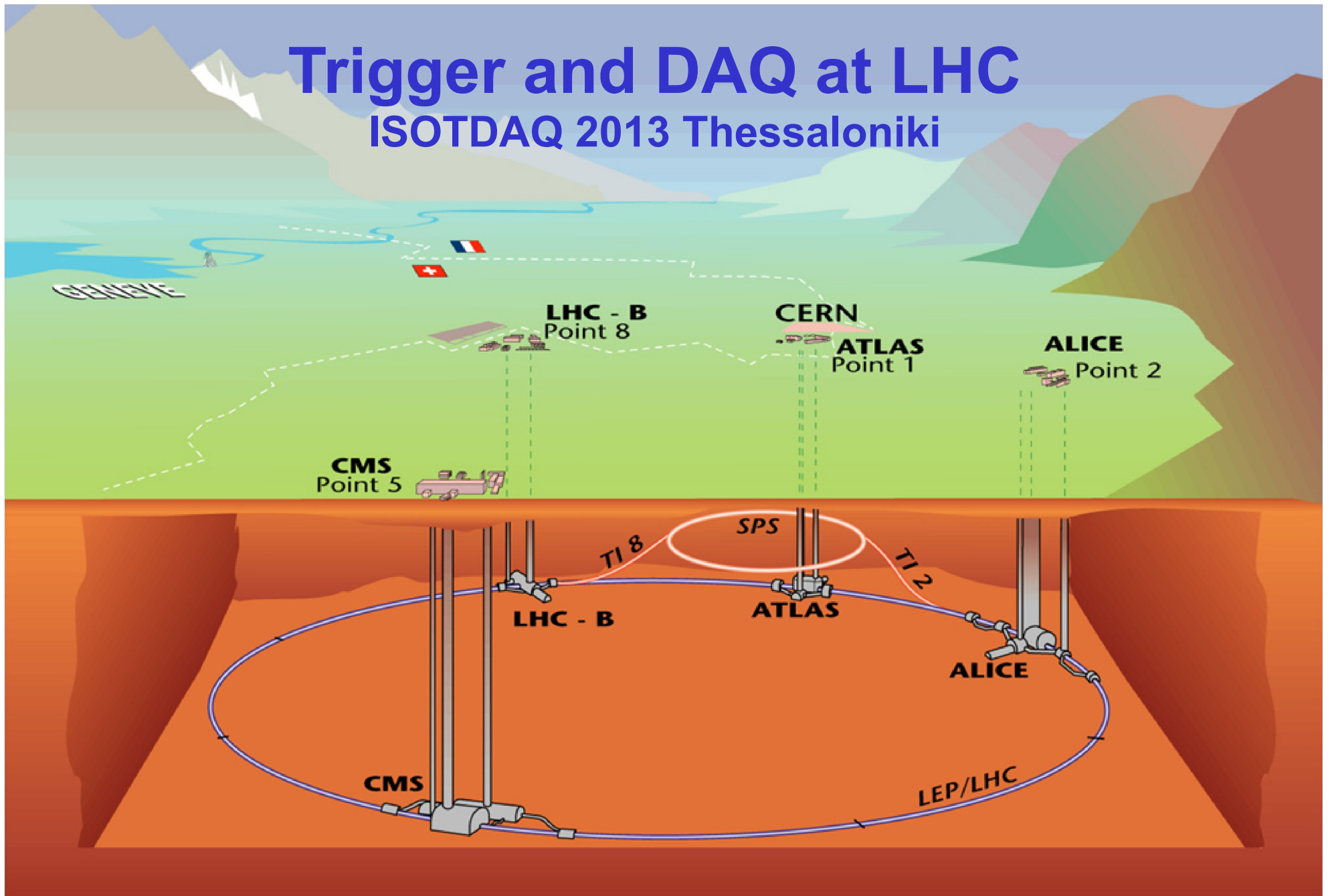


# Trigger and DAQ at LHC

## ISOTDAQ 2013 Thessaloniki

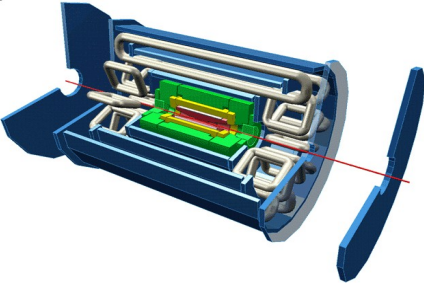
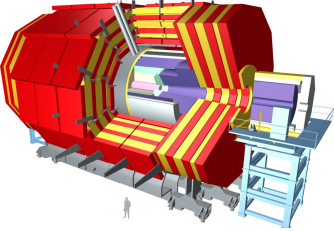
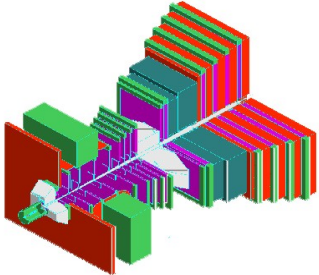
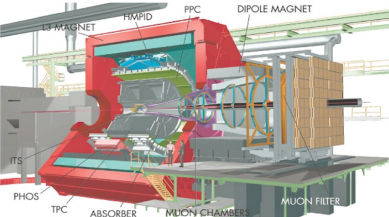


# Contents

---

- **Introduction:**
  - The context: LHC & experiments
- **Part 1: Trigger at LHC (hardware trigger)**
  - Requirements & Concepts
  - Muon and Calorimeter triggers (CMS and ATLAS)
  - Specific solutions (ALICE, LHCb)
  - Ongoing and future upgrades
- **Part2: Readout Links, Dataflow, and Event Building**
  - Data Readout (Interface to DAQ)
  - Data Flow of the 4 LHC experiments
  - Event Building: CMS as an example
  - Software: Some techniques used in online systems
  - Ongoing and future upgrades
- **Acknowledgement**
  - Thanks to many of my colleagues in ALICE, ATLAS, CMS, LHCb for the help they gave me while preparing these lectures; and in particular to Sergio Cittolin who provided me with many slides (probably those you will like most are from him!)

# Trigger/DAQ parameters

	No.Levels	Lvl 0,1,2	Event	Evt Build.	High Level Trigger
	Trigger	Rate (Hz)	Size (Byte)	Bandw.(GB/s)	HLT Out MB/s (Event/s)
	<b>3</b>	LV-1 <b>105</b> LV-2 <b>3x103</b>	<b>1.5 MB</b>	<b>4.5</b>	<b>300</b> (200)
	<b>2</b>	LV-1 <b>105</b>  Pb-Pb 1500MB/s	<b>1.0 MB</b>	<b>100</b>	<b>300</b> (200)
	<b>2</b>	LV-0 <b>106</b>	<b>30 kB</b>	<b>30</b>	<b>60</b> (2 kHz)
	<b>4</b>	Pb-Pb <b>500</b> p-p <b>103</b>	<b>70 MB</b> <b>2 MB</b>	<b>25</b>	<b>1250</b> (100) <b>200</b> (100)

---

# Data Flow: Architecture

# Data Acquisition:

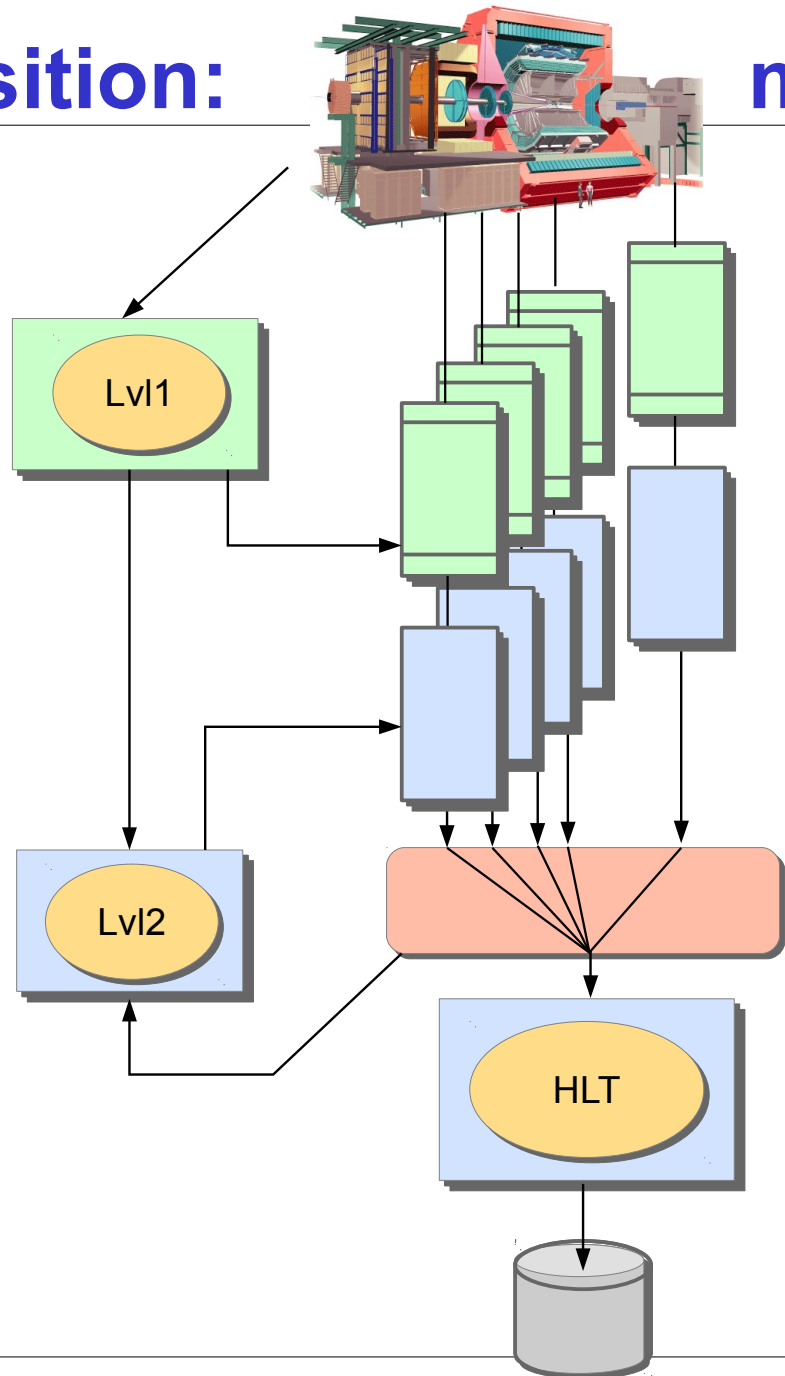
# main tasks

- custom hardware
- PC
- network switch

hardware trigger

Provide higher level trigger with partial event data

Our "Standard Model" of Data Flow



Lv1 pipelines

Data readout from Front End Electronics

Temporary buffering of event fragments in **readout buffers**

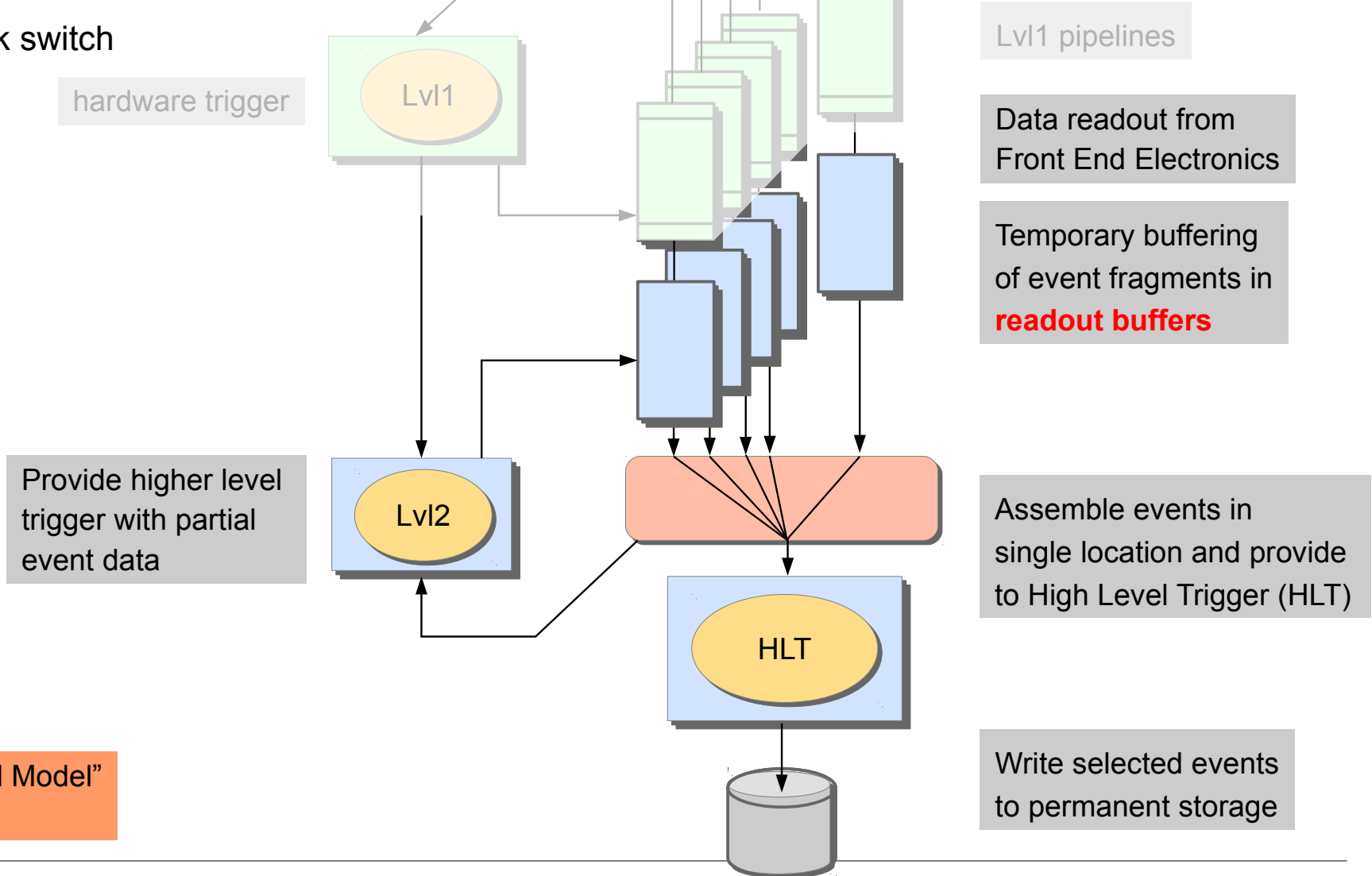
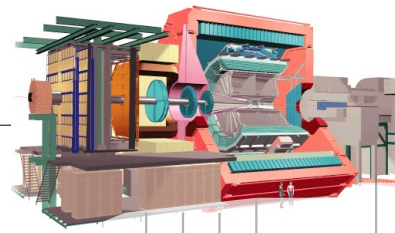
Assemble events in single location and provide to High Level Trigger (HLT)

Write selected events to permanent storage

# Data Acquisition:

# main tasks

- custom hardware
- PC
- network switch

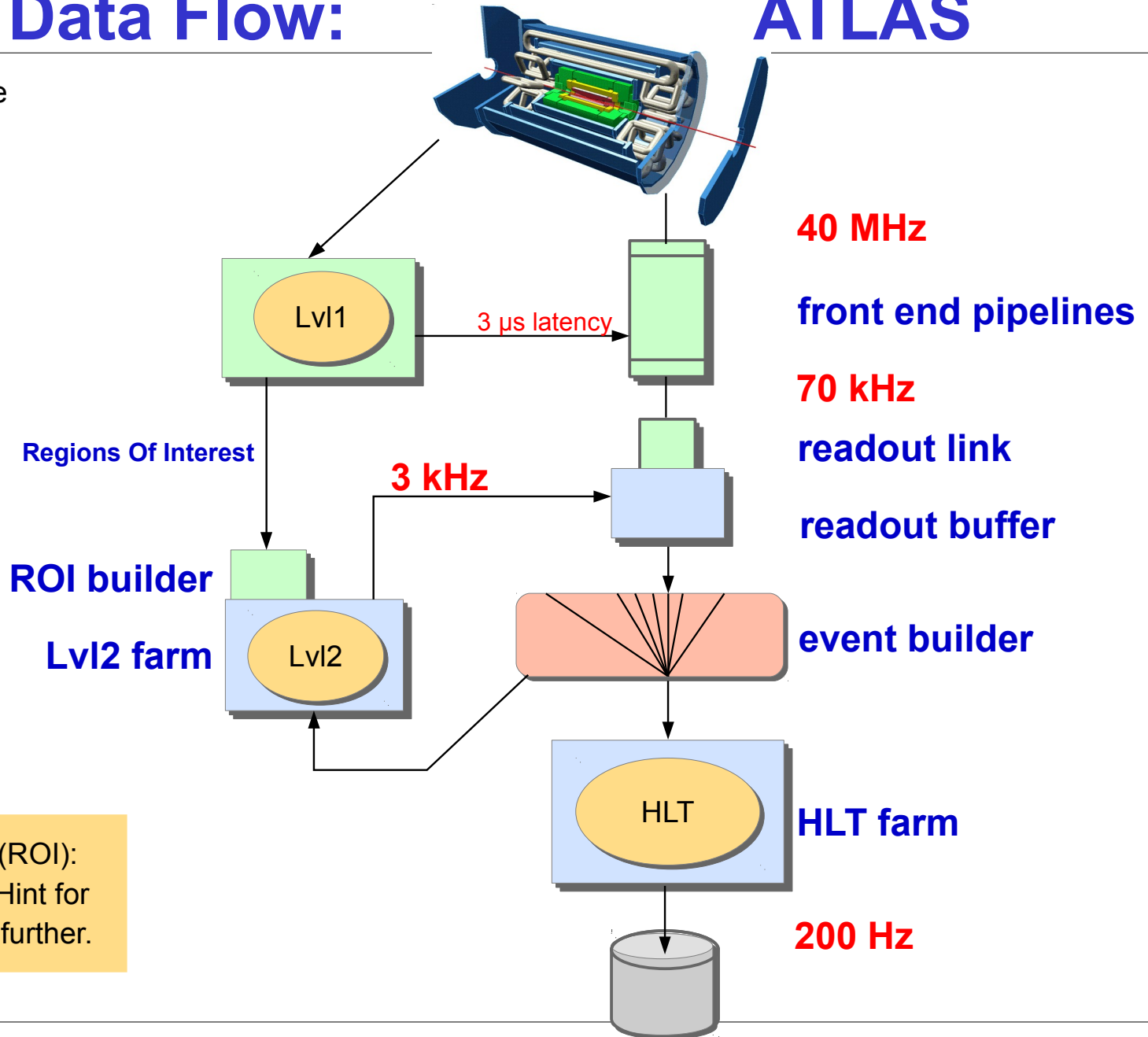


Our "Standard Model" of Data Flow

# Data Flow:

# ATLAS

- custom hardware
- PC
- network switch

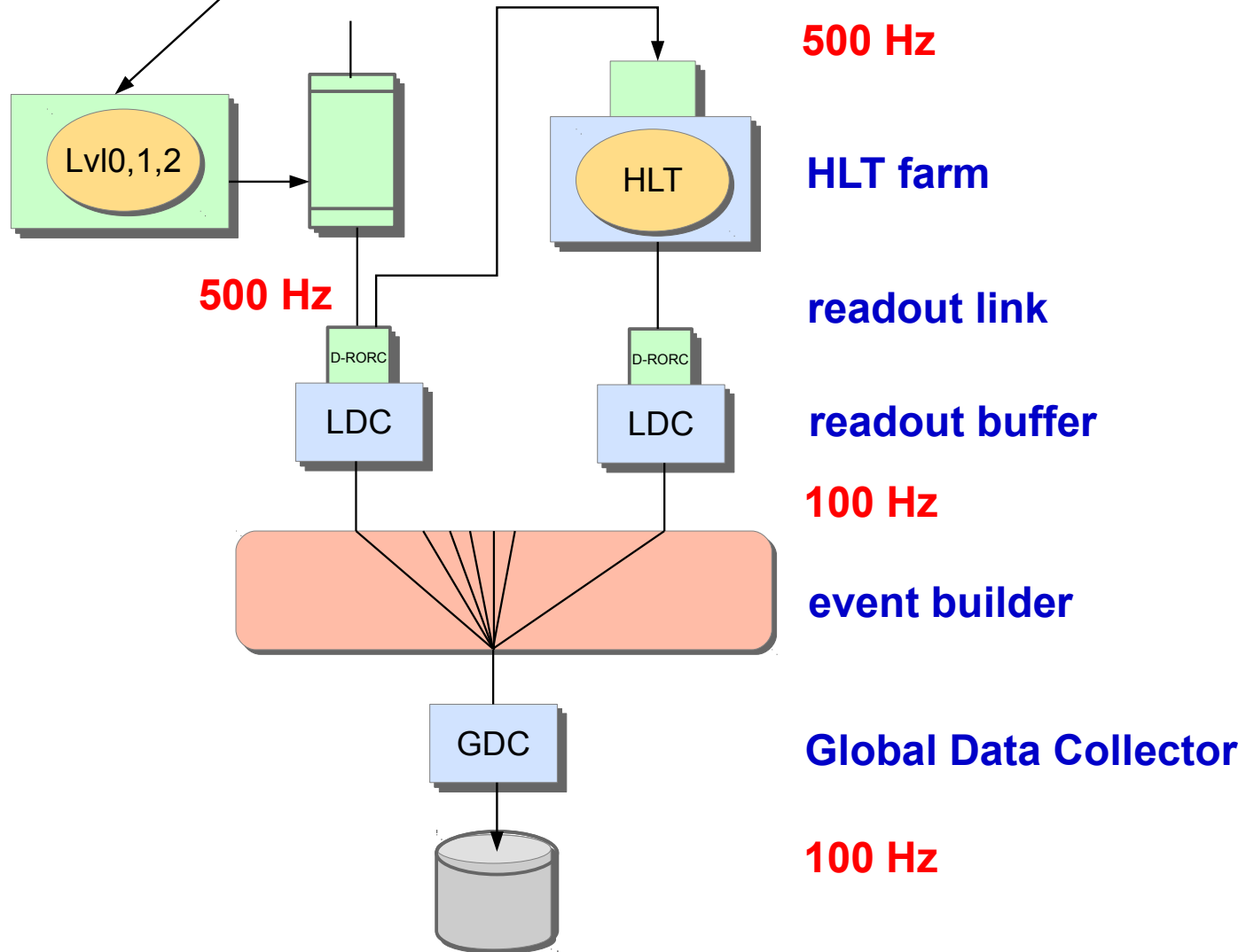
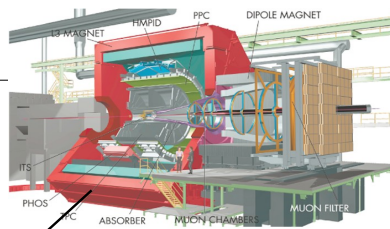


Region of Interest (ROI):  
Identified by Lvl1. Hint for  
Lvl2 to investigate further.

# Data Flow:

# ALICE

- custom hardware
- PC
- network switch

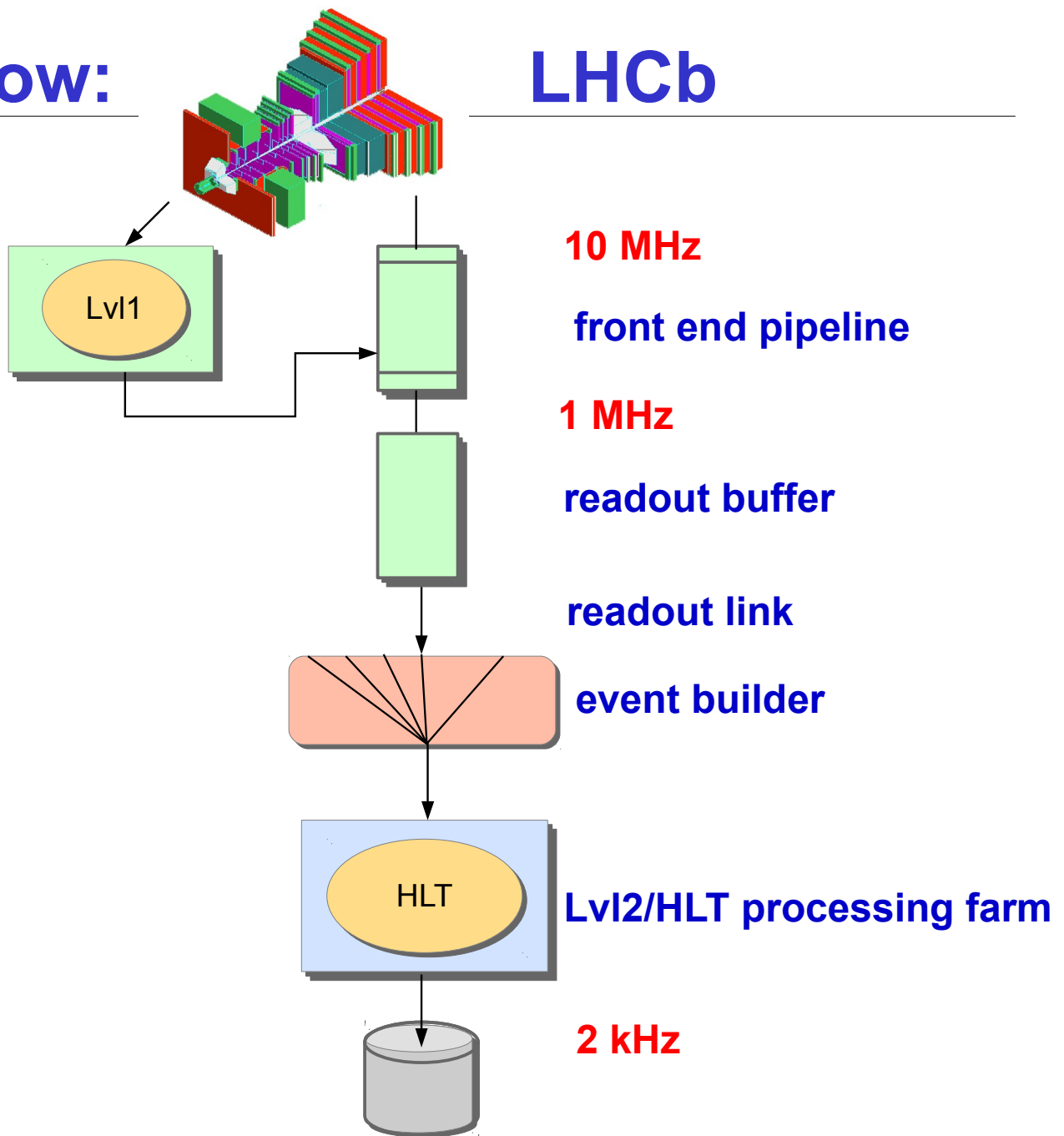




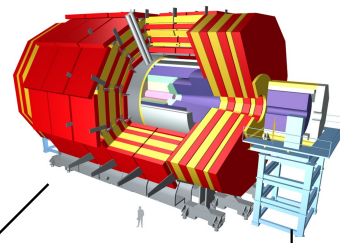
# Data Flow:

# LHCb

- custom hardware
- PC
- network switch

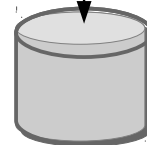
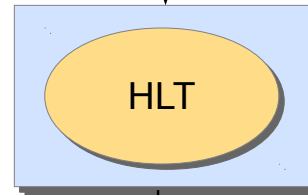
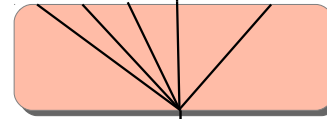
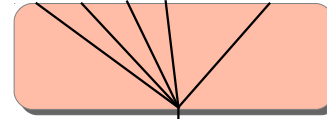
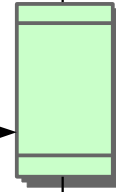
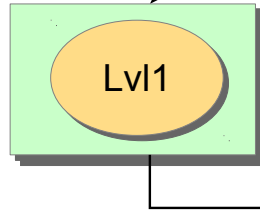


# Data Flow:



# CMS

- custom hardware
- PC
- network switch



**40 MHz**

**Front end pipeline**

**100 kHz**

**readout link**

**event builder: stage 1**

**readout buffer**

**event builder: stage 2**

**100 kHz**

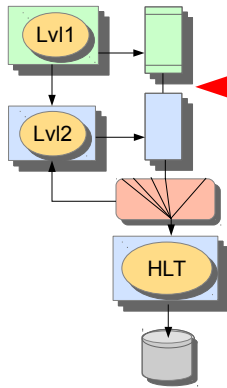
**HLT processing farm**

**1kHz**

---

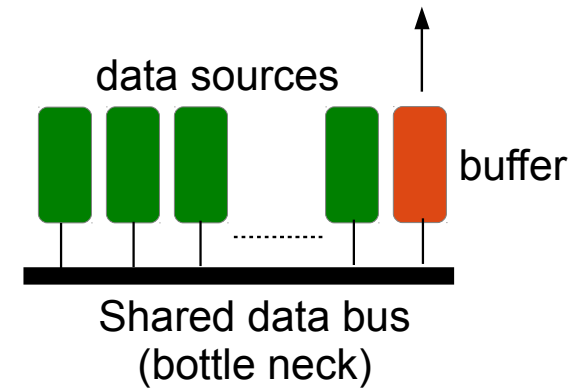
# Data Flow: Readout Links

# Data Flow: Data Readout



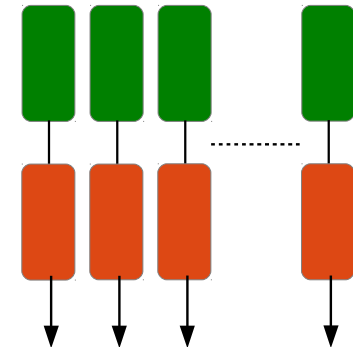
- **Former times: Use of bus-systems**

- VME or Fastbus
- Parallel data transfer (typical: 32 bit) on shared bus
- One source at a time can use the bus



- **LHC: Point to point links**

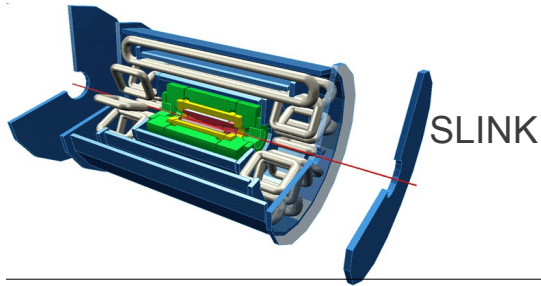
- Optical or electrical
- Data serialized
- Custom or standard protocols
- All sources can send data simultaneously



- **Compare trends in industry**

- 198x : ISA, SCSI(1970), IDE, parallel port, VME(1982)
- 199x : PCI(1990, 66Mhz 1995), USB(1996), FireWire(1995)
- 200x : USB2, FireWire 800, PCIExpress, Infiniband, GbE, 10GbE

# Readout Links of LHC Experiments

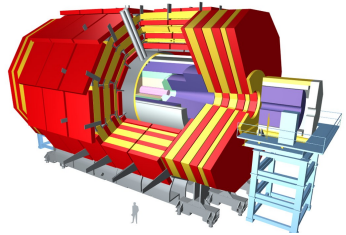


SLINK

Optical: 160 MB/s ≈ 1600 Links  
Receiver card interfaces to PC.

## Flow Control

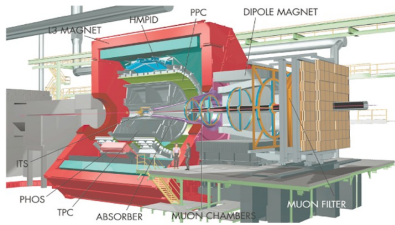
Yes



SLINK 64

LVDS: 400 MB/s (max. 15m) ≈ 500 links  
(FE on average: 200 MB/s to readout buffer)  
Receiver card interfaces to commercial NIC (Network Interface Card)

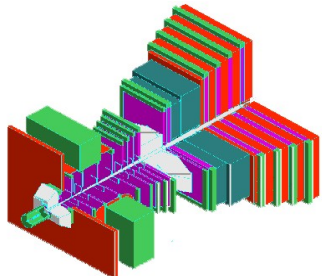
yes



DDL

Optical 200 MB/s ≈ 500 links  
Half duplex: Controls FE (commands, Pedestals, Calibration data)  
Receiver card interfaces to PC

yes



TELL-1  
& GbE Link

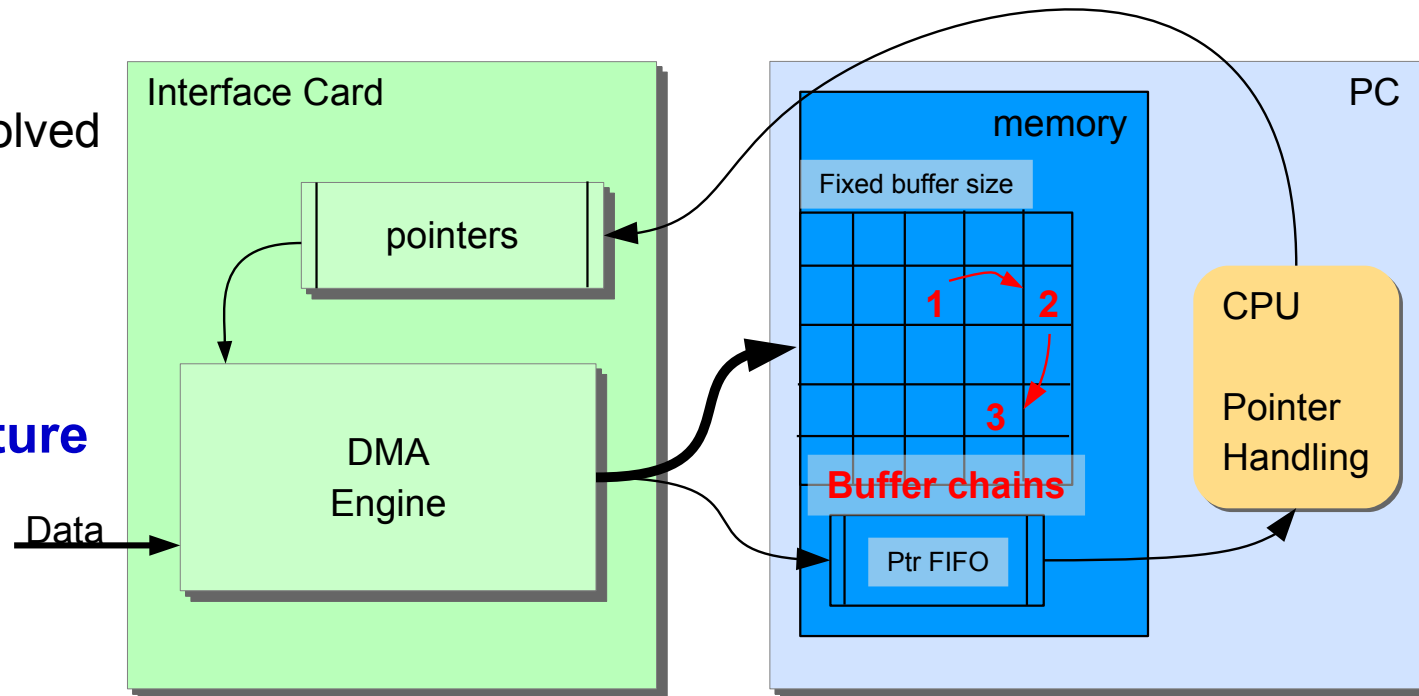
Copper quad GbE Link ≈ 400 links  
Protocol: IPv4 (direct connection to GbE switch)  
Forms “Multi Event Fragments”  
Implements readout buffer

no

# Readout Links: Interface to PC

- **Problem:**
  - Read data in PC with high bandwidth and low CPU load
  - Note: copying data costs a lot of CPU time!
- **Solution: Buffer-Loaning**
  - Hardware shuffles data via DMA (Direct Memory Access) engines
  - Software maintains tables of buffer-chains
- **Advantage:**
  - No CPU copy involved

used for links of  
**CMS, Alice,**  
**Atlas may be in future**



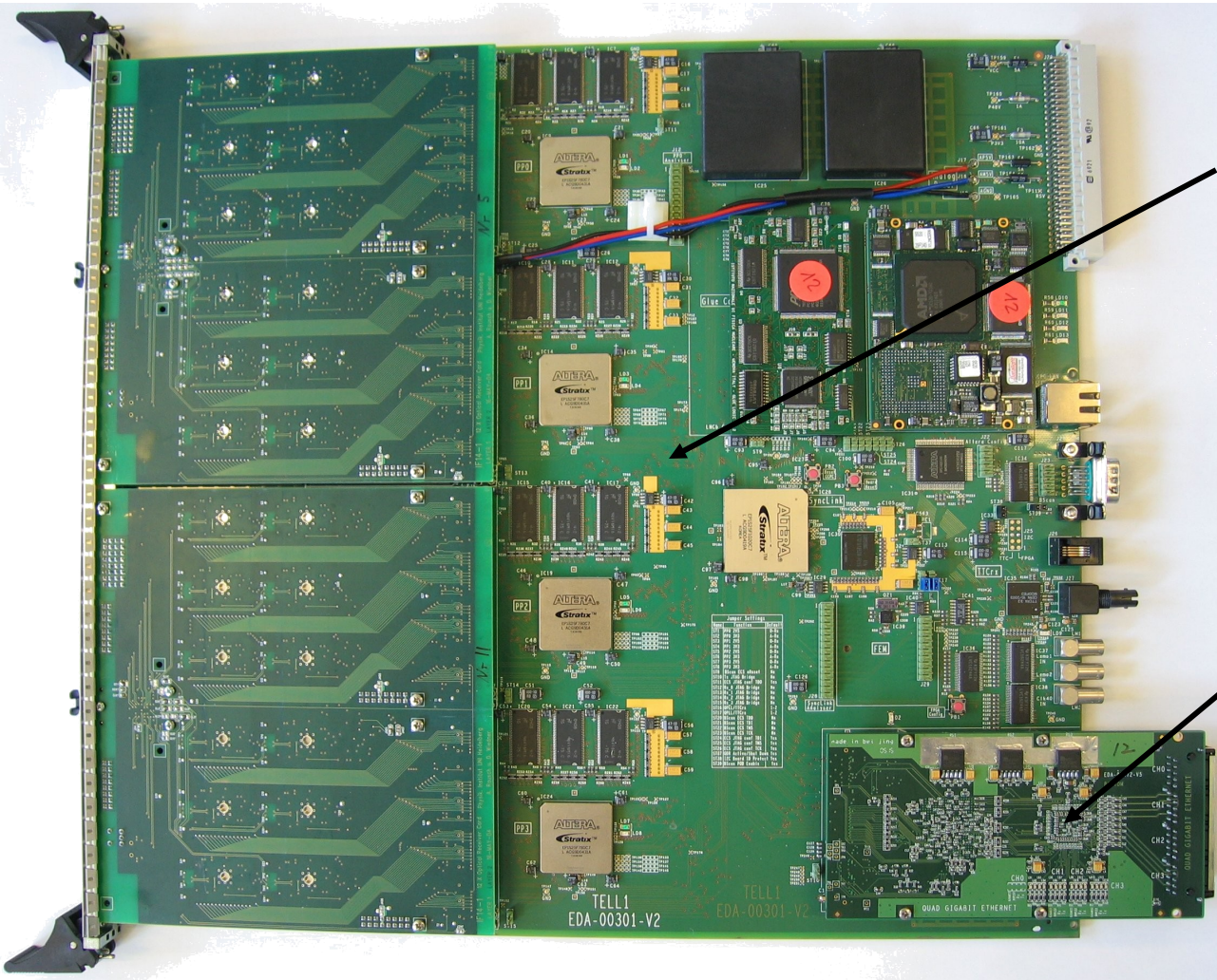
# Example readout board: LHCb

## Main board:

- data reception from “Front End” via optical or copper links.
- detector specific processing

## Readout Link

- “highway to DAQ”
- simple interface to main board
- Implemented as “plug on”

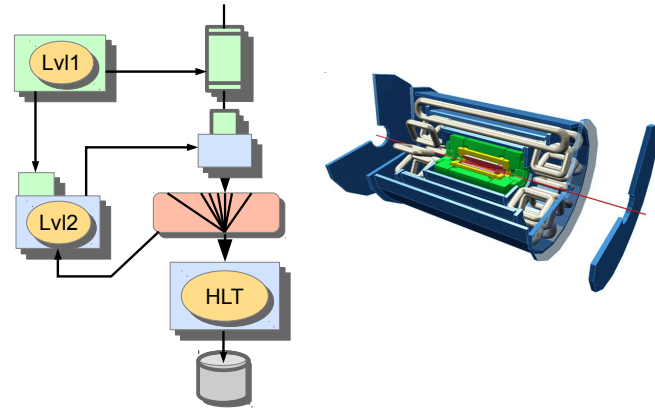


---

# Event Building: example CMS



# Event Building: Atlas vs CMS



**Challenging**

Concept of “Region Of Interest” (ROI)  
Increased complexity

- ROI generation (at Lv1)
- ROI Builder (custom module)
- selective readout from buffers

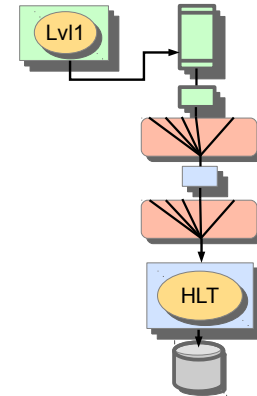
“Commodity”

1kHz @ 1 MB = O(1) GB/s

## Readout Buffer

“Commodity”

Implemented with commercial PCs



## Event Builder

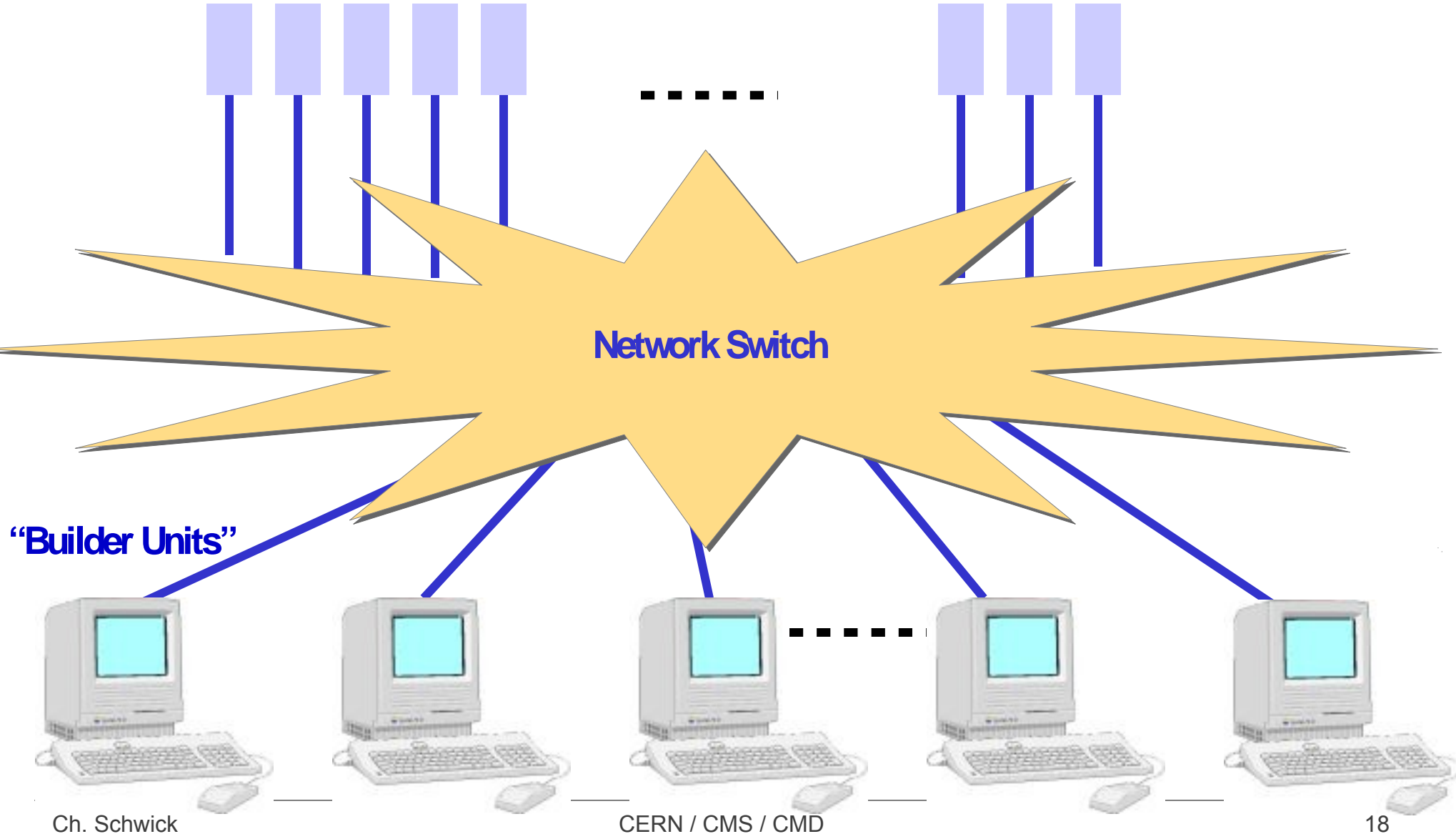
**Challenging**

100kHz @ 1 MB = 100 GB/s  
Increased complexity:

- traffic shaping
- specialized (commercial) hardware

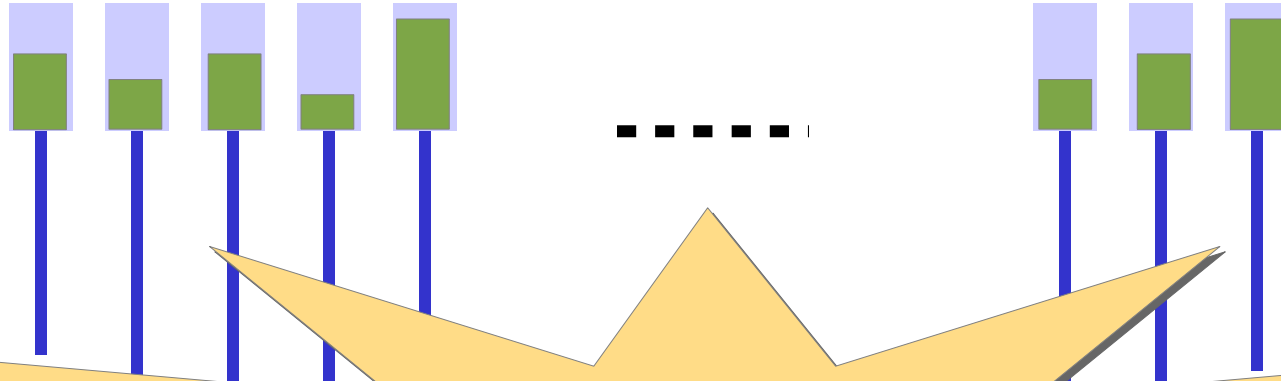
# Networking: EVB traffic

Readout Buffers



# Networking: EVB traffic

Readout Buffers



Network Switch

"Builder Units"



Ch. Schwick

CERN / CMS / CMD

# Networking: EVB traffic

Readout Buffers



Network Switch

"Builder Units"



Ch. Schwick

CERN / CMS / CMD

20

# Event Building dilemma

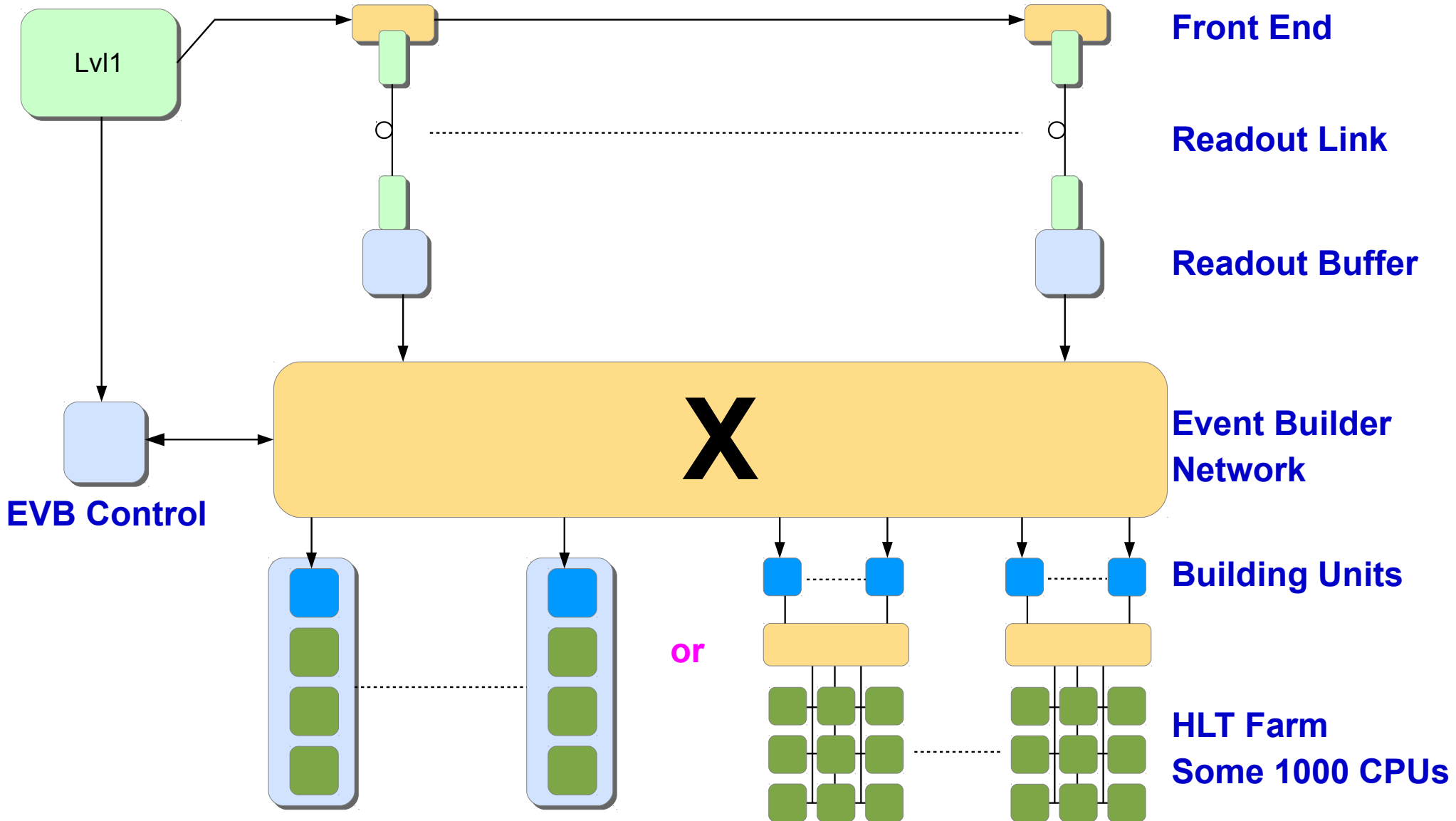
---



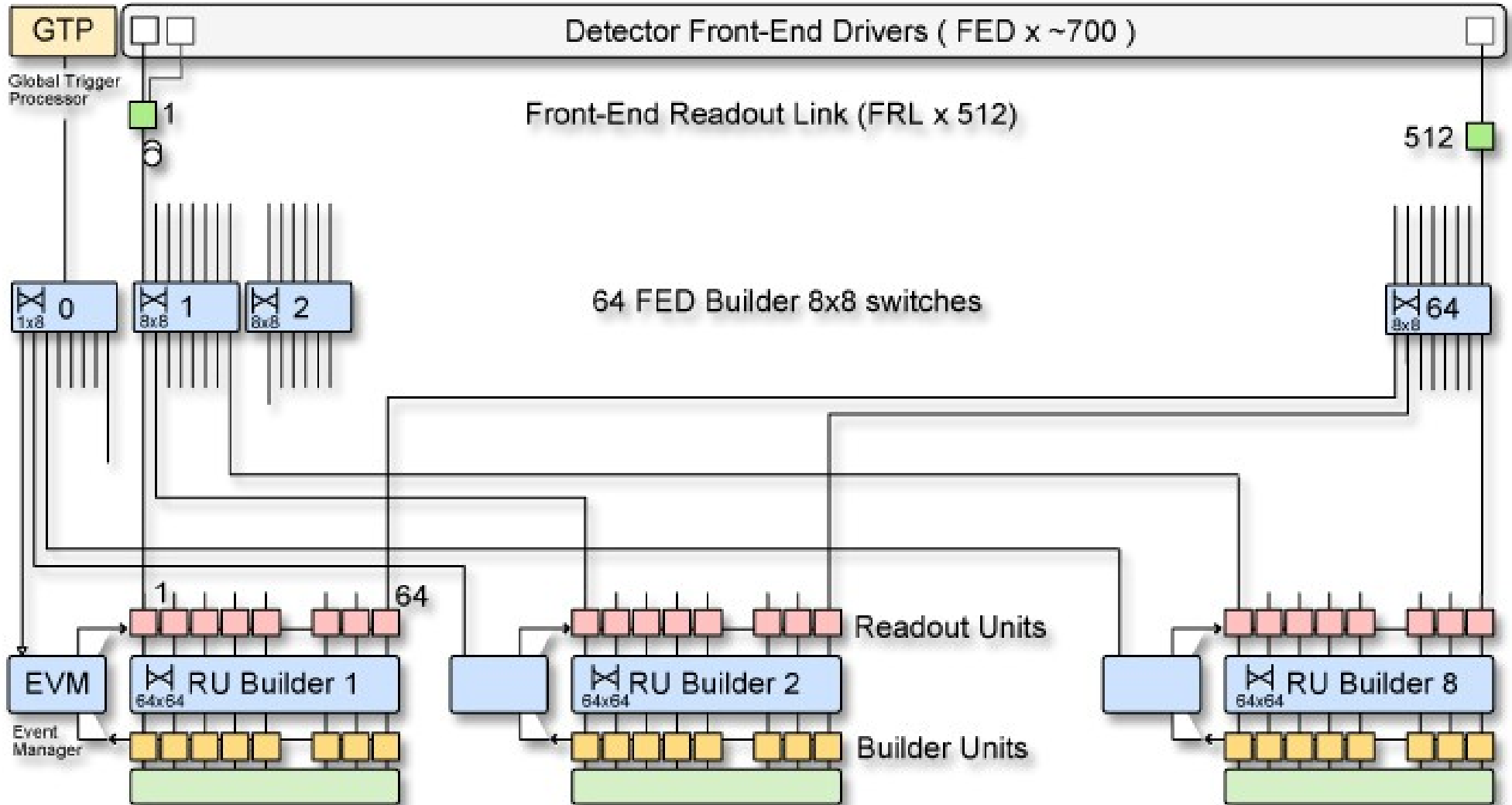
**For Event builder traffic pattern congestion is a problem...**



# Modern EVB architecture

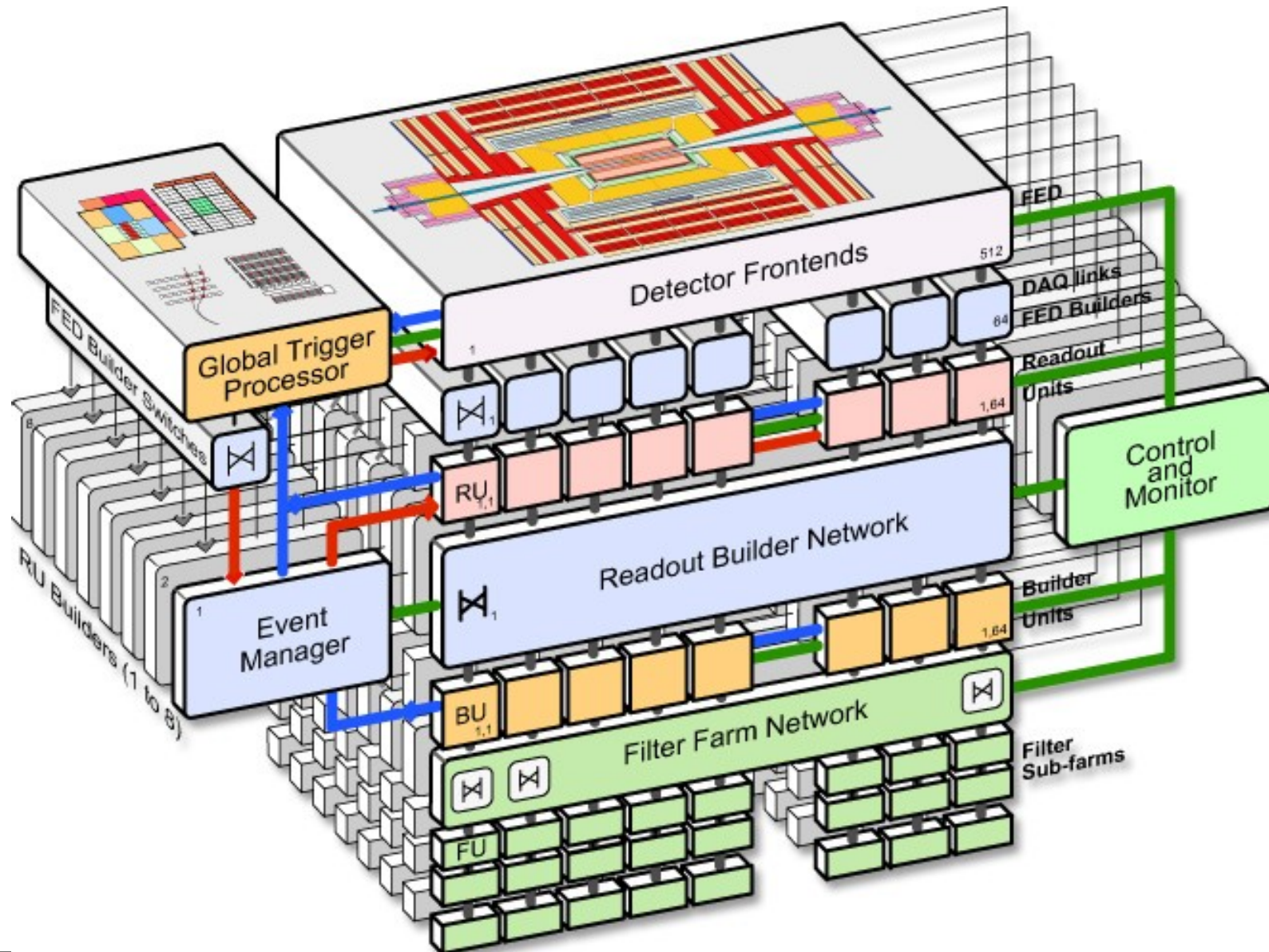


# EVB CMS: 2 stages





# CMS: 3D - EVB



# Advantages of 2 stages & “sliced” EVB

---

- **Relaxed requirements for 2nd stage:**
  - Every RU-Builder works at 12.5 kHz (instead of 100kHz)
- **Staging in time: building the system step by step**
  - To start up the experiment not the entire hardware needs to be present.  
Example:
    - If an Event Builder operating at 50 kHz is sufficient for the first beam, only 4 RU-builders need to be bought and set up.
- **Scalability**
- **Technology independence:**
  - The RU-Builder can be implemented with a different technology than the FED-Builder
  - Even different RU-Builders can be implemented with different technologies.
- **Redundancy**
  - In case of sever problems in a slice (e.g. with the storage system) you can just mask the slice and continue running (with only 12.5% less maximal performance)

---

**The first stage of the Event-Builder**

**“FEDBuilder”**

**FED = Front End Driver**

# Stage1: FED-Builder implementation

- **FED Builder functionality**

- Receives event fragments from approx. 8 Readout Links (FRLs).
- FRL fragments are merged into “super-fragments” at the destination (Readout Unit).

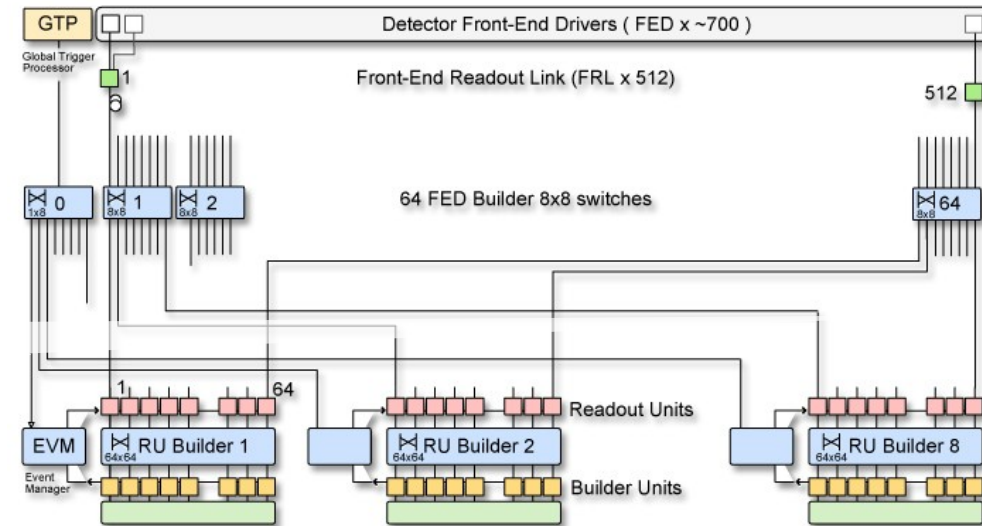
- **FED Builder implementation**

- **Requirements:**

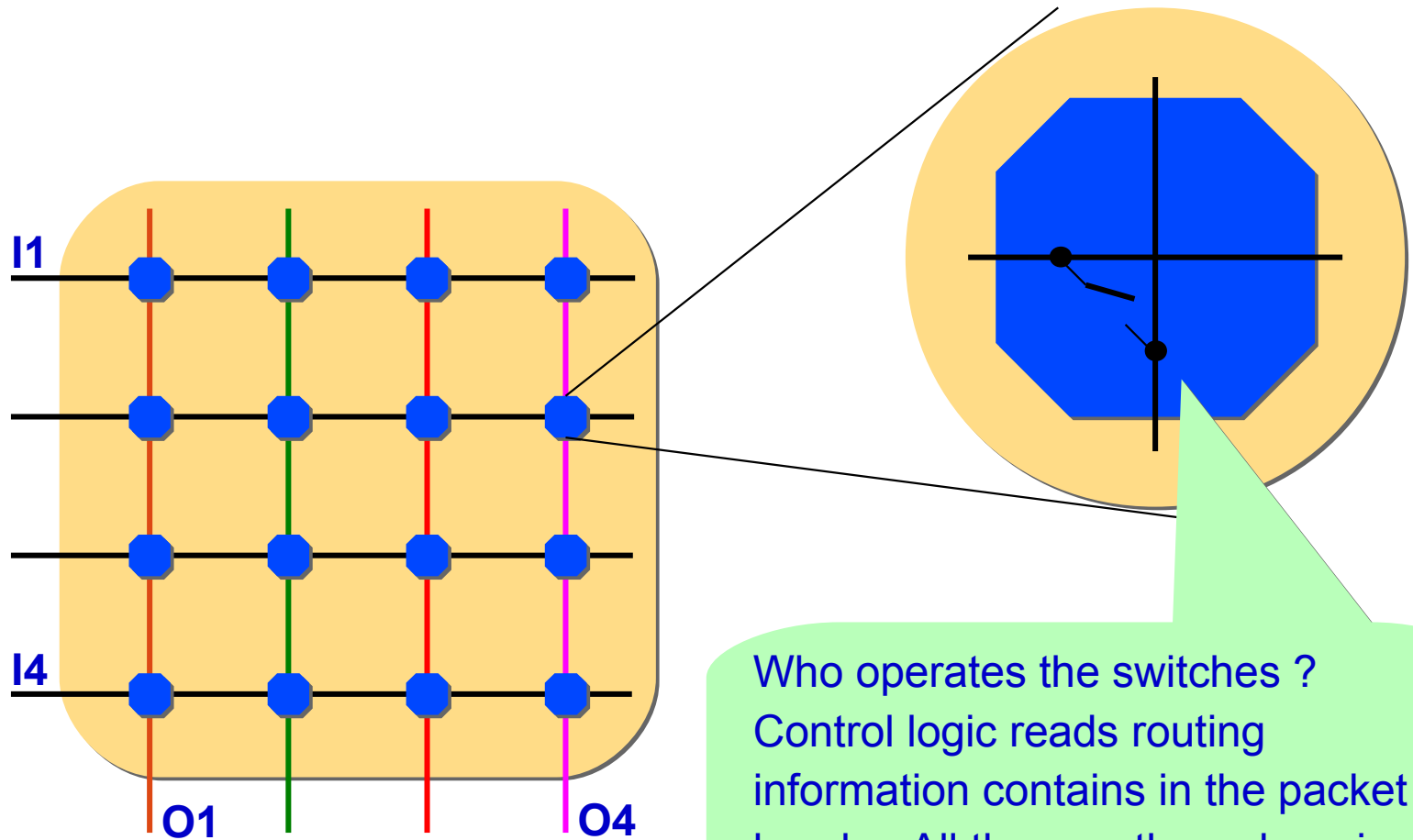
- Sustained throughput of 200MB/s for every data source (500 in total).
- Input interfaces to FPGA (in FRL) -> protocol must be simple.

- **Chosen network technology: Myrinet**

- NICs (Network Interface Cards) with 2x2.5 Gb/s optical links ( $\approx 2 \times 250$  MB/s)
- Full duplex with flow control (no packet loss).
- NIC cards contain RISC processor. Development system available.  
Can be easily interfaced to FPGAs (custom electronics: receiving part of readout links)
- Switches based on cross bars (predictable, understandable behavior).
- Low cost!



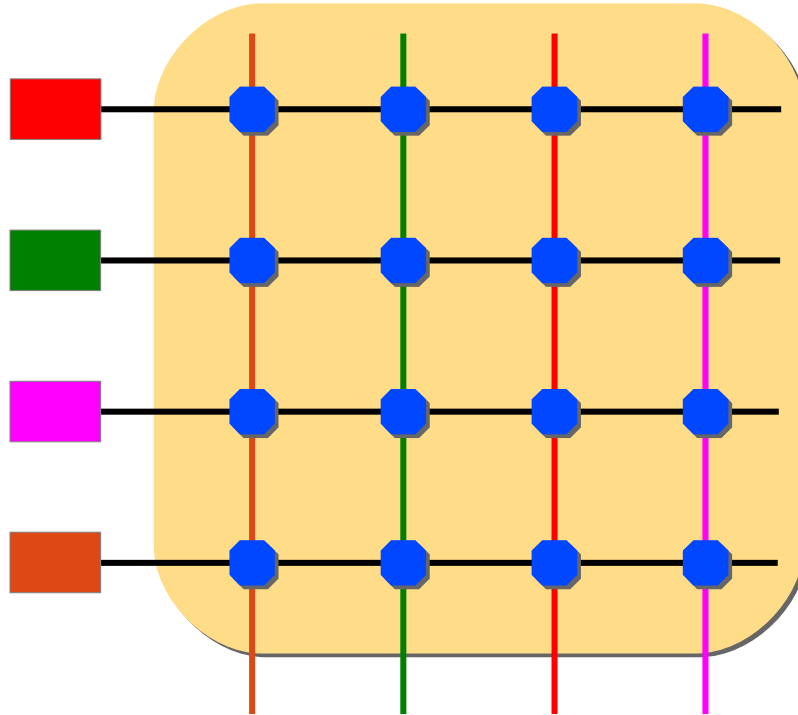
# Switch implementation: Crossbar



Every input/output has  
A given “wire speed”  
(here 2.5 Gb/s)

Who operates the switches ?  
Control logic reads routing  
information contains in the packet  
header. All the way through various  
layers of switches are defined there.

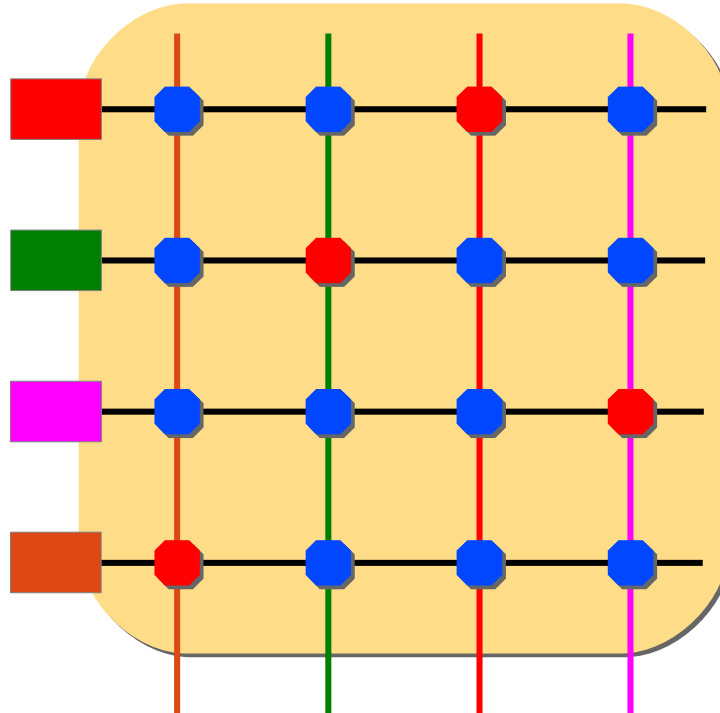
# Switch implementation: Crossbar



Best possible scenario:

no congestion since all packets find a free way through the switch.

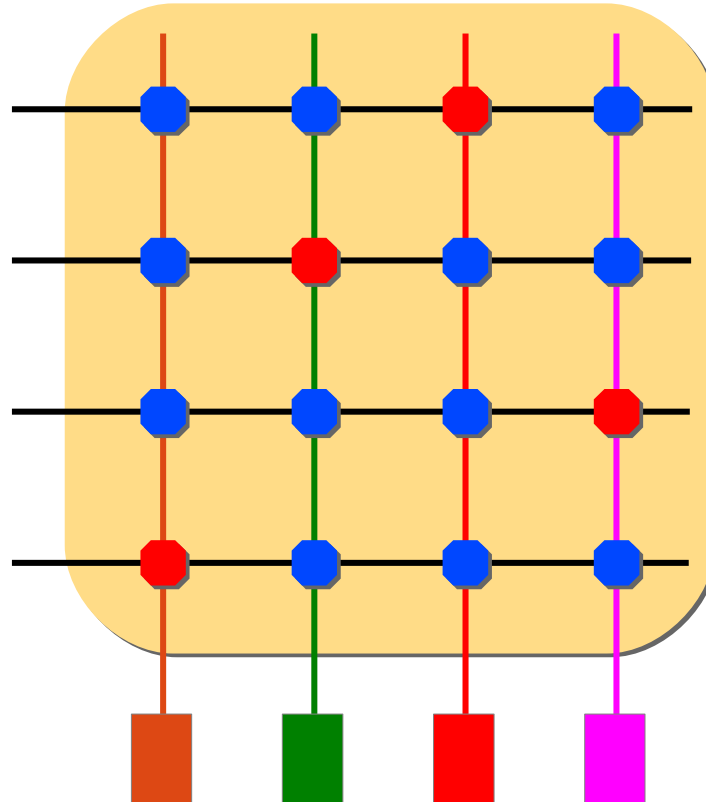
# Switch implementation: Crossbar



Best possible scenario:

no congestion since all packets find a free way through the switch.

# Switch implementation: Crossbar



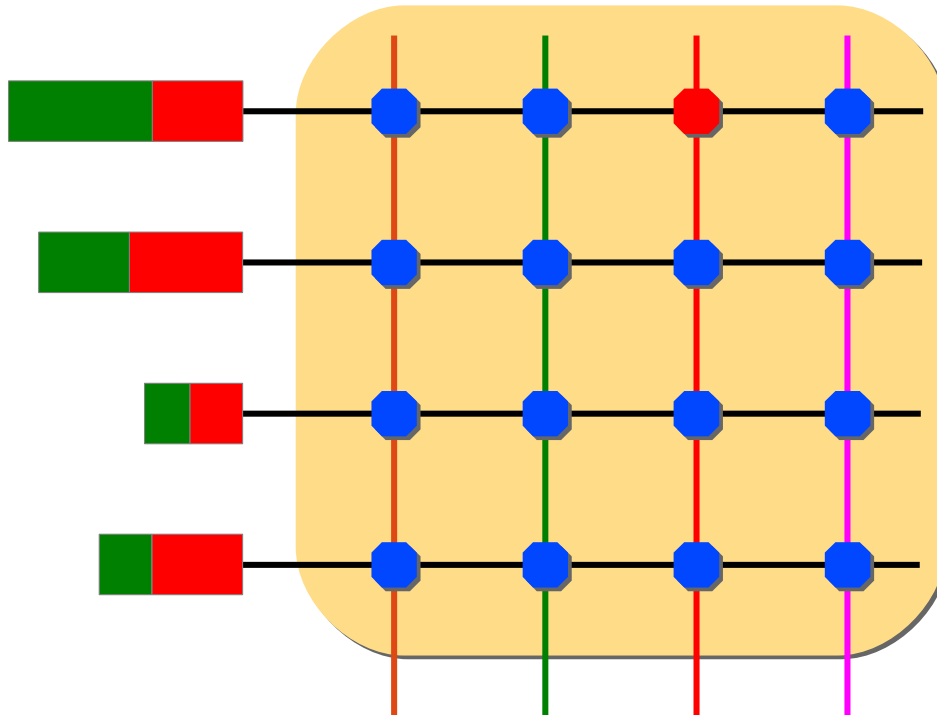
Best possible scenario:

Data traverses the switch at “wire speed”.



# Switch implementation: Crossbar

Crossbar switch: **A lot of congestion for typical EVB traffic**



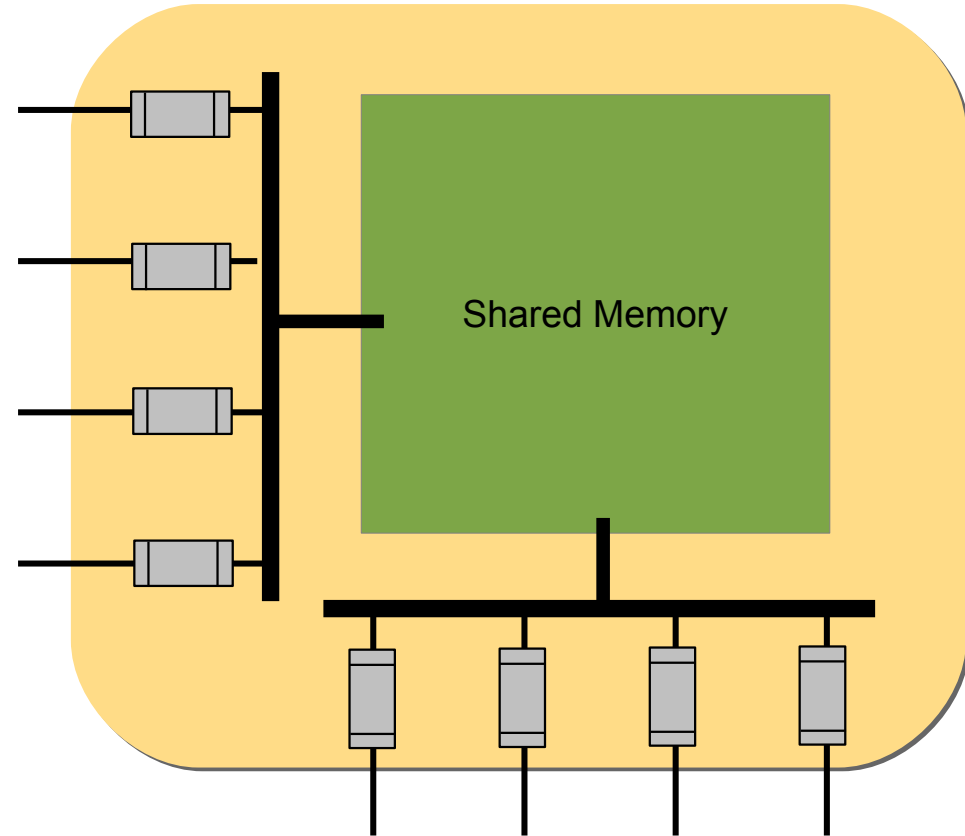
Only one packet at a time  
can be routed to the  
destination.

“Head of line” - blocking

# Alternative Switch Implementation

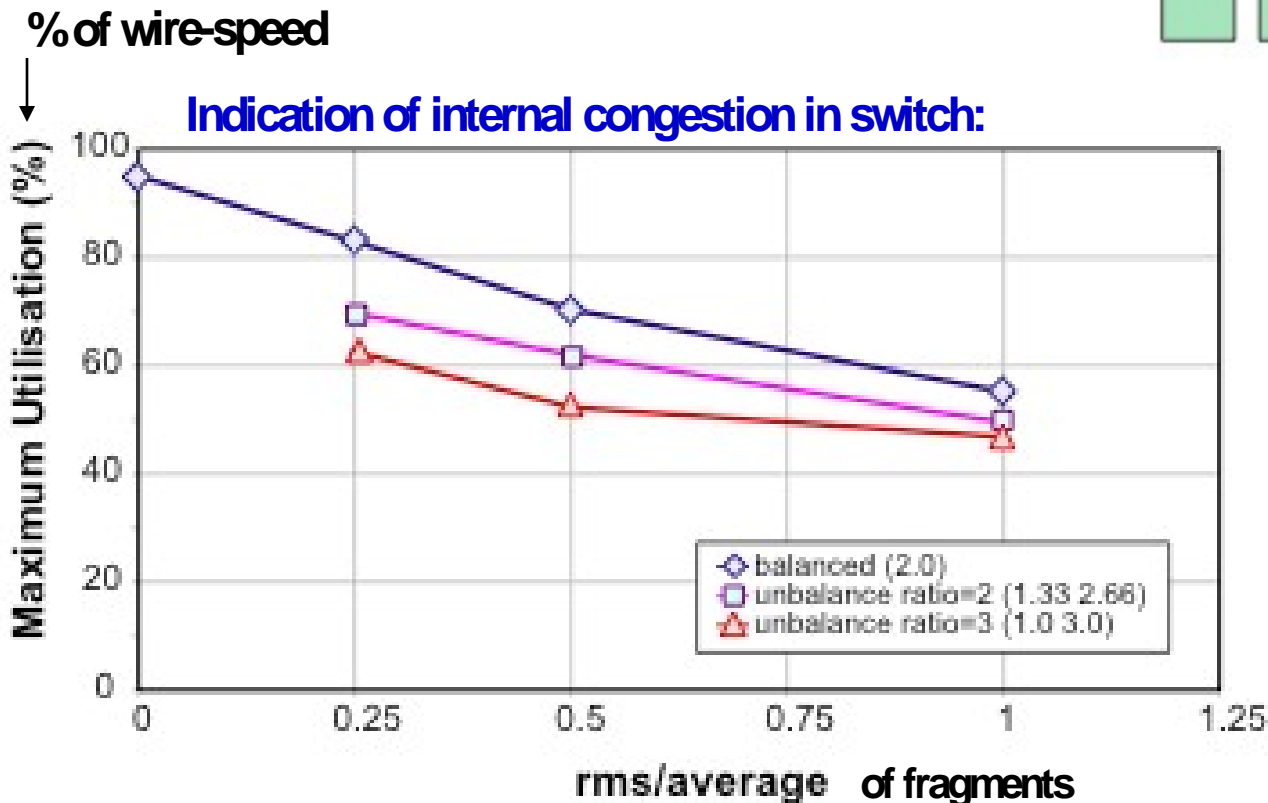
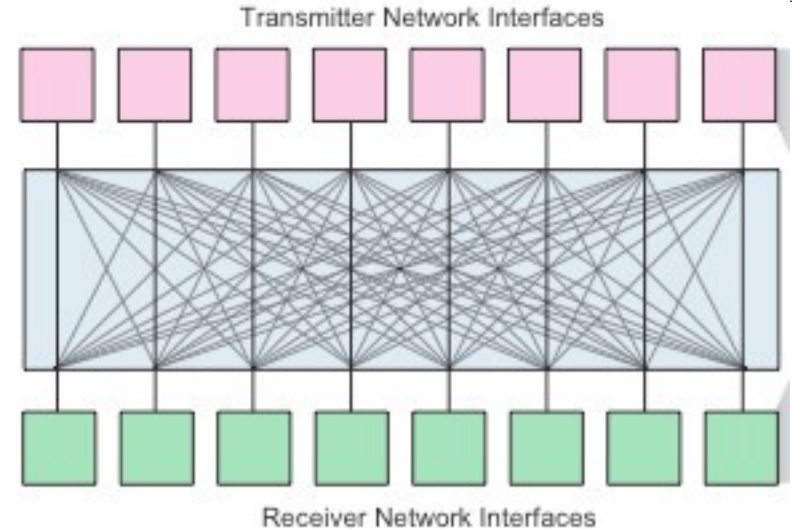
- **Ethernet switch with a lot of internal memory**

- A lot of ports (> 1000)
- Congestion mitigated through with a lot of fast internal memory.
- Higher latency (not relevant for event building)
- Very expensive...



# Performance of “1 rail” FEDBuilder

Measurement configuration:  
8 sources to 8 destinations



Measured switch utilization:

**Blue:** all inputs 2 kB avg

**Magenta:** 4 x 1.33 kB

4 x 2.66 kB

**Red:** 4 x 1 kB

4 x 3 kB

**≈ 50 %**

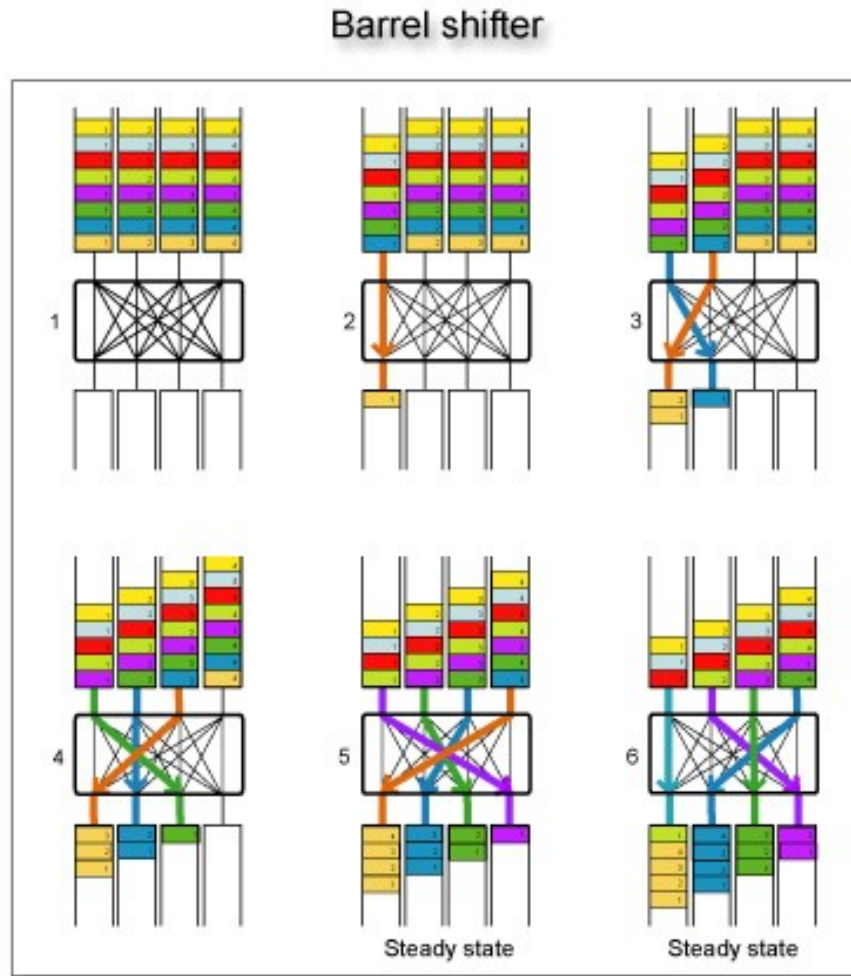
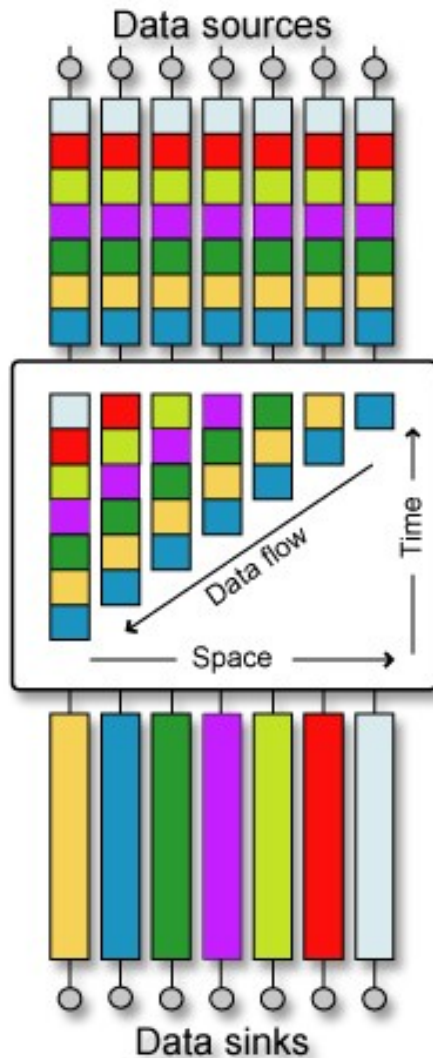
# Conclusion: EVB traffic and switches

---

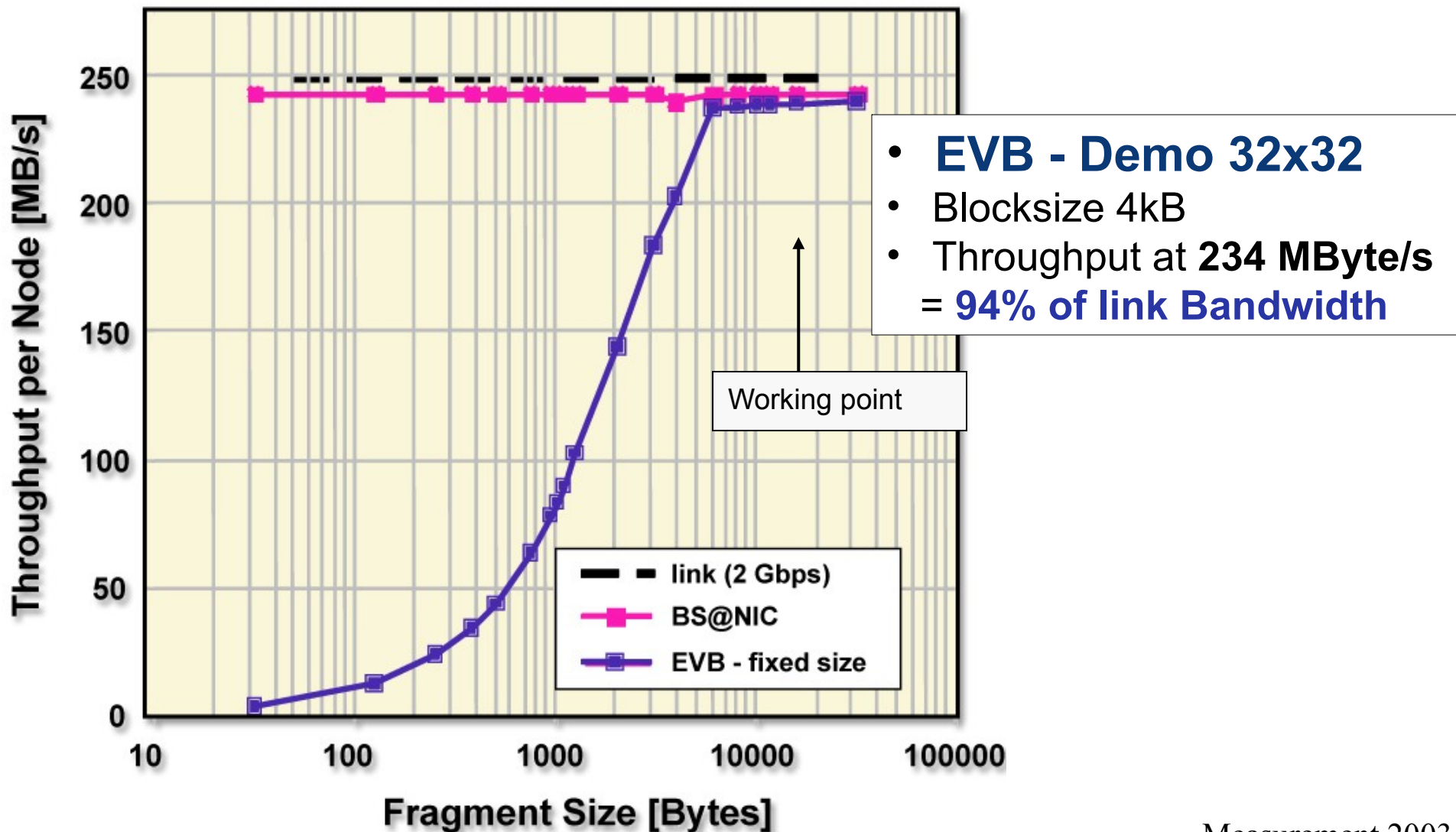
- **EVB network traffic is particularly hard for switches**
  - The traffic pattern is such that it leads to congestion in the switch.
  - The switch either “**blocks**” (= packets at input have to “wait”) or **throws away** data packets (Ethernet switches)
  
- **How to deal with this ???**  
  
→ **2 possible solutions...**

# 1st : the clever solution: traffic shaping

## Example: Barrel Shifter



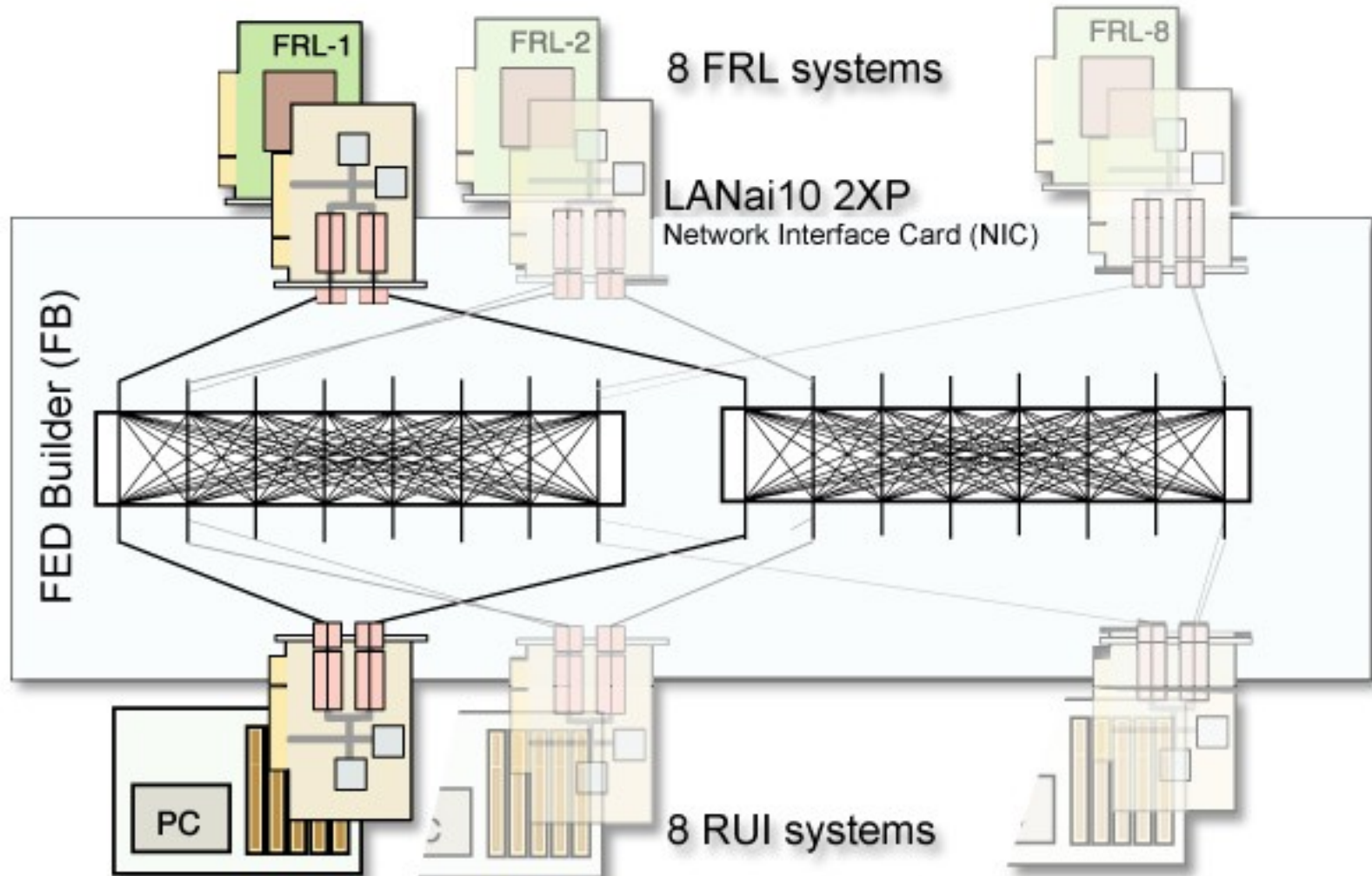
# Barrel Shifter: Measured Performance



Measurement 2003  
(still valid)

# 2nd Solution: “take the hammer”

Over-dimension the system: buy twice as much hardware



# What did CMS do???

---

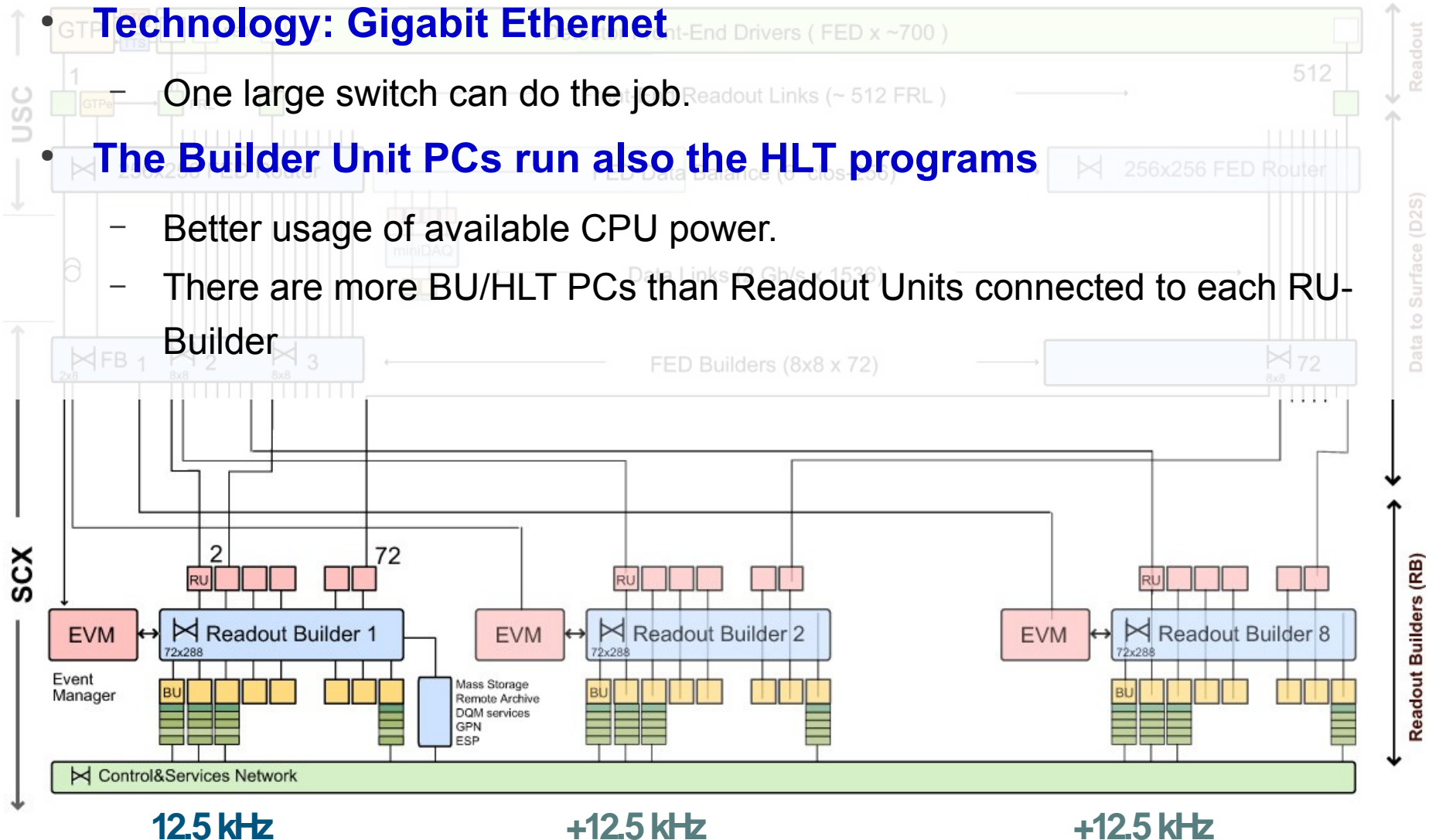
😊 Of course: we took the hammer 😊

- **Advantages:**

- Much **less development work**
- No dependence on internal working of the switch
- Much **less maintenance work**
- **Most important: redundancy**
  - If one rail fails: continue to run with one leg ( → less performance but still taking data !!!!)



# 2<sup>nd</sup> stage Event Builder: “bread and butter”



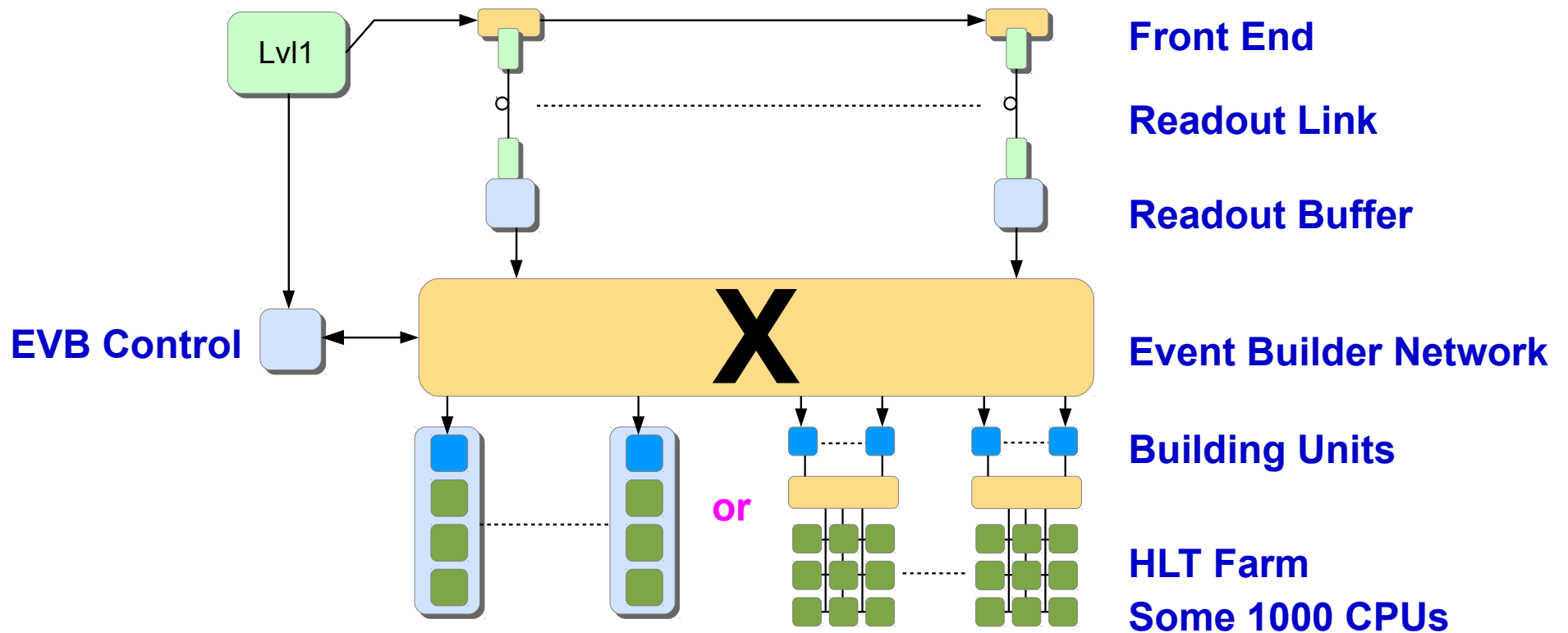
# Link Aggregation

---

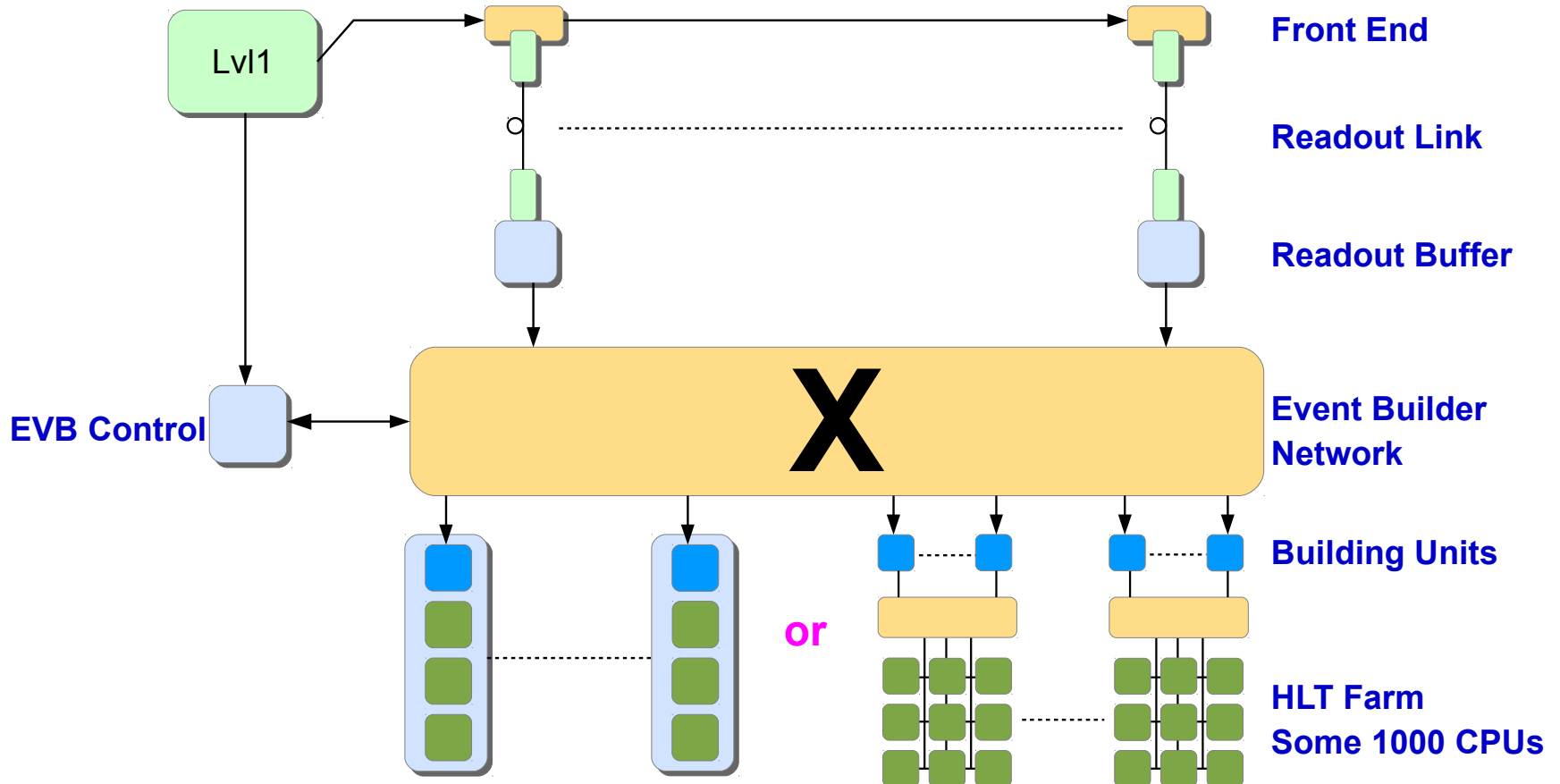
- **We want at least 200 MB/s traffic from RU to BU**
  - One Gigabit Ethernet line can transfer “only” 120MB/s
  - Need multiple Network Interfaces in RU and BU
  - Divide network between RUs and BUs into virtual LANs
    - Connect every RU to BUs in different VLANs
    - Connect every BU to RUs in different VLANs
    - Every RU can send data simultaneously on different VLANs
    - Every BU can receive data simultaneously on different VLANs
  - Alternative solution
    - Use Link Aggregation (IEEE standard exists)

# Event Building: EVB-Protocol

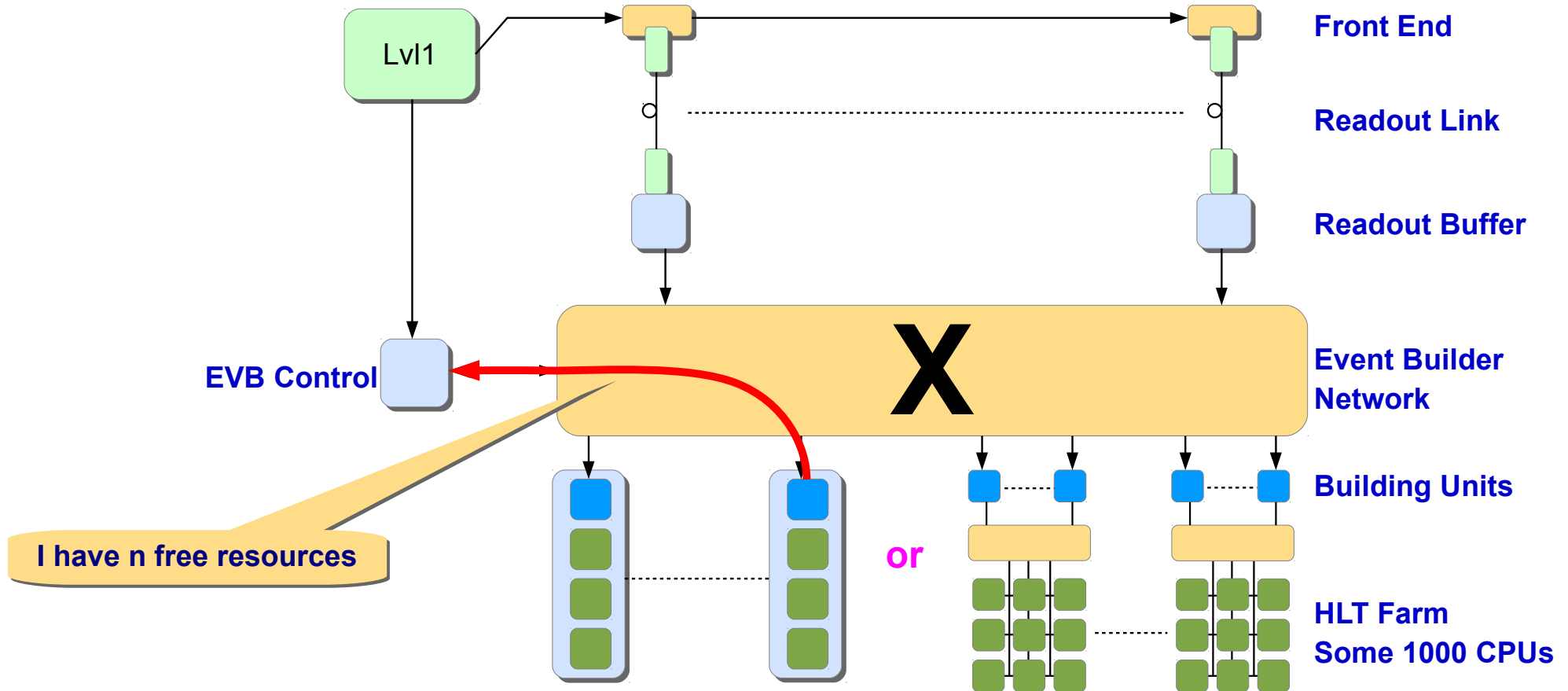
- **Aim: Event Builder should perform load balancing**
  - If for some reason some destinations are slower than others this should not slow down the entire DAQ system.
  - Another form of **traffic shaping**



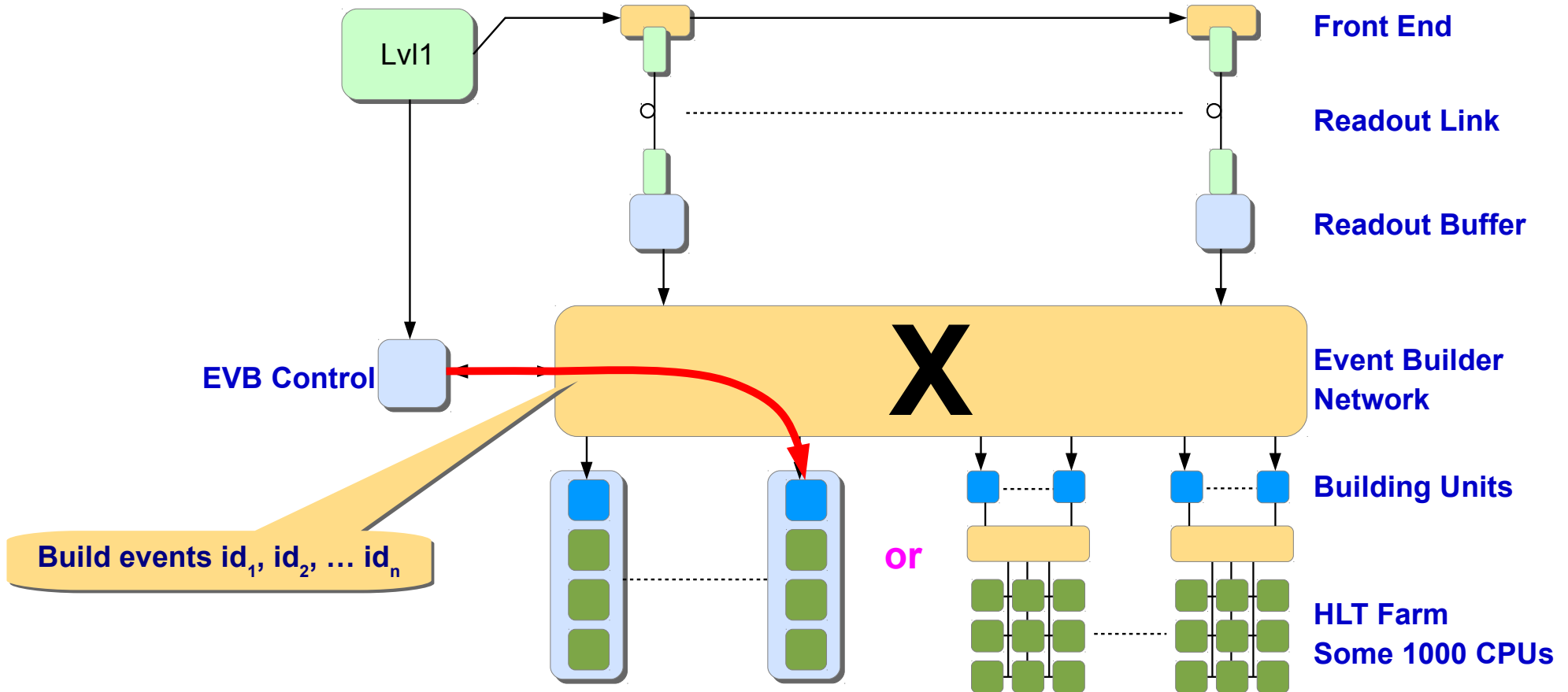
# Event Building: EVB Protocol



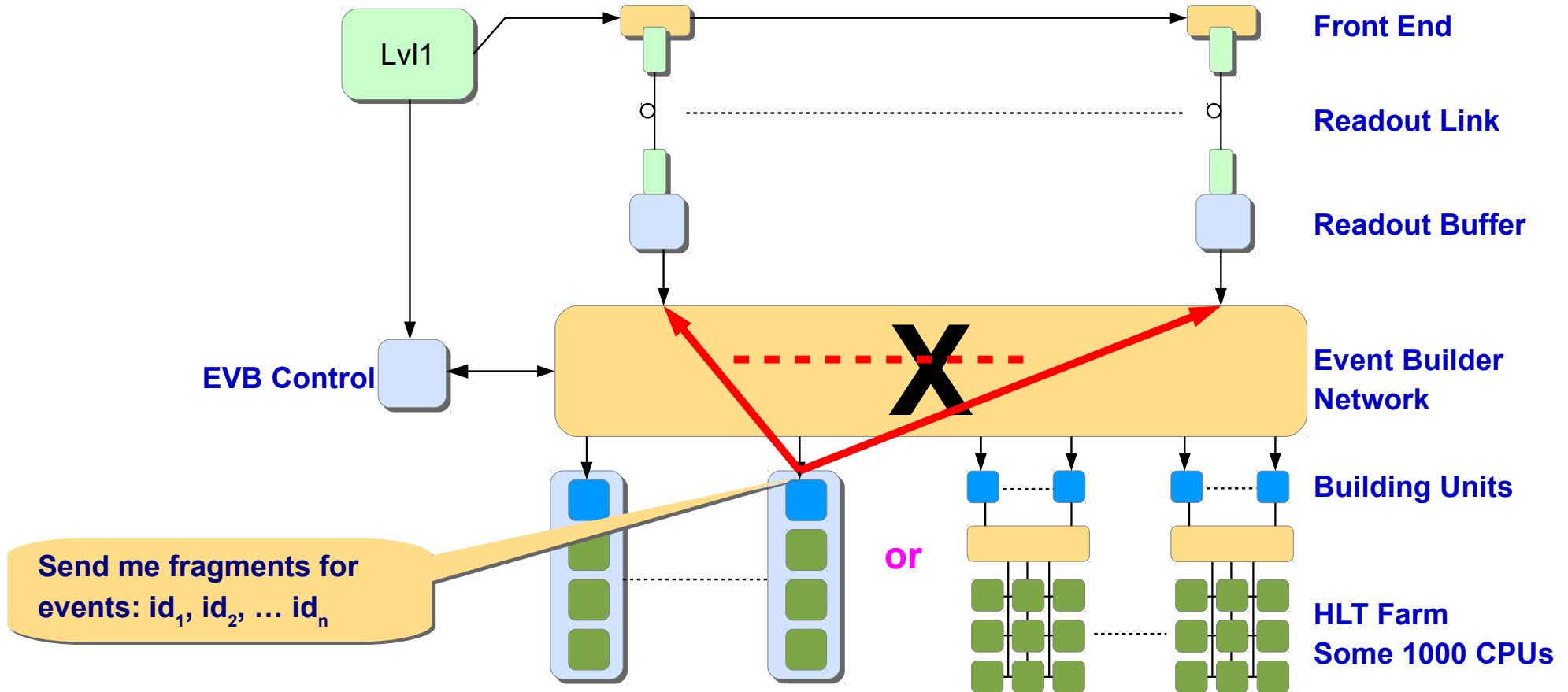
# Event Building: EVB Protocol



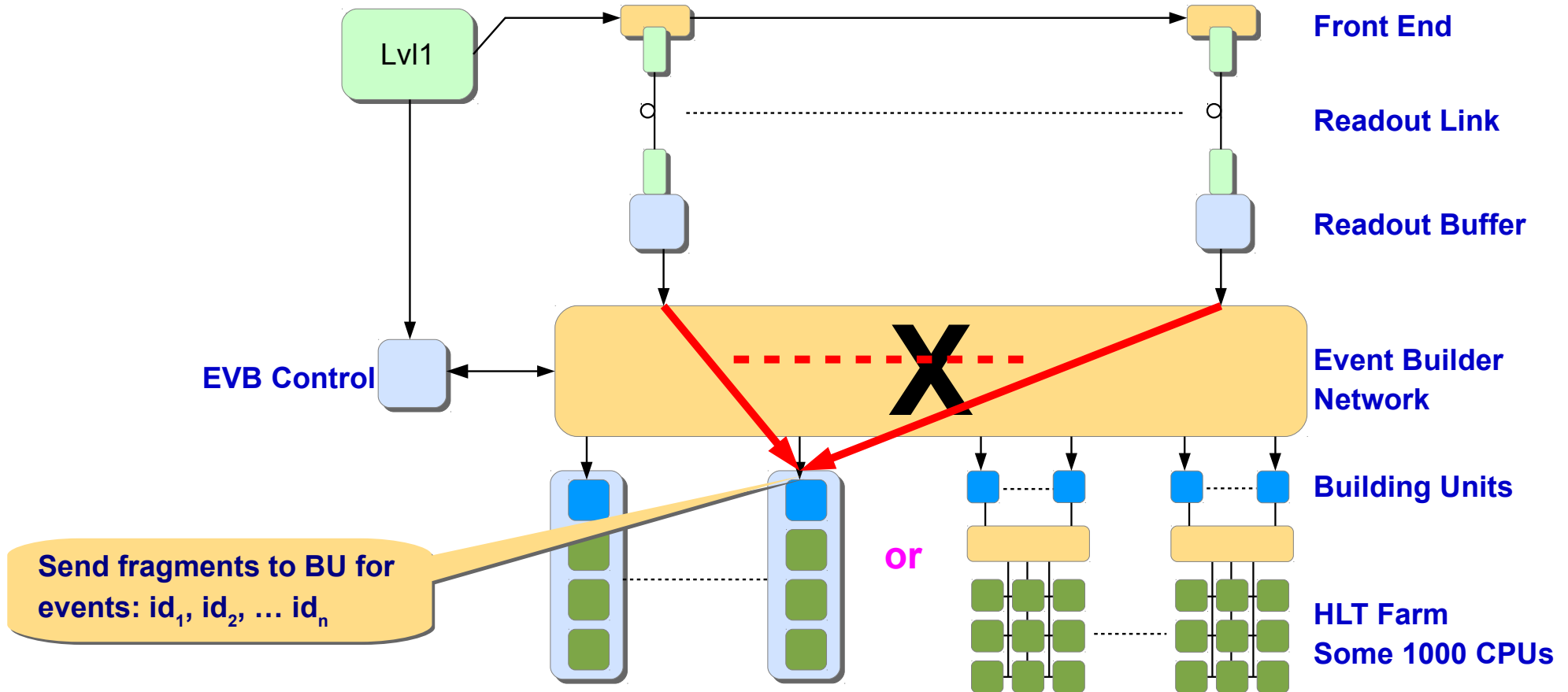
# Event Building: EVB Protocol



# Event Building: EVB Protocol



# Event Building: EVB Protocol





# Event Builder Components



## Half of the CMS FED Builder

One half of the FEDBuilder is installed close to the experiment in the underground. The other half is on the surface close to the RU-Builder and the Filter Farm implementing the HLT. The FEDBuilder is used to transport the data to the surface.

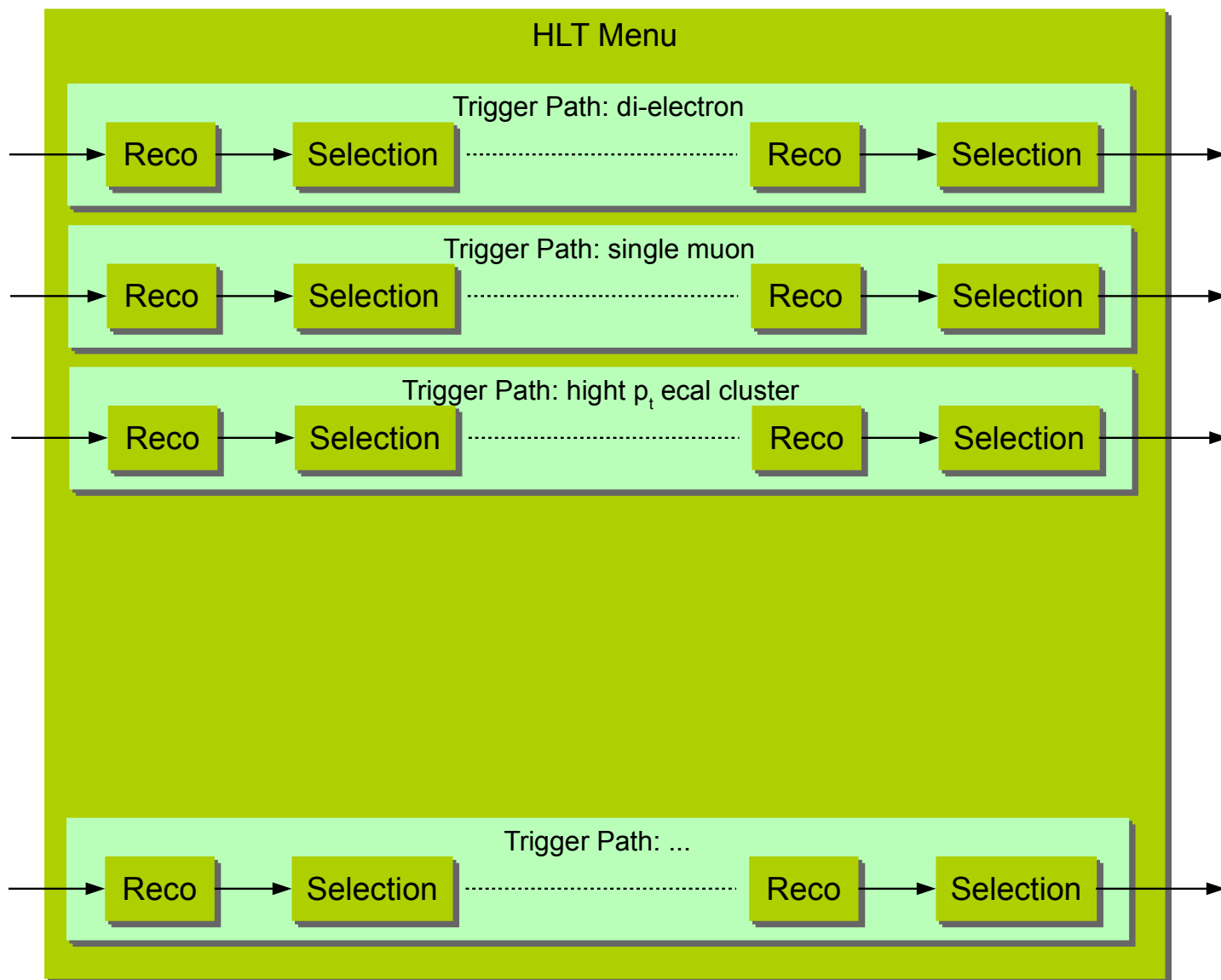
---

# HLT trigger implementation

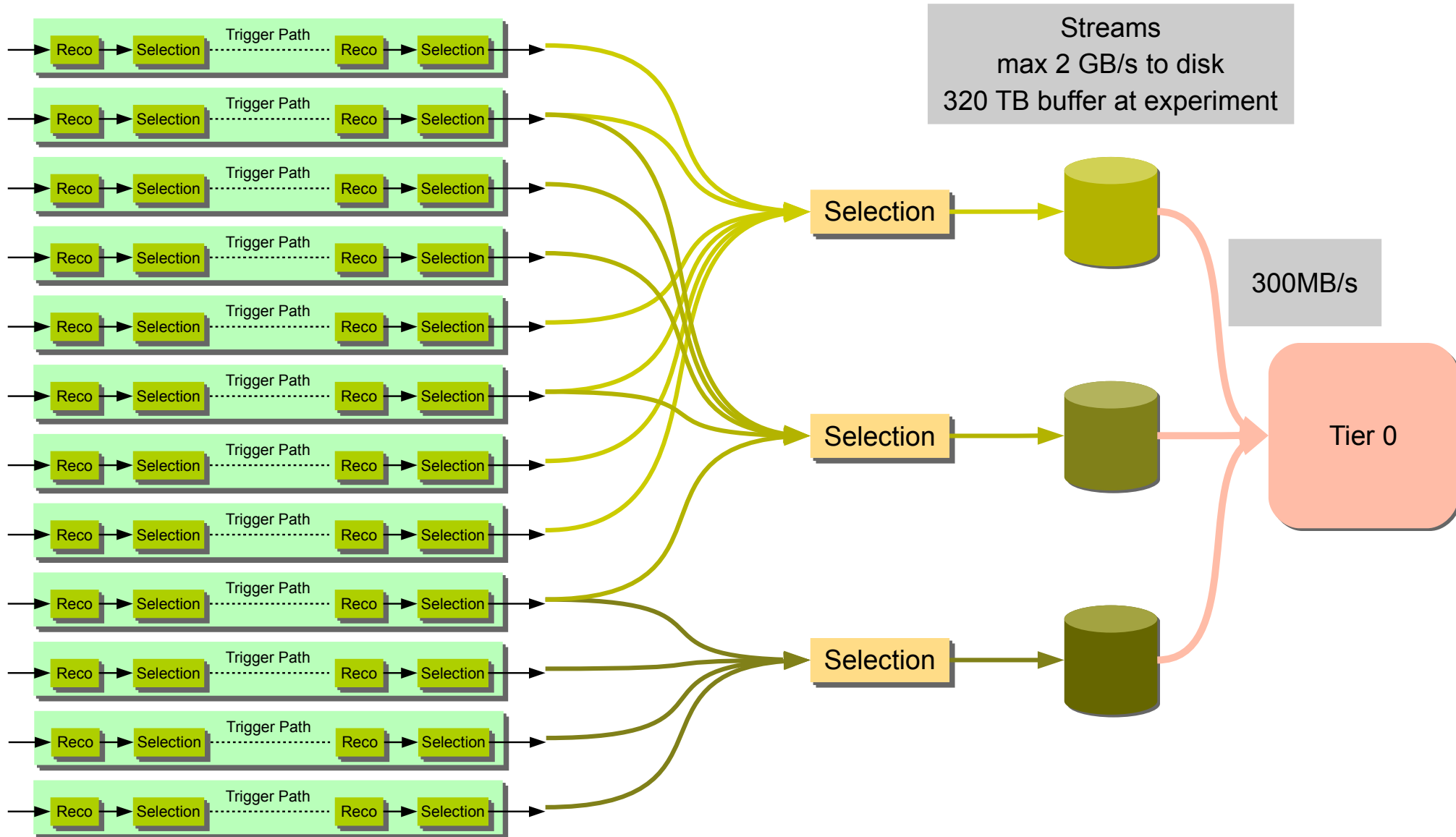
# CMS High Level Trigger of Filter

## Filter processes

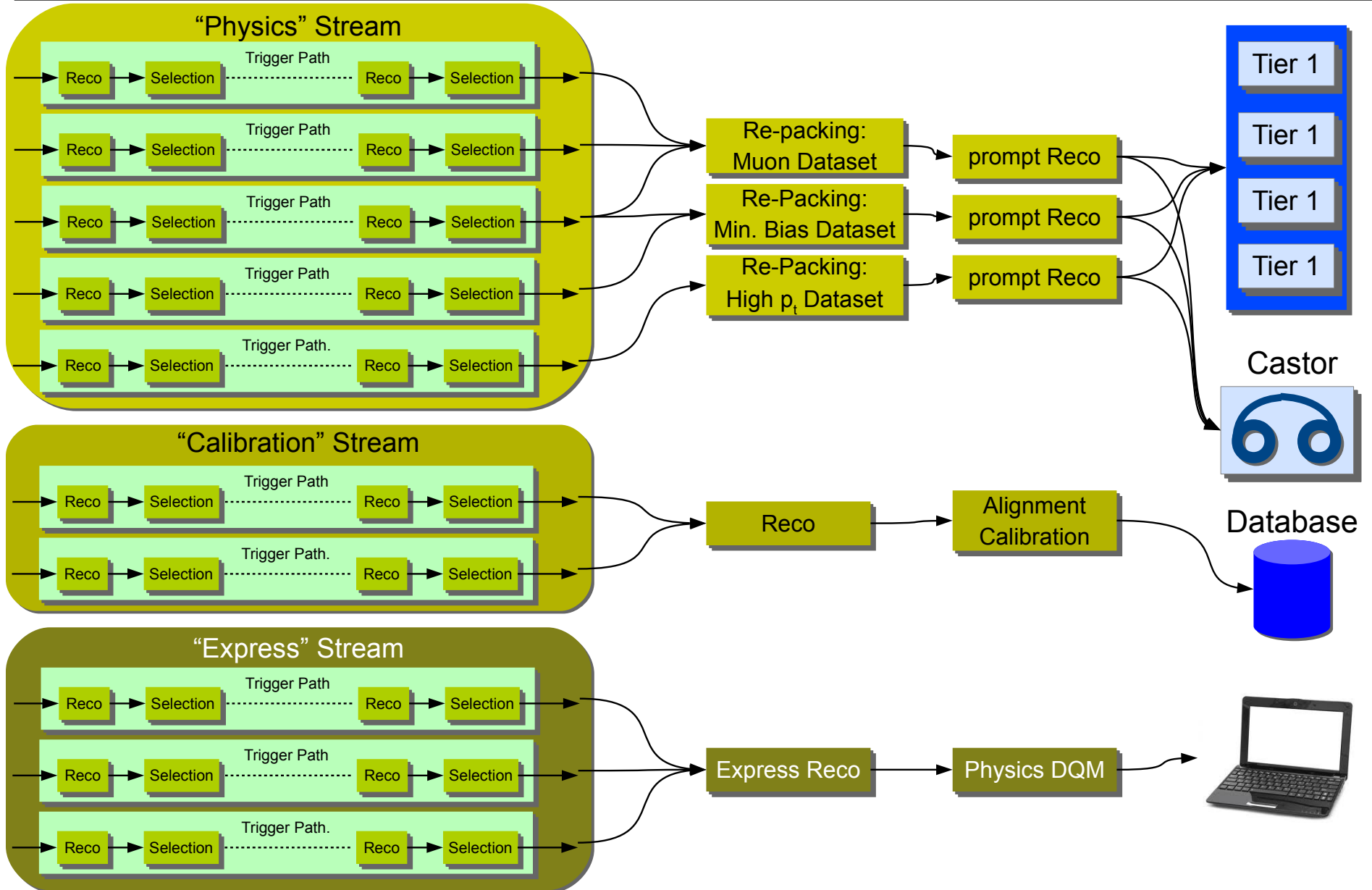
- Run the same code as offline analysis
- Trigger decision based on “trigger paths”



# Output Streams



# Tier0 processing



---

# **Online software: Some aspects of software design**

# History: Procedural programming

---

- **Up to the 90's: procedural programming**
  - Use of libraries for algorithms
  - Use of large data structures
    - Data structures passed to library functions
    - Results in form of data structures
- **Typical languages used in Experiments:**
  - Fortran for data analysis
  - C for online software

# Today: Object Oriented Programming

- **Fundamental idea of OO:**

**Data is like money: completely useless...if you don't do anything with it...**

- Objects (instances of classes) contain the data and the functionality:
  - Nobody wants the data itself: you always want to do something with the data (you want a “service”: find jets, find heavy particles, ...)
  - **Data is hidden** from the user of the object
  - Only **the interface** (= methods =functions) **is exposed** to the user.
- Aim of this game:
  - Programmer should not care about data representation but about functionality
  - Achieve better robustness of software by encapsulating the data representation in classes which also contain the methods:
    - The class-designer is responsible for the data representation.
    - He can change it as long as the interface(= exposed functionality) stays the same.
- Used since the 90s in Physics experiments

- **Experience so far:**

- It is true that for large software projects a **good OO design is more robust and easier to maintain.**
- Good design of a class library is difficult and time consuming and **needs experienced programmers.**



# Frameworks vs Libraries

- **What is a software framework?**

- Frameworks are programming environments which offer enhanced functionality to the programmer.
- Working with a framework usually implies programming according to some rules which the framework dictates. This is the difference wrt use of libraries.

- **Some Examples:**

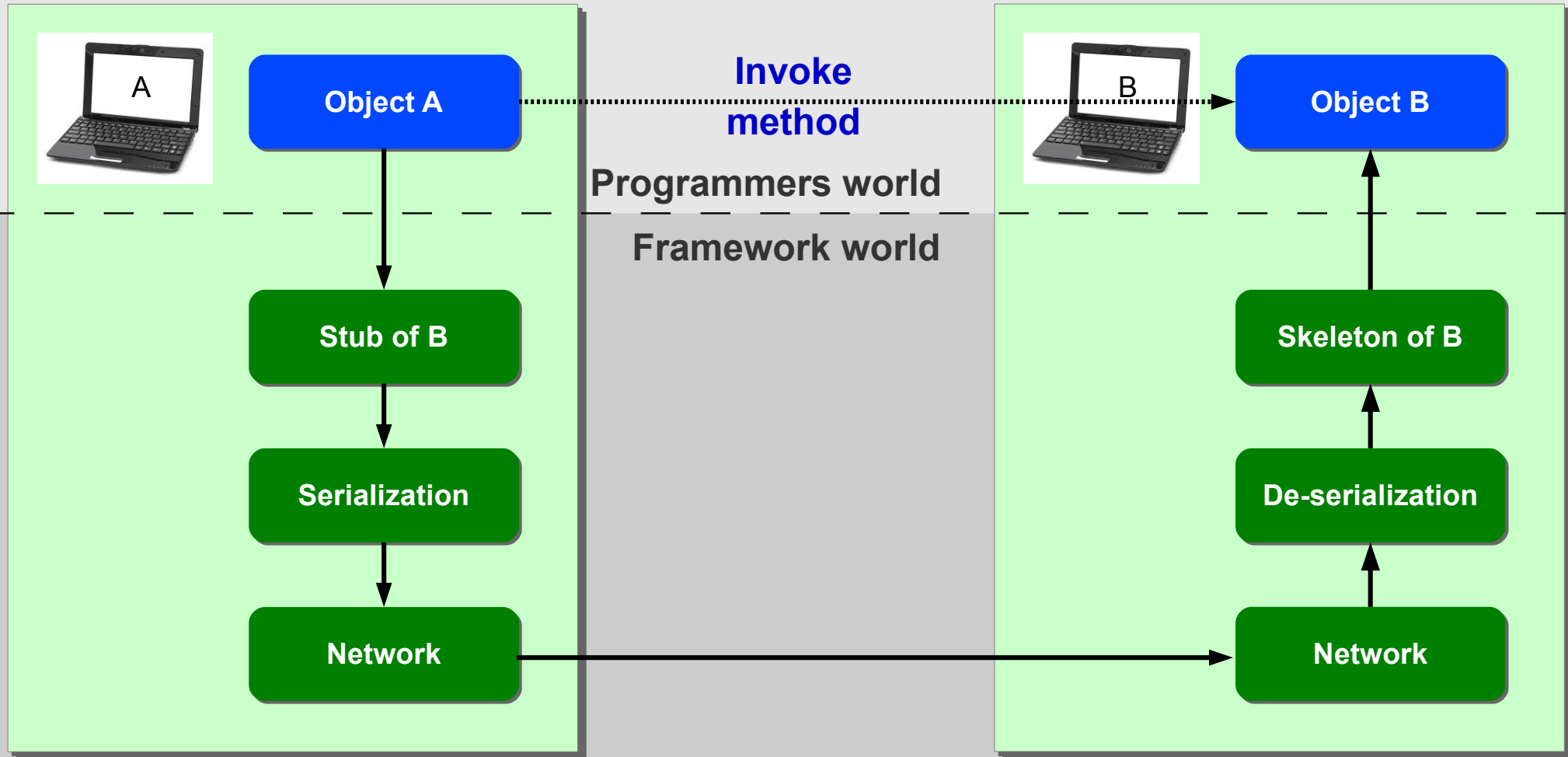
- Many frameworks for programming GUIs “own” the main program. The programmer’s code is only executed via callbacks if some events are happening (e.g. mouse click, value entered, ...)
- An Physics Analysis framework usually contains the main loop over the events to be analyzed.
- An online software framework contains the functionality to receive commands from a Run-Control program and executes specific call-backs on the programmer’s code.  
It contains functionality to send “messages” to applications in other computers hiding the complexity of network programming from the application.

# Distributed computing

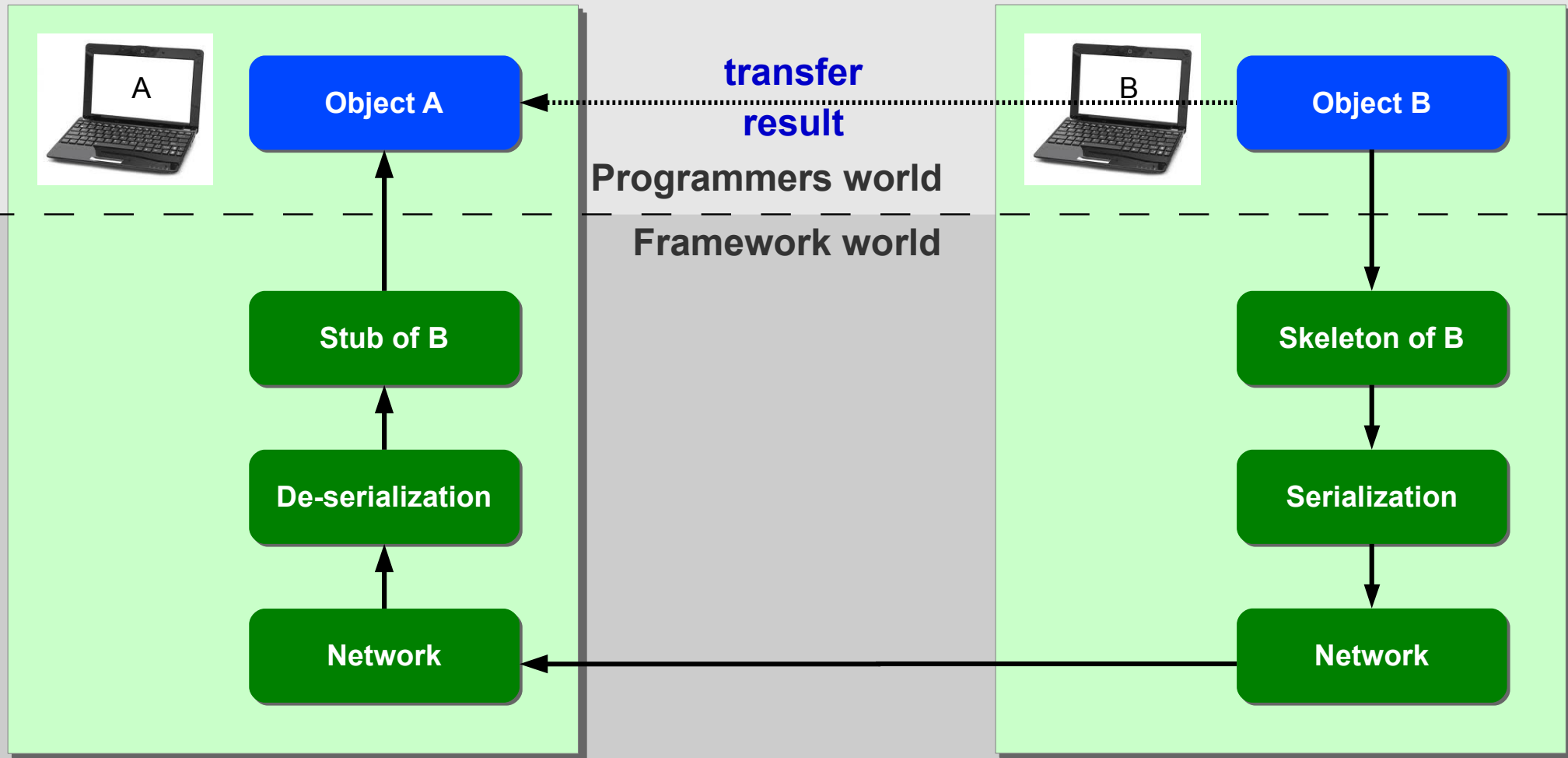
---

- **A way of doing network programming:**
  - “Normal Program”: runs on a single computer. Objects “live” in the program.
  - Distributed Computing: An application is distributed over many computers connected via a network.
    - An object in computer A can call a method (service) of an object in computer B.
    - Distributed computing is normally provided by a framework.
    - The complexity of network programming is hidden from the programmer.
- **Examples:**
  - CORBA (Common Object Request Broker Architecture)
    - Used by Atlas
    - Works platform independent and programming language independent
  - SOAP (Simple Object Access Protocol)
    - Used by CMS
    - Designed for Web Applications
    - Based on xml and therefore also independent of platform or language

# Distributed computing: Making a request

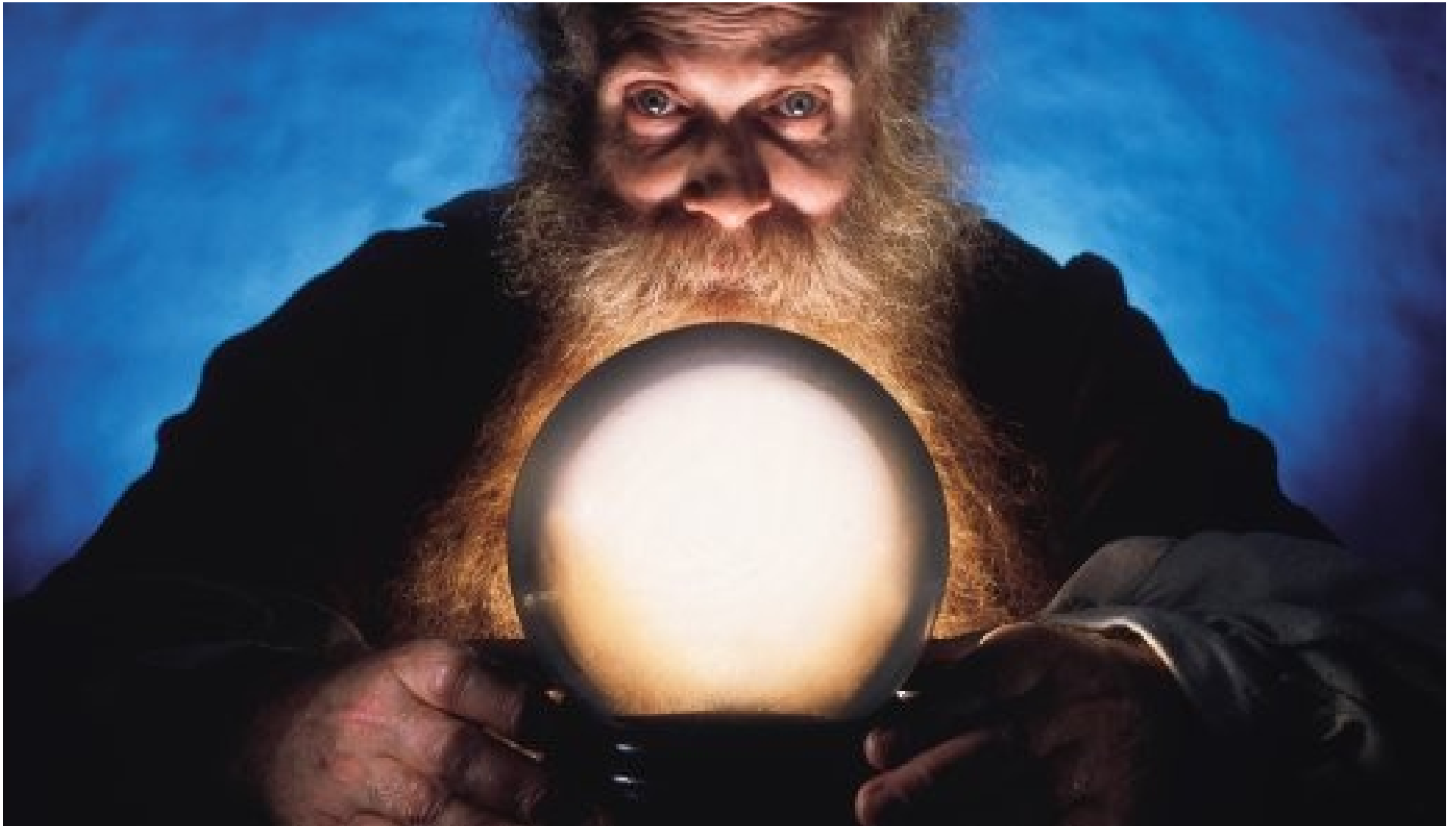


# Distributed computing: Return of result



# ??? What does the future bring us ???

---



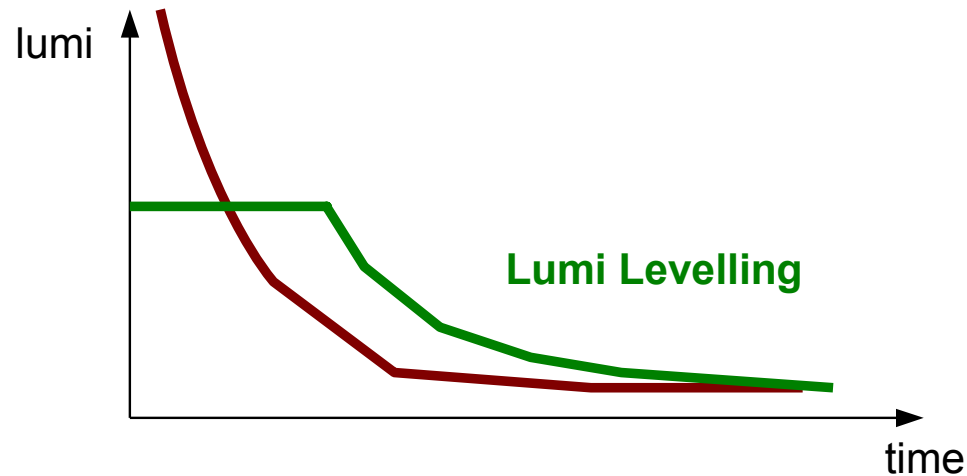
# Upgrade of LHC

- Remember from the Trigger Upgrade Section...
- The event size scales with the number of underlying Min. Bias Events
  - The experiment was designed for pileup of 24. (...fortunately with some margin... )
  - In 2012 we had a pileup > 30
  - After LS 1 we must be ready to accept pileups > 50

BX spacing [ns]	Beam current [ $\times 10^{11}$ e]	Emittance [ $\mu\text{m}$ ]	Peak Lumi [ $\times 10^{34}\text{cm}^{-2}\text{s}^{-1}$ ]	Pileup
25	1.15	3.5	0.92	21
25	1.15	1.9	1.6	43
50	1.6	2.3	0.9-1.7	40-76
50	1.6	1.6	2.2	108

# Upgrade of LHC: Consequences for DAQ

- **Event sizes are dominated by underlying events**
  - The number of pile up events might double
  - The event size might double
  - In case the pile up exceeds 50: LHC will do lumi levelling:
    - Beams will be separated at the start of a fill not not exceed a pile up of 50.
    - The separation increases the life time of the beams
    - The beams will be steered closer when the pileup decreases.
    - As a consequence there will be long fills with high luminosity



# In the far future... (year 2018...20xy)

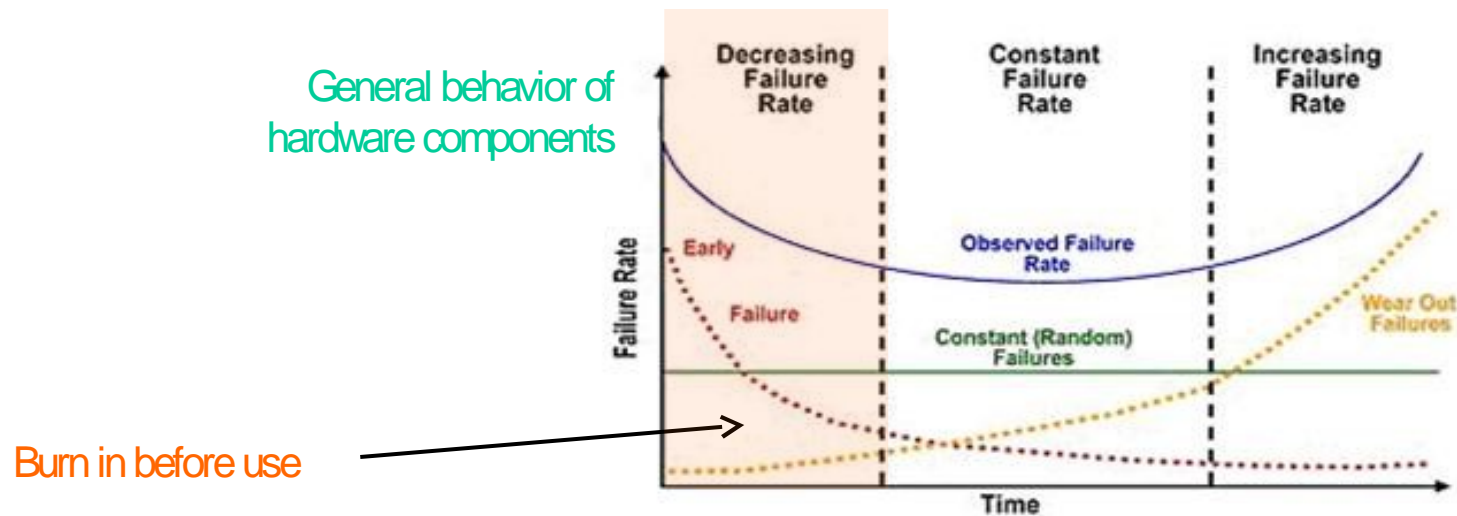
---

- **Luminosity of LHC might reach multiple of design lumi**
  - Today people talk about  **$5 \times 10^{34}$**  which is 5 x design lumi
  - Trigger and DAQ need to be heavily upgraded for this scenario.
  - Event size will grow due to detector upgrades (more channels) and more underlying min. bias events
    - DAQ needs substantially **higher data throughput**



# Real life... another reason to “upgrade”

- **A lot of hardware components become old ...**
  - System reliability decreases
    - It makes sense to replace PCs every 4 years
    - It make sense to replace network equipment every 7 years
    - Custom hardware is usually kept longer... but of course it also starts breaking...

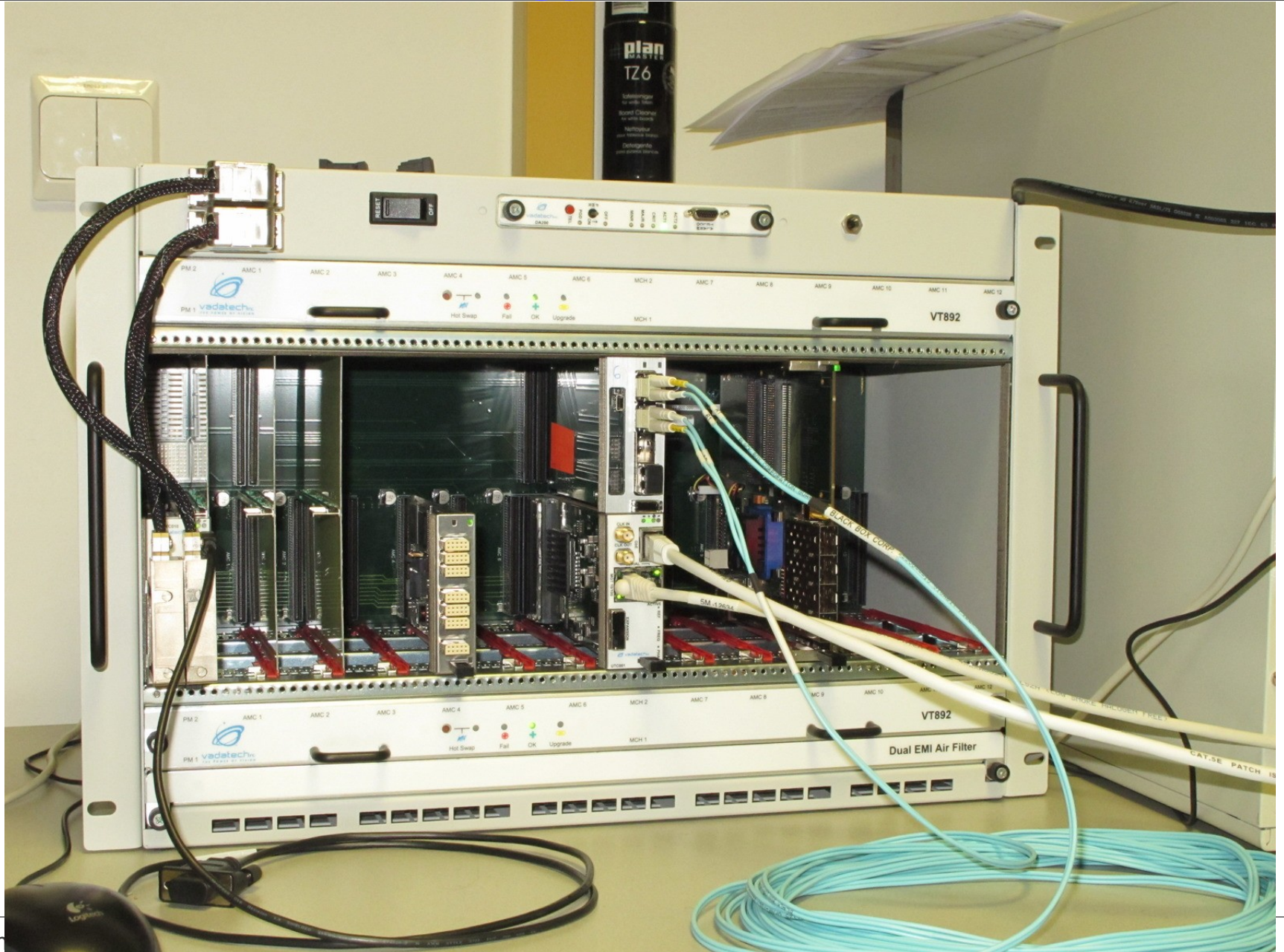


# Trigger/DAQ upgrade: technology

---

- **Upgrade technology for very high lumi**
  - Larger state of the art **FPGA devices**
    - Larger granularity needed
    - The trigger needs to cope with more channels
  - **Modern link technology** to interconnect processing boards
    - Multi Gigabit serial links
    - Telecommunication technology (uTCA crates with customized backplanes)

# uTCA Technology: Follower of VME?

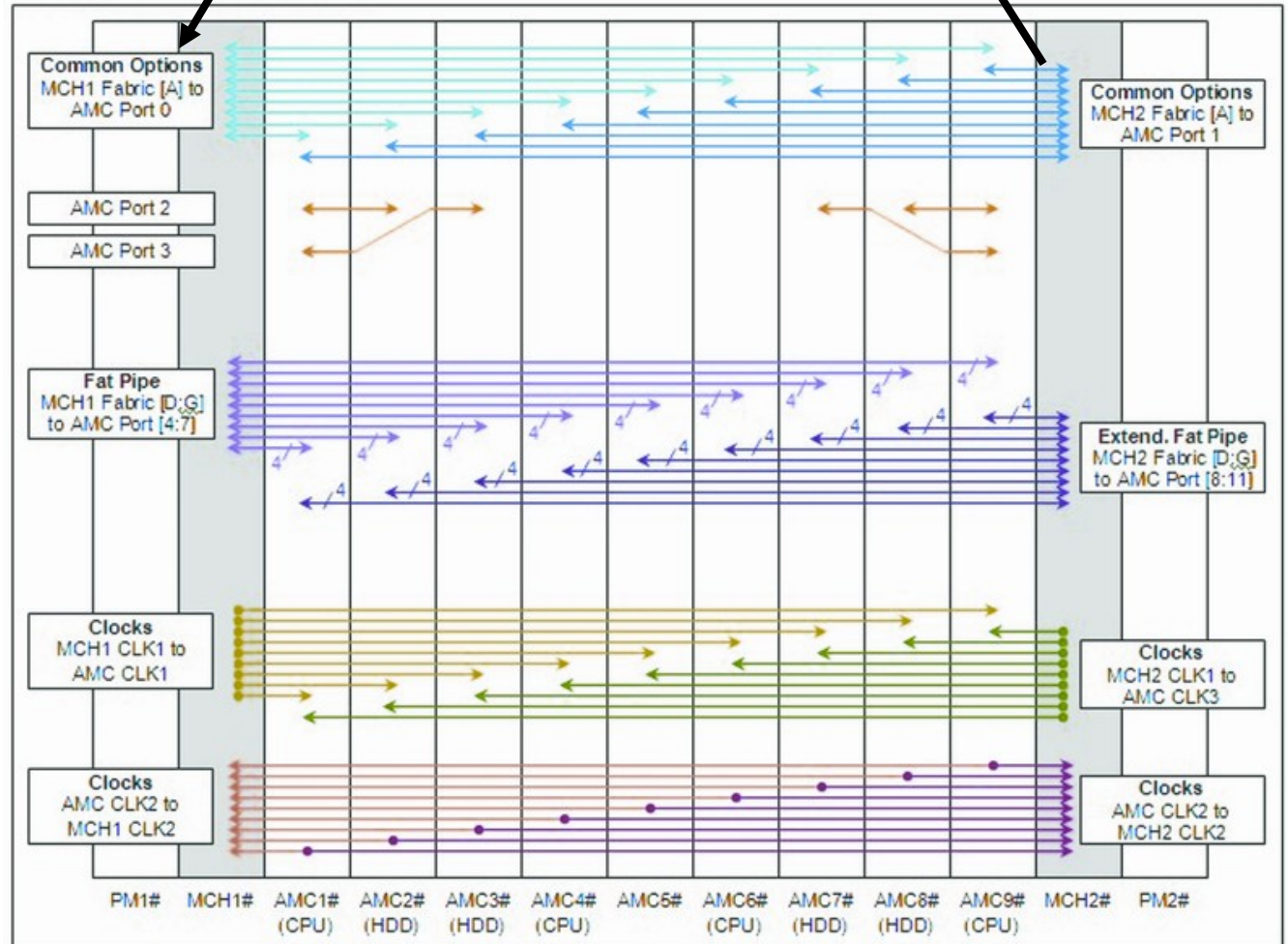


# uTCA technology: backplane

MCH: crate controllers; redundancy with 2 controllers per crate.

Port 0/1: can be used for control (ethernet).  
Connected to switch in MCH

“Fatpipes” : High Speed interconnect  
Ideal for trigger modules



# xTCA in LHC

---

- **Atlas and Alice will (most probably) build future hardware in ATCA**
- **CMS has decided to build new hardware in uTCA**
  - Calorimeter trigger already now in uTCA
  - Trigger and HCAL electronics is being built for after LS1
- **To be considered...**
  - The uTCA infrastructure is highly complicated:
    - IPMI in order to manage the crate / cards in the crate.
      - Accessing the cards in the crate (AMCs) with IPMI from “outside” via the MCH is highly problematic due to a lack of standardization or available tools.
    - The uTCA crate needs a small embedded computer to “boot up” and to control it: the MCH
    - uTCA is designed for highly redundant systems (in general in physics we do not use these features)
    - Inter-operation of cards of different vendors is not always easy.
    - The uTCA standard does not specify a unique way to control the AMCs
      - VME did this very well → you could buy solutions for crate control
        - VME-bridges which connect to PC (including software)
        - Crate Controller PCs in form of a VME card
        - Chip sets which implement VME interfaces to put on your cards

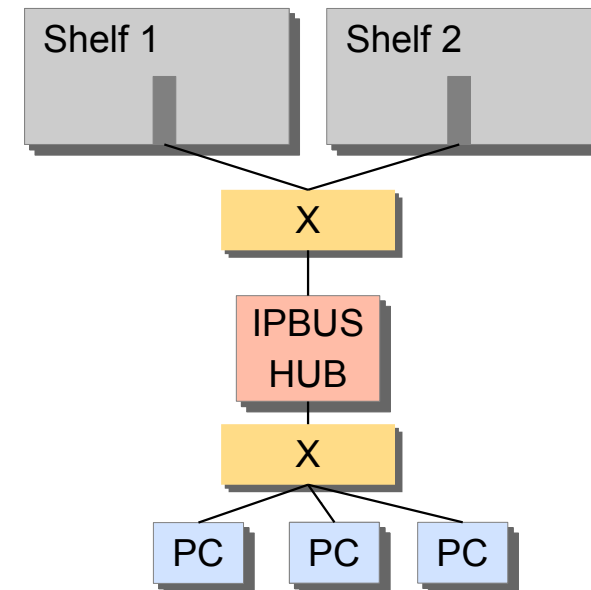
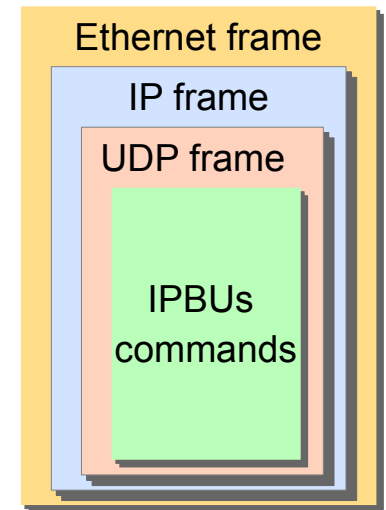
# uTCA: Accessing the modules in the shelf

---

- **One possibility: PCIe via a FATPIPE**
  - FATPIPE on the backplane to implement PCIe switched point to point connection to a central controller which interfaces to an outside PC.
  - **Advantages:**
    - Standard Protocol
    - Standard software can be used.
    - High data throughput
    - PCIe cores for FPGAs or PCIe chips are available on the market
  - **Disadvantage:**
    - You eat up one of the FatPipes.
    - The central controller needs to be implemented in some central MCH module
      - This is possible on a MCH add on card (user specific tongue)

# Accessing uTCA AMC's in CMS

- **A custom solution based on UDP (chosen by CMS):**
  - send ethernet packets on port0 via the MCH to the ethernet port.
  - **Advantages:**
    - All Fatpipes available
    - Easy to build a distributed system (UDP is IP based)
  - **Disadvantages:**
    - Completely custom solution
    - UDP is not reliable: Need complicated software to guarantee that packets arrive at their destination and that the uplinks are not over-committed
    - Custom firmware blocks for all FPGAs types need to be maintained
    - Low data throughput (1Gb/s per crate is the wire speed: You will need to stay below this in order not to loose packets.)



# DAQ upgrade projects

---

- **Increase bandwidth of EventBuilder**
  - **New Readout links**
    - Possibly with standard protocols
    - Connect directly to industrial network technology (TCP/IP?)
  - **Event builder switch network**
    - Move to 10Gb/Ethernet or Infiniband (56Gb/s)
  - **HLT farm**
    - Multi-core machines with more power.
- **Specific DAQ problem: backwards compatibility**
  - Not all sub-systems do the upgrade at the same time
  - Old and new readout systems need to co-exist
    - This prevents the possibility of radical changes (and unfortunately radical improvements are not feasible even though technical possible)

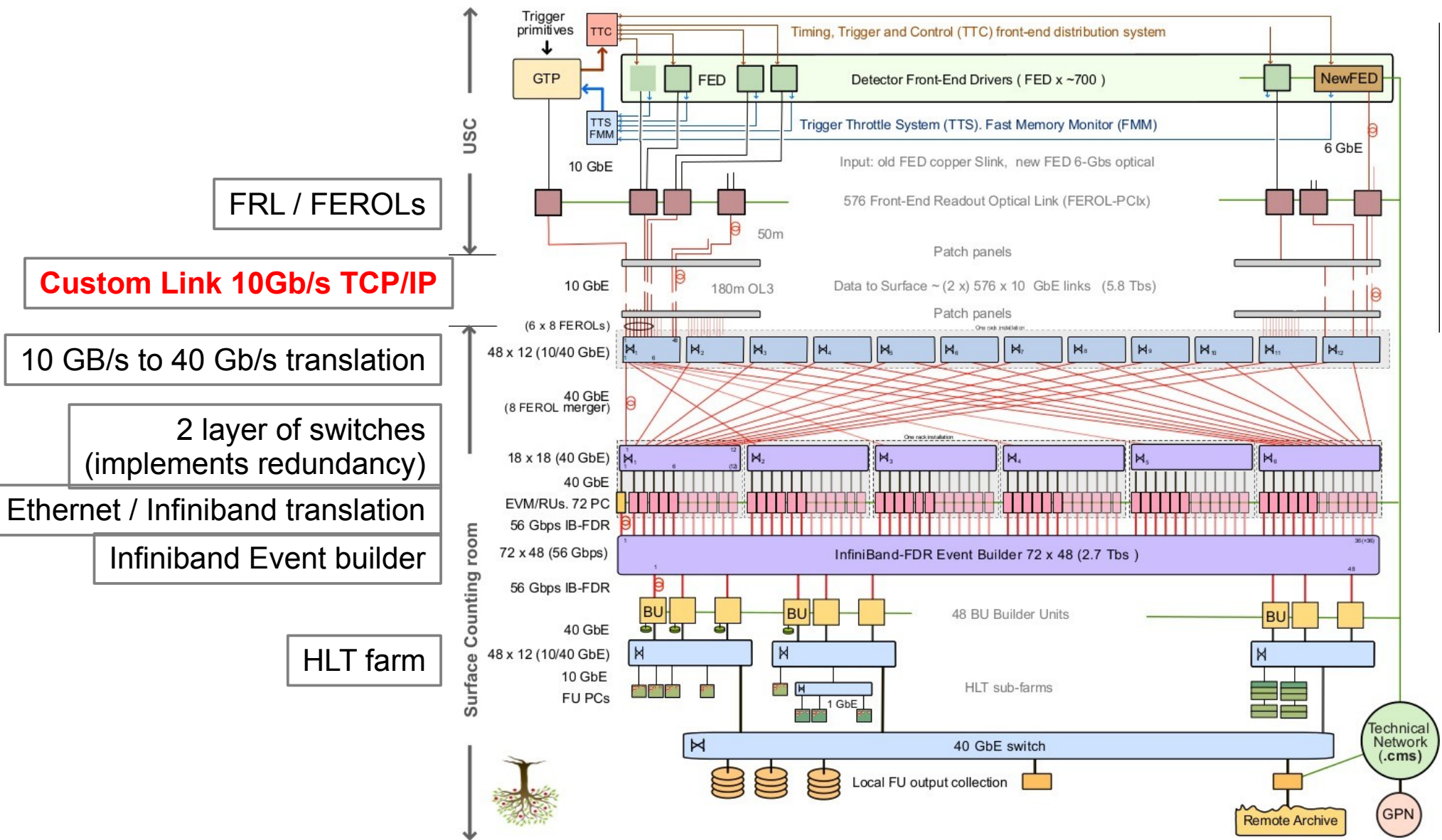


# DAQ Upgrade: a few examples

---

- **ATLAS**
  - Integration of Lvl2 and HLT Farm into one single computer farm
- **CMS**
  - 10Gb/s custom readout link based on TCP/IP
  - uTCA hardware will replace VME hardware
    - Currently being used for Trigger and HCAL
  - 56 Gbit/s or Infiniband based Event Builder under study
- **Alice**
  - Read out detector via Ethernet / UDP (10Gbit)
    - Goal: Read out Pb collisions at 50kHz
  - Restructuring of Computer Farms: HLT farm integrated with event builder farm
- **LHCb**
  - Readout at 30 MHz
  - Trigger only in Software

# Upgraded CMS DAQ system



FRL / FEROLs

Custom Link 10Gb/s TCP/IP

10 GB/s to 40 Gb/s translation

2 layer of switches (implements redundancy)

Ethernet / Infiniband translation

Infiniband Event builder

HLT farm

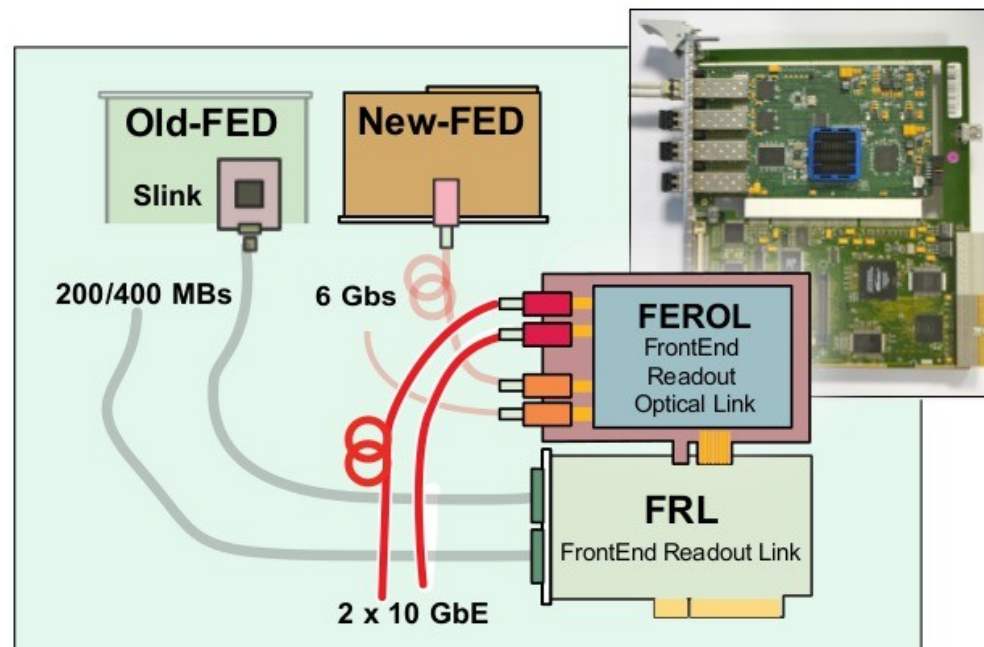
USC

Surface Counting room

# Use of the new Link

- **Legacy and new “Front End Drivers” are supported**

- The existing FRL receiver stays in place to support legacy FEDs.
- New Ferol has 2x10Gb/s links and 2 x 6Gb/s links
  - One 10Gb/s link with TCP/IP to DAQ
  - One 10Gb/s to support new fast FEDs via optical input
  - 2 x 6Gb/s links to support new FEDs with lower bandwidth
  - Data from legacy FEDs from FRL via PCI-x bus to Ferol

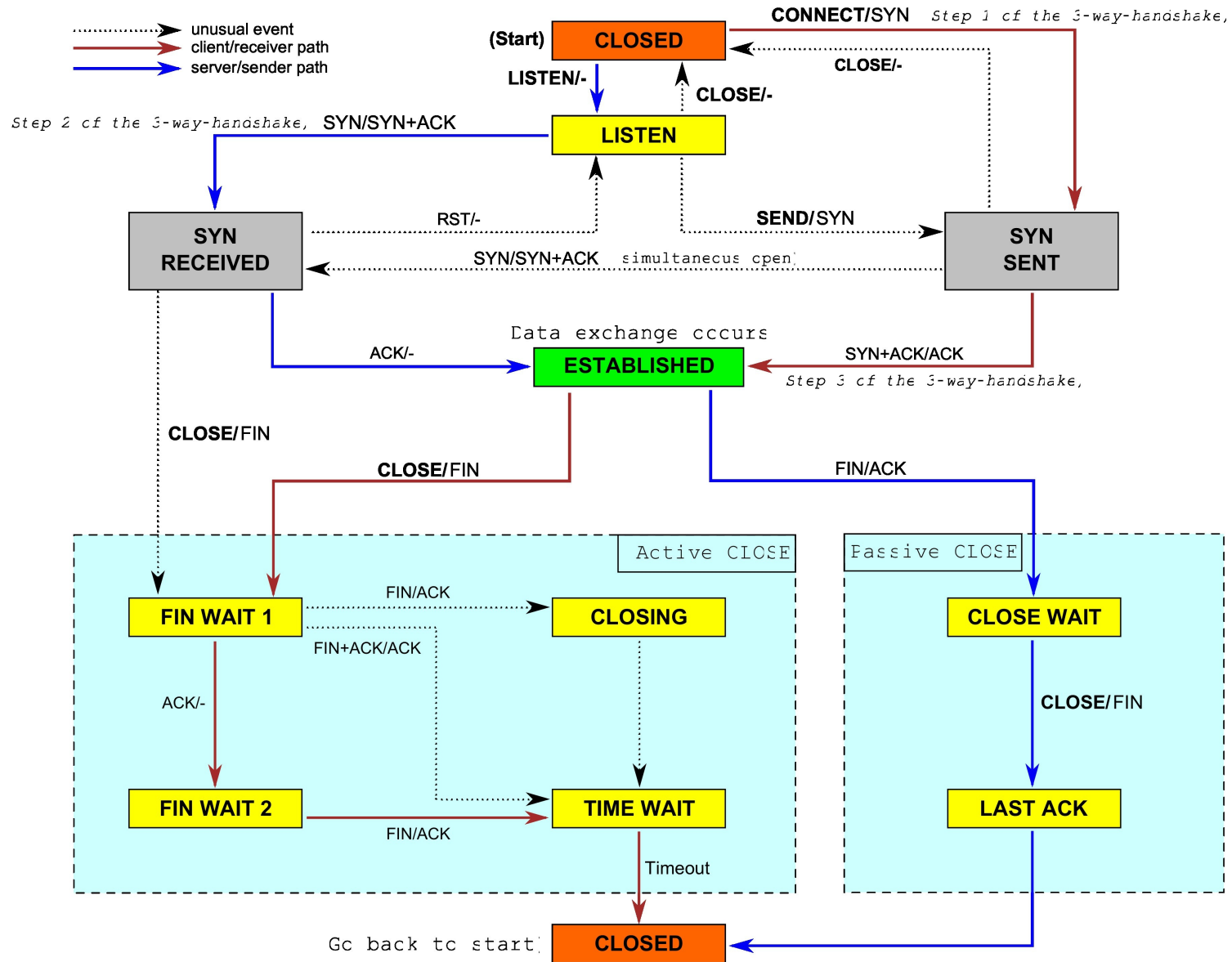


# CMS: an new custom readout link

---

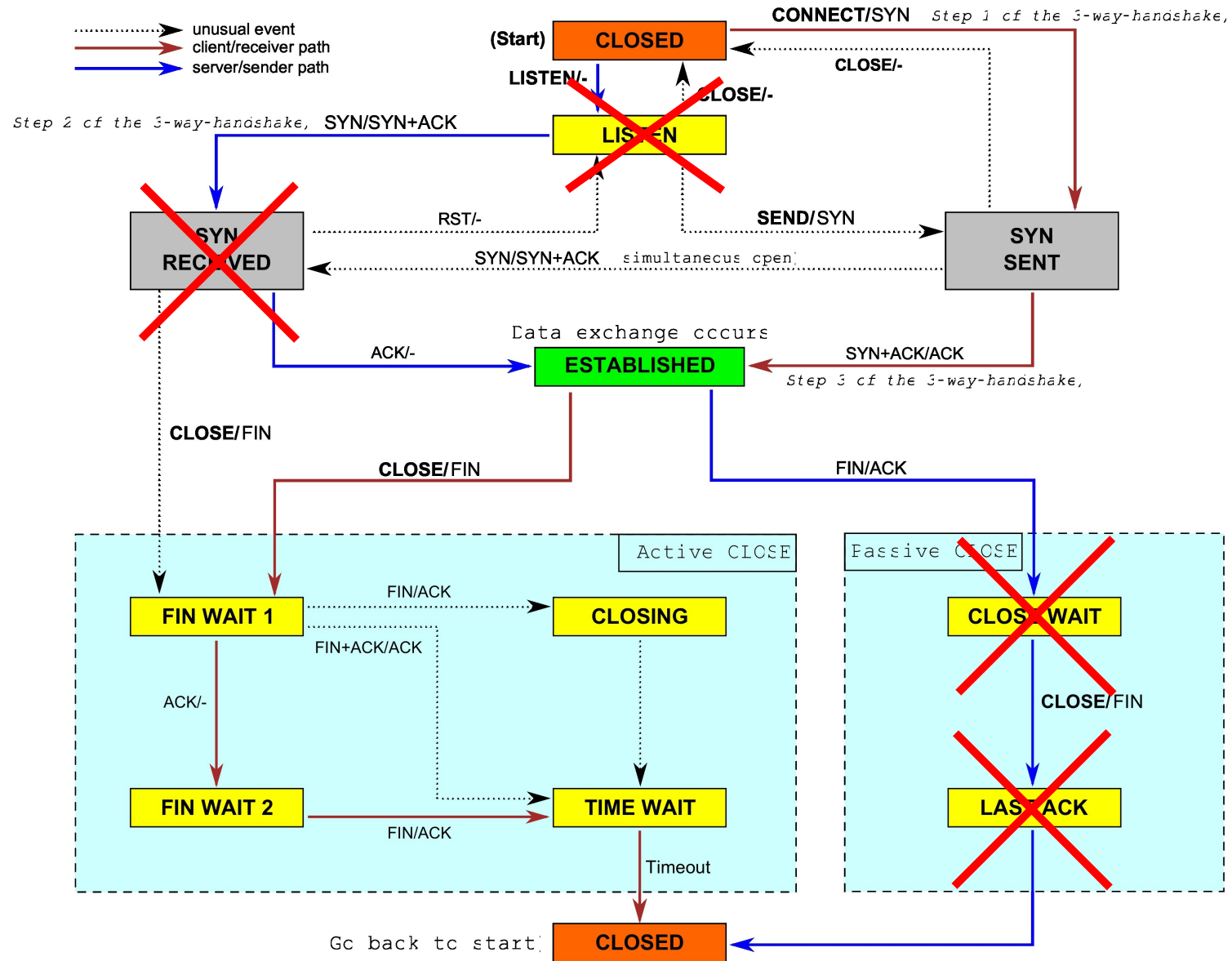
- **New custom readout link based on 10 Gb/s TCP/IP**
- **In principle a very difficult task for an FPGA**
  - Even for a Computer CPU TCP/IP is a CPU hungry protocol
- **Simplifications are possible due to specific traffic pattern**
  - We do not need to implement a universal network interface
  - We want to send data from one source, over a network to one destination.
  - The data traffic goes only in one direction for a readout link
    - Of course acknowledge packets must be sent; they are part of the protocol.
  - The handling of the connections can be drastically simplified

# TCP/IP Protocol Statediagram (simplified)



# Simplifying: TCP/IP (I)

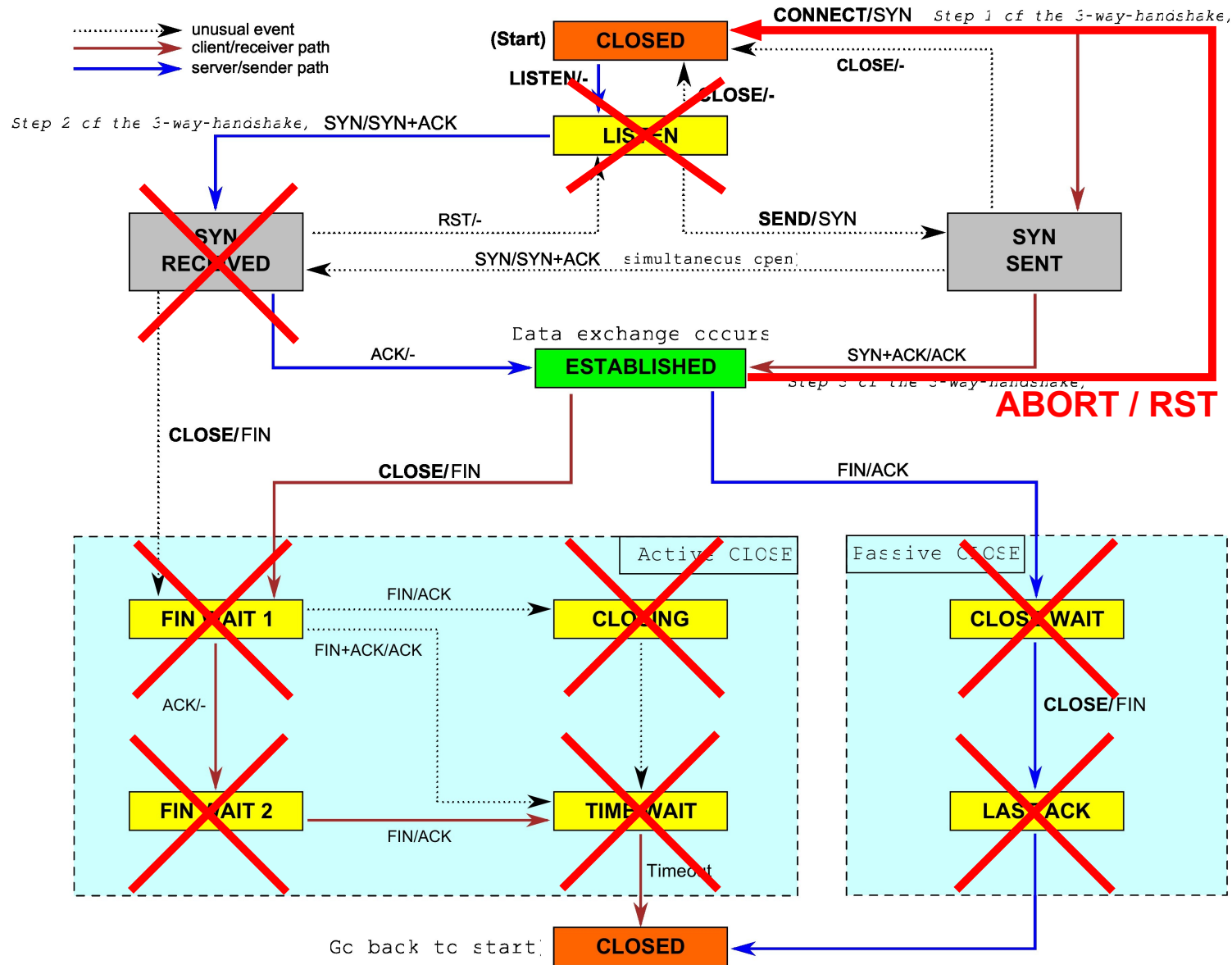
## no listening or receiving of data



Go back to start:

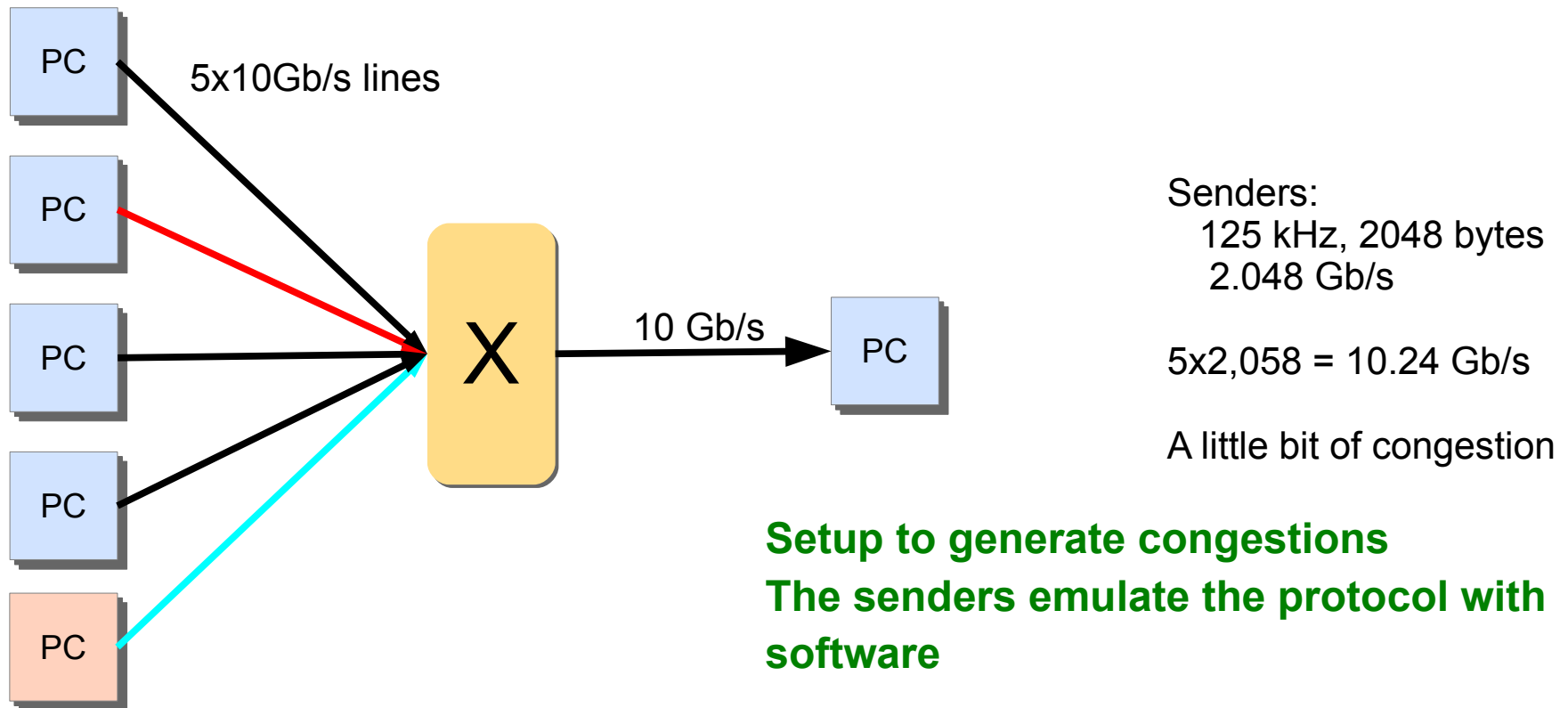
# Simplifying: TCP/IP (II)

no connection close – only brute force abort



# Congestion Control

- **TCP/IP has a sophisticated congestion control algorithm**
- **Do we need congestion control?**
  - Test setup with up to 5 10Gb/s links aggregated to one via a switch.
  - The PCs actually emulate our protocol with software.



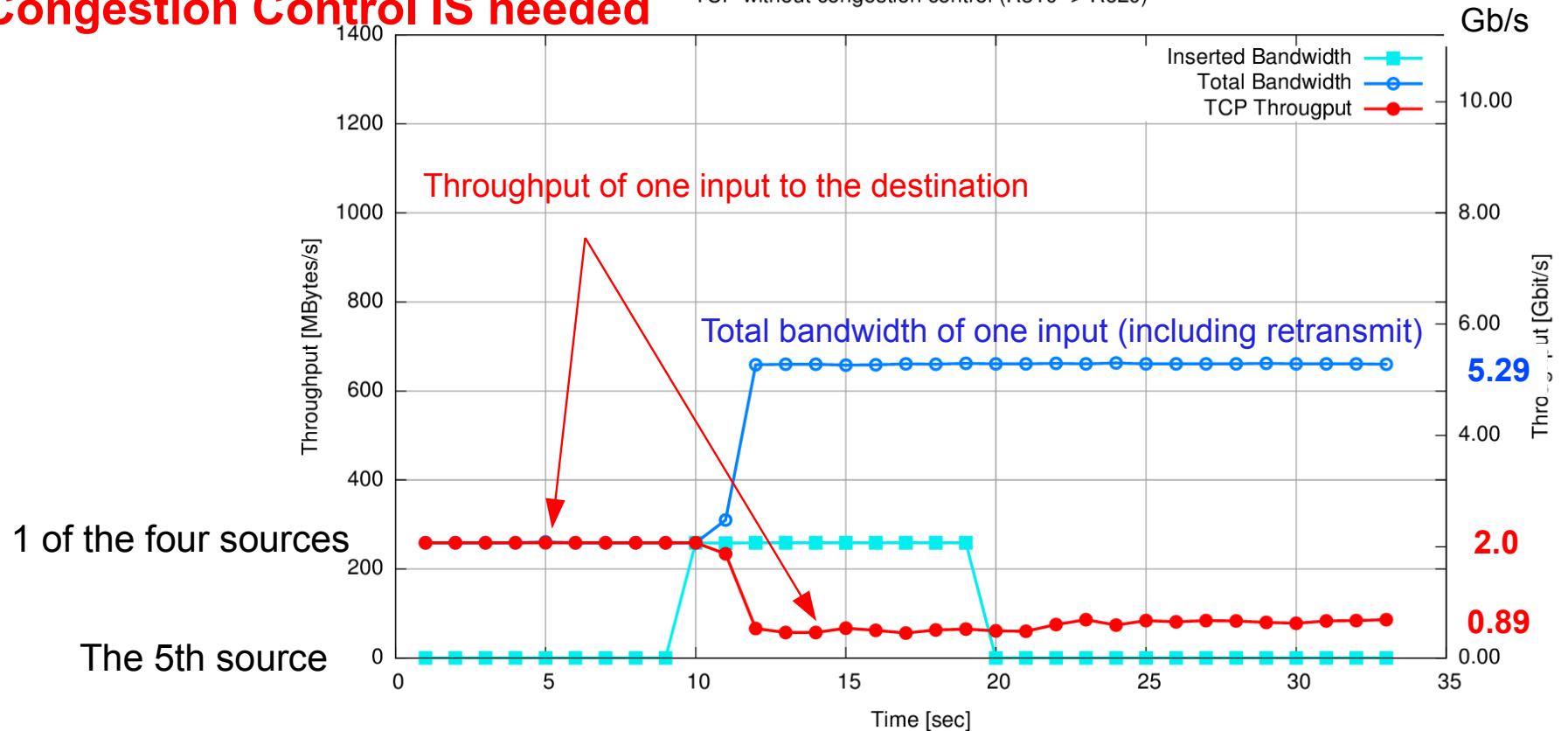


# Congestion Control

- **The system does not recover**
  - The available bandwidth is eaten up by a lot of re-transmits. Due to the re-transmits congestion becomes worse and worse and throughput becomes very small.

## Congestion Control IS needed

TCP without congestion control (R310 -> R620)

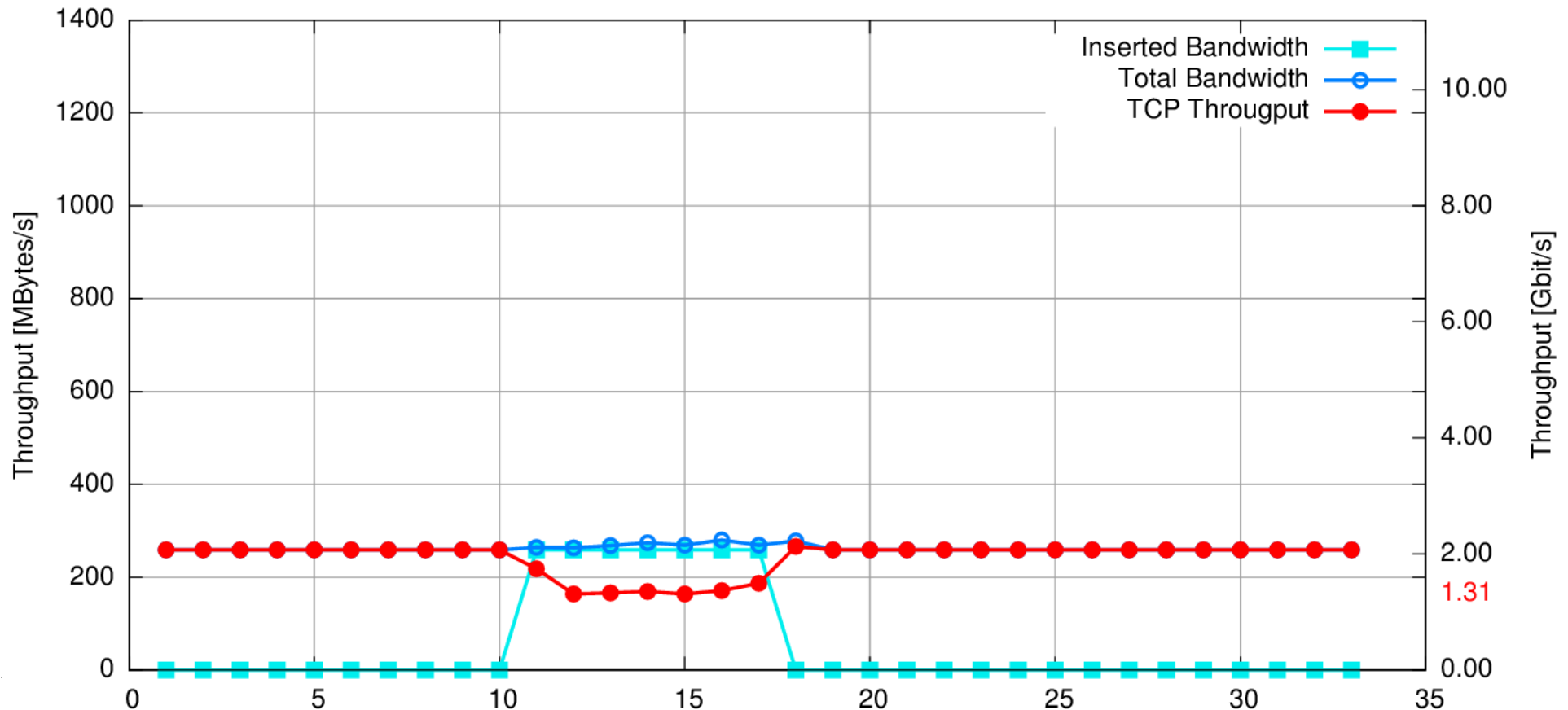


# Congestion control

- **Add simple congestion control**

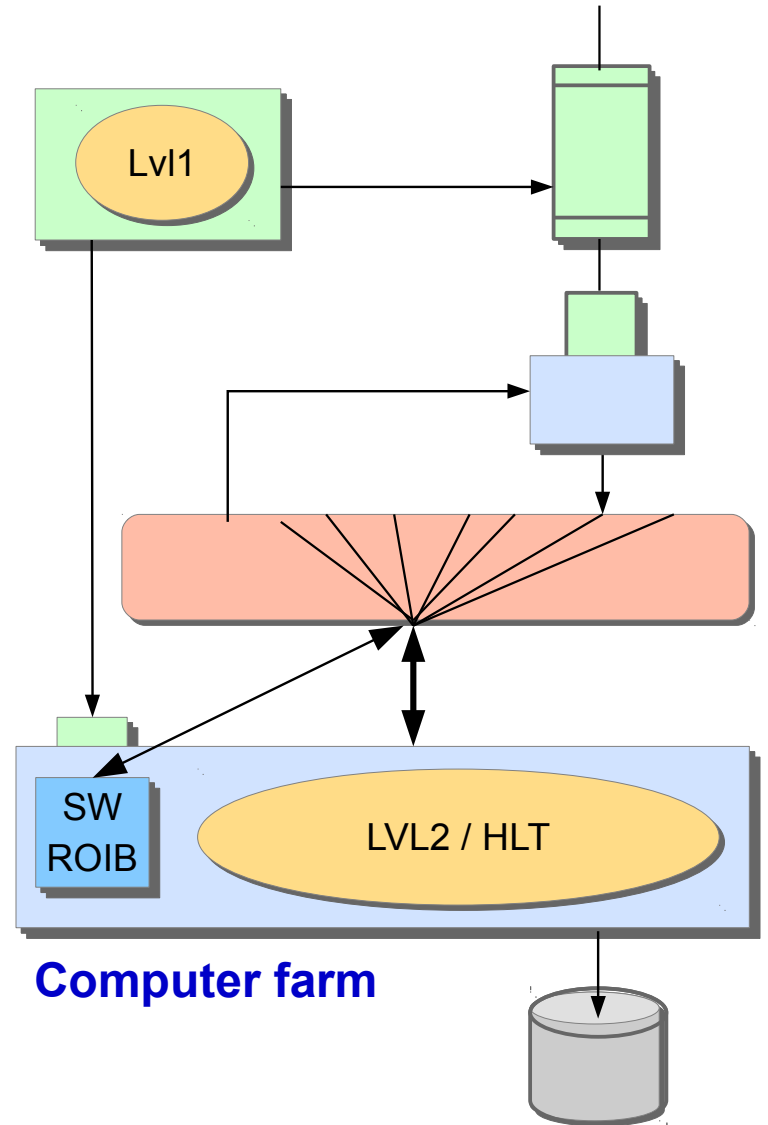
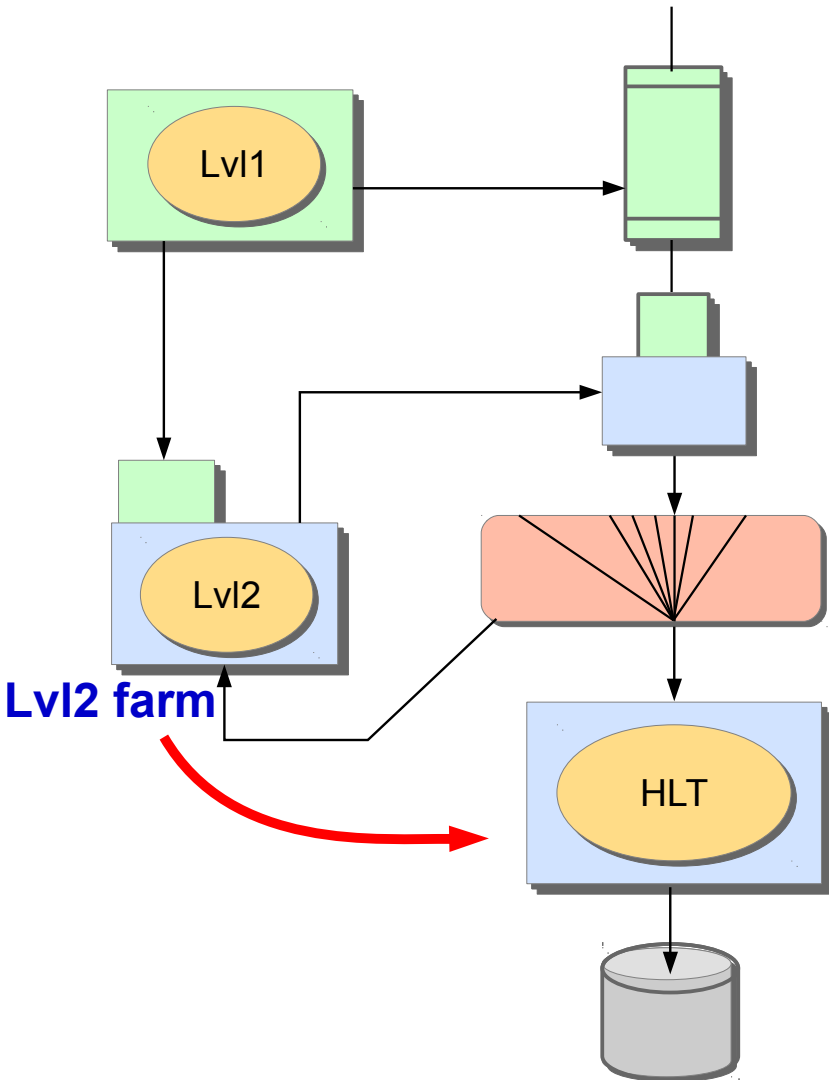
- Reduce the congestion window size
- Implement exponential backoff for re-transmit: double the re-transmit timeout if a packet is not acknowledged if a packet is not acknowledged

TCP without congestion control (R310 -> R620)



# ATLAS upgrade

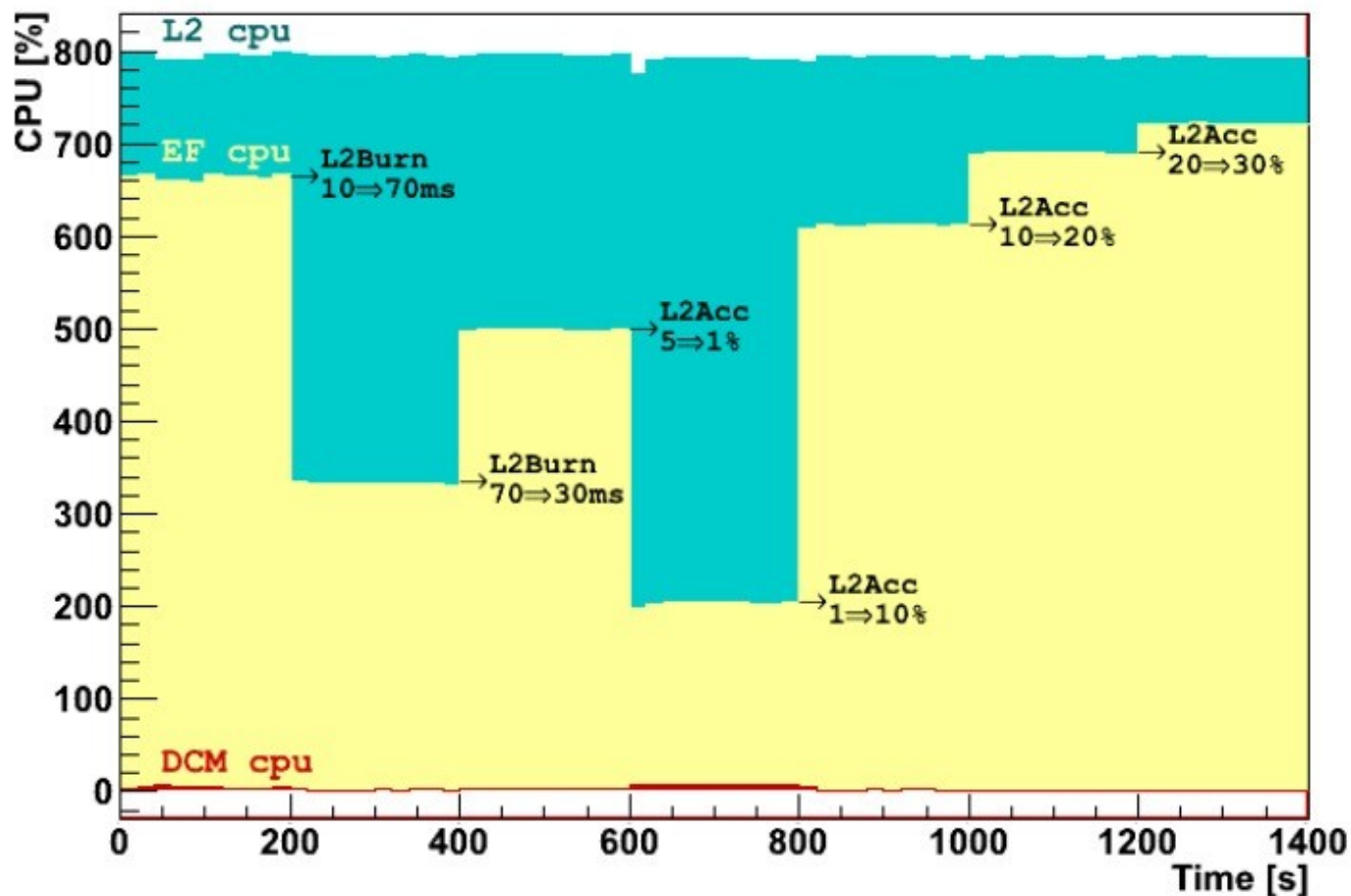
NOW FUTURE



# Atlas: Load Balancing

- **Tests with a Prototype**

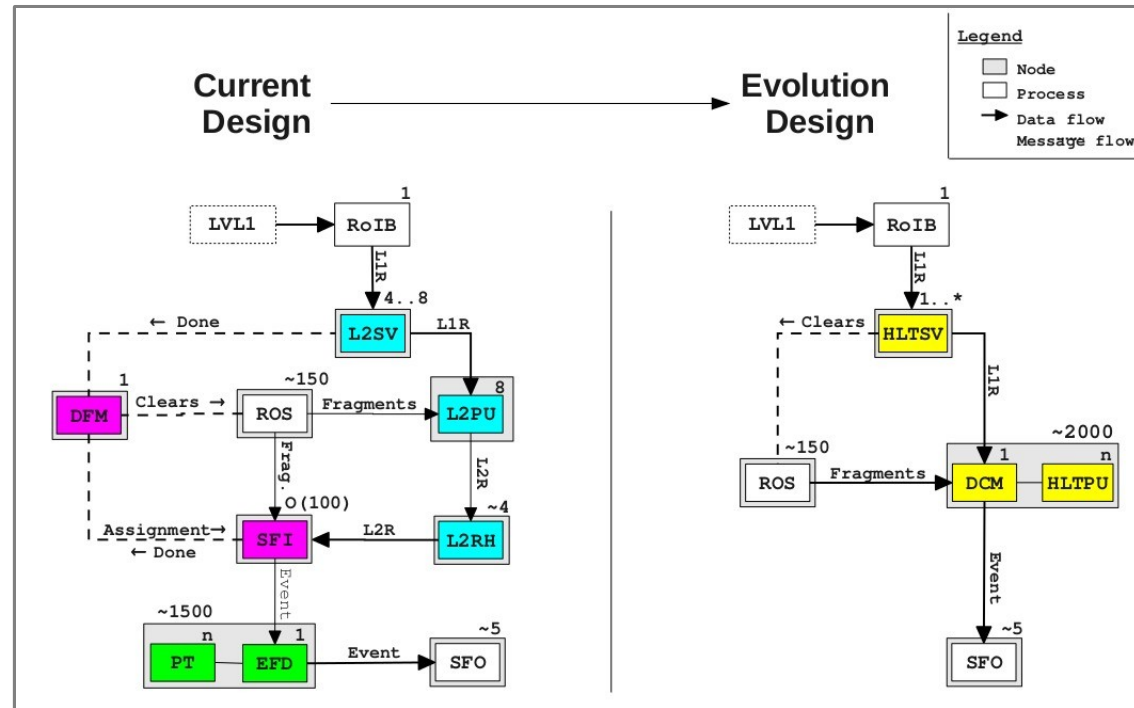
- The test shows that computing resources can be shuffled automatically between HLT and LVL2 according to the needs of the Physics run



# Atlas Data Flow Upgrade

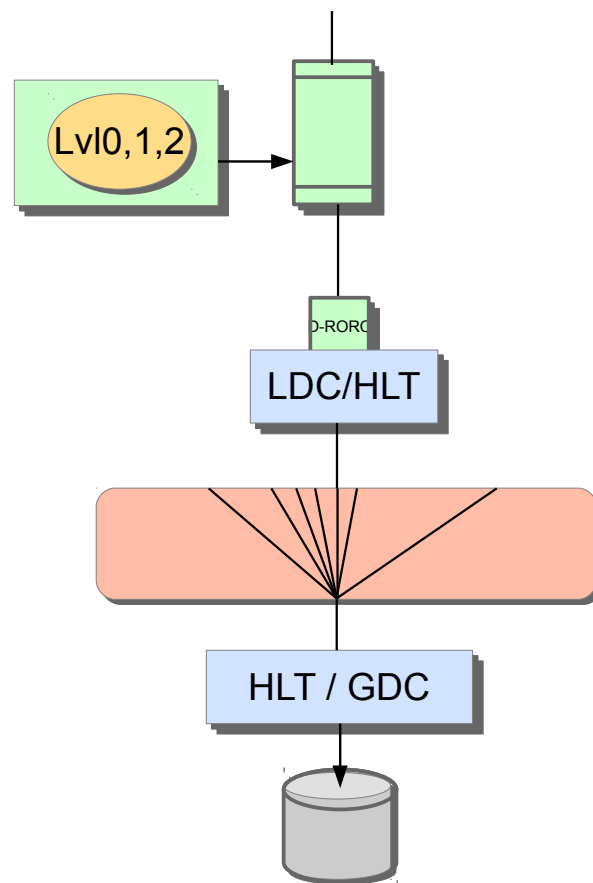
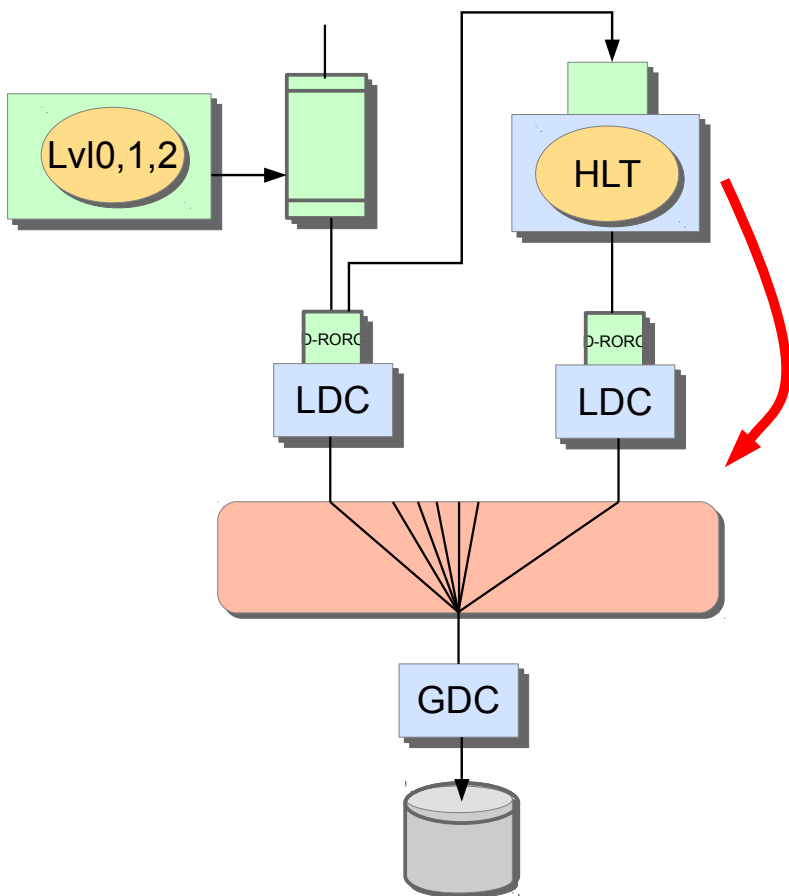
- **Unify Lvl2 Farm an HLT Farm**
- **Multiple benefits:**
  - Simpler architecture
    - Simpler Configuration
    - Less software applications
  - Automatic load balancing of the Farm Computers
    - Load can be divided as needed between Lvl2 and HLT
    - Better use of available computing resource
    - More flexibility for various HLT strategies
  - Less connections to ROS PCs

## Software Simplification

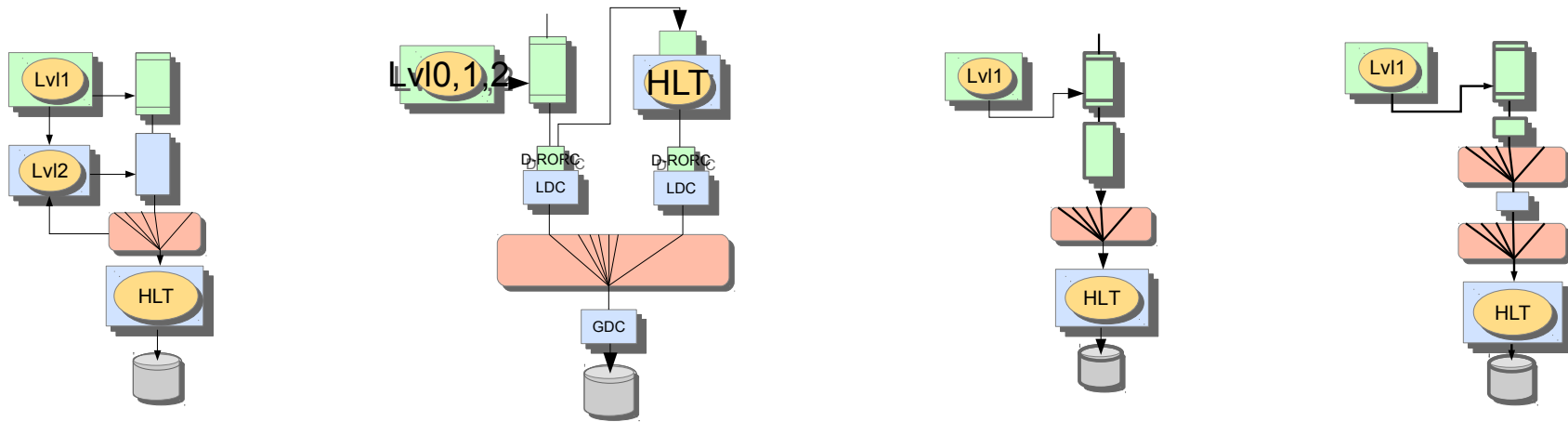


# ALICE: Unification of HLT and EVB Farm

NOW FUTURE



# Summary



- **Exiting Times ahead**

- The LHC experiments have started a large upgrade program
- Trigger systems will be completely exchanged
- DAQ system are being upgraded or plans to upgrade them exist.
- Technologies are changing to keep up with performance requirements
- This is the time to join since interesting developments are ongoing

# The END

---

**Thank you**

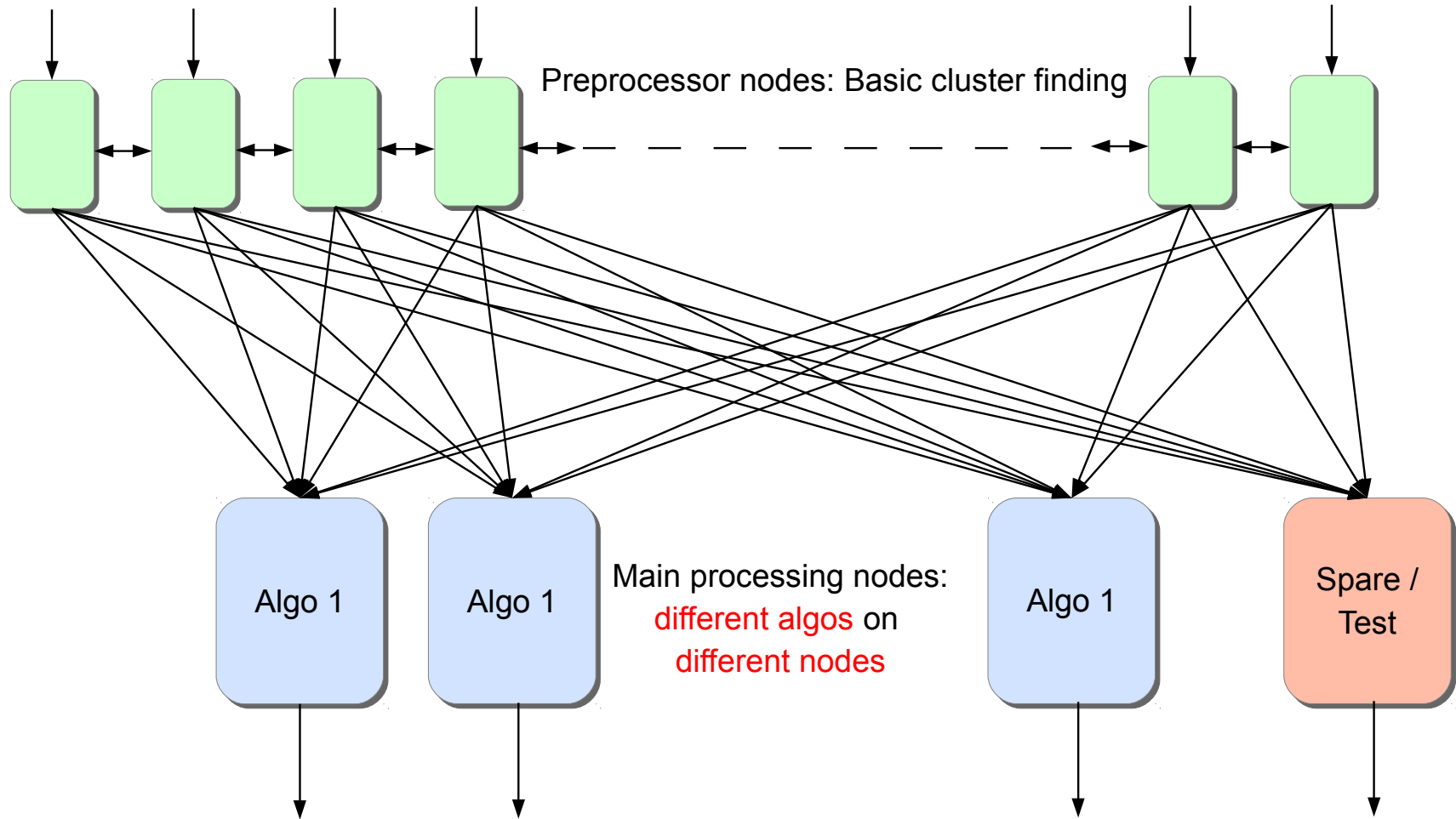
**and**

**Have a lot of fun in future projects !!!**



# Upgrade of CMS Calorimeter Trigger: Variant 1

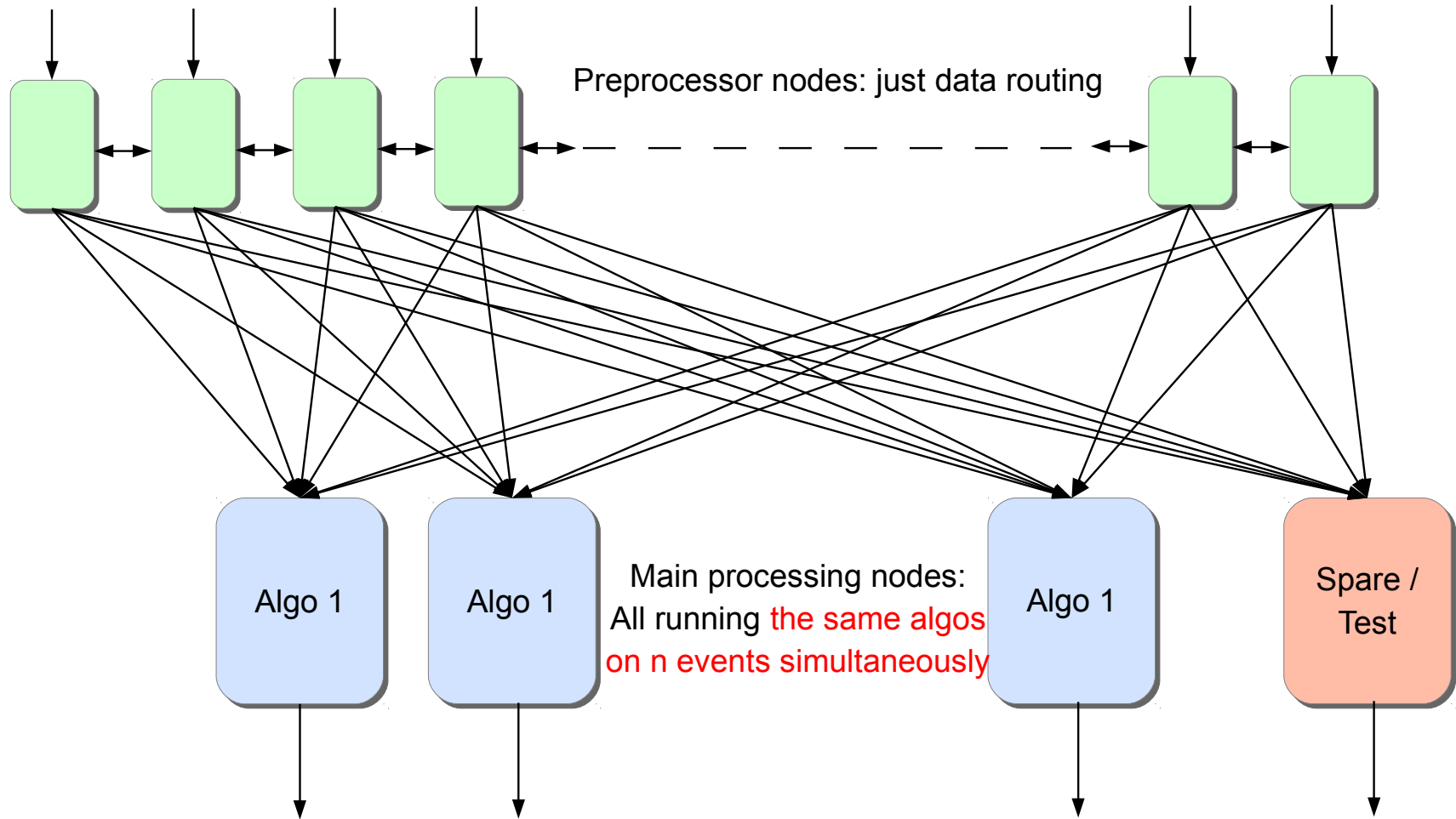
## Incoming Calorimeter Data



To Global Trigger Processor

# Upgrade of CMS Calorimeter Trigger: Variant 2

## Incoming Calorimeter Data



## To Global Trigger Processor

# Interesting for you...

---

- ..in case you want to participate

- “Playing” with high tech technology guaranteed

- ..but... the golden time of electronics are over...:

Once upon a time (in the 90s) a physicist could stick some FPGAs together, write some VHDL code and then claim: I have done an electronics board...

- Digital electronics has become challenging since analogue aspects play a major role in the meantime
- This is due to the high clock frequencies
- A connection becomes a transmission line where waves propagate
- System issues like power distribution, PCB layout become major challenges
  - A PCB board is a complicated passive electronic device
- But this is also a major fun in electronics design  
(see : Highspeed Digital Design: A Handbook of Black Magic)

# Interesting for you...

---

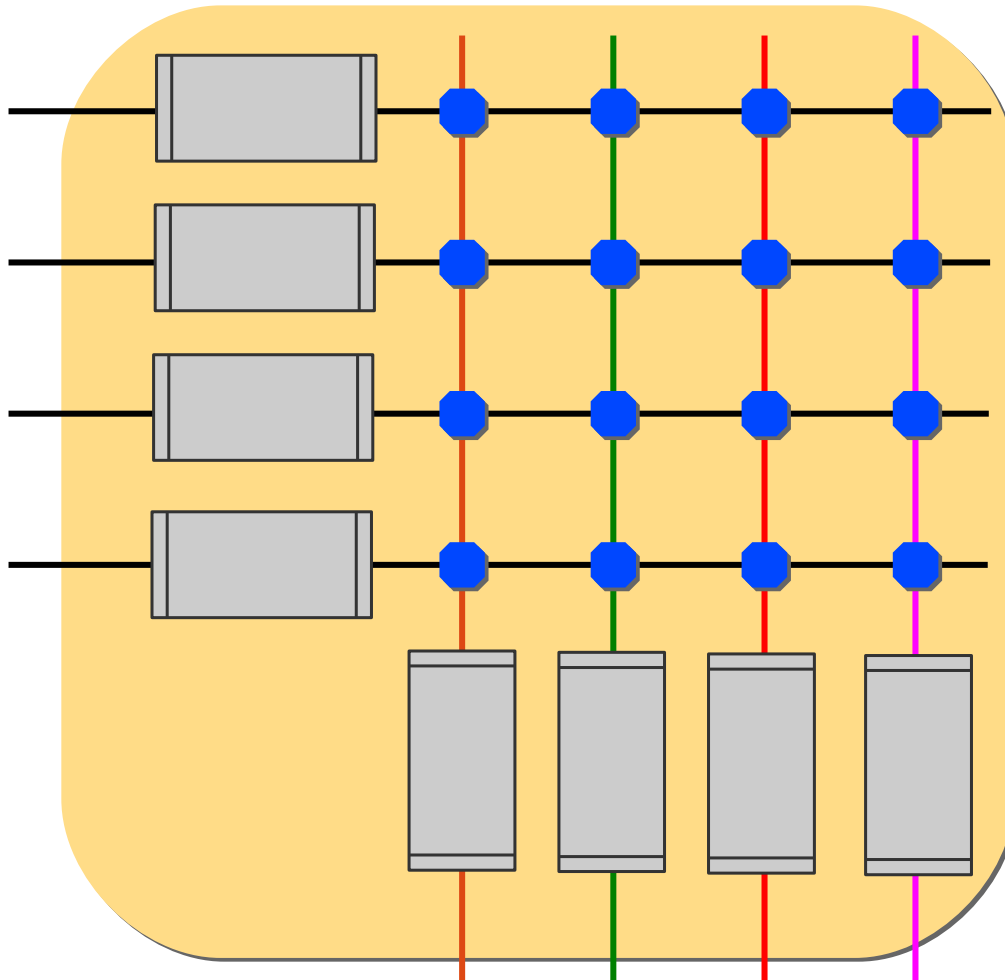
- **...in case you want to participate**
  - “Playing” with high tech technology guaranteed
- **...but... the golden time of electronics are over...:**  
**Once upon a time (in the 90s) a physicist could stick some FPGAs together, write some VHDL code and then claim: I have done an electronics board...**
  - Digital electronics has become challenging since analogue aspects play a major role in the meantime
  - This is due to the high clock frequencies
  - A connection becomes a transmission line where waves propagate
  - System issues like power distribution, PCB layout become major challenges
    - A PCB board is a complicated passive electronic device
  - But this is also a major fun in electronics design  
(see : Highspeed Digital Design: A Handbook of Black Magic)

---

# EXTRA SLIDES

# Switch implementation: Crossbar

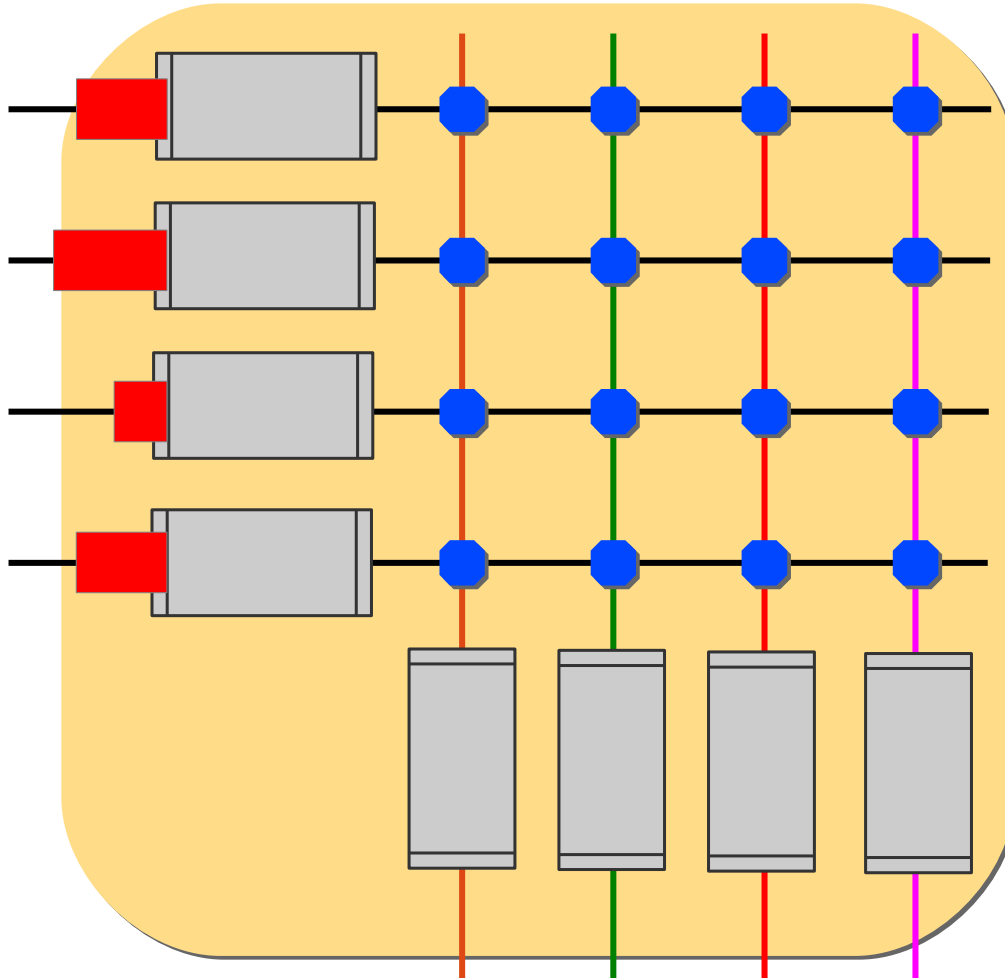
Mitigation of the congestion problem with additional memory.



Fifos can “absorb” congestion  
...until they are full...

# Switch implementation: Crossbar

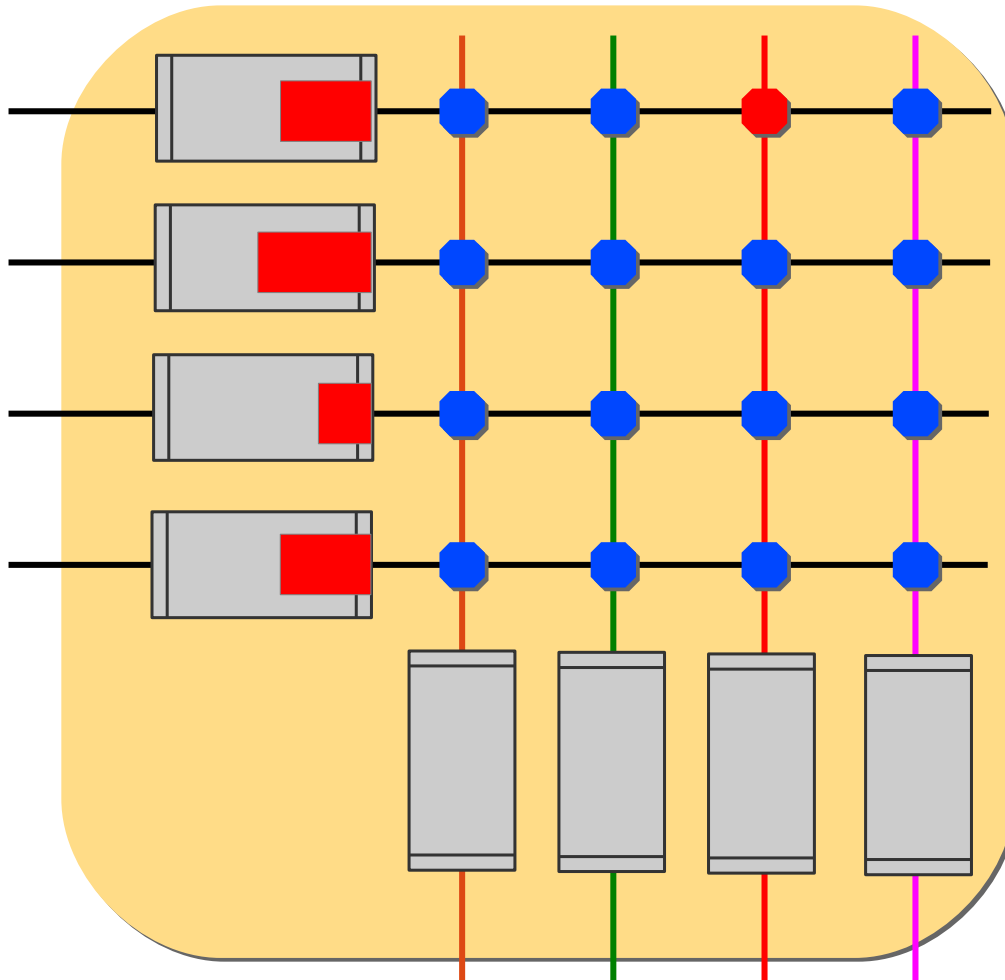
Mitigation of the congestion problem with additional memory.



Fifos can “absorb” congestion  
...until they are full...

# Switch implementation: Crossbar

Mitigation of the congestion problem with additional memory.

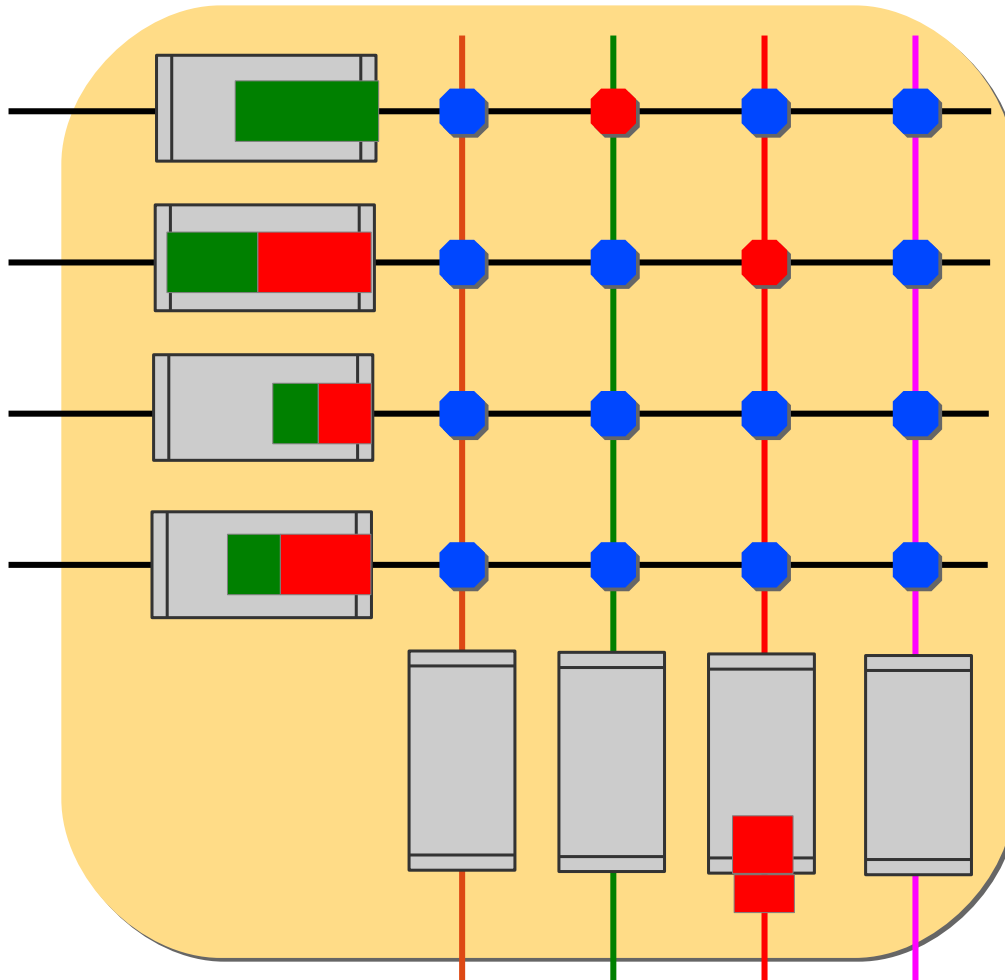


Fifos can “absorb” congestion  
...until they are full...



# Switch implementation: Crossbar

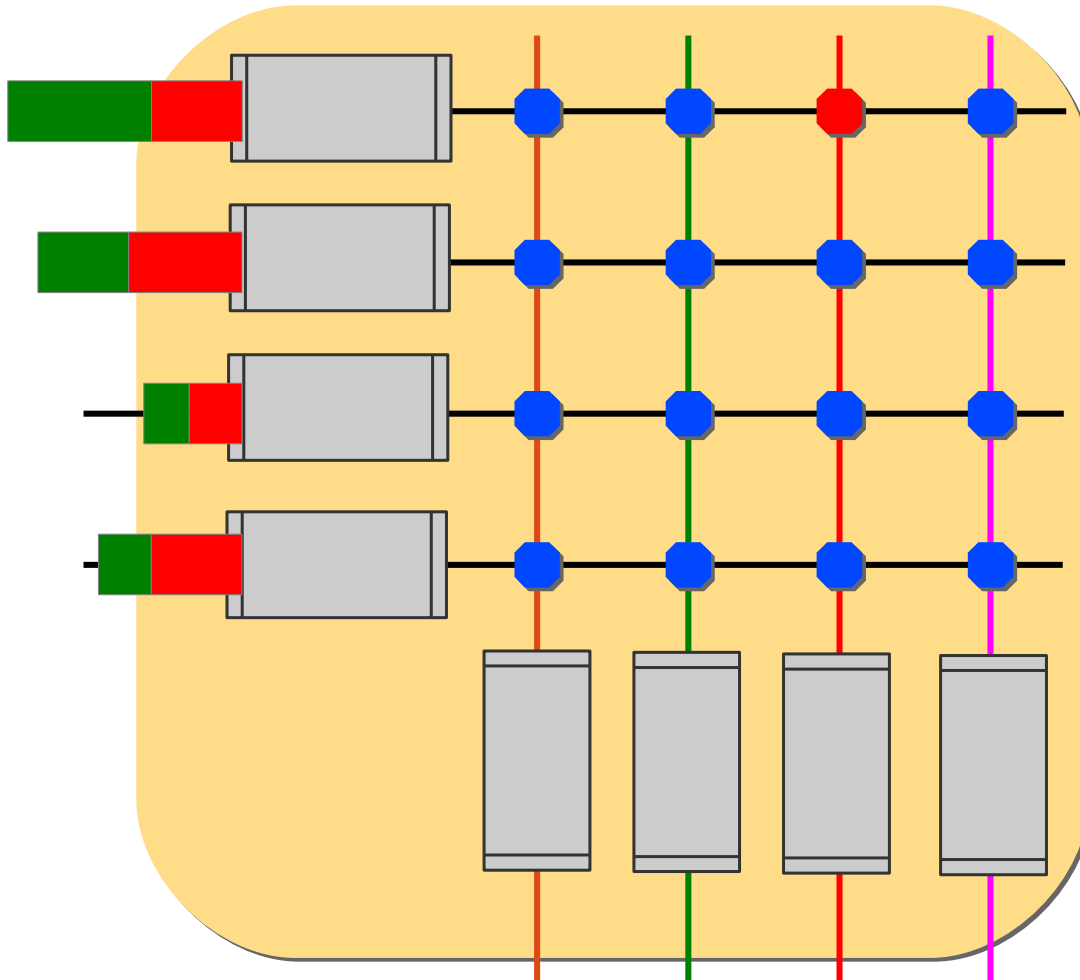
Mitigation of the congestion problem with additional memory.



Fifos can “absorb” congestion  
...until they are full...

# Switch implementation: Crossbar

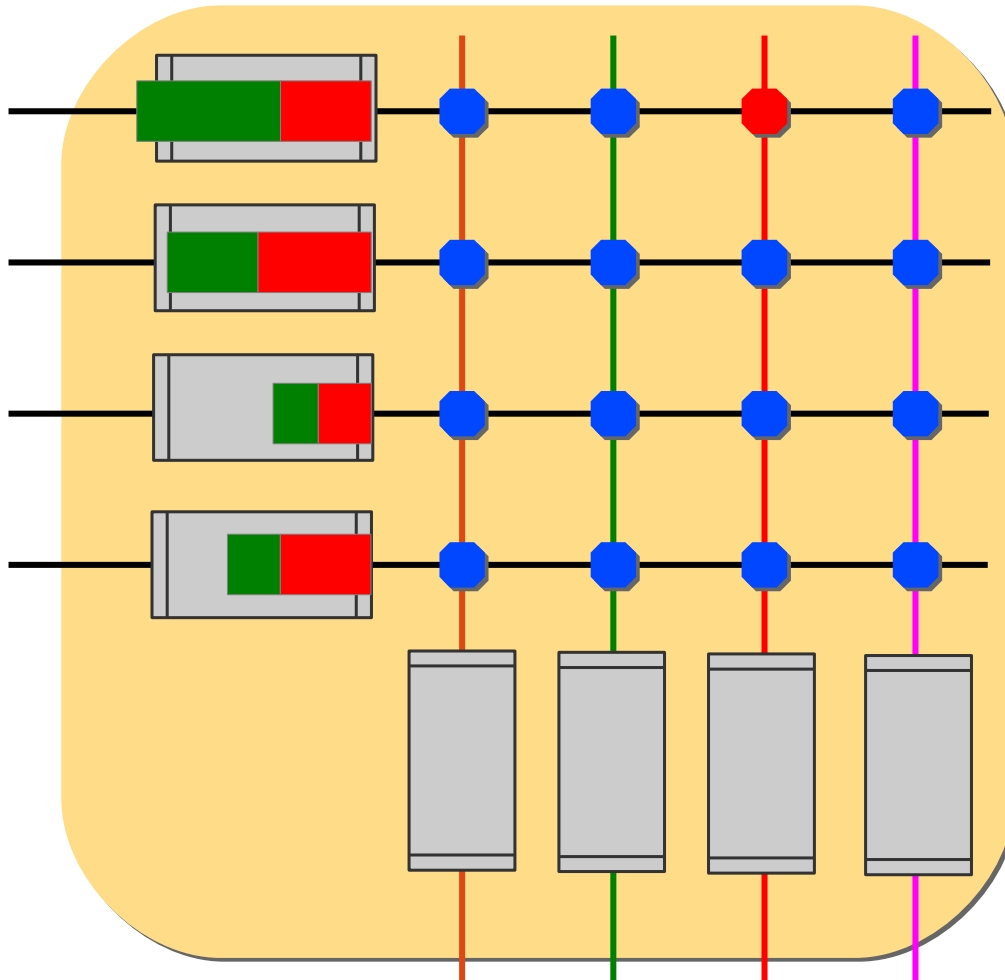
Mitigation of the congestion problem with additional memory.



Fifos can “absorb” congestion  
...until they are full...

# Switch implementation: Crossbar

Mitigation of the congestion problem with additional memory.



Fifos can “absorb” congestion  
...until they are full...

