UNIVERSITÀ DI PAVIA

INFN Istituto Nazionale di Fisica Nucleare

Introduction to Data AcQuisition

ISOTDAQ 2018: 9th International School of Trigger and Data Acquisition

Vienna, 14 Feb 2017

Andrea.Negri@pv.infn.it

Acknowledgment

- Lecture inherited from **Wainer Vandelli**
  - Material and ideas have been taken from:
    - CERN Summer Student lectures, **N.Neufeld** and **C.Gaspar**
    - the “Physics data acquisition and analysis” lessons given by **R.Ferrari** at the University of Parma, Italy
- Errors and flaws are mine

Introduction

- Aim of this lesson is to introduce the **basic DAQ concepts** avoiding as many technological details as possible
  - The following lectures will cover these aspects
  - I'll mostly refer to DAQ in High-Energy Physics

Outline

- Introduction
  - What is DAQ?
  - Overall framework
- Basic DAQ concepts
  - Digitization, Latency
  - Deadtime, Busy, Backpressure
  - De-randomization
- Scaling up
  - Readout and Event Building
  - Buses vs Network
- Do it yourself

DAQ

- Wikipedia: data acquisition (DAQ) is the process of sampling signals that measure real world physical conditions and converting the resulting samples into digital numeric values that ….
- **D**ata **A**c**Q**uisition is an heterogeneous field
  - Boundaries not well defined
  - An alchemy of physics, electronics, networking, computer science, …
  - Hacking and experience
  - …, money and manpower matter as well

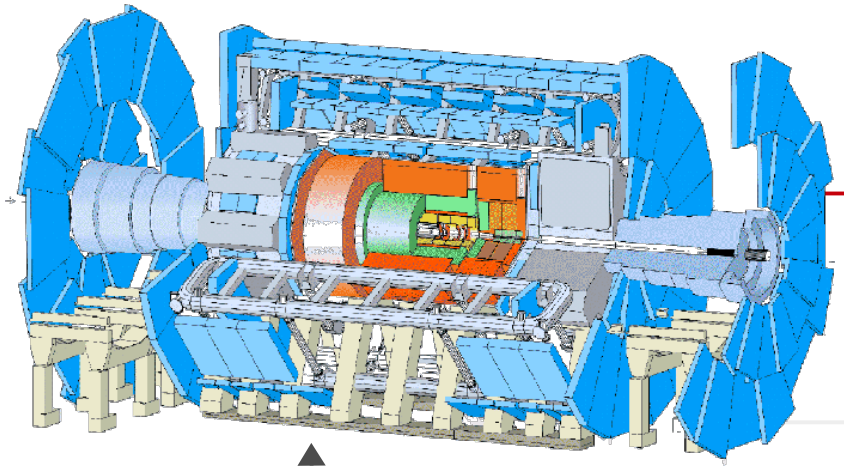
DAQ

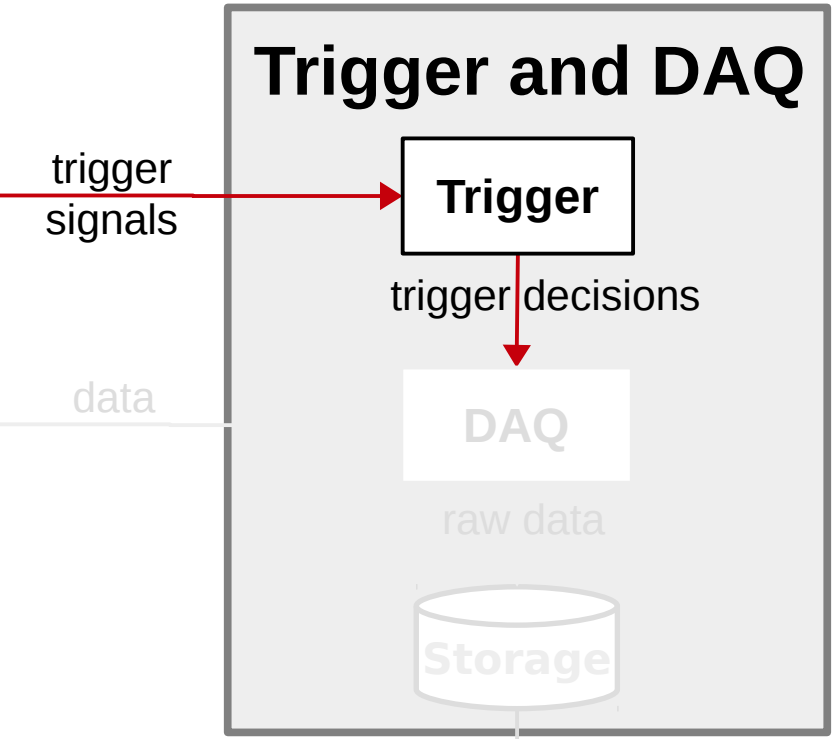
- Wikipedia: data acquisition (DAQ) is the process of sampling signals that measure real world physical conditions and converting the resulting samples into digital numeric values that ….

- **D**ata **A**c**Q**uisition is an heterogeneous field
  - Boundaries not well defined
  - An alchemy of physics, electronics, networking, computer science, …
  - Hacking and experience
  - …, money and manpower matter as well

Overview

- Overall the main role of T & DAQ is to process the signals generated in a detector and saving the interesting information on a permanent storage



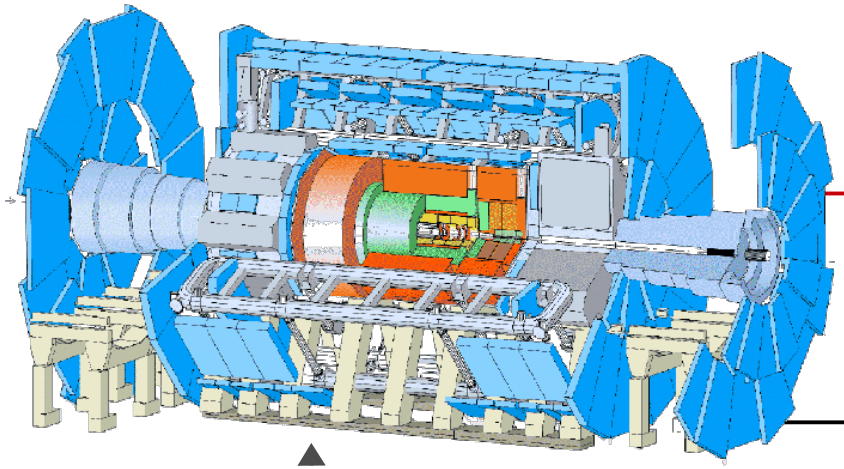


design feedback

Detector & Trigger Simulation → **Reconstruction & Analysis** → **Physics Results**

Overview

- Overall the main role of T & DAQ is to process the signals generated in a detector and saving the interesting information on a permanent storage



Trigger & DAQ

- Trigger — Alessandro's talk (intro) / Mandred's talk (HW)
  - Either selects interesting events or rejects boring ones, in real time
  - i.e. with minimal *controlled* **latency**
    - time it takes to form and distribute its decision

- DAQ
  - Gathers data produced by detectors: **Readout**
  - Forms complete events: **Data Collection** and **Event Building**
  - Possibly feeds several trigger levels: **HLT**
  - Stores event data: **Data Logging**

  } **Data Flow**

  - Provides **Run Control**, **Configuration**, **Monitoring**

Trigger & DAQ

- Trigger — Alessandro's talk (intro); Mandred's talk (HW)
  - Either selects interesting events or rejects boring ones, in real time
  - i.e. with minimal *controlled* **latency**
    - time it takes to form and distribute its decision

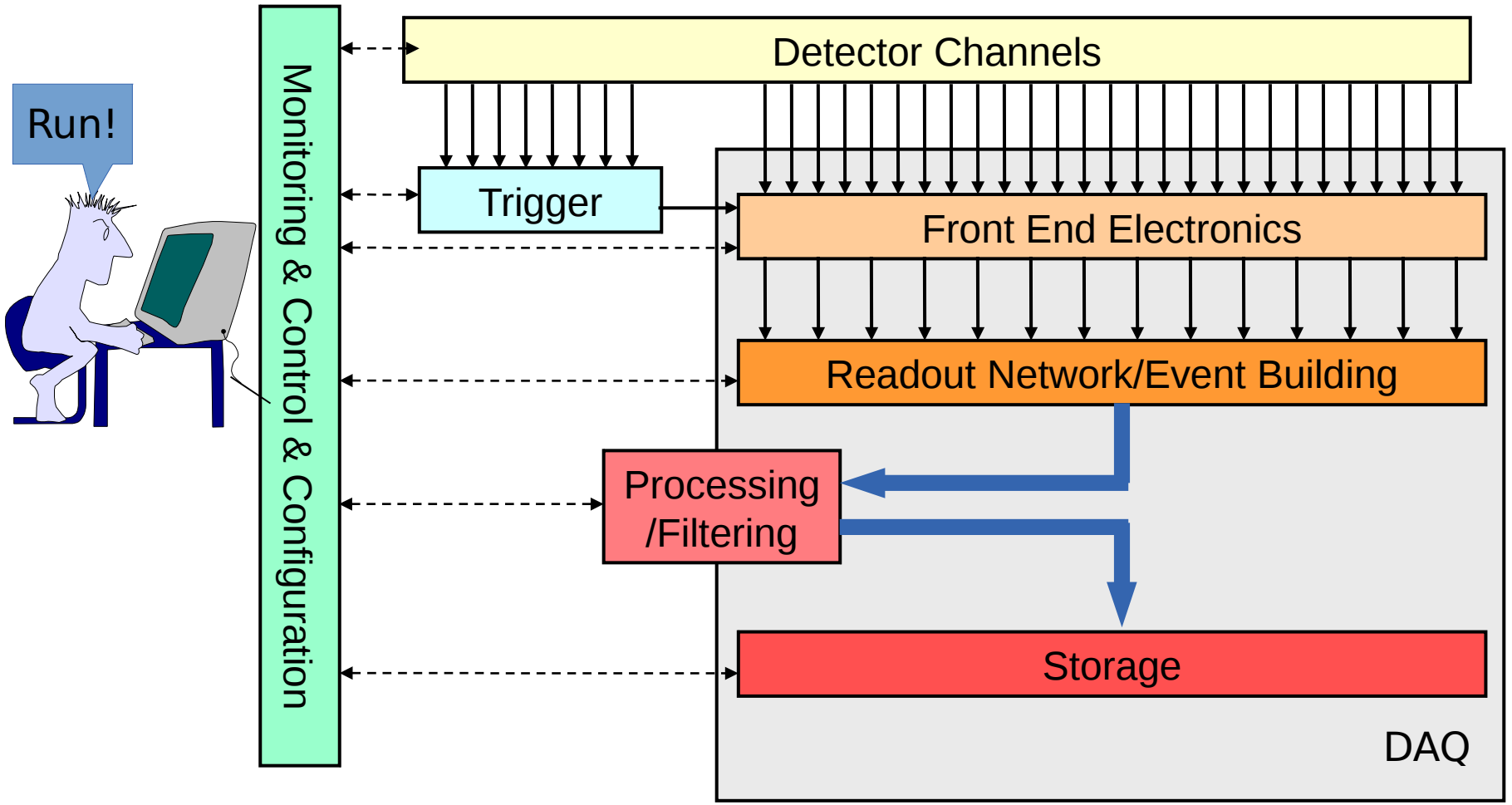
- DAQ — Kostas' talk (HW)
  - Gathers data produced by detectors: **Readout** — Gokhan's talk
  - Forms complete events: **Data Collection** and **Event Building**
  - Possibly feeds several trigger levels: **HLT**
  - Stores event data: **Data Logging**

Data Flow

  - Provides **Run Control**, **Configuration**, **Monitoring** — Serguei's talk

Lab 12

Trigger, DAQ and Controls

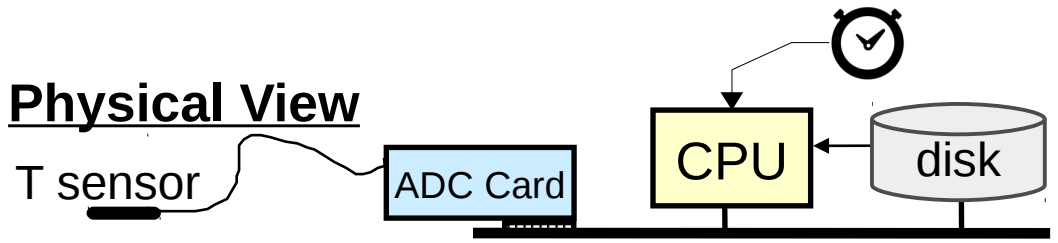


Outline

- Introduction
  - What is DAQ?
  - Overall framework
- Basic DAQ concepts
  - Digitization, Latency
  - Deadtime, Busy, Backpressure
  - De-randomization
- Scaling up
  - Readout and Event Building
  - Buses vs Network
- Do it yourself

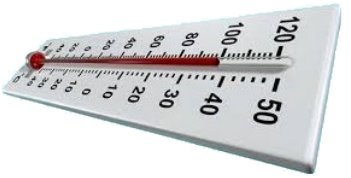
Basic DAQ: periodic trigger

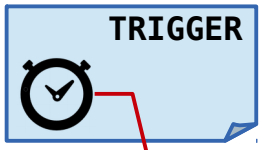
- Eg: measure temperature at a fixed frequency
  - ADC performs analog to digital conversion, digitization (our front-end electronics)
  - CPU does readout and processing
- System clearly limited by the time τ to process an “event”
  - ADC conversion + CPU processing + Storage
- The DAQ maximum sustainable rate is simply the inverse of τ, e.g.:
  - $\tau = 1$ ms $\rightarrow R = 1/\tau = 1$ kHz

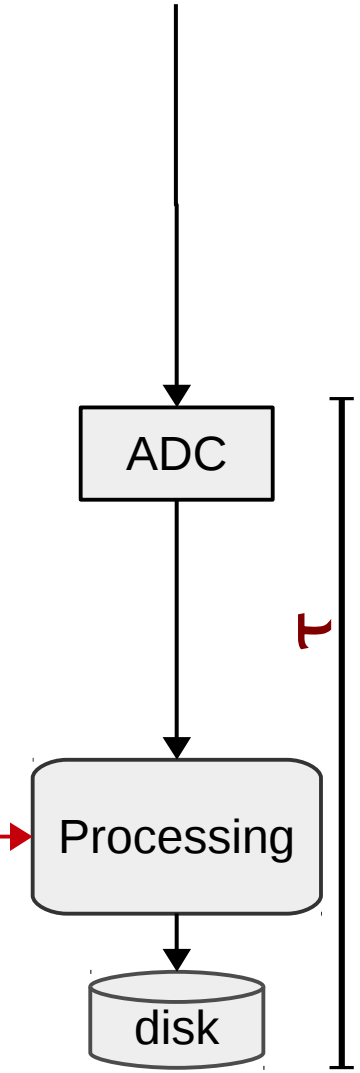


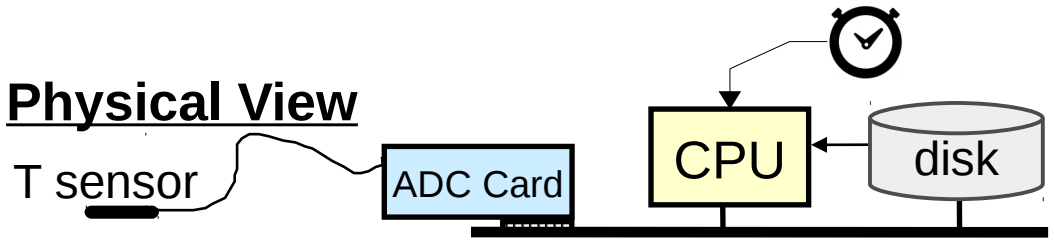
Basic DAQ: periodic trigger

- Eg: measure temperature at a fixed frequency
  - ADC performs analog to digital conversion, digitization (our front-end electronics)
  - CPU does readout and processing
- System clearly limited by the time τ to process an "event"
  - ADC conversion + CPU processing + Storage
- The DAQ maximum sustainable rate is simply the inverse of τ, e.g.:
  - $\tau = 1$ ms $\rightarrow R = 1/\tau = 1$ kHz



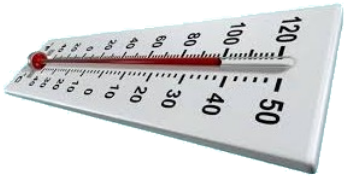


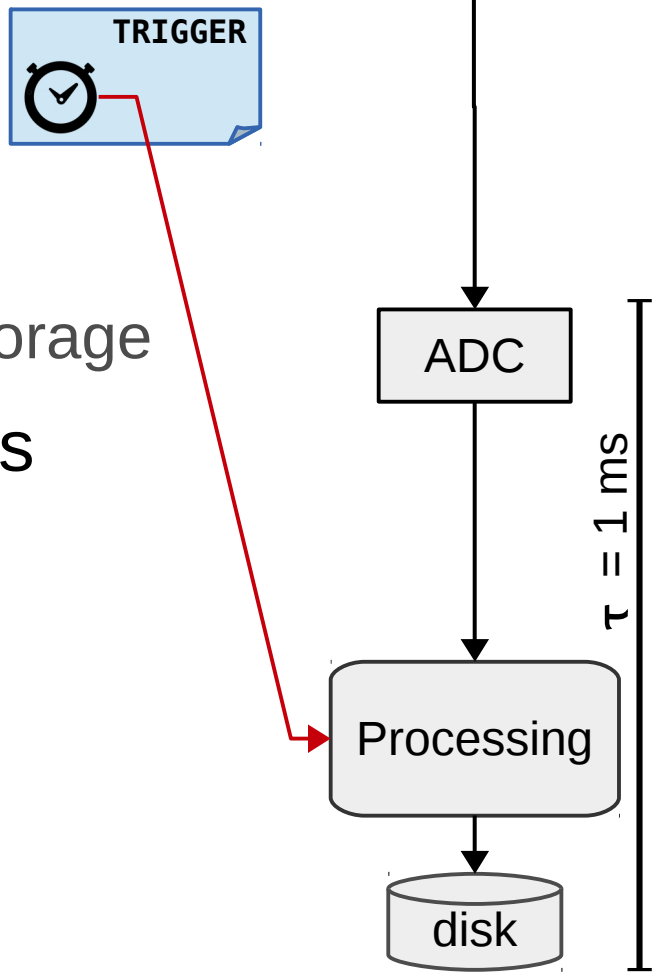


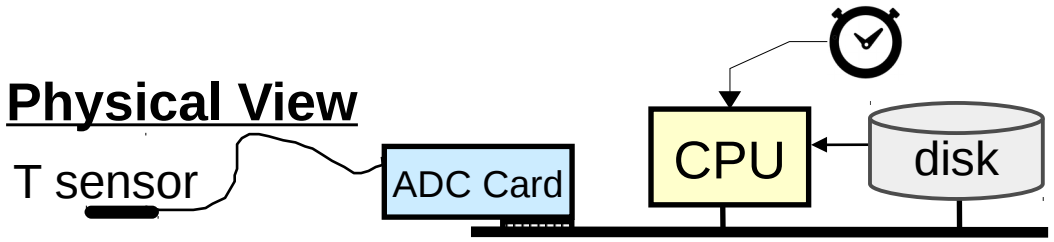


Basic DAQ: periodic trigger

- Eg: measure temperature at a fixed frequency
  - ADC performs analog to digital conversion, digitization (our front-end electronics)
  - CPU does readout and processing
- System clearly limited by the time **τ** to process an "event"
  - ADC conversion + CPU processing + Storage
- The DAQ maximum sustainable rate is simply the inverse of **τ**, e.g.:
  - $\tau = 1$ ms $\rightarrow R = 1/\tau = 1$ kHz



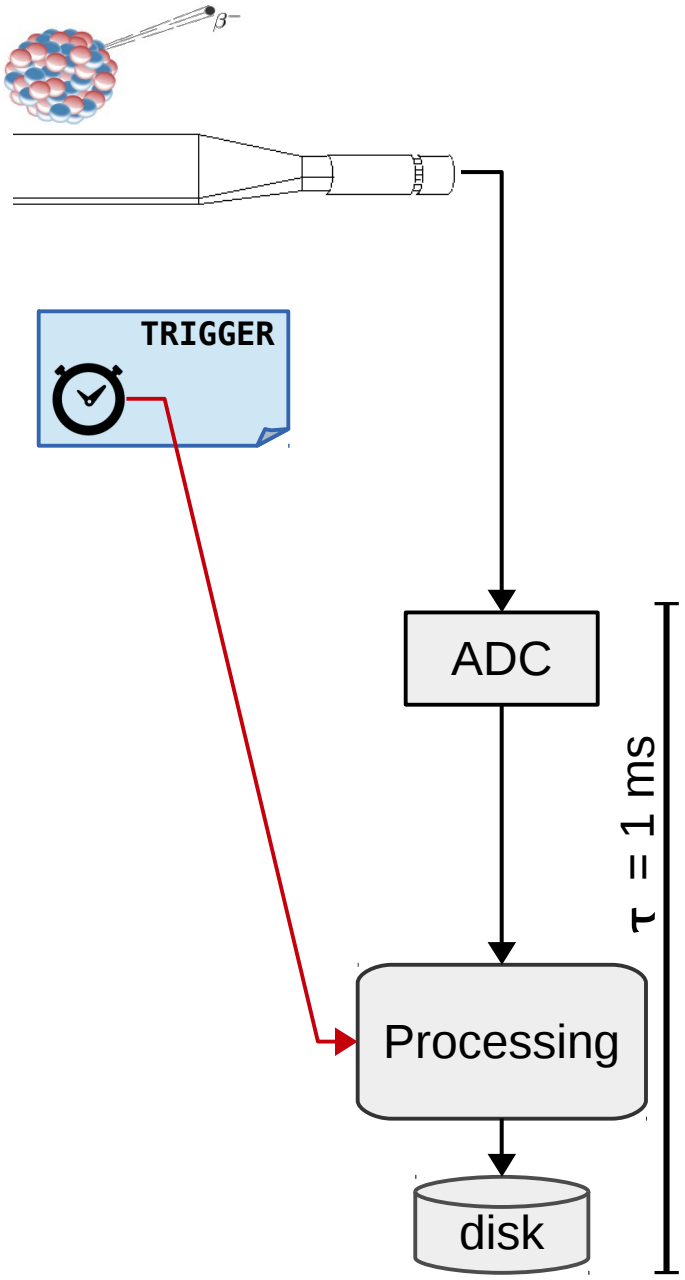




Basic DAQ: “real” trigger

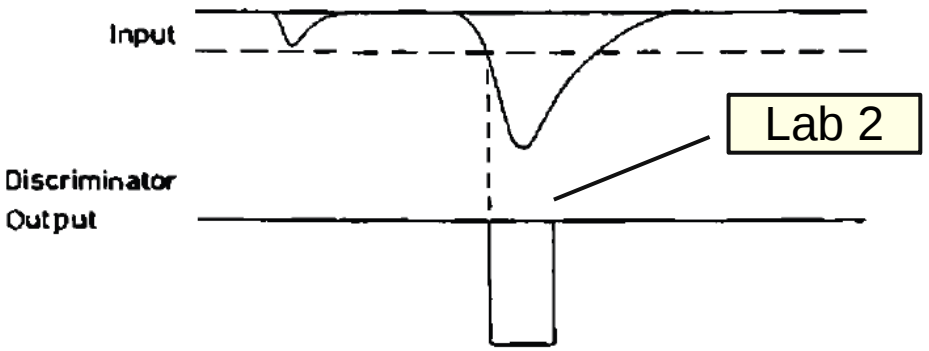
- Events asynchronous and unpredictable
  - E.g.: beta decay studies
- A physics trigger is needed
  - **Discriminator**: generate an output signal only if amplitude of input pulse is greater than a given threshold

  - delay introduced to compensate for the **trigger latency**

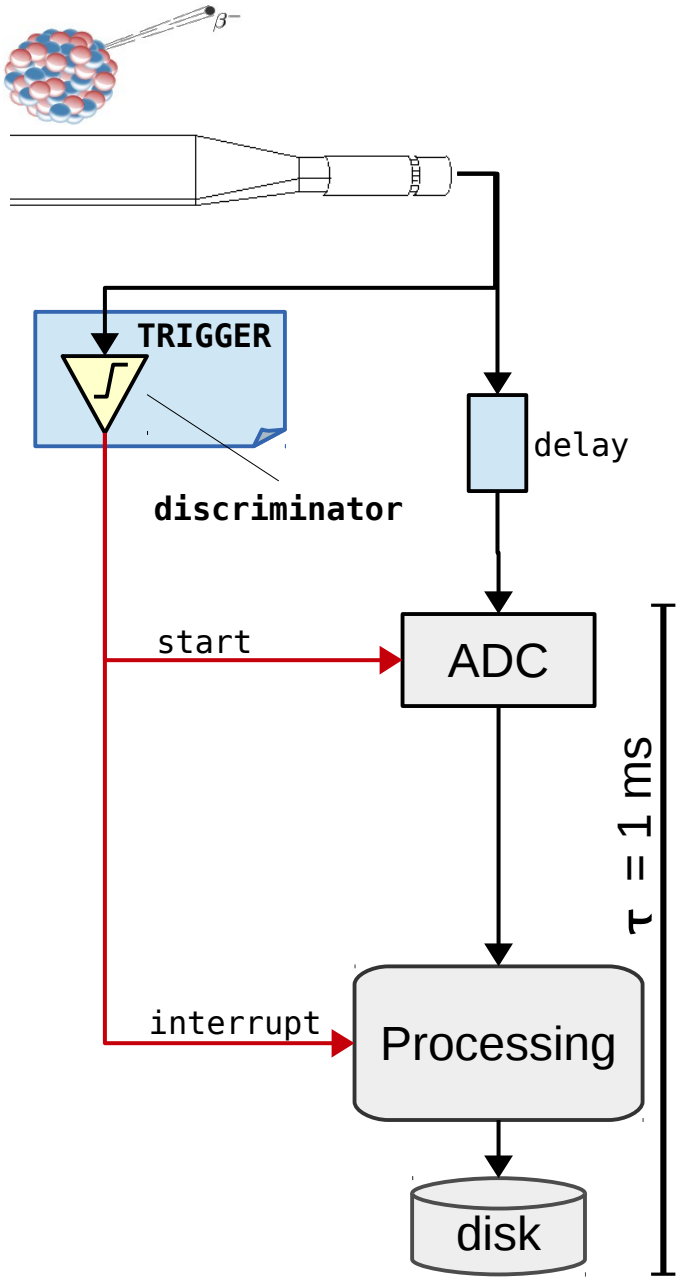


Basic DAQ: “real” trigger

- Events asynchronous and unpredictable
  - E.g.: beta decay studies
- A physics trigger is needed
  - **Discriminator**: generate an output signal only if amplitude of input pulse is greater than a given threshold

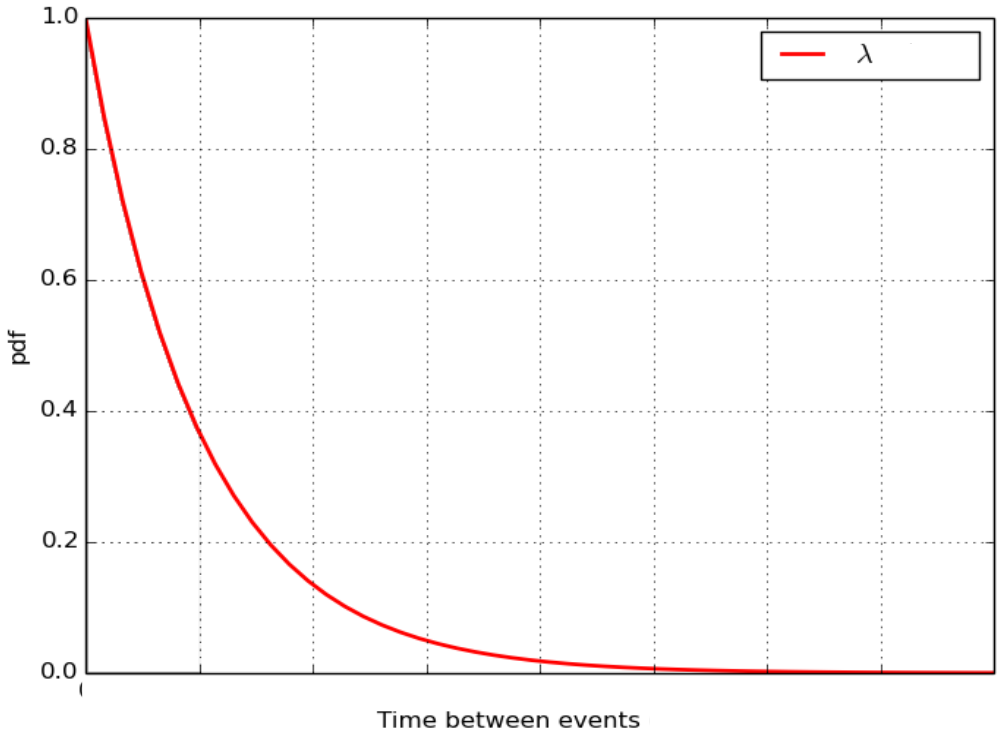


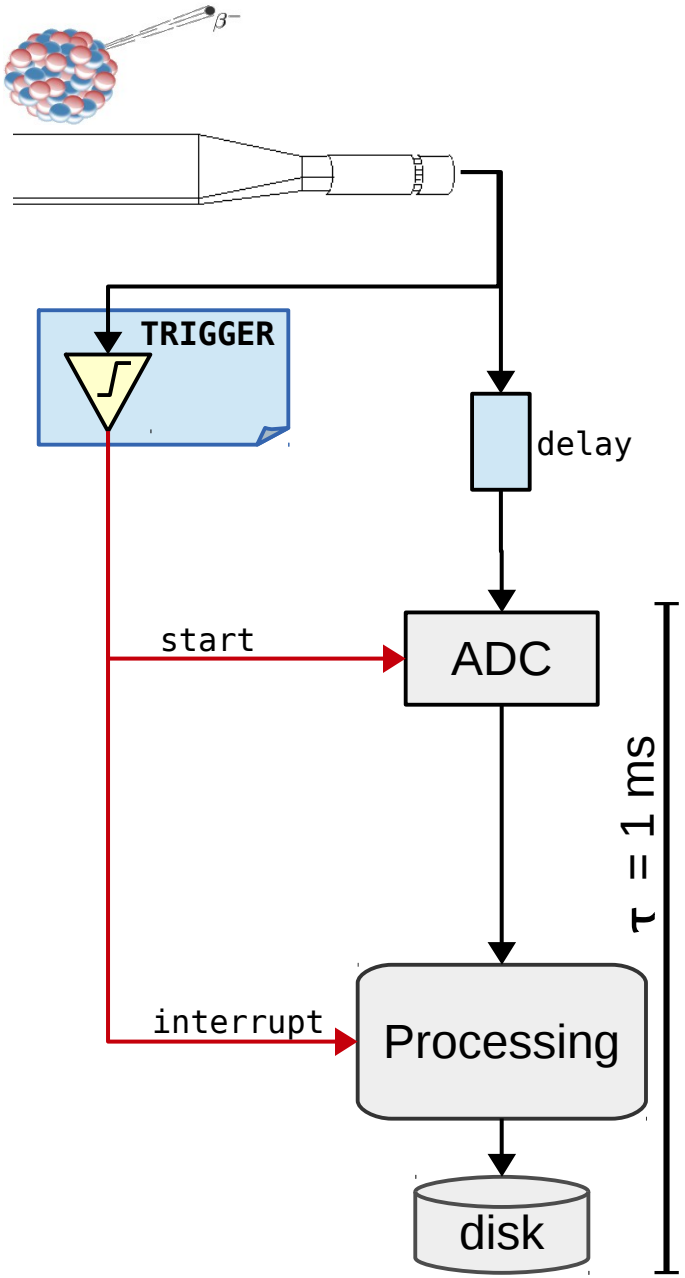
  - delay introduced to compensate for the **trigger latency**



Basic DAQ: “real” trigger

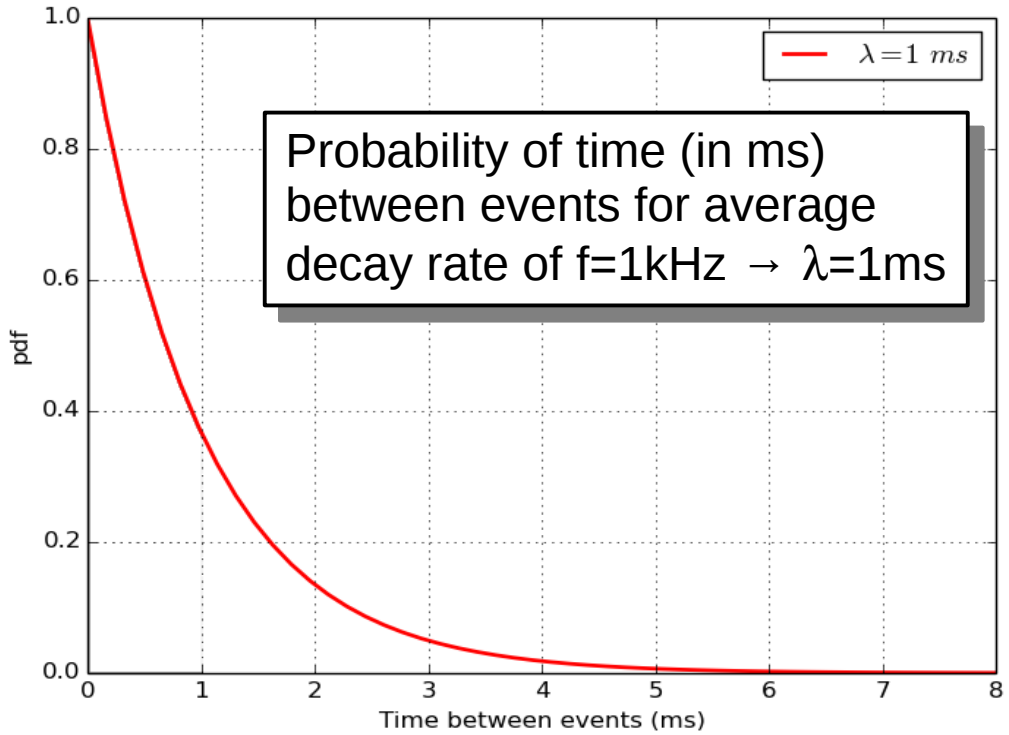
- Stochastic process
  - Fluctuations in time between events
- Let's assume for example
  - a process rate f = 1 kHz, i.e. λ = 1 ms
  - and, as before, τ = 1 ms

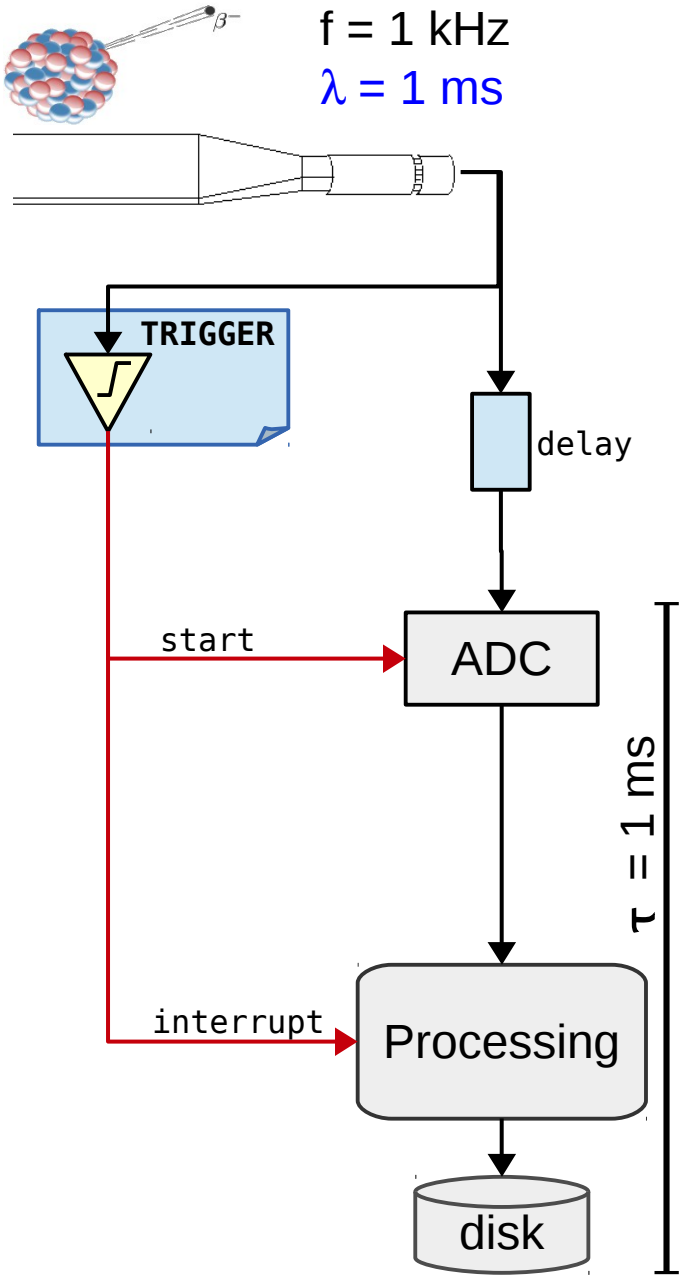




Basic DAQ: “real” trigger

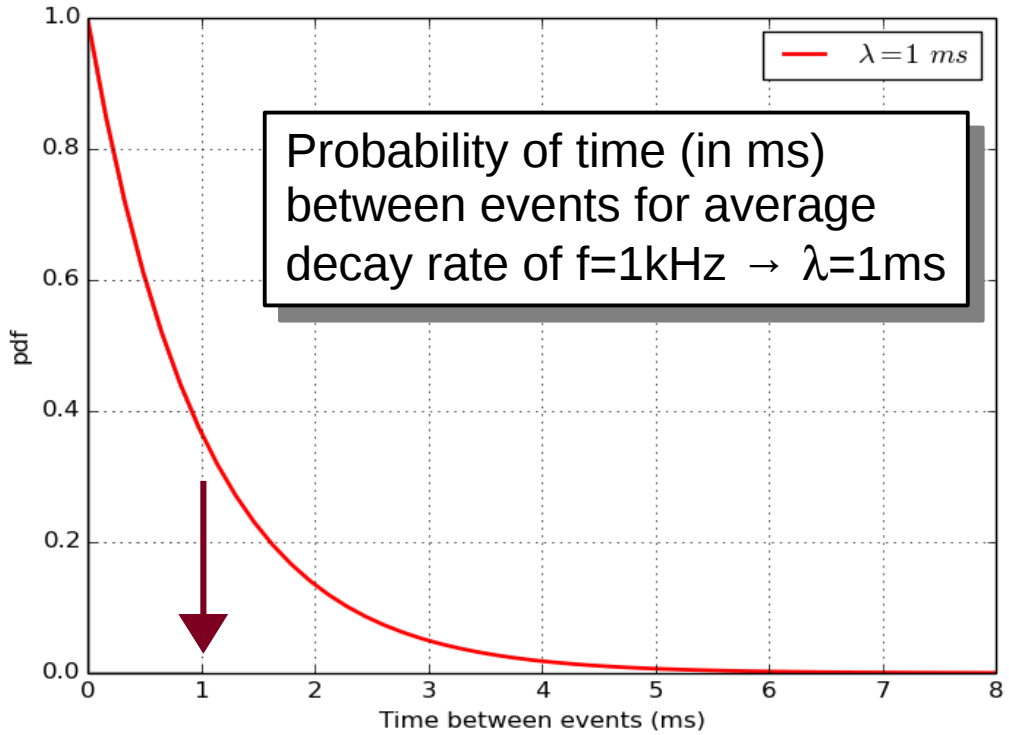
- Stochastic process
  - Fluctuations in time between events
- Let's assume for example
  - a process rate f = 1 kHz, i.e. $\lambda = 1$ ms
  - and, as before, $\tau = 1$ ms

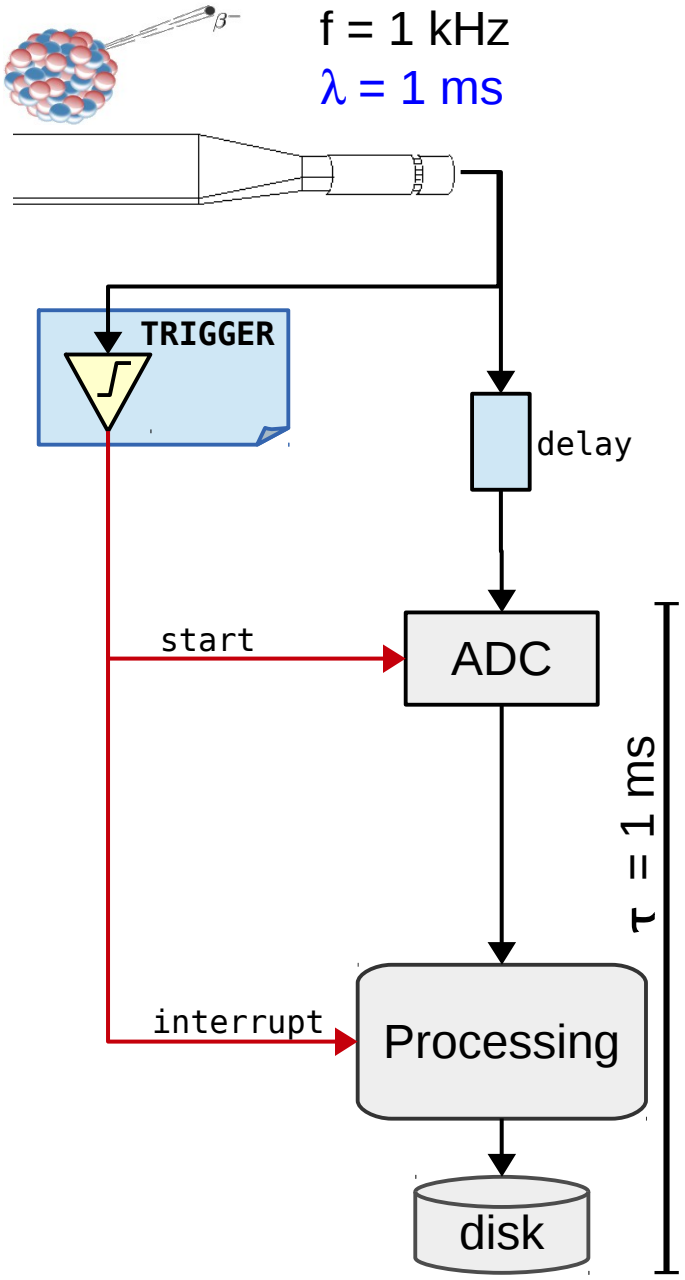




Basic DAQ: “real” trigger

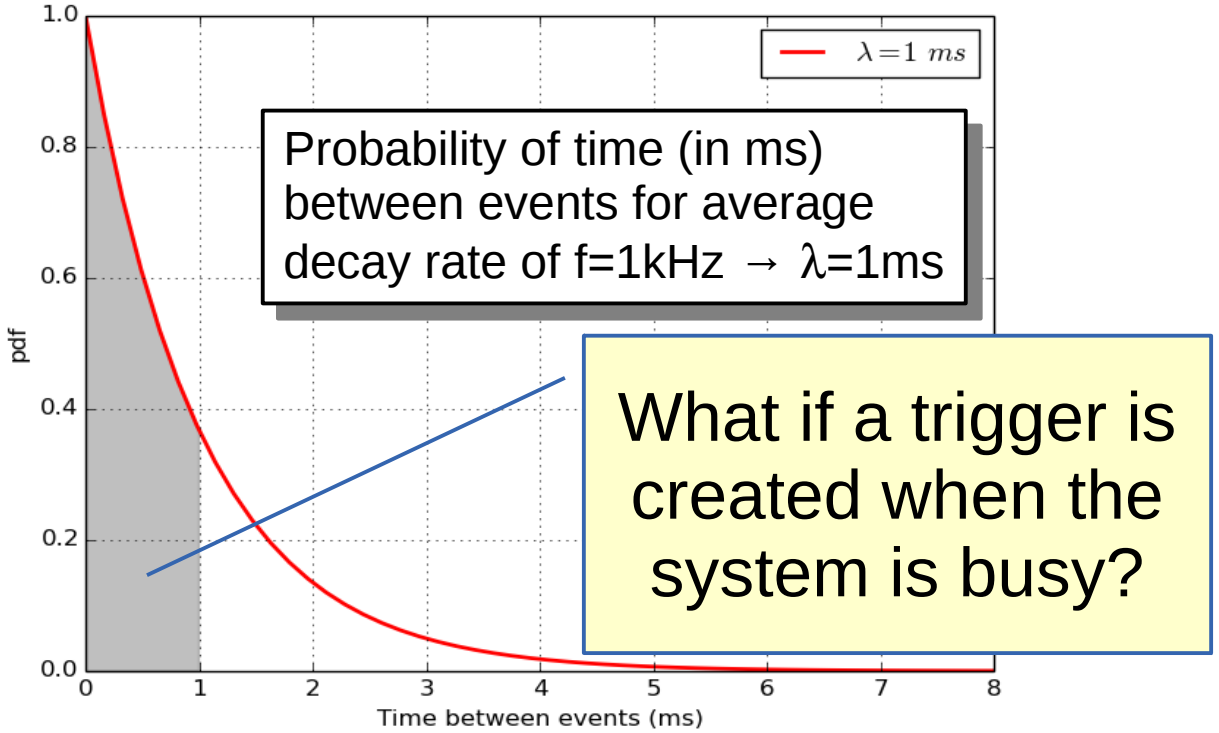
- Stochastic process
  - Fluctuations in time between events
- Let's assume for example
  - a process rate f = 1 kHz, i.e. λ = 1 ms
  - and, as before, τ = 1 ms

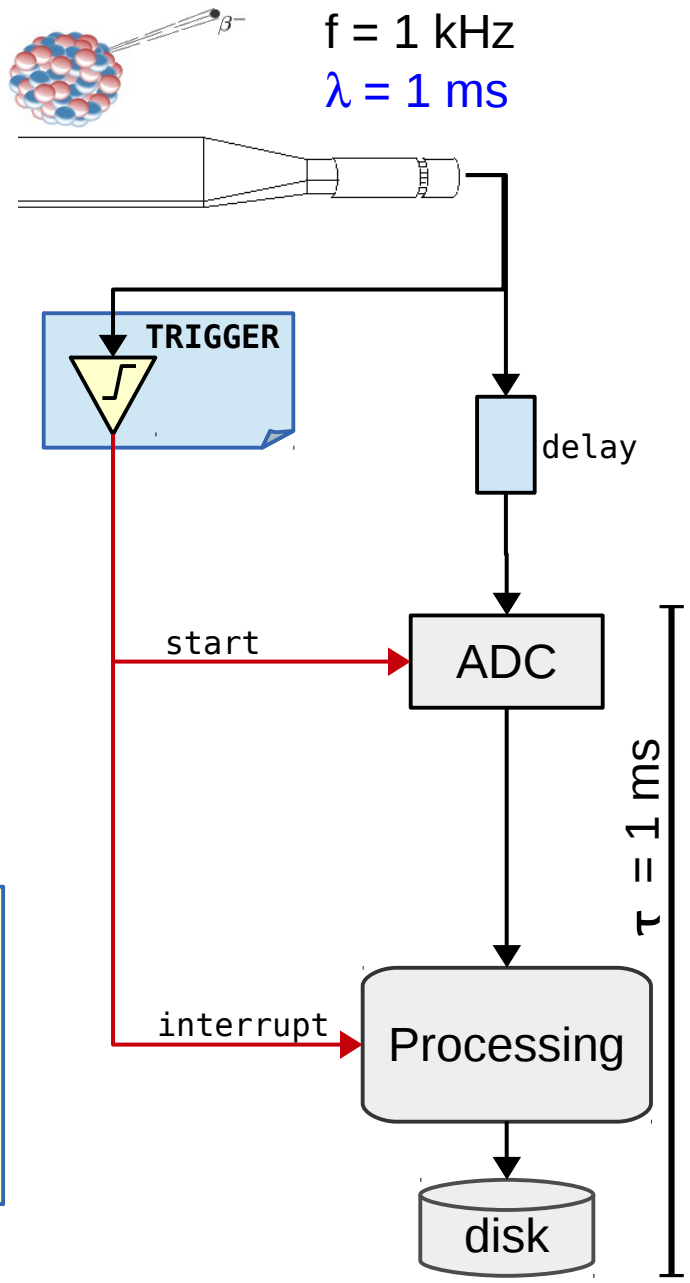




Basic DAQ: “real” trigger

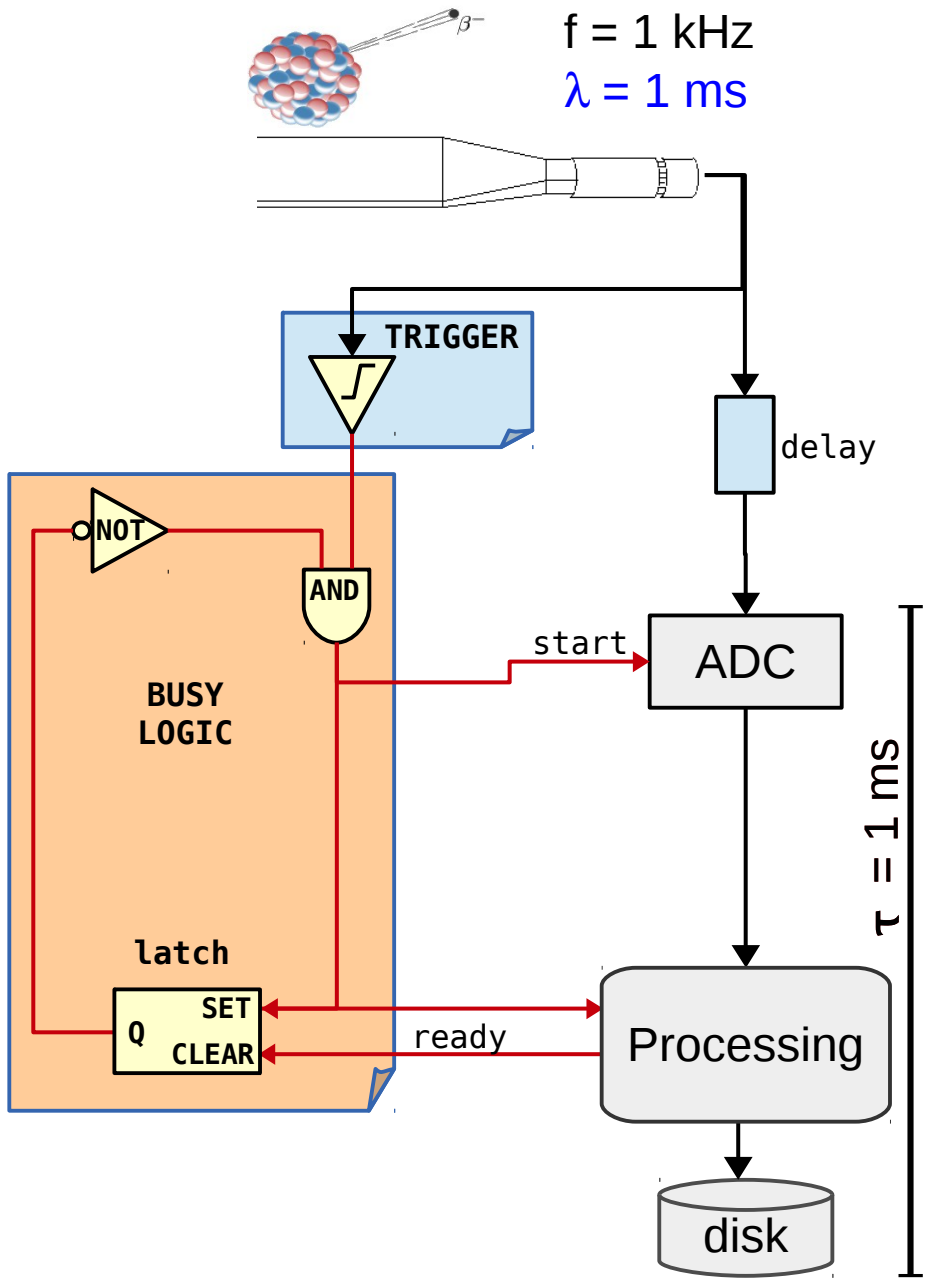
- Stochastic process
  - Fluctuations in time between events
- Let's assume for example
  - a process rate f = 1 kHz, i.e. $\lambda = 1$ ms
  - and, as before, $\tau = 1$ ms





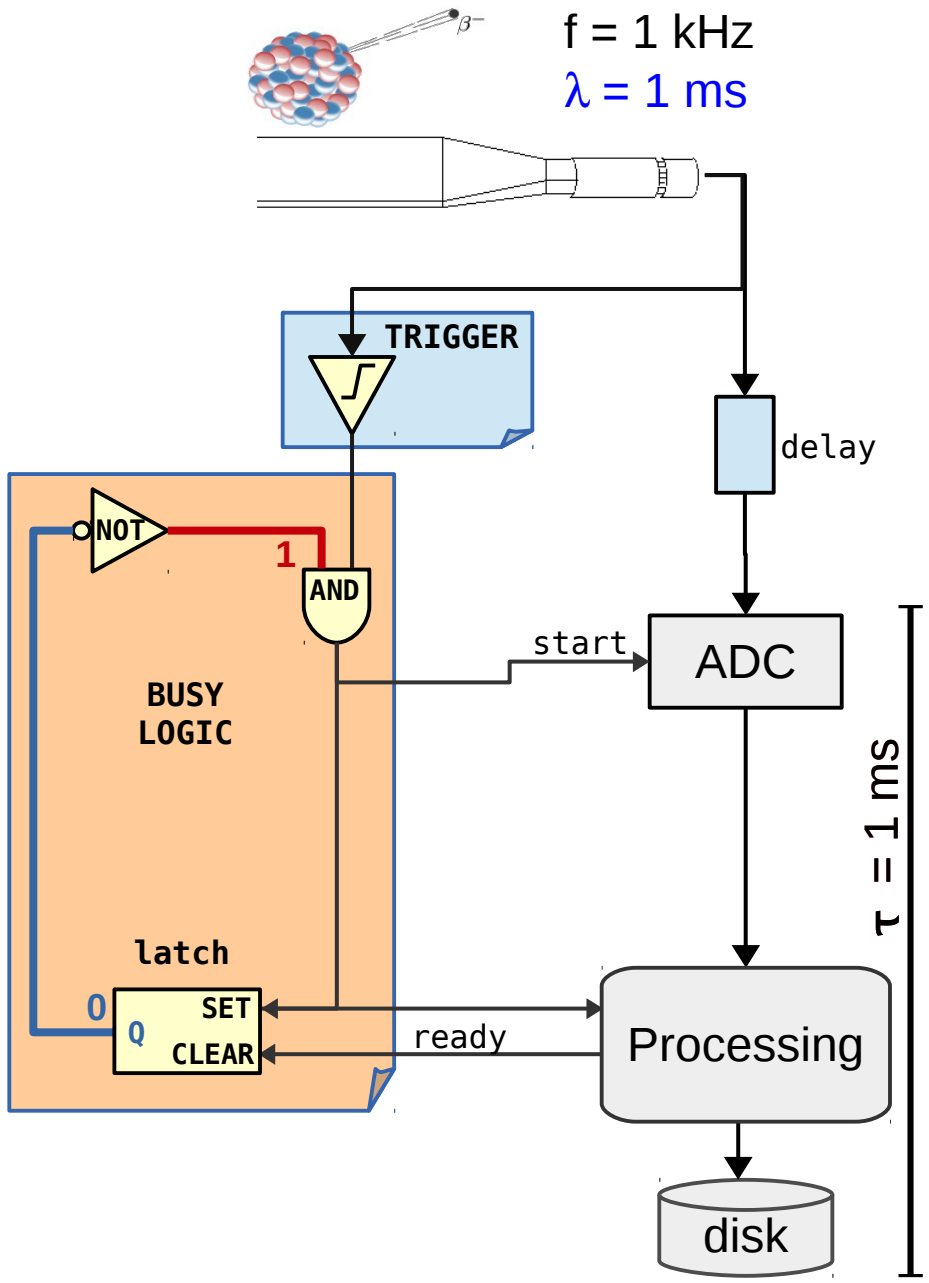
Basic DAQ: “real” trigger

- **Busy logic** avoids triggers while the system is busy in processing
  - E.g.: AND gate and a latch
- Latch (**flip-flop**):
  - a bistable circuit that changes state (Q) by signals applied to the control inputs (SET, CLEAR)



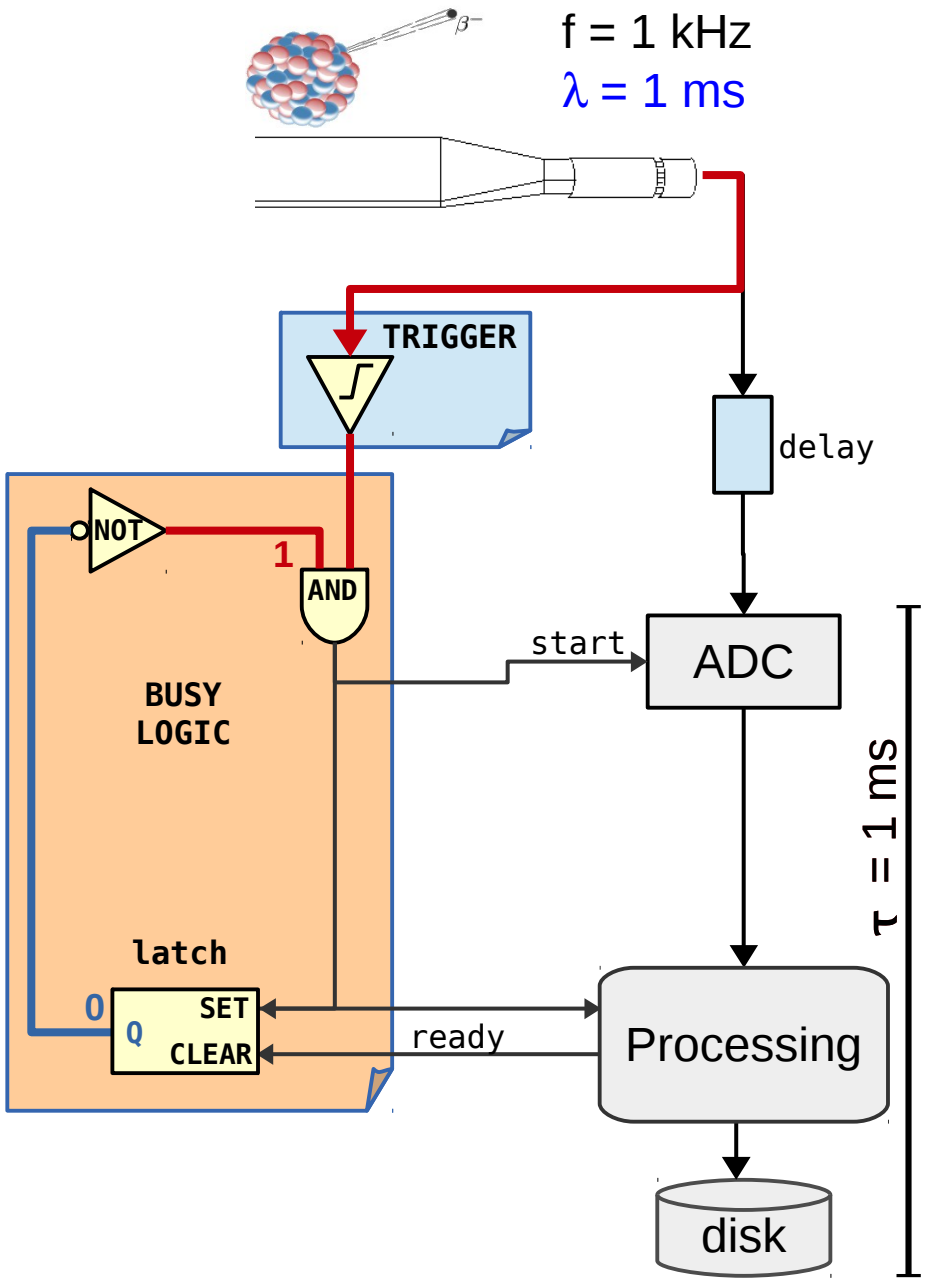
Basic DAQ: “real” trigger

- **Busy logic** avoids triggers while the system is busy in processing
  - E.g.: AND gate and a latch
- Latch (**flip-flop**):
  - a bistable circuit that changes state (Q) by signals applied to the control inputs (SET, CLEAR)
  - At the beginning the flip-flop state is down and so one input of the AND gate is always up
    - via the NOT gate



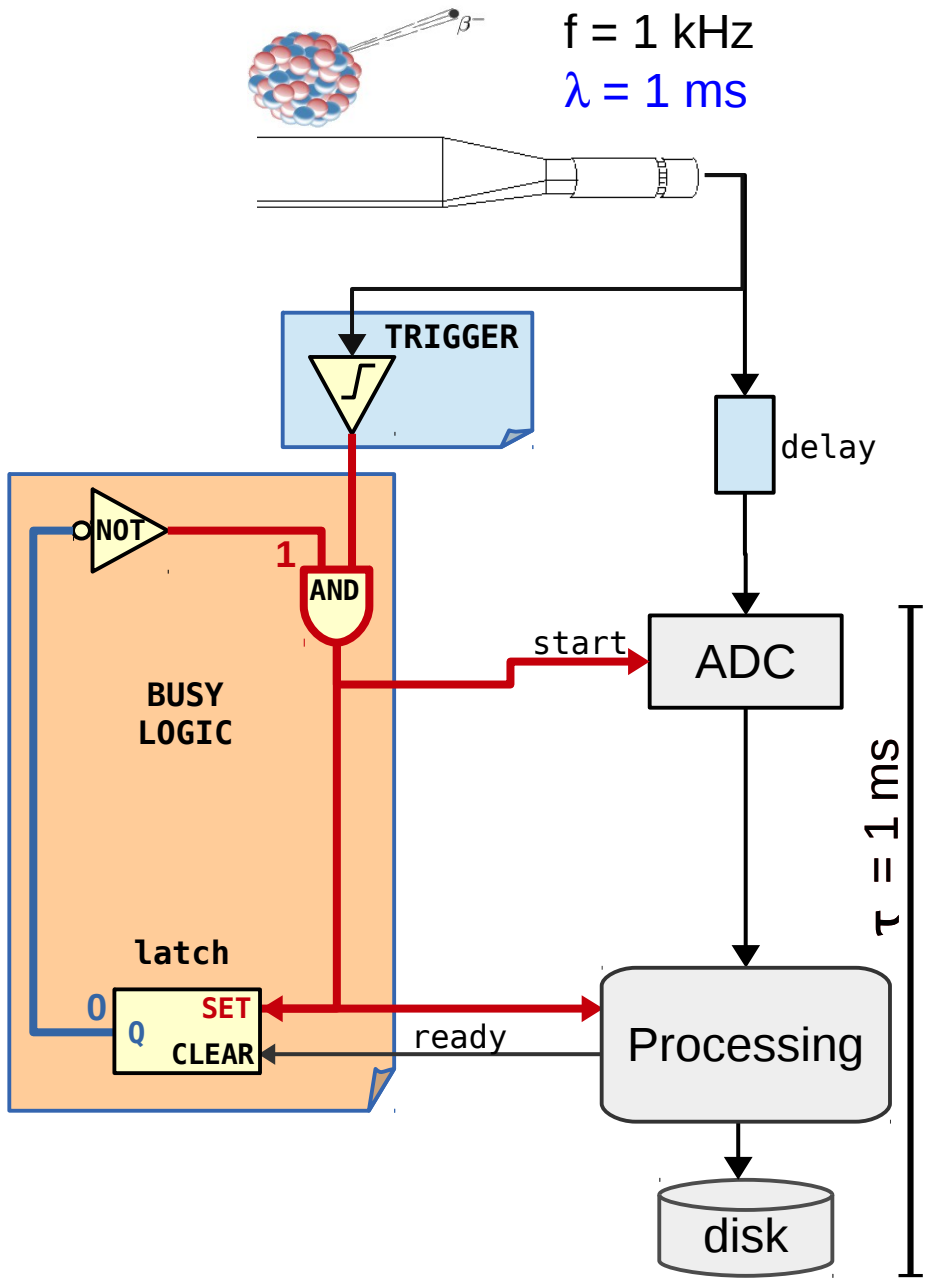
Basic DAQ: “real” trigger

- **Busy logic** avoids triggers while the system is busy in processing
  - E.g.: AND gate and a latch
- Latch (**flip-flop**):
  - a bistable circuit that changes state (Q) by signals applied to the control inputs (SET, CLEAR)
  - If a trigger arrives, it finds the AND gate open
    - ADC is started
    - Flip-flop is flipped



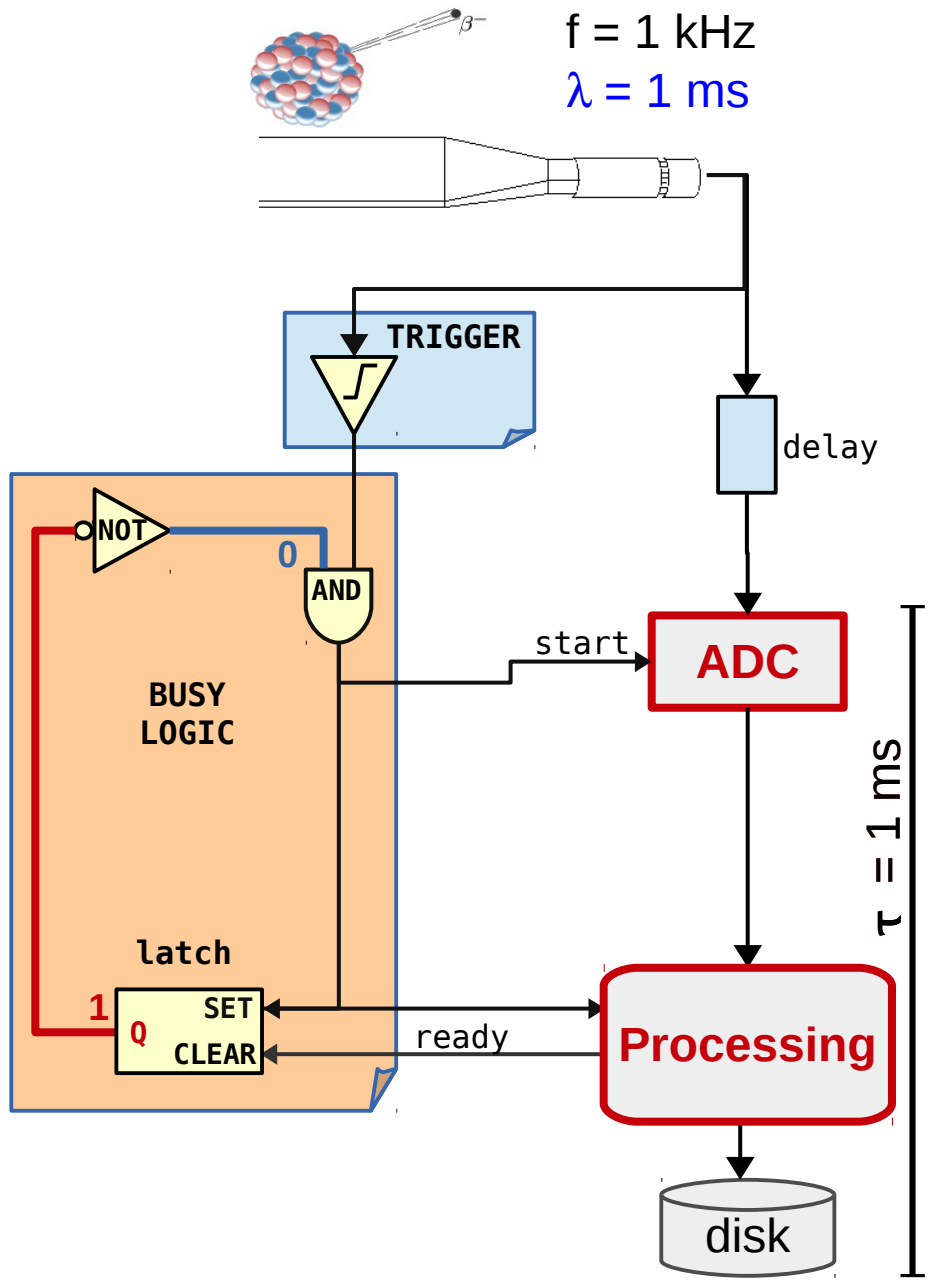
Basic DAQ: “real” trigger

- **Busy logic** avoids triggers while the system is busy in processing
  - E.g.: AND gate and a latch
- Latch (**flip-flop**):
  - a bistable circuit that changes state (Q) by signals applied to the control inputs (SET, CLEAR)
  - If a trigger arrives, it finds the AND gate open
    - ADC is started
    - Flip-flop is flipped



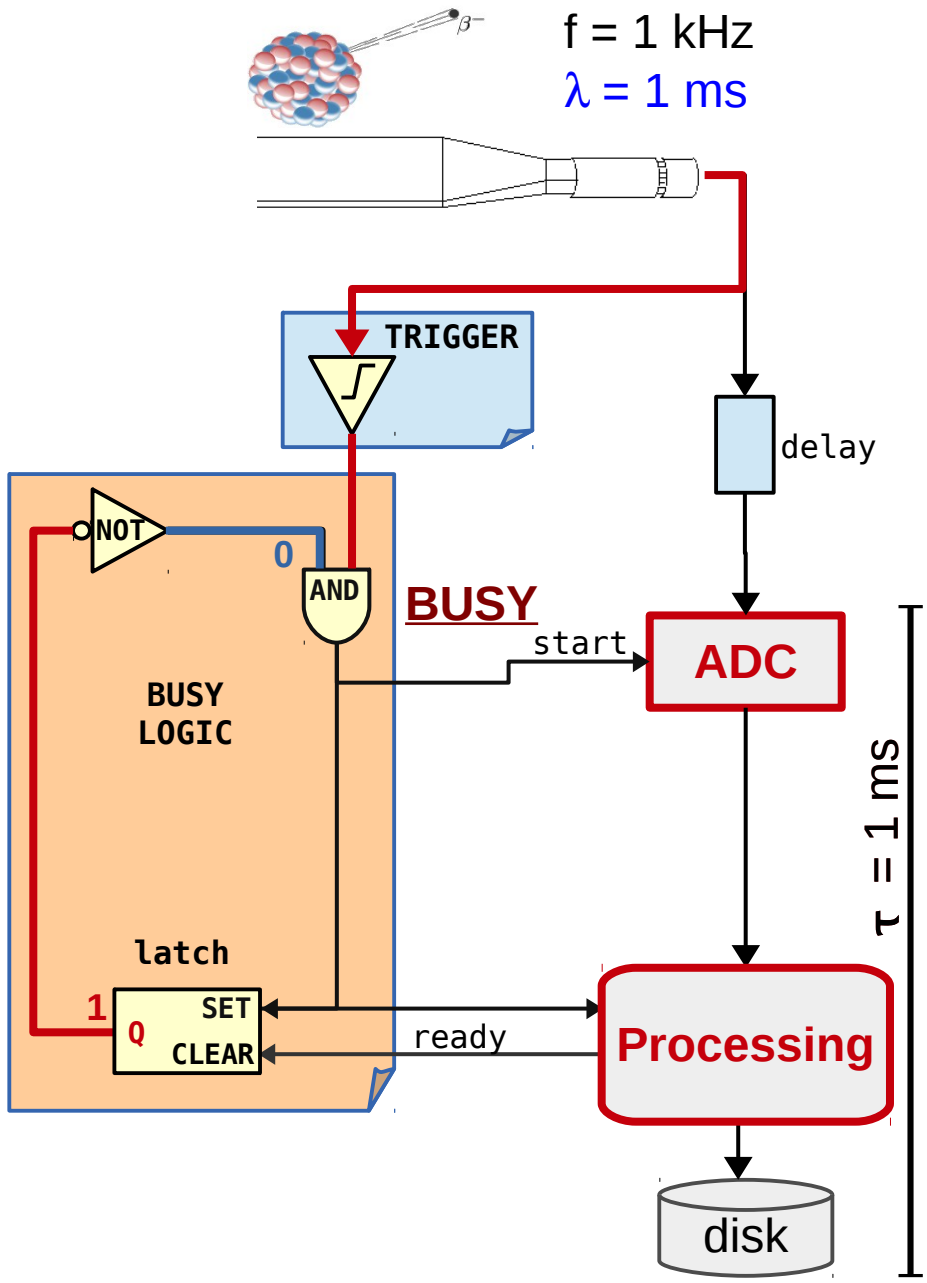
Basic DAQ: “real” trigger

- **Busy logic** avoids triggers while the system is busy in processing
  - E.g.: AND gate and a latch
- Latch (**flip-flop**):
  - a bistable circuit that changes state (Q) by signals applied to the control inputs (SET, CLEAR)
  - Flip-flop is flipped
    - Now one AND input gate is steadily down



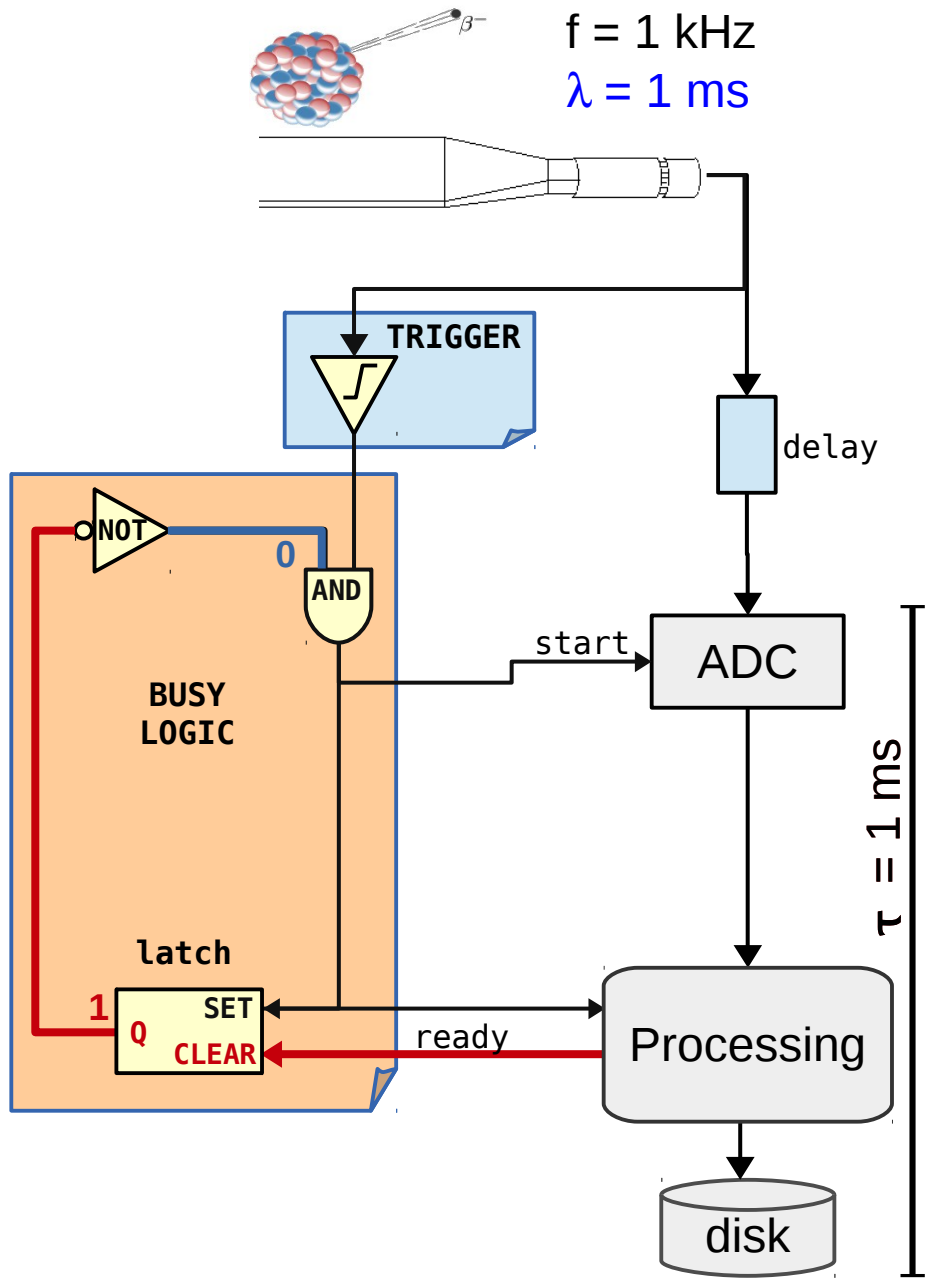
Basic DAQ: “real” trigger

- **Busy logic** avoids triggers while the system is busy in processing
  - E.g.: AND gate and a latch
- Latch (**flip-flop**):
  - a bistable circuit that changes state (Q) by signals applied to the control inputs (SET, CLEAR)
  - Any new trigger is inhibited by the AND gate
    - busy



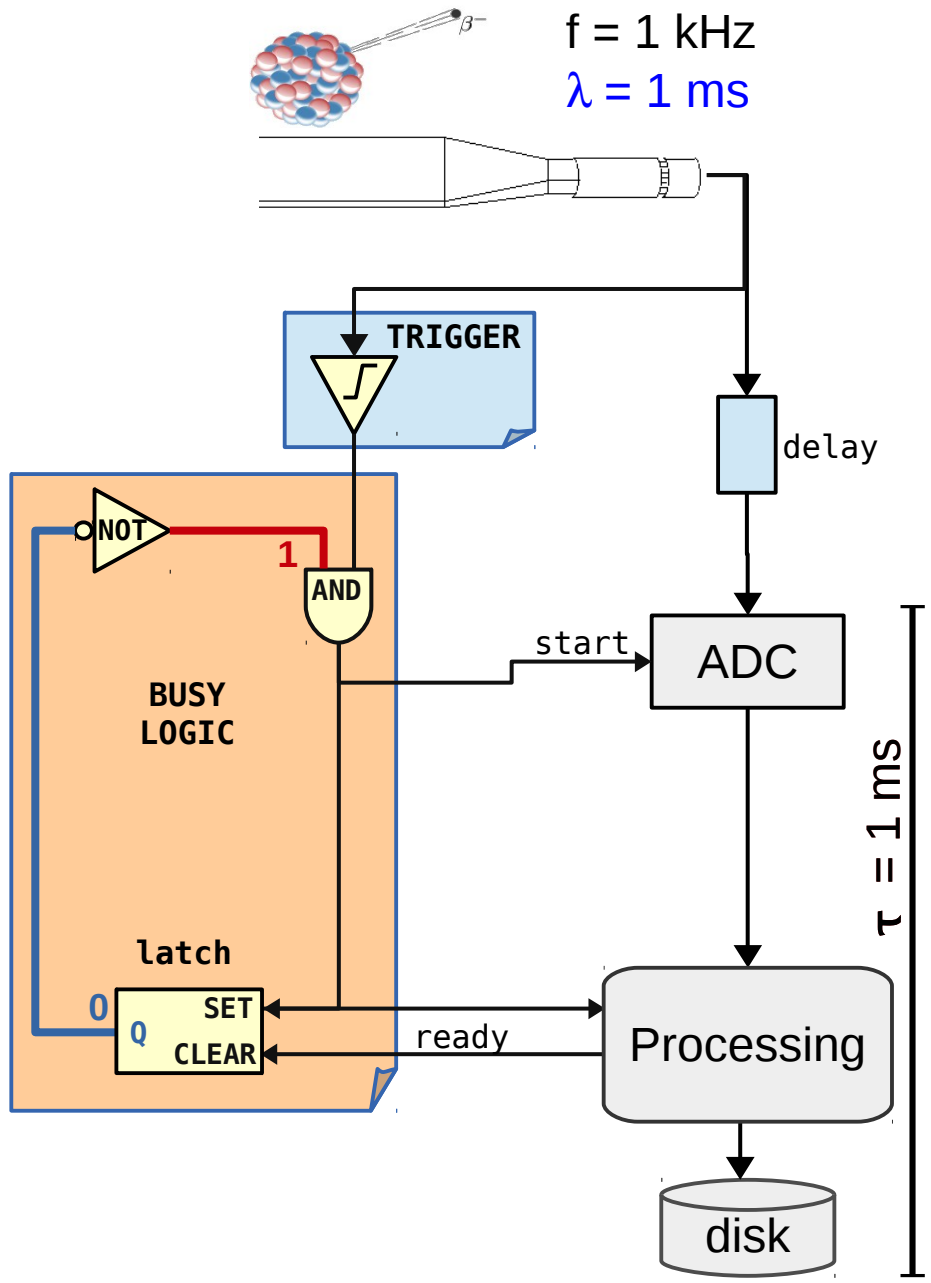
Basic DAQ: “real” trigger

- **Busy logic** avoids triggers while the system is busy in processing
  - E.g.: AND gate and a latch
- Latch (**flip-flop**):
  - a bistable circuit that changes state (Q) by signals applied to the control inputs (SET, CLEAR)
  - At the end of processing flip-flop is flipped again
    - The system is ready to accept a new trigger

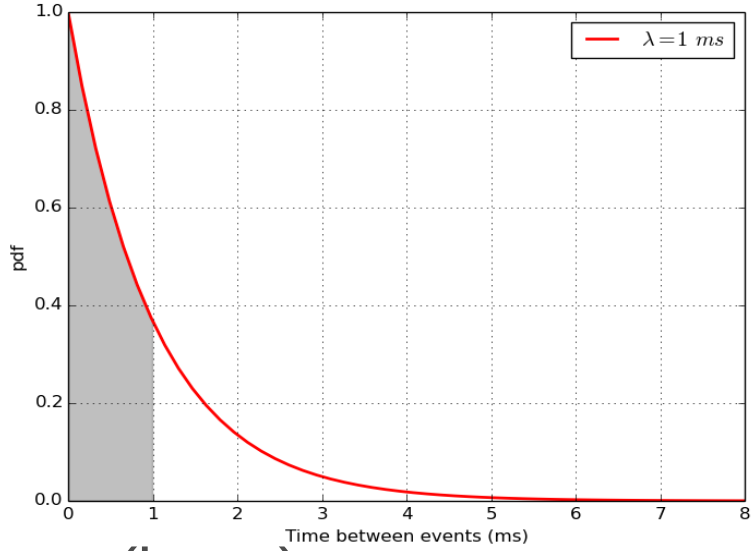


Basic DAQ: “real” trigger

- **Busy logic** avoids triggers while the system is busy in processing
  - E.g.: AND gate and a latch
- Latch (**flip-flop**):
  - a bistable circuit that changes state (Q) by signals applied to the control inputs (SET, CLEAR)
  - At the end of processing flip-flop is flipped again
    - The system is ready to accept a new trigger



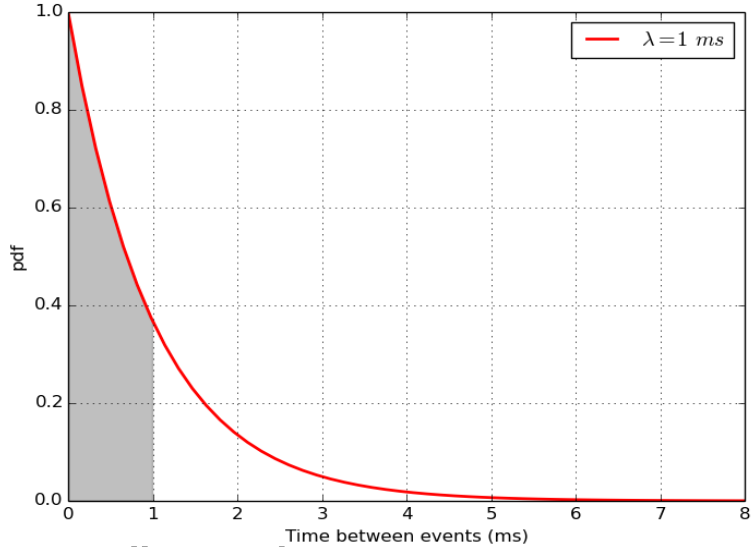
Deadtime and efficiency



- Which (average) DAQ rate can we achieve now?
  - Reminder: w/ a clock trigger and τ = 1 ms the limit is 1 kHz
- Definitions
  - **f:** average rate of physics phenomenon (input)
  - **ν:** average rate of DAQ (output)
  - **τ: deadtime**, the time the system requires to process an event, without being able to handle other triggers
  - probabilities: P[busy] = $\nu\,\tau$; P[free] = 1 - $\nu\,\tau$
- Therefore:

$$\nu = f\,P[free] \;\Rightarrow\; \nu = f(1-\nu\,\tau) \;\Rightarrow\; \nu = \frac{f}{1+f\,\tau}$$

Deadtime and efficiency

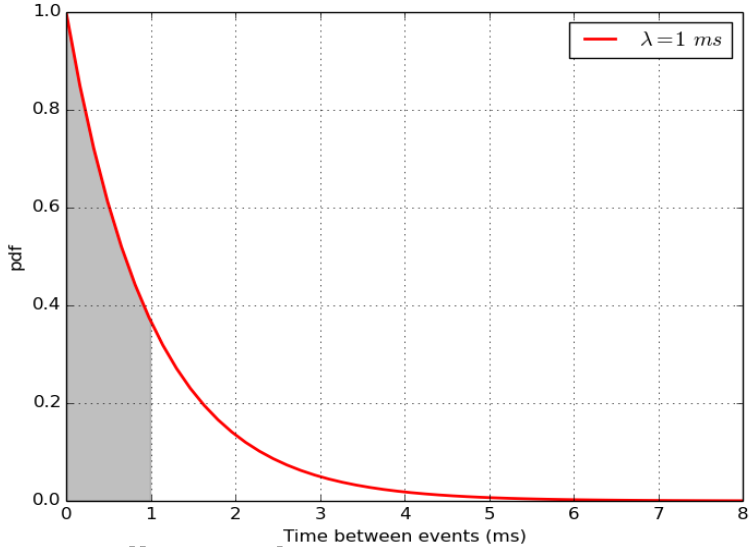


- Which (average) DAQ rate can we achieve now?
  - Reminder: w/ a clock trigger and τ = 1 ms the limit is 1 kHz
- Definitions
  - **f:** average rate of physics phenomenon (input)
  - **ν:** average rate of DAQ (output)
  - **τ: deadtime**, the time the system requires to process an event, without being able to handle other triggers
  - probabilities: P[busy] = $\nu\,\tau$; P[free] = 1 - $\nu\,\tau$
- Therefore:

$$\nu = f\,P[free] \;\Rightarrow\; \nu = f(1-\nu\,\tau) \;\Rightarrow\; \nu = \frac{f}{1+f\,\tau}$$

Deadtime and efficiency

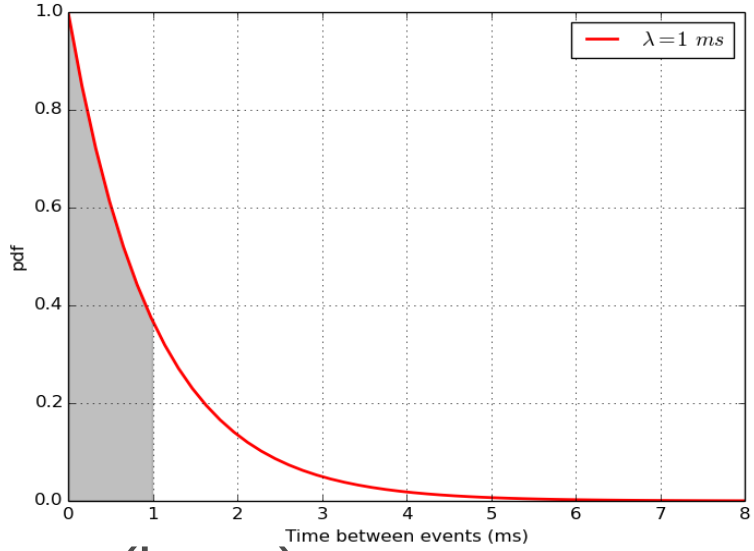
- Which (average) DAQ rate can we achieve now?
  - Reminder: w/ a clock trigger and τ = 1 ms the limit is 1 kHz



- Definitions
  - **f:** average rate of physics phenomenon (input)
  - **ν:** average rate of DAQ (output)
  - **τ: deadtime**, the time the system requires to process an event, without being able to handle other triggers
  - probabilities: P[busy] = $\nu\,\tau$; P[free] = 1 - $\nu\,\tau$
- Therefore:

$$\nu = f\,P[free] \;\Rightarrow\; \nu = f(1-\nu\,\tau) \;\Rightarrow\; \nu = \frac{f}{1+f\,\tau}$$

Deadtime and efficiency



- Which (average) DAQ rate can we achieve now?
  - Reminder: w/ a clock trigger and τ = 1 ms the limit is 1 kHz
- Definitions
  - **f:** average rate of physics phenomenon (input)
  - **ν:** average rate of DAQ (output)
  - **τ: deadtime**, the time the system requires to process an event, without being able to handle other triggers
  - probabilities: P[busy] = $\nu\,\tau$; P[free] = 1 - $\nu\,\tau$
- Therefore:

$$\nu = f\,P[free] \;\Rightarrow\; \nu = f(1-\nu\,\tau) \;\Rightarrow\; \nu = \frac{f}{1+f\,\tau}$$

Deadtime and efficiency

- Due to stochastic fluctuations
  - DAQ rate always < physics rate $\nu = \frac{f}{1+f\,\tau} < f$
  - Efficiency always < 100% $\epsilon = \frac{N_{saved}}{N_{tot}} = \frac{1}{1+f\,\tau} < 100\,\%$

- So, in our specific example

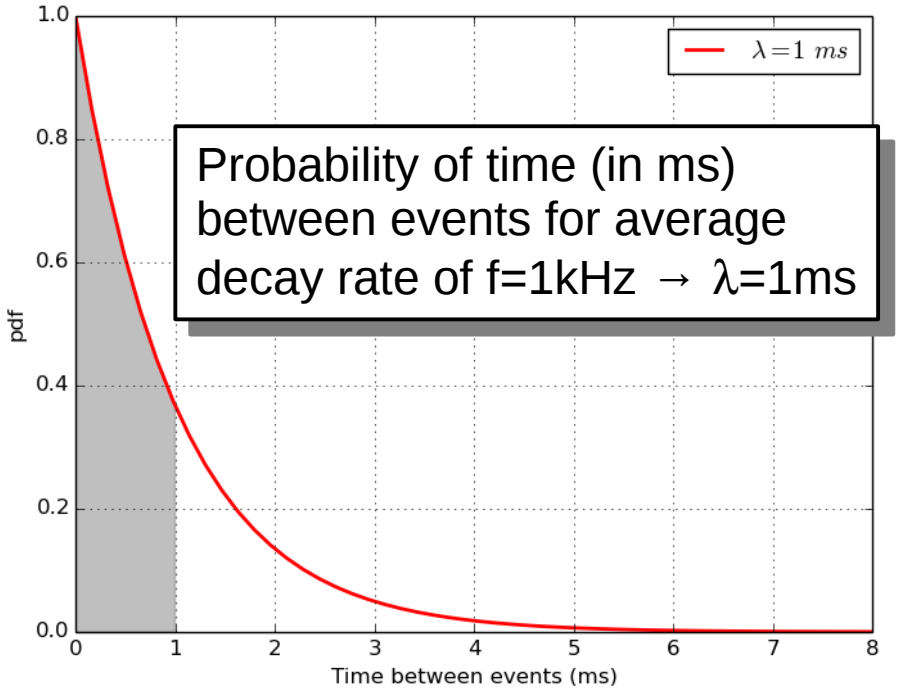
$$\left| \begin{array}{l} f = 1\,kHz \\ \tau = 1\,ms \end{array} \right. \rightarrow \left| \begin{array}{l} \nu = 500\,Hz \\ \epsilon = 50\,\% \end{array} \right.$$

Deadtime and efficiency

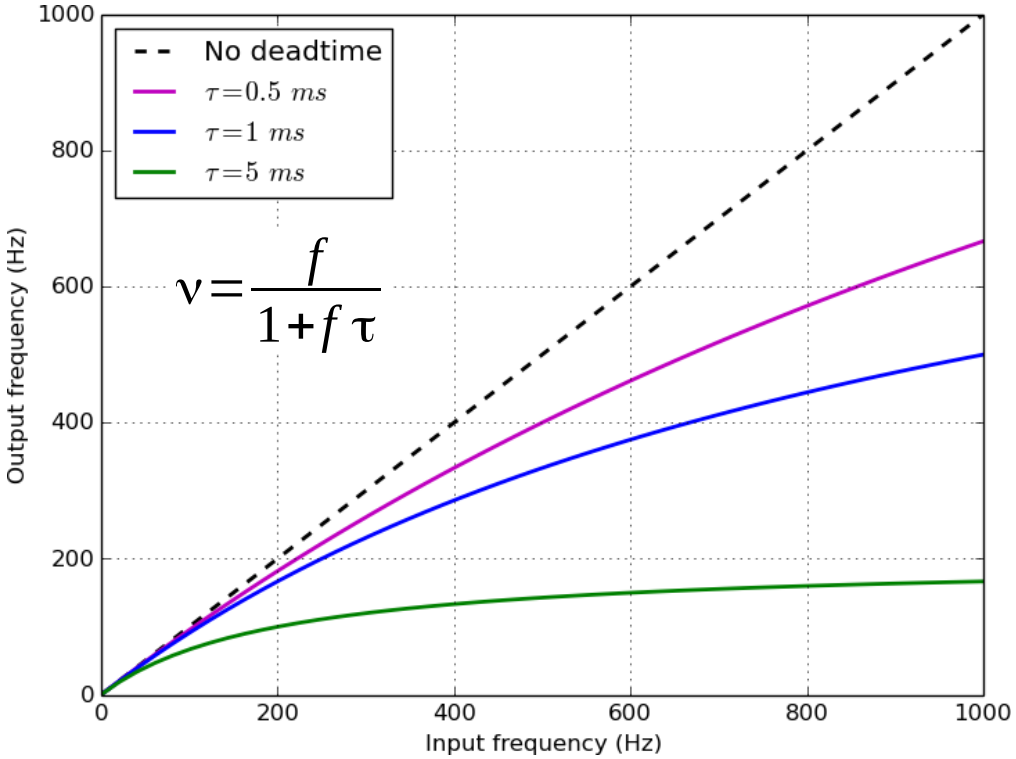
- Due to stochastic fluctuations
  - DAQ rate always < physics rate $\nu = \frac{f}{1+f\,\tau} < f$
  - Efficiency always < 100% $\epsilon = \frac{N_{saved}}{N_{tot}} = \frac{1}{1+f\,\tau} < 100\,\%$

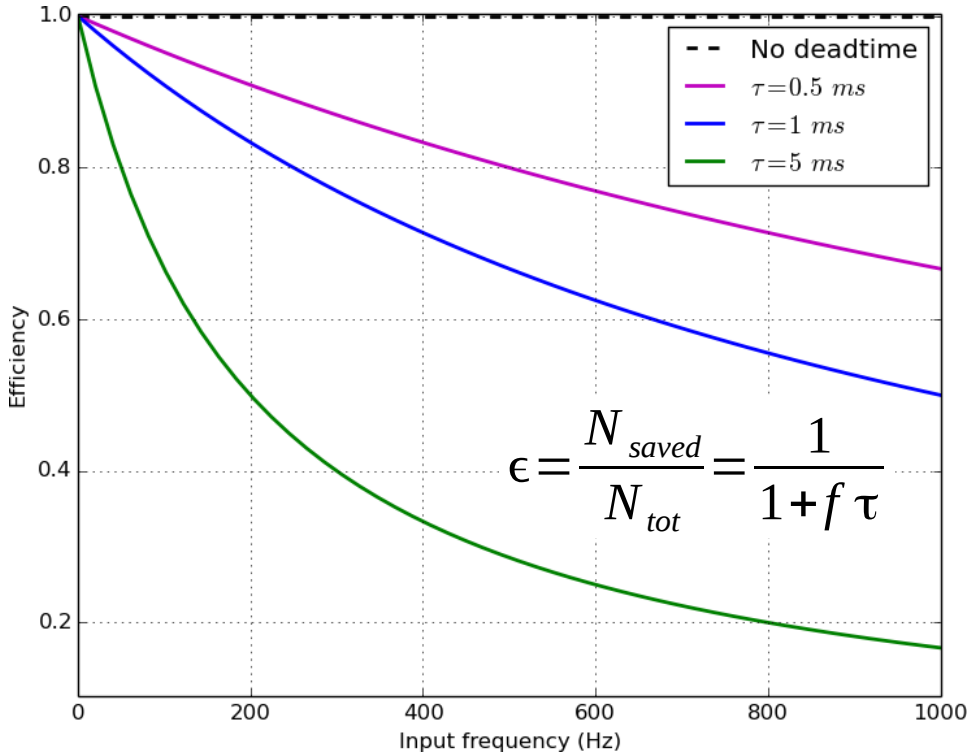
- So, in our specific example

$$\left| \begin{array}{l} f = 1\,kHz \\ \tau = 1\,ms \end{array} \right. \rightarrow \left| \begin{array}{l} \nu = 500\,Hz \\ \epsilon = 50\,\% \end{array} \right.$$



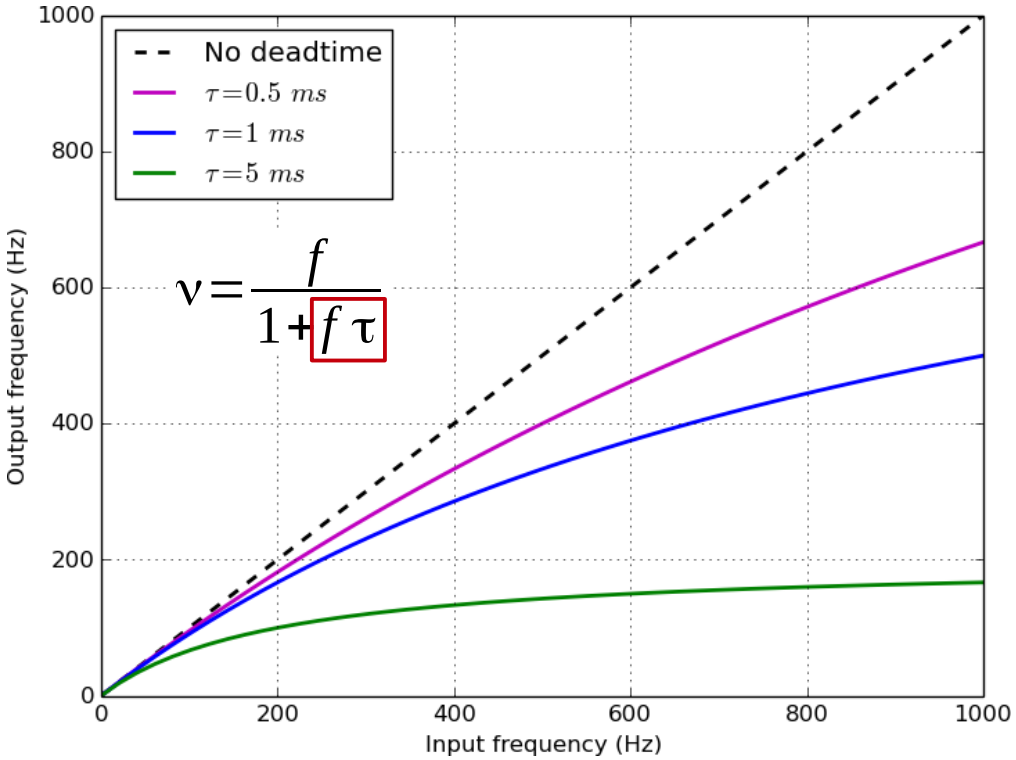
Deadtime and efficiency

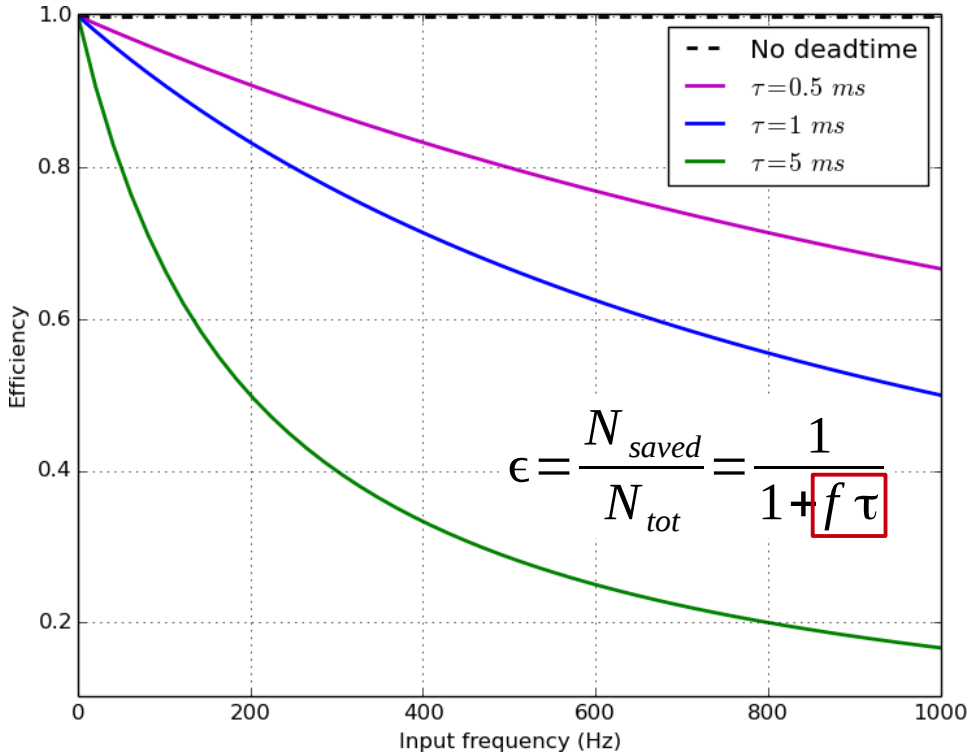




- In order to obtain ε~100% (i.e.: ν~f) $\rightarrow f\tau << 1 \rightarrow \tau << \lambda$
  - E.g.: ε~99% for f = 1 kHz $\rightarrow$ τ < 0.01 ms $\rightarrow 1/\tau$ > 100 kHz
  - To cope with the input signal fluctuations, we have to **over-design** our DAQ system **by a factor 100!**
- How can we mitigate this effect?

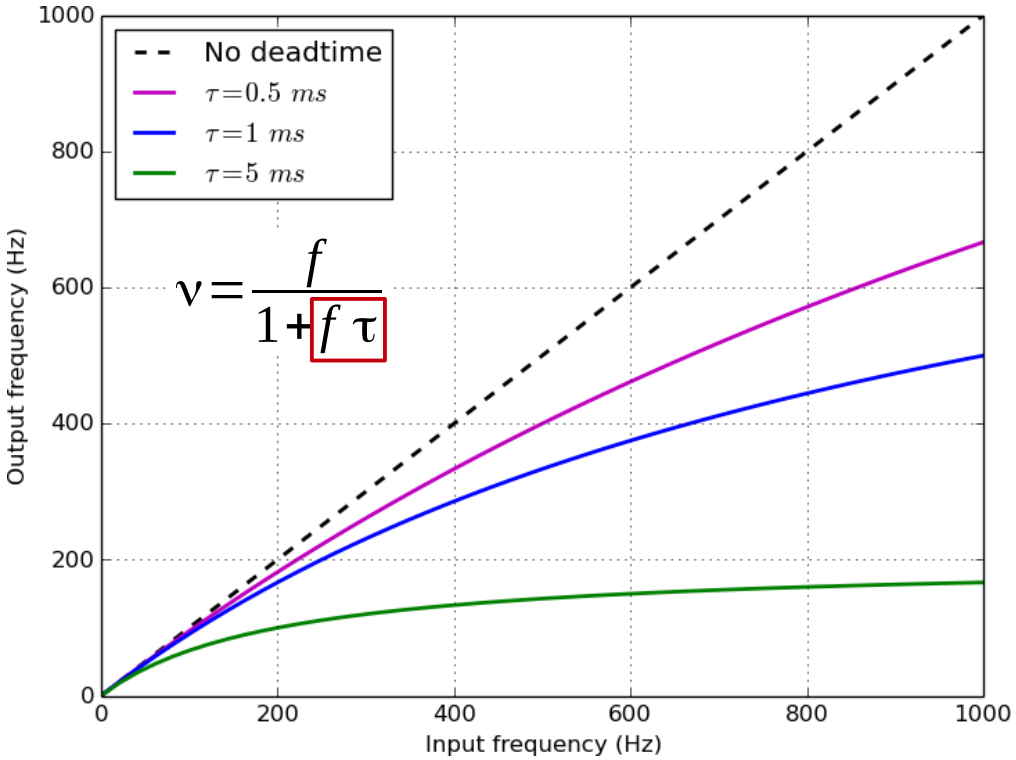
Deadtime and efficiency





- In order to obtain ε~100% (i.e.: ν~f) $\rightarrow f\tau << 1 \rightarrow \tau << \lambda$
  - E.g.: ε~99% for f = 1 kHz $\rightarrow$ τ < 0.01 ms $\rightarrow 1/\tau$ > 100 kHz
  - To cope with the input signal fluctuations, we have to **over-design** our DAQ system **by a factor 100!**
- How can we mitigate this effect?

Deadtime and efficiency

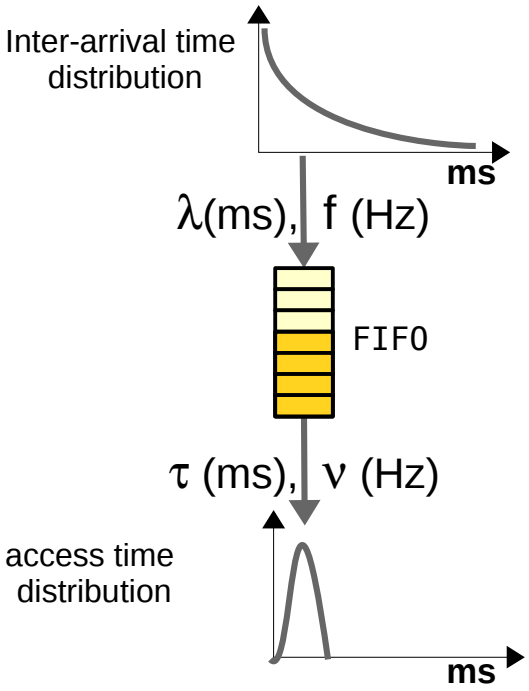


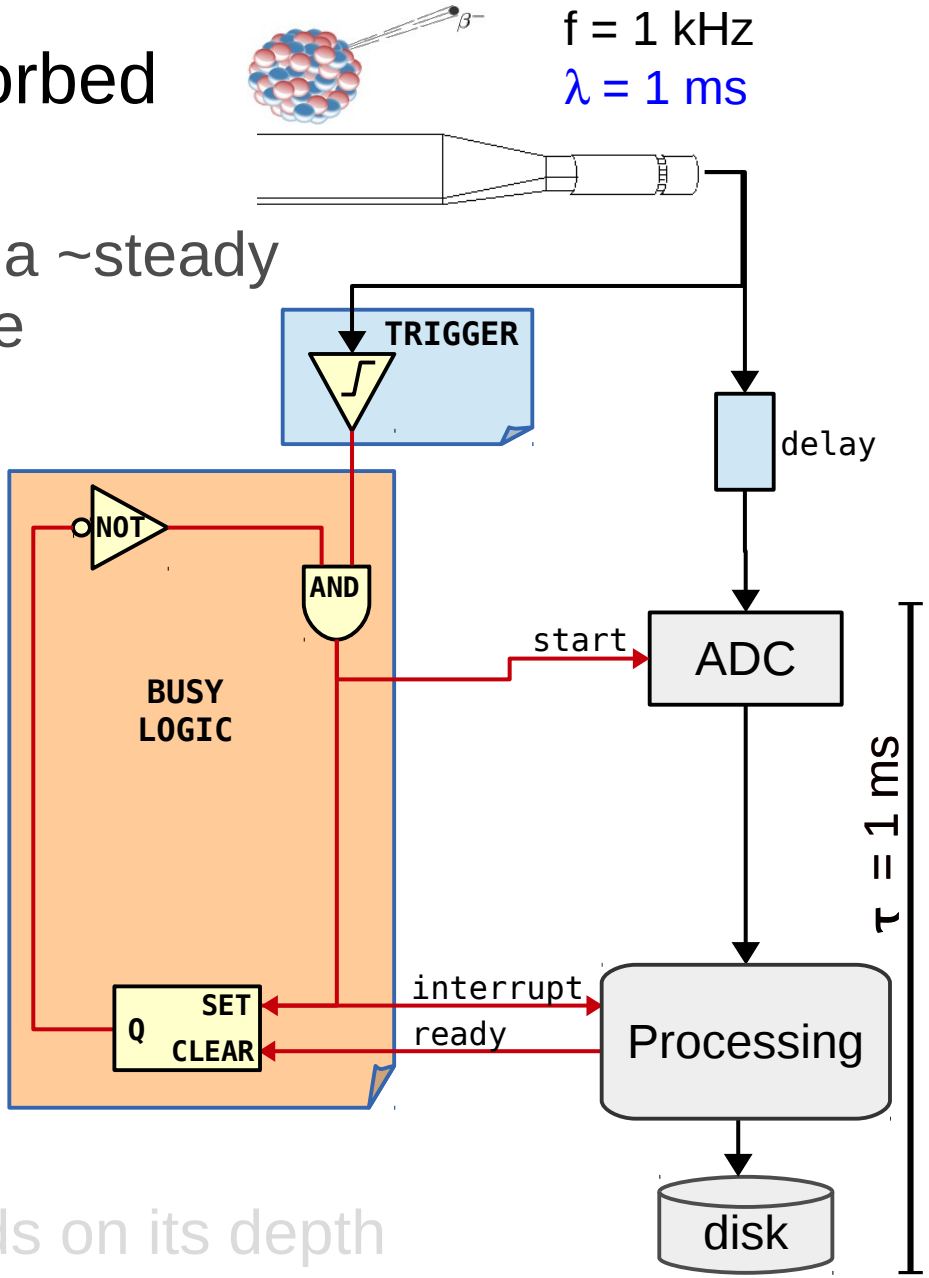
- In order to obtain ε~100% (i.e.: ν~f) $\rightarrow f\tau << 1 \rightarrow \tau << \lambda$
  - E.g.: ε~99% for f = 1 kHz $\rightarrow$ τ < 0.01 ms $\rightarrow 1/\tau$ > 100 kHz
  - To cope with the input signal fluctuations, we have to **over-design** our DAQ system **by a factor 100!**

- How can we mitigate this effect?

De-randomization

- Input fluctuations can be absorbed and smoothed by a queue
  - A First In First Out can provide a ~steady and **de-randomized** output rate

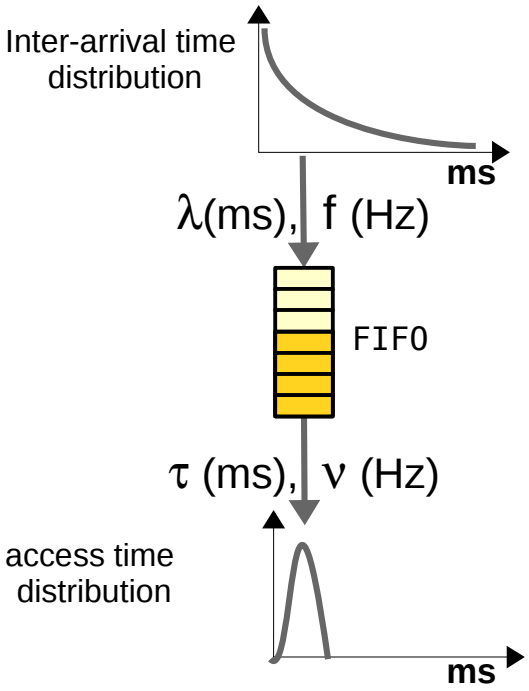


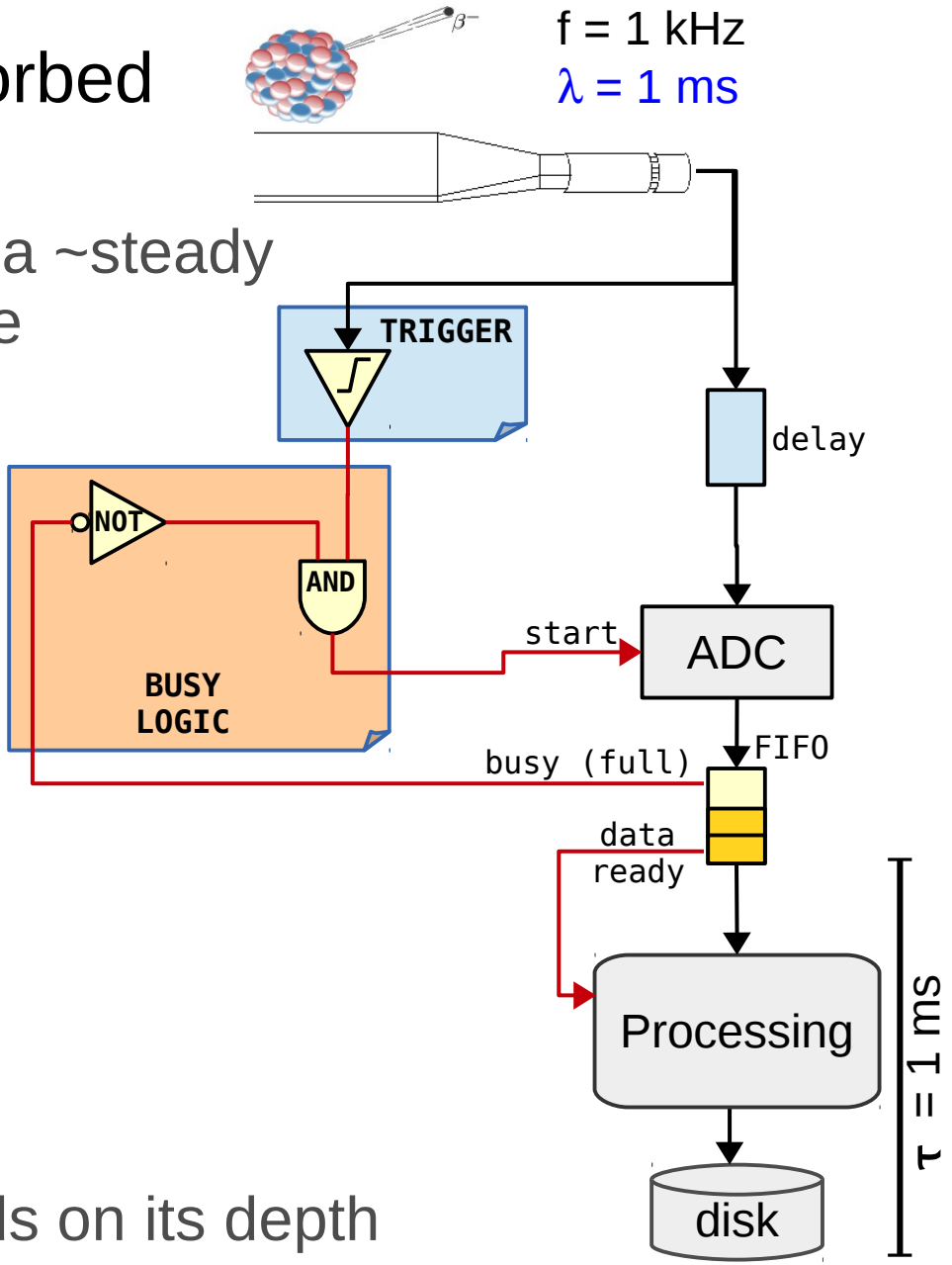


  - The effect of the queue depends on its depth

De-randomization

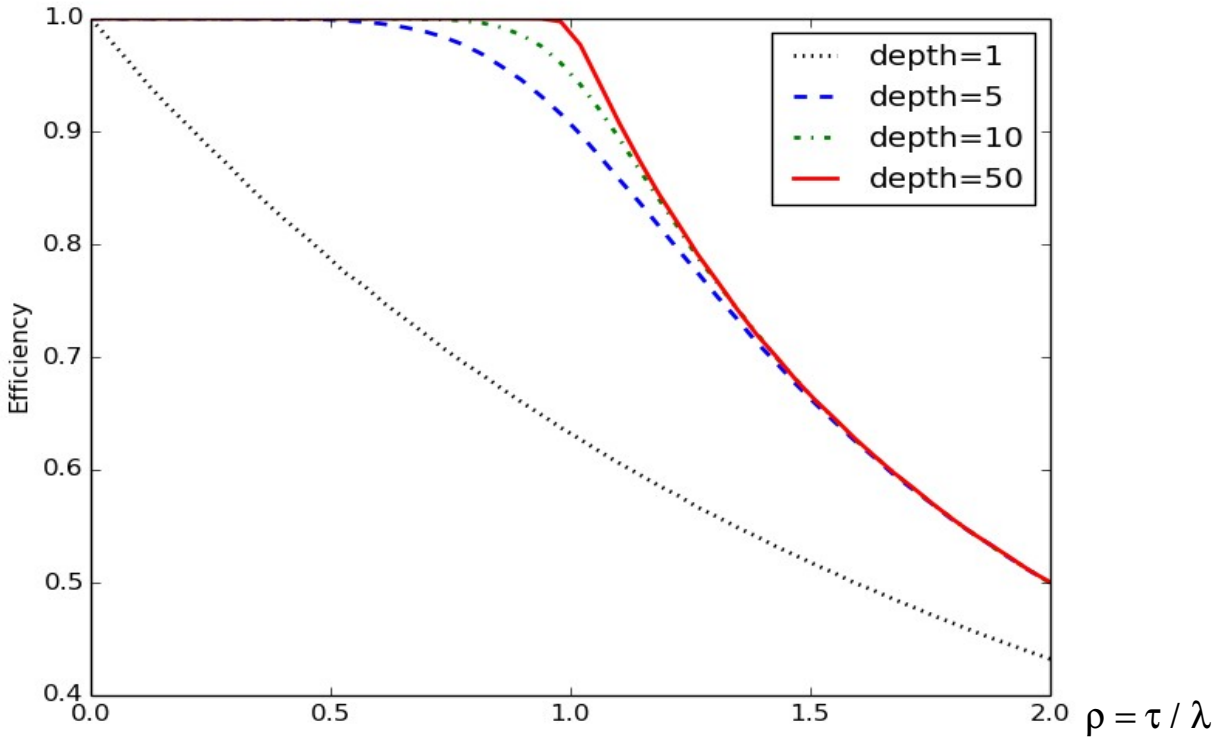
- Input fluctuations can be absorbed and smoothed by a queue
  - A First In First Out can provide a ~steady and **de-randomized** output rate

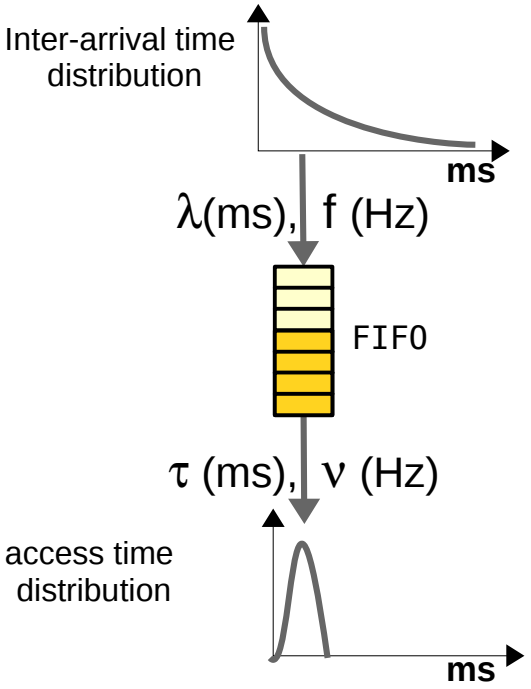




  - The effect of the queue depends on its depth

Queuing theory

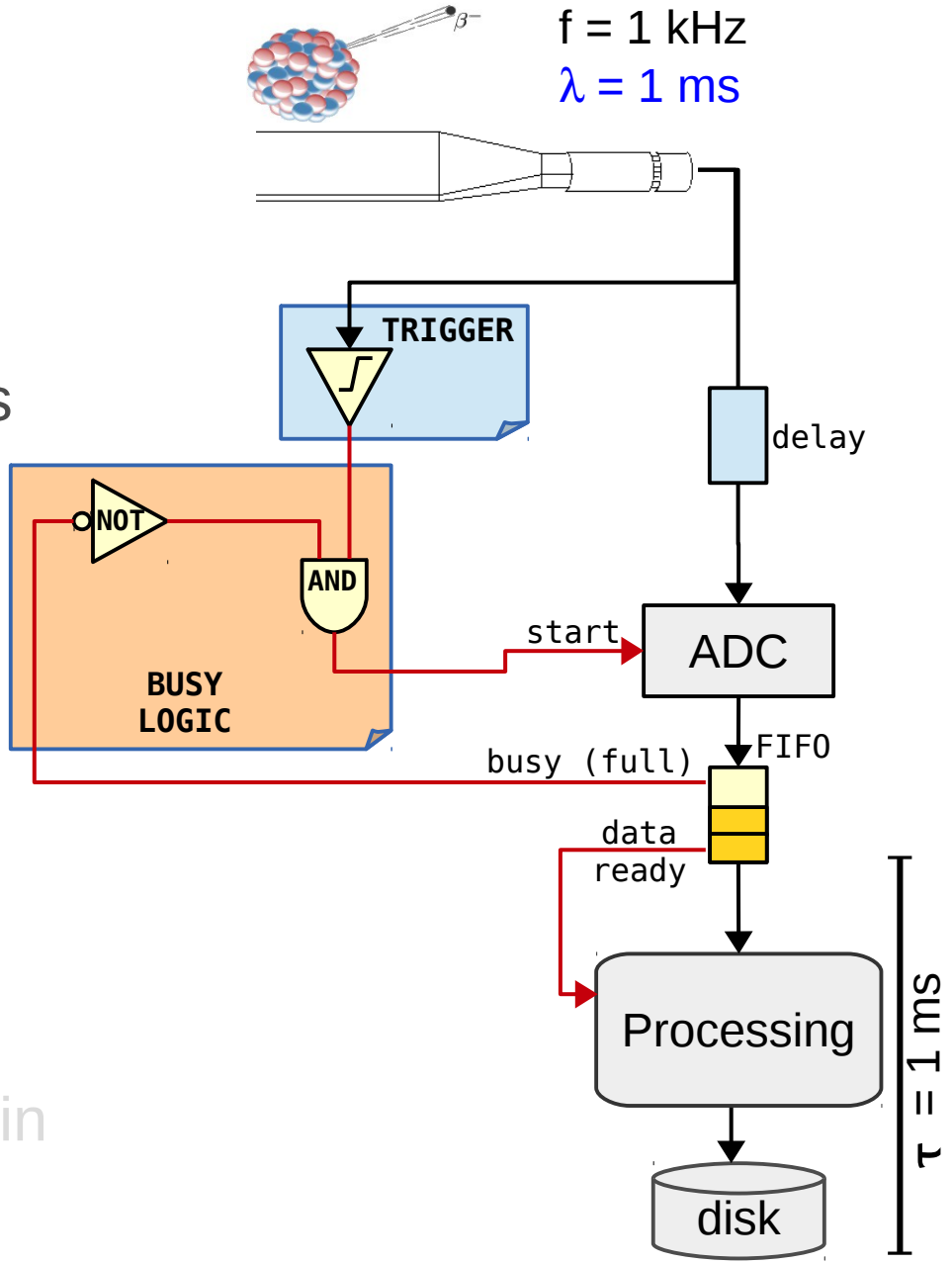




- Efficiency vs traffic intensity ($\rho = \tau / \lambda$) for different queue depths
  - **$\rho > 1$**: the system is overloaded ($\tau > \lambda$)
  - **$\rho << 1$**: the output is over-designed ($\tau << \lambda$)
  - **$\rho \sim 1$**: using a queue, high efficiency obtained even w/ moderate depth
- Analytic calculation possible for very simple systems only
  - Otherwise MonteCarlo simulation is required

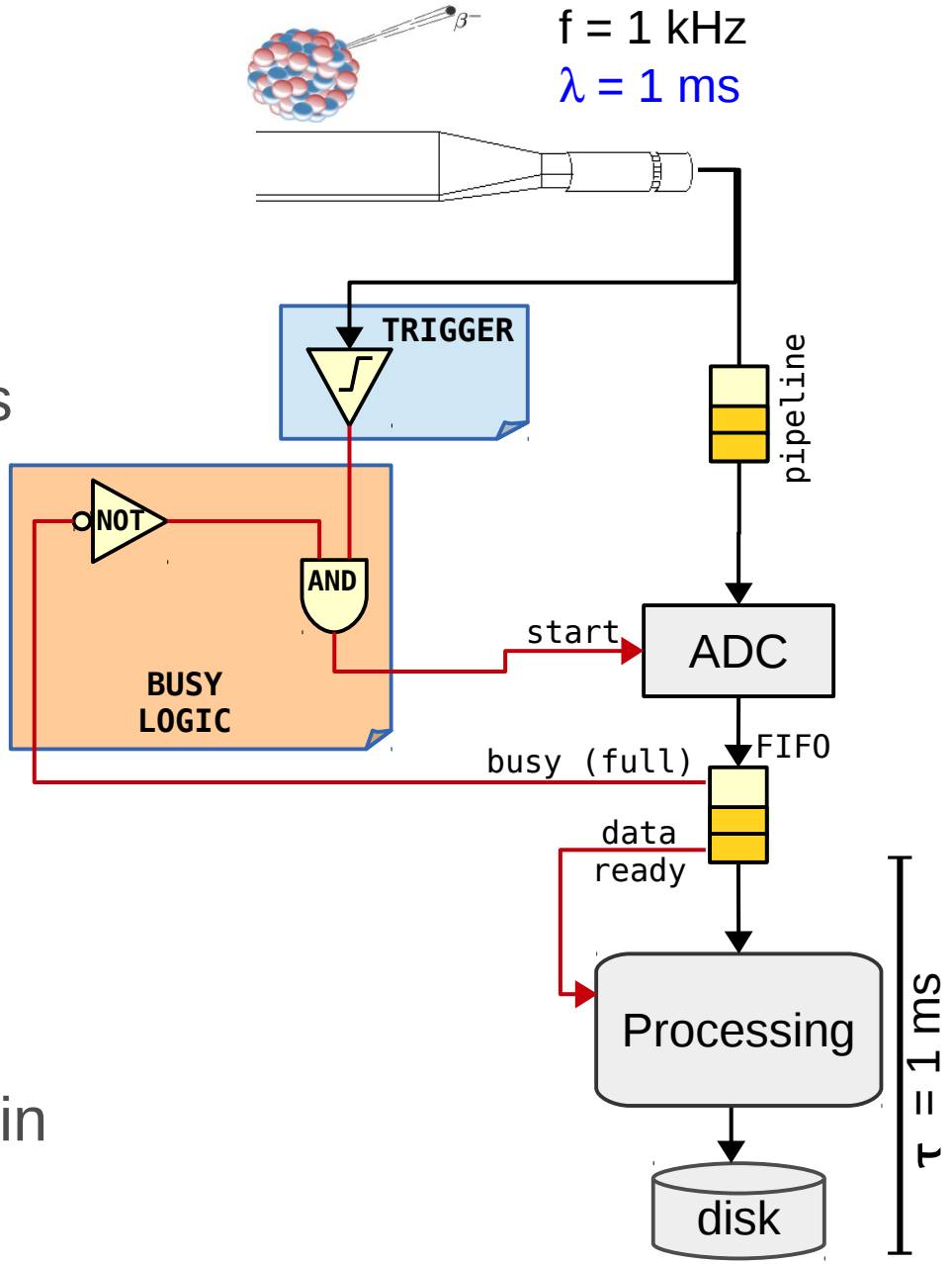
De-randomization summary

- The FIFO decouples the low latency front-end from the data processing
  - Minimize the amount of “unnecessary” fast components
- ~100% efficiency w/ minimal deadtime achievable if
  - ADC can operate at rate >> f
  - Data processing and storing operate at a rate ~ f
- Could the delay be replaced with a “FIFO”?
  - Analog pipelines, heavily used in LHC DAQs



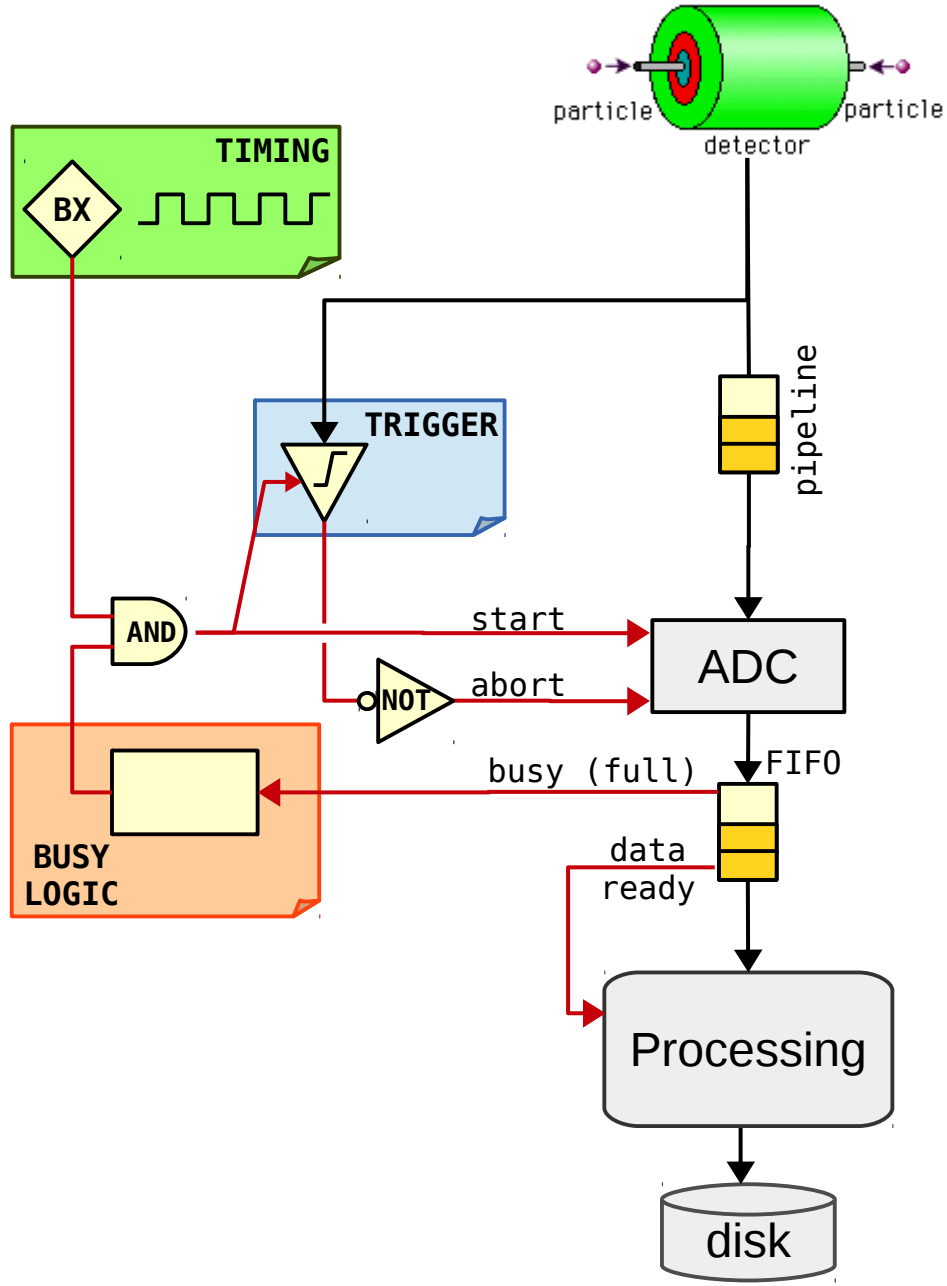
De-randomization summary

- The FIFO decouples the low latency front-end from the data processing
  - Minimize the amount of “unnecessary” fast components
- ~100% efficiency w/ minimal deadtime achievable if
  - ADC can operate at rate >> f
  - Data processing and storing operate at a rate ~ f
- Could the delay be replaced with a “FIFO”?
  - Analog pipelines, heavily used in LHC DAQs



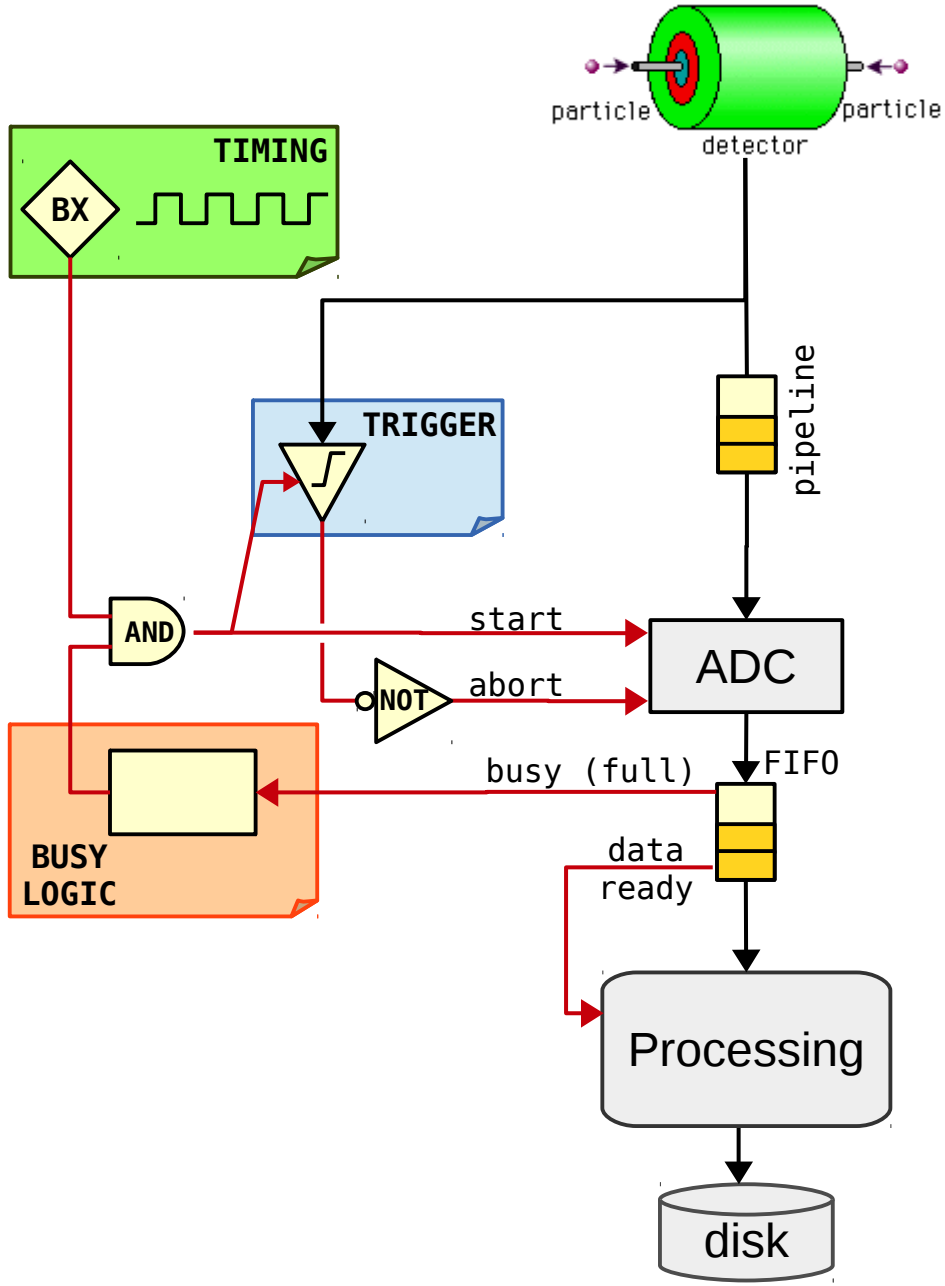
Collider setup

- Particle collisions are synchronous
  - So, do we still need de-randomization buffers?
- But the time distribution of triggers is random
  - Good events are unpredictable
- De-randomization still needed
- More complex busy logic to protect buffers and detectors
  - Eg: accept n events every m bunch crossings
  - Eg: prevent some dangerous trigger patterns



Collider setup

- Particle collisions are synchronous
  - So, do we still need de-randomization buffers?
- But the time distribution of triggers is random
  - Good events are unpredictable
- De-randomization still needed
- More complex busy logic to protect buffers and detectors
  - Eg: accept n events every m bunch crossings
  - Eg: prevent some dangerous trigger patterns

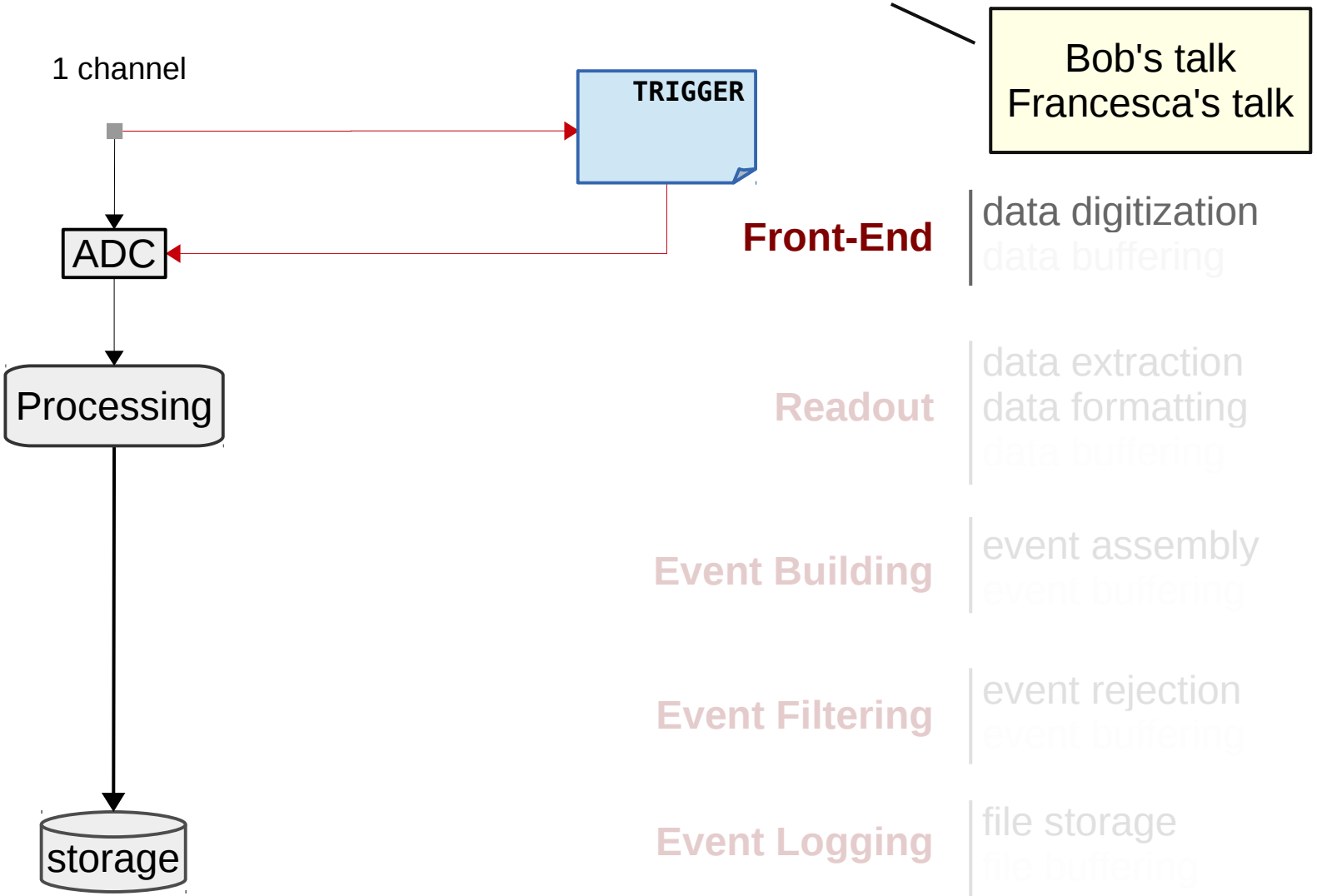


Outline

- Introduction
  - What is DAQ?
  - Overall framework
- Basic DAQ concepts
  - Digitization, Latency
  - Deadtime, Busy, Backpressure
  - De-randomization
- Scaling up
  - Readout and Event Building
  - Buses vs Network
- Do it yourself

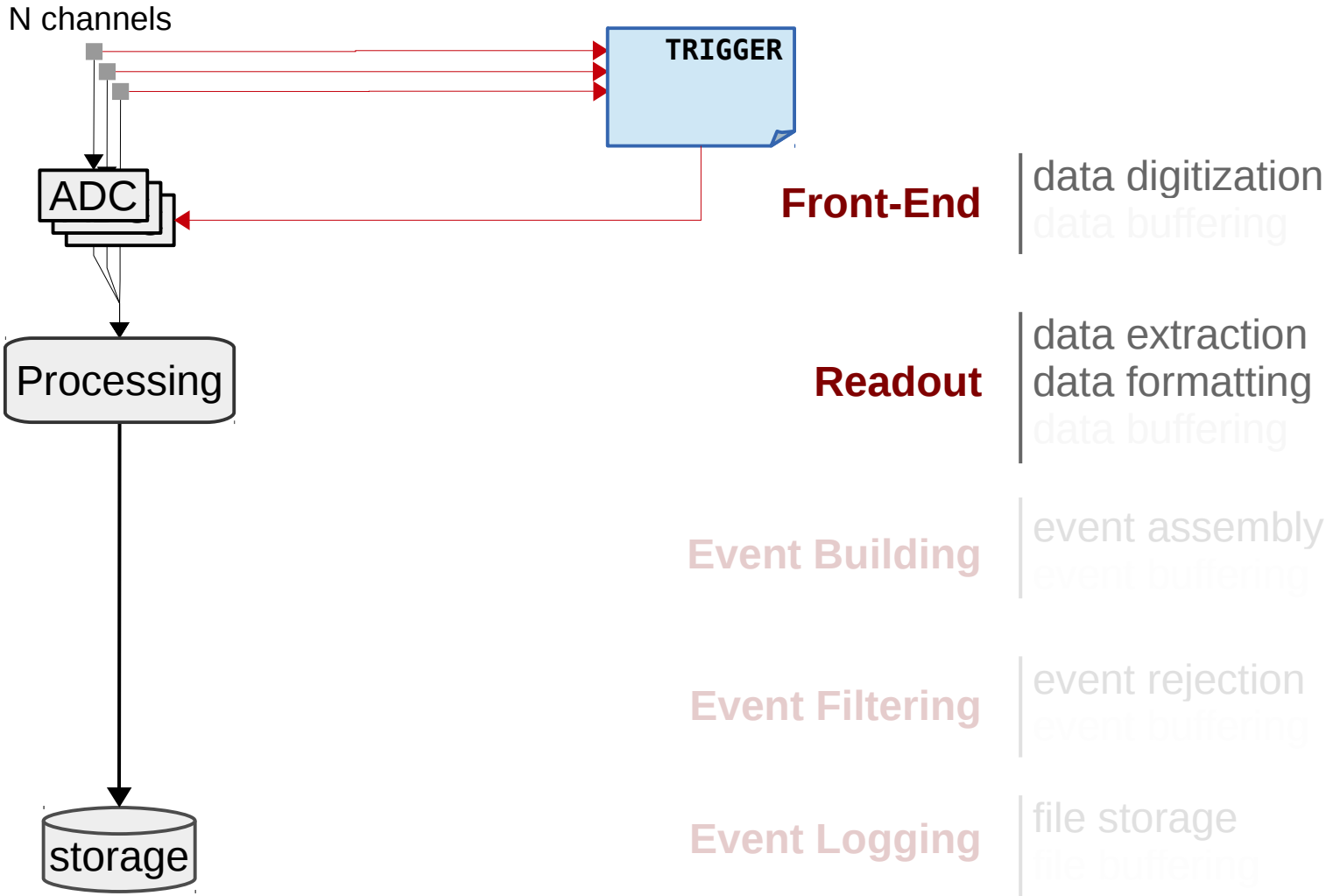
Adding more channels

- Adding more channels requires a hierarchical structure committed to the data handling and conveyance



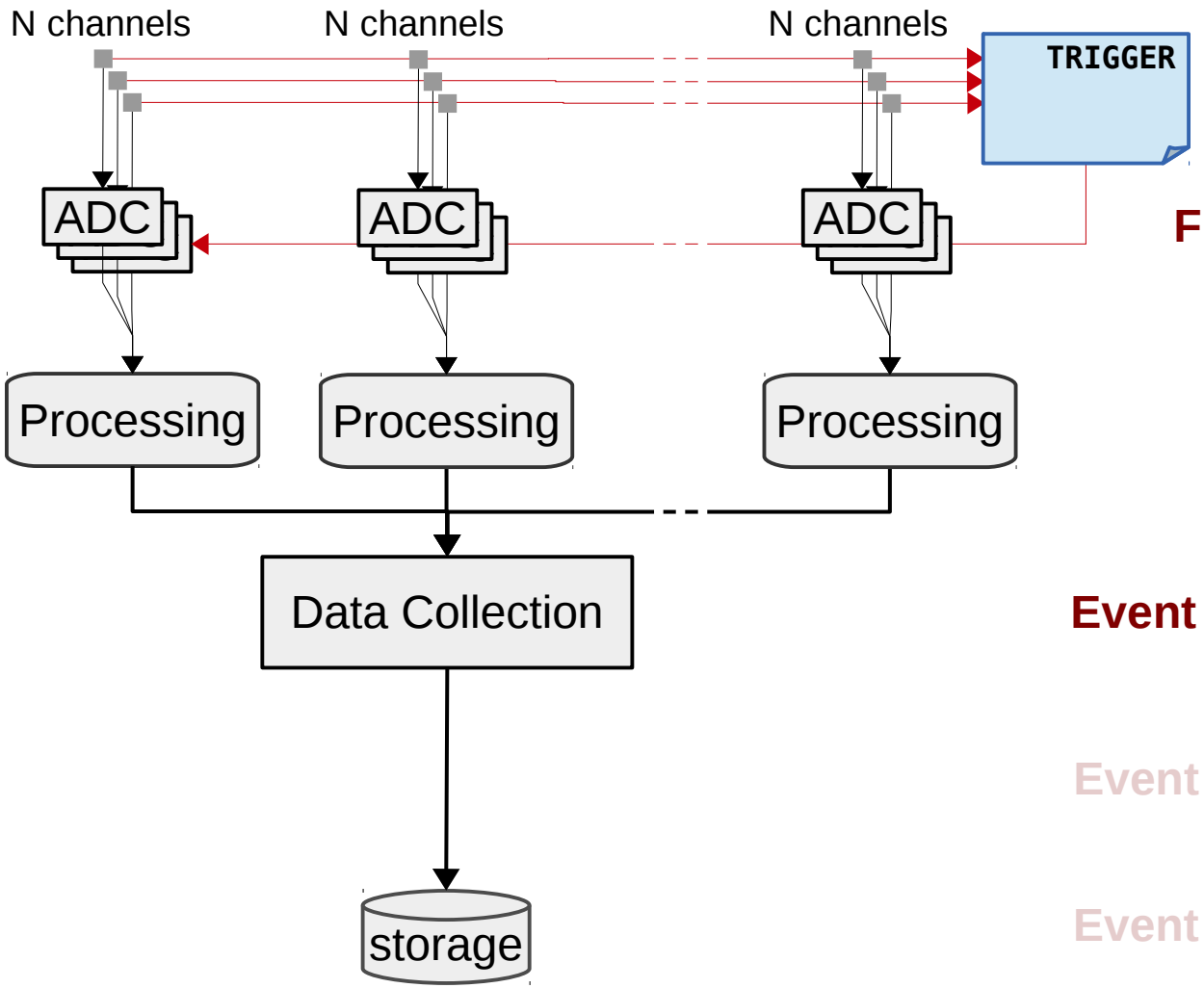
Adding more channels

- Adding more channels requires a hierarchical structure committed to the data handling and conveyance



Adding more channels

- Adding more channels requires a hierarchical structure committed to the data handling and conveyance



Front-End — data digitization, data buffering

Readout — data extraction, data formatting, data buffering

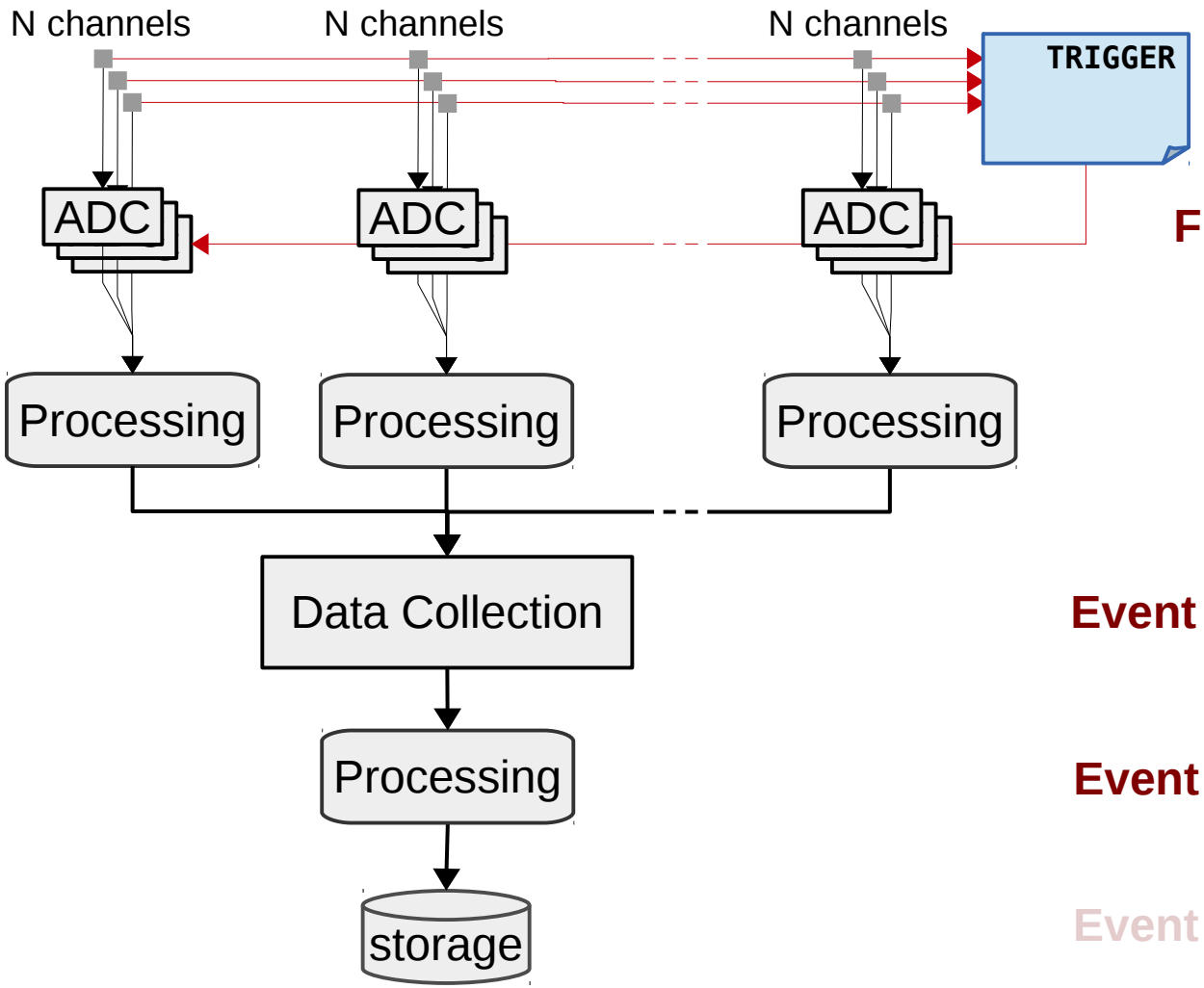
Event Building — event assembly, event buffering

Event Filtering — event rejection, event buffering

Event Logging — file storage, file buffering

Adding more channels

- Adding more channels requires a hierarchical structure committed to the data handling and conveyance



Front-End — data digitization, data buffering

Readout — data extraction, data formatting, data buffering

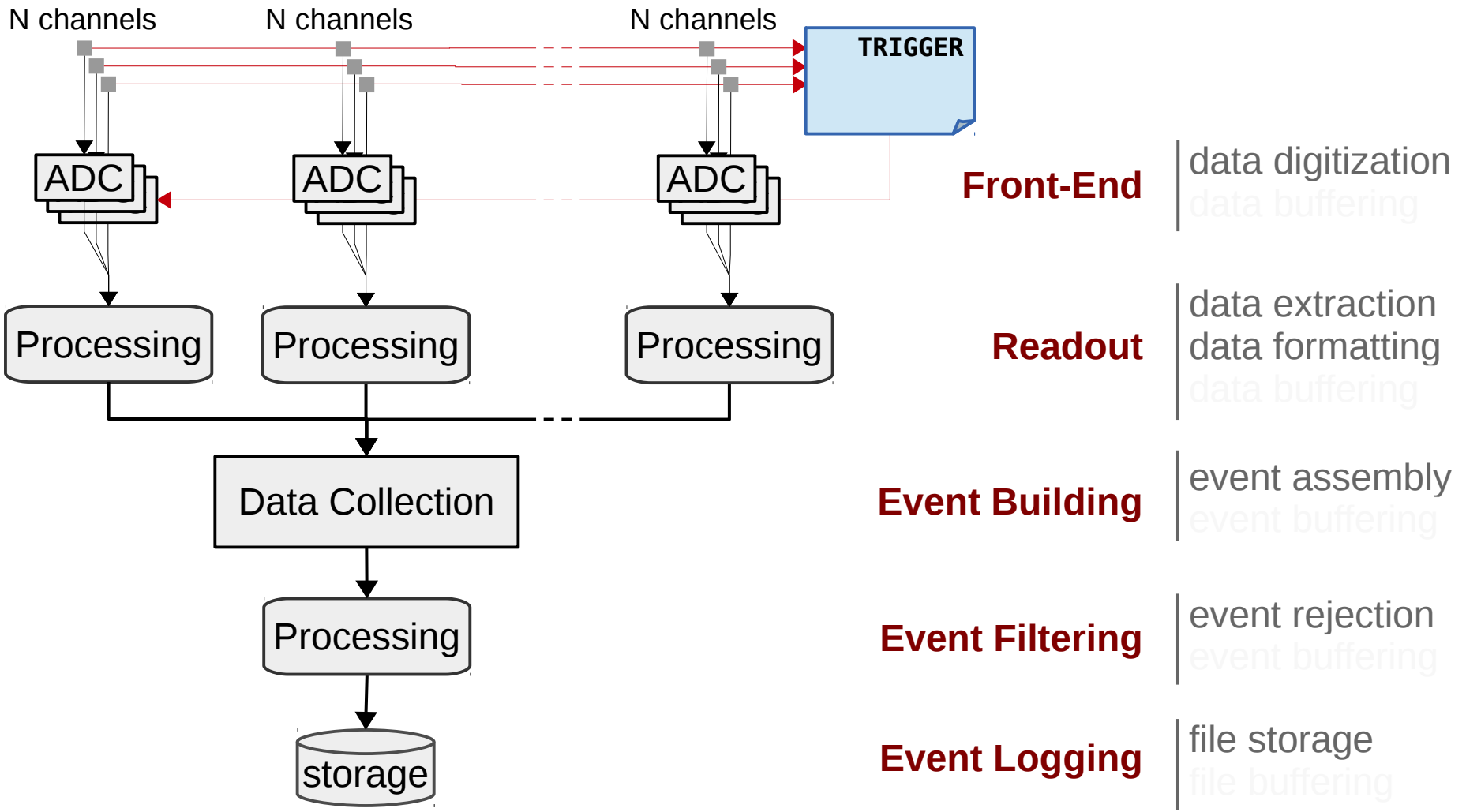
Event Building — event assembly, event buffering

Event Filtering — event rejection, event buffering

Event Logging — file storage, file buffering

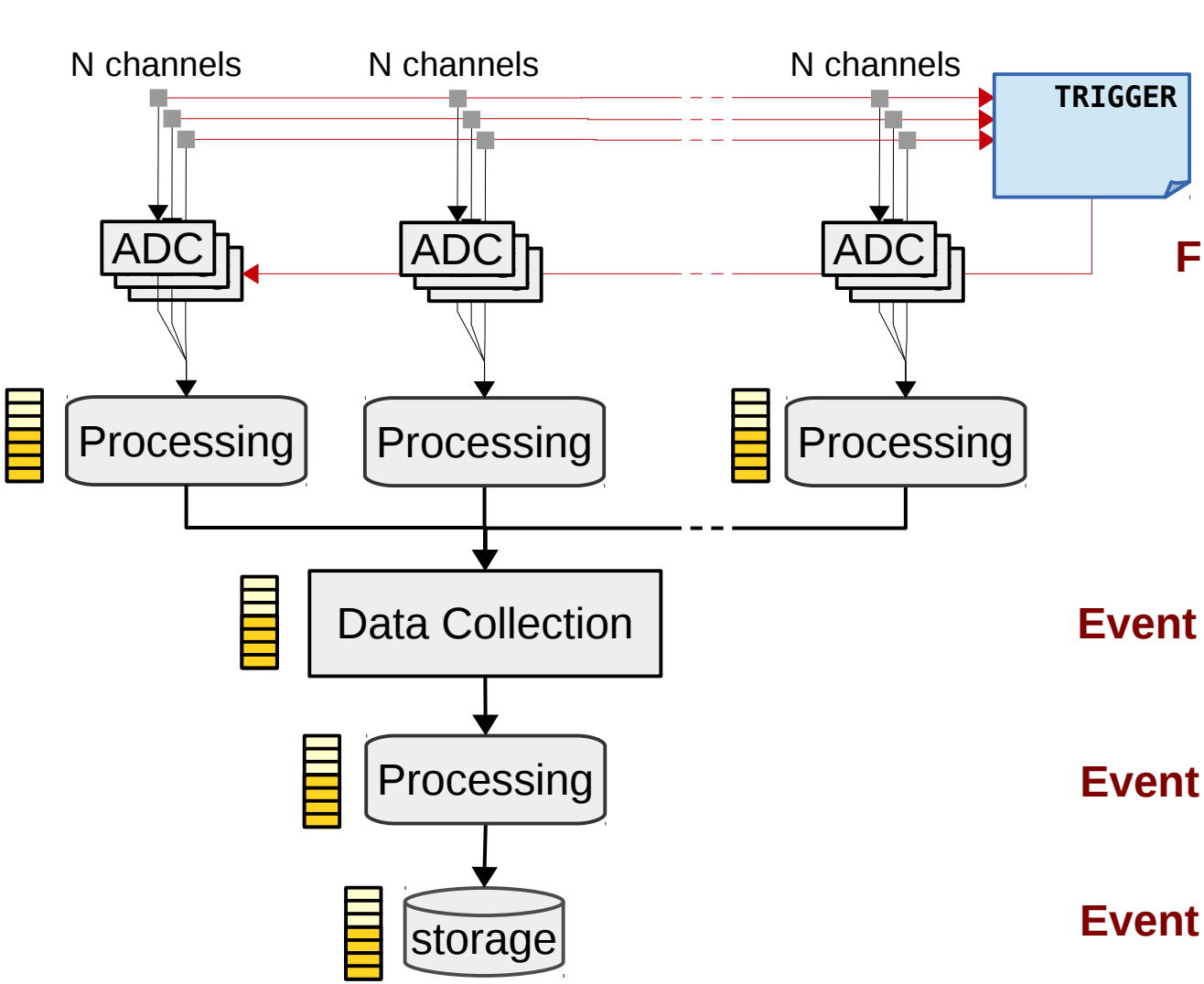
Adding more channels

- Adding more channels requires a hierarchical structure committed to the data handling and conveyance



Adding more channels

- Buffering usually needed at every level
  - DAQ can be seen as a multi level buffering system



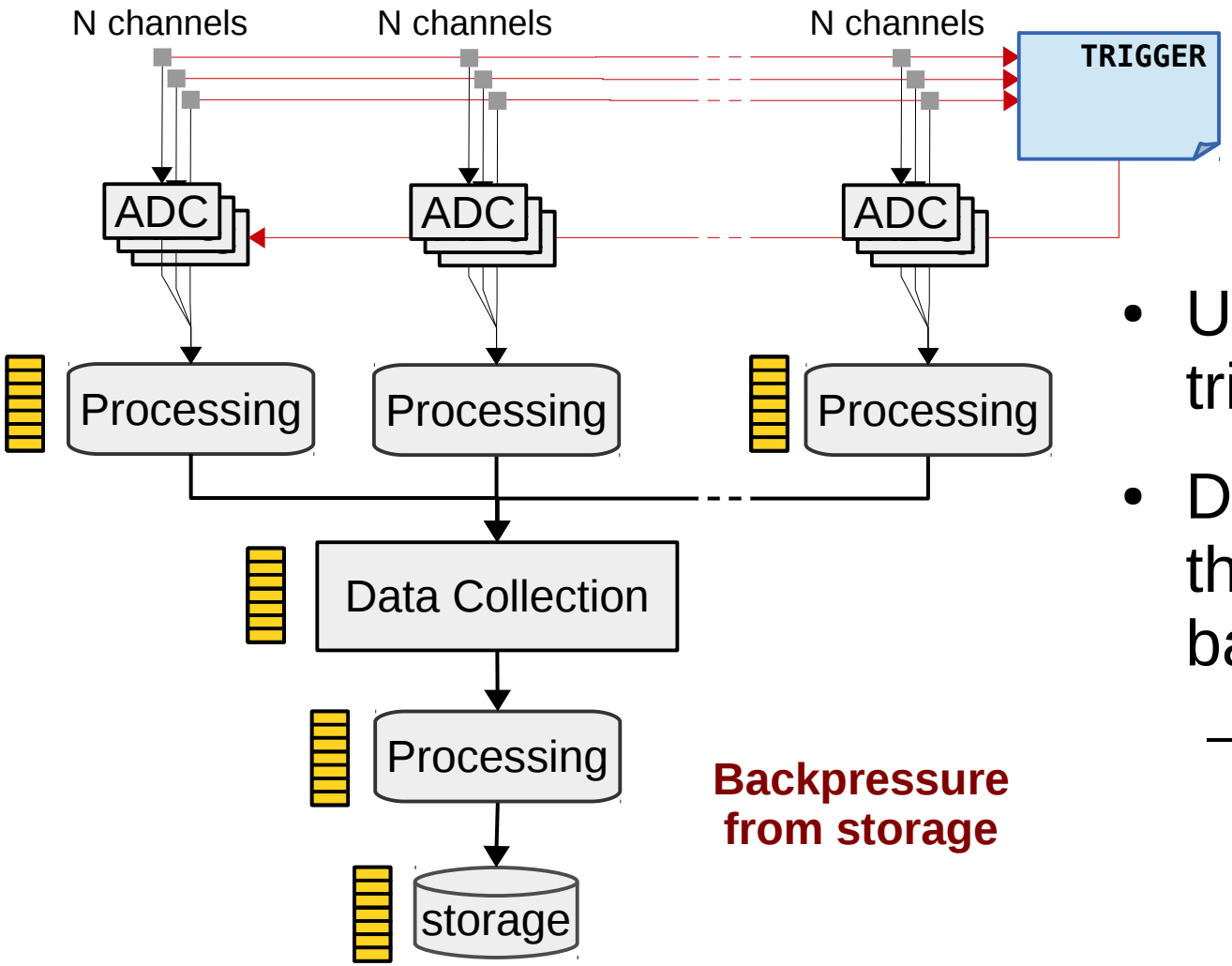
Enrico's talk

Enrico's talk

Front-End	data digitization data buffering
Readout	data extraction data formatting data buffering
Event Building	event assembly event buffering
Event Filtering	event rejection event buffering
Event Logging	file storage file buffering

Backpressure

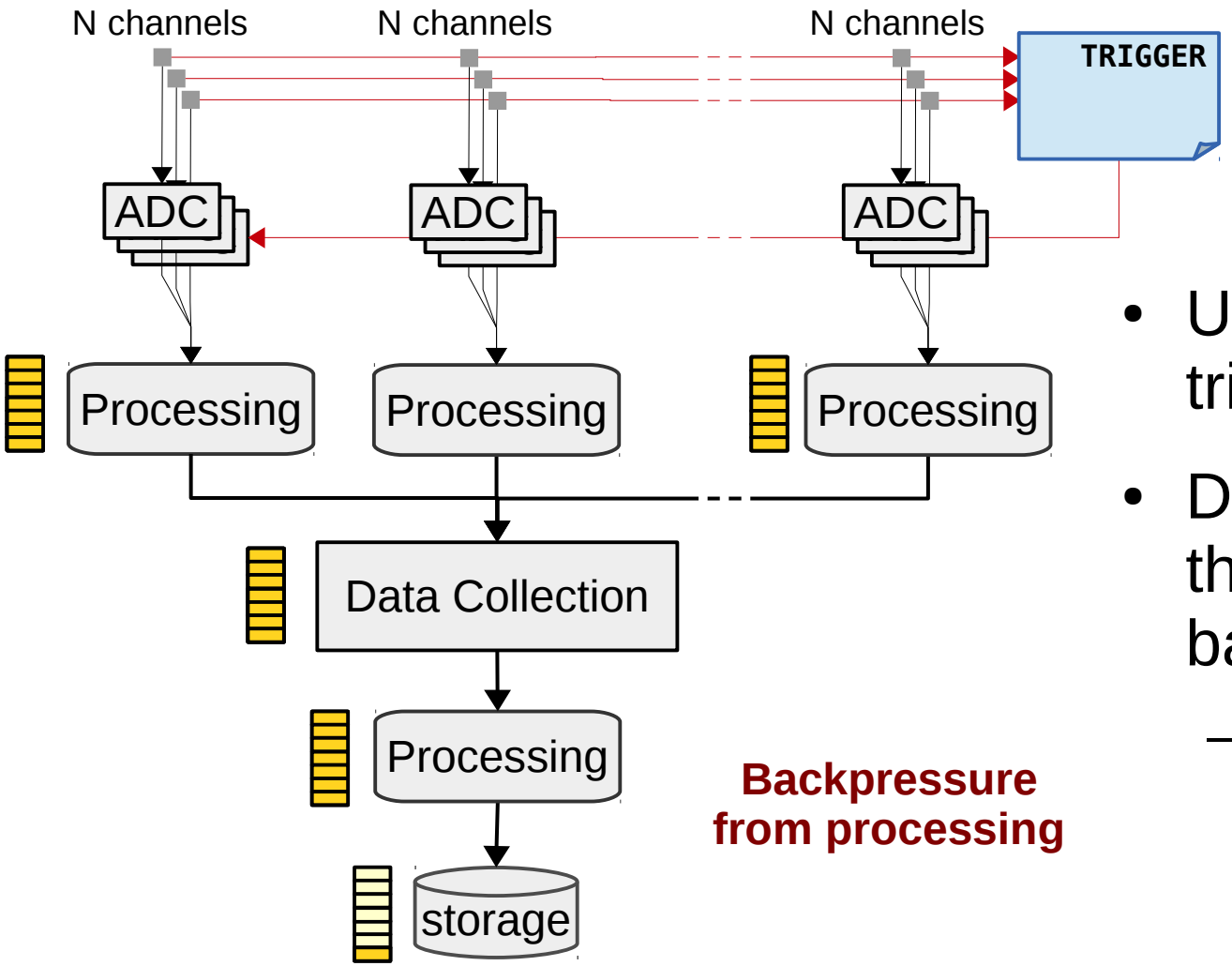
- If a system/buffer gets saturated
  - the “pressure” is propagated upstream (**back-pressure**)



- Up to exert busy to the trigger system
- Debugging: where is the source of backpressure?
  - follow the buffers occupancy via the monitoring system

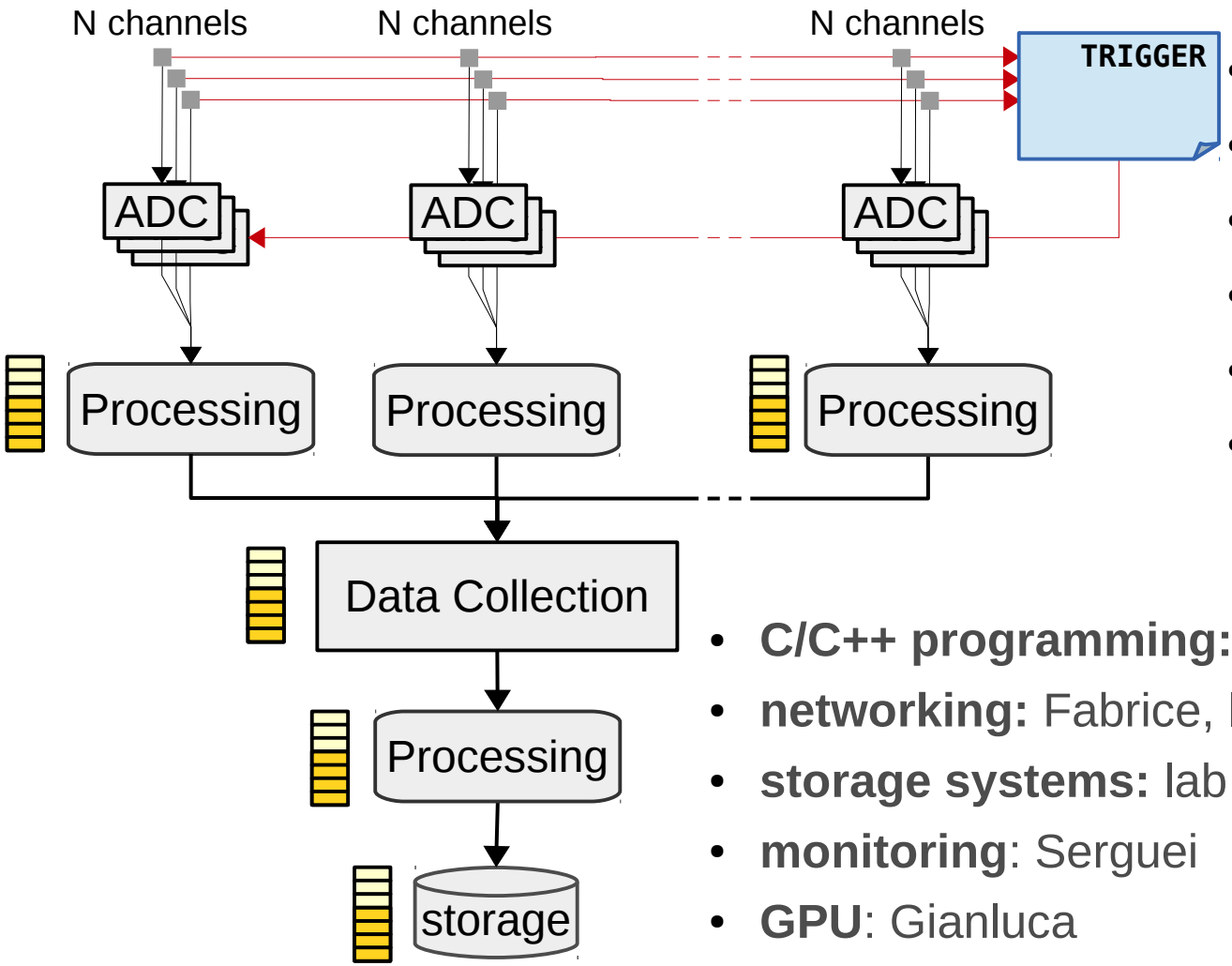
Backpressure

- If a system/buffer gets saturated
  - the “pressure” is propagated upstream (**back-pressure**)



- Up to exert busy to the trigger system
- Debugging: where is the source of backpressure?
  - follow the buffers occupancy via the monitoring system

Talks and Labs on specific topics

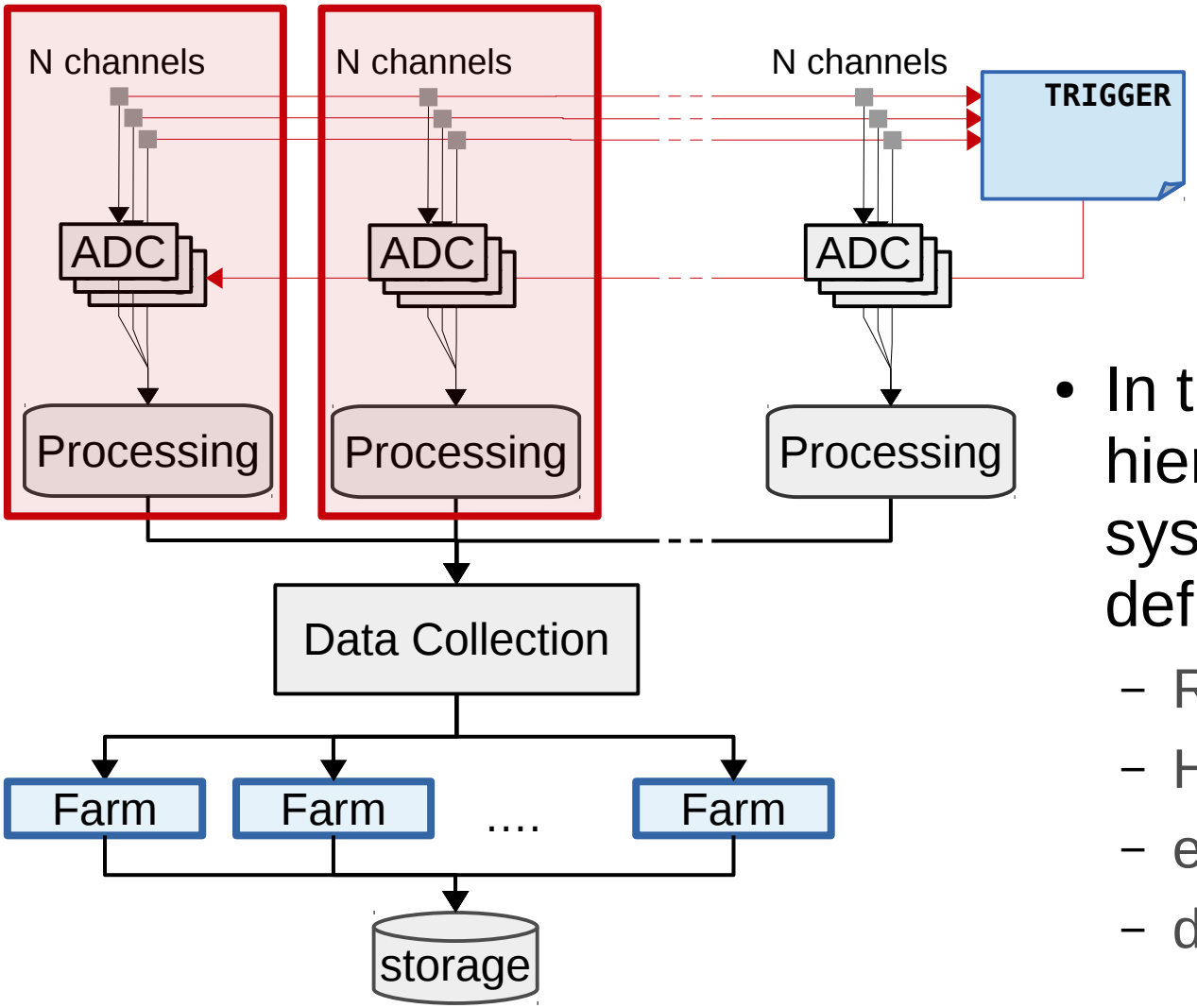


- **asic:** Ozgur
- **fpga**: Hannes, Manoel, lab 5
- **µcontr.**: Mauricio, lab 10
- **vme**: Markus, lab 1
- **atca**: lab 6
- **pci**: Paolo
- **optical links**: Paolo
- **assocative memories**: AN
- **timing**: Sophie

- **C/C++ programming:** Alessandro
- **networking:** Fabrice, lab 9
- **storage systems:** lab 11
- **monitoring**: Serguei
- **GPU**: Gianluca

Building blocks

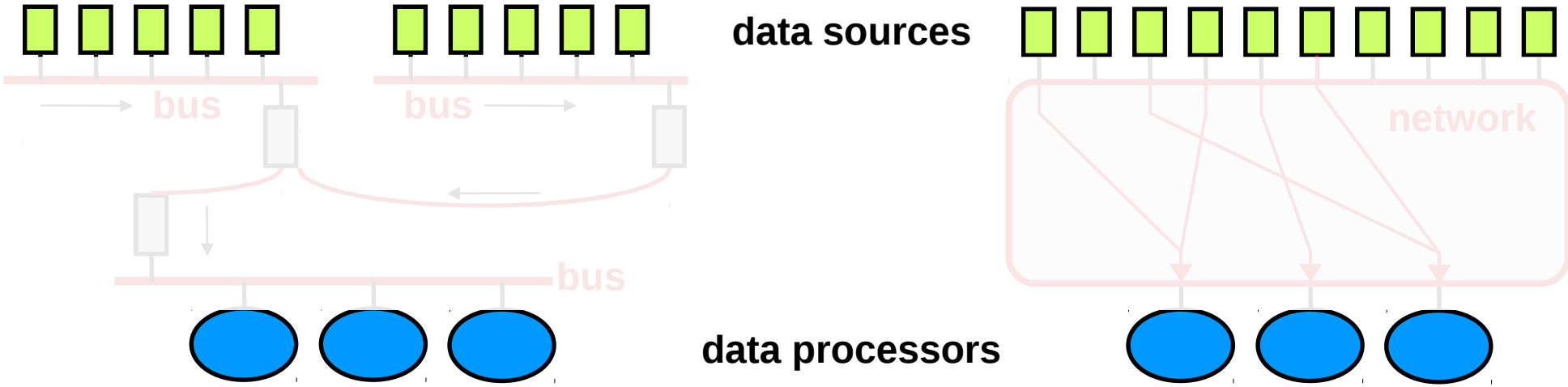
- Reading out data or building events out of many channels requires many components



- In the design of our hierarchical data-collection system, we have better define "**building blocks**"
  - Readout crates
  - HLT racks
  - event building groups
  - daq slices

Readout Topology

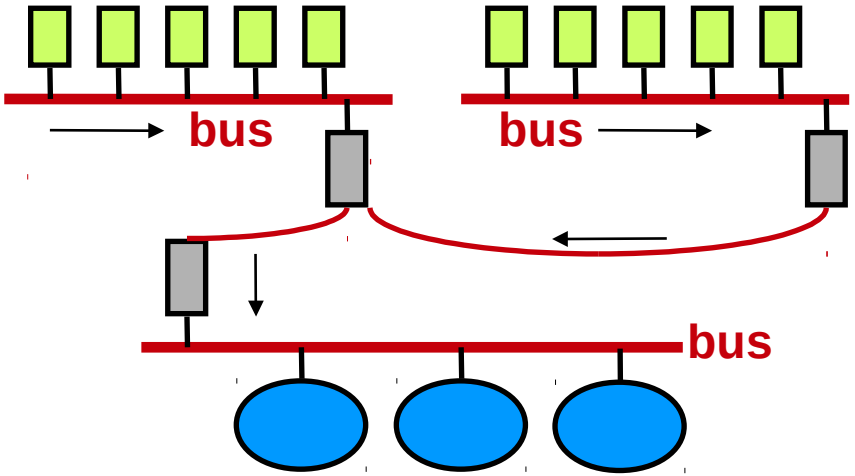
- How to organize the interconnections inside the building blocks and between building blocks?
  - How to connect data sources and data destinations?
  - Two main classes: **bus** or **network**

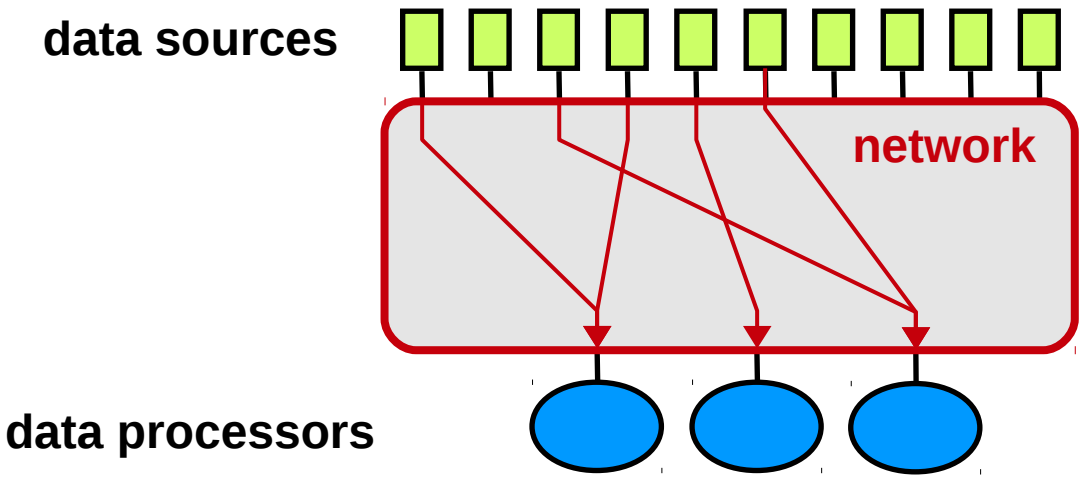


  - Warning: bus and network are generic concepts that can be easily confused with their most common implementations

Readout Topology

- How to organize the interconnections inside the building blocks and between building blocks?
  - How to connect data sources and data destinations?
  - Two main classes: **bus** or **network**





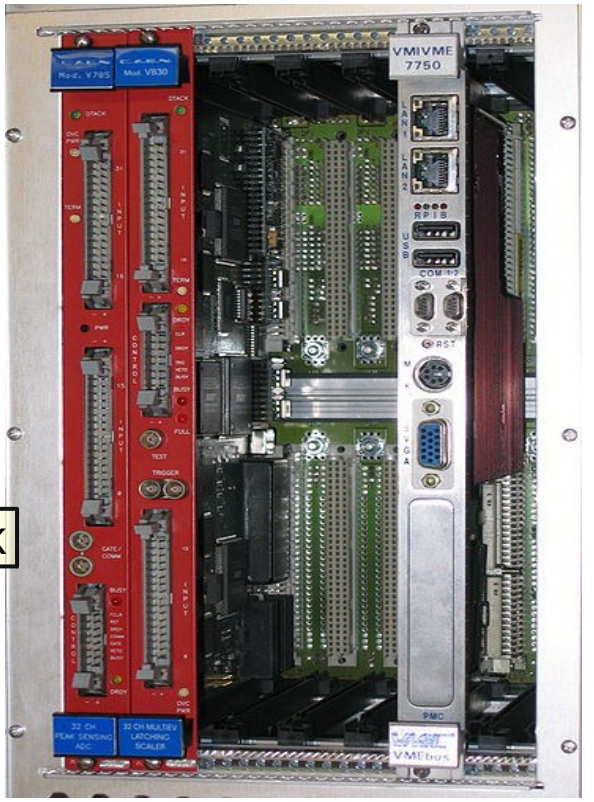
  - Warning: bus and network are generic concepts that can be easily confused with their most common implementations

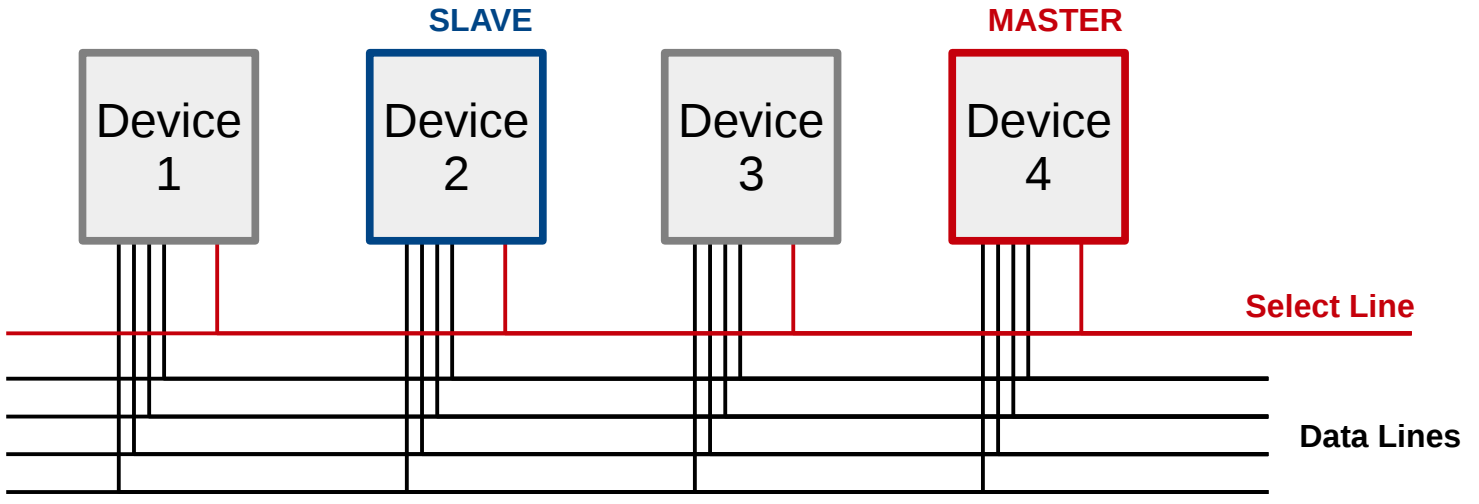
Buses

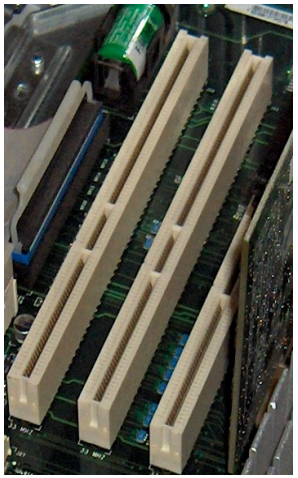
- Devices connected via a **shared bus**
  - Bus → group of electrical lines
- Sharing implies **arbitration**
  - Devices can be **master** or **slave**
  - Devices can be addresses (uniquely identified) on the bus
- E.g.: SCSI, Parallel ATA, VME, PCI …
  - local, external, crate, long distance, ...

Markus' talk

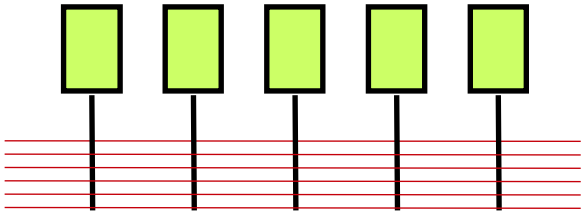
Paolo's talk





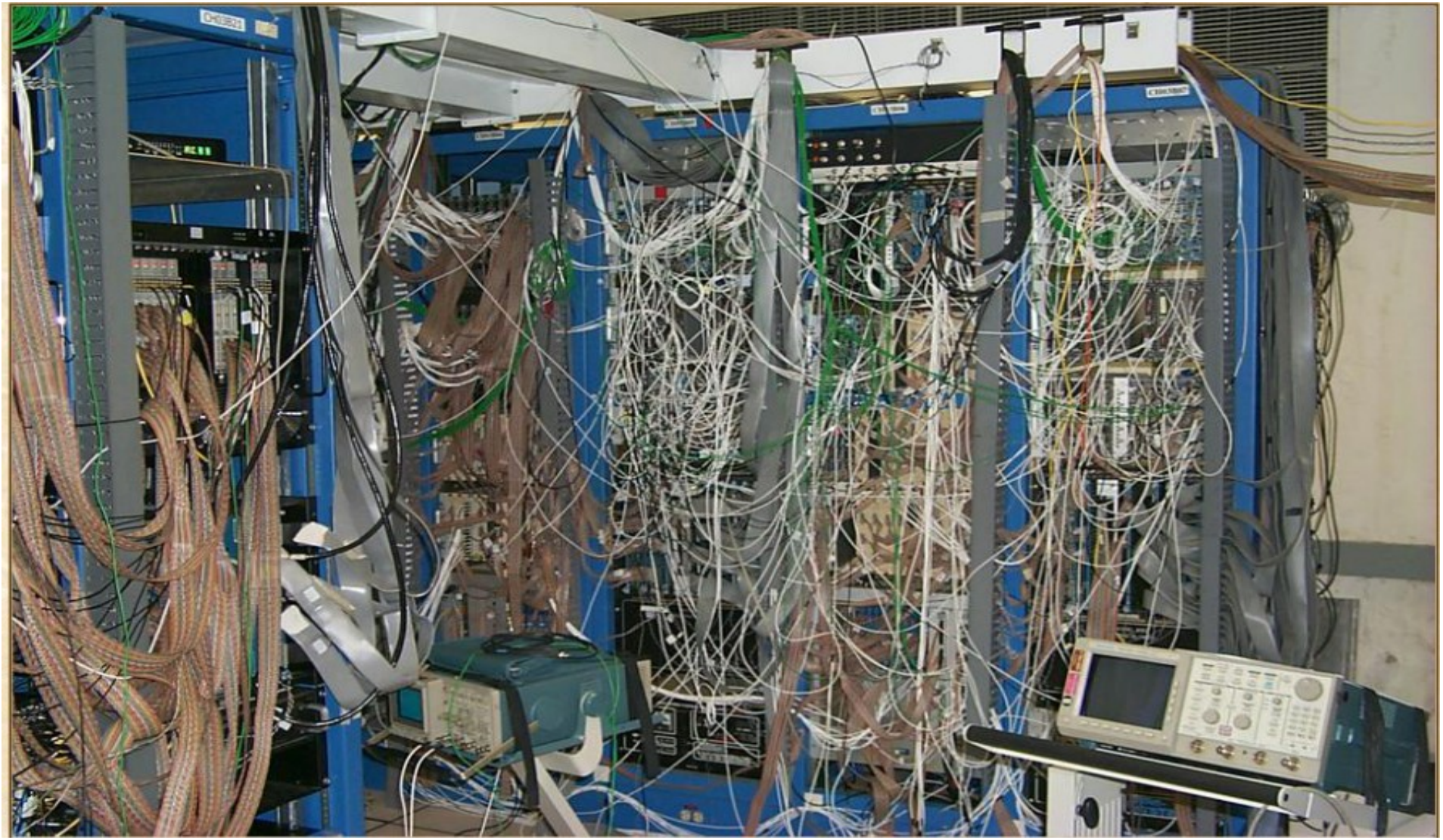


Bus facts



- Simple :-)
  - Fixed number of lines (bus-width)
  - Devices have to follow well defined interfaces
    - Mechanical, electrical, communication, ...
- **Scalability** issues :-(
  - Bus bandwidth is shared among all the devices
  - Maximum bus width is limited
  - Maximum number of devices depends on bus length
  - Maximum bus frequency is inversely proportional to the bus length
  - On the long term, other “effects” might limit the scalability of your system

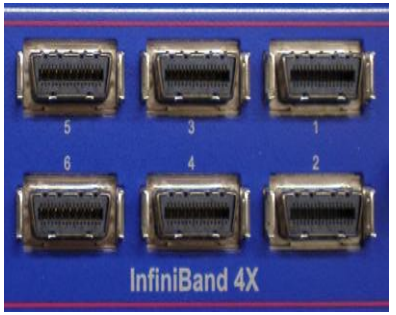
Bus facts

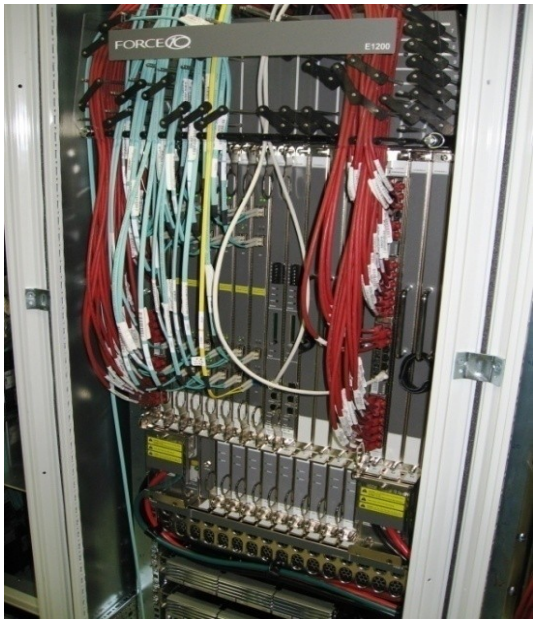


- On the long term, other “effects” might limit the scalability of your system

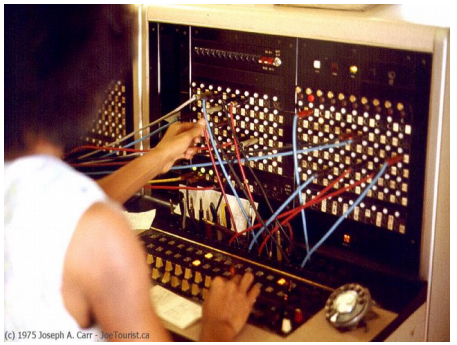
Network

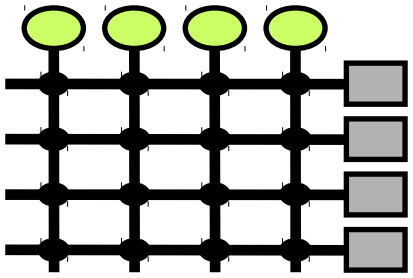
Lab 9





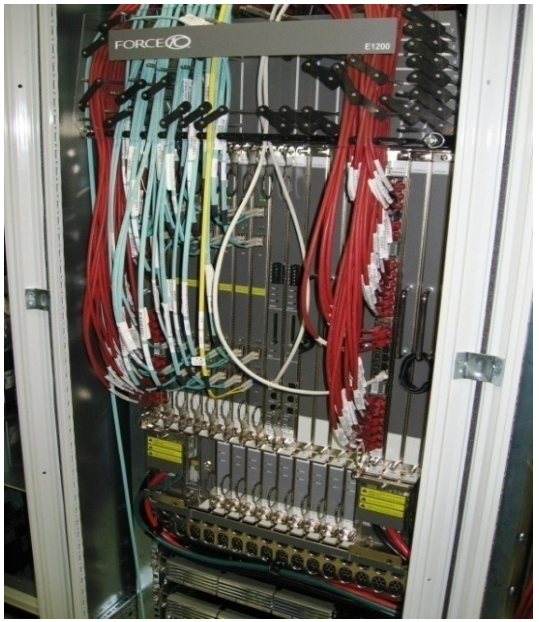
- All devices are **equal**
  - Devices communicate directly with each other via messages
  - No arbitration, simultaneous communications
- Examples:
  - Telephone, Ethernet, Infiniband, …
- In switched networks, **switches** move messages between sources and destinations
  - Find the right path
  - Handle **congestions** (two messages with the same destination at the same time)
    - The key is ....



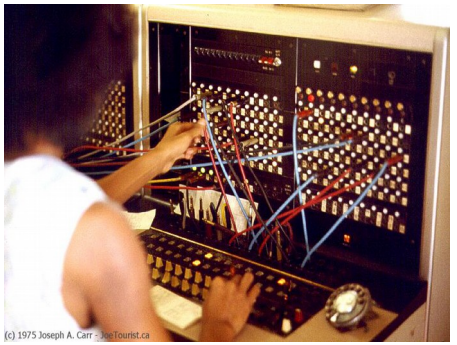


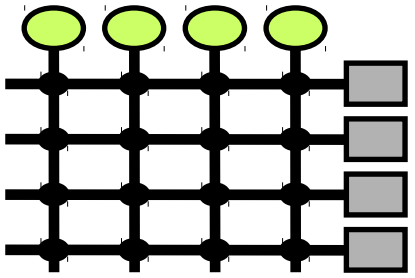
Network

Lab 9



- All devices are **equal**
  - Devices communicate directly with each other via messages
  - No arbitration, simultaneous communications
- Examples:
  - Telephone, Ethernet, Infiniband, …
- In switched networks, **switches** move messages between sources and destinations
  - Find the right path
  - Handle **congestions** (two messages with the same destination at the same time)
    - The key is .... **buffering**

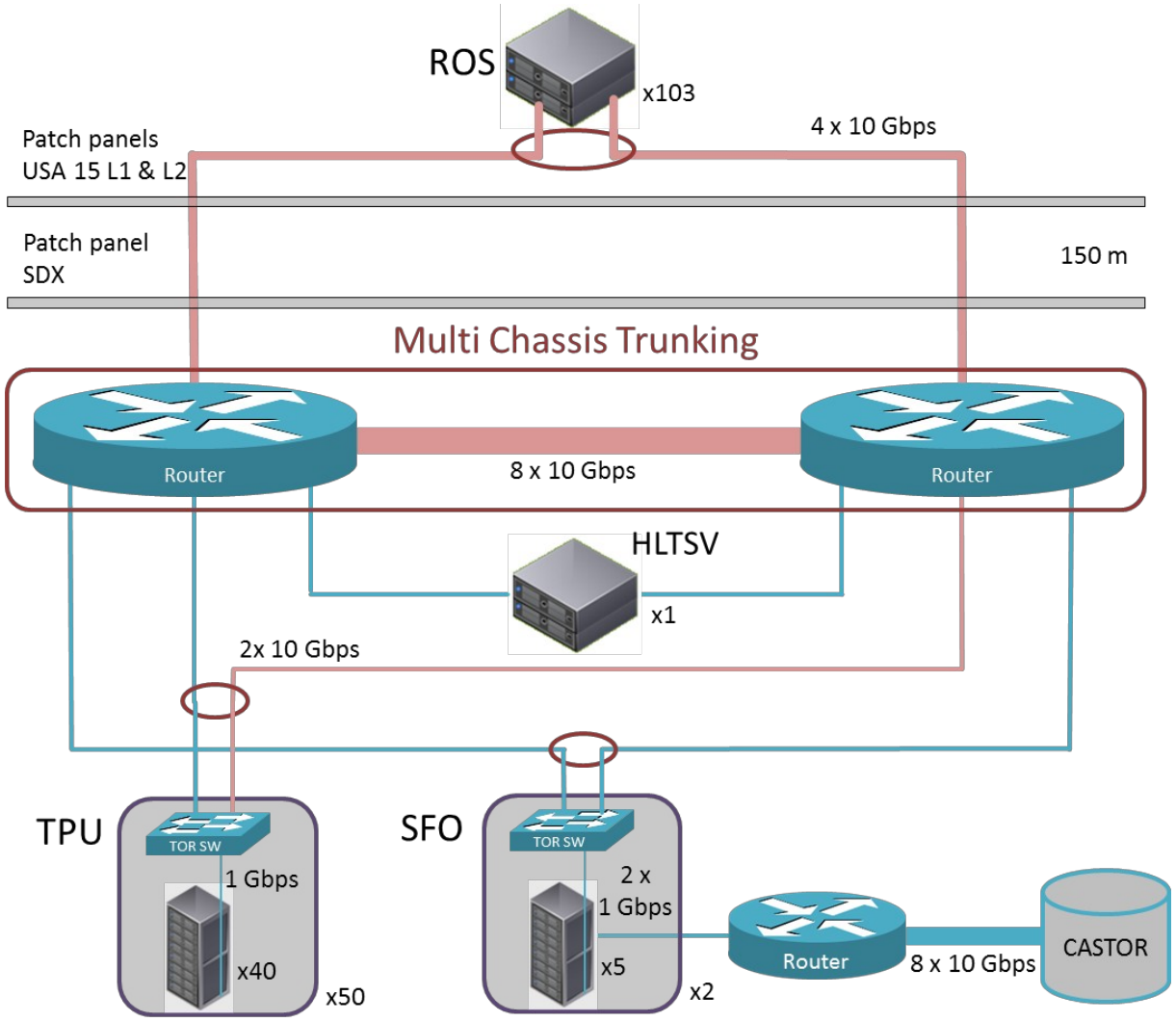




Network

Fabrice's talk

- Networks scale well (and allow redundancy)
  - They are the backbones of LHC DAQ systems



Outline

- Introduction
  - What is DAQ?
  - Overall framework
- Basic DAQ concepts
  - Digitization, Latency
  - Deadtime, De-randomization
- Scaling up
  - Readout and Event Building
  - Buses vs Network
- Do it yourself

DAQ Mentoring

- Study the trigger properties
  - Periodic or stochastic, continuous or bunched
- Consider the needed efficiency
  - It is good to keep operation margins, but avoid over-sizing
- Identify the fluctuation sources and size adequate buffering mechanisms
  - Watch out: (deterministic) complex systems introduce fluctuations: multi-threaded software, network communications, ...
- An adequate buffer is not a huge buffer
  - Makes your system less stable and responsive, prone to divergences and oscillations. Overall it decreases reliability

DAQ Mentoring

- Keep it simple, keep under control the number of free parameters without losing flexibility
  - Have you ever heard about SUSY phase-space scans? Do you really want something like that for your DAQ system?
- Problems require perseverance
  - Be careful, a rare little glitch in your DAQ might be the symptom of a major issue with your data
- In any case, ...

DAQ Mentoring

- Keep it simple, keep under control the number of free parameters without losing flexibility
  - Have you ever heard about SUSY phase-space scans? Do you really want something like that for your DAQ system?
- Problems require perseverance
  - Be careful, a rare little glitch in your DAQ might be the symptom of a major issue with your data
- In any case, ...