Alma Mater Studiorum – Università di Bologna

Dottorato di ricerca in
Fisica
Ciclo XXXII

Settore Concorsuale: 02/A1
Settore Scientifico disciplinare: FIS/01

Study of the data acquisition network for the triggerless data acquisition of the LHCb experiment and new particle track reconstruction strategies for the LHCb upgrade

Presentata da: Flavio Pisani

Coordinatore Dottorato
Prof.ssa Silvia Arcelli

Supervisore
Prof. Angelo Carbone

Esame finale anno 2020

Abstract

The LHCb experiment will receive a major upgrade by the end of February 2021. This upgrade will allow the recording of proton-proton collision data at $\sqrt{s} = 14$ TeV with an instantaneous luminosity of $2 \cdot 10^{33}$ cm^{-2}s^{-1}, making possible measurements of unprecedented precision in the b and c-quark flavour sectors.

For taking advantage of the increased luminosity provided, the data acquisition system will receive a substantial upgrade. The upgraded system will be capable of processing the full collision rate of 30 MHz, without any low-level hardware preselection. This new design constraint poses a non-trivial technological challenge, both from a networking and computing point of view.

A possible design of a 32 Tb/s data acquisition network is presented, and low-level network simulations are used to validate the design. Those simulations use an accurate behavioural model developed and optimised for this specific purpose.

It is mandatory to optimise the reconstruction algorithms using a computing and physics approach, to perform the online reconstruction of the full 30 MHz pp collisions rate. A new parametrisation of the charged particles' bending generated by the dipole of the LHCb experiment is presented. The accuracy of the model is tested against Monte Carlo data. This strategy can reduce by a factor four the size of the search windows needed in the SciFi sub-detector. The `LookingForward` algorithm in the Allen framework uses this model.

Contents

Acronyms

100GbE	100 Gigabit Ethernet
a2a	All-to-All
ALICE	A Large Ion Collider Experiment
ATAPC	All-To-All Personalised Communication
ATLAS	A Toroidal LHC ApparatuS
BU	Builder Unit
CMS	Compact Muon Solenoid
COTS	Commercial Off-The-Shelf
CRC	Cyclic Redundancy Check
DAQ	Data Acquisition
DAQPIPE	DAQ Protocol-Independent Performance Evaluator
EB	Event Building
ECAL	Electromagnetic Calorimeter
EDR	Enhanced Data Rate
EM	Event Manager
FCCL	Flow Control Credit Limit
FCTBS	Flow Control Total Blocks Sent
FIFO	First In First Out

flit Flow Control Unit

FPGA Field Programmable Gate Array

GbE Gigabit Ethernet

GPL General Public License

HCA Host Channel Adapter

HCAL Hadronic Calorimeter

HDR High Data Rate

HEP High Energy Physics

HLT High Level Trigger

HPC High Performance Computing

IB InfiniBand

IBTA InfiniBand Trade Association

IP Impact Parameter

J-sim JavaSim

LAN Local Area Network

LC Link Controller

LFT Linear Forwarding Table

LHCb Large Hadron Collider beauty

LHC Large Hadron Collider

LPCRC Link Packet CRC

LS2 Long Shutdown 2

MaPMT Multi anode PhotoMultiplier Tube

MC Monte Carlo

MEP Multi Event Packet

MFP Multi Fragment Packet

MPI Message Passing Interface

MS Multiple Scattering

MWPC Multi-Wire Proportional Chamber

ns-3 network simulator 3

OFA Open Fabric Alliance

OMNeT++ Objective Modular Network Testbed in C++

Op Operation

OS Operating System

PCIe Peripheral Component Interconnect Express

PFC Priority-based Flow Control

PID Particle IDentification

PSB Proton Synchrotron Booster

PS Proton Synchrotron

PTP Precision Time Protocol

PV Primary Vertex

QoS Quality of Service

RDMA Remote Direct Memory Access

RF Radio Frequency

RICH Ring Imaging Cherenkov

RoCE RDMA over Converged Ethernet

RU Readout Unit

SAF Store and Forward

SciFi Scintillating Fiber

SPS Super Proton Synchrotron

UT Upstream Tracker

VCT Virtual-cut-through

VELO VErtex LOcator

VL Virtual Lane

WLS WaveLenght Shifter

WR Work Request

Introduction

The Large Hadron Collider beauty (LHCb) experiment will receive a major upgrade by the end of a scheduled maintenance period of the Large Hadron Collider (LHC), called Long Shutdown 2 (LS2) (Dec 2018-Feb 2021). During the third run of the LHC (Run 3), this upgrade will allow the recording of proton-collision data at $\sqrt{s} = 14$ TeV with an instantaneous luminosity of $2 \cdot 10^{33}$ cm^{-2}s^{-1}. The upgraded LHCb experiment is expected to collect at least 50 fb^{-1} in less than ten years of operation, making possible unprecedented precision measurements in the b and c-quark flavour sectors.

In order to take advantage of the increased luminosity provided during Run 3, it is necessary to remove several limitations present in the current experiment. One of the most challenging ones is the removal of the 1.1 MHz fixed latency readout rate limit. In the current system, the full inelastic collision rate of 30 MHz is reduced by a factor ~ 30 by a fixed latency hardware trigger called Level-0. The inefficiencies introduced by the Level-0 decision are the dominating factor in the entire chain, and in order to take advantage of the luminosity that will be delivered during Run 3, the Level-0 trigger will be removed.

Removing the Level-0 rate reduction requires that the full collision rate of 30 MHz will be injected into the readout system, corresponding to a data-rate of 32 Tb/s. This new design constraint requires an architectural change of the entire Data Acquisition (DAQ) system of the experiment, from the low-level front-end detector electronic to the complex algorithms needed to reconstruct in the data generated by the pp collisions. This upgraded DAQ system poses a non-trivial technological challenge, both from a networking and computing point of view.

The LHCb DAQ networking will be completely rebuilt in order to transfer the full data-rate produced by the experiment. This task requires the use of the most advanced network technologies available at the time of writing, and extensive planning and testing before the actual procurement of the system. In this thesis, multiple aspects of the problem will be discussed and analysed, such as the scheduling of the traffic; the allocation and the management of mission-critical resources; the selection of the network technology and the global architecture of the network. Before the commissioning of the system, an *a priori* quantitative evaluation of all the aspects mentioned above is almost impossible; therefore, as part of this thesis project, a low-level network simulation framework has been developed. This powerful tool provides quantitative predictions about the behaviour of real network infrastructure, and how it reacts to the traffic generated by the data recorded from the *pp* collision in the LHCb detector.

Transferring the data over the DAQ interconnection system is a necessary step in the full DAQ process, but it is not the only one. In order to provide useful data for the physics analysis, the full data-rate of 32 Tb/s has to be reconstructed – i.e. the physical quantities have to be extracted from the data – and filtered – i.e. the interesting collisions have to be selected. This process poses a non-trivial computing challenge, which can be tackled both from a computer science and a physics point of view. From the computer scientist's point of view, the existing code-base can be optimised to take advantage of newer capabilities provided by more modern computing architectures. From the physicist's point of view, it is necessary to develop new algorithms and strategies in order to reduce the computing power needed to achieve the same physics results. The second strategy has been followed for the second part of this thesis project; in particular, a parametric description of the charged particles deflection introduced by the LHCb magnet has been developed. This particles' trajectory description can be used to reduce the computing power needed by the track reconstruction of the LHCb experiment. The `LookingForward` algorithm in the Allen framework uses this model.

The organisation of the thesis is as follows. Chapter one gives an introduction about the CERN accelerator complex and the LHCb experiment. The configuration described is the one that will be used during Run 3 of the LHC.

Chapter two gives a theoretical introduction about interconnection networks; the focus of this overview being all the aspects needed in the design on the LHCb DAQ network.

Chapter three contains a detailed description of the networking infrastructure needed by the upgraded LHCb experiment. The network constraints and the design strategy are described in detail, together with the implementation details of the low-level simulation model and the results from simulated systems.

Chapter four presents the magnetic field deflection model, including the derivation of the model itself and a quantitative test of the model's prediction accuracy. Finally, in order to conclude this thesis, there is a summary of the results and discussion over the conclusions.

Chapter 1

The LHCb experiment at the LHC

1.1 The CERN

After the second world war, in Europe, the need for peace and cohesion was felt. A significant number of distinguished scientists consider the possibility to build a world-class atomic physic research laboratory to provide a force for unity in post-war Europe and allow scientists all around the world to share the costs of an expensive research facility; consequently in 1952 a provisional council was set up in order to create this facility, that nowadays we know as CERN. Geneva was chosen for its central Europe position, the presence of other international organisation, such as the Red Cross and, last but not least, the neutrality of Swiss during WWII. 12 countries, including Germany, France and Italy, founded CERN; and the number of member state increased over time and nowadays there are 23 member state and 7 associate member state, and this makes CERN the largest particle physics research facility in the world and an actual meeting point for people from all the word and a laboratory of peace. CERN is not just a symbol of a united Europe, but mainly a cutting-edge laboratory for particle physics research. Several notable achievements have been made trough experiments performed at CERN, they include:

- 1973: the discovery of neutral current with the Gargamelle bubble chamber [1];
- 1983: the discovery W and Z bosons with the UA1 and UA2 [2];
- 1989: the determination of the number of light neutrino families with ALEPH, DELPHI, L3 and OPAL [3];

- 1995: the first creation of antihydrogen [4];
- 1999: the discovery of direct CP violation with NA48 [5];
- 2012: The discovery of Higgs boson with ATLAS and CMS [6, 7];
- 2015: the discovery of pentaquarks with LHCb [8];
- 2019: the discovery of CP violation in charm hadrons with LHCb [9].

1.2 The Large Hadron Collider

The Large Hadron Collider (LHC) is a particle accelerator and collider designed to collide hadrons. It is installed in a 27 km tunnel located 100 m underground. The LHC is designed to accelerate protons beams up to an energy of 7 TeV and then to collide them with a centre-of-mass energy of 14 Tev, and accelerate heavy ions beams, such as Pb and Xe, up to an energy of 1.4 TeV per nucleon and centre-of-mass energy of 2.8 TeV per nucleon. The accelerator provides a peak luminosity of $10^{34}\ cm^{-2}s^{-1}$ and $10^{27}\ cm^{-2}s^{-1}$ when colliding protons and heavy ions respectively. The LHC can achieve the aforementioned characteristic thanks to its state-of-the-art design based on superconducting elements; including 1232 dipole magnets for beam steering, and 392 quadrupole magnets for beam focusing.

The protons needed to generate the collisions in the LHC can be efficiently produced from hydrogen atoms, by stripping off all the electrons from the atoms. In order to reach the final collision energy from their quasi-rest condition, the protons have to pass through a complex chain of accelerators, shown in Figure 1.1. The new upgraded accelerator chain will start with a source of H^- ions, right after the source, the ions will be injected into Linac4, a linear accelerator which accelerates the H^- ions to 160 MeV. At this point, all the electrons are removed from the H^- ions, and the resulting protons are injected in the Proton Synchrotron Booster (PSB), a set of four superimposed synchrotrons, which accelerate the protons to an energy of 1.4 GeV. The next step in the chain is the Proton Synchrotron (PS), a 628 m circular accelerator which brings the proton up to an energy of 25 GeV. Then, the protons are transferred from PS to Super Proton Synchrotron (SPS); their energy is boosted up to 450 GeV; this is the last step before being injected into the LHC.

Because the LHC has two beams circulating in opposing directions, the injection takes place via two separate tunnels, one for every beam. The LHC is the filled

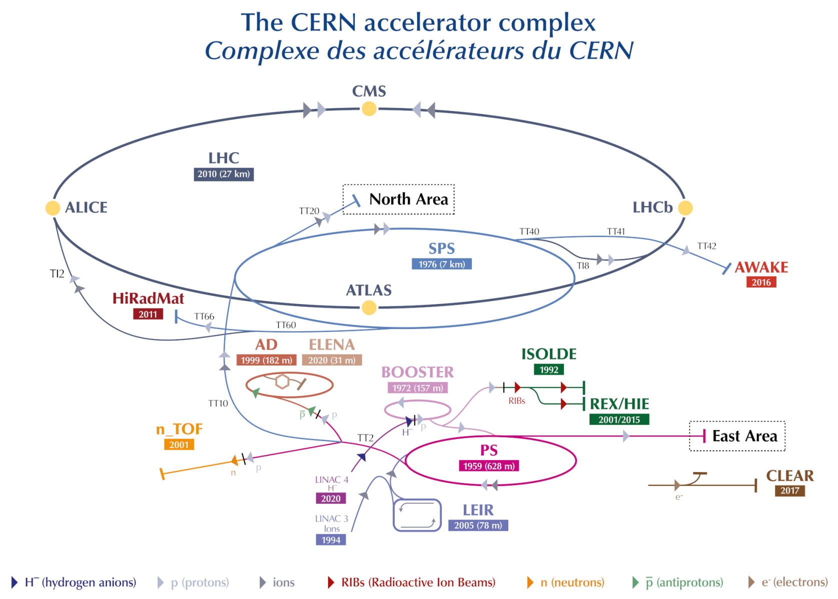


Figure 1.1. Schematic view of the CERN accelerator complex as it will during Run 3. Original image from CERN.

up with 2808 bunches of 10^{11} protons per ring[1]. When the filling is completed the protons can be accelerated up their final energy of 7 TeV. In order to keep the particles beams on the correct circular trajectory, the superconductive dipole magnets have to generate a magnetic field of 8.34 T.

When the machine is ready to generate collisions the two beams collide at four interaction points around the ring every 25 ns, generating a collision rate of 40 MHz; around these four collision points, four experiments have been built:

- A Toroidal LHC ApparatuS (ATLAS): one of the two general-purpose detectors; it was designed to find the Higgs boson and particles predicted by the supersymmetry theory.
- Compact Muon Solenoid (CMS): the second general-purpose detector, ATLAS and CMS have the same goals, but different design; this choice was made to ensure that, on a given discovery, there was a double-blind response.
- A Large Ion Collider Experiment (ALICE): the heavy-ion detector. It was designed to study the properties of quark-gluon plasma and antimatter, to increase the insight into quantum chromodynamics.

[1]Those numbers refer to the design filling scheme of the LHC; the actual numbers may vary according to the operation conditions of the accelerator and the experiments.

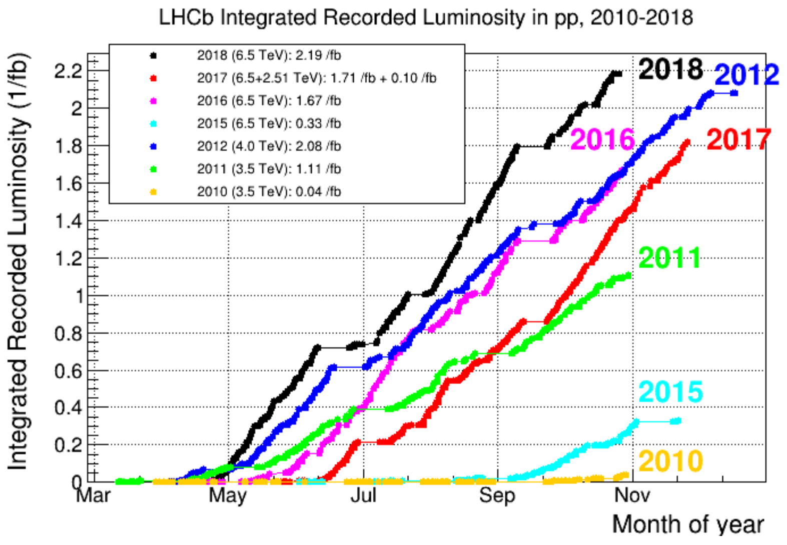


Figure 1.2. Integrated recorded luminosity by the LHCb experiment for each year. The data was collected during Run 1 in the 2010-2012 period, and during Run 2 in the 2015-2018 one.

- Large Hadron Collider beauty (LHCb): the specialised heavy quark flavour detector. The principal goal of LHCb is the determination of CP violation parameters. A detailed description of the LHCb experiment will be given in section 1.3.

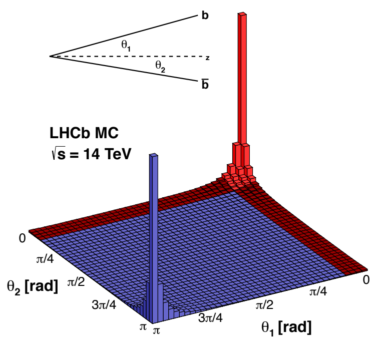
The LHC has delivered excellent performance during Run 1 and Run 2, allowing the LHCb experiment to record more than 9 fb^{-1} of integrated luminosity, as depicted in Figure 1.2.

1.3 The LHCb detector

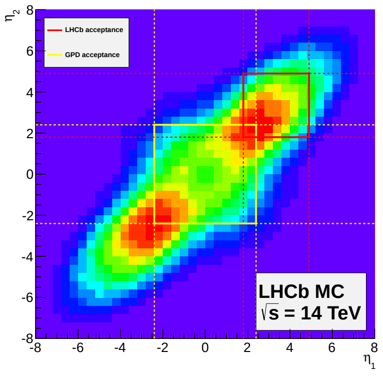
The LHCb experiment [10] was created to exploit the production cross-section of $b\bar{b}$ pairs in $p\bar{p}$ collision at the LHC and to study the flavour physics using hadrons containing b or c quarks. In fact, the cross-section values for $b\bar{b}$ collisions are [11]:

$$\sigma(pp \to b\bar{b}X)_{\sqrt{s}=7\,\text{TeV}} = (72.0 \pm 0.3 \pm 6.8)\ \mu\text{b}$$
$$\sigma(pp \to b\bar{b}X)_{\sqrt{s}=13\,\text{TeV}} = (154.3 \pm 1.5 \pm 14.3)\ \mu\text{b}$$

Those characteristics allow LHCb to be an ideal experiment for the study of b physics, but the same characteristics are optimal for the c physics as well; because



(a) $b\bar{b}$ production angle



(b) $b\bar{b}$ production η

Figure 1.3. Production angles with respect to the beam direction and pseudorapitidy of $b\bar{b}$ pairs produced in pp collisions with $\sqrt{s}$ = 14 TeV as obtained from fully simulated events. The LHCb aceptance region is highlighted in red.

the cross-section of cc production is even bigger than $b\bar{b}$ production [12, 13]:

$$\sigma(pp \to c\bar{c}X)_{\sqrt{s}=7\,\text{TeV}} = (1230 \pm 190)\ \mu\text{b}$$
$$\sigma(pp \to c\bar{c}X)_{\sqrt{s}=13\,\text{TeV}} = (2369 \pm 3 \pm 152 \pm 118)\ \mu\text{b}$$

There is a high asymmetry in the momenta of the two partons colliding in pp collisions, the b and c quarks are produced strongly boosted along the direction of the beam-line. Consequently, the angular distribution of the b and c hadrons is prominently in a specific forward or backward region, and with a small-angle within respect to the beam direction, as shown in Figure 1.3. For this reason, the geometry of LHCb is entirely different from the other LHC detectors; instead of having a cylindrical geometry, and therefore covering the full solid angle, it is a forward spectrometer, as shown in Figure 1.4. The geometrical acceptance of LHC is (10-300) *mrad* in the horizontal plane and (10-250) *mrad* in the vertical plane. Given the spectrometer nature of LHCb the bending of charged particle is obtained via a dipolar magnet that bends particles in the horizontal plane. The presence of the magnet justifies the asymmetry in the angular acceptance between the horizontal and the vertical plane. Therefore, the LHCb detector can detect particles that lie in a pseudorapidity η range (1.8 - 4.9); where η is defined as:

$$\eta = -ln\,tan\left(\frac{\theta}{2}\right) = \frac{1}{2}\,ln\frac{|\overrightarrow{p}| + p_L}{|\overrightarrow{p}| - p_L}$$

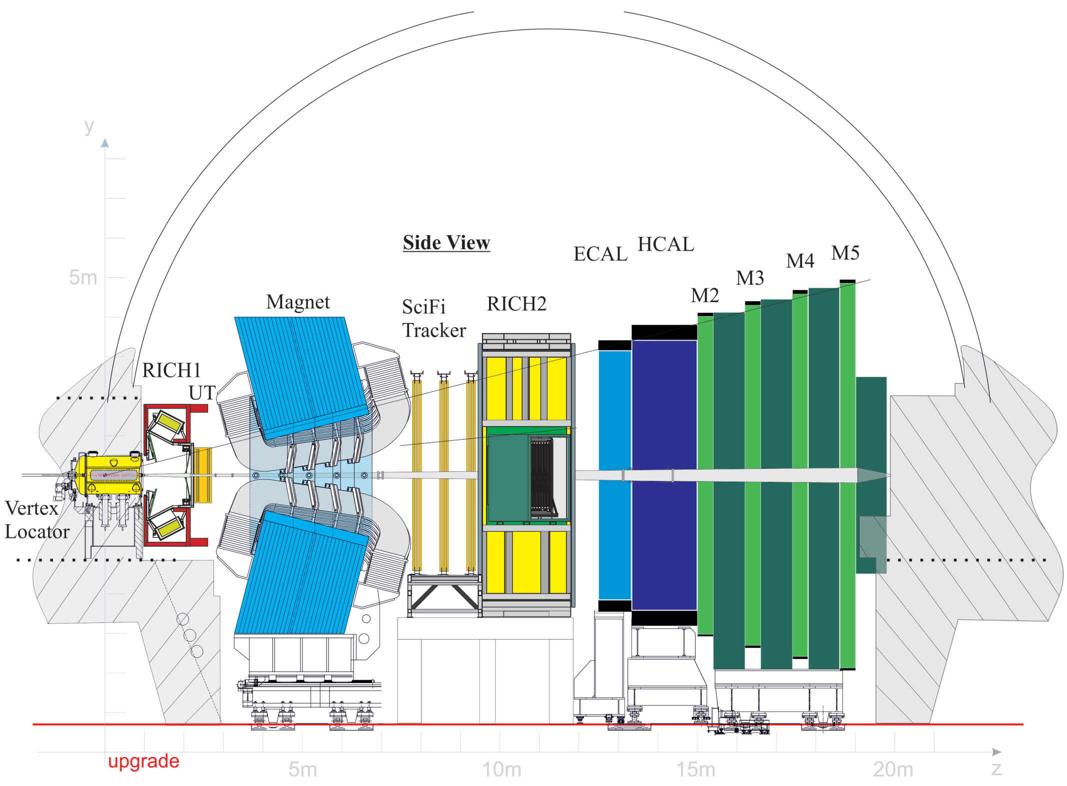


Figure 1.4. The LHCb detector layout in the configuration that will be used during Run 3. From left to right all the different sub-detectors are shown: VELO, RICH1, UT, SciFi tracker, RICH2, ECAL, HCAL and muons stations from M2 to M5.

where θ is the angle between the particle and the beam axis and p_L is the longitudinal momentum.

1.4 The LHCb tracking system

This section give an overview of the LHCb tracking system, as it will be during Run 3.

1.4.1 VErtex LOcator

The VErtex LOcator (VELO) is the closest sub-detector to the beam interaction point. Its primary purpose is to locate Primary Vertices (PVs) produced by pp collisions, identify tracks and assign them to the correct PV, and evaluate the Impact Parameter (IP). The distance between the PV and the closest approach of the track of the PV defines the IP. Because, at the LHCb production conditions, ground

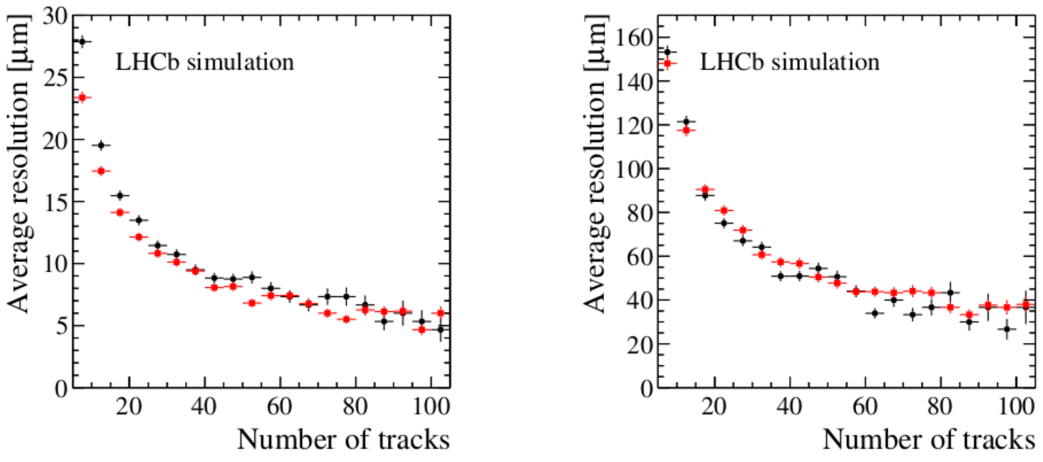


Figure 1.5. PV resolution in x (left) and z (right) as a function of the number of reconstructed tracks in the vertex. The current VELO is shown with black circles and the upgrade VELO with red squares. The resolutions in x and y are similar. As presented in [14].

state charm and beauty hadrons fly $\sim$ 1 cm before decaying, a good IP resolution is fundamental for reducing the combinatorial background and a correct b and c identification. The PV and IP resolutions are therefore crucial VELO performance indicators. The plots in Figure 1.5 show the PV resolution in x and z as a function of the number of tracks composing the vertex. Figure 1.6 shows the IP resolution as a function of the p_T of the track.

The key factor behind the excellent performances achieved by the VELO is in the design of the detector itself. This silicon-based tracking device is made of 52 modules placed on the two sides of the beamline and perpendicular to the beam itself. Each module is composed of 4 200 μm thick silicon sensors; the active area of each module is 42.46 $\times$ 14.08 mm^2, and it is segmented in 55 $\times$ 55 μm^2 pixels. In order to maximise the PV and the IP resolutions, the sensors need to be as close as possible to the beamline, in the upgraded VELO the distance of approach to the LHC beams will be just 5.1 mm. Because the beam configuration changes from one fill to the next one and from the injection phase to the collisions phase; the VELO is divided into two retractable halves. Figure 1.7 shows a 3D rendering of the full VELO detector in the closed position.

In order to reduce the material budget before the first sensors, the VELO modules are encapsulated in an isolated vacuum vessel. The first vacuum of the LHC is separated from the secondary vacuum of the VELO by the RF foil, a thin aluminium

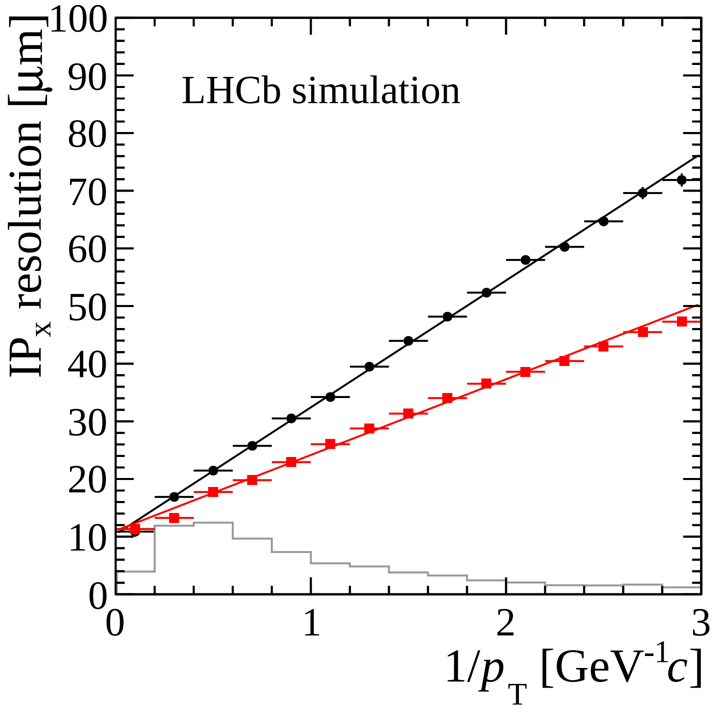


Figure 1.6. IPx resolution for long tracks for VELO Upgrade, in red, compared to the expected performance of current VELO design in upgrade conditions, in black, as described in [14].

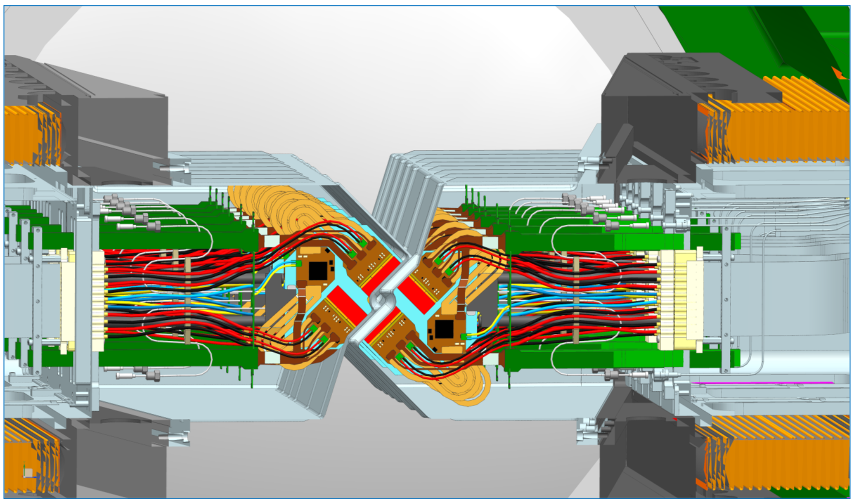


Figure 1.7. X section of the VELO detector in the closed position. Credits of the image to Freek Sanders.

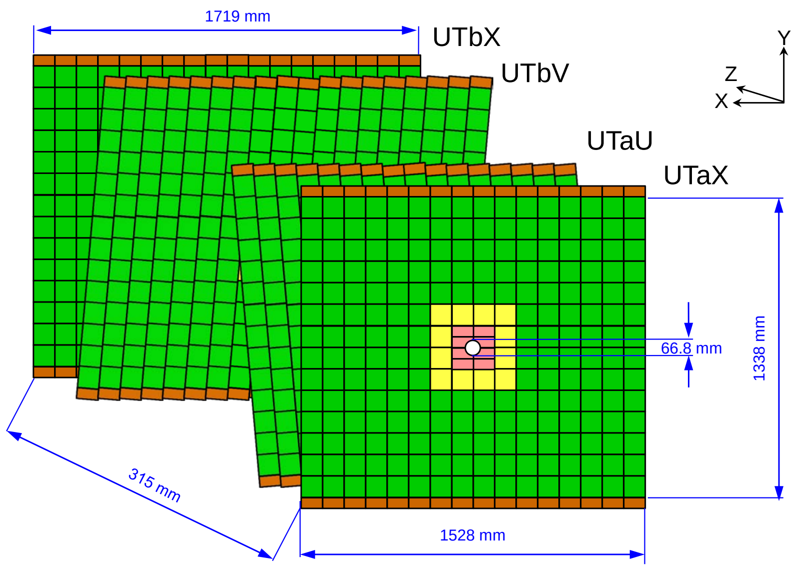


Figure 1.8. Geometry of the UT detector, as presented in [15]. The four colours intensify the silicon sensor used in every region.

foil which shields Radio Frequency (RF) interference produced by the circulating beams. In order to dissipate heat produced by the detector and the detector electronics inside the vacuum vessel, an advanced CO_2-based cooling system has been implemented which is designed to dissipate 1000 W on each half of the detector. More details about the physical implementation of the detector can be found in [14].

1.4.2 Upstream Tracker

The Upstream Tracker (UT) is placed right before the magnet in a region pervaded by a fringe magnetic field. The detector is composed of four planes of silicon strips, as depicted in Figure 1.8. The planes are divided into two stations separated by 315 mm and called UTa and UTb. In order to provide x/y coordinates and increase the reconstruction efficiency, the strips in the planes have different orientations; in particular the first and the last planes form a $0°$ angle with the y-axis, while the second and the third layers are installed at a $+5°$ and $-5°$ angle respectively.

In every plane, the sensors are arranged into columns, called "stave". Four different sensors are installed in various regions of the detector, as shown in Figure 1.8. All the sensors have a 250 μm thickness and an expected hit resolution of 50 μm.

The main purpose of the UT detector is to perform a fast estimation of p and p_T of charged particles using solely VELO and UT data. This first momentum

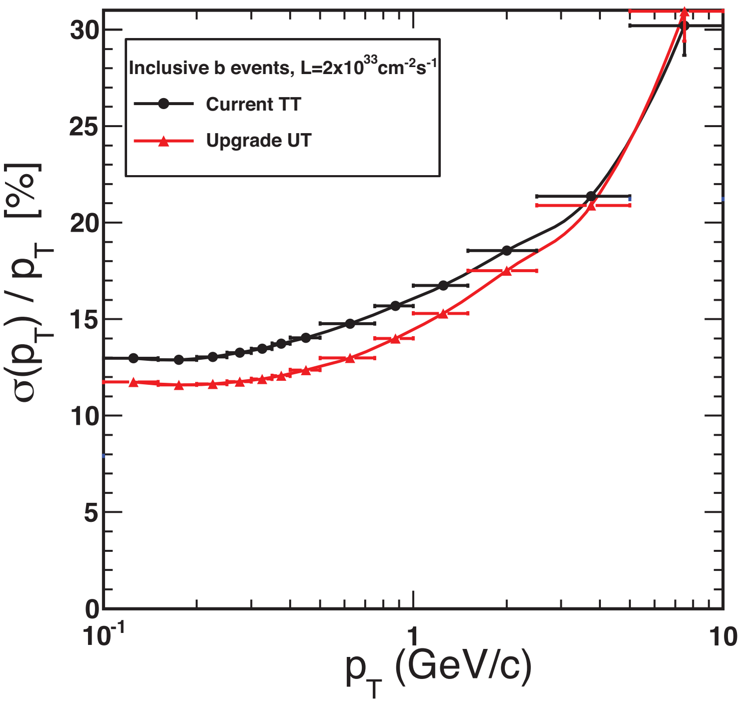


Figure 1.9. p_T resolution as a function of p_T, the previous detector is shown with (black) circles, the expected performance of the upgraded UT is shown in (red) triangles. As presented in [15].

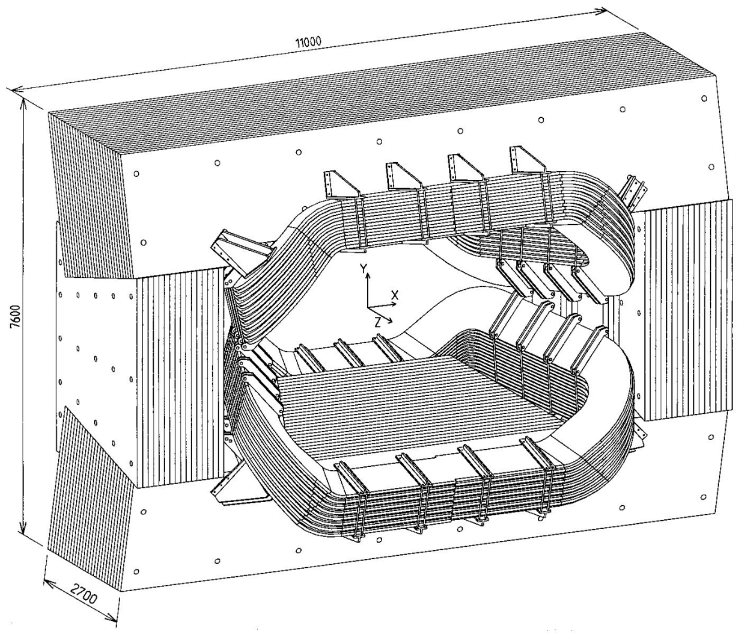


Figure 1.10. Scheme of the LHCb dipole magnet, as presented in [16].

estimation is essential to improve the online reconstruction quality of downstream detectors and reduce the computation time needed by the online reconstruction framework. The expected p_T resolution is shown in Figure 1.9.

1.4.3 Magnet

The magnetic field needed to measure p and p_T of charged particles is provided by a warm dipole, which generates an integrated magnetic field of approximately 4 Tm. The magnetic field is generated by two 25 tons identical coils symmetrically placed inside a 1450 tons yoke, as depicted in Figure 1.10.

The main component of the $\overrightarrow{B}$ filed is directed along the y axis, and it generates a bend in the charged particles trajectory on the x-z plane allowing measurement of their momentum. In order to define the expected trajectory of a charged particle, it is crucial to know the value over the space of the magnetic field $\overrightarrow{B}(x, y, z)$ to integrate it along the track trajectory. Figure 1.11 depicts the magnetic field values along a central track a one with a horizontal and vertical angle of 197 mrad.

During data-taking, the magnetic field polarity is frequently reversed to allow the evaluation of any left-right asymmetry in the detector. Because oppositely charged particles are bent in the opposite direction by the magnetic field, any difference in

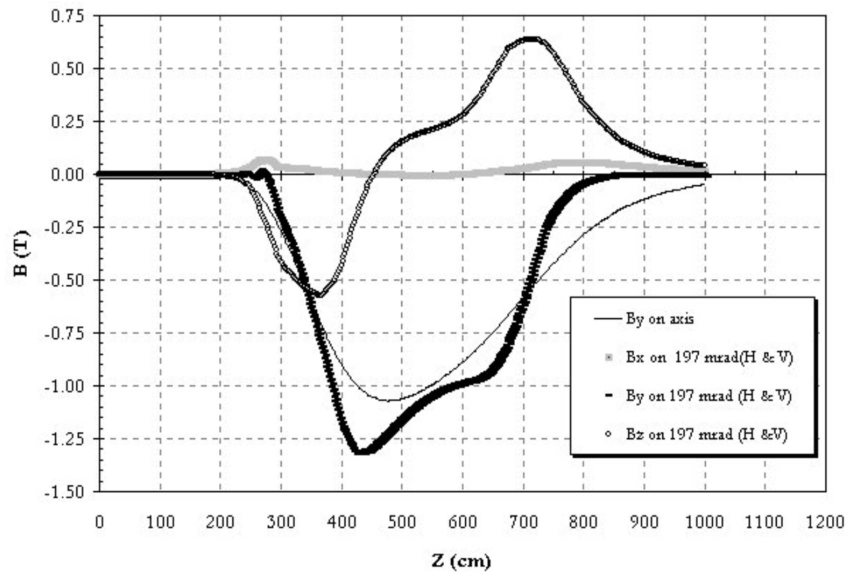


Figure 1.11. Magnetic field $\overrightarrow{B}$ along a central track and one with a horizontal and vertical angle of 197 mrad, as presented in [16].

the reconstruction efficiency between the left side and the right side of the detector can affect the CP symmetry measurements.

1.4.4 Scintillating Fiber (SciFi)

The Scintillating Fiber tracker is between the LHCb dipole and the RICH2 as shown in Figure 1.12. The tracker layout is arranged in three stations (T1, T2, T3) with four layers each, as depicted in Figure 1.13.

Every layer uses 2.5 m long multi cladding wavelength shifting scintillating fibres as an active material. The fibres in the first and the last layers in every station are parallel to the y-axis (*x-layers*), while in the two middle layers they are tilted by $\pm 5°$ (*u-layers* and *v-layers*). Every layer is built out of 12 modules, and a total of 144 modules are required to build the complete detector.

Thanks to the 250 μm fibres diameter, the modules provide a high granularity active region. The position resolution will be better than 100 μm in the bending plane. A higher resolution is not needed because the extrapolation of tracks from the VELO is dominated by the multiple scattering effects in the detectors upstream of the magnet.

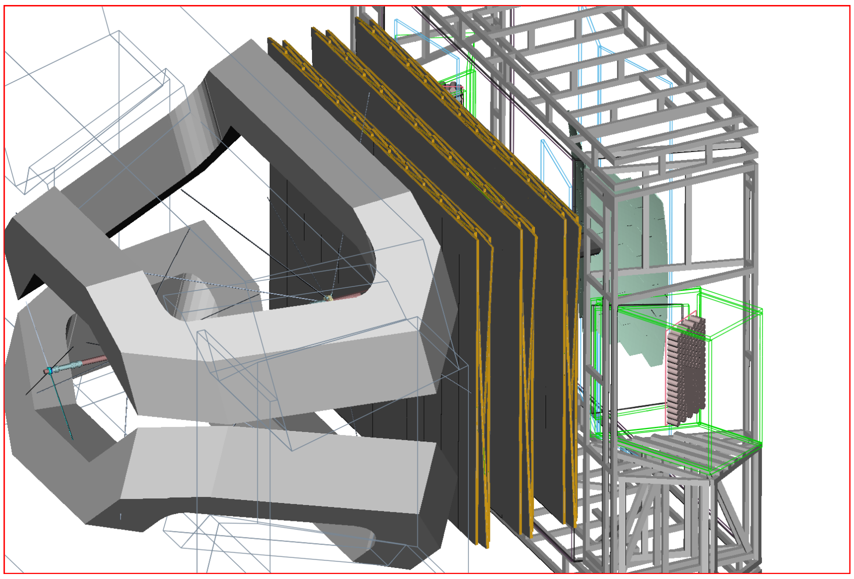


Figure 1.12. 3D model of the three stations of the SciFi tracker, as presented in [15].

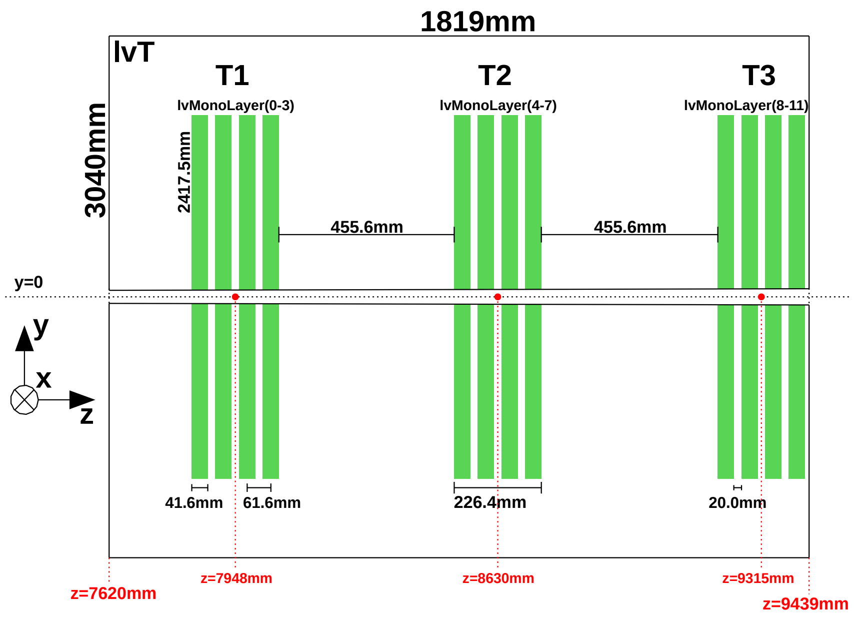


Figure 1.13. The layout of the 12 detection layers within the full tracker volume, as presented in [15].

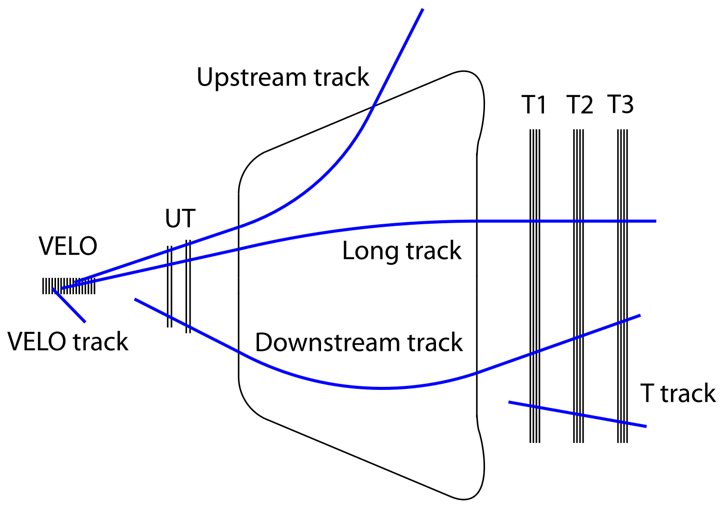


Figure 1.14. Reconstructed track type categorised by sub-detector, as shown in [15].

1.4.5 Track reconstruction and performance

The trajectory of the charged particles detected by the LHCb tracking system is reconstructed using specific algorithms. The *tracking* process in the LHCb is divided in two parts: pattern recognition and track fitting. The first part consists of combining individual measurements from various tracking into track candidates, in order to combine data from multiple sub-detectors multiple specialised algorithms are used and combined. The second part determines the optimal set of track parameters by performing a Kalman filter based fitting [17].

The tracks reconstructed in the LHCb detector can be categorised according to the sub-detectors in which they were reconstructed, as depicted in Figure 1.14. A full description of the tracks' categories follow:

- **long tracks:** these tracks are reconstructed in all the tracking sub-detectors. They have an excellent spatial resolution close the PV, thanks to characteristics of the VELO, and they have precise momentum information, thanks to the combined track slope before and after the magnet; therefore these tracks are the most important and the most used for physics analysis;

- **upstream tracks:** these tracks are reconstructed only in the VELO and in the UT. They are usually associated with low momentum particles, which are bent out of the SciFi geometrical acceptance area by the magnet. Thanks to the fringe magnet field present in the UT area it is possible to estimate the momentum with a resolution of $\frac{\sigma p}{p} \sim 15 \div 20\%$. These tracks are used to reconstruct low momentum particles and perform background-related studies for the RICH sub-detectors;

- **downstream tracks:** these tracks are reconstructed only by the UT and the SciFi. They are usually generated by long-lived particles decays which decay outside of the VELO, such as K_S^0 and Λ. These tracks are important for the reconstruction of the daughters of the aforementioned long-lived particles;

- **VELO tracks:** these tracks are reconstructed only by the VELO. They usually generated by large-angle or backward particles. Backwards tracks are essential for an unbiased reconstruction of the PVs;

- **T tracks:** these tracks are reconstructed only in the SciFi. They are not used in physics analysis, but they can be used for RICH2 studies.

1.5 Particle identification

Particle IDentification (PID) in the LHCb experiment is provided by three sub-detectors: Ring Imaging Cherenkov (RICH) system (RICH1 and RICH2), the calorimeter system (ECAL and HCAL) and the muon system. In this section, an overview of the system, as it will be during Run 3, will be given. A full description can be found in [18].

1.5.1 RICH1 & RICH2

Ring Imaging Cherenkov (RICH) detectors use the photons produced by the Cherenkov effect to identify the nature of particles. Every time a charged particle is traversing a dielectric medium with a velocity higher than the speed of light in the medium, it emits photons. For a medium with a refractive index n and a particle with a velocity relative to the speed of light $\beta = \frac{\beta}{c}$ the photons are emitted in a cone at a specific angle (θ_C):

$$\cos\theta_C = \frac{1}{n\beta} \tag{1.1}$$

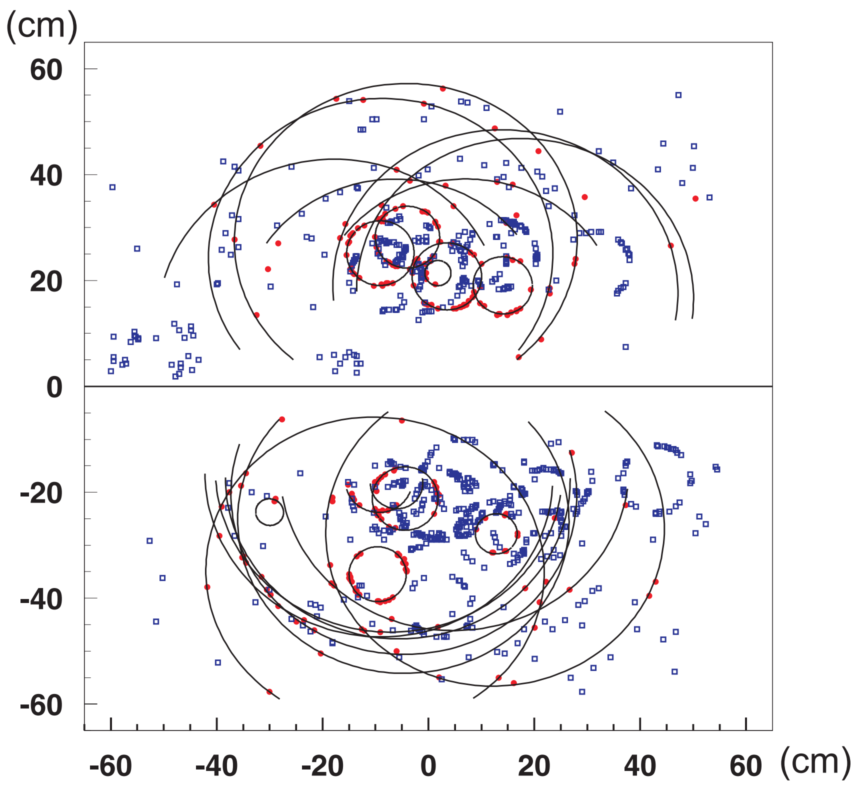
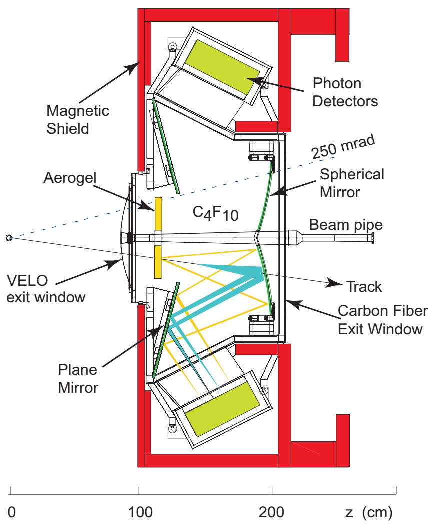


Figure 1.15. RICH1 displays an example (Run 1 data). The photon detectors detect the photons, and the rings are then reconstructed.

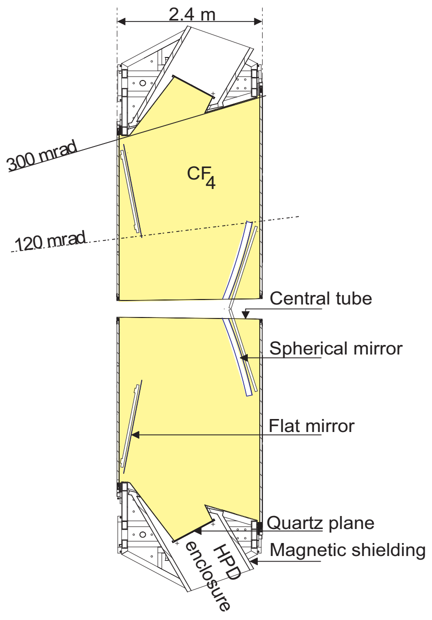
If $n\beta < 1$, i.e. the particle does not have a velocity higher than the speed of light in the medium, the effect is not observed; therefore different media can cover different β ranges, and therefore different momentum ranges.

The primary purpose of the RICH system is to identify and discriminate: π, K, e, μ and p, in different momentum ranges. The principle of operation is to use a medium in which the photons are produced, called radiator, collect the photons an arrange of mirrors and then detect the photons with high-efficiency photon detectors. The light from the different particles will form rings, as depicted in Figure 1.15, by measuring the radius of every ring (i.e. θ_C) it is possible to know the β of the particle. The combinations of the β and p of the particle are then used to determine its mass, i.e. its identity.

RICH1 is designed to operate in a momentum range of 1 GeV/c $< p <$ 60 GeV/c and it is located between the VELO and the UT. The radiator is C_4F_{10} with a refractive index of $n = 1.0014$. The light produced in the radiator is the reflected by the mirror system and detected by an array of Multi anode PhotoMultiplier



(a) RICH1



(b) RICH2

Figure 1.16. Schematic views of the two RICH sub-detectors. Side view of the RICH1 detector on the left. Top view of the RICH1 detector on the right.

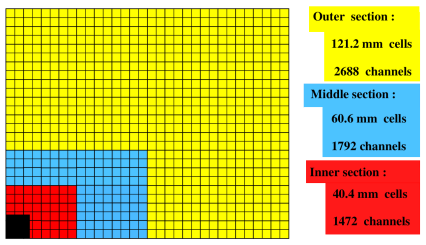
Tubes (MaPMTs), as shown in Figure 1.16a.

RICH2 is designed to operate in a momentum range of 15 GeV/c $< p <$ 100 GeV/c and it is located downstream of the SciFi. The radiator is CF_4 with a refractive index $n = 1.0005$, in order to quench scintillation 5% of CO_2 is added to the gas. The schematic representation of RICH2 is shown in Figure 1.16b. More details about the RICH system can be found in [18, 19].

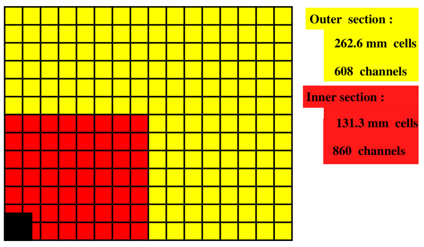
1.5.2 Calorimeter system

The calorimeter system achieves identification for photons, electrons and hadrons. By measuring their energy. The LHCb experiment, during Run 3, will use a system made of two sub-detectors an ECAL and a HCAL.

The ECAL is realised using a shashlik design, and it separated into independent modules. The shashlik design implements a sampling structure of alternating slices of absorber and scintillators; the layers are then penetrated perpendicularly by WaveLenght Shifter (WLS) fibres. The WLS fibres guide the scintillating light produced by the conversion of the energy deposited in the scintillating material to



(a) ECAL segmentation



(b) HCAL segmentation

Figure 1.17. Segmentation of the ECAL on left, and of the HCAL on right, as presented in [20].

a photon detector. Every module of the ECAL is made of alternating layers of lead and scintillating tyles, the lead layers are 2 mm thick while the scintillating ones are 4 mm thick. The total number of alternating layers is 66 per module, resulting in a total depth of 25 radiation lengths. All the modules are $120 \times 120\ \text{mm}^2$ but the read-out granularity changes in different regions of the detector, the segmentation schema of the ECAL is shown in Figure 1.17a, in particular the cell size is: $40 \times 40\ \text{mm}^2$ in the inner region, $60 \times 60\ \text{mm}^2$ in the middle region and $120 \times 120\,\text{mm}^2$ in the outer region.

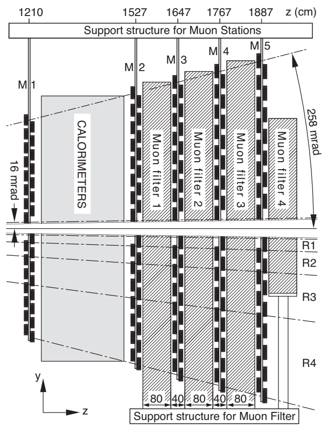
The HCAL has a sampling design made out of steel absorbers, and scintillating tiles, the orientation of the sampling structure is parallel to the beamline. The scintillating light is guided out of the detector by WLS fibres. The modular structure is similar to the one of the ECAL with an absorber to scintillator ratio of 5.5 : 1 and a total depth of 5.6 interaction lengths. The readout granularity changes in different regions of the detector, the segmentation schema of the HCAL is shown in Figure 1.17b, in particular the cell size is: $131 \times 131\ \text{mm}^2$ in the inner region and $263 \times 263\,\text{mm}^2$ in the outer region.

The energy resolution of the ECAL is $\sigma(E)/E = ((8 \div 10)/\sqrt{E} \bigoplus 0.9)\%$, while the resolution of the HCAL is $\sigma(E)/E = ((69 \pm 5)/\sqrt{E} \bigoplus (9 \pm 2)\%$.

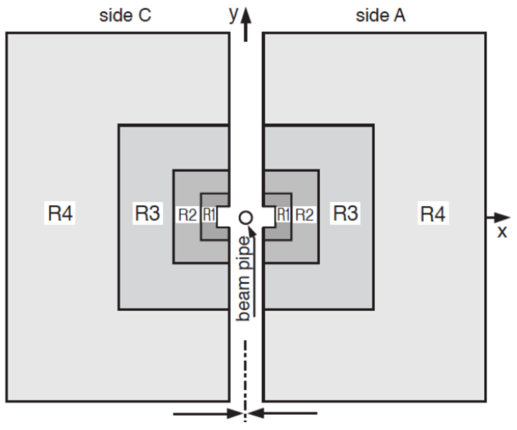
More information about the calorimeter system can be found in [18, 20]

1.5.3 Muon system

Muons are present in the final state of several b-hadron decay modes, e.g. $B^0_s \to J/\psi(\mu^+\mu^-)\phi$, $B^0_{(s)} \to J/\psi(\mu^+\mu^-)K^0_s$, $B^0_s \to \mu^+\mu^-$, etc. Moreover, high p_T muons are used by b tagging algorithms to identify the spectator b-hadron associated with the signal one. For the aforementioned reasons, excellent muon identification capa-



(**a**) Side view



(**b**) Station layout view

Figure 1.18. (Left) Side view of the LHCb muon system, the M1 station will be removed for Run 3. (Right) Station layout with the four regions R1-R4 indicated. As shown in [18].

bilities are essential to the physics program of the LHCb experiment.

The muon system during Run 3 will be composed of 4 muon stations (M2-M5), with an angular acceptance of ±300 mrad and ±300 mrad in the horizontal and vertical plane respectively. Thanks to the high penetration capability of muons, the stations can be located after the calorimeters, in order to reduce misidentification probability. Figure 1.18a shows a lateral view of the muon system, the M1 station located before the calorimeter system will be removed before the start of Run 3. The active layers are interleaved with 80 cm thick iron absorbers to select high momentum muons. The minimum momentum of a muon in order to pass all the M stations is 6 GeV/c.

Every station is composed of Multi-Wire Proportional Chambers (MWPCs) filled with a mixture of $Ar/CO_2/CF_4$ in with a 50:40:10 ratio. According to their geometrical position, the chambers' readout granularity varies, and in every station, four regions are defined (R1-R4). Figure 1.18b depicts the stations layout in the x/y plane, while the dimensions of the readout regions are listed in Table 1.1. The different regions are designed to achieve the same occupancy in all the detector.

From the physics performance point of view, the global muon identification efficiency is above 96%, and the hadrons misidentification rates are usually less than 1%. More information about the muon system can be found in [18, 21].

Region	M2 [mm^2]	M3 [mm^2]	M4 [mm^2]	M5 [mm^2]
R1	6.6×31	6.7×34	29×36	31×39
R2	12.5×63	13.5×67	58×73	62×77
R3	25×125	27×134	116×145	124×155
R4	50×250	54×268	231×290	248×310

Table 1.1. Sizes of the logical pads in the LHCb muon system. The dimensions are expressed as x-size $\times$ y-size.

Chapter 2

Interconnection networks

In this chapter an overview of different aspects of computer networks that are relevant for High Energy Physics (HEP) DAQ systems will be presented. A comprehensive deep analysis of all the main aspects related to understanding and designing networks, can be found in in [22, 23], which have been used as main reference for the content of this chapter.

In particular the concepts of: network topology, performance metric, flow control and routing will be explained in this chapter.

2.1 Terminology and basic concepts

An interconnection network is a programmable system made of a set of devices called *nodes* interconnected together via interconnection media called *channels*. The system purpose is to exchange *messages* among a subset of the nodes called *terminal-nodes* or simply *terminals*, the term *messages* refers to the data exchanged over the network.

2.1.1 Network classification

The nature of the interconnection medium and the way the nodes are interconnected can be used to classify the network itself. The proposed classification scheme is based on the one presented in [24], and this classification is not fully exhaustive; nevertheless, it provides all the context needed. Interconnection network can be divided in four major categories: *shared-medium networks*, *direct networks*, *indirect networks* and *hybrid networks*; a full description follows:

- **shared-medium networks:** every network that uses a shared medium to interconnect the nodes is in this category. The main advantage of this architecture is an easier deployment while the main drawback is that multiple devices cannot send data at the same time. Most modern wireless network technologies use a shared medium to communicate;

- **direct networks:** those networks use individual point-to-point links to interconnect the different terminal nodes without any other device on the communication path; because the communication medium is not shared multiple nodes can send information at the same time. Because the number of the point-to-point links available on a single terminal is limited the size of those networks is usually small, in order to increase the number of nodes interconnected, it is possible to add to the nodes the functionality of forwarding traffic towards other nodes. Those networks are usually used in dedicated High Performance Computing (HPC) clusters;

- **indirect networks:** those networks are similar to the direct ones because the nodes are connected via independent point-to-point links, but the way the terminal nodes are connected is different; terminals are not directly connected among themselves, but they are interconnected via another node called *router* or *switch*[1]. The addition of a network switch solves the problem of limited available point-to-point links on the terminals without adding any extra functionality to the terminals themselves. This flexibility comes at the price of the introduction of an extra class of devices. Ethernet-based Local Area Networks (LANs) are usually indirect i.e. switch based networks;

- **hybrid networks:** any network that uses a mixture of the solutions mentioned above is considered a hybrid network. For example, a modern usual campus or domestic network combines shared medium networks like WiFi with indirect networks like switched Gigabit Ethernet (GbE).

For the LHCb DAQ network the use of a shared network is not possible because of the large number of nodes that need to send information at the same time, while the use of a direct approach imposes strong constraints onto the nodes limiting the possibilities in terms of available Commercial Off-The-Shelf (COTS) hardware;

[1] A description of the OSI model, is not needed for this overview. Therefore there is no distinction between a layer2 device (switch) and a layer3 device (router), which can be threaded in the same way at this level.

therefore the LHCb Event Building (EB) network will be an indirect network based on COTS switches.

2.2 Network topology

How the various nodes are interconnected is called *topology*, and it can be expressed formally using the graph theory [23].

2.2.1 Channels and nodes

According to the previously introduced terminology a network is specified by a set of nodes N interconnected via a set of channels C. Because the messages are only originated and terminated in terminal nodes, it is useful to identify this subset of nodes N^* where $N^* \subseteq N$. In a direct network, every node is a terminal node; therefore, $N^* = N$. Every channel, $c = (x, y) \in C$, connects a pair of nodes $x, y \in N$ called source and destination node respectively, the source and destination nodes of a channel c are indicated with s_c and d_c respectively, if the interconnection between two nodes is bidirectional it will be represented by two channels $c_i = (x, y)$ and $c_o = (y, x)$. The network topology is represented by a directed graph in which the nodes are the N nodes of the network, and the edges are the channels in C.

Every channel c is characterised by two main characteristics: *bandwidth* and *latency*.

- **Bandwidth**: this property of a channel expresses the maximum amount of data that can be transported over the medium in a unit of time. Usually it is indicated in bits per second or Bytes per second. This quantity is limited by the physical interconnection used and should not be confused with the throughput which indicates the actual amount of data per unit of time that has been transferred over the link at a given time.

- **Latency**: this property of a channel expresses the amount of time needed to transfer a bit from s_c to d_c, for wired interconnections it can be expressed in terms of the propagation velocity v and the length of the interconnection l_c in this way $t_c = \frac{l_c}{v}$ where t_c is the latency. This quantity is the time needed to transfer one bit over the link and should not be confused with an end-to-end latency of a message from its source to its destination; the link latency does not take into account the latency introduced by the nodes and the extra delay introduced by other network traffic.

In a similar way it is possible to define three main characteristics for every node n: *degree*, *switching capacity* and *port-to-port latency*:

- **degree**: this property is simply the number channels connected to a node and it can be defined as: $\delta_n = |\{c \in C|((s_c = n) \vee (d_c = n))\}|$.
- **Switching capacity**: this property represents the total amount of data that can be exchanged by the node per unit of time. Usually it is indicated in b/s or B/s. It is important to note that a node of a given degree δ connected to channels of throughput T does not have necessarily a switching capacity of $S = \delta T$, in others words the switching capacity is not linked one-to-one with the degree and the throughput of the channels.
- **Port-to-port latency:** this property indicates the amount of time needs to transfer one bit from one input port to an output port. It should be noted that the port-to-port latency does not include the delay introduced by other traffic.

For every node n in the network, it is possible to define four sets of channels according to the connectivity with the node itself:

- **Input channels**: all the channels with n as destination
$$C_{In} = \{c \in C|d_c = n\}$$
- **Output channels**: all the channels with n as source
$$C_{On} = \{c \in C|s_c = n\}$$
- **Connected channels**: all the channels connected to n
$$C_n = C_{On} \cup C_{In}$$
- **Not connected channels**: all the channels not connected to n
$$U_n = \{C \setminus C_n\}$$

Similarly, it is possible to define two sets of nodes using the same guide principles:

- **Neighbours**: all the nodes with a direct connection with n

$$D_n = \{x \in N | (C_{Ox} \cap C_{In} \neq \emptyset) \vee (C_{Ix} \cap C_{On} \neq \emptyset)\}$$

- **Non neighbours**: all the nodes without a direct connection with n

$$I_n = N \setminus D_n$$

2.2.2 Cuts and Bisections

A *cut* of a network is a set of channels that divides the nodes into two disjoint sets, N_1 and N_2. Therefore every channel in the cut, $C(N_1, N_2)$ will have either its source in N_2 and its destination in N_1, or vice versa. The total bandwidth of $C(N_1, N_2)$ will be the sum of all the bandwidths of all the channels in the cut itself.

$$B(N_1, N_2) = \sum_{c \in C(N_1, N_2)} b_c \tag{2.1}$$

Bisection is a particular cut that divides the entire network nearly in half; moreover, the terminal nodes are also divided nearly in half as follows:

$$|N_2| \leq |N_1| \leq |N_2| + 1 \tag{2.2}$$

$$|N_2 \cup N| \leq |N_1 \cup N| \leq |N_2 \cup N| + 1 \tag{2.3}$$

Among all the possible bisections the one with the minimum channel count is called channel bisection, B_C. In a similar way it is possible to define the bisection bandwidth, B_B as the minimum bandwidth over all the bisections.

$$B_C = \min_{\text{bisections}} |C(N_1, N_2| \tag{2.4}$$

$$B_B = \min_{\text{bisections}} B(N_1, N_2) \tag{2.5}$$

The bisection bandwidth is a fundamental metric for representing the bandwidth characteristic of given network topology, and for estimating how it will perform with traffic from different algorithms as shown in [25] and [26].

2.2.3 Paths

A *path* or *route* is defined as an ordered set of connected channels.

$$P = \{c_1, c_2, \cdots, c_n\} \tag{2.6}$$

$$d_{c_i} = s_{c_{i+1}} \quad \forall\, i \in [0 : n-1] \tag{2.7}$$

The source and the destination of a path are the source node of the first channel and the destination node of the last channel respectively, using the aforementioned notation they can be identified as $s_P = s_{c_1}$ and $d_P = d_{c_n}$. If all the terminal nodes of the network have at least one path that connects them, then the network is *connected.* It must be noted that if all the components of a full network are considered, the routing function has to be take into account. The routing function is the algorithm that selects which paths can be used when connecting a given source with a given destination, therefore the set of path available on the network, for a given source destination pair P_r will verify this relation $P_r \subseteq P$. From this simple relation it is obvious that a non-connected topology cannot be connected at a routing function level, while a connected topology can be made non-connected by adopting a different routing function. A detailed description of routing function will be provided in 2.6.

If a path is connecting a pair of node with the minimal number of hops, then the path is called *minimal path.* The length of the minimal paths from node x to node y is indicated by $H(x, y)$. The diameter of a network $H_{\max}$ is the longest minimal path available on the network.

$$H_{\max} = \max_{x,y \in N^*} H(x, y) \tag{2.8}$$

It is possible to define an average minimum path between the terminal nodes $H_{\min}$ as follows:

$$H_{\min} = \frac{1}{|N^*|^2} \sum_{x,y \in N^*} H(x, y) \tag{2.9}$$

The minimum source-destination latency can be calculated by combining the latency number for all the individual nodes along the minimal path, assuming that the port-to-port latency t_p is the same for all the nodes and that the latency t_c is the same for all the channels:

$$t(x, y) = H(x, y) \cdot (t_c + t_p) \tag{2.10}$$

In order to estimate a global source-destination latency for a given network topology either $H_{\max}$ or $H_{\min}$ can be used, to get the maximum or the average latency respectively.

2.2.4 Symmetry and design considerations

A topology is *vertex-symmetric* if there is an automorphism that maps any node into another node. This property can simplify the routing function because the same local information can be used to route messages between nodes with the same relative position.

A network is *edge-symmetric* if there is an automorphism that maps any channel into another channel. This particular symmetry can mitigate load spikes across the different channels of the network because there is no reason to prefer one channel to another.

2.3 Performance measurements: throughput and latency

An interconnection network is a complex system with many different variables that can be optimised; moreover the optimal working point of the system depends heavily of the amount and the nature of the traffic that is injected into the network. It is therefore crucial to use significative and universally defined metrics, that can be used to evaluate the general performance level of the infrastructure. The two metrics that are usually taken into account are *latency* and *throughput*.

Latency is the amount of time spent by a message into the network, from the beginning of the transmission to the reception of the message at the destination terminal. This time interval is constituted by two major contributes one introduced by the network itself and one introduce by the network traffic. The first contribution can be calculated a priori by adding all the contributions introduced by nodes and channels that the message will pass through, and it provides a lover bound to the achievable latency. The second contribution is difficult to estimate because it depends on the evolution of the full traffic state of the system over time; therefore this quantity is measured either on real systems or on simulated ones. Latency is a per message quantity but it can be combined into a global quantity by either measuring the average latency over time or by populating an histogram with the latency distribution; Different network-based applications can be affected by latency in different ways, and can be penalised more by an high average latency or by outliers in the

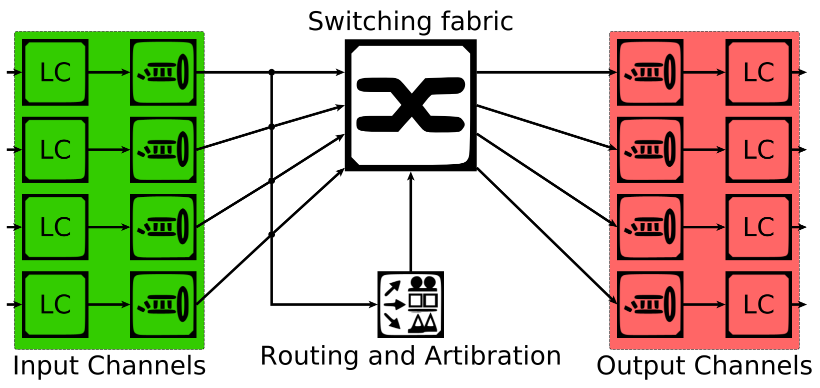


Figure 2.1. Internal structure of a generic router as the one described in [22]. From left to right the diagram shows: the input channels with the input buffers, the switching fabric with the switching and arbitration logic, the output buffers with the output channels.

latency distribution.

Throughput is the amount of information delivered by the network per unit of time. Similarly to latency, throughput is affected statically by the characteristics of the network and dynamically by the interaction with other messages. The static contribution is defined, on the path taken by a specific message, by the channel with the lowest bandwidth which provides an upper limit to the maximum achievable throughput. Because the final throughput depends on the full traffic state usually it is measured either on real systems or on simulated ones. According to its definition throughput is a global quantity and it represents the total amount of data delivered by the full system; nevertheless it is possible to define a subset of terminal nodes and measure the throughput only on this subset, this metric can be used to determine where the network is not delivering the expected performances.

2.4 Router model

Routers constitute the backbone of an indirect network, and they are responsible for the actual dispatching of the messages from the source node to the destination node. Figure 2.1 depicts the internal structure of a generic router model, providing a logical representation of the inner components that provide the basic functionality of a router; a detailed description of the internal components follows:

- **I/O Buffers:** *First In First Out (FIFO)* buffers are used for storing messages in transit. The model in Figure 2.1 has buffers for input and output channels, but alternative designs may have input or output buffers only.

- **Switching fabric:** This component constitutes the heart of the device itself, providing a programmable interconnection that is used to forward the messages from the input channel to the appropriate output one.
- **Routing and arbitration unit:** Those components constitute the brain of the router: implementing the routing algorithm, selecting the appropriate output channel for an incoming message and configuring the switching fabric accordingly. If the same output channel is requested by multiple messages at the same time, the arbitration unit must resolve the contention. The arbitration policy can vary for a simple round-robin to a complex priority algorithm.
- **Link Controller (LC):** This component is responsible for the data flow over the channel.

Incoming messages will enter the device from the input channels via a point-to-point link; then the information is stored into the input buffer until the Routing and arbitration logic configures the switching fabric, allowing the data to flow towards the right output channel. Before getting to the output channel, the data can be buffered a second time in an output buffer.

In a real-world scenario hardware resources are limited, in particular, both the switching capacity (i.e. how much data can be handled at the same time by the the switching fabric) and the amount space available in the buffers are critical resources, and they must be carefully managed. Every time the buffer of a specific channel is full, it cannot accept any new messages. The behaviour of the LC when there are no available resources will be discussed in section 2.5.

2.5 Flow Control

Flow control is responsible for managing and allocating the resources needed by the different messages flowing through the network. The most critical resources in network infrastructure are the I/O *buffers* and the *channels*. Flow control policies cannot select which path is used by the messages to go from their sources to their destinations, but they can enforce which message move forward and which message should wait at any given time, and control which resources are reserved to which message. The optimisation of the flow control strategy is crucial for achieving the required performances in terms of throughput and latency.

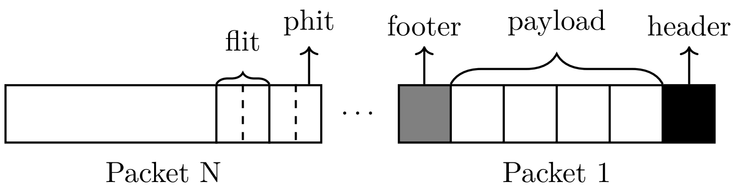


Figure 2.2. Fragmentation of a message into packets, flits and phits.

2.5.1 Data fragmentation

Data sent over the network has been up to now referred to as *messages*, and no size nor content limitations had been enforced. Handling efficiently multiple arbitrary size data streams crossing each other using real-world hardware is complex and inefficient. Therefore messages are usually split into *packets* of a predefined maximum size, as shown in Figure2.2. In order to have a deliverable packet it cannot only be made of user-defined data, but it has to contain additional information like: the source terminal address, the destination terminal address and additional data; this collection of values is generally called protocol information. The addressing information is used by the routing logic to select the appropriate path for the packet. Therefore it is usually stored into the first Bytes of it; this part of the packet is called *header*. In addition to the header most of the network protocols store some information in the last Bytes of the packet, this portion of a network packet is referred to as *footer*; most protocols implement error detection or error correction strategies, usually in the form of a Cyclic Redundancy Check (CRC). Because CRCs are computed using all the packet data, it is convenient to store them in the footer. A network packet is therefore composed of: an *header*, a portion of the message called *payload* and an optional *footer*; and it is the smallest structure that can be delivered from one terminal to another.

The maximum payload size M_{size} is a key parameter that needs to be optimised, because the size of the protocol information is fixed and it is contained in every packet it is useful to define the data fraction of a packet f_d and the effective bandwidth b_{eff}:

$$f_d(P_{\text{size}}) = \frac{P_{\text{size}}}{S_h + P_{\text{size}} + S_f} \tag{2.11}$$

$$b_{\text{eff}}(P_{\text{size}}) = f_d(P_{\text{size}}) \cdot b \tag{2.12}$$

Where: P_{size} is the payload size, S_h is the header size and S_f is the footer size. If a message exceeds the maximum payload size, this must be fragmented into multiple packets; consequently multiple headers and footers need to the transferred over the network. From (2.12), it is possible to derive the effective bandwidth for a multi-packet message:

$$f_d(T_{\text{size}}) = \frac{T_{\text{size}}}{(S_h + S_f) \cdot \left\lceil \frac{M_{\text{size}}}{T_{\text{size}}} \right\rceil + M_{\text{size}}} \tag{2.13}$$

$$b_{\text{eff}}(T_{\text{size}}) = f_d(T_{\text{size}}) \cdot b \tag{2.14}$$

Where T_{size} is the total message size and $\lceil a \rceil$ represent the ceil operation. In both cases, given a fixed S_h and S_f, the largest is the payload the more the packetization is efficient; moreover because the overhead is fixed in size, the closest the message size is to the maximum payload size the highest the effective bandwidth will be. On the other hand, increasing the packet size negatively affects latency and increases the hardware complexity of the nodes; therefore, in real systems, a tradeoff is made on the maximum payload size.

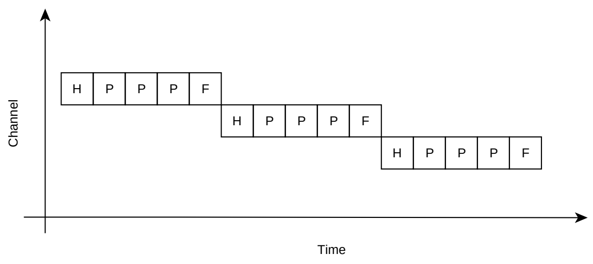
As depicted in Figure 2.2 a packet can be subsequently divided in *flits* and *phits*, which represents respectively *units of flow control* and *physical units* that can be transferred in one iteration over one channel.

2.5.2 Bufferless flow control

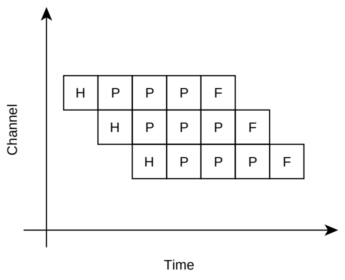
A bufferless flow control method uses a very simplistic approach and only manages the allocation of channels. It is remarkably essential to notice that the bufferless property is only applied to the flow control and not necessarily to the routers; the flow control system is unaware of the buffer status.

Every time a packet arrives into a router the flow control logic will try to reserve the output channel selected by the routing algorithm, if one or multiple packets already reserve the channel, an arbitration policy is used to balance the resources among the different packets. Because the flow control is not aware of the status of the destination node of the selected channel, it is possible to forward a packet towards the input port of a router that cannot take any more packets. For example, the aforementioned situation happens every time there is channel contention: if a packet cannot be forwarded, because the requested channel is busy, consequently it cannot free the input channel resources. Every time a router receives a packet and has no resources available, it cannot be processed, and the router discards it. The action of discarding packets is responsible for performance degradation impacting both latency and throughput, and it causes data retransmission. If there is no systematic resource contention, and therefore the fraction of dropped packets is under control, a bufferless flow control provides an easy and efficient solution to a complex problem; on the other hand in case of high resource contention the drop rate can quickly go out of control resulting in massive performance degradation.

2.5.3 Buffered flow control



(a) Store and Forward flow control



(b) Virtual-cut-through flow control

Figure 2.3. Time-space diagrams showing different flow control used to send a five flit long packet over three channels.

As previously mentioned flow control is responsible for managing which packet can progress towards its destination using the channels assigned by the routing function, and which packet should wait, if the algorithm that takes this decision is aware of the I/O buffer status, then the flow control is buffered. Because every packet can be split into several flits, flow control policies are divided into two major categories: packet-based flow control and flit-based flow control. The first category allocates resources to packets, and the latter allocates resources to individual flits.

The first and simpler buffered flow control strategy is called Store and Forward (SAF) and it is completely packet-centric. Every time a packet is received on the input buffer of a node the packet is stopped until the last flit of it is stored into the buffer. After the packet is completely stored into the node the flow control logic checks if the receive buffer of the output channel selected by the routing logic has enough capacity to store the packet, if the buffer is not full the packet is forwarded otherwise it is stopped until the resources are available. A SAF approach to packet flow control simplifies the hardware implementation of the routers because all the packets are treated as monolithic entities and no interruptions into the flow of a single packet need to be handled. The main disadvantage of SAF concerns latency, because the packet is forwarded only after it has been completely received the transmission delay cannot be pipelined from one channel to the next, as depicted in Figure 2.3a. The minimum latency of a single packet of n flits travelling over a distance d from its source to its destination can be calculated in this way:

$$t(n,d) = t_{\text{flit}} n d \tag{2.15}$$

where t_{flit} is the time needed to forward one flit from one channel to the next[2].

The second buffered flow control strategy is called Virtual-cut-through (VCT) and it tries to overcome the significant latency penalty introduce by a SAF approach. A VCT flow control implementation is still a packet-based solution but it introduces flit awareness into the process, making low latency transmissions possible. Every time the first flits of a packet are received into the input buffer of a node, as soon as the next channel has been selected by the routing unit, the flow control logic checks if the receive buffer of the output channel has enough capacity to store the full packet, if the buffer is not full the space is reserved and the packet is forwarded in a flit-by-flit way. By forwarding the packet one flit at a time the transmission latency can be significantly reduced by pipelining the transmission delays, as depicted in Figure 2.3b. The minimum latency of a single packet of n flits travelling over a distance d is:

$$t(n,d) = t_{\text{flit}} \cdot (n + d) \tag{2.16}$$

It is important to note that VCT is not a flit-based flow control strategy because the smallest unit that can be controlled – i.e. the smallest unit that can the either stopped or forwarded by the flow control – is a full packet and not an individual flit. Individual flits can be pipelined. However, once the forwarding of a packet starts, it cannot be halted, and all the buffer resources are allocated for entire packets and not to individual flits.

The third buffered flow control strategy proposed is called wormhole flow control, and it is flit-based. Wormhole uses the same flit pipelining technique used by VCT flow control, but it controls the flow of individual flits requesting only a one flit slot into the receiving buffer. Because the resource allocation is done on a per-flit basis the I/O buffers inside the various nodes can be smaller and the buffer space can be allocated in a more efficient and optimised way. The minimum latency can be calculated using (2.16) because allocating resources in flits units rather than in

[2]In this simplistic model all channel are supposed to have the same transmission latency.

packets units does not change the minimum latency. However, more efficient resource allocation can reduce the average latency when traffic is taken into account.

2.5.4 Buffer allocation and backpressure

All the aforementioned buffered flow control methods need to be aware of the buffer status to stop the data transfer and prevent overflow in the receiving channel; the term backpressure refers to this process. In order have an updated status of the buffer occupancy, the two LCs need to exchange data periodically. The buffer status information can be exchanged in several ways; two, in particular, will be examined: *Credit-based* flow control and *on/off* flow control.

Credit-based flow control

In Credit-based flow control, the sending router keeps a count of the amount of buffer available, in flit units, for every receiving channel. Every time the flow control logic reserves some space in the receiving buffer, the appropriate available space counter is decreased accordingly. When the counter goes below the size of a flit, for flit-based flow control, or the size of the next packet, for packet-based flow control, the buffer becomes unavailable, and the flow control pauses the transmission. In order to increment the counter value, the sender needs to know how many flits have been processed by the receiver, because resource contention can prevent a packet from progressing on its path towards its destination, the sender router cannot make any assumption about the number of flits processed. The only way for the sender to update the receiver buffer status is to receive an update on the buffer status called *Credit*. Because sending information between two nodes consumes available bandwidth on the channels, it is crucial to optimise the Credit send frequency f_C. If the sender does not receive Credits before the counter reaches zero the flow control has to stop the data transfer; on the other hand, sending too many Credits update will reduce the available bandwidth causing performance degradation. The Credit exchange optimisation is a time-sensitive problem; therefore, it is mandatory to take into account both the latency of the channel and the time needed to generate and process the Credit. In particular for a given: channel latency t_c, channel bandwidth in flit units b_f and flow control process time t_f the minimum amount of buffer needed to operate the system at full speed is:

$$F \geq (2t_c + t_f + \frac{1}{f_C})b_f \tag{2.17}$$

Therefore optimising a Credit-based flow control is a trade-off between buffer size and Credit frequency, increasing the buffer size requires more hardware resources while increasing the Credit frequency reduces the available channel bandwidth.

on/off flow control

In on/off flow control, the exchanged information between the sender and the receiver is reduced to the minimum, a 1-bit state. The receiver will send an off-signal to the sender every time the transfer should be paused, and it will send an on-signal when it is ready to receive more flits. The buffer condition in the receiving channel triggers the stop or start status. The flow control process is regulated via two thresholds T_{off} and T_{on}. Every time the amount of available flit slots in the receiving buffer drops below T_{off} an off-signal is sent and the transfer stops. When the free space in the buffer is greater than T_{on} an on signal is sent and the transfer is restarted. The two thresholds values are critical for the correct operation of the network.

If the T_{off} value is set too low then the transmission latency and process time of the status can lead to a buffer overflow condition and consequently a data loss, on the other hand; if the T_{on} is set too high then the receiver node can process all the flits in the buffer before receiving new flits from the sender resulting in performance degradation. Given the channel latency t_c, the channel bandwidth in flit units b_f and the flow control process time t_f maximum value for T_{off} is the following:

$$T_{\text{off}} \geq (2t_c + t_f)b_f \tag{2.18}$$

Similarly, it is possible to calculate the minimum amount of flits F needed into the receiving buffer needed to prevent performance degradation when the on signal is sent:

$$F \geq (2t_c + t_f)b_f + T_{\text{on}} \tag{2.19}$$

Relation (2.18) shows that the minimum buffer size needed for an on/off flow control system to work without data losses $(2t_c + t_f)b_f$. By adding the obvious condition $T_{\text{on}} \geq T_{\text{off}}$ to (2.19) it is possible to define the minimum amount of buffering needed to operate the system at full speed

$$F \geq (2t_c + t_f)b_f + T_{\text{on}} \geq (2t_c + t_f)b_f + T_{\text{off}} \geq 2(2t_c + t_f)b_f \tag{2.20}$$

Therefore, with an adequate amount of buffering, an on/off flow control can operate efficiently and with a limited amount of flow control information sent back from the receiver to the sender.

2.6 Routing

Routing is the process of selecting which path a given message should use from its source to its destination. The selection of the path is one of the critical factors that determine how much of the network potential can be expressed. A good routing algorithm will try to share the load on all the multiple channels in order to maximise the bandwidth available to the traffic. The performance of a routing algorithm can change drastically from one traffic pattern to another, making the optimisation of the routing function a crucial and challenging task.

The minimum requirement for a routing algorithm is to exchange messages across all the terminal nodes in a finite amount of time. In order to achieve this a routing function must be *connected*, and it has to prevent the arise of *deadlock* and *livelock* conditions; a routing function is connected if all the terminal nodes have at least one routing path that connects them, deadlock and livelock are static and dynamic conditions which prevent packets from being delivered and will be discussed in section 2.6.4 and 2.6.5 respectively.

2.6.1 Classification of Routing Algorithms

Different routing algorithm can be classified in many different ways according to different properties, a classification based on the way routing algorithms select paths from the set of all the possible paths R_{xy} from source node x to destination node y is given. For a fully exhaustive taxonomy of routing algorithms refer to [27].

The first category of algorithms is the one of *deterministic* routing algorithms. A routing function is deterministic if the selection of the path depends only on the source and the destination of the message. The routing decision can be taken as soon as the data enters the network or at every subsequent node, in the first case, the algorithm performs *source routing* in the latter *distributed routing.* Because a deterministic algorithm always uses the same path to connect any given pair of nodes it usually tends to provide sub-optimal load balancing across the various channels, on the other hand, the complexity required for the implementation of a deterministic algorithm is usually lower and, especially for distributed algorithms, the system provides better and easier scalability.

The second category is the *adaptive* algorithms one. An adaptive routing algorithm selects different paths according to the status of the network. Adaptive algorithms can be further divided into two sub-classes the *fully adaptive* algorithms and the *patrly adaptive* algorithms; the first ones take into account all the possible paths between the source and the destination nodes, while the ones in the second sub category use only a subset of R_{xy} for the routing decision. The selection of the optimal path through a given network can be a complex problem, especially for large networks, and the complexity of the routing function itself can increase very rapidly with the number of variables that are considered during the path selection process. The tuning and optimisation of adaptive algorithm require a lot of expertise and resources, but those algorithms can provide to the network a higher capacity of delivering messages and better resiliency against faults.

The third category is the *oblivious* routing algorithm one. Oblivious algorithms select their path among the ones in R_{xy}, but their decision is not on the network status-based, and therefore should not be confused with adaptive algorithms. Those algorithms require more resources than deterministic algorithms, but they provide some of the load balancing capabilities of the adaptive ones. Oblivious algorithms provide better scalability because they do not collect any information about the network status and can be easily implemented in a distributed way.

If at every hop, the message gets closer to its destination the algorithm is called *minimal*, on the other hand, a function that allows the message to be misrouted – i.e. moved further away from its destination – is called *nonminimal.* By selecting a path which is longer than the topological distance between source and destination, it is possible to provide better traffic spreading and lower end-to-end latency.

2.6.2 Formal definition of the routing function

A routing algorithm can be split into to two distinct parts: the routing relation R and the selection function ρ. R is responsible for returning a set of paths, and ρ selects which path. This division allows disentangling problems related to the algorithm itself from issues related to adaptivity. R can be defined in the following way:

$$R : N^* \times N^* \mapsto \mathcal{P}(P) \tag{2.21}$$

where $\mathcal{P}(A)$ indicates the power set of the set X, allowing the routing function to return multiple paths, the selection function will then select only one.

A generic routing algorithm routing a packet from x to y will select a path P in the following way $P = \rho(R(x, y))$. A deterministic routing algorithm only selects one path; therefore, R will return only the selected path, and ρ will be the identity function.

A routing subfuction R' can be defined in an analogous way by either reducing the domain of the routing relation or reducing the subset returned:

$$R' : S \times D \mapsto M \tag{2.22}$$

where $S \subseteq N^*$, $D \subseteq N^*$ and $M \subseteq R[N^* \times N^*]$. If $S = D = N^*$ the routing subfunction is connected, if the routing algorithm is deterministic the only connected subfunction is R itself.

The routing relation defined in (2.21) returns a set of full paths, and it is particularly convenient when defining global properties of the algorithm, but it represents only source routing algorithms because it calculates the entire path; therefore, it is convenient to define a distributed version of the routing relation which can be used to describe all the distributed algorithms:

$$R : N^* \times N^* \mapsto \mathcal{P}(C) \tag{2.23}$$

It is possible to map a source routing algorithm into a distributed one. Therefore the routing relation defined in (2.21) can be used to define global properties of distributed routing.

2.6.3 Virtual channels

In a buffered flow control the critical resources while sending packet are the receiving buffers, in order to optimise the channel usage and reduce the network congestion, it is possible to use virtual channels. Virtual channels consist of a set duplicated buffers multiplexed and demultiplexed into the same physical channel by the link control logic. From the routing algorithm and the flow control point of view, there is no difference between an extra physical channel and a virtual one because all the allocated resources can be reserved independently, and the link controller handles all the multiplexing onto the physical level. Adding a virtual channel is a cost-effective way of increasing the number of channels in the network without adding the extra cost and complexity of a new physical link.

All the virtual channels share the bandwidth of the physical link, and a predefined policy assigns the shares, any non-uniform policy can be used to implement Quality of Service (QoS) mechanisms providing different priority levels to different virtual channels.

The next section will present a different use of the virtual channels in the design of a deadlock-free routing algorithm.

2.6.4 Deadlock

Deadlock is a *configuration* of the network which contains packets that are indefinitely blocked by the flow control and therefore cannot be delivered. A *configuration* is a specific assignment of buffer resources to all the packets currently present in the network. The packets blocked in a deadlocked configuration are waiting for resources occupied by other blocked packets, resulting in network disruption.

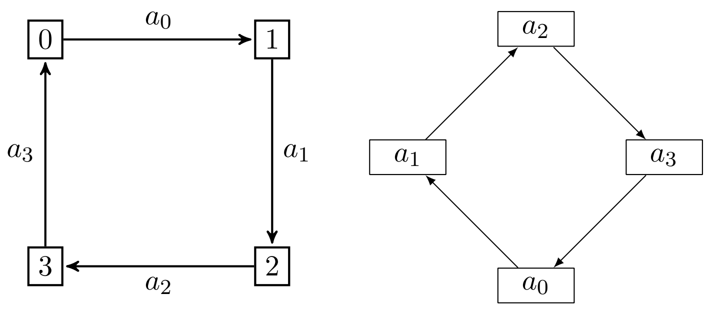
A deadlocked configuration is *canonical* if all the packets are blocked. Because any deadlocked configuration as a corresponding canonical one, only canonical configurations will be discussed. In order to obtain a canonical deadlock configuration, it is sufficient not to inject any traffic and wait for the configuration to become static, i.e. to wait for all the non-blocked packets to be delivered.

This paragraph will analyse two different approaches to avoid deadlock: deadlock avoidance and deadlock recovery. The first one consists of designing the routing algorithm to be deadlock-free on the specific network topology, i.e. regardless of the traffic no deadlocked configurations can be generated; the latter consists of detecting a deadlocked configuration and free resources to break the stall.

Deadlock avoidance

In order to prevent deadlocked configurations from happening, it is possible to use a deadlock-free routing function. Because this property must be granted regardless of the traffic injected into the network, it is not trivial to verify if a routing algorithm is deadlock-free. In order solve this problem it is possible to use some graph theory and an interesting theorem.

In order to provide a formal description of the problem it is fundamental to introduce the concept of *channel dependency* and the *channel dependency graph*: channel i depends on channel j if the routing algorithm can forward a packet holding resources on channel i to channel j; the channel dependency graph is a directed graph which has all the unidirectional channels of network as vertices, and the edges represent the dependencies between the channels. The following theorem provides a necessary and sufficient condition for establishing if a routing algorithm is deadlock-free [28].



(a) Network topology **(b)** Channel dependency graph

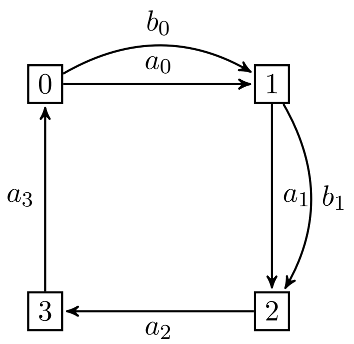
Figure 2.4. A four node circular network with its channel dependency graph.

Theorem 1 *A connected routing function R for an interconnection network I is deadlock-free if and only if there exists a routing subfunction R_1 that is connected and has no cycles in its channel dependency graph.*

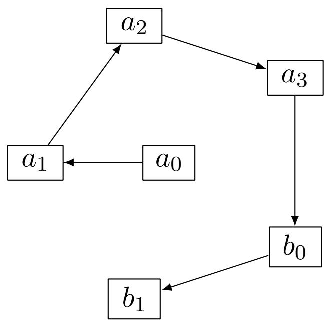
Theorem 1 is applicable to VCT and wormhole flow control, and it can be used to generate deadlock-free routing algorithms as in the following example: the network topology selected is a circular direct network of four nodes connected by unidirectional channels, as the one depicted in Figure 2.4a. Because of the simplicity of this topology, the only connected, deterministic and minimal routing function is the one that forwards packets through the only available channel until they arrive at their destination. The dependency graph for this algorithm is shown in Figure 2.4b and it is not acyclic. Therefore the routing relation is not deadlock-free. In this trivial example, it is straightforward to find a deadlocked configuration: any configuration with all the buffers full and no packets arrived at their destination is deadlocked.

In order to make this network deadlock-free both the topology and the routing function must be modified, the new topology is depicted in 2.5a and a distributed version of a deadlock-free routing algorithm is shown in Listing 2.1.

This new routing function uses the extra channels for packets with a source index smaller than the current node; this modification changes the channel dependency graph into the one depicted in Figure 2.5b which is acyclic and therefore this new network is deadlock-free. Because adding real channels to prevent deadlock from happening is costly, it is common to use a virtual channel instead. The Independence between virtual channel from the routing and flow control point of view makes them a perfect and cost-effective candidate for breaking cyclic channel dependency graphs, and designing deadlock-free algorithms.



(a) Network topology



(b) Channel dependency graph

Figure 2.5. Modified deadlock-free circular network and channel dependency graph of the proposed routing algorithm.

Listing 2.1. Pseudo-code implementation of a deadlock-free routing algorithm for the topology shown in Figure2.5a.

```
if ( destination == current )
  packet_delivery;
else if ( source < current )
  reserve(a_i+1);
else
  reserve(b_i+1);
```

Deadlock recovery

In order to prevent deadlocked configuration from persisting it is possible to utilise deadlock recovery techniques and break the deadlock state. Because a deadlocked configuration has a circular pattern in the resource dependency, it is sufficient to free resources to one of the packets in the critical resource loop to restore normal operation. From this first superficial analysis putting in place a deadlock recovery strategy seems an easy task, it is sufficient to drop a single packet from the main circular dependency; in reality this task is much more complex and it requires several non trivial elements: detect a deadlocked configuration, identify the main dependency loop and coordinates the nodes to drop the required packet.

Because deadlock is global status of the network it is not possible to establish if a packet cannot be forwarded because of network congestion or deadlock; therefore a fully deterministic deadlock detection is not feasible, on large network, and costly on small ones. Nevertheless, it is possible to use an heuristic timeout-based system to identify deadlock occurrence. As per any other heuristic implementation there will be a certain probability of misidentifying heavy traffic with deadlock, resulting in an higher number of packets dropped and transmitted. Increasing the timeout value will result in a lover misidentification probability but it will increase the reaction time to a real deadlocked configuration, therefore this value must be carefully tuned.

The next important step in deadlock recovery is selecting which packet should be dropped, an heuristic time based solution can be followed by dropping first packets that have been blocked for longer, because those packets have a greater probability of being part of the main dependency loop.

In all the aforementioned solutions the selection of appropriate timeout values is critical for the correct operation of the system. If the threshold is too aggressive then the network will drop a considerable number of packets resulting in increased latency and numerous retransmissions; on the other hand a too conservative value will lead to a slow reaction time to a real deadlocked configuration.

Another aspect that has not yet been introduced is the deadlock frequency, extensively studied in [29], which can be expressed in terms of normalised deadlocks – i.e. the ratio between the average number of deadlocks over the number number of delivered packets. This number is heavily influenced by: the network topology, the routing algorithm and the injected traffic, and it complex to give an a priori prediction. In any case if the reaction time of the deadlock recovery system is not adequately fast it is possible to have a new deadlock loop forming before the previous one has been broken, resulting in a constant pile-up of deadlocked configurations which will cause a massive performance degradation.

2.6.5 Livelock

Livelock is misbehaviour of the routing function which forwards a packet in an indefinite loop without delivering it to its destination terminal. Livelock conditions impact the network in two ways: the packets affected are never delivered; therefore, the basic functionality of delivering messages between terminal nodes is broken. Moreover, the affected packets are continuously misrouted, wasting static and dynamic resources that cannot be used by other packets.

To be affected by livelock, a routing algorithm has to be non-minimal, because minimal algorithms ensure packet delivery in a number of hops which equal to the distance between the source terminal and the destination terminal. Any non-minimal algorithm should include protection against livelock occurrence, a simple and effective way to avoid livelock is to limit the number of times that a packet can be misrouted granting the delivery in a finite number of hops. Misrouting can be limited by either designing the algorithm in a way that prevents infinite misrouting or by adding a misroute counter to the protocol information of the packet. The first option is more suitable for centralised or source-based algorithms; the latter is more suitable for distributed algorithms. The simplicity of adding a counter to the protocol information comes with the price of adding more information into the protocol section of the packet, because the effective bandwidth b_{eff} defined in (2.12) is negatively affected by the size of the protocol information, adding this counter is going to produce either lower effective bandwidth or larger packets and consequently bigger buffers.

A fully adaptive algorithm, without any misrouting restriction, can be statistically livelock-free if it uses the approach described in [30] for the so-called deflection routing algorithm. This algorithm uses the minimal path with a higher priority than the non-minimal ones, because the probability of having all the minimal paths busy decreases with the number of tries, the livelock probability goes to zero after a long enough period. This method is weaker than a deterministic one, but it is the only known one that is proven to be both deadlock and livelock-free without limiting the number of misrouted steps.

Chapter 3

Fast networks for the next generation LHCb Data Acquisition

To reach the full 40 MHz event-building rate the event building (EB) network must be capable of forwarding an aggregated peak throughput of 40 Tb/s between ~ 500 end-points, therefore resulting in a single link throughput of ~ 80 Gb/s. At the time of writing, there are multiple COTS solutions that provide 100 Gb/s link bandwidth, like InfiniBand (IB) Enhanced Data Rate (EDR) or 100 Gigabit Ethernet (100GbE). Given those requirements and the technological constraints the EB network has to connect ~ 500 end-point with an average link utilisation of ~ 80 %, moreover, the EB traffic tends to generate heavily unbalanced instantaneous link load, resulting therefore in network congestion and potential performance degradation.

In this chapter, an accurate description of the upgraded LHCb EB system is given. The first section provides a detailed physical description of the system, a high-level description of the EB process and the requirements and constraints imposed on the network infrastructure. The second section describes in detail the two different EB traffic emulators developed by the LHCb online team to evaluate the performance of different systems. The fourth section describes in detail the simulation model used to simulate the EB system including low-level tuning of the model to match, as much as possible, the behaviour of the actual network hardware and EB traffic pattern.

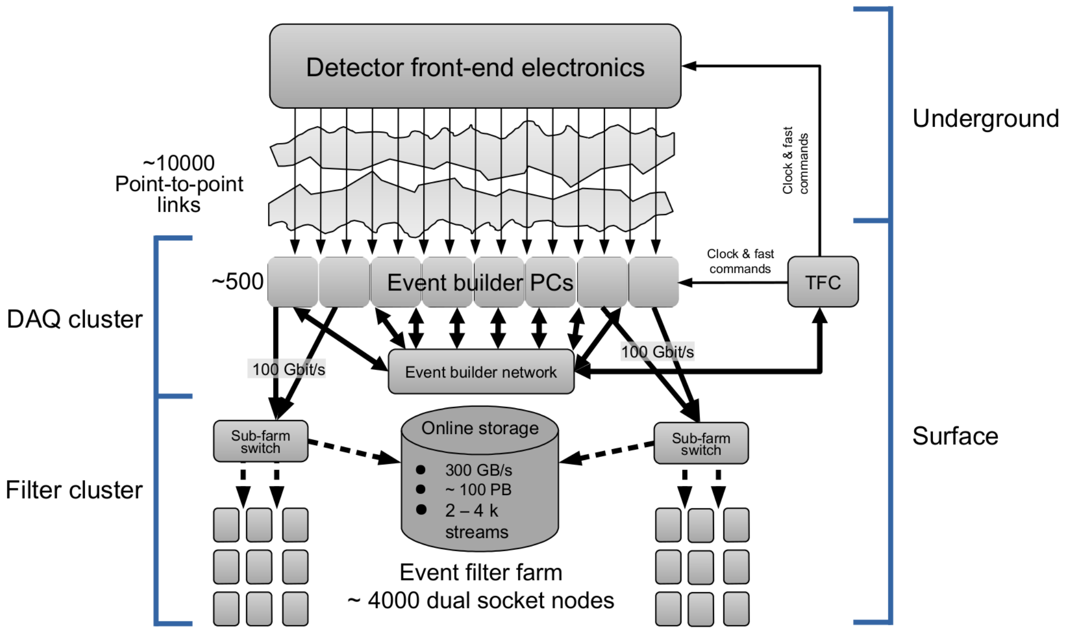


Figure 3.1. The architecture of the upgraded LHCb readout system.

3.1 The LHCb Event Building

3.1.1 High level description of the Event Building process

As described in [31] the data flow of the LHCb detector is divided across ~ 10000 optical fiber links terminated onto ~ 500 custom FPGA-based readout readout cards.

To collect data from all the different sub-detector channels, the EB process must take place, and all the data fragments of a single event have to be collected and assembled in the same place. Therefore all the different event fragments must be sent over some interconnection network in an all-to-one communication. Three logical units can be defined to provide a high-level description of this task: Builder Units, Readout Units and the Event Manager; a functional description follows:

- **Readout Unit (RU)** collects the fragments from the Peripheral Component Interconnect Express (PCIe)-based DAQ board and sends them to the Builder Units (BUs)

- **Builder Unit (BU)** receives and aggregates the fragments into full events

- **Event Manager (EM)** assigns which event is built on which BU

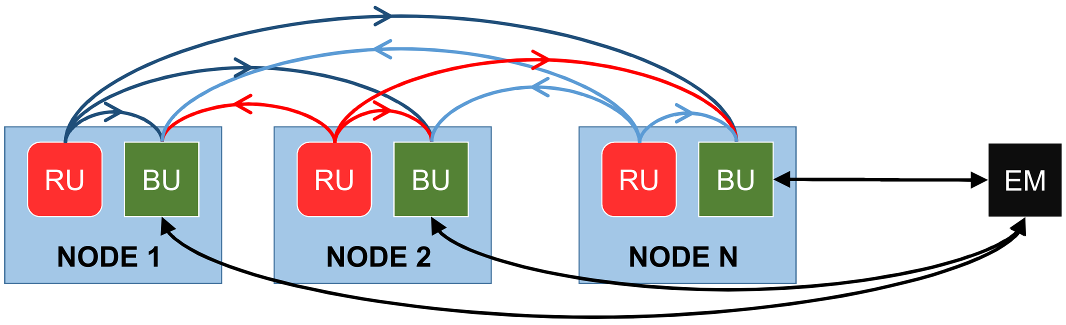


Figure 3.2. The logic representation of the LHCb EB process. The different arrows represent the multiple fragments gathered by the BU while the black ones the control messages to and from the EM.

Those functional units can be used to describe most of the modern HEP DAQ systems, by either changing the communication pattern or the physical layout of the units within the system. Figure 3.2 depicts the logical interconnection and the physical layout of the logical units in the LHCb DAQ. All the EB nodes (i.e. the servers that are hosting the required hardware and running the EB software) have two units: one RU and one BU; because data always flows from the RUs to the BUs, a "folded" structure can profit from the full-duplex nature of COTS network technologies; and it leads to a reduction by a factor two in the number of physical machines used. In the common communication schema, the traffic pattern of a folded event builder can be mapped to an All-To-All Personalised Communication (ATAPC) with different data size for every fragment.

The events produced by the upgraded LHCb detector will have a nominal event size of 100 kB; therefore, the amount of data stored in every fragment will be $\mathcal{O}(100B)$. Modern interconnection technologies are not designed and optimised to transfer messages of a few hundred Bytes. To make the transfer more efficient event fragments, and consequently, events, are grouped into Multi Fragment Packets (MFPs) and Multi Event Packets (MEPs). During Run 3 of the LHC, the event packing factor of the LHCb experiment will $\mathcal{O}(10^3)$ events per MEP, generating larger MFPs that can be efficiently transferred over the network.

In order to make the notation more natural to read, and because grouping fragments from multiple events does not change the behaviour of the event building network, the terms event and event fragment will be used instead of the acronyms MEP and MFP when referring to the event building system.

3.1.2 The Event Building network

The interconnection network constitutes the heart of the EB system providing the infrastructure for the intensive I/O activity performed by the aforementioned EB logic units. By looking at the nature of the generated traffic, it is clear how it can potentially be extremely challenging for the underlying network infrastructure. Because data from multiple sources must be aggregated into a single destination, this communication pattern naturally generates instantaneous overloading of specific links, resulting in network congestion. If the buffers associated with the overloaded channels do not have enough capacity to absorb the traffic peak; the link flow control has to intervene by either reducing the effective injected throughput, or dropping packets. The choice is led according to the nature of the flow control implemented by the selected network technology.

Different strategies can be put in place to avoid performance degradation both in the EB software and in the COTS network technology selected, the first strategy will be discussed in section 3.2 and the latter in section 3.3.

3.2 Event Building traffic generators

Because the full EB system needs to be designed and optimised before the commissioning of the detector, it is crucial to develop a traffic generator that replicates the expected output from the detector. The LHCb online team has been working on different traffic generators since 2014, producing different benchmark applications and exploring different strategies to solve the problem of EB.

3.2.1 Linear shifting scheduling

In an EB scenario, all the data from one event has to be gathered in one terminal node of the network. In a network with N data sources, all trying to send data with a datarate r, this generates throughput towards the destination terminal of Nr. In the foreseen scenario for Run-3, the LHCb experiment, the data rate will be 80% of the channel bandwidth, and the number of nodes will be 500 resulting in a massive channel oversaturation.

Because there are multiple data destinations, assuming an even distribution of the events across all the BUs, every target node will receive a new event every N events. Therefore the average input data rate for every BU will be r. The fact that the sustained requested throughput does not exceed the terminal available input bandwidth implies that it is possible to redistribute the traffic in a way that generates no congestion.

Because of the ATAPC nature of the EB traffic it is possible to use take advantage of the extensive work performed on optimising an ATAPC by the HPC community [32, 33, 34, 35, 36, 37, 38]. In order to schedule the communication, the full all-to-all exchange must be considered, therefore in a balanced event builder with N BUs and RUs the scheduling will consider N events in parallel. The rationale behind reducing the network congestion introduced by an ATAPC is to divide the full exchange into phases and schedule conflict-free communication between node pairs. This technique is referred to as shifting, and it is widely used with different shift patterns. The most straightforward pattern that can be used to produce conflict-free scheduling is the linear shifting, which can be summarised as follows:

- In a folded event builder architecture every node needs to receive $N - 1$ fragments from the other RUs via the network plus one extra additional fragment from the RU located on the same node, therefore from the networking point of view the exchange can be divided into $N - 1$ phases;
- In any phase, every RU sends data to one BU, and every BU receives data from one RU;
- During phase n RU x sends to BU $(n + x)\%N$[1];
- When a previously agreed condition is met all the units synchronously switch from phase n to phase $n + 1$.

The use of the aforementioned scheduling technique provides an easy way to solve the destination conflict issue generated by the event building traffic. On the other hand, it requires the processing of multiple events in parallel; some degree of synchronisation among the nodes; and it does not guarantee end-to-end congestion-free network traffic. In order to have the certainty of no resource contention, the underlying network has to be *nonblocking.*

A network is non-blocking if it is always possible to establish a free path between a pair of unused terminal nodes regardless of the other traffic. Because all the traffic generated by a linear shifter is occurring between exclusively used pairs of ones, a non-blocking network makes the full exchange conflict-free.

Because the traffic generated by a linear shifter is known *a priori* it is possible to relax the non-blocking condition and to request the network to be *rearrangeably*

[1]The % symbol indicates the modulo operation

nonblocking, i.e. it is possible to establish a free path between any pair of unused terminals regardless of the traffic by rearranging existing data flows. This condition is equivalent to this one: it is always possible to have a specific set of paths which allows connecting any number of terminal nodes pairs. Because only a specific set of paths can be used, the routing algorithm must select the right combination of channels for every phase.

Processing multiple events in parallel have an impact on the amount of buffering needed in the event builder PCs. Every RU/BU needs to store 500 events to complete all the phases of the linear shifting while receiving new data from the detector. The last problem is the synchronisation between the nodes which can be solved with different approaches [39]: full synchronisation, partial synchronisation and no synchronisation.

- **Full synchronisation:** this method uses stable synchronisation between all the nodes; this approach is the only one which is granted to provide a fully deterministic conflict-free scheduling. This strong alignment between the nodes can be achieved by either sending synchronisation messages over the network or by the occurrence of predefined external condition, for example, a time-based phase shifting;
- **Partial synchronisation:** this method ensures synchronisation only between a subset of the nodes reducing the overhead of a full synchronisation. The rationale behind a weaker phase alignment is that the network infrastructure can handle a small degree of network congestion without a significant performance drop;
- **No synchronisation:** this method removes the overhead thoroughly introduce by nodes synchronisation by removing the synchronisation itself. The idea is that, if all the nodes have to send the same amount of data, they will keep the phase alignment by themselves because the amount of time needed to complete one phase is uniform. This approach is less resilient and can lead to phase skipping in case of unexpected issues.

3.2.2 DAQPIPE

DAQ Protocol-Independent Performance Evaluator (DAQPIPE) [40, 41, 42] is a benchmark application that generates event building-like traffic; this software has been developed by the LHCb online team to test the response of different network technologies and scheduling techniques to the traffic generated by the LHCb DAQ system.

The core of this project provides a C++ based implementation of all the logic units needed to replicate the full data-flow of the event building system, including a dummy data source that allows DAQPIPE to be run without the need of using the real readout hardware. The software is designed with a modular structure that allows an effective decoupling of the network layer from the rest of the code, providing an ideal framework for testing different communication libraries like OpenMPI, LIBFABRIC, VERBS and PSM2.

DAQPIPE can be used either in a PUSH – i.e. the RUs push data towards the BUs – or PULL – i.e. the BUs request data to the RUs – schema and it supports different traffic shaping strategies to reduce network congestion, including two different linear shifters and a random one. The linear shifting-like traffic pattern can be generated either with or without a stable phase synchronisation. The first scenario uses in-band message signalling to implement a barrier synchronisation which keeps all the nodes phase aligned. The second scenario does not implement any phase synchronisation technique within one event; nevertheless, the EM enforces strong alignment on a per-event basis to prevent an excessive phase drifting over time. The random shifter is useful in case of a network that does not provide non-blocking behaviour, and it is not intended to be used in the real system. The rationale behind the development of this particular scheduler was to quantify the performance loss introduced by a sub-optimal network setup. The default scheduling used is the unsynchronised linear shifter.

The lack of perfect traffic shaping leads to local short-living link contention by different messages, which can potentially severely degrade the throughput delivered by the full system. In particular, if the fragment processing is implemented in a fully sequential way, the EM will not assign the next event until the slowest fragment is delivered. In order to mitigate this effect, DAQPIPE introduces some parallelism both in the number of fragments of the same events in flight towards a specific builder unit and in the number of events processed in parallel by one builder unit. Within the framework, the parallelism is configurable via two parameters named *credits* and *parallel sends* with the following effect:

- **credits:** number of events processed in parallel by the BU;
- **parallel sends:** number of fragments of the same event in flight to the same BU from the different RUs.

Every BU announces to the EM the selected number of credits; the EM will then consume those credits by assigning specific events to the BUs, every time an event is fully received by a BU a new credit will be announced to the EM. Every credit is processed asynchronously from the others, following its linear shifting-like scheduling. Processing multiple credits in parallel reduces the amount of network starvation introduced by the communication delay between the event manager and builder units, and waiting for the delivery of a specific fragment. The number of parallel sends defines a sliding window mechanism on top of the barrel shifting scheduling which has the purpose of absorbing extra communication delays between the BUs and RUs because every credit is processed independently from the others the number of parallels sends is per credit.

The parallelism, as mentioned above, has the side effect of inherently generating some network congestion by breaking the ideal linear shifting traffic pattern. This behaviour is acceptable because the application does not enforce, and it does not rely on perfect scheduling. Nevertheless, the behaviour of the network in the presence of channel oversaturation depends heavily on the specific implementation of the flow control, in particular, bufferless flow control can drop a significant number of packets resulting in a consequently high number of retransmissions. For this reason, the traffic generated by DAQPIPE is more suitable for a network with buffered flow control.

The amount of parallelism introduced, other than affecting positively or negatively the aggregate throughput of the system, changes the minimum amount of buffering needed in the readout units and the builder units. The minimum amount of buffering needed can be easily calculated via the following model:

$$B(c, f_{\text{size}}) = mNcf_{\text{size}} \tag{3.1}$$

Where c is the number of credits N is the number of RUs/BUs and f_{size} is the size of a single fragment, the factor m is the safety margin and it is a natural number greater than 1. Because the data from the experiment will continue flowing while the event building process is taking place, the extra safety margin is needed in order to store the next batch of data while the current one is being processed.

Selecting the optimal value for credits and parallel sends for a given network infrastructure *a priori* is not a simple operation. It requires an accurate model of the interaction between the traffic generated by the application and the flow control system of the network, the results achieved by both running the application on smaller-scale clusters and running a discrete event simulation of the system will be presented in section 3.4.6.

3.2.3 a2a

All-to-All (a2a) is a micro-benchmark recently developed to test how a fully synchronous linear shifter approach will perform in an event building scenario; therefore a2a faces the problem from an opposite point of view compared to DAQPIPE. At the time of writing this benchmark enforces robust phase synchronisation via a time-based method, this strategy has been initially preferred to in-band signalling to reduce the communication latency penalty.

The transmission pattern generated by a2a is shaped by using two parameters: the *transmission window* and the *idle window*; the first parameter is the amount of time in which every RU is allowed to send fragments to the target BU selected according to the linear shifting scheduling; the second one is buffer time period in which no RUs is supposed to send. The purpose of the idle window is to absorb any desynchronisation that may occur between the various nodes preventing phase shifting because the benchmark relies on a conflict-free communication it fundamental to prevent node misalignment. The maximum achievable data-rate per RU using a2a can be easily calculated:

$$d_{\text{a2a}} = \overline{b_{\text{eff}}} \frac{T_{\text{send}}}{T_{\text{send}} + T_{\text{idle}}} \tag{3.2}$$

Where: $\overline{b_{\text{eff}}}$ represent the effective bandwidth, as define in (2.12), calculated for the average message size used; T_{idle} represent the idle period and T_{send} the transmission window period. In order to maximise d_{a2a} it is important to properly balance the two time windows, in particular if $T_{\text{send}} \gg T_{\text{idle}}$ then the performance penalty introduced by the idle period is negligible.

The calculation of the optimal values for the transmission and idle windows is a complex problem and strongly depends on the underlying hardware and software infrastructure, by introducing some approximations it is possible to understand how changing those parameters interacts with the rest of the DAQ system. To a first approximation T_{idle} and T_{send} can be considered independent, under this condition the value of T_{idle} will be calculated upon various considerations based on the final implementation of the system, such as the amount of clock jitter between the nodes; the Operating System (OS) kernel scheduler latency and granularity; the end-to-end network latency; the standard deviation of the software stack processing time. Once the value of T_{idle} is set the value of T_{send} can be selected in order to meet the desired throughput requirements.

The data-rate per RU in case there is an imperfection in the scheduling – i.e. two or more RUs are sending to the same BU because the synchronisation mechanism failed – is given by:

$$d_{\text{a2a}} = \overline{b_{\text{eff}}} \left(\frac{T_{\text{send}} - T_{\text{conf}}}{T_{\text{send}} + T_{\text{idle}}} + \frac{1}{2} \cdot \frac{T_{\text{conf}}}{T_{\text{send}} + T_{\text{idle}}} \right) \tag{3.3}$$

Where T_{conf} is the amount of time the scheduling conflict lasts – i.e. the amount of time the misbehaving node will continue sending to the wrong destination after the idle period. In presence of network conflicts having $T_{\text{send}} \gg T_{\text{conf}}$ helps reducing the performance degradation introduced by a non-ideal scheduling.

In the two models proposed in (3.2) and (3.3) having a large value for T_{send} provides higher data-rates, unfortunately this performance improvement comes with an implementation cost drawback: increasing the transmission window requires the sender and the receiver to exchange a more significant amount of data in every slice. In an event building scenario increasing the amount of data exchanged results in more events to be processed in parallel and therefore requires buffer space in both the RUs and the BUs, the amount of buffer needed as a function of T_{send} is:

$$B(T_{\text{send}}) = mN\overline{b_{\text{eff}}}T_{\text{send}} \tag{3.4}$$

Where N is the number of BUs, and under the assumption that the average amount of data injected from the detector into a RU is less than the average amount of data that the network can ingest, if the last assumption is not verified then the full system is overloaded and should be redesigned regardless of the buffer size. The factor m represent a safety margin with the same purpose as the one introduced in (3.1).

In order to tune the two-time windows needed by a2a accurate measurements of the possible jitter introduced by the system must be taken, and then a tradeoff between cost and performance has to be made.

3.3 Network implementation

In this section, a detailed description concerning all specific implementation details of the network will be given.

3.3.1 Network technology

The event building network of the LHCb experiment will be based on COTS hardware capable of delivering a channel bandwidth of 100 Gb/s; moreover, in order to reduce the amount of CPU and memory bandwidth used by the event building application, the selected network technology has to support Remote Direct Memory Access (RDMA) transfer.

Currently, several COTS network technologies can fulfil the requirements needed by the LHCb event building system, in particular, three of them will be presented: InfiniBand, OmniPath and RDMA over Converged Ethernet (RoCE).

InfiniBand

InfiniBand is a network communication standard designed for HPC applications, and it provides high throughput and low latency connectivity. The underlying fabric takes advantage of credit-based buffered flow control which prevents packet retransmission when congestion occurs. The standard is defined and promoted by the InfiniBand Trade Association (IBTA), a multi-vendor consortium established in 1999. Since then multiple companies have joined and left the IBTA, but the development of the InfiniBand technology never stop producing the two most recent variants of it EDR and High Data Rate (HDR), providing a channel bandwidth 100 Gb/s and 200 Gb/s respectively.

The technology is based on RDMA, and the Open Fabric Alliance (OFA) developed a standardised Linux-based InfiniBand software stack. At the time of writing the dominant vendor of InfiniBand-based solutions is Mellanox Technologies.

OmniPath

OmniPath is a network technology, developed by Intel Corporation, and it targets low latency and high throughput HPC applications. Similarly to InfiniBand, this communication standard takes advantage of RDMA and credit-based flow control, to avoid the increased latency and lower throughput generated by frequent retransmissions. The current OmniPath generation is implemented using 100 Gb/s capable point-to-point links. At the time of writing Intel has cancelled any plan for a future 200 Gb/s version of OmniPath.

RDMA over Converged Ethernet

RoCE is a network protocol defined in an supplement to the InfiniBand specifications [43, 44] and the rationale behind its introduction was the idea of porting the advantages introduces by an RDMA transmission protocol to Ethernet – i.e. IEEE 802.3 [45] – installations.

The main disadvantage of this solution, compared to modern HPC-focused fabric designs, is the lack of an efficient flow control system. The default operation of an Ethernet-based network implements bufferless flow control, and allow congested nodes to drop incoming packets every time an input port does not have enough buffer space. In revision IEEE 802.3x of the Ethernet standard the concept of the *pause frame* was introduced, making the Ethernet flow control buffer-aware. Pause frames implement an on/off flow control similar to the one described in 2.5.4: every time a node is experiencing congestion and cannot receive new data it sends a pause frame, i.e. an off flow control message. Upon the reception of a pause frame, the sender will immediately stop injecting new packets. The standard does not implement any resume frame and the typical operation resumes after a predefined amount of time.

This approach to flow control was not designed with modern HPC in mind, and it has been improved in the IEEE 802.1Qbb standard with the introduction of Priority-based Flow Control (PFC). This improved flow control implementation allows a node to send pause frames that target only a subset of packets, called class. This addition to the standard allows a better granularity it the flow control implementation.

The improved flow control capabilities provided by properly configured PFC-capable network allow an Ethernet-based interconnection to meet the reliability requirements imposed by the InfiniBand protocol, and therefore it is possible to encapsulate an InfiniBand packet as the payload of an Ethernet one. This strategy allows to transparently reuse all the software that uses the IB software stack and all the advantages of RDMA over RoCE capable Ethernet networks. The main drawback is that IB flow control is superior to the one provided by PFC.

At the time of writing RoCE-capable network cards are produced by several vendors, and are available with different channel bandwidth from 25 Gb/s to 200 Gb/s.

3.3.2 Network topology

The selection of an appropriate network topology for the event building network is crucial for the correct operation of the entire system. As mentioned in section

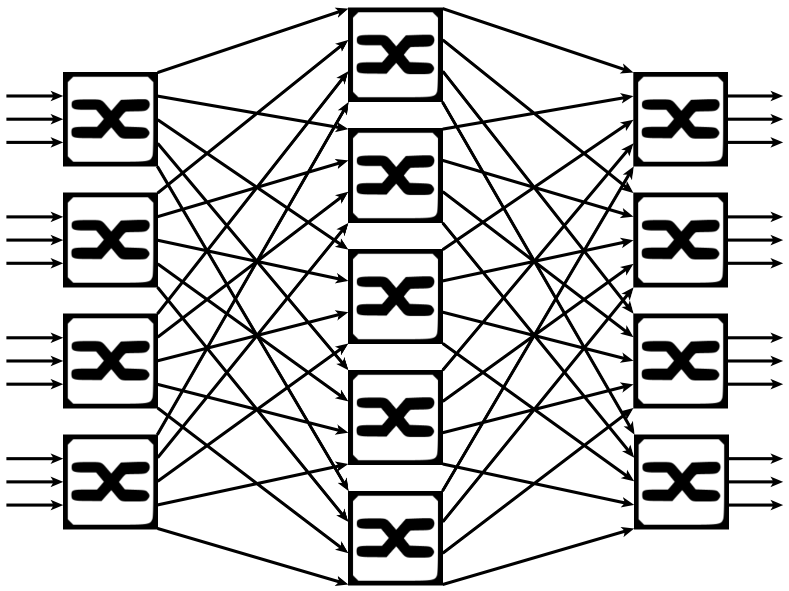


Figure 3.3. Example of a non blocking Clos network with: $n = 3$, $r = 4$ and $m = 5$. From left to right, the switches are arranged into three layers: ingress layer, middle layer and egress layer.

3.2.1, a linear shifting scheduling is a conflict-free solution to the ATAPC problem only if the network is non-blocking; therefore, this will be an essential requirement. Besides, the network has to be buildable out of COTS hardware; therefore, it has to be a cost-effective composition of available network switches.

The mathematical investigation on the problem of designing cost-effective and non-blocking switching systems started way before the HPC and large HEP experiments era. In 1953 Charles Clos published a study [46] about non-blocking switching networks for telephone switching. The solution proposed by Clos consisted of a three-stage switch-based network as the one depicted in Figure 3.3; the three layers are called from left to right ingress layer, middle layer and egress layer. Today this topology is referred to as Clos network after Charles Clos. The topology is completely defined by three integer numbers n, m and r: n is the number of terminal nodes that are connected to every ingress or egress switch. r is the number of the ingress or egress switches, which corresponds to half of the radix of the middle layer switches; m is the number of middle layer switches which corresponds to the number of ports in any of the ingress or egress switches connected the middle layer. In order for the network to be either non-blocking or rearrangeable non-blocking relations

	k-ary fat-tree	k = 16 fat-tree
Number of terminals	$2k^2$	512
Switch radix	$2k$	32
Number of leaf switches	$2k$	32
Number of spine switches	k	16
Total number of switches	$3k$	38

Table 3.1. Characteristics of a generic fat-tree network and a specific topology with $k = 16$

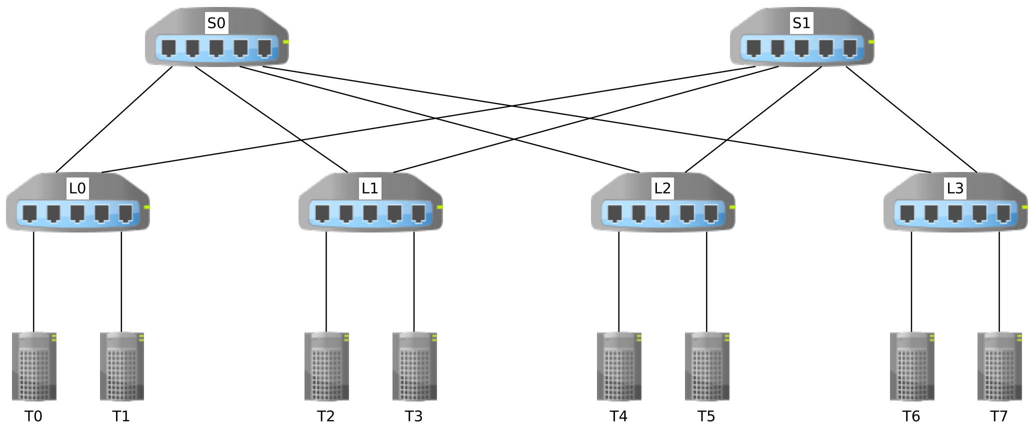


Figure 3.4. Example of a folded Clos network with $k = 2$. This particular example connects 8 terminal nodes via four leaf switches and two spine switches; the switch radix is 4 for all the switches.

(3.5) or (3.6) respectively must be satisfied [46].

$$m \geq 2n - 1 \tag{3.5}$$

$$m \geq n \tag{3.6}$$

The topology proposed by Clos has been revisited to be more HPC-centric, generating the folded Clos network topology, introduced in [47]. In the data centre world, and by the authors of [47], this topology is often referred to as fat-tree. This definition can be misleading because the term fat-tree has been already used previously to define a different network topology designed to interconnect CPUs in HPC systems, and introduced by Charles E. Leiserson in [48]. In this thesis, the term fat-tree will be only used to specify folded Clos networks. As depicted in Figure 3.4, the network topology is organised into two layers of switches: the ones in the bottom layer are called leaf switches; while the ones in the top one are called spine switches.

Contrary to a typical Clos network topology a fat-tree is fully defined by only one number k which defines the k-arity of the tree. All the leaf switches in the network have $2k$ ports, half of them are connected to k terminal nodes and the other half are connected to the k spine switches. A comprehensive view of the characteristics of a fat-tree network, can be found in Table 3.1, together with a configuration which is suitable for a LHCb event building scenario.

In order to be used as event building network, the topology must be at least rearrangeable nonblocking. Because a folded Clos network can be mapped into a normal Clos network it is sufficient to use (3.5) and (3.6). A fat-tree of a given arity k can be mapped onto a Clos network in the following way:

$$m = 2k$$
$$n = k$$
$$r = k$$

Therefore a fat-tree network topology is rearrangeably non blocking, because it verifies (3.6), but it does not satisfies (3.5).

In conclusion, a fat-tree network topology is a perfectly suitable solution for building a folded event building network of 512 RUs/BUs with a linear shifting-like traffic shaping, like the one needed by the new DAQ system of the LHCb experiment.

3.3.3 Routing algorithms

The selection of an appropriate routing algorithm is the next important step in designing the 40 MHz DAQ for the LHCb experiment. As any fully working algorithm, the one selected has to fulfil all the requirements presented in section 2.6, i.e. it must be: connected, deadlock-free and livelock-free.

In order to ensure that the algorithm is livelock-free, it is possible to limit the selection only to minimal algorithms. Because the traffic generated by a linear shifter is predictable, it is possible to design an algorithm that performs a fair load-distribution across the channels, without requiring non-minimal path to be included by the routing function.

The implementation of the static distributed deadlock-free algorithm can be done by using the up*/down* approach proposed in [49]. This class of algorithms generates an acyclic channel dependency graph by splitting the channels into two classes *up channels* and *down channels*. The attribution of a channel to a specific class is based on the definition of a spanning tree of all the link and the following rules:

- A channel belongs to the up class if moves packets closer to the root of the spanning tree
- A channel belongs to the down class if moves packets farther away from the root of the spanning tree

For a fat-tree topology, every channel that moves packets from the leaf switches towards the spine ones will be in the up class. The channels that move a packet from the spine switches towards the leaf ones will be in the down class[2]. To prevent any cyclic dependency it is sufficient to ensure that in any of the path selected by the routing function, the transition from a channel in the down class to a channel in the up class is forbidden. This strategy is used to generate deadlock-free routing algorithms on different topologies, e.g. an application on InfiniBand-based networks is presented in [50].

In addition to those basic properties, in order to achieve a conflict-free direct shifter routing, the algorithm has to select the right combination of paths, among all the ones that can be selected using an up*/down* strategy. A specialised algorithm for conflict-free routing on InfiniBand-based fat-tree based topologies has been presented in [51]. This static distributed algorithm defines the Linear Forwarding Table (LFT), i.e. the local correspondence between an output channel and the destination address of the packet, in a recursive way. This algorithm is designed to work on a wide range of generalised fat-trees [52], and a full characterisation of the algorithm can be found in [51], a specialised version designed to work on the topology described in section 3.3.2 will be presented.

In order to spread the communication generated by the event building traffic it is useful to introduce a channel usage counter, this counter represents the number of times that the routing algorithm selects a specific channel, and it is used to generate the LFTs.

The pseudo-code in Listing 3.1 describes the algorithm.

The presented algorithm distributes the up going traffic to a different spine switch according to destination switch port used by the destination terminal. Figure 3.5 depicts the paths selected by the algorithm for the second phase of a linear shift.

[2]As mentioned in section 2.2, a bidirectional link is considered as two unidirectional channels.

Listing 3.1. Pseudo-code implementation of the fat-tree optimised routing algrithm presented in [51]

```
def assing_down_port_ascending(switch, addr)
    local_port = find_least_used_ascending_port(switch)
    set_lft on local_port->remote_device addr to local_port->
        get_remote_port
    local_port->channel_usage_increase
    assingn_up_port_by_descending(switch, addr)
    assingn_down_port_by_ascending(local_port->remote_device, addr)

def assing_up_port_descending(switch, addr)
    for port in down_going_ports
        if lft (addr) != port and port->remote_device->is_terminal
            remote_port = port->get_remote_port
            set_lft on port->remote_device addr to remote_port
            local_port->channel_usage_increase
            assign_uo_port_descending(remote_port->local_device, addr)

main_loop
for leaf_switch in leaf_switches
    for terminal in leaf_switch->get_terminals
        set_lft on leaf_switch terminal->get_addr to terminal->
            get_remote_port
        assing_down_port_ascending(leaf_switch, terminal->addr)
```

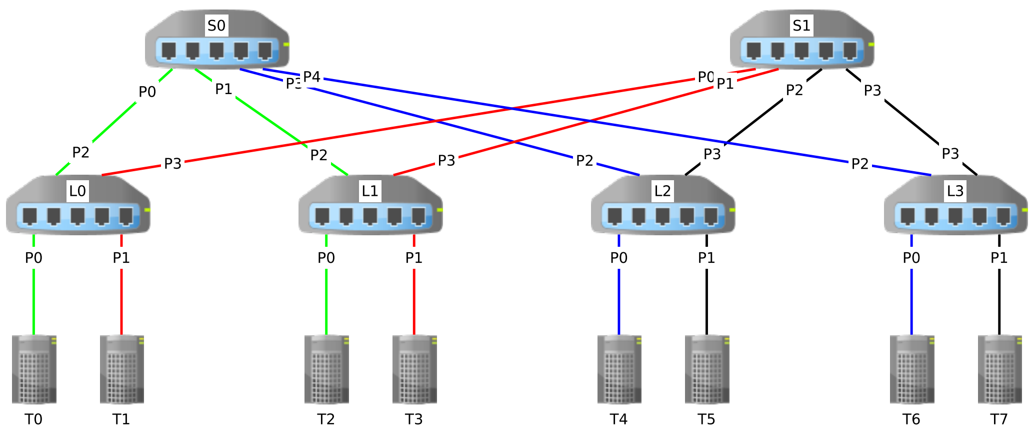


Figure 3.5. Example of the fat-tree optimised routing algorithm operating on 2-ary fat-tree. The different colours represent the paths selected by the algorithm for different packets.

For clarity sake, only half of the traffic is shown in the diagram in order to avoid confusion over bidirectional links.

3.4 Event Building network simulation

This section will describe the development and the performance tuning of a Flow Control Unit (flit) level behavioural network simulator. The network technology targeted for the simulations will be InfiniBand, in particular, the 100 Gb/s version of it – i.e. EDR. Moreover, an accurate model of the event building traffic will be discussed.

3.4.1 Simulation libraries overview

The process of simulating network infrastructures steers an intense development by the HPC community. In particular, many simulation frameworks have been developed over the years. In order to ease the framework selection, developers conducted several surveys such as the ones in [53, 54]. In particular, three of the most used open-source frameworks will be discussed: ns-3, J-sim and OMNeT++.

J-sim

JavaSim (J-sim) is a component-based compositional simulation environment. In order to ease the task of modelling complex hierarchical structures, the framework is implemented using the autonomous component programming model. The simulator software is written in Java, and many ready-to-use models are available. At the time of writing the latest release of the software dates back to 2006; therefore, the project seems to be abandoned.

ns-3

network simulator 3 (ns-3) is a discrete event simulator developed for educational and research purpose. The codebase is written in C++, and it offers Python bindings. A selection of pre-implemented, commonly used, network topologies are provided to the user. Moreover, it is possible to implement custom ones via the provided API in either C++ or Python. Similarly, behavioural models of commonly used protocols are available, and not available ones can be implemented. At the time of writing, there is no available model for the InfiniBand protocol, and the full simulation engine seems to be more focused towards the simulation of internet-used protocols. In order to speed-up the simulation of large systems, this framework provides a Message Passing Interface (MPI)-based parallel implementation.

OMNeT++

Objective Modular Network Testbed in C++ (OMNeT++) [55, 56] is a discrete event simulation framework developed for modelling a wide spectrum of interconnection networks from: telecommunication ones to distributes multiprocessor systems. The simulation library is written in C++ and, thanks to an extensible and modular design, it can be used to implement any discrete event simulation. Given that OMNeT++ is focused on network simulations, the simulation model is defined via a hierarchical structure of modules interconnected via channels. Modules can have two different nature: *simple* modules and *compound* modules.

Simple modules define the behaviour of the system, and they are implemented in C++ as an object-oriented specialisation of a simple module base-class. Compound modules are simple containers for an arbitrary number of simple or compound modules.

In order to define how the modules are interconnected, and therefore the full system topology, it is possible to use a topology description language called NED. This tool is provided together with the rest of the simulation software, and it is fully integrated with the software stack. The use of NED is not mandatory, and the user can directly instantiate the modules and define the topology in C++.

OMNeT++ supports parallel distributed simulations via a MPI, and tools for debugging the simulation model and collecting useful data from it.

This simulation framework is very well supported by both the open-source community and the industry, providing many models for multiple network protocols, including a Mellanox contributed InfiniBand behavioural implementation.

Framework Selection

Among all the frameworks presented in the surveys [53, 54], and after all the considerations expressed in the three presented in this section, OMNeT++ has been chosen to be the one used to implement a simulation of the LHCb event building network for the following reasons:

- it is an actively maintained project under constant development;

- it offers a powerful and modular architecture which can be easily customised to add support for different network architectures;
- the scientific community and the industry widely use it;
- it offers a powerful topology description language;
- it provides support for parallel distribute simulation via MPI;
- There is an InfiniBand protocol model which can be used as a starting point.

3.4.2 flit level InfiniBand simulation model

In order to implement an OMNeT++-based simulation model, it essential to understand the basic functioning of the framework.

Overview of the simulation engine

OMNeT++ is a discrete event simulator, therefore the simulated time advances from one event to the next one. This approach is very convenient for behavioural network simulation, because it allows to significantly reduce the amount of computing power available by only executing code for components that are not idling. Events can be scheduled at any time in the future and they are internally called *messages*. It must be noted that the messages defined by OMNeT++ are generic entities used by the simulation engine and they are not necessarily related with the network messages defined in chapter 2.

The main loop of the simulation engine will therefore store all the messages in a time ordered queue and implement the following steps:

1. pop the first message from the message queue
2. get the message scheduled simulation time and advance the simulation time to the same value;
3. execute all the relevant message callback functions
4. update the message queue with all the new messages that may have been generated by the callback functions

Within this framework the developer has to first identify which *modules* are needed to implement the various functionalities, and subsequently implement a behavioural description of all of them. As previously mentioned, the only entry point used by the simulation engine to interact with the user defined code of the modules is the message handling function. This function is called every time a message is received by a specific module. The callback function code can include arbitrary user defined tasks, like updating internal state variables or sending an arbitrary number of messages to other modules, including self messages. In order to receive or send messages modules need *ports*, an arbitrary number of input, output or input/output port can be specified for every module.

The last piece needed to implement an interconnection network simulation is a way of connecting the modules. In OMNeT++ this is done via the use of C++ object called *channel*. In the default operation mode the topology is defined via the NED language, and the modules can only send messages to and receive message from modules which are directly connected. This connection schema allows to implement point-to-point connections. In order to simulate shared-medium networks it possible either to model the behaviour of the shared-medium with a module or to leave the ports unconnected, and send messages directly to the remote port of another module.

The channels implemented in OMNeT++ can be either ideal channels with infinite bandwidth and no latency, or realistic ones with appropriate numbers for both parameters. Ideal channels are useful to model the internal connection of complex modules (like a switch), while realistic channels can be used to implement external connection between network devices.

Mellanox contributed InfiniBand flit model

An OMNeT++-based InfiniBand flit level simulation model has already been developed by Mellanox technologies, and it has been released under the General Public License (GPL) version 2 license. This model provides a behavioural description of several features of the InfiniBand fabric including: credit-based flow-control, static distributed routing based on LFTs, Virtual Lanes (VLs) arbitration[3], packet generation and fragmentation and packet arbitration.

Unfortunately this behavioural model is no longer maintained by the original developers and it has not been updated to support InfiniBand EDR. Moreover the provided software does not replicate the behaviour of any existing hardware, and several of the model parameters need to be set to realistic values in order to achieve accurate simulation results.

In order to achieve meaningful results: the original simulation model has been

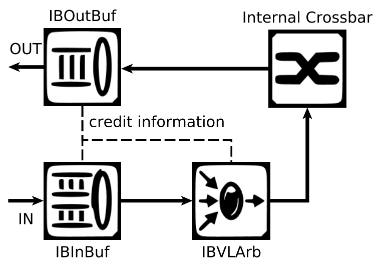
[3]A VL is the InfiniBand implementation of the virtual channel concept explained in section 2.6.3.

updated and improved; the relevant parameters has been directly on indirectly measured on the actual COTS hardware.

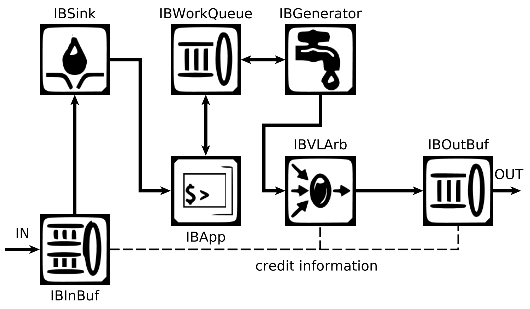
Modules implementation

Hereafter a detailed list of the implemented modules [57] will be given, including a description of their behaviour:

- **IBOutBuf:** it can be used as an output buffer for the outgoing flits. In addition to the buffer functionality this module implements the link controller and the credit based flow control;
- **IBInBuf:** it provides a set of buffers for incoming flits. Every VL has a dedicated buffer. Similarly to the IBOutBuf module this one implement link and flow control features;
- **IBVLArb:** it implements arbitration among the different VLs and selects which input channel should forward the next packet, if there are multiple suitable input channels a round robin scheduling is applied;
- **PktFwd:** it simulates the routing logic by selecting which output port should be used to forward the packet. This implementation is based on static distributed forwarding tables (LFTs) which are used to assign an output port based on the destination address of the packet, like in the actual hardware. Every instance of this module will load the routing information from a file;
- **SwitchPort:** it is a compound module that combines input and output buffers with the VL arbitration logic;
- **IBApp:** it submits an InfiniBand Work Request (WR) [58] to the appropriate Work Queue according to the selected traffic pattern. Multiple specialisations of this class can be used to generate different traffic, and different applications can be used at the same time;
- **IBWorkQueue:** it receives all the WRs from all the *IBApps* and it serves them when the *IBGenerator* is ready to process a new message. The communication between this module and the *IBGenerator* can be delayed in order to simulate the latency of the PCIe bus.



(a) Switch port implementation. The Internal Crossbar is not part of the switch port but it has been added for clarity.



(b) Terminal node implementation.

Figure 3.6. Internal structure of a switch port and a terminal node. The solid lines represent the data path, the dashed lines represent propagation path for the link layer flow control credits.

- **IBGenerator:** it pulls the WRs from the *IBWorkQueue*, and it fragments the message contained into the WR, first into packets and then into individual flits;
- **IBSink:** it receives the packets and notifies the *IBApp* module upon completion. This last feature is critical for simulating real world application because it makes the *IBApp* aware of the inbound traffic, and therefore able to reply to incoming messages

The aforementioned modules can be combined together to implement InfiniBand network devices such as switches and terminal node, Figure 3.6 depicts the implementation of a switch port and a terminal node.

The implementation of modules required is a first step towards implementing the simulation of a realistic system, but more software components are needed. In particular, because the *PktFwd* module implements a LFT-based routing algorithm, an automatic way of generating those tables has been implemented. Because the aim of this thesis is to simulate event building traffic, the fat-tree optimised routing algorithm [51] described in section 3.3.3 has been accurately implemented; moreover, because it is interesting to simulate the behaviour of existing systems, a translation layer that converts InfiniBand LFTs into the format used by the simulator has been implemented. With this set of tools it is therefore possible to either implement any routing algorithm or to replicate existing installation's routing.

The last missing software component concerns the topology generation. The creation of parametric versions of regular topologies, like the fat-tree described in 3.3.2, can be implemented using NED as shown in Listing 3.2. The code is split in three part: *parameters*, *submodules* and *connections*; the first section contains the definition of all the relevant parameters needed to specify a specific implementation of a parametric topology, in this case the arity of the network plus some other values derived from it; the second section defines the number and the type of the modules used, in this case there switches and hosts – i.e. terminals; the last part defines the interconnection between the modules, in this case there two blocks of connections, the first one for the terminal/leaves connectivity, the second one for the leaves/spines backbone.

Parametric NED-based topology descriptions are a powerful tool, but they are not convenient for simulating a replica of an existing HPC cluster, taking into account: unavailable terminals, storage and login services, asymmetric additions of terminal nodes and matching the addressing schema used by the routing algorithm. In order to facilitate this task a script has been developed, this program can parse the topology dump collected from an existing cluster, and generate a non parametric NED-based description of the network topology. This tool, together with the one that converts the LFTs, makes the simulation of a real world system possible. This process allows the user to perform simulations before running benchmarks, resulting in a more efficient use of the limited and precious time available on the cluster; moreover it is possible to compare the results collected from HPC installations against the ones collected from a simulation of the same system. This last operation is crucial part of the model validation process.

3.4.3 Model tuning

The simulation model described in section 3.4.2 provides a behavioural description of an InfiniBand based network; in order to make software flexible and capable of implementing both different generations of InfiniBand and different hardware devices, all the low level details of the implementation are configured via parameters. To achieve an high grade of realism, the parameters of the OMNeT++-based library need to be set to the right value, as described in [57].

Model's parameters can be divided into two main categories: protocol specific and hardware specific. The first set contains quantities that are fully defined in the InfiniBand architecture specification [59] therefore their values are fully fixed after

Listing 3.2. Parametric description of a fat-tree network topology

```

network fat_tree
{
    parameters:
        int arity = default(18);
        int num_spine = arity;
        int nodes_per_leaf = arity;
        int num_leaf = arity*2;
        int num_nodes = num_leaf*nodes_per_leaf;
    submodules:
        switch[num_spine + num_leaf] : Switch {
     parameters:
            numSwitchPorts = arity*2;
            @display("i=abstract/router");
        gates:
            port[arity *2];
        }
        hosts[num_nodes] : HCA {parameters: srcLid = -1;}

    connections allowunconnected:
        for i=0..(num_nodes - 1){
            hosts[i].port <--> IB4XEDRWire <--> switch[i/(
                nodes_per_leaf)].port[i%(nodes_per_leaf)];
        }

        for sw=num_leaf..(num_leaf + num_spine - 1) , for p=0..(arity*2 - 1)
            {
            switch[sw].port[p] <--> IB4XEDRWire <--> switch[p].port[sw-
                nodes_per_leaf];
        }
}
```

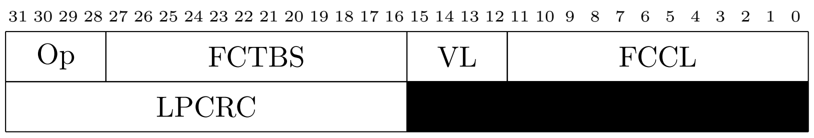


Figure 3.7. InfiniBand flow control packet format, as defined in [59]

the selection of the specific InfiniBand implementation, such as: channel bandwidth, maximum payload size, flit size, protocol overhead and flow control behaviour. Because the purpose of this thesis is to evaluate an implementation for the LHCb DAQ system the protocol configuration has been set to match the EDR one. Hardware specific are not specified in the protocol specification and they need to be estimated performing real measurements and reverse engineering on the actual hardware.

The most important values for accuracy of the simulation model are: switch buffer size, link layer latency and PCIe latency; the buffer size of the switch is critical for a realistic behaviour of the InfiniBand flow control, an accurate modelling of the latency is needed to reproduce the same link congestion as in a real system. In all the tests describe in the following section the hardware tested was a Mellanox SB7700 EDR switch and Mellanox MT27700 ConnectX-4 Host Channel Adapters (HCAs), unless explicitly specified.

Switch buffer measurement

The switch buffer size can be measured following two different approaches[4]: the first one consists of analysing the flow control credits; the second one is based on generating controlled network congestion and monitoring the congestion indicator[5].

The first method proposed consists of extracting the buffer occupancy information from the flow control packets. In order to extract this information it is crucial to understand the anatomy of an InfiniBand credit. Figure 3.7 describes all fields present in an InfiniBand flow control credit, a description of the field function follows:

- **Operation (Op):** this 4 bits value identifies the nature of the flow control packet, a value of 0x0 indicates a normal credit, while a value of 0x1 indicates a special one sent a the initialisation of the link;

- **Flow Control Total Blocks Sent (FCTBS)**: this 12 bits counter contains

[4]The methodology described in this section is based on the presentation held by Qian Liu at the OpenFabrics Software User Group Workshop in 2015.

[5]The congestion indicator is the PortXmitWait counter which indicates the time, expressed in clock ticks, that the data transfer on a given port has been paused by the flow control.

the number flits sent over the link;

- **VL**: this 4 bits value indicates the virtual lane involved by this specific credit
- **Flow Control Credit Limit (FCCL)**: this 12 bits counter contains the sum of the number of flits received over the channel plus the available space in the input buffer;
- **Link Packet CRC (LPCRC)**: this 16 bits CRC code is used to detect transmission error and bit flips.

It is important to note that the buffer status information is encoded in an incremental way; this encoding protects the system against data loss or data corruption on the credit itself, because the flow control information is critical to the link operation the remote buffer status must be reconstructible after an arbitrary number of missed credits. In order to extract the buffer size from the flow control information it is needed collect a credit sent from each of the 2 devices sharing the channel; the credit sent from the receiver A contains the buffer size information, while the credit sent from the sender B contains the amount of data sent over the link. To disentangle the available buffer space b_{free} from the number of sent flits this procedure it is sufficient to subtract the relevant fields of the credit packets:

$$b_{\text{free}} = \text{FCTBS}_B - \text{FCCL}_A \tag{3.7}$$

If the link has been idling for a sufficient amount of time and there is no congestion on the network devices under test we can assume than $b_{\text{free}} = b_{\text{size}}$, and therefore the the amount of buffering available can be measured.

This approach has two disadvantages: the first one is that according to the IB specifications [59] the receiver should not announce more that 2048 available slots in the buffer, regardless of the actual capacity; therefore any buffer size of more that 128 KiB cannot be directly measured. The second issues is related to how the flow control information can be analysed; because the flow control information is only used by the link control hardware it is not possible to acquire this information in software. The only way of monitoring the flow control information is to use a low level InfiniBand protocol analyser: a low level hardware device which inspects and decode all the packets send over a channel. At the time of writing there are no EDR-capable protocol analysers on the market; therefor the second strategy was put in place.

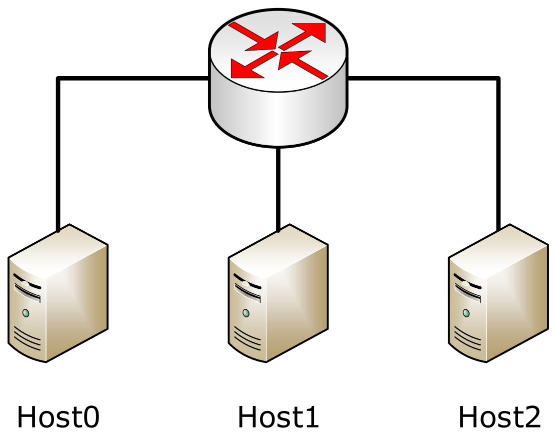


Figure 3.8. Setup used to generate congestion and estimate the switch buffer size. Three terminals named Host0, Host1 and Host2 are interconnected via a switch. Host0 sends data to Host2 at the maximum data rate possible, at the same time host1 sends packets of different sizes to create controlled congestion.

The second strategy aims to create controlled network congestion and monitor the status of the network devices. Figure 3.8 shows the network topology used to create the controlled congestion, which is constituted by three terminal nodes and one switch. The procedure to generate the controlled network congestion can be summarised in the following steps:

1. *Host0* sends continuously to *Host2* at the maximum data rate possible, i.e. trying to fully saturate the channel bandwidth;

2. *Host1* sends to *Host2* messages of increasing size at regular intervals, to create congestion;

3. A monitoring program reads the PortXmitWait performance counter and logs it together with the size of the message that were causing the controlled congestion.

In order to estimate the buffer size it is possible to interpolate the time that the link was stop by the flow control with the size of the the messages sent by Host1. Following this procedure the estimated buffer size si 64 KiB per port per VL with 4 VLs enabled.

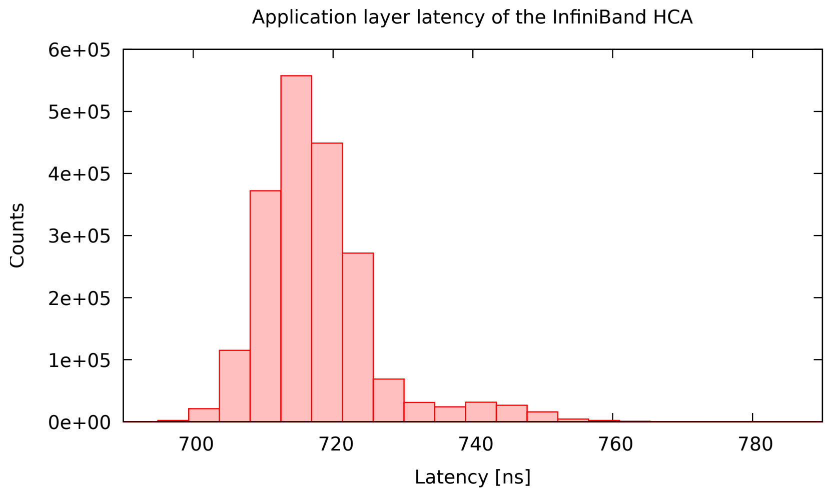


Figure 3.9. Application and PCIe latency of an InfiniBand EDR HCA.

Link-layer latency

In order to measure the link layer latency, i.e. the latency introduced by the channel and the link controller, the hardware timestamping feature of the IEEE 1588-2008 standard [60] – i.e. Precision Time Protocol (PTP) – has been used.

PTP is a high precision time synchronisation protocol designed to achieve sub microsecond precision. In order to achieve the synchronisation level required a PTP-capable device has to precisely measure the link latency with the highest accuracy possible. To perform this accurate measurement extra hardware onto the network device which allow a low level timestamping of the synchronisation packets, and therefore making possible to accurately measure the link-layer latency without using a protocol analyser.

The path latency measure using PTP produced an estimation of 170 ns full delay using a 3 m long direct attached copper cable, between two directly connected MT27700 ConnectX-4 HCAs.

PCIe latency

The final missing parameter in this simulation model is a realistic model of the combined latency introduced by the PCIe bus and the InfiniBand software stack; because of the non-real-time nature of modern computing systems and software, instead of developing a theoretical model of the system a phenomenological approach has been taken.

The latency has been measured on the same test setup used for measuring the link-layer latency. In order to perform the measurements the *ib_write_lat* benchmark has been used and the previously measured link layer latency has been subtracted; therefore this measurement includes all the latency introduced from the hardware and software chain before the link controller.

Figure 3.9 shows the histogram of the latency measurements, the simulation model draws random number generated from this distribution to replicate latency and jitter of the real system.

3.4.4 Traffic injector implementation

An accurate model of the interconnection network is not complete without a realistic replica of the injected traffic. Because the target of this thesis is to evaluate solutions for the LHCb event building network, the simulation of DAQPIPE and a2a generated traffic is a strong requirement.

Simulated a2a

The traffic pattern generated by the a2a benchmark is a linear shifter with a throughput limit; therefore the implementation of this traffic injector is extremely simple. If the underlying network topology and routing algorithm provide conflict free paths for every phase of the shift, then this traffic pattern will provided the target bandwidth and there is no need of simulating it; on the other hand if the network is not optimised for a linear shift or if the nodes lose their phase synchronism, a dynamic simulated study of the network can provide a better understanding of the problem.

Simulated DAQPIPE

An accurate replica of the DAQPIPE benchmark has been implemented, this traffic injector replicates all the major features described in section 3.2.2, including all the configurable parameters that allow to change the traffic scheduling.

The traffic injector replicates the behaviour of the three logic units that implement all the task required by the event building process; the functionality is statically assigned to the terminal nodes at the beginning of the simulation.

Because the simulated application is part of a discrete event simulation, the flow of the program has to be steered by the incoming messages. In order to achieve this messages need to be tagged and identified according to their function. A list and description of the different classes of messages follows:

Listing 3.3. State machine of the DAQPIPE-like traffic injector

```
[. . .]
    if (app_msg_kind == EM_EVENT_ASSIGN){
        send_ru_requests();
    } else if (app_msg_kind == RU_REQUEST){
        send_event_fragment(p_msg);
    } else if ((app_msg_kind == EM_REQUEST) && (event_manager_lid
        == srcLid)){
        send_event_assign(p_msg);
    } else if (app_msg_kind == DATA){
        handle_event_fragment(p_msg);
    } else {
        handle_error(p_msg);
    }
[. . .]
```

- `EM_REQUEST`: this message emulates the announcement of a new available credit sent by a BU;

- `EM_EVENT_ASSIGN`: this message is sent by the EM every time a new event is assigned to a particular BU;

- `RU_REQUEST`: this message is the pull request sent by the BU to the RU;

- `DATA`: this message contains the actual event fragment.

When the simulation starts every BU sends to the EM a number of `EM_REQUEST` messages equal to the number of credits configured. When the EM receives the `EM_REQUEST` message it will reply with an `EM_EVENT_ASSIGN` one. When an event is assigned to a BU will send a number of `RU_REQUEST` messages equal to the number of parallel sends configured. The sends are scheduled in a linear shifter-like pattern, as previously mentioned in section 3.2.2. This behaviour is implemented by the state machine listed in 3.3.

This traffic injector is capable of matching the messages sent by DAQPIPE, and together with the rest of the simulation model allows to study the performance of the system and to find the optimal working point for any give network topology.

3.4.5 Fast model implementation

The simulation of a 100 Gb/s interconnection network with a flit-level accuracy produces a high number of events in the simulation engine. The order of magnitude of the event rate can be easily calculated from the channel bandwidth and the size of a flit S_f:

$$E_{\text{rate}} = \frac{b}{S_f} \tag{3.8}$$

An InfiniBand EDR network has a bandwidth of 100 Gb/s and a flit size of 64 B; therefore according to (3.8) the rate of events generated in the simulation engine is $\simeq 200 \cdot 10^6$ s^{-1} on every unidirectional channel.

The amount of events that can be processed every second by an OMNeT++-based application depends heavily on the specific implementation of the code. For this specific project event processing rate on a single core of a modern x86 CPU it is $\sim 1 \cdot 10^6$ s^{-1}, resulting in a simulation which is 200 times slower than realtime.

The full LHCb DAQ network is composed of $\sim 10^3$ bidirectional links. Under the hypotheses of linear increase in the number of events generated and a constant event processing rate, the simulation will be $4 \cdot 10^5$ times slower than realtime. It should be noted that this estimate is rather optimistic because it does not take into account all the events generated by the internal components of compound modules, and that the event processing time does not remain constant when the size of the event queue increases.

From this preliminary estimation perform a full event building simulation for the LHCb experiment, using a flit model, is highly demanding in terms of computing power. Moreover, in order to find the optimal working point of the different applications, multiple simulations are needed with different: topologies, routing algorithms and parameter configurations.

A first way of reducing the time needed is to take advantage of the parallelism of modern compute architectures and distributed computing clusters. The OMNeT++ framework provides parallel processing via MPI, and therefore it would be possible to run a single simulation on a distributed set of compute cores. An other approach will be run all the multiple independent configurations in parallel, each running on a single compute core. Because of the large number of simulations needed and the perfect scaling provided by the second solution, for this project, the second strategy has been chosen.

In order to further reduce the amount of compute power needed it is possible to reduce the amount of events generated by aggregating multiple flits into one simulation event. This approach reduces the realism of the simulator, and the impact on the final results and the speedup achievable will be evaluated in section 3.4.6. Because InfiniBand implements a VCT flow control the global effect on the full simulation should not be too severe. On the other hand, the amount of events generated by every channel will be reduced by a factor equal to the flit aggregation.

3.4.6 Simulation results

In this section the results obtained with the previously mentioned simulation will be presented and analysed, from the validation of the model itself to the simulation of the full LHCb event building network.

Validation of the simulation model

In order to validate the simulation model it is crucial to compare the simulation results against data gathered on real clusters.

The most complex event-building-like benchmark implemented in the simulation framework is DAQPIPE therefore the validation of the model will be against a real run of DAQPIPE on a HPC cluster.

The first step required for this validation is to build a simulated network with the same topology and routing algorithm as the original one. In this specific case the network topology is a fat-tree-like network with 64 terminals; this particular network was not a one-to-one implementation of the one described in section 3.3.2, and there were missing nodes and swapped cables. The routing algorithm implemented on the network was not the one described in section 3.3.3 and therefore the selected paths were sub-optimal for a linear shift traffic.

The decision of using this particular topology for the simulation validation was driven by two major motivations:

- the network topology of this specific installation was well documented and the all relevant information was provided by the administrators of the site;
- running the benchmark in a sub-optimal environment will generate higher levels of network congestion, resulting in higher stress for the flow control system, and providing a more challenging environment to the simulation model.

The metric used for this validation is the average effective event building throughput, and the same criterion is applied to the tests on the real system and on the

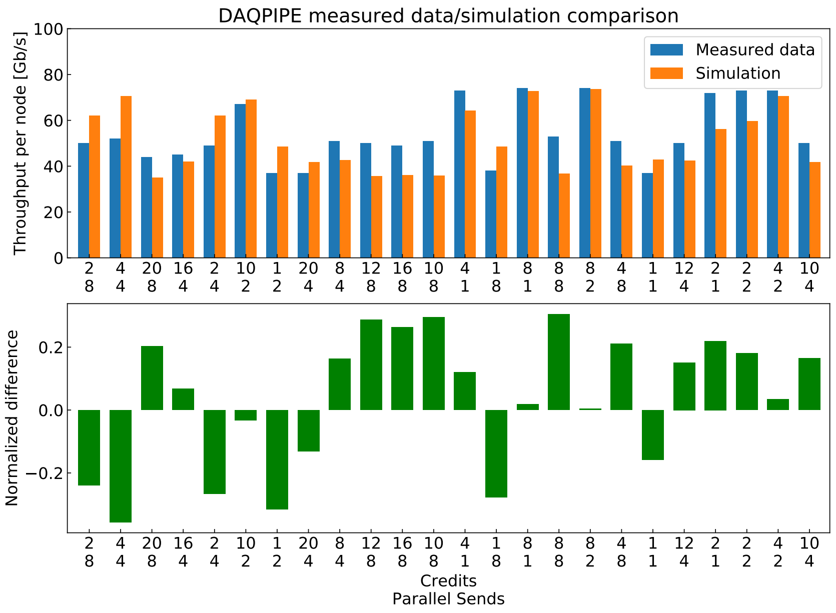


Figure 3.10. Comparison between measured and simulated DAQPIPE on an HPC cluster of 64 nodes, as presented in [57]. The upper panel shows the average event-building throughput rate, and the lower panel shows the difference between measured and simulated throughput, normalised to the measured data.

simulated one.

A comparison of the simulated and the real DAQPIPE for different values of the *credits* and *parallel sends* parameters is shown in Figure 3.10. From this comparison it is possible to confirm that the simulation model is accurate, and that it can reproduce the behaviour of an event building benchmark application on a given network topology. The simulation can reproduce the throughput trend changing the parameters, and the absolute throughput value within 30% error across the full parameters space. In particular for the high throughput configurations the simulation is well within 20% from the measured data. Given the complexity of simulating such a complicated system, this simulation model provides an efficient and powerful tool to evaluate the scalability of the system.

Sub-optimal network configurations

The main disadvantage of a linear shifting scheduling technique is the high susceptibility to changes in the network configuration; if the network topology and the routing algorithm do not allow a perfect traffic balancing the resulting network congestion will impact the full event building performance.The final system will be

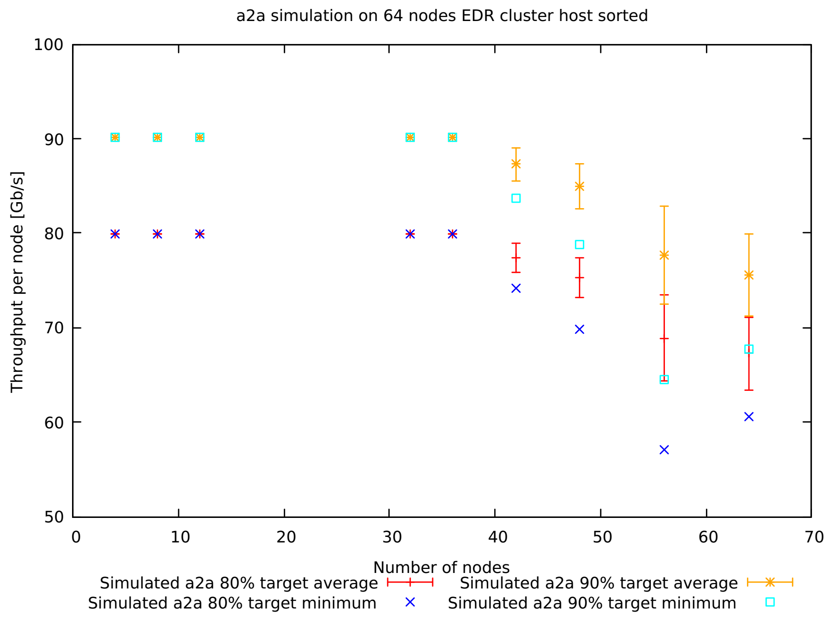


Figure 3.11. Simulation of the a2a benchmark on real HPC 64 nodes cluster. The plot depicts the average and minimum node throughput for different configurations of the application. The permutation used in the linear shift is based on the host names of the terminal nodes.

designed to operate in optimal conditions with the appropriate network topology and routing algorithm; nevertheless it is important to evaluate the impact potential partial failures, and determine the impact on DAQPIPE and a2a.

The plots in Figure 3.11 and 3.12 show the simulation results for different configurations. In all the tested scenarios a2a can reach the requested throughput if the number of nodes stays below 64, i.e. the number of switches involved is lower, because the probability of getting congestion introduced by the sub-optimal network configuration is lower. When the number of involved nodes increases the performance degradation becomes more severe and the average throughput drops by $\sim$ 10 Gb/s while the minimum throughput drop by $\sim$ 20 Gb/s. Because a2a needs to keep a strong phase alignment among the nodes, and because the event building needs to build full events, the effective event building throughput of an a2a-like approach will the throughput of the slowest node rather than the average one.

The different shape and magnitude of the minimum throughput drop between Figure 3.11 and 3.12 shows how the particular node permutation affects the performance. In particular the simulations depicted in Figure 3.11 are configured with a

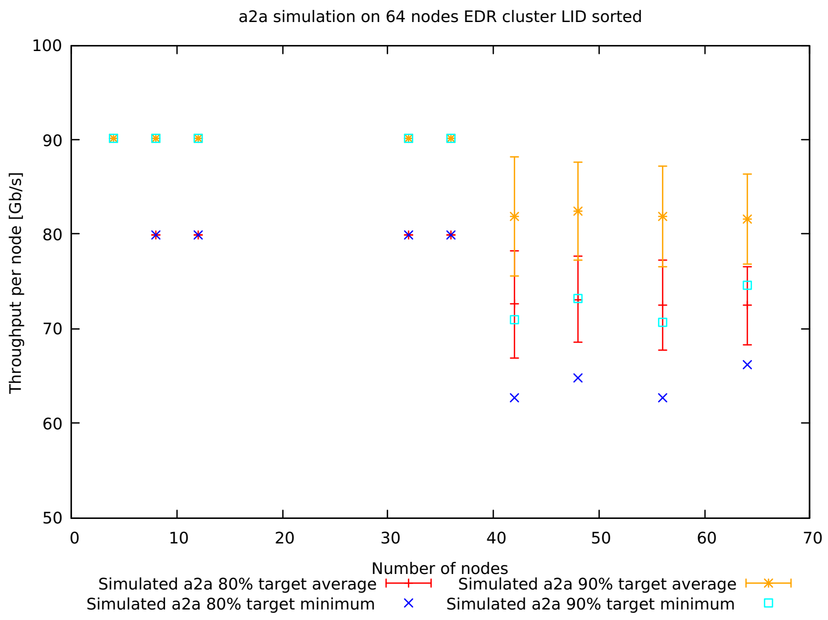


Figure 3.12. Simulation of the a2a benchmark on real HPC 64 nodes cluster. The plot depicts the average and minimum node throughput for different configurations of the application. The permutation used in the linear shift is based on the network address of the terminal nodes.

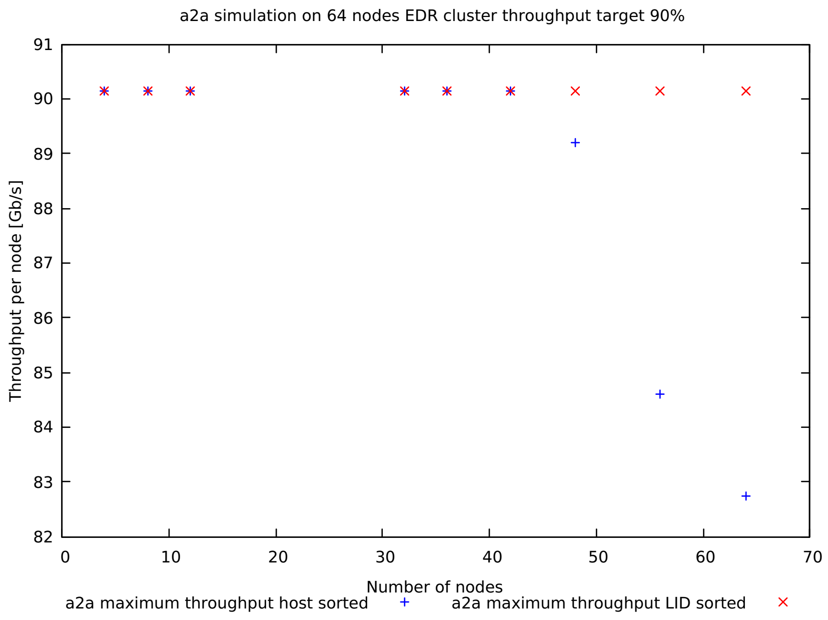


Figure 3.13. Simulation of the a2a benchmark on real HPC 64 nodes cluster. The plot depicts the maximum node throughput for the host name-based configuration and the network address based one.

host-name-based permutation, which is unrelated to the network topology and the routing algorithm. On the other hand the simulations shown in Figure 3.12 use a network address-based one, which is strongly correlated with the network topology and the routing algorithm. In particular the minimum throughput is more stable when the permutation is address based and the minimum value for high node counts is higher. In order to understand this behaviour it is useful to check the maximum throughput for the two different nodes permutation, as depicted in Figure 3.13; the analysis of the maximum throughput show how the network congestion affects the nodes in a more predictable way when the scheduling is linked with the network configuration. Resulting a lower number of nodes being affected by the network congestion. On the other hand a network independent scheduling generates heavier and less predictable congestion which affects all the nodes.

The a2a benchmark is therefore very stable when the network can provide a conflict-free packet delivery; on the other hand it suffers from network congestion introduced by non ideal-scheduling. In particular the results simulated on a 64 nodes non conflict-free topology show significant performance degradation of ~ 20 Gb/s.

Figure 3.14 depicts a performance comparison of the simulated DAQPIPE on two different topologies: a clean fat-tree of 72 nodes and an HPC cluster of 64 nodes. The performance degradation introduced by a non conflict-free scheduling is highly dependant on the parameters of DAQPIPE, and can be as high as 50%; nevertheless the bandwidth drop for the fastest configuration is 6%.

The effects of a non-ideal network infrastructure affect DAQPIPE's performance and makes it more unstable, the throughput degradation can vary significantly and it is highly influenced by the configuration parameters and the topology itself. Configurations with many *parallel sends* are affected in a more severe way because they increase the number of nodes communicating at every phase, especially if in conjunction with a high *credit* count, increasing the probability of local link congestion.

The DAQPIPE benchmark can absorb the performance degradation introduced by a non conflict-free network by having an adequate number of message in flight. On the other hand the parameter tuning is critical, and a misconfiguration of the parameters can lead to severe throughput degradation.

Fast model validation and scaling

As mentioned in section 3.4.5 it is possible to aggregate multiple flits into a single simulation event; this procedure should reduce the amount of compute time needed

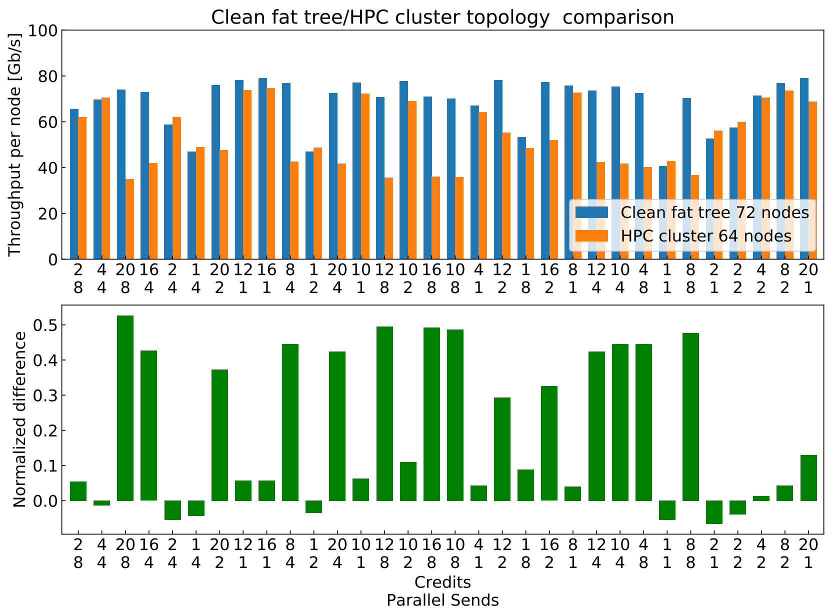


Figure 3.14. Comparison between simulated DAQPIPE on an HPC cluster topology of 64 nodes and on a fat-tree of 72 nodes,as presented in [57]. The upper panel shows the average simulated event-building throughput rate, and the lower panel shows the difference in simulated throughput between the clean topology and the topology available on the HPC system, normalised to the throughput of the 72 nodes fat tree topology cluster.

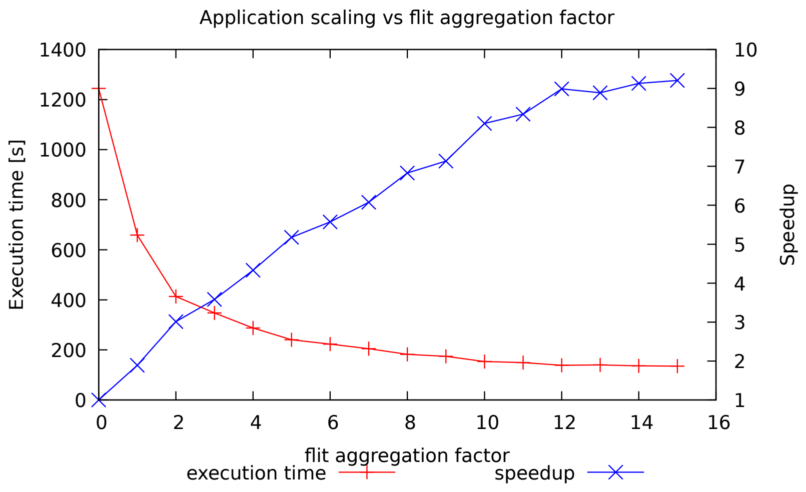


Figure 3.15. Computational scaling of the fast simulation model as a function of the flit aggregation factor. The execution time refers to the left y axis, while the computation speedup refers to the right one.

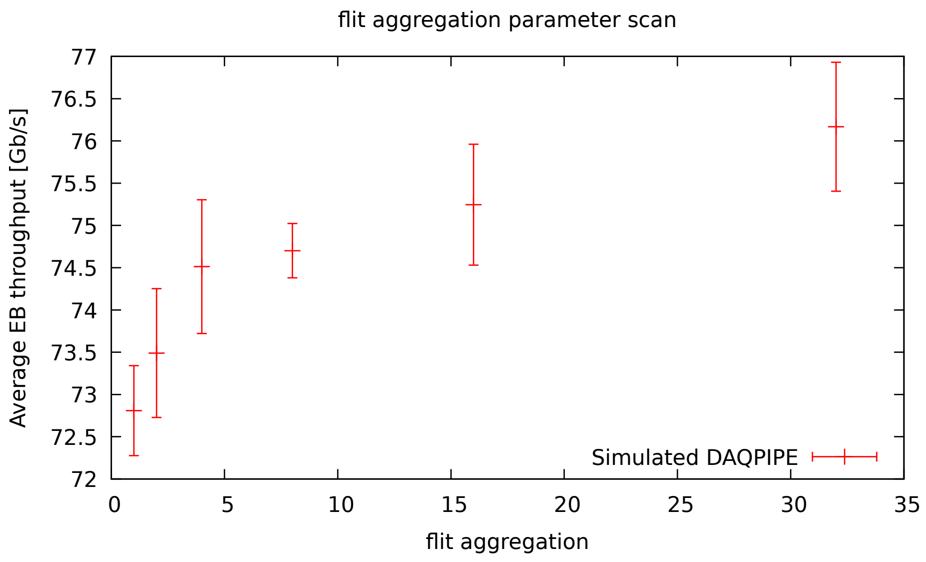


Figure 3.16. Comparison of different DAQPIPE simulations with different values for the flit aggregation parameter.

to perform a given simulation task. The computational scaling of a DAQPIPE simulation is depicted in Figure 3.15; the plot shows a significant speedup of the application of up to a factor 9. The fast simulation model can therefore reduce the amount of compute power needed to simulate the LHCb event building network by an order of magnitude.

The coalescing of multiple flits into a single simulation event reduces the realism of the simulation model; therefore a validation against the normal model is needed. Figure 3.16 shows a comparison between different simulations of DAQPIPE with different values of the flit aggregation parameter; this comparison shows how the variation in the average EB throughput is modest over the full parameter range, in particular if the flit aggregation factor is limited to 16 the average EB throughput difference is 2.4 Gb/s.

The fast simulation model provides a significant reduction the the computing resources needed, and it introduces a small deviation in the overall accuracy of the simulation.

Full scale simulations

The ultimate objective of the simulation model is to provide performance figures for untested systems; and in particular to perform simulations of large scale sys-

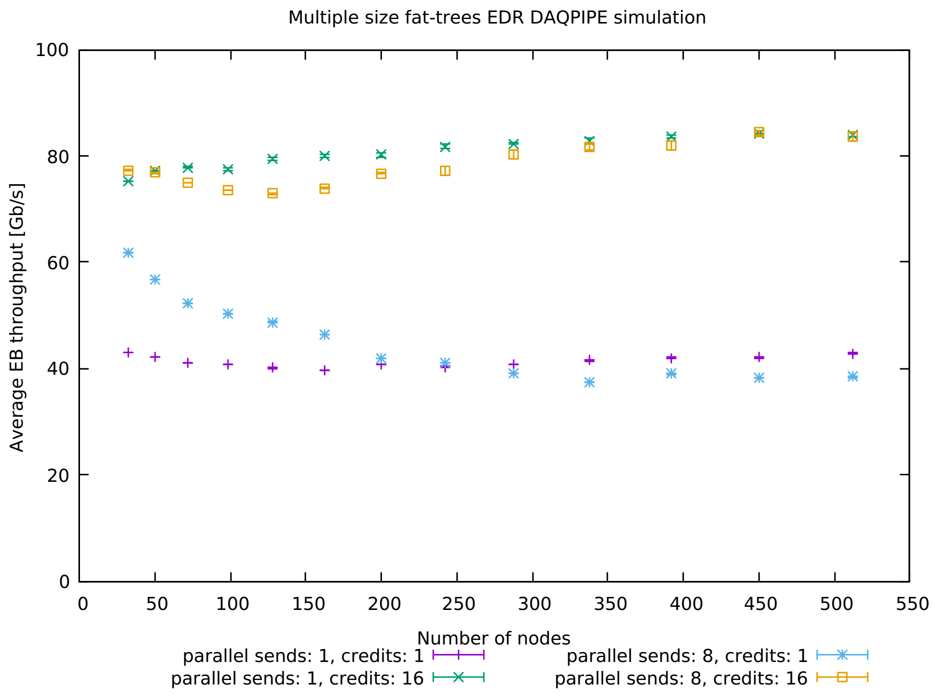


Figure 3.17. Average EB throughput of simulated DAQPIPE vs number of nodes. All the network topologies tested are full fat-trees, the different node count is achieved by changing the number of terminal nodes on every leaf switch.

tems. Because simulating large systems requires a significant amount of computing resources, the fast model will be used with an aggregation factor of 16.

Among the two available benchmarks DAQPIPE is the most interesting to simulate for the following reasons: the performance variation introduced by a change in the number of credits or parallel sends are highly unpredictable; the application can generate non trivial congestion patterns that need to be evaluated in simulation. On the other hand a2a produces a more predictable traffic pattern, and it can be negatively affected by two factors: node desynchronisation and conflicts in the routing paths. The simulation of the first phenomenon requires a low level implementation of the OS scheduler and the interaction between various processes inside the terminal node, introducing an high grade of non-reproducibility into the model. The conflicts in the routing paths have been previously analysed on smaller scale systems, and they can be studied on larger scale with a static path analysis.

The scalability of the simulated DAQPIPE has been tested in two different scenarios: the first one was a test run on different fat-tree topologies with different

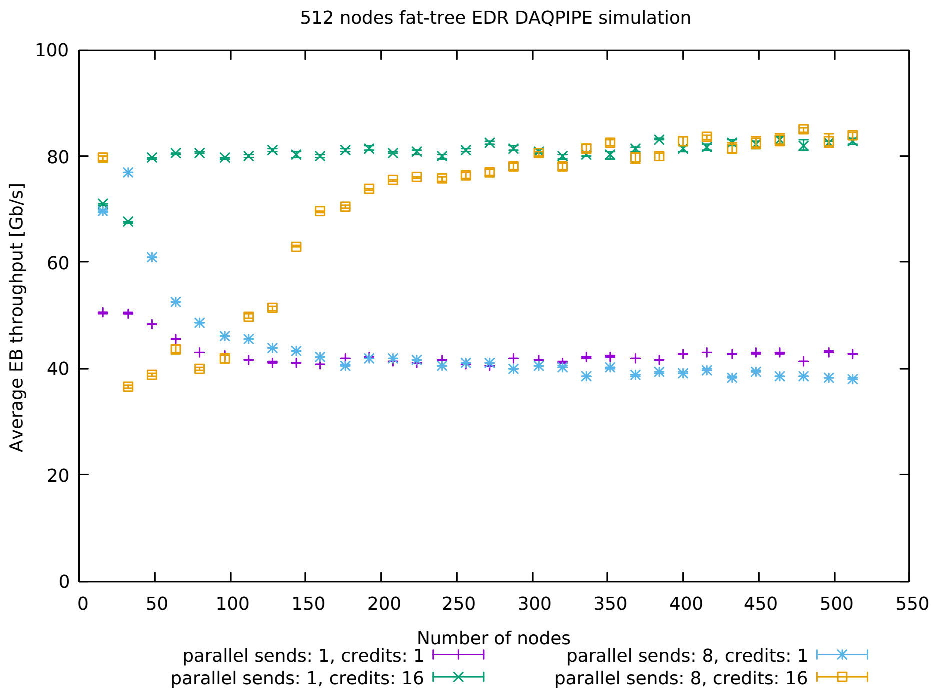


Figure 3.18. Average EB throughput of simulated DAQPIPE vs number of nodes. The network topology tested is a 512 nodes fat-tree, the different node count is achieved by changing the number of leaf switches with active nodes connected.

number of terminal nodes; the second one was simulation of a 512 nodes fat-tree with different number of active terminals. In order to keep the two configurations tested conflict-free, in the second scenario nodes were added but the leaf switches population was kept homogeneous – i.e. the number of active nodes per leaf switch was either zero or the arity of the fat-tree.

The graph in Figure 3.17 presents the results obtained simulating the first of the two aforementioned scenarios. The simulations were executed with different parameter configurations, in order to test the behaviour of DAQPIPE under different conditions. The configurations with 16 credits show solid performance in term of scaling and throughput, with the best configuration delivering more than 80 Gb/s in most of the tests. On the other hand the configurations with only one credit suffer from the lower number of fragments being transferred in parallel, and the higher latency in the communication path with the EM; in particular the configuration with 1 credit and 8 parallel sends delivers high throughput in small systems, and experiences performance degradation in larger ones.

The plot in Figure 3.18 shows the results obtained simulating the second of the two aforementioned scenarios. The results under those different testing conditions show a more irregular behaviour, with a high throughput variability especially in the 16 credits 8 parallel sends configuration. This behaviour can be explained by the fact that DAQPIPE does not operate in congestion-free condition, and in particular the number of parallel sends generates inter-RU resource contention. Because the two topologies are different it is not surprising that the behaviour under congested conditions is different. It is important to note that the particular particular performance drop, i.e. the one that affects the 8 parallel send 16 credits configuration, it is not easily predictable by a static traffic analysis. When the number of nodes increases the difference between the two topologies becomes more and more negligible and the average EB throughput tends to the same value. On the other hand the configuration with 16 credits and 1 parallel send show a consistent and above 80 $textGb/s$ throughput, and it experiences only a small performance penalty when running with a very low number of terminal nodes.

In conclusion the simulations show that the scalability of DAQPIPE is solid and that the application can deliver more than 80 Gb/s at the scale required by the LHCb DAQ system.

Chapter 4

Parametric model of the LHCb magnet

In order to reconstruct long tracks in the LHCb detector, it is possible to propagate the tracks found in the downstream sub-detector to the upstream ones. In particular, tracks reconstructed in the VELO can be extrapolated to the UT and the SciFi. This method consists of extrapolating the track in the next sub-detector, and then defining a region, called search window, in which possible matching hits are searched. The hits found inside the search window are then processed by the tracking algorithm, which can add the hits to the track or reject them accordingly. The width of the search window has to take into account several effects like deviation in the track trajectory introduce by Multiple Scattering (MS), detector resolution effects and approximated particle propagation models.

The SciFi detector is located downstream respect to the LHCb magnet and therefore charged particles are bent by the interaction with the magnetic field. In order to ensure a proper matching by the tracking algorithm, either the track extrapolation or the search window has to take into account the curvature introduced by the magnet. The first method requires a fast parametric description of the interaction between the magnet and a charged particle of a given momentum, and particle momentum estimation; the latter requires wider search windows, and therefore a more significant number of hits to be tested.

In this chapter, a parametric description of the interaction between charged particles and the magnet of the LHCb experiment will be introduced. This model can be used to predict the trajectory after the magnetic field region, reducing the size of the search windows used in the SciFi. Because the momentum of the particle affects the magnitude of the curvature, an estimation of it has to be provided as an input to the model; in particular, in the SciFi sub-detector, the momentum estimation provided by the UT can be used. It is important to note that the momentum resolution provided UT is $\sigma(p)/p \sim 15 \div 20\,\%$.

4.1 Magnetic field effects and the p_T-kick method

The trajectory of a charged particle of momentum $\overrightarrow{p}$, charge q and velocity $\overrightarrow{v}$ in a static magnetic field $\overrightarrow{B}(\overrightarrow{x})$ is given by:

$$\frac{d\overrightarrow{p}}{dt} = q\overrightarrow{v} \times \overrightarrow{B}(\overrightarrow{x}) \tag{4.1}$$

To solve the differential equation (4.1) requires a local description of the magnetic field, and calculating the resulting integral can be computing intensive depending on the magnetic field functional expression; therefore, High Level Trigger (HLT) tracking algorithms use approximated models to describe the interaction of a charged particle with the magnetic field. In particular, the effect of a magnet between two detectors region can be approximated with the p_T-kick method.

The p_T-kick method consists of assuming the deflection of the particle as an instantaneous change to the momentum of the particle, called kick. This change in the particle's trajectory is applied at the centre of the magnet, and its value $\Delta\overrightarrow{p}$ is given by:

$$\Delta\overrightarrow{p} = q\int_L d\overrightarrow{l} \times \overrightarrow{B} \tag{4.2}$$

Where L is the trajectory of the particle. Because in the LHCb magnet $B_y \gg B_x$, and B_z is directed alongside the flight direction of the particle, the equation can be reduced to:

$$\Delta p_x = p\left(\frac{t_{x,f}}{\sqrt{1+t_{x,f}^2+t_{y,f}^2}} - \frac{t_{x,i}}{\sqrt{1+t_{y,i}^2+t_{x,i}^2}}\right) = q\int_L |d\overrightarrow{l} \times \overrightarrow{B}|_x \tag{4.3}$$

Where t_x and t_y are the slopes of the the tracks in the x and y direction respectively. This equation allows calculating the momentum of a charged particle knowing the deflection due to the magnetic field.

In order to predict the trajectory of the particle after the magnet, equation (4.3)

Station	C_B [(MeV/c)$^-$1]
T1	1171.1 ± 0.2
T2	1214.7 ± 0.3
T3	1238.6 ± 0.2

Table 4.1. Measurement of C_B from 5000 MC events, selecting long track of non leptonic particles with $p > 2$ GeV/c and $p_T > 300$ MeV/c

can be approximated by:

$$\Delta s = t_{x,f} - t_{x,i} \simeq \frac{q}{p} \int_L |d\overrightarrow{l} \times \overrightarrow{B}|_x \tag{4.4}$$

The last piece of information requires the integration of the magnetic field itself. As previously mentioned, for HLT track reconstruction purposes, it is not feasible to integrate the exact values of Bx along the track; therefore, a parametric description needs to be produced.

4.2 Parametrisation of the magnetic field

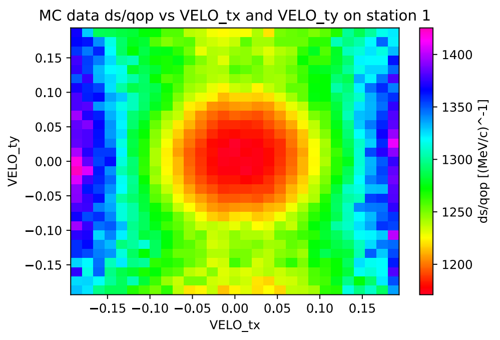
In this section, a parametric description of the action of the magnet in the region between UT and SciFi will be given. The following model has been developed using Monte Carlo (MC) data. In a first approximation, it is possible: to consider $\overrightarrow{B}$ as a perfect dipole, to ignore the fringe fields effects and to neglect the length difference of the various tracks. Under those conditions the magnetic field contribution can be expressed as:

$$\int_L |d\overrightarrow{l} \times \overrightarrow{B}|_x \simeq B(\Delta z)L(\Delta z) \tag{4.5}$$

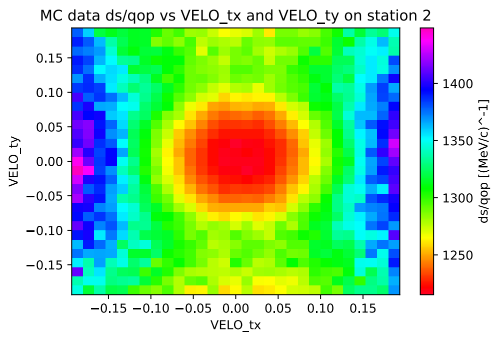
Where $B(\Delta z)$ is the integral of the magnetic field between z_f and z_i, L is the average track length and $\Delta z = z_f - z_i$. From this point onward $B(\Delta z)L(\Delta z)$ will be called $C_B(\Delta z)$.

To determine the magnitude of C_B a sample of 5000 MC events has been used, and only the long tracks of non leptonic particles[1]with $p > 2$ GeV/c and $p_T >$ 300 MeV/c particles were considered. The value of Δs has been calculated between the VELO t_x and the SciFi t_x and it has been normalised over the $\frac{q}{p}$ of the MC generated particle. The combined values of C_B are shown in Table 4.1 for every SciFi station.

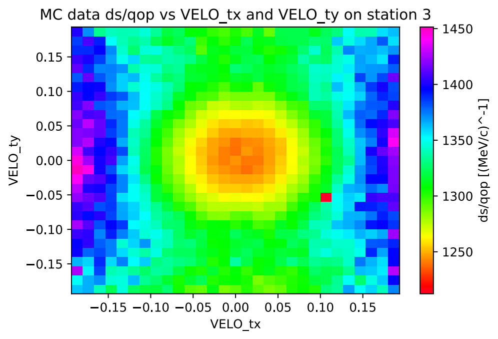
[1]Due to the energy loss introduced by bremsstrahlung radiation, this approximated model cannot describe leptonic particles.



(a) Station 1



(b) Station 2



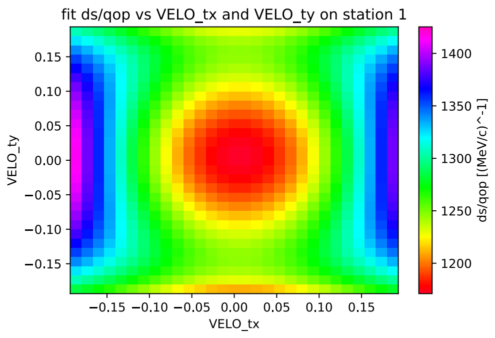
(c) Station 3

Figure 4.1. C_B versus t_x and t_y of the incoming VELO track.

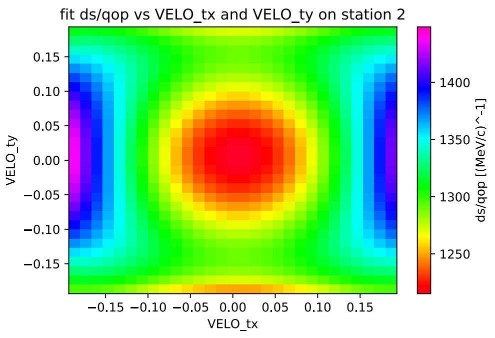
The aforementioned magnetic field approximated model is too simplistic, and it does not describe the behaviour of a particle in the real magnetic field of the LHCb experiment. In particular, the assumption of a magnetic field description which is independent of the particle trajectory is too strict and leads to an incorrect description of the magnetic field bending.

In order to analyse the dependencies of C_B from the VELO trajectory's parameters, (i.e. $t_{x,i}$ and $t_{y,i}$) a bi-dimensional study has been conducted, and it is shown in Figure 4.1. The 2-D plots show an evident and non-trivial dependency between the normalised Δs and the track parameters; any model that does not take this effect in consideration will produce a biased prediction of the particle trajectory, and it will, therefore, result either in a lower reconstruction or computational efficiency.

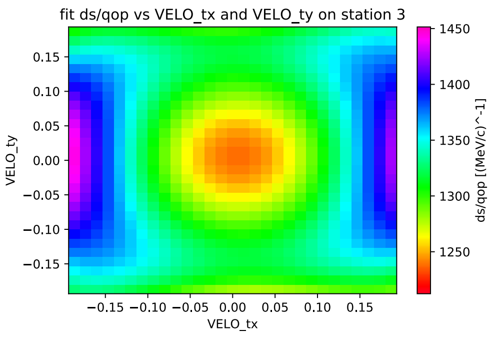
In order to include the $t_{x,i}$ and $t_{y,i}$ dependency into the model, a bi-dimensional polynomial fit has been performed on the 2-D distributions presented in Figure 4.1.



(a) Station 1

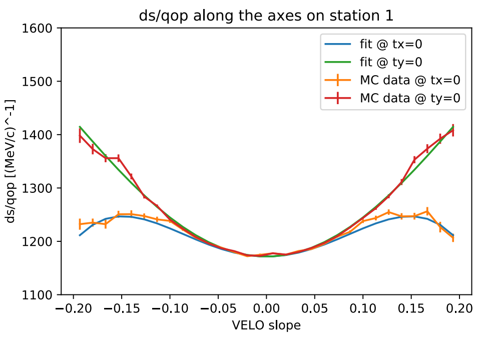


(b) Station 2

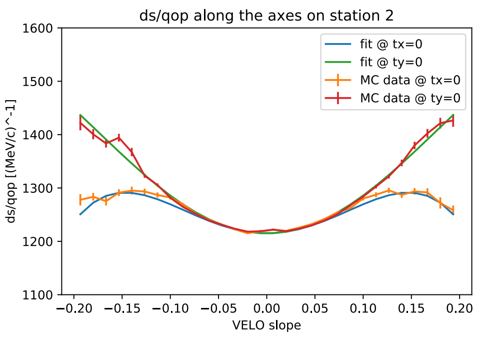


(c) Station 3

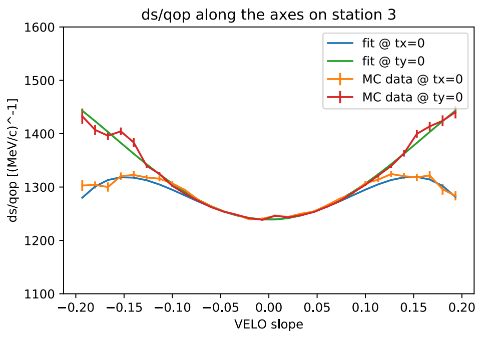
Figure 4.2. C_B versus t_x and t_y of the incoming VELO track.



(a) Station 1



(b) Station 2



(c) Station 3

Figure 4.3. C_B along the t_x and t_y axes.

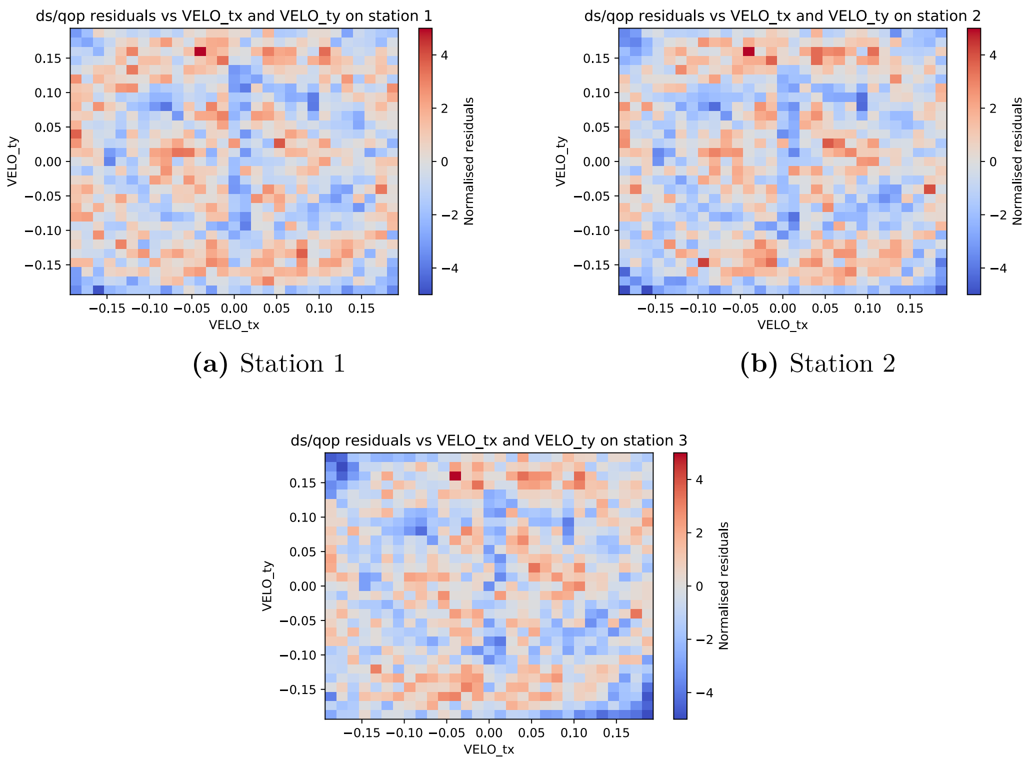
The fit model is a fourth degree polynomial in t_x and t_y:

$$\Delta s \cdot \frac{p}{q} = C_B = \sum_{i=0}^{4}\sum_{j=0}^{4} c_{i,j} t_x^i t_y^j \tag{4.6}$$

Where $c_{i,j}$ are the fit parameters which can be represented as a 5×5 matrix $\mathbf{C}$. Figure 4.2 shows the full model after the fit; Figure 4.3 depicts the fitted model and the C_B calculated from the MC tracks along the t_x and the t_y axes; Figure 4.4 shows the 2-D distribution of the residuals of the fit[2]. This improved model replicates with high fidelity the general structure of the t_x and t_y dependency of C_B.

The model described in (4.6) has 25 free parameters but, by looking at their statistical significance, most of them are compatible with zero. Table 4.2 shows all

[2]The residuals are defined as $\frac{y_{\text{data}}-f(x)}{\sigma(y_{\text{data}})}$, where: y_{data} represents the data points, $\sigma(y_{\text{data}}$ represents the uncertainty on the data points and $f(x)$ is the fit model



(a) Station 1 **(b)** Station 2 **(c)** Station 3

Figure 4.4. Fit residuals versus t_x and t_y of the incoming VELO track.

$(1.1711 \pm 0.0002) \times 10^3$	-4 ± 4	$(6.82 \pm 0.03) \times 10^3$	$(3.9 \pm 3.0) \times 10^2$	$(-1.54 \pm 0.03) \times 10^5$
6 ± 4	$(-1.2 \pm 0.9) \times 10^2$	$(-0.2 \pm 1.0) \times 10^3$	$(1 \pm 5) \times 10^4$	$(0.3 \pm 5.0) \times 10^4$
$(7.64 \pm 0.07) \times 10^3$	$(0.5 \pm 1.0) \times 10^3$	$(-2.5 \pm 0.2) \times 10^5$	$(-4 \pm 7) \times 10^4$	$(3.4 \pm 0.8) \times 10^6$
$(-2 \pm 3) \times 10^2$	$(6 \pm 5) \times 10^3$	$(2 \pm 8) \times 10^4$	$(-6 \pm 2) \times 10^5$	$(0.4 \pm 3.0) \times 10^6$
$(-3.0 \pm 0.3) \times 10^4$	$(-1 \pm 5) \times 10^4$	$(-1.1 \pm 0.5) \times 10^6$	$(1 \pm 3) \times 10^6$	$(6 \pm 3) \times 10^7$

Table 4.2. Fit parameters on station 1. The values of $c_{i,j}$ are expressed in $[(\mathrm{MeV/c})^-1]$ and they are presented in a matrix were the row and columns indicate i and j respectively.

	station 1	station 2	station 3
$c_{0,0}$	$(1.1715 \pm 0.0003) \times 10^3$	$(1.2147 \pm 0.0002) \times 10^3$	$(1.2387 \pm 0.0002) \times 10^3$
$c_{2,0}$	$(7.58 \pm 0.07) \times 10^3$	$(7.51 \pm 0.08) \times 10^3$	$(7.23 \pm 0.08) \times 10^3$
$c_{4,0}$	$(-2.9 \pm 0.2) \times 10^4$	$(-4.2 \pm 0.3) \times 10^4$	$(-4.7 \pm 0.3) \times 10^4$
$c_{0,2}$	$(6.75 \pm 0.08) \times 10^3$	$(7.09 \pm 0.09) \times 10^3$	$(7.25 \pm 0.09) \times 10^3$
$c_{0,4}$	$(1.51 \pm 0.04) \times 10^5$	$(-1.64 \pm 0.03) \times 10^5$	$(-1.64 \pm 0.03) \times 10^5$
$c_{2,2}$	$(-2.5 \pm 0.2) \times 10^5$	$(-3.1 \pm 0.2) \times 10^5$	$(-3.1 \pm 0.2) \times 10^5$
$c_{2,4}$	$(3.8 \pm 0.7) \times 10^5$	$(5.0 \pm 0.8) \times 10^6$	$(4.8 \pm 0.8) \times 10^6$

Table 4.3. Fit parameters for the reduced model on all the SciFi stations. All the values are expressed in [(MeV/c)$^-$1].

the values of the 25 fit parameters with their uncertainties for the fit on Station one. The other stations have similar results; therefore, it is possible to reduce the number o free parameters from 25 to 7 and to simplify the model proposed in (4.6):

$$C_B = c_{0,0} + c_{2,0}t_x^2 + c_{4,0}t_x^4 + c_{0,2}t_y^2 + c_{0,4}t_x^4 + c_{2,2}t_x^2t_y^2 + +c_{2,4}t_x^2t_y^4 \tag{4.7}$$

Table 4.3 contains all the parameter values on all the SciFi stations.

4.3 Test of the parametrisation

The parametric model described in section 4.2 can be used to predict where a charged particle will hit on the SciFi stations. In order to make this prediction possible several conditions must be met:

- The track of the particle has to be reconstructed by the VELO;
- The momentum and charge of the particle has to be known;
- All the model's parameters have to be determined from MC events.

If all the conditions above are verified, then, assuming the validity of the p_T-kick model, the trajectory of the particle can be written as:

$$x(z) = \begin{cases} x_0 + t_{x,v}z & : z < z_m \\ x_0 + t_{x,v}z_m + t_{x,s}(z - z_m) & : z > z_m \end{cases} \tag{4.8}$$

$$y(z) = y_0 + t_{y,v}z \tag{4.9}$$

Where: $t_{x,v}(t_{y,v})$ is the slope of the $x(y)$ coordinate calculated from the VELO track; $t_{x,s}$ is the slope of the x coordinate inside the SciFi and z_m is z coordinate of the kick point. The value of $t_{x,s}$ can be calculated by inverting equation (4.6):

$$t_{x,s} = C_B \cdot \frac{q}{p} + t_{x,v} \tag{4.10}$$

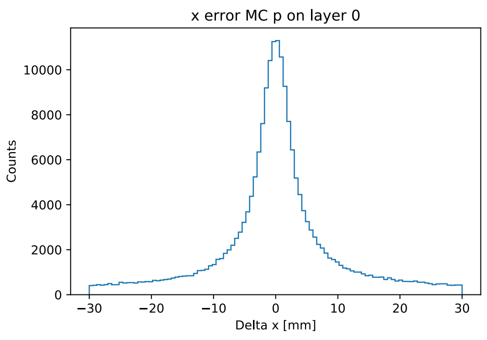
The accuracy of the model can be measured by executing this procedure on MC-generated events:

1. The selection is restricted to long tracks from non leptonic patricles with $p > 2$ GeV/c and $p_T > 300$ MeV/c;
2. The VELO track parameters are calculated;
3. The projection of the particle onto the different SciFi stations is computed
4. The quantity *x error* (i.e. the difference between the predicted x position and the MC truth x position) is calculated

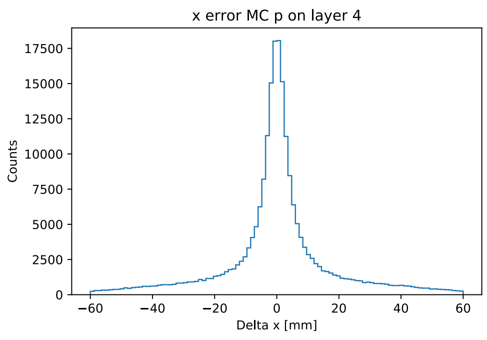
The plots in Figure 4.5 show the distribution of *x error* on a set of 5000 MC event, when the real momentum of the particle is used. Because curvature of the track is proportional to $\frac{1}{p}$ the distribution of *x error* is not expected to be flat in p, the plots in Figure 4.6 show the p dependency of *x error*, the results are obtained by grouping the particles in momentum bins and calculating a 1σ interval in *x error*, which corresponds to the half-width of a search window that includes 68% of the tracks. This model can, therefore, predict the trajectory of a charged particle with a *x error* of less than $50\,mm$ for low momentum particles and less than $5\,mm$ for the high momentum ones.

In a real SciFi tracking scenario, the only momentum estimation available is the one provided by the UT; therefore the evaluation as mentioned above of the parametric model has been conducted by applying a 15 % Gaussian resolution model for the particles' momentum. The *x error* distribution and the momentum-binned one are shown in Figure 4.7 and 4.8 respectively. Under those more realistic conditions, the accuracy of the model's prediction is lowered by a factor four.

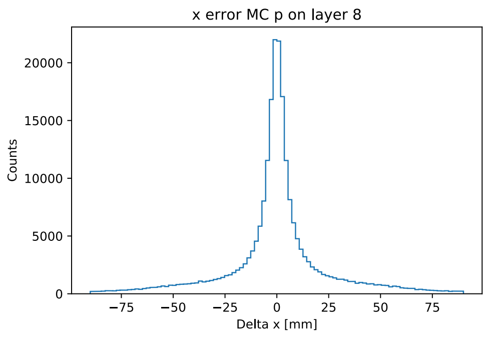
In order to quantify the reduction in the search window size introduced by the track model described in (4.8), it is useful to evaluate the *x error* distribution when a linear model is used. The plots in Figure 4.9 show the deflection introduced by the magnet field on positively charged particles. The plots in Figure 4.10 show the p dependency of the deflection; because the deflection in the momentum bins is



(a) Station 1



(b) Station 2

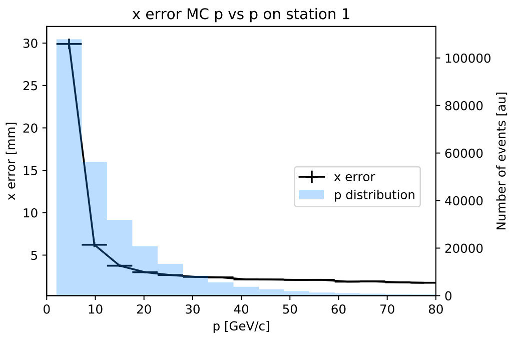


(c) Station 3

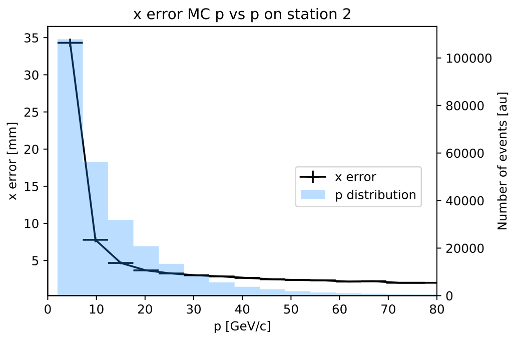
Figure 4.5. Distribution of *x error* on the first x layer of every station. The momentum is extracted from the MC truth.

non-Gaussian, in order to provide numbers that are comparable with the ones in Figure 4.6 and 4.8, the half-width of an asymmetric window which includes 68% of the tracks has been calculated.

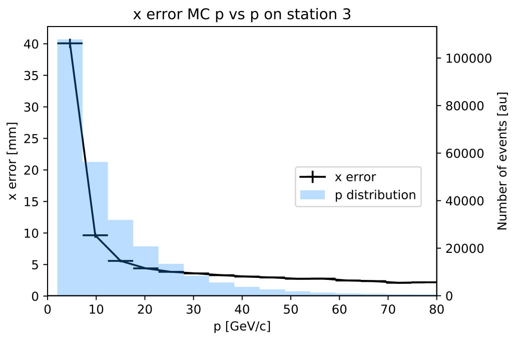
In conclusion, the model, as mentioned earlier, allow for a reduction by a factor 4 in the size of the search windows needed to reconstruct long tracks in the SciFi sub-detector.



(a) Station 1

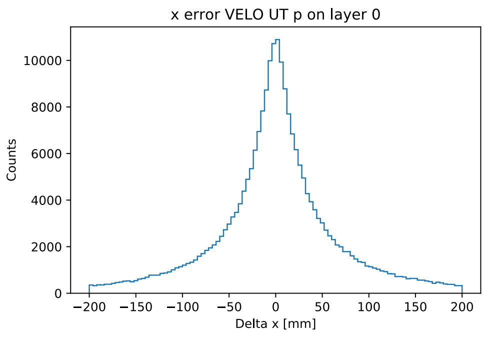


(b) Station 2

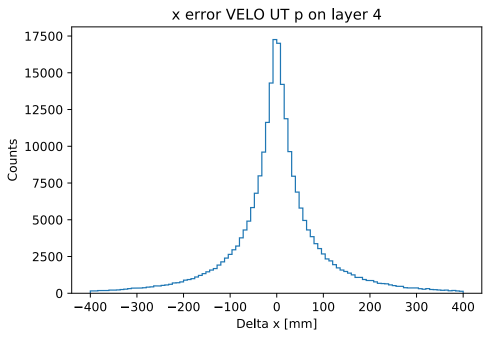


(c) Station 3

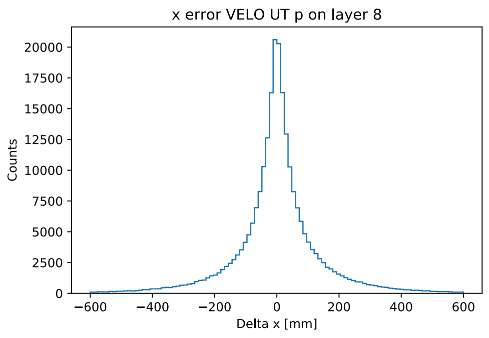
Figure 4.6. The x error vs the momentum of the track on the first x layer of every station. The light blue histogram represents the momentum distribution of the particles. The momentum is extracted from the MC truth.



(a) Station 1

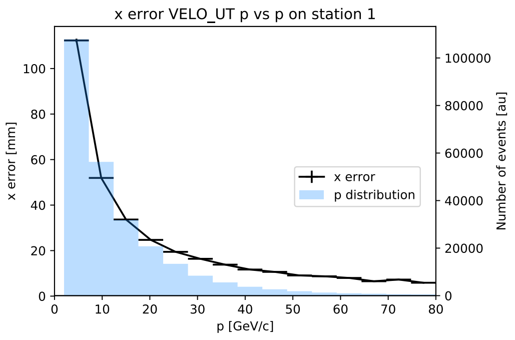


(b) Station 2

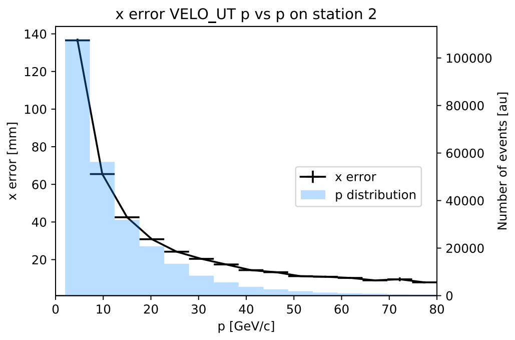


(c) Station 3

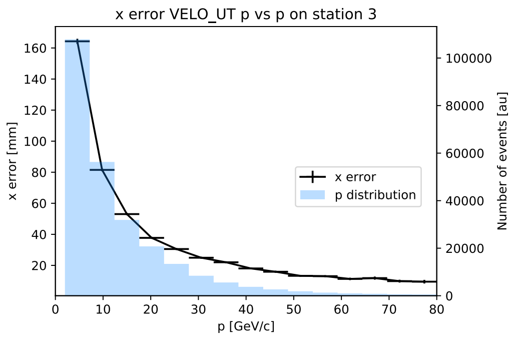
Figure 4.7. Distribution of *x error* on the first x layer of every station. The momentum is extracted with an unbiased 15 % resolution from the MC truth.



(a) Station 1

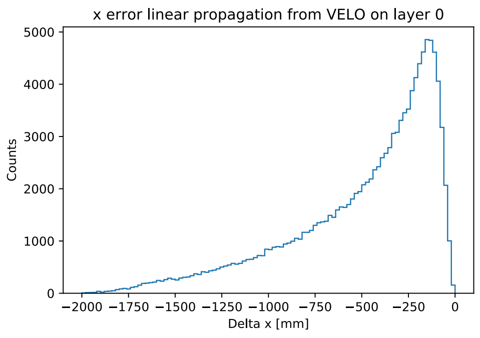


(b) Station 2

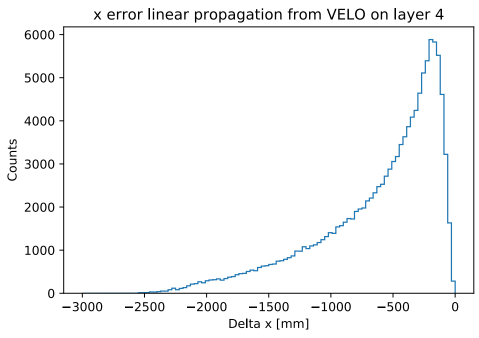


(c) Station 3

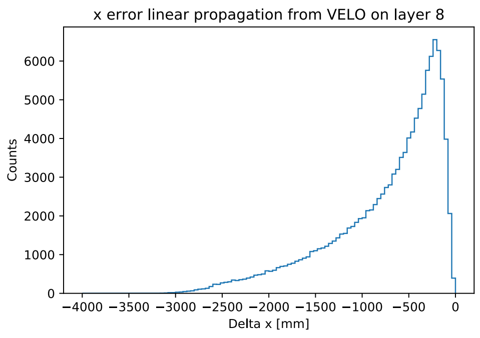
Figure 4.8. *x error* vs the momentum of the track on the first x layer of every station. The light blue histogram represents the momentum distribution of the particles. The momentum is extracted with an unbiased 15 % resolution from the MC truth.



(a) Station 1

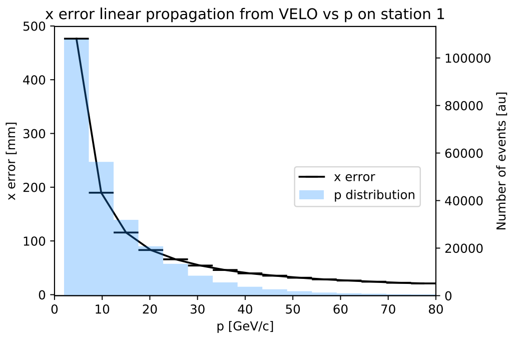


(b) Station 2

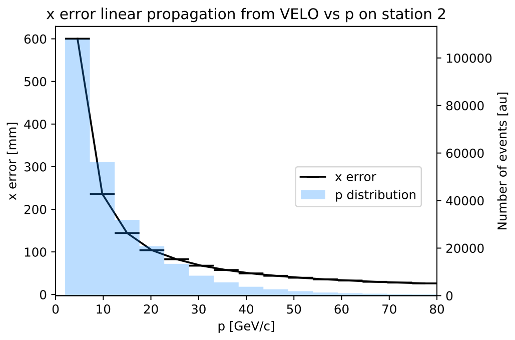


(c) Station 3

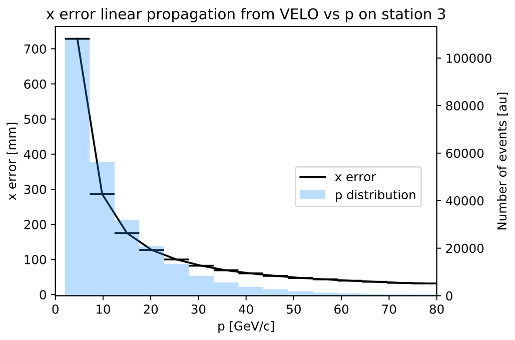
Figure 4.9. Distribution of *x error* on the first x layer of every station for a linear extrapolation from the VELO, for positive particles.



(a) Station 1



(b) Station 2



(c) Station 3

Figure 4.10. *x error* vs the momentum of the track on the first x layer of every station for a linear extrapolation from the VELO, for positive particles. The light blue histogram represents the momentum distribution of the particles.

Conclusions

This thesis presents two technological problems introduced by the working conditions of the LHCb experiment during Run 3: the higher throughput required by the DAQ network infrastructure, and the higher compute power needed to process the full collision rate produced.

The challenging design aspects of a 32 Tb/s DAQ system are analysed. Possible solutions are proposed via extensive use of the linear shifting traffic scheduling, and by requiring more sophisticated flow control capabilities in the network. A low-level simulation model is developed to test the performance of the system before the actual procurement. Measurements conducted on real systems validate this model, and the simulation results can replicate the measured data with a 20% accuracy. Full-scale simulations of the DAQPIPE event building benchmark show that it is possible to achieve the required throughput of 80 Gb/s per node. Simulations of the a2a benchmark show a perfect scalability under the optimal network configuration. In conclusion, both strategies can be used to implement the event building network of the LHCb experiment.

To lower the computing power needed to perform the track reconstruction at the full 30 MHz *pp* collision rate, a new parametric description of the interaction between charged particles and the magnet of the LHCb experiment is given. This fast model derived from MC generated events can be used to predict the particle trajectory in the SciFi sub-detector. The accuracy of the model is tested against MC data under different conditions. If the real momentum of the particle is used the model can predict the particle trajectory with an error of less than 50 mm for low momentum particles and less than 5 mm for the high momentum ones. If the momentum estimation provided by the UT is used the accuracy of the simulation model is lowered by a factor 4. A comparison between this model and a linear extrapolation from the VELO shows a potential reduction by a factor four of the search windows needed to reconstruct long tracks in the SciFi sub-detector. This model is successfully used by the `LookingForward` algorithm used in the Allen framework.

Bibliography

[1] F. J. Hasert et al. "Observation of Neutrino Like Interactions Without Muon Or Electron in the Gargamelle Neutrino Experiment". In: *Phys. Lett.* 46B (1973), pp. 138–140. DOI: 10.1016/0370-2693(73)90499-1.

[2] G. Arnison et al. "Experimental observation of isolated large transverse energy electrons with associated missing energy at s=540 GeV". In: *Physics Letters B* 122.1 (Feb. 24, 1983), pp. 103–116. ISSN: 0370-2693. DOI: 10.1016/0370-2693(83)91177-2. URL: http://www.sciencedirect.com/science/article/pii/0370269383911772 (visited on 10/30/2019).

[3] D. Decamp et al. "A Precise Determination of the Number of Families With Light Neutrinos and of the Z Boson Partial Widths". In: *Phys. Lett.* B235 (1990), pp. 399–411. DOI: 10.1016/0370-2693(90)91984-J.

[4] G. B. Andresen et al. "Trapped antihydrogen". In: *Nature* 468.7324 (2010), pp. 673–676. ISSN: 1476-4687. DOI: 10.1038/nature09610. URL: https://doi.org/10.1038/nature09610.

[5] J.R Batley et al. "A precision measurement of direct CP violation in the decay of neutral kaons into two pions". In: *Physics Letters B* 544.1 (Sept. 2002), pp. 97–112. ISSN: 0370-2693. DOI: 10.1016/s0370-2693(02)02476-0. URL: http://dx.doi.org/10.1016/S0370-2693(02)02476-0.

[6] Georges Aad et al. "Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC". In: *Phys. Lett.* B716 (2012), pp. 1–29. DOI: 10.1016/j.physletb.2012.08.020. arXiv: 1207.7214 [hep-ex].

[7] Serguei Chatrchyan et al. "Observation of a New Boson at a Mass of 125 GeV with the CMS Experiment at the LHC". In: *Phys. Lett.* B716 (2012), pp. 30–61. DOI: 10.1016/j.physletb.2012.08.021. arXiv: 1207.7235 [hep-ex].

[8] LHCb Collaboration. “Observation of $J/\psi p$ Resonances Consistent with Pentaquark States in $\Lambda_b^0 \to J/\psi K^- p$ Decays”. In: *Phys. Rev. Lett.* 115.7 (Aug. 12, 2015), p. 072001. DOI: 10.1103/PhysRevLett.115.072001. URL: https://link.aps.org/doi/10.1103/PhysRevLett.115.072001 (visited on 10/30/2019).

[9] Roel Aaij et al. “Observation of CP Violation in Charm Decays”. In: *Phys. Rev. Lett.* 122.21 (2019), p. 211803. DOI: 10.1103/PhysRevLett.122.211803.

[10] The LHCb Collaboration et al. “The LHCb Detector at the LHC”. In: *Journal of Instrumentation* 3.8 (2008), S08005.

[11] LHCb Collaboration. “Measurement of the b-Quark Production Cross Section in 7 and 13 TeV pp Collisions”. In: *Phys. Rev. Lett.* 118.5 (Feb. 3, 2017), p. 052002. DOI: 10.1103/PhysRevLett.118.052002. URL: https://link.aps.org/doi/10.1103/PhysRevLett.118.052002 (visited on 10/13/2019).

[12] LHCb collaboration. “Prompt charm production in pp collisions at sqrt(s)=7 TeV”. In: *Nuclear Physics B* 871.1 (June 2013), pp. 1–20. ISSN: 05503213. DOI: 10.1016/j.nuclphysb.2013.02.010. arXiv: 1302.2864. URL: http://arxiv.org/abs/1302.2864 (visited on 10/13/2019).

[13] LHCb collaboration. “Measurements of prompt charm production cross-sections in pp collisions at $\sqrt{s} = 13$ TeV”. In: *J. High Energ. Phys.* 2017.5 (May 2017), p. 74. ISSN: 1029-8479. DOI: 10.1007/JHEP05(2017)074. arXiv: 1510.01707. URL: http://arxiv.org/abs/1510.01707 (visited on 10/13/2019).

[14] LHCb Collaboration. “LHCb VELO Upgrade Technical Design Report”. In: (2013).

[15] LHCb Collaboration. *LHCb Tracker Upgrade Technical Design Report.* CERN-LHCC-2014-001. LHCB-TDR-015. Feb. 2014. URL: https://cds.cern.ch/record/1647400.

[16] LHCb Collaboration. *LHCb magnet: Technical Design Report.* Geneva: CERN, 2000. URL: https://cds.cern.ch/record/424338.

[17] Rudolph Emil Kalman. “A New Approach to Linear Filtering and Prediction Problems”. In: *Transactions of the ASME–Journal of Basic Engineering* 82 (Series D 1960), pp. 35–45.

[18] LHCb Collaboration. *LHCb PID Upgrade Technical Design Report.* CERN-LHCC-2013-022. LHCB-TDR-014. Nov. 2013. URL: https://cds.cern.ch/record/1624074.

[19] LHCb Collaboration. *LHCb RICH: Technical Design Report.* Technical Design Report LHCb. Geneva: CERN, 2000. URL: `https://cds.cern.ch/record/494263`.

[20] LHCb Collaboration. *LHCb calorimeters: Technical Design Report.* Technical Design Report LHCb. Geneva: CERN, 2000. URL: `https://cds.cern.ch/record/494264`.

[21] LHCb Collaboration. *LHCb muon system: Technical Design Report.* Technical Design Report LHCb. Geneva: CERN, 2001. URL: `https://cds.cern.ch/record/504326`.

[22] Jose Duato, Sudhakar Yalamanchili, and Ni Lionel. *Interconnection Networks: An Engineering Approach.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002. ISBN: 1-55860-852-4.

[23] William James Dally and Brian Patrick Towles. *Principles and Practices of Interconnection Networks.* Elsevier, Mar. 6, 2004. 582 pp. ISBN: 978-0-08-049780-8.

[24] Ni. "Issues in designing truly scalable interconnection networks". In: *1996 Proceedings ICPP Workshop on Challenges for Parallel Processing.* 1996 Proceedings ICPP Workshop on Challenges for Parallel Processing. Aug. 1996, pp. 74–83. DOI: `10.1109/ICPPW.1996.538592`.

[25] Clark David Thompson. "A Complexity Theory for VLSI". PhD thesis. Pittsburgh, PA, USA: Carnegie Mellon University, 1980.

[26] The MIT Press. *Complexity Issues in VLSI.* The MIT Press. URL: `https://mitpress.mit.edu/books/complexity-issues-vlsi` (visited on 09/10/2019).

[27] P.T. Gaughan and S. Yalamanchili. "Adaptive routing protocols for hypercube interconnection networks". In: *Computer* 26.5 (May 1993), pp. 12–23. ISSN: 0018-9162, 1558-0814. DOI: `10.1109/2.211888`.

[28] J. Duato. "A necessary and sufficient condition for deadlock-free routing in cut-through and store-and-forward networks". In: *IEEE Transactions on Parallel and Distributed Systems* 7.8 (Aug. 1996), pp. 841–854. ISSN: 1045-9219, 1558-2183, 2161-9883. DOI: `10.1109/71.532115`.

[29] S. Warnakulasuriya and T. M. Pinkston. "Characterization of deadlocks in interconnection networks". In: *Proceedings 11th International Parallel Processing Symposium.* Proceedings 11th International Parallel Processing Symposium. Apr. 1997, pp. 80–86. DOI: `10.1109/IPPS.1997.580852`.

[30] S. Konstantinidou and L. Snyder. “The Chaos router”. In: *IEEE Transactions on Computers* 43.12 (Dec. 1994), pp. 1386–1397. ISSN: 0018-9340, 1557-9956, 2326-3814. DOI: 10.1109/12.338098.

[31] *LHCb Trigger and Online Upgrade Technical Design Report.* CERN-LHCC-2014-016. LHCB-TDR-016. May 2014. URL: https://cds.cern.ch/record/1701361.

[32] D. P. Bertsekas et al. “Optimal communication algorithms for hypercubes”. In: *Journal of Parallel and Distributed Computing* 11.4 (Apr. 1, 1991), pp. 263–275. ISSN: 0743-7315. DOI: 10.1016/0743-7315(91)90033-6. URL: http://www.sciencedirect.com/science/article/pii/0743731591900336 (visited on 10/05/2019).

[33] Naijie Gu. “Efficient indirect all-to-all personalized communication on rings and 2-D tori”. In: *J. Comput. Sci. & Technol.* 16.5 (Sept. 1, 2001), pp. 480–483. ISSN: 1860-4749. DOI: 10.1007/BF02948967. URL: https://doi.org/10.1007/BF02948967 (visited on 10/05/2019).

[34] Vassilios V. Dimakopoulos and Nikitas J. Dimopoulos. “A Theory for Total Exchange in Multidimensional Interconnection Networks”. In: *IEEE Trans. Parallel Distrib. Syst.* 9.7 (July 1998), pp. 639–649. ISSN: 1045-9219. DOI: 10.1109/71.707541. URL: http://dx.doi.org/10.1109/71.707541 (visited on 10/05/2019).

[35] S. H. Bokhari. “Multiphase complete exchange: a theoretical analysis”. In: *IEEE Transactions on Computers* 45.2 (Feb. 1996), pp. 220–229. DOI: 10.1109/12.485374.

[36] Yu-Chee Tseng et al. “Bandwidth-Optimal Complete Exchange on Wormhole-Routed 2D/3D Torus Networks: A Diagonal-Propagation Approach”. In: *IEEE Trans. Parallel Distrib. Syst.* 8.4 (Apr. 1997), pp. 380–396. ISSN: 1045-9219. DOI: 10.1109/71.588613. URL: https://doi.org/10.1109/71.588613 (visited on 10/05/2019).

[37] Christina Christara, Xiaoliang Ding, and Ken Jackson. “An Efficient Transposition Algorithm for Distributed Memory Computers”. In: *High Performance Computing Systems and Applications.* Ed. by Andrew Pollard, Douglas J. K. Mewhort, and Donald F. Weaver. The International Series in Engineering and Computer Science. Boston, MA: Springer US, 2000, pp. 349–370. ISBN: 978-0-

306-47015-8. DOI: 10.1007/0-306-47015-2_38. URL: https://doi.org/10.1007/0-306-47015-2_38 (visited on 10/05/2019).

[38] Young-Joo Suh and Sudhakar Yalamanchili. "All-To-All Communication with Minimum Start-Up Costs in 2D/3D Tori and Meshes". In: *IEEE Trans. Parallel Distrib. Syst.* 9.5 (May 1998), pp. 442–458. ISSN: 1045-9219. DOI: 10.1109/71.679215. URL: https://doi.org/10.1109/71.679215 (visited on 10/05/2019).

[39] A. Faraj, X. Yuan, and P. Patarasuk. "A Message Scheduling Scheme for All-to-All Personalized Communication on Ethernet Switched Clusters". In: *IEEE Transactions on Parallel and Distributed Systems* 18.2 (Feb. 2007), pp. 264–276. ISSN: 1045-9219. DOI: 10.1109/TPDS.2007.19.

[40] D. H. Cámpora Pérez, R. Schwemmer, and N. Neufeld. "Protocol-Independent Event Building Evaluator for the LHCb DAQ System". In: *IEEE Transactions on Nuclear Science* 62.3 (June 2015), pp. 1110–1114. ISSN: 0018-9499. DOI: 10.1109/TNS.2015.2428891.

[41] Flavio Pisani, Daniel Hugo Cámpora Pérez, and Niko Neufeld. "High-speed zero-copy data transfer for DAQ applications". In: *Journal of Physics: Conference Series* 608.1 (2015), p. 012029. URL: http://stacks.iop.org/1742-6596/608/i=1/a=012029.

[42] Adam Otto et al. "A first look at 100 Gbps LAN technologies, with an emphasis on future DAQ applications." In: *Journal of Physics: Conference Series* 664.5 (2015), p. 052030. URL: http://stacks.iop.org/1742-6596/664/i=5/a=052030.

[43] InfiniBand SM Trade Association. *InfiniBand Architecture Specification Annex A 16: RoCE.* 2010.

[44] InfiniBand SM Trade Association. *InfiniBand Architecture Specification Annex A 17: RoCEv2.* 2014.

[45] *802.3-2018 - IEEE Standard for Ethernet - IEEE Standard.* URL: https://ieeexplore.ieee.org/document/8457469 (visited on 10/10/2019).

[46] C. Clos. "A study of non-blocking switching networks". In: *The Bell System Technical Journal* 32.2 (Mar. 1953), pp. 406–424. DOI: 10.1002/j.1538-7305.1953.tb01433.x.

[47] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. “A Scalable, Commodity Data Center Network Architecture”. In: *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication.* SIGCOMM ’08. event-place: Seattle, WA, USA. New York, NY, USA: ACM, 2008, pp. 63–74. ISBN: 978-1-60558-175-0. DOI: `10.1145/1402958.1402967`. URL: `http://doi.acm.org/10.1145/1402958.1402967` (visited on 10/11/2019).

[48] C. E. Leiserson. “Fat-trees: Universal networks for hardware-efficient supercomputing”. In: *IEEE Transactions on Computers* C-34.10 (Oct. 1985), pp. 892–901. DOI: `10.1109/TC.1985.6312192`.

[49] M. D. Schroeder et al. “Autonet: a high-speed, self-configuring local area network using point-to-point links”. In: *IEEE Journal on Selected Areas in Communications* 9.8 (Oct. 1991), pp. 1318–1335. DOI: `10.1109/49.105178`.

[50] Xuan-Yi Lin, Yeh-Ching Chung, and Tai-Yi Huang. “A multiple LID routing scheme for fat-tree-based InfiniBand networks”. In: *18th International Parallel and Distributed Processing Symposium, 2004. Proceedings.* 18th International Parallel and Distributed Processing Symposium, 2004. Proceedings. Apr. 2004, pp. 11–. DOI: `10.1109/IPDPS.2004.1302913`.

[51] Zahavi Eitan et al. “Optimized InfiniBandTM fat-tree routing for shift all-to-all communication patterns”. In: *Concurrency and Computation: Practice and Experience* 22.2 (), pp. 217–231. DOI: `10.1002/cpe.1527`. URL: `https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.1527`.

[52] S. R. Ohring et al. “On generalized fat trees”. In: *Proceedings of 9th International Parallel Processing Symposium.* Proceedings of 9th International Parallel Processing Symposium. Apr. 1995, pp. 37–44. DOI: `10.1109/IPPS.1995.395911`.

[53] Elias Weingärtner, Hendrik Vom Lehn, and Klaus Wehrle. “A Performance Comparison of Recent Network Simulators”. In: *Proceedings of the 2009 IEEE International Conference on Communications.* ICC’09. event-place: Dresden, Germany. Piscataway, NJ, USA: IEEE Press, 2009, pp. 1287–1291. ISBN: 978-1-4244-3434-3. URL: `http://dl.acm.org/citation.cfm?id=1817271.1817510`.

[54] Atta ur Rehman Khan, Sardar Muhammad Bilal, and Mazliza Othman. “A Performance Comparison of Network Simulators for Wireless Networks”. In: *CoRR* abs/1307.4129 (2013). URL: `http://arxiv.org/abs/1307.4129`.

[55] András Varga et al. “The OMNeT++ discrete event simulation system”. In: *Proceedings of the European simulation multiconference (ESM’2001)*. Vol. 9. sn, 2001, p. 65.

[56] András Varga and Rudolf Hornig. “An Overview of the OMNeT++ Simulation Environment”. In: *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*. Simutools ’08. event-place: Marseille, France. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, 60:1–60:10. ISBN: 978-963-9799-20-2. URL: `http://dl.acm.org/citation.cfm?id=1416222.1416290`.

[57] Tommaso Colombo et al. “Flit-Level InfiniBand Network Simulations of the DAQ System of the LHCb Experiment for Run-3”. In: *IEEE Transactions on Nuclear Science* 66.7 (July 2019), pp. 1159–1164. ISSN: 0018-9499, 1558-1578. DOI: `10.1109/TNS.2019.2905993`.

[58] Paul Grun. “Introduction to infiniband for end users”. In: *White paper, InfiniBand Trade Association* (2010).

[59] InfiniBand SM Trade Association. *InfiniBand Architecture Specification Volume 1 and 2*. 2015. URL: `http://www.infinibandta.org/content/pages.php?pg=technology_public_specification`.

[60] “IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems”. In: *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)* (July 2008), pp. 1–300. DOI: `10.1109/IEEESTD.2008.4579760`.