On Software Defined Networks for Particle Accelerators

Eray Duran & George Caraus
`wir14edu@student.lu.se & wir15gca@student.lu.se`

Department of Electrical and Information Technology
Lund University

Advisor:
Jens A Andersson
Anders J Johansson
Examiner:
Maria Kihl

February 18, 2018

Printed in Sweden
E-huset, Lund, 2018

Abstract

Particle accelerators and their connected laboratories for scientific experiments are equipped with a multitude of data acquisition nodes. Example of such site is the European Spallation Source (ESS), based in Lund, Sweden. The nodes are connected to a communication network, designed for quick data transport in the normal operating mode. In case of emergency or malfunction, the accelerator has to be halted and then restarted. Data produced from the nodes is valuable, so it has to be stored during the restart time.

The data communication network adapted for normal production mode is not suitable for such emergency situation. Software Defined Networking (SDN) is paradigm in networking that has gained a great attention during the years and is a promising technique that has many features over the traditional network.

In this Master's thesis work we give a theoretical information for SDN and present SDN architectures in case of a failure of the particle accelerator. A general network architecture for emergency situation is presented, where different use cases for the network behavior are shown.

Popular Science Summary

The particle accelerator is a machine that accelerates electrons or protons at very high energies and is used to investigate different aspects of particle physics. These facilities are used for new discoveries in different areas such as science or medicine.

The European Spallation Source (ESS) will be world's largest pulsed neutron source and will contribute to the ongoing research in fields like life science, cultural heritage and fundamental physics until 2065. During the production mode, malfunctions may occur and in this case, the accelerator requires a shut down and the huge amount of the produced data needs to be stored for later analysis.

Traditional data management tools can't not be used for Big-Data management because of it's volume and complexity. To overcome the drawbacks of the traditional network architecture, a Software Defined Network (SDN) architecture is introduced. The major change is the decoupling of control and data plane and thus having a programmable network that would make it easier to switch from normal mode to emergency mode.

Since there is no knowledge of ESS's network topology, the normal production mode architecture is based on assumption after discussions with our supervisors.

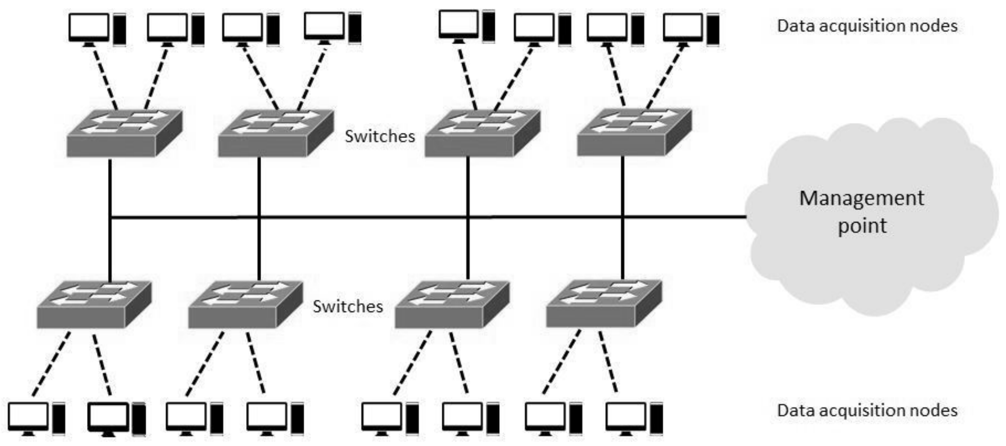

Figure 0.1: Normal production mode

The thesis proposes an SDN network architecture that supports Big-Data applications. This solution offers more flexibility and a global view of the network by having the SDN controller as the "brain" of the network. The application layer will receive information about the network status from the controller. Furthermore, the controller will forward the application's requests to the data layer.

In case of a malfunction, the controller is informed and changes the network architecture from normal production mode to the emergency mode, which is presented in the below picture.

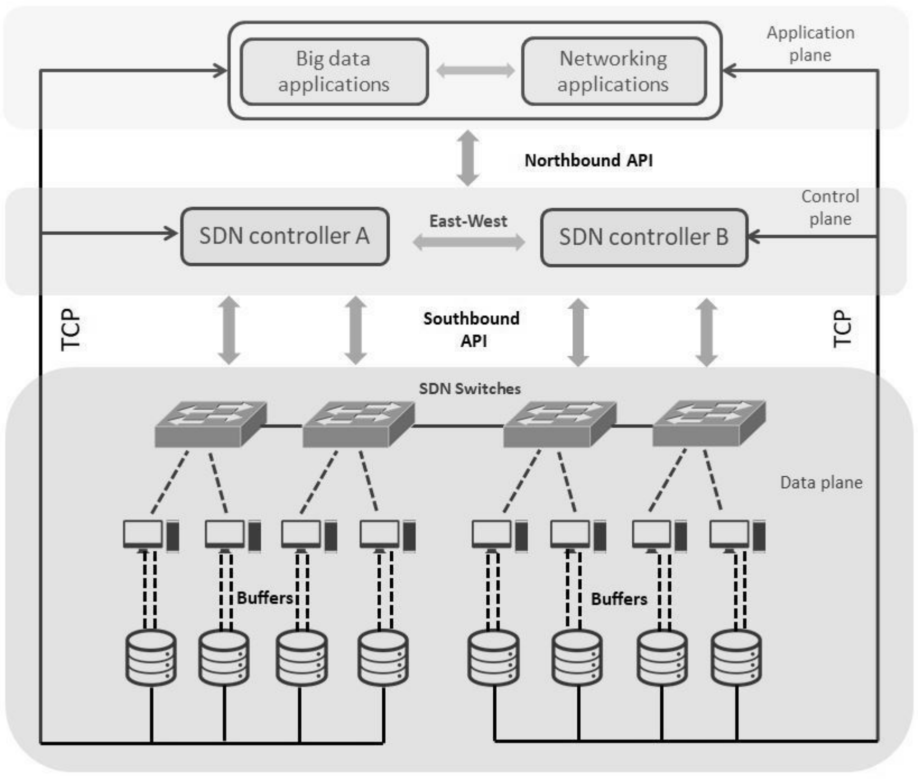

Figure 0.2: Particle accelerators architecture in emergency mode

The suggested solution with a buffer attached to each accelerator node is a reliable one for any particle accelerator, since we are able to meet the requested performance requirements such as:

- Reliability - no data should be lost
- Scalability - the network should scale if the number of data acquisition node is increased
- Time sensitivity - the data should be saved and stored as soon as possible during the restart time of the facility.

The presentation of the network architecture model is supported by different use cases. We define network parameters where the assigned values were estimated. For better understanding of the network’s bottlenecks graphical representations are shown.

Acknowledgemets

This thesis work was carried out at the Department of Electrical and Information Technology at Lund University, Sweden.

We would like to express our deepest gratitude to all those who confirmed the permission and thus made it possible to complete the thesis work at Lund University.

Most notably, we would like to convey our thanks to our project supervisor, Jens A Andersson for his endless support, advice and guidance throughout the period of the thesis.

We would like to specially thank to our technical advisor Saeed Bastani who has shared his knowledge and experience with us, has been supporting us and helped us throughout our project with his best.

We would like to thank Anders J Johansson who has been our contact person with ESS.

Eray Duran, George Caraus
Lund, 2018

Contents

List of Acronyms

- APIs - Application Programming Interfaces
- BLED - Beam Line Element Databases
- CERN - European Organization for Nuclear Research
- EB - Exabyte
- EIGRP - Enhanced Interior Gateway Routing Protocol
- EIT - Electrical and Information Technology
- ESS - European Spallation Source
- HMI - Human Machine Interface
- I/O - Input/Output
- ICS - Integrated Control System
- IDC - International Data Corporation
- IT - Information Technology
- IoT - Internet of Things
- KPRs - Key Performance Requirements
- LHC - Large Hadron Collider
- MPS - Machine Protection System
- NAPI - Northbound Application Programming Interface
- OF - OpenFlow
- ONF - Open Networking Foundation
- ONOS - Open Network Operating System
- PB - Petabyte
- PLC - Programmable Logic Controller
- PPS - Personnel Protection System

- SAPI - Southbound Application Programming Interface
- SDN-NGeNIA - SDN Next Generation Integrated Architecture
- SDN - Software Defined Network
- TCP - Transmission Control Protocol

List of Figures

List of Tables

Chapter 1

Introduction

This chapter gives the reader a general overview of our project. It starts with a background information for particle accelerators and Software Defined Network (SDN). The main problems of our thesis are highlighted to give the auditory better understanding of the goals. The chapter ends with a brief description of how we approached our work and with a thesis outline.

1.1 Background

The particle accelerator is one of the marvelous instruments that has been ever built by physicist and represents a machine that accelerates electrons or protons at very high energies. Initially constructed to broaden our knowledge in physics and astronomy, numerous accelerators are used nowadays in many different research areas such as medicine, industrial processing, molecular sciences, life sciences. Those powerful facilities give the scientist the ability to look deep inside the materials and study them in finest detail.

The European Spallation Source (ESS) is expected to be the world's most powerful neutron source intended for engineering and life sciences. It is being built in Lund, Sweden and therefore it is a case of interest in our work. Another particle accelerator that is used as an example in this project is the Large Hadron Collider (LHC).

Particle accelerators are establishments with high complexity and malfunctions are expected. Huge amount of valuable data with high intensity are produced, which means that those facilities must be equipped with a reliable communication network in order to process this important information.The operation of the accelerator is stopped if a malfunction occurs, which means that we need to store the generated information. The peculiarity of the produced data makes the traditional networking not the best solution.

SDN turns to be quite good approach which overcomes the limitations of the traditional networking architecture and allows for quick redesign of the whole network once the accelerator is halted.

1.2 Problem Definition

The aim with this Master's thesis work is to make a theoretical study and present a general architecture case of SDN for particle accelerators. In this study we investigate how a SDN could be a profitable solution and could help to collect and save all the data from the nodes, when the accelerator is in restart phase due to malfunction.

We need to first identify key performance requirements (KPRs) that we need to take into consideration during the network design phase.

Our network should be:

- reliable - no data should be lost before it is processed
- scalable - if we increase the number of our data acquisition points (nodes), our architecture should be able to support them
- time sensitive - we need to process the data without excessive delay as fast as possible.

The nature of the data that is produced in this facility fits in the definition of "Big-Data". A traditional network, where the control and data plane are set within the same network device, is not capable to deal with this case. In contrast, some features of SDN can facilitate Big-Data acquisition. Chapter 3 provides the readers with more detailed information about what Big Data is and how SDN relates to Big Data.

1.3 Related work

SDN has gained a great research attention during the last several years and thus has been implemented and used in different areas of interest.

In 2015, the networking equipment vendor Brocade Communication Systems has been deployed by the CERN's Large Hadron Collider, the world's biggest particle accelerator and started a project where the aim was to develop a long-term software defined networking strategy that can support future infrastructure requirements. It is stated that an overriding issue for CERN is the huge quantity of data that they produce every second and it is a challenge to store and analyze it.[1] A complete scientific report related to the project stated above was not found and there is no information whether the project is still ongoing.

Open Network Operating System (ONOS) represents an open source SDN networking operating system, which is designed for high performance, scale and availability. The emergence of ONOS, SDN and end-to-end operating systems could provide an intelligent and adaptive network that can meet the needs of a fast and efficient data distribution network, required for scientific research, such as the LHC experiments.[2]

In [12], the author presents an SDN enabled architecture that is able to handle the bursting data in Big-Data applications. It is shown that the programmability and flexibility features of SDN can benefit data delivery for Big-Data applications.

The SDN Next Generation Integrated Architecture (SDN-NGeNIA) project addresses the challenges that science facilities are facing when it comes to processing, distribution and analysis of data sets with sizes of petabytes (PB) or even exabytes (EB). The aim of SDN-NGeNIA project is to enable LHC and other science domains to operate with a new level of efficiency and control. This is based on intelligent SDN-driven network systems which are coupled to high throughput applications. [3]

Previous work that is tightly connected to our main goal which is to collect and save all the data before restarting the particle accelerator in case of emergency situation or malfunction is not found.

1.4 Limitations

Here are listed the limitations associated with our work:

- The normal production mode of ESS is based on assumptions after discussions with our supervisors, since access to it was not given.
- Our main solutions that are presented are based on pure theoretical research without any simulations due to the lack of relevant input. Furthermore, the time needed to find relevant information has shortened the time for simulations.
- The values assigned to the parameters in Chapter 4 are based on estimations.

Given the fact that there is almost no work done in this field or even if it is done it is not available to the public, we believe that our work presents useful information that could be further developed.

1.5 Thesis outline

The following section represents an outline of the thesis work which consist of 7 chapters. Brief description of the content in each chapter is presented.

- Chapter 2 Theory - informs the reader about the architecture in normal operation mode, reviews what SDN is and what are the advantages over the traditional network. A brief description of the European Spallation Source, as well as it's role is presented given that it is a case of interest
- Chapter 3 SDN meets Big-Data - gives information what Big-Data is, why we use this characteristics and how SDN is used with Big-Data
- Chapter 4 A general case - in this chapter an architecture that could be used as a possible solution to the addressed main problem is shown.
- Chapter 5 Discussion - contains a discussion about the results and data analysis
- Chapter 6 Conclusion - describes the outcome of the main problem in a short summary
- Chapter 7 Future work - includes suggestions of how this work could be used and developed in the future

1.6 Organization of the project

This Master's thesis was written both by Eray Duran and George Caraus and took place at the Department of Electrical and Information Technology (EIT) at Lund University.

Both the authors took active part in the theoretical investigation process. Eray got acquainted with the SDN part and made a research, whether this could be a solution to the problem formulation and studied the advantages of this type of network over the traditional one. George focused mostly on the Big-Data scenario and the ESS case. He made a detailed study and came to the conclusion that we need to make a general case where Big-Data characteristics should be taken into consideration and that we can use ESS case as an example that suits to the general case.

Chapter 4, was written by Eray, while the rest of the thesis was written by George.

Chapter 2 Theory

A main part of our project was to make a detailed theoretical study and get well acquainted with the particle accelerators and the current status of the networking world.

We begin this chapter with a description of the network architecture behaviour in normal production mode. Traditional network approach is used in this mode which means that software and hardware are coupled together in the network devices. We explain why we need to make a transition to programmable network.

To be able to formulate our main goal and achieve beneficial outcome we finalize this chapter with theoretical information regarding Software Defined Networking and the European Spallation Source.

2.1 Normal production mode

Particle accelerators are complex facilities composed of a lot of self-controlled nodes that send constantly flows of data to a management station. Figure 2.1 shows the network architecture of a particle accelerators when there is no failure and everything works smoothly. As stated in the limitations section, we were not given an access to the exact network topology in the normal production mode. This topology is an assumption based on our early discussions with our supervisor. From this moment on, we base our solution on this assumption.

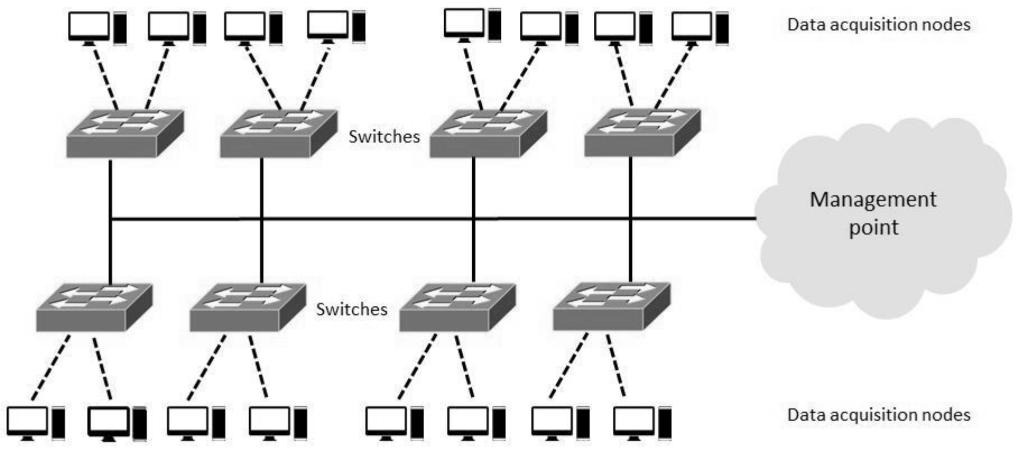

Figure 2.1: Normal production mode

According to [19], during October 2017, CERN’s data centre stored the huge amount of 12.3 petabytes (PB) of data. This came as a result of the experiments that took place at LHC. Furthermore, the volume of stored data has increased at an exponential rate during the past ten years, achieving the milestone of 200 PB stored data in June 2017.[19]

Particle accelerators require high data rates to collect and save all the produced data. The estimated data rate from all four experiments that are running at LHC site is about 25 gigabyte per second (GB/s). [20]

It is clear from the huge amounts of data that is produced and transferred that those facilities are very complex, hard to manage and require more robustness.For example, in case of a overloaded switch, we will lose the information from the attached nodes. In case of a link failure we will also lose the data and with this approach the reaction time is not fast. Therefore, even if we assume that this traditional network approach is able to handle the data transfer process in normal operation mode, it is not suitable in case of a alarm situation. We need to look for another solution that will offer us more robustness, scalability and flexibility and this is why we have turned our attention to a technique called SDN.

2.2 Software Defined Network

The networking world evolves everyday as new devices with new technologies come out. The traditional network infrastructure approach, as shown in the previous sub-chapter, is a complex system of many physical elements on which the information technology (IT) industry has been dependent over the years for communication and services. A small change in any part of the network could be a reason for a failure of the whole system. [4].

To grasp better the SDN concept, the beginning of this sub-chapter introduces the traditional network architecture so that we focus and understand what are the needs for a programmable network and what are the advantages of this new approach.

The sub-chapters later on, inform the readers regarding the SDN devices, controllers and applications.

2.2.1 Transition to the programmable network

Over the years, the conventional networks have been successful in delivering services to the users. However, in our case the number of nodes and data that is produced is big. Therefore, more forwarding devices such as switches, routers and middle-boxes are installed. Since the network is static and adapted only for normal production mode needs, the growth of the network introduced several limitations that quickly became a challenging task to solve when a failure occurs:

- complexity in routing and flow management became a problem since each device operates on a particular network-level protocol and therefore requires an appropriate configuration with each other[8]
- the separate configuration of each device become costly to operate and required highly skilled personnel to maintain the network nodes in case of an error as they typically come from different vendors.

In traditional networks the control and data plane are set within the same network device. The paths for data flows are configured and programmed by the control plane and then that control information is used for data forwarding. The drawback with that method, however, is that the control information cannot be changed unless re-configuration is done. Moreover, the inflexibility is a major drawback of the traditional networking. Nowadays more dynamic user applications like dynamic load-balancing are needed and due to the static nature of the conventional networks the performance is sub-optimal.

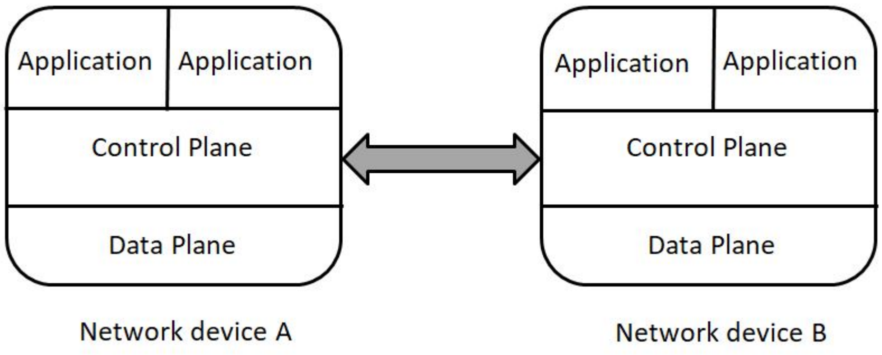

Figure 2.2: Traditional network device architecture

In the last decade companies invested huge amount of money to build a new architecture type that will reduce the cost and offer more flexibility and reliability to meet the current needs. Companies re-built their networks with less expensive and off-the-shelf equipment controlled by software, which is called Software Defined Network[14].

SDN is a more flexible and dynamic approach that has the capability to meet the drawbacks of the traditional networking, described above and is a network type that is cost-effective and provides an overall view of the network with a single point of control.[13]

Particle accelerators are research facilities where data sets are large, complex and are typically not processed, analyzed and visualized at the location that are produced. Due to variety of reasons data is moved to another locations which leads to an even higher network complexity and dependence on a functioning distributed architecture. In such case dynamic traffic change could be required which means that the traditional networking approach would be less reactive and efficient. To overcome the limitations described above, a transition to a next-level network that meets our current needs is inevitable.

2.2.2 SDN - Decoupling the control and data plane

As we have seen above the traditional network was no longer capable to operate today's network efficiently, mainly due to to lack of a common control platform. The researchers proposed to separate the control plane from the data plane.

The decision on how to forward data is taken from the control plane, whereas the data plane is the system which takes care of forwarding the traffic.

The SDN controller is a server that runs SDN software and communicates with the data plane through a protocol called OpenFlow. It can transmit different rules to the data plane according to dynamic needs such as new policies, network events or alerts.

The role of the data plane is simple, since it needs only to transmit a given packet to the next hop by following some simple rules. In this way we can increase the performance of our network which is a critical point, since huge amounts of data should be quickly transported without big delays and packet losses. Complications in network configurations are effectively reduced, since it is only the application used by the controller that needs to be changed and thus canceling the need of implementing different vendor specific settings and commands. Therefore, SDN happens to be a flexible, easy manageable and dynamic system in case of a change in the application requirements.

2.2.3 SDN Architecture

The SDN architecture consists of three layers - application, control and data/infrastructure plane. Application programming interfaces (APIs) are used for cross-layer communication. This architecture is presented in the following Figure 2.3.

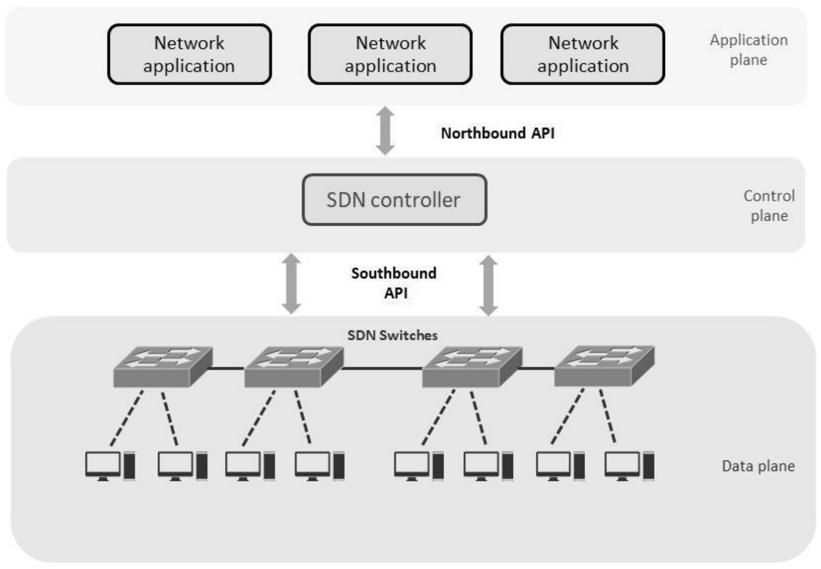

Figure 2.3: SDN architecture

Application plane: The Application plane runs different SDN applications to specify the resources and behavior that they require from the network. They communicate with the SDN controller via the Northern API. The application layer is provided with a network view which is used in decision making process. Examples of some applications are network topology discovery, network resource provisioning, path reservation.

Control plane: The control plane or the SDN controller is the core of the network. It receives instructions from the application plane via the Northbound API (NAPI), processes the information and then transmits it to the data plane via the Southbound API (SAPI).
The controller also has the duty to transports back hardware information

to the application level with the network status - for example statistics and events. The configuration of the devices "below" is done using the OpenFlow protocol.

Data/Infrastructure plane: The data plane consists of SDN enabled network devices such as routers and switches that route the packets according to SDN standards such as having API which allows them to utilize a centralized controller as the brain that governs their actions.

2.2.4 Northern API

In a SDN architecture, the NAPIs are used to communicate between the SDN controller and the services and applications running in the application plane.[15] NAPIs are considered as the most critical ones in a SDN environment, because they must support a wide variety of applications, which means that one particular API could not fit all. Due to that reason, there is no specified standard, meaning that each controller may expose its own API corresponding to the application's specificity.

Applications can use the controller via the NAPI to gather information about the network status or to set new rules throughout the network or to signal the controller for threats.[8]

2.2.5 Southern API

SAPIs are used to communicate between the controller and the switches and routers that are situated in the data plane. This APIs support the control over the network and enable the SDN controller to dynamically make changes according to real-time demands and needs. Unlike the NAPIs, the SAPIs are standardized and the first and most well-known interface is the OpenFlow (OF).

2.2.6 OpenFlow

OF protocol was developed by the Open Networking Foundation (ONF) and is an industry standard that defines the way a SDN controller should interact with the infrastructure plane and adapt the network to changing business requirements. To make the network more responsive to traffic demands, the controller could set rules about forwarding behaviour to the data plane units. Through the OF protocol, the controller could set the routers and switches to add, remove, en-queue and forward a packet belonging to particular flow.

OpenFlow has been supported by a number of switch and router vendors such as Cisco, Juniper, Big Switch Networks, Brocade, Arista, Extreme Networks, Dell, NoviFlow, Hewlett-Packard, NEC.[16] It is important to state that OpenFlow is the first and most widely used SAPI. However, it is not the only one available on the market.

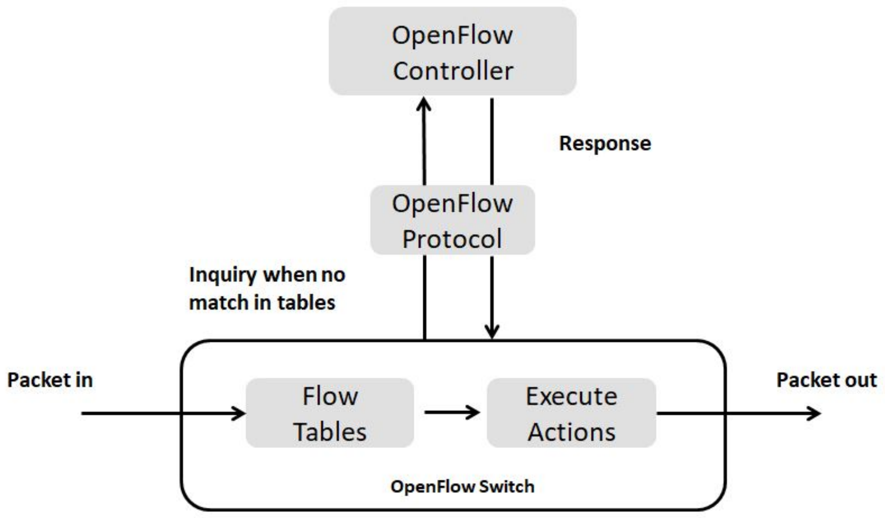

Figure 2.4: OpenFlow Architecture

A flow is defined as set of packets that belong to a specific flow entry.Switches situated in the infrastructure layer have flow tables, which consist of flow entries. These entries could be seen as forwarding rules with three important components: a command list, a bit pattern that shows particular flow's properties and a set of counters. Figure 2.4 shows the function of an OF-based network. When a packet arrives to the OF switch it can apply one of the following cases:

- If the packet matches to a particular flow entry of the flow tables of the switch, then the packet is forwarded following the rules of that specific entry.
- If the packet matches to a particular flow entry but there are no actions related with that flow entry, the packet is dropped.
- If the packet does not match a particular flow entry, a request is sent to the controller. Then it assigns a new OF entry where future packets are handled by the switch.

The OF channel is the interface that is used to connect the controller with the OF switch. Therefore we can identify three types of OF protocol messages [14]:

- Controller - Switch Messages:
  Those messages are initiated by the controller and used to manage and configure the underlying devices in the infrastructure layer.
- Asynchronous Messages:
  Those are initiated by the switches to send information back to the controller. Typical examples are - notification of a packet arrival, error or alarm messages, switch state change reports.

- Symmetric Messages:
  Those can be initiated by both the controller or the switch and are sent periodically to check the existence of a device (the so-called echo messages) or to test latency.

2.2.7 SDN controller types

Below is given a brief description of different SDN controllers. There are various types of controllers that have been developed and offer different possibilities.

- NOX:

  NOX is the first OpenFlow controller and has been widely used by researchers since the majority of SDN or OpenFlow applications in papers are based on it. Being open-source and written in C++ programming language makes that controller provide a developer-friendly environment in which it is easy to make modifications. No major changes were made on this controller since the last 5 years.

- POX:

  POX represents an early version of NOX. It is also an open-source controller, but unlike NOX, it is written in Python. This offers the opportunity to implement new features, but his performance is lower when conparing with other controllers.

- Beacon:

  Beacon represents a stable controller that is written in Java. It was first produced in 2010 and its performance make him a viable solution for real conditions use.

- Floodlight:

  Floodlight has its root from the Beacon controller and is an open-source Java-Based OpenFlow Controller. It is a controller which is easy to set up and with great performances, being the most mature OpenFlow controller. [8]

2.3 The European Spallation Source

The European Spallation Source is a case of interest in our thesis, therefore overall information about the facility is presented. ESS is a research facility, where neutron beams will be used to investigate scientific questions. ESS Steering Committee consists of 17 partner countries with financial contributions and contributions of equipment and other technical services. The construction of the site began in 2013, will reach its full specifications in 2025 and it will serve the European research community until 2065. [5]

2.3.1 Integrated Control System

The main parts of the facility are: accelerator, target, neutron scattering instruments and conventional facilities. The Integrated control system (ICS) will unify all these components and in this way will keep the costs relatively low. The components of ICS are:

- The control system core represents a set of systems that provides data and services to engineers, physicists and operators. The core includes the timing systems, the machine protection system (MPS) and the personnel protection system (PPS).
- Control Boxes represents servers that control a set of equipment. The ICS will include many control boxes (e.g. for an internal team), dividing the responsibilities, which leads to a smoother integration.
- The ICS central data management system is named beam line element database (BLED). Its main characteristic is to store and manage the information about the machine.
- The human-machine interface (HMI) encloses user interfaces, provides monitoring of facility's status and remote access.
- The development environment is responsible for development procedures, code storage or upgrades of the ICS software.[6]

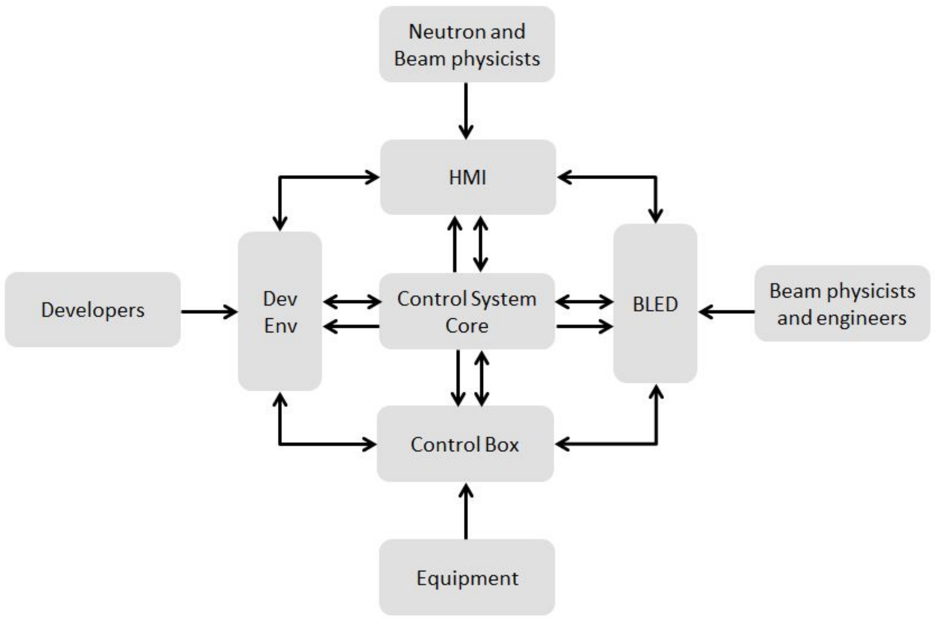

Figure 2.5: Control System Architecture

2.3.2 Networks

ESS will be the world’s most powerful science site and this leads to specific security needs for accessing it’s control system. One solution that would provide perimeter security is the complete isolation of ICS from the Internet. However, this is not a suitable solution since it is desirable to access and control the instruments remotely. The solution that the engineers came with is to have some dedicated gateway hosts and the remote access to ICS will be possible only through these hosts.[6]

ICS operation will be provided by seven independent networks:

- A control network - used by channel access
- A timing network - used for time synchronization
- A Machine Protection System (MPS) - used for machine protection
- A Personnel Protection System (PPS) - used for personnel protection
- A video network - used for video streaming
- A network for collecting beam diagnostic data
- A programmable logic controller (PLC) network - used for reliable implementation of safety. [6]

Network	Transfer rate	Protocol
Core control	1 Gbps	TCP
Timing	2.5 Gbps	Custom
Machine protection system		Custom
Programmable logic controller	100 Mbps	TCP or UDP
Video	> 1 Gbps	UDP
Beam diagnostics	maximum available	UDP

Table 2.1: Transfer rates and protocols for different networks [6]

2.3.3 Failure detection

Operations activities will be coordinated and monitored from the main control room. Here, a large monitor will provide an analysis of the state of the machine. If there is an error or a malfunction of the machine, then the status "Error" will be displayed, otherwise the status will be "all OK". More detailed information will be displayed on additional monitors, giving the opportunity to the main monitor to display specific information upon request.

Machine protection system

The main role of Machine Protection System (MPS) is to protect the equipment from damage caused by a malfunctioning equipment. If a malfunction is detected, then MPS will initiate an emergency shutdown of the machine. Moreover, MPS could trace the responsible subsystem and the nature of the problem that caused the emergency shutdown. Every second that the machine is not on the production mode is crucial, that’s why the MPS will try to avoid false emergency shutdowns, which will lead to a lower amount of time with the system shouted down.

Chapter 3

SDN meets Big-Data

Big-Data represents one of the most popular research topics in both academia and industry. When referring to big data, they are data sets with such volume and complexity that traditional data management tools are unsuitable to deal with it. Big-Data can be characterized according to five salient features, named as the "5Vs":

- Volume - represents the most obvious characteristic and refers to the amount of data. Many companies are collecting data from different sources (social media, business transactions or information from sensors) and with the rapid development of new technologies, the amount of data will increase significantly.
- Velocity - represents speed of data, which increases proportionally with the amount of produced data. For example, RFID tags or sensors requires to deal with data in near-real time.
- Variety - represents the range of data type and source. Generated data comes in many types of formats-from numeric data to audio data or financial transactions.
- Value - represents the usefulness of data. Collecting endless amounts of data is useless if it can not be turned into value.
- Veracity - represents the trustworthiness of data. For example, think about the people that are posting on Twitter or Instagram while using hash tags and abbreviations and the reliability of all that content. [9]

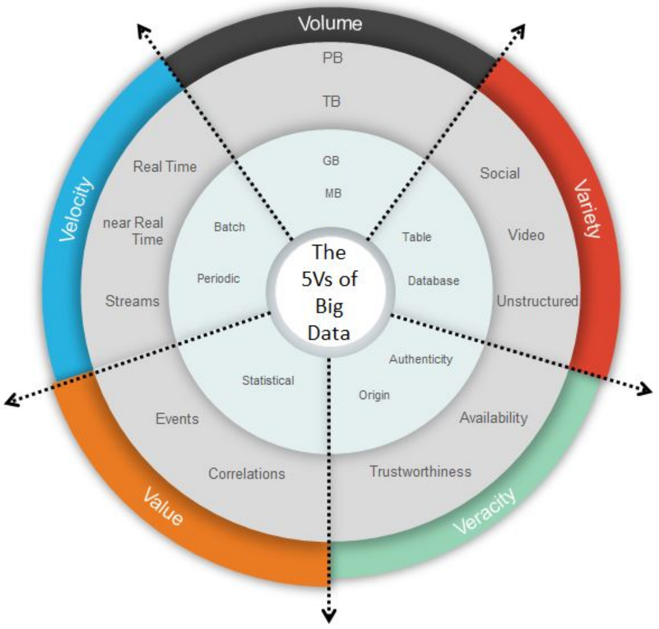

Figure 3.1: Big-Data: Expanding on 5 fronts at an increasing rate

Considering the ongoing development of Internet of Things (IoT) and scientific research, there is an obvious increasing of the amount of data. According to International Data Corporation (IDC), it is expected that the global data volume will grow from 130 exabytes (EB) in 2005 to 40,000 EB in 2020.[9]

3.1 Particle accelerator - a Big-Data scenario

To be able to classify a particle accelerator as a Big-Data scenario, the next characteristics should be identified: volume, velocity, value.

Given the fact that the ESS site is under construction at the moment of writing, we can compare its structure with the LHC site, which represents one of the world's most powerful particle accelerator and is located at the European Organization for Nuclear Research (CERN).

Volume

At LHC, the protons collide some 1 billion times per second and this process generates approximately one petabyte (PB) of collision data per second. However, the data is filtered by the experiments and only the important one are kept. As an example of the data volumes generated by an particle accelerator, CERN generated 49 petabytes of data during the year of 2016. [11]

Velocity

A consequence of the huge data volume is represented by an increased demand for data transfer since all the produced data needs to be stored and then transferred to permanent storage and analysis. Moreover, the data rates will increase in case of a malfunction is detected and a shut down of the accelerator is required. In this case, the data needs to be saved as reliable and as fast as possible.

Value

The produced data at ESS facility has an extremely importance since it will be used for research in various fields.

3.2 How can SDN benefit Big-Data applications

During the past years, there has been performed intensive research for both SDN and Big-Data, but they were mostly addressed separately. However, SDN's features can have an important impact on Big-Data applications. For example, features like separation the control plane and data plane, the centralized controller or the ability to program the network can facilitate Big-Data acquisition, storage and processing. Furthermore, this section borrows greatly from [9].

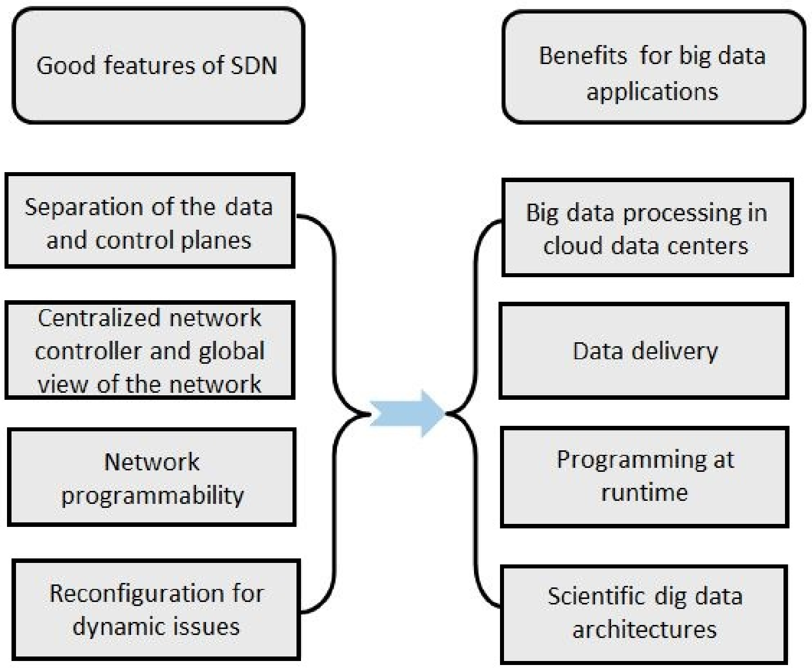
Good features of SDN
Benefits for big data applications
Separation of the data and control planes
Centralized network controller and global view of the network
Network programmability
Reconfiguration for dynamic issues
Big data processing in cloud data centers
Data delivery
Programming at runtime
Scientific dig data architectures

Figure 3.2: Features of SDN that can benefit Big-Data applications

3.2.1 SDN can benefit Big-Data processing in cloud data centers

A characteristic of an SDN enabled data center is that networking and storage infrastructure is automated by SDN. The cloud data centers represents the cloud platform which is built on the basis of the data centers. In addition, this infrastructure for big data applications can be hosted by cloud service providers.

3.2.2 SDN can benefit data delivery for Big-Data applications

One of the main concerns when referring to Big-Data applications is represented by the data delivery, which represents a great challenge due to the large volumes of data. In [12], the author present a prototype demonstration of Big-Data applications which shows that SDN features like programmability and flexibility can benefit data delivery for Big-Data applications.

3.2.3 SDN can benefit scientific Big-Data architectures

Science produce nowadays huge amounts of data and in some cases, due to different reasons, the data can not be processed and analyzed at the same location where they are generated. As the volumes of generated data increases, a scalable end-to-end network architectures and implementations that enable applications to use the network more efficiently is needed.

The authors of [13] introduce SDN to scientific Big-Data models and propose a new architecture model for campus networks that use SDN for multi-science disciplines within the same campus network. Finally, the capability of this new architecture which implies SDN is demonstrated for scientific big data applications.

3.2.4 Programming at Runtime for optimizing big data applications

In general, Big-Data applications require frequent reconfigurations due to changing environments. For this issue, SDN's capacity of programming at runtime is very useful.
The fact that there does not exist an standardized SDN controller creates issues when discussing the programming language in SDN. Some controllers uses Java, whereas others uses Phyton. Big-Data applications causes frequent flow table changes, and for this there is required a common programming language in SDN. [9]

Chapter 4
A general case

Particle accelerators as described earlier are facilities that produce huge amount of valuable data in normal production mode. ESS is used as an example in our work, but the provided solutions could be implemented in similar facilities. We have assumed that the normal production mode architecture could be able to handle the transport of the produced data.

According to [1], published in 2015, Kevin Woods - Brocade director of product management of software networking, says that CERN produces huge amount of data every second. Given the quantity of the produced data, he says that it is a challenging task to transport, store or analyze that information. Also, according to Tony Cass, who is the head of the IT communication systems group at CERN, SDN is expected to make their network more flexible. He claims also that SDN is more reactive when there are sudden changes in the topology.

In case of a failure, data should be collected and saved from the nodes as fast as possible, before the facility is restarted. The nature of the produced data falls into the category of Big-Data and as presented in the previous chapters this complexity makes the traditional network unable to support the needs of the site. Therefore, we presented SDN which has been a promising technique to overcome the drawbacks of the traditional network.

So far we have a traditional architecture for normal operation mode and we switch to SDN architecture when a failure occurs. However, we have seen that SDN offers more advantages and robustness. Therefore, we suggest the use of SDN network for both situations - in normal and alarm state. We base our proposal on the advantages of SDN and based on the information that CERN is also looking forward to implement SDN to handle the data in normal mode.

This chapter introduces a software defined communication network which is adapted for both normal and emergency mode. Sub-section 4.1 presents the normal production mode, while sub-section 4.2 explains why we use distributed controllers rather than a single centralized one. In sub-section 4.3 we present a list of variables that are important for our system design. Given those variables, we show several use cases as examples.

Sub-section 4.4 presents the network design which is based on the examples from sub-section 4.3. Due to lack of other resources, the solutions that we show in this sub-chapter are based mainly on discussions with Saaed Bastani, a former researcher in the department of Electrical and Information Technology (EIT) at

Lund University.

Our network design is based on three KPRs that were taken into account during our design phase - reliability, scalability and time sensitivity.

4.1 Network architecture for normal production mode

SDN is used to separate the control and data plane and provides an opportunity to enable a network to be application aware. Below we explain the 3-layered SDN architecture that is adapted for normal operation mode with a description of the components that build each plane.

Application plane: As could be seen from Figure 2.3 the application plane is responsible to push information to the controller for making decisions. Big-Data Analytic are taken into account in our case of interest, therefore, we have Big-Data applications that interact with other applications that run on the top of our network. With the emergence of SDN, it is easier for the network to integrate with those applications. Network resources can be allocated based on the application’s requirements to increase the speed of Big-Data delivery.[17]

Control plane: The main role of it is to keep the forwarding table updated so that the data plane can forward the traffic. The controller sends information about the network state to the applications through NAPIs and translates application requirements to data plane through SAPIs. The controller is informed for any malfunction that occurs in the data plane by the asynchronous OF messages. In this situation the controller changes the network architecture from normal production mode to the one with the buffers shown later on. This unit, however, is a single point of failure.It is important to make this plane more robust in a situation where we need to store the data as fast as possible. The next sub-chapter explains how we can mitigate the drawbacks of a single controller by having another control layer model.

Data/Infrastructure plane: Information about the exact view of this plane was not provided to us. Therefore, we look at this part as a "black-box". Typically data acquisition/accelerator nodes are situated in this layer where they pass the information to the switches situated above.

The bottom of Figure 4.1 shows the data plane where it is important to understand that the attachment of two nodes to a switch is given as an example. The main idea here is to show that a sufficient number of switches must be installed to process the data as quick as possible in case of a failure.

OpenFlow switches process a packet according to their flow tables. The OF switch is connected to the control plane (or in particular the SDN controller) by the SAPI. It is typically using OpenFlow and provides an open and standard way for a switch to pass information to the controller.

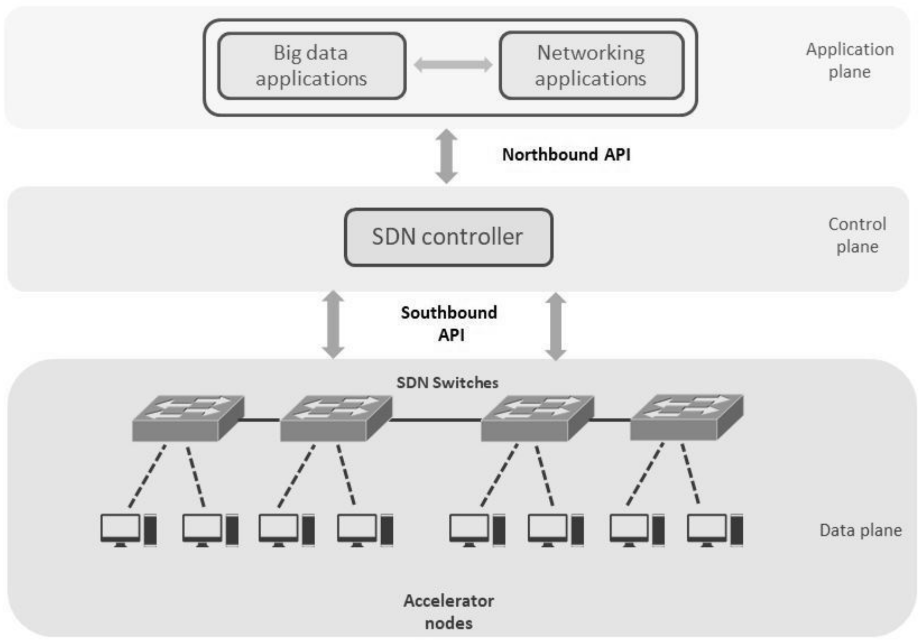

Figure 4.1: General SDN architecture for particle accelerator

4.2 From Centralized to Distributed Controllers

This sub-chapter gives an overview of why we chose the distributed controller architectures rather than the centralized one.

Centralized architecture do not meet two of the performance requirements - reliability and scalability, which are critical for our network architecture when we detect a failure.

When SDN was introduced, the main idea was to have a centralized controller. However, one of the most significant aspects with such a controller is that it is a single point of failure. Due to variety of reasons, the controller can stop functioning at any time and thus abandon the network without its control plane. Another disadvantage with this model is the big delays that we can experience from far-situated data plane units that transfer information to the controller. The switches with which the controller will communicate are supposed to be quite many, which will require more controllers that are physically situated to overcome the problems with the network scalability.

The distributed controller architecture, as show in Figure 4.2, is an approach that suits better to our case and clearly meets better the network performance requirements. Here we have a cluster of controllers that work together in the control plane to cover the entire network. Those controllers are geographically distributed but logically centralized. Logically centralized means that each controller makes decision based on a global view of the the entire network. In this type of architecture, the controllers have the same responsibilities and are always aware of the changes in the network topology by instant network synchronization. If one of the controllers fails to operate, the SDN operation continues to operate without any significant problems.

The communication between the multiple SDN controllers is made via eastbound and westbound APIs. Some tasks supported by these interfaces include changing data between controllers and monitoring if all controllers are up. Since each controller implements it's own east/west - bound APIs, the drawback is represented by the lack of standard API. This leads to a lower inter-operability between different controllers.[18]

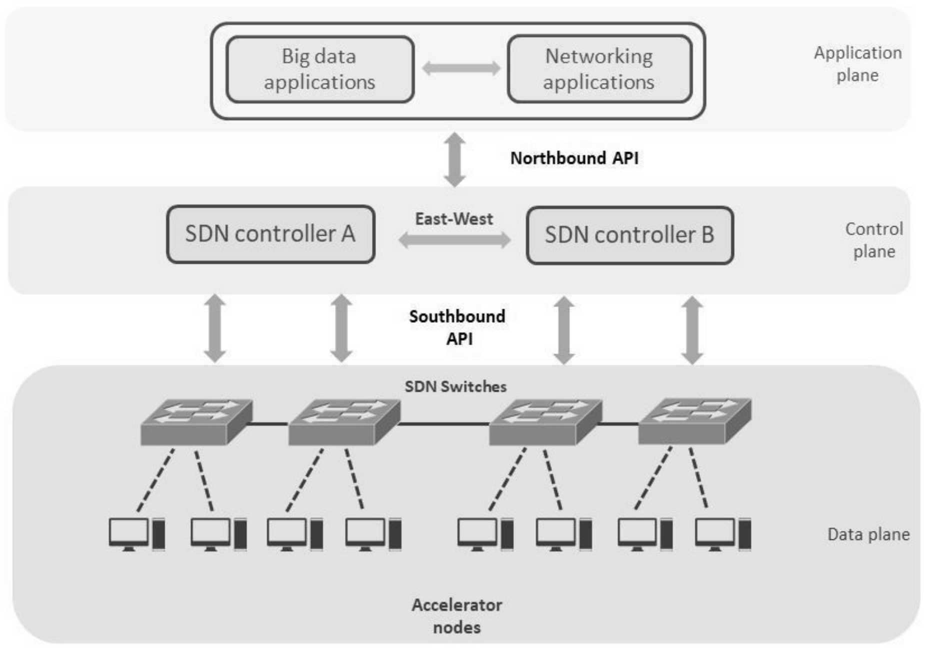

Figure 4.2: SDN with distributed controllers

4.3 Network architecture for emergency case

In the previous sub-sections we have discussed how we will meet the reliability and scalability requirements from control plane perspective. In this sub-section we present our network architecture for emergency case. We put more attention to the data plane, where we show how this design will help us meet better the KPRs.

The accelerator nodes in the data plane are expected to be quite many. This will require more switches to be attached to a given number of nodes to overcome the problems with network scalability and time sensitivity.

The network functions in the following way: when there is a malfunction, the control plane is informed for the failure from the data plane via SAPI. Then, the controller updates the application plane by NAPI for the error. The applications push information to the control plane to change the network architecture from normal production mode to the one shown below. The next step is to update the flow tables to push the data from the nodes to the buffers. When the process of storing data has been completed, a transmission control protocol (TCP) connection could be used, to transport the data reliably back to the application layer.

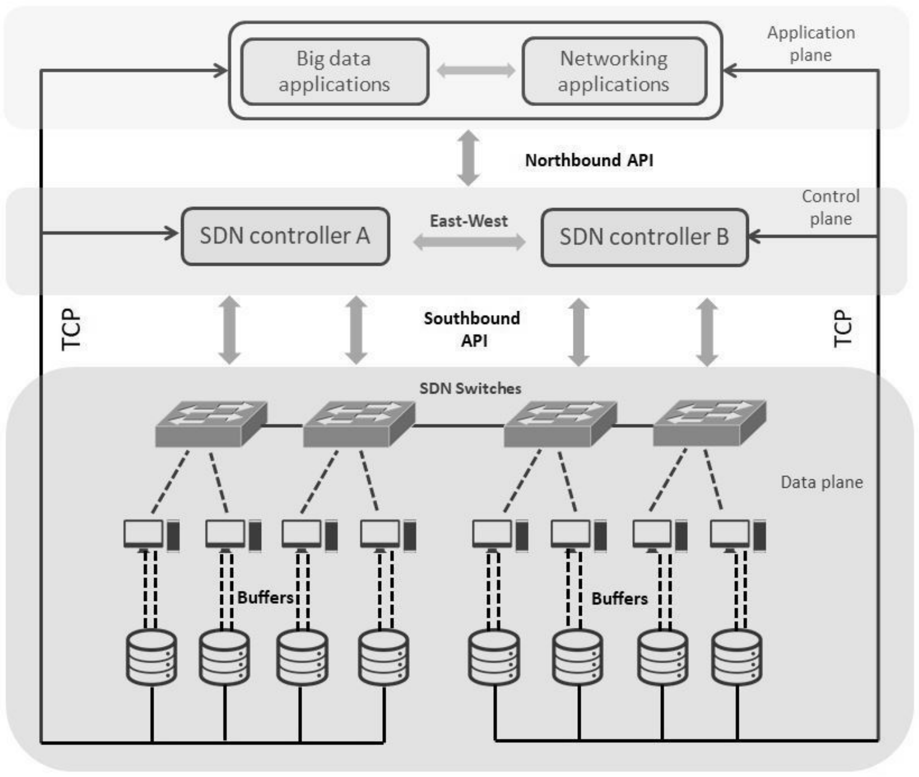

Figure 4.3: General architecture case for particle accelerators with buffer attached to each node

In Figure 4.3 we present an architecture where a buffer is attached to each node. Depending on the system parameters, the number of nodes that are attached to a buffer can differ. The system parameters and the different use cases will be shown in the next sub-section. This construction will improve the reliability and the time-sensitivity of the network. Data will be collected in the buffers faster and only a particular node’s data could be lost if it fails.

We want to process our data as quick as possible, before we restart the facility. In certain cases we would not be able to do that, if a node is attached to a buffer using one communication link, given its certain capacity. For faster transfer of the data, we suggest to attach a parallel communication link to increase the bandwidth, as could be seen in Figure 4.3. If the bandwidth of the first link is sufficient to handle our data, the second link could be set on stand-by mode to save power. Therefore, it could be used only if there is a failure in the first link (i.e having the second link as a back-up) or if there is not enough bandwidth to support the data flow.

4.4 Estimations of network parameters

The architecture model presented above is not based on any data. In this sub-chapter, we support our solution by showing different use cases, where the input data is estimated. We need to know how much data has to be transported, what time is required to send the data to the buffers and where are the bottlenecks.

4.4.1 Network parameters

In this sub-section, a list of network parameters that are used to make the calculations for the use cases is presented.

- **Total produced data per second.** This represents the summed data rate from all the nodes, assuming that all the nodes produce the same amount of data.
- **Saved data duration.** In case of an emergency shut down and restart of the facility, there is a need to save the last produced data for further investigation.
- **Restart time of the accelerator.** The time required for the facility to restart, when a malfunction occurs.
- **Number of nodes.** This variable depends on the complexity of the site and represents the number of data acquisition nodes.
- **Buffer Input/Output (I/O) data transfer rate.** Depending on the buffer type that will be used, there are different data rates which represent the maximum rate that a buffer can save the produced data from a node.
- **Capacity of a link.** The nodes are connected to buffers with links with certain capacity.

- **Number of nodes per switch.** This is the maximum number of nodes that can be attached to the ports of a switch.
- **Data rate per node.** This is the data that each node produces per second, when we are in emergency situation.
- **Number of buffers.** The number of buffers that is needed to store the data from all the nodes.
- **Number of links.** This is the number of links that are used to attach the nodes with the buffers.

4.4.2 Scenarios

In this sub-section we make valid estimations of the values for the network parameters that are presented above. Some of the variables have a constant value (e.g. capacity of a link), others are variables with multiple values (e.g. the number of nodes).

Table 4.1 shows the description of the parameters together with the values that we estimate.

Parameter	Type	Values
Total produced data per second	Variable	20 GB/s, 50 GB/s, 100 GB/s
Saved data duration	Constant	120 sec
Restart time	Variable	5 sec, 30 sec, 2 min, 10 min
Number of nodes	Variable	50, 200, 500
Buffer I/O data transfer rate	Variable	600 MB/s, 19.2 GB/s
Capacity of the link	Constant	10 GB/s
Number of nodes per switch	Constant	24
Data rate per node	Variable	To be calculated
Number of buffers	Variable	To be calculated
Number of links	Variable	To be calculated

Table 4.1: Values of the network's parameters

By using the information in the table, we will elaborate different use cases and set thresholds for the system availability. The goal is to find the values of the last three parameters from Table 4.1. Please note that the values for the total produced data per second are valid when we have an emergency situation.

The data rate produced by a node in case of an emergency is calculated by using the following mathematical expression:

$$d = \frac{p \cdot 120}{t \cdot n}$$

, where d=data rate per node, p=produced data per second, t=restart time and n=number of nodes and the constant value of 120 seconds, which is the saved data duration (in case of a failure, we need to save the data that is produced by the accelerator for the last 2 minutes).

There are two types of buffer that we take as an example for our solution.

- Buffer 1: Seagate ST8000VN0002
  Size of a buffer - 8 TB [21]
- Buffer 2: IBM V9000
  Size of a buffer - scalable up to 456 TB [22]

Depending on the buffer type that will be used, there are different data rates that a buffer can save the produced data from a node.

- Seagate ST8000VN0002: I/O data transfer rate - 600 MB/s
- IBM V9000: I/O data transfer rate - 19.2 GB/s

If the node produces big amount of data, which is more than the I/O data transfer rate of any of the buffers, then we have a bottleneck on the buffer side. In this case, the system will be not able to process the information that is sent from the nodes.

The nodes are connected to buffers with links. Those links have a certain capacity and in our case we assume that it has a fixed value of 10 GB/s. Another situation where the system could face a bottleneck is when the nodes produce more data than the capacity of the link. However, this could be solved by increasing the number of links so that we can achieve higher the capacity if needed.

Table 4.2 shows the resulted data rate per node for a facility with 50 nodes. It can be observed that different values are taken into consideration for total produced data per second and restart time for a better understanding of the system behaviour.

Total produced data per second	Restart time	Data rate per node
20 GB/s	5 seconds	9.6 GB/s
20 GB/s	30 seconds	1.6 GB/s
20 GB/s	2 minutes	0.4 GB/s
20 GB/s	10 minutes	0.08 GB/s
50 GB/s	5 seconds	24 GB/s
50 GB/s	30 seconds	4 GB/s
50 GB/s	2 minutes	1 GB/s
50 GB/s	10 minutes	0.2 GB/s
100 GB/s	5 seconds	48 GB/s
100 GB/s	30 seconds	8 GB/s
100 GB/s	2 minutes	2 GB/s
100 GB/s	10 minutes	0.4 GB/s

Table 4.2: Calculations for a facility with 50 nodes

Figure 4.4 shows the graphical representation of the system behaviour using the values from table 4.2. It could be seen that the thresholds are dependent on the buffer I/O data transfer rate.

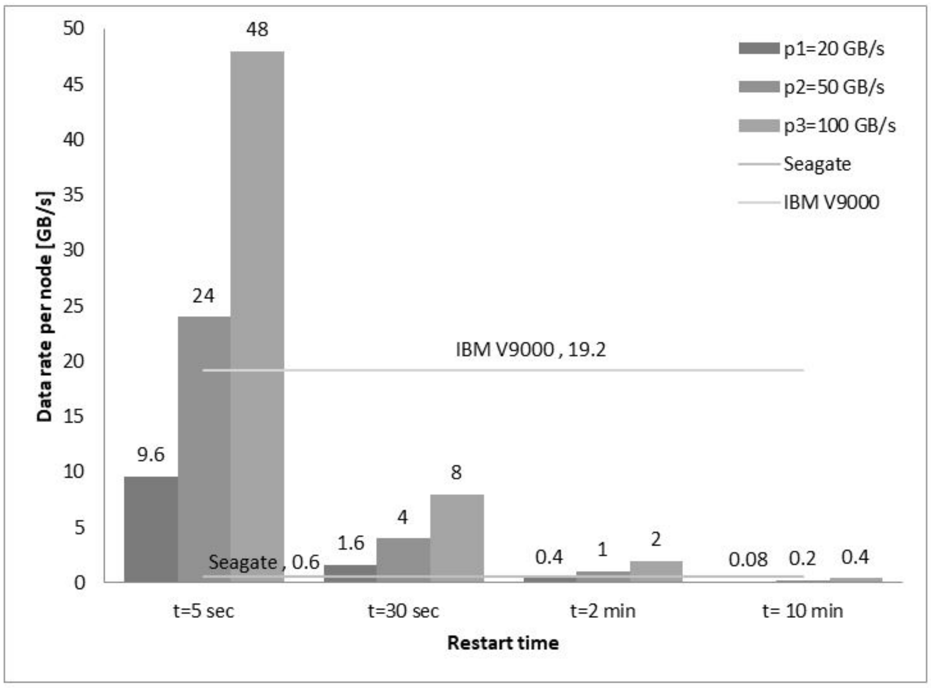

Figure 4.4: Graphical representation of the parameter's values from Table 4.2

In a similar way as above, Table 4.3 shows the resulted data rate per node, but in this case it is assumed that the facility has 200 nodes.

Total produced data per second	Restart time	Data rate per node
20 GB/s	5 seconds	2.4 GB/s
20 GB/s	30 seconds	0.4 GB/s
20 GB/s	2 minutes	0.1 GB/s
20 GB/s	10 minutes	0.02 GB/s
50 GB/s	5 seconds	6 GB/s
50 GB/s	30 seconds	1 GB/s
50 GB/s	2 minutes	0.25 GB/s
50 GB/s	10 minutes	0.05 GB/s
100 GB/s	5 seconds	12 GB/s
100 GB/s	30 seconds	2 GB/s
100 GB/s	2 minutes	0.5 GB/s
100 GB/s	10 minutes	0.1 GB/s

Table 4.3: Calculations for a facility with 200 nodes

Figure 4.5 shows the graphical representation of the system behaviour using the values from table 4.3. It could be seen again that the thresholds are dependent on the buffer I/O data transfer rate. In this figure, the scale has been changed to achieve better view of the threshold when Seagate buffer is used. The buffer provided by IBM handles all the cases without bottleneck on the buffer side.

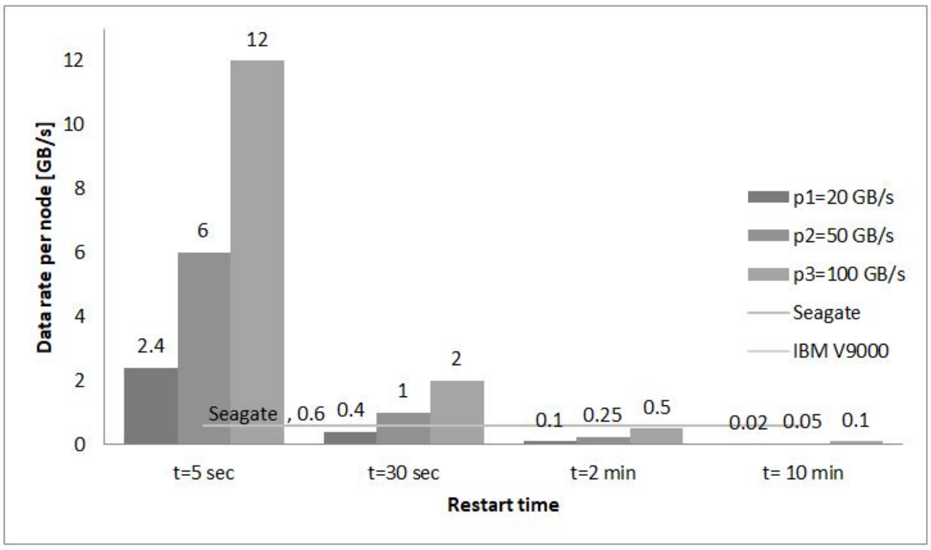

Figure 4.5: Graphical representation of the parameter's values from Table 4.3

The last case is represented by a facility with 500 nodes. The resulted data rate per node are shown in Table 4.4.

Total produced data per second	Restart time	Data rate per node
20 GB/s	5 seconds	0.96 GB/s
20 GB/s	30 seconds	0.16 GB/s
20 GB/s	2 minutes	0.04 GB/s
20 GB/s	10 minutes	0.008 GB/s
50 GB/s	5 seconds	2.4 GB/s
50 GB/s	30 seconds	0.4 GB/s
50 GB/s	2 minutes	0.1 GB/s
50 GB/s	10 minutes	0.02 GB/s
100 GB/s	5 seconds	4.8 GB/s
100 GB/s	30 seconds	0.8 GB/s
100 GB/s	2 minutes	0.2 GB/s
100 GB/s	10 minutes	0.04 GB/s

Table 4.4: Calculations for a facility with 500 nodes

Figure 4.6 shows the graphical representation of the system behaviour using the values from table 4.4. It could be seen again that the thresholds are dependent on the buffer I/O data transfer rate. In a similar manner, in this figure, the scale has been changed to achieve better view of the threshold when Seagate buffer is used. The buffer provided by IBM handles all the cases without bottleneck on the buffer side.

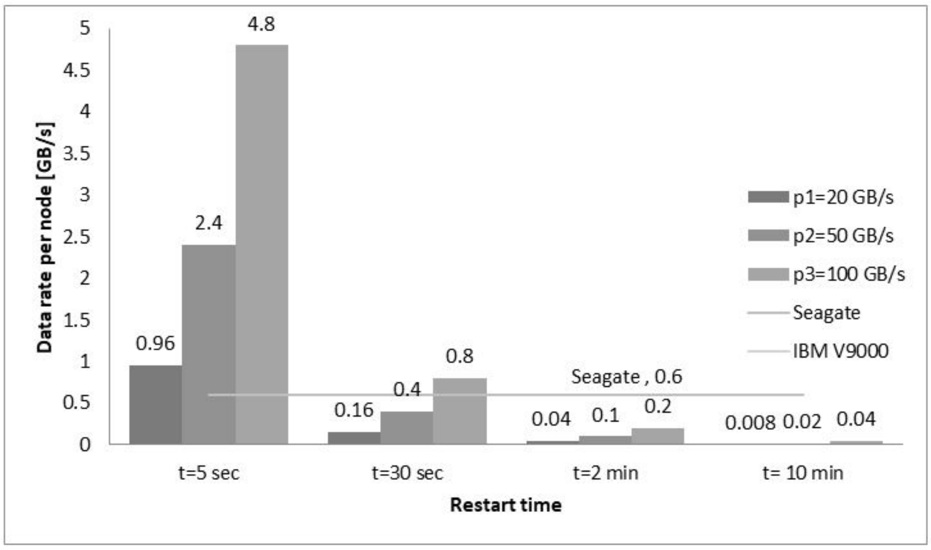

Figure 4.6: Graphical representation of the parameter's values from Table 4.4

Chapter 5 Discussion

With all the parameters presented in the previous chapter, it could be seen that a lot of combinations are possible.

To give the reader a better view of the presented network architecture, we show three use cases as an example in sub-chapter 5.1. Each case is specific and there is a dependency between the variables.

Sub-chapter 5.2 shows extreme cases where the network is not able to process the data due to a bottleneck on the buffer side.

Another important assumption here is that a single switch has 24 ports which means that at most 24 nodes can be attached to it.

5.1 Use cases

Case 1: From table 4.2 we choose the case with 50 nodes, a total produced data rate of 100 GB/s and restart time of 10 minutes. This results in 0.4 GB/s (400 MB/s) of data produced by each node. In this case, 3 switches will be required to cover the number of nodes.

- If the Seagate ST8000VN0002 buffer is used, with maximum I/O data-transfer rate - 600 MB/s, it follows that at most one node can be attached to a single buffer. Thus, for the given network, 50 buffers will be needed.

  Each node is connected to the buffer with 2 links, where one of them is not used in this case, but acts as a backup link. Therefore, the total number of links is 100.

- If the IBM V9000 buffer is used, with maximum I/O data-transfer rate - 19.2 GB/s, it follows that at most 48 nodes can be attached to a single buffer. Thus, for the 50 nodes, 2 buffers will be needed.

  Each node is connected to the buffer with 2 links, where one of them is not used in this case, but acts as a backup link. Therefore, the total number of links is 100.

Case 2. From table 4.3 we choose the case with 200 nodes, a total produced transfer rate of 50 GB/s and restart time of 5 seconds. This results in 6 GB/s of data produced by each node. In this case, 9 switches will be required to cover the number of nodes.

- If the Seagate ST8000VN0002 buffer is used, with maximum I/O data-transfer rate - 600 MB/s, it follows that there is a bottleneck on the buffer side. The buffer is not able to support the data produced by a node given the short restart time.
- If the IBM V9000 buffer is used, with maximum I/O data-transfer rate - 19.2 GB/s, it follows that at most 3 nodes can be attached to the buffer. Thus, for our given network case, 67 buffers will be needed.

  Each node is connected to the buffer with 2 links, where one of them is not used in this case, but acts as a backup link. Therefore, the total number of links is 400.

Case 3. From table 4.4 we choose the case with 500 nodes, a total produced transfer rate of 50 GB/s and restart time of 2 minutes. This results in 0.1 GB/s (100 MB/s) of data produced by each node. In this case, 21 switches will be required to cover the number of nodes.

- If the Seagate ST8000VN0002 buffer is used, with I/O data-transfer rate - 600 MB/s, it follows that at most 6 nodes can be attached to the buffer. Thus, for the given network with 500 node 84 buffers will be needed.

  Each node is connected to the buffer with 2 links, where one of them is not used in this case, but acts as a backup link. Therefore, the total number of links is 1000.
- If the IBM V9000 buffer is used, with maximum I/O data-transfer rate - 19.2 GB/s, it follows that at most 190 nodes can be attached to the buffer. Thus, for our 500 node network we will need at 3 buffers.

  Each node is connected to the buffer with 2 links, where one of them is not used in this case, but acts as a backup link. Therefore, the total number of links is 1000.

5.2 Extreme cases

- For a system with 50 nodes, a restart time of 5 seconds, with a total produced data per second of 50 GB/s, the resulted data rate per node will be 24 GB/s. In this case, the bottleneck will be in the communication link and at the buffer side, for both types of buffer.
- For a system with 50 nodes, a restart time of 5 seconds, with a total produced data per second of 100 GB/s, the resulted data rate per node will be 48 GB/s. Again, the bottlenecks will be in the communication link and at the buffer side, for both types of buffer.

Chapter 6

Conclusion

Particle accelerators are facilities with great importance for the humans, since they produce data that is used to gain more insight in different research areas. With the innovations in those science centers, the production of valuable research data increases with high temps. Such information is typically dependent on a communication network, designed for quick data transport in normal operating mode that ensures successful transfer.

In case of emergency situation, however, the accelerator has to be stopped and restarted and the produced data must be stored within the restart time. In this situation the traditional networking techniques are not able to guarantee robustness and flexibility, mainly because of the the nature of data that is produced. Therefore, there is a need for a new architecture type that would be able to collect and store the information in case of a failure in the accelerator.

6.1 Why SDN?

In this Master thesis we have made a study about SDN and how it could be used to save and collect all of our data as fast as possible from the accelerator's data acquisition nodes. We have explored the benefits of using SDN as a technique that would offer more flexibility for data acquisition in big research environments. The Big-Data characteristics were discussed since the information that is produced in such big facility falls into that category of data type.

Network programmability is an advantage of SDN which allows us to change policies easier and in this way we can react faster to a network modification or a network event, which is important since our network is time sensitive. With the traditional network techniques, this was something that we would need to do manually, which led to more errors and slower reaction times.

The controller sets low level rules to specified requirements and in this way it is more flexible to re-route traffic, inspect new policies or test particular flows. This thesis proposed an network architecture with a distributed SDN controller that solves the problem with the reliability.

The forwarding devices in the data plane are not required to provide many features and capabilities. Being less vendor specific, the switches will make the network equipment low-cost. This means that, with SDN, adding a new device is not an issue and makes the network scalable.

6.2 Proposed general case

As a summary of Chapter 4, having a buffer attached to each data acquisition node makes the network more reliable and makes it less time sensitive since the data does not need to be processed by the switch before saving it in the buffer.

We have defined network parameters and made calculations with data that has been estimated due to lack of exact information.
In chapter 5, readers got familiar with different use cases. The extreme cases showed that if the produced data rate per node is high and the restart time is too small, we can experience bottlenecks on the buffer side.

We can conclude that for all the cases when produced data rate per node is less than 10 GB/s (capacity of the link) and with the IBM v9000 storage, there will be no bottlenecks in the network. In the same way, for the cases when the produced data rate per node is less than 600 MB/s (I/O data-transfer rate for the Seagate buffer), there will be no bottlenecks in the network.

As a final conclusion, taking into consideration the KPRs during the network design phase - reliability, scalability and time sensitivity, we can conclude that SDN meets them and could be used in our case with particle accelerators.

Chapter 7

Future work

This master thesis was a pure theoretical research and the presented use cases are based on estimated data. We believe that a challenging and useful next step would be to have an exact network architecture model, with given parameters and assigned values. Therefore, our general case could be used as an example for real-life implementations.

Throughout our research, we have seen many simulations that have been done for different scenarios. Mininet is an example of a software that supports simulations of SDN and OpenFlow and has been broadly used. We suggest that it could be studied and used as a simulation tool.

When there is information about the network's parameters, tests could be made to see how exactly the network will behave and where will be the bottlenecks. We strongly suggest this as a future step, since it is important to have a network architecture which will be suitable to handle the burst of data that is produced and needs to be collected in case of an emergency situation.

The protocol proposed to be used for transferring the collected data from the storage units to the application layer is TCP, but specific details of how the protocol is designed are not presented in our work. It is strongly recommended to make a detailed research for this part. This is an important part of the whole design process, since by using the TCP protocol we could be sure that the data is collected without errors. Other alternatives than using TCP protocol could be used, so we think that it could be further investigated.

Issues with the network security have been not discussed in our work, but they are quite important given the peculiarity of the produced data and should not be underestimated. For example, storing the data in cloud database is not recommended solution in our case due to reliability and security issues. The use of SDN solves a lot of problems in the networking world nowadays, but adding a centralized controller that governs the management makes the network a perfect target for attacks and in particle accelerator case it could cause a lot of troubles. Furthermore, the 3-layered architecture gives the possibility of malicious attacks on every layer:

- Application layer - attack some applications or the NAPIs
- Control layer - attack the controller, the NAPIs or the SAPIs
- Data layer - attack switches or the SAPIs.

Bibliography

[1] http://www.networkcomputing.com/networking/cern-taps-brocade-sdn/2108472247 , Retrieved March 2017

[2] Harvey Newman, Azher Mughal, Dorian Kcira, Iosif Legrand, Ramiro Voicu, Julian Bunn. *"High Speed Scientific Data Transfers using Software Defined Networking"*, Retrieved March 2017

[3] J. Balcas, T. W. Hendricks, D. Kcira, A. Mughal, H. Newman, M Spiropulu, J. R. Vlimant, et al 2017 J. Phys.: Conf. Ser. 898 112001 *"SDN-NGenIA, a software defined next generation integrated architecture for HEP and data intensive science"*, Retrieved March 2017

[4] Md. HumayunKabir. *"Software Defined Networking (SDN): A Revolution in Computer Network"*, IOSR Journal of Computer Engineering, Nov. - Dec. 2013, Retrieved March 2017

[5] http://eval.esss.lu.se/DocDB/0002/000274/014/TDR_online_ver_overview.pdf , Retrieved March 2017

[6] http://eval.esss.lu.se/DocDB/0002/000274/014/TDR_online_ver_ch5.pdf , Retrieved March 2017

[7] https://www.citrix.se/products/netscaler-adc/resources/sdn-101.html , Retrieved July 2017

[8] Thomas Paradis, "Software-Defined Networking", Master of Science Thesis, January, 2014, Retrieved August 2017

[9] Laizhong Cui, F. Richard Yu, Qiao Yan. *"When Big Data Meets Software-Defined Networking: SDN for Big Data and Big Data for SDN"*, Retrieved August 2017

[10] http://searchcloudcomputing.techtarget.com/definition/big-data-Big-Data, Retrieved August 2017

[11] https://home.cern/about/updates/2017/07/cern-data-centre-passes-200-petabyte-milestone, Retrieved August 2017

[12] A. Sadasivarao. *"Bursting Data Between Data Centers: Case for Transport SDN"*, Retrieved September 2017

[13] I. Monga, E. Pouyoul, and C. Guok. *"Software-Defined Networking for Big-Data Science - Architectural Models from Campus to the WAN"*, Retrieved September 2017

[14] Muhammad Zeshan Naseer. *"Modelling Control Traffic in Distributed Software Defined Networks"*, Degree Project in Electrical Engineering, Stockholm, 2016, Retrieved September 2017

[15] https://www.sdxcentral.com/sdn/definitions/north-bound-interfaces-api/,Retrieved October 2017

[16] https://www.sdxcentral.com/sdn/definitions/southbound-interface-api/,Retrieved October 2017

[17] Li-Wei Cheng, Shie-Yuan Wang. *"Application-Aware SDN Routing for Big Data Networking"*, Retrieved October 2017

[18] Diego Kreutz, Fernando M. V. Ramos, Paulo Esteves Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky and Steve Uhlig. *"Software-Defined Networking: A Comprehensive Survey"*, Retrieved October 2017

[19] https://home.cern/about/updates/2017/12/breaking-data-records-bit-bit, Retrieved December 2017

[20] https://home.cern/about/computing/processing-what-record, Retrieved October 2017

[21] https://www.seagate.com/www-content/product-content/nas-fam/nas-hdd/en-us/docs/100724684f.pdf, Retreieved Januari 2018

[22] http://www.hamburgnet.de/products/ibm/IBM_FlashSystem_V9000.pdf, Retreieved Januari 2018