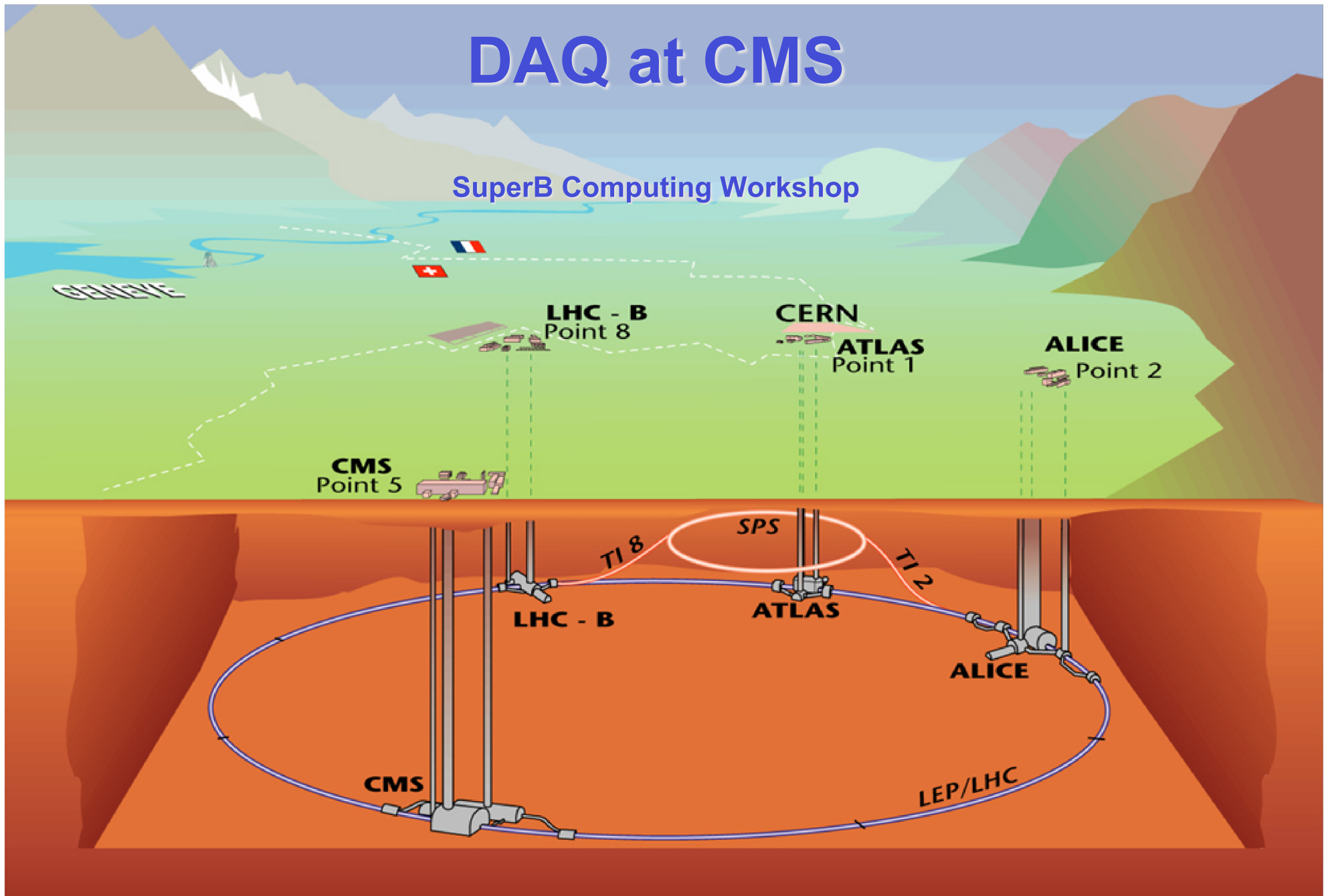


DAQ at CMS

SuperB Computing Workshop



Contents

The context: LHC & CMS experiment

Data Flow of the DAQ system

The Interface to the Front End: Readout-links

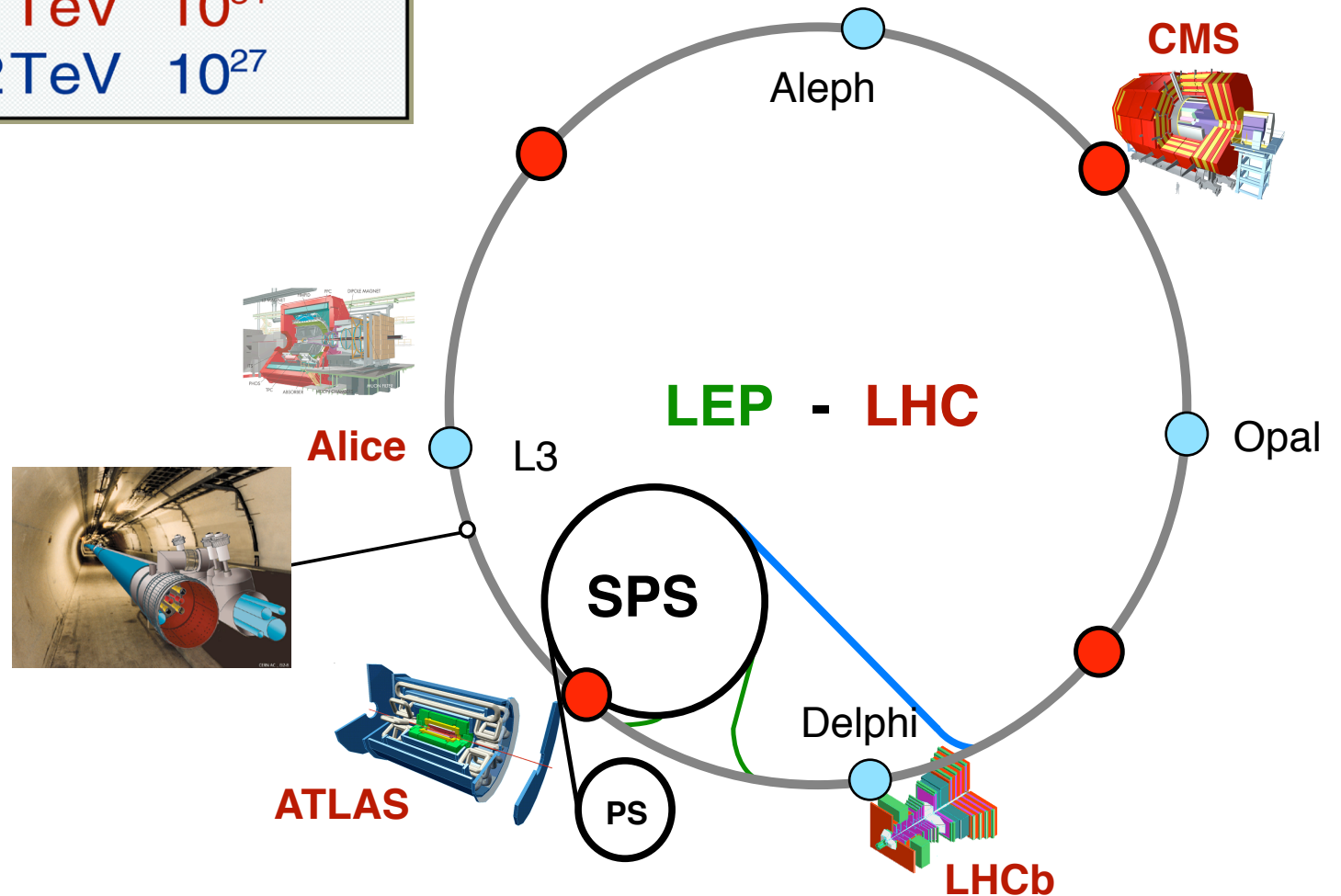
Event Building in CMS

Online Software techniques

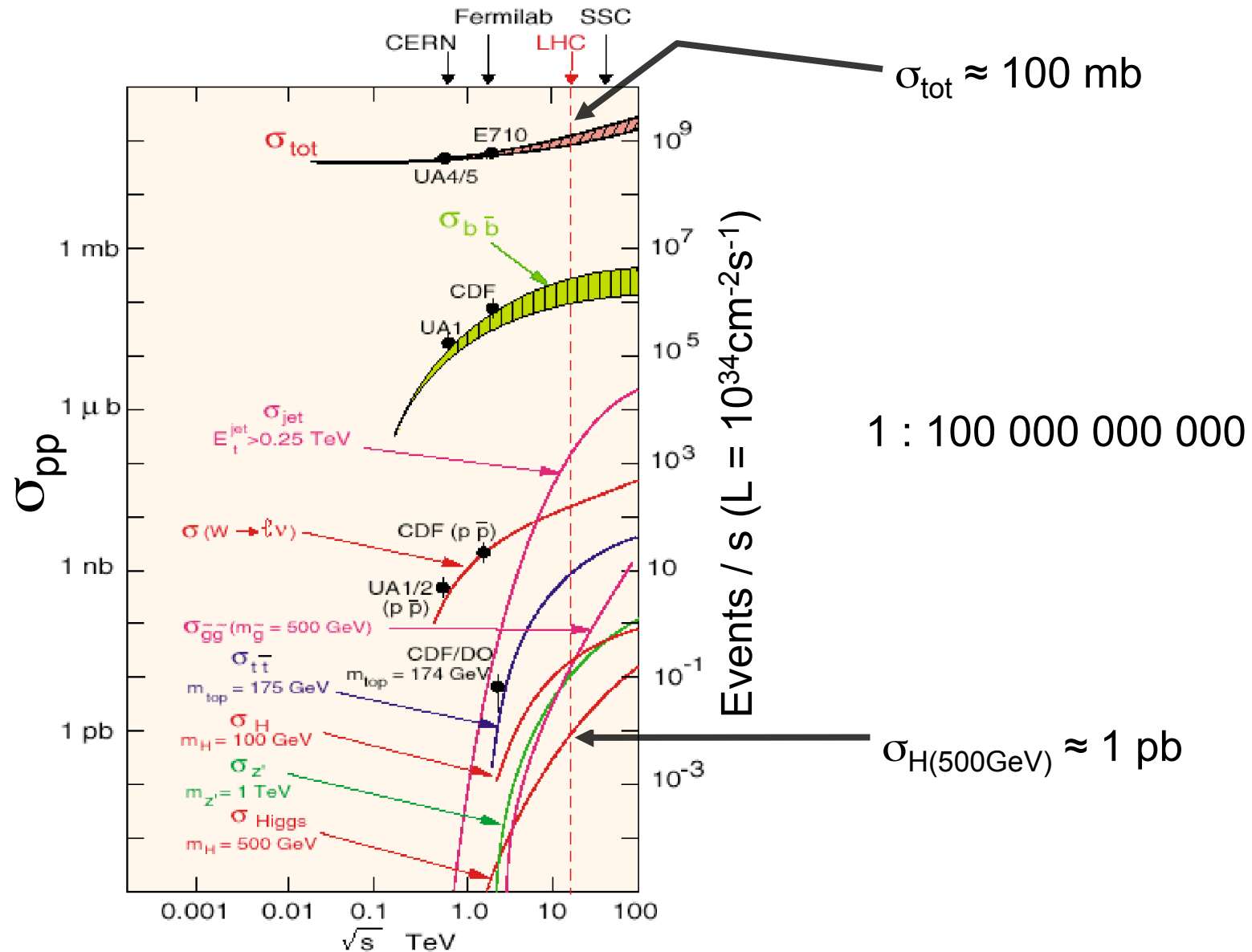
Introduction: LHC

LHC: a “discovery” machine

	Beams	Energy	Luminosity
LEP	e^+e^-	200 GeV	$10^{32} \text{ cm}^{-2}\text{s}^{-1}$
LHC	$p p$	14 TeV	10^{34}
	$P_b P_b$	1312 TeV	10^{27}



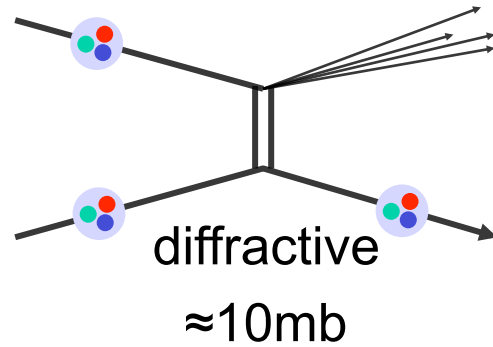
Interesting Physics at LHC



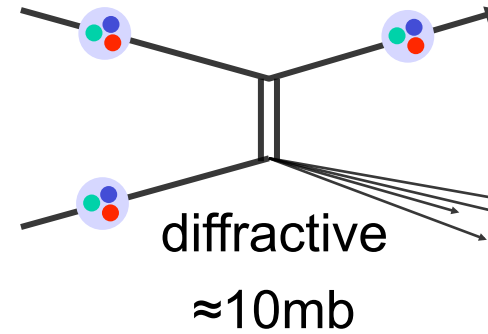
p-p interactions at LHC

$$\sigma_{\text{tot}} =$$

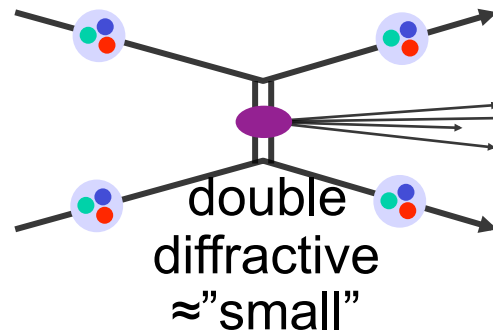
$\approx 100\text{mb}$



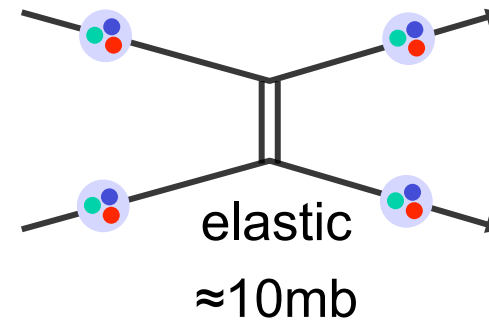
+



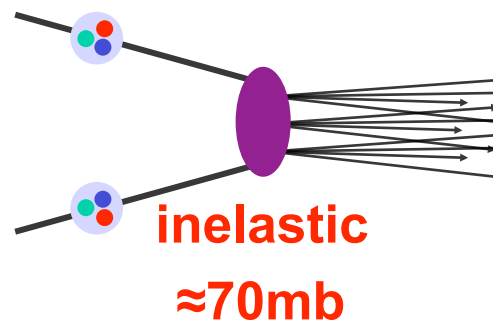
+



+

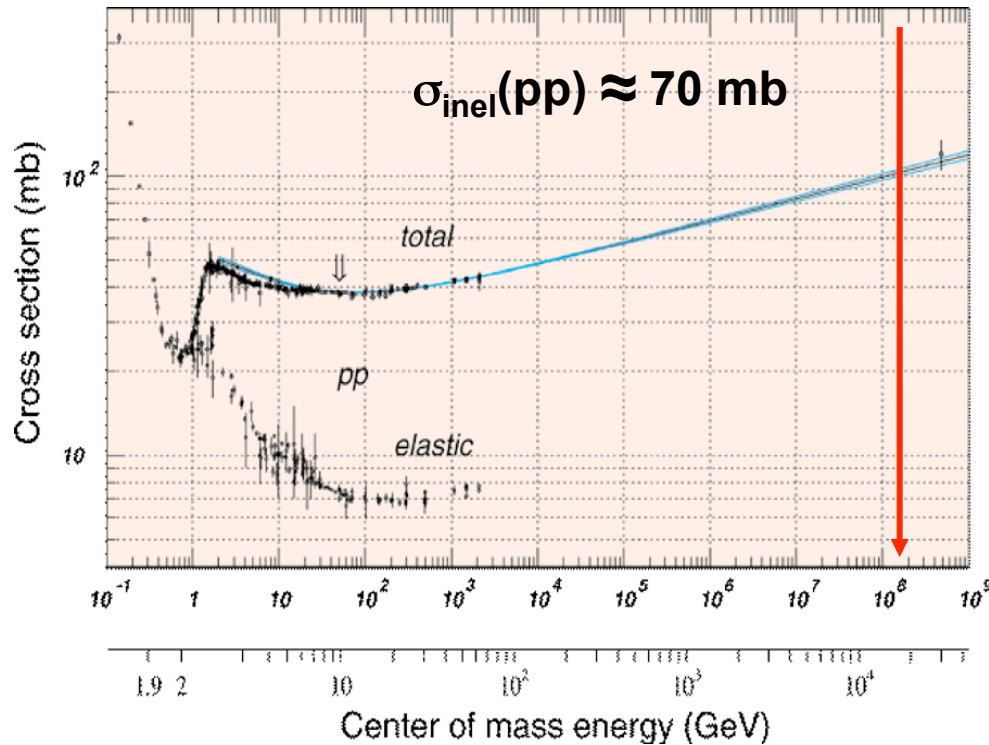


+



Interesting Physics
Many signals in detector

LHC: experimental environment



- $L=10^{34}\text{cm}^{-2}\text{s}^{-1}$
- $\sigma_{inel}(pp) \approx 70\text{mb}$
event rate = $7 \times 10^8\text{Hz}$
- $\Delta t = 25\text{ns}$
events / 25ns = 17.5
- Not all bunches filled (2835 out of 3564)
events/crossing = 23

The CMS Experiment

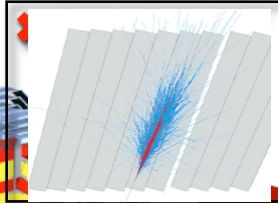
CMS detector: pp and heavy ions

SUPERCONDUCTING COIL

Total weight : 12,500 t
 Overall diameter : 15 m
 Overall length : 21.6 m
 Magnetic field : 4 Tesla

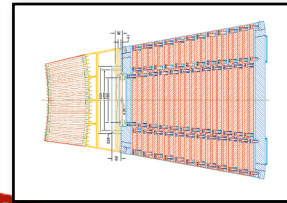
CALORIMETERS

ECAL Scintillating PbWO_4 Crystals



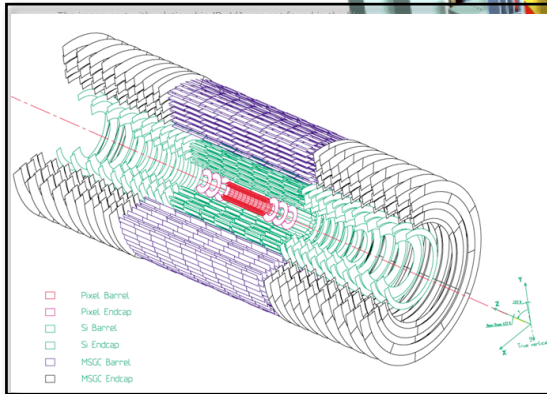
HCAL Plastic scintillator

brass sandwich



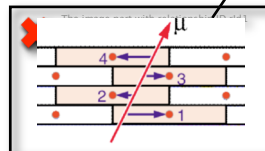
IRON YOKE

TRACKERS



Silicon Microstrips
 Pixels

MUON BARREL

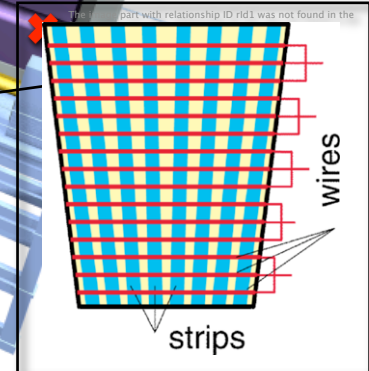


Drift Tube Chambers (**DT**)



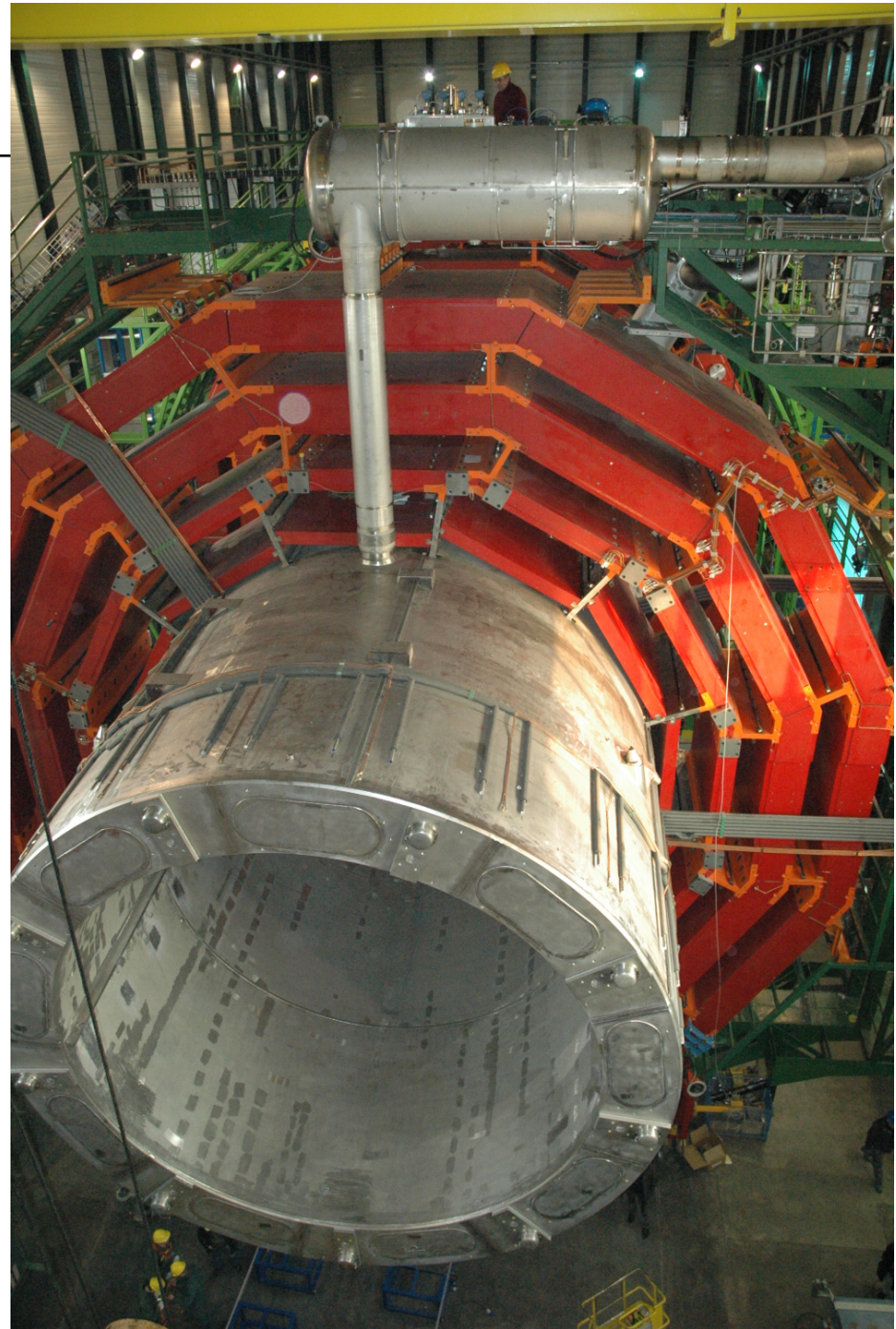
Resistive Plate Chambers (**RPC**)

MUON ENDCAPS

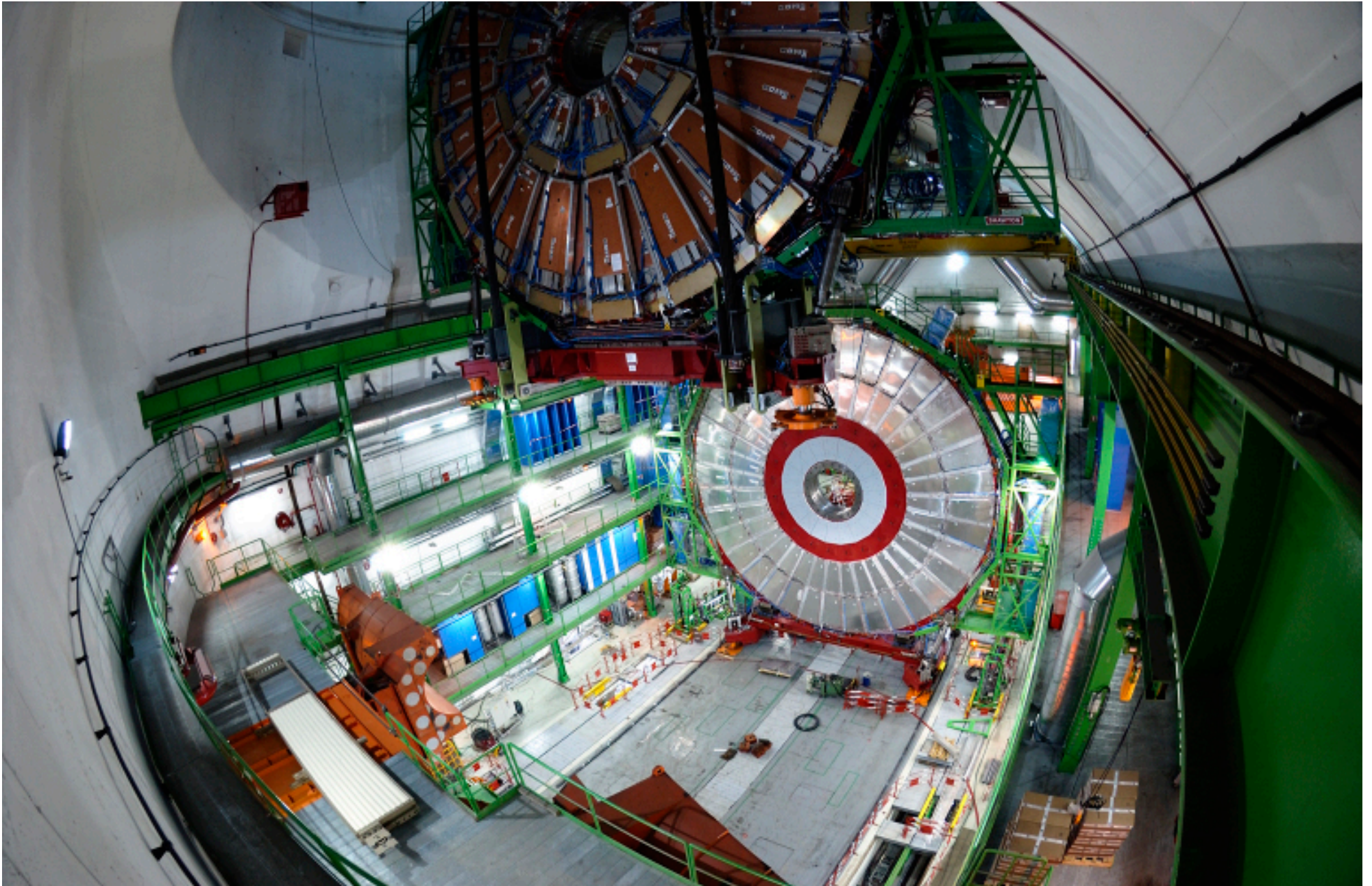


Cathode Strip Chambers (**CSC**)
 Resistive Plate Chambers (**RPC**)

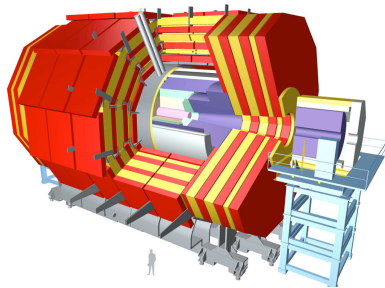
Superconducting 4T Magnet of CMS



Heavy lowering in CMS



The LHC Experiments



CMS

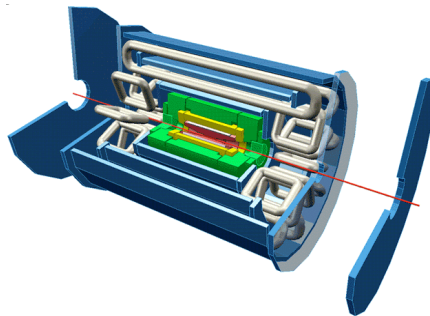
Study of pp & heavy ion collisions

Tracker: Si (Pixel, Strips, Discs)

Calorimeters: BGO, Brass Scintillators, Preshower

Muon System: RPC, MDT, CSC,

Supraconducting solenoid



ATLAS

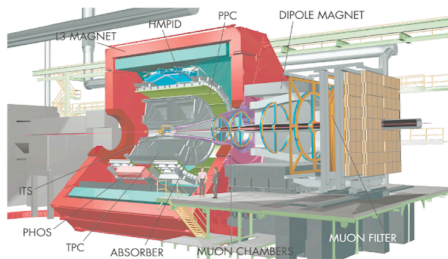
Study of pp collisions

Tracker: Si (Pixel and SCT), TRT

Calorimeters: LAr, Scintillating Tiles

Muon System: MDT, RPC, TGC, CSC,

Magnets: Solenoid and Toroid



ALICE

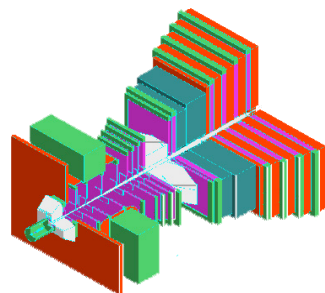
Study of heavy ion collisions

Tracker: Si (ITS), TPC, Chambers, TRD, TOF

Particle Id: RICH, PHOS (scintillating crystals)

RPC, FMD (forward mult.; Si) ZDC (0 degree cal)

Magnets: Solenoid, Dipol



LHCb

Study of CP violation in B decays (pp)

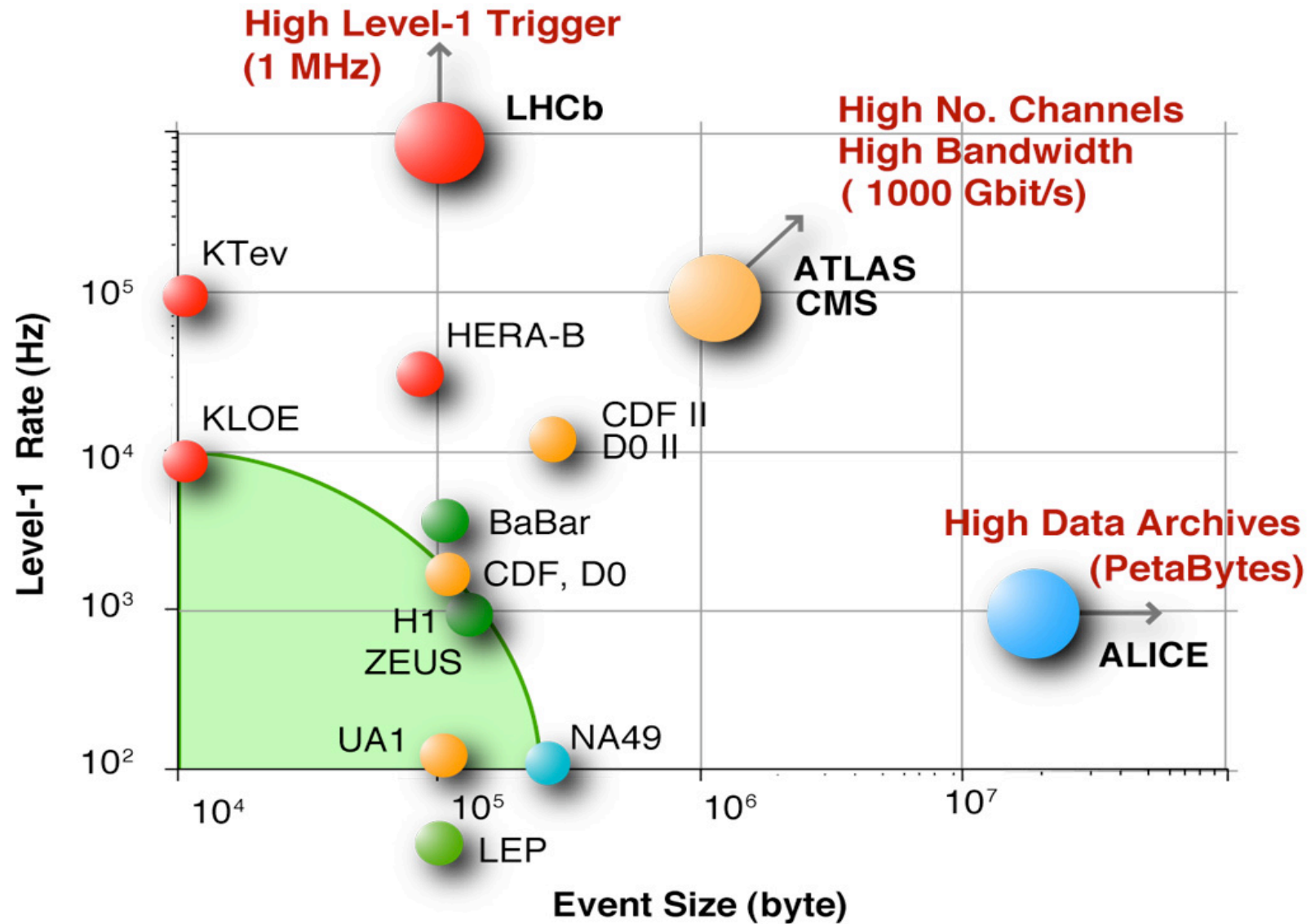
Tracker (Si, Velo), 2 RICH, 4 Tracking stations (Straw-

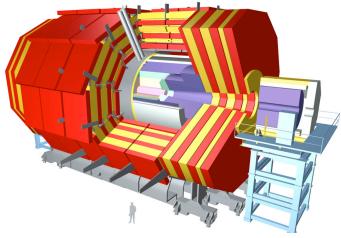
Tubes, Si), SPD (scintill. Pads), Preshower, ECAL (lead

scintillator) HCAL (steel scintillator), Muon stations (MWPCs)

Trigger & DAQ parameters

LHC experiments: Lvl 1 rate - evt. size





Trigger/DAQ parameters

Item	Today (2008)	2009	Final design
Lvl1 trigger rate	20-60kHz	50kHz	100 kHz
Event size	1.5 MB	1.5 MB	1.5 MB
Readout / EVB Bandwidth	100 GB/s	100 GB/s	100 GB/s
High Lvl Trigger Output to local disk	~500 MB/s	~2 GB/s	~2 GB/s
Local disk space available (at CMS)	~80 TB	~300 TB	???
Transfer line to Tier0	1x10 Gbit/sec	2x10 Gbit/sec (for redundancy)	2x10 Gbit/sec (for redundancy)
Data to tape at Tier0	300 MB/s	???	???

- **Remarks:**

- These values evolve continuously depending on
 - Needs of the experiment (experience, HLT software, physics, ...)
 - Technology available (within budget)

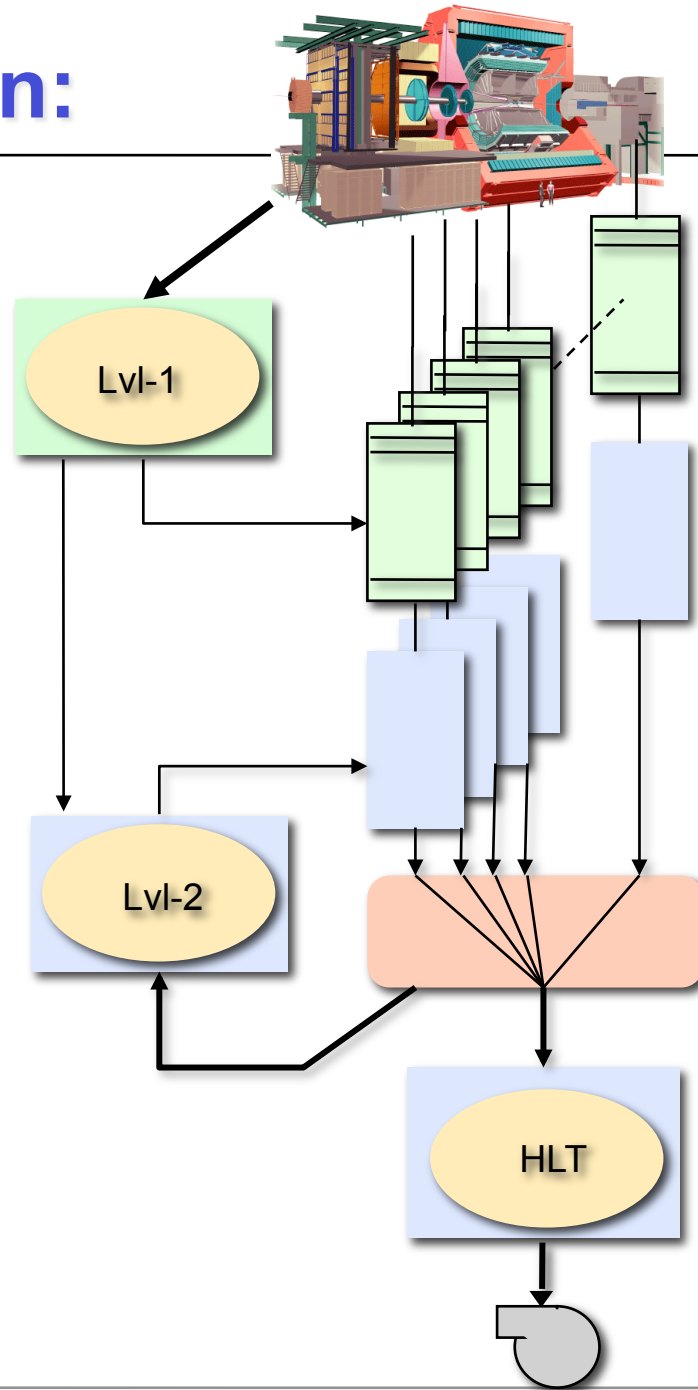
Data Flow: Architecture

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

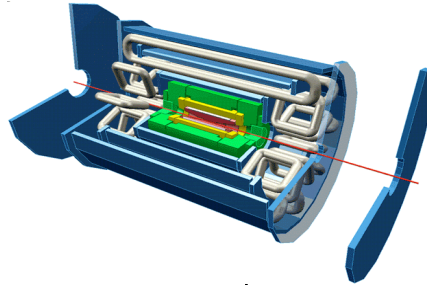
Assemble events in single location and provide to High Level Trigger (HLT)

“Standard Model”
of Data Flow

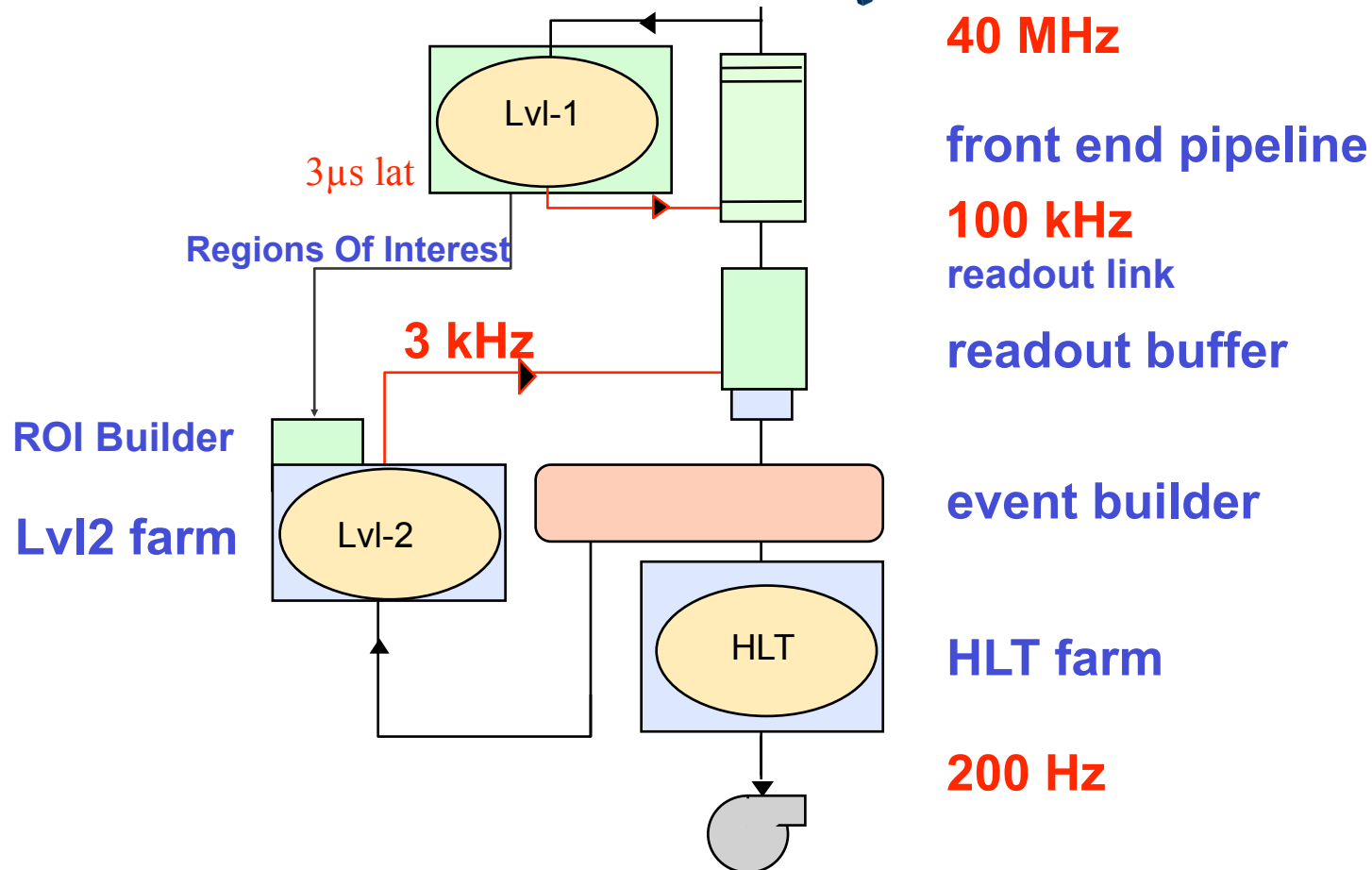
Write selected events to permanent storage

Data Flow: ATLAS

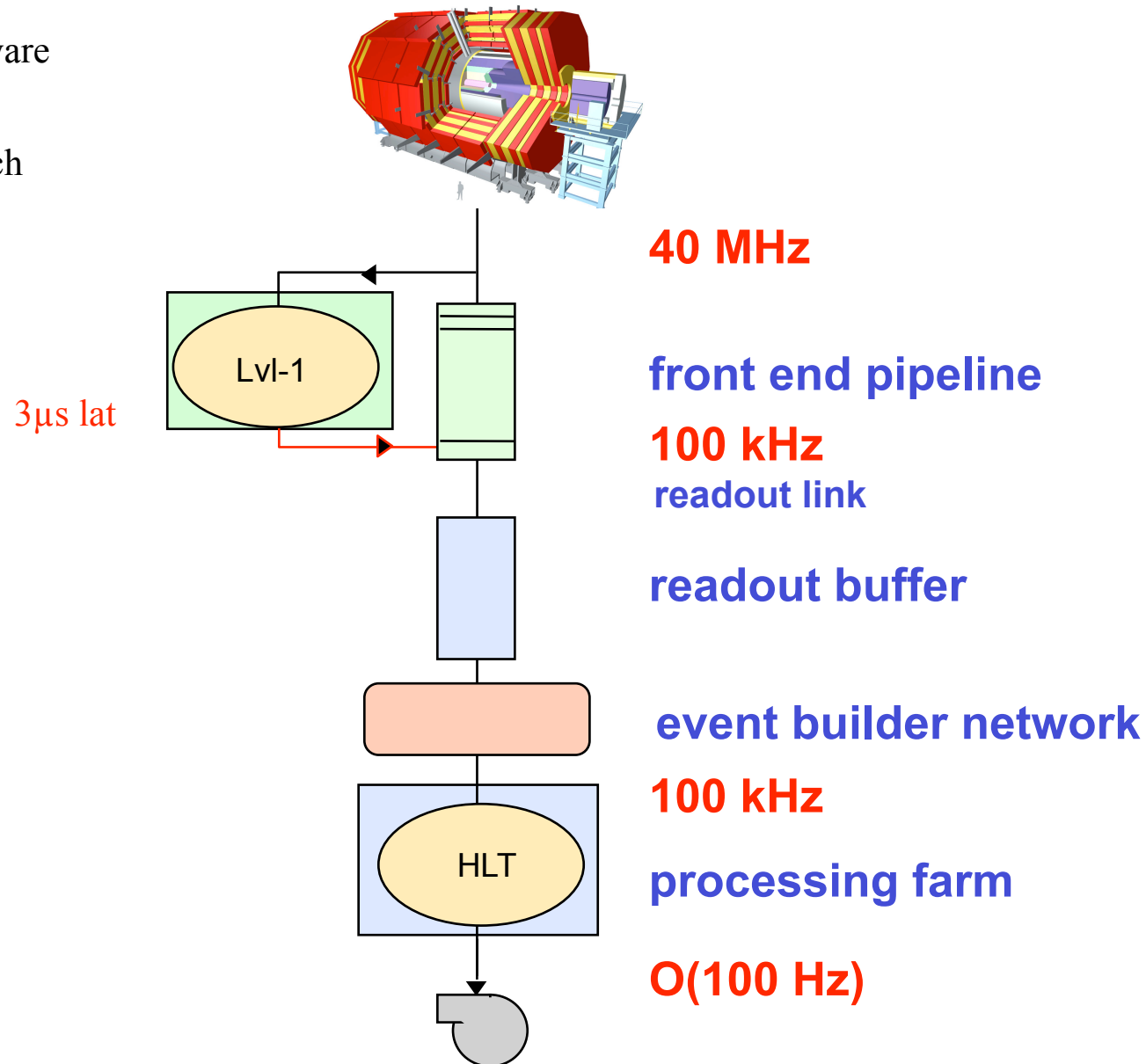
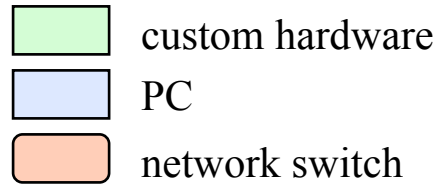
- custom hardware
- PC
- network switch



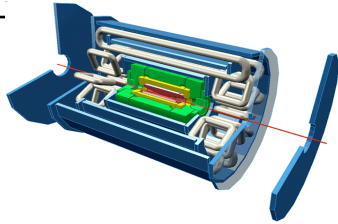
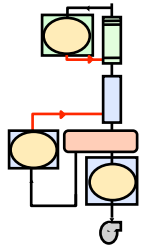
Region Of Interest (ROI):
Identified by Lvl1. Hint for Lvl2
to investigate further



Data Flow: CMS

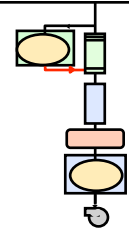
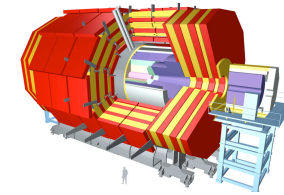


Data Flow: Atlas vs CMS



Challenging

Readout Buffer



“Commodity”

Concept of “Region Of Interest”
Increased complexity

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

Implemented with commercial
PCs

“Commodity”

Event Builder

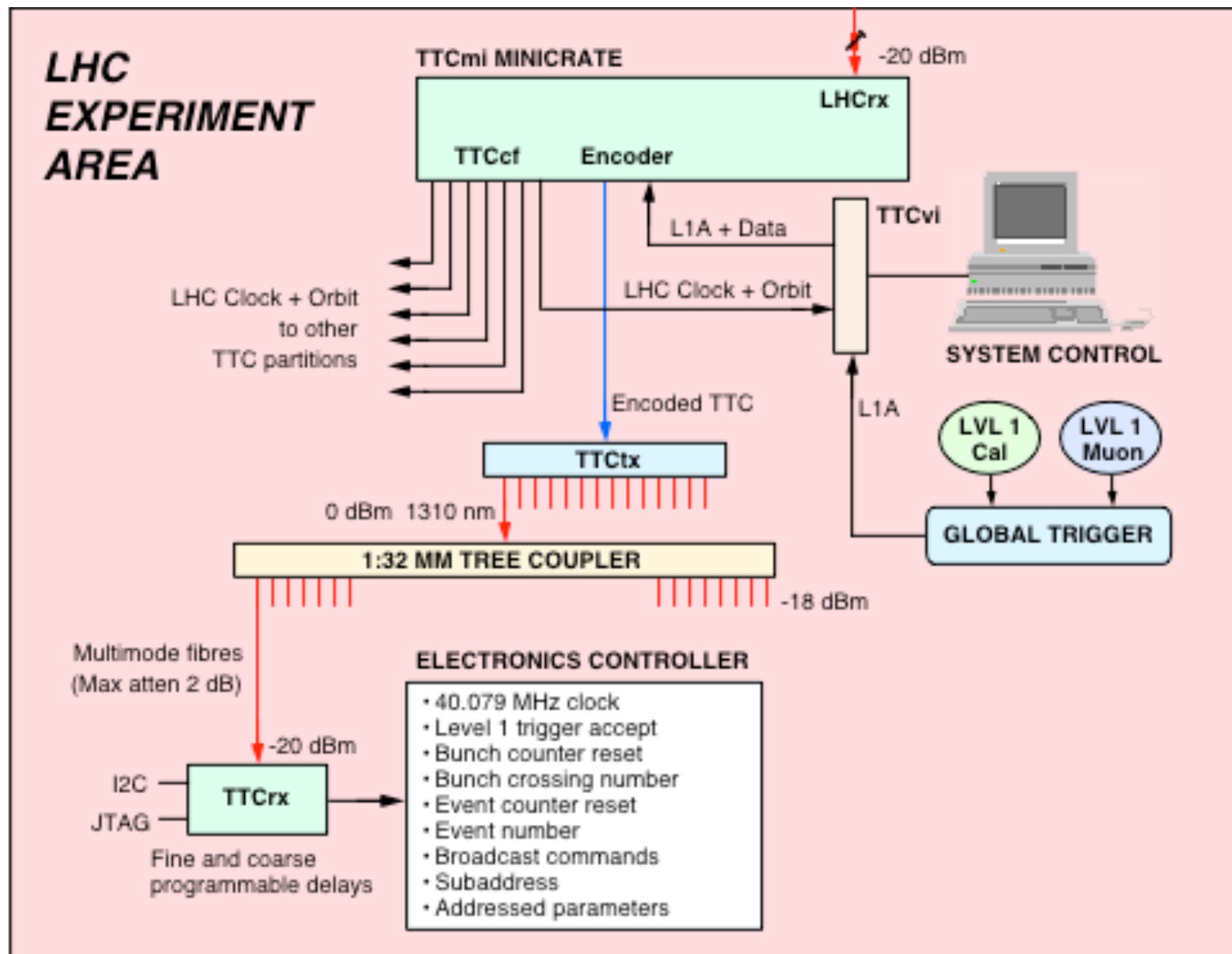
Challenging

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

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

Implementation TTC: Trigger Timing and Control System

Trigger distribution: TTC system



Shown is a single partition.

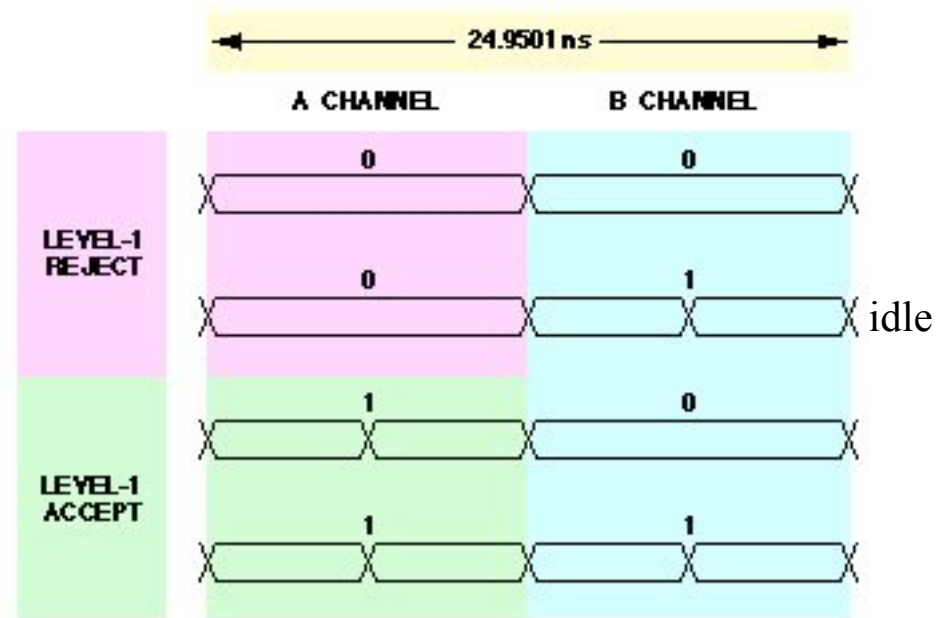
Every partition has it's own TTCvi with an optical tree.

TTC encoding: 2 Channels

- Channel A:
 - One bit every 25ns
 - low and constant latency
 - For distribution of LVI1 accept

- Channel B:

- One Bit every 25 ns
- **Synchronous** commands
 - Arrive in fixed relation to LHC Orbit signal
- **Asynchronous** commands
 - No guaranteed latency or time relation
- “**Short**” broadcast-commands (Bunch Counter Reset, LHC-Orbit)
- “**Long**” commands with addressing scheme
 - Addressing scheme
 - Can be used for calibration, re-synch, ...



TTC Designed in the early '90s

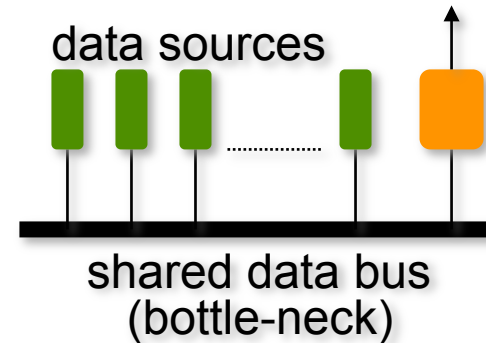
Implementation: Readout Links

Data Flow: Data Readout

- **Readout here:** Transfer of data from sub-detector dependent readout modules into DAQ system

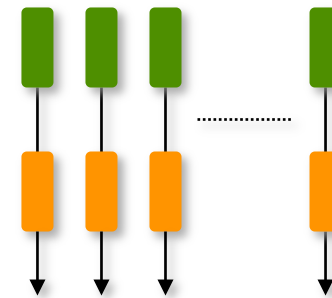
- **LEP: Use of bus-systems**

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



- **LHC: Point to point links**

- All sources can send simultaneously
- Optical or electrical
- Data serialized
- Custom or standard protocols
- Related to powerful network technology (see event-builder section)



- **Compare industrial market:**

- 198x: ISA, SCSI(1979), IDE, parallel port, VME(1982)
- 199x: PCI (1990, 66MHz 1995), USB(1996), FireWire(1995)
- 200x: USB2, FireWire 800, PCIeexpress, Infiniband, GbE, 10GbE

Readout Links: Interface to PC

Problem:

Read data in PC with high bandwidth and low CPU load

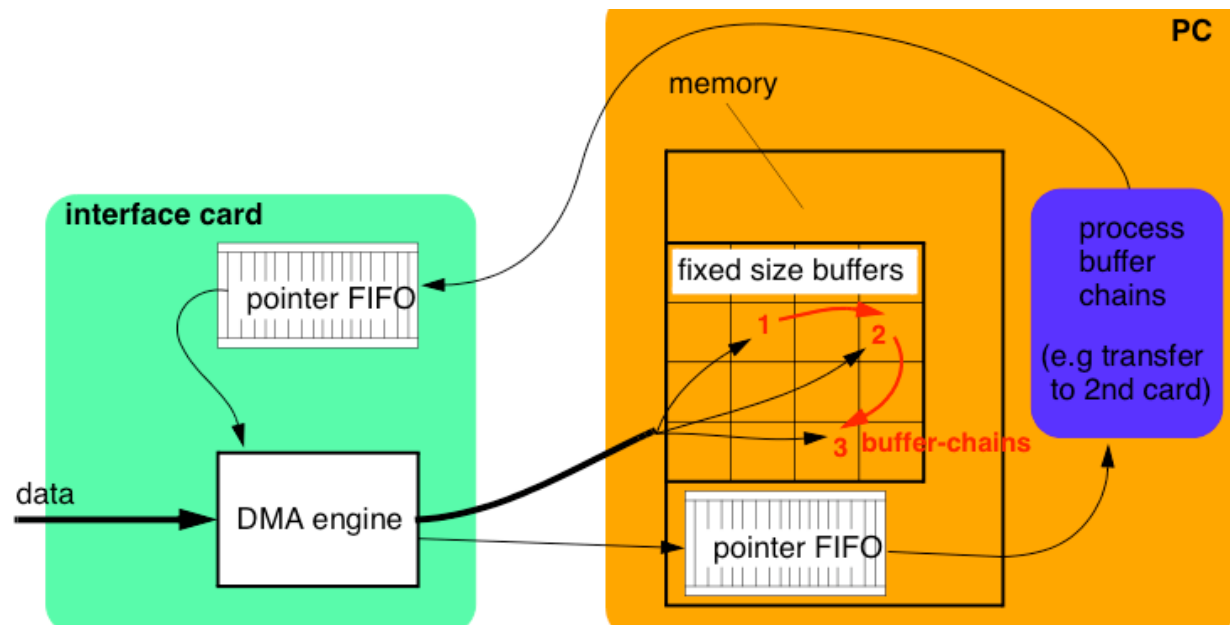
Solution: Buffer-Loaning

- Hardware shuffles data via DMA engines
- Software maintains tables of buffer-chains

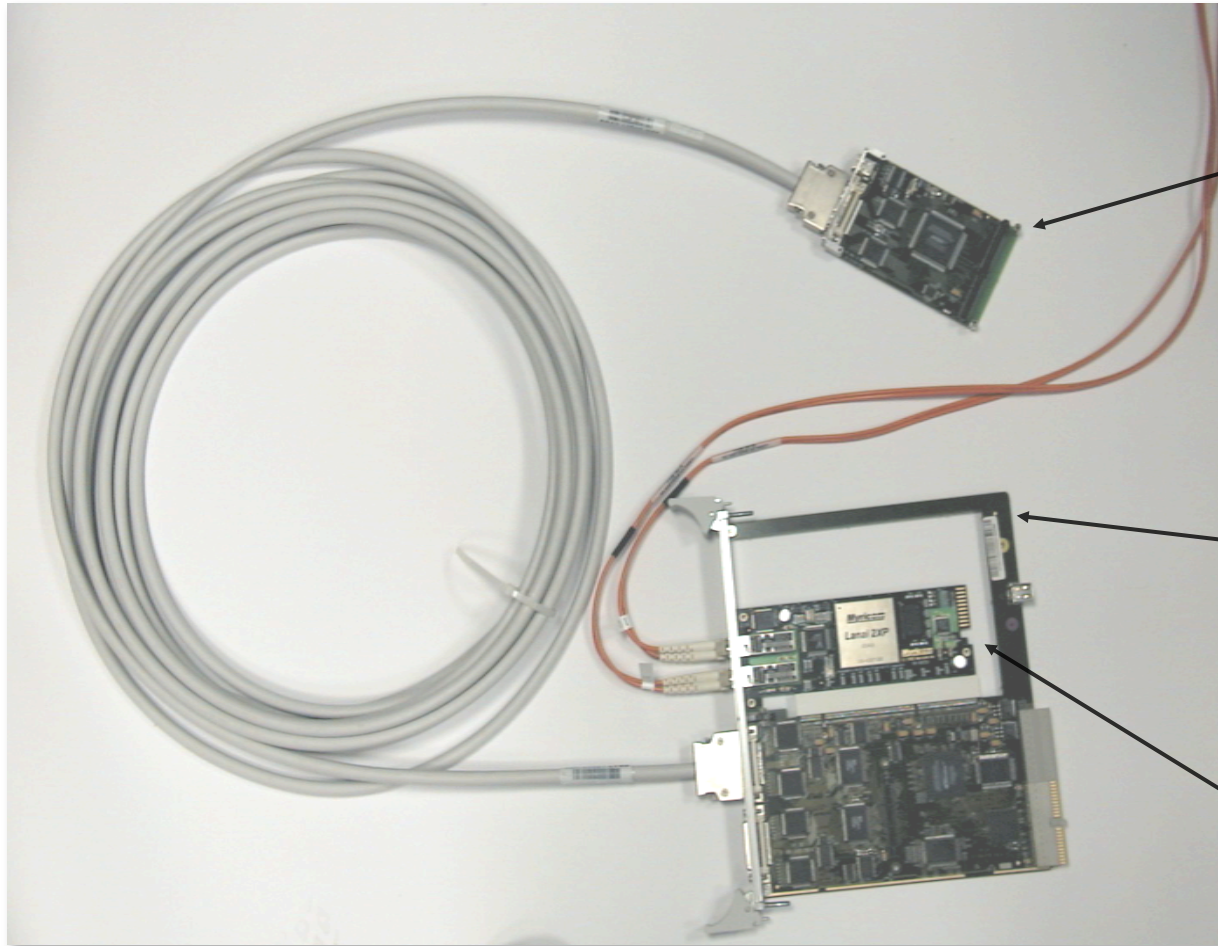
Advantage:

- No CPU copy involved

used for links of
Atlas, CMS, ALICE
(LHCb uses Gb Ethernet.)



CMS: The SLINK-64 / FRL



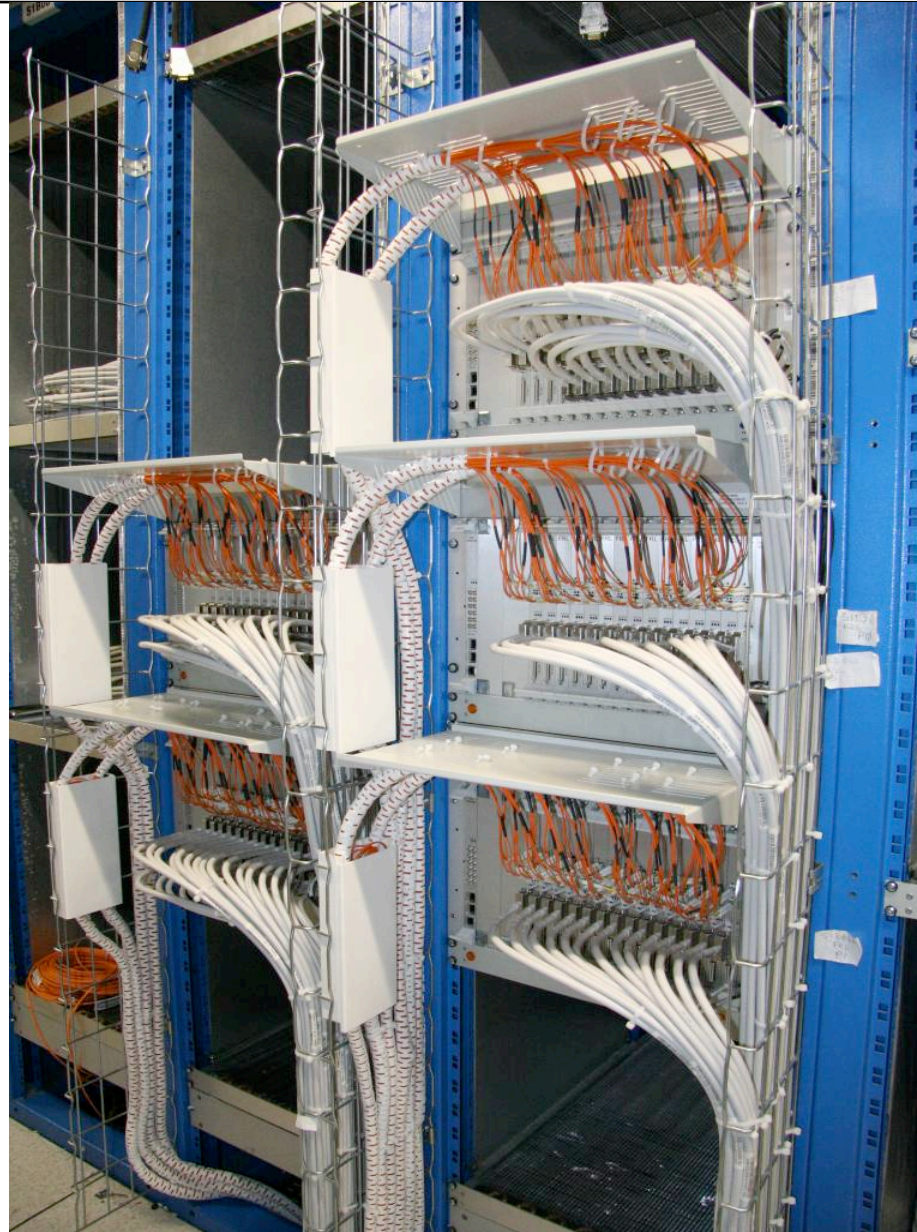
Sender Card:

- **Fifo like interface** to Detector card
- LVDS link up to 11m

Receiver Card: FRL

- Compact PCI
- 2 input channels (can merge 2 incoming data blocks)
- interface to commercial NIC (internal PCI-X bus): Myrinet with 2 x 2.5 Gb/s optical link

SLINK-64 Receivers (FRLs) in place



Experience with the SLINK 64

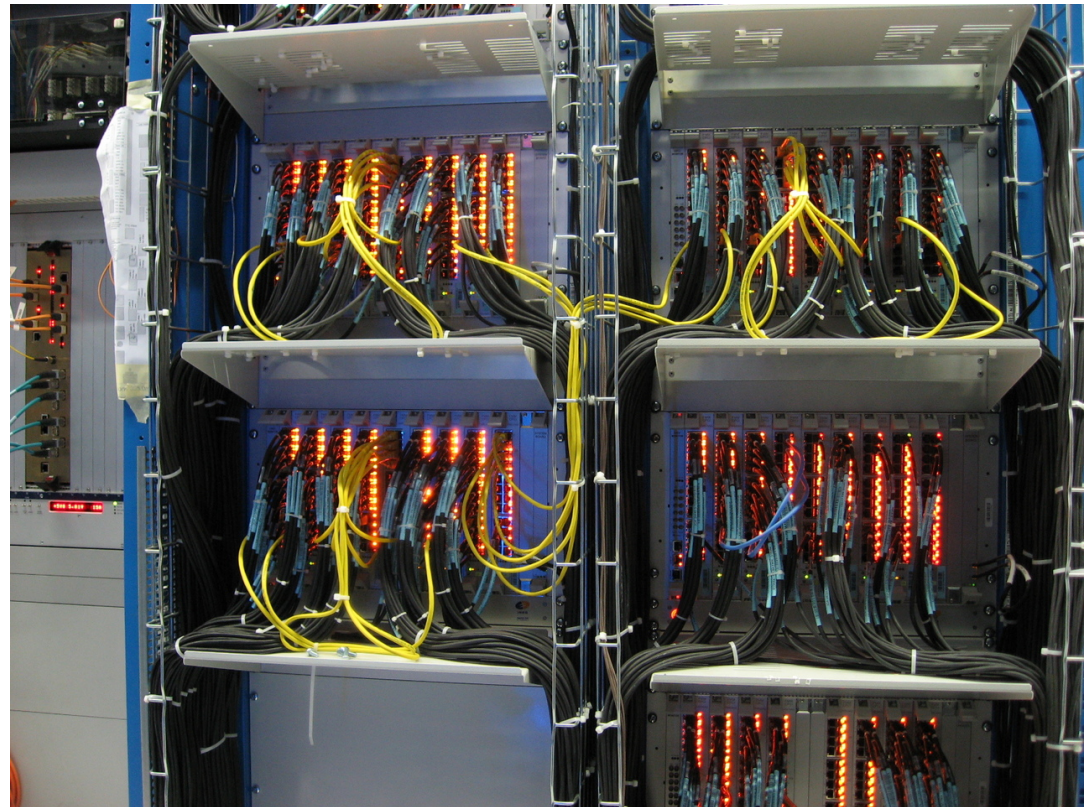
- Series production operational since 2005
 - SLINK implementation has proven to work rock-solid
 - Works at 400 MB/s (tested successfully up to 800 MB/s in laboratory)
 - Identified 3 links (out of 600) with rare CRC errors. Exchanged hardware.
 - No component failures so far.
 - Important feature during commissioning:
 - Double checking of CRC allows to identify unambiguously source of problem (FED or Link components):
 - Once data arrives at SLINK sender CMC
 - Once data arrives in SLINK receiver
 - Simple SLINK protocol has proven to be helpful for Sub-detector developers

Future of Readout Links *

- **Optical Link has clearly advantages**
 - Longer distance achievable
 - Electrical cables are bulky and difficult to handle
 - Grounding less critical
 - LVDS is delicate (1V common mode only)
 - Needs very good cables
- **Standard protocol desirable (Ethernet, TCP/IP)**
 - Direct interface into commercial event building network
 - See LHCB !
 - At the time the SLINK was developed it was not obvious how to implement this at the given requirements.

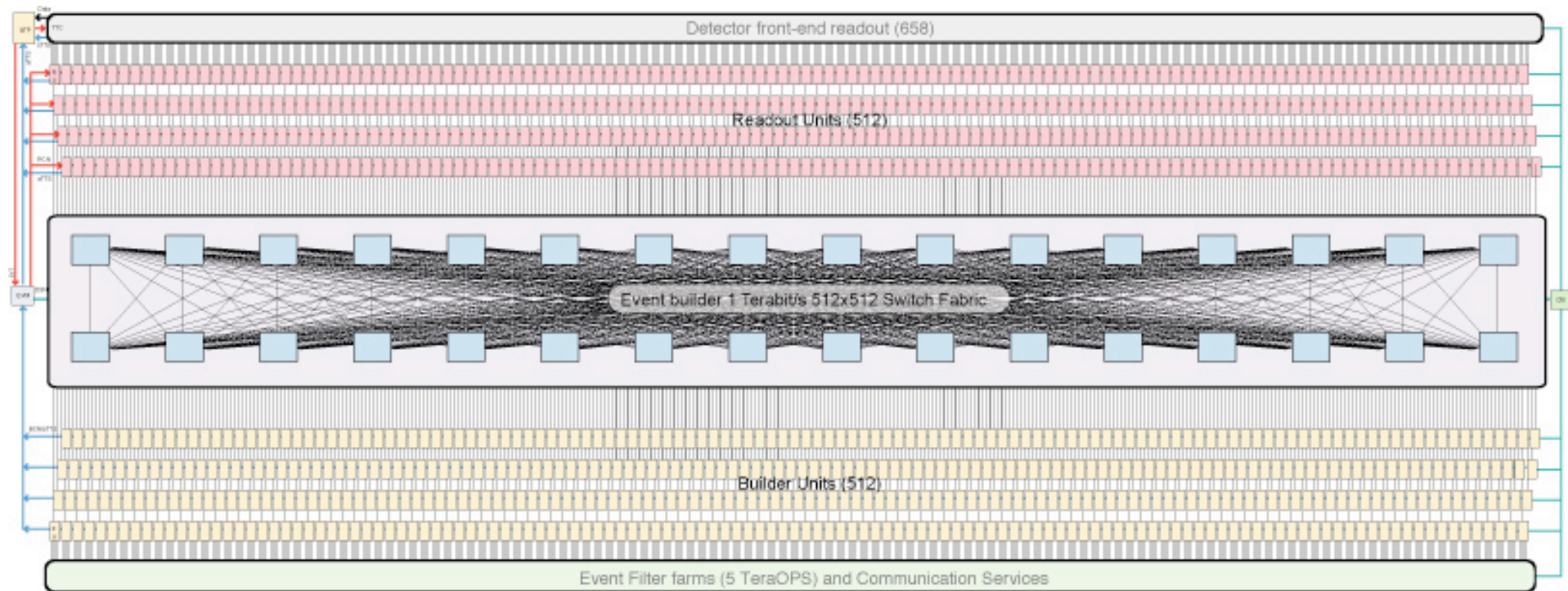
Backpressure: the sTTS system

- sTTS: synchronous Trigger Throttling System
 - If incoming data rate $>$ DAQ capacity:
Need to throttle trigger in order to avoid data corruption in Front End.
 - Fast messaging system
FrontEnd -> Lvl1 Trigger
based on 4 LVDS lines.
 - Encodes various signals:
 - Ready,
 - Almost full,
 - Full,
 - Error,
 - Synch-error
 - Disconnected.
 - Merger Modules 'OR' the signals for each partition.



Implementation: Event Building in CMS

Event Building in CMS

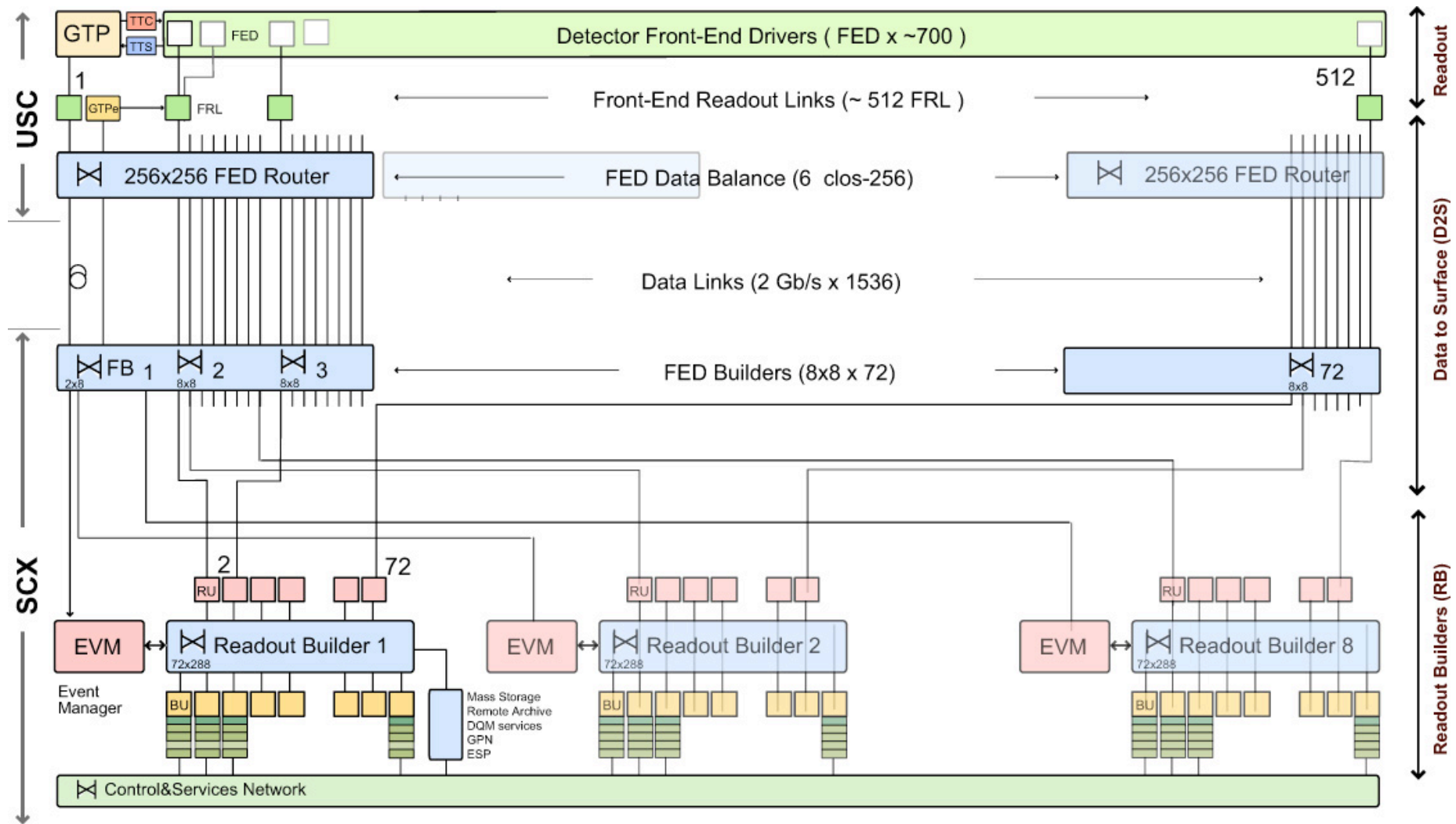


Level-1 maximum trigger rate	100 kHz	No. Readout systems	≈ 512
Average event size	1 Mbyte	No. Filter Subfarms	≈ 512 x n
Builder network	1 Terabit/s	No. (C&D) network ports	≈ 10000
Event filter computing power	5 10⁶ MIPS	No. programmable units	≈ 10000
Event flow control	≈ 10 ⁶ Mssg/s	System dead time	≈ %

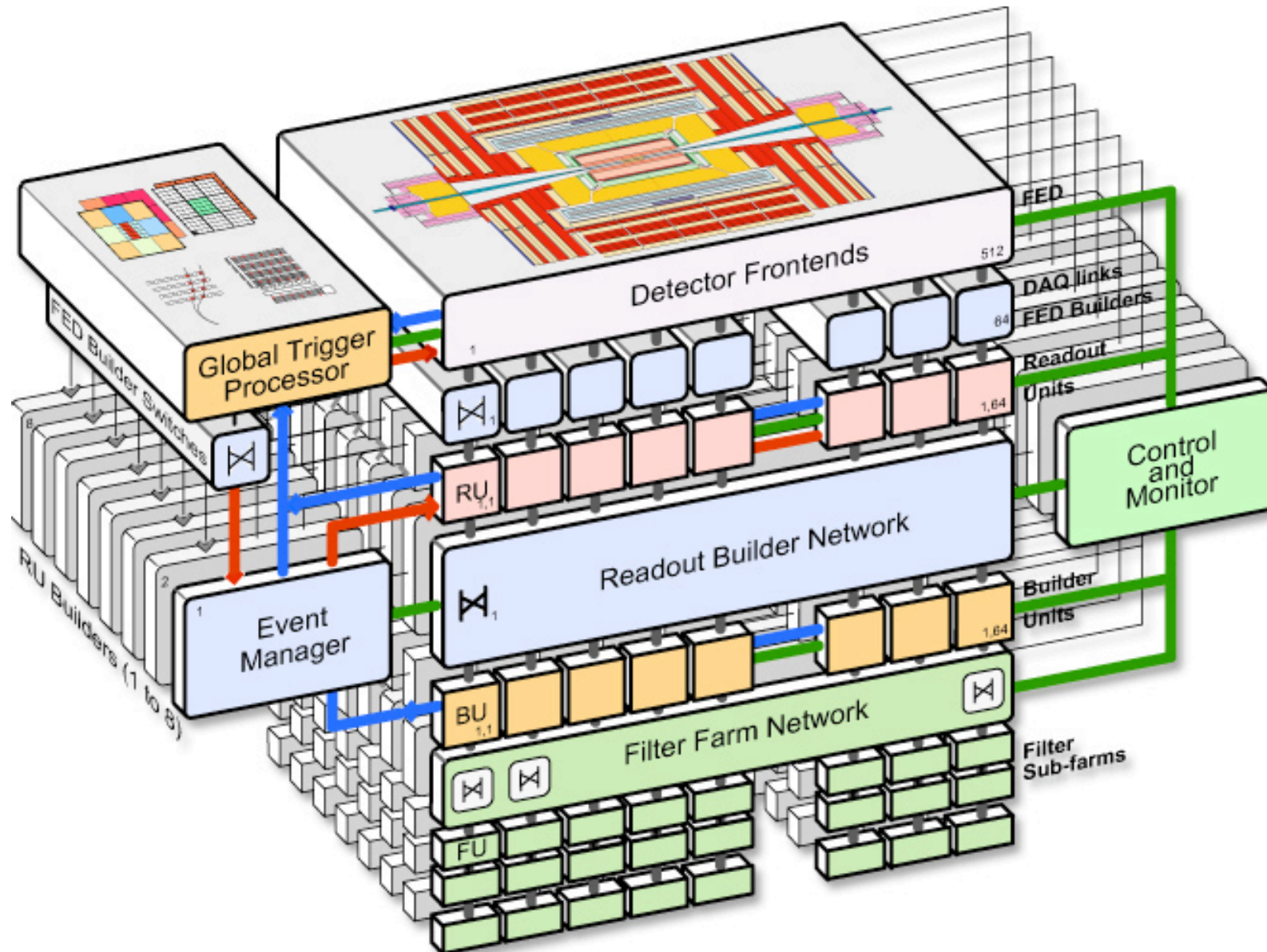
Achronyms

- TPO Trigger Primitive Generator
- RTP Regional Trigger Processor
- L1L1 Level-1 Trigger Processor
- GTP Global Trigger Processor
- TTC Timing, Trigger and Control
- eTTS experimental Trigger Threshold System
- eTTS experimental Trigger Threshold System
- FES FrontEnd System
- FSD FrontEnd Driver
- FEC FrontEnd Controller
- DOS Data In Subsystem
- FSL FrontEnd Subsystem Link
- BU Builder Unit
- FS Filter Subfarm
- EVM Event Manager
- FMA FrontEnd Manager
- BM Builder Manager
- EVB Event Builder
- FCN FrontEnd Control Network
- BCN Builder Control Network
- CSN Computing Service Network
- DCN Detector Control Network
- DSN Data Service Network
- DCS Detector Control System
- FCS Farm Control System

EVB CMS: 2 stages



CMS: 3D - EVB

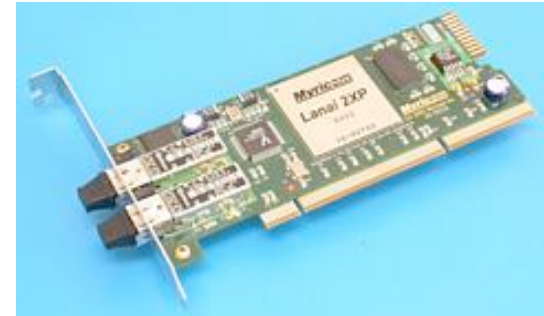


Advantages of 2 stage EVB

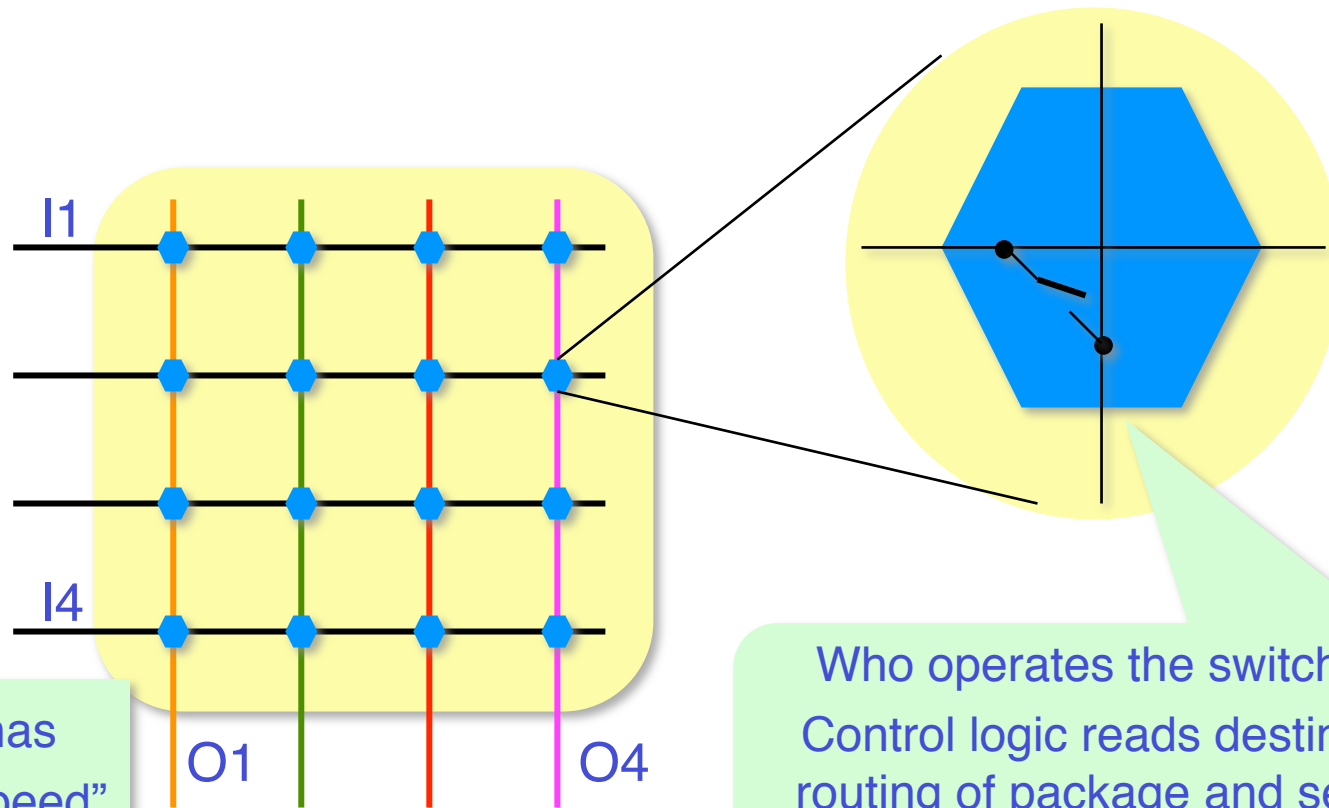
- **Relaxed requirements:**
 - Every RU-Builder works at 12.5 kHz (instead of 100kHz)
- **Staging**
 - To start up the experiment not the entire hardware needs to be present. Example:
 - If an Event Builder operating at 50 kHz is sufficient for the first beam, only 4 RU-builders need to be bought and 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.

Stage1: FED-Builder

- FED Builder implementation
 - Boundary conditions:
 - Sustained throughput of 200MB/s for every input.
 - Input interfaces to FPGA --> limits protocol complexity.
 - Chosen network technology: Myrinet
 - NICs with 2 x 2.0 Gb/s optical links
 - Full duplex with flow control (no packet loss).
 - NIC cards contain RISC processor. Development system available.
 - Custom protocol has been developed (buffer loaning scheme at input and output side).
 - Switches based on cross bars



Switch implementation: crossbar

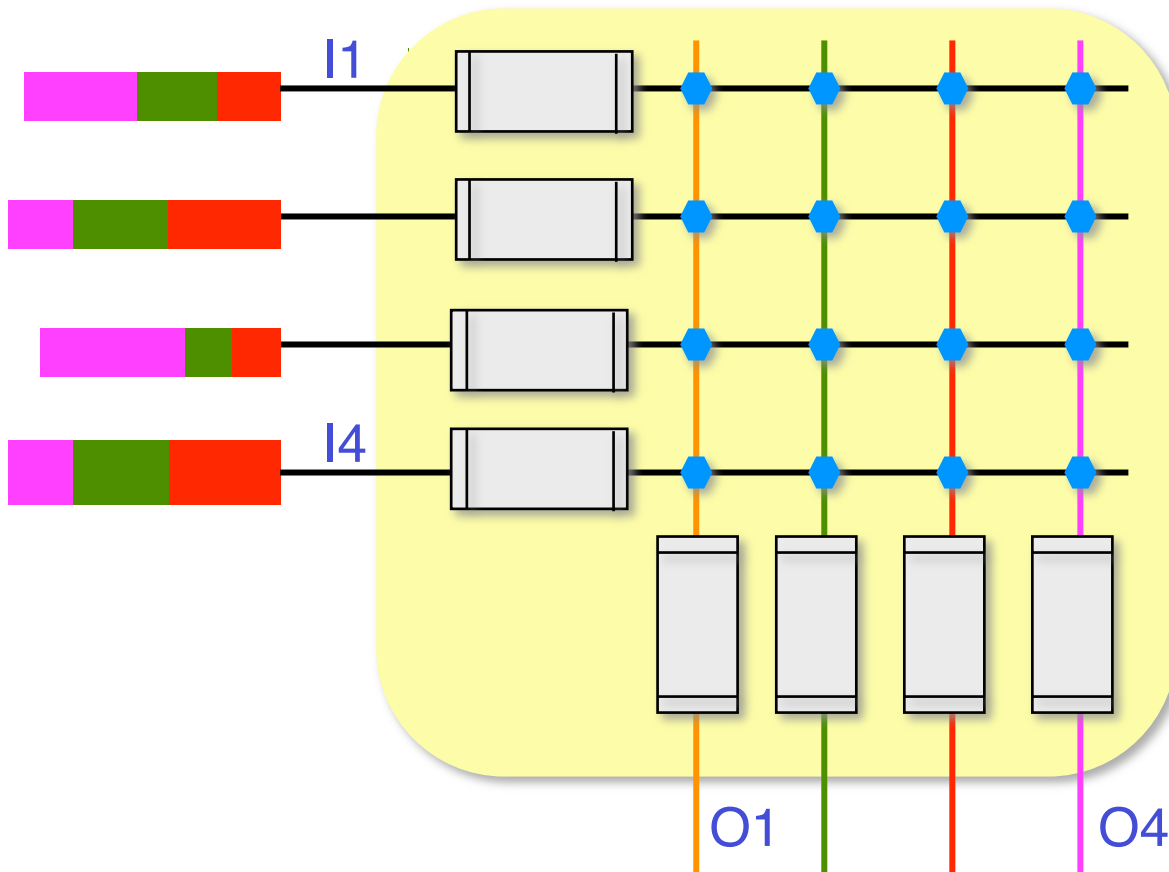


Every input / output has
A given max. “wire-speed”
(e.g. 2.0 Gbits/sec).
Internal connections are
much faster!

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

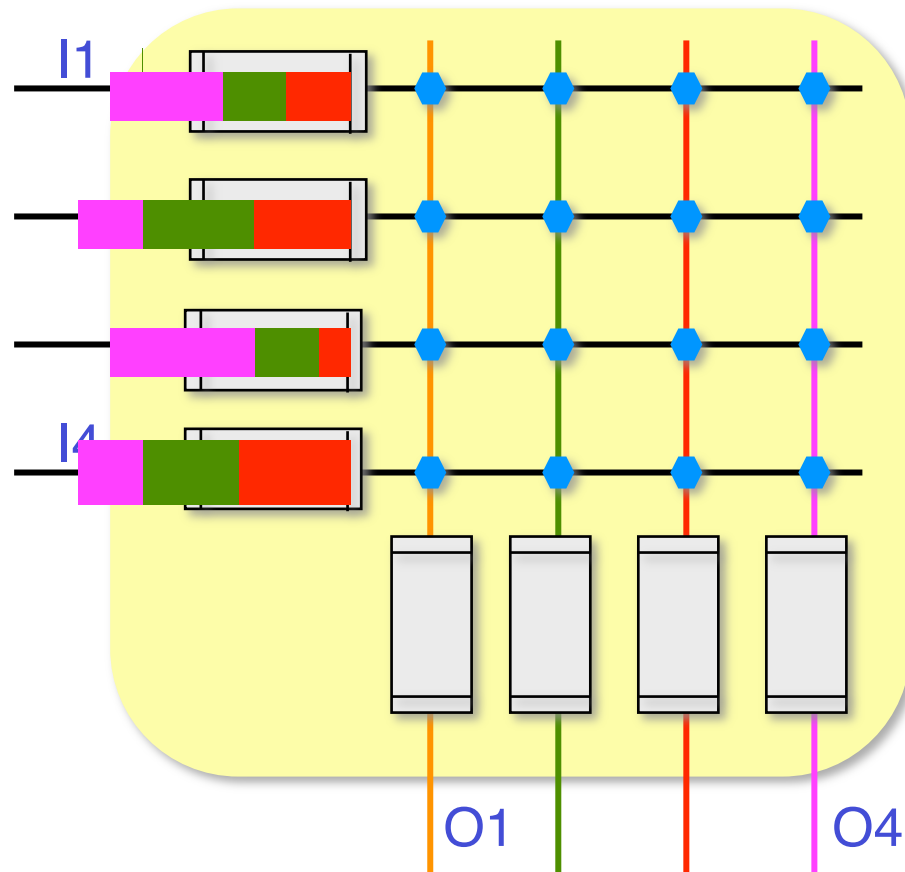
Switch implementation

Ideal situation: continuously send data into switch with wire speed on all inputs at any time (100% switch utilization)

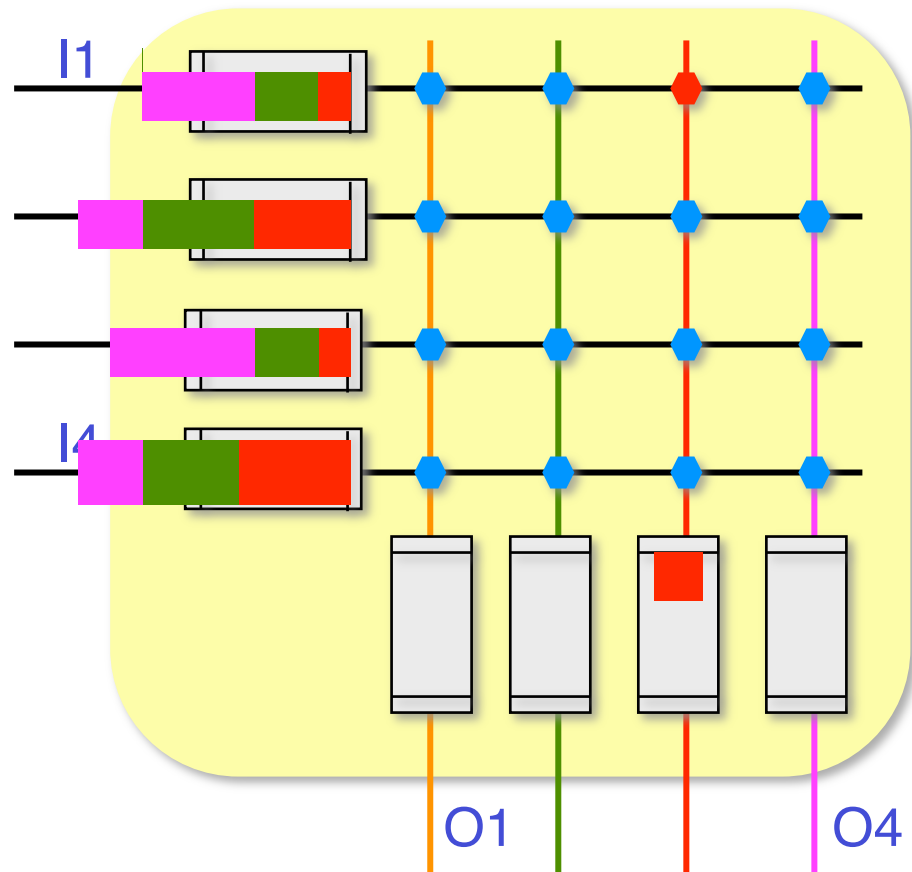


Fifos can “absorb” congestion
...until they are full.

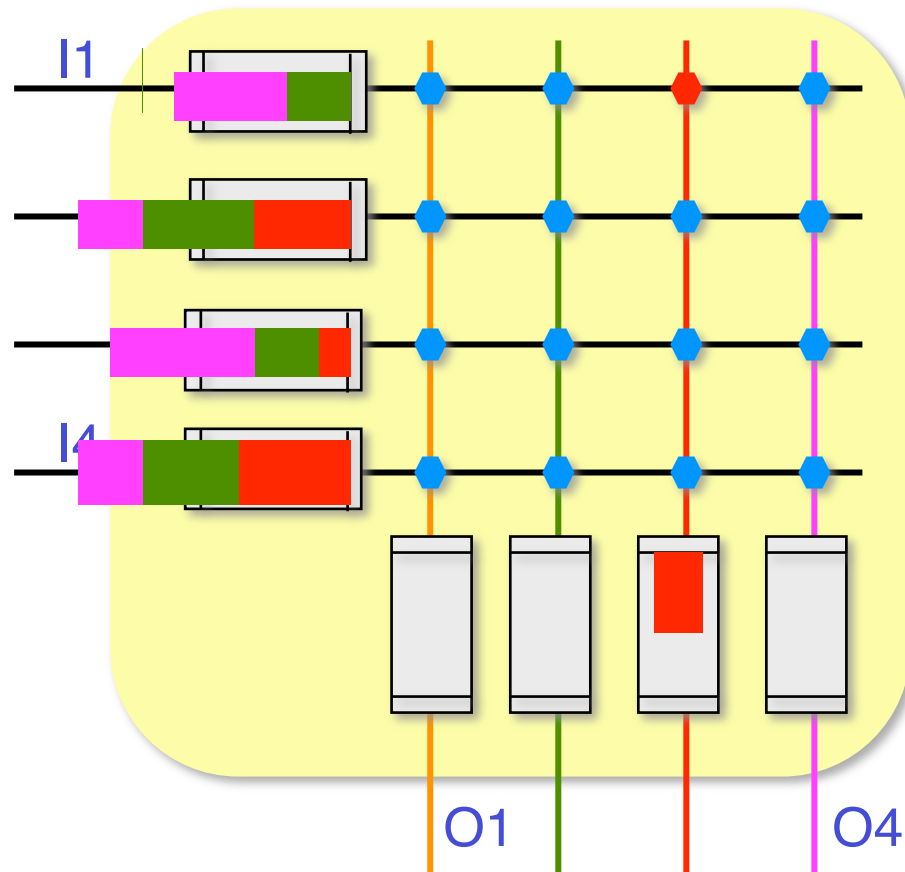
Switch implementation



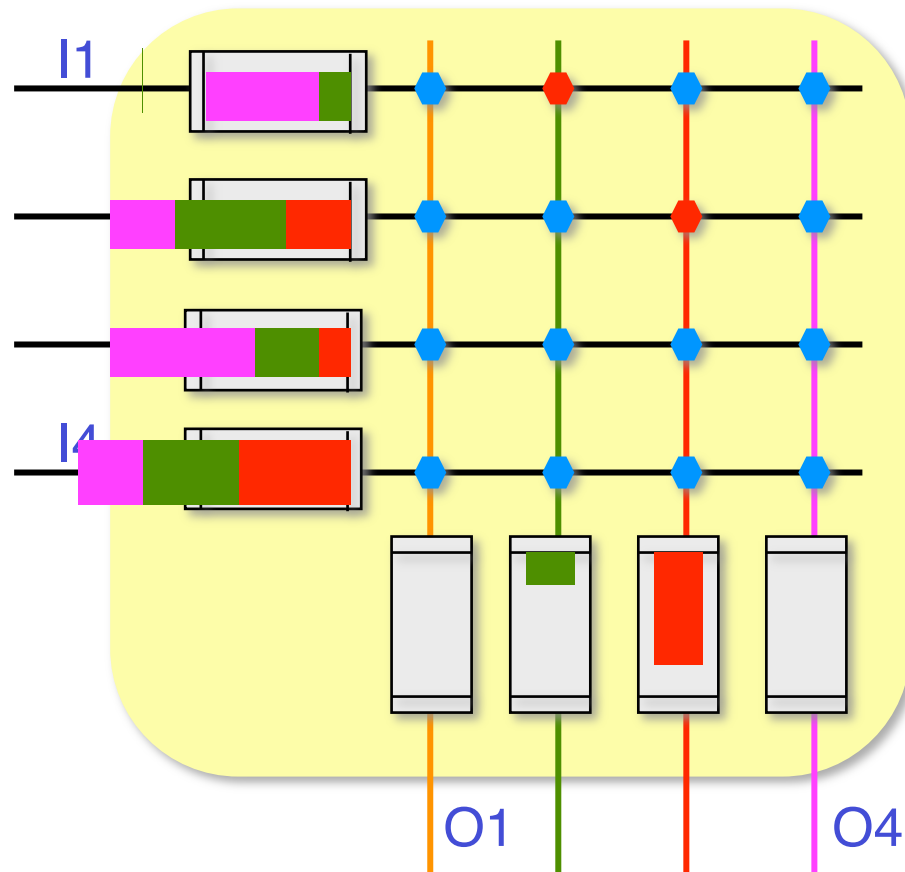
Switch implementation



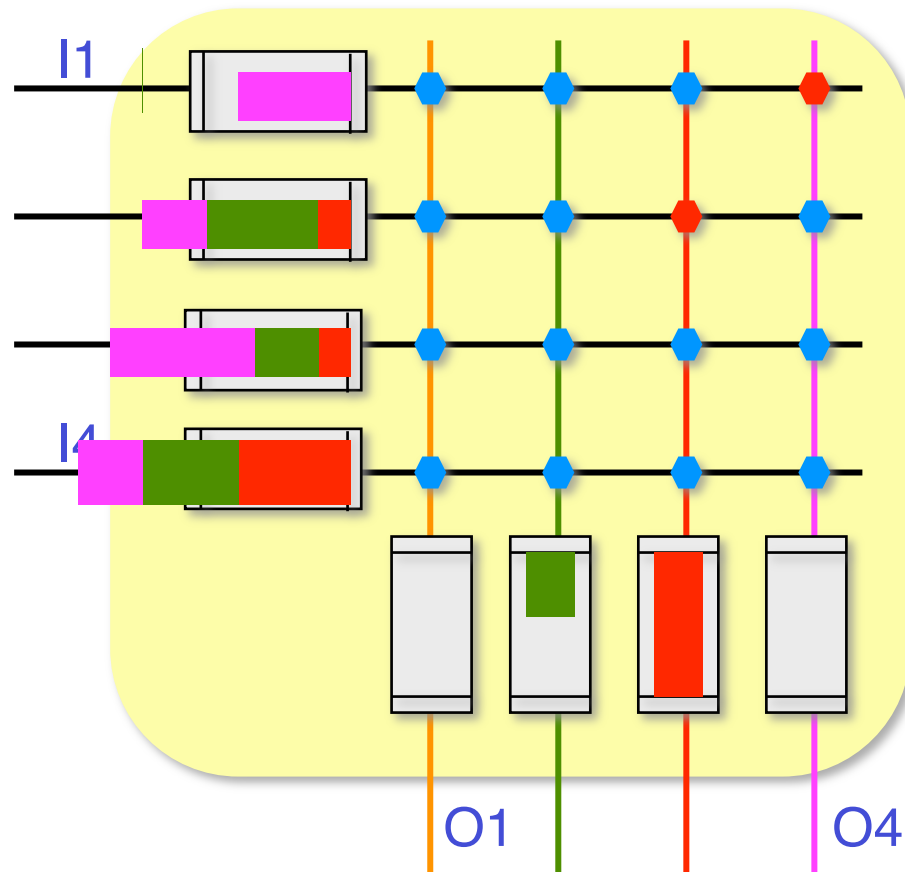
Switch implementation



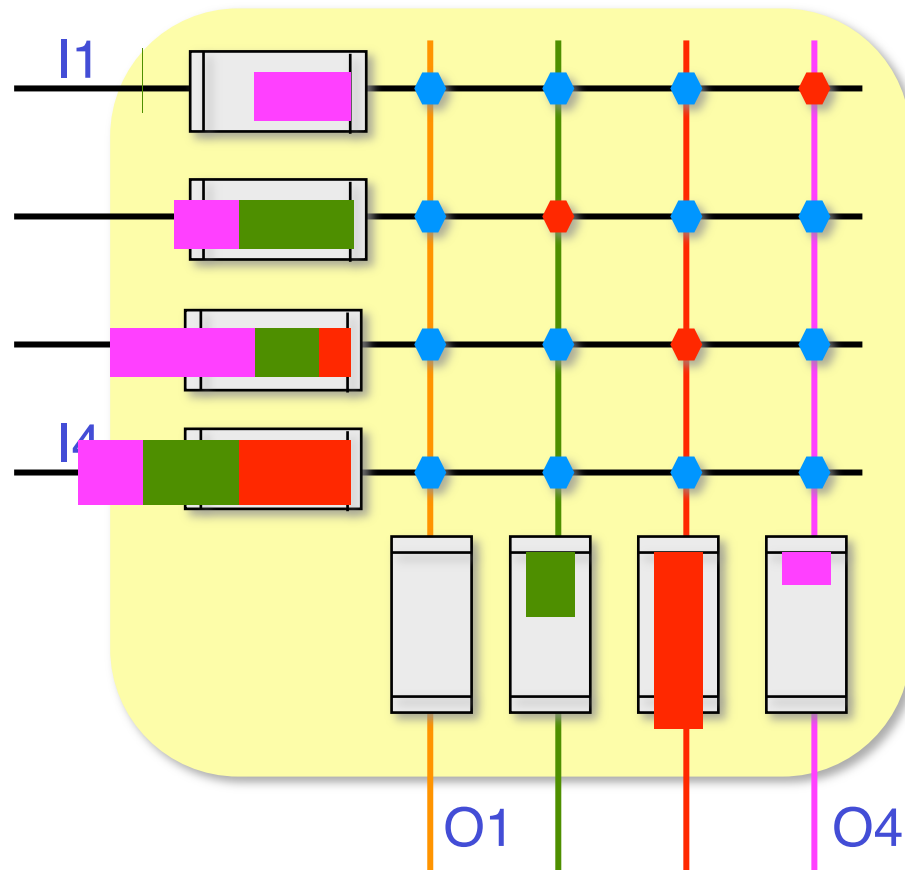
Switch implementation



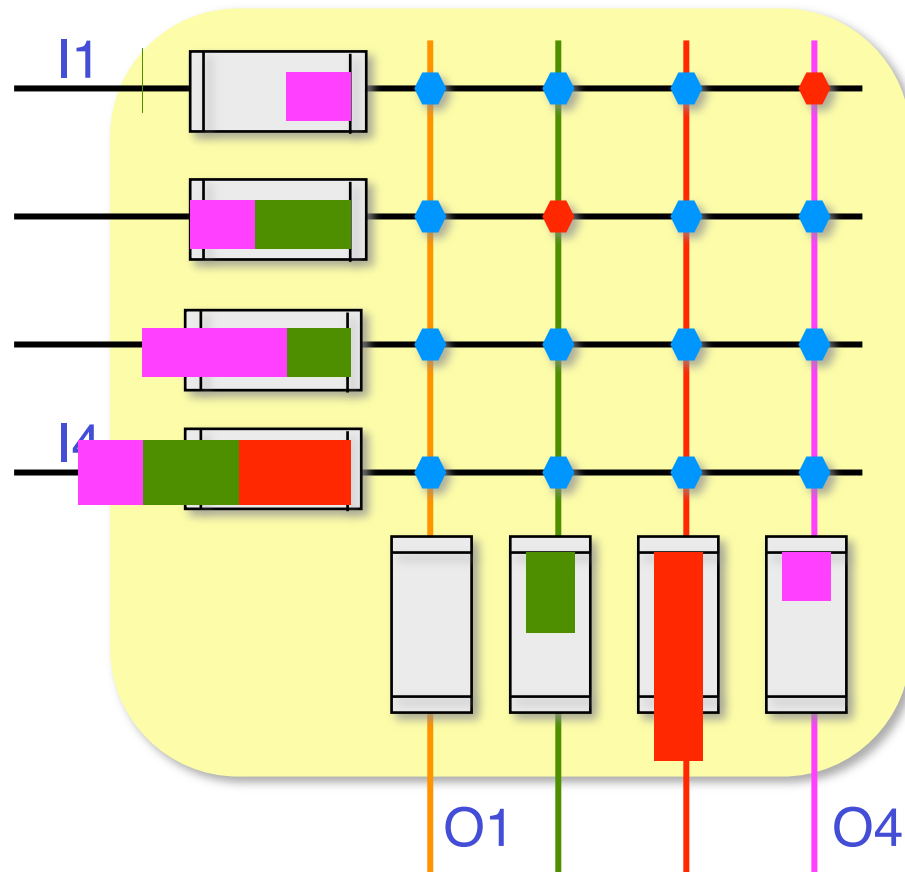
Switch implementation



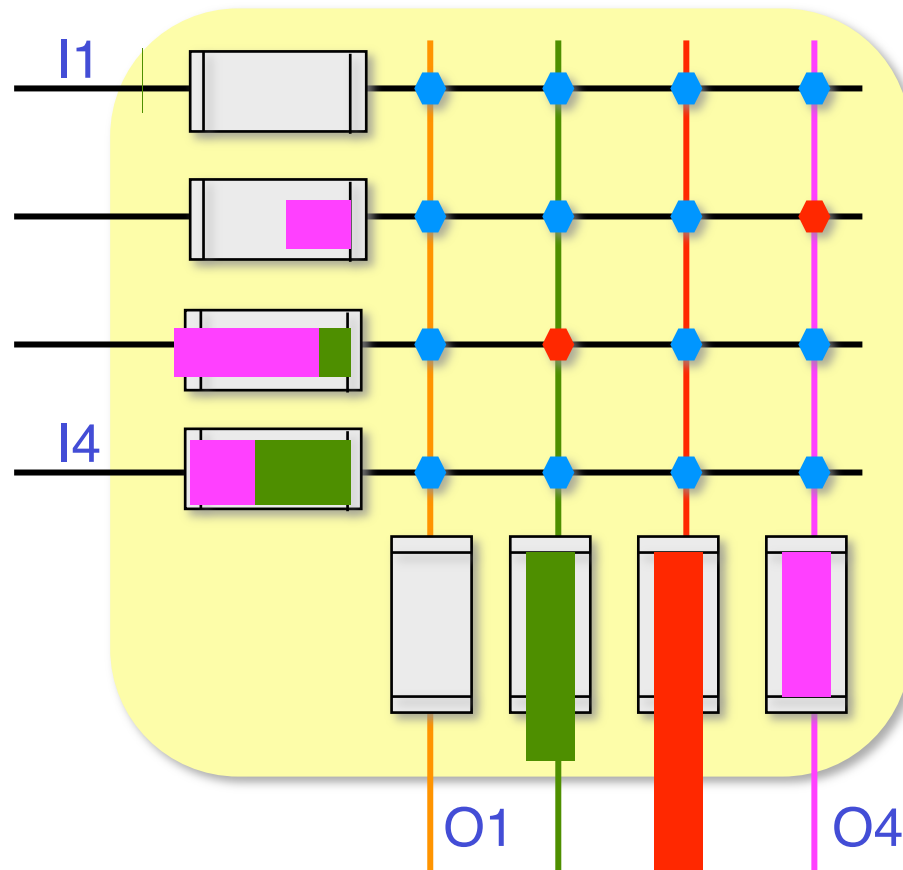
Switch implementation



Switch implementation



Switch implementation



Problematic:

Input Fifos can absorb data fluctuations until they are full. All fine if:

Fifos capacity $> n \cdot \text{max event size}$
 n : no of inputs

In practice: sizes of FIFOs are much smaller!

EVB traffic: switch will partially block

Event Building dilemma

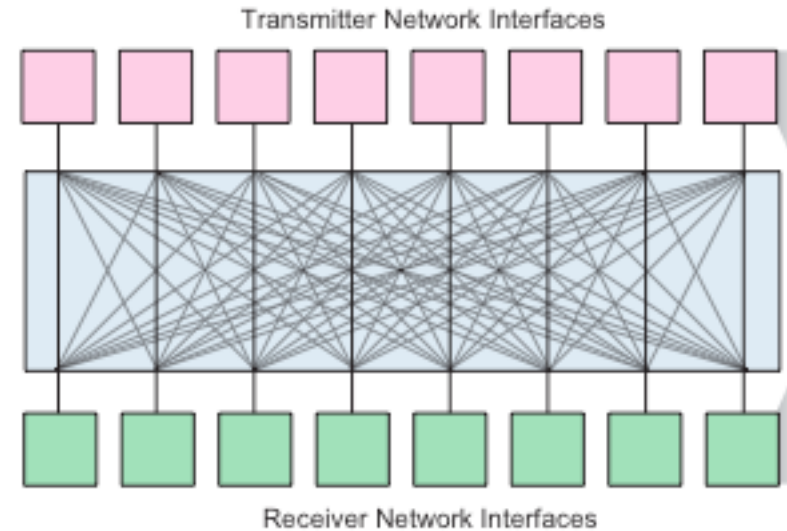
To be avoided:



In spite of the Event builder traffic pattern congestion should be avoided: Either do traffic shaping, or over-dimension the system

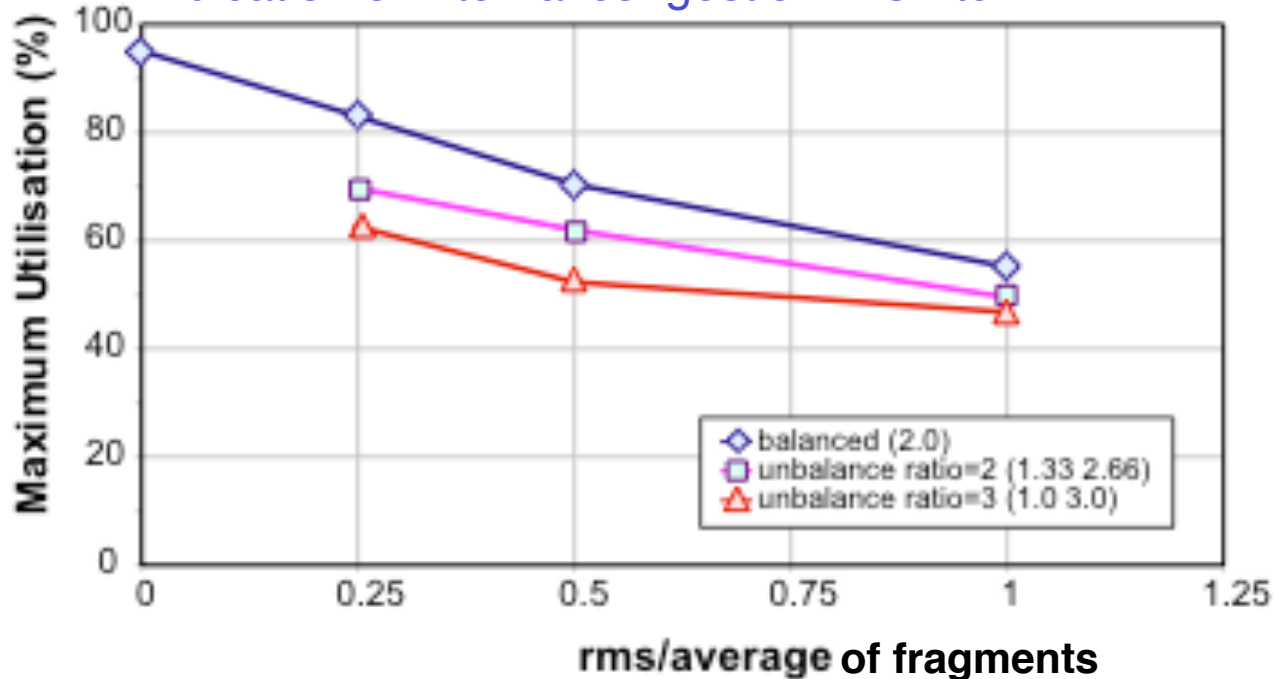
Performance of “1 rail” FEDBuilder

Measurement configuration:
8 sources to 8 destinations



% of wire-speed

Indication of internal congestion in switch:



Measured switch utilization:

Blue: all inputs 2 kB avg

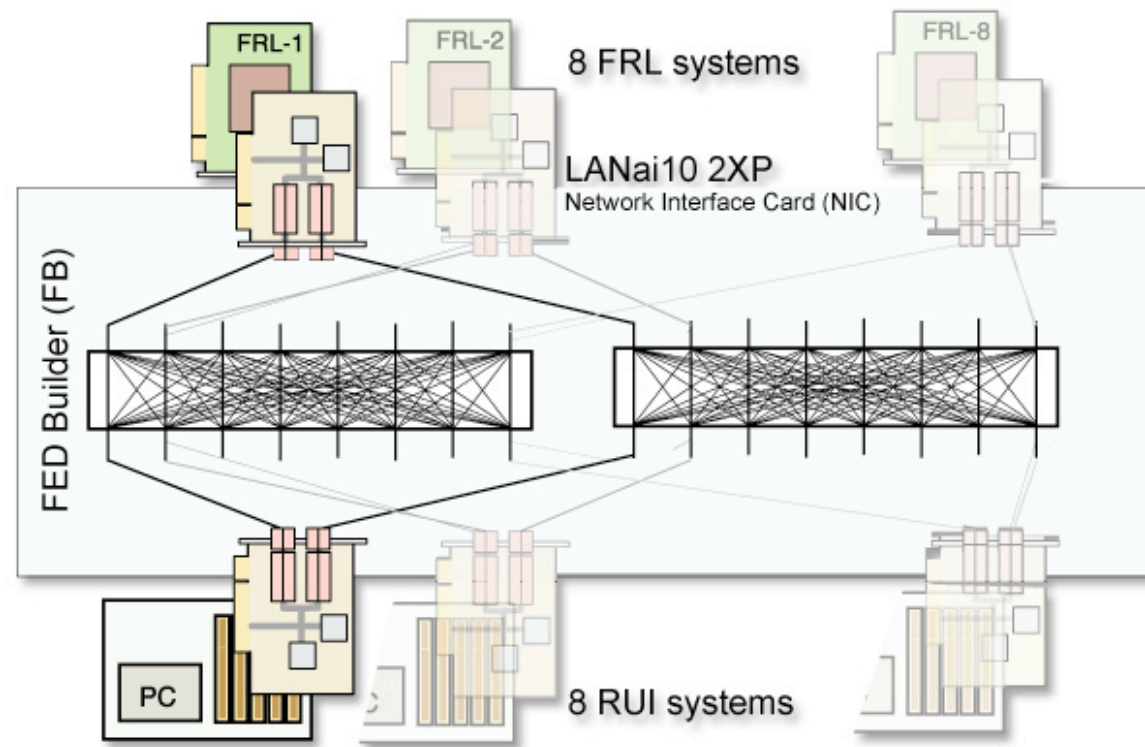
Magenta: 4 x 1.33 kB
4 x 2.66 kB

Red: 4 x 1 kB
4 x 3 kB

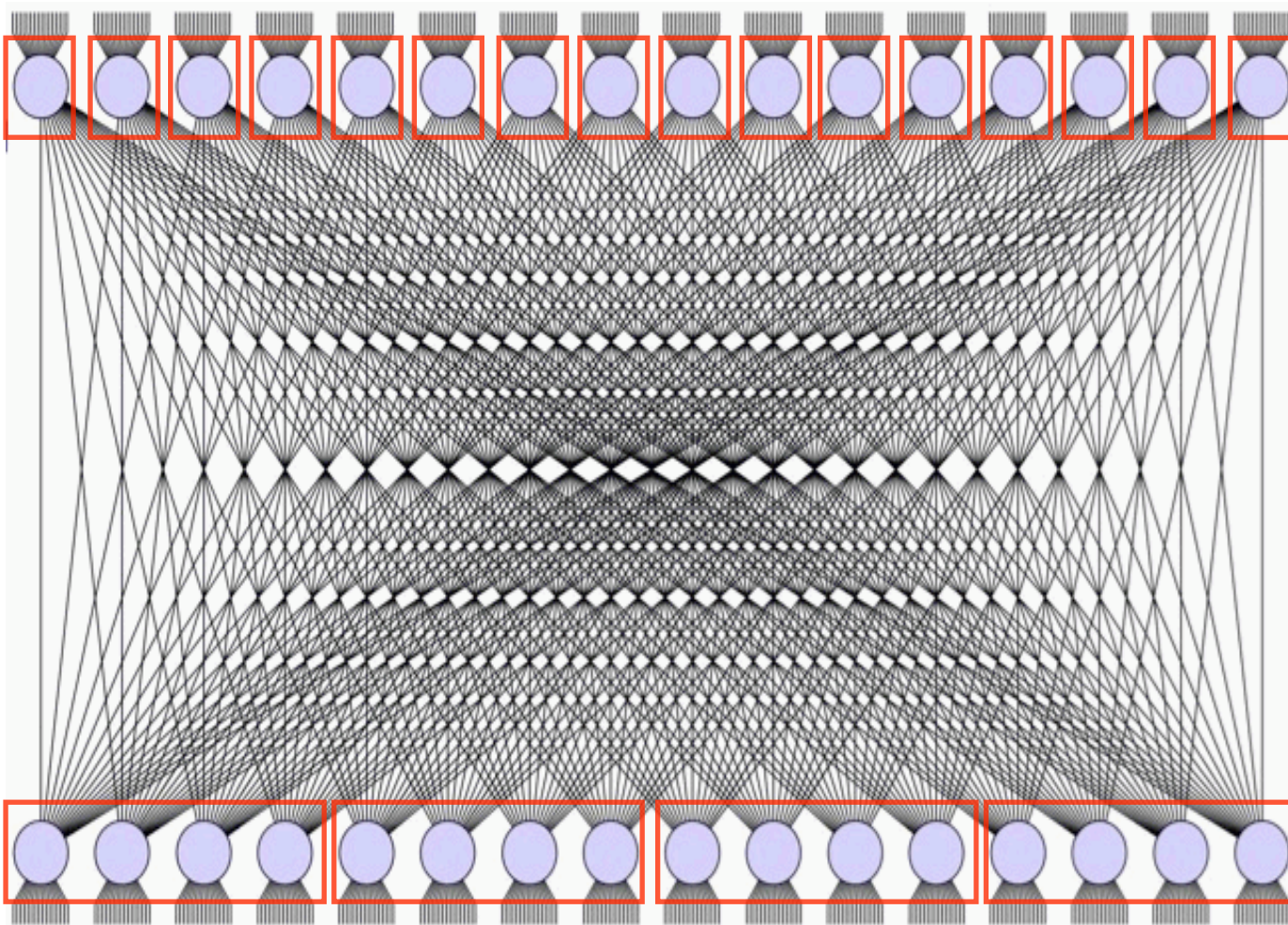
≈ 50 %

FED-Builder: 2 switches in parallel

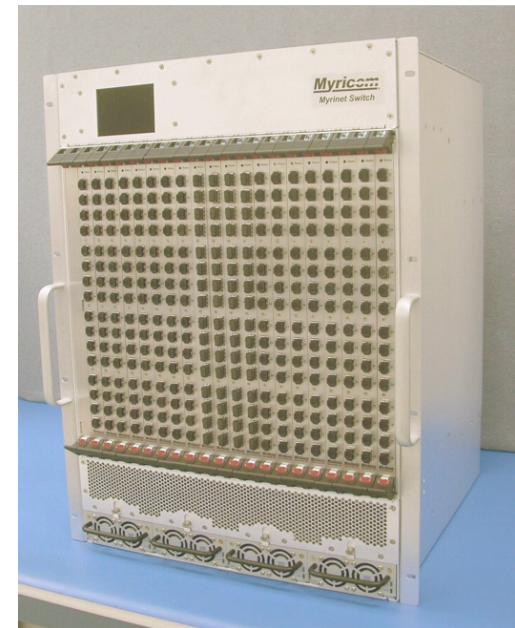
- Almost no extra software/firmware development
- Easy to maintain
- Adds redundancy to the system



Inside the Myrinet Switch



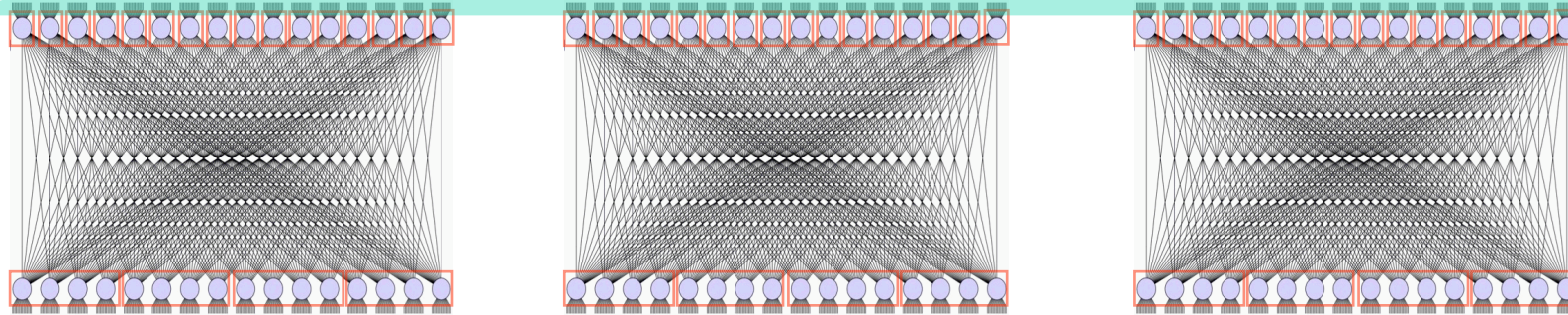
16 linecards with each 16 inputs and 1 XBar with 16 bi-directional ports to the Front Panel and 16 ports to the backplane



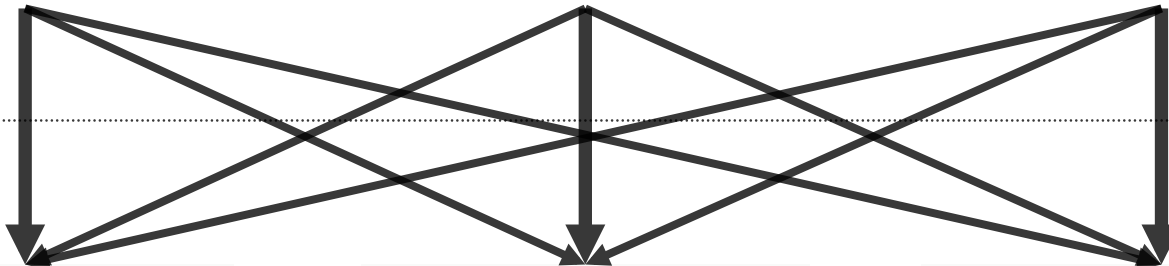
4 "Quad" linecards with 4 Xbars

1/2 FED-Builder

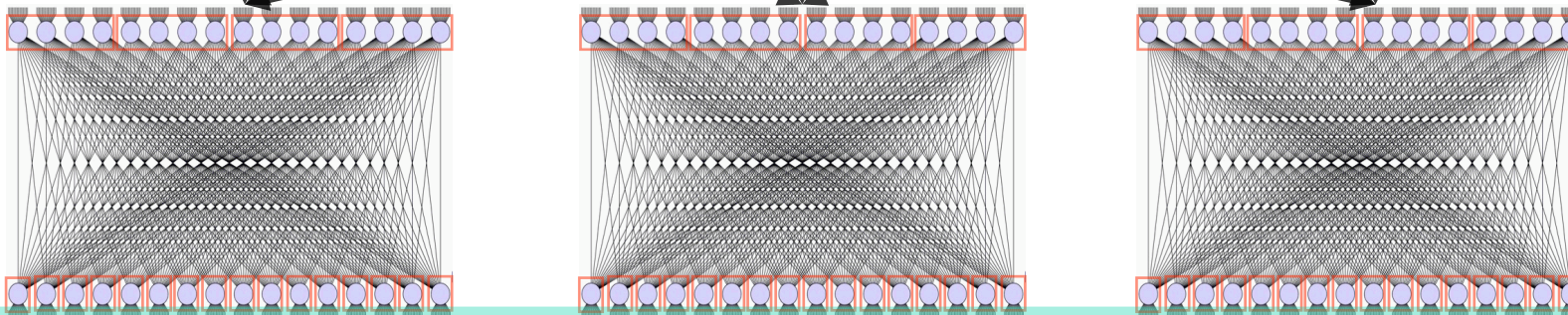
Readout Links



underground
surface



Cross connections allow flexible routing algorithms: load balancing, hardware fault tolerance, ...



2nd stage Builder

Event Builder Components

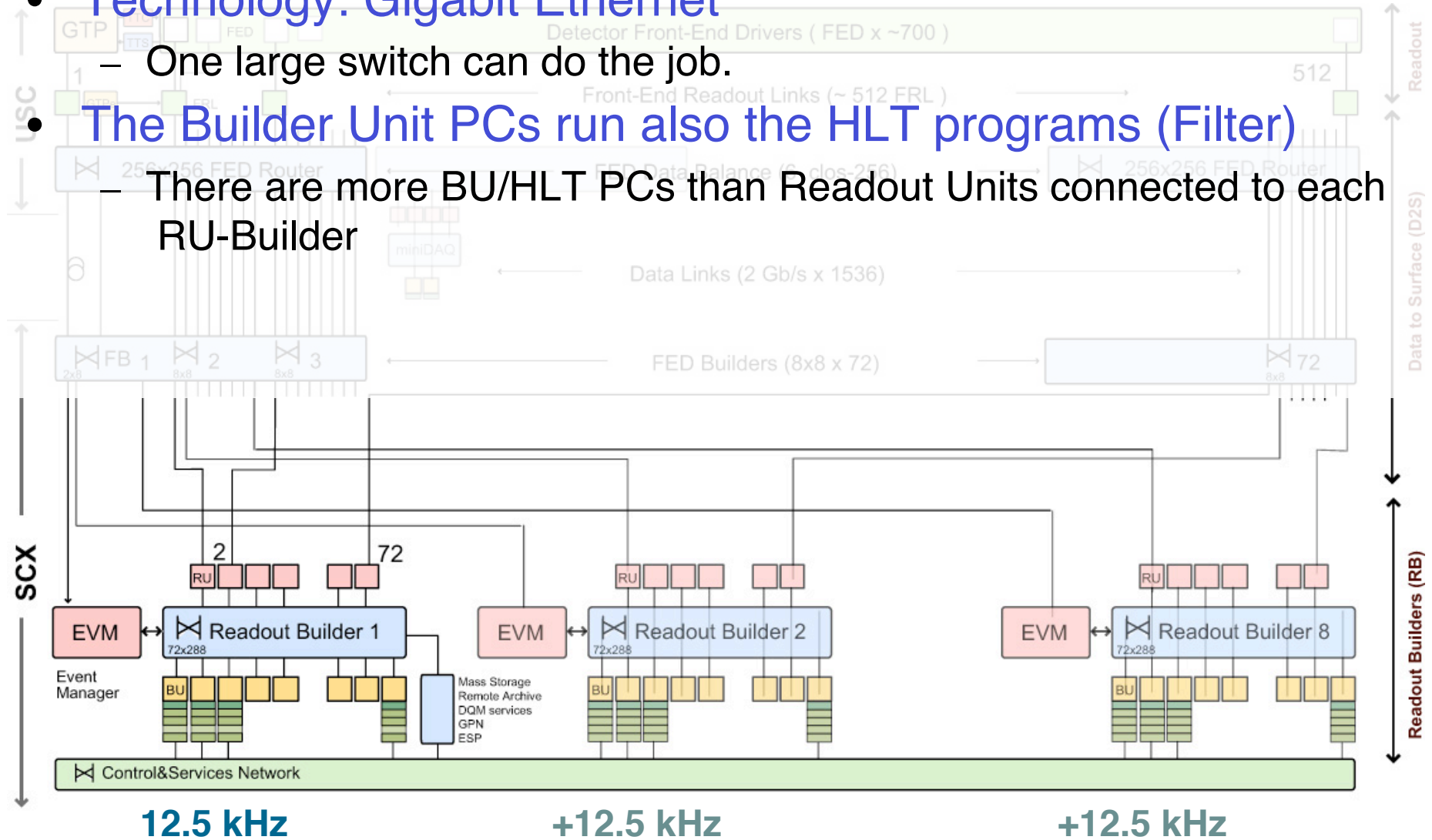


Half of the CMS FED Builder

One half of the FEDBuilder is installed close to the experiment in the underground. The other half is on the surface close to the 2nd stage-Builder and the Filter Farm Implementing the HLT. The FEDBuilder is used to transport the data to the surface.

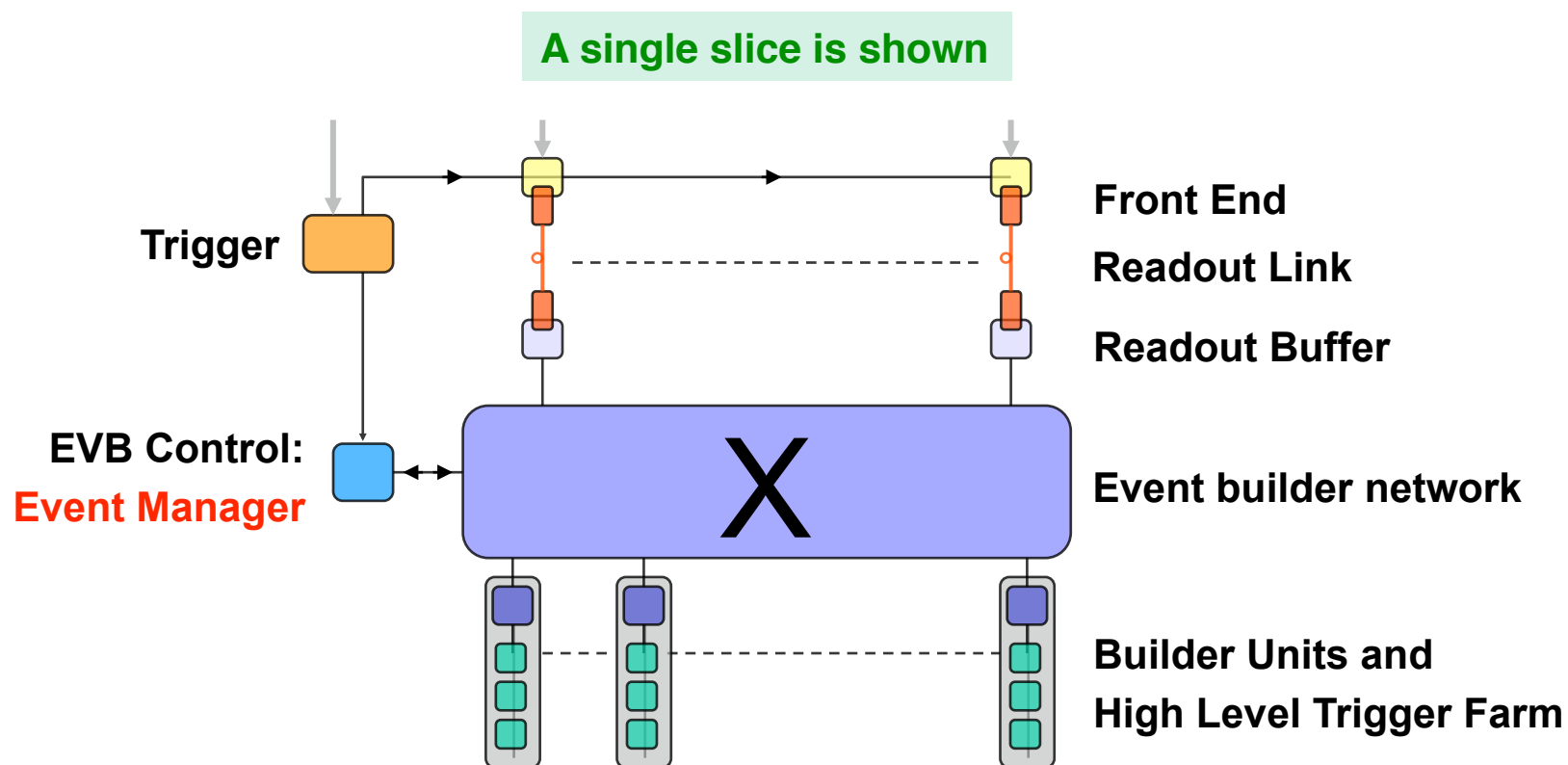
EVB 2nd stage: Gigabit Ethernet

- **Technology: Gigabit Ethernet**
 - One large switch can do the job.
- **The Builder Unit PCs run also the HLT programs (Filter)**
 - There are more BU/HLT PCs than Readout Units connected to each RU-Builder

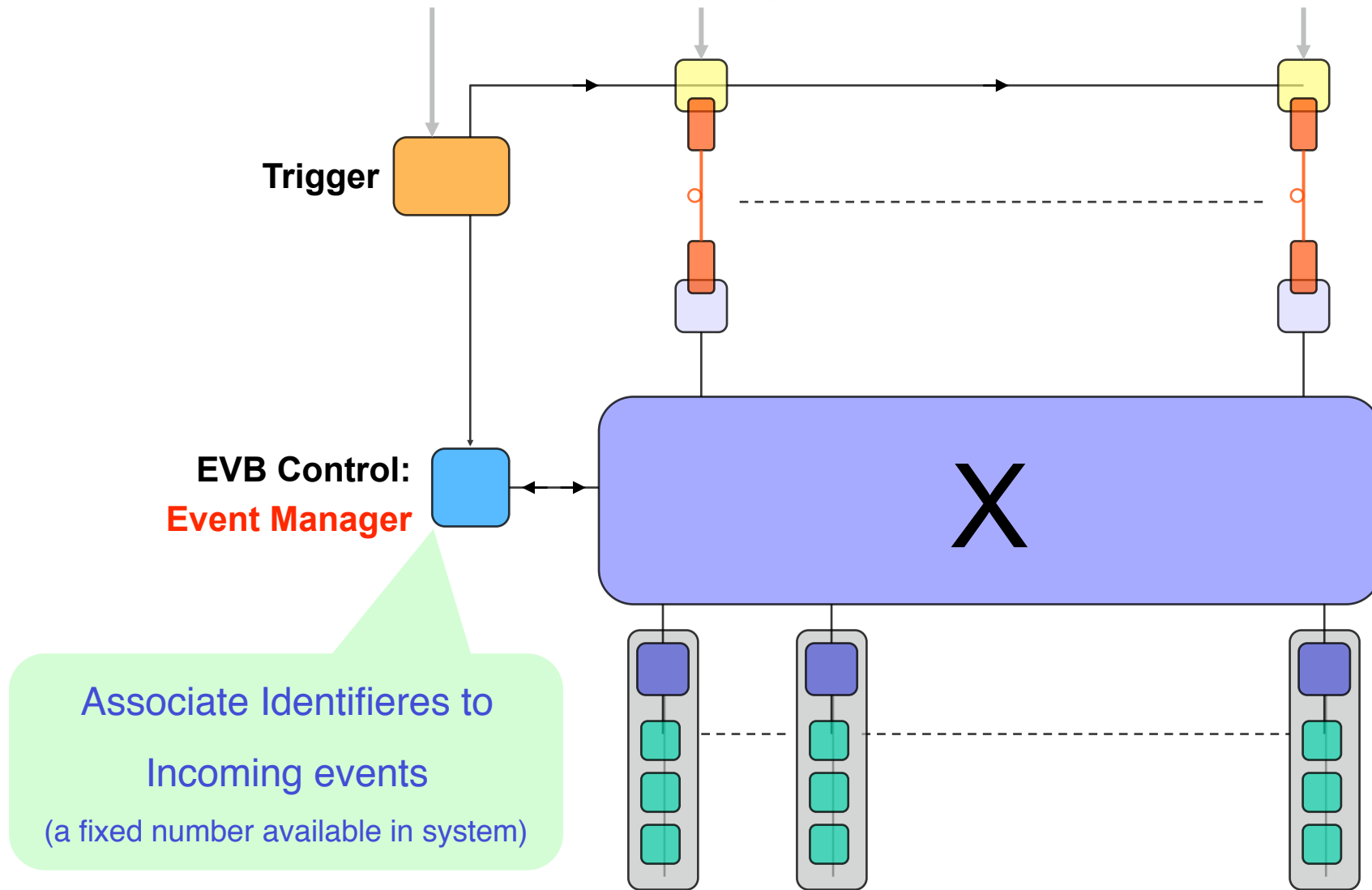


Event Building: EVB-Protocol

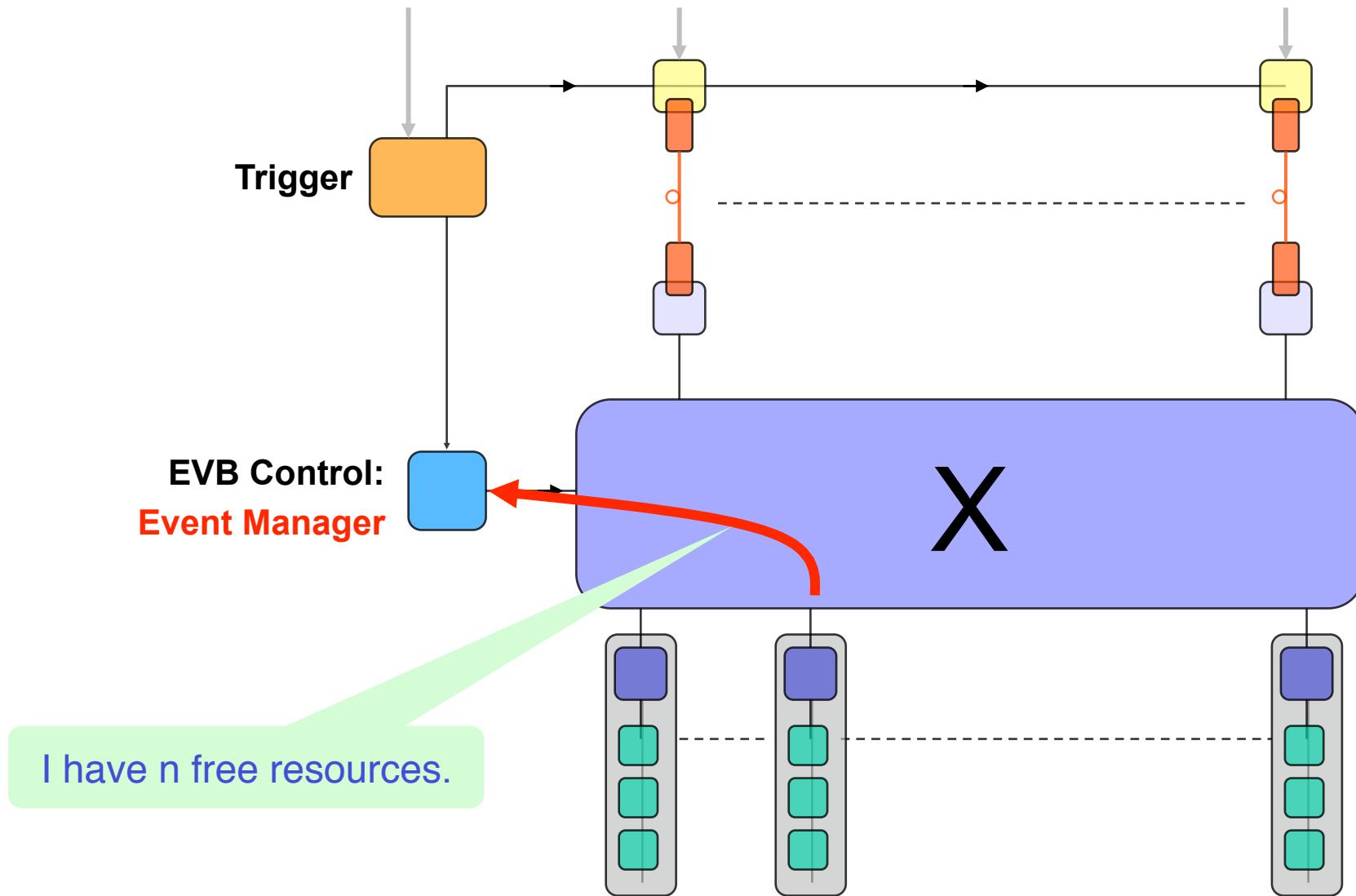
- Aim: Event Builder should perform load balancing
 - If for some reason some destinations are slower than others this should not slow down the entire DAQ system.
 - A form of **traffic shaping**



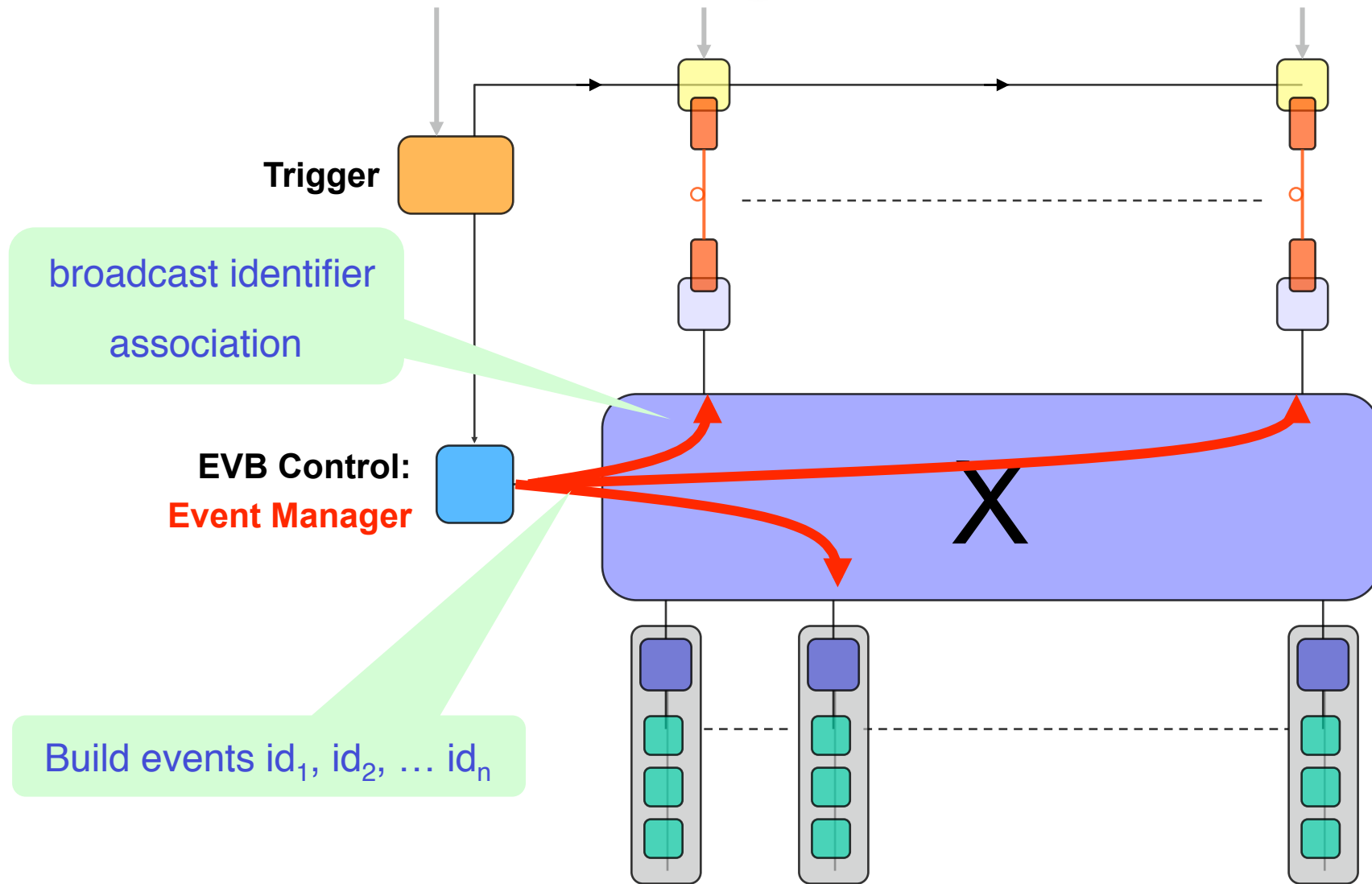
Event Building: EVB-Protocol



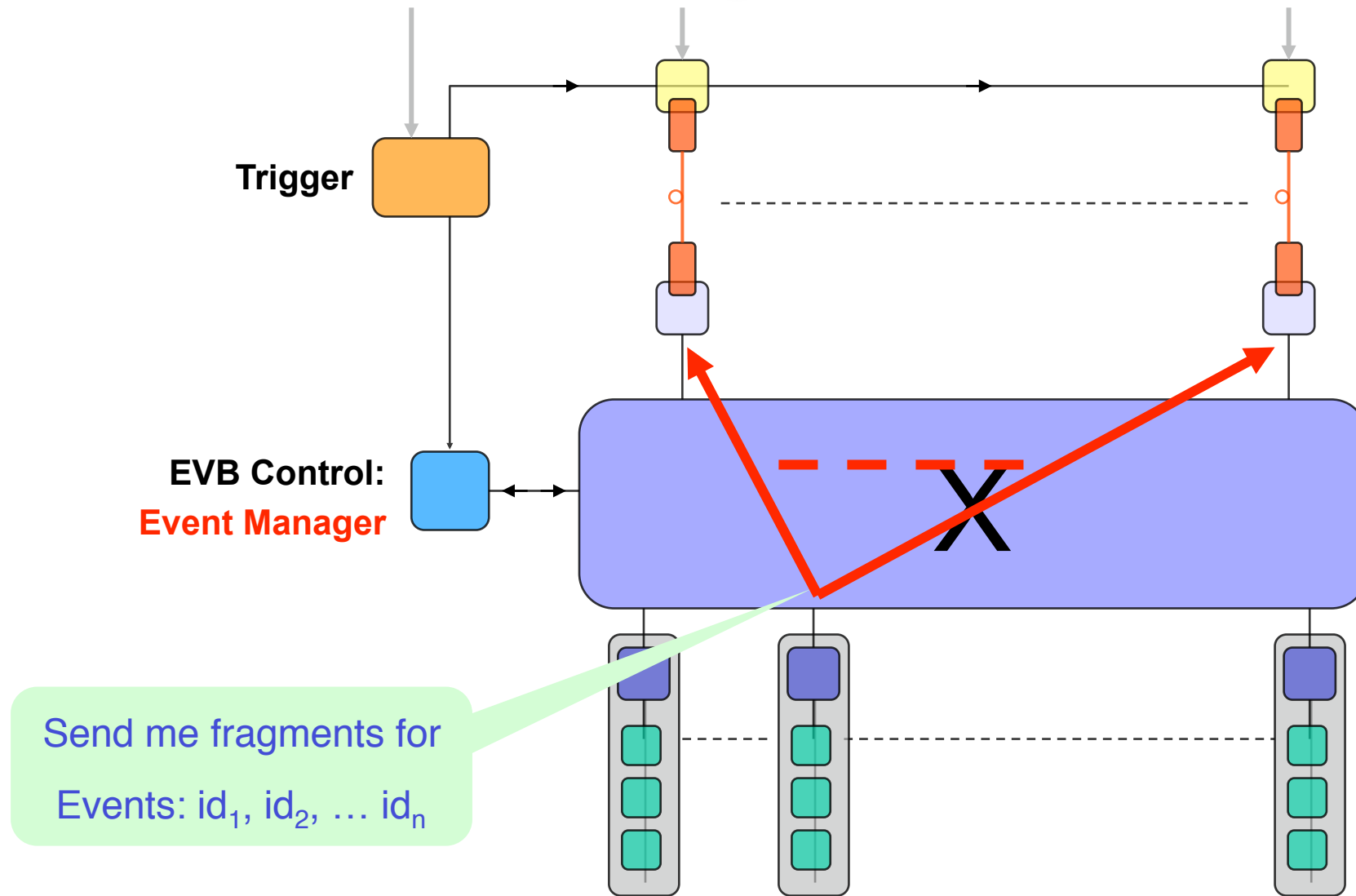
Event Building: EVB-Protocol



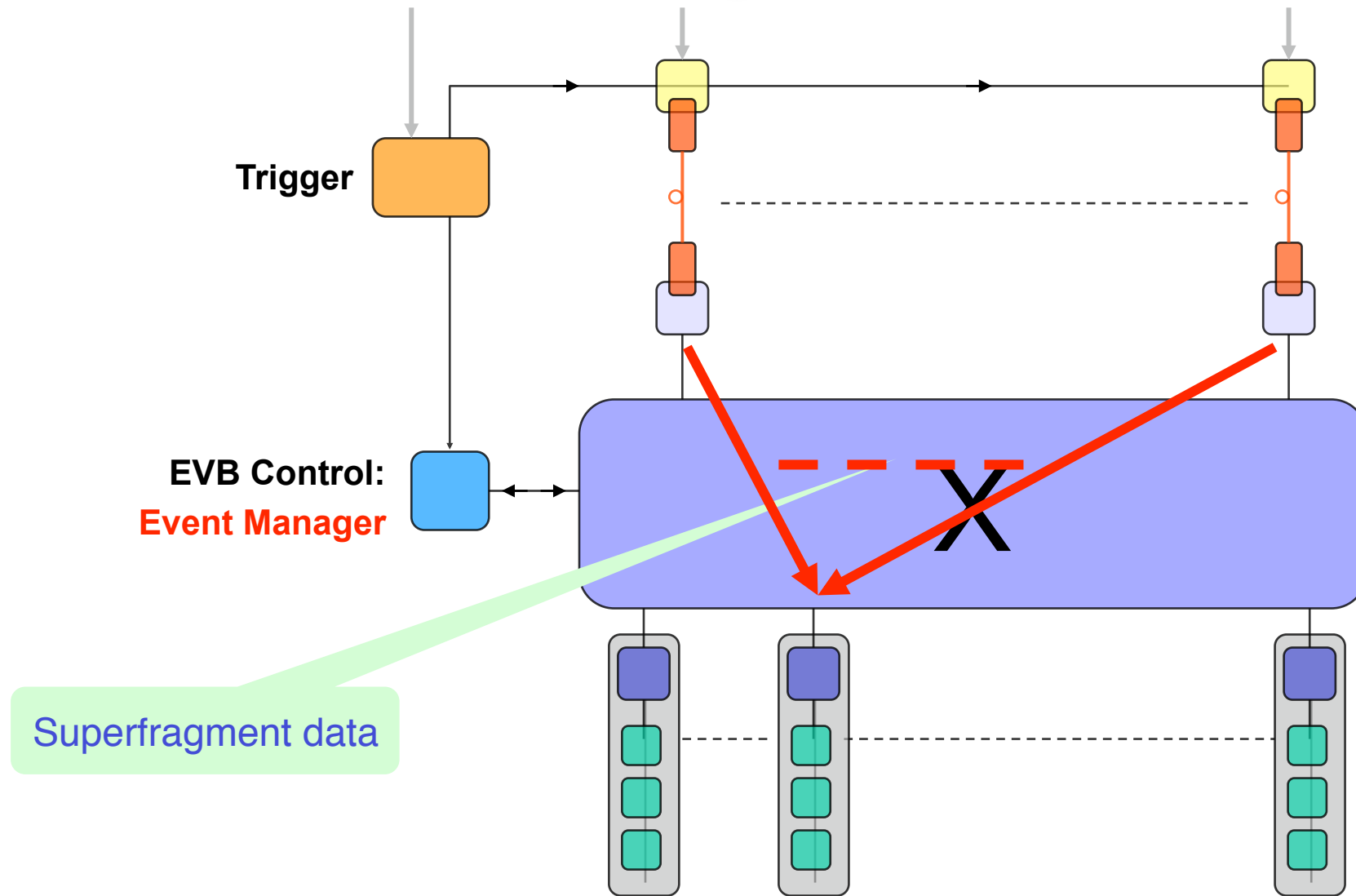
Event Building: EVB-Protocol



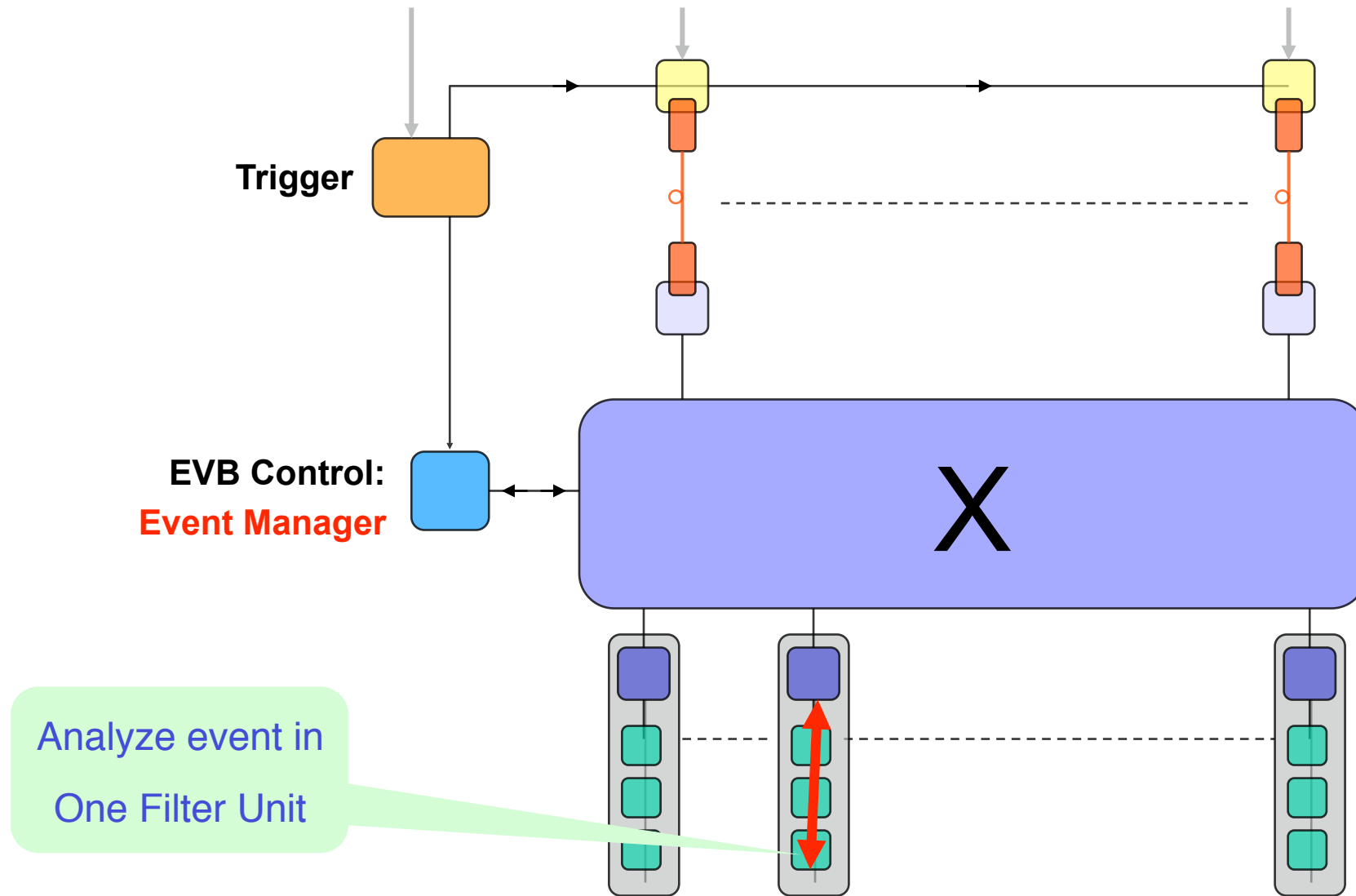
Event Building: EVB-Protocol



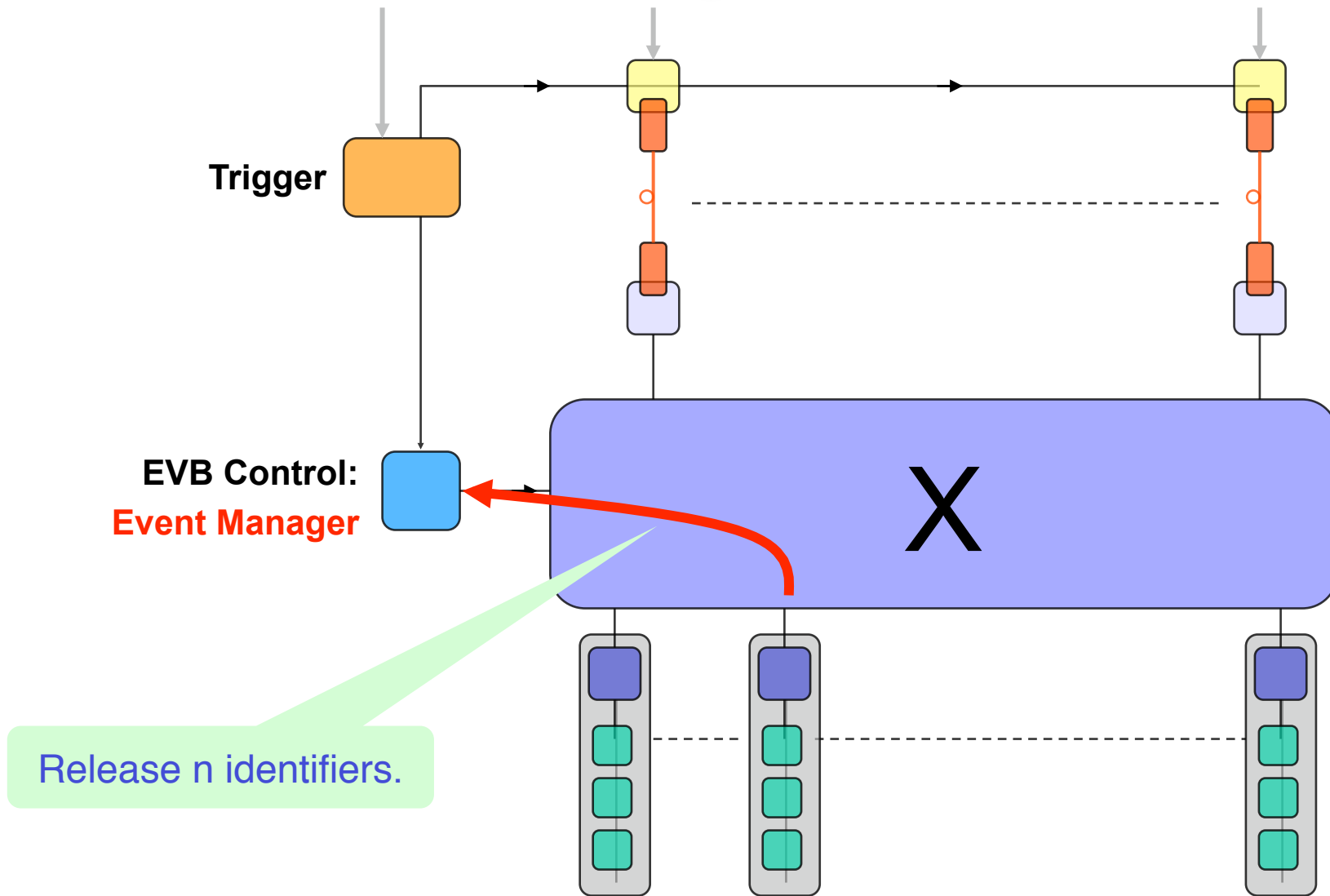
Event Building: EVB-Protocol



Event Building: EVB-Protocol



Event Building: EVB-Protocol



Event Builder Components



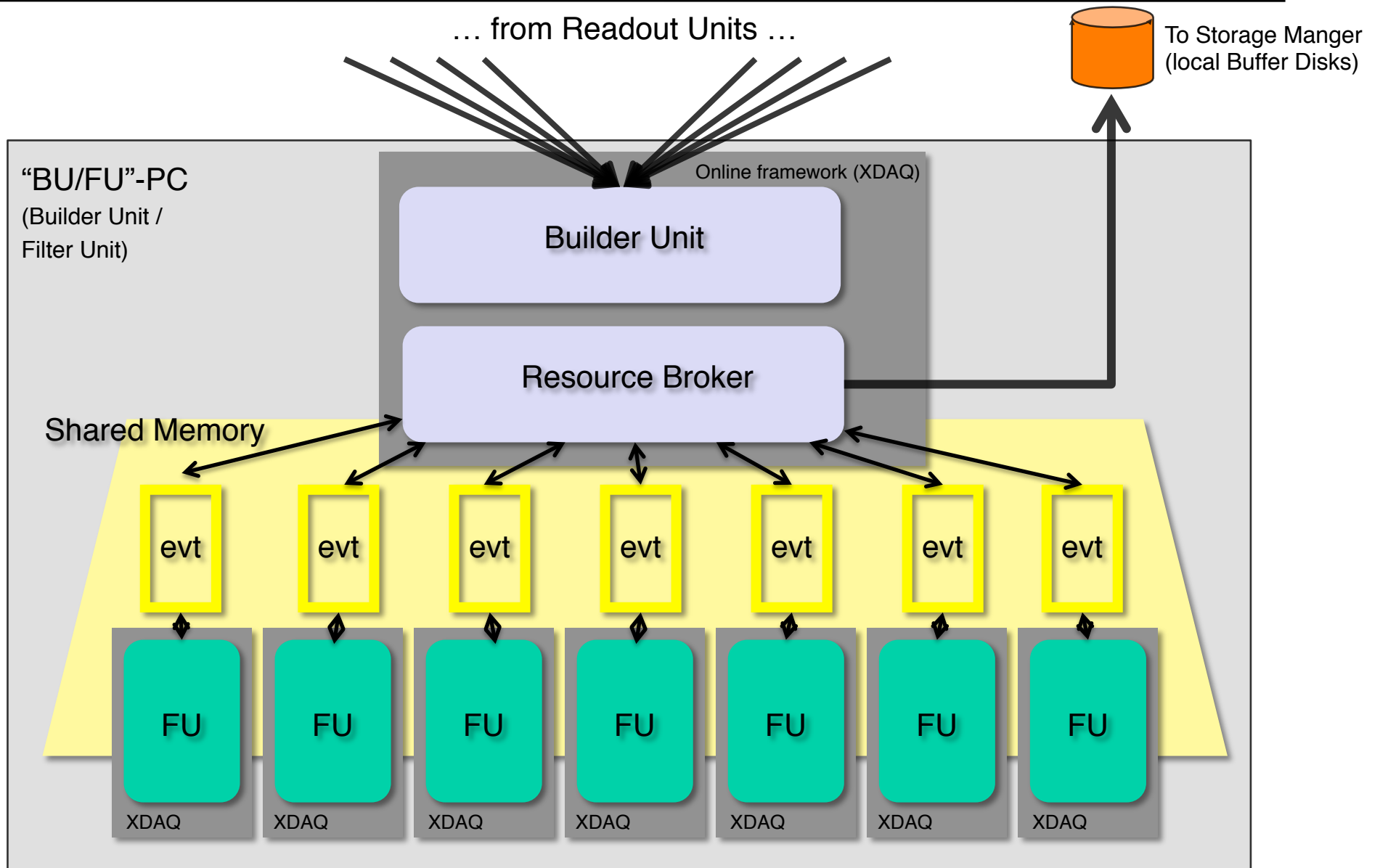
The RU-Builder Switch

Based on

- FORCE 10 E1200 chassis
- 90 port linecards (quad connectors)
- 8 chassis (one for each slice)

Implementation Filter Farm: Architecture and Components

Software Components of a Filter Node



Event Builder: Experience and outlook

Operation experience

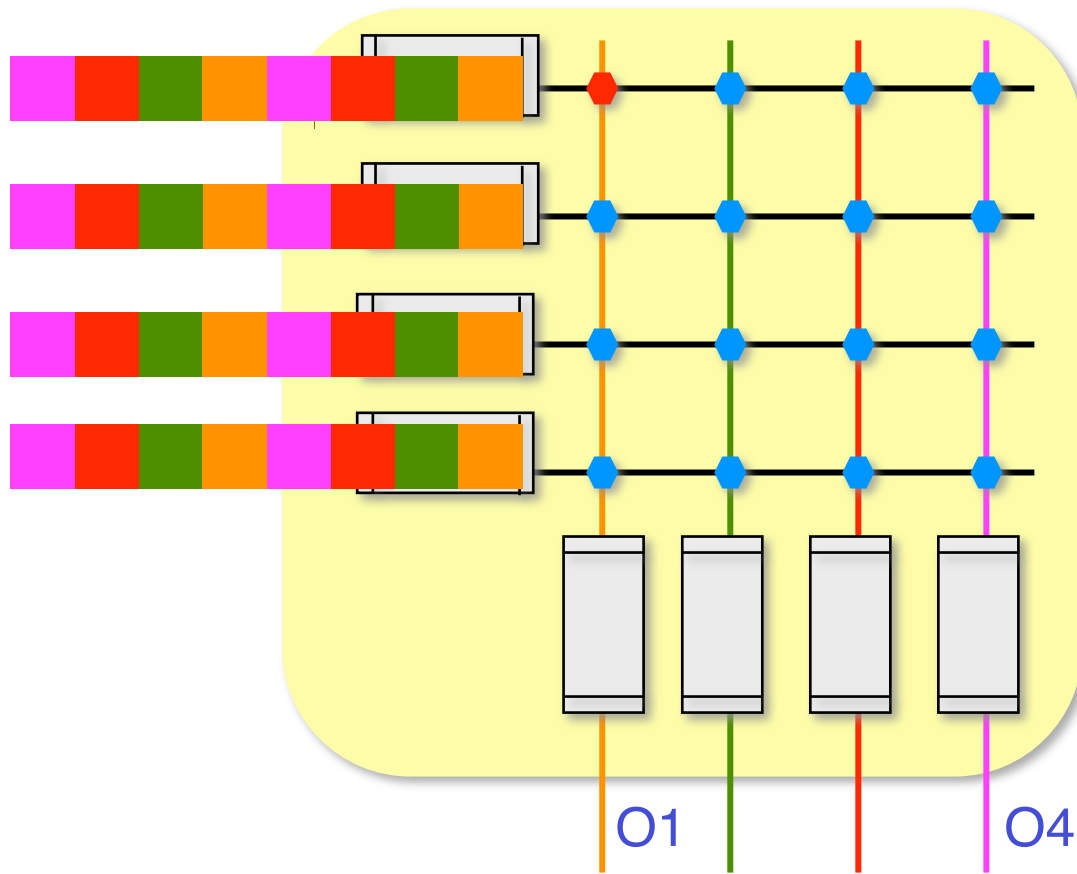
- The Fedbuilder and the RU builder have been operated successfully during the past year (and before)
 - Eventbuilder performs according to design specs: 100 GB/s
 - Design has proven to be very flexible:
 - Multiple test systems can be operated in parallel
 - Simultaneous operation of “global runs” and commissioning tests (hardware or software)
 - Systems can be tailored for specific needs:
 - High data throughput
 - Large number of input channels
 - High CPU power in filter
 - High data rate to local disk
- In progress
 - Automatic Failure handling
 - A Front End sends corrupted or no data in the middle of a run

Future: performance potential of EVB

- Fedbuilder performance increase via Traffic Shaping
 - Currently two Myrinet switch networks are operated at 50% wire speed.
 - Via “traffic shaping” the performance can be doubled.

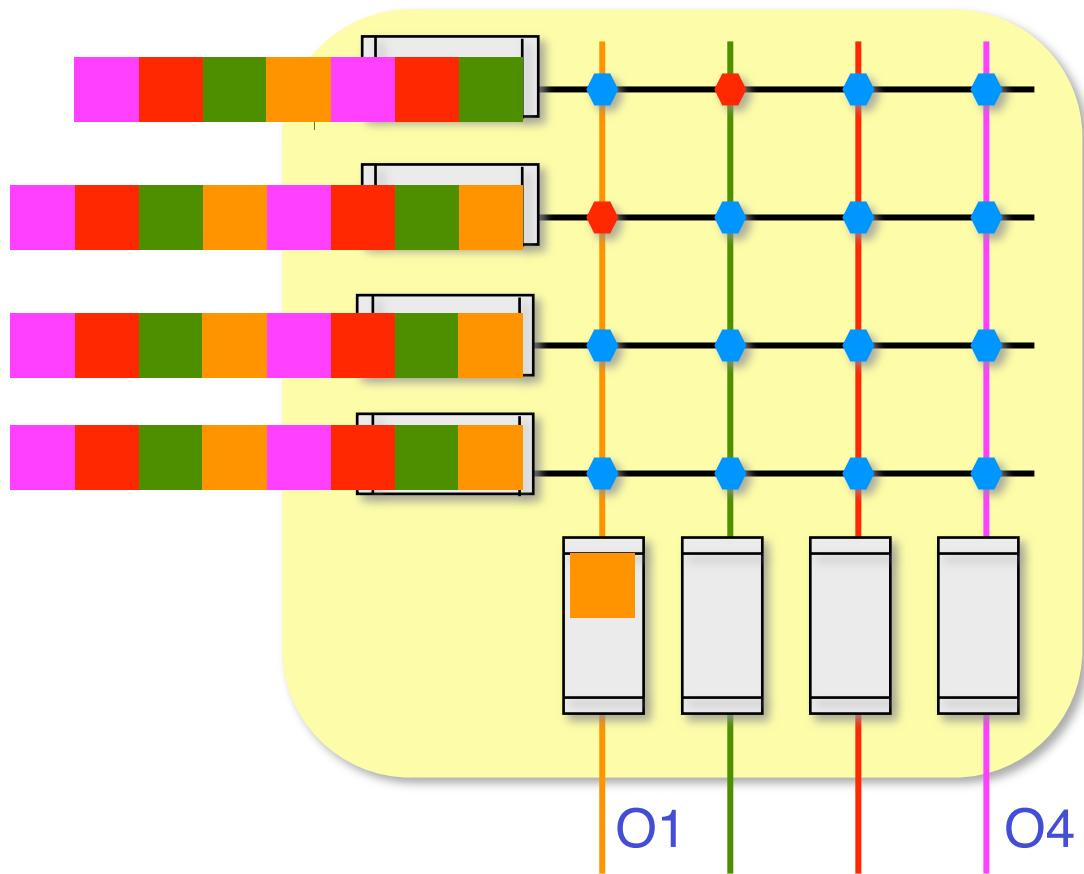
Switch implementation

Traffic shaping for the Crossbar switch: **Barrel Shifter**



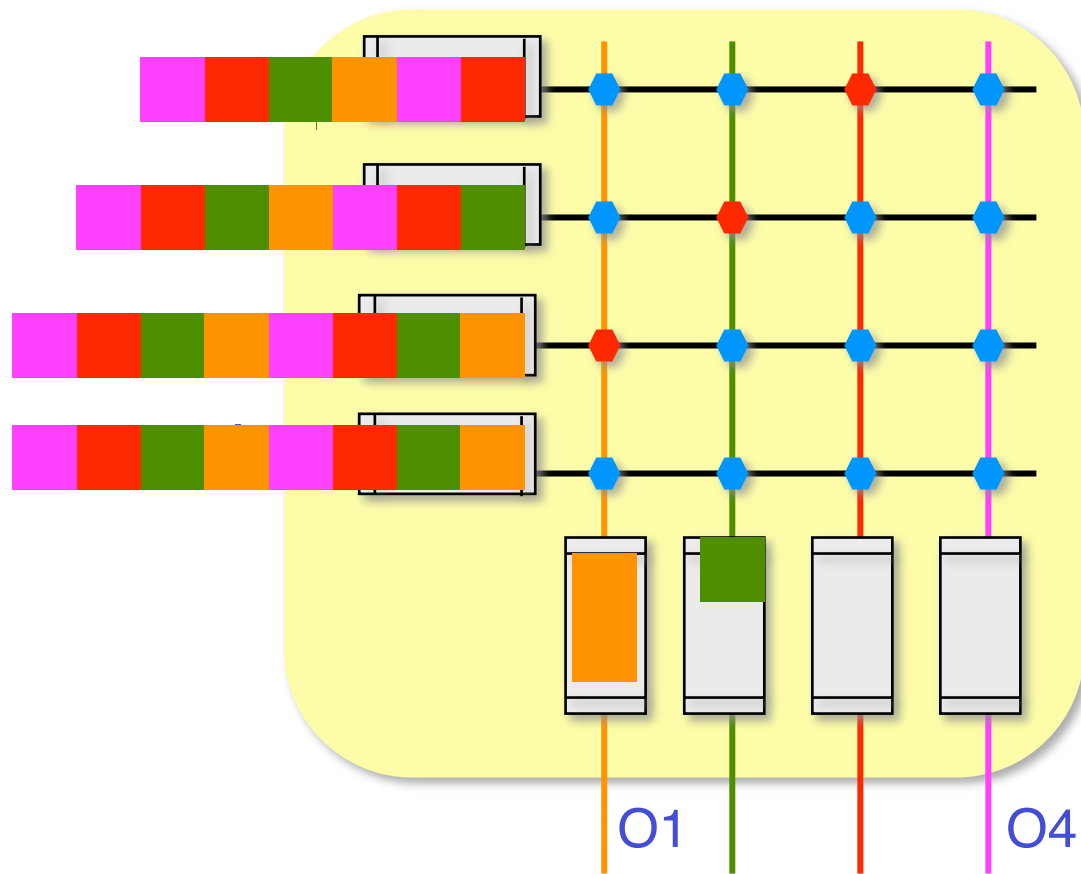
Switch implementation

Crossbar switch: perfect scenario



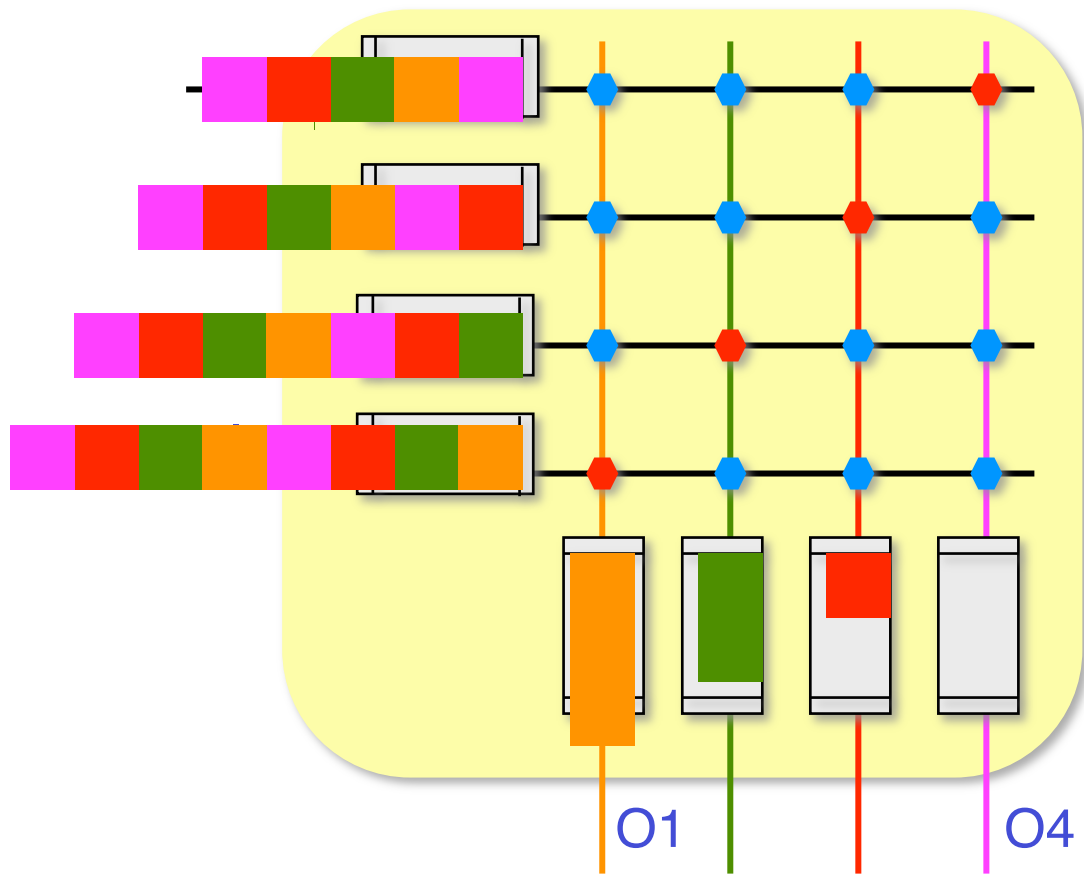
Switch implementation

Crossbar switch: perfect scenario



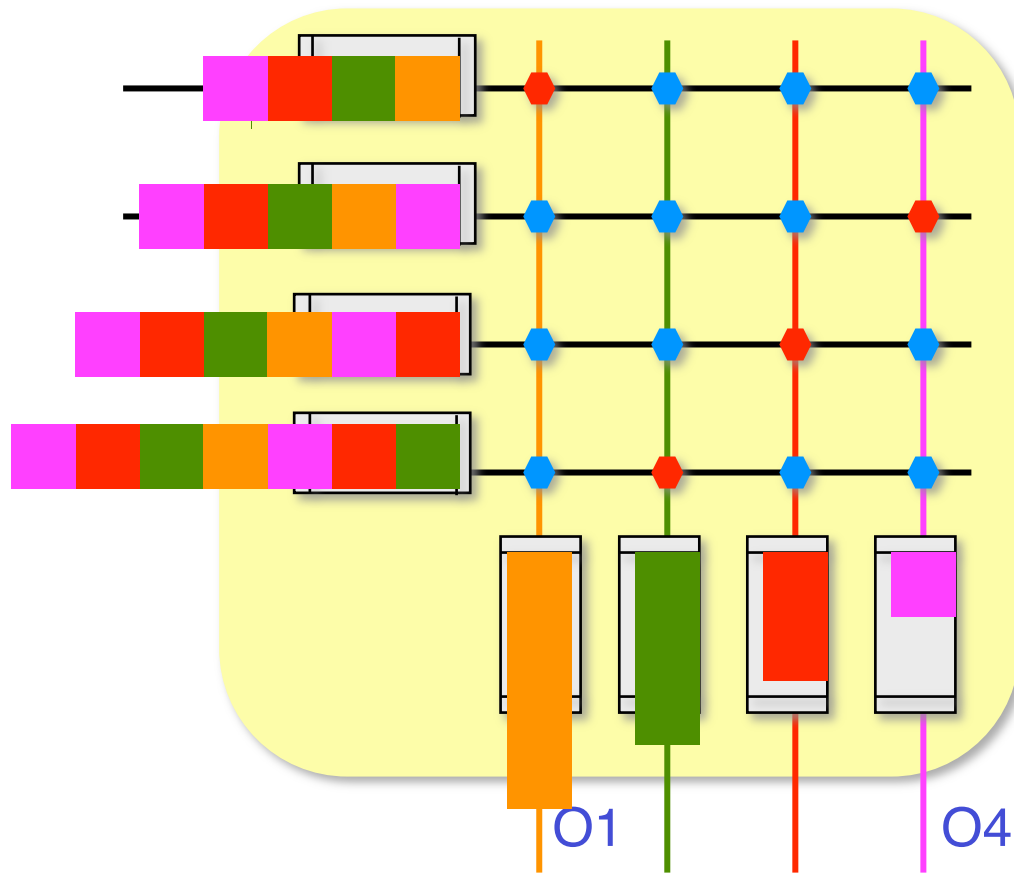
Switch implementation

Crossbar switch: perfect scenario



Switch implementation

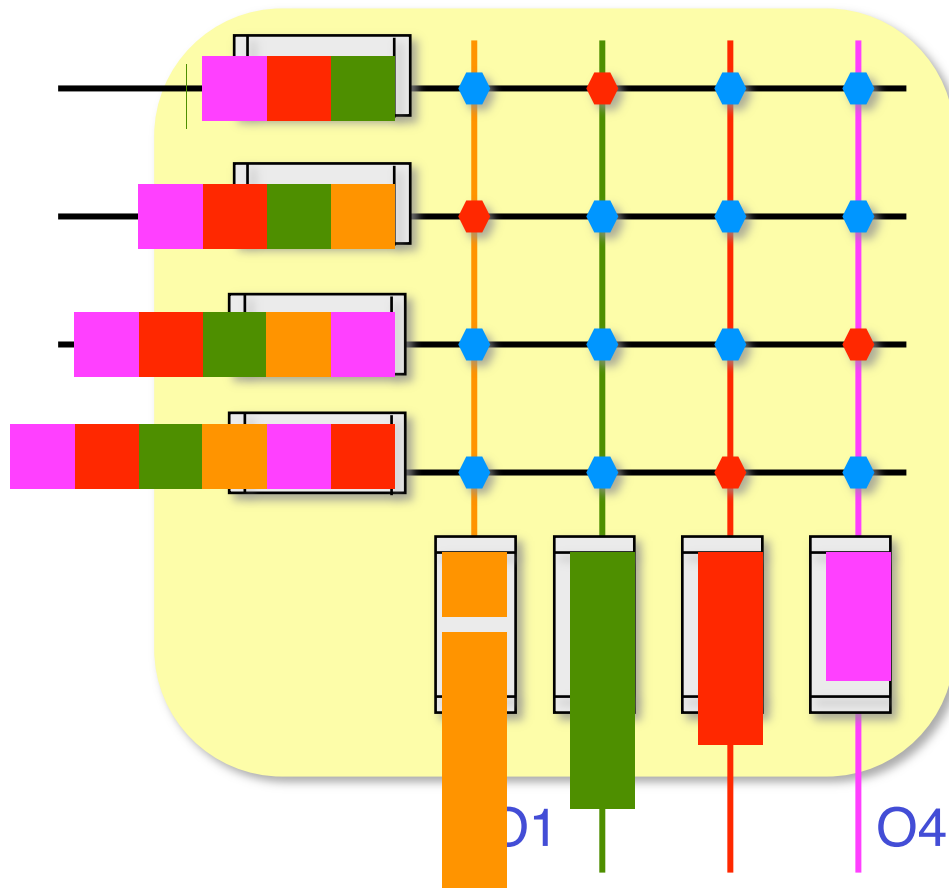
Crossbar switch: perfect scenario



Full wirespeed can be reached
(sustained) !

Switch implementation

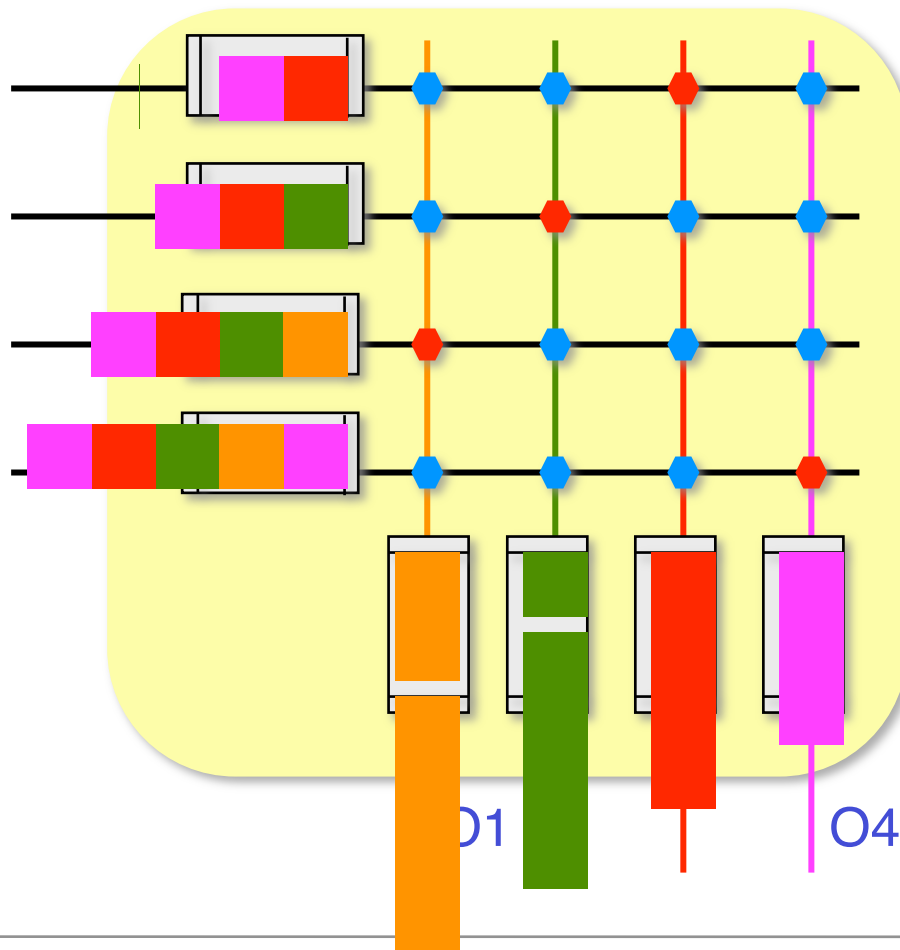
Crossbar switch: perfect scenario



Full wirespeed can be reached
(sustained) !

Switch implementation

Crossbar switch: perfect scenario



100% switch
utilization !

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** (i.e. methods, functions, **services**) **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.**
- Special situations: need of highest efficiency, only limited processing power (e.g. FPGA processors, firmware for special cards (NICs)) C is the better choice.

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.

Online software frameworks in CMS

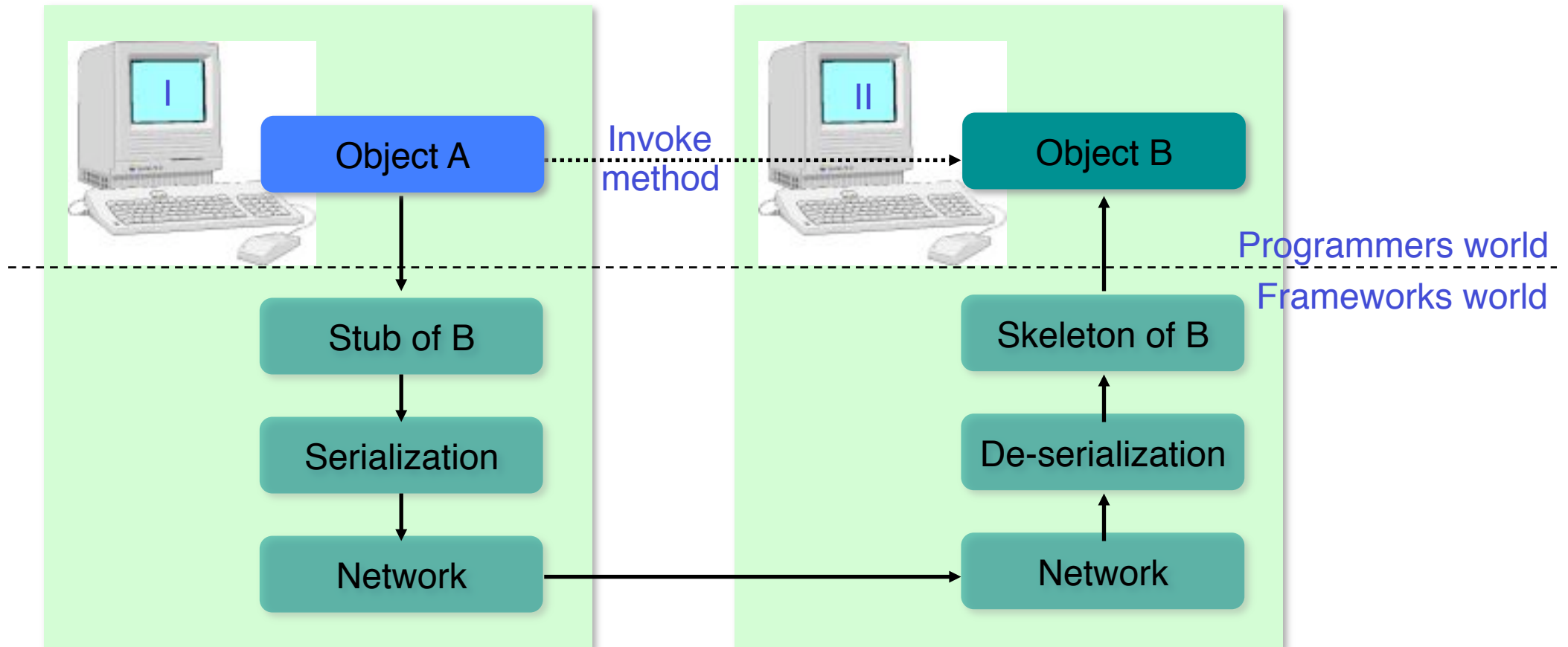
- **Event builder software and Control software for Front End**
 - C++ Framework “XDAQ” developed at CERN for
 - Data transport
 - Configuration
 - Monitoring
 - Error Messages
- **Run Control software**
 - Framework (RCMS) developed at LNL in collaboration with CERN
 - Web based (Tomcat Server houses the RCMS system)
 - Services for DAQ Control Applications (Function Managers)
 - Database access to retrieve configuration and write conditions
 - Logging
 - Error handling
 - Communication with XDAQ applications
 - Monitoring
 - State Machines
 - User interface

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

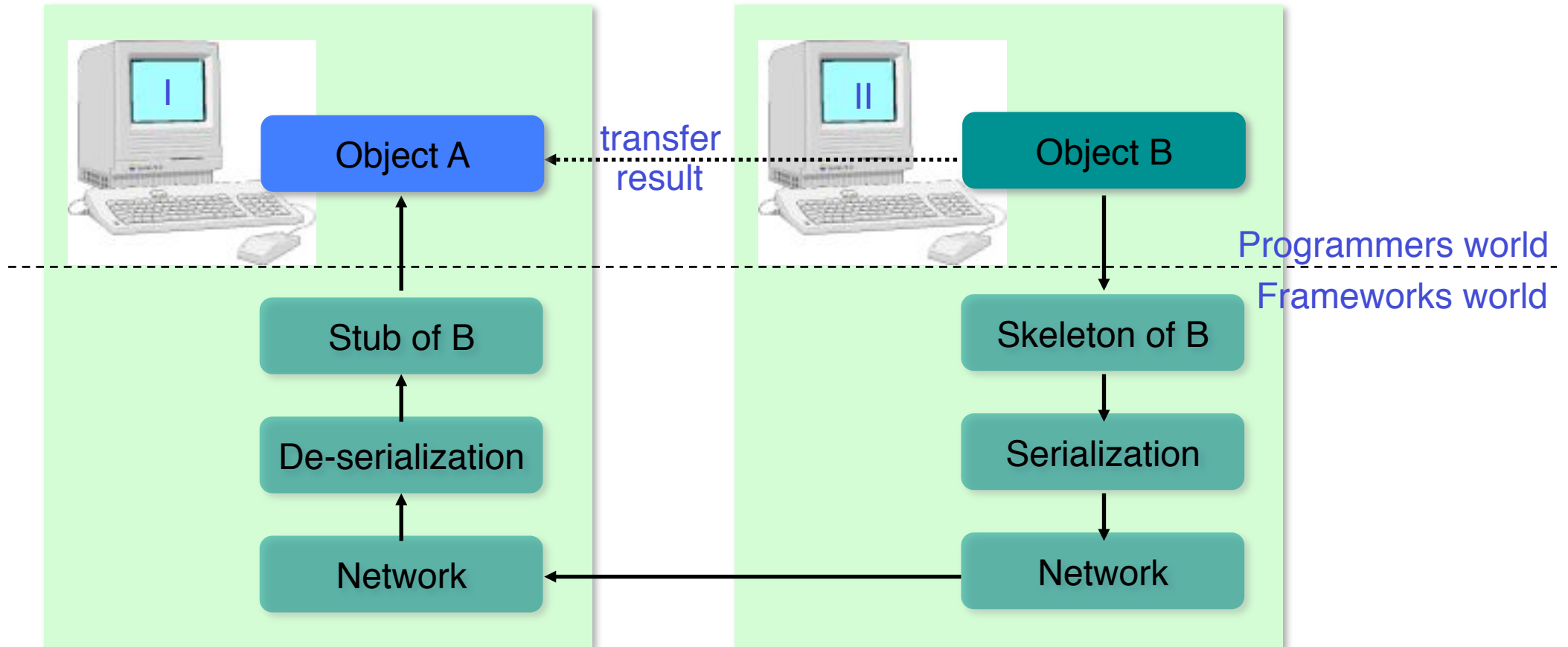
Distributed computing

A method on a remote object is called:



Distributed computing

The result is coming back:



Handling of Multi Core machines in CMS

- **XDAQ framework**
 - Framework supports multiple threads
 - Event building applications exploit multi threading
 - Software design takes into account the number of cores (2x2 cores for event-building machines)
- **Run Control (RCMS)**
 - Heavily uses multi threading (a lot of support in Java/Tomcat)
- **HLT Filter farm**
 - Runs on 2x4 core machines with 16 GB
 - 7 distinct processes
 - Event Data is specific to each event.

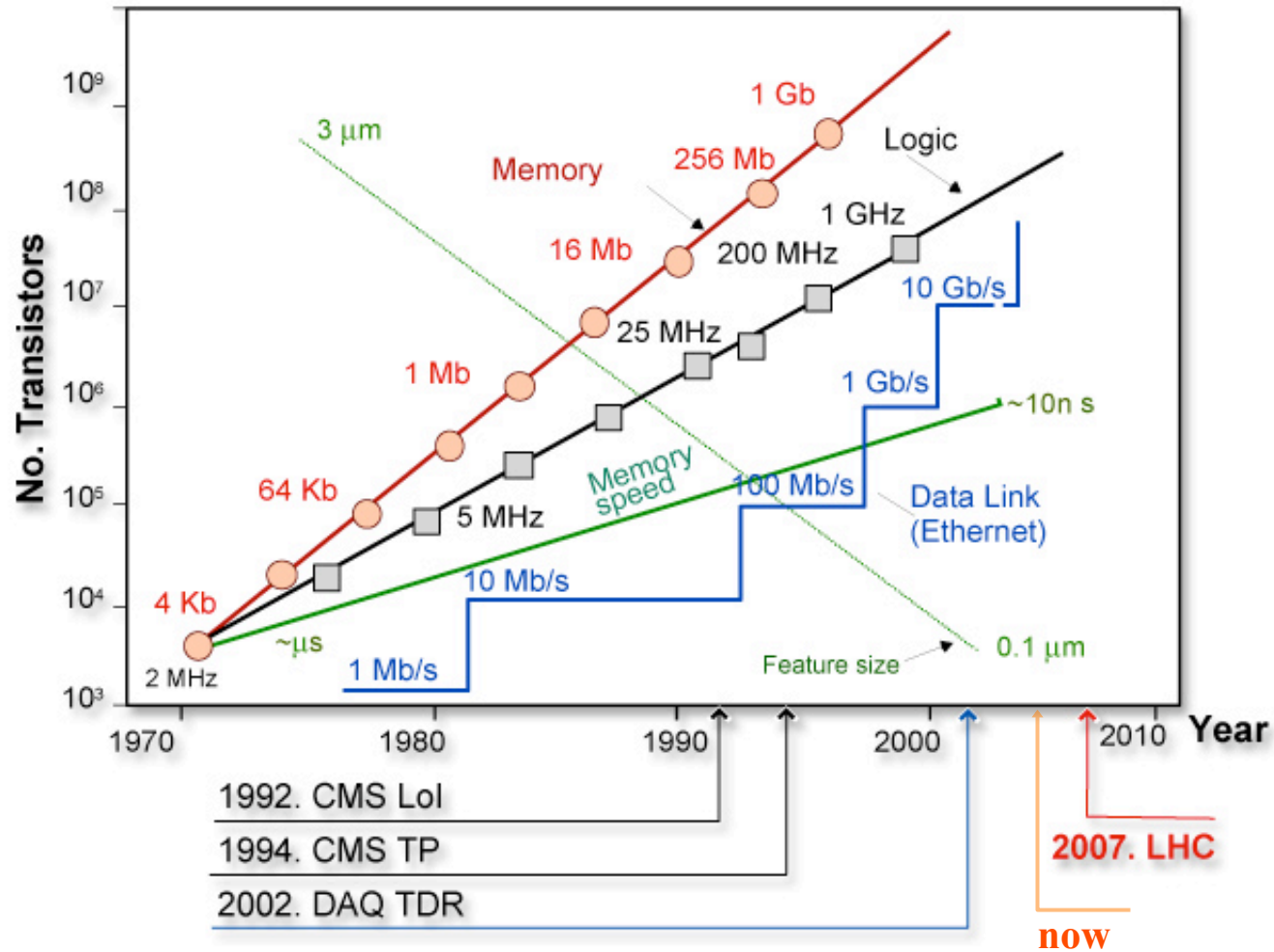
Conclusions

- **CMS DAQ system is fully functional**
 - Performance is according to original specifications
 - No fundamental problems found
 - During the first beam and various cosmic tests data has been successfully taken with all sub-detectors
- **Still to be done**
 - Error and fault tolerance in the central DAQ system
 - Automatic recovery on reception of corrupted data or on missing data from the front end.
 - On a CMS wide scale: Improve robustness and speed if starting a large system containing all front end channels, and the entire Filter Farm (5000 cores).

EXTRA SLIDES

Outlook

Moore's Law

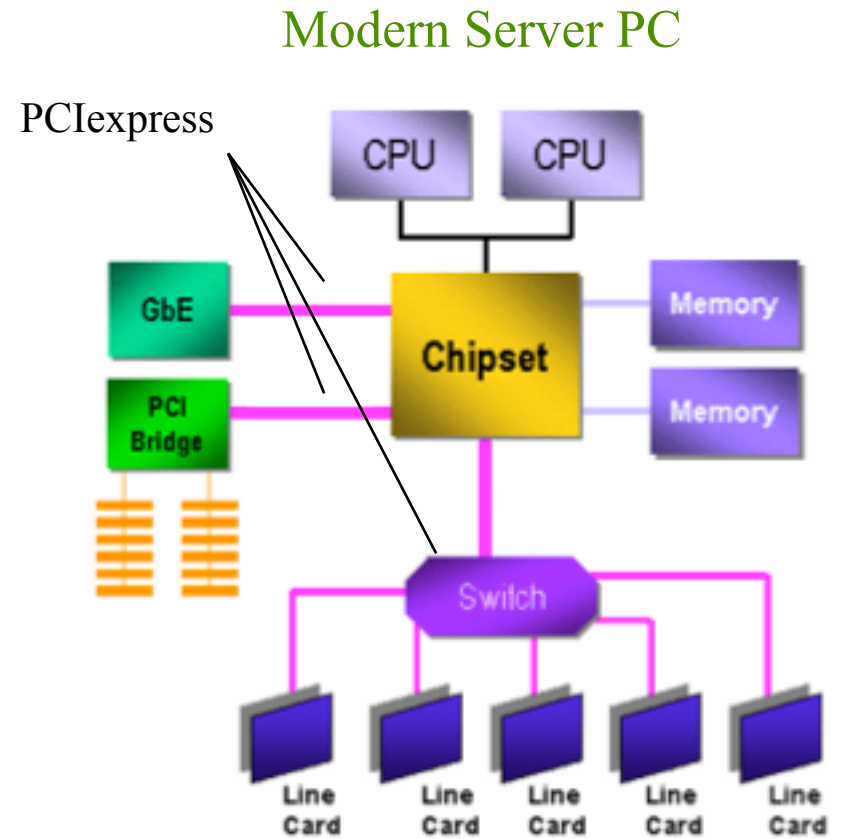


Technology: FPGAs

- 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

Technology: PCs

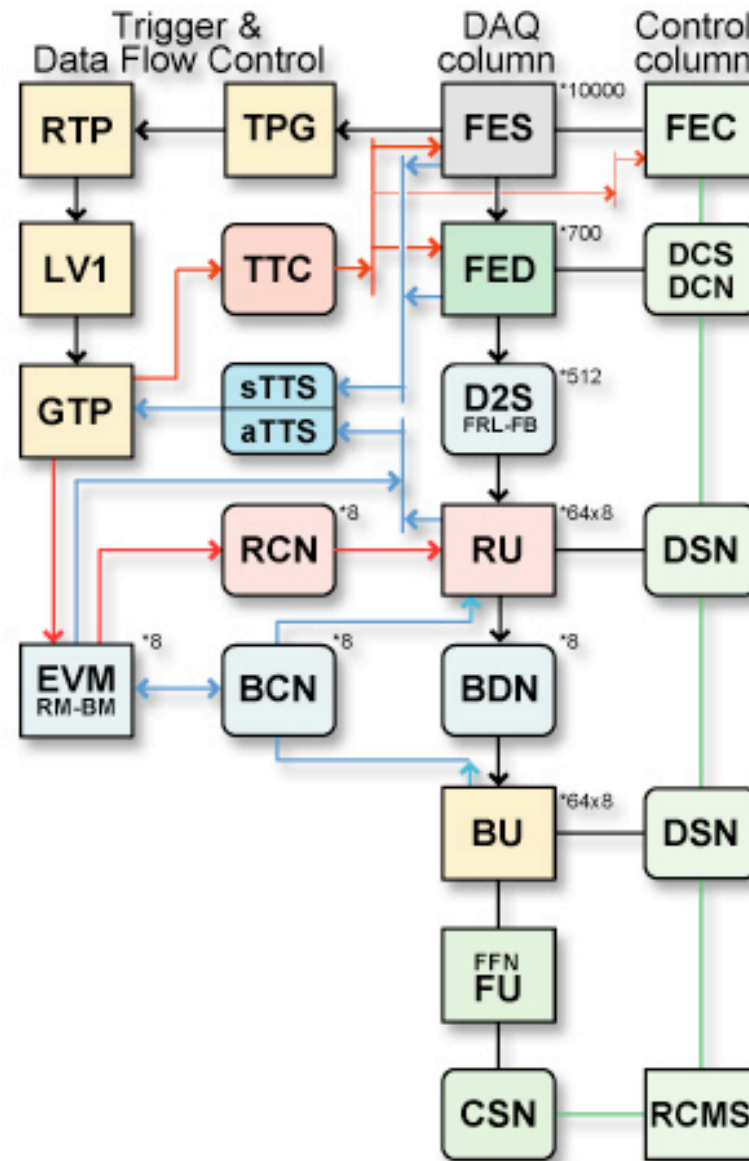
- **Connectivity**
 - PCI(x) --> PCI express
 - 10GbE network interface
- **INTELS Processor technology**
 - Future processors (2015):
 - Parallel processing
 - Many CPU-cores on the same silicon chip
 - Cores might be different (e.g. special cores for communication to offload software)



Conclusions

- LHC experiments will produce an unprecedented amount of data at all levels:
 - Front end
 - Data readout (after the hardware trigger levels)
 - At the higher trigger level
 - At the permanent storage
- Readout and event building
 - Many common concepts (driven by similar requirements and technology development)
 - Still some choices to be made (compare EVBs of ATLAS and CMS)
- Summary: I believe we will have a lot of fun ...and stress... in the forecoming months

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

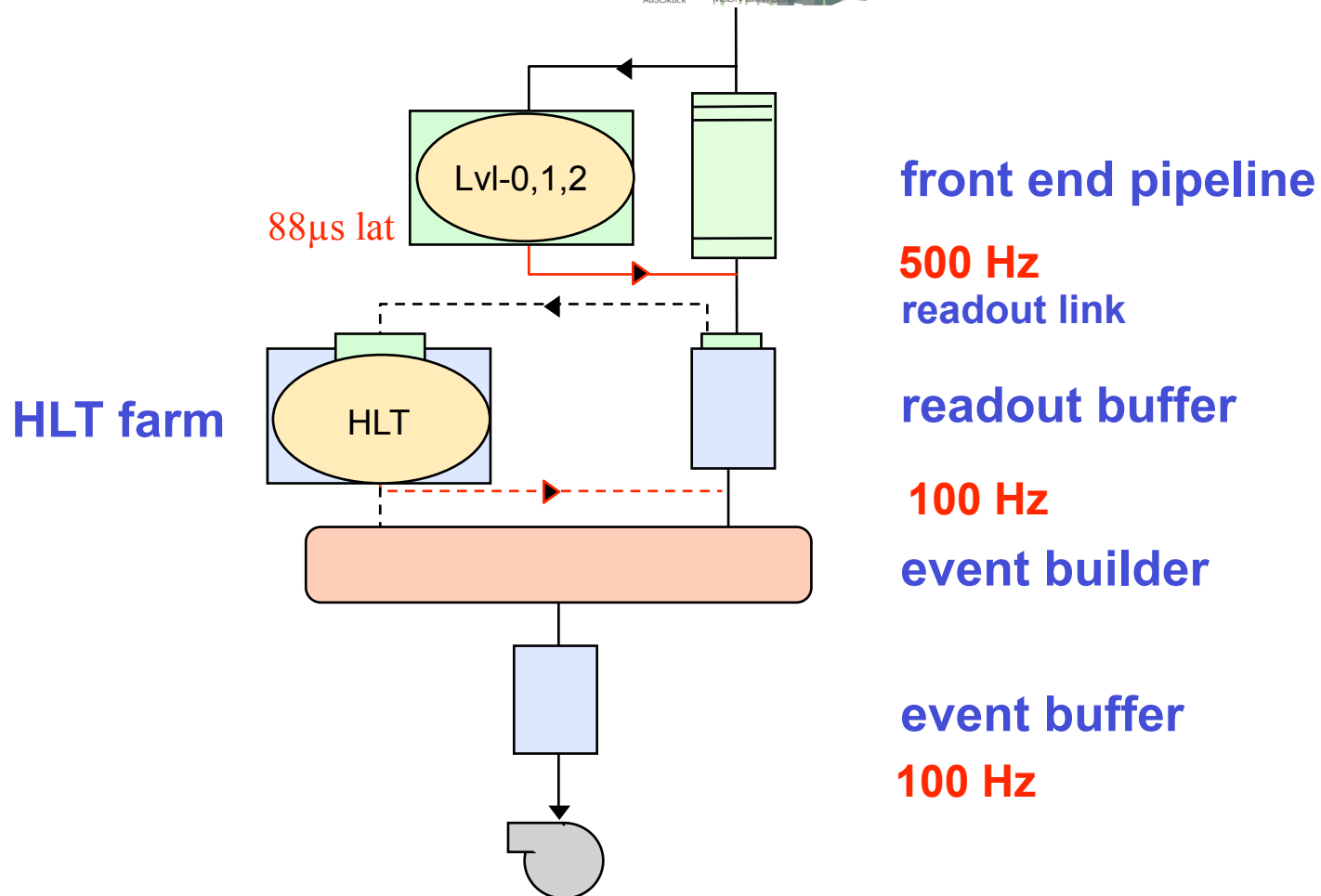
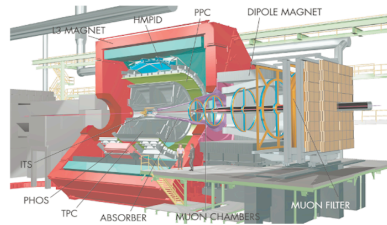
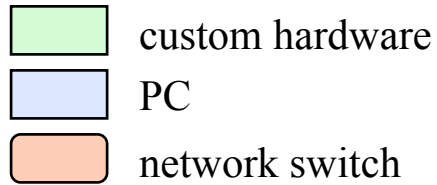
ALICE: optimizing efficiency

- **Concept of Trigger clusters:**
 - Trigger cluster: group of sub-detectors
 - one sub-detector can be member of several clusters
 - Every trigger is associated to one Trigger Clusters
 - Even if some sub-detectors are busy with readout (Silicon Drift Detector needs $260 \mu\text{s}$ to read out), triggers for not-busy clusters can be accepted.
- **Triggers with “rare” classification:**
 - In general at LHC: stop the trigger if readout buffer almost full
 - ALICE:
 - “rare” triggers fire rarely and contain potentially interesting events.
 - when buffers get “almost-full” accept only “rare” triggers

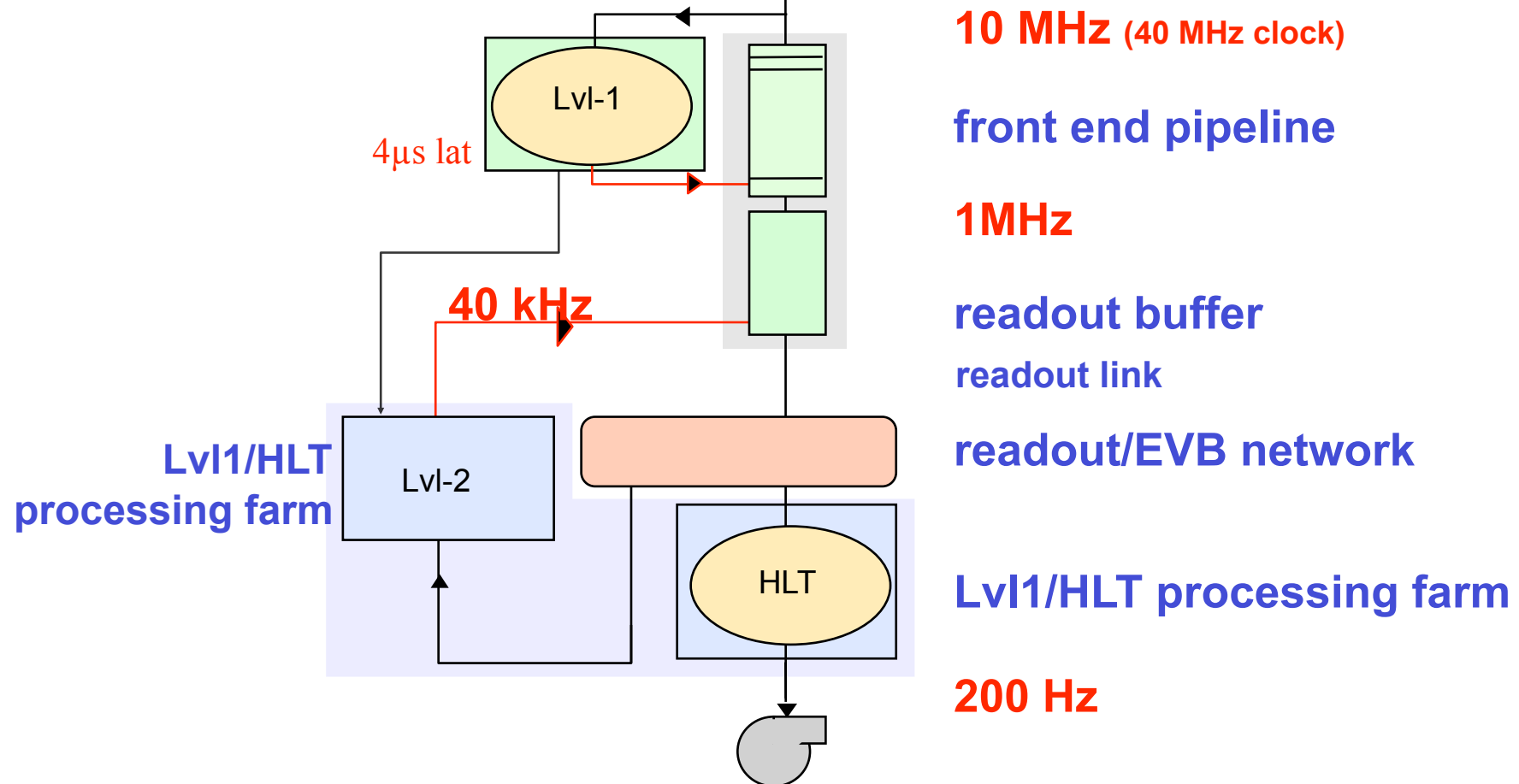
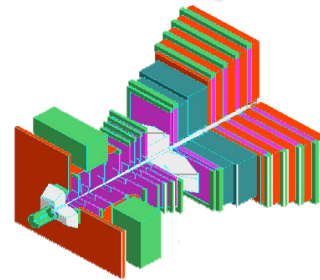
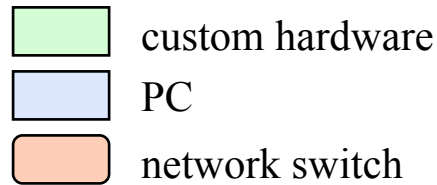
First level trigger: Implementation

- Custom Electronics design
 - Preprocessing and final trigger logic based on ASICs and FPGAs
 - Use of ASICs
 - Can be produced radiation tolerant (for “on detector” electronics)
 - Can contain “mixed” design: analog and digital
 - In some cases more economic (large numbers, or specific functionality)
 - But: Higher development “risk”, longer development cycles
- Extremely high connectivity with low latency
 - Large cards because of large number of IO channels
 - Many identical channels processing data in parallel
 - Custom links
 - Backplane parallel busses for in-crate connections
 - LVDS links for short ($O(10\text{m})$) inter-crate connections

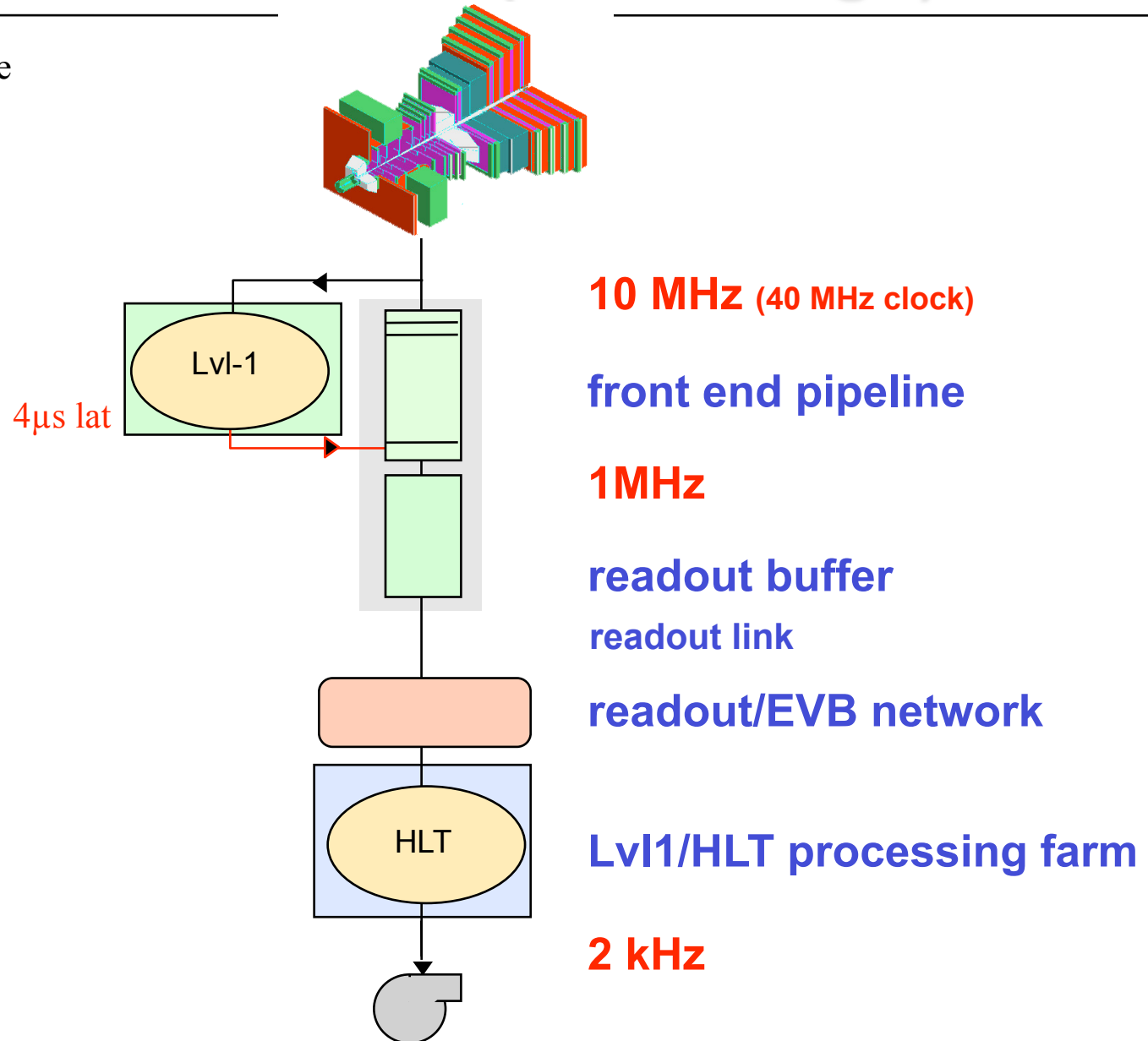
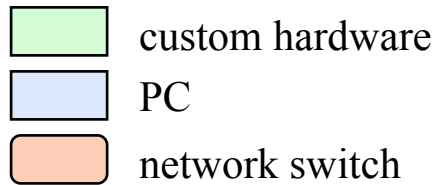
Data Flow: ALICE



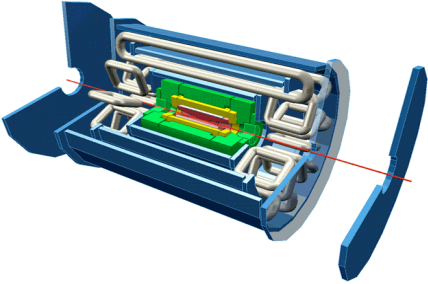
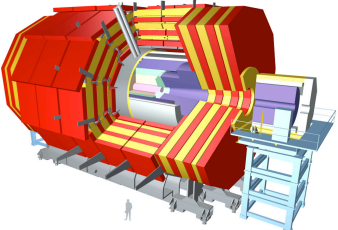
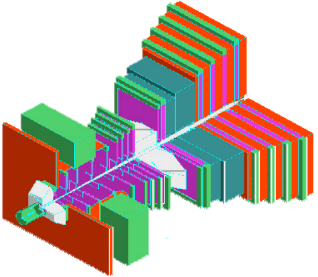
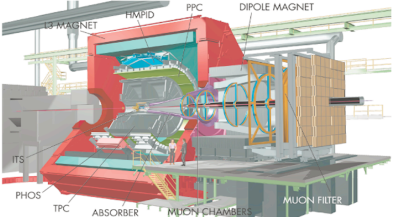
Data Flow: LHCb (original plan)



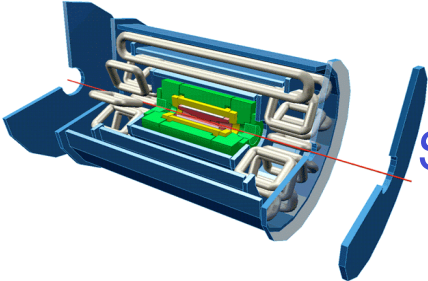
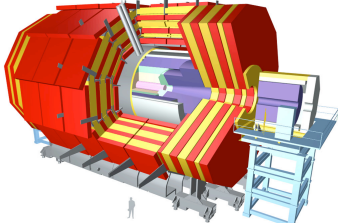
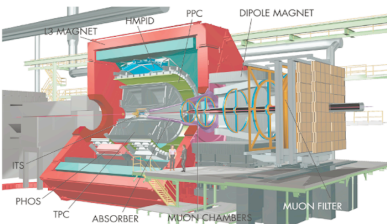
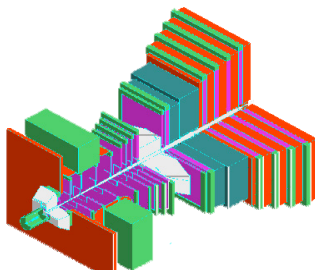
Data Flow: LHCb (final design)



Trigger/DAQ parameters

	No.Levels	Level-0,1,2	Event	Readout	HLT Out
	Trigger	Rate (Hz)	Size (Byte)	Bandw.(GB/s)	MB/s (Event/s)
	3	LV-1 10^5 LV-2 3×10^3	1.5×10^6	4.5	300 (2×10^2)
	2	LV-1 10^5	10^6	100	O(1000) (10^2)
	2	LV-0 10^6	3×10^4	30	40 (2×10^2)
	4	Pb-Pb 500 p-p 10^3	5×10^7 2×10^6	25	1250 (10^2) 200 (10^2)

Readout Links of LHC Experiments

				Flow Control
	SLINK	Optical: 160 MB/s Receiver card interfaces to PC.	≈ 1600 Links	Yes
	SLINK 64	LVDS: 200 MB/s (max. 15m) Peak throughput 400 MB/s to absorb fluctuations Receiver card interfaces to commercial NIC (Myrinet)	≈ 500 links	yes
	DLL	Optical 200 MB/s Half duplex: Controls FE (commands, Pedestals, Calibration data) Receiver card interfaces to PC	≈ 400 links	yes
	TELL-1 & GbE Link	Copper quad GbE Link Protocol: IPv4 (direct connection to GbE switch) Forms "Multi Event Fragments"	≈ 400 links	no

Technical aspects: Implementation choices

Custom electronics

- Choice for electronics in Counting House
 - Sub-detectors house their electronics in VME crates
 - Solid Mechanics and Electrics
 - VME only used for configuration and local DAQ: no performance issue
 - Controller is a bridge to a PC.
 - Standard Linux PC maintained by the system administration like all other PCs in the computing farm.
 - CPU power of a normal PC for user processes
 - Easy upgrade possible
 - Simple interface (VME is easy to implement in custom electronics)
 - DAQ hardware uses CompactPCI
 - Higher data rate
 - Seamless integrated into PC world (bridge extends internal PCI bus)

Computing nodes in EVB

- **Eventbuilding nodes and Computing nodes**
 - Multi core (4-8) Server PCs running all CERN Linux (currently SLC4)
 - Installation and configuration utility for the cluster: Quattor
 - initially developed by CERN; now on sourceforge.
 - Allows to re-install from scratch a machine within 7 minutes.
 - Allows to distribute software packages within 20 minutes on 2000 nodes.
 - Compares the configurations foreseen for a node with the actual configuration, and enforces coherence after every reboot.
 - It works with some limits (on RPM basis, not on file basis)
 - This is a configurable option
 - Difficult to use

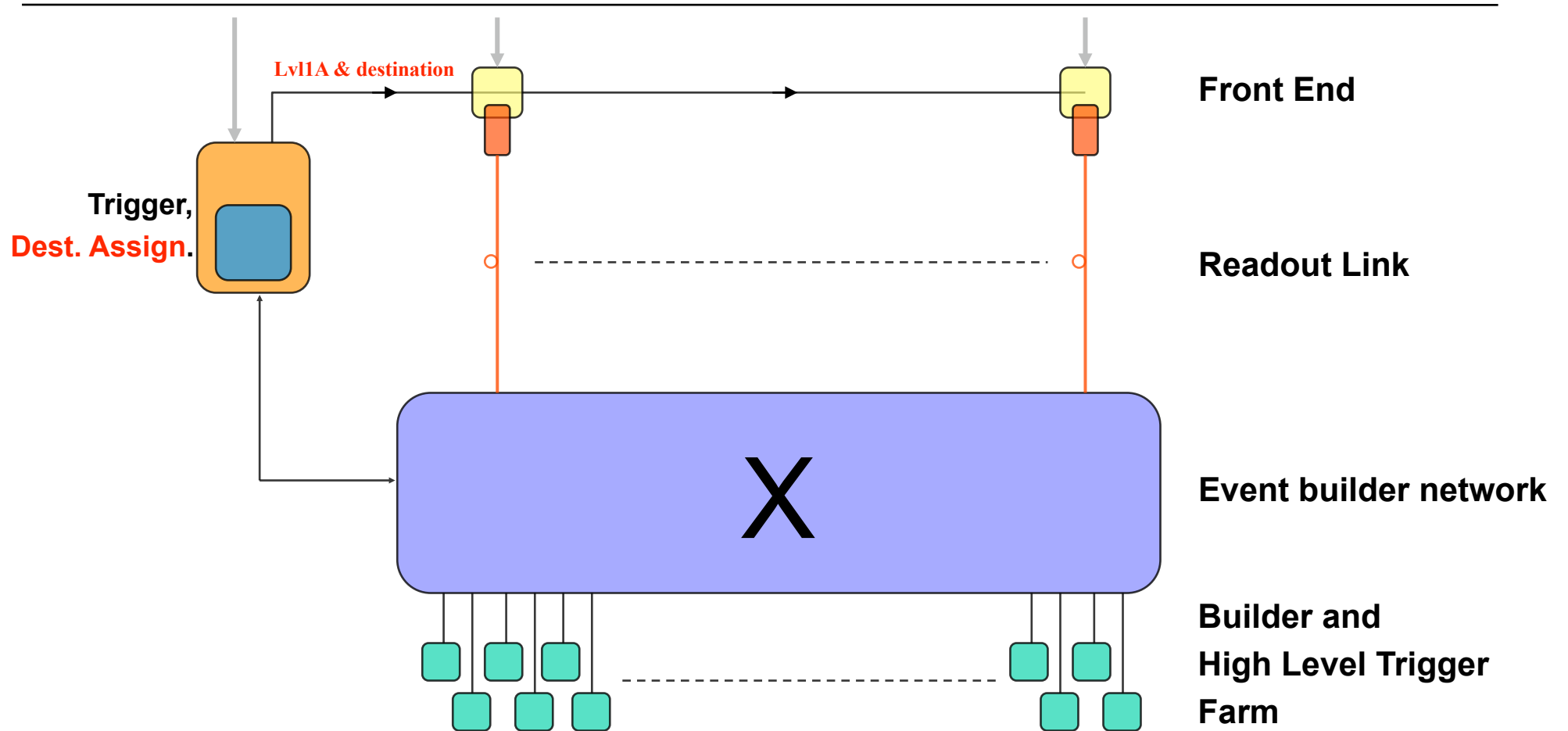
Event Builder Tasks

- **Synchronization of fragments**
 - Fragments of the same BX can be assembled into event
 - Possible means to achieve this
 - Trigger Number
 - Bunch counter (no overflow)
 - Time-stamp (resolution of time needs to match machine parameters)
- **Destination assignment**
 - Event fragments of one interaction need to be sent to the same destination in order to assemble the event
 - Possible implementations:
 - An “Event Manager” is distributing destinations for events identified by its Trigger Number
 - Use “static” destination assignment based on Trigger Number
 - Use “static” destination assignment based on BX Number
 - Use algorithm based on time stamp

Cont'ed: Event Builder Tasks

- Load balancing for destination units:
 - Needed optimally exploit the available resources
 - Needs a “feed back” system:
 - Work load of destinations must be fed back to Event Manager:
 - Possible Implementations:
 - “Send on demand”: Destination assignment is done as a function of a requests from destination
 - Static destination assignment: destination tables need to be updated if destinations start to become idle.

Future EVB architecture



Future EVB architecture wo trigger III

