GASOLine
Generic Acquisition
System OnLine

Document préliminaire
de Spécifications Logicielles

Simon CHOLLET (LLR) – s.chollet@llr.in2p3.fr
Nicolas ROCHE (LLR) – nicolas.roche@llr.in2p3.fr

1. Introduction

Le but de ce document est de présenter succinctement l'acquisition actuelle. Nous présentons ensuite, l'architecture logicielle envisagée pour assurer de meilleures performances, mais aussi une meilleure segmentation des modules logiciels.

Dans la première version du logiciel d'acquisition, la librairie « LibLDA » actuellement utilisée, présente quelques défauts de stabilité et de gestion des ressources. Ces problèmes sont en partie expliqués par des défauts de conception générale. De plus, la « LibLDA » a évolué depuis le début en intégrant au fur et à mesure du temps des algorithmes de corrections, et différents modes d'utilisation selon les types de détecteurs lus.

Le but est de revenir, avec les éléments et l'expérience passée, sur une nouvelle architecture plus modulaire, permettant notamment de rendre à ce logiciel d'acquisition son rôle initial, c'est-à-dire l'acquisition générique. Les fonctions auxiliaires qu'elle assurait auparavant seront déportées autant que possible vers des applications extérieures.

2. Architecture matérielle

Dans cette partie, nous détaillons les différents éléments matériels qui permettent d'assurer l'acquisition globale du détecteur.

Nous rappelons rapidement l'architecture matérielle actuelle, à l'aide de la figure suivante :

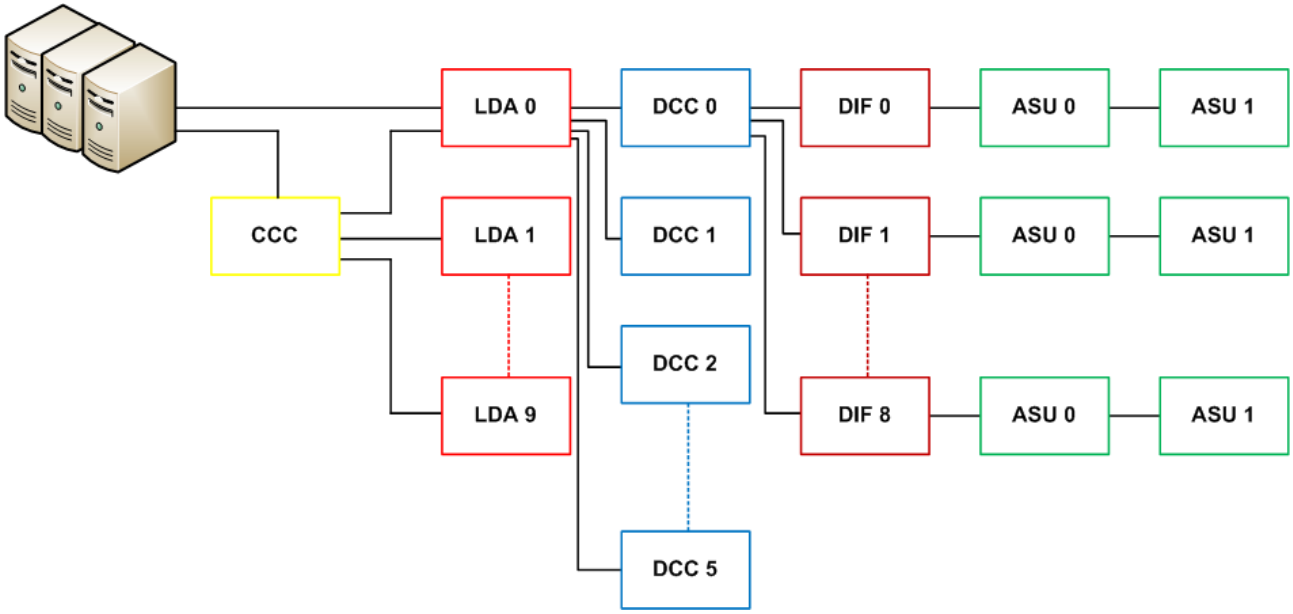

Figure1. Architecture matérielle

2.1. ASU

Composant permettant d'interfacer les détecteurs silicium : **A**ctive **S**ensor **U**nit.

Chaque ASU est composé de 24 ASICs, d'une liaison avec un autre ASU, et d'une sortie vers une DIF. Les entrées / sorties vers d'autre ASU sont optionnelles et utilisent le même bus propriétaire que vers la DIF.

Nous appelons le regroupement de plusieurs ASU, un SLAB.

SDU

2.2. DIF

Carte d'interface : **D**etector **I**nter**f**ace.

SDU

2.3. DCC

Carte de concentration de données : **D**ata **C**oncentrator **C**ard.

La carte DCC est composée de 9 entrées au format HDMI pouvant recevoir 9 DIF. Elle possède une sortie, aussi au format HDMI, qui se connecte à une carte LDA.

SDU

2.4. LDA

Carte permettant de concentrer plusieurs liens de données : **L**ink **D**ata **A**gregator.

La carte LDA est possède 10 entrées au format HDMI pouvant recevoir jusqu'à 10 cartes DCC ou DIF. Elle possède une sortie au format Ethernet optique ou cuivre. Elle possède aussi une entrée au format HDMI pour recevoir les commandes, les signaux de power pulsing et le trigger en provenance de la CCC.

SDU

2.5. CCC

Carte de synchronisation de commandes : **C**lock **C**ontrol **C**ard.

Cette carte possède 1 entrée trigger, et 8 sorties au format HDMI qui peuvent se connecter vers 8 cartes LDA. Le trigger permet de fonctionner en mode « beamtest », si ce trigger n'est pas utilisé, c'est le mode « ILC ».

SDU

3. Architecture logicielle

Dans cette partie, nous décrivons globalement l'architecture logicielle proposée. Les paragraphes suivants permettront de détailler les différents modules.

Le système présente plusieurs modules qui sont exécutables sur plusieurs machines différentes. Cette architecture a l'avantage d'être modulaire et permet de répartir les charges de traitement et de ressources sur des CPU différents.

Nous présentons ici, l'architecture logicielle envisagée pour cette nouvelle version d'application :

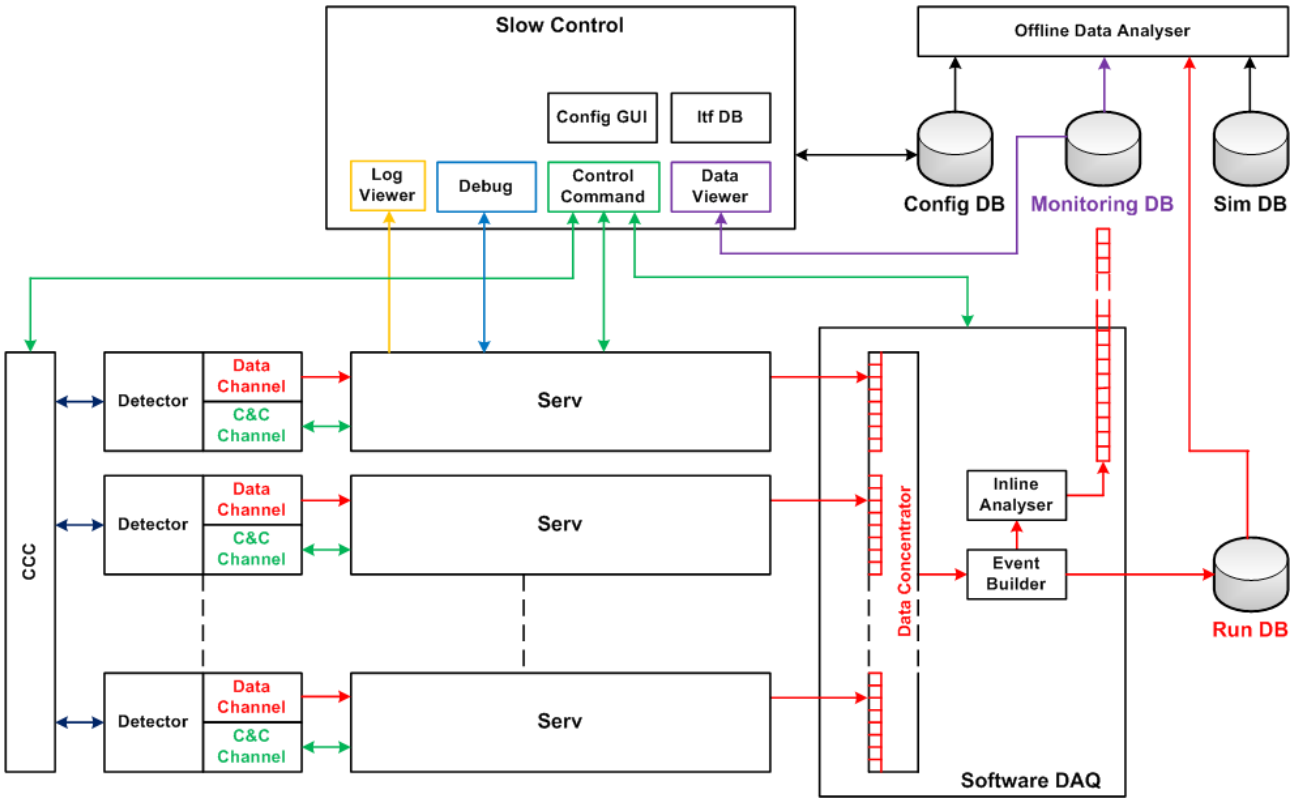

Figure2. Architecture logicielle.

3.1. Détecteur

Le détecteur est composé d'une ou plusieurs cartes LDA, qui elles-mêmes pilotent plusieurs cartes DCC qui concentrent plusieurs cartes DIF.

Chaque carte LDA possèdent en théorie 2 canaux de communication qui permettent de séparer le flux des données du flux des commandes et acquittements. En pratique, nous utiliserons un serveur afin de s'affranchir de cette limitation.

3.2. Serveur central

Au cœur du système d'acquisition, nous retrouvons le module `Server` qui permet de contrôler le flux des données provenant du détecteur avec deux canaux : un canal pour les données et un canal pour les commandes et acquittements. Plusieurs applications peuvent se connecter au `Server` avec des liens de communications de type socket.

Une fois l'acquisition lancée, le serveur traite les données acquises sur chaque canal pour les renvoyer vers le module de concentration `Data Concentrator`.

Ce module reçoit ses commandes et configurations de l'application de `Slow Control`.

3.3. Acquisition logicielle

Le module d'acquisition logicielle regroupe plusieurs applications : `Event Builder`, `Inline Analyser` et `Data Concentrator`. Le module `Data Concentrator` permet de gérer les flux asynchrones d'arrivée de données à l'aide de FIFO, et de multiplexage. Chaque événement reçu est stocké en attendant une reconstruction globale par le module `Event Builder`. Le module peut être

exécuté sur plusieurs machines différentes. Une fois l'événement reconstruit, il est stocké dans la base données dite d'acquisition : Run DB.

Le module Inline Analyser permet d'effectuer des analyses en ligne de tous les événements reconstruits, il enregistre ensuite les résultats de ses traitements dans la base de données Monitoring DB.

Les modules Data Concentrator, Event Builder, et Inline Analyser sont configurés et commandés par l'application de Slow Control.

C'est le module Event Builder qui détermine si un événement partiellement reconstruit doit être jeté ou stocké et étiqueté d'une manière particulière dans la base d'enregistrements Run DB. Il gère aussi un timeout qui permet de s'affranchir des paquets retardataires afin de ne pas pénaliser les enregistrements.

3.4. Bases de données

Il y a plusieurs bases de données prévues :

- **Config DB** : Base de données de gestion de **configurations**.
- **Run DB** : Base de données d'enregistrement des données **acquises**.
- **Sim DB** : Base de données d'enregistrement des données **simulées**.
- **Monitoring DB :** Base de données d'enregistrement des spectres produits en temps réel.

La base de données Config DB permet d'enregistrer plusieurs configurations différentes du système, et plus précisément les paramètres de configuration des différents modules, qu'ils soient logiciels ou matériels.

La base de données Monitoring DB est la base qui stocke les spectres élaborés à la volée, durant l'acquisition en se basant sur une partie des événements seulement.

La base de données Run DB est la base qui sera la plus sollicitée, elle permet de stocker les différentes acquisitions. Elle doit être dimensionnée pour pouvoir recevoir les données aussi vite que l'ensemble du détecteur lui les envoit. Le débit est borné à 300Mb.s-1 par LDA soit à 1Gb.s-1 pour 3 LDA.

La base de données Sim DB est la base de données qui stocke les données simulées. Elle n'intervient pas directement dans le processus d'acquisition. Elle permet principalement de faire de l'analyse de données dite OffLine.

3.5. Application Slow Control

Le module Slow Control regroupe plusieurs applications, et interfaces graphiques qui permettent de contrôler les différents modules logiciels (Control Command), d'afficher les messages (Log Viewer), les spectres issus des données acquises (Data Viewer), et de visionner la configuration (Config GUI).

Cette application permet aussi de paramétrer les modules Data Concentrator, Event Builder, et Inline Analyser. Elle a accès aux bases de données de configuration Config DB, et de suivis des données Monitoring DB.

Des applications de mesure de données lentes seront implémentées pour mesurer des températures, hydrométries, etc. Ces données seront jointes à la base de données d'enregistrements des acquisitions `Run DB`, mais sans passer par le module `Event Builder`, ni l'acquisition logicielle.

3.6. Analyse des données offline

Ce module n'intervient pas directement dans le processus d'acquisition des données, il contient les algorithmes de traitements et d'analyse de données. Certains de ces algorithmes pourront aussi, si les charges et les ressources le permettent être intégrés dans l'application de surveillance des données dans le module `Inline Analyser`.

4. Détails des modules

Dans cette partie, nous détaillons plus les différents modules logiciels.

4.1. DAQ électronique

La DAQ électronique est composée de détecteurs connectés à une CCC. ...

SDU

4.2. Slow Control

Les applications de `Slow Control` pourront être exécutées sur différentes machines selon la répartition des charges de calcul nécessaire.

Dans un premier temps, pour faciliter les étapes de développements, chaque interface graphique sera développée indépendamment les unes des autres.

Une application globale permettra, dans les phases finales de regrouper toutes ces interfaces dans une seule.

4.2.1 Interface avec la carte CCC

Le module `Control Command` pilote le séquencement général de l'acquisition, il pilote donc aussi la carte CCC, au travers d'un lien Ethernet.

Les commandes à prévoir sont les suivantes :

- **Start :** SDU.
- **Stop :** SDU.
- **Synchronize :** Nécessite en plus de positionner un registre sur toutes les DIFs + SDU.
- **OneEvt :** SDU.
- *SDU*

4.2.2 Interface avec le Serveur Central

Le module `Control Command` prend en charge le pilotage de la DAQ électronique. Il dialoguera avec le serveur central pour lui envoyer différentes commandes de demande de démarrage, de

configuration et de debug ou de demande de statut.

Pour communiquer avec le serveur central, l'application ouvrira une socket cliente de communication.

Les commandes à prévoir sont les suivantes :

- **Init :** SDU.
- **Config :** SDU.
- **Start :** SDU.
- **Stop :** SDU.
- **Close :** SDU.
- *SDU*

4.2.3 Module de Debug

Le module `Debug` permet d'affiner les requêtes de Debug et de statut demandées au cours d'une acquisition.

Compatibilité acquisition ? → *SDU*

SDU

4.2.4 Interface avec les bases de données

Nous prévoyons de développer une interface graphique contenant un arbre représentant une configuration générale du système, chaque branche décompose un élément logiciel ou matériel. La sélection d'un élément permet d'obtenir une interface propre à l'élément sélectionné.

Un format de fichier arborescent du type XML permettra d'enregistrer une ou plusieurs configurations.

Un module d'exportation, et d'importation sera mis en place pour transférer la configuration choisie depuis ou vers la base de données `Config DB`, en respectant la structure de celle-ci.

L'application de configuration pourra ainsi soit travailler directement avec la base de données `Config DB`, soit avec des fichiers stockés localement.

SDU

4.2.5 Interface avec l'Acquisition Logicielle

Le dialogue entre le module `Control Command` et l'acquisition logicielle sera établi avec une socket.

Les commandes à prévoir sont les suivantes :

- **Config :** réglage de la configuration, notamment le numéro de run, SDU.
- **Start :** Démarre la reconstruction et l'enregistrement des données.
- **Stop :** Arrête la reconstruction et l'enregistrement des données.

- *SDU*

SDU

4.3. Serveur central

Le serveur central est le cœur du système, il dialogue directement avec les canaux du détecteur, mais aussi avec les modules logiciels de contrôle-commande et d'acquisition. Il jour le rôle d'un switch.

C'est pourquoi, il intégrera au minimum 2 threads, l'un affecté au transfert des données, l'autre pour la prise en compte des commandes.

Pour l'implémentation des couches de dialogue, le module ouvrira des sockets vers les autres modules. Pour le partage des données, notamment avec le concentrateur de données, le serveur pourra implémenter soit une socket, soit un mécanisme de mémoire partagée.

Dans un premier temps, l'interface avec les modules logiciels se fera avec des commandes dites de « bas niveau », permettant d'assurer les performances. Une interface du type SOAP sera ensuite implémentée pour permettre des connexions logicielles de plus haut niveau utilisant le même protocole que les logiciels d'acquisition générique comme, par exemple XDAQ, Narval, Tango, etc.

La couche d'interface SOAP est une solution envisagée, il est possible d'implémenter aussi d'autre type d'interface pour d'autre type de SCADA.

4.3.1 Interface avec la DAQ électronique

Pour cette interface entre chaque LDA et le serveur central, nous nous référerons aux documentations suivantes :

- Documentation technique de la LDA : ref. *SDU*.
- Documentation technique de la DCC : ref. *SDU*.

SDU

4.3.2 Machine d'états

La machine d'état de ce serveur central peut être représentée ainsi :

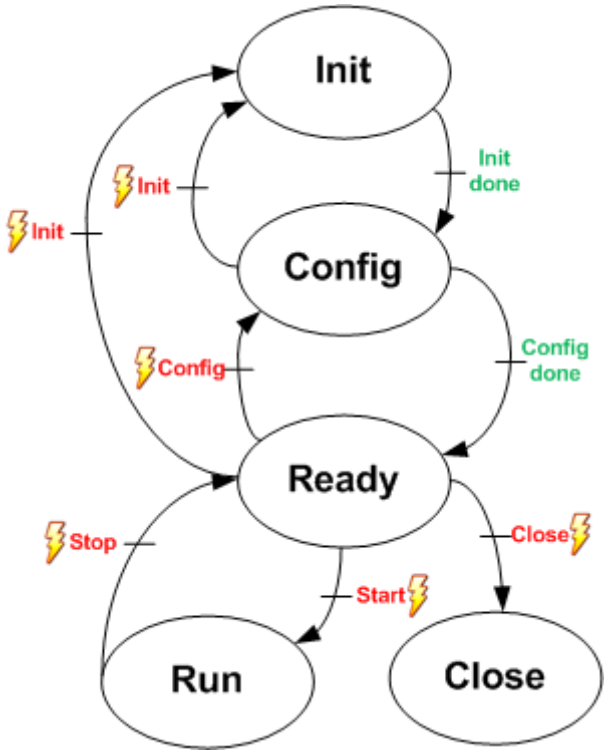

Figure3. Machine d'état du serveur central.

Au démarrage du serveur, le module `Server` va, dans un premier temps :

- **Ouvrir** les sockets de communication avec les autres applications.
- **Initialiser** les threads de gestion de commandes et de transferts de données.
- *SDU :* **Découvrir** plus ou moins automatiquement le matériel connecté : fonctionnalité « Plug&Play ».

SDU : Une fois que cette initialisation a été effectuée, il passe alors dans l'état `Config`., où il configure tous ses paramètres avec des valeurs par défaut.

Il arrive alors dans l'état où il est prêt à démarrer : état `Ready`.

Dans l'état `Ready`, il est possible alors de demander une nouvelle initialisation avec une commande `Init`, une nouvelle configuration avec le message `Config`. Nous pouvons demander directement la fermeture de l'application avec le message `Close`.

Le démarrage de l'acquisition des données acquises sur les détecteurs est demandé avec le message `Start`, le module passe alors dans l'état `Run`.

Dans l'état `Run`, le module aspire les données sur les détecteurs et les transmet au concentrateur de données. Le module sort de cet état lorsqu'il reçoit un message `Stop`.

Dans l'état `Close`, le module va :

- Fermer les sockets de communication.
- Arrêter les threads de traitements.

4.3.3 Paramètres de configuration

Dans cette partie, nous listons les paramètres de configuration du serveur central :

- Taille des buffers DIF.
- Taille des buffers PCAP.
- *SDU*.

SDU : La taille des buffers DIF est réglée pour permettre de traiter les événements acquis.

SDU : La taille des buffers PCAP permet de régler le nombre d'octets disponibles pour les communications.

4.3.4 Sockets de communication

Plusieurs sockets seront utilisées sur le serveur central pour établir la communication avec les composants logiciels extérieurs. Nous pouvons lister :

- Une socket **LOG**, en mode UDP : pour la journalisation de certains messages.
- Une socket **SLC**, en mode TCP : pour recevoir des commandes ou envoyer des acquittements.
- Une socket **DAQ**, en mode TCP : pour transférer les données au module de concentration de données.

Le mode TCP permet de garantir l'envoi et la réception des trames sur un lien Ethernet, il y a une gestion d'erreur plus évoluée que pour le mode UDP. Le mode UDP est plus rapide que le mode TCP, mais il utilise de plus petits buffers et il y a un risque que les paquets de données soient perdus.

Nous choisirons donc le mode TCP couplé à de gros buffers d'envois quand la communication demande de grosses performances.

SDU

4.3.5 Threads de traitements

Il y a plusieurs threads implémentés dans le serveur central :

- Thread **RDATA** de récupération des données depuis une LDA.
- Thread **WDATA** d'envoi des données vers le concentrateur.
- Thread **SLC** de gestion du contrôle-commande.
- *SDU*.

Chaque thread utilisera l'ordonnancement « FIFO » du noyau Linux (ne pas confondre avec les files de stockage) afin de garantir leur priorité sur tout autre processus utilisateur. Nous donnons la priorité maximale au thread de gestion des commandes (SLC), pour pouvoir réagir aux demandes provenant du Slow Control. Celles-ci sont des messages courts, simples et rapides à traiter. Si aucune demande n'est effectuée, les deux autres threads prendront alors la priorité.

SDU

4.4. Event Builder

Le module Event Builder, associé au module Data Concentrator ont plusieurs rôles. Ils permettent de sérialiser toutes les données acquises, pour les stocker dans la base de données Run DB. Mais ils permettent aussi de répartir les données sur plusieurs machines. Ceci permet de distribuer la charge CPU sur plusieurs éléments.

La répartition de charge est propre au module Data Concentrator. La figure suivante représente la méthode de répartition des flux de données :

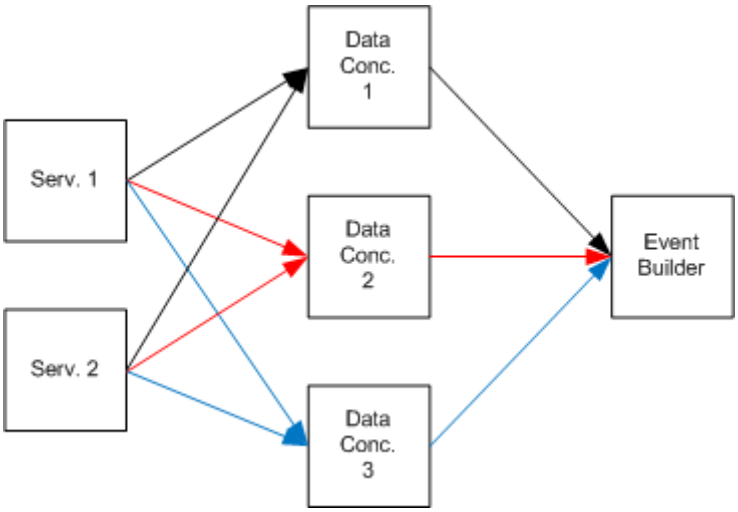

Figure4. Répartition des flux de données.

Dans l'exemple ci-dessus, les flèches noires représentent les premières réception de données, les flèches rouges les deuxièmes et, les flèches bleues les troisièmes. Les données sont envoyées sur des concentrateurs de données différents à chaque fois.

L'ordonnancement de cette distribution de flux de données est réglée par un algorithme round robbin de type modulo sur le numéro d'événement (ie trigger).

Cette distribution est configurée par le module Control Command du Slow Control. Cette méthode permet de s'affranchir d'un prévisible goulot d'étranglement au niveau du module Event Builder. En effet, le débit sera le même que pour la Run DB : 1Gb.s-1 d'événements à réceptionner, reconstruire et transmettre.

L'arrivée de ces données n'étant pas synchrone, il est possible de tomber sur des cas où des morceaux d'événements manquent. C'est le module Event Builder qui détermine alors la politique de garder ou non les événements même s'ils ne sont que partiellement remplis.

De même, si l'électronique ou le serveur ne délivre pas à temps un morceau d'événement, le module Data Concentrator relayera l'événement incomplet et jettera les éventuels morceaux retardataires.

Ces modules dialoguent avec le module de Slow Control :

- Ils seront configurés par une des applications du module Slow Control.
- Il sera possible de connaître l'état de ce module (nombre d'événements enregistrés, taux d'enregistrement, etc.).

SDU

5. Glossaire

ASU : Active **S**ensor **U**nit.

CCC : Clock **C**ontrol **C**ard.

DCC : Data **C**oncentrator **C**ard.

DIF : Detector **I**nter**f**ace.

GUI : Graphical **U**ser **I**nterface.

IHM : Interface **H**omme-**M**achine.

LDA : Link **D**ata **A**gregator.

SCADA : Supervisory **C**ontrol **A**nd **D**ata **A**cquisition.

SDU : Sera **D**éfini Ultérieurement.

SLAB : Regroupement de plusieurs ASU.

SOAP : Single **O**bject **A**ccess **P**rotocol, protocole de type RPC orienté objet et bâti sur XML.