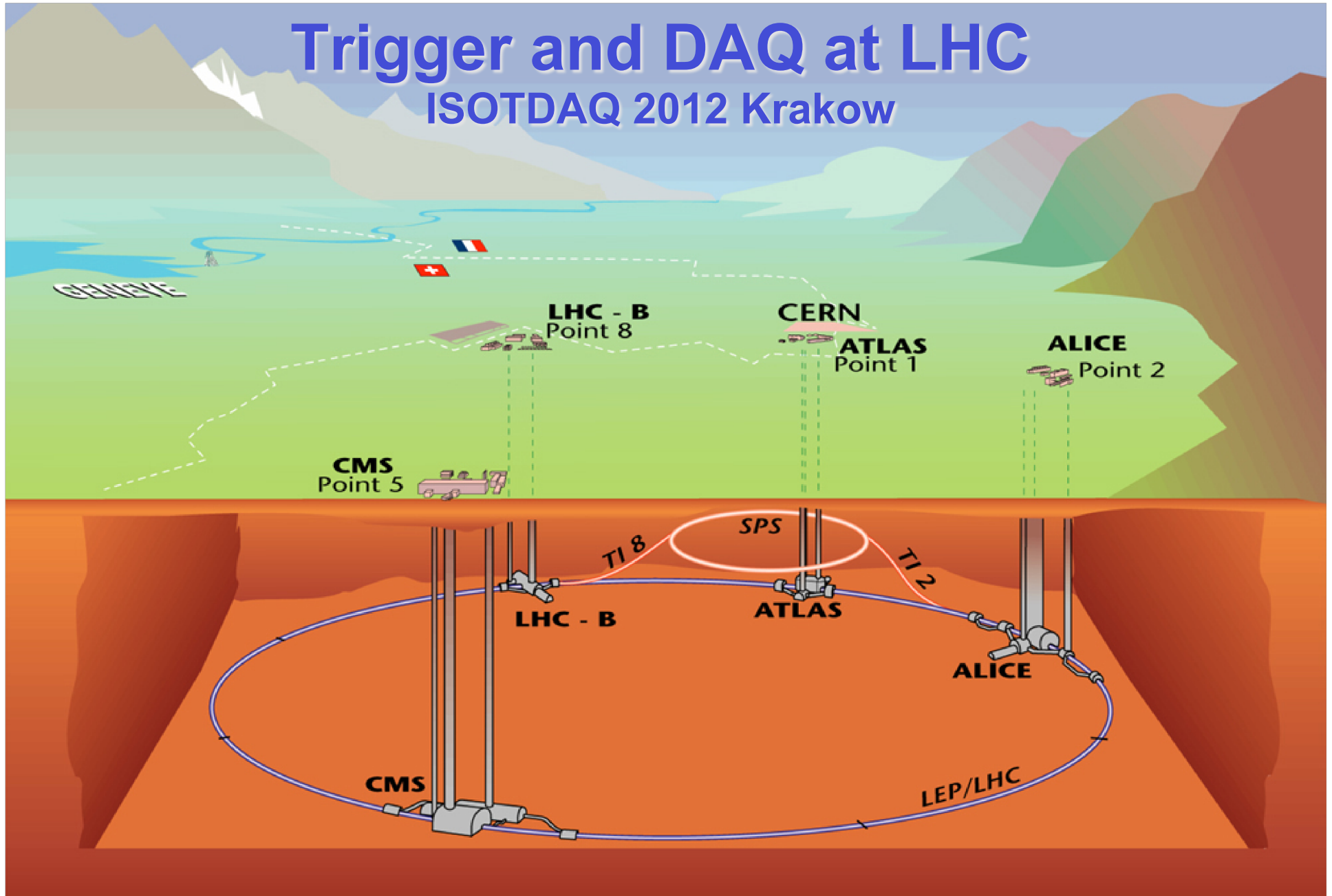


# Trigger and DAQ at LHC

ISOTDAQ 2012 Krakow



# Contents

---

## INTRODUCTION

The context: LHC & experiments

## PART1:

Trigger at LHC

Requirements & Concepts

Muon and Calorimeter triggers (CMS and ATLAS)

Specific solutions (ALICE, LHCb)

Hardware implementation

## Part2:

Data Flow, Event Building and higher trigger levels

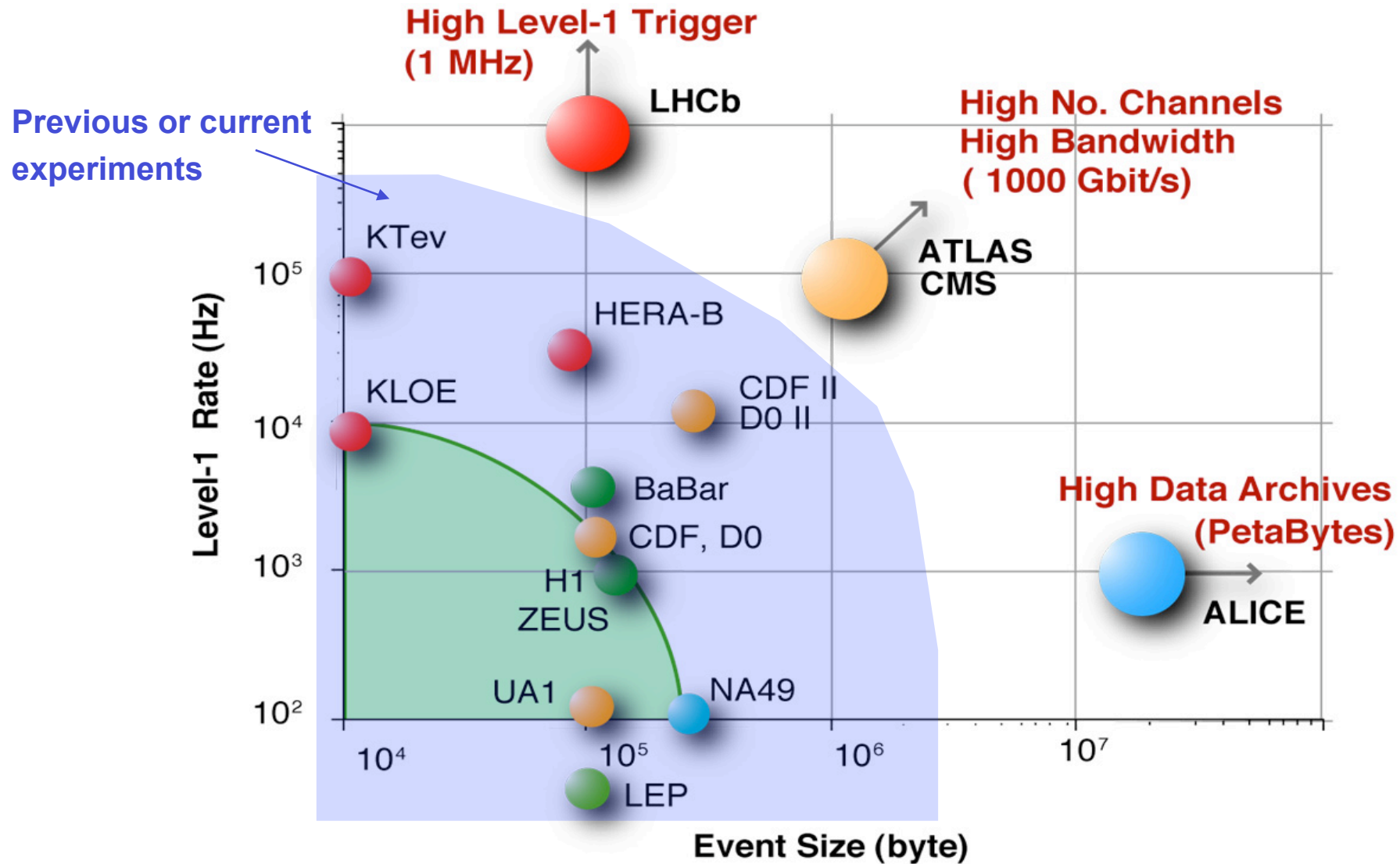
Data Flow of the 4 LHC experiments

Data Readout (Interface to central DAQ systems)

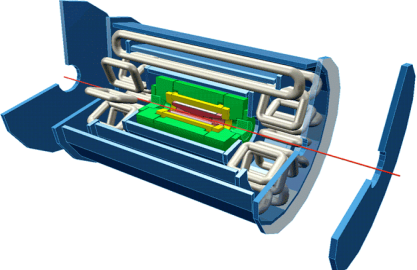
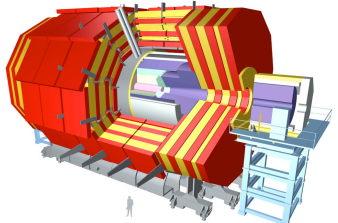
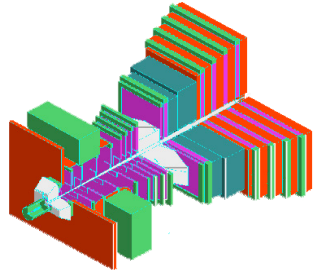
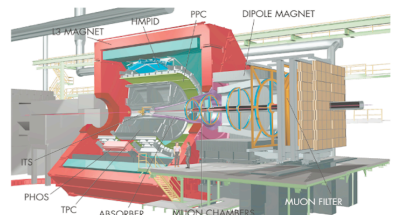
Event Building: CMS as an example

Software: some technologies

# LHC experiments: Lvl 1 rate vs size



# 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><math>10^5</math></b> LV-2 <b><math>3 \times 10^3</math></b>	<b>1.5 MB</b>	<b>4.5</b>	<b>300 (200)</b>
	<b>2</b>	LV-1 <b><math>10^5</math></b>	<b>1.0 MB</b>	<b>100</b>	<b>300 (200)</b> Pb-Pb 1500MB/s
	<b>2</b>	LV-0 <b><math>10^6</math></b>	<b>30 kB</b>	<b>30</b>	<b>60 (2 kHz)</b>
	<b>4</b>	Pb-Pb <b>500</b> p-p <b><math>10^3</math></b>	<b>70 MB</b> <b>2 MB</b>	<b>25</b>	<b>1250 (100)</b> <b>200 (100)</b>



---

# Data Flow: Architecture

# Data Acquisition:

# main tasks

- custom hardware
- PC
- network switch

Lvl1 trigger

Provide higher level trigger with partial event data

Lvl1 pipelines

Data readout from Front End Electronics

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

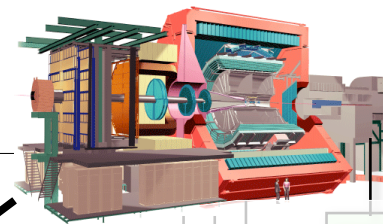
Assemble events in single location and provide to High Level Trigger (HLT)

Write selected events to permanent storage

Our "Standard Model" of Data Flow

# Data Acquisition:

# main tasks



- custom hardware
- PC
- network switch

Lvl1 trigger

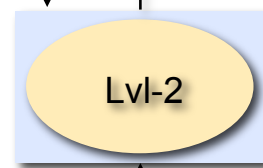


Lvl1 pipelines

Data readout from Front End Electronics

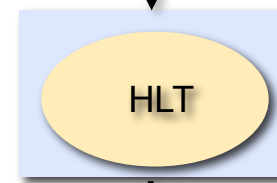
Temporary buffering of event fragments in **readout buffers**

Provide higher level trigger with partial event data



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

Our "Standard Model" of Data Flow

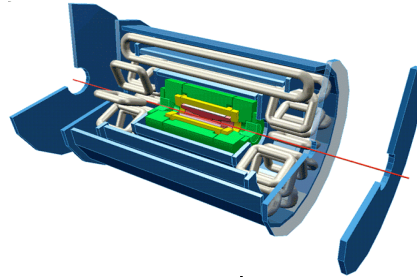


Write selected events to permanent storage

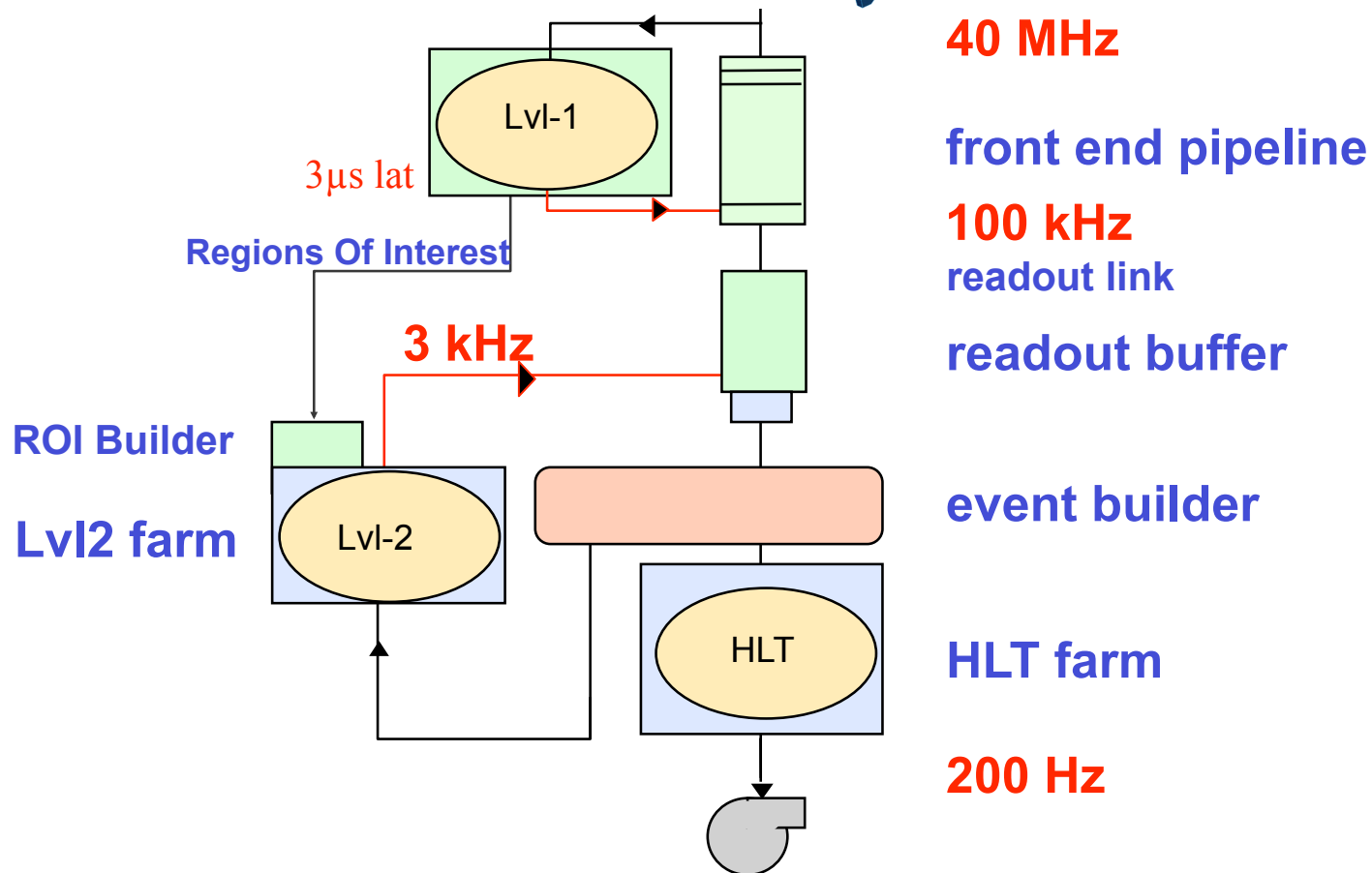


# Data Flow: ATLAS

- custom hardware
- PC
- network switch

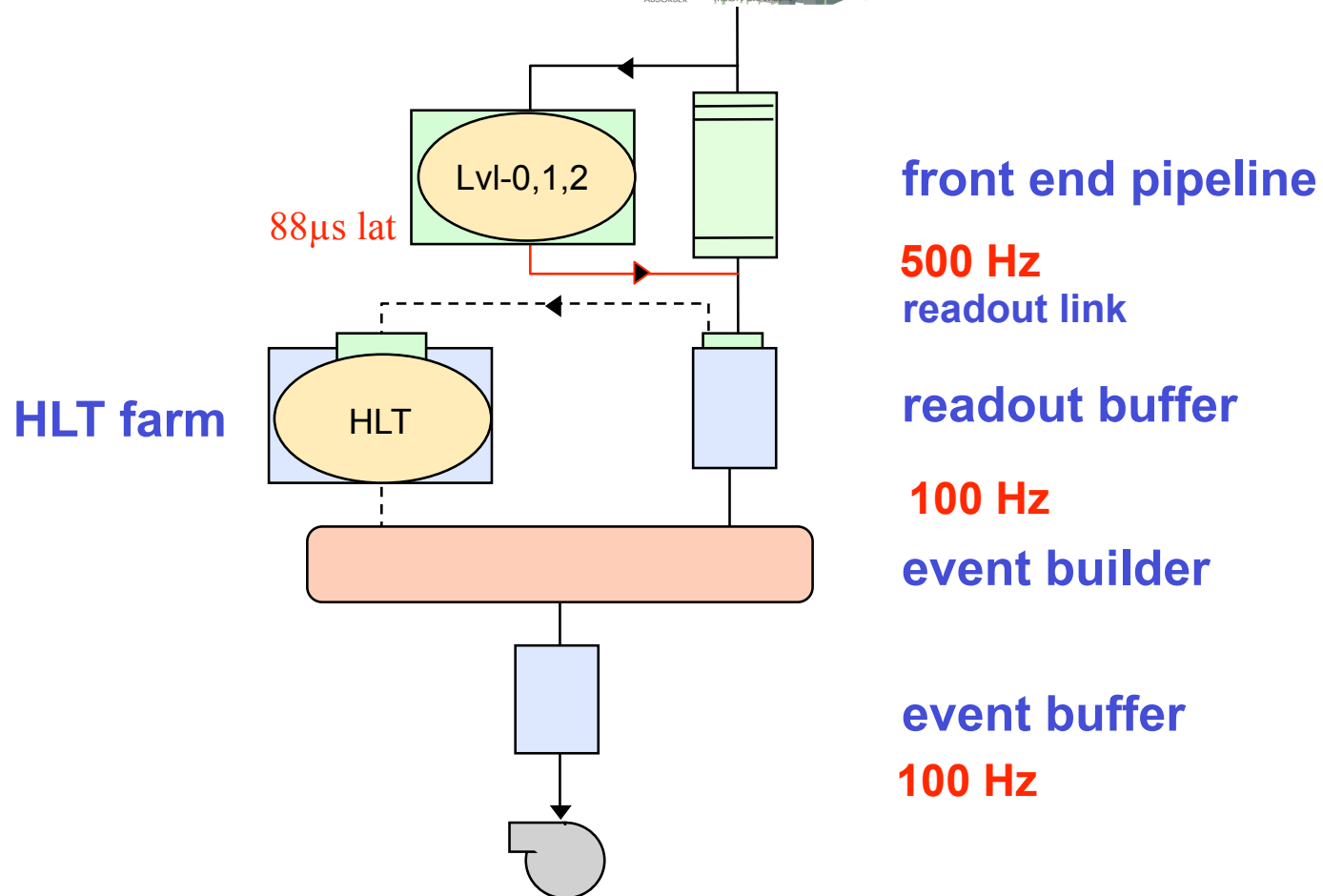
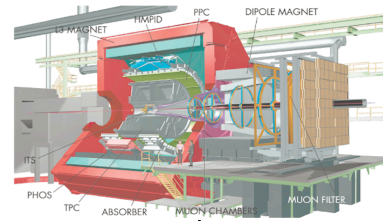


Region Of Interest (ROI):  
Identified by Lvl1. Hint for Lvl2  
to investigate further



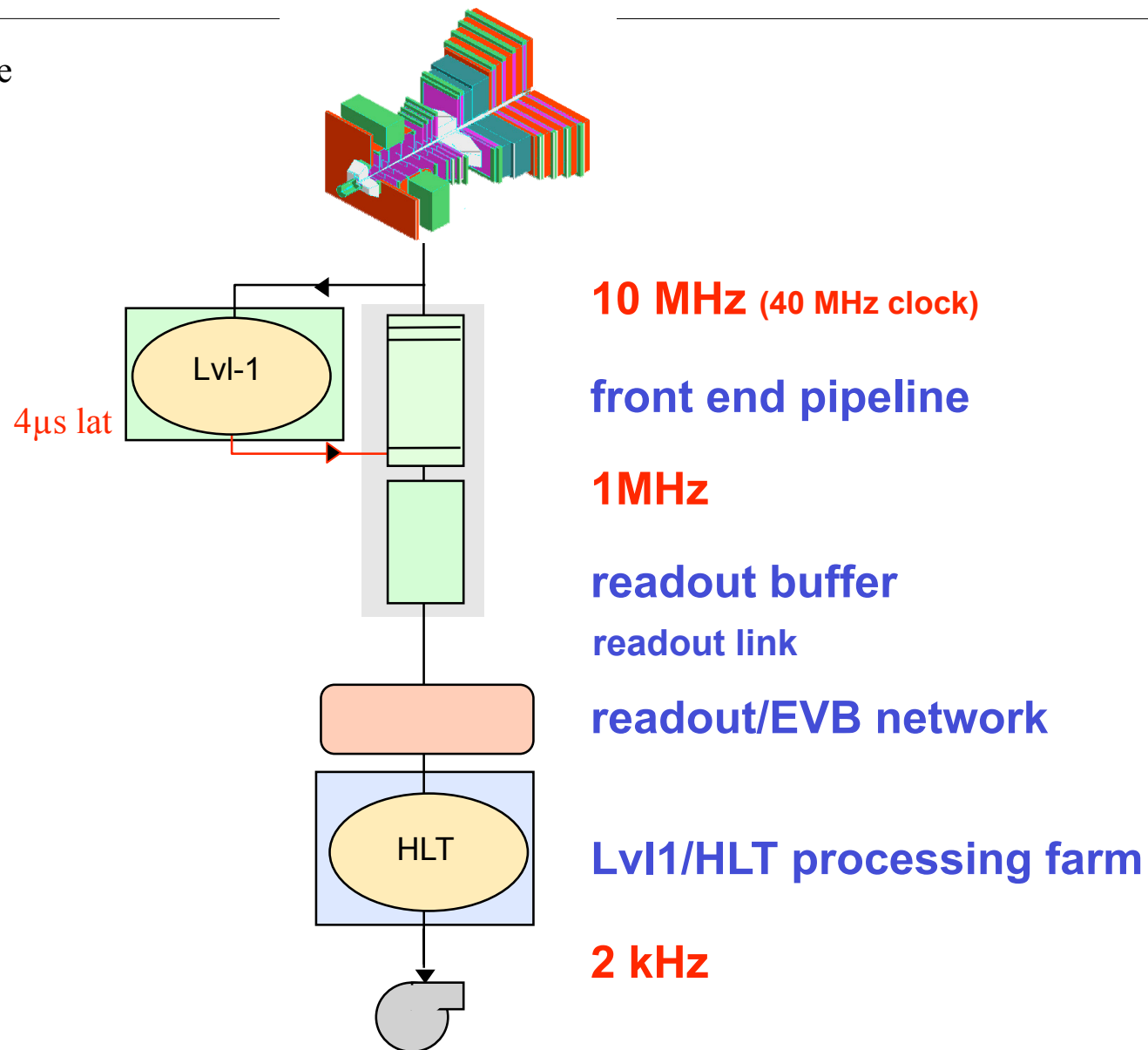
# Data Flow: ALICE

- custom hardware
- PC
- network switch



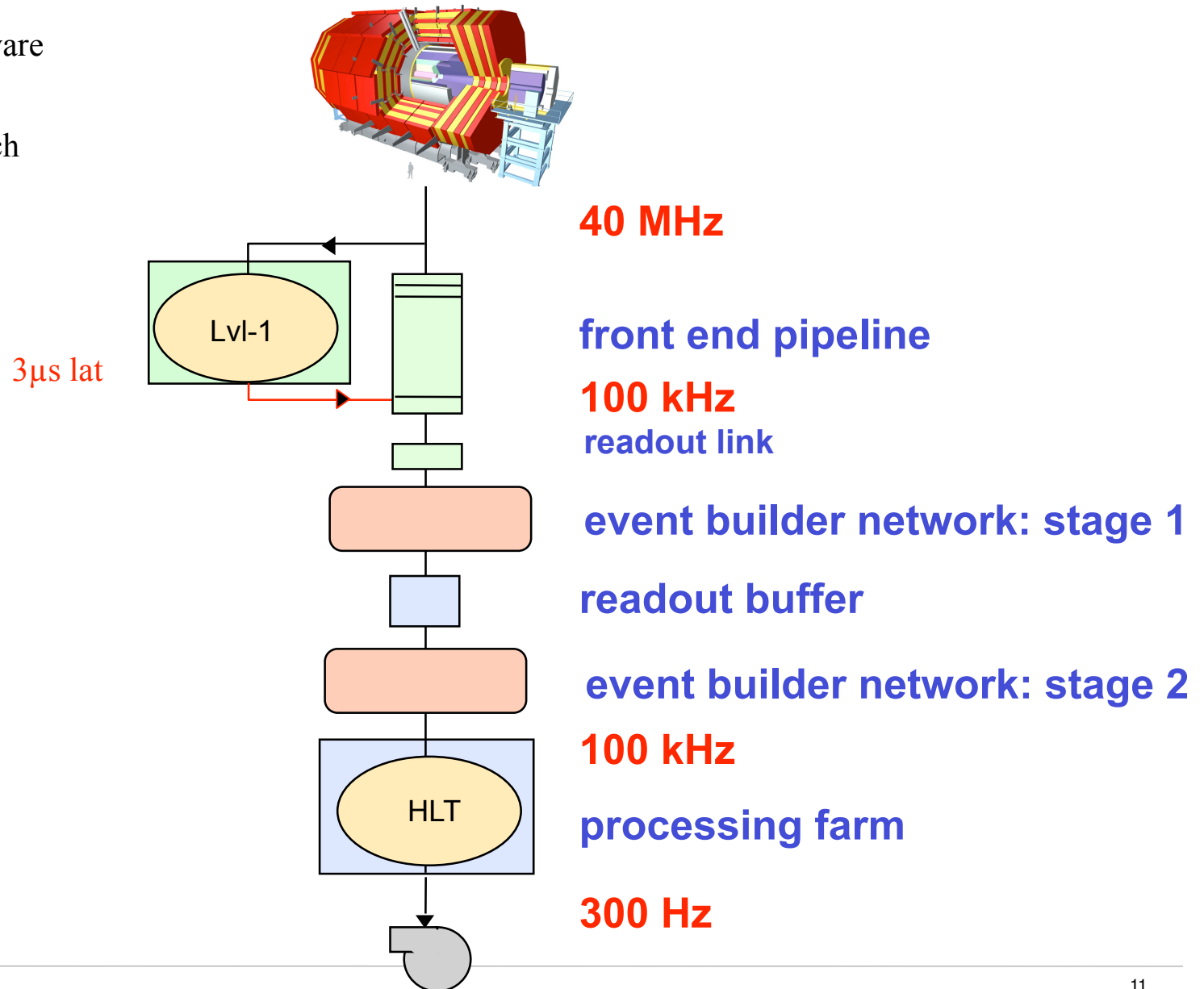
# Data Flow: LHCb

- custom hardware
- PC
- network switch



# Data Flow: CMS

- custom hardware
- PC
- network switch

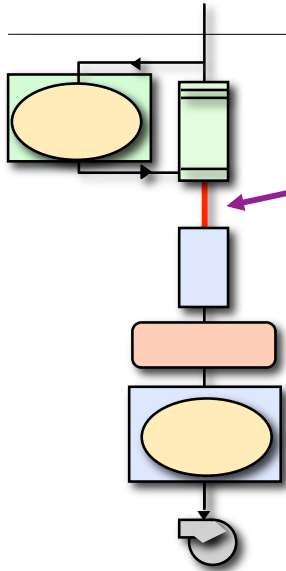




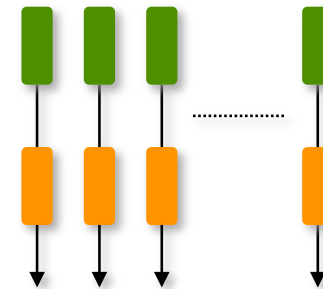
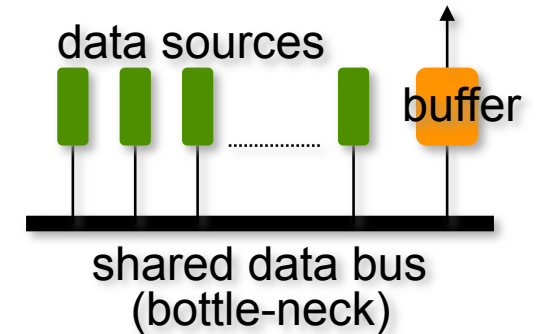
---

# Data Flow: Readout Links

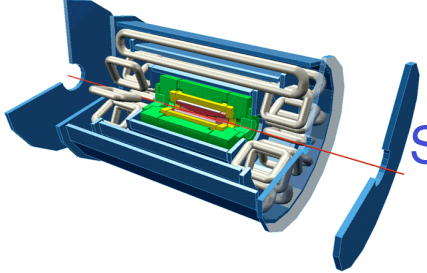
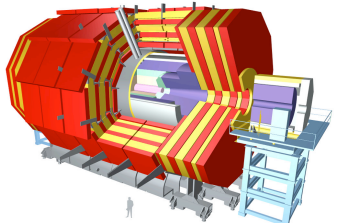
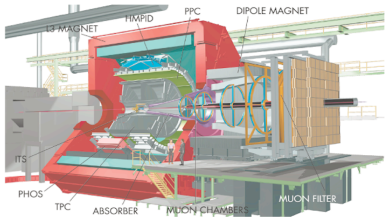
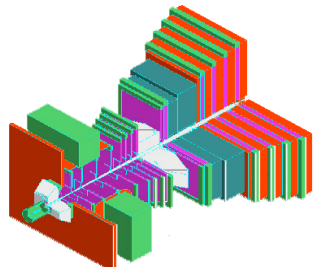
# Data Flow: Data Readout



- **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 market:**
  - 198x: ISA, SCSI(1979),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

				Flow Control
	SLINK	Optical: 160 MB/s Receiver card interfaces to PC.	≈ 1600 Links	Yes
	SLINK 64	LVDS: 400 MB/s (max. 15m) (FE on average: 200 MB/s to readout buffer) Receiver card interfaces to commercial NIC (Network Interface Card)	≈ 500 links	yes
	DDL	Optical 200 MB/s Half duplex: Controls FE (commands, Pedestals, Calibration data) Receiver card interfaces to PC	≈ 500 links	yes
	TELL-1 & GbE Link	Copper quad GbE Link Protocol: IPv4 (direct connection to GbE switch) Forms “Multi Event Fragments” Implements readout buffer	≈ 400 links	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!

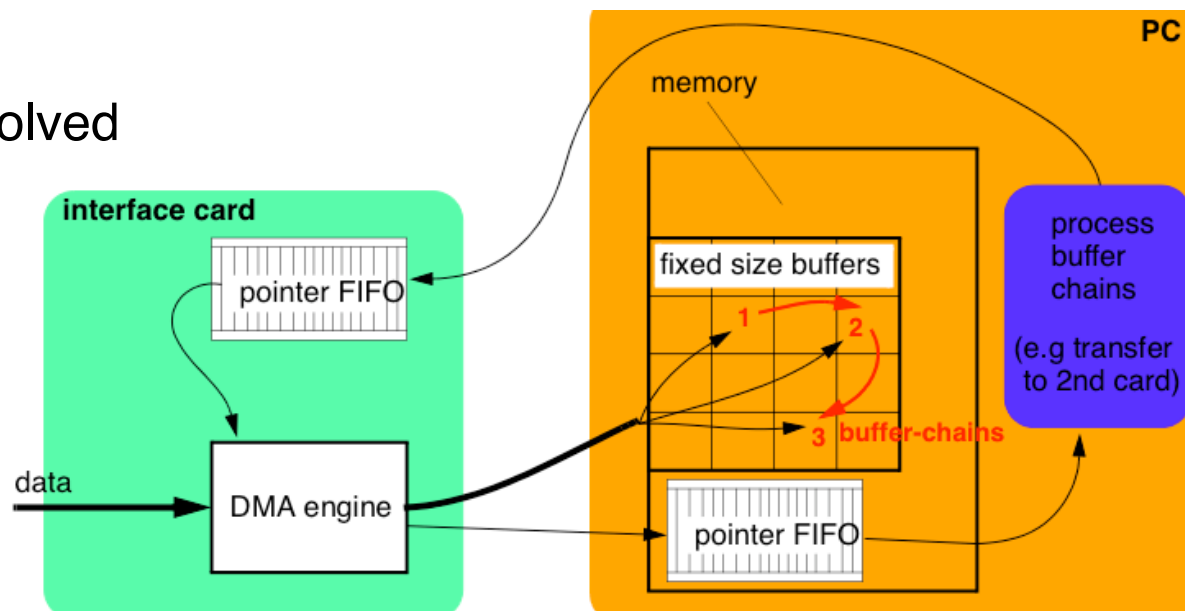
## Solution: Buffer-Loaning

- Hardware shuffles data via DMA (Direct Memory Access) engines
- Software maintains tables of buffer-chains

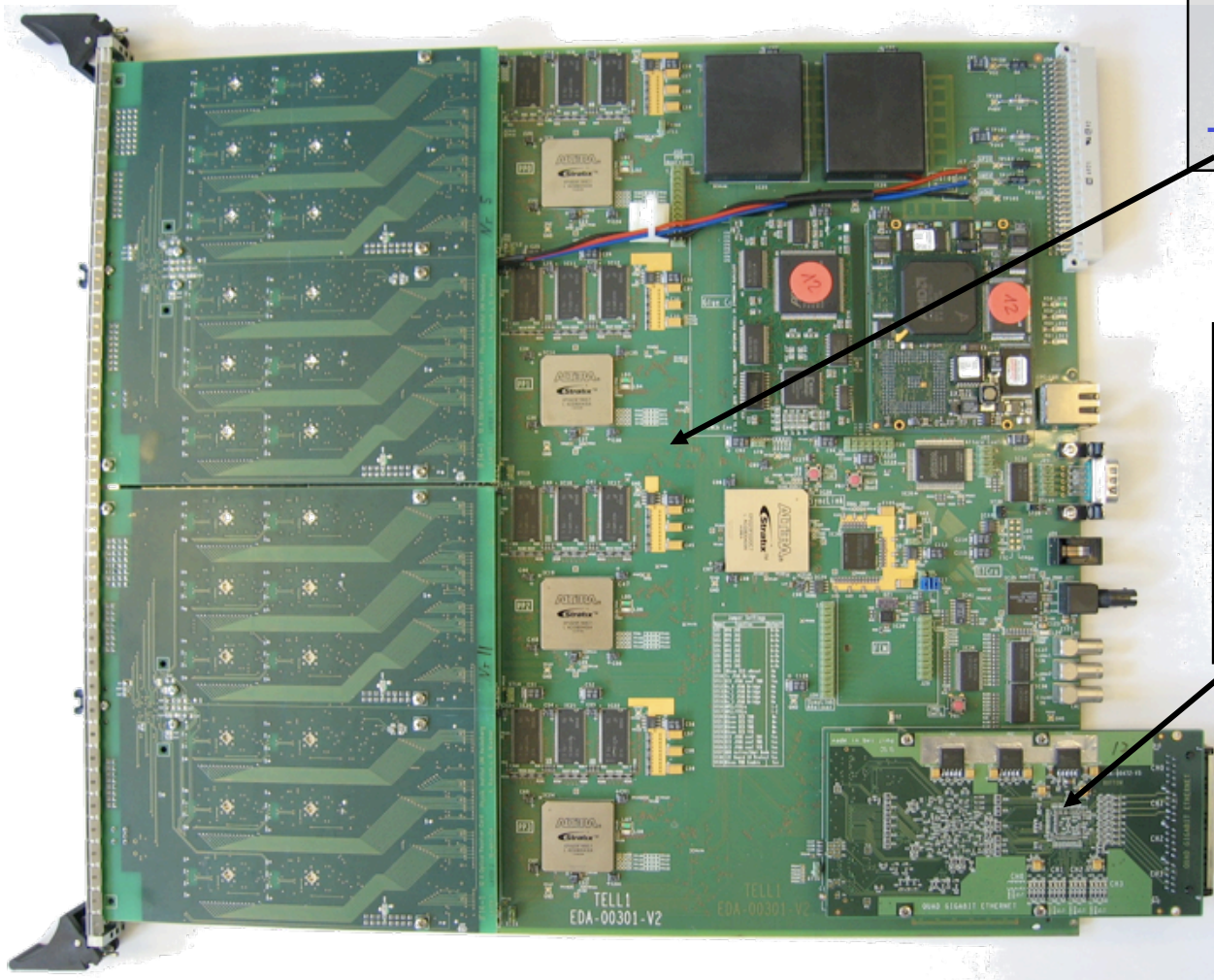
## Advantage:

- No CPU copy involved

used for links of  
Atlas, CMS, ALICE



# Example readout board: LHCb



## Main board:

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

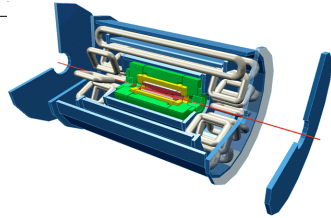
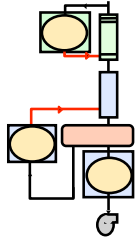
## Readout Link

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

---

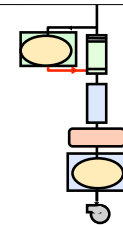
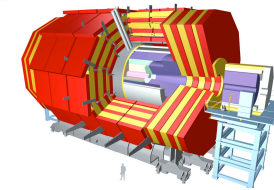
# Event Building: example CMS

# Event Building: Atlas vs CMS



**Challenging**

**Readout Buffer**



**“Commodity”**

Concept of “Region Of Interest” (ROI)

Increased complexity

- ROI generation (at Lvl1)
- ROI Builder (custom module)
- selective readout from buffers

Implemented with commercial PCs

**“Commodity”**

**Event Builder**

**Challenging**

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

100kHz @ 1 MB = 100 GB/s

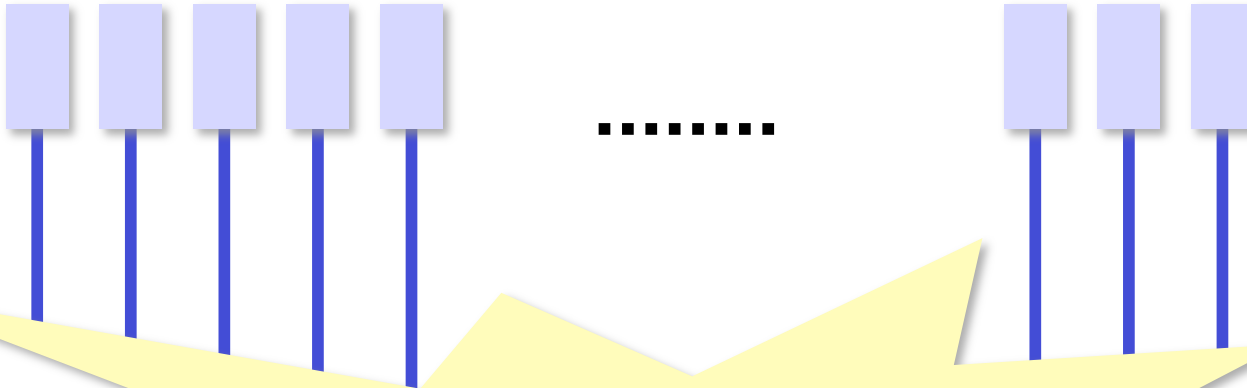
Increased complexity:

- traffic shaping
- specialized (commercial) hardware



# Networking: EVB traffic

Readout Buffers



Network Switch

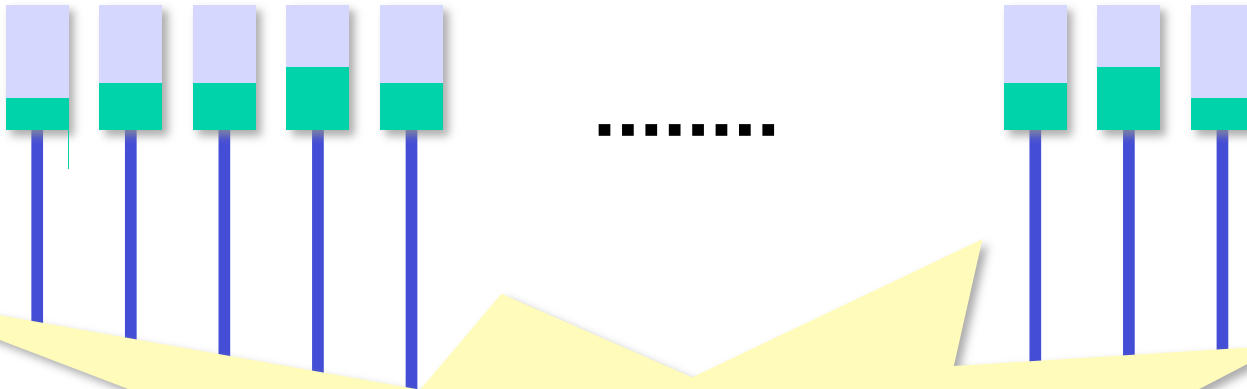
“Builder Units”



# Networking: EVB traffic

Readout Buffers

Lvl1 event



Network Switch

“Builder Units”



# Networking: EVB traffic

Readout Buffers (N)

Lvl1 event



Network Switch

**EVB traffic**  
all sources send to  
the same destination  
at (almost) concurrently.  
**Congestion**

Builder Units (M)



# Event Building dilemma

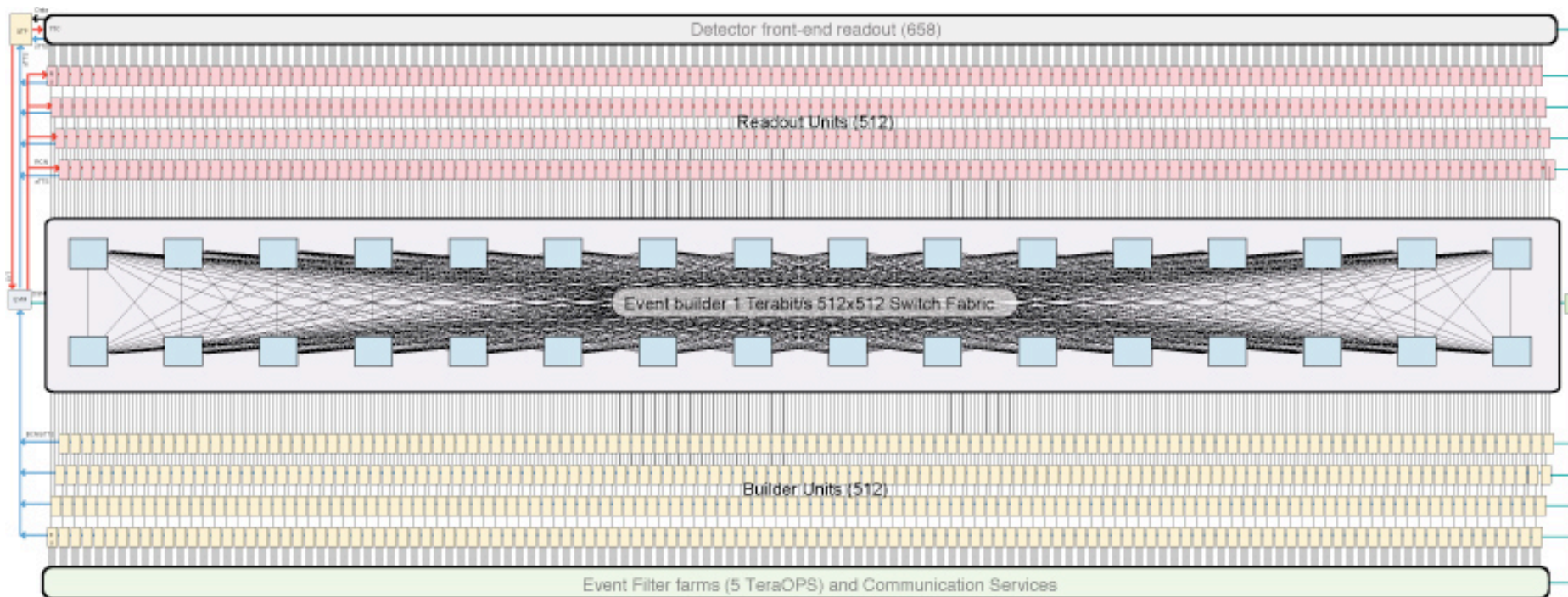
To be avoided:



**In spite of the Event builder traffic pattern congestion should be avoided.**



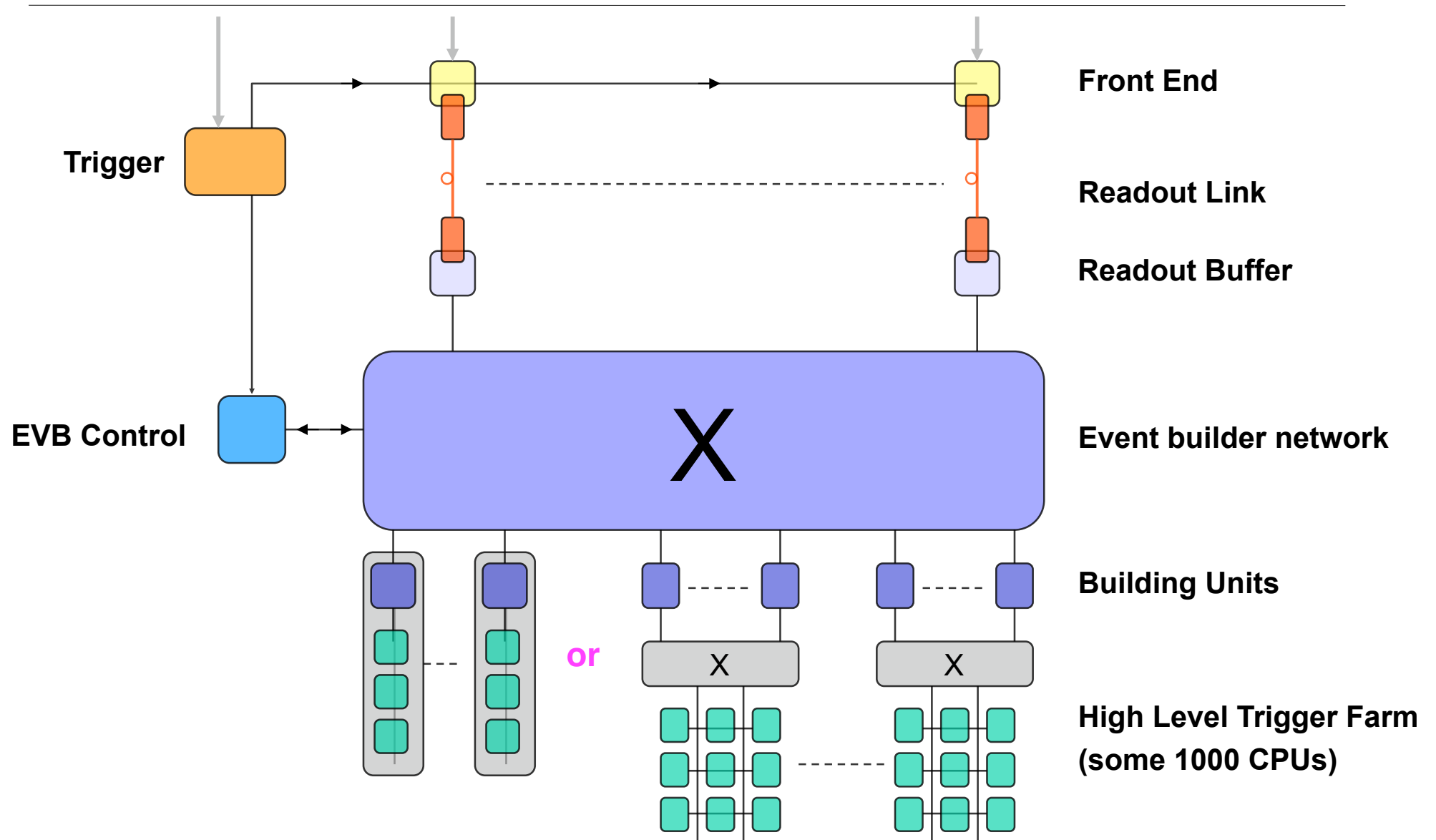
# EVB example: CMS



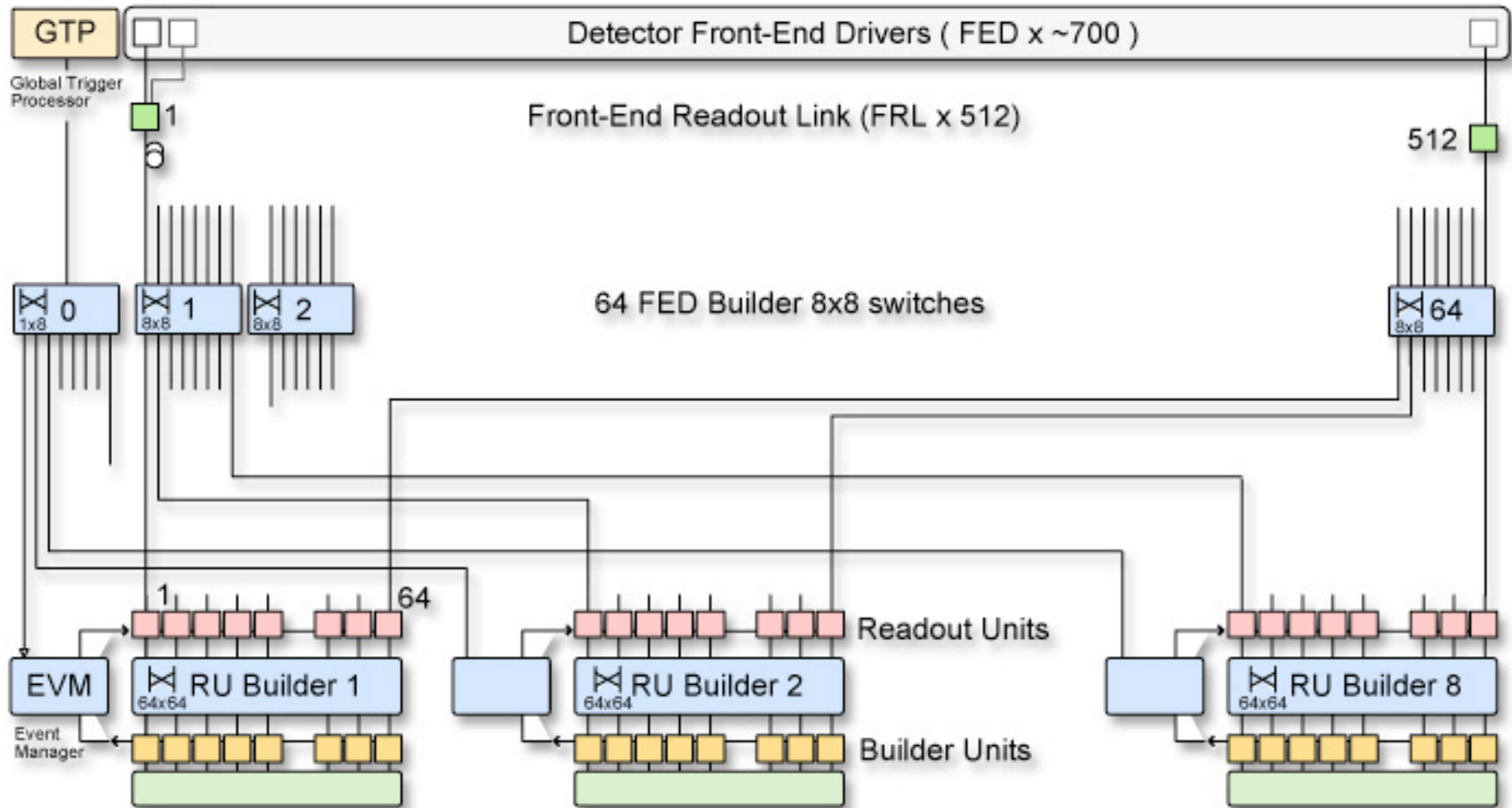
<b>Level-1 maximum trigger rate</b>	<b>100 kHz</b>	No. Readout systems	≈ 512
<b>Average event size</b>	<b>1 Mbyte</b>	No. Filter Subfarms	≈ 512 x n
<b>Builder network</b>	<b>1 Terabit/s</b>	No. (C&D) network ports	≈ 10000
<b>Event filter computing power</b>	<b>5 10<sup>6</sup> MIPS</b>	No. programmable units	≈ 10000
<b>Event flow control</b>	<b>≈ 10<sup>6</sup> Mssg/s</b>	System dead time	≈ %

- Acronyms**
- TPG Trigger Primitive Generator
  - RTP Regional Trigger Processor
  - LVT Level 1 Trigger Processor
  - STP Global Trigger Processor
  - TTC Timing, Trigger and Control
  - aTTS asynchronous Trigger Threshold System
  - aFES asynchronous Front-End System
  - FES Front-End System
  - FSC Front-End Control
  - FEC Front-End Controller
  - DOS Data to Surface
  - FRL Front-End Readout Link
  - FEL Front-End Link
  - BU Builder Unit
  - FS Filter Subfarm
  - FMM Event Manager
  - FMS Event Manager
  - DM Subfarm Manager
  - EVB Event Builder
  - FCN Front-End Control Network
  - SCN Subfarm Control Network
  - CSN Computing Service Network
  - DCN Detector Control Network
  - DSN Data Service Network
  - DCS Detector Control System
  - FCS Farm Control System

# “Modern” EVB architecture

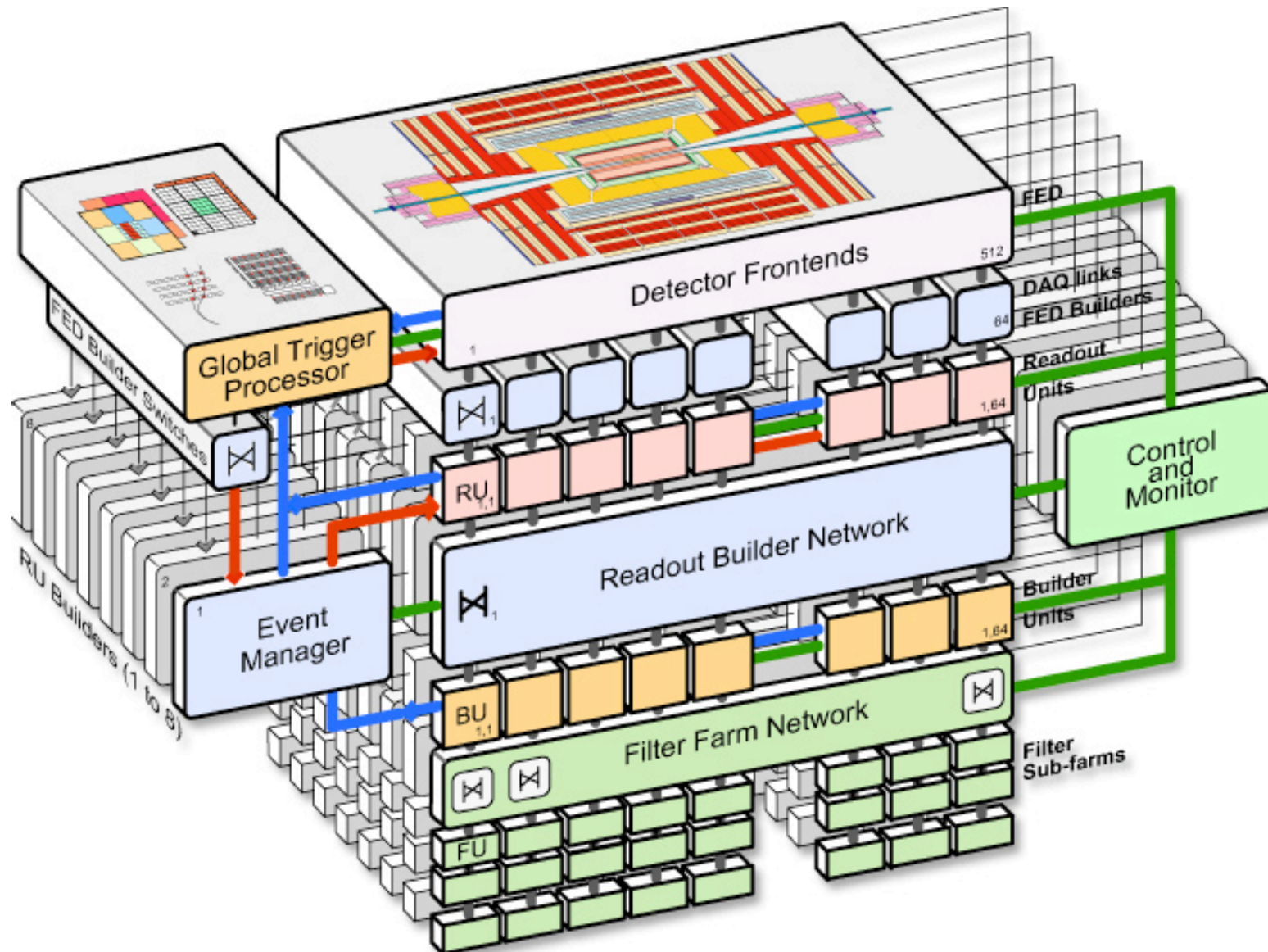


# EVB CMS: 2 stages





# CMS: 3D - EVB



# Advantages of 2 stage EVB

---

- Scalability
- Relaxed requirements for 2<sup>nd</sup> 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.
- 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.

---

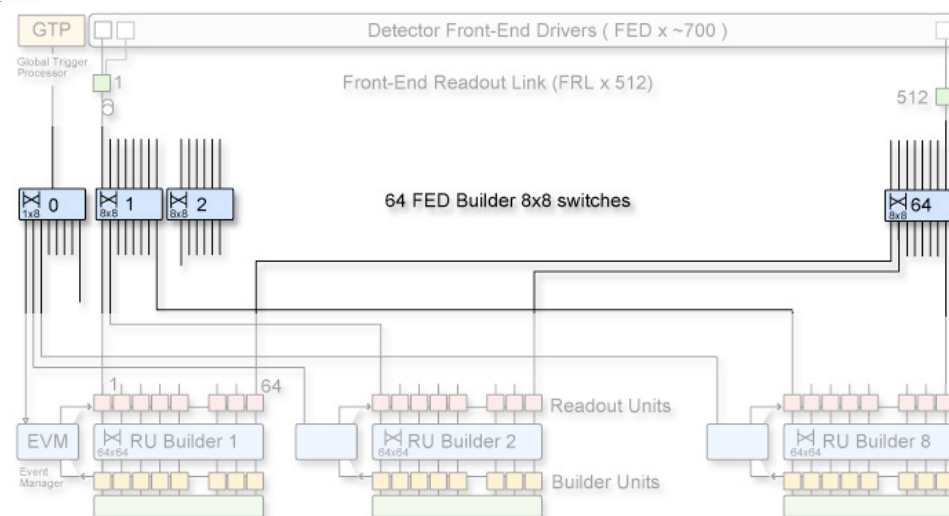
## The first stage of the Event-Builder

“FEDBuilder”

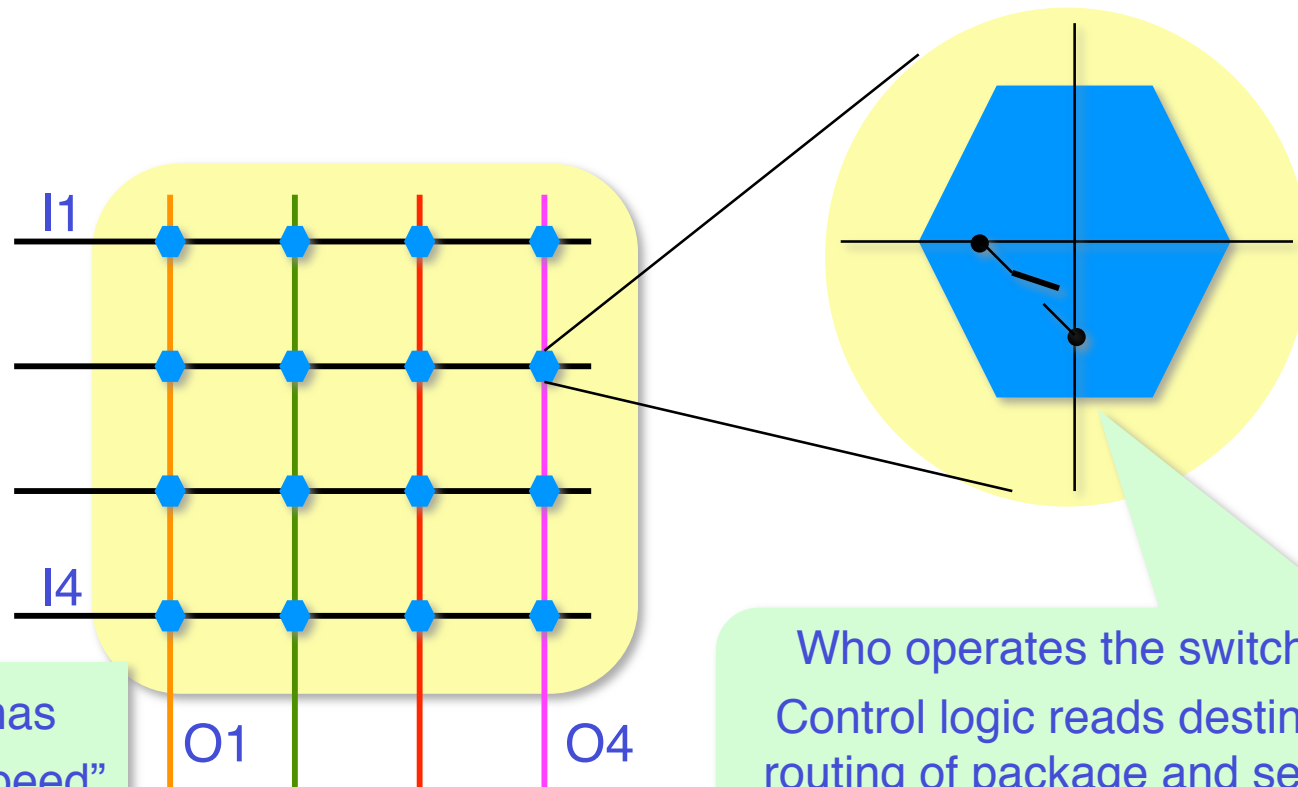
FED = Front End Driver

# 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!



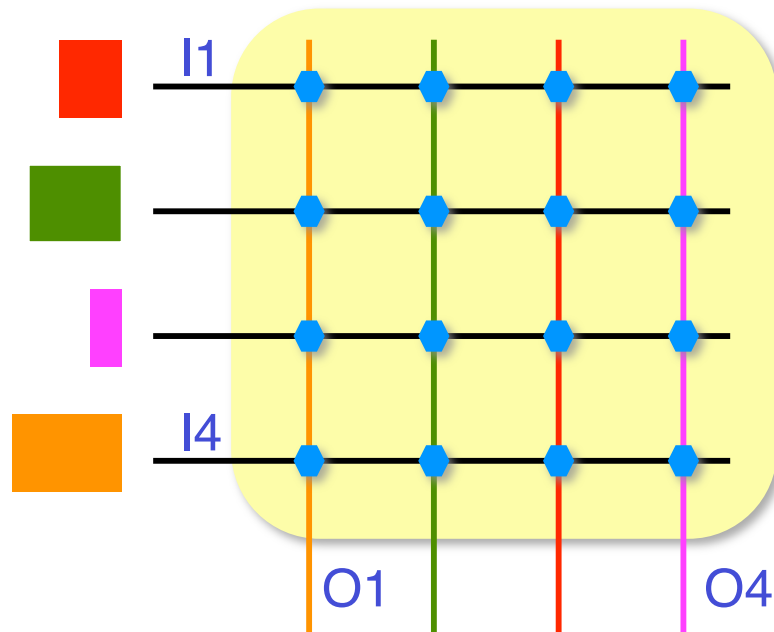
# Switch implementation: crossbar



Every input / output has  
A given max. “wire-speed”  
(e.g. 2Gbits/sec).  
Internal connections are  
much faster!

Who operates the switches ?  
Control logic reads destination  
routing of package and sets the  
switches appropriately.

# Switch implementation: crossbar

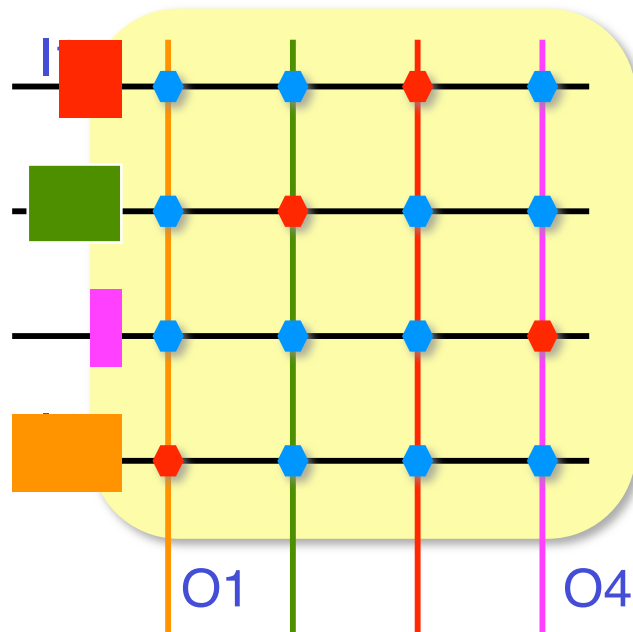


Paradise scenario:

All inputs want to send data to different destinations

# Switch implementation: crossbar

---

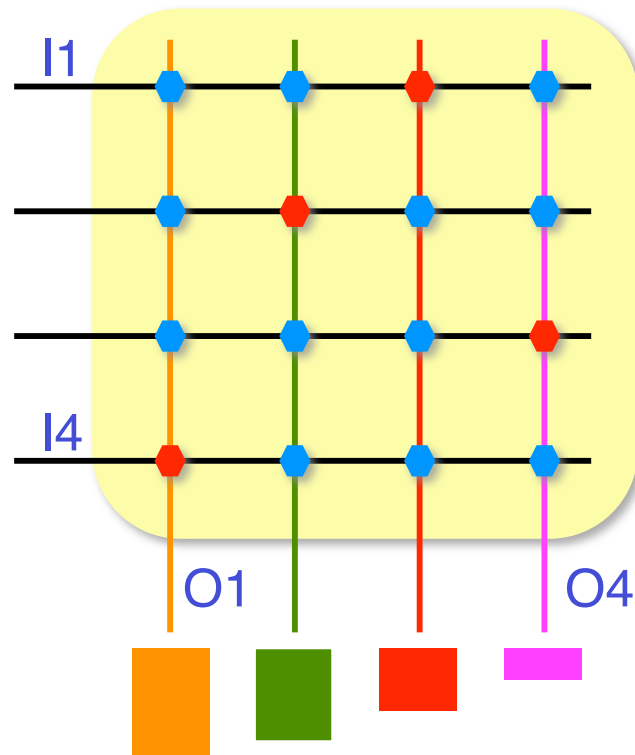


Paradise scenario:

No congestion, since every data package finds a free path through the switch.



# Switch implementation: crossbar

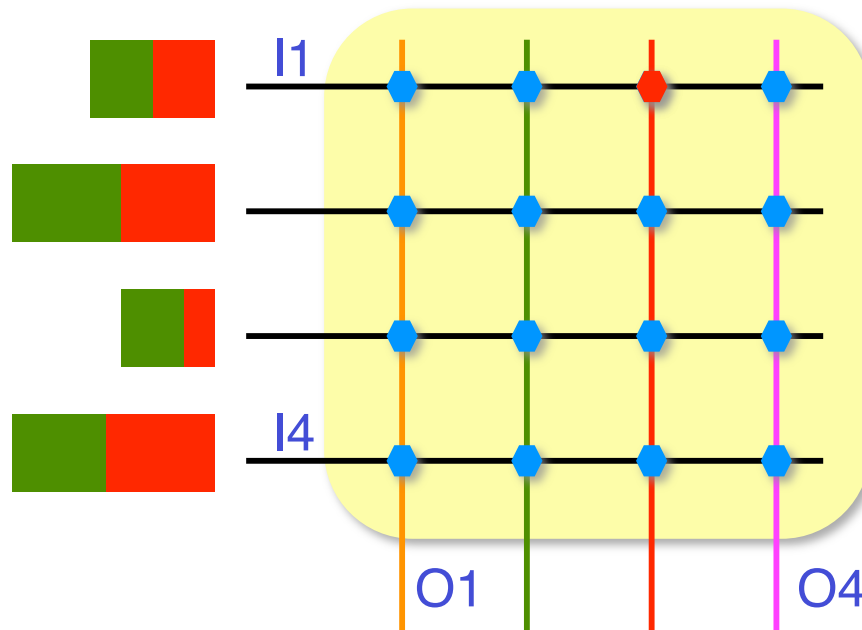


Paradise scenario:

Data traffic performs with  
“wire speed” of switch

# Switch implementation

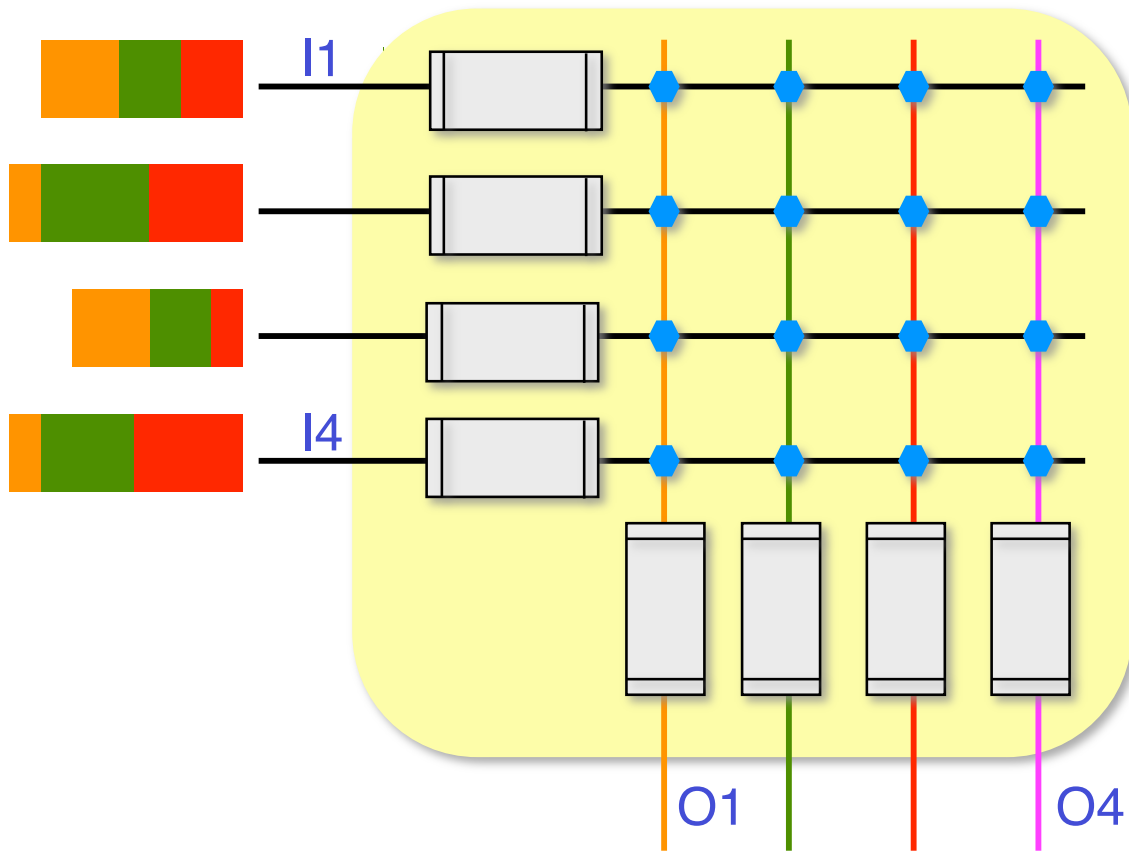
## Crossbar switch: **Congestion in EVB traffic**



Only one packet at a time can be routed to the destination.  
“Head of line” blocking

# Switch implementation

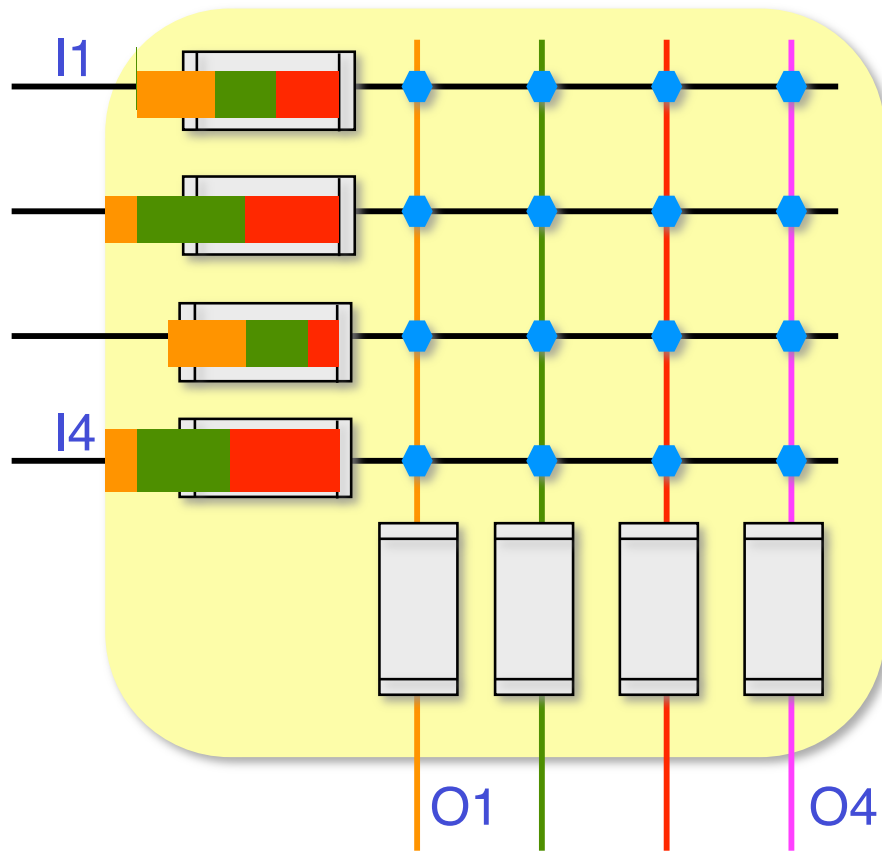
Crossbar switch: **Improvement : additional input FIFOs**



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

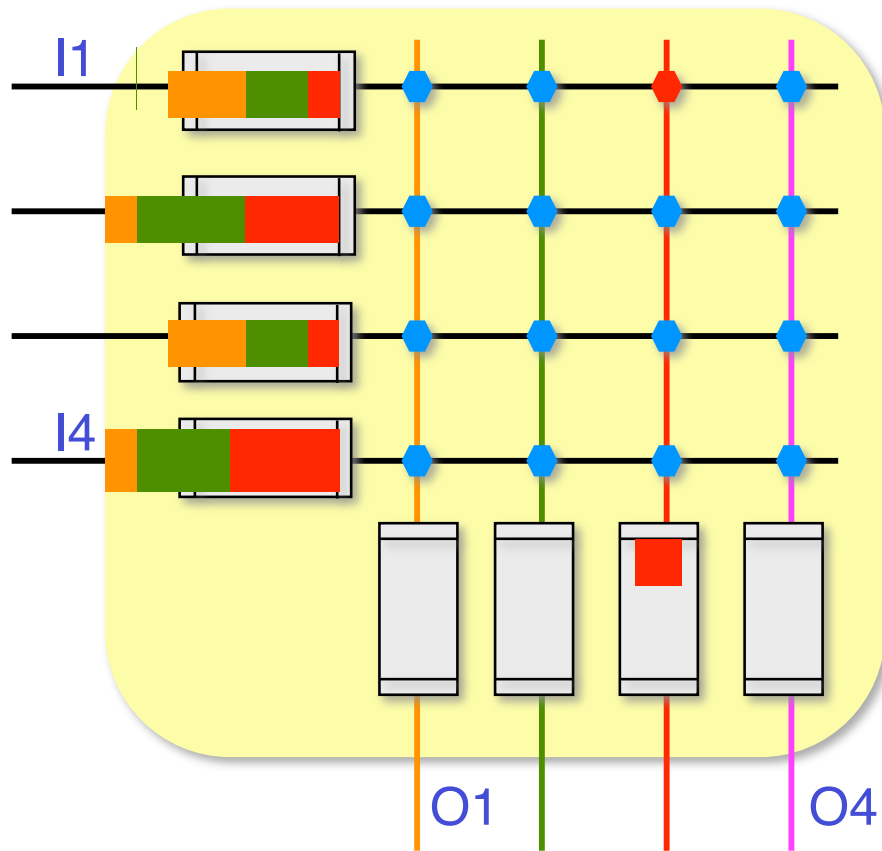
# Switch implementation

Crossbar switch: **Improvement : additional input FIFOs**



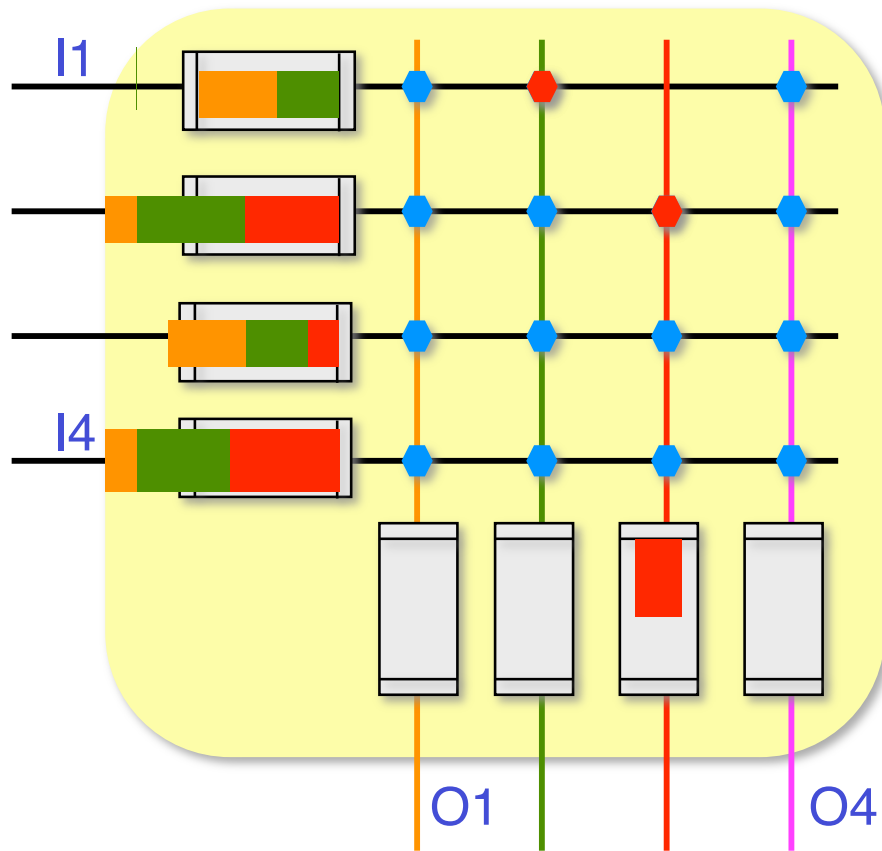
# Switch implementation

Crossbar switch: **Improvement : additional input FIFOs**



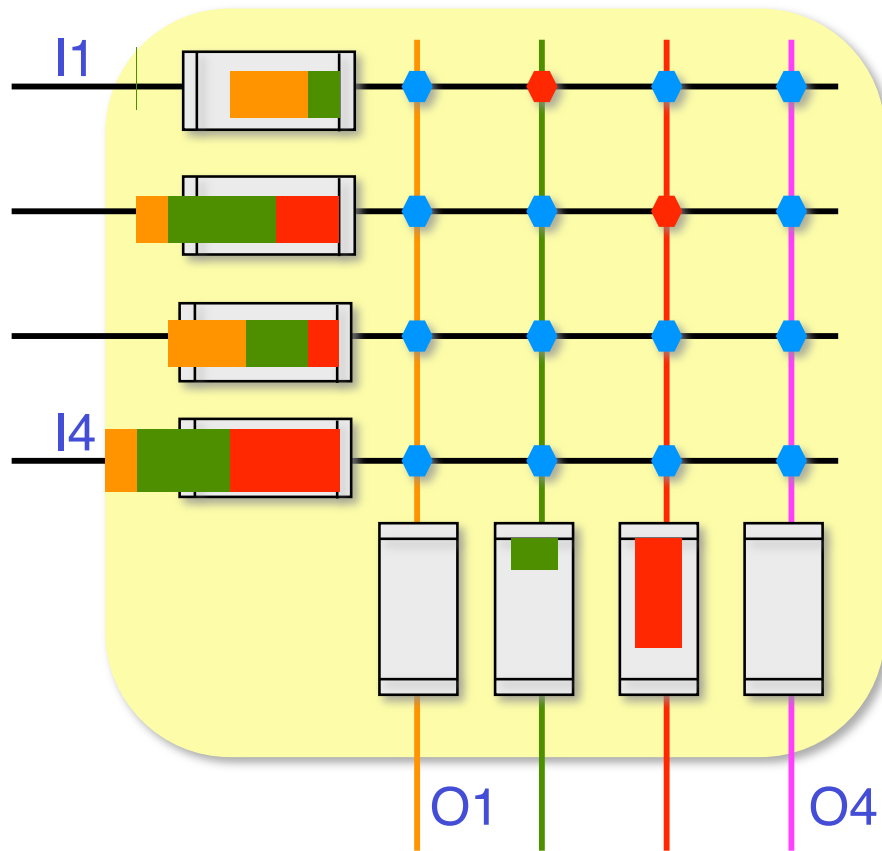
# Switch implementation

Crossbar switch: **Improvement : additional input FIFOs**



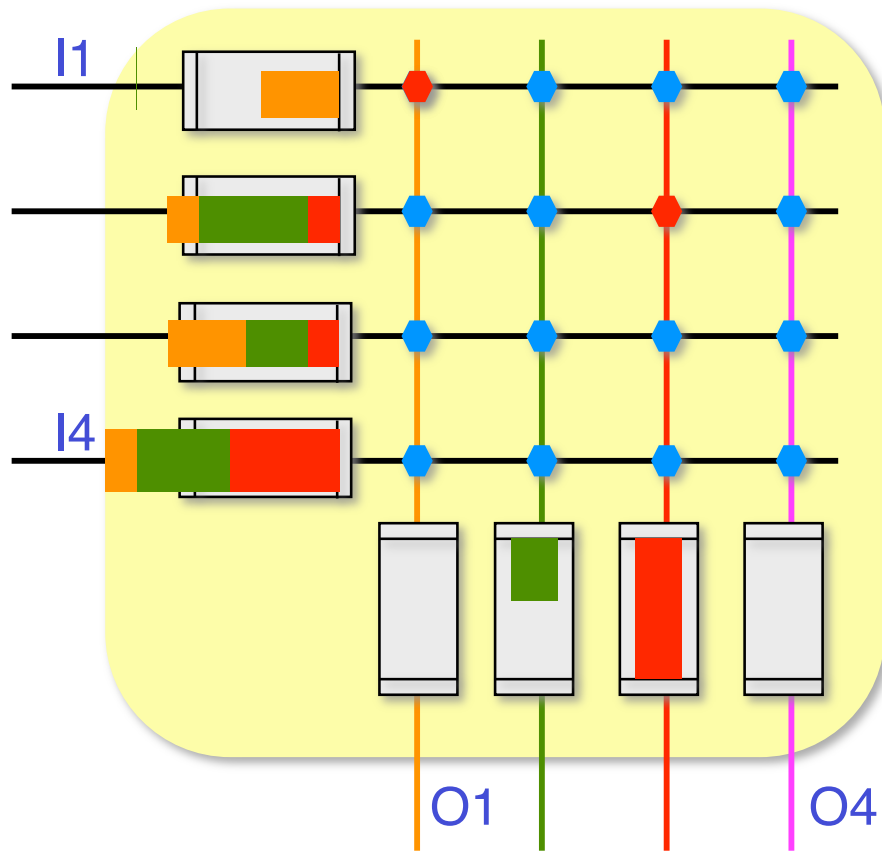
# Switch implementation

Crossbar switch: **Improvement : additional input FIFOs**



# Switch implementation

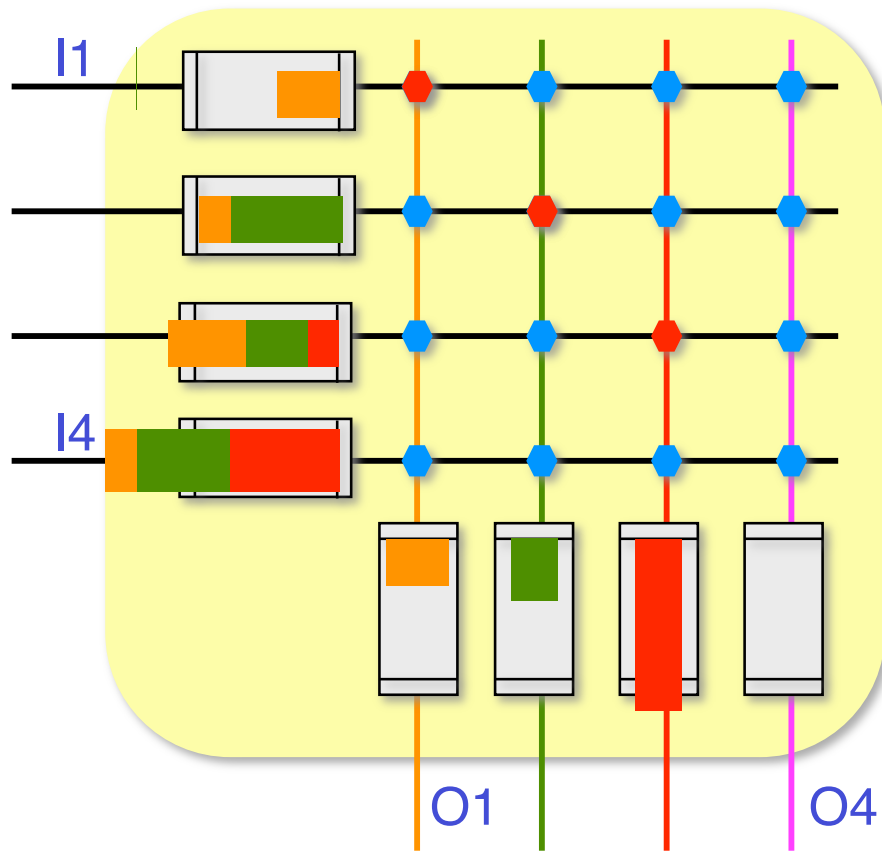
Crossbar switch: **Improvement : additional input FIFOs**





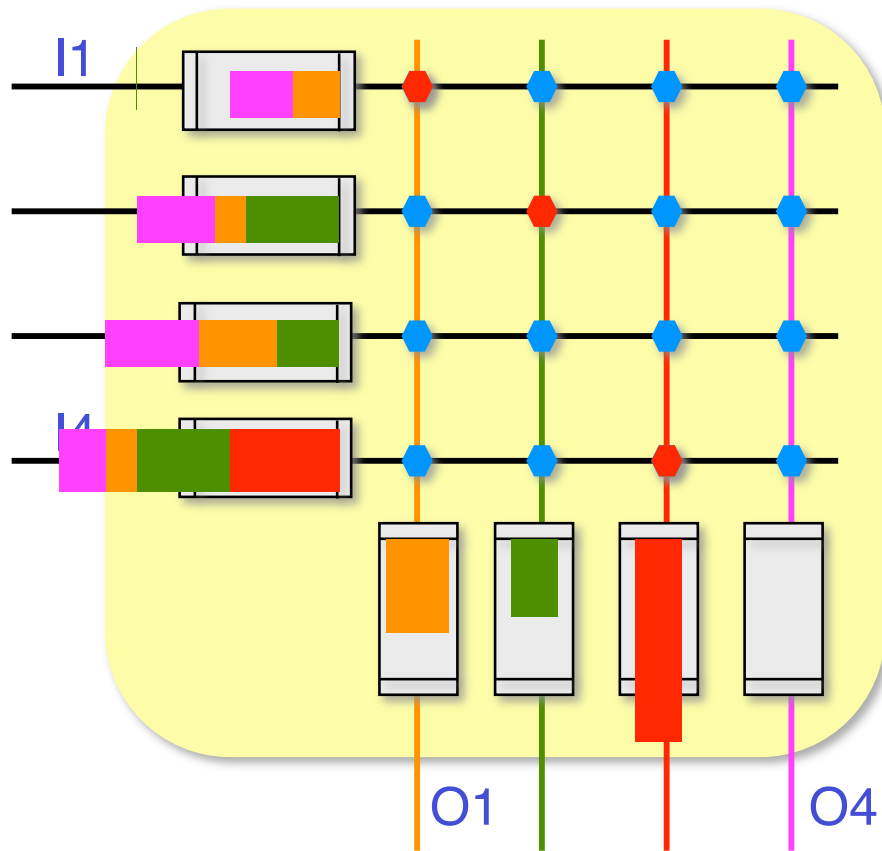
# Switch implementation

Crossbar switch: **Improvement : additional input FIFOs**



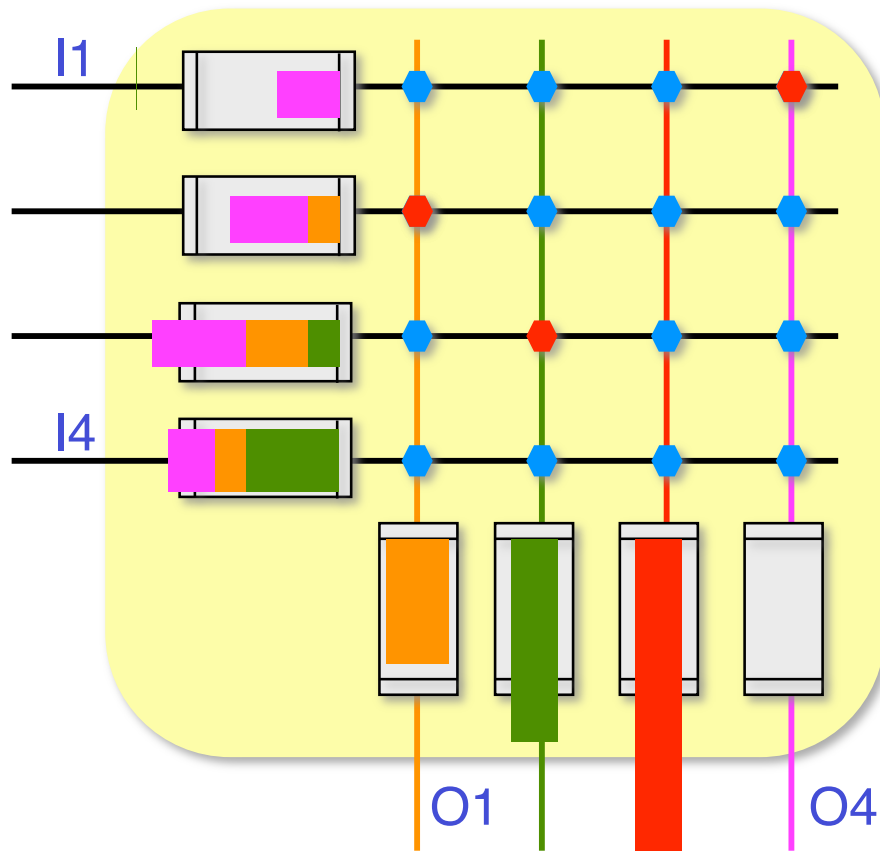
# Switch implementation

Crossbar switch: **Improvement : additional input FIFOs**



# Switch implementation

Crossbar switch: **Improvement : additional input FIFOs**



## Still problematic:

Input Fifos can absorb data fluctuations until they are full. How good it works depends on:

Fifos capacity

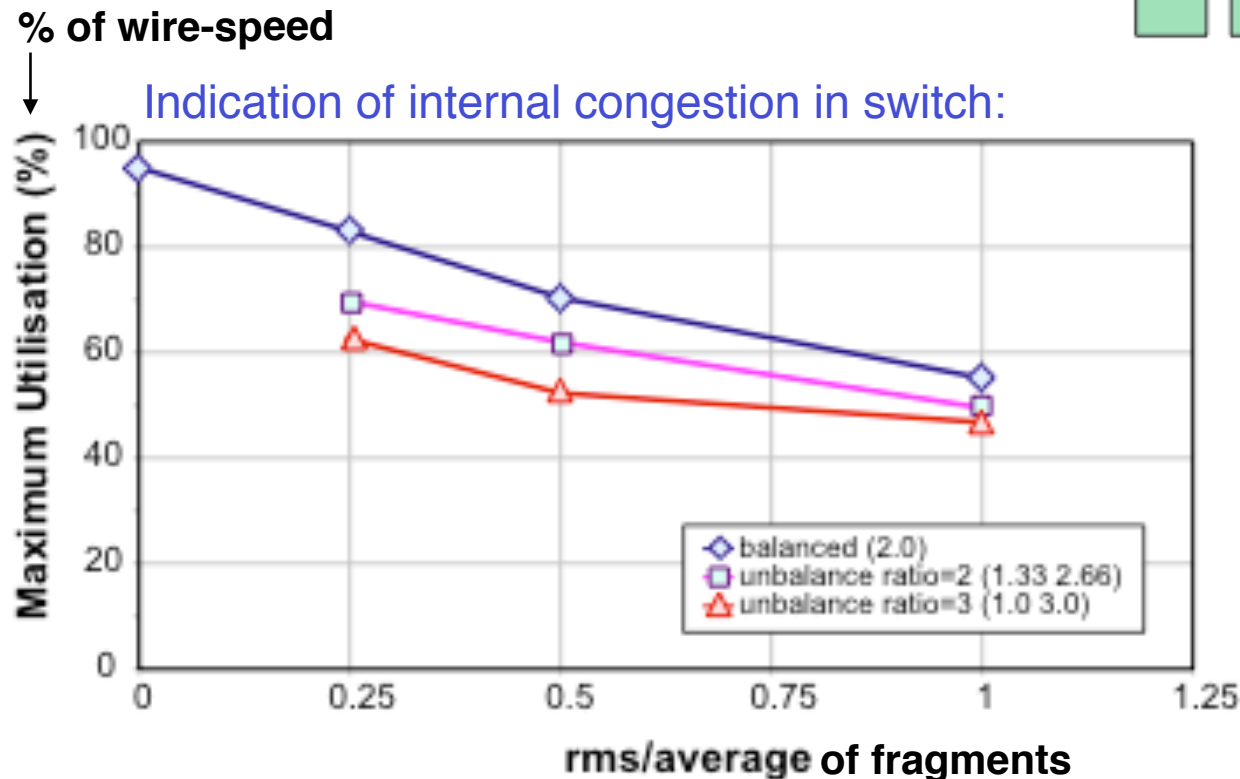
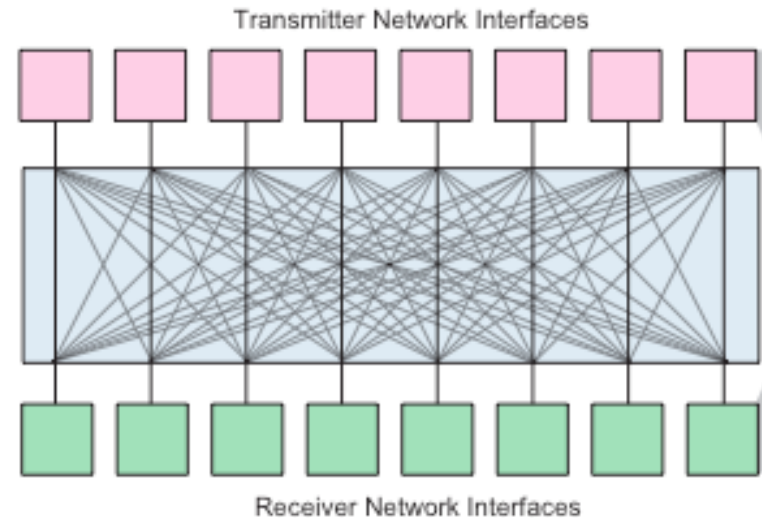
event size distribution

Internal speed of the switch

EVB traffic: blocking problem remains to some extent

# 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

---

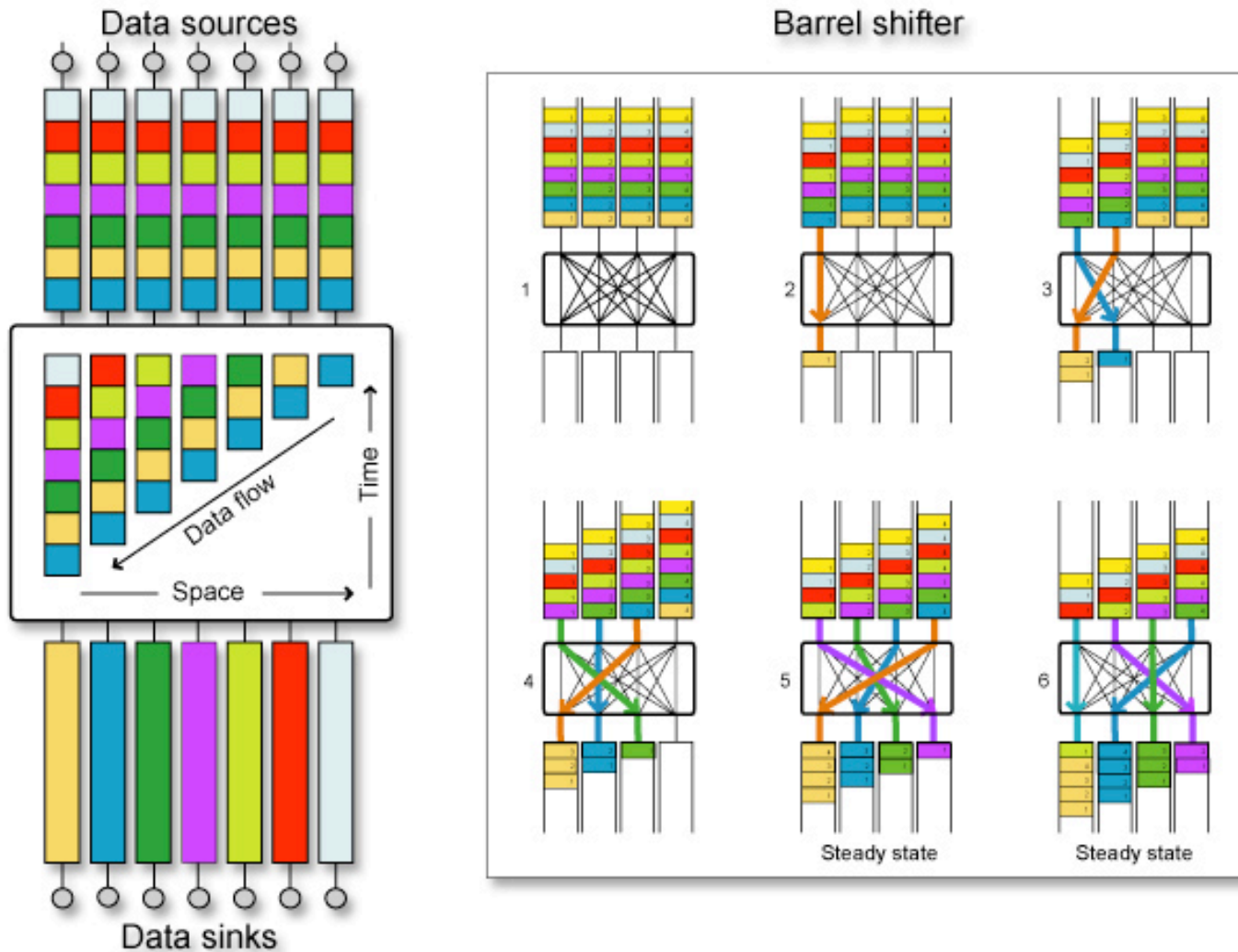
- EVB network traffic is particularly hard for switches
  - The traffic pattern is such that it leads to congestion in the switch.
  - The switch either “blocks” (= packets at input have to “wait”) or **throws away** data packets (Ethernet switches)

How to deal with this ???

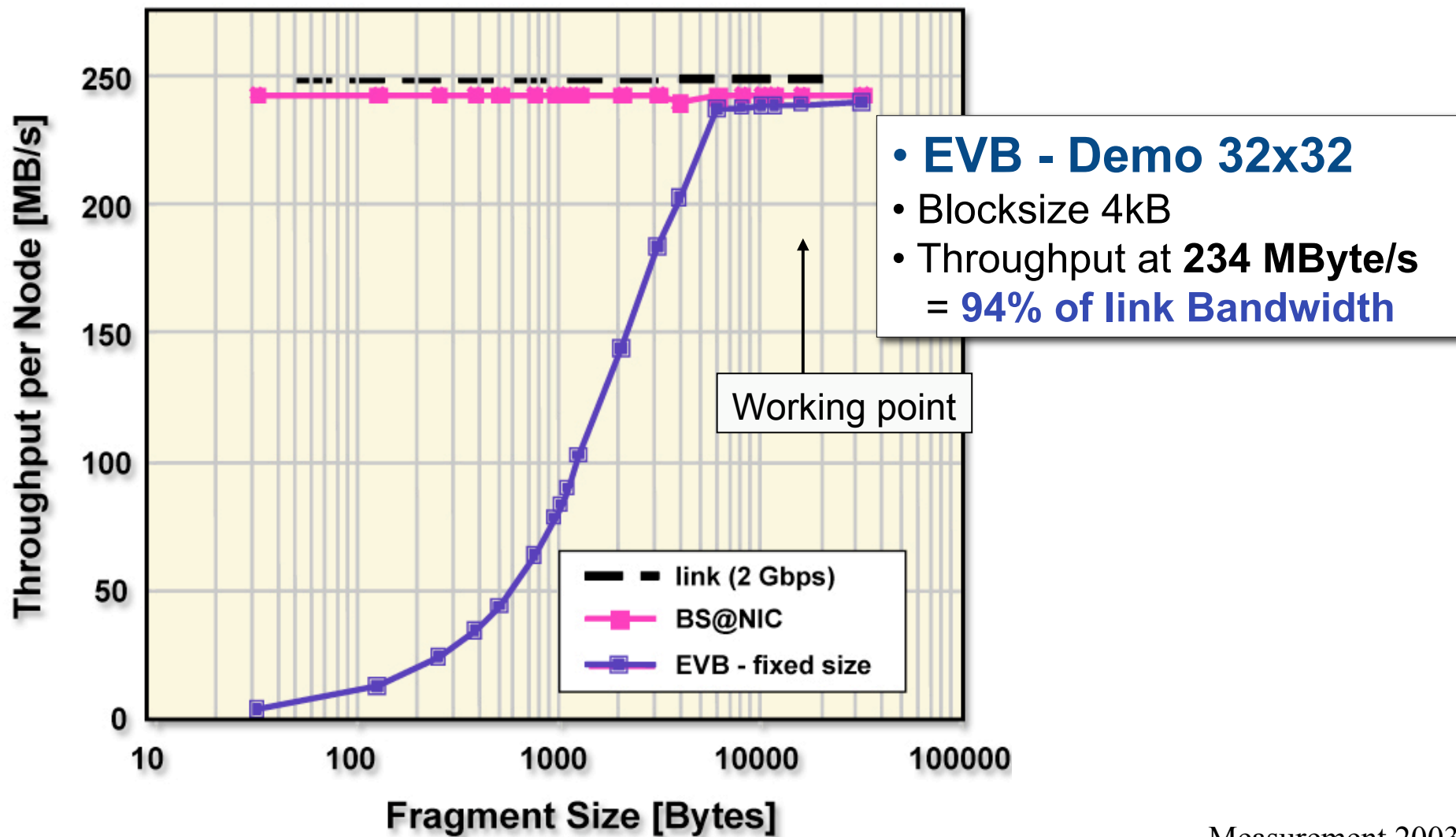
→ 2 possible solutions

# 1<sup>st</sup> : the clever solution: traffic shaping

## Example: Barrel Shifter



# Barrel Shifter: Measured Performance



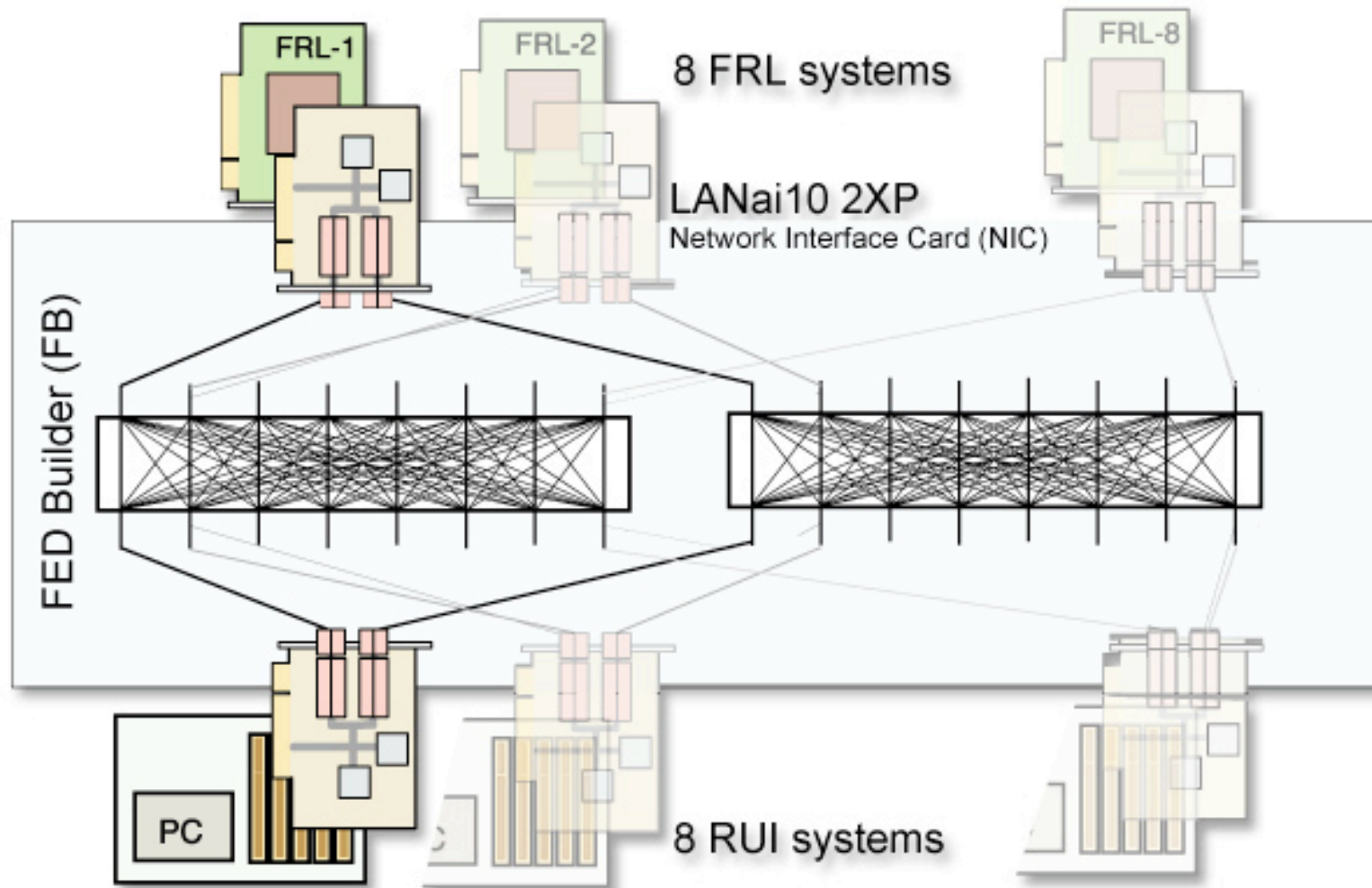
Measurement 2003  
(still valid)



## 2<sup>nd</sup> Solution: “take the hammer”



Over-dimension the system: buy twice as much hardware





# What did CMS do???

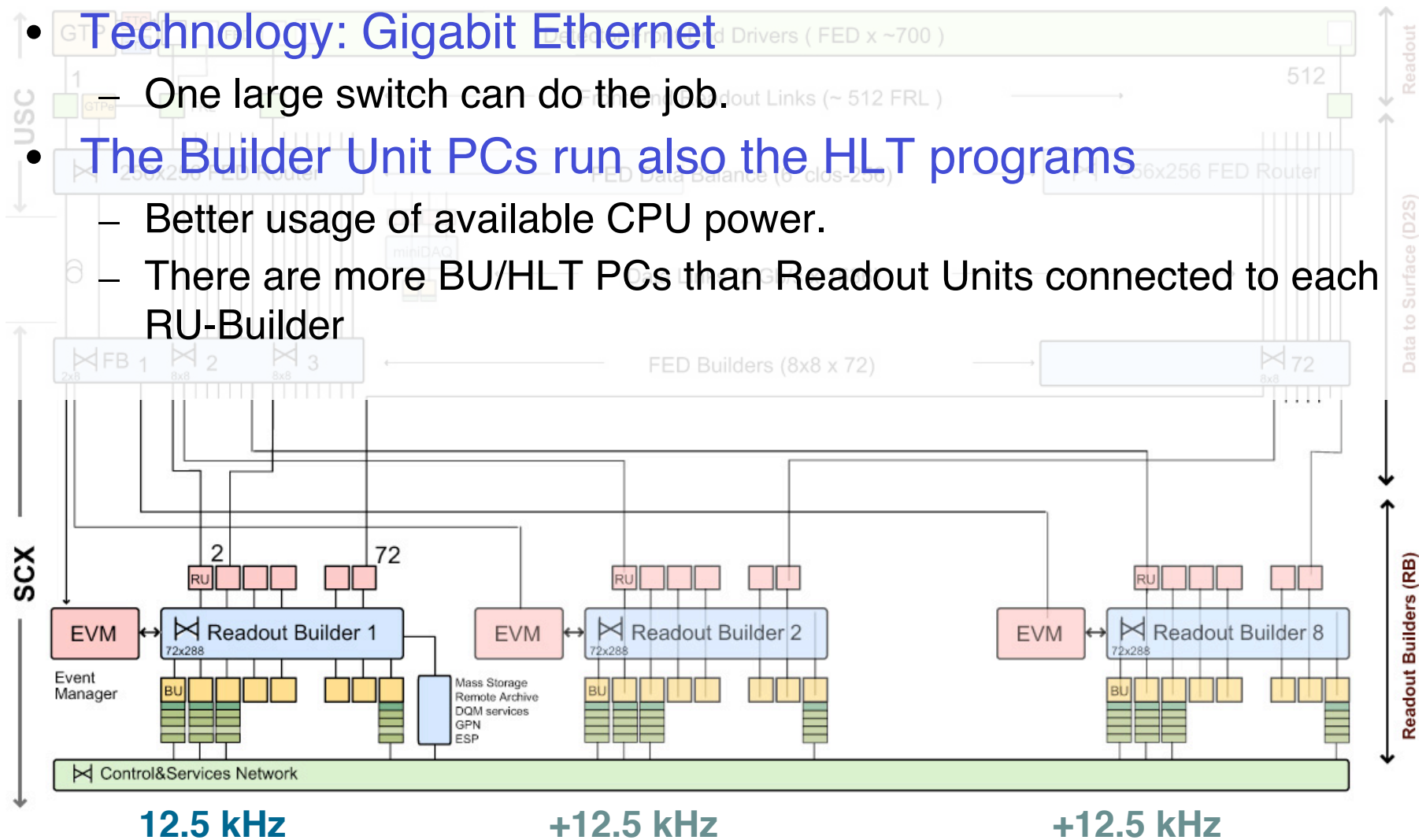
---

☺ 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”

- **Technology: Gigabit Ethernet**
  - One large switch can do the job.
- **The Builder Unit PCs run also the HLT programs**
  - Better usage of available CPU power.
  - There are more BU/HLT PCs than Readout Units connected to each RU-Builder



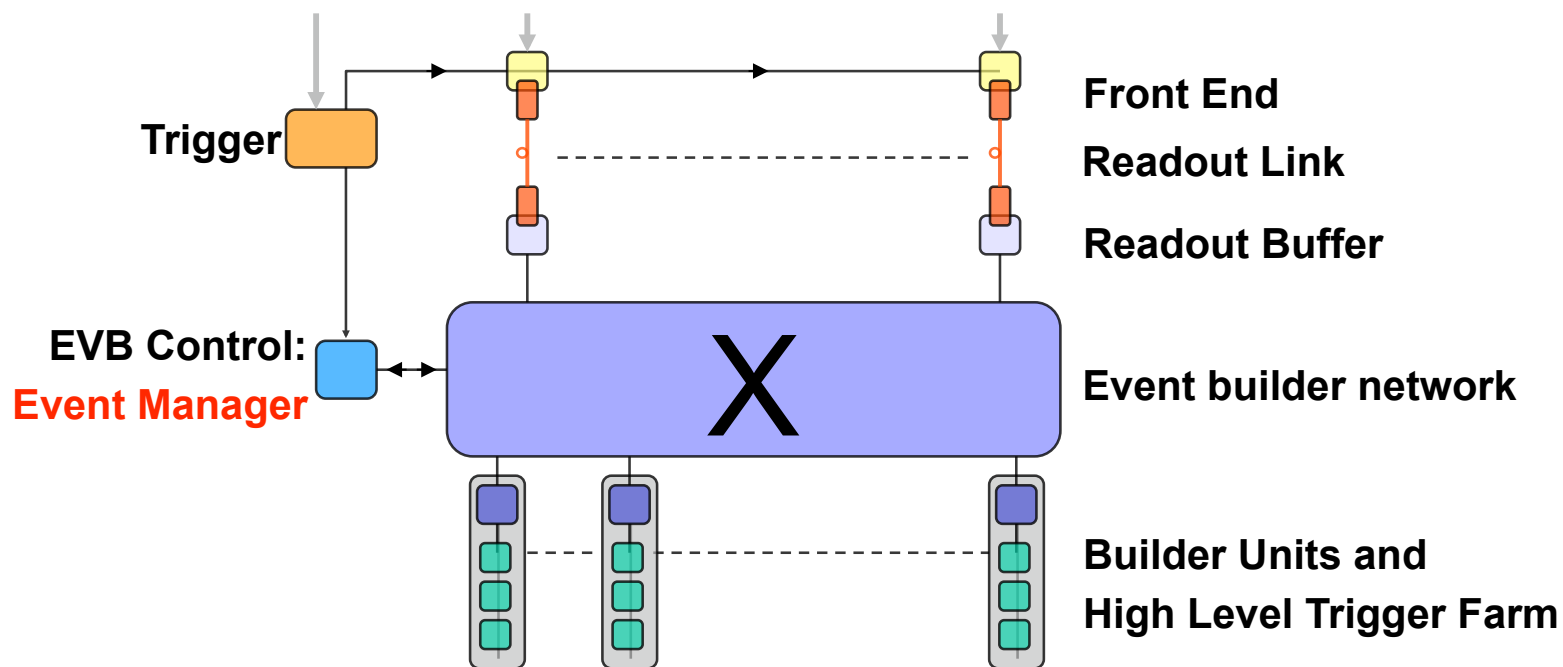
# Link Aggregation

---

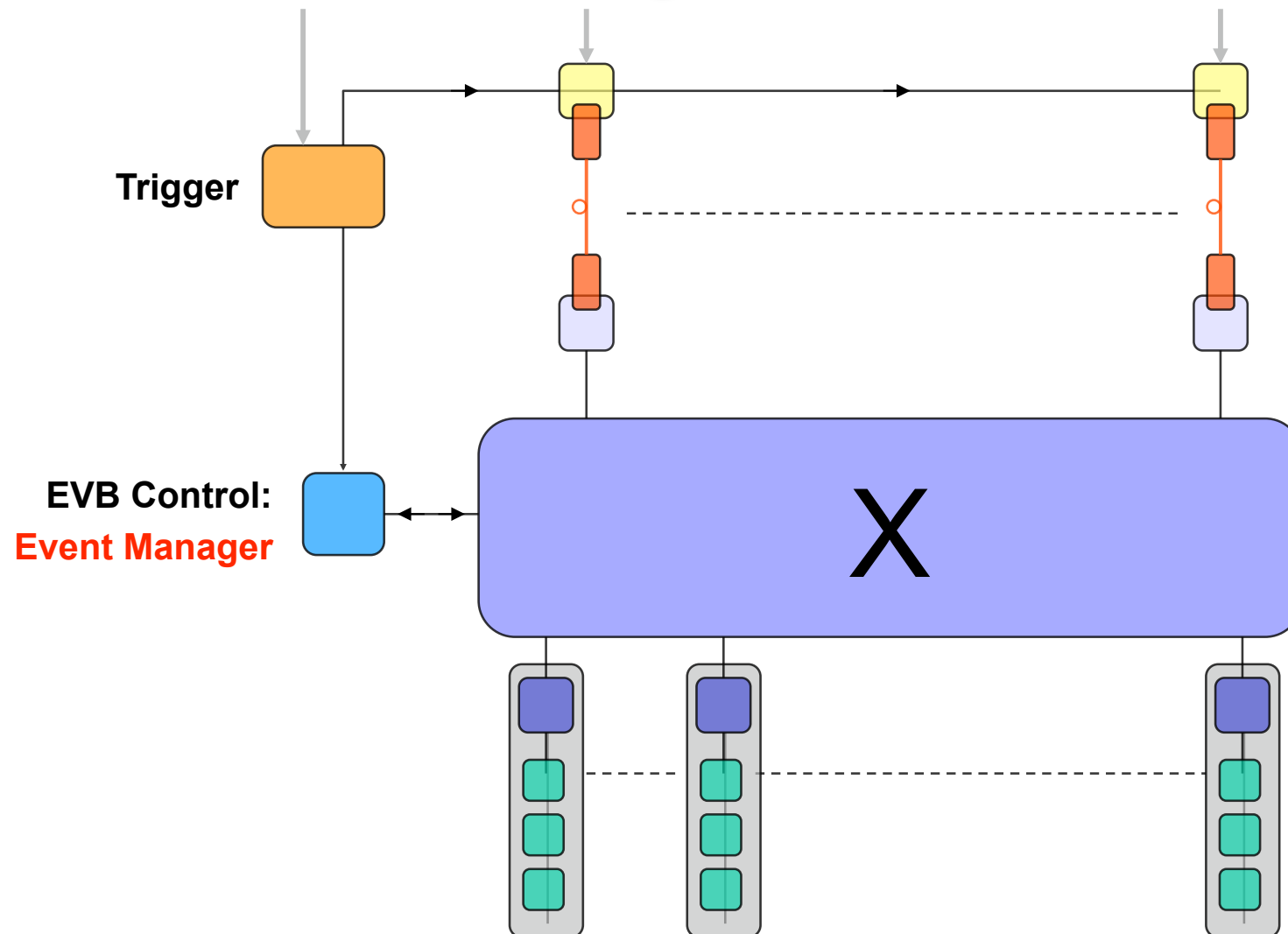
- We want at least 200 MB/s traffic from RU to BU
  - One Gigabit Ethernet line can transfer “only” 120MB/s
  - Need multiple Network Interfaces in RU and BU
    - Divide network between RUs and BUs into virtual LANs
      - Connect every RU to BUs in different VLANs
      - Connect every BU to RUs in 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

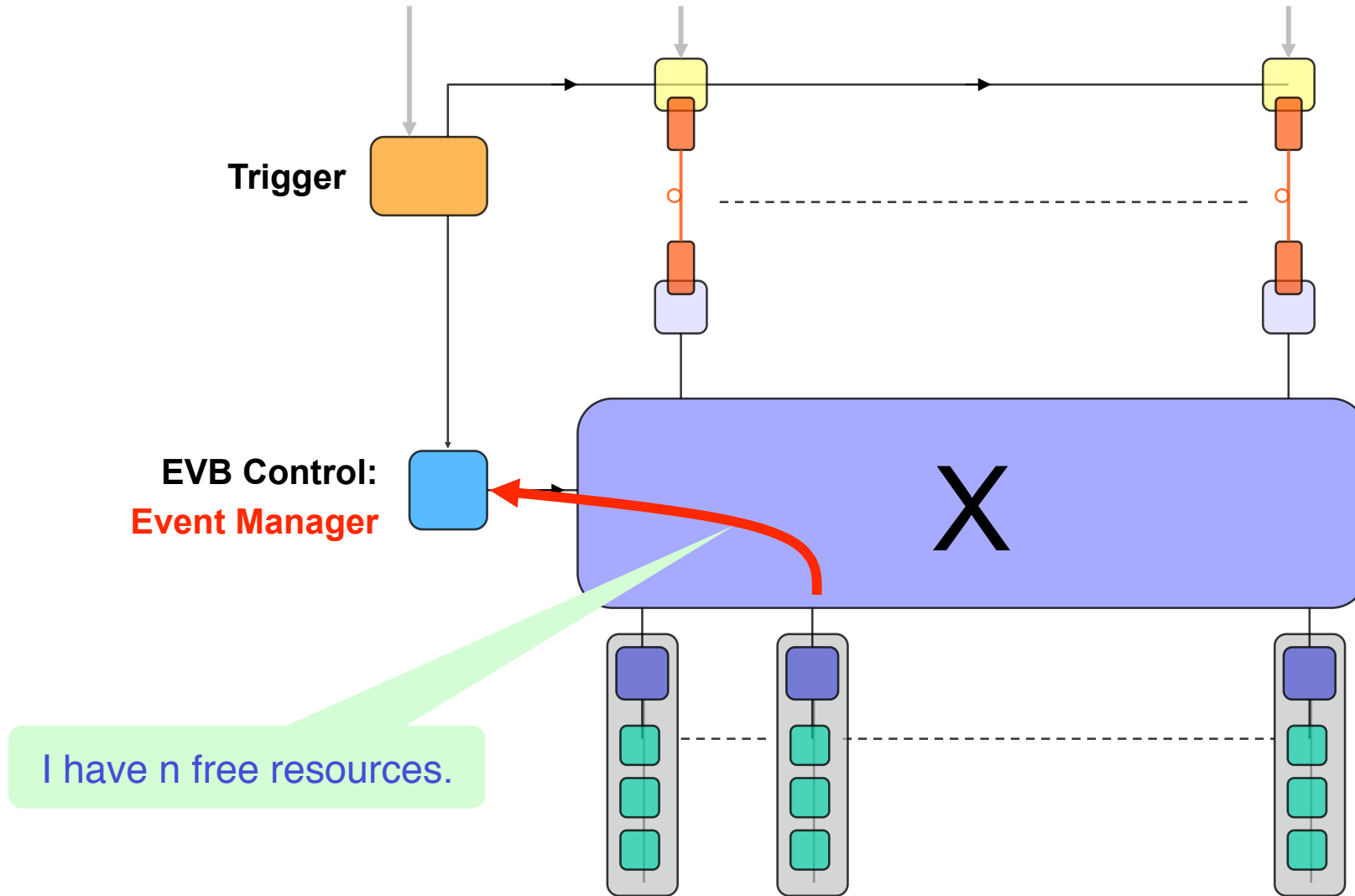
- Aim: Event Builder should perform load balancing
  - If for some reason some destinations are slower than others this should not slow down the entire DAQ system.
  - Another form of **traffic shaping**



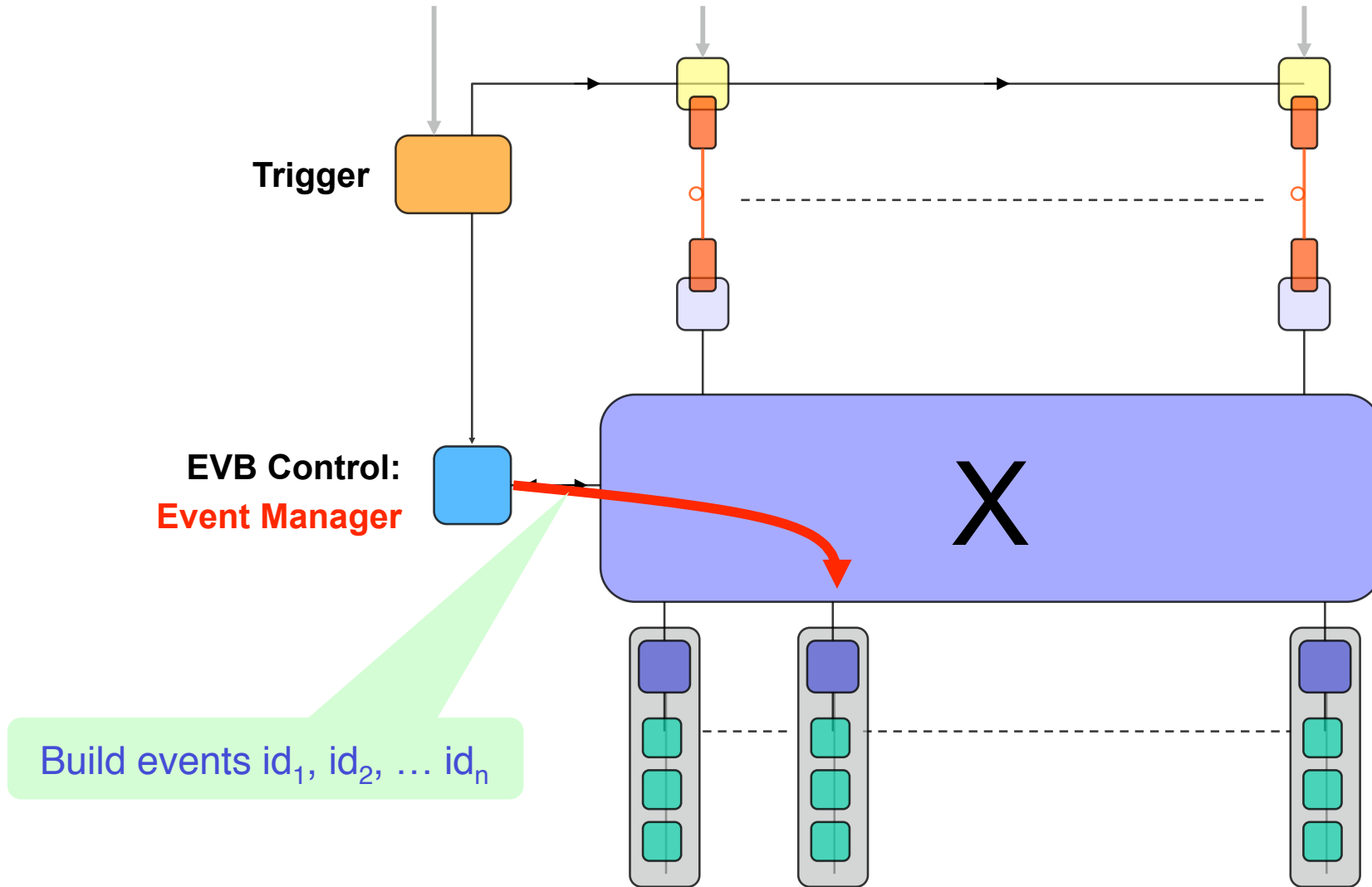
# Event Building: EVB-Protocol



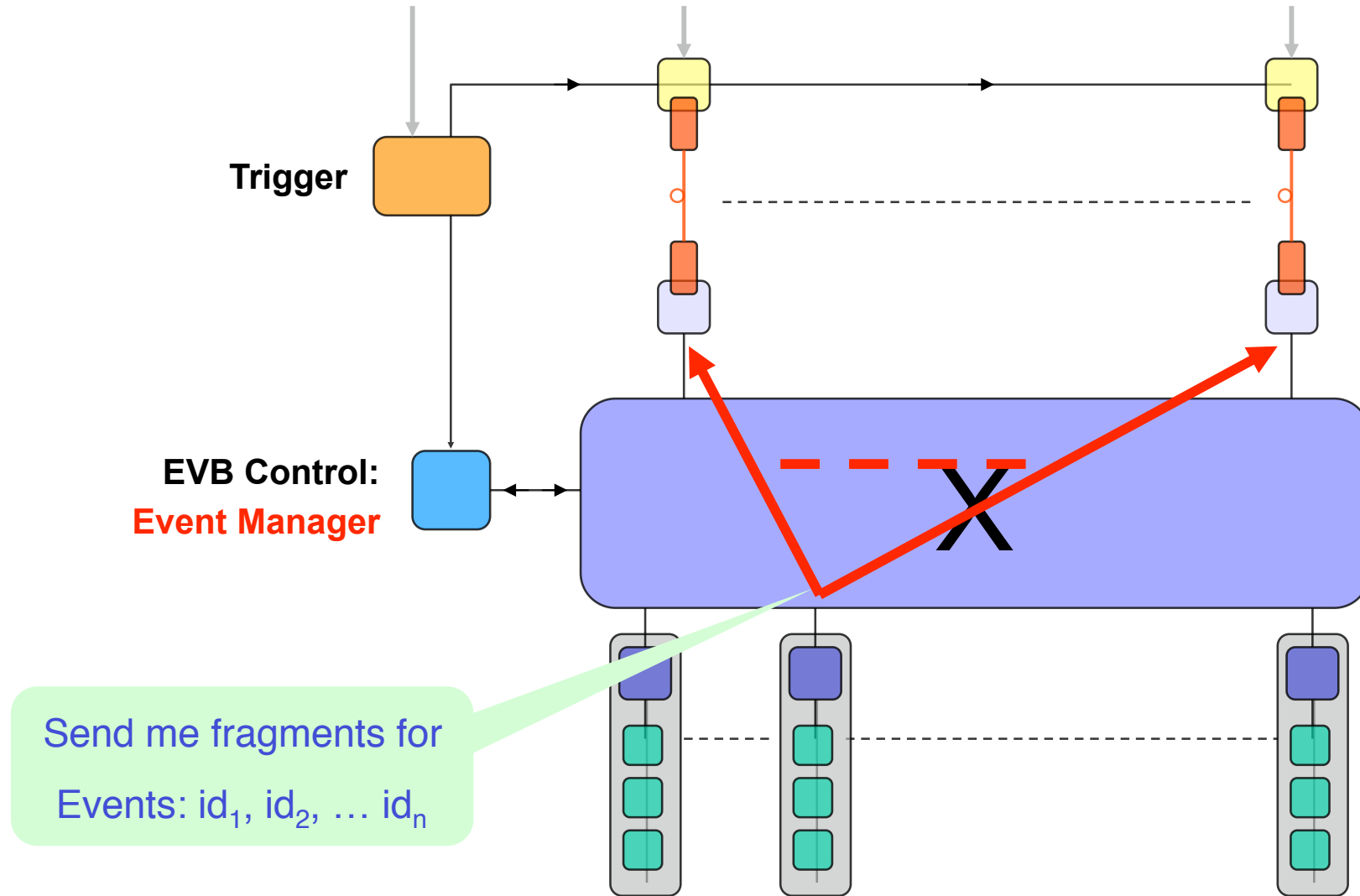
# Event Building: EVB-Protocol



# Event Building: EVB-Protocol

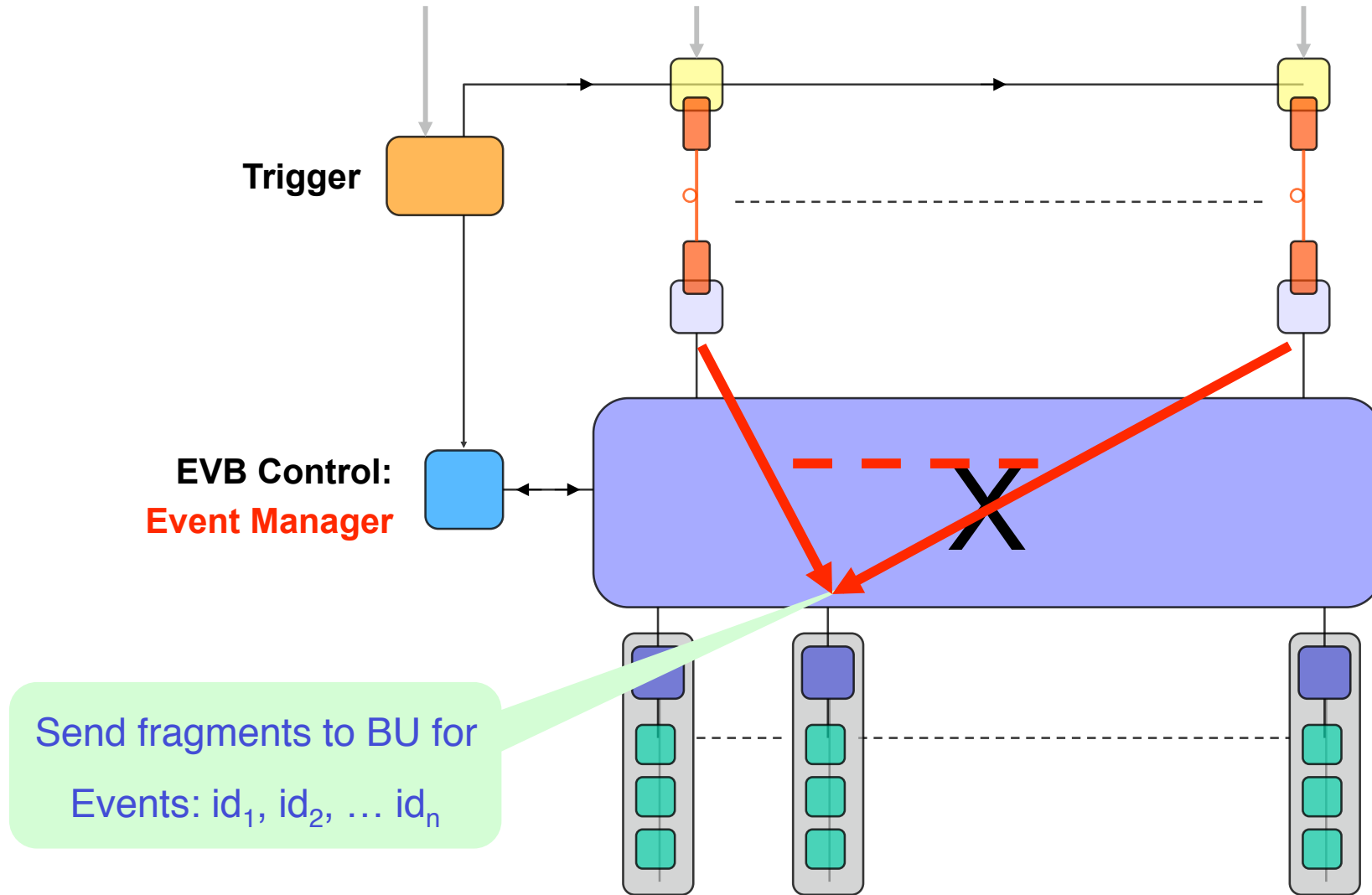


# Event Building: EVB-Protocol





# 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.



# Event Builder Components

---



The RU-Builder Switch  
(for 2 slices)

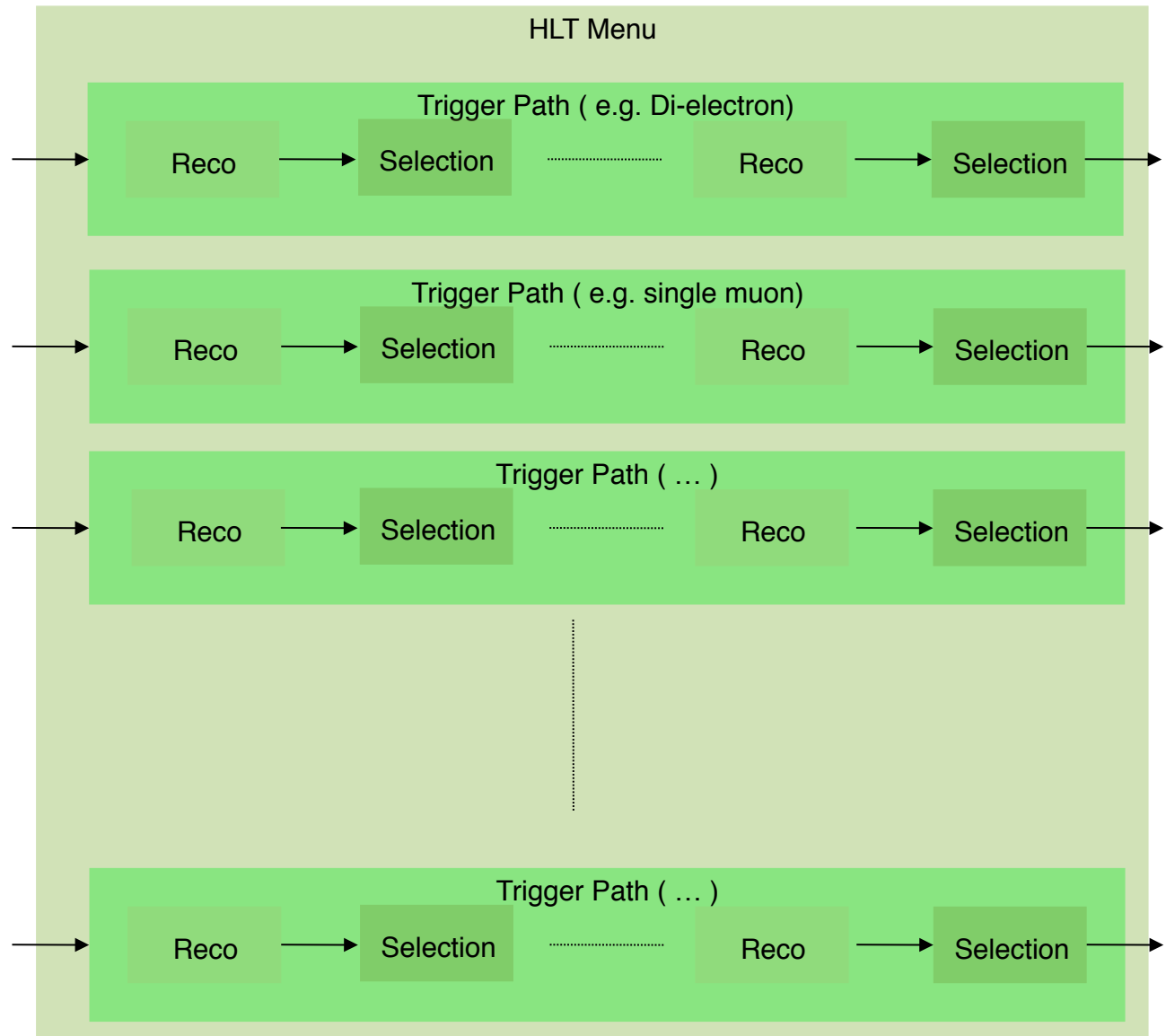
---

# HLT trigger implementation

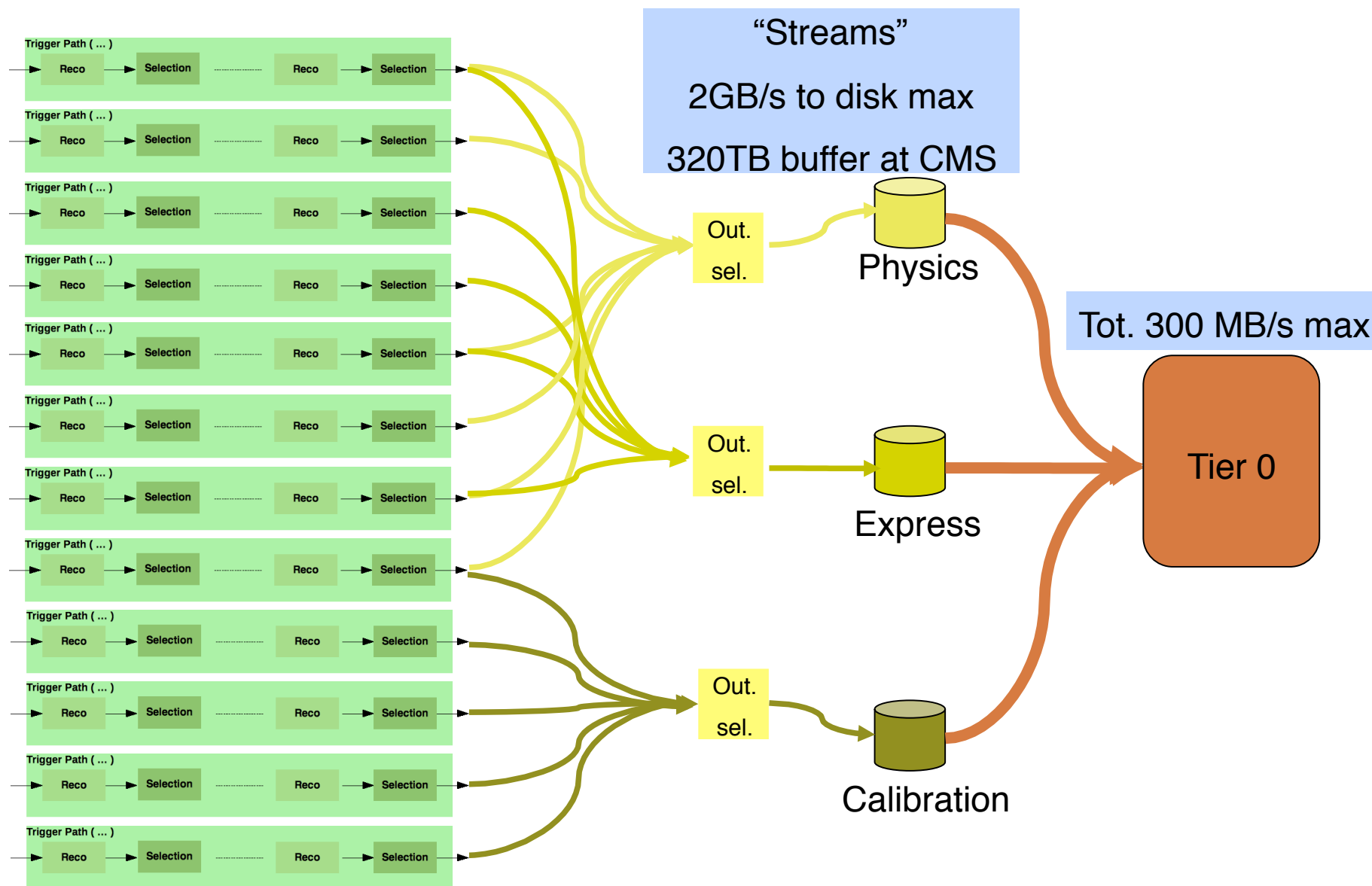
# CMS High Level Trigger or Filter

## Filter processes

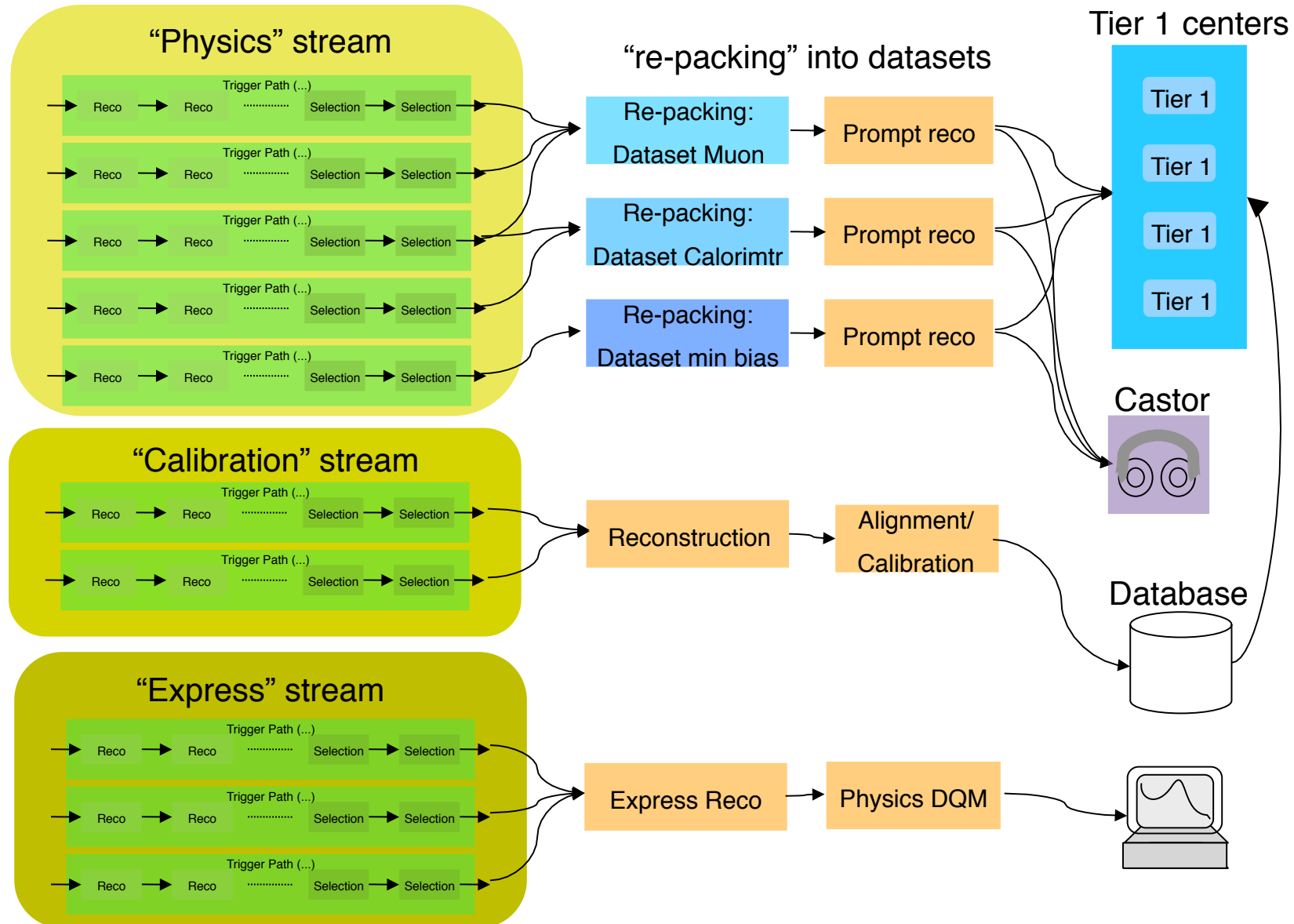
- Run the same code as offline analysis jobs
- Trigger decision based on “trigger paths”



# 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.

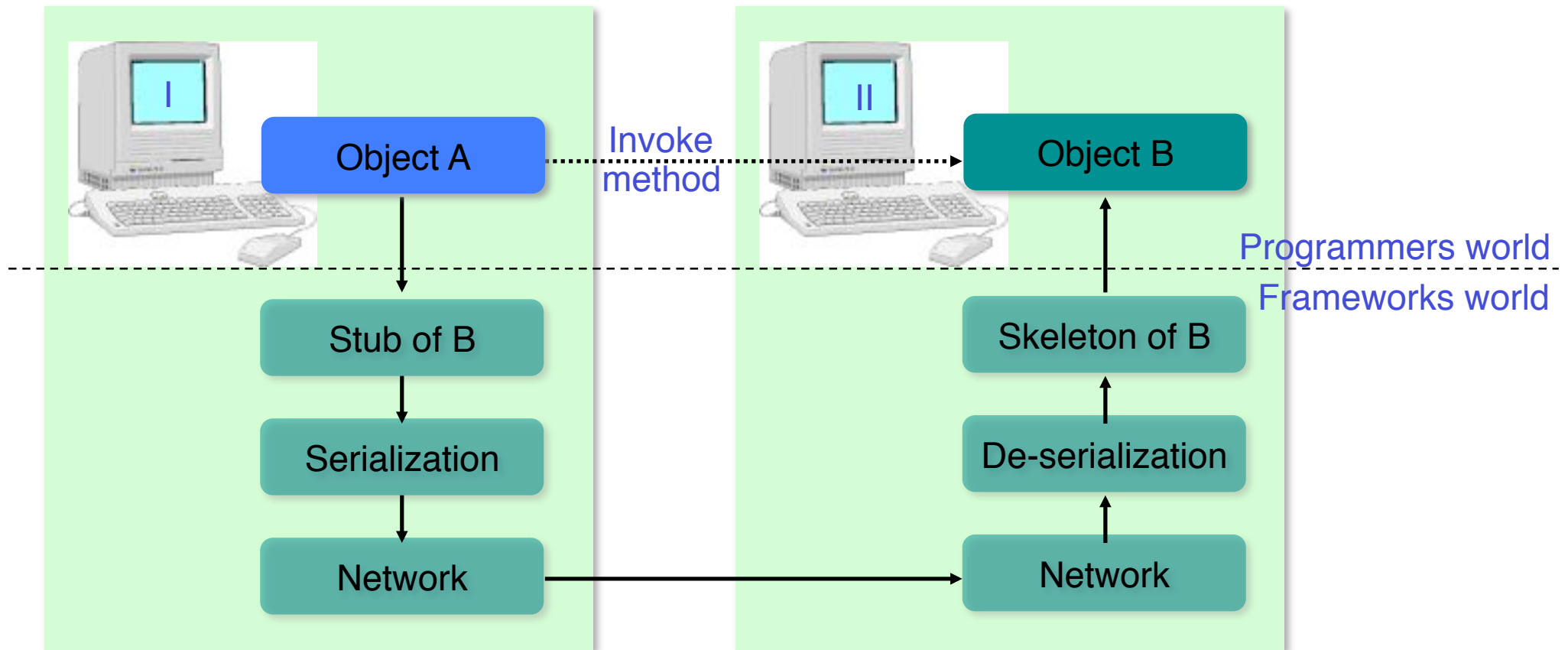
# Distributed computing

---

- **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

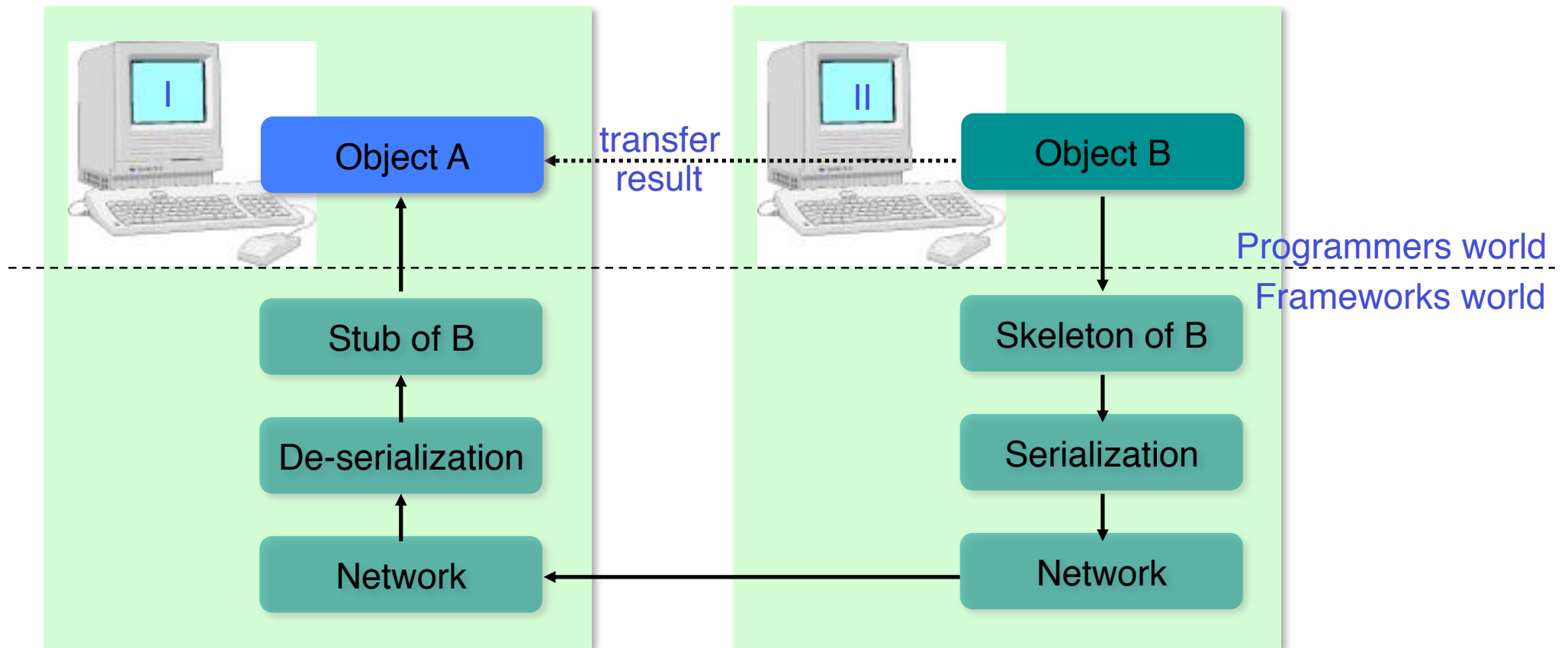
A method on a remote object is called:





# Distributed computing

The result is coming back:



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

---



# Possible scenarios at LHC

---

Everything here is speculation, but anyway...

- The way LHC reaches the “design luminosity”:
  - Due to technical issues with the LHC it might be possible that design luminosity ( $10^{34}$ ) will be reached with bunch spacing of 50ns instead of 25ns originally foreseen.
    - Difficult issue for experiments because:

$$T_{\text{InterBunch}} \times 2 \rightarrow N_{\text{bunches}} \times 0.5 \rightarrow L_{\text{per bunch}} \times 2 \rightarrow$$

**Twice the number of underlying interactions (min.bias) per event !!!**

**46 instead of 23**



---

- A consequence

- Larger occupancies in the detector
- Trigger algorithms become perform worse
  - Isolation cuts need to be modified
  - Purity becomes worse (rate goes up) or efficiency drops
  - Thresholds need to be raised
- Events become larger
  - Expect a factor of 2 since **average** event size dominated by min. bias
  - Need to shuffle twice as much data with DAQ system
- HLT needs to work harder
  - E.g. to do tracking at higher occupancies, algorithms need more time

To keep the performance of the Detector a lot of components need to be upgraded in Trigger and DAQ

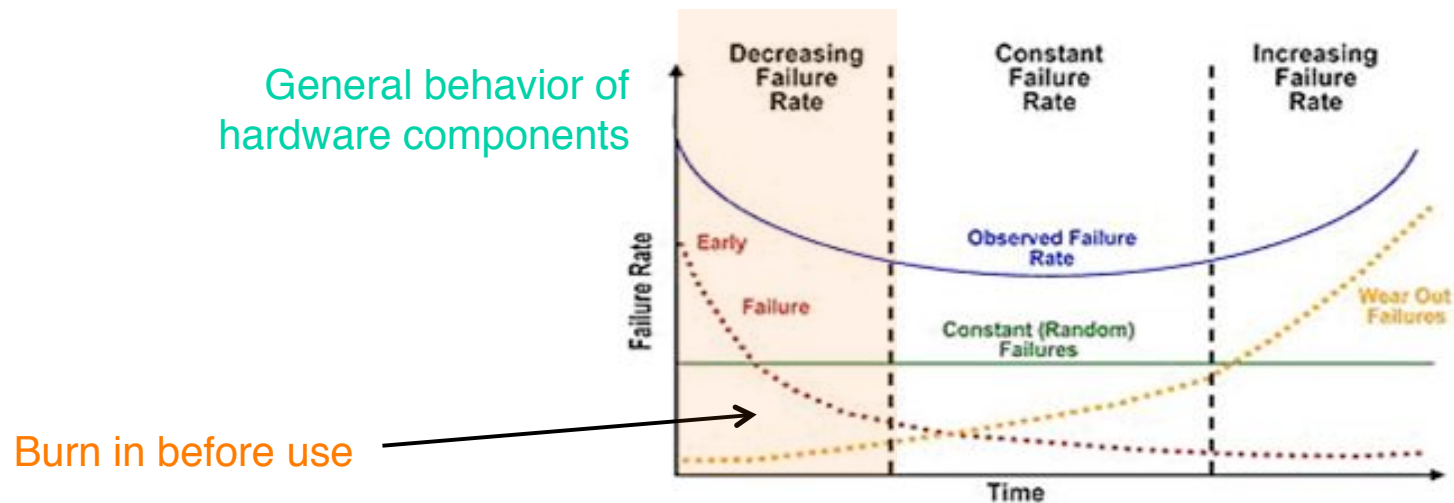
# In the far future... (year 2018...20xy)

---

- 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.
  - Probably **Tracker information** need to be added to the trigger
  - Calorimeter triggers need to work with **finer granularities** in order to be able to do effective isolation cuts.
  - Event size will grow due to detector upgrades (more channels) and more underlying min. bias events
    - DAQ needs substantially **higher data throughput**

# Real life...

- 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

---

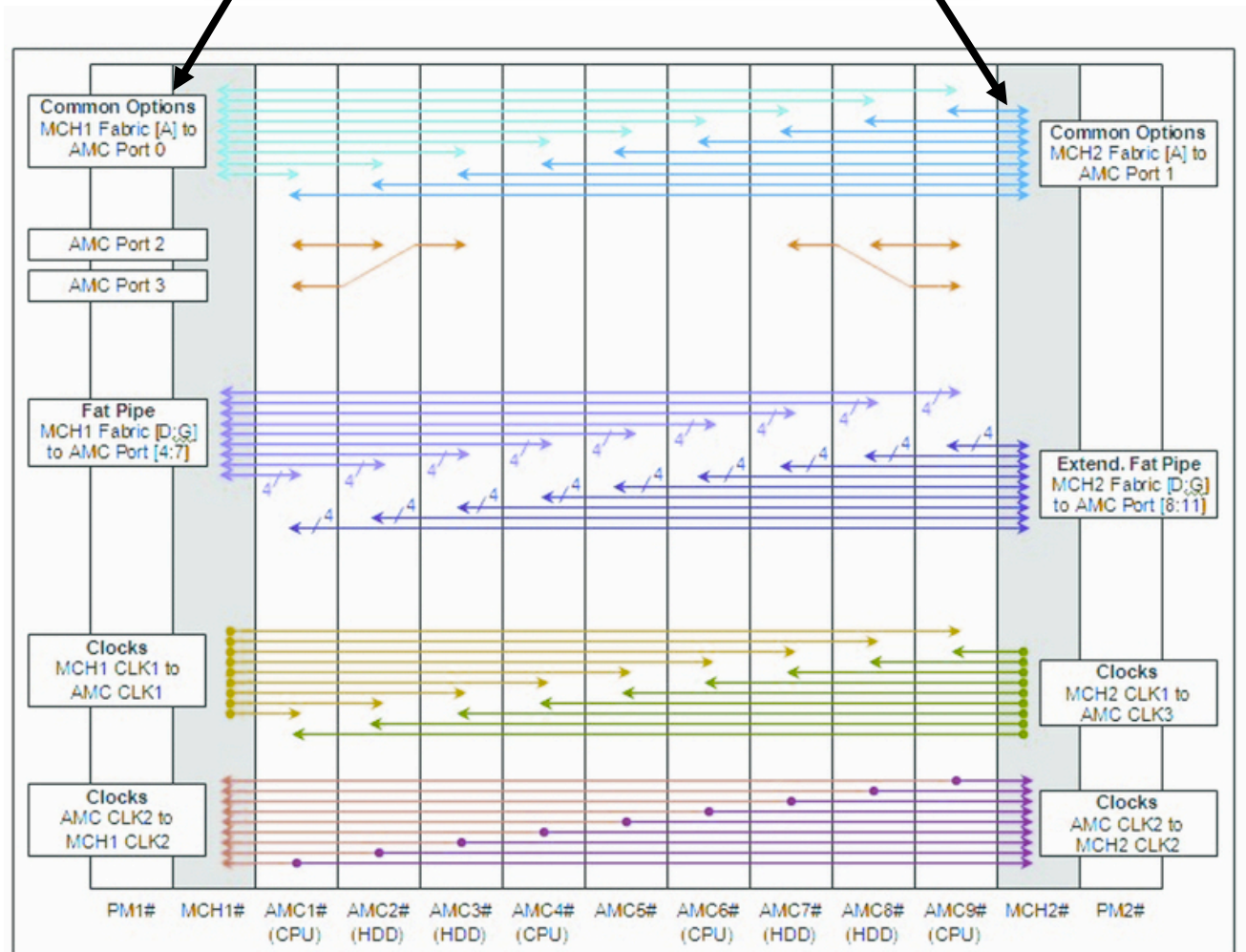
- Upgrade technology for very high lumi
  - Larger state of the art **FPGA devices**
    - Larger granularity needed
    - The trigger needs to cope with more channels
  - **Modern link technology** to interconnect processing boards
    - Multi Gigabit serial links
    - Telecommunication technology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

# uTCA technology: backplane

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

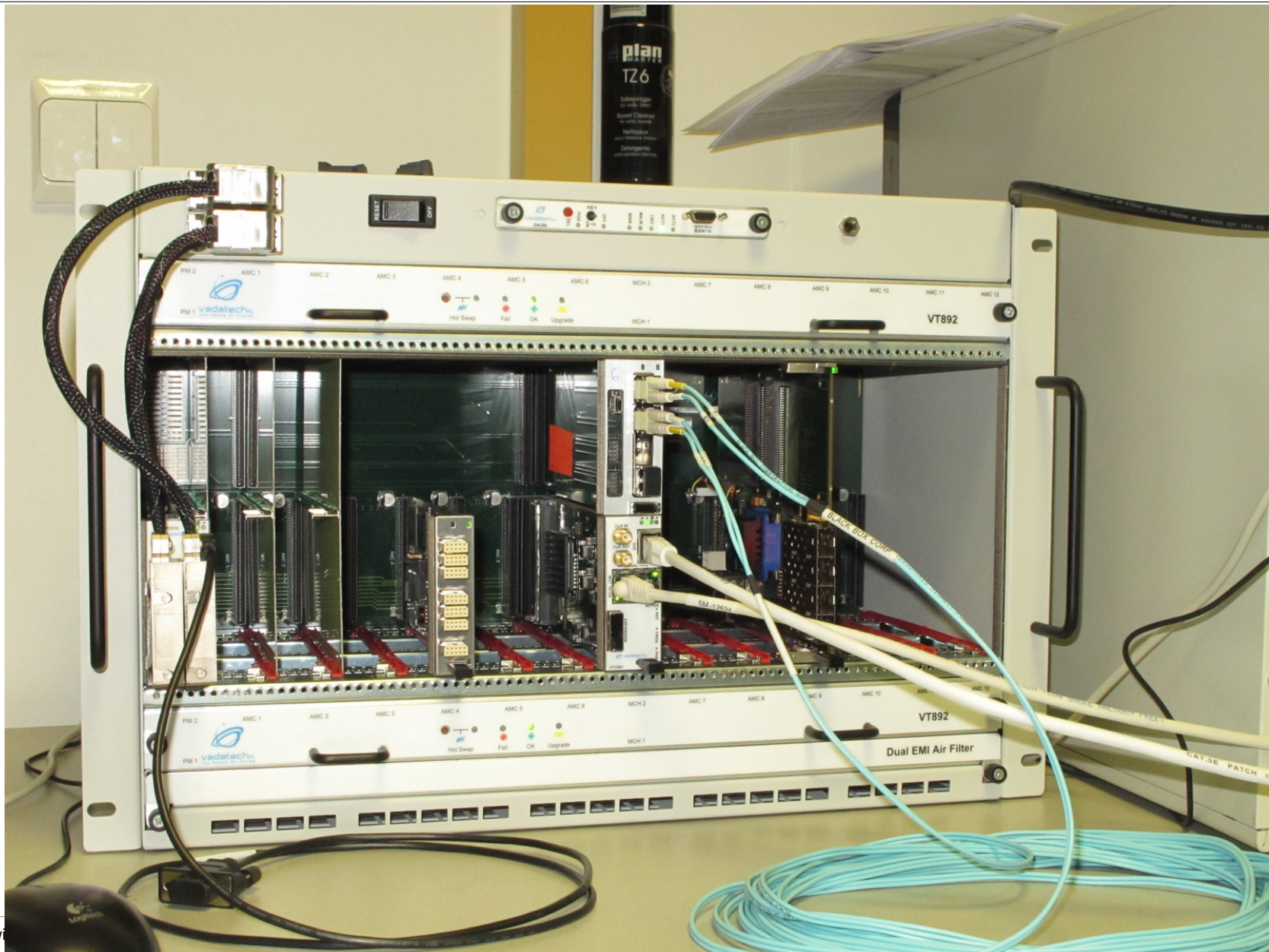
MCH: crate controllers; redundancy with 2 controllers per crate.

“Fatpipes” : High Speed interconnect  
Ideal for trigger modules





# uTCA Technology



# 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 (40Gb/s)
  - HLT farm
    - Multi-core machines
- 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)

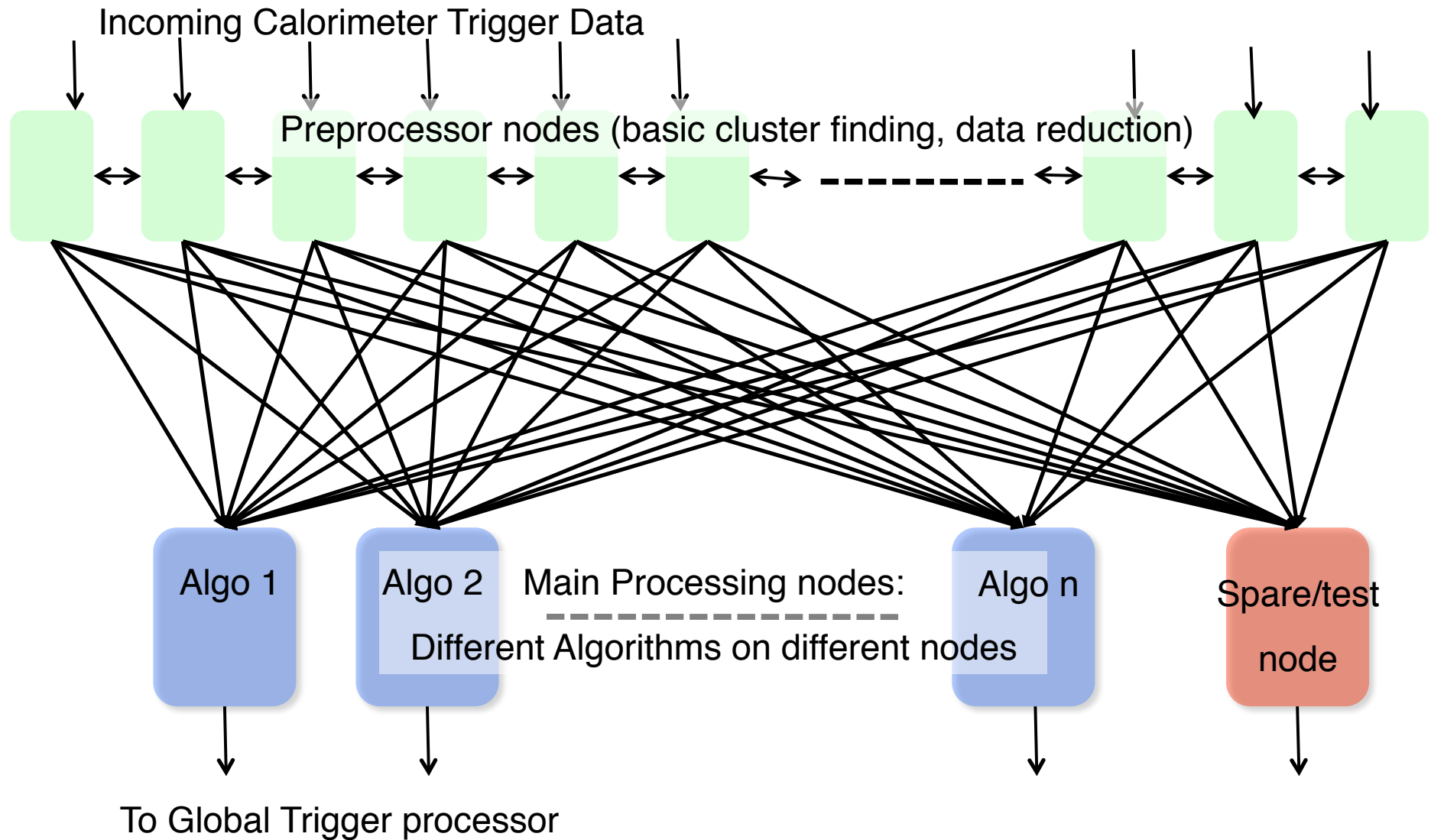
# Upgrade: a few examples

---

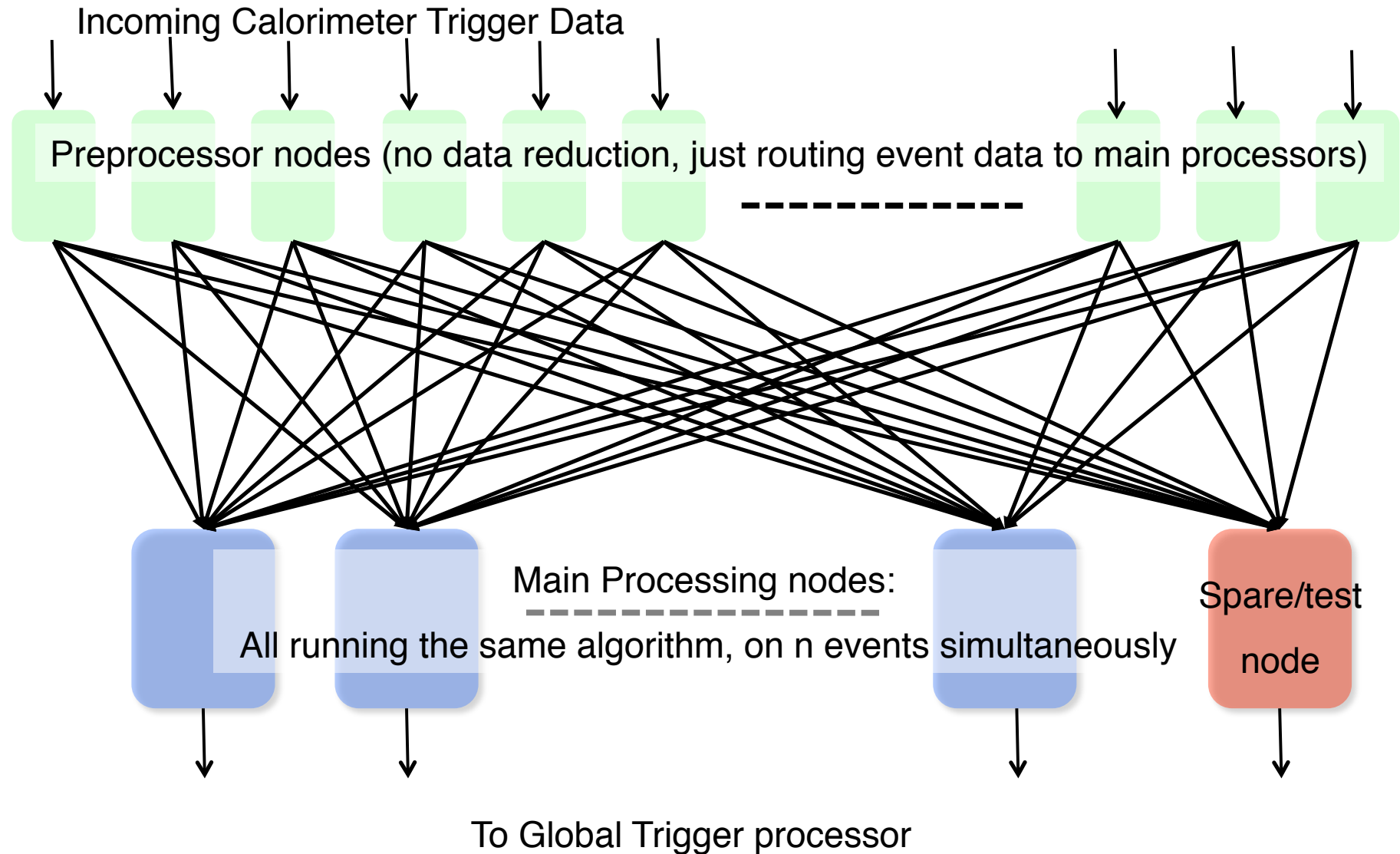
- **ATLAS**
  - Fast Tracker for being used at Lvl2.
  - Generic ATCA modules under study
- **CMS**
  - 10Gb/s readout link. Protocol not yet defined
  - uTCA hardware in Trigger and HCAL (already used in ECAL trigger)
  - 10Gbit or Infiniband based Event Builder under study
- **Alice**
  - Read out detector via Ethernet / UDP (10Gbit)
    - Do the splitting of data via a switch and multicast
    - Goal: Read out Pb collisions at 50kHz
- **LHCb**
  - Readout at 40MHz
  - Trigger in Software



# Upgrade of CMS Calorimeter Trigger: Variant 1

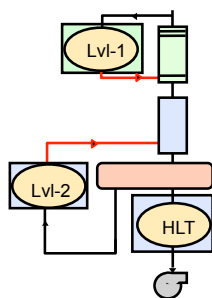


# Upgrade of CMS Calorimeter Trigger: Variant 2



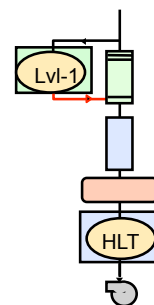
# Conclusions

- Detector Readout: Custom Point to Point Links
- Trigger / DAQ at LHC experiments



Many Trigger levels:

- partial event readout
- complex readout buffer
- “straight forward” EVB



One Trigger level (CMS, LHCb):

- “simple” readout buffer
- high throughput EVB
- complex EVB implementation (custom protocols, firmware)

- **Event-Building**

- Implemented with commercial Network technologies
- Event building is done via “Network-switches” in large distributed systems.
- Event Building traffic leads to network congestion  
Traffic shaping copes with these problems

- **Upgrade**

- The next generation of Trigger/DAQ systems is on the way: **JOIN NOW!**

# The E N D

---

Thank you

and

Have a lot of fun in future projects !!!

# Interesting for you...

---

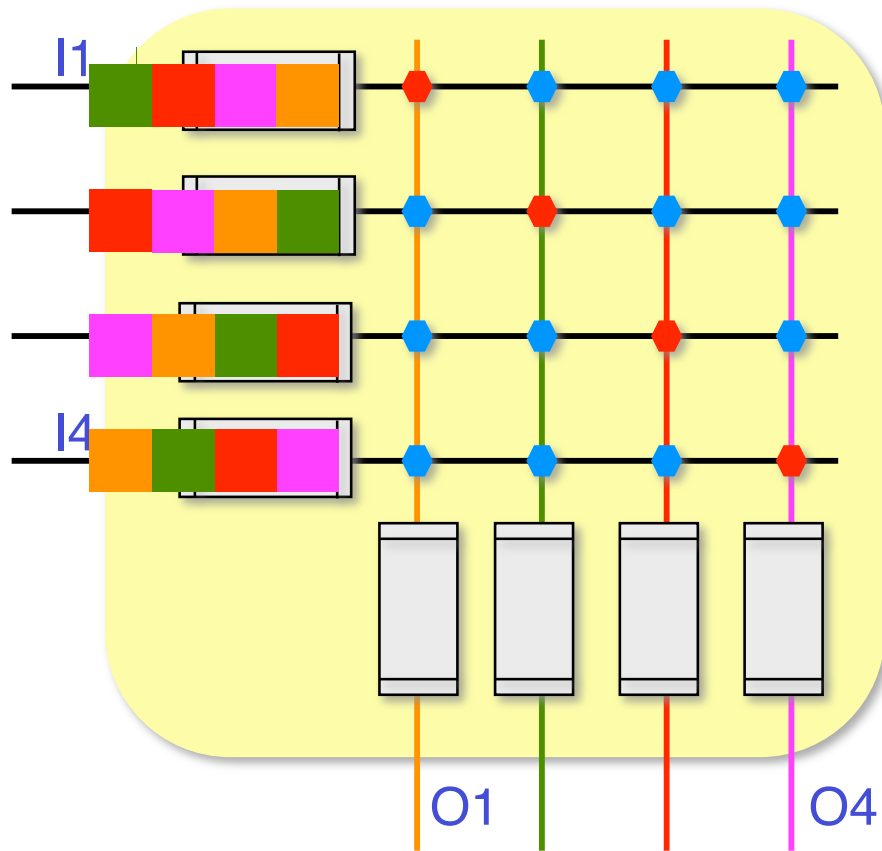
- ...in case you want to participate
  - “Playing” with high tech technology guaranteed
- ...but... the golden time of electronics are over...:  
Once upon a time (in the 90s) a physicist could stick some FPGAs together, write some VHDL code and then claim: I have done an electronics board...
  - Digital 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 switch: perfect scenario



Full wirespeed can be reached  
(sustained) !

# Technology: FPGAs

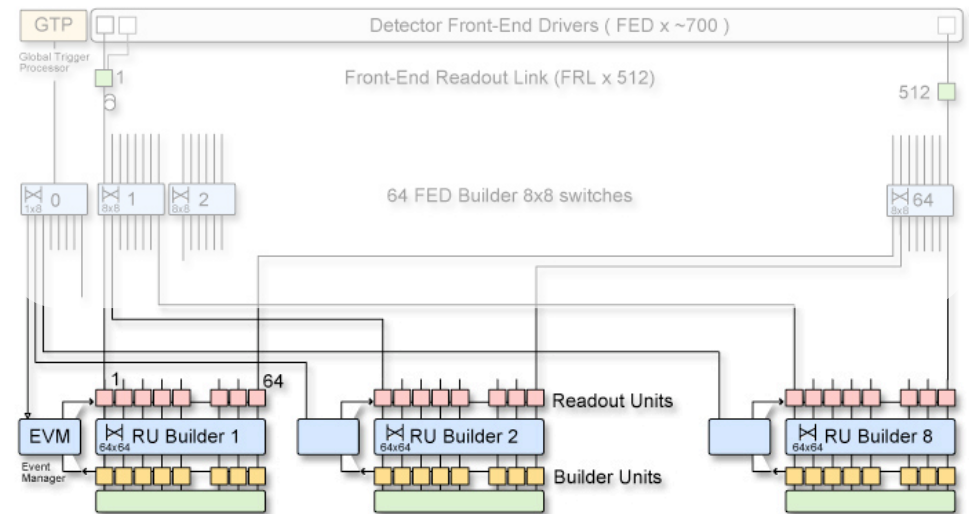
---

- Performance and features of today's "Top Of The Line"
  - XILINX:
    - High Performance Serial Connectivity (3.125Gb/s transceivers):
      - 10GbE Cores, Infiniband, Fibre Channel, ...
    - PCI-express Core (1x and 4x => 10GbE ready)
    - Embedded Processor:
      - 1 or 2 400MHz Power PC 405 cores on chip
  - ALTERA:
    - 3.125 Gb/s transceivers
      - 10GbE Cores, Infiniband, Fibre Channel, ...
    - PCI-express Core
    - Embedded Processors:
      - ARM processor (200MHz)
      - NIOS "soft" RISC: configurable



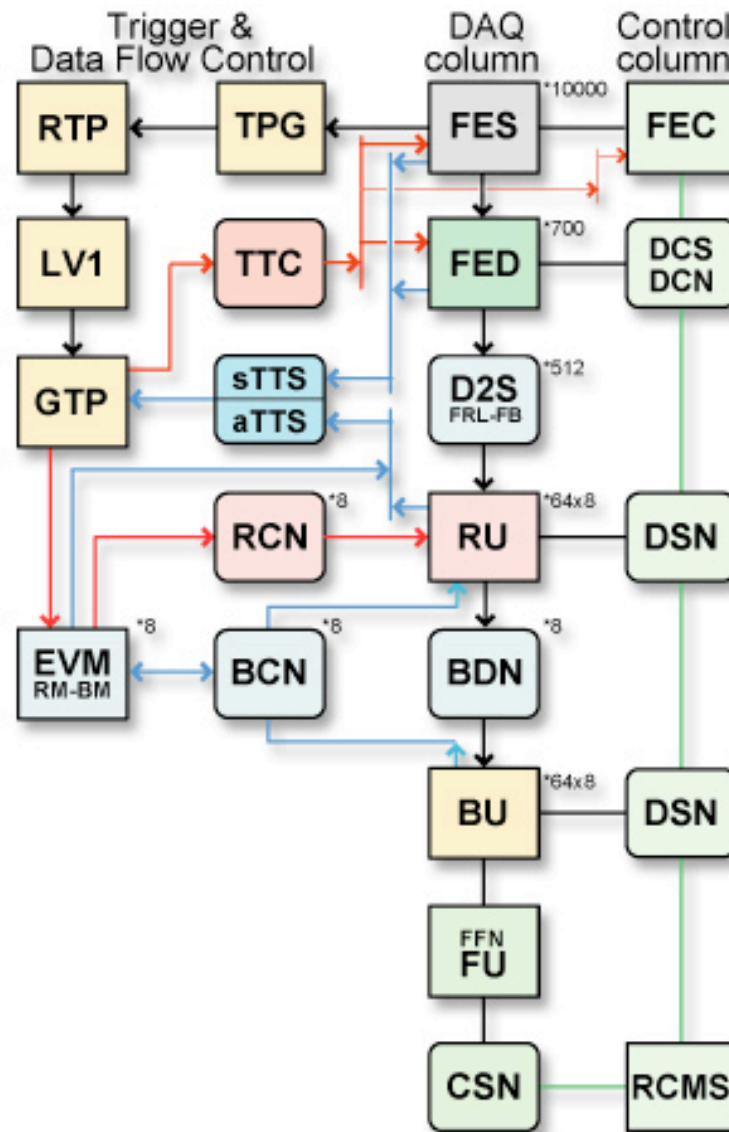
# Stage2: RU-Builder (original plan: 2004)

- **Implementation: Myrinet**
  - Connect 64 Readout-Units to 64 Builder-Units with switch
  - Wire-speed in Myrinet: 250MB/s



- **Avoid blocking of switch: Traffic shaping with Barrel Shifter**
  - Chop event data into fixed size blocks (re-assembly done at receiver)
  - Barrel shifter (next slides)

# Example CMS: data flow

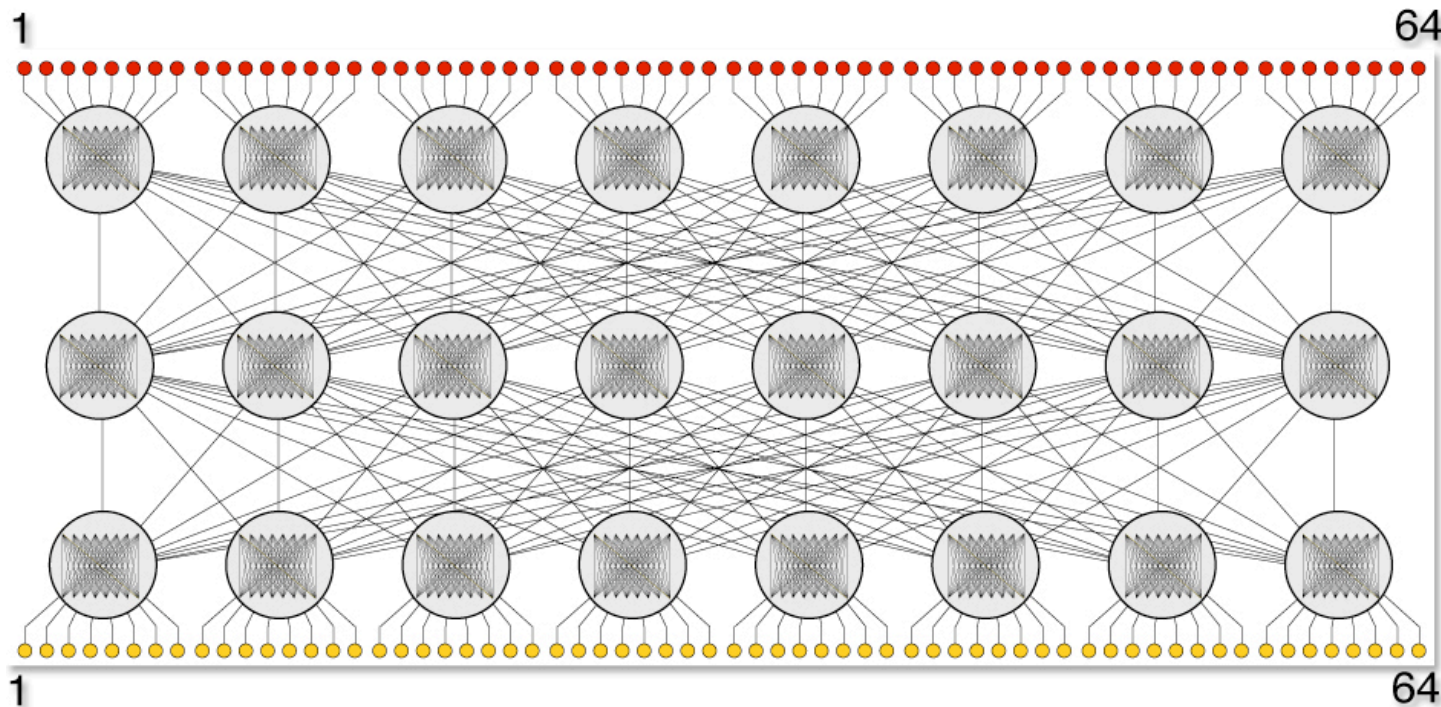


## Acronyms

BCN	Builder Control Network
BDN	Builder Data Network
BM	Builder Manager
BU	Builder Unit
CSN	Computing Service Network
DCS	Detector Control System
DCN	Detector Control Network
DSN	DAQ Service Network
D2S	Data to Surface
EVM	Event Manager
FB	FED Builder
FEC	Front-End Controller
FED	Front-End Driver
FES	Front-End System
FFN	Filter Farm Network
FRL	Front-End Readout Link
FS	Filter Subfarm
GTP	Global Trigger Processor
LV1	Level-1 Trigger Processor
RTP	Regional Trigger Processor
RM	Readout Manager
RCN	Readout Control Network
RCMS	Run Control and Monitor System
RU	Readout Unit
TPG	Trigger Primitive Generator
TTC	Timing, Trigger and Control
sTTS	synchronous Trigger Throttle System
aTTS	asynchronous Trigger Throttle System

# Myrinet (old switch)

- network built out of crossbars (Xbar16)
- wormhole routing, built-in back pressure (no packet loss)
- **switch**: 128-Clos switch crate
  - **64x64 x 2.0** Gbit/s port (bisection bandwidth **128 Gbit/s**)
- **NIC**: M3S-PCI64B-2 (LANai9 with RISC), custom Firmware

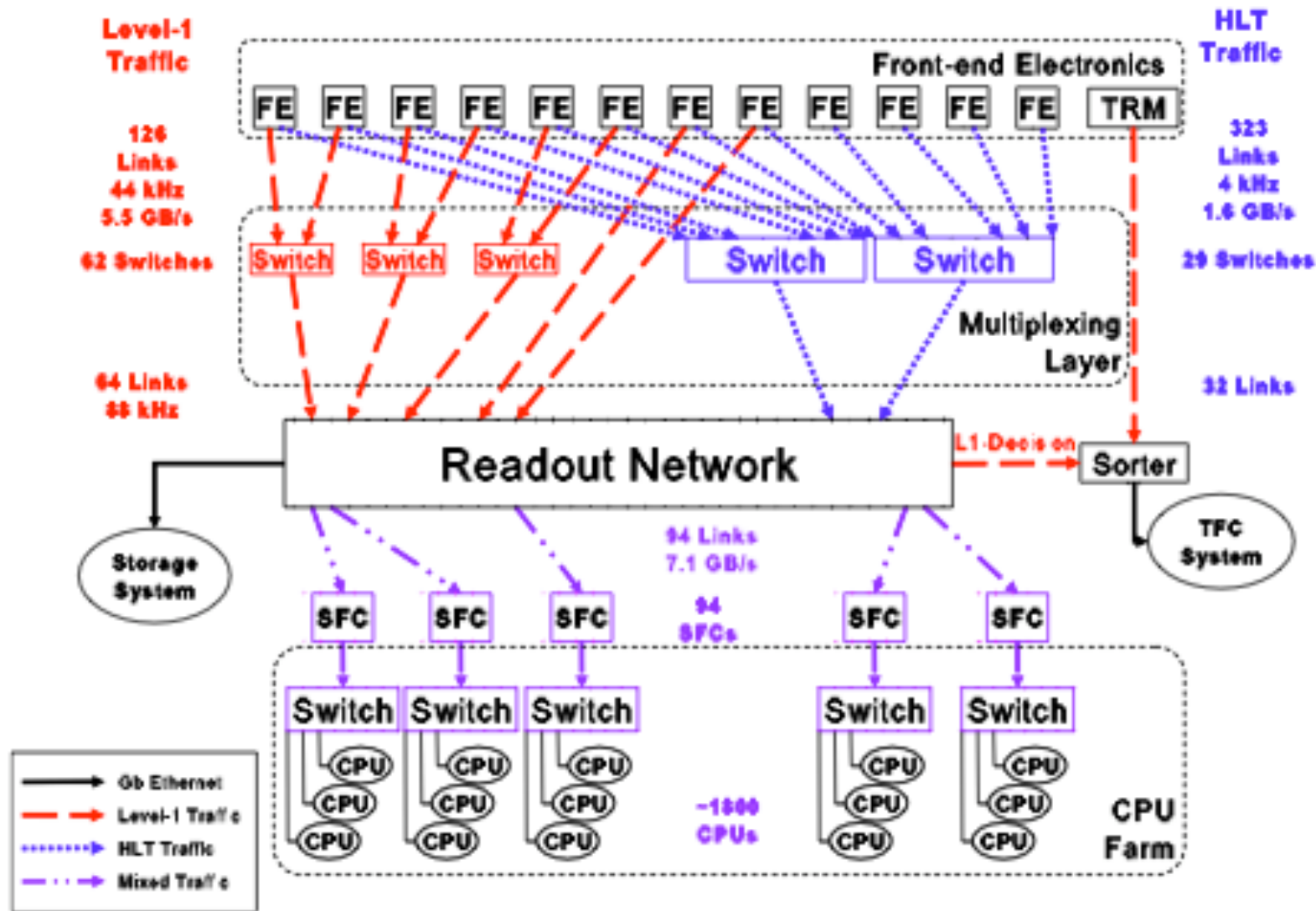


# LHCb

---

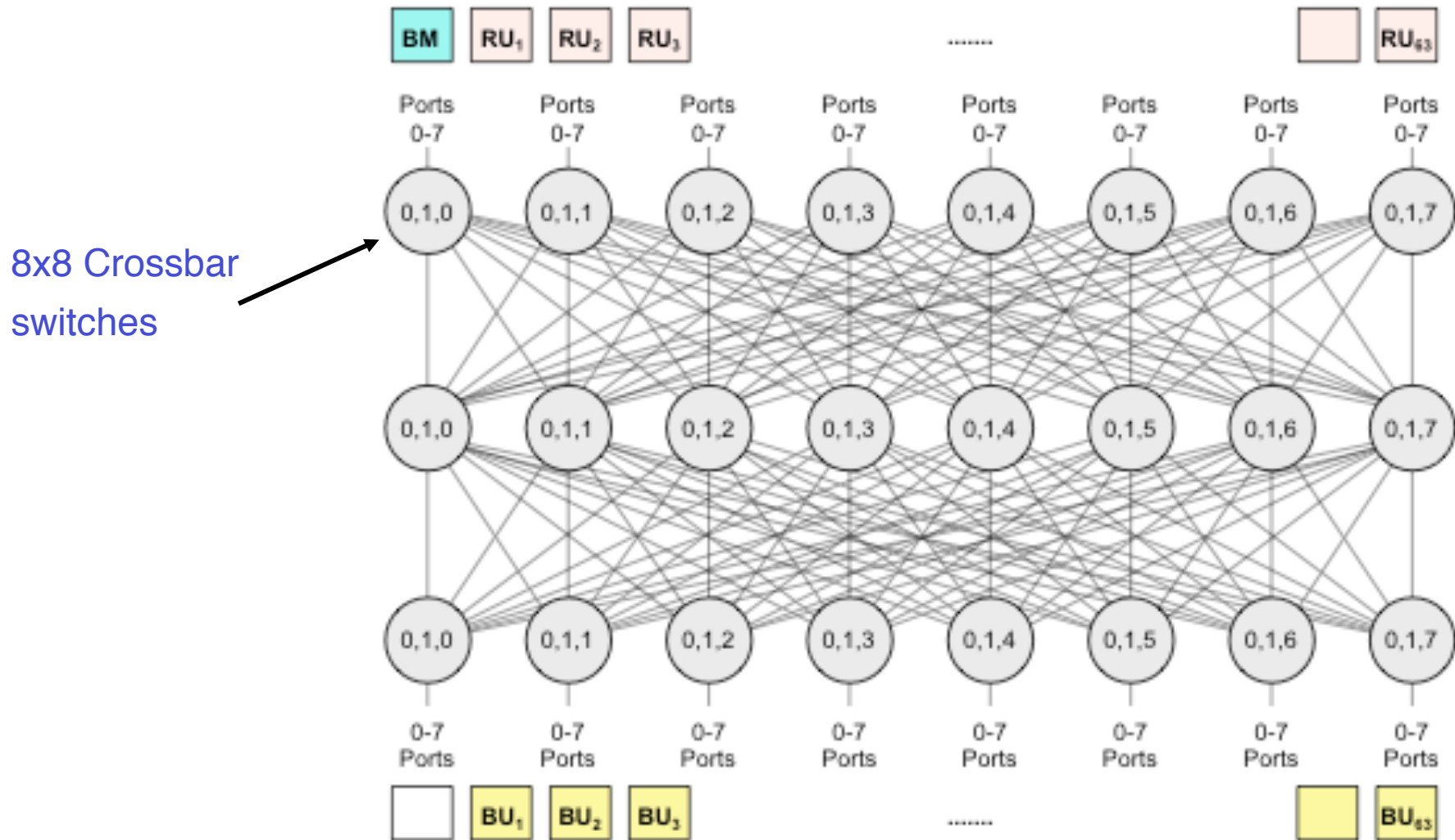
- Operate at  $L = 2 \times 10^{32} \text{ cm}^{-2}\text{s}^{-1}$ : 10 MHz event rate
- Lvl0: 2-4 us latency, 1MHz output
  - Pile-up veto, calorimeter, muon
- Lvl1: 52.4ms latency, 40 kHz output
  - Impact parameter measurements
  - Runs on same farm as HLT, EVB
- Pile up veto
  - Can only tolerate one interaction per bunch crossing since otherwise always a displaced vertex would be found by trigger

# LHCb L1-HLT-Readout network





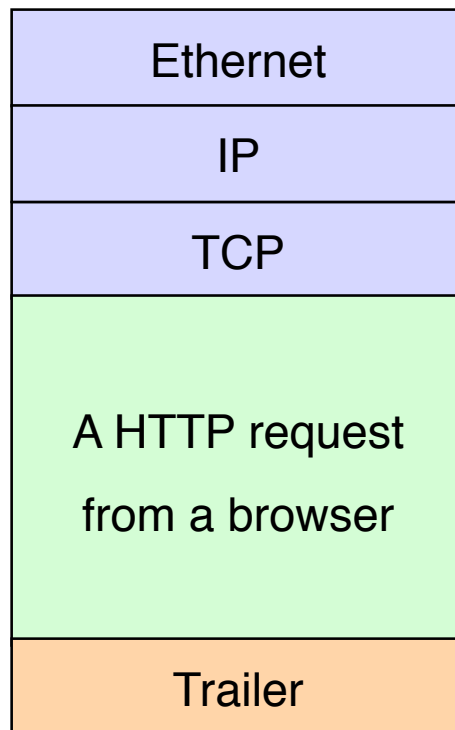
# Clos-switch 128 from 8x8 Crossbars



# Intermezzo: Networking

---

- TCP/IP on Ethernet networks
  - All data packets are surrounded by headers and a trailer



Ethernet:

- Addresses understood by hardware (NIC and switch)

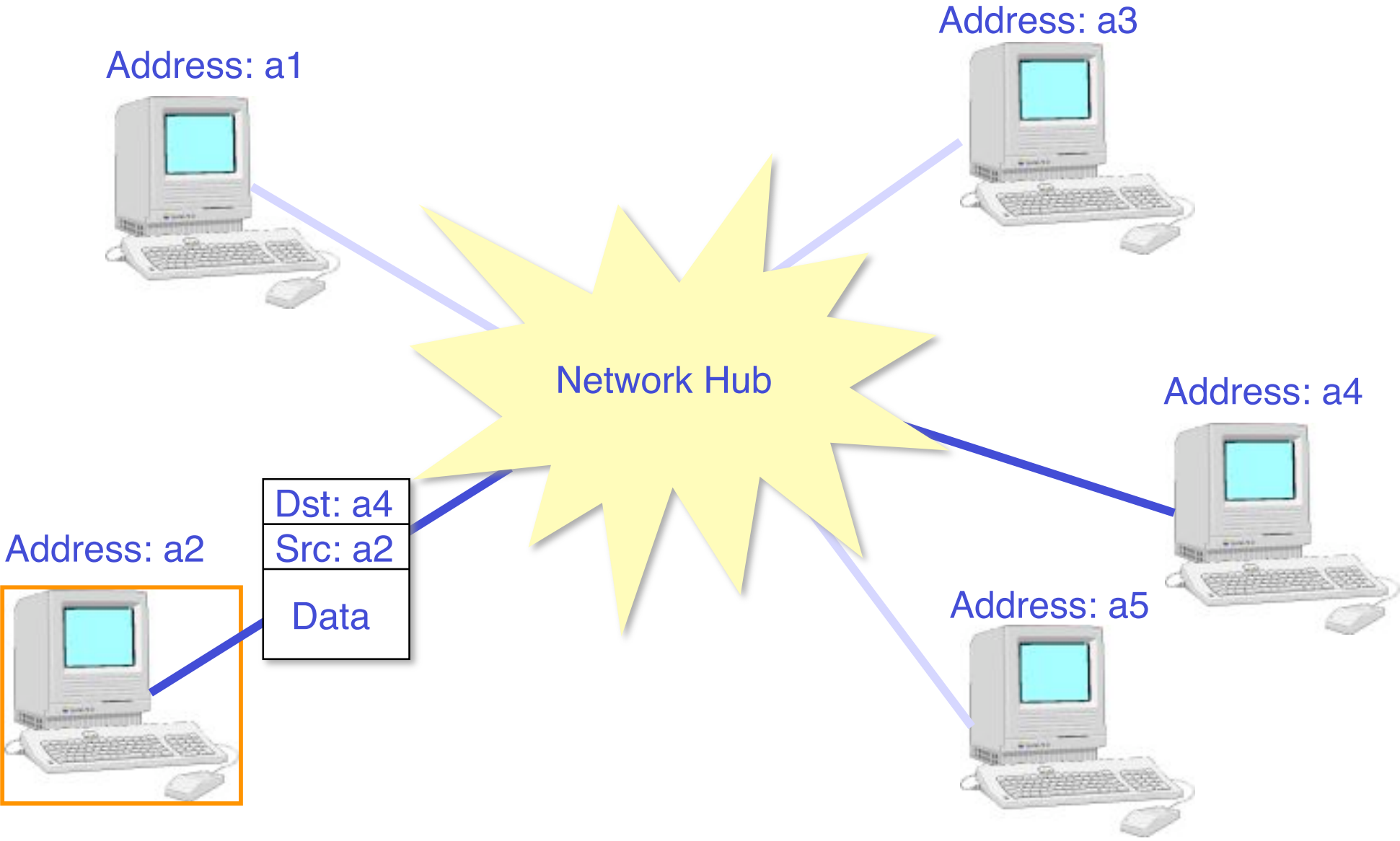
IP:

- unique addresses (world wide) known by DNS (you can search for [www.google.com](http://www.google.com))

TCP:

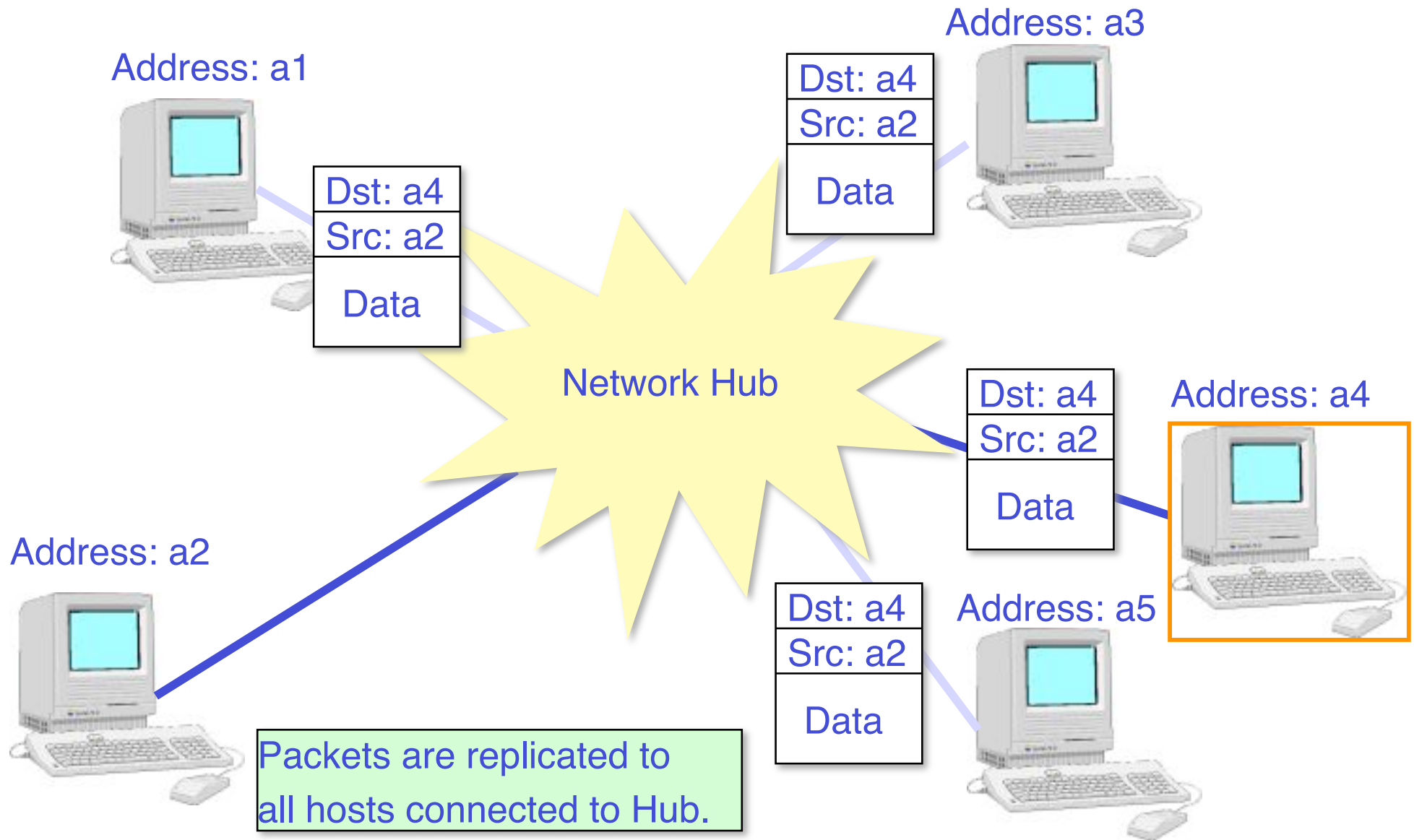
- Provides programmer with an API.
- Establishes “connections” = logical communication channels (“socket programming”)
- Makes sure that your packet arrives: requires an acknowledge for every packet sent (retries after timeout)

# Intermezzo: Networking & Switches

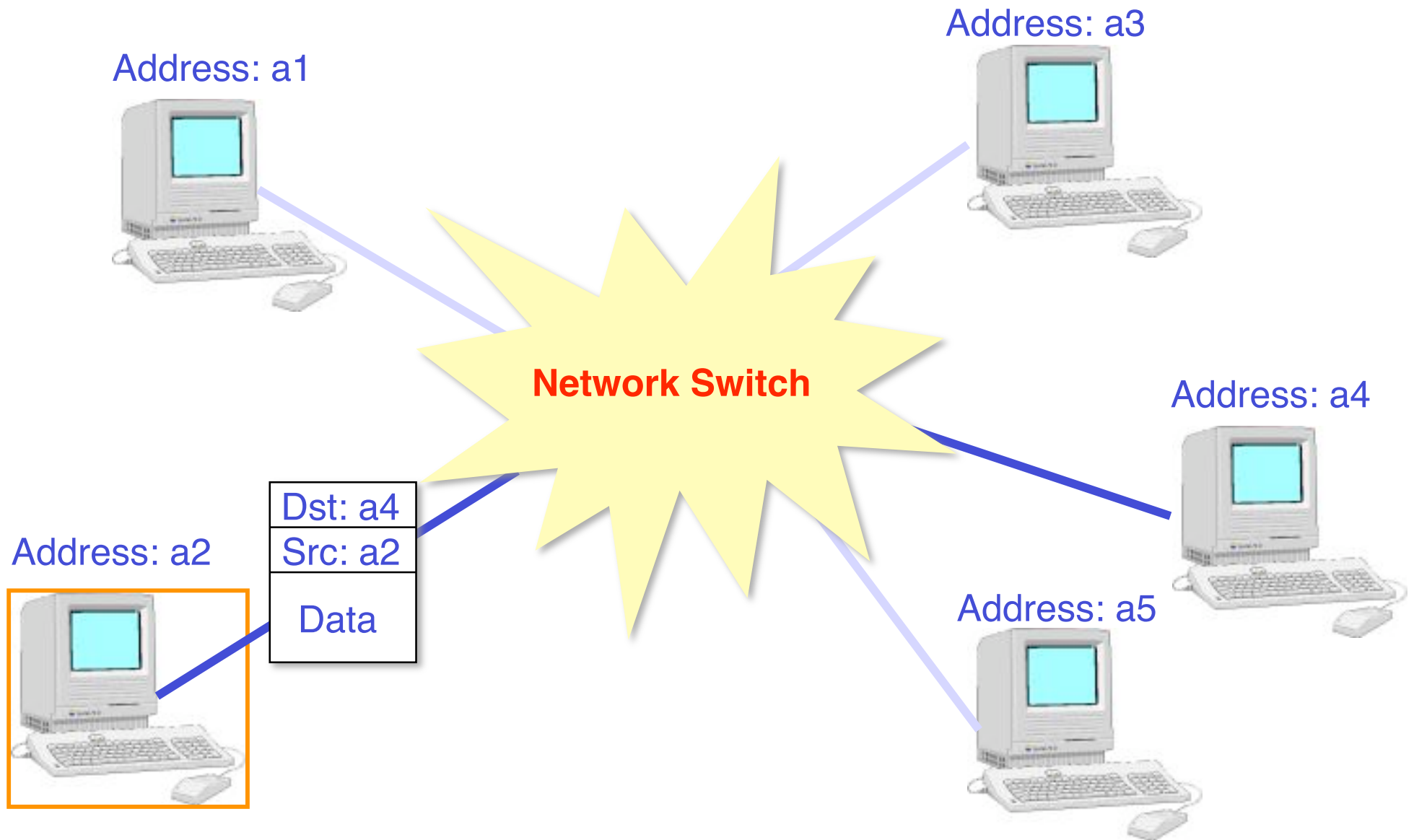




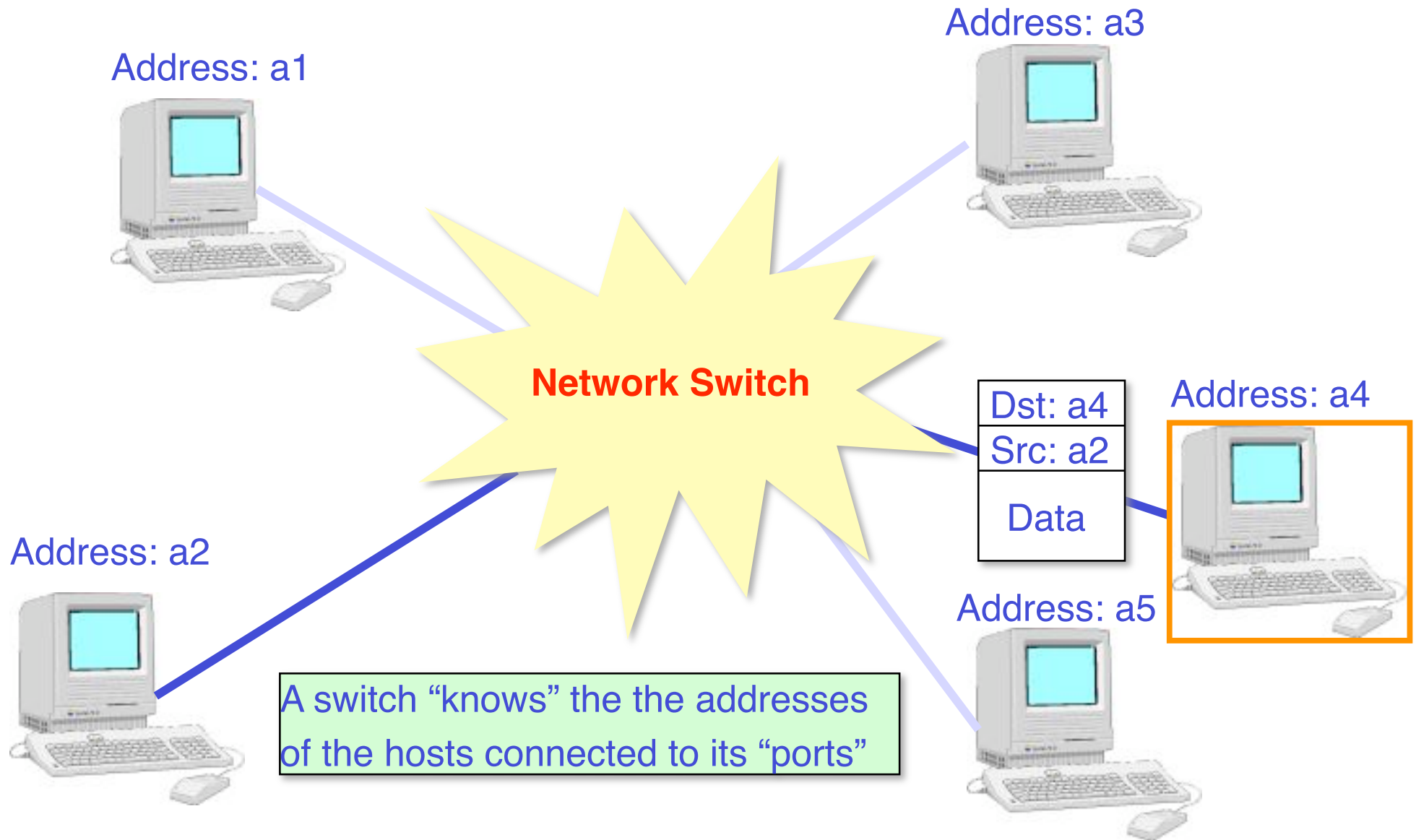
# Intermezzo: Networking & Switches



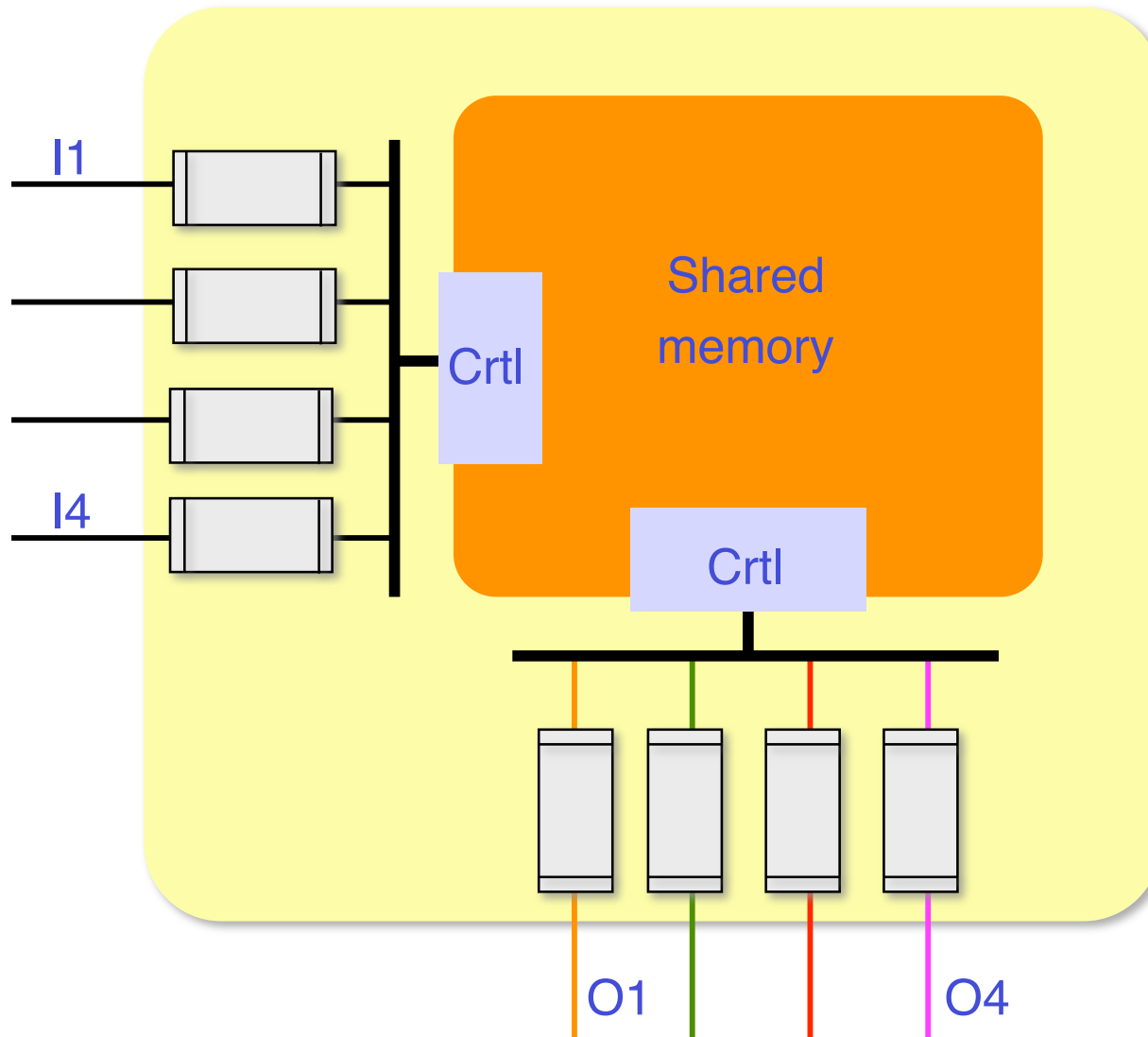
# Intermezzo: Networking & Switches



# Intermezzo: Networking & Switches



# Alternative switch implementation



Similar issue:

The behavior of the switch (blocking or non-blocking) depends largely on the amount of internal memory (FIFOs and shared memory)



# Implementation of EVBs and HLTs today

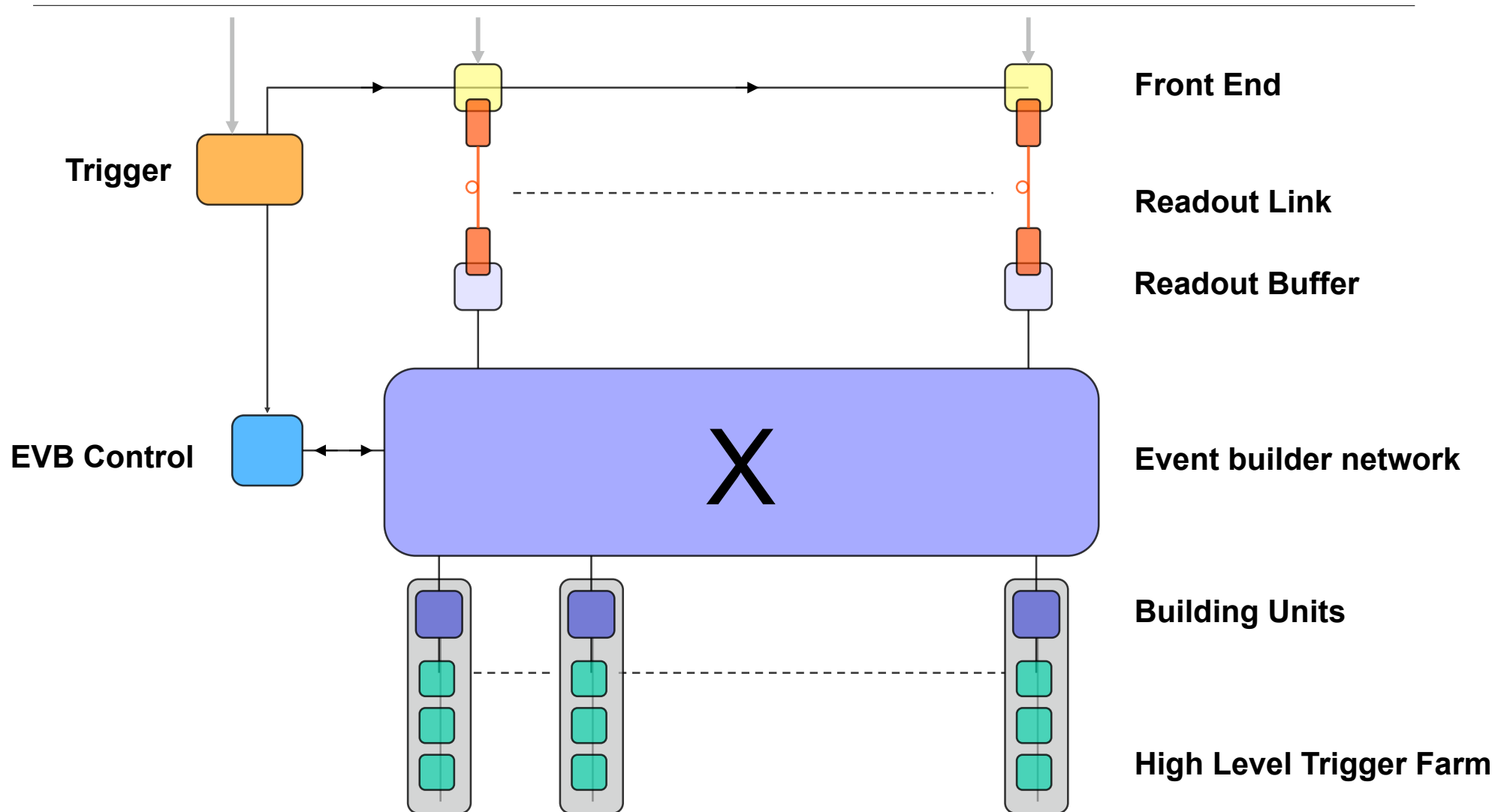


Higher level triggers are implemented in software. Farms of PCs investigate event data in parallel.

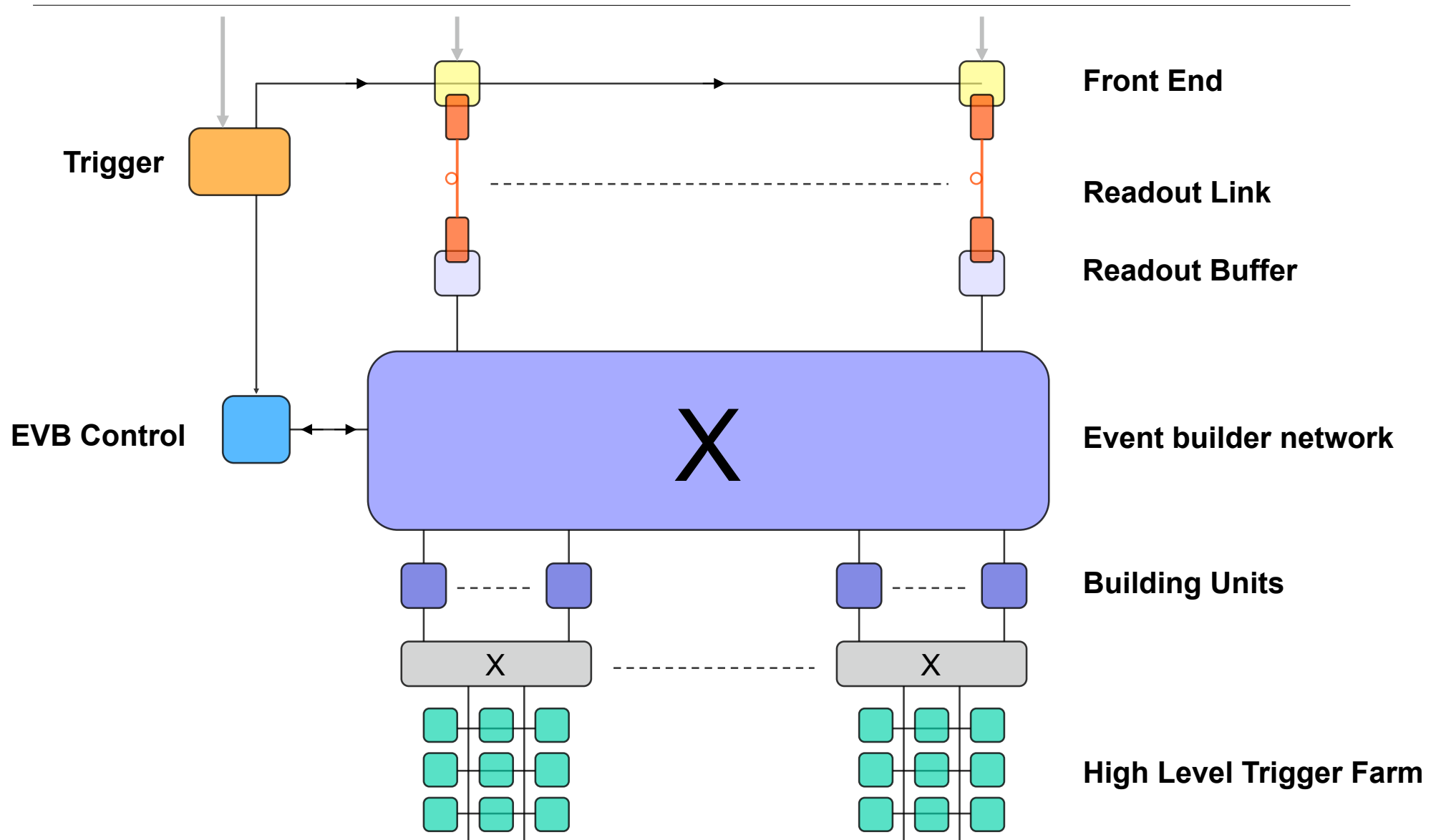


Eventbuilder and HLT Farm resemble an entire “computer center”

# Current EVB architecture



# Future EVB architecture?



# Future EVB architecture I

