DAQ

EIROForum School of Instrumentation 2015

Rainer Schwemmer, CERN-PH

Outline

- Introduction
  - Data acquisition
  - The first data acquisition campaign
- A simple DAQ system
  - One sensor
  - More and more sensors
- Read-out with buses
  - Crates & Mechanics
  - The VME Bus
- A DAQ for a large experiment
  - Sizing it up
  - Trigger
  - ~~Front-end Electronics~~
  - Readout with networks
    - Event building in switched networks
    - Problems in switched networks

Disclaimer

- Electronics, Trigger and DAQ are vast subjects covering a lot of physics and engineering
- Based entirely on personal bias I have selected a few topics
- While most of it will be only an overview at a few places we will go into some technical detail
- Some things will be only touched upon or left out altogether – information on those you will find in the references at the end
  - Quantitative treatment of detector electronics & physics behind the electronics
  - Derivation of the "physics" in the trigger → field theory lectures
  - DAQ of experiments outside HEP/LHC
  - Management of large networks and farms &High-speed mass storage

Thanks

- This Lecture is a part of the Summer Student Lectures on Electronics, Trigger and DAQ
- Lots of material and lots of inspiration for this lecture was taken from these lectures by: P. Mato, P. Sphicas, J. Christiansen, Niko Neufeld

Front-End Electronics
Physicists stop reading here

- It is well known that

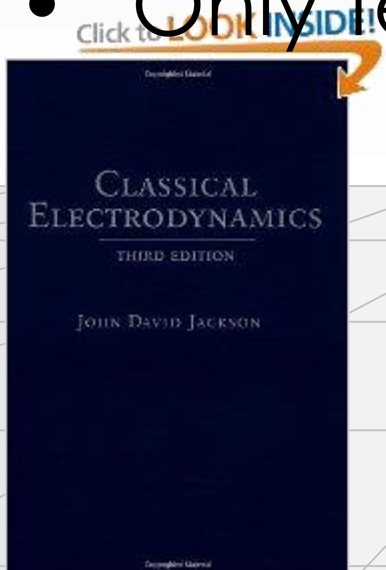
$$d\mathbf{F} = 0$$

$$d\mathbf{G} = \mathbf{J}$$

$$C : \Lambda^2 \ni \mathbf{F} \mapsto \mathbf{G} \in \Lambda^{(4-2)}$$

- "Only technical details are missing"

Werner Heisenberg, 1958



A physicist is someone who learned
Electrodynamics from Jackson

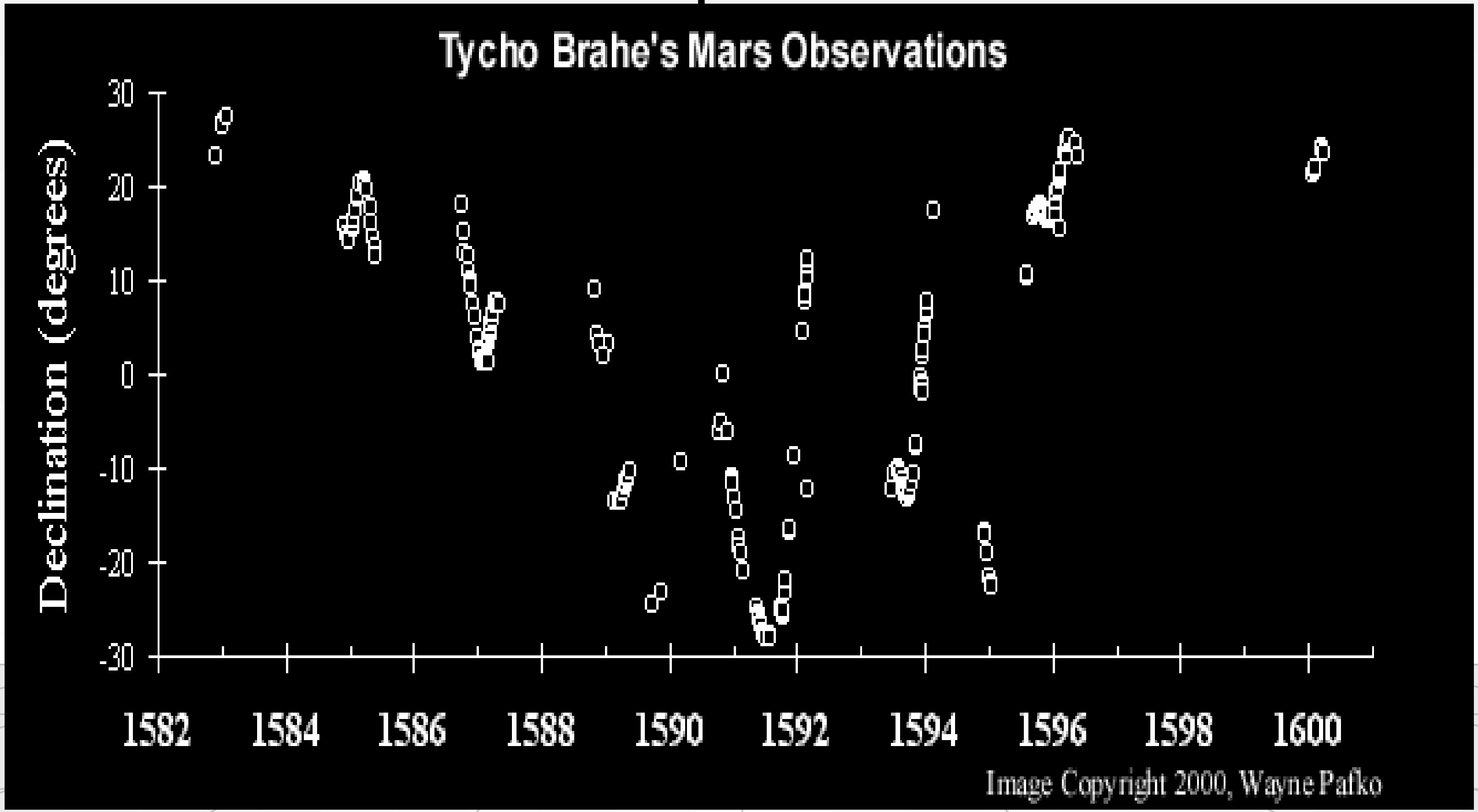
Tycho Brahe and the Orbit of Mars

I've studied all available charts of the planets and stars and none of them match the others. There are just as many measurements and methods as there are astronomers and all of them disagree. What's needed is a long term project with the aim of mapping the heavens conducted from a single location over a period of several years.

Tycho Brahe, 1563 (age 17).

- First measurement campaign
- Systematic data acquisition
  - Controlled conditions (same time of the day and month)
  - Careful observation of boundary conditions (weather, light conditions etc...) - important for data quality / systematic uncertainties

The First Systematic Data Acquisition



- Data acquired over 18 years, normally every month
- Each measurement lasted at least 1 hr with the naked eye
- Red line (only in the animated version) shows comparison with modern theory

Tycho's DAQ in Today's Terminology

- Bandwith (bw) = Amount of data transferred / per unit of time
  - "Transferred" = written to his logbook
  - "unit of time" = duration of measurement
  - bw_{Tycho} = ~ 100 Bytes / h (compare with LHCb 10.000.000.000.000 Bytes / s)
- Trigger = in general something which tells you when is the "right" moment to take your data
  - In Tycho's case the position of the sun, respectively the moon was the trigger
  - the trigger rate ~ 3.85 x 10^{-6} Hz (compare with LHCb 1.0 x 10^{6} Hz)

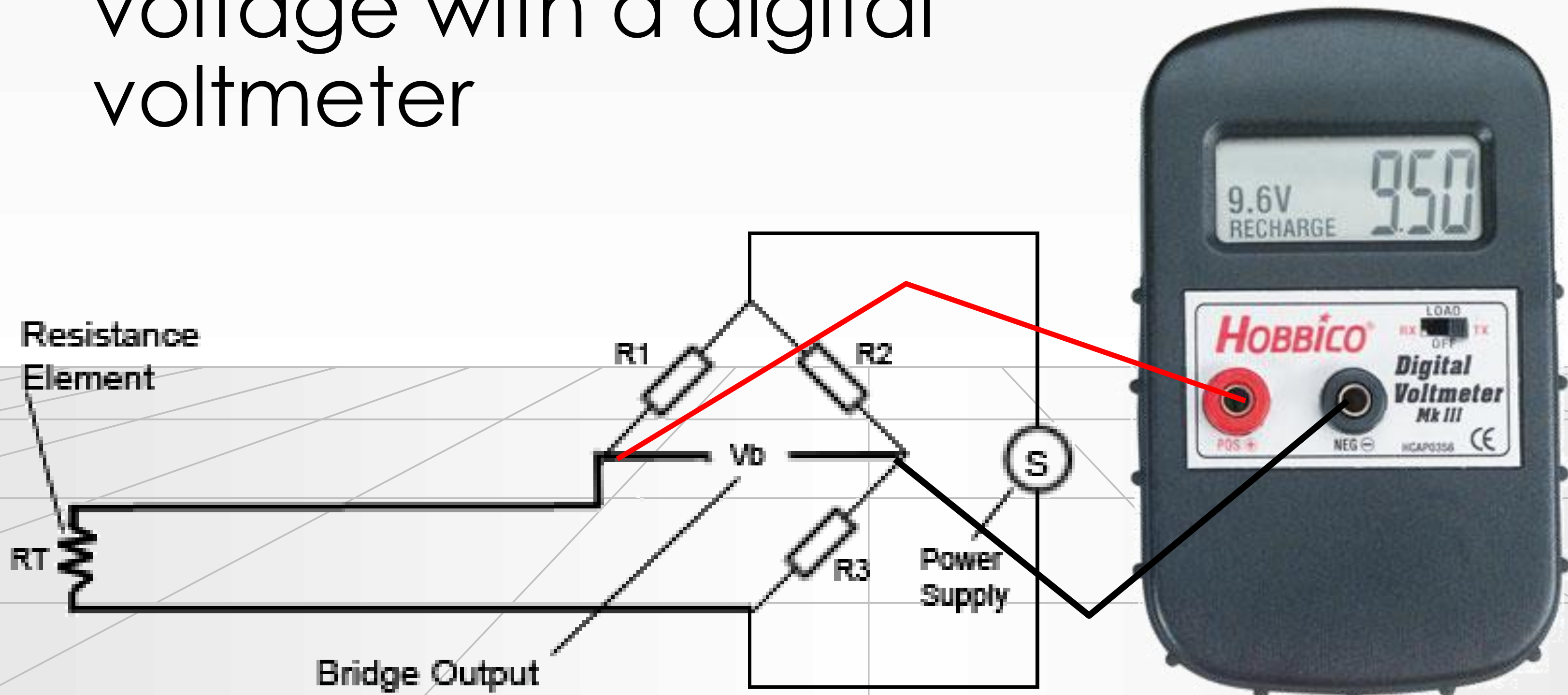
Some More Thoughts on Tycho

- Tycho did not do the correct analysis of the Mars data, this was done by Johannes Kepler (1571-1630), eventually paving the way for Newton's laws
- Morale: the size & speed of a DAQ system are not correlated with the importance of the discovery!

A Very Simple Data Acquisition System

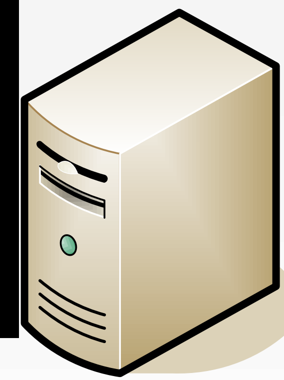
Measuring Temperature

- Suppose you are given a Pt100 thermo-resistor
- We read the temperature as a voltage with a digital voltmeter



Reading Out Automatically

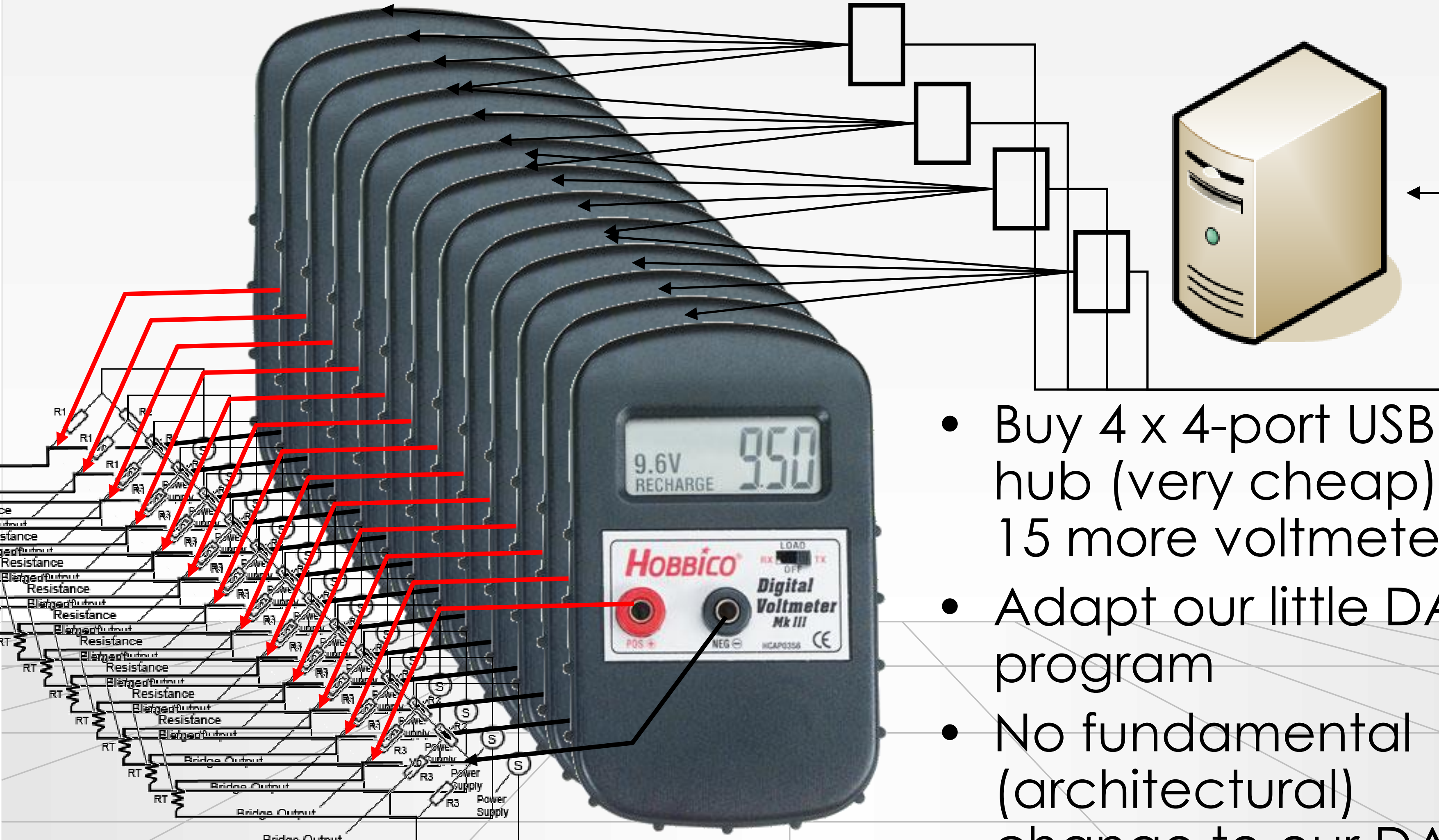
```
#include <libusb.h>
struct usb_bus *bus;
struct usb_device *dev;
usb_dev_handle *vmh = 0;
usb_find_busses(); usb_find_devices();
for (bus = usb_busses; bus; bus = bus->next)
          for (dev = bus->devices; dev; dev = dev-
>next)
                    if (dev->descriptor.idVendor ==
HOBBICO) vmh = usb_open(dev);
usb_bulk_read(vmh ,3,&u,sizeof(float),500);
```



Note how small the sensor has become.
In DAQ we normally need not worry about the details of the things we readout

USB/RS232

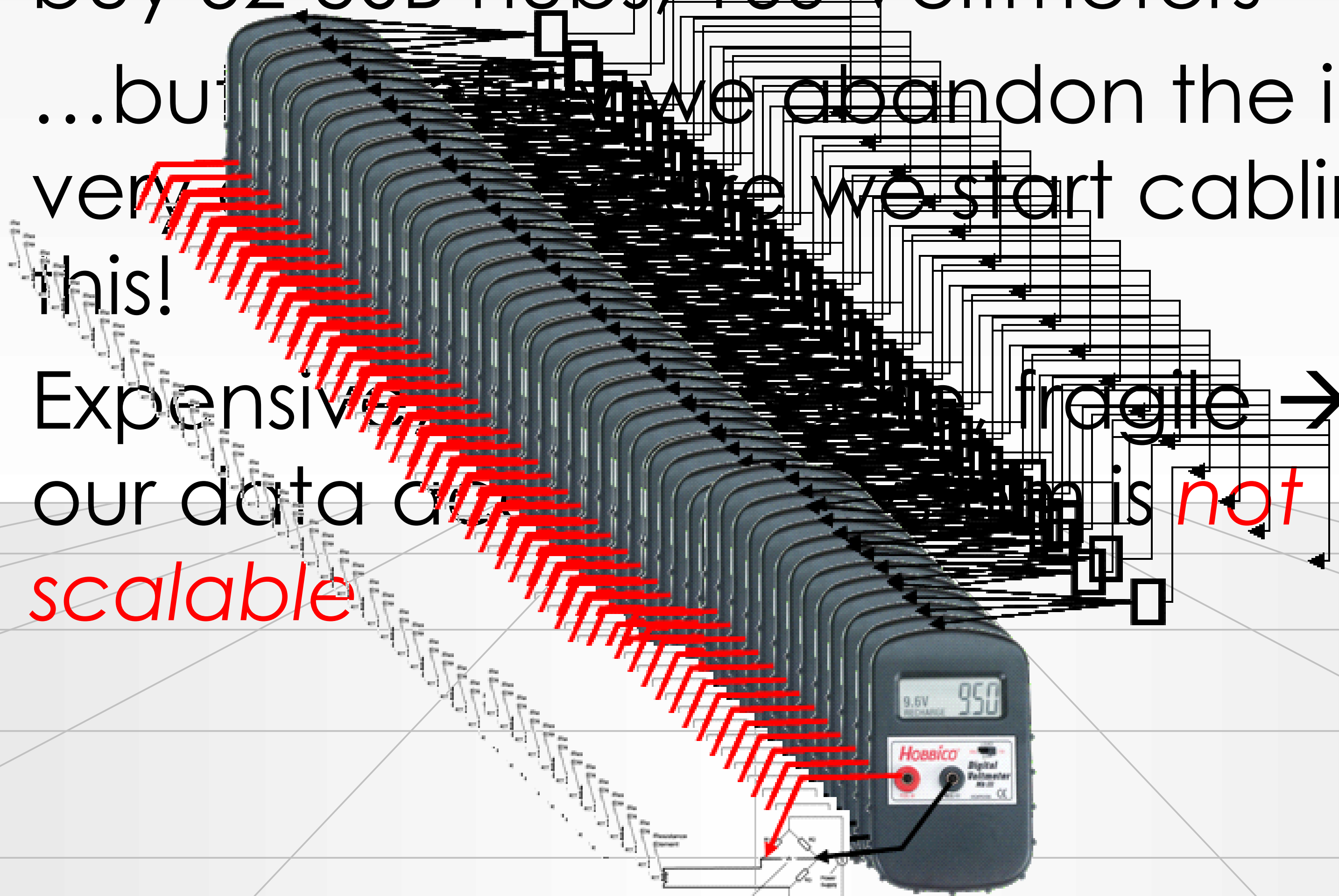
Read-out 16 Sensors



- Buy 4 x 4-port USB hub (very cheap) (+ 15 more voltmeters)
- Adapt our little DAQ program
- No fundamental (architectural) change to our DAQ

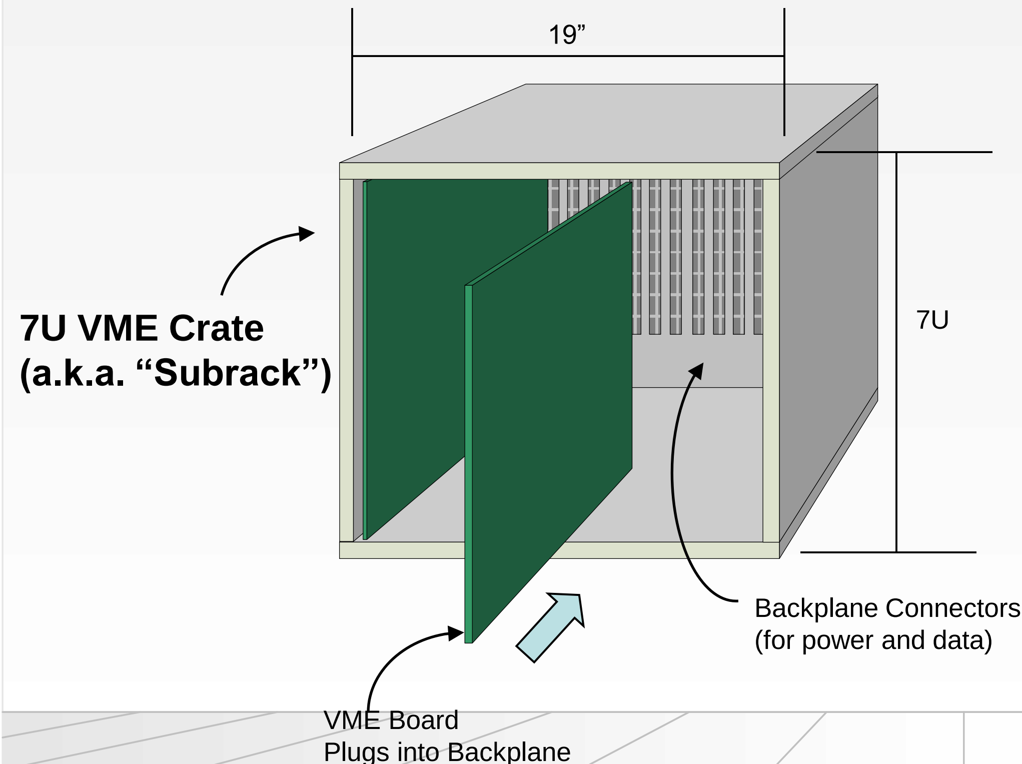
Read-out 160 Sensors

- For a moment we (might) consider to buy 52 USB hubs, 160 Voltmeters
- …but we abandon the idea very we start cabling this!
- Expensive, fragile → our data is *not scalable*



Read-out with Buses

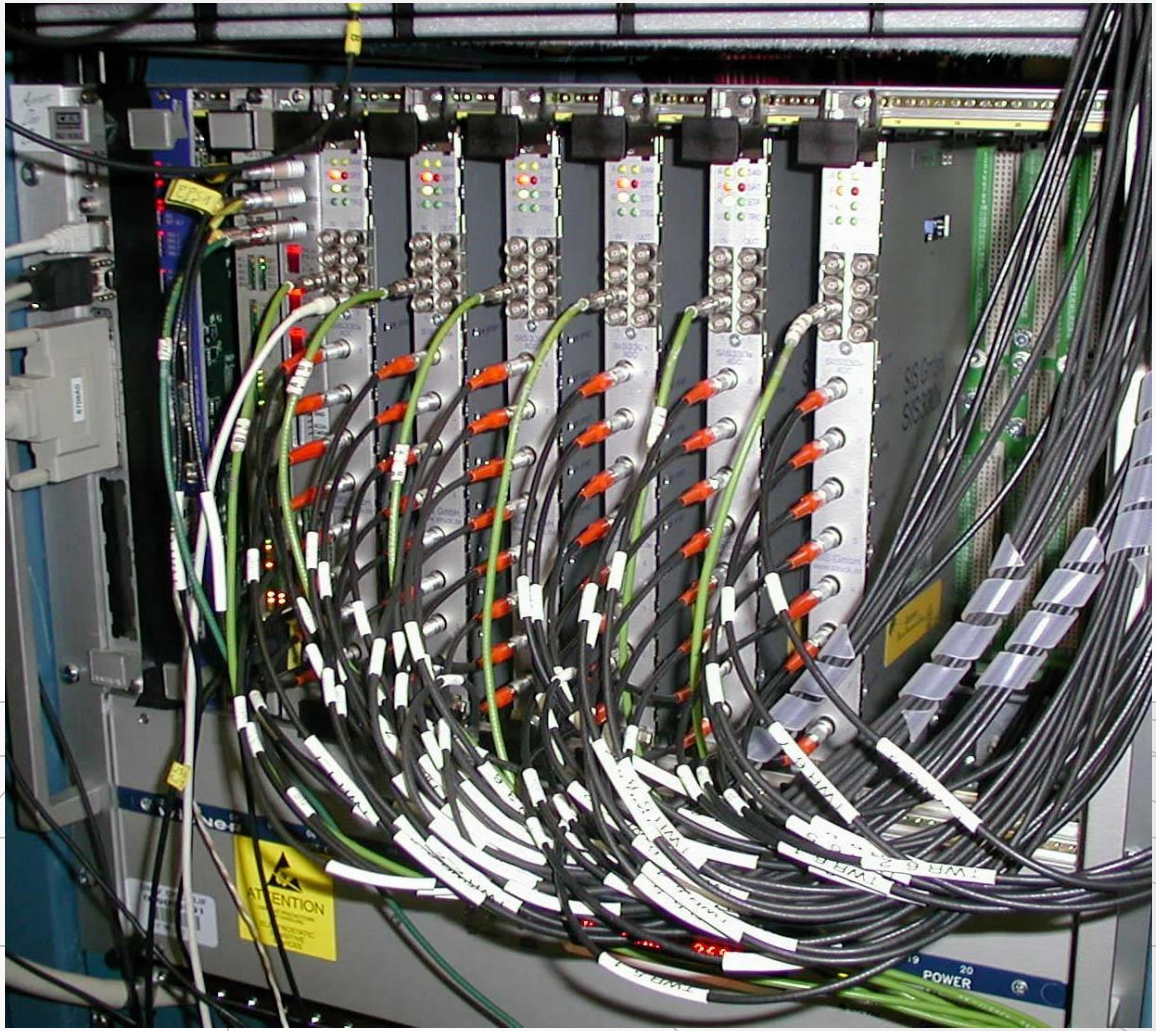
A Better DAQ for Many (temperature) Sensors



- Buy or build a compact multi-port volt-meter module, e.g. 16 inputs
- Put many of these multi-port modules together in a common chassis or crate
- The modules need
  - Mechanical support
  - Power
  - A standardized way to access their data (our measurement values)
- All this is provided by standards for (readout) electronics such as VME (IEEE 1014)

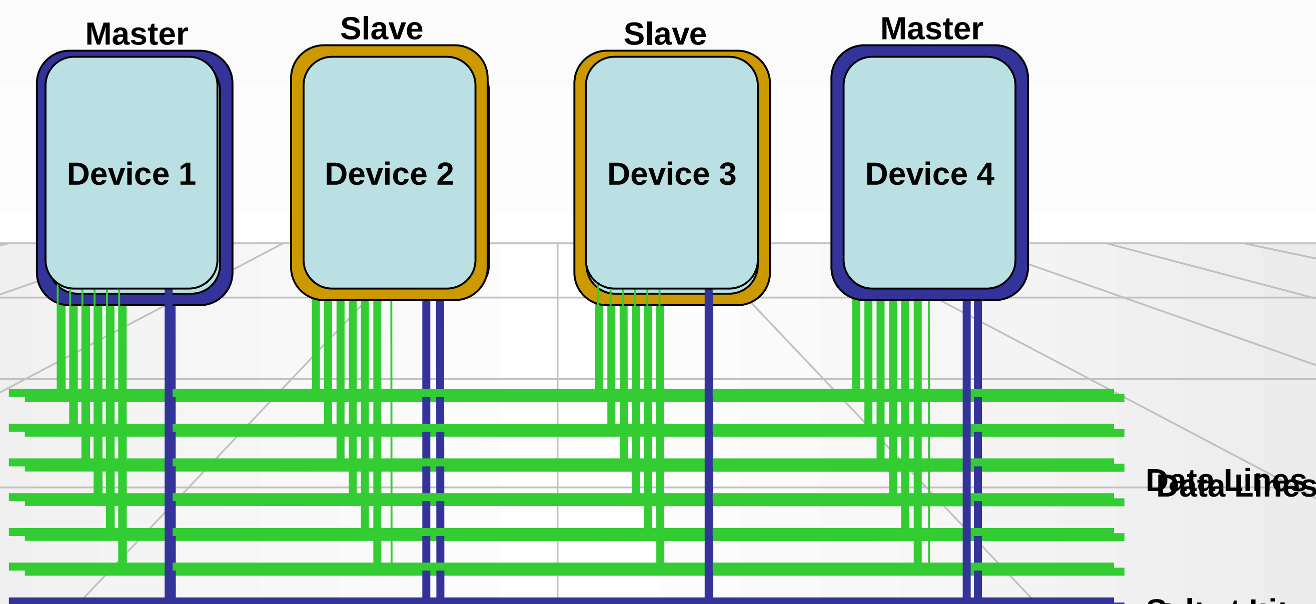
DAQ for 160 Sensors Using VME

- Readout boards in a *VME-crate*
  - mechanical standard for
  - electrical standard for power on the backplane
  - signal and protocol standard for communication on a *bus*



Communication in a Crate: Buses

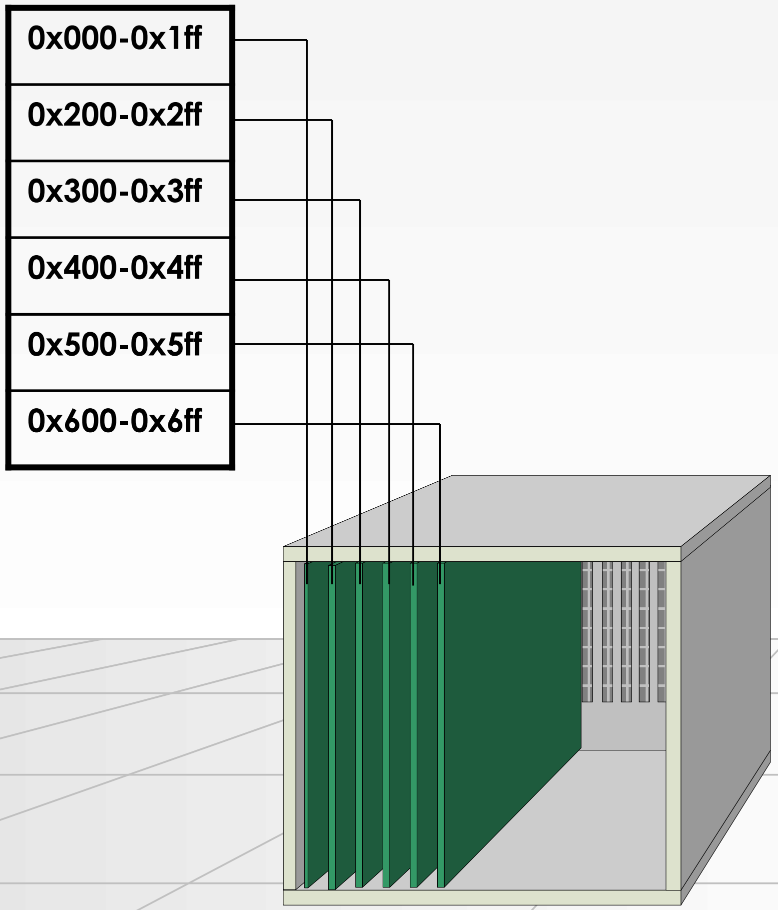
- A bus connects two or more devices and allows the to communicate
- The bus is shared between all devices on the bus → arbitration is required
- Devices can be masters or slaves (some can be both)
- Devices can be uniquely identified ("addressed") on the bus



Buses

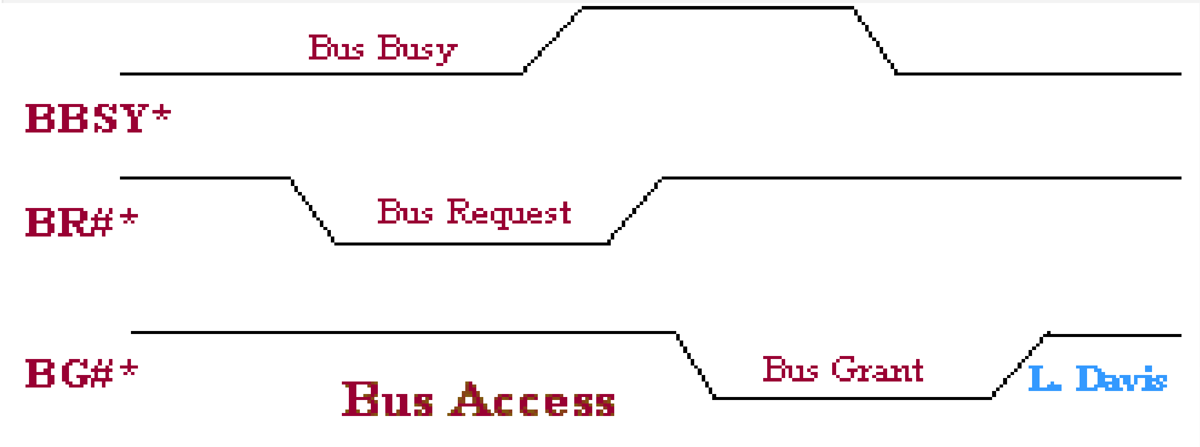
- Famous examples: PCI, USB, VME, SCSI
  - older standards: CAMAC, ISA
  - upcoming: ATCA
  - many more: FireWire, I2C, Profibus, etc…
- Buses can be
  - local: PCI
  - external peripherals: USB
  - in crates: VME, compactPCI, ATCA
  - long distance: CAN, Profibus

The VME Bus



- In a VME crate we can find three main types of modules
  - The controller which monitors and arbitrates the bus
  - Masters read data from and write data to slaves
  - Slaves send data to and receive data from masters
- Addressing of modules
  - In VME each module occupies a part of a (flat) range of addresses (24 bit to 32 bit)
  - Address range of modules is hardwired (conflicts!)

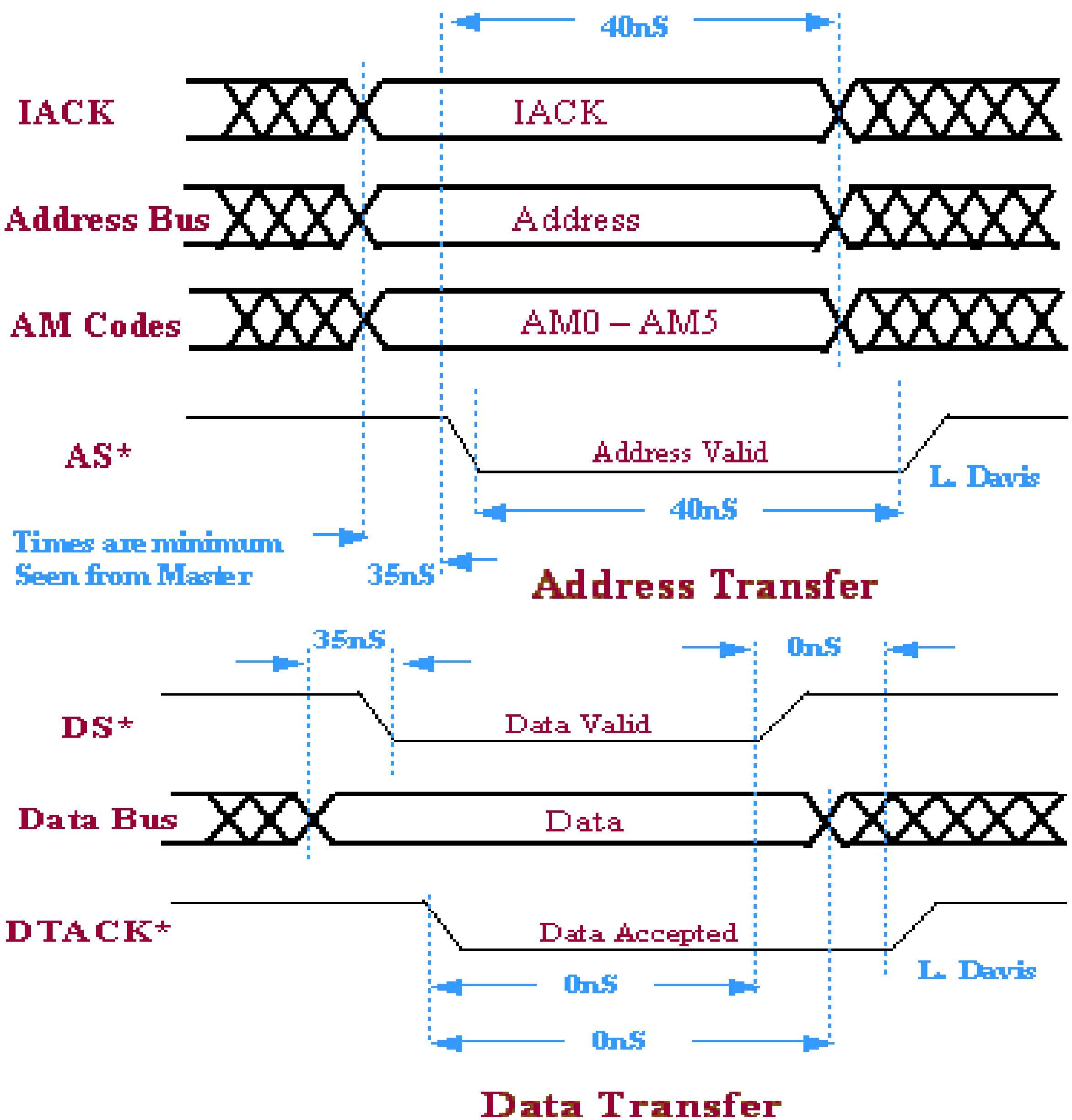
VME protocol 1) Arbitration



- Arbitration: Master asserts*) BR#, Controller answers by asserting BG#
- If there are several masters requesting at the same time the one physically closest to the controller wins
- The winning master drives BBSY* high to indicate that the bus is now in use

Pictures from http://www.interfacebus.com
*) assert means driving the line to logical 0 (VME control lines are inverted or active-low)

VME protocol 2) Write transfer

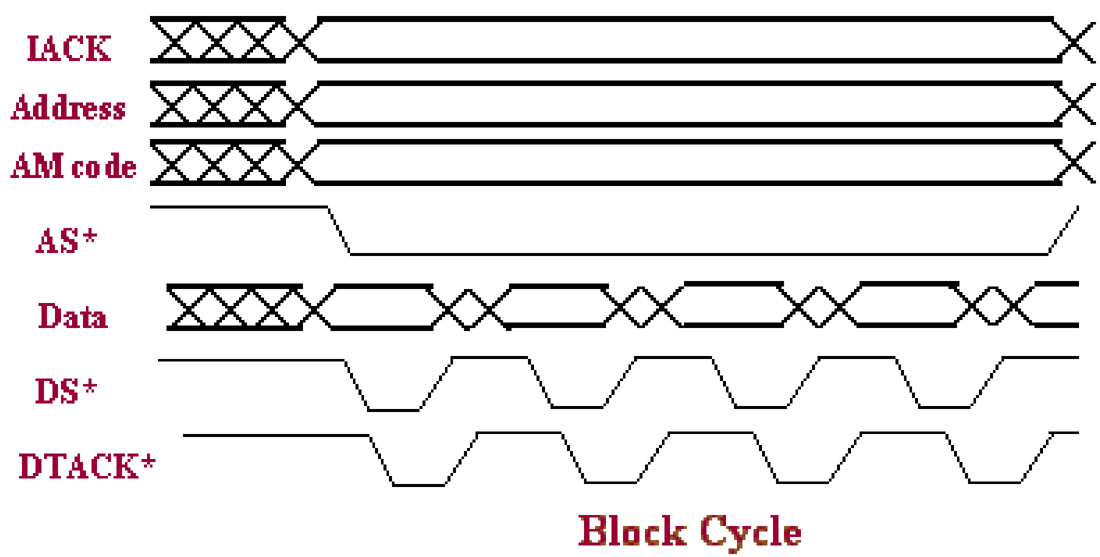


- The Master writes data and address to the data / respectively data bus
- It asserts DS* and AS* to signal that the data and address are valid
- The slave reads and acknowledges by asserting DTACK
- The master releases DS*, AS* and BSBSY*, the cycle is complete
- Note: there is no clock! The slave can respond whenever it wants. VME is an asynchronous bus

Speed Considerations

- Theoretically ~ 16 MB/s can be achieved
  - assuming the databus to be full 32-bit wide
  - the master never has to relinquish bus master ship
- Better performance by using block-transfers

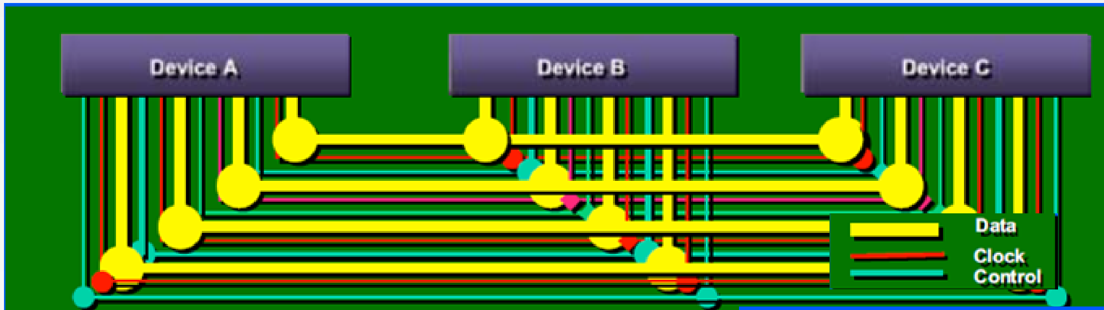
VME protocol 3) Block transfer

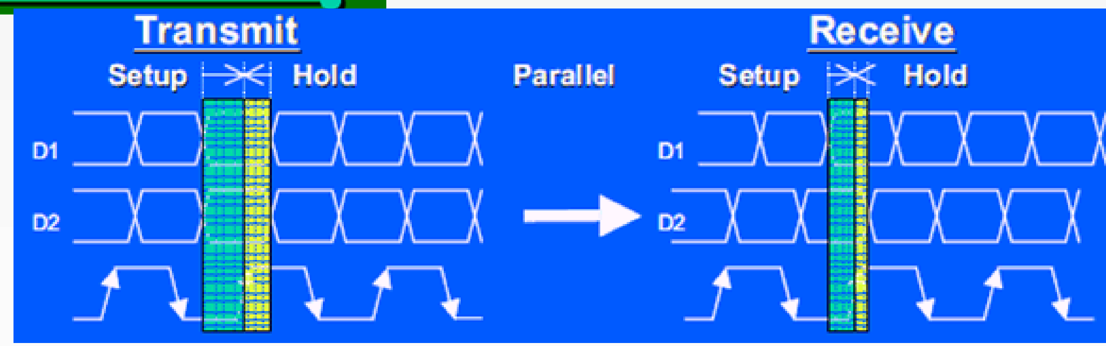


- Block transfers are essential for Direct Memory Access (DMA)
- More performance can be gained by using the address bus also for data (VME64)

- After an address cycle several (up to 256) data cycles are performed
- The slave is supposed to increment the address counter
- The additional delays for asserting and acknowledging the address are removed
- Performance goes up to 40 MB/s
- In PCI this is referred to as "burst-transfer"

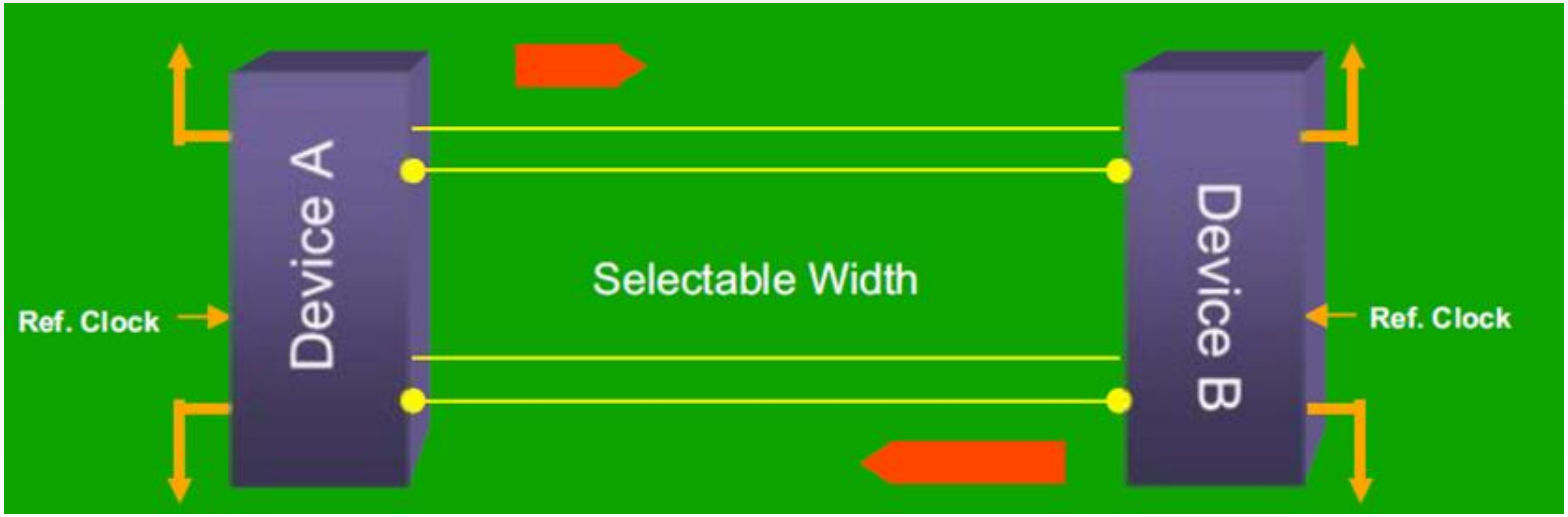
More Modern Busses PCI vs. PCIe





- Similar concept as VME
- Stringent routing requirements to ensure timing
- Not feasible at high speeds O(GB/s)

More Modern Busses PCI/PCIe



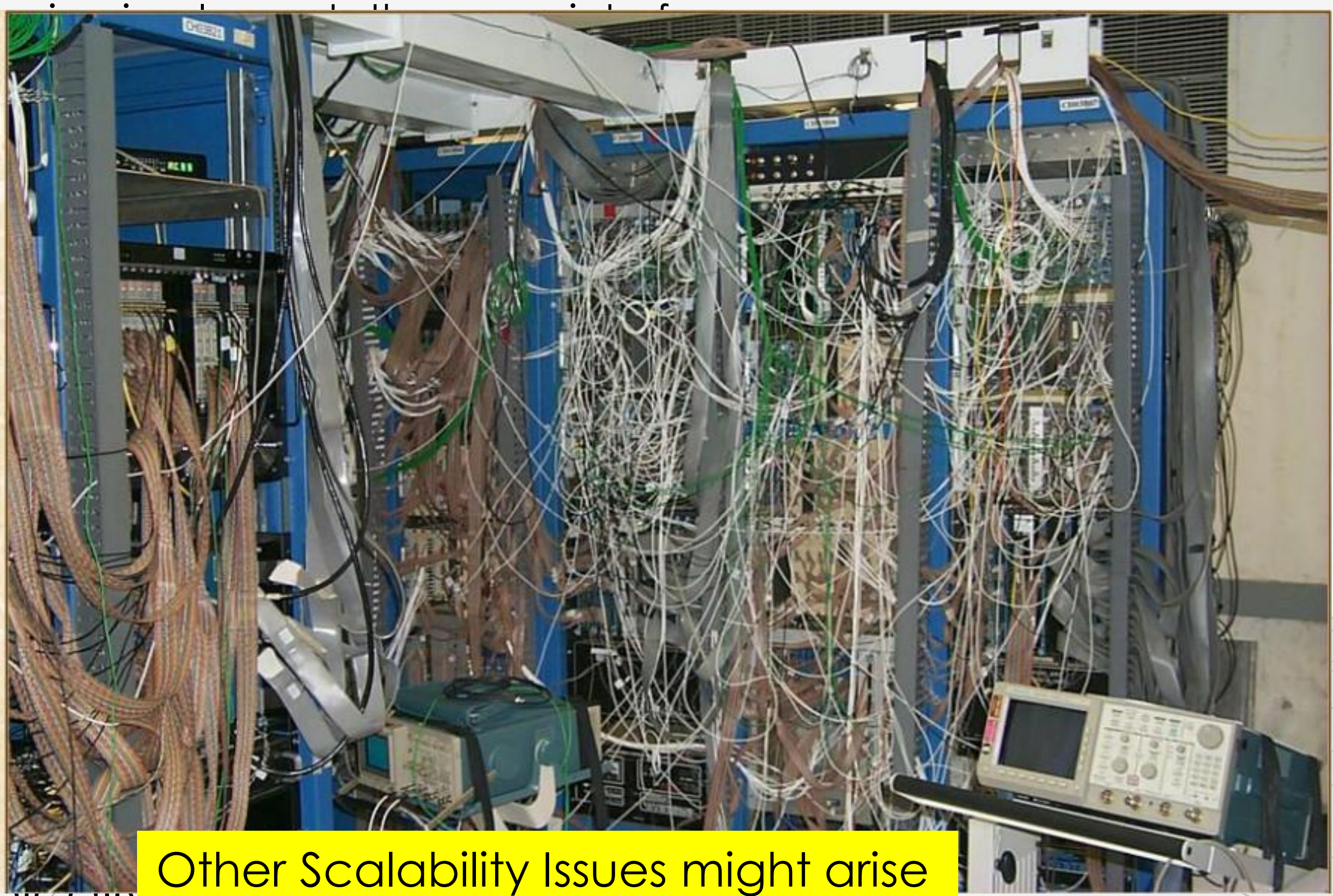
- Serial lane based "bus"
- Data is transmitted as messages/packets
- More like a switched network
- Multiple independent, serial lanes transmit data in parallel
  – Currently up to 32 lanes @ 8 Gbit/s

Buses Pros/Cons

- Pros:
  - Relatively simple to implement
    - Constant number of lines
    - Each device implements the same interface
  - Easy to add new devices
    - topological information of the bus can be used for automagically choosing addresses for bus devices: this is what plug and play is all about.
- Cons:
  - A bus is shared between all devices (each new active device slows everybody down)
    - Bus-width can only be increased up to a certain point (128 bit for PC-system bus)
    - Bus-frequency (number of elementary operations per second) can be increased, but decreases the physical bus-length
  - Number of devices and physical bus-length is limited (scalability!)
    - For synchronous high-speed buses, physical length is correlated with the number of devices (e.g. PCI)
    - Typical buses have a lot of control, data and address lines (look at a SCSI or ATA cable)
- Buses are typically useful for systems O(GB/s)

Buses Pros/Cons

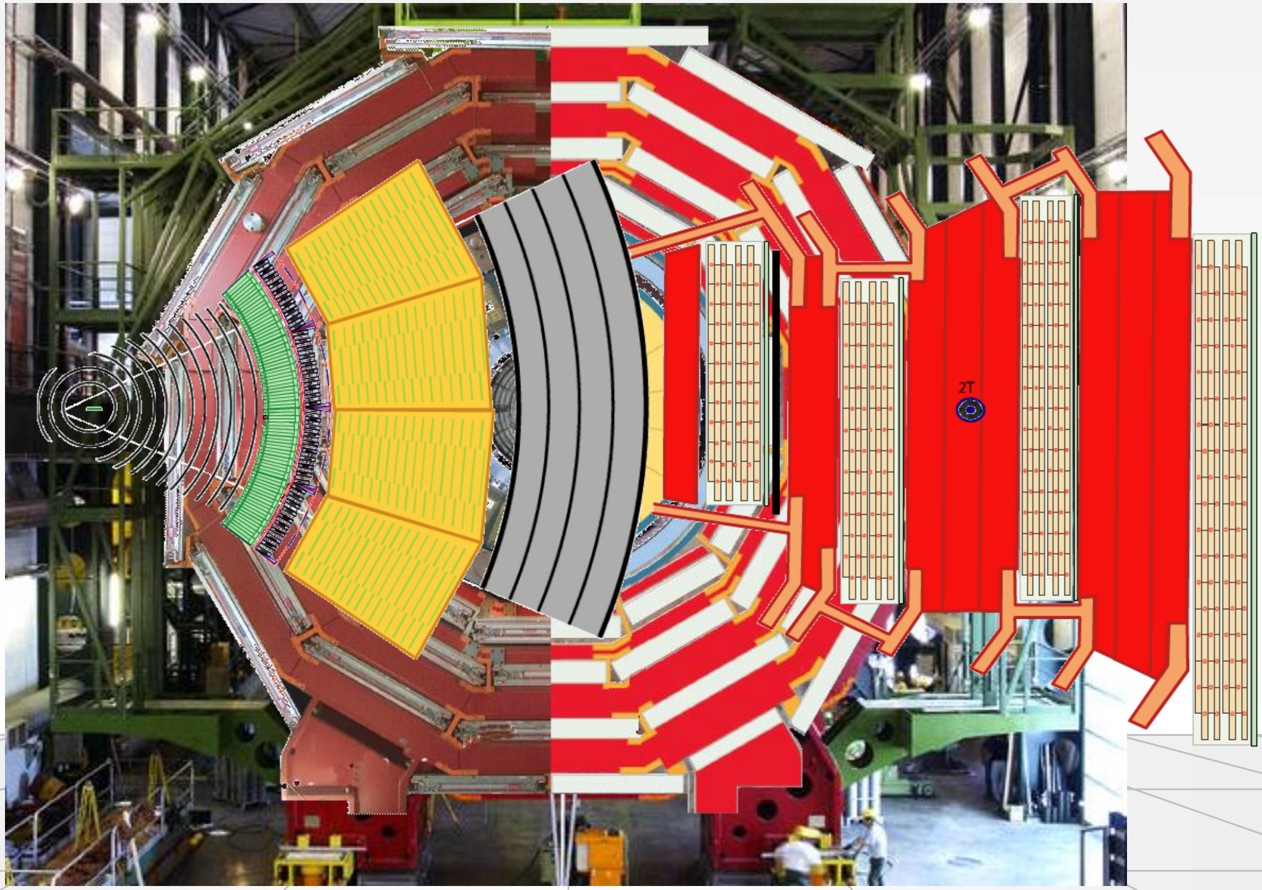
- Pro:
  - Relatively simple to implement
    - Constant number of lines
    - Each d
  - Easy to ac
    - topolog
      choosir
      about.
- Con:
  - A bus is sh
    slows ever
    - Bus-wid
      PC-syst
    - Bus-freq
      be incr
  - Number o
    (scalability
    - For sync
      with the
    - Typical
      a SCSI c
- Buses are typically



Other Scalability Issues might arise

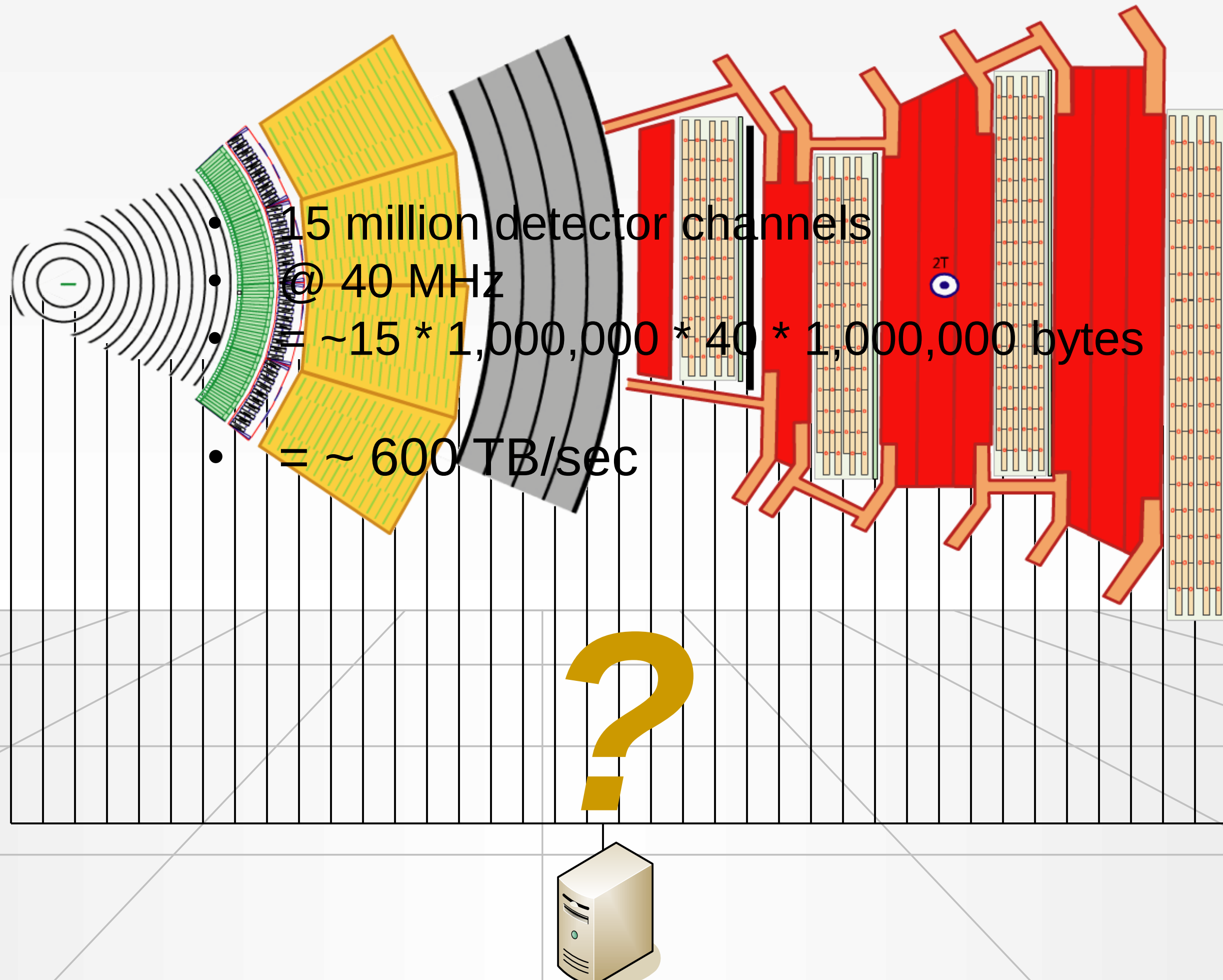
Data Acquisition for a Large Experiment

Moving on to Bigger Things…



The CMS Detector

Moving on to Bigger Things…



- 15 million detector channels
- @ 40 MHz
- = ~15 * 1,000,000 * 40 * 1,000,000 bytes
- = ~ 600 TB/sec

Designing a DAQ System for a Large (HEP) Experiment

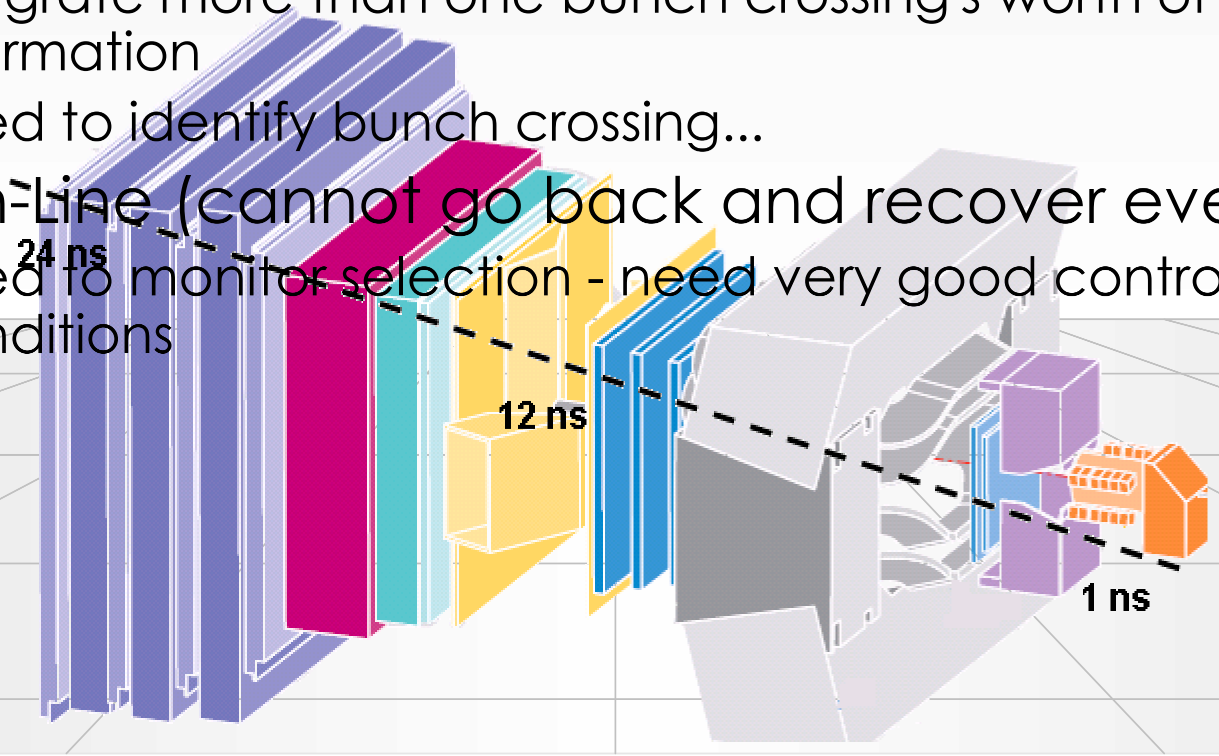
- What defines "large"?
  - The number of channels: for LHC experiments $O(10^7)$ channels
    - a (digitized) channel can be between 1 and 14 bits
  - The rate: for LHC experiments everything happens at 40.08 MHz, the LHC bunch crossing frequency (This corresponds to 24.9500998 ns or 25 ns among friends)
- HEP experiments usually consist of many different sub-detectors: tracking, calorimetry, particle-ID, muon-detectors

First Questions

- Can we or do we want to save all the data?
- How do we select the data
- Is continuous read-out needed, i.e. an experiment in a collider? Or are there idle periods mixed with periods with many events – this is typically the case for fixed-target experiments
- How do we make sure that the values from the many different channels refer to the same original event (collision)

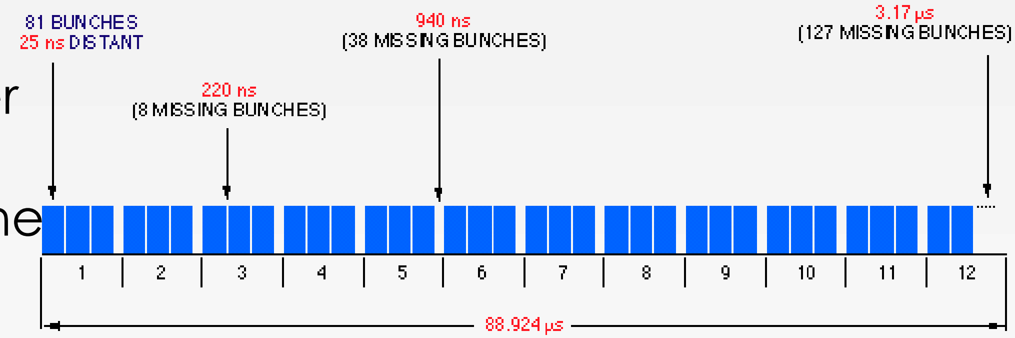
Challenges for the L1 at LHC

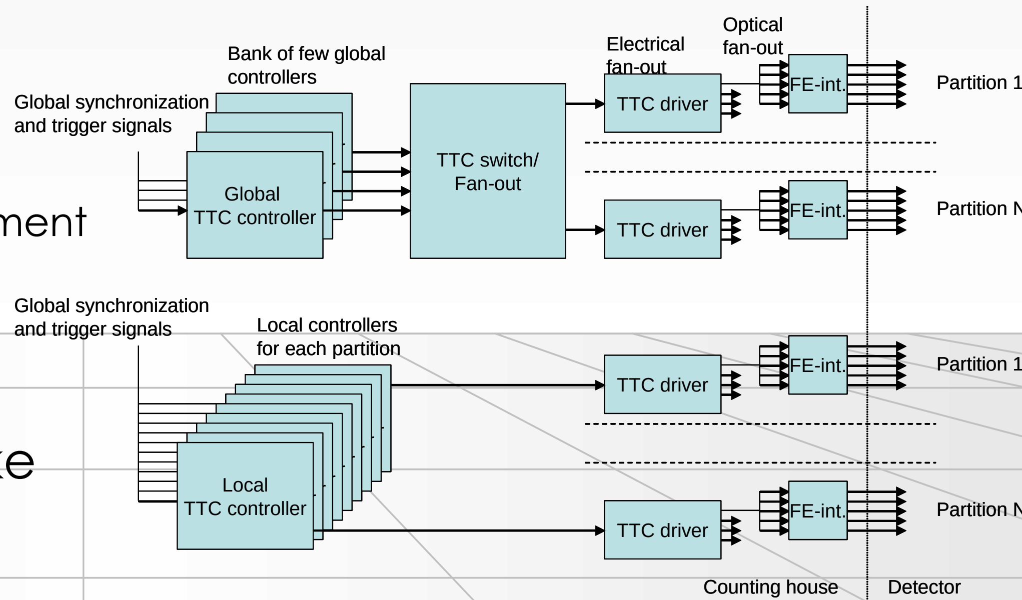
- N (channels) ~ O(10^7); ≈20 interactions every 25 ns
  - need huge number of connections
- Need to synchronize detector elements to (better than) 25 ns
- In some cases: detector signal/time of flight > 25 ns
  - integrate more than one bunch crossing's worth of information
  - need to identify bunch crossing...
- It's On-Line (cannot go back and recover events)
  - need to monitor selection - need very good control over all conditions



Timing & sync control

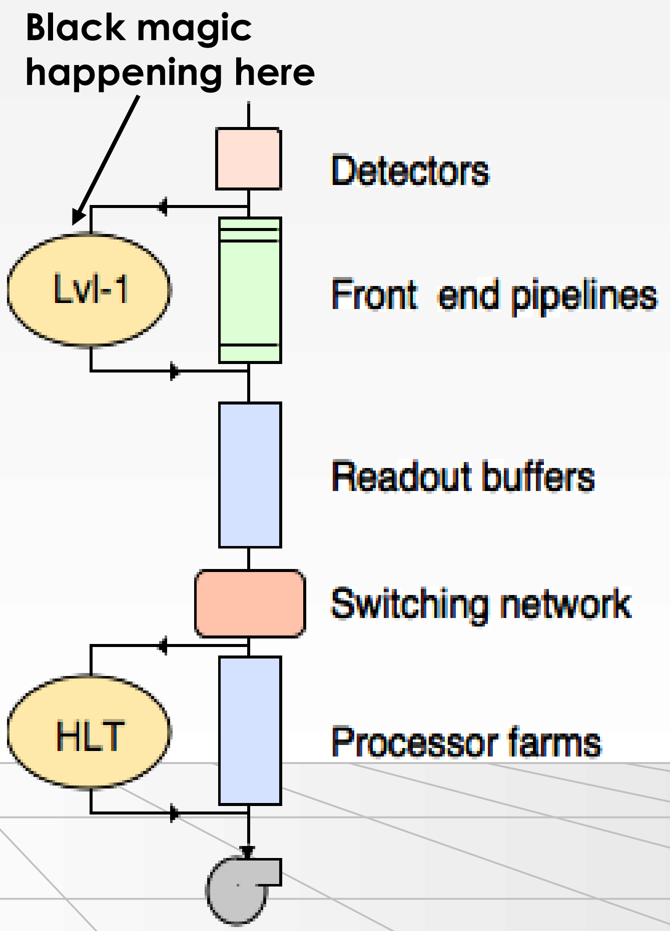
- Sampling clock with low jitter
- Synch reset
- Synchronization with machine bunch structure
- Calibration
- Trigger (with event type)
- Time align all the different sub-detectors and channels
  - Programmable delays
- Fan-out – unidirectional
  - Global fan-out to whole experiment or
  - Sub-detector fan-out
- Must be reliable as system otherwise may get de-synchronized which may take quite some time to correct



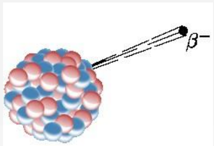


Trigger

- No (currently affordable) DAQ system could store O(10^7) channels at 40 MHz → 400 TBit/s to read out – even assuming binary channels!
- What's worse: most of these millions of events per second are totally uninteresting: one Higgs event every 0.02 seconds
- A *first level trigger (Level-1, L1)* must somehow select the more interesting events and tell us which ones to deal with any further

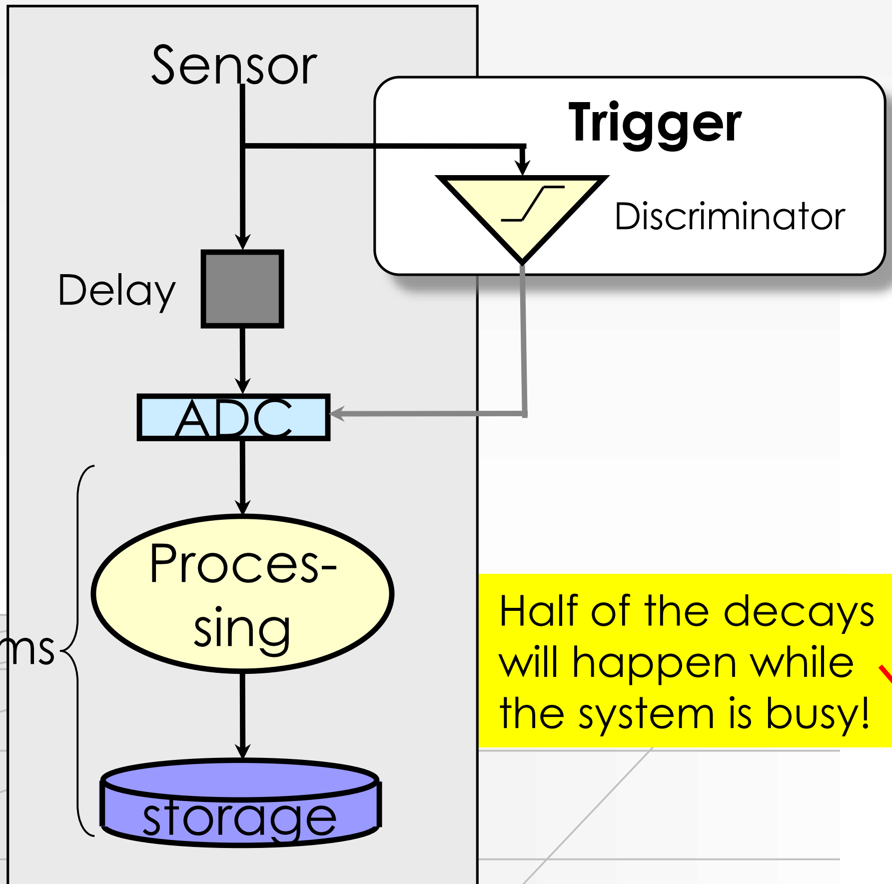


Beware the dead time!

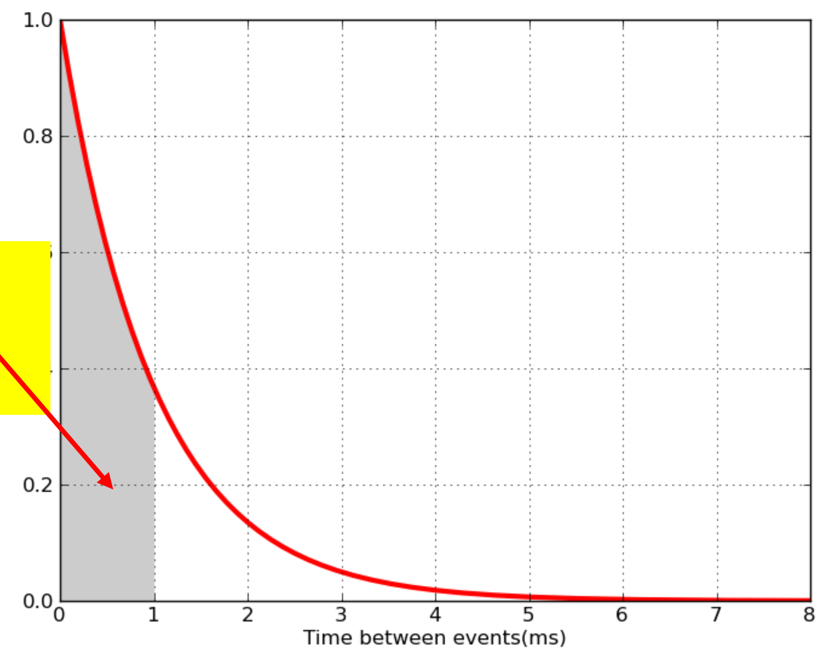


- Beta Decay @ 1kHz
- Processing takes 1 ms
- Stochastic process
- Can only capture 50% of events!

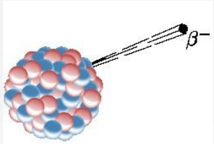
➔ Data needs to be buffered or Derandomized

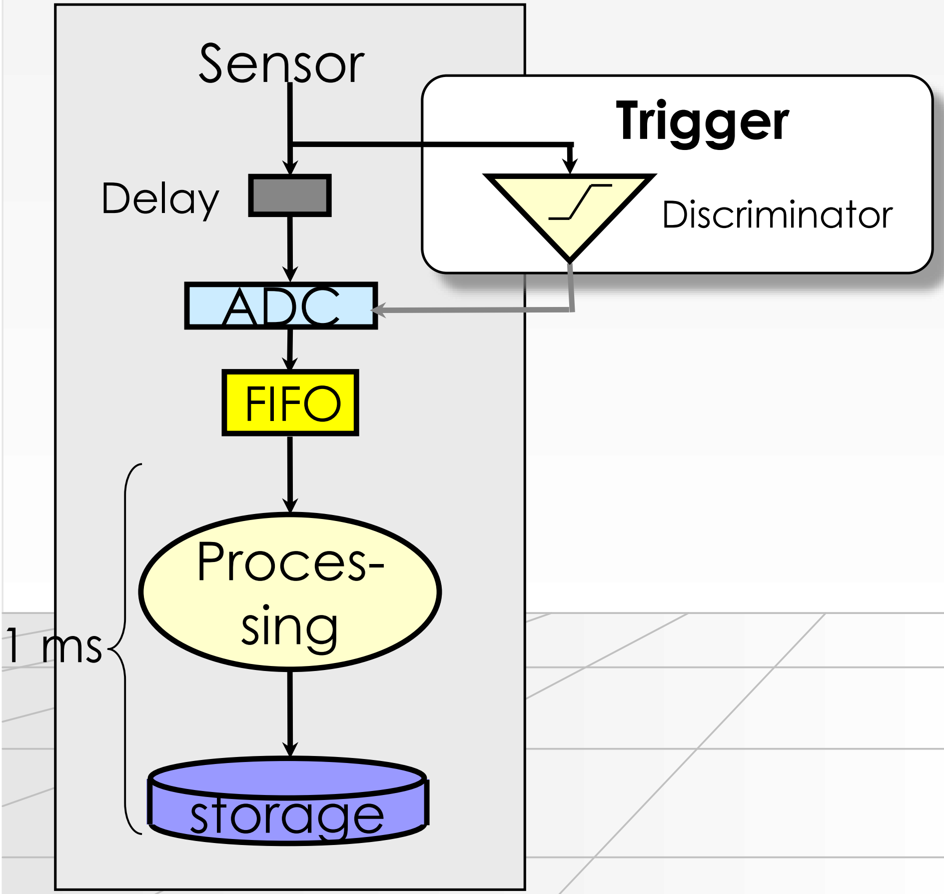


Half of the decays will happen while the system is busy!

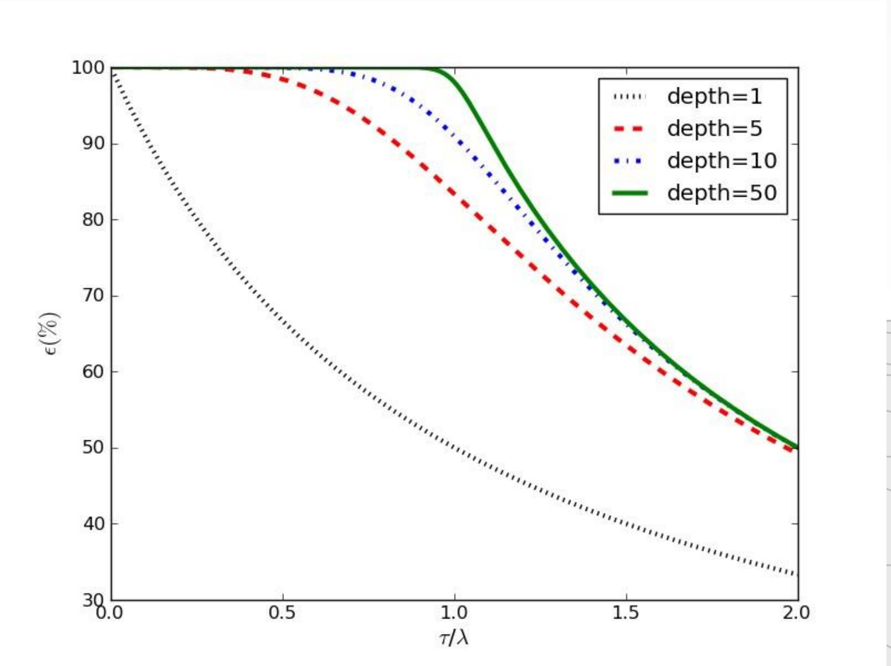


Derandomization/Queuing





- Add small buffer before the processing
- Processing requests data from FIFO when ready
- FIFO full → data still discarded
- Caveat: Memory in rad area is expensive



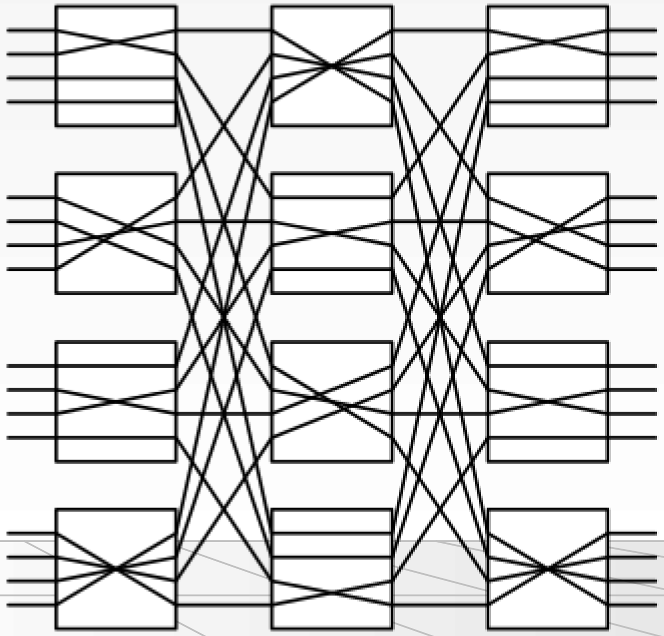
Large DAQ

Data Acquisition

- Event-data is digitized, pre-processed and tagged with a unique, monotonically increasing number
- The event data is distributed over many *read-out boards* ("sources")
- For the next stage of selection, or even simply to write it to tape we have to get the pieces together: enter the DAQ Network

Network based DAQ

- In large (HEP) experiments we typically have thousands of devices to read, which are sometimes very far from each other → *buses can not do that*
- Network technology solves the scalability issues of buses
  - In a network devices are equal ("peers")
  - In a network devices communicate directly with each other
    - no arbitration necessary
    - bandwidth guaranteed
  - data and control use the same path
    - much fewer lines (e.g. in traditional Ethernet only two)
  - At the signaling level buses tend to use parallel copper lines. Network technologies can be also optical, wire-less and are typically (differential) serial



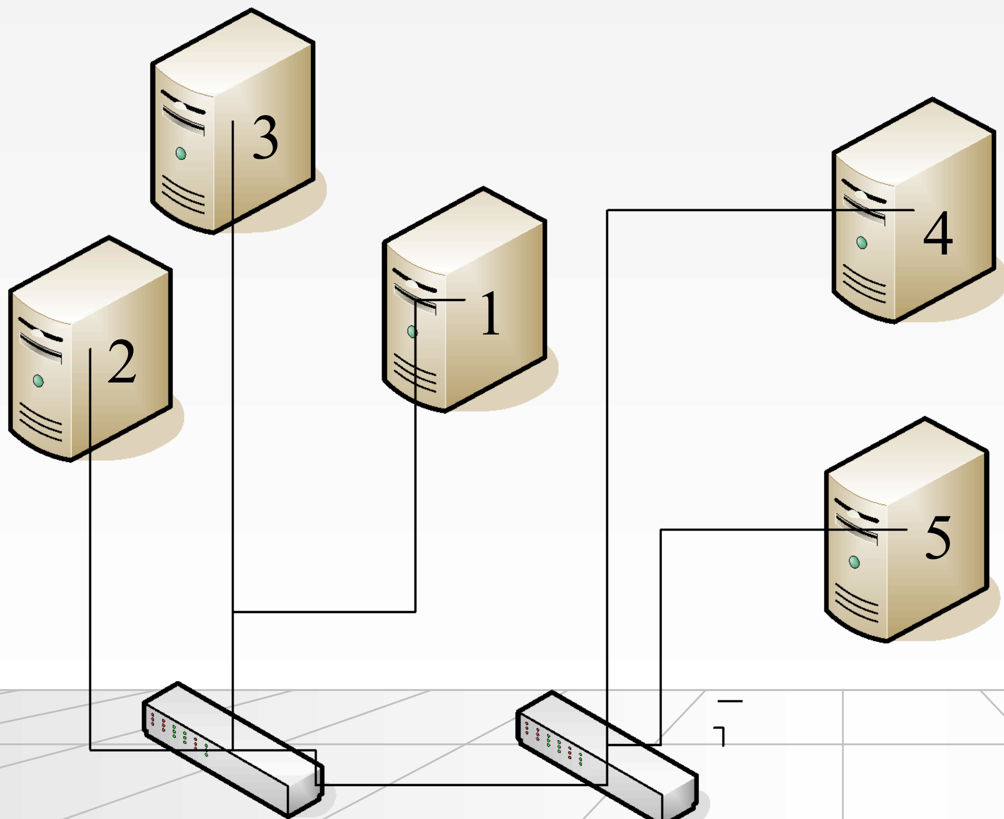
Network Technologies

- Examples:
  - The telephone network
  - Ethernet (IEEE 802.3)
  - ATM (the backbone for GSM cell-phones)
  - Infiniband
  - many, many more
- Note: some of these have "bus"-features as well (Ethernet, Infiniband)
- Network technologies are sometimes functionally grouped
  - Cluster interconnect (Myrinet, Infiniband) 15 m
  - Local area network (Ethernet), 100 m to 10 km
  - Wide area network (ATM, SONET) > 50 km

Connecting Devices in a Network

- On an network a device is identified by a network address
  - eg: our phone-number, the MAC address of your network card
- Devices communicate by sending messages (frames, packets) to each other
- Some establish a connection like the telephone network, some simply send messages
- Modern networks are *switched with point-to-point links*
  - circuit switching, packet switching

A Switched Network



- While 2 can send data to 1 and 4, 3 can send at full speed to 5
- 2 can distribute the bandwidth between 1 and 4 as needed

Switches

- Switches are the key to good network performance
- They must move frames reliably and as fast as possible between nodes
- They face two problems
  - Finding the right path for a frame
  - Handling congestion (two or more frames want to go to the same destination at the same time)

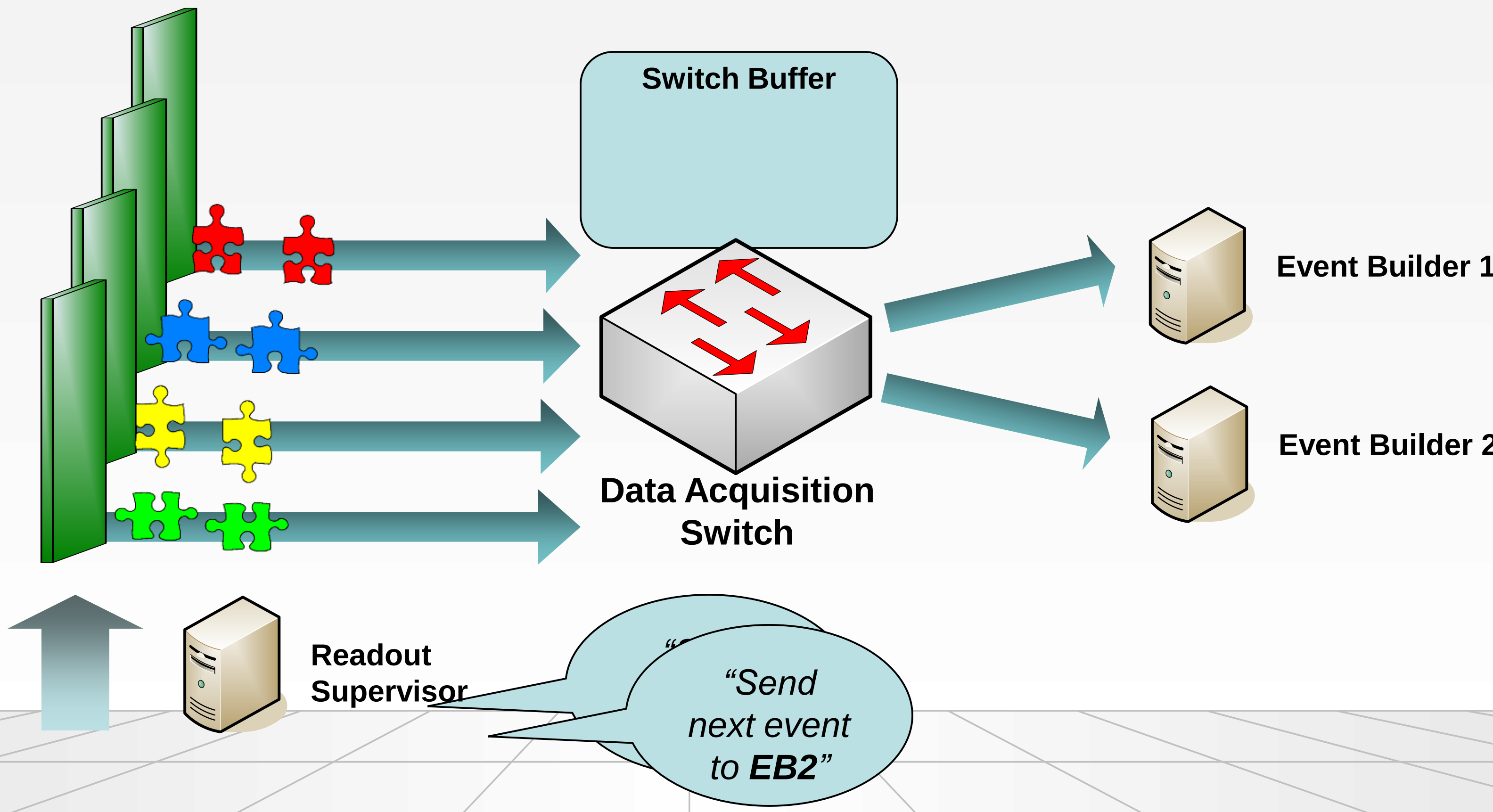
Ethernet

- Cheap
- Unreliable – but in practice transmission errors are very low
- Available in many different speeds and physical media
- We use IP or TCP/IP over Ethernet
- By far the most widely used local area network technology (even starting on the WAN)

Event Building

(putting it all back together)

Push-Based Event Building

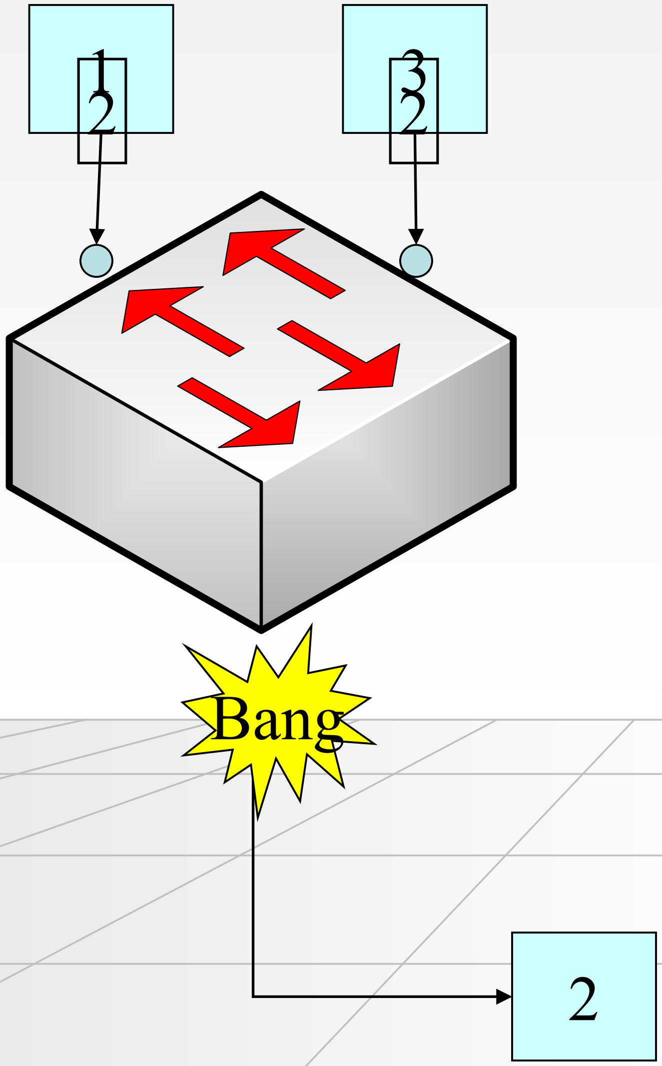


1 Readout Supervisor tells readout boards where events must be sent (round-robin)

2 Readout boards do not buffer, so switch must

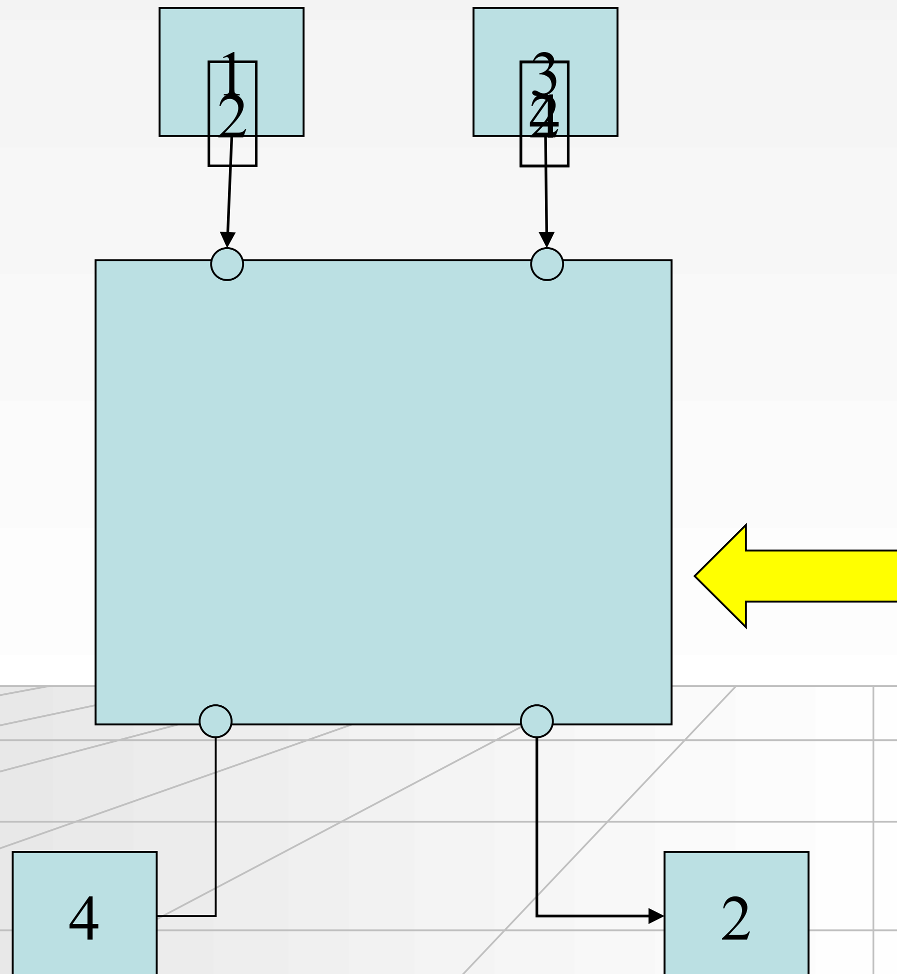
3 No feedback from Event Builders to Readout system

Congestion



- "Bang" translates into random, uncontrolled packet-loss
- In Ethernet this is perfectly valid behavior and implemented by very cheap devices
- Higher Level protocols are supposed to handle the packet loss due to *lack of buffering*
- This problem comes from ***synchronized*** sources ***sending*** to the same destination at the ***same time***

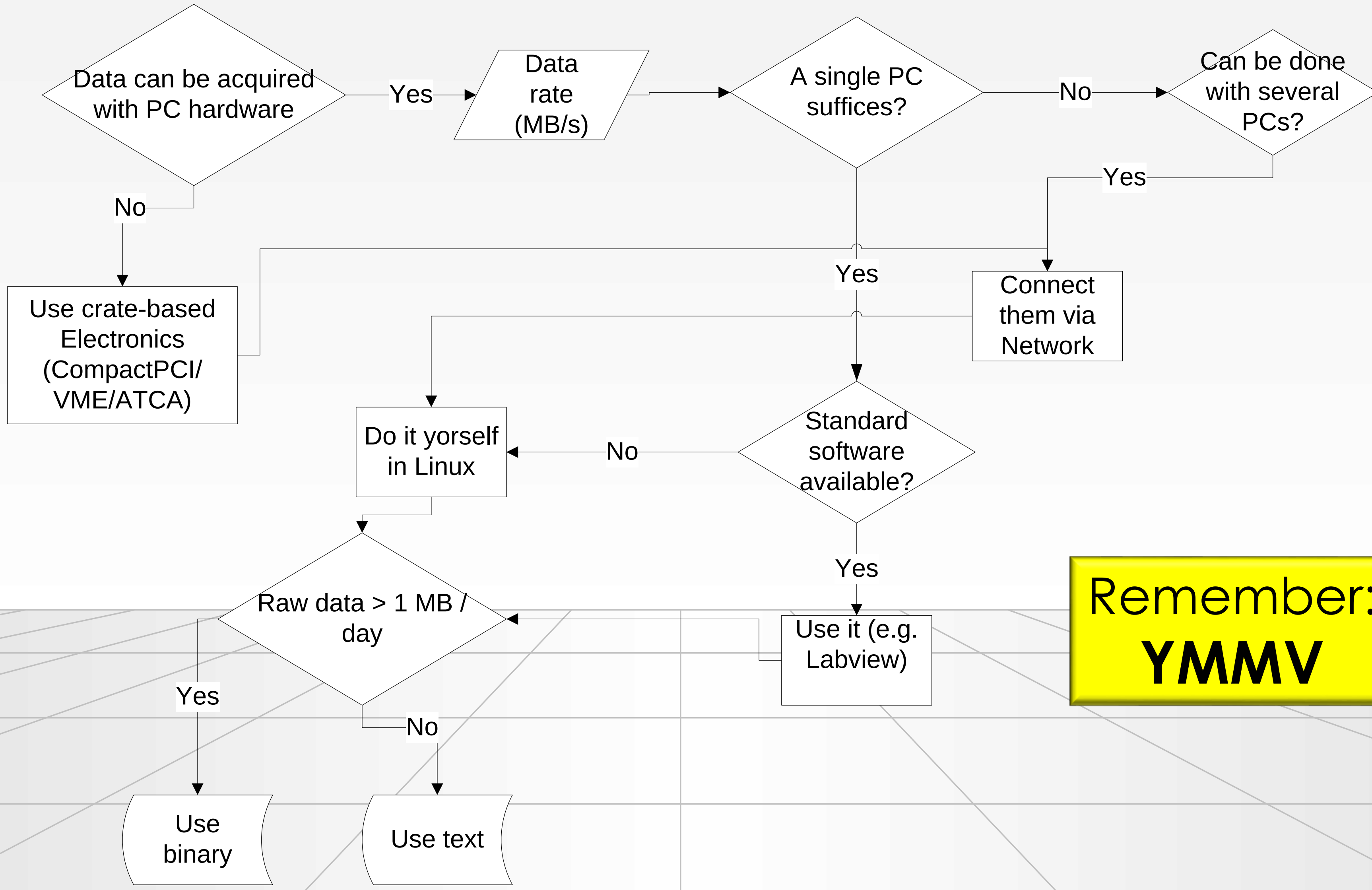
Overcoming Congestion



- In practice virtual output queueing is used: at each input there is a queue → for n ports $O(n^2)$ queues must be managed
- Assuming the buffers are large enough(!) such a switch will sustain random traffic at 100% nominal link load

Packet to node 2 waits at output to port 2. Way to node 4 is free

A little checklist for your DAQ



Remember: YMMV

The end

Further Reading

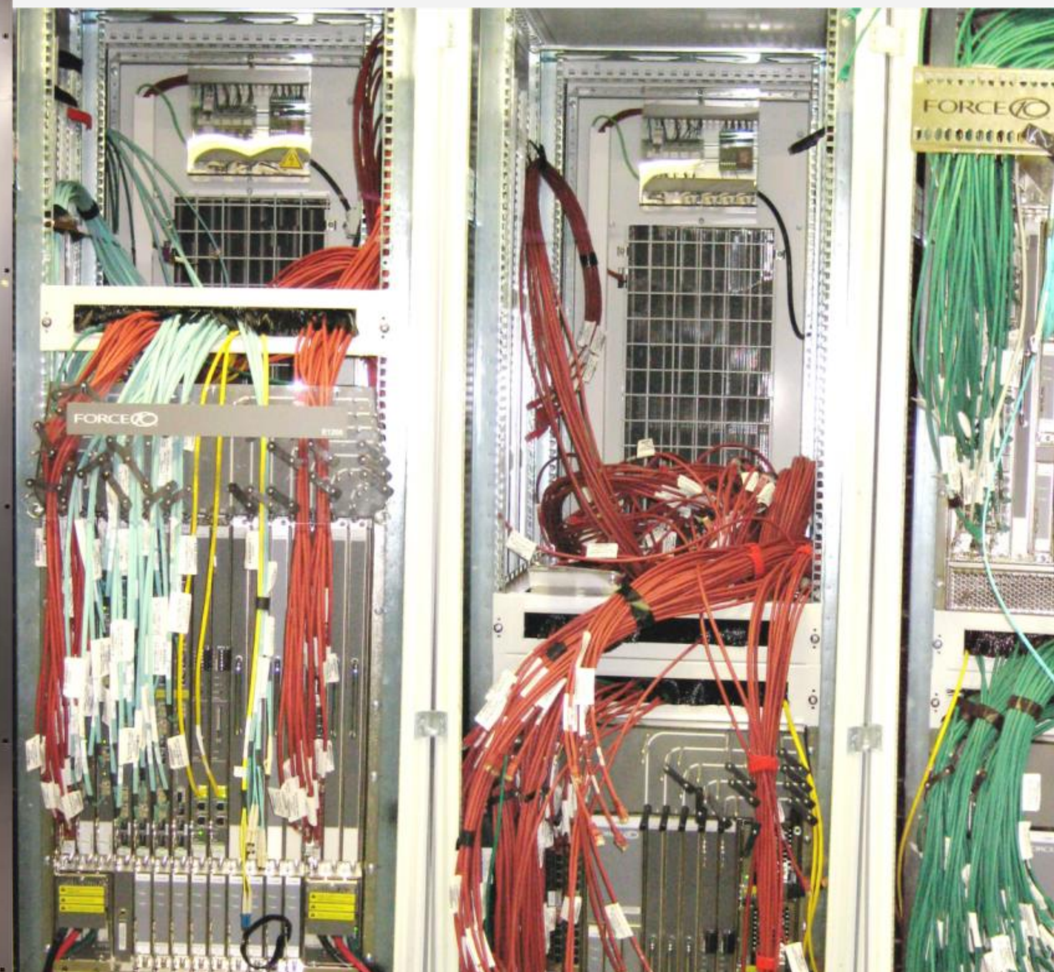
- Electronics
  - Helmut Spielers web-site: http://www-physics.lbl.gov/~spieler/
- Buses
  - VME: http://www.vita.com/
  - PCI http://www.pcisig.com/
- Network and Protocols
  - Ethernet "Ethernet: The Definitive Guide", O'Reilly, C. Spurgeon
  - TCP/IP "TCP/IP Illustrated", W. R. Stevens
  - Protocols: RFCs www.ietf.org in particular RFC1925 http://www.ietf.org/rfc/rfc1925.txt "The 12 networking truths" is required reading
- Wikipedia (!!!) and references therein – for all computing related stuff this is usually excellent
- Conferences
  - IEEE Realtime
  - ICALEPCS
  - CHEP
  - IEEE NSS-MIC
- Journals
  - IEEE Transactions on Nuclear Science, in particular the proceedings of the IEEE Realtime conferences
  - IEEE Transactions on Communications

Backup

Gallery

ALICE Storage System

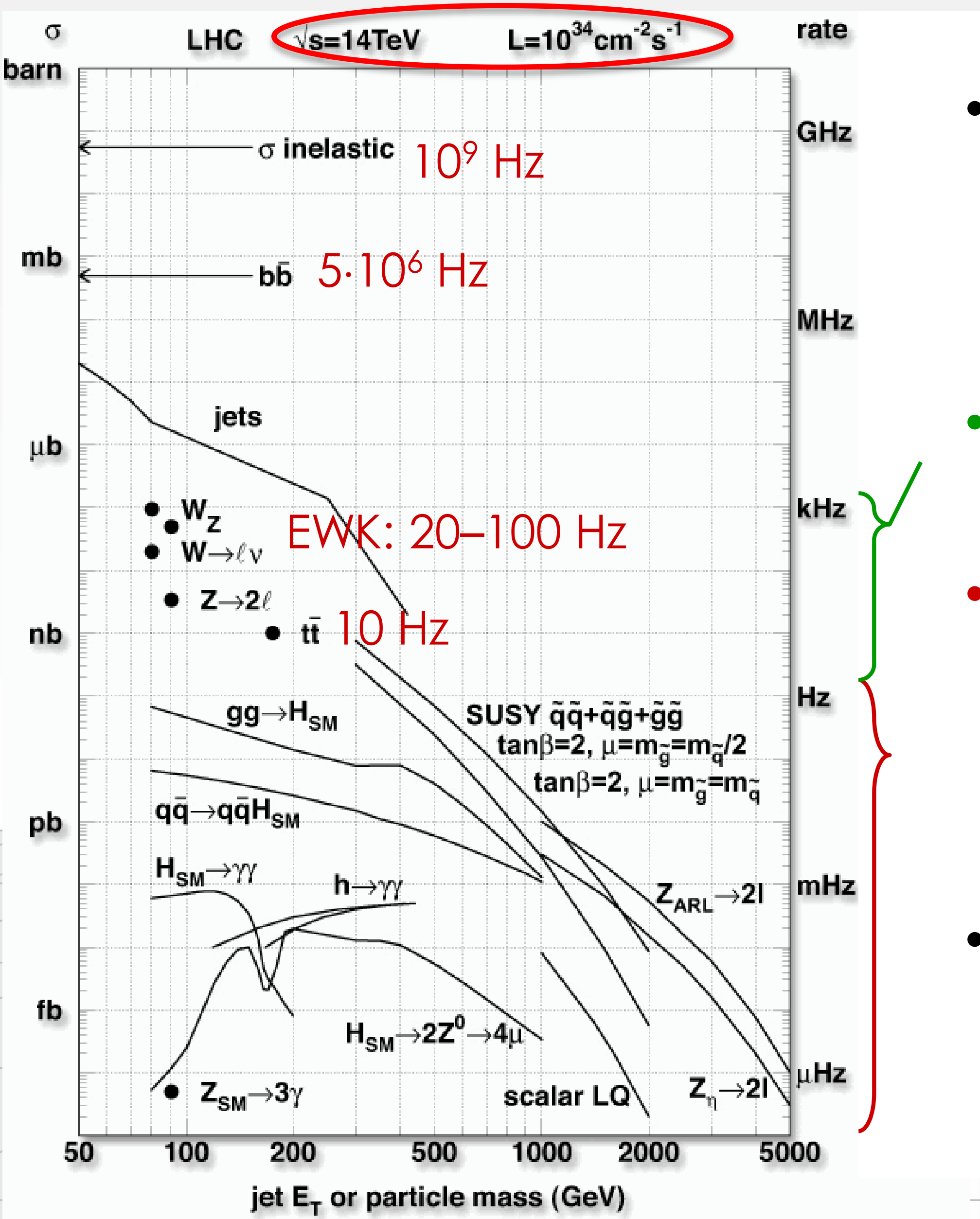
Online Network Infrastructure



Building a trigger (recap)

- Keep it simple! (Remember Einstein: “As simple as possible, but not simpler”)
- Even though “premature optimization is the root of all evil”, think about efficiency (buffering)
- Try to have few adjustable parameters: scanning for a good working point will otherwise be a night-mare

Should we read everything?

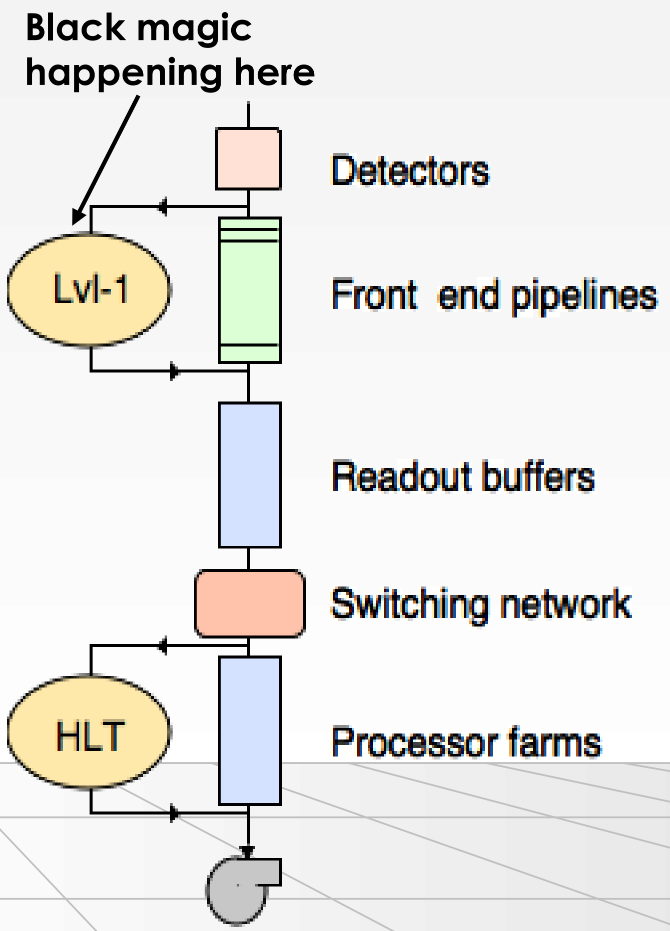


- A typical collision is "boring"
  - Although we need also some of these "boring" data as cross-check, calibration tool and also some important "low-energy" physics
- "Interesting" physics is about 6–8 orders of magnitude rarer (EWK & Top)
- "Exciting" physics involving new particles/discoveries is ≥ 9 orders of magnitude below σ_{tot}
  - 100 GeV Higgs 0.1 Hz
  - 600 GeV Higgs 0.01 Hz

- We *just* ☺ need to efficiently identify these rare processes from the overwhelming background before reading out & storing the whole event

Trigger Condensed

- No (affordable) DAQ system could read out O(10^7) channels at 40 MHz → 400 TBit/s to read out – even assuming binary channels!
- What's worse: most of these millions of events per second are totally uninteresting: one Higgs event every 0.02 seconds
- A *first level trigger (Level-1, L1)* must somehow select the more interesting events and tell us which ones to deal with any further

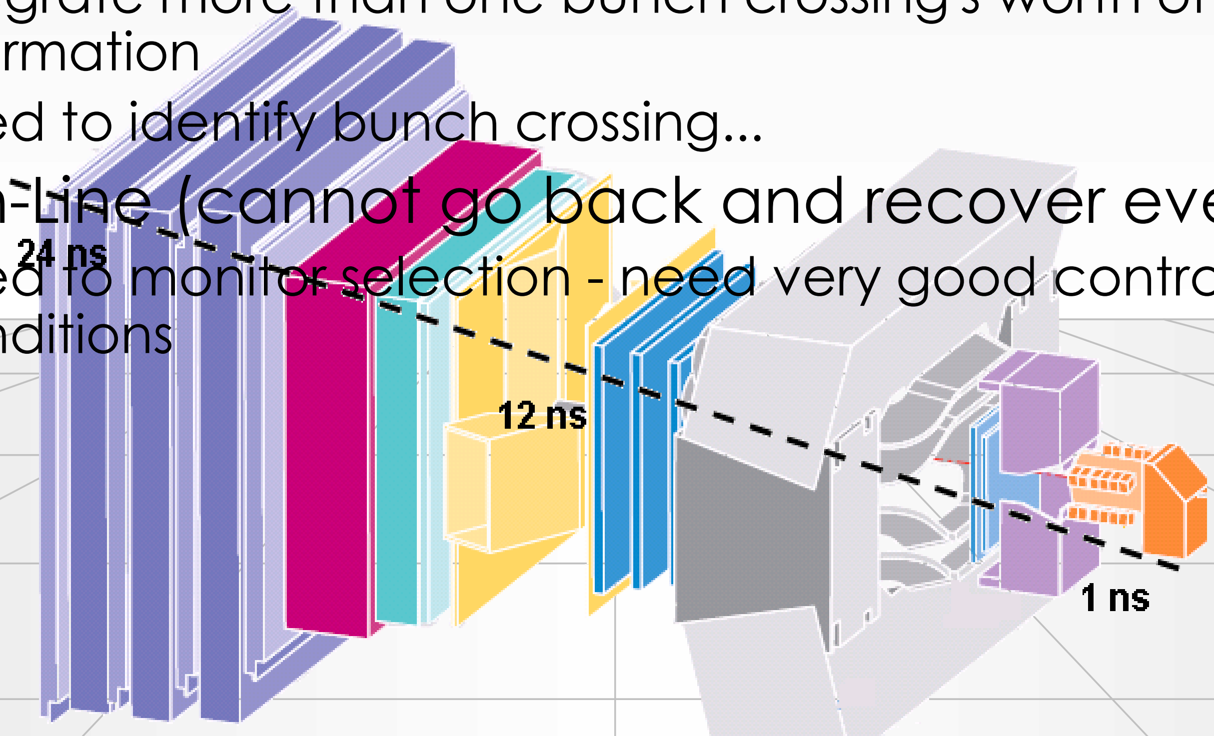


Inside the Box: How does a Level-1 trigger work?

- Millions of channels →: try to work as much as possible with "local" information
  - Keeps number of interconnections low
- Must be fast: look for "simple" signatures
  - Keep the good ones, kill the bad ones
  - Robust, can be implemented in hardware (fast)
- Design principle:
  - fast: to keep buffer sizes under control
  - every 25 nanoseconds (ns) a new event: have to decide within a few microseconds (μs): trigger-latency

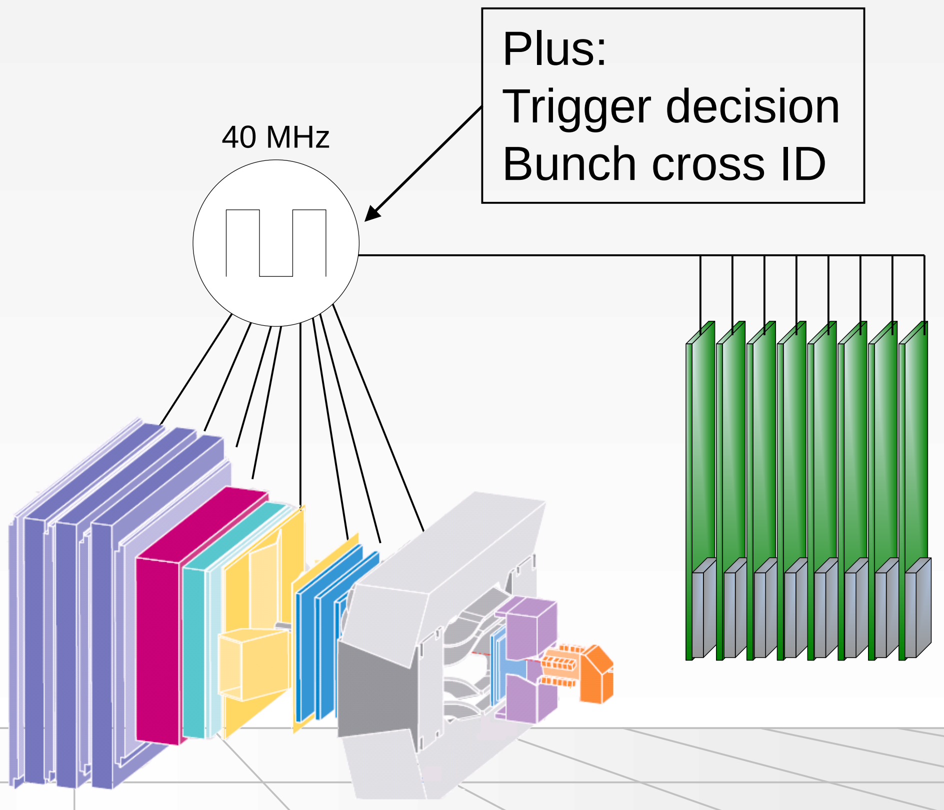
Challenges for the L1 at LHC

- N (channels) ~ $O(10^7)$; ≈20 interactions every 25 ns
  - need huge number of connections
- Need to synchronize detector elements to (better than) 25 ns
- In some cases: detector signal/time of flight > 25 ns
  - integrate more than one bunch crossing's worth of information
  - need to identify bunch crossing...
- It's On-Line (cannot go back and recover events)
  - need to monitor selection - need very good control over all conditions



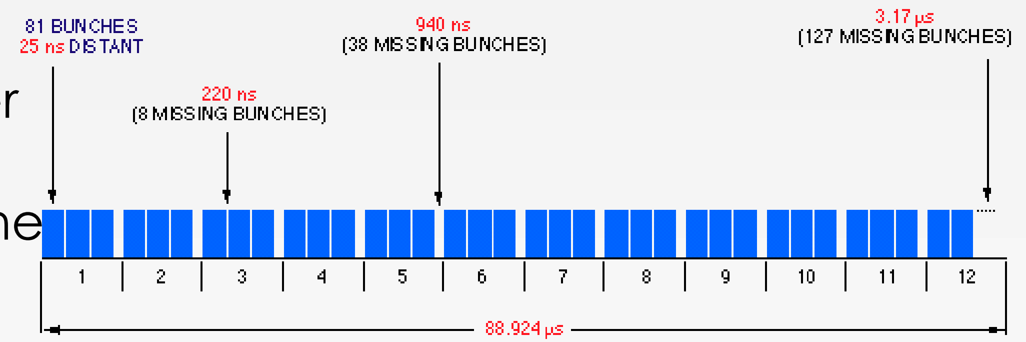
Clock Distribution and Synchronisation

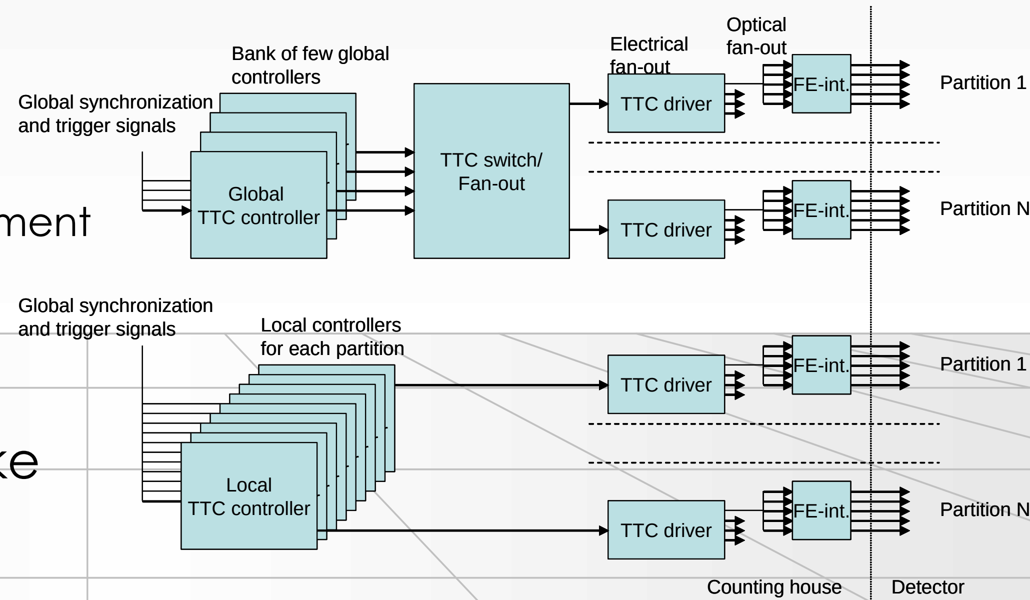
- An *event* is a snapshot of the values of all detector front-end electronics elements, which have their value caused by the same collision
- A common clock signal must be provided to all detector elements
  - Since the c is constant, the detectors are large and the electronics is fast, the detector elements must be carefully time-aligned
- Common system for all LHC experiments TTC based on radiation-hard opto-electronics



Timing & sync control

- Sampling clock with low jitter
- Synch reset
- Synchronization with machine bunch structure
- Calibration
- Trigger (with event type)
- Time align all the different sub-detectors and channels
  - Programmable delays
- Fan-out – unidirectional
  - Global fan-out to whole experiment or
  - Sub-detector fan-out
- Must be reliable as system otherwise may get de-synchronized which may take quite some time to correct



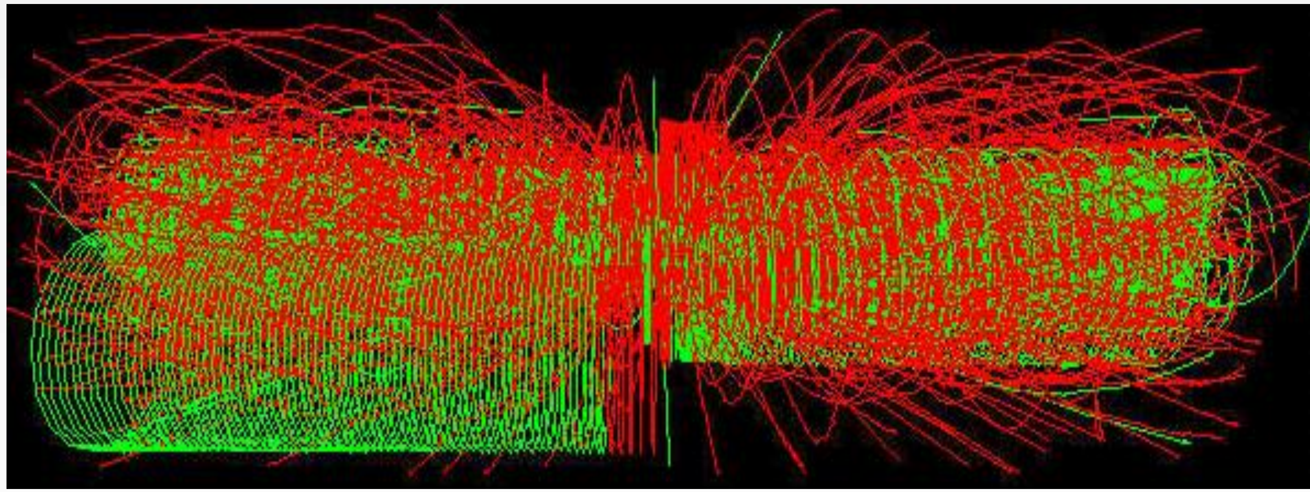


Know Your Enemy: pp Collisions at 14 TeV at 10^{34} $cm^{-2}s^{-1}$

- $\sigma(pp)$ = 70 mb --> >7 x 10^8 /s (!)
- In ATLAS and CMS* 20 min bias events will overlap
- $H \to ZZ$

$Z \to \mu\mu$

$H \to$ 4 muons: the cleanest ("golden") signature

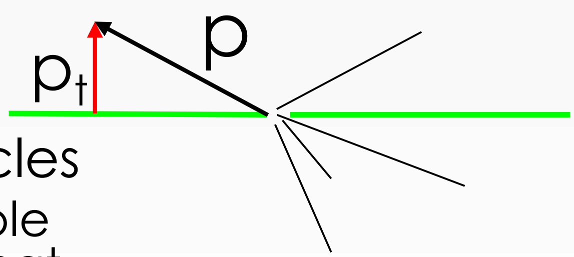


And this
(not the H though…)
repeats every 25 ns…

*) LHCb @2×10^{33} cm^{-2}-1 isn't much nicer and in Alice (PbPb) it will be even worse

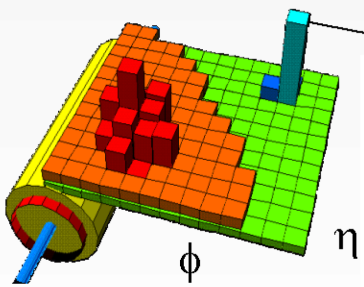
Mother Nature is a … Kind Woman After All

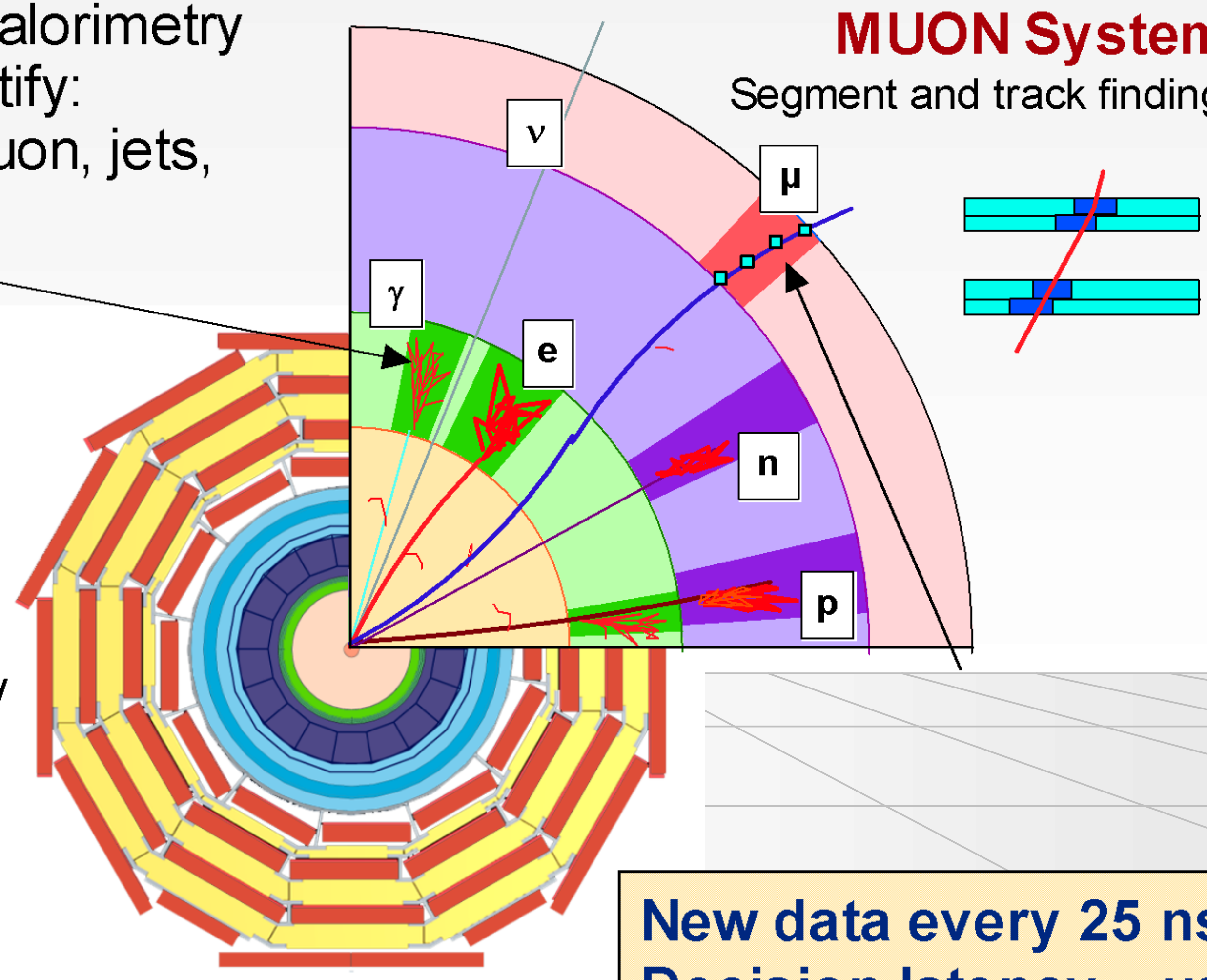
- pp collisions produce mainly hadrons with transverse momentum "p_t" ~1 GeV
- Interesting physics (old and new) has particles (leptons and hadrons) with large p_t:
  - $W \rightarrow e\nu$: $M(W)=80$ GeV/c^2; $p_t(e)$ ~ 30-40 GeV
  - H(120 GeV)$\rightarrow\gamma\gamma$: $p_t(\gamma)$ ~ 50-60 GeV
  - $B \rightarrow \mu D^{*+}\nu$ $p_t(\mu)$ ~ 1.4 GeV
- Impose high thresholds on the p_t of particles
  - Implies distinguishing particle types; possible for electrons, muons and "jets"; beyond that, need complex algorithms
- Conclusion: in the L1 trigger we need to watch out for high transverse momentum electrons, jets or muons



How to defeat minimum bias: transverse momentum p_t

Use prompt data (calorimetry and muons) to identify:
High p_t electron, muon, jets, missing E_T



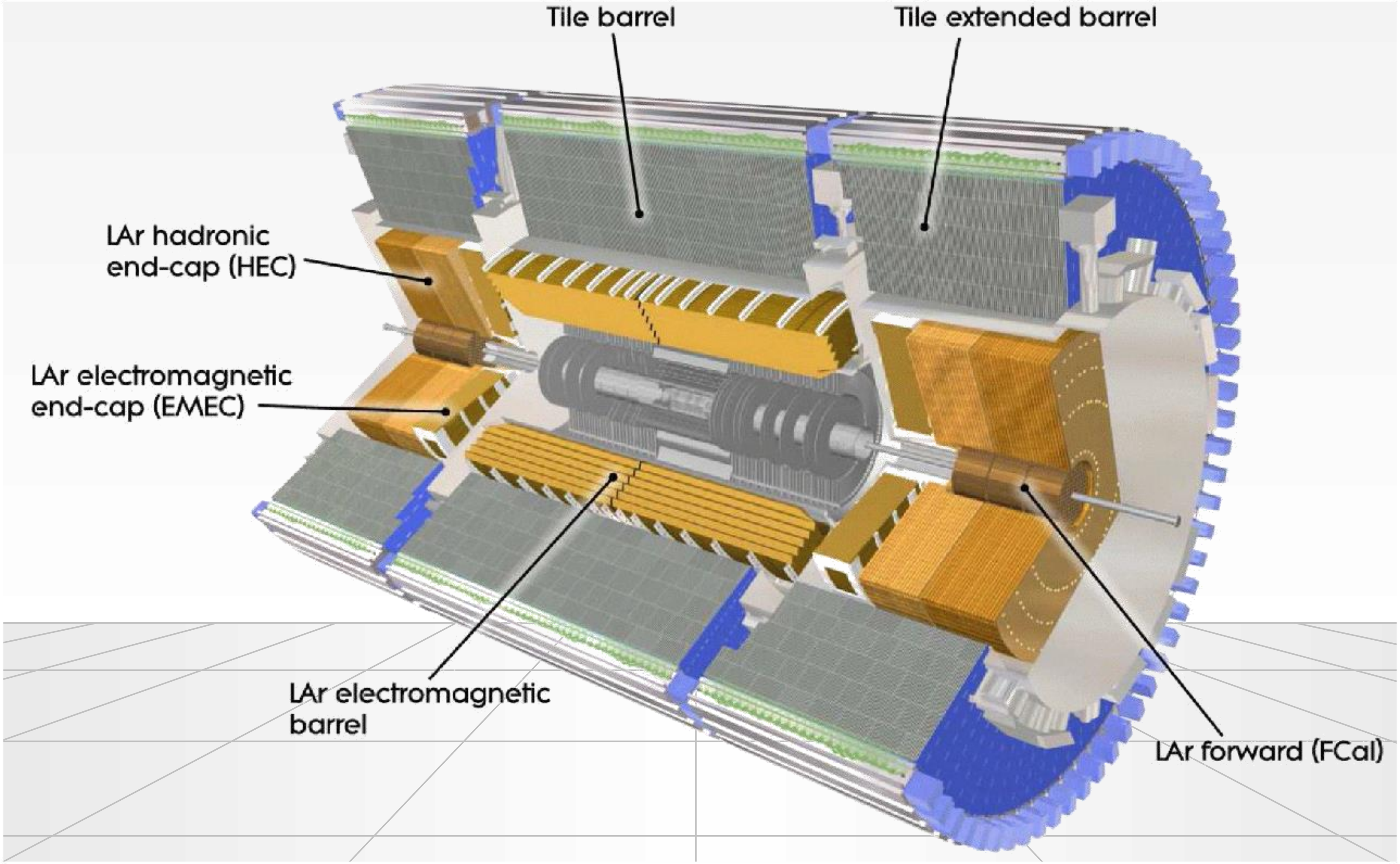


MUON System
Segment and track finding

CALORIMETERs
Cluster finding and energy deposition evaluation

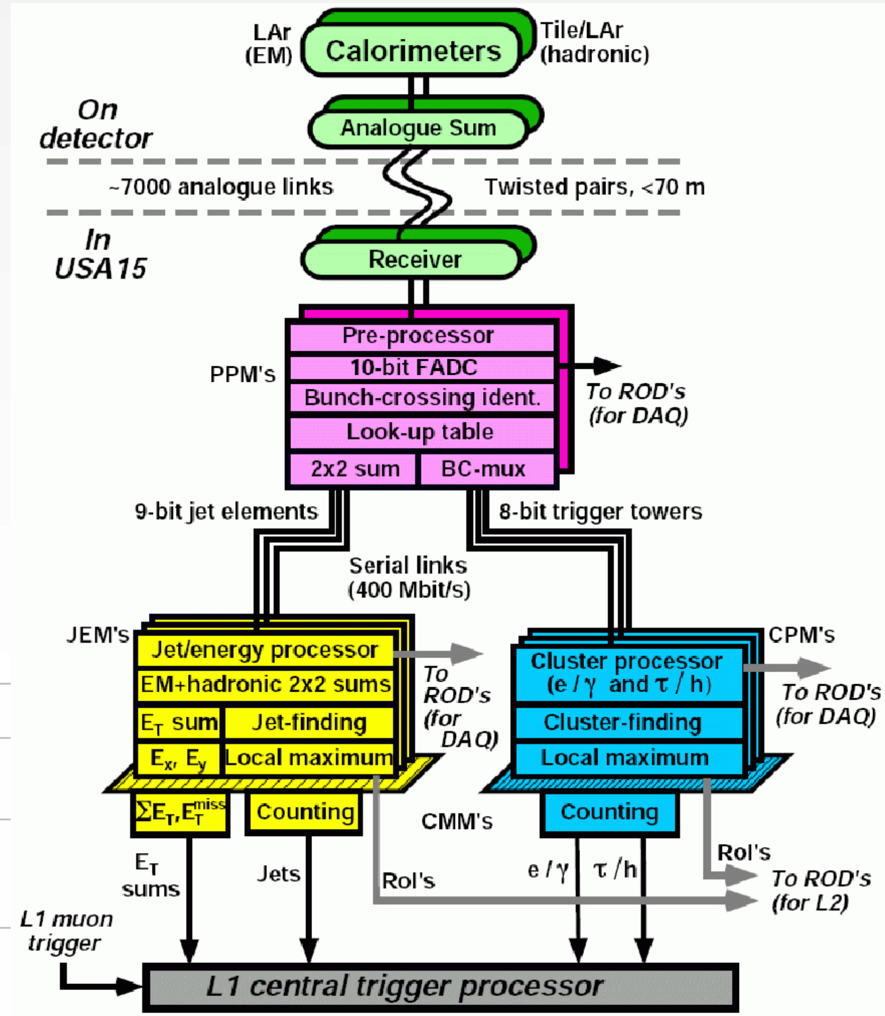
New data every 25 ns
Decision latency ~ μs

ATLAS Calorimeters



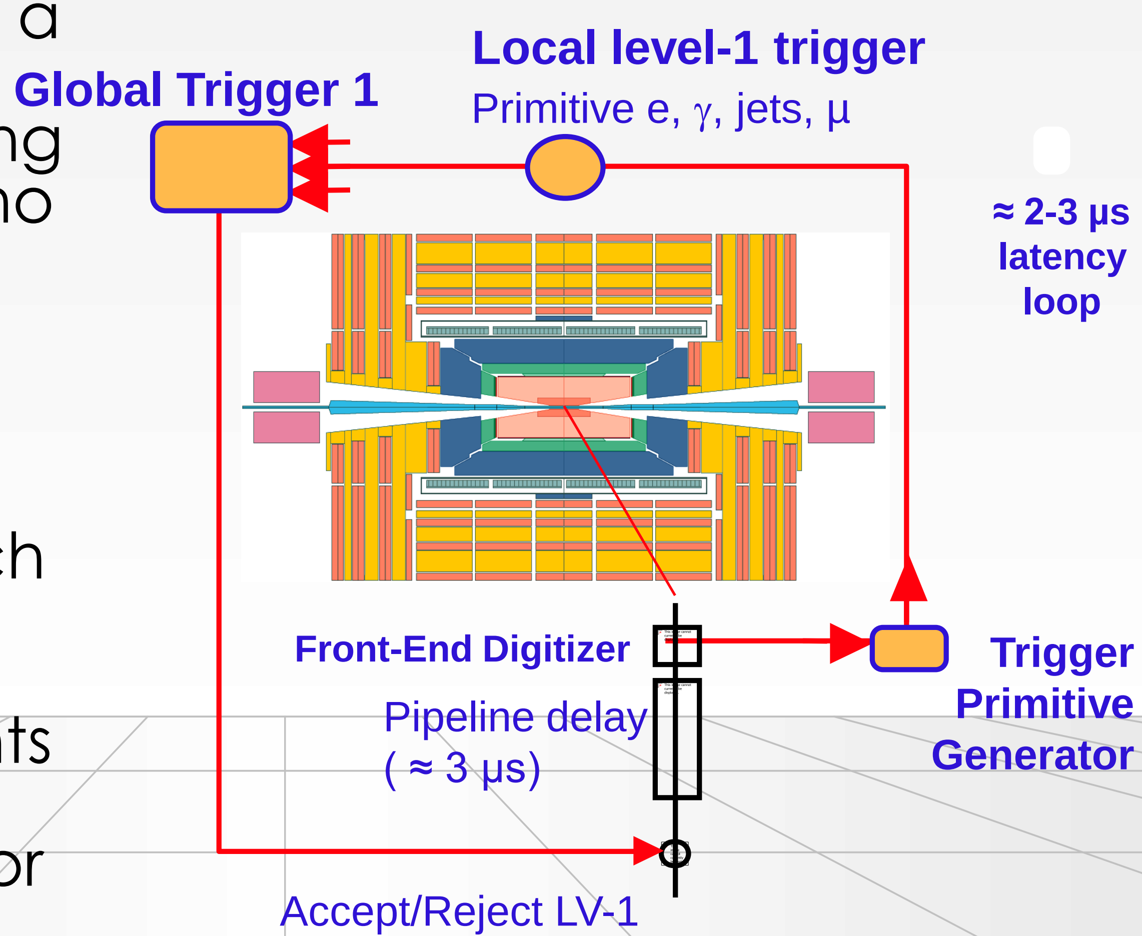
ATLAS L1 Calo Trigger

- Form analogue towers 0.1 x 0.1 ($\delta\eta$ x $\delta\varphi$)
- Digitize, identify
- bunch-xing, Look-Up Table (LUT) → E_T
- Duplicate data to Jet/Energy-sum
- (JEP) and Cluster (CP) processors
- Send to CTP 1.5 μs after bunch-crossing ("x-ing").
- Store info at JEP and CP to seed next level of trigger

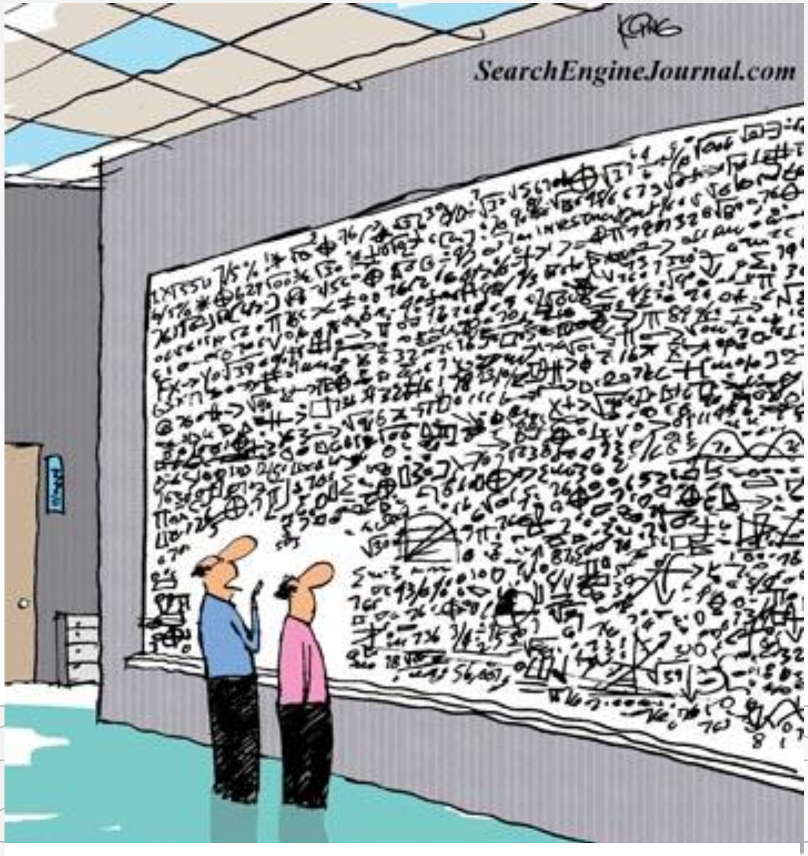


Distributing the L1 Trigger

- Assuming now that a magic box tells for each bunch crossing (clock-tick) yes or no
  - Triggering is not for philosophers – "perhaps" is not an option
- This decision has to be brought for each crossing to all the detector front-end electronics elements so that they can send of their data or discard it

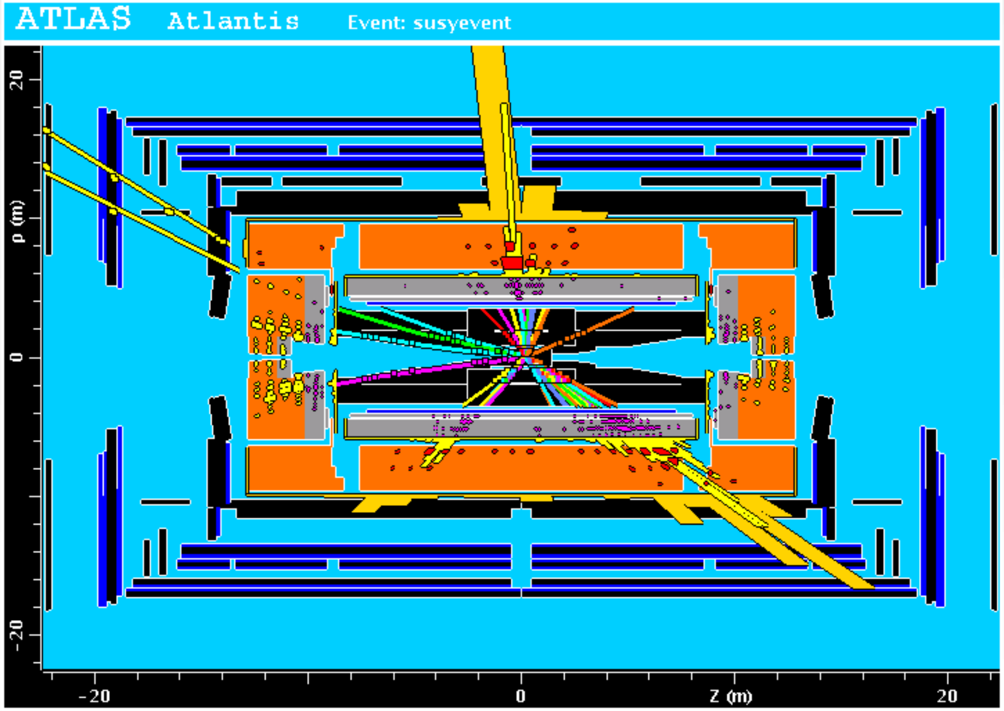


High Level Trigger

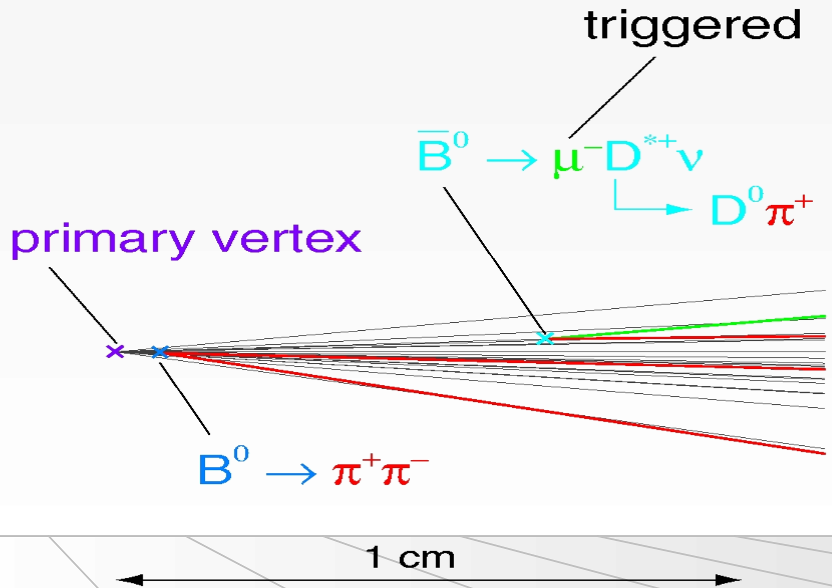


And that, in simple terms, is what we do in the High Level Trigger

High Level Trigger

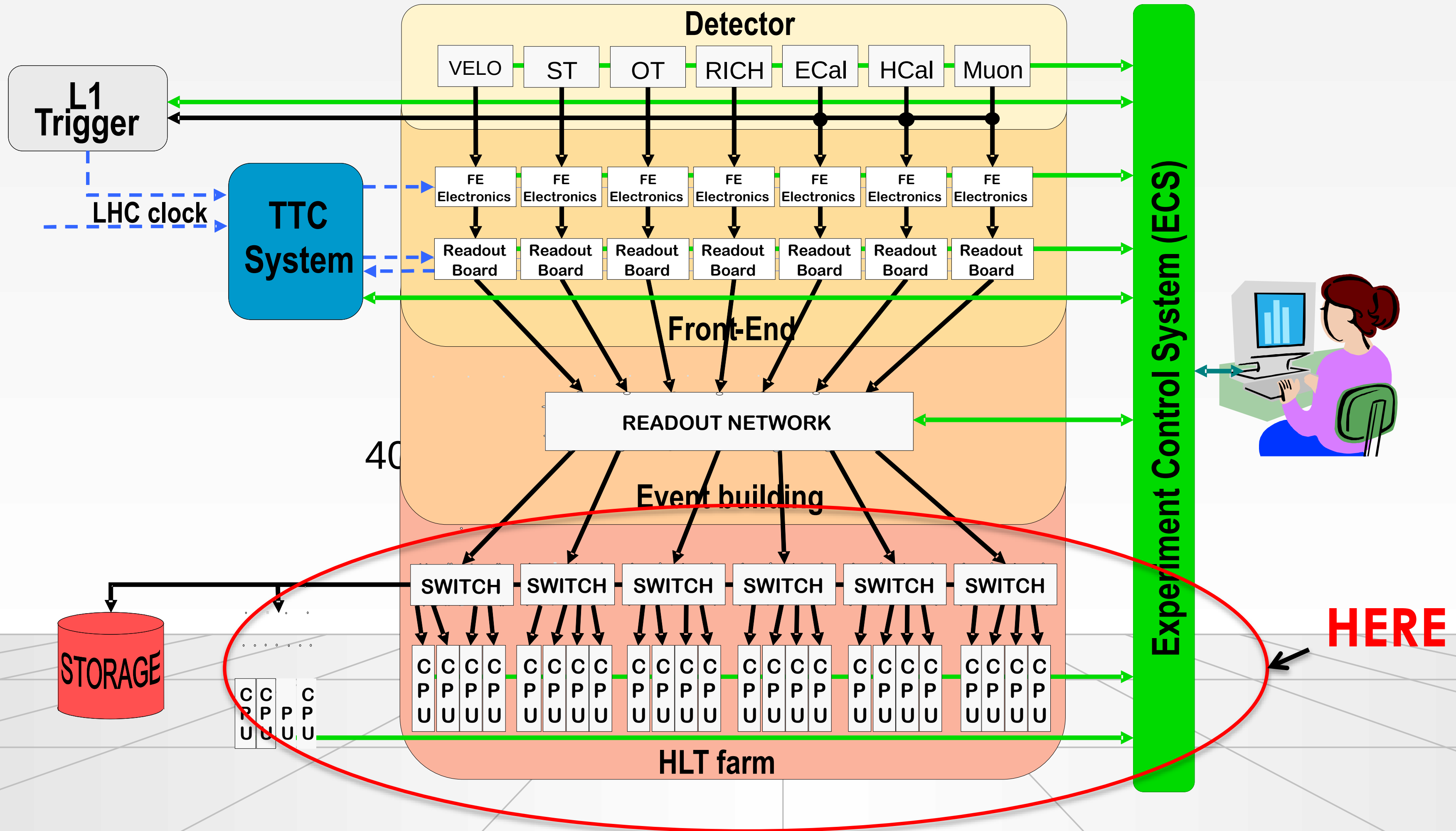


Complicated Event structure with hadronic jets (ATLAS) or secondary vertices (LHCb) require full detector information



Methods and algorithms are the same as for offline reconstruction (Lecture "From raw data to physics")

The High Level Trigger is …



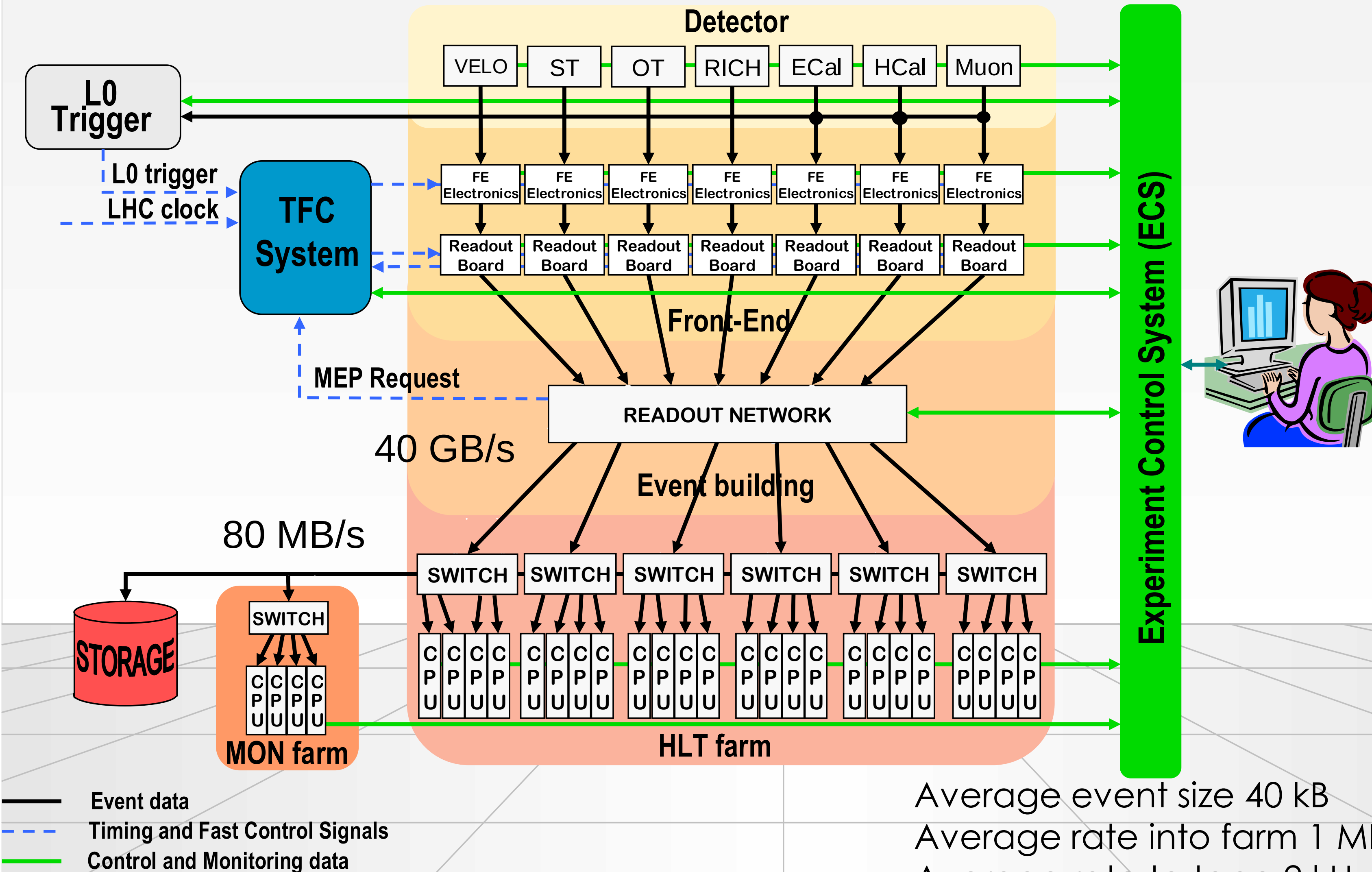
After L1: What's next?

- Where are we after L1
  - ATLAS and CMS : rate is ~75 to 100 kHz, event size ~ 1 - 2 MB
  - LHCb: rate is 1 MHz, event size 40 kB / ALICE: O(kHz) and O(GB)
- Ideally
  - Run the real full-blown physics reconstruction and selection algorithms
  - These application take O(s). Hence: even at above rates still need ***100 MCHF server farm (Intel will be happy!)***
- In Reality:
  - Start by looking at only part of the detector data **seeded by what triggered the 1st level**
  - LHCb: 1st level Trigger confirmation" algorithms: < 10 ms/event
  - Atlas: Region of Interest" (RoI): < 40 ms/event
- ➔ Reduce the rate by factor ~ 30, and then do offline analysis

Event Building

(providing the data for the High Level Trigger)

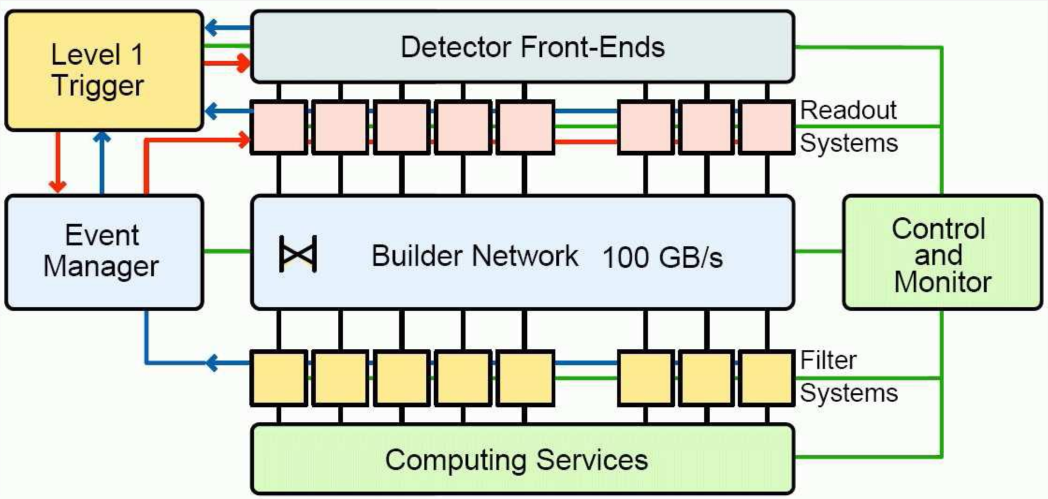
LHCb DAQ



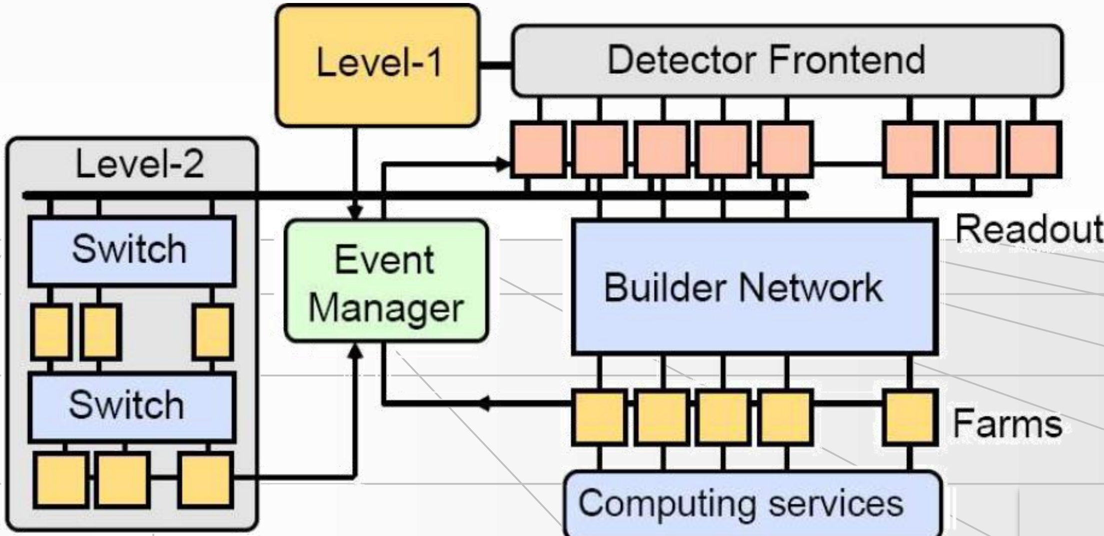
Average event size 40 kB
Average rate into farm 1 MHz
Average rate to tape 2 kHz

Two philosophies

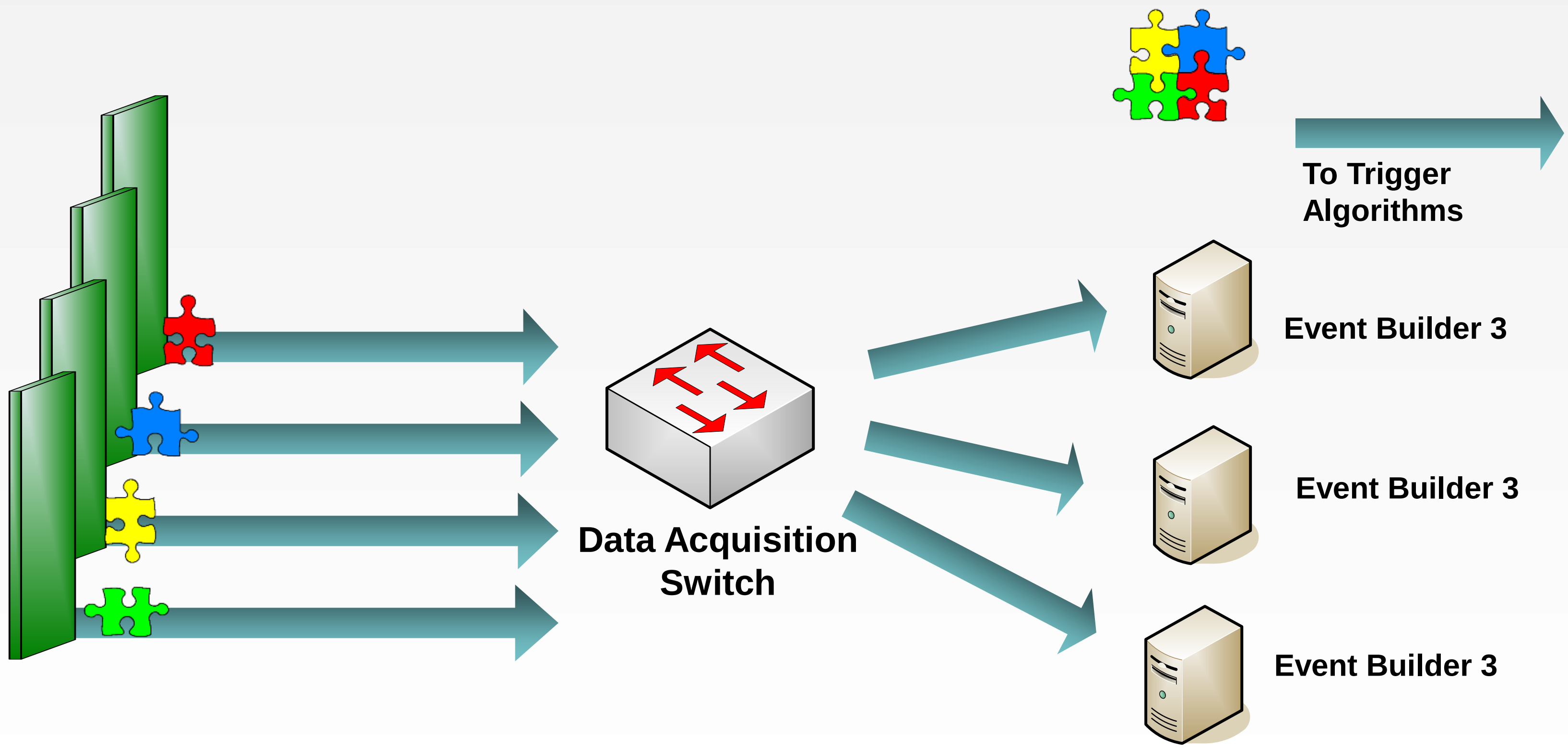
- Send everything, ask questions later (ALICE, CMS, LHCb)



- Send a part first, get better question
  Send everything only if interesting (ATLAS)



Event Building



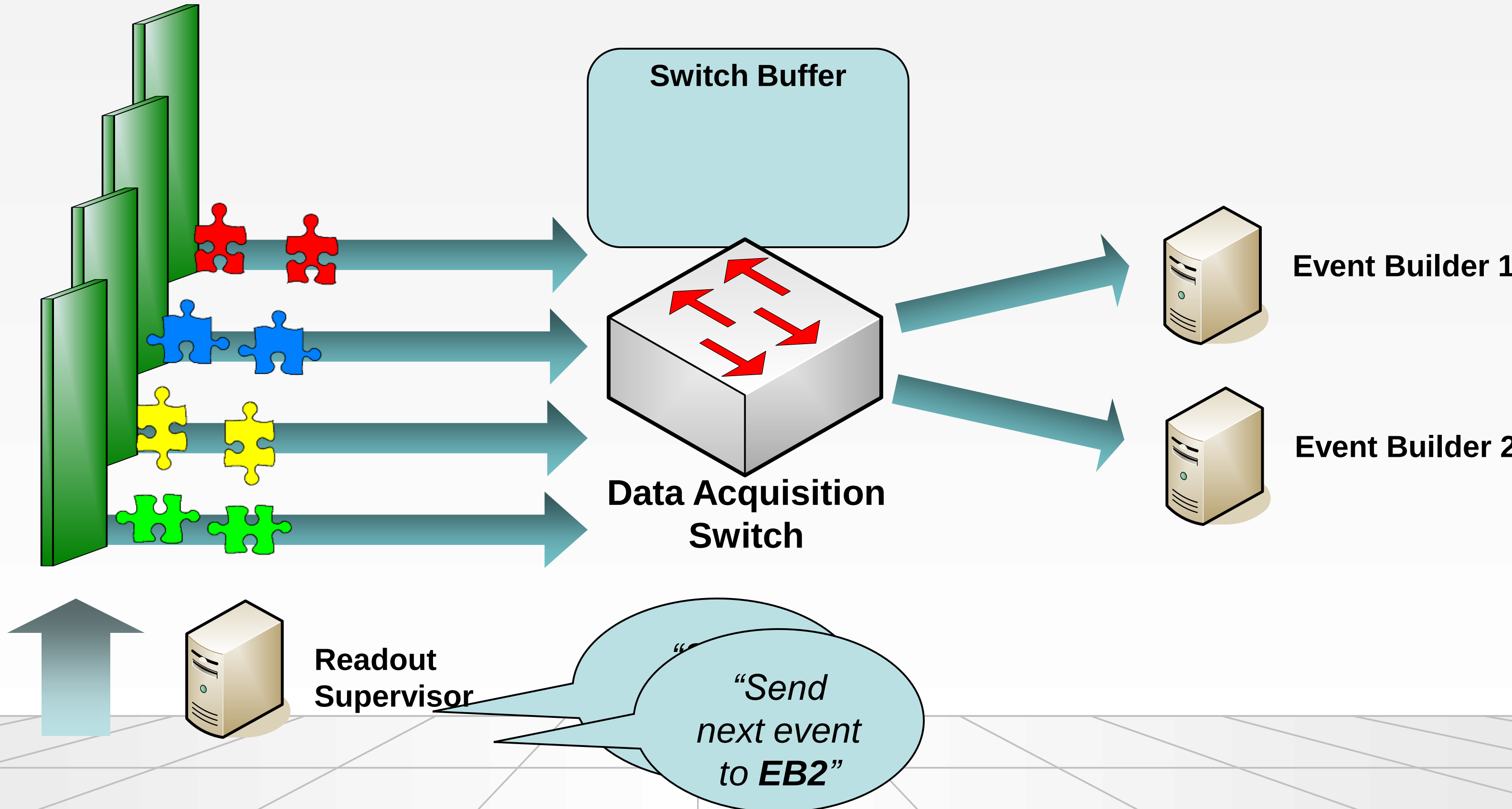
1 Event fragments are received from detector front-end

2 Event fragments are read out over a network to an event builder

3 Event builder assembles fragments into a complete event

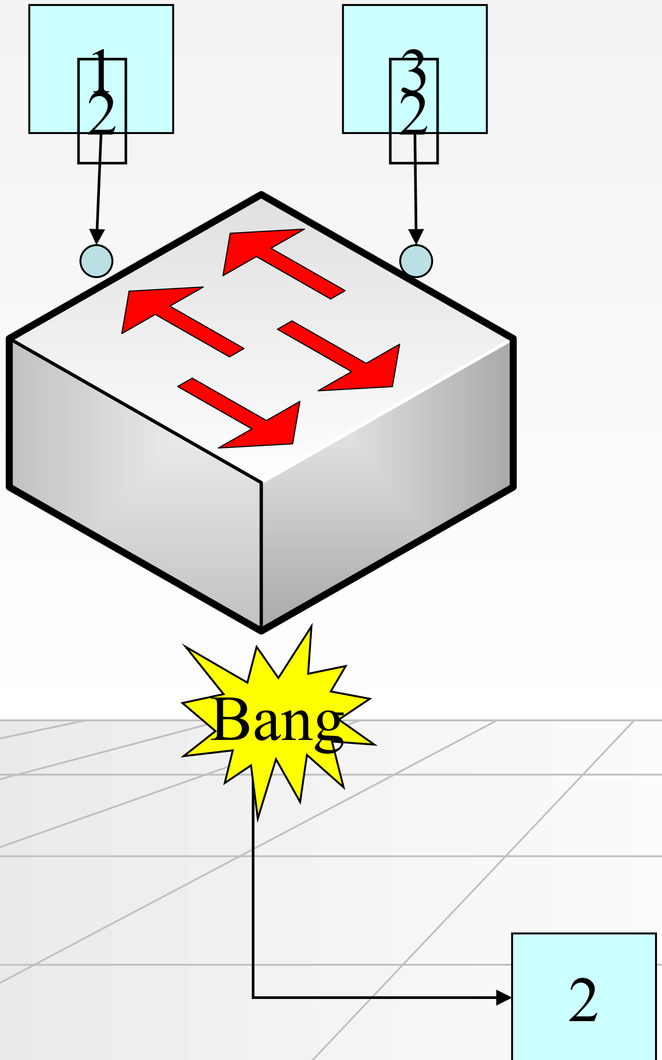
4 Complete events are processed by trigger algorithms

Push-Based Event Building



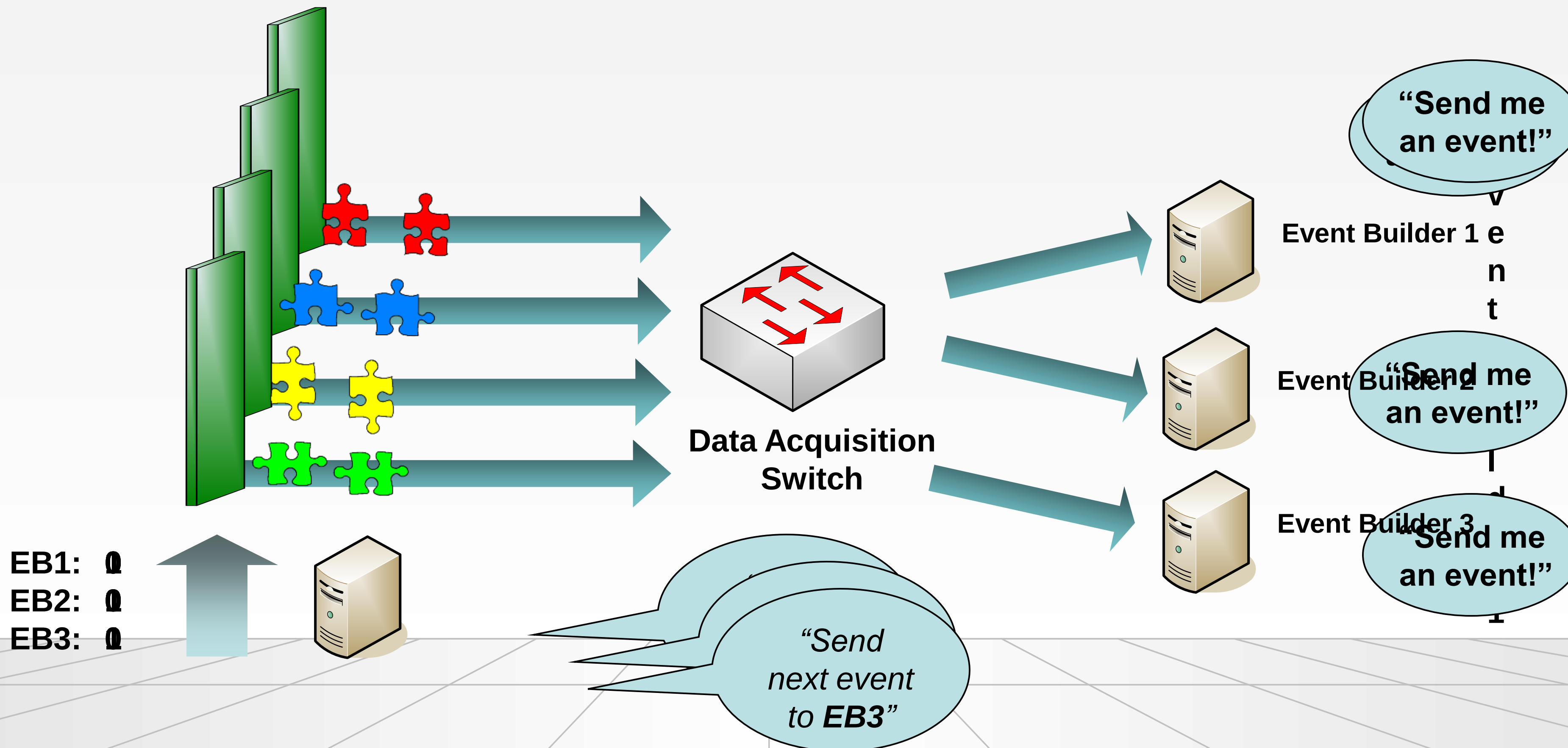
1. Readout Supervisor tells readout boards where events must be sent (round-robin)
2. Readout boards do not buffer, so switch must
3. No feedback from Event Builders to Readout system

Congestion



- "Bang" translates into random, uncontrolled packet-loss
- In Ethernet this is perfectly valid behavior and implemented by very cheap devices
- Higher Level protocols are supposed to handle the packet loss due to *lack of buffering*
- This problem comes from ***synchronized*** sources ***sending*** to the same destination at the ***same time***

Pull-Based Event Building



1 Event Builders notify Readout Supervisor of available capacity

2 Readout Supervisor ensures that data are sent only to nodes with available capacity

3 Readout system relies on feedback from Event Builders

The end

Further Reading

- Electronics
  - Helmut Spielers web-site: http://www-physics.lbl.gov/~spieler/
- Buses
  - VME: http://www.vita.com/
  - PCI http://www.pcisig.com/
- Network and Protocols
  - Ethernet "Ethernet: The Definitive Guide", O'Reilly, C. Spurgeon
  - TCP/IP "TCP/IP Illustrated", W. R. Stevens
  - Protocols: RFCs www.ietf.org in particular RFC1925 http://www.ietf.org/rfc/rfc1925.txt "The 12 networking truths" is required reading
- Wikipedia (!!!) and references therein – for all computing related stuff this is usually excellent
- Conferences
  - IEEE Realtime
  - ICALEPCS
  - CHEP
  - IEEE NSS-MIC
- Journals
  - IEEE Transactions on Nuclear Science, in particular the proceedings of the IEEE Realtime conferences
  - IEEE Transactions on Communications

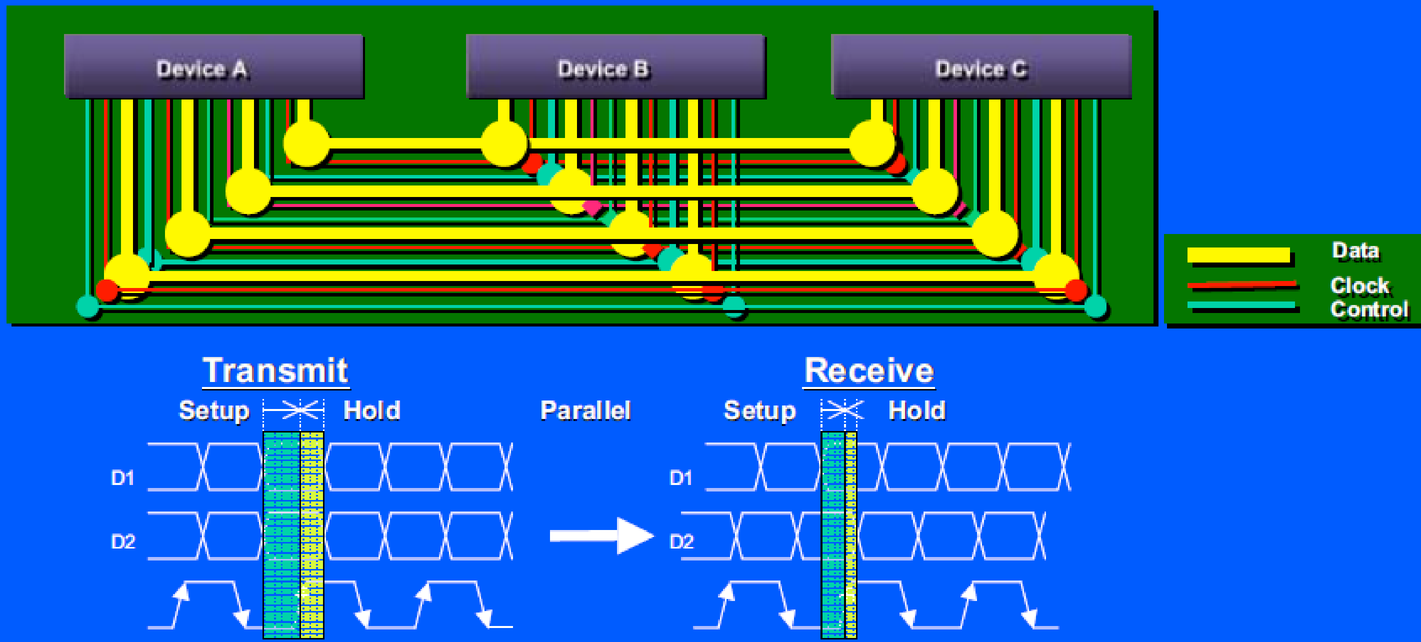
More Recent “Buses”

PCI-Express

- Not anymore a Bus in the classical sense
- Serial instead of parallel
- Variable Amount of Lanes
- More like a network

PCI vs. PCI-E

PCI Bus

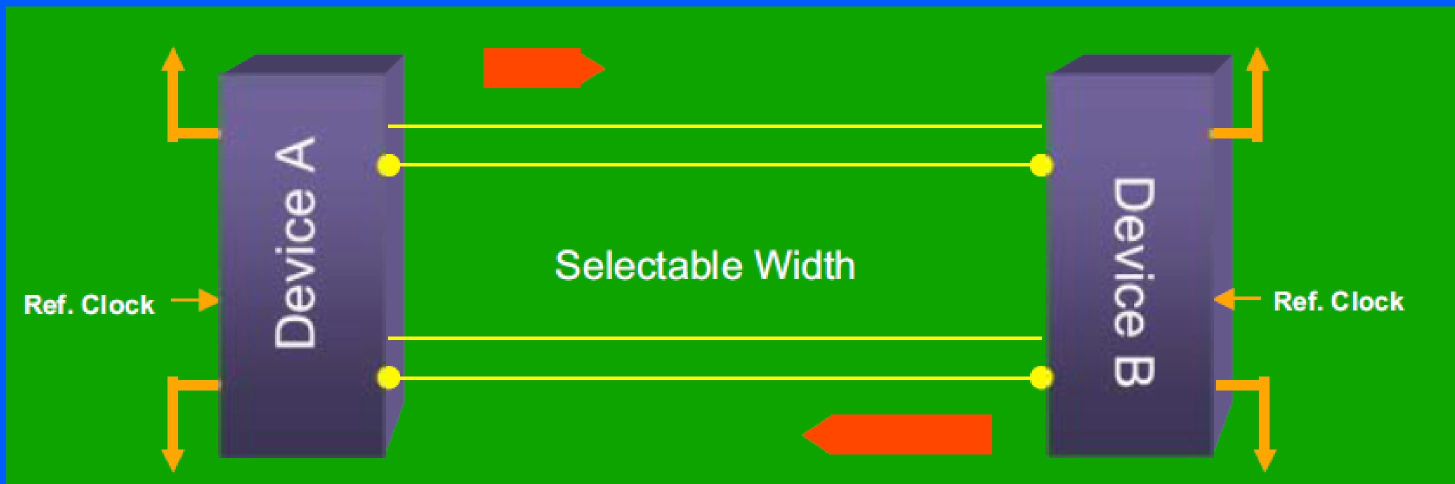


- Stringent routing rules
- Limited scaling with frequency and voltage
- Lack of advanced features

Parallel I/O interconnects have reached practical limit

PCI-E Bus

PCI Express Interconnect



- Dual Simplex Point to point topology
- Differential low voltage interconnect
- Bit rate: > 2.5Gb/sec/lane/direction and beyond
- Selectable lane width:
  - ✓x1, x2, x4, x8, x12, x16, x32

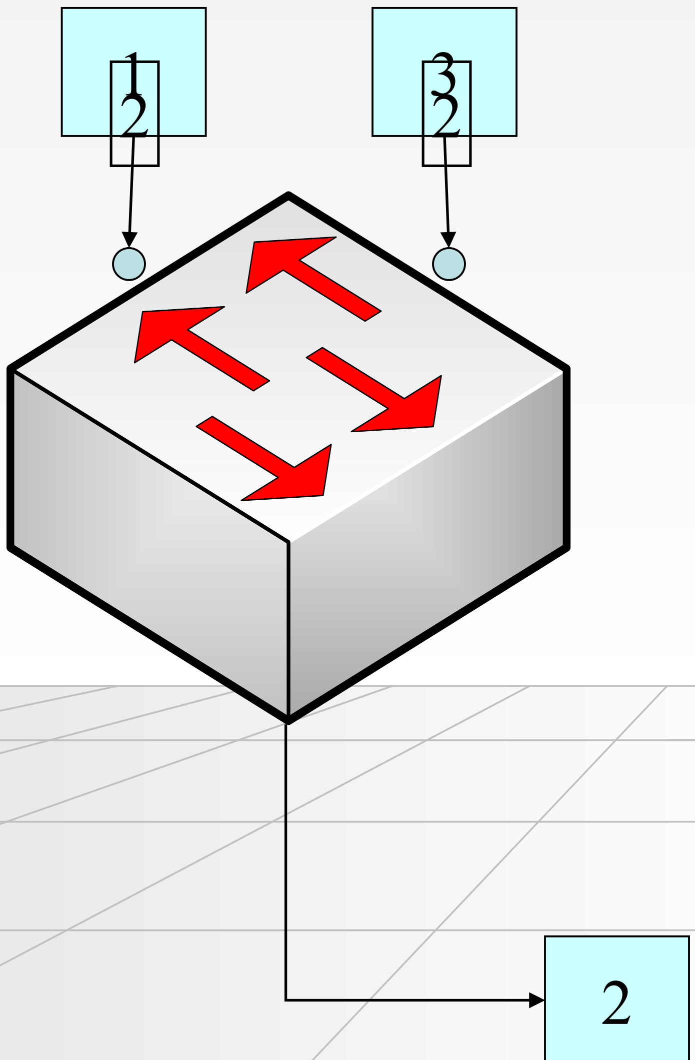
Micro TCA/ATCA

- Going even further with the network idea
- XAUI Protocol
  - Essential the physical connection protocol of 10 G Ethernet
- Backplane is now a completely serialized multi-Gbps dual star or mesh architecture
- Standardized Mezzanine Cards (AMC) that can be hot plugged into backplane
- IPMI infrastructure for managing hardware and diagnosing/fixing problems
- Can be made fully redundant (dual star)
- RTM provides Trigger and Clock infrasturcture

More Stuff

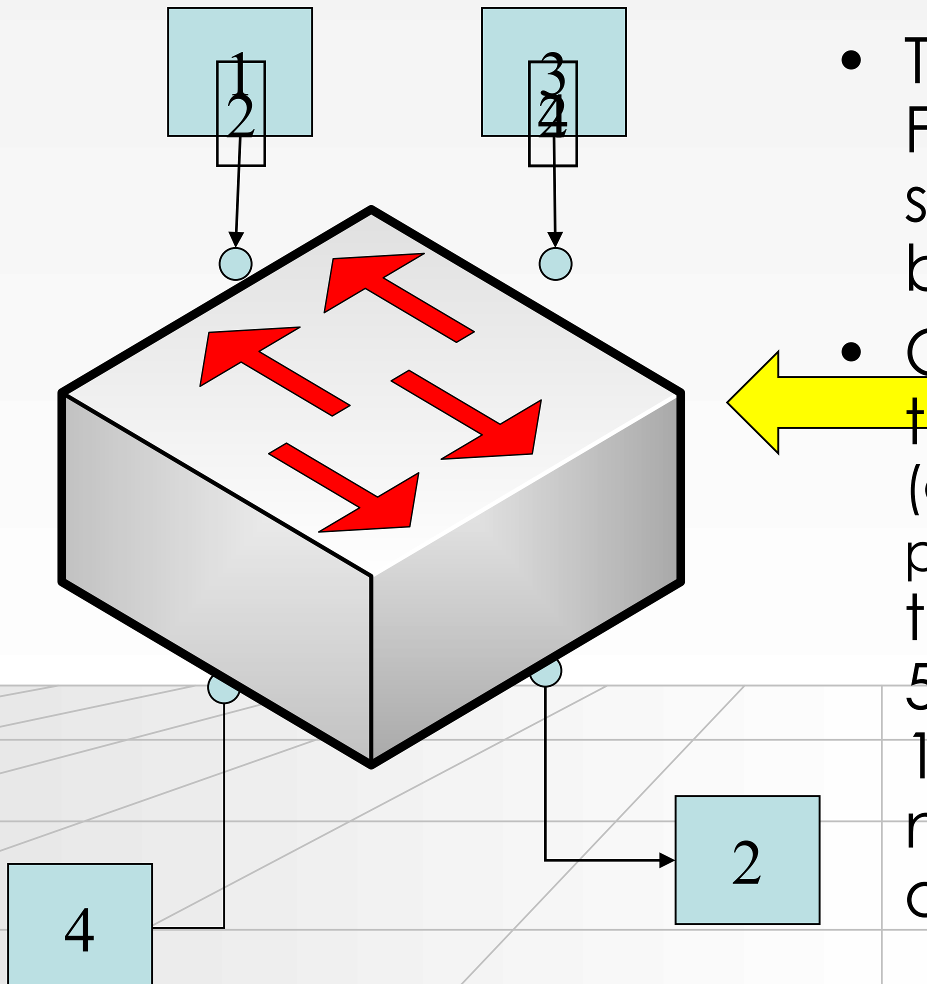
Data format, DIY DAQ, run-control

Overcoming Congestion: Queuing at the Input



- Two frames destined to the same destination arrive
- While one is switched through the other is waiting at the input port
- When the output port is free the queued packet is sent

Head of Line Blocking

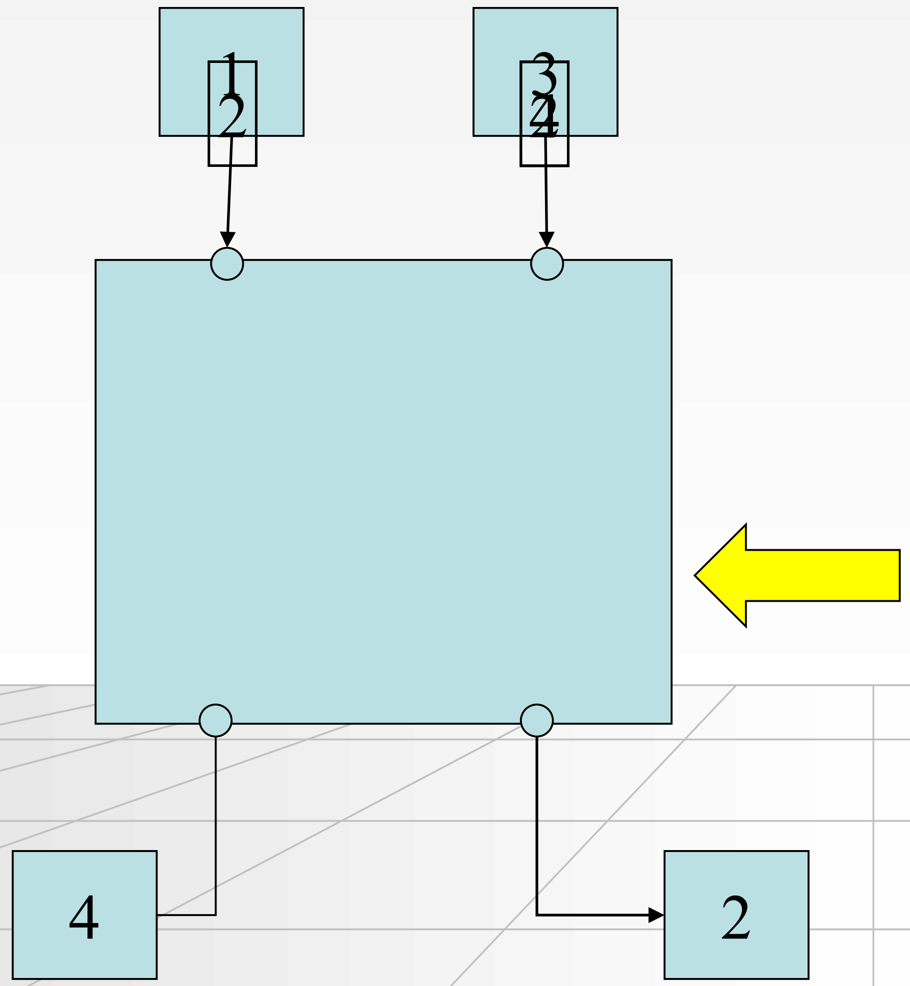


- The reason for this is the First in First Out (FIFO) structure of the input buffer
- Queuing theory tells us* that for random traffic (and infinitely many switch ports) the throughput of the switch will go down to 58.6% → that means on 100 MBit/s network the nodes will "see" effectively only ~ 58 MBit/s

Packet to node 4 must wait even though port to node 4 is free

*) "Input Versus Output Queueing on a Space-Division Packet Switch"; Karol, M. et al. ; IEEE Trans. Comm., 35/12

Output Queuing



- In practice virtual output queueing is used: at each input there is a queue → for *n* ports $O(n^2)$ queues must be managed
- Assuming the buffers are large enough(!) such a switch will sustain random traffic at 100% nominal link load

Packet to node 2 waits at output to port 2. Way to node 4 is free

Raw data format

There are 10 kinds of people in the world


```
0000260 0101 0101 0101 0000 0000 0000 0000 0201
0000300 0403 0605 0807 0a09 010b 0300 0101 0101
0000320 0101 0101 0001 0000 0000 0100 0302 0504
0000340 0706 0908 0b0a 0010 0102 0303 0402 0503
0000360 0405 0004 0100 017d 0302 0400 0511 2112
0000400 4131 1306 6151 2207 1471 8132 a191 2308
0000420 b142 15c1 d152 24f0 6233 8272 0a09 1716
0000440 1918 251a 2726 2928 342a 3635 3837 3a39
0000460 4443 4645 4847 4a49 5453 5655 5857 5a59
0000500 6463 6665 6867 6a69 7473 7675 7877 7a79
0000520 8483 8685 8887 8a89 9392 9594 9796 9998
0000540 a29a a4a3 a6a5 a8a7 aaa9 b3b2 b5b4 b7b6
```



```
<ADCVALUE>
<TIME>00:04:10</TIME>
<VALUE>0.2334</VALUE>
<PCISTATUS>OK</PCISTATUS>
</ADCVALUE>
<ADCVALUE>
<TIME>00:05:10</TIME>
<VALUE>0.9999</VALUE>
<PCISTATUS>ERROR</PCISTATUS>
</ADCVALUE>
<ADCVALUE>
<TIME>00:06:10</TIME>
<VALUE>0.6334</VALUE>
<PCISTATUS>OK</PCISTATUS>
</ADCVALUE>
<ADCVALUE>
<TIME>00:07:10</TIME>
<VALUE>0.8334</VALUE>
<PCISTATUS>OK</PCISTATUS>
</ADCVALUE>
```

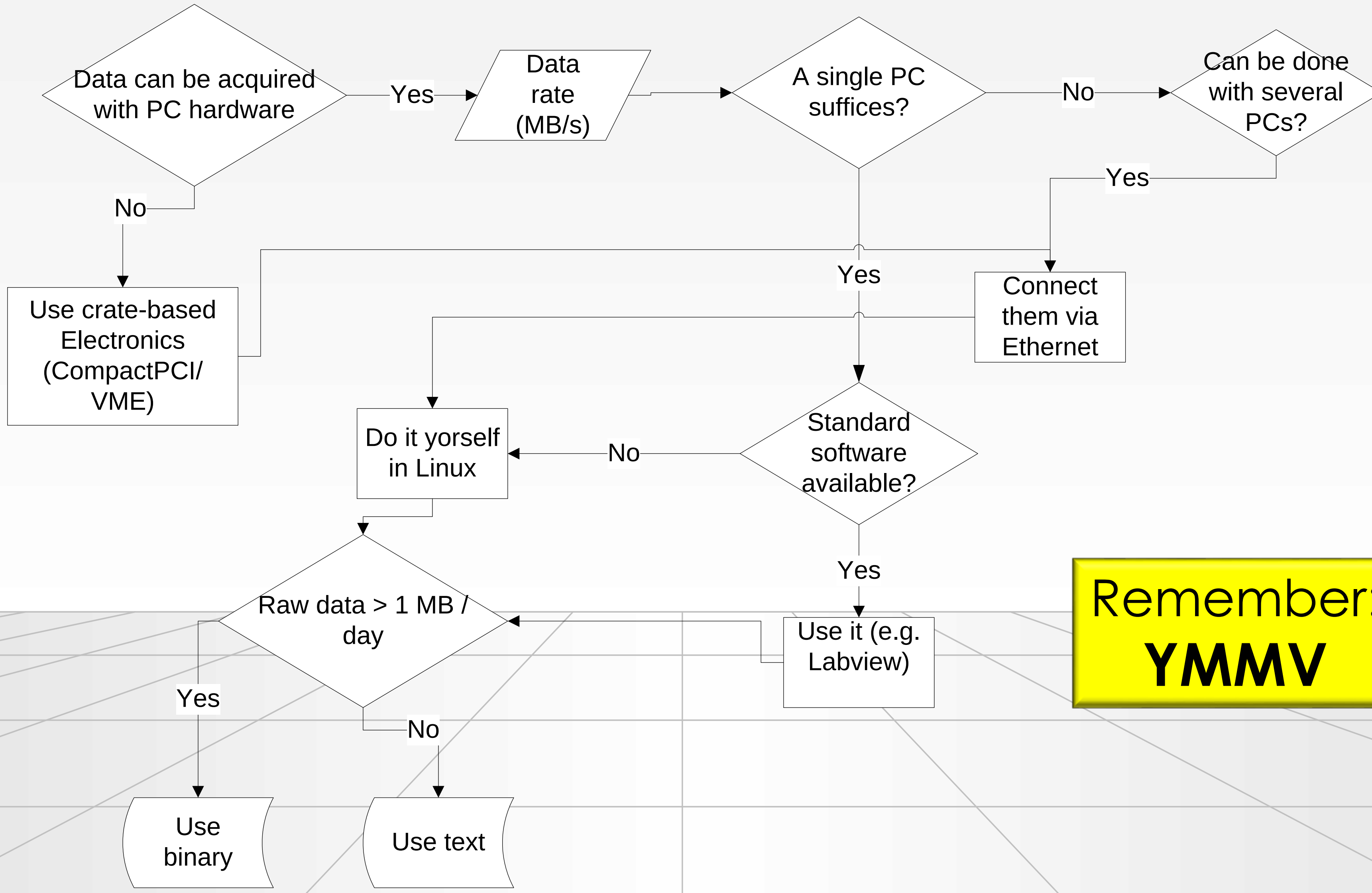

Those who can read binary and those who cannot

Binary vs Text

- 11010110 Pros:
  - compact
  - quick to write & read (no conversion)
- Cons:
  - opaque (humans need tool to read it)
  - depends on the machine architecture (endinaness, floating point format)
  - life-time bound to availability of software which can read it

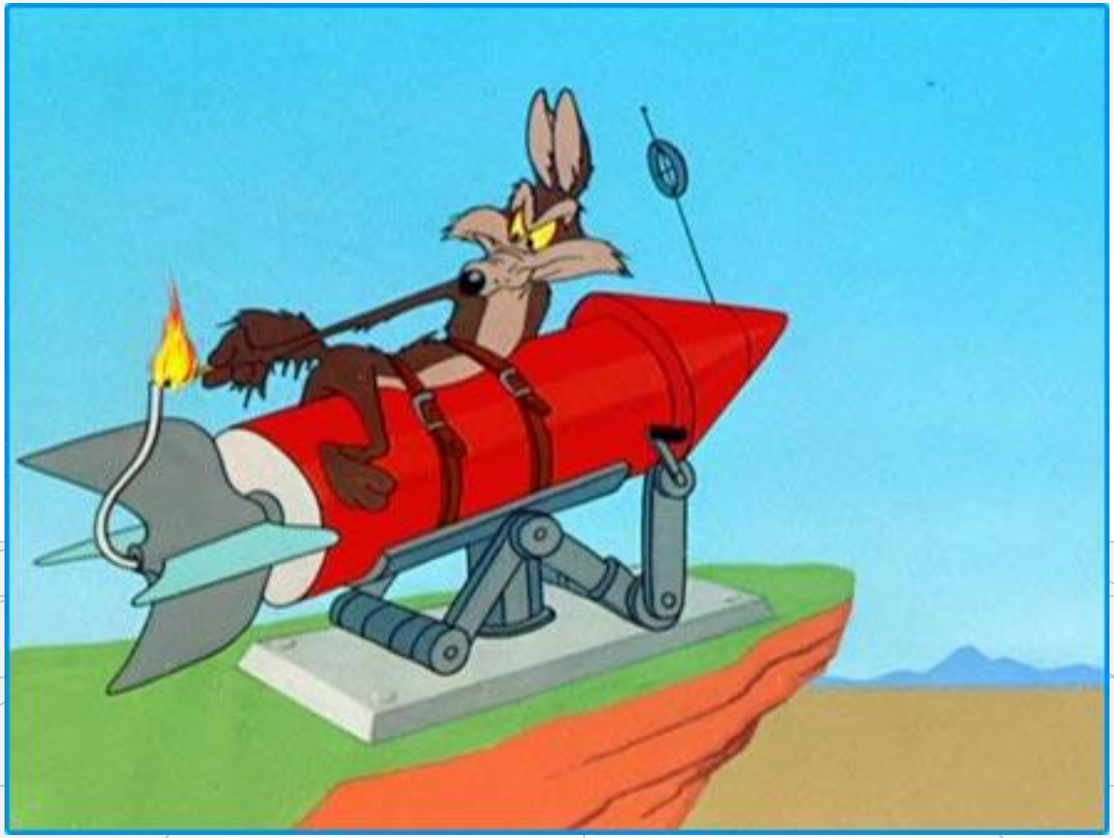
- <TEXT></TEXT> Pros:
  - universally readable
  - can be parsed and edited equally easily by humans and machines
  - long-lived (ASCII has not changed over decades)
  - machine independent
- Cons:
  - slow to read/write
  - low information density (can be improved by compression)

A little checklist for your DAQ



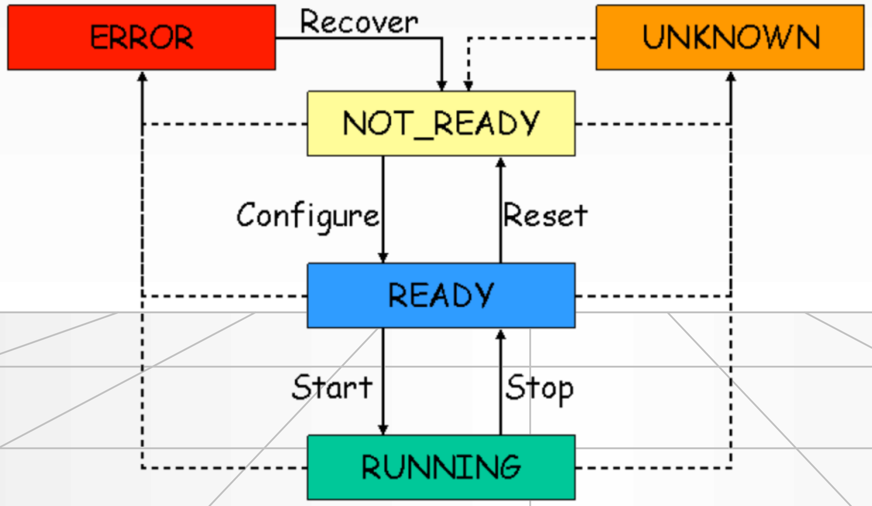
Remember:
YMMV

Runcontrol



Run Control

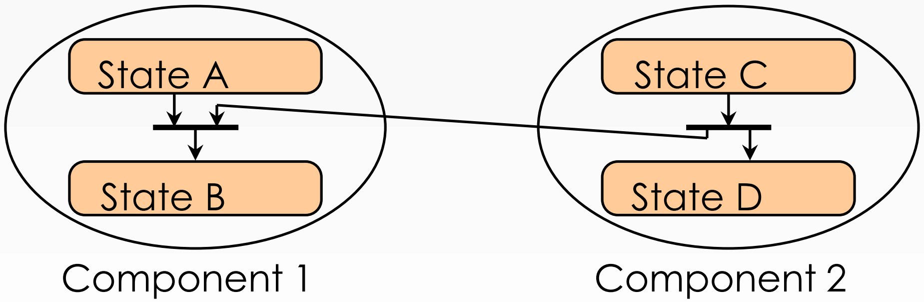
- The run controller provides the control of the trigger and data acquisition system. It is the application that interacts with the operator in charge of running the experiment.
- The operator is not always an expert on T/DAQ. The **user interface** on the Run Controller plays an important role.
- The complete system is modeled as a **finite state machine**. The commands that run controller offers to the operator are state transitions.



LHCb DAQ /Trigger Finite State Machine diagram (simplified)

Finite State Machine

- Each component, sub-component of the system is modeled as a *Finite State Machine*. This abstraction facilitates the description of each component behavior without going into detail
- The control of the system is realized by inducing transitions on remote components due to a transition on a local component



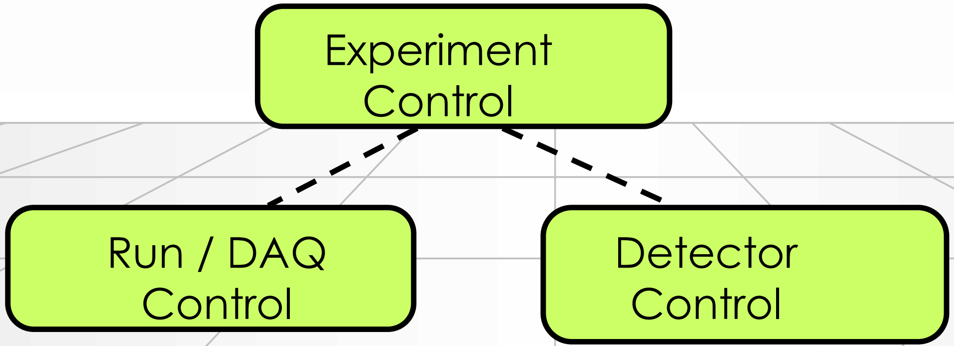
Component 1 can only complete the transition to State B if Component 2 is in state D.

- Each transition may have actions associated. The action consist of code which needs to be executed in order to bring the component to its new state
- The functionality of the FSM and state propagation is available in special software packages such as SMI

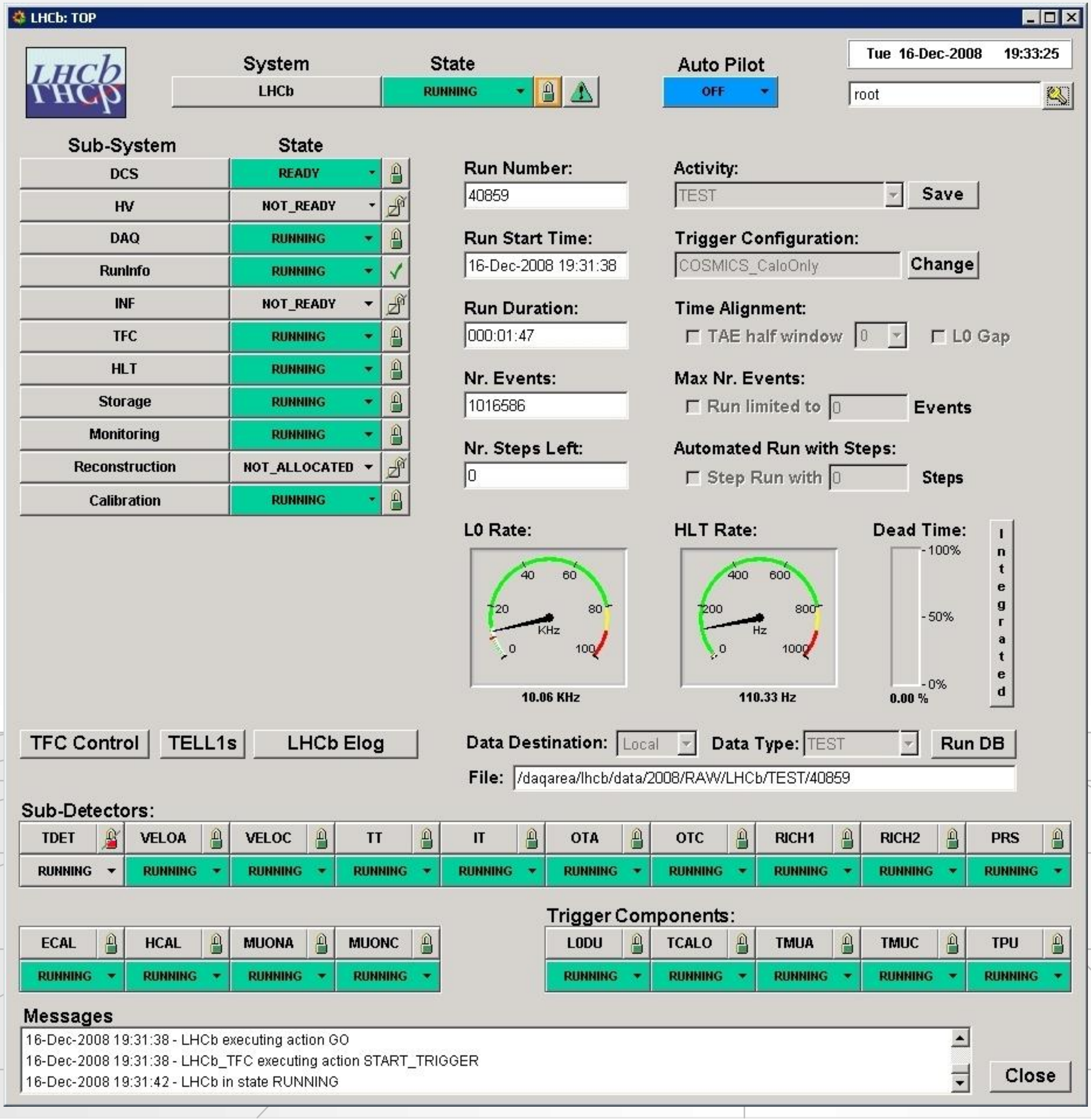
Detector Control

- The detector control system (DCS) (also Slow Control) provides the monitoring and control of the detector equipment and the experiment infrastructure.
- Due to the scale of the current and future experiments is becoming more demanding: for the LHC Experiments: ≈ 100000 parameters

Control hierarchy



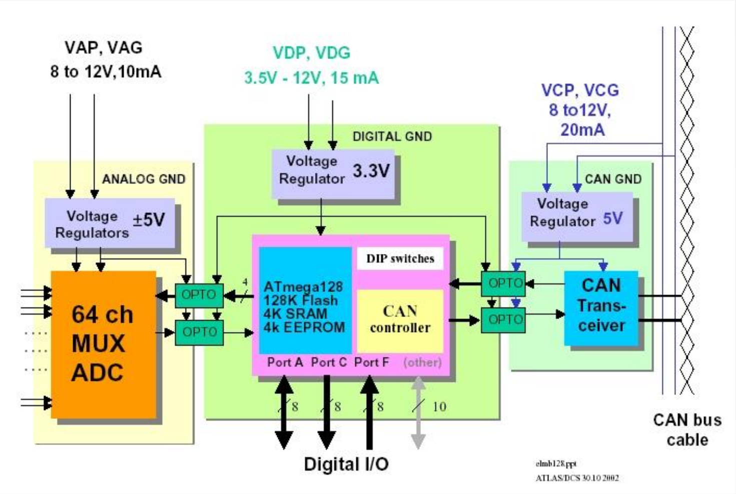
Run Control GUI

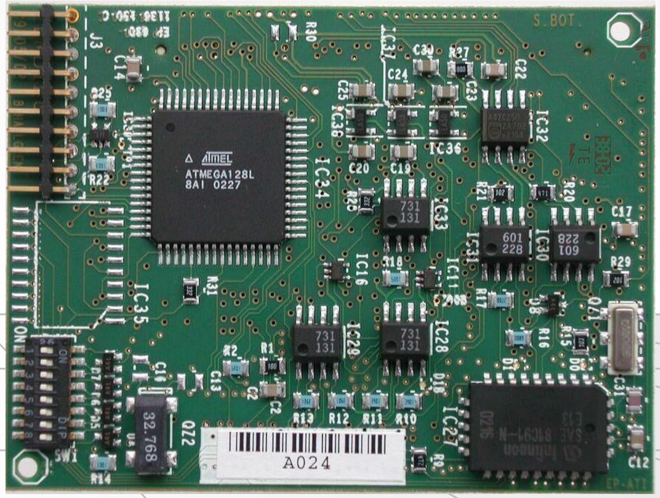


Main panel of the LHCb run-control (PVSS II)

Control and monitoring

- Access to setup registers (must have read-back)
- Access to local monitoring functions
  - Temperatures, power supply levels, errors, etc.
- Bidirectional with addressing capability (module, chip, register)
- Speed not critical and does not need to be synchronous
  - Low speed serial bus: I^2C, JTAG, SPI
- Must be reasonably reliable (read-back to check correct download and re-write when needed)





Example: ELMB

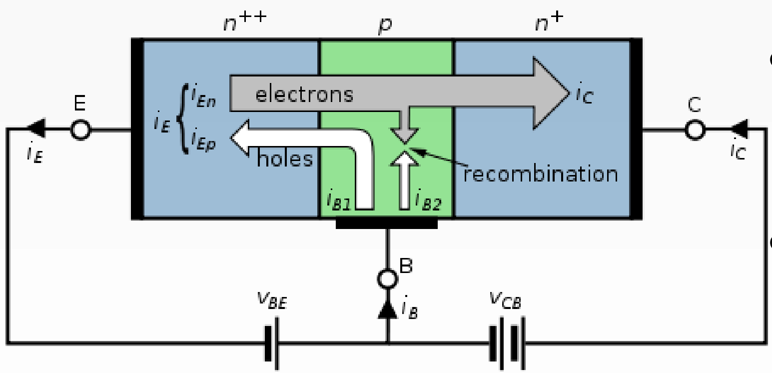
Online Trigger Farms 2009

	ALICE	ATLAS	CMS	LHCb	CERN IT
# servers	81 (1)	837	900	550	5700
# cores	324	~ 6400	7200	4400	~ 34600
total available power (kW)		~ 2000(2)	~ 1000	550	2.9 MW
currently used power (kW)		~ 250	450(3)	~ 145	2.0 MW
total available cooling power	~ 500	~ 820	800 (currently)	525	2.9 MW
total available rack-space (Us)	~ 2000	2449	~ 3600	2200	n/a
CPU type(s)	AMD Opteron	Intel Hapertown	Intel (mostly) Harpertown	Intel Harpertown	Mixed (Intel)

(1) 4-U servers with powerful FPGA preprocessor cards H-RORC
(2) Available from transformer (3) PSU rating

Even more stuff

Transistors

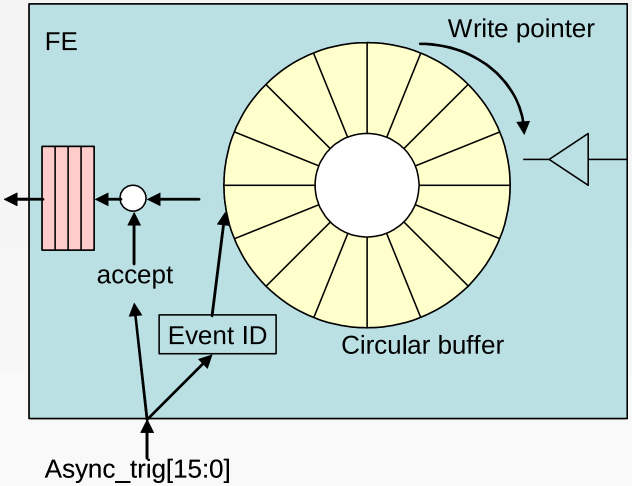


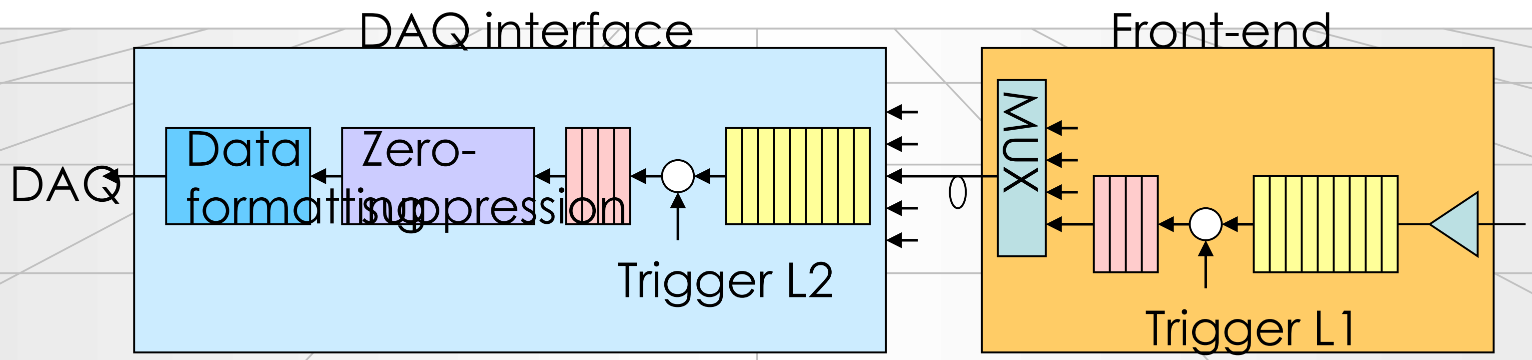
from Wikipedia

- Exampe: bi-polar transistor of the NPN type
- C collector, E emitter, B Base
- EB diode is in forward bias: holes flow towards np boundary and into n region
- BC diode is in reverse bias: electrons flow AWAY from pn boundary
- p layer must be thinner than diffusion length of electrons so that they can go through from E to N without much recombination

Multilevel triggering

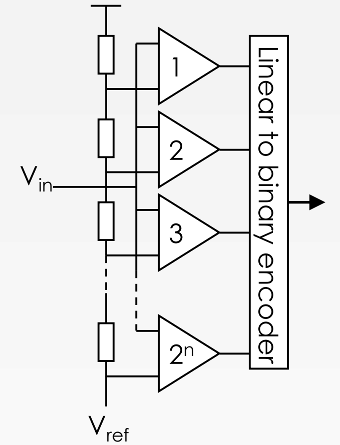
- First level triggering.
  - Hardwired trigger system to make trigger decision with short latency.
  - Constant latency buffers in the front-ends
- Second level triggering in DAQ interface
  - Processor based (standard CPU's or dedicated custom/DSP/FPGA processing)
  - FIFO buffers with each event getting accept/reject in sequential order
  - Circular buffer using event ID to extracted accepted events
    - Non accepted events stays and gets overwritten by new events
- High level triggering in the DAQ systems made with farms of CPU's: hundreds – thousands. (separate lectures on this)

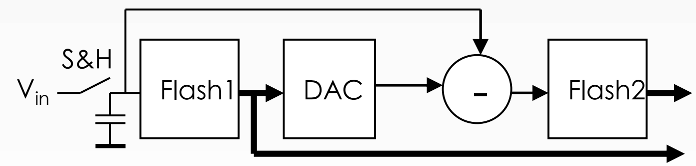




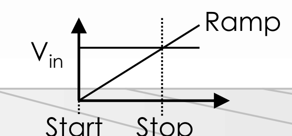
ADC architectures

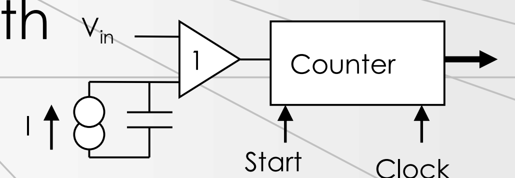
- Flash
  - A discriminator for each of the 2^n codes
  - New sample every clock cycle
  - Fast, large, lots of power, limited to ~8 bits
  - Can be split into two sub-ranging Flash $2 \times 2^{n/2}$ discriminators: e.g. 16 instead of 256 plus DAC
    - Needs sample and hold during the two stage conversion process





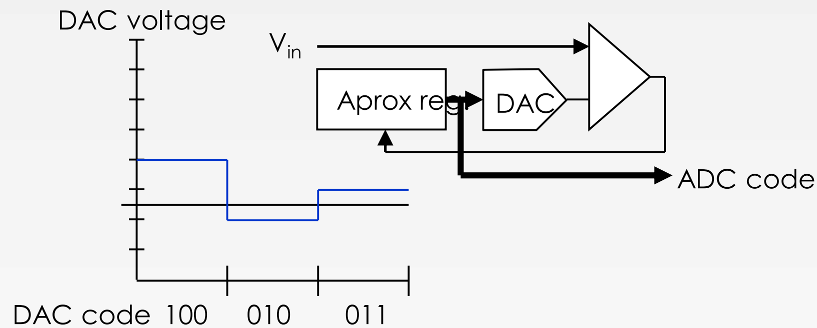
- Ramp
  - Linear analog ramp and count clock cycles
  - Takes 2^n clock cycles
  - Slow, small, low power, can be made with large resolution



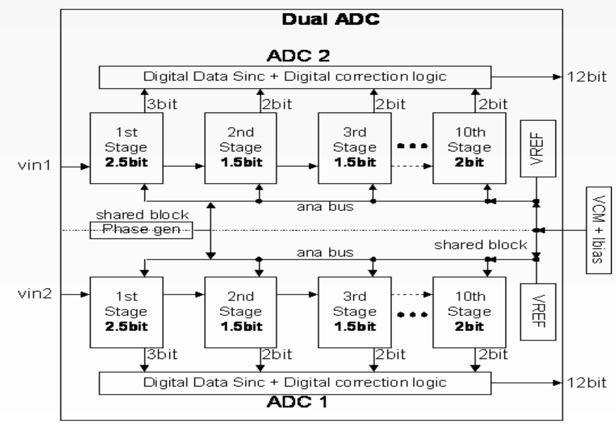


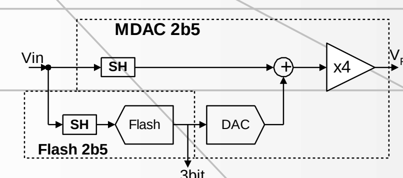
ADC architectures

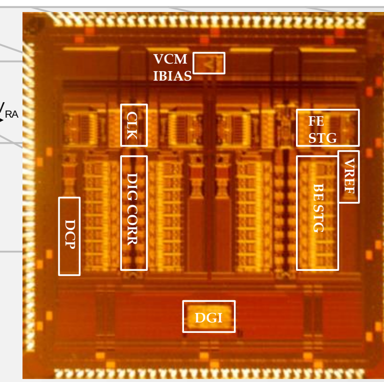
- Successive approximation
  - Binary search via a DAC and single discriminator
  - Takes n clock cycles
  - Relatively slow, small, low power, medium to large resolution



- **Pipelined**
  - Determines "one bit" per clock cycle per stage
    - Extreme type of sub ranging flask
  - n stages
  - In principle 1 bit per stage but to handle imperfections each stage normally made with ~2bits and n*2bits mapped into n bits via digital mapping function that "auto corrects" imperfections
  - Makes a conversion each clock cycle
  - Has a latency of n clock cycles
    - Not a problem in our applications except for very fast triggering
  - Now dominating ADC architecture in modern CMOS technologies and impressive improvements in the last 10 years: speed, bits, power, size







ADC imperfections

- Quantization (static)
  - Bin size: Least significant bit (LSB) $=V_{max}/2^n$
  - Quantization error: RMS error/resolution: $LSB/\sqrt{12}$
- Integral non linearity (INL): Deviation from ideal conversion curve (static)
  - Max: Maximum deviation from ideal
  - RMS: Root mean square of deviations from ideal curve
- Differential non linearity (DNL): Deviation of quantization steps (static)
  - Min: Minimum value of quantization step
  - Max: Maximum value of quantization step
  - RMS: Root mean square of deviations from ideal quantization step
- Missing codes (static)
  - Some binary codes never present in digitized output
- Monotonic (static)
  - Non monotonic conversion can be quite unfortunate in some applications. A given output code can correspond to several input values.