Configuring the LHCb Readout System using a Database

Lana Abadie
CERN, Switzerland Geneva and GET/INT

***Abstract* - The LHCb readout system is composed of hundreds of electronics boards, an event-building network based on Gigabit Ethernet switches and an online processing farm. The Experiment Control System (ECS) configures the system from an online configuration database. This database must contain device parameters, the hierarchical structure and the connectivity information of the system. In addition the switches in the event-building network require routing tables that have to be generated according to the connectivity while ensuring optimal load balancing.**
We apply the Entity Relationship model to represent the connectivity of the system. PL/SQL code dynamically builds the routing tables using the information contained in the database.

I. Introduction

The LHCb experiment is one of the four experiments at the CERN LHC (Large Hadron Collider).
The LHCb Online Data Acquisition (DAQ) system collects data from front-end electronics and transfers them to a CPU farm for execution of the software trigger algorithms. There are two traffics intermixed corresponding to the data for Level-1 trigger (L1) and for the High Level Trigger (HLT).
For more details about the DAQ system, see [1].
The DAQ as shown in Figure 1 is composed of:

- About 300 hundred front-end electronics boards (FEs)
- About twenty switches to reduce the number of links (multiplexing layer)
- A readout network to access the SubFarms
- About a hundred of SubFarms to process data: a SubFarm is composed of a SubFarm Controller (SFC) and about twenty PCs

The monitoring, configuration and operation of all the experimental equipment will be handled by the Experiment Control System (ECS) [2] All configurable information required for the control of the equipment will reside in a configuration database [1].
In this paper we focus on an effective way to configure the LHCb Online Data Acquisition (DAQ) system.

II. Configuration of the DAQ : Applied Methodology

This section explains in detail what needs to be configured from the DAQ.

A. Objectives & requirements

Configuring the DAQ network system mainly consists of providing 2 kinds of information:

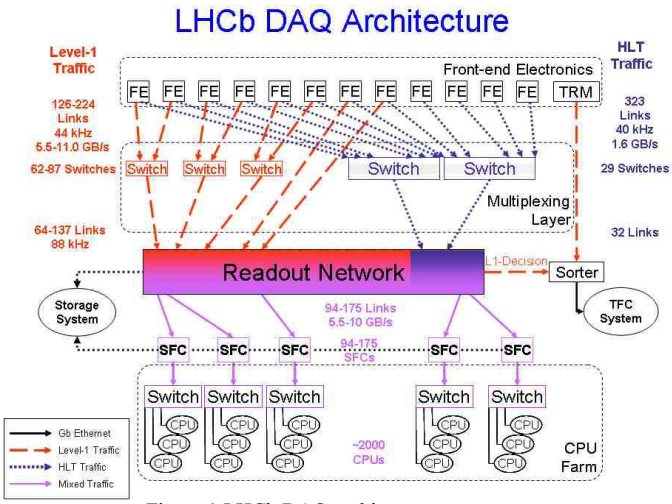


Figure 1 LHCb DAQ architecture

- Routing tables for the switches (see II.C for more details)
- Specific configuration files such as Dynamic Host Configuration Protocol (DHCP) [3], Domain Name System (DNS)[1] [4] for servers and for PCs in the subfarms.

The design of the configuration database has to satisfy the following requirements:

- Reduce the use of the switch « Automatic» or « self-management » routing mode. The behavior of any networking devices in a DAQ system has to be deterministic to control and manage the routing of data.
- Keep track of all routing tables and configuration files to enable the system to recover after a crash and also to analyze the cause of the failure afterwards.

[1] A DNS converts an IP name into an IP address and vice versa.

B. Use of a database

To satisfy the previous requirements, we use the Online Configuration database [5] to configure the DAQ system since databases are nowadays a powerful and safe technology to save data for large systems.

Information stored in the database should be:

- **Complete**, to be able to start up the network after a crash
- **Scalable,** to support any extension or changes to the network topology or in the hardware of the components
- **Consistent**, to avoid errors and incompatibilities in routing tables or in configuration files
- **Easy to maintain**; to minimize the data inserted by users

C. Scenarios

For the DAQ network configuration, the two main scenarios (use cases) are:

1. generating routing tables for all switches
2. generating configuration files for PCs and servers

A *routing table* is a table stored in a router or a switch that keeps track of routes to particular network host interfaces. There is one routing table per switch or router. Different kinds of routing table algorithms exist; in this paper, we use the most common one - the shortest path – which consists in minimizing the number of network devices crossed i.e. number of hops.

We can distinguish 2 kinds of routing tables, IP for routers (layer 3) and MAC for switches (layer 2).

A *configuration file* is a text file stored in a server which enables network devices to get configured. In this paper, we take the example of the Dynamic Host Configuration Protocol (DHCP) configuration file.

The DHCP configuration file provides a PC with an IP address (dynamically or set by the user) and also sends configuration information and BOOTP (bootstrap protocol) parameters [6]. It is very useful when a PC is newly installed in the network or if the PC is diskless.

III. REPRESENTATION OF THE DAQ NETWORK

We will focus on determining the necessary information to be stored in the database and on finding an efficient table schema.

A. Network key words

A network is a graph composed of a certain number of **nodes** and **links.**

In a network, we have the following key elements:

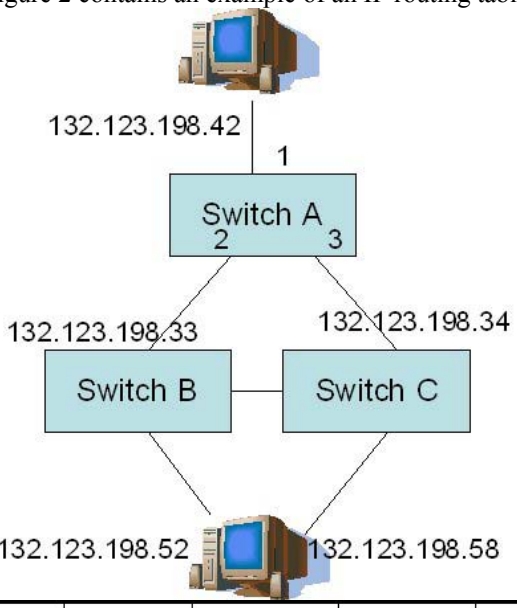
- **Host nodes** corresponding to equipment which processes data such as PCs
- **Switch nodes** or intermediate nodes which forward data through the network
- **A port** which is an entity to interconnect nodes
- **A network interface card** (NIC) is connected to a port number of a node. Each NIC is characterized by a MAC address and an IP address. For example, the PC in the bottom of Figure 2 has 2 interfaces.
- **A link** connects two nodes

B. Routing table properties

An IP (resp. MAC) routing table usually has the following entries for each destination:

- Port number to forward data
- IP (resp. MAC) address of the next hop
- IP (resp.MAC) address of the destination
- Subnet Mask [7](resp.VLAN prefix[8]): property of a subnet (VLAN) i.e. a pool of networking devices.
- Path length or round trip down to the destination

Figure 2 contains an example of an IP routing table.



Port nb	IP @ Next hop	IP @ destin.	Subnet mask	Path length
1	132.123. 198.42	132.123. 198.42	255.255. 255.0	1
2	132.123. 198.33	132.123. 198.52	255.255. 255.0	2
3	132.123. 198.34	132.123. 198.58	255.255. 255.0	2

Figure 2 Example of the IP **routing table** of switch A

C. Required User data

.The different properties of a link are:

- a start and an end;
- a type corresponding to the data type that the link carries such as control or data traffic (Fig.1);
- orientation : bidirectional or unidirectional;
- status : either functional or dead

A destination is characterized by a MAC or an IP address respectively if it's a MAC or an IP routing table.

D. Table design

The information described previously needs to be stored in the database. To do the table design, we use the entity relationship model because of its simplicity and its flexibility to model many systems.

The entity relationship model (ERM) is a graphical representation of entities and their relationships to each other, typically used in computing in regard to the organization of data within databases or information systems. An entity is a piece of data—an object or concept about which data is stored. A relationship is how the data is shared between entities.

Figure 3 shows the table schema which represents the DAQ network system. The primary key of a table, to refer to other tables is in bold. The arrows represent the relationships between the different entities (or tables).

In the database, we can distinguish two kinds of information:

- Inserted by the user (table names in upper case in Fig. 3) such as the list of equipment, properties of links, IP and MAC addresses. This type of information can't be generated automatically.
- Derived by the information stored in the database (table names in lower case in Fig. 3) such as routing tables, configuration files…

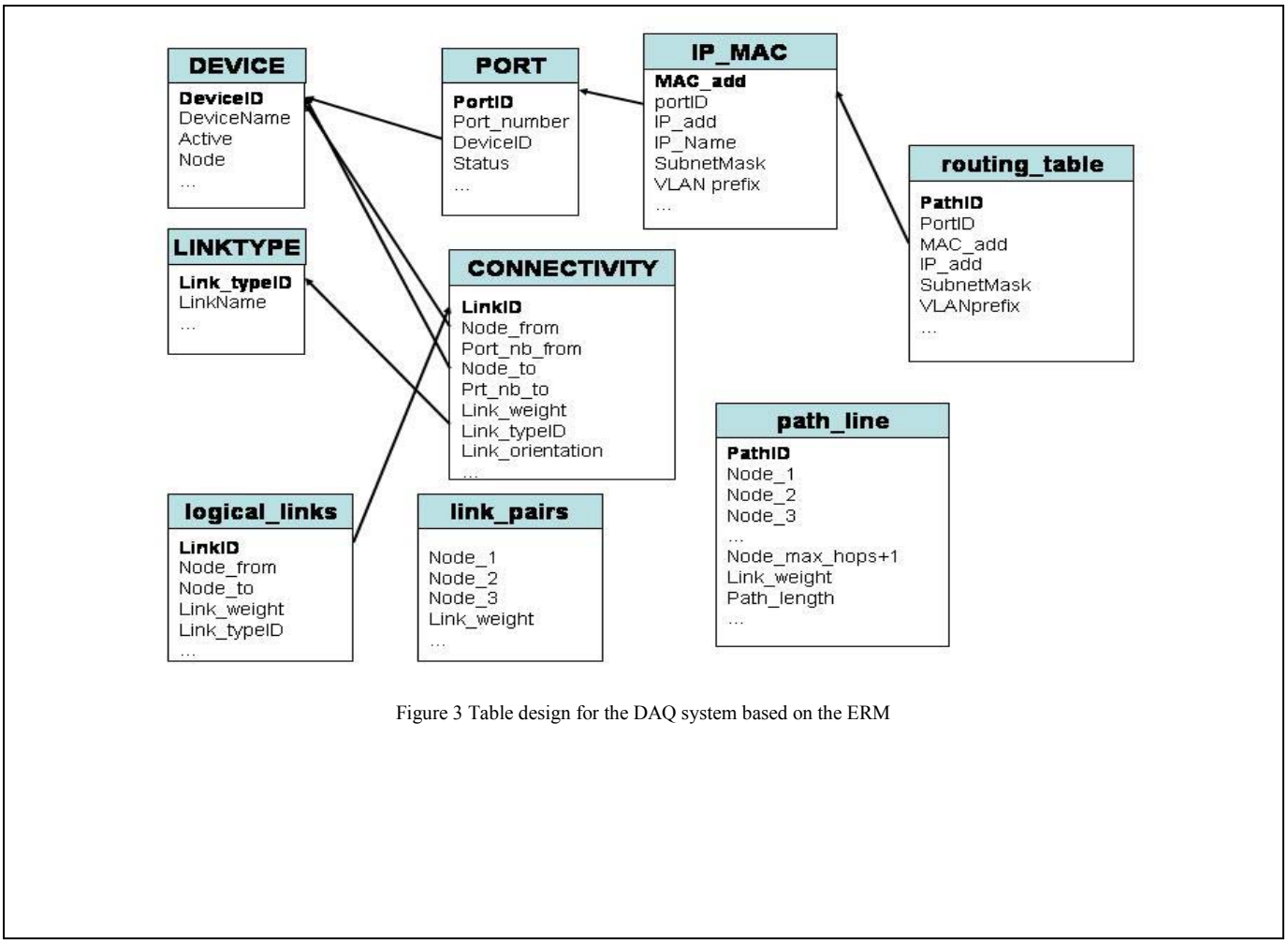


Figure 3 Table design for the DAQ system based on the ERM

IV. Generating Routing Tables

This section explains how we can generate a routing table using the table schema of Fig. 3.

A. Link and path weights.

In the **DEVICE** table from Figure 3, the **Node** attribute is equal to 1 if it's a host node and 0 if it's an intermediate node.

In the **CONNECTIVITY** table from Figure 3, the **Link_weight** is computed as follows:

- **Link_weight** is equal to 0 if the link is between 2 switches
- **Link_weight** is equal to 1 if the **Node_from** is a host and **Node_to** is a switch.
- **Link_weight** is equal to 2 if the **Node_to** is a host node and **Node_from** is a switch.
- **Link_weight** is equal to 3 if **Node_from** and **Node_to** are both host nodes (although not used here).

The path weight is the sum of the link weights along the path.

B. Algorithm

The main idea to generate a routing table for a switch A is to determine all paths starting from switch A and ending at a host node, whose length is less than a maximum number of hops.

A routing path has **one and only one host** node which is always **the terminated node**. Thus the weight of a routing path is always greater than 0.

The **path_line** table contains the complete routing paths of a given switch: there is one **path_line** table per switch.

The input parameters of the routing algorithm are the name of the switch node and the maximum hops of a path to eliminate the too long paths.

The algorithm to generate the routing table is based on the following steps:

- Create the **logical_links** table which is the logical (or macroscopic) view of the system. It is derived from the **CONNECTIVITY** table (cf Figure 3).
  In the **logical_links** table, multiple physical links between 2 nodes are represented by one single link.
- Create the **link_pairs** table in which all valid pairs of successive links are stored.
  To create the **link_pairs** table, we perform a self-join of the **nodelink** table. It is like a Cartesian product of the table with itself with the following constraints:
  - Node_to of link1 is equal to Node_from of link2:
  - Node_to of link1 is not a host node.

  The **logical_links** and **link_pairs** tables are regenerated only when a change occurs in the **CONNECTIVITY** table.
- For each switch, a **path_line** table is created which is initialized with the elements from table **link_pairs** which have this switch as a starting node.An iterative join between the **link_pairs** and the **path_line** tables is executed. The core of this step is the following:

  i=3*;*
  while (stop=0 and *i*<max_length) **loop**

  Select t.**Node_3**, f.**Node_1**,…, f.**Node_*i***

  from **link_pairs** t, **path_line** f

  where t.**Node2**=f.**Node_*i*** and t.**Node_1**=f.**Node_*i*-1** and

  f.**Path_weight=0** and f.**Path_type=t.Pair_type** and

  [$\prod_{k \text{ between } 1 \text{ and } i}$ (**t.Node_3-f.Node_*k***)]!=0 (check cycles)

  Insert the new valid paths found (of length *i*) in the **path_line** table:

  Delete from **path_line** the rows where **Path_weight**=0 and **Node_*i*+1**=0 (old rows)

  Count the number of **Path_weight**=0 in the **path_line** table.
  If all the **Path_weight** > 0 then
  stop=1;
  else
  i=*i*+1;
  end if;
  end loop;

- Finally, we then select the shortest routing path per destination to obtain a consistent routing table.

This algorithm works in all network architectures including full mesh layouts.

C. Implementation and tests

The Configuration database is a central database based on Oracle technology (Oracle 9i) [9]

To generate the routing tables, we have developed a PL/SQL package. PL/SQL is a portable Oracle language [10], i.e. it can be embedded in any language such as JAVA, C/C++. The code is executed at the server side. We have tested this package with different layouts of the DAQ network system which has been represented with 2350 nodes and 4216 links.

V. CONCLUSION

This paper proposes a way to configure a network with a focus on how generating routing tables using a database. The next step is to implement other packages to generate configuration

files by still using the same representation. One ongoing project is to automatically generate a DHCP configuration file.

ACKNOWLEDGMENTS

I would like to thank Arthur Barczik, Jean-Pierre Dufey, Benjamin Gaidioz and Niko Neufeld for their explanation of the LHCb DAQ network.

REFERENCES

[1] LHCb Online Computing, Trigger System TDR, December 2001
[2] the LHCb ECS: http://lhcb-comp.web.cern.ch/lhcb-comp/ECS/default.htm
[3] DHCP: RFC 2131 http://www.faqs.org/rfcs/rfc2131.html
[4] DNS: RFC 1035 http://www.faqs.org/rfcs/rfc1035.html
[5] LHCb Online Configuration Database: http://lhcb-online.web.cern.ch/lhcb-online/configurationdb/
[6] BOOTP Protocol, RFC 1497 : http://www.faqs.org/rfcs/rfc1497.html
[7] Subnet Mask definition : http://compnetworking.about.com/od/workingwithipaddresses/l/aa043000a.htm
[8]: VLAN information : http://www.cisco.com/univercd/cc/td/doc/product/rtrmgmt/sw_ntman/cwsimain/cwsi2/cwsiug2/vlan2/laneapp.htm
[9] Oracle Technology : http://www.oracle.com/technology/index.html
[10] PL/SQL language : http://www.oracle.com/technology/tech/pl_sql/index.html