DIAGNOSTIC AND MONITORING CERN ACCELERATOR CONTROLS INFRASTRUCTURE: THE DIAMON PROJECT FIRST DEPLOYMENT IN OPERATION

Mark Buttner, Pierre Charrue, Joel Lauener, Maciej Sobczak, CERN, Geneva, Switzerland

Abstract

The CERN accelerator controls infrastructure spans over several machines and several thousands of devices are used to collect and transmit piece of control data. Each of these remote devices might fail and therefore prevent correct operation. A complete diagnostic and monitoring infrastructure has been developed in order to provide Operation crews with complete and easy to use graphical interface presenting the state of the controls system. Simple agents running in each surveyed item periodically report monitoring information to a central server. Graphical JAVA clients in the operation centers subscribe to this monitoring data and display a view of the current state of the machines. Mouse actions from these clients allows for diagnostic commands to be sent to the agent to get additional details or to repair a faulty situation. This presentation will describe the overall architecture of DIAMON, present the different agents running in the controls system and a few views of the graphical clients. The outcome of the first months in operation of the DIAMON tools will also be presented. Finally, the future plans will be exposed.

OVERALL ARCHITECTURE OF DIAMON

DIAMON is a typical 3-tier application, communicating over JMS (Java Messaging Service). The "data acquisition tier" (agents) collects the data to be monitored, sends it to a middle-tier based on SonicMQ message brokers and a business logic implemented using OC4J (the Oracle application server for J2EE). The presentation tier is a Java Swing application receiving the information from the middle-tier over JMS. Figure 1 gives an overview of all components of the DIAMON application.

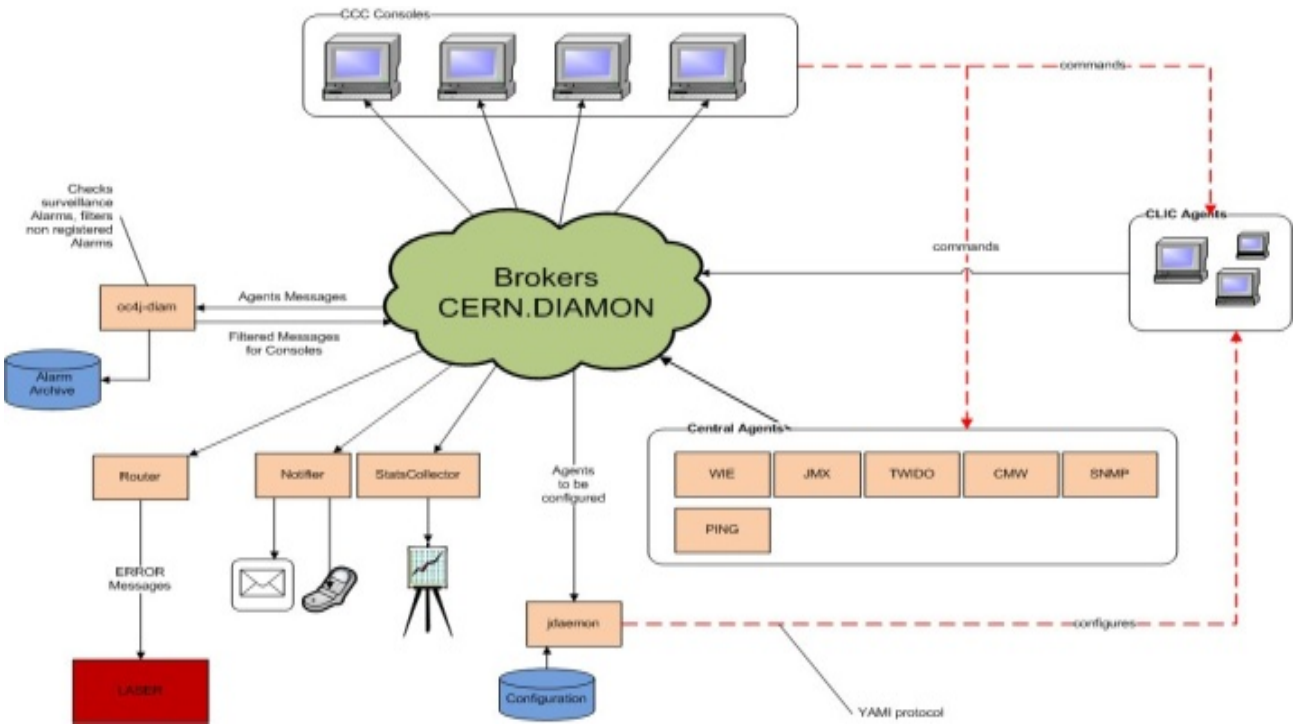

Figure 1: Overview of the DIAMON architecture. Agents (right side of the diagram) send JMS messages to the brokers, which make the information available to the business logic (oc4j-diam) and various clients.

The central component of the DIAMON architecture is the cluster of JMS brokers (center of Figure 1). Various agents (see next paragraph) inform the brokers about the state of the equipment they monitor. The business logic ("oc4j-diam") validates the messages against the configuration data, and prepares the information for use by the client applications, which are:

- The graphical user interface
- The "router", which forwards certain problems to the alarm system.
- The "notifier" in charge of sending mails and SMS messages to subscribers
- The dynamic configuration server ("jdaemon")

The agents, auxiliary clients (i.e. notifier, router, etc.) and graphical user interface were developed especially for DIAMON. The middle-tier (broker and business logic) was provided by the LASER project [3].

DIAMON AGENTS

DIAMON agents are divided in two categories:

- Embedded agents
- Centralized agents

The embedded agents do run on the device they monitor. They collect data about the general system health state (mainly operating system parameters like memory usage, CPU load, etc.) as well as data about connected hardware (timing boards, World FIP bus). As it requires to install software on the equipment to be monitored, this kind of agent is installed on computers only. DIAMON actually uses embedded agents for computers running various flavours of Linux, as well as LynxOS 4.0.

Centralized agents run on a server and poll lists of devices to obtain the information to be monitored. This approach is used to monitor controls infrastructure equipment, where software can not easily be installed, like PLCs, video converters, power supplies etc.

Technically, all DIAMON agents use the same API (the "DIAMON lib") which encapsulates all communication aspects. The library is available for Java (used for centralized agents) and C++ (used the embedded agents and the centralized agent for Schneider and Siemens PLCs). Agent developers need to extend a framework by calling a method for initialization (identify the equipment monitored), for publishing the status of the monitored equipment and provide on request additional details. Certain agents are able to execute commands like restarting a process. The usage of the API is documented on our wiki pages [2].

GRAPHICAL CLIENTS

The data is provided to the end users (mainly the operators in the CERN Control Center) through two different GUIs (Graphical User Interfaces), both based on Java Swing and therefore able to run on almost all platforms (Linux and Windows at CERN).

The Standalone GUI

The main flavour of DIAMON GUI is the "standalone" version, which is designed to cover all functionality and make the use as simple as possible (Figure 1.)

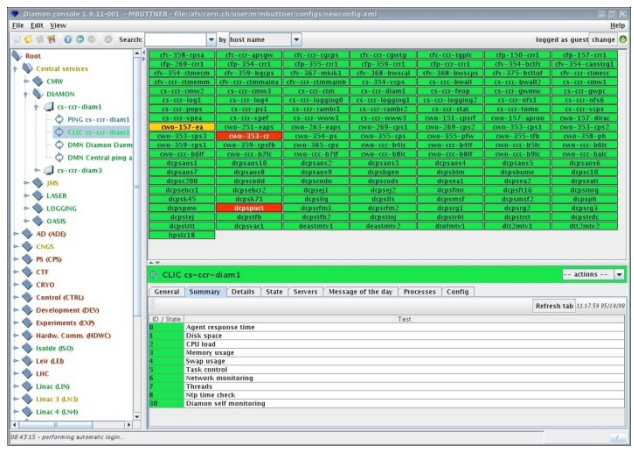

Figure 2: Diamon GUI.

The application window contains three separate panels:

- The tree on the left-hand side provides a hierarchical view on the monitored equipment, for instance with 3 levels: the accelerator, the controlling computer, the connected equipment. Users can configure the levels and content of the tree..
- The top right panel displays the full list of equipment corresponding to the element selected in the tree. The background color clearly displays the status of each item (green: OK; yellow: warning, red: error).
- The bottom right panel (the "Detail panel") displays all details for the element selected in the left hand-side panel (if the selected element is a leaf node) or in the top right panel.

A plugin architecture allows to develop dedicated display components, providing an optimum presentation of the data. For instance, a plugin for WoldFIP monitoring is available (see Figure 3), which displays individually the status of each device connected to the bus.

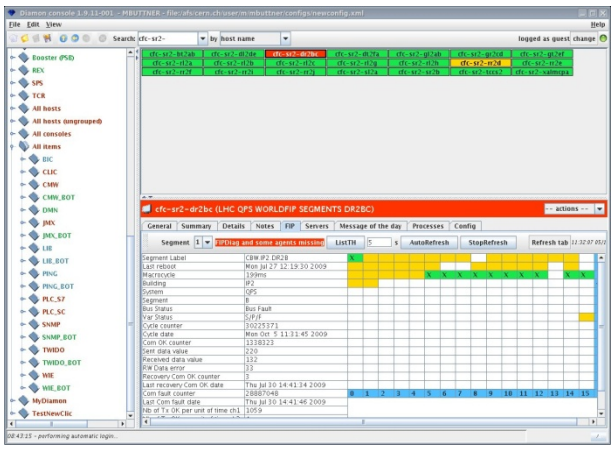

Figure 3: WorldFIP plugin in the Diamon GUI.

The DIAMON Panel in the LASER Console

Certain operators, especially those having already lots of open windows on their screen, prefer a single application providing both alarms and monitoring information. By configuration, a DIAMON panel can appear in the LASER alarms console (Figure 4).

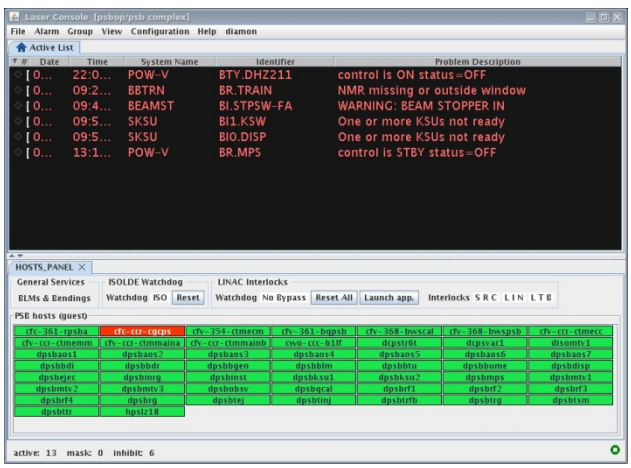

Figure 4: LASER alarms console with DIAMON panel.

In this configuration, the DIAMON panel inserted at the bottom of the LASER is the same as the top right panel of the standalone GUI, with a slightly modified behaviour: As no separate details panel is available, the user can not simply select an item: A double-click is required, which displays the detail panel in the same location.

OUTCOME OF EXPLOITATION

DIAMON is now used in the CERN Control Center for more than 1 year. Approximately 1'500 computers, 300 PLCs, 300 fan trays and many other equipments are actually monitored by DIAMON. Almost the whole controls infrastructure of CERN is visible in the application in a unified way. The tool provides rich functionality: ranging from the display of equipment status to the possibility to reboot frontend computers or restart failing processes. Of course, possible improvements were also reported by the operators:

- The start up time of the DIAMON GUI is perceived as being too long.
- The software suffers from similar problems than LASER [3], which is used by DIAMON as communication infrastructure. For instance, database and network failures can affect the availability of DIAMON.

FUTURE PLANS

The future plans for DIAMON are based on the weaknesses observed during exploitation: the application must become more robust, faster in some areas and should make a better use of CERN standard components.

To achieve this, we mainly plan to develop a new middle-tier to replace LASER. Key elements to the new development should be:

- completely isolate the middle-tier from the database (no direct connection)
- the data-tier to middle-tier communication should use JAPC [4] (Java API for parameter control) instead of the pure JMS approach. JAPC is the recommended API to collect data from devices in CERN's controls infrastructure
- implement the possibility to load/replace definitions in the configuration without restarting the server
- optimize the way configuration data is structured, in order to reduce the start up time of the DIAMON GUI.

REFERENCES

[1] http://wikis.cern.ch/display/alarms/home
[2] http://wikis.cern.ch/display/DIAMON/Home
[3] K. Sigerud, N. Stapley, M.. Misiowiec, T.Zygula, "First operational experience with LASER", ICALEPCS 2005.
[4] http://wikis/display/JAPC/Home