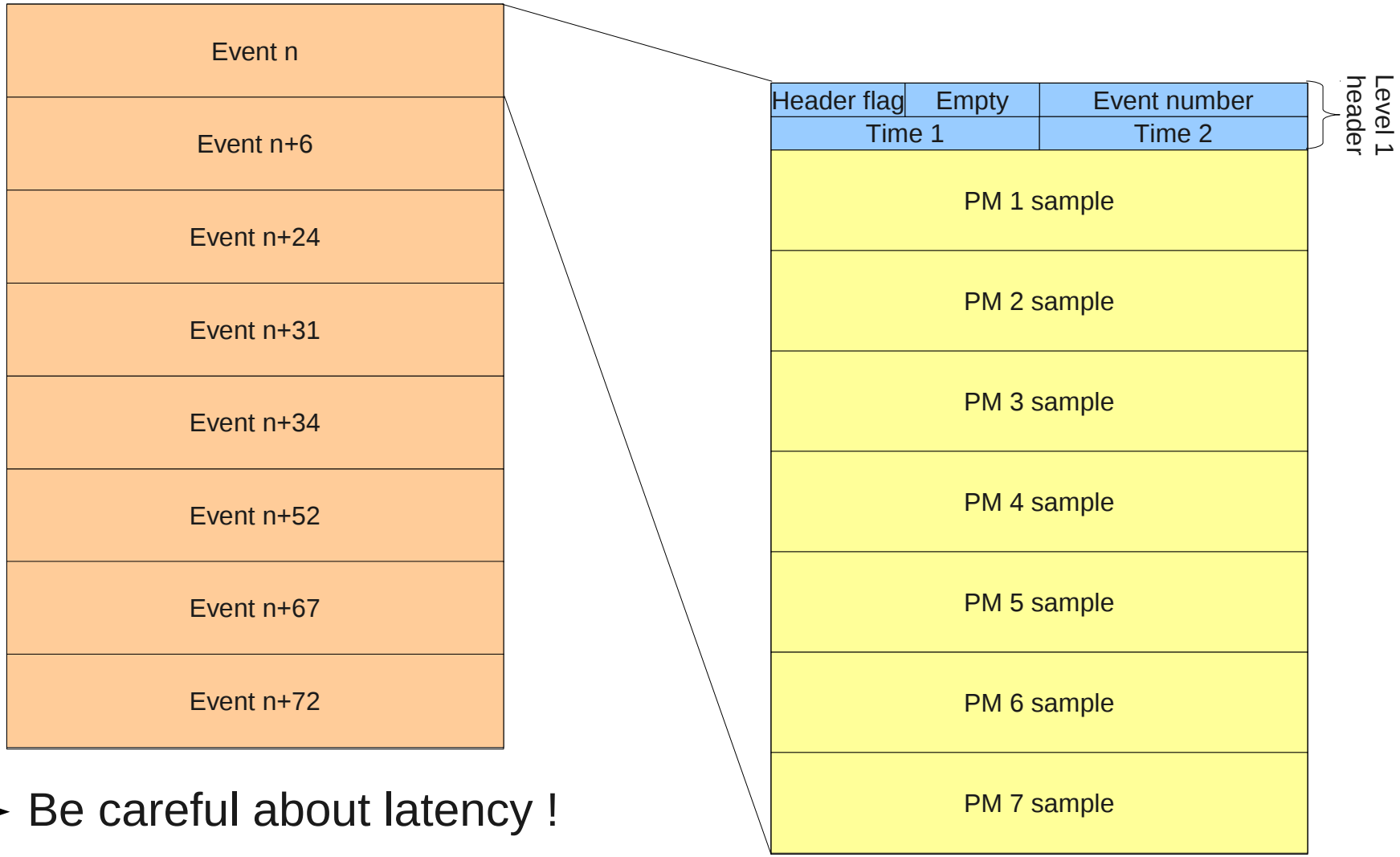
Data format : type 1.1

The front end electronics transmit a single value for each PM for all events (after a L1 triggering)



Data format : type 2.0

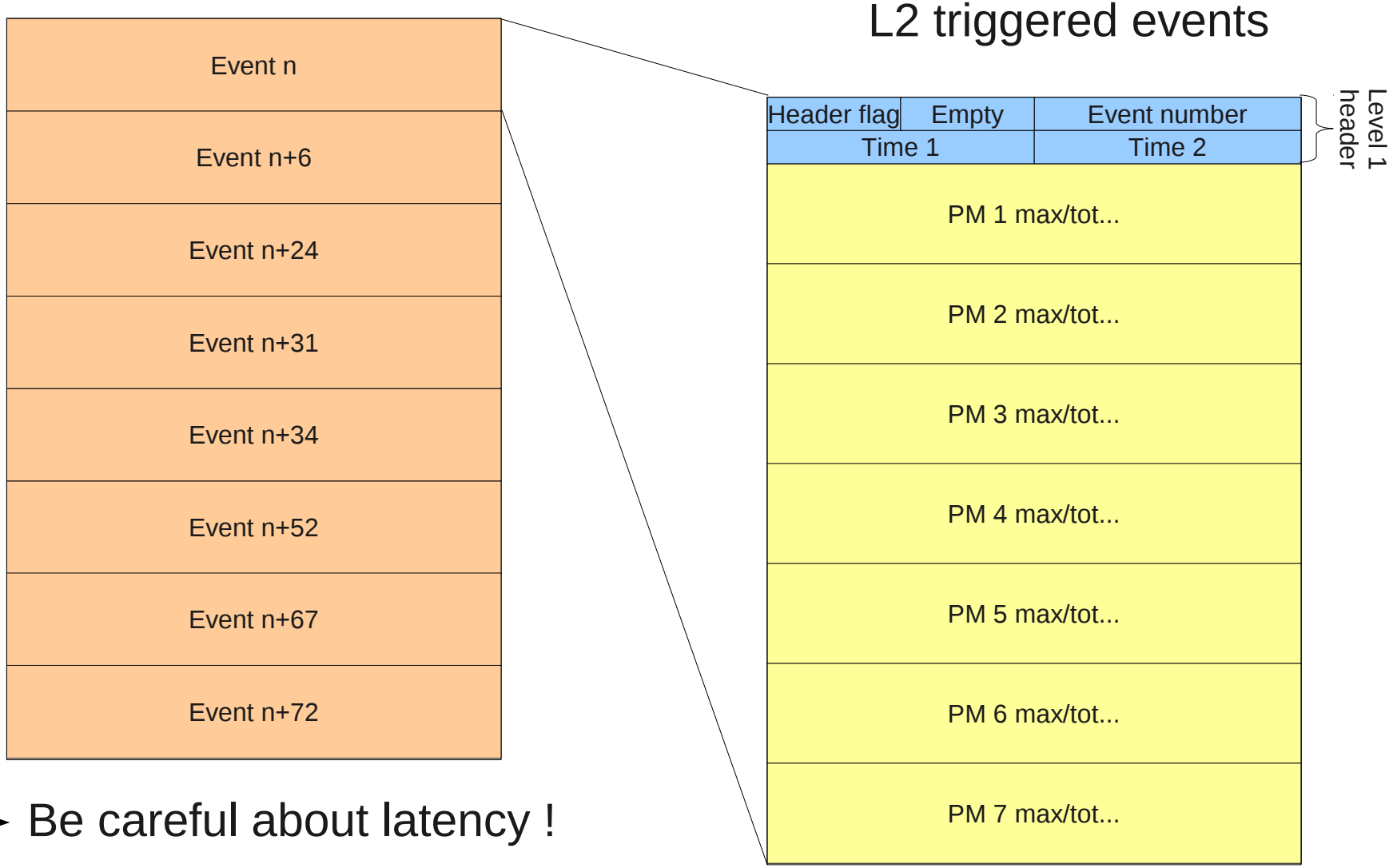
The front end electronics only transmit L2 triggered events



→ Be careful about latency !

Data format : type 2.1

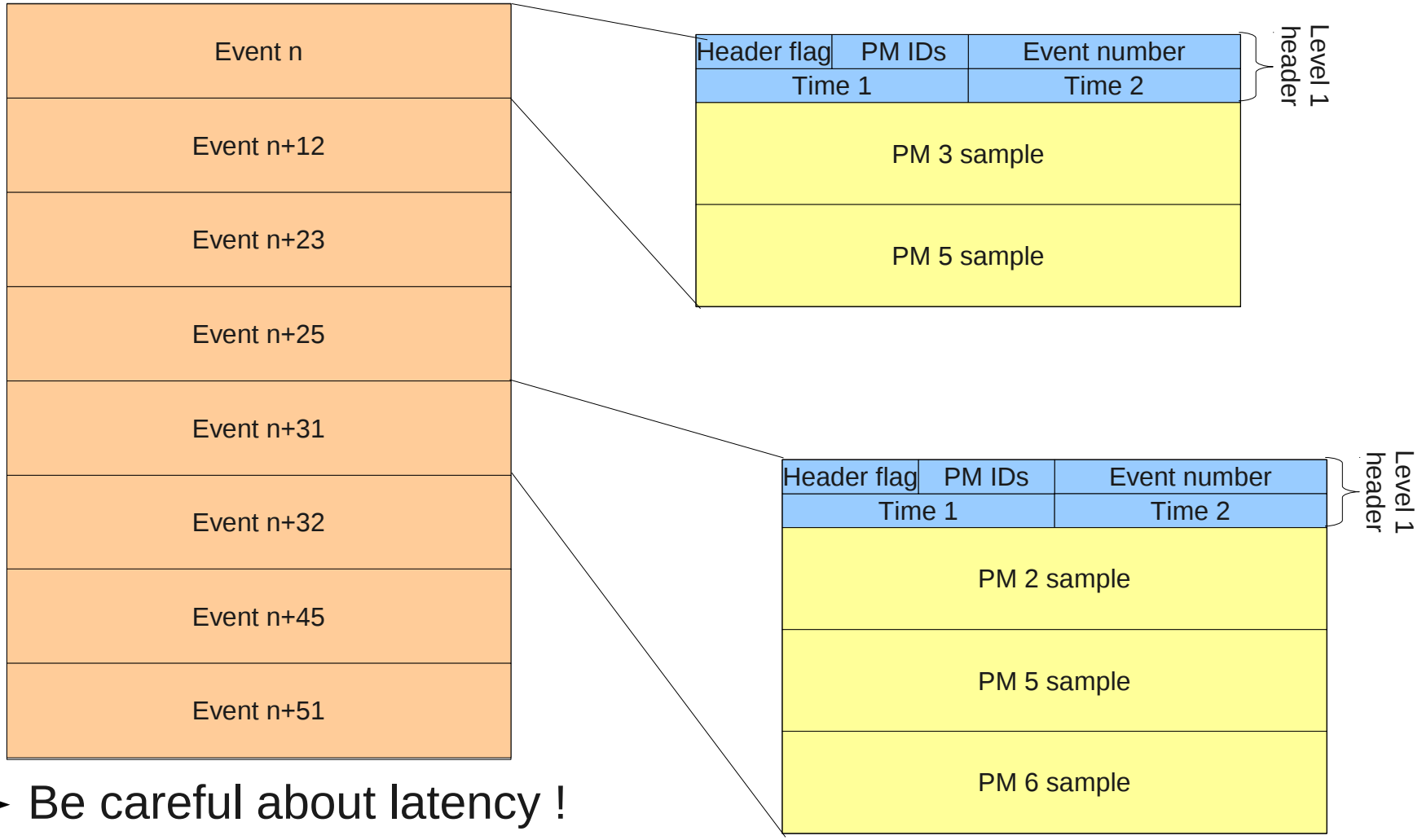
The front end electronics transmit a single value for each PM in L2 triggered events



→ Be careful about latency !

Data format : type 3.0

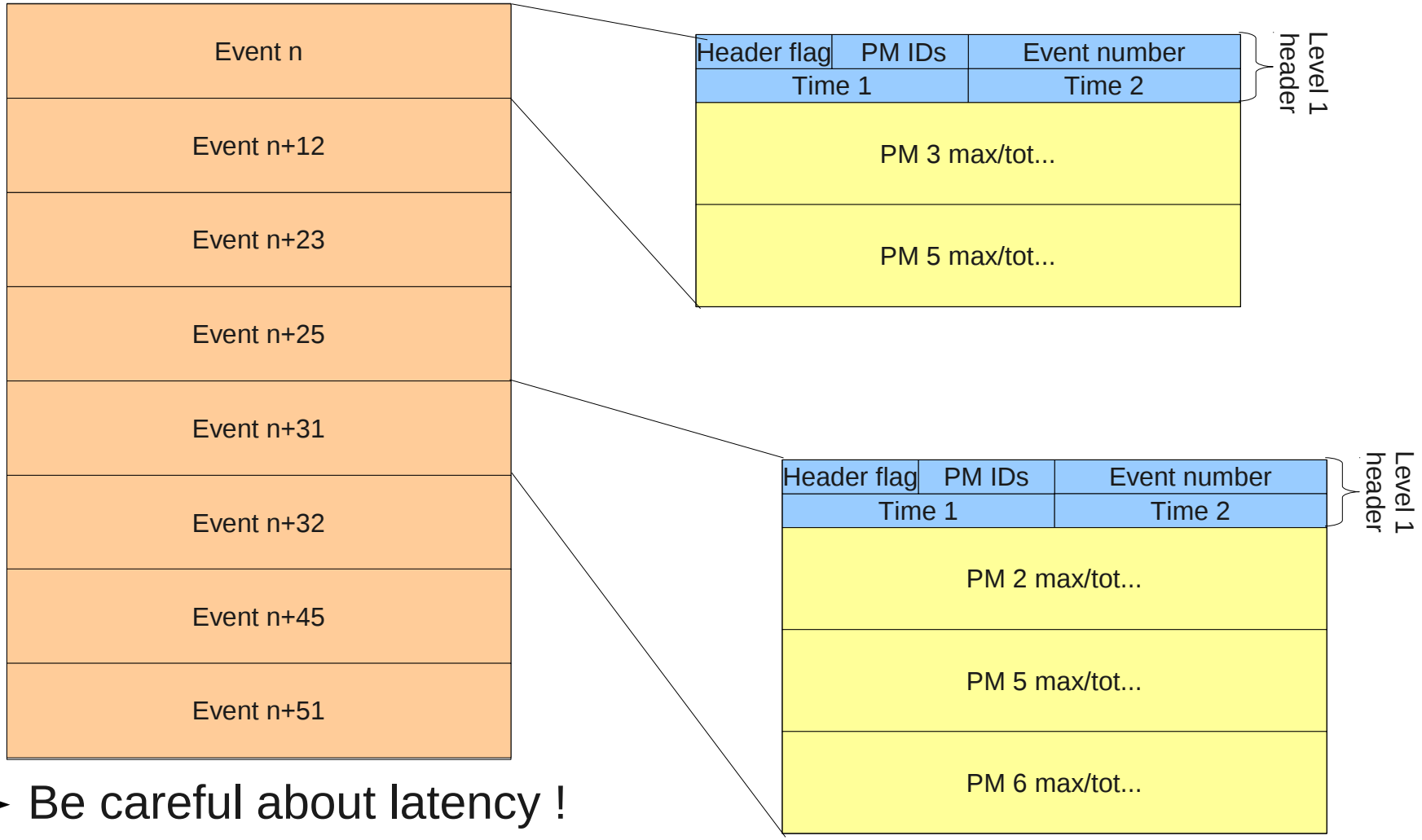
The front end electronics only transmit triggering PM in L2 triggered events



→ Be careful about latency !

Data format : type 3.1

The front end electronics transmit a single value for triggered PM in L2 triggered events

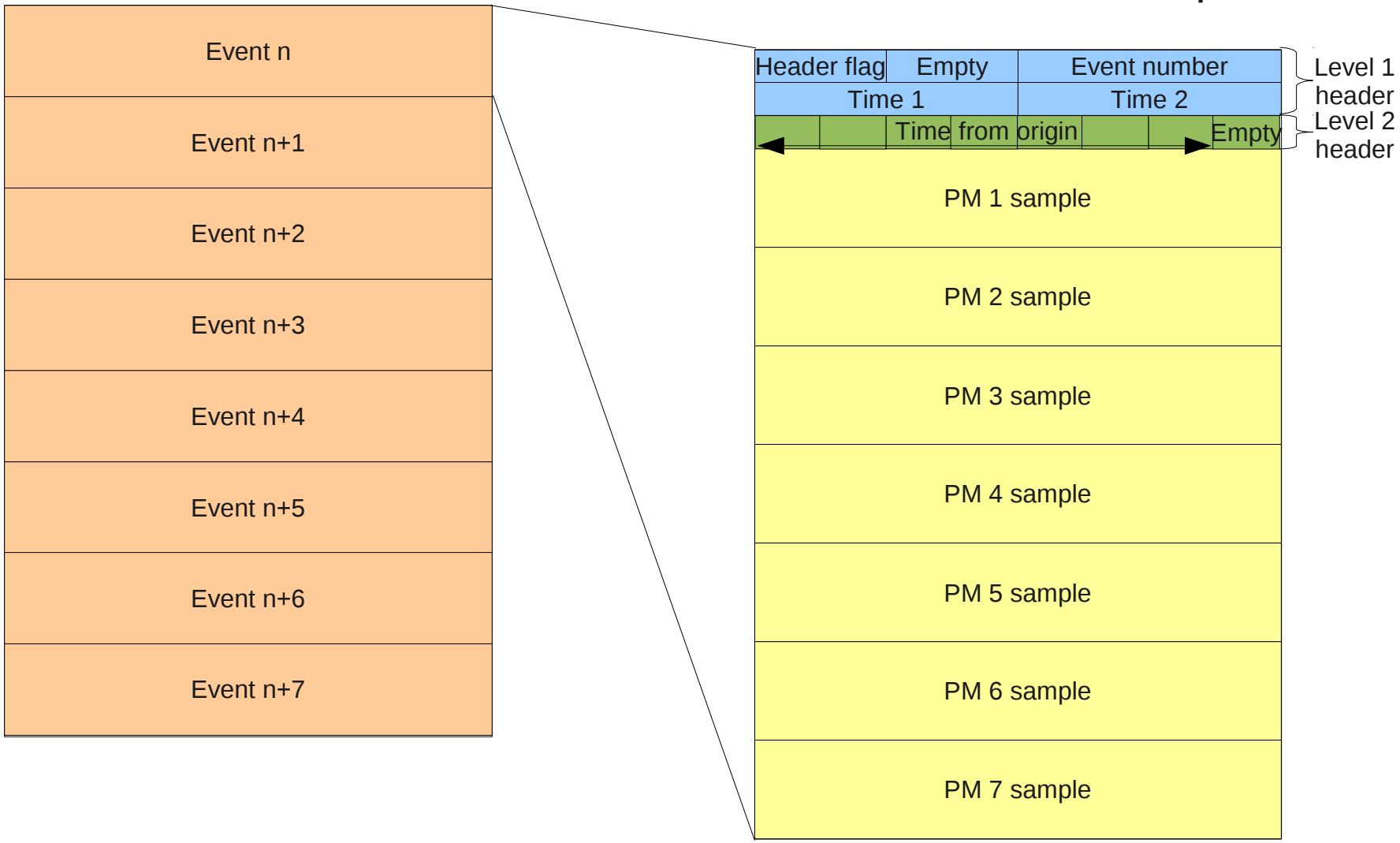


→ Be careful about latency !

Data format : type X.Y.1

Samples beginnings are truncated, can be applied to the types described

Example 1.0.1

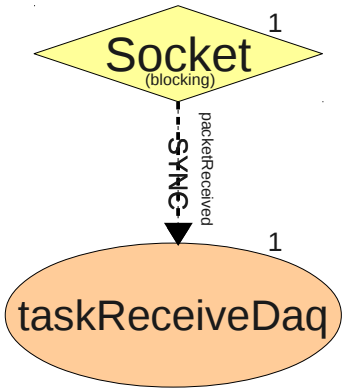


Data format : discussion

The formats exposed are just a proposition and must be discussed with all the concerned teams

Backup

Optimizing data reception



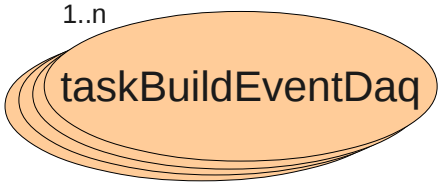
Receive data and copy to the packets buffer

- Multiply reception tasks to share the load
- Use jumbo frames
- Allocate interrupts to the core executing the task
- Enable interrupts coalescing
- Increase the Linux receive buffer size

To do :

- Experiment the Intel Direct Cache Access (DCA) : packets prefetched in cache memory
- Experiment zero copy solutions

Optimizing memory to memory copy



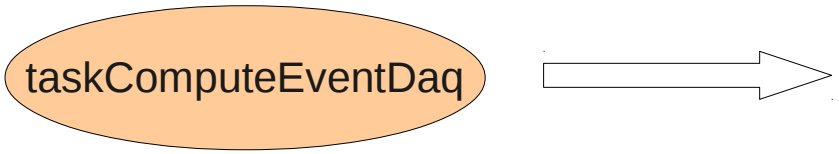
- Woken up each time a packet is copied to the packets buffer
- Copy from the packets buffer to the events buffer

- Use the SSE instructions
- Bypass cache memory to improve the other tasks performances
- Take advantage of the NUMA architecture : coordinate tasks affinity and memory allocation on nodes for the packets buffers
- Allocation of contiguous and aligned memory areas

To do :

- Why not synchronizing when several packets have been received ?
- Think about the allocation of the event buffer and nodes distribution
- Why not introducing calculations to take advantage of the memory loads ?
- Experiment the new Intel Advanced Vector Extensions (AVX)

Events processing



- Is woken each time X events have been completely rebuilt
- Has access to the complete events

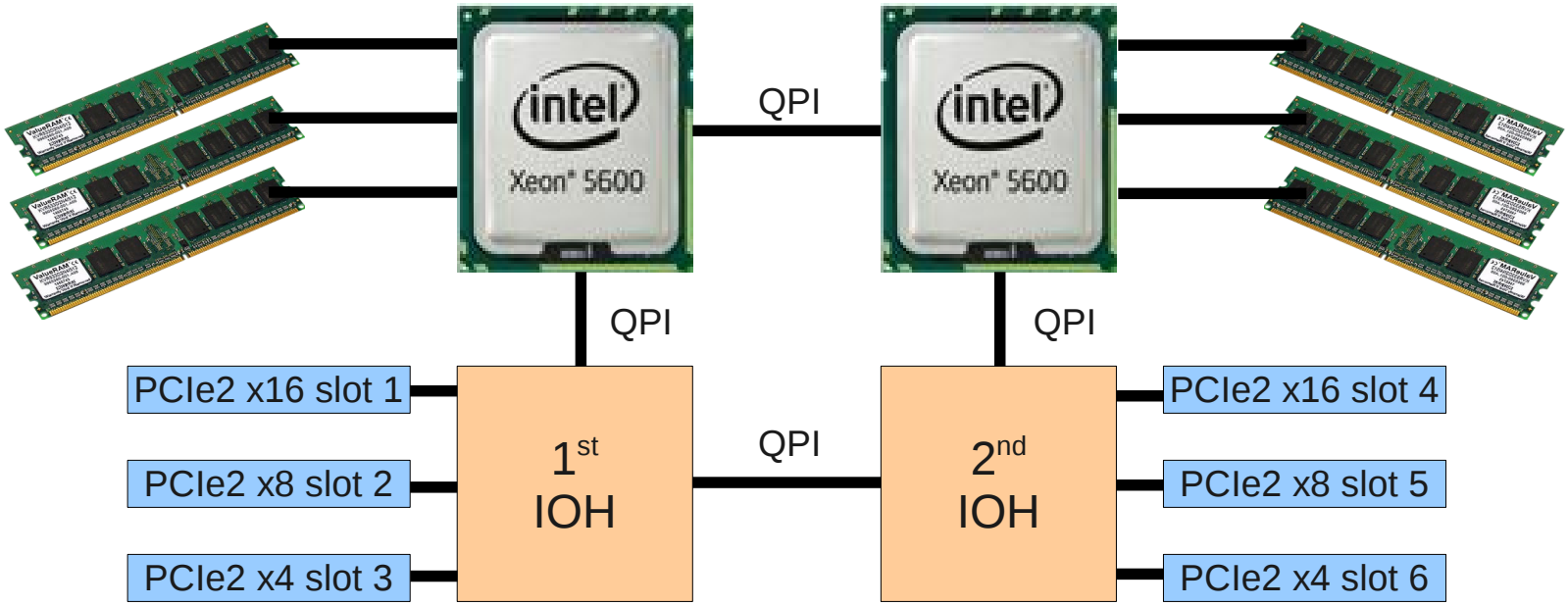
Currently :

- only checks the integrity of the complete events when ready

Later :

- Perform calculations for L2 triggering
- Transfer data to GPU if needed
- Store data to a local storage if needed
- Send data to the central server
- Be multiplied if load is important

Non Uniform Memory Access



QPI @ 6.4 GT/s -> 25.6 GB/s (12.8 GB/s unidirectional)

Triple channel memory @ 1333 MHz -> 32 GB/S