Calice DAQ2 Software

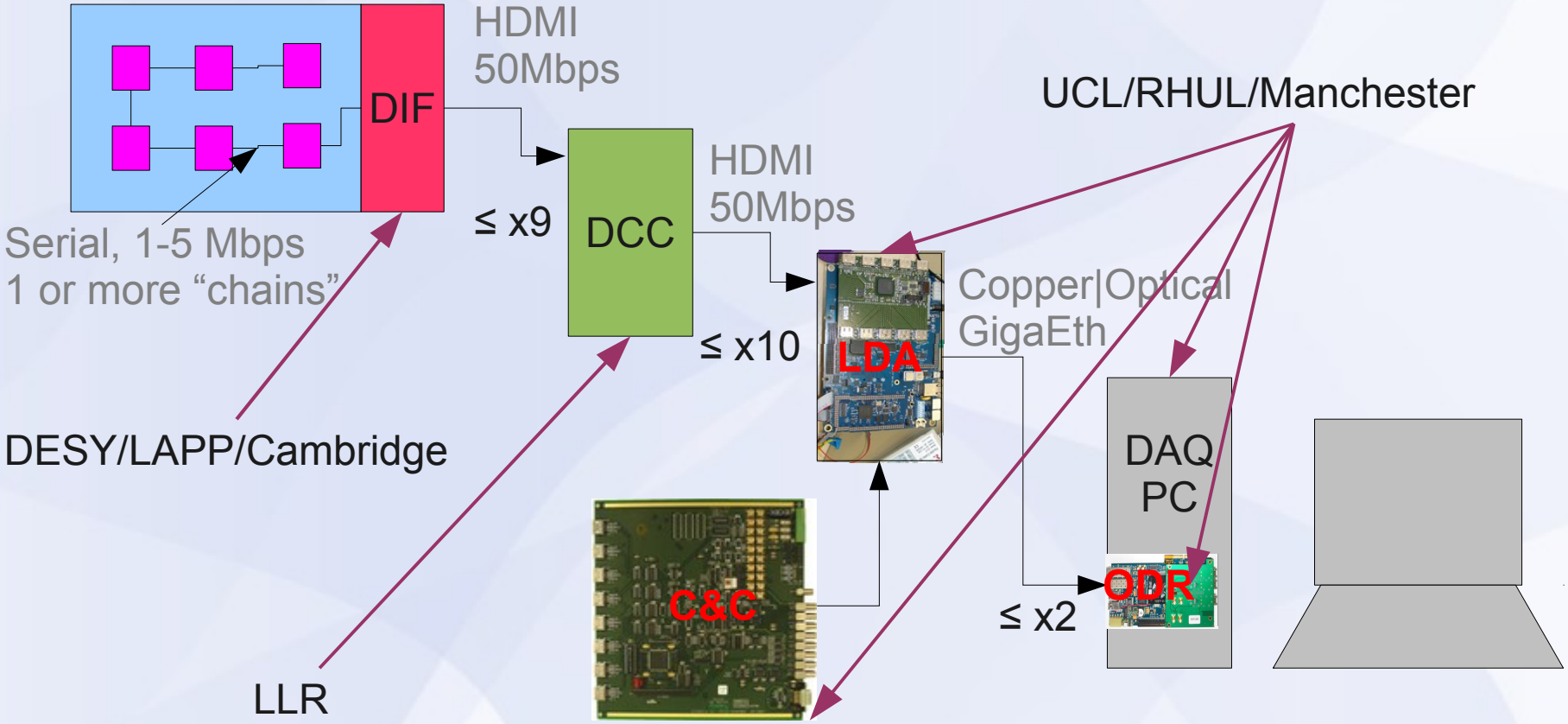
David Decotigny

Outline

- DAQ2 SW overview and status
- DAQ2 Read-Out raw data formats and reassembly (Chips-wire-LCIO)
- Simple bandwidth usage study
- DAQ2 SW perspective
- Open questions / subdetector integration

DAQ2 SW overview and status

“DAQ2” Calice: System Overview



A network-oriented setup: a tree of data concentrator cards

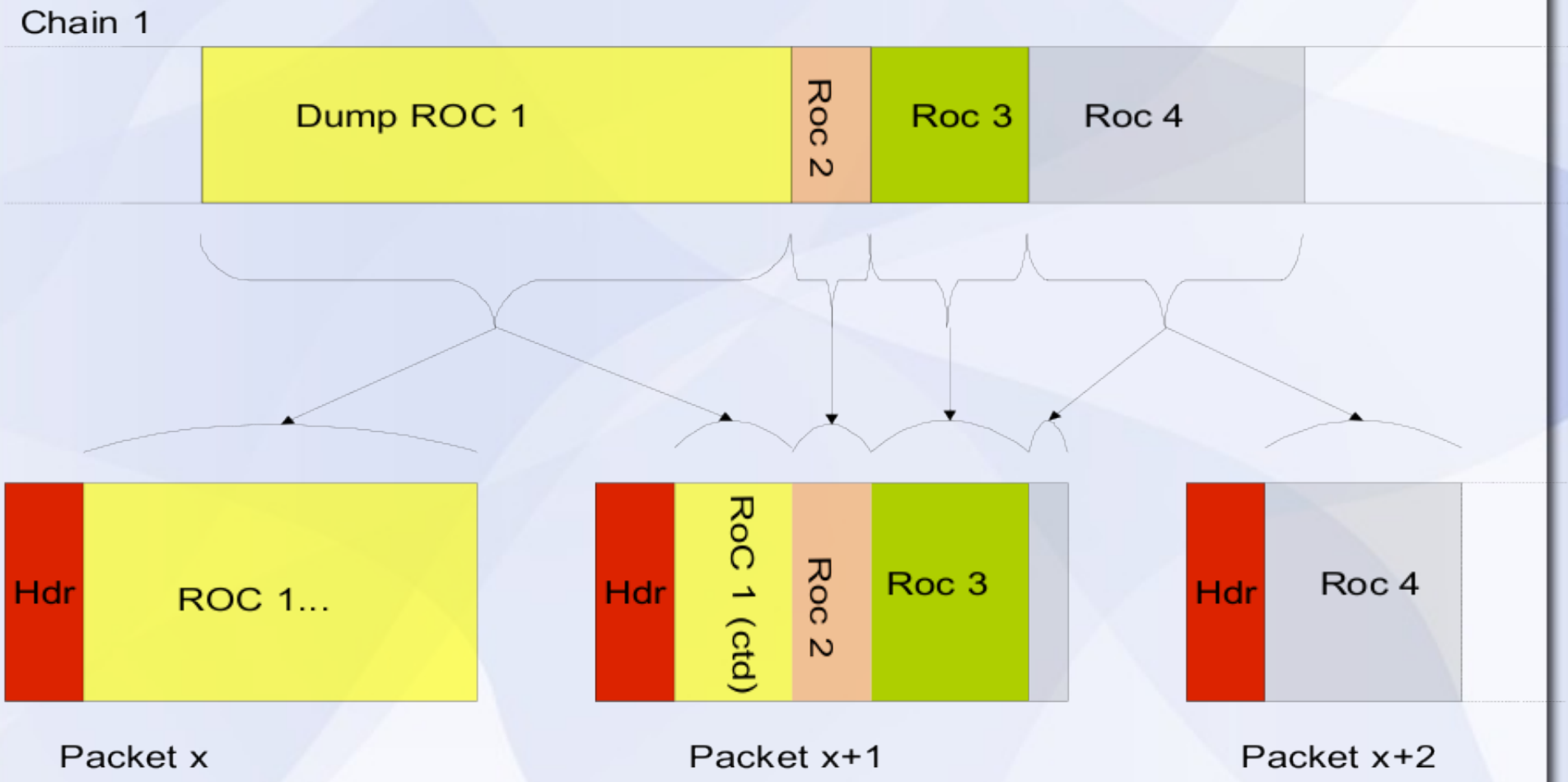
Software status

Hardware/Software interfaces

- ODR kernel driver by A. Misiejuk (Manchester)
  - Able to store data packets to disk
- CCC interface (DOOCS + Agnostic)
  - Register accesses
- LDA interface (Agnostic)
  - Register accesses through ethernet
- DCC interface
  - Tests through USB
- DIF interface
  - Tests through USB
- Integrated DAQ Software (DOOCS, V. Bartsch)
  - GUIs to control the ODR + CCC
  - FSM to control the device servers
  - DB framework to retrieve configs

DAQ2 Read-Out raw data formats and reassembly (Chips-wire-LCIO)

Read-out data format
On the wire at DIF level



https://svn.in2p3.fr/calice/docs/trunk/DIF_readout_format.doc?view=co

Read-out data format

On the wire at DIF level

- "ROC Dump" = a series of "frames":
  - Local timestamp
    - Allows zero-suppresion
  - Digital values (1 value for each channel)
    - eg.: 64 channels x 2 bits for SDHCAL (+ timestamp), 36 channels x 24 bits for ECAL (+ gain/hit/timestamp)
- DIF Packets:
  - Max packet size chosen = 1kB
    - Compatible with Ethernet or UDP encapsulation at LDA level
  - Header: allows to reassemble the ROC dumps after DIF has split them

Read-out data format

On the wire at LDA/ODR level

- A series of Ethernet packets
  - Each packet = encapsulates 1 DIF packet
  - Direct ethernet or UDP encapsulation foreseen (for now: ethernet)...: ?
- This is what the PC receives
  - Has to be efficient when delivering them to the app (eg. With 1 Byte payload, freq packets = 1.5 M packets / s max on ethernet 1Gbps)
  - ODR designed to support this
  - Default linux drivers & IP stack pushed to their limits (better with UDP or Ethernet ?... needs evaluation)

Read-out packet reassembler
Prototype: overview

- Goal: from a stream of un-ordered interleaved packets, reconstruct the original ROC read-out data sequences
- Parse and store the re-assembled data as structured data (LCIO format)

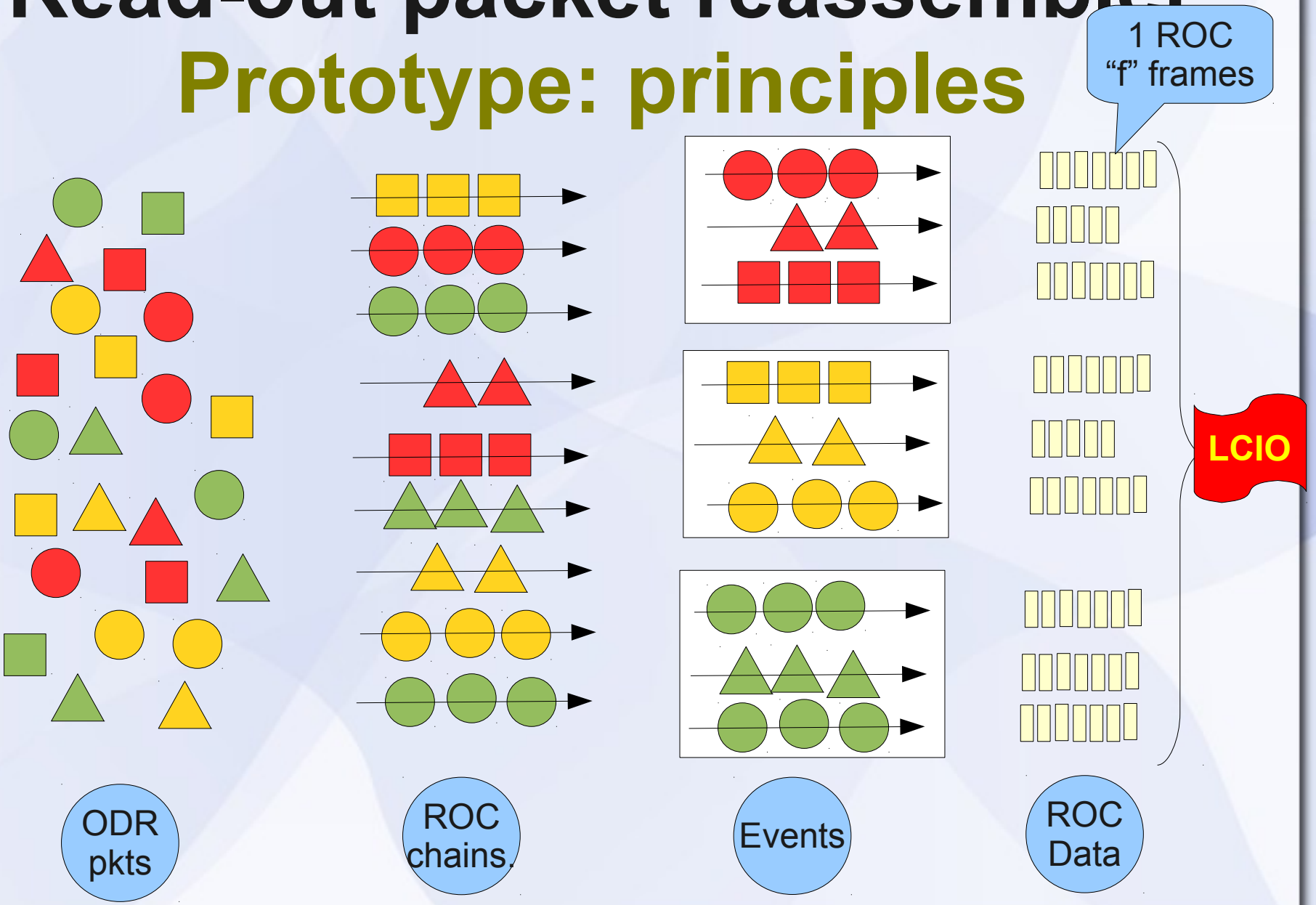
➔ Available as a simulation only... for now

  - Soon to be evaluated over a real 1Gbps link + test machine

https://svn.in2p3.fr/calice/online-sw/trunk/daq/

Read-out packet reassembler

Prototype: principles



Read-out packet reassembler
Prototype: characteristics

- Prototype: 3-level pipeline
  - Identification of the ROC chain data sequences (containing the ROC hit data)
  - Assemble the ROC chain data belonging to the same “event” (ie. Start-readout event) together
  - Parse the ROC data sequences to get the ROC hit data
- Meant to be parallelized (with threads)
- Note: great care about memory leaks/errors

Read-out packet reassembler
LCIO storage format

- LCIO Format:
- 1 Event =
  - Detector Name, Run#, Event#, Timestamp
  - 1 LCCollection per DIF
    - Unique DIF ID (ODR/LDA/LDAlink/DIF Id)
    - 1 LCGenericObject per ROC Data
      - Unique ROC ID (ROC Chain ID/ROC Id)
      - Chip config (type, acqMode)
      - Number of frames
      - Blocks of frame data (for HR2: 20B / frame)

https://svn.in2p3.fr/calice/online-sw/trunk/daq/reassembler/lcio_dump.hpp?view=markup

Simple bandwidth usage study

Simple bandwidth usage study

Calo Prototypes

- SDHCAL: cubic meter
  - 40 layers * 3 DIFs
  - → 2 LDA needed
  - 48 chips/DIF
  - Max 128 frames (20B) per chip per read-out
- ECAL prototype:
  - 30 layers * 1 DIF
  - → 1 LDA needed
  - ~130 chips/DIF
  - Max 16 frames (146B) per chip per read-out

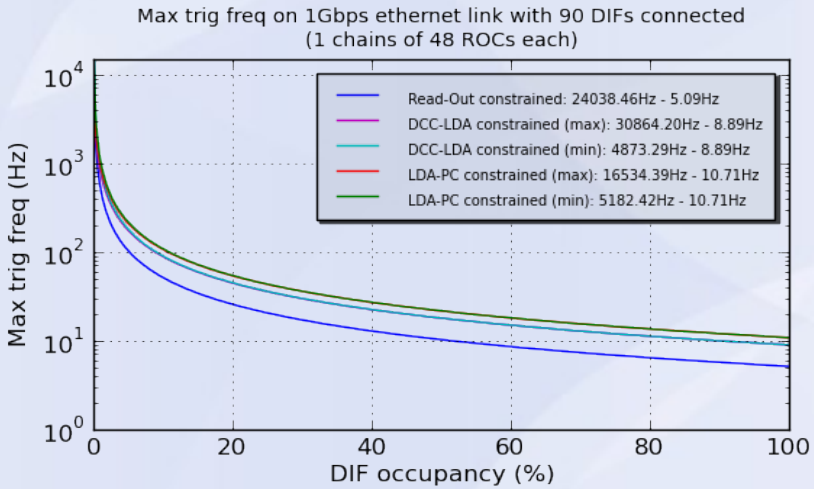
Simple bandwidth usage study

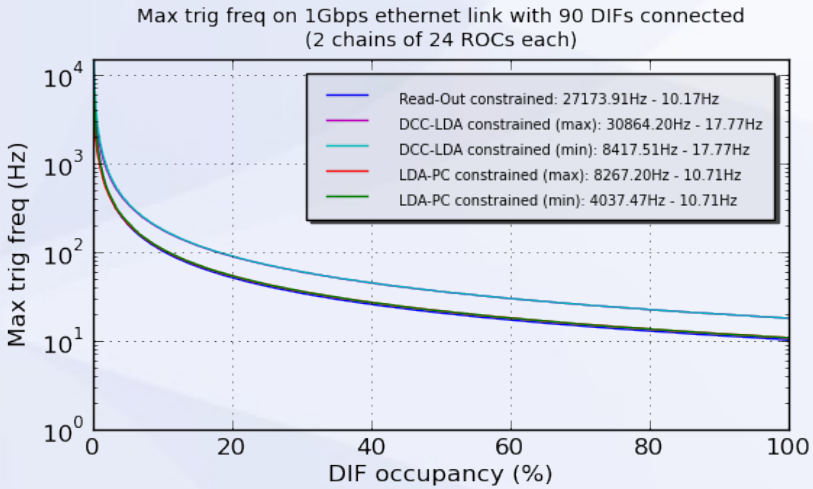
- Per 1Gbps ethernet link (ie. 1 LDA):
  - eg. DHCAL with 1 LDA x 10 DCC x 9 DIFs x 48 HR2
  - How many read-out achievable per second ?
- ➔ Depends on:
  - Chip read-out constraint: **max** number of frames stored in a DIF chain for a bunch train
    - Next read-out allowed after all chips read-out
    - Slow read-out rate (1-5 Mbps)
  - Network bandwidth constraints (DCC->LDA & LDA->PC): **average** number of frames stored in a DIF chain

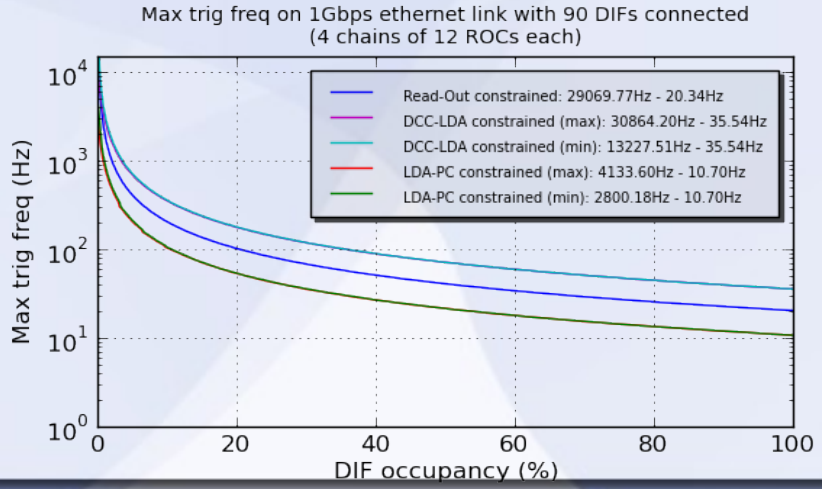
Simple bandwidth usage study

Upper limits (analytically)

DHCAL Trigger rates with raw Ethernet packets







Similar profiles for the ECAL

Similar profiles when using UDP over Ethernet

Simple bandwidth usage study
First observations

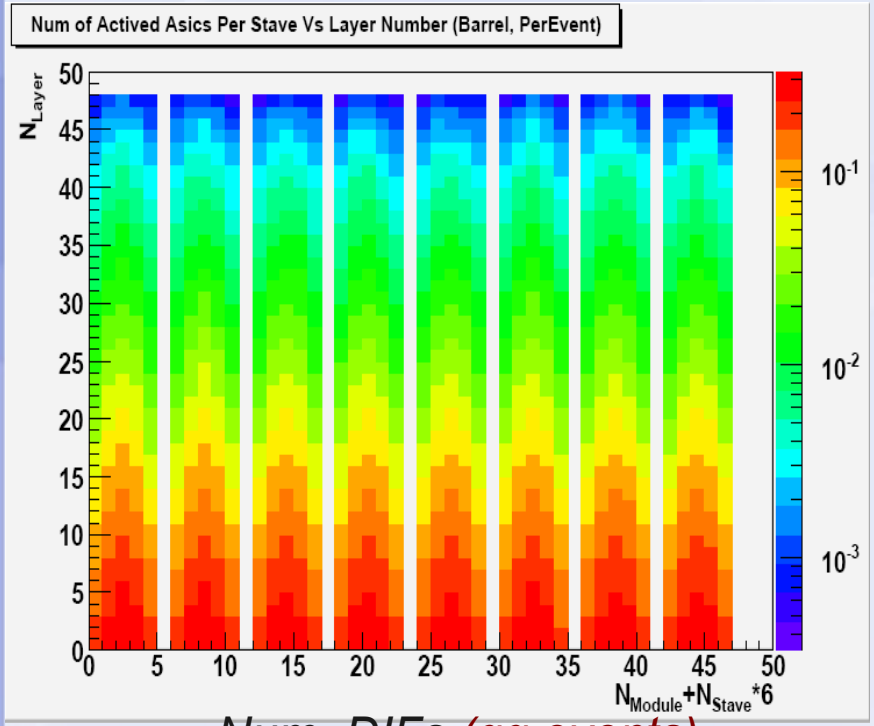
- Supports 1 fully-loaded SDHCAL LDA at up to ~10Hz (calibration, 2 chains/DIF)
- Mainly limited by chip read-out
  - The shorter the chains, the smaller the dead time needed to read-out the chips
- ➔ Most loaded DIF determines the performances
  - ➔ Fundamental in calibration mode

Simple bandwidth usage study

Side note: expected behavior

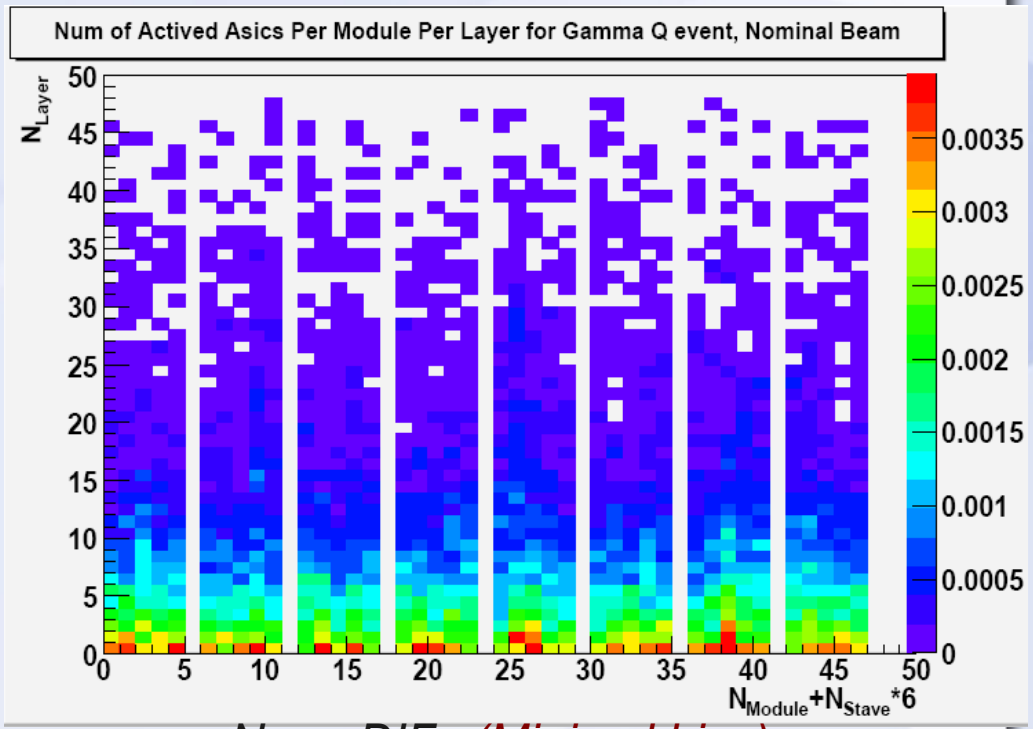
- From Manqi Ruan's simulations in the barrel:

(ILD mode, qqbar with GigaZ option)



Num. DIFs (qq events)

Worst expected DIF occupancy:
3% per train (GigaZ)



Num. DIFs (Minimal bias)

Worst expected DIF occupancy:
12% per train (ILD nominal)

Agenda and open questions

DAQ2: SW work ahead...

- Following weeks: evaluate reassembly + LCIO storage on real PC (simulated DAQ2 hardware)
- With Lyon:
  - Summer 2010: validate first setup with DCC+CCC controlling a m2 over USB
  - Fall 2010: integrate the LDA
- In parallel:
  - Fall 2010: Topology-aware slow-control config database + tools to generate the SC blocks
  - Fall 2010: Abstract SW layer to indifferently drive LDA/Ethernet or DCC/USB
    - Xdaq + DOOCS + Tango

Open questions

- Common data pipeline across subdetectors ?
  - Common scheme (load balancing to several servers, how ?) and data format ?
  - Prefer UDP over Ethernet ?
- Common configuration database ?
  - Common way to configure devices (same FSM ?) ?
- Common way to identify coming data from the same train across subdetectors ?

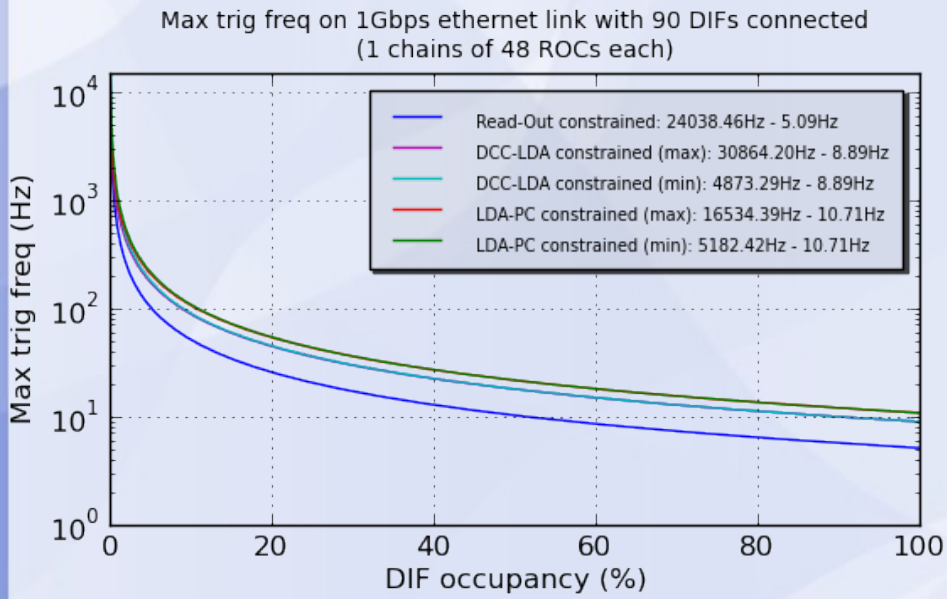
Thank you !

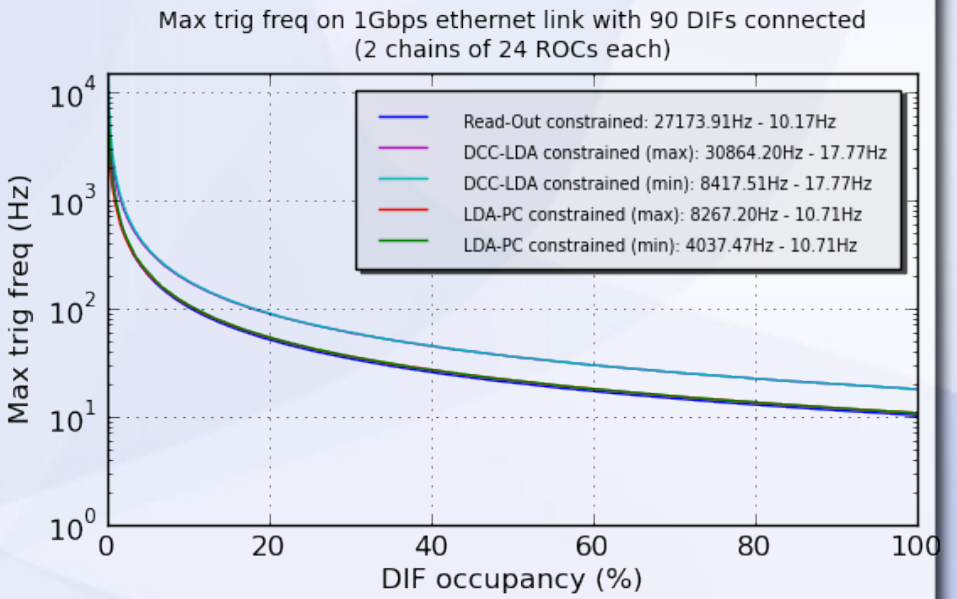
Bonus slides

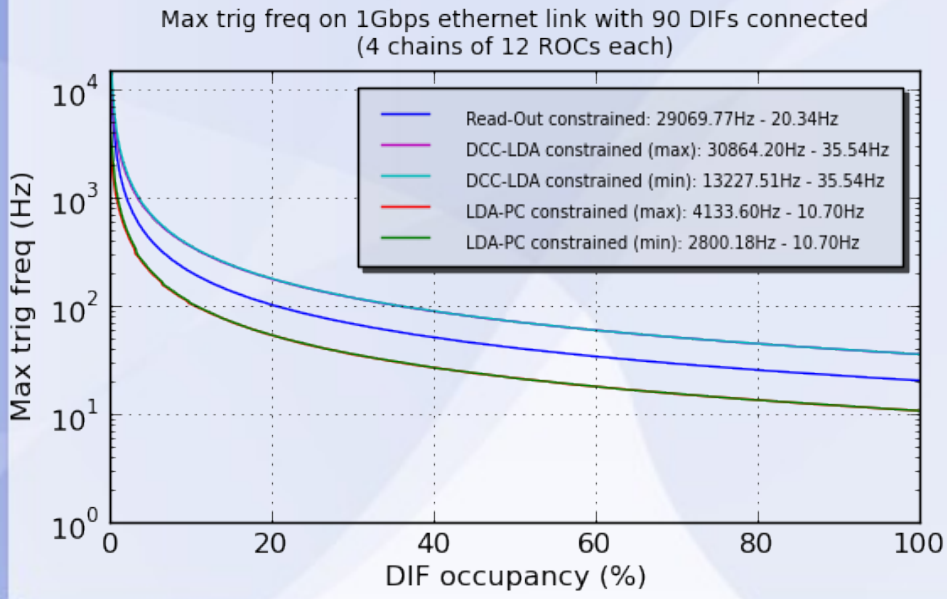
UDP vs. raw ethernet encapsulation...

- Would only affect LDA firmware
- To be confirmed soon:
  - Better bufferization in kernel (?)
  - Better compatibility with standard network switches (?)
  - Could be used with default NIC linux drivers (?)
    - First tests with ee1000 and UDP: 458000 packets / s OK (limit = sender)

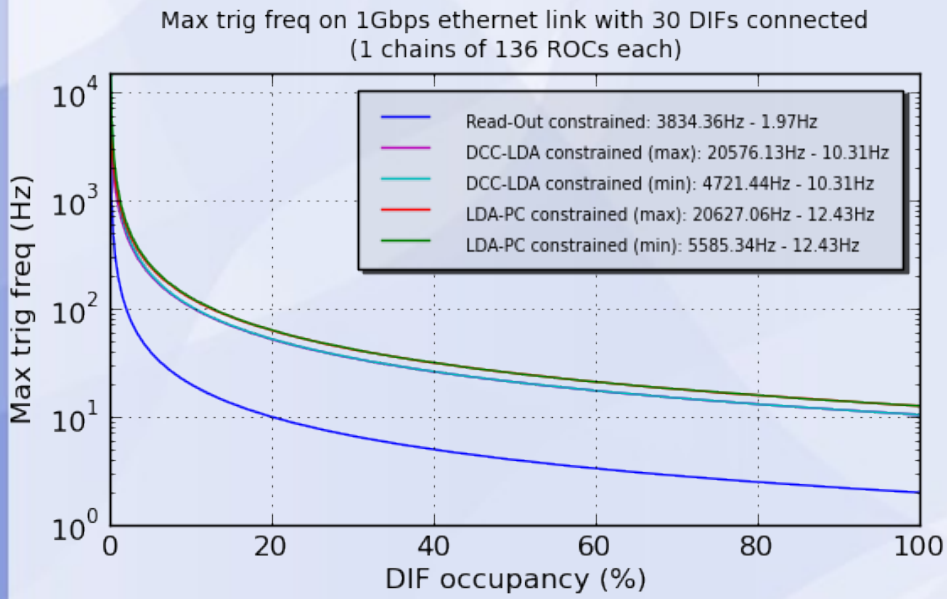
DHCAL Trigger rates with raw Ethernet packets

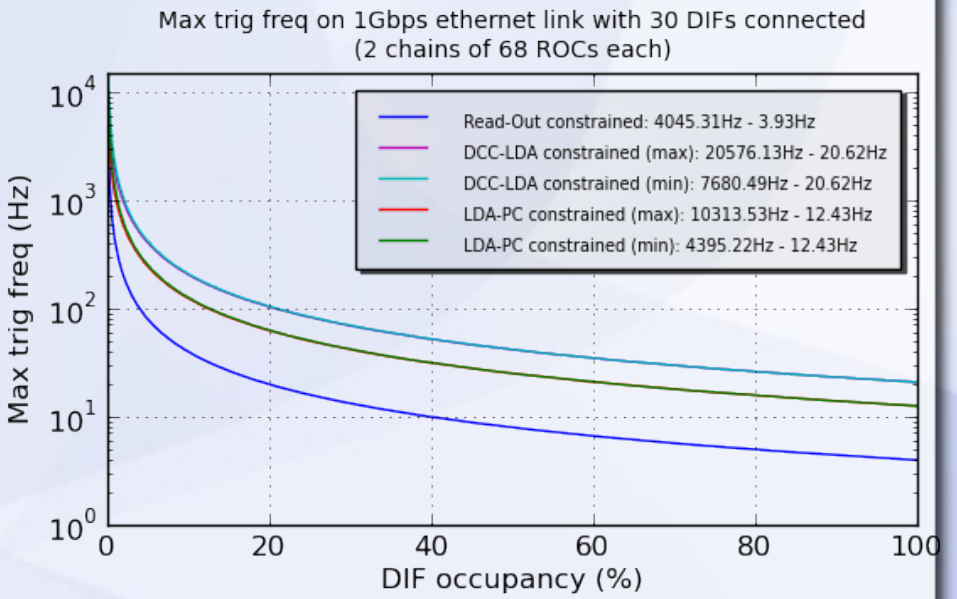


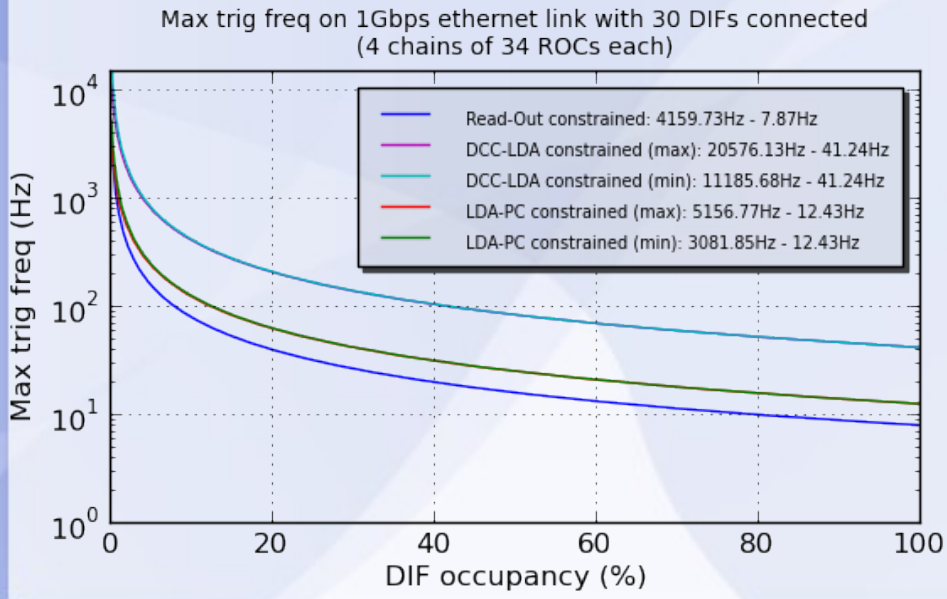




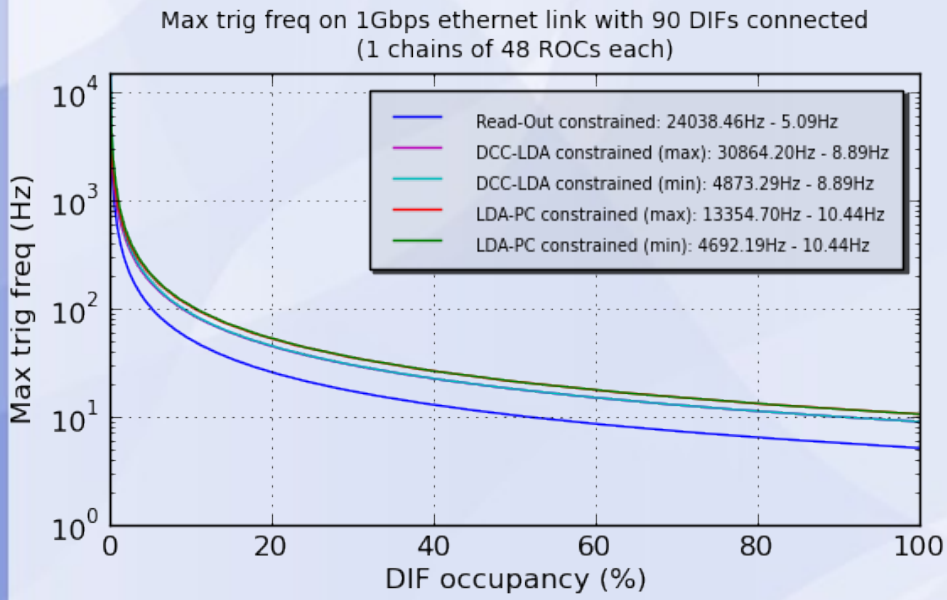
ECAL Trigger rates with raw Ethernet packets

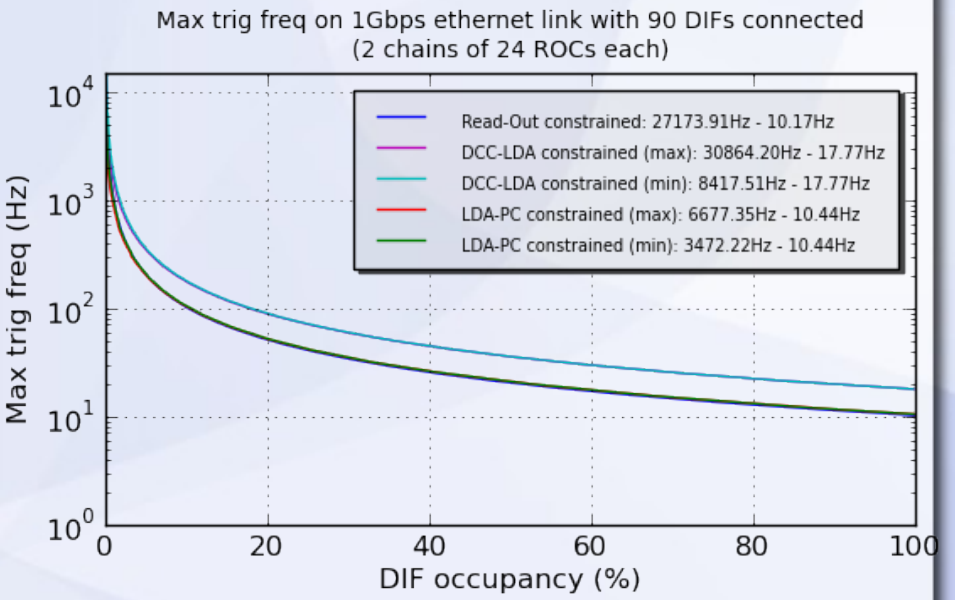


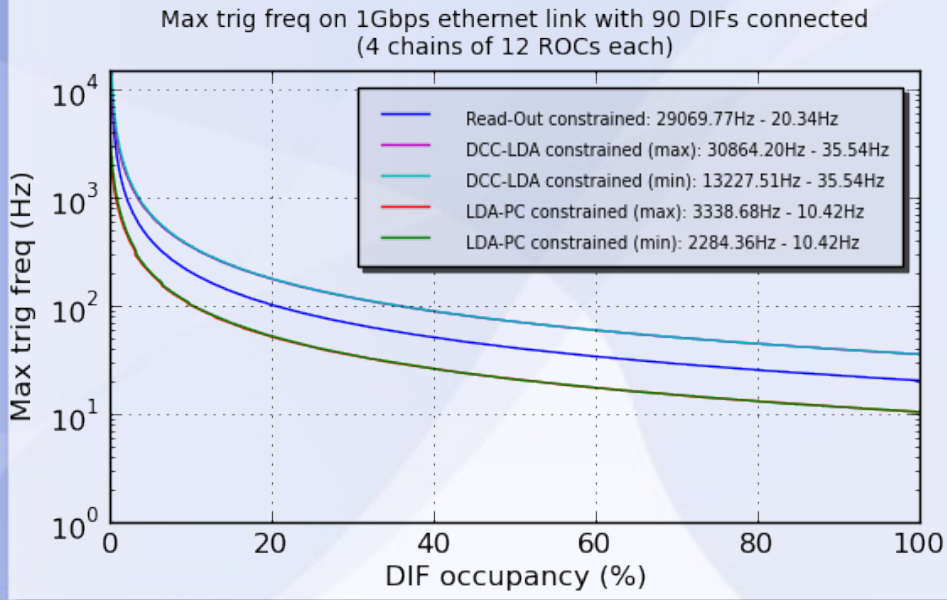




DHCAL Trigger rates with UDP packets







ECAL Trigger rates with UDP packets