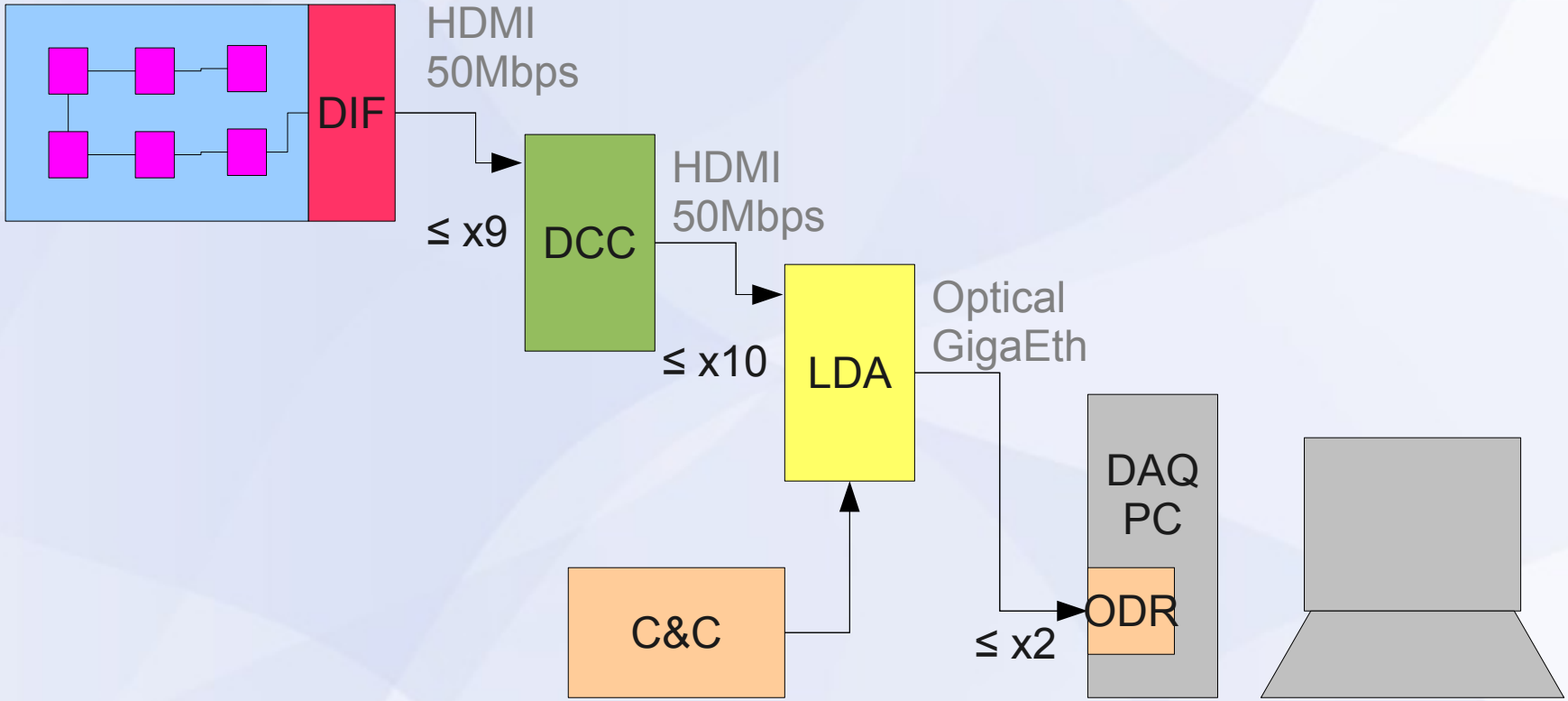
DAQ Software Status and Simulation

David Decotigny (LLR)
First part in collaboration with Valeria Bartsch (UCL)

Overview

- DAQ Software status
- DAQ Read-out: a simulation framework

EUDET DAQ System overview



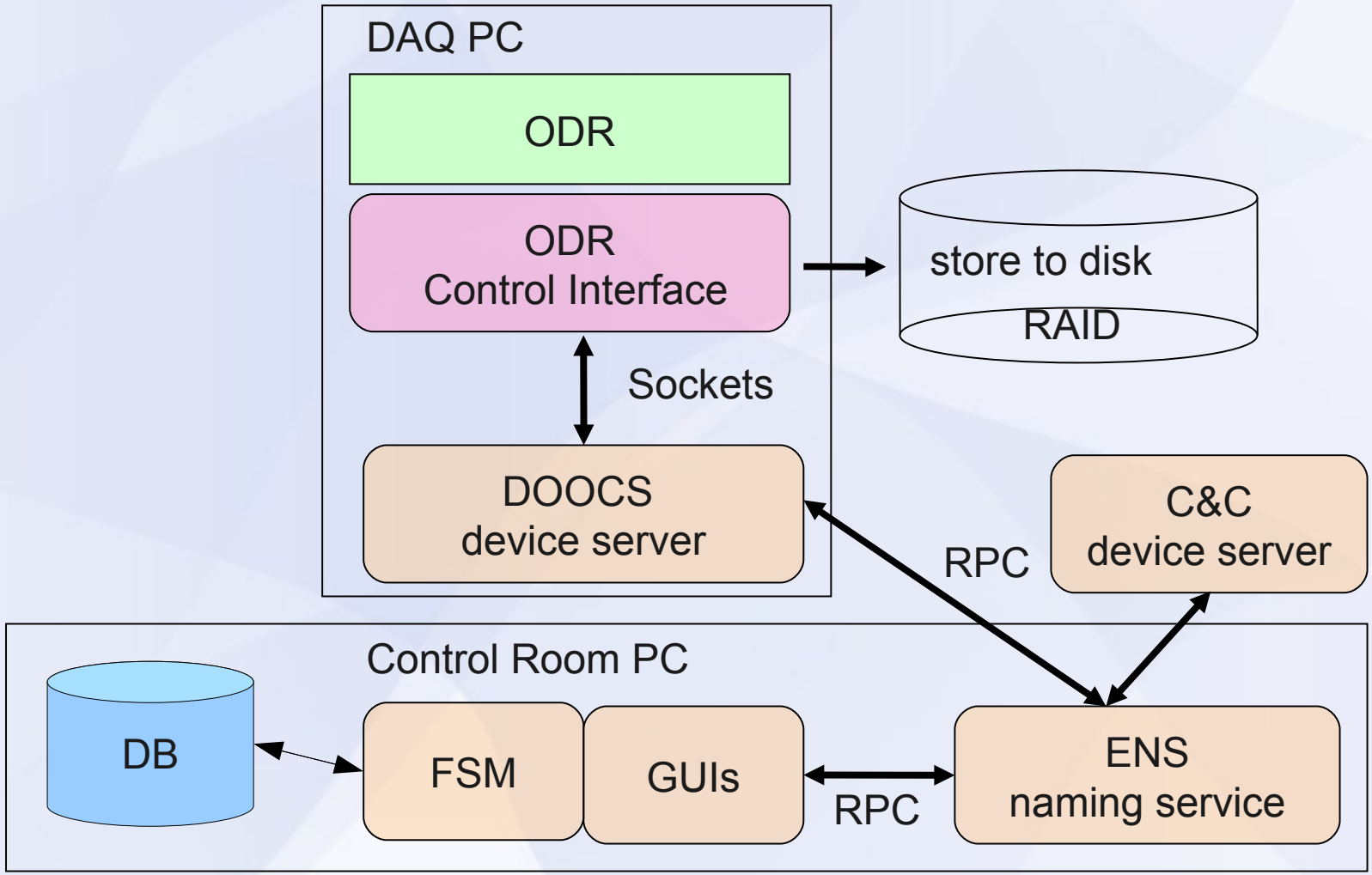
Hardware/Firmware status

- LDA+ODR+C&C+DAQ PC:
  - Hardware Provided to LLR by UCL/RHUL/Manchester
  - Firmware developed by UCL/RHUL/Manchester
- DCC:
  - Hardware available @LLR
  - Firmware developed by F. Gastaldi
- DIF:
  - Several prototypes available @LLR
  - DIF Task Force for the firmware

Software status

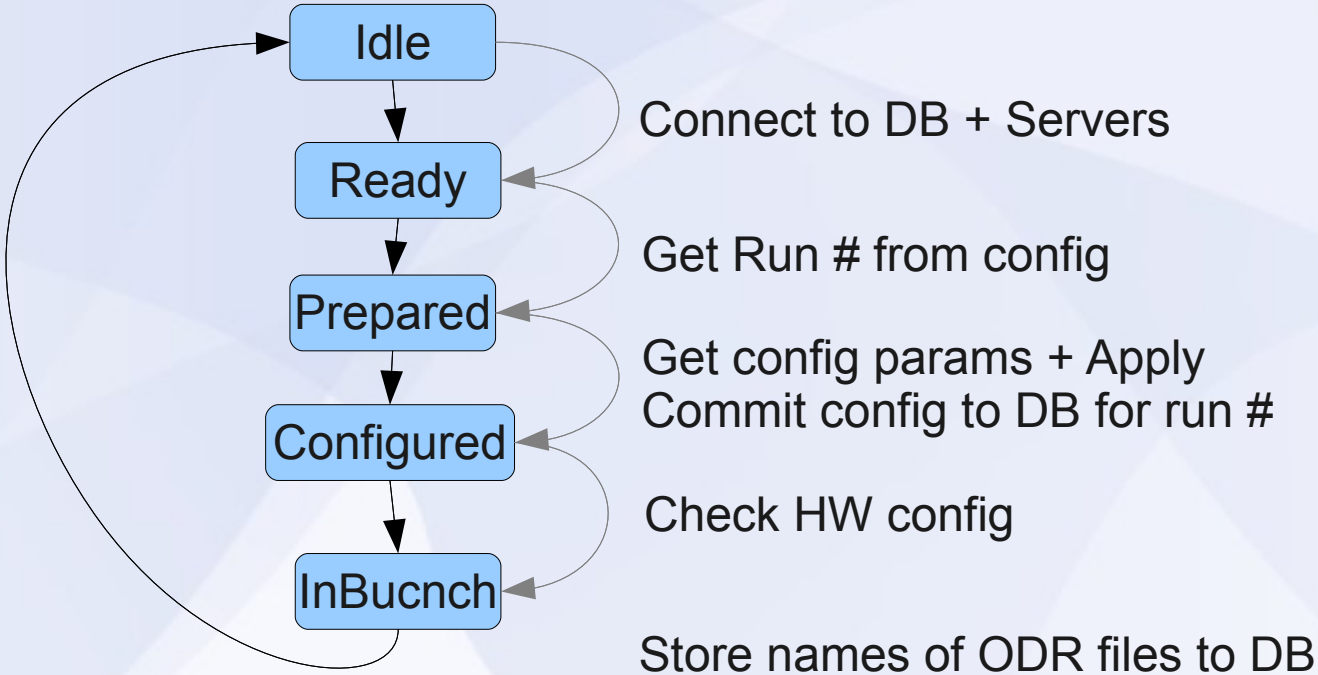
- Prototype running on DOOCS
  - ODR kernel driver by A. Misiejuk (Manchester)
    - Able to store data packets to disk
  - CCC interface for DOOCS
    - Register accesses
  - DAQ Software by V. Bartsch (UCL)
    - GUIs to control the ODR + CCC
    - FSM to control the device servers
    - DB framework to retrieve configs

DAQ SW architecture

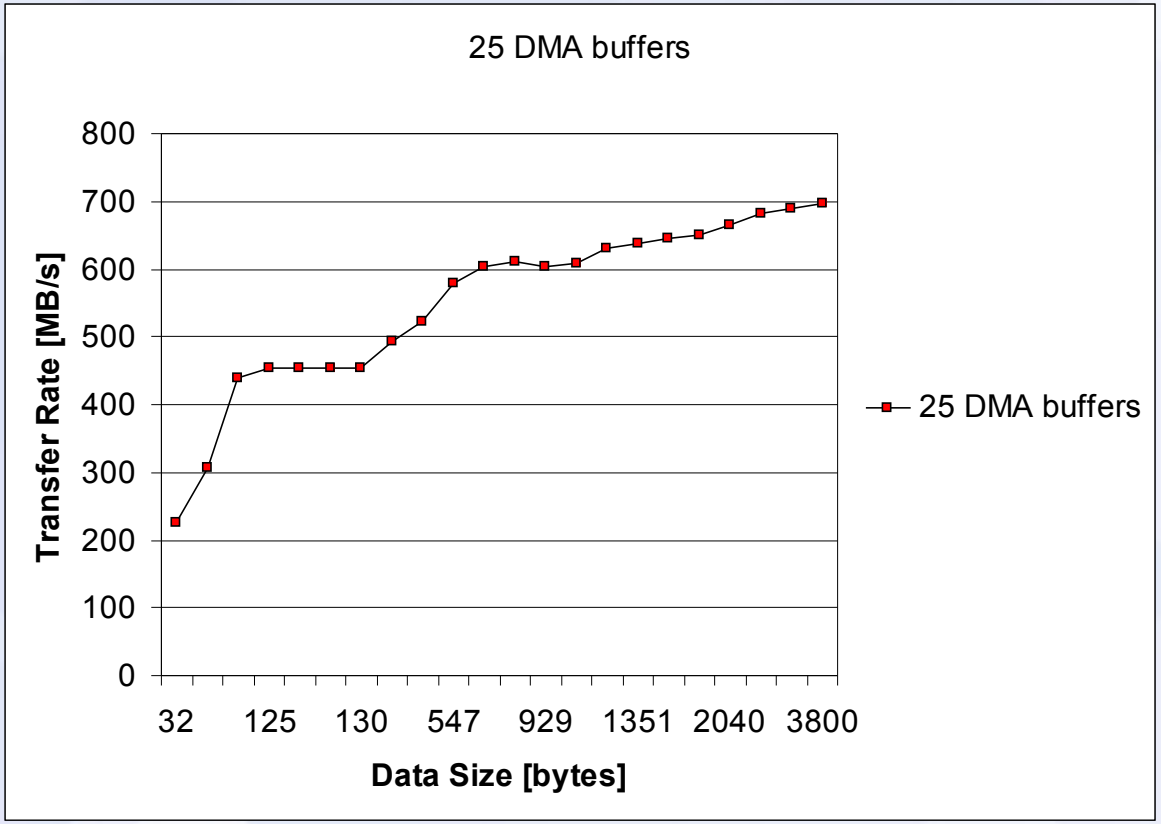


DAQ SW: FSM & States

- Current implementation: 2-level hierarchical FSM
  - "Super" FSM + Device servers
- State Machines:



DAQ Prototype: Performances



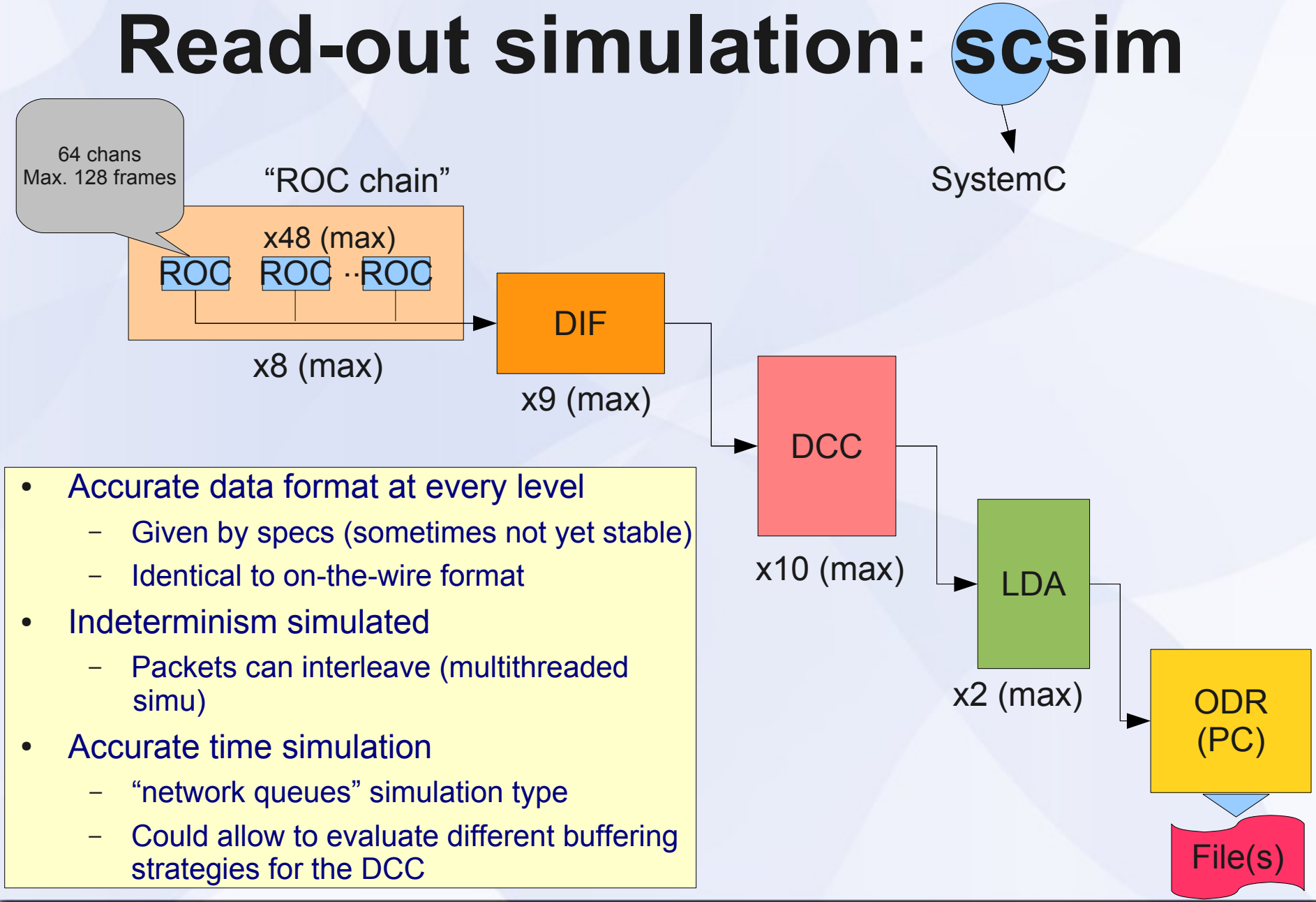
Evaluation by V. Bartsch/T. Wu/UCL/Manchester

DAQ Prototype: Perspectives

- Integrate/debug the whole chain
  - Validate ODR/LDA connection
  - Validate LDA/DCC connection
  - Validate DCC/DIF connection
  - A few tools available @LLR (SFP modules)
    - Require a minor ODR firmware fix
- Assess the performances of read-out reassembly algos

A simulation of the read-out chain

Read-out simulation: scsim



- Accurate data format at every level
  - Given by specs (sometimes not yet stable)
  - Identical to on-the-wire format
- Indeterminism simulated
  - Packets can interleave (multithreaded simu)
- Accurate time simulation
  - "network queues" simulation type
  - Could allow to evaluate different buffering strategies for the DCC

Read-out simulation: scsim

- Example:
  - 1 ODR: 2 LDA * 2 DCC * 2 DIF * 7 chains * 48 ROCs

```
0 s: send read-out signal...
Received pkt (1052 bytes) at 4312616 ns
Received pkt (1052 bytes) at 4312616 ns
Received pkt (1052 bytes) at 4312616 ns
Received pkt (1052 bytes) at 4312616 ns
Received pkt (1052 bytes) at 4321032 ns
Received pkt (1052 bytes) at 4321032 ns
Received pkt (1052 bytes) at 4321032 ns
Received pkt (1052 bytes) at 4321032 ns
Received pkt (1052 bytes) at 4415216 ns
...
```

1 simulated second (5 read-outs)
34160 ODR packets
File = 39MB

  - Full setup: 878400 packets, File = 989MB

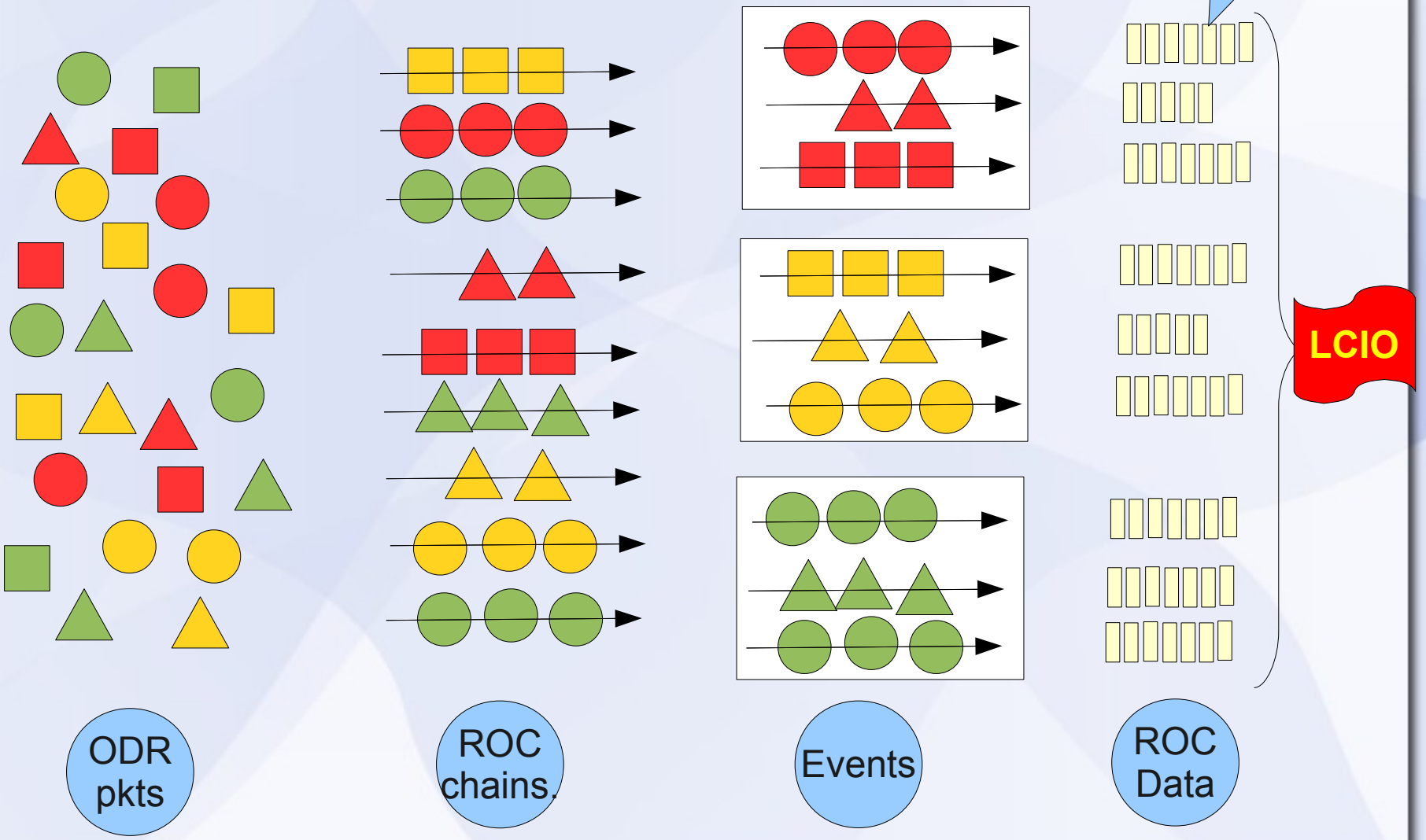
Read-out simulation: reassembler

- 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)
  - For now: HR2 format only, but should be easily generalized (Spiroc, Skiroc, ...)

Read-out simulation: reassembler

- 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)

Read-out simulation: reassembler



Read-out simulation: reassembler

- 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)

Read-out simulation: reassembler

- Calice packet formats described in
  https://svn.in2p3.fr/calice/online-sw/trunk/daq/calice_packets/calice_raw_formats.h?view=markup
- LCIO Format detailed in
  https://svn.in2p3.fr/calice/online-sw/trunk/daq/reassembler/lcio_dump.hpp?view=markup

(Open ?) questions

- Online event-reconstruction + LCIO conversion/storage realistic ?
- Use a common DAQ framework ?
  - ➔ Integrate smoothly with the existing DAQ chains:
    - C. Combaret @IPNL (Xdaq)
    - EUDAQ
    - Si tracking DAQ
    - ALICE-based
    - Other testbeams
    - ....
  - – EUDAQ ? Xdaq ? DOOCS ? Tango ? PVSS ? Custom ?

Backup slides

Software: read-out simulation

- In svn: online-sw/trunk/daq (https://svn.in2p3.fr/calice/online-sw/trunk/daq/)
  - calice_packets: C++ library to parse/generate ODR/LDA/DCC/DIF (+USB) read-out packets
  - scsim: SystemC simulation to accurately simulate readout on ROC → DIF → DCC → LDA → ODR → PC
  - reassembler: C++ framework to reorder/reassemble ODR packets and save them in LCIO format with a simple structure "run/event/DIF/ROC_chain data"

DAQ Framework: Tango ?

- http://www.tango-controls.org/
- Used by many synchrotron exp. (ESRF, DESY, Soleil…)
- But: the base is NOT synchrotron-specific
- Generic DAQ distributed system (slow-control)
  - Distributed configuration stored in DB
  - Service interaction through an ORB (~ RPC bus)
- Nothing really technically original, but more modern, nicer, more mature, less DESY-centric than Doocs

DAQ Framework: Tango ?

- The things that made me enthusiastic:
  - A real 3rd-party blob: we are *users* of the framework, we don't go *inside* the framework
  - Simple generic & composable message data types between services
  - Reasonably language-agnostic: C++, java, python
  - Can use jddd (Doocs GUI builder)
  - Management can be achieved via GUIs
  - A fast “device server design” approach (GUI code-generator)

DAQ Framework: Tango ?

- Tango or not Tango ?
  - Generated only 3 device servers (C++, java, python) and played with them
  - Had a positive experience with the Tango mailing-list
- To be continued ?...