The Power Manager for the LHCb On-Line Farm

Public Note

Issue: 2
Revision: 0

Reference: LHCb-2007-094
Created: March 10, 2006
Last modified: August 6, 2007

Prepared by: Federico Bonifazi[a], Daniela Bortolotti[b], Angelo Carbone[a], Domenico Galli[c], Daniele Gregori[a], Umberto Marconi[b], Gianluca Peco[b], Vincenzo M. Vagnoni[b]

[a]INFN-CNAF, Bologna, I-40127, Italy.
[b]INFN Sezione di Bologna, I-40126, Italy.
[c]Alma Mater Studiorum – Università di Bologna and INFN Sezione di Bologna, Bologna, I-40126, Italy.

Abstract

The Power Manager is a tool of the LHCb **FMC** (Farm Monitoring and Control System) which allows — in an OS-independent manner and without requiring expensive network-controlled power distributors — to **switch** the farm nodes **on** and **off**, and to **monitor** their physical condition: **power status** (on/off), **temperatures**, **fan speeds** and **voltages**. The Power Manager can operate on farm nodes whose motherboards and network interface cards implement the **IPMI** (Intelligent Platform Management Interface) specifications, version 1.5 or subsequent, and copes with several IPMI limitations.

Document Status Sheet

1. Document Title: The Power Manager for the LHCb On-Line Farm			
2. Document Reference Number: LHCb-2007-094			
3. Issue	**4. Revision**	**5. Date**	**6. Reason for change**
Draft	0	March 10, 2006	First version. Abstract only.
Draft	1	May 30, 2007	First complete version. Unchecked.
1	0	July 3, 2007	Refers to FMC version 3.6.10.
2	0	August 6, 2007	Refers to FMC version 3.7.0.

Contents

1 Requirements **4**
1.1 Power Control 4
1.2 Power and Sensor Monitor 4

2 The IPMI Interface **4**
2.1 IPMI Specifications 5
2.2 The IPMI LAN Interface 6
2.3 Proprietary Alternatives to IPMI 7
2.3.1 HP iLO 7
2.3.2 IBM RSA 7
2.3.3 Dell DRAC 8
2.3.4 Sun ALOM 8

3 Implementation **9**
3.1 The DIM Network Communication Layer 9
3.2 Power Manager Server Logics 10
3.2.1 Coping with the long IPMI Response Time 10
3.2.2 Coping with the single command limitation 10
3.2.3 Accessing the IPMI Interface 11
3.2.4 Power Manager Configuration 11
3.2.5 Sensor Classification 12

4 The Power Manager Server 12
4.1 Synopsis 12
4.2 Description 12
4.3 Command Line Options 12
4.4 Environment 13
4.5 Configuration String Format 13
4.5.1 Sample configuration strings 14
4.6 Published DIM Commands and Services 14
4.6.1 CMD: **/⟨NODE_NAME⟩/power_switch** 14
4.6.2 SVC: **/⟨NODE_NAME⟩/power_status** 14
4.6.3 SVC: **/⟨NODE_NAME⟩/power_status_timestamp** 14
4.6.4 SVC: **/⟨NODE_NAME⟩/sensors/current/names** 15
4.6.5 SVC: **/⟨NODE_NAME⟩/sensors/current/units** 15
4.6.6 SVC: **/⟨NODE_NAME⟩/sensors/current/input** 15
4.6.7 SVC: **/⟨NODE_NAME⟩/sensors/current/status** 15
4.6.8 SVC: **/⟨NODE_NAME⟩/sensors/fan/names** 15
4.6.9 SVC: **/⟨NODE_NAME⟩/sensors/fan/units** 15
4.6.10 SVC: **/⟨NODE_NAME⟩/sensors/fan/input** 15
4.6.11 SVC: **/⟨NODE_NAME⟩/sensors/fan/status** 15
4.6.12 SVC: **/⟨NODE_NAME⟩/sensors/temp/names** 16
4.6.13 SVC: **/⟨NODE_NAME⟩/sensors/temp/units** 16
4.6.14 SVC: **/⟨NODE_NAME⟩/sensors/temp/input** 16
4.6.15 SVC: **/⟨NODE_NAME⟩/sensors/temp/status** 16
4.6.16 SVC: **/⟨NODE_NAME⟩/sensors/voltage/names** 16
4.6.17 SVC: **/⟨NODE_NAME⟩/sensors/voltage/units** 16
4.6.18 SVC: **/⟨NODE_NAME⟩/sensors/voltage/input** 16
4.6.19 SVC: **/⟨NODE_NAME⟩/sensors/voltage/status** 16
4.6.20 SVC: **/⟨NODE_NAME⟩/sensors_timestamp** 17
4.6.21 SVC: **/⟨CTRL_PC_NAME⟩/power_manager/success** 17
4.6.22 SVC: **/⟨CTRL_PC_NAME⟩/power_manager/actuator_version** 17
4.6.23 SVC: **/⟨CTRL_PC_NAME⟩/power_manager/server_version** 17

5 The Power Manager Command-Line Clients 17
5.1 The **pwSwitch** Command-Line Client 17
5.1.1 Synopsis 17
5.1.2 Description 17
5.1.3 Command Line Options 18
5.1.4 Environment 18
5.1.5 Warning 18
5.1.6 Examples 18
5.2 The **pwStatus** Command-Line Client 18
5.2.1 Synopsis 18

5.2.2 Description . . . 19
5.2.3 Command Line Options . . . 19
5.2.4 Environment . . . 19
5.2.5 Warning . . . 19
5.2.6 Examples . . . 19
5.2.7 Output sample . . . 19
5.3 The **`ipmiViewer`** Command-Line Client . . . 19
5.3.1 Synopsis . . . 19
5.3.2 Description . . . 20
5.3.3 Command Line Options . . . 20
5.3.4 Environment . . . 20
5.3.5 Warning . . . 20
5.3.6 Examples . . . 20
5.3.7 Output sample . . . 20

6 The Power Manager PVSS Clients . . . 21
6.1 The Power Tab . . . 21
6.2 The Sensor Tabs . . . 21

7 Component versions . . . 23

8 Conclusion . . . 23

9 References . . . 23

List of Figures

1 The IPMI v2.0 Architecture. . . . 5
2 The IPMI LAN Interface. . . . 6
3 Diagram of the Power Manager deployment. . . . 9
4 Diagram of the DIM network communication mechanism. . . . 10
5 The Power Manager PVSS Client: the Power Tab. . . . 22
6 The Power Manager PVSS Client: the Temperature Sensor Tab. . . . 22

List of Tables

1 Versions of the Power Manager components in FMC v. 3.7.0. . . . 23

1 Requirements

1.1 Power Control

The need of a tool to **switch on** and **off** and to **power-cycle** the farm nodes on a large on-line farm — such as the LHCb Event Filter Farm, consisting of ∼2000 PCs — arises from several requirements in the farm administration, e.g.:

- All the farm nodes need to be switched off before a **scheduled power cut** and to be switched on again after power restoration.
- All the farm nodes need to be switched on after a **power failure**.
- All the farm nodes need to be power-cycled if the whole farm must be rebooted in a **different configuration** (e.g. a kernel upgrade).
- A farm node needs to be power-cycled if it hangs-up due to a **hardware defect**, an **OS bug** or a **driver bug**, because in these cases a software reboot cannot be started, since the OS is not responding.
- A farm node needs to be power-cycled if it hangs-up due to a **bug in a user process scheduled by a Linux real-time scheduler** (FIFO or Round-Robin), since the hung process itself can have higher static priority than one of the processes needed to start a software reboot (`xterm(1)`, `bash(1)`, `reboot(8)`). The FMC Task Manager [1] avoids this event in starting new processes, by keeping the static priority of its command thread higher than the ones of the started processes; however it could happen for processes started in a different way.

1.2 Power and Sensor Monitor

Since the LHCb Event Filter Farm nodes are disk-less, the fans are the only mechanical devices on the PCs and thus the devices most prone to failure. Thus a very important issue is to keep all the node **temperatures** and **fan speeds** monitored, in order to detect anomalous conditions in the cooling system and to take the necessary steps as soon as possible to minimise the downtime.

Moreover, in order to supervise the overall status of the farm, the **power status** (on/off) of each node as well as the **voltages** provided by the power supplies have to be surveyed.

2 The IPMI Interface

IPMI [2] (Intelligent Platform Management Interface) is a standardised, abstracted, message-based interface to Intelligent Platform Management hardware which defines records for describing platform management devices and their characteristics.

The IPMI interface can be used, at present (version 2.0):

- to query platform status, e.g. power status (since version 1.5), temperatures, fan speeds, voltages;
- to view hardware logs;
- to access from a remote console the text-based interfaces for BIOS, utilities, operating systems, and applications (since version 2.0);
- to issue other requests, e.g. switch on/off the power (since version 1.5).

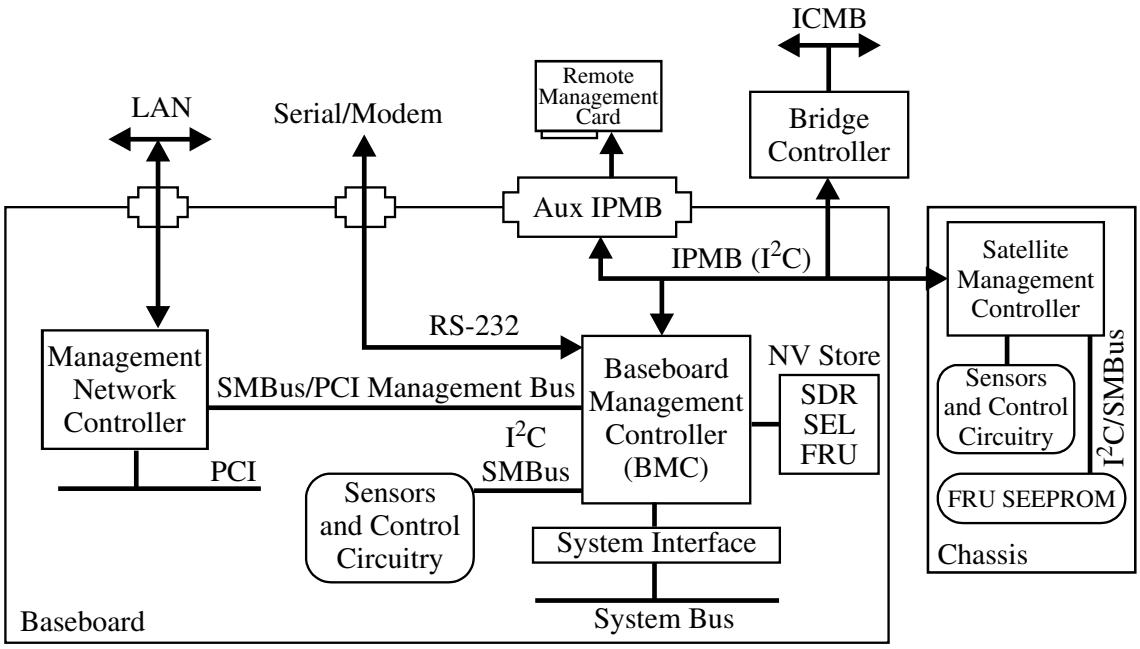

Figure 1 The IPMI v2.0 Architecture.

2.1 IPMI Specifications

The first IPMI specifications were published by Dell, HP, Intel Corporation and NEC Corporation, in 1998 (version 1.0), then refined in 2001 (version 1.5) and 2004 (version 2.0). IPMI is now adopted by ∼200 industries [3]. The IPMI initiative comprises three separate specifications: IPMI [4], the Intelligent Platform Management Bus (IPMB) [5], and the Intelligent Chassis Management Bus (ICMB) [6]. The IPMI specification, which is the main one, defines the messages and system interface to the platform management hardware. The IPMB specification defines an internal management bus for extending platform management within a chassis (intra-platform management). Finally, the ICMB specification defines the external management bus between IPMI-enabled systems (inter-platform management).

The IPMI architecture (Fig. 1) is centred on a main micro-controller, the BMC, i.e. **Baseboard Management Controller**, embedded into the motherboard and powered even when the PC is switched off. The BMC is interfaced through the I^2C (Inter-Integrated Chip specification)/SMBus to the temperature, fan speed and voltage **sensors**, to the **FRU** (Field Replaceable Unit, containing the inventory data, like the manufacturer, the serial number, the date of manufacture, etc., of the devices that are replaceable including power supplies, memory devices, plug-in-boards, etc.), to the **SEL** (System Event Log, containing messages for sensor threshold violations, memory ECC errors, power on/off requests, etc.) and to the **SDR** (Sensor Data Repository, containing the information about the sensors properties of the individual sensors present on the board: sensor address, name, type, unit, threshold).

The BMC can also be connected, through the **IPMB** [5] to Satellite Management Controllers located on a **different chassis board in the same platform**, in turn connected with other sensors (intra-platform management). Moreover, the **ICMB** [6] bridge provides a mechanism for transferring internal messages on the IPMB bus to devices on **different platforms** (inter-platform management).

IPMI can be accessed by using three kinds of interfaces (Fig. 1):

- the **KCS** (Keyboard Controller Style) **interface** (also known as **open interface**), which is a local interface to the host operating system through the system bus, unauthenticated, which can be accessed through the `OpenIPMI` [7] Linux software running on the same node, and thus cannot be used to switch on a PC or to power cycle a hung-up PC;
- the **LAN interface** (since version 1.5) which is a network interface, session-based, authenticated, designed to be always available, even when the system is powered down or when the OS is

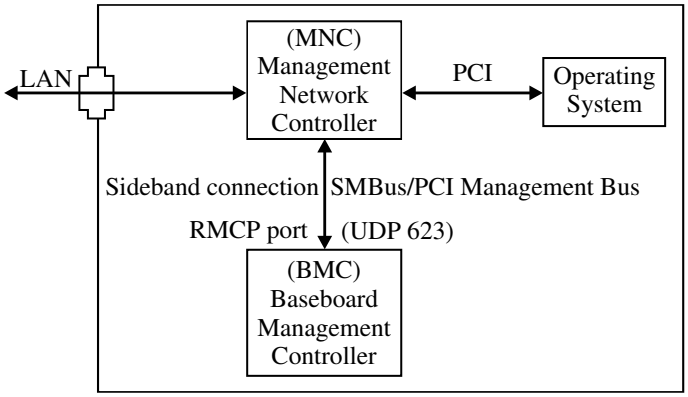

Figure 2 The IPMI LAN Interface.

hung or inactive, and thus is hardware implemented through the SMBus/PCI Management Bus and the **Management Network Controller**, as well as OS independent.

- the **Serial/Modem interface** (since version 1.5), designed to be always available, hardware implemented and OS independent, accessible through a serial RS-232 connection.

The version 1.5 of the IPMI specifications introduces the LAN and Serial/Modem interfaces (by using the RCMP, Remote Control Management Protocol, with basic authentication), the PCI Management Bus, several new sensor types, and an ACPI System Power State sensor to support out-of-band monitoring of the system power state.

The version 2.0 of the IPMI specifications introduces VLAN support, Serial Over LAN — SOL, which provides a mechanism that enables the serial controller of a managed system to be redirected to an IPMI session over IP, enabling remote console applications to access text-based interfaces for BIOS, utilities, operating systems, and applications while simultaneously providing access to IPMI platform management functions — enhanced security authentication and encryption for network packets, incorporating authentication based on SHA-1, Secure Hash Algorithm-1, supporting AES (Advanced Encryption Standard) and using the more robust authentication and encryption provided by RCMP+ **Remote Management and Control Protocol**, and firmware firewall.

2.2 The IPMI LAN Interface

The IPMI messages can be transferred between the BMC and a remote management console over an Ethernet LAN using the User Datagram Protocol (UDP). The IPMI messages are encapsulated in **RMCP** packets at the **Management Network Controller** (MNC) and sent out on the network.

The IPMI LAN interface can be implemented either using a network interface which is **dedicated** to the BMC, or — more typically — using a network interface which is **shared** by the system and the BMC, but with two different MAC addresses for system and BMC. The BMC communicates with the MNC by means of the IPMB bus (either I^2C or SMBus).

The MNC has the capability to detect the packets addressed to the **RMCP port** (port 623), and to deliver these packets to the BMC, while forwarding all the other packets to the system (Fig. 2). The same way, packets sent by the BMC are injected into the network through the shared LAN interface by the MNC, interleaving them with the network packets generated by the system.

The LAN controller is usually designed such that the interface for the management port is **powered by standby power**. This approach enables the BMC to be active and available even when the system is locked up or powered down.

IPMI can be configured by means of the PC start-up configuration utility and can make use of DHCP [8] to set the network parameters.

The IPMI LAN interface needs **no software agent on the server side** (the side of the controlled nodes) since the server has a hardware implementation, while there are several Linux packages for the client (controller) side e.g. IPMItool [9], IPMIutil [10] (formerly known as panicsel), freeIPMI [11], etc..

2.3 Proprietary Alternatives to IPMI

Besides IPMI, there are other **tools** (which are **not**, however, **standardised interfaces**) with similar functionalities, usually vendor-proprietary, generally developed before the IPMI specifications and sometimes providing more functions than IPMI. They include: HP Integrated Lights Out (**iLO**), IBM® Remote Supervisor Adapter (**RSA**), Dell Remote Assistant Cards (**DRAC**) and Sun Advanced Lights Out Management (**ALOM**).

Some vendors are still investing in their proprietary service processors and continue to implement their own proprietary tools (instead of IPMI) on their PCs: this is the case, for example, of several HP ProLiant rack-mountable servers.

It must be remarked that — apart the portability issues which can arise from a proprietary implementation — typically **these proprietary tools cannot be used by a custom application or by a custom script** since they have a closed implementation and **expose only telnet or Web interfaces**. On the contrary, IPMI encapsulates messages in the more suitable RMCP/RMCP+ packets and provides the interface specifications, which are used by several open source tools, (e.g. IPMItool [9], IPMIutil [10] and freeIPMI [11]). Thus, custom IPMI applications can be developed following the IPMI specifications, while custom IPMI scripts (`sh`, `perl`, `python`, etc.) can be developed by exploiting the open source IPMI tools.

2.3.1 HP iLO

There are two versions of iLO (Integrated Lights Out) — basic and advanced. Basic iLO provides access to text consoles only in text mode. Advanced iLO provides remote KVM and virtual media access. Like IPMI, iLO is implemented as a separate chip, an application-specific integrated circuit (ASIC) on the motherboard of the server, and includes a dedicated Ethernet port on the server. iLO remains active either if the server is turned off or on.

iLO has two interfaces - a **Web interface** and a **Telnet interface**. When the server is turned off, the only way to access iLO is via the Web interface. Once logged in to the Web interface, an administrator is able to view the status of many events, such as hardware event logs, perform diagnostics, upgrade the iLO firmware, configure iLO settings, etc.. The user is also able to control the power button, turning the server on or off.

Through the Web interface, the administrator can access the **Java-based** remote console, which provides a view of all activities on the server — from power-on self tests (POSTs) and messages to BIOS, machine boot, boot-time options and kernel loading, etc..

Similar functionality exists through the Telnet interface, but it is only accessible when the server is turned on. The iLO virtual media option provides the administrator with a virtual floppy disk drive and a virtual CD drive that can direct a remote host server to boot and use standard media from anywhere on the network. Virtual media devices are available when the host system is booting.

2.3.2 IBM RSA

RSA (Remote Supervisor Adapter) II provides many options for alerting, monitoring and remote management including notifications, alerts, event logging, storage of the last screen before a failure, virtual disk, power control and Web interface for console management. Like other service processors, RSA II provides sensor, temperature and voltage readings.

RSA II functionality exists on a PCI card that manages the BMC located on the motherboard of a server. A version of the RSA II, called SlimLine, is an internal card that includes the BMC and uses a dedicated Ethernet connector on the server for communication. The IBM BladeCenter uses a form of RSA with an integrated KVM switch for accessing individual server blades.

For secure communication with the RSA II or the BladeCenter management module, especially when using a wide area network (WAN) connection, Secure Sockets Layer (SSL) or Secure Shell (SSH) can be used. RSA II and the BladeCenter management module can act as SSL servers for secure Web server (Secure Hypertext Transfer Protocol [HTTPS]) or as secure LDAP clients (LDAPS) for a LDAP server

like Windows® Active Directory Service (ADS) or Linux OpenLDAP. The SSH feature provides secure access to the CLI and the serial (text console) redirect features.

Users may access RSA II and the BladeCenter management module through a Web browser as both include built-in Web servers. In addition to a **Web interface**, many of IBM xSeries service processors also have a built-in interface that is accessible through **Telnet via Ethernet**, **SSH via Ethernet** or American National Standards Institute (ANSI) **terminal via serial**.

2.3.3 Dell DRAC

Dell was one of the original supporters of IPMI because of its ability to manage servers from many different vendors. Currently, IPMI 1.5 is supported on many Dell PowerEdge servers. However, these servers also include a PCI card slot for the Dell's proprietary service processor solution DRAC (Dell Remote Assistant Card).

DRAC is designed to provide IT administrators with continuous access to servers and full control of the server hardware and operating system from any client workstation running a Web browser. The IT administrator has the ability to manage the server even if the server is down. DRAC allows access to the console via virtual network computing (**VNC**) software. A **Telnet interface** provides only some of the features of the **Web user interface**; however, it does enable console access. DRAC provides alert notification when the system is down and allows full access to the system. In addition, DRAC logs the probable causes of system crashes and saves current error displays. By communicating with the embedded system management hardware, DRAC can report warnings or errors related to voltages, temperatures and fan speeds. DRAC includes software modules that provide a set of operating system-specific services. These services communicate with the DRAC hardware to allow in-band configuration as well as console redirection to the out-of-band connection.

There are three versions of DRAC: DRAC II, DRAC III and DRAC 4. DRAC III provides enhancements in graphical console redirection, POSTs, virtual remote disk, and allows access for up to 16 users per card. DRAC III is a half-length PCI card while DRAC II is a full-length card. For DRAC III, the Dell OpenManage Server Administrator resides on the card, installs the driver and supplies both a GUI and CLI. In order to use DRAC II, Dell OpenManage IT Assistant software must be installed on the client management workstation.

DRAC 4 can be configured to send e-mail alerts for warnings or errors related to voltages, temperatures and fan speeds. DRAC 4 also logs event data and the most recent crash screen (for systems running the Windows operating system only) to help diagnosing the probable cause of a system crash. Depending on the system, DRAC 4 hardware is either a daughter card (DRAC 4/I) or a half-length PCI card (DRAC 4/P).

When accessing the remote service processor system, the DRAC Web console offers the following options for system management: System Health window (displays information such as the DRAC ambient temperature, system firmware version and BIOS version installed, current status of DRAC battery, wall adapter voltage, and PCI bus voltage), System Information window (specifies which system is being accessed and the system BIOS version, baseboard, microprocessor(s), slots, ports and chassis), Event Log window (records remote system events, DRAC events and POST logs), Remote Access window (contains the console redirect service, which provides views of the remote system management screen). From the last window, one can use the management station mouse and keyboard to perform management functions such as system reset, power off and on, power cycle and graceful shutdown. One can also save screen images of the remote system and store the image as a .bmp file or as a .txt file in text-only mode.

2.3.4 Sun ALOM

ALOM (Advanced Lights Out Manager) is a Sun proprietary service processor technology. Referred to as the standard System Controller (SC) for remote out-of-band management for all Sun VSP servers, ALOM 1.5 replaces Remote System Control (RSC) used on VSP servers and Lights Out Management (LOM & LOMlite) used on Sun Netra servers. ALOM SC hardware is embedded in the server and works independently as a separate embedded computer. ALOM firmware is pre-installed by default;

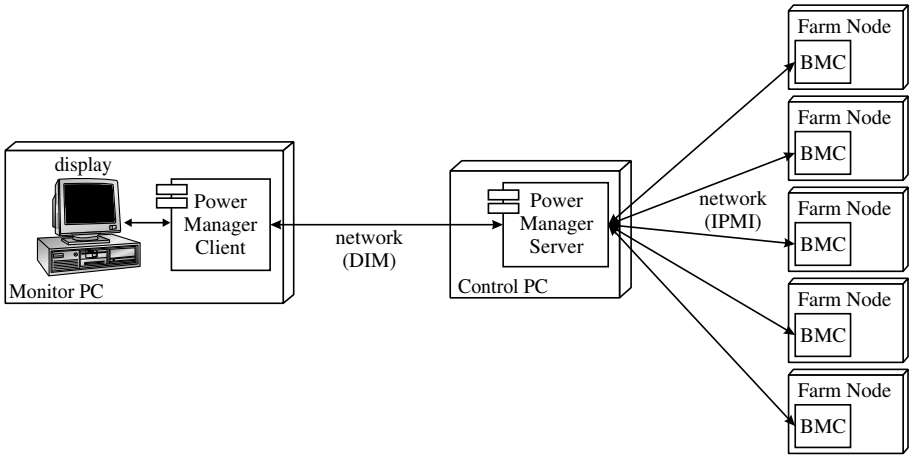

Figure 3 Diagram of the Power Manager deployment.

an **external ASCII terminal** can be connected to the **serial management port** and ALOM functionality is available immediately without the need to configure ALOM software.

ALOM functions allow for monitoring, logging, alerting and for basic control of the system. Multiple users can monitor the host server, but only one user at a time can issue any command that requires permissions. ALOM is particularly useful for remotely managing a server in a typical "lights out" environment. ALOM monitors the following server components: disk drives (if each slot has a drive present and slot status), fans (fan speed and status), CPUs (if a CPU is present, the temperature at the CPU and any thermal warning or failure conditions), power supplies (if each bay has a power supply present, and whether it reports OK status), system enclosure temperature (system ambient temperature, enclosure thermal warning or failure conditions), circuit breakers and voltages (if circuit breakers have been tripped, and if correct voltages are reported), server front panel (system key-switch position and status of LEDs).

3 Implementation

The Power Manager is a tool to switch-on, switch-off, power-cycle and show the power status and the sensor information retrieved through the I^2C bus (e.g. temperatures, fan speeds, voltages, etc.) of every farm node from a central console.

The Power Manager Server runs on a few *control PCs*, each one watching $\sim$200 farm nodes (Fig. 3). To communicate with the Power Manager Clients (which send commands and get the status), the Power Manager Server uses **DIM** [12] (Distributed Information Management System) as server, while to communicate with nodes it uses **IPMI** as client.

3.1 The DIM Network Communication Layer

DIM is a network communication layer built on top of the TCP protocol and based on a client/server paradigm, making use of a name server (Fig. 4).

The *server* provides *data services* and *command services* to clients by "registering" them with the *name server* (normally once, at startup). The *client* "subscribes" to services by asking the name server which server provides the service and then contacting the server directly.

The update of the data on the client side can happen ONCE_ONLY (the client will receive the data once and then will disconnect from the server), TIMED (the client will receive the data at regular time intervals) or MONITORED (the client will receive the data only when the server updates them).

The client can also send a command to a command service to require the execution of a procedure on the server side.

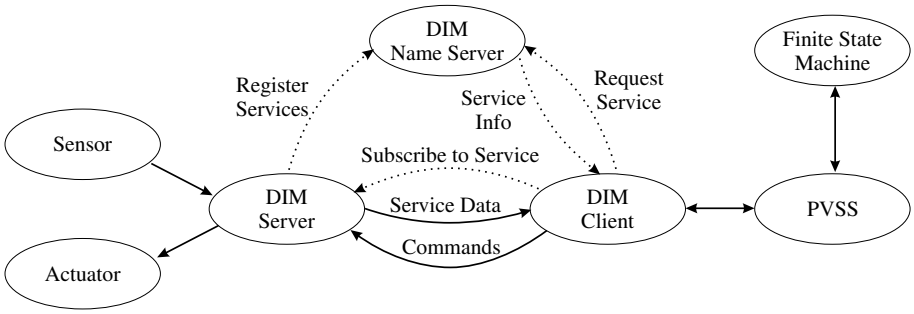

Figure 4 Diagram of the DIM network communication mechanism.

3.2 Power Manager Server Logics

The Power Manager Server is a DIM server which interfaces DIM with the IPMI LAN interface and copes with several IPMI limitations. It consists of 3 long-living threads (the **main thread**, the **I/O thread** and the **timer thread**) and many **short-living threads** started to access the IPMI interfaces.

The **main thread** performs the main loop, which **periodically** starts short-living threads (one thread for each controlled node) to **query** the controlled nodes for the power state and the sensor measurements. The update period varies between 4 s and 60 s, increasing by 20% at every iteration and it is reset to 4 s every time a power switch command is issued. This way, in normal operation, the power and sensor data are updated once in a minute, while, soon after a power-on/off command is issued, the power and sensor data are updated more frequently to report soon the status change.

The **I/O thread** manages DIM commands and services. To update the **DIM services** it simply sends the last stored measurements, retrieved by the main thread. To perform a **DIM command** it starts a short-living thread for each controlled node to send the IPMI commands.

3.2.1 Coping with the long IPMI Response Time

The first IPMI limitation (at least in version 1.5) is the **long IPMI response time**, which can be long up to 0.7 s/node on the Dell PowerEdge SC 1450 computer used as test-bed, which means 23 minutes to access once, sequentially, all the 2000 nodes. The situation can even be worse if a few nodes are disconnected (but their names are still resolved by the DNS): as a matter of fact the **connection timeout** would be ~ 16 s/node.

The Power Manager Server copes with the long IPMI response time by parallelly accessing each node from a different short-living thread. Every IPMI access request starts as many short-living threads as the number of the controlled nodes and each short-living thread access one node only.

Between two subsequent periodic IPMI sensor accesses, the performed measurements together with their timestamps are kept stored in a runtime database by the main thread, so that the Power Manager Server can immediately answer to a Power Manager Client request.

3.2.2 Coping with the single command limitation

The second IPMI limitation is the ability to process **only one command at a time**. While the BMC is processing a command it is unable either to process another command or to enqueue another command for a deferred execution: simply, the second command fails immediately.

Overlapping accesses to IPMI can happen, for example, if a power-switch command is issued while the BMC is working to retrieve the sensor information for a periodical update. The net effect could be that a power-switch command is actually performed only on a part of the nodes (those having the BMC free), while it fails and thus it is not performed on the other part of the nodes (those having the BMC busy).

The Power Manager Server copes with this limitation by *arbitrating* among the IPMI commands sent to the same node. In case of overlapping commands or service updates, the Power Manager Server

enqueues commands, in order for all the received commands to be executed, one at a time, exactly in the same order they were issued, and *cancel periodical sensor updates*, in order to avoid indefinite thread pile-up for not-responding IPMI interfaces.

Technically speaking, the Power Manager keeps the array of booleans `bmc.busy[ctrldNodeN]` (one boolean for each controlled node) containing the busy state of all the node BMCs. The BMC of the i-th node is defined as busy (`bmc.busy[i]=1`) since just before an IPMI command is sent to the node until when the IPMI response is received from the node. The access to the `bmc.busy` array is protected by a mutex (`bmc.mutex`) and any change of the array values is signalled to the other waiting threads by means of a condition variable (`bmc.cond`).

If the *main thread* finds the BMC busy on the i-th node it does not start the periodical power-service and sensor-service short-living threads on the node, i.e. sensor updates are skipped. The *I/O thread*, on the contrary, starts the power-command short-living thread anyhow, even if the BMC is busy.

If a started power-command short-living thread trying to access the i-th node finds the node BMC busy the thread identifier is added to the queue (linked list) `bmc.cmdQ[i]` and the thread waits (`pthread_cond_wait(3)` [13]) until `bmc.busy[i]=0` AND the thread identifier is the first of the queue `bmc.cmdQ[i]`.

If a started power-service or sensor-service short-living thread trying to access the i-th node finds the node BMC busy the thread identifier is added to the linked list `bmc.pwSvcQ[i]` or `bmc.sensSvcQ[i]` and the thread waits until `bmc.busy[i]=0` AND the length of the linked list `bmc.cmdQ[i]` is zero (no power command enqueued) AND the thread identifier is the first of the queue `bmc.pwSvcQ[i]` or `bmc.sensSvcQ[i]`.

3.2.3 Accessing the IPMI Interface

In the first FMC releases (until version 1.2), the Power Manager Server was linked to the IPMItool `libintf_lan.so` library, version 1.5.9. The library was patched in order to become thread-safe (global variables were substituted by local variables passed by reference to the functions, timeout mechanisms based on `signals(7)` [14] and `longjmps(3)` [15] were substituted with different timeout mechanisms, etc.).

Since then, many versions of the IPMItool library have been released — the last one being the 1.8.9, released on March 6, 2007 — and thus it has become very difficult to maintain on the new releases the patches to make IPMItool thread-safe.

We decided therefore — waiting for a maintained thread-safe IPMI library — to execute the `ipmitool` command in a different process, getting its output through a pipe (`popen(3)` [16]).

3.2.4 Power Manager Configuration

The Power Manager Server *Configuration String*, i.e. the list of the nodes to be controlled, together with the parameters needed to access them, can be loaded by `ipmiSrv` in 2 different ways:

- through a file on a file system mounted on the machine running `ipmiSrv`, by specifying the `-c conf_file` command line option (e.g.: `-c /etc/ipmiConfFile.txt`);
- through the DIM service `/⟨CTRL_PC_NAME⟩/config_manager/ipmi/get` provided by the FMC Configuration Manager `cmSrv`, if the `-c conf_file` option is not specified on the `ipmiSrv` command line.

In the former case the file `conf_file` contains the ASCII Configuration String.

In the latter case the Configuration Manager provides the ASCII Configuration String to the Power Manager. If the Power Manager is started before the Configuration Manager, it waits for the Configuration Manager to be running and configured.

The latter option is useful to set the Power Manager Configuration through PVSS.

3.2.5 Sensor Classification

The sensor names are different in different PCs, and very different in PCs from different brands [a].

The sensor classification, i.e. the grouping of sensor in temperature sensors, fan sensors, etc. can be retrieved from the IPMI **SDR**, but this option can delay up to 300 seconds the server start-up in the worst case (node not responding but resolved by DNS).

To avoid such a delay every time the Power Manager Server is started, in its first execution the Power Manager Server stores the sensor classification retrieved by the SDR in a file on the control PC; in the subsequent execution the sensor classification will be loaded from that file, unless the `-d` switch is specified in the server start-up command line (this is required if a different PC is added to the nodes controlled by the control PC).

4 The Power Manager Server

The Power Manager Server, whose executable image name is **`ipmiSrv`**, runs on the Control Nodes of the farm. Each server typically controls ∼200 farm nodes. The Power Manager Configuration String (i.e. the list of the nodes controlled by the each control node together with the parameters needed to access them) should be available either through the FMC Configuration Manager or through the file **`conf_file`**.

4.1 Synopsis

```
ipmiSrv[-l logger_unit][-c conf_file][-s sens_file][-d]
ipmiSrv[-h]
```

4.2 Description

Starts the Power Manager Server **`ipmiSrv`** on the current control node and sends its diagnostic messages to a logger unit.

4.3 Command Line Options

`-h` — Get basic usage help from the command line and exit.

`-l logger_unit` — Send diagnostic messages to the logger units defined in the logger mask, which is the bitwise OR of the following values:

- **`0x1`** — **`L_DIM`**, the default FMC Event Logger (**`/tmp/logSrv.fifo`**);
- **`0x2`** — **`L_STD`**, the standard error stream;
- **`0x4`** — **`L_SYS`**, the Linux **`syslog`** facility.

By default, `ipmiSrv` does not send diagnostic messages.

`-c conf_file` — Read the Power Manager Configuration String (i.e. the list of the farm nodes controlled by the current Power Manager, running on the current node, together with the parameters needed to access them) from the file **`conf_file`**. If this option is not specified, the configuration is read through the DIM Configuration Manager **`cmSrv`**, which must be running on this host.

`-s sens_file` — Read and write the sensor dictionary from/to the file **`sens_file`**. Default: use the file **`/etc/ipmi_sensor_dictionary.txt`**.

`-d` — Discover sensor dictionary by using IPMI SDR and update the dictionary file **`sens_file`**.

[a] For example, the temperature sensors in a Dell PC PowerEdge SC 1450 are named: `Temp`, `Planar Temp`, `VRD 0 Temp` and `VRD 1 Temp`, while in a Sun PC SunFire W20z they are named `ambienttemp`, `cpu0.dietemp`, `cpu0.memtemp`, `cpu1.dietemp`, `cpu1.memtemp`, `gbeth.temp`, `hddbp.temp` and `sp.temp`.

4.4 Environment

DIM_DNS_NODE *(mandatory)* — **hostname.domain** of DIM dns node.

LD_LIBRARY_PATH *(mandatory)* — Variable, in PATH format, which must contain the path to the shared libraries **libdim.so** and **libFMCutils.so**.

IPMI_USER *(mandatory)* — The default username used to access IPMI. Can be overridden by the per-host specifications provided by the Configuration String.

IPMI_PASSWD *(mandatory)* — The default password used to access IPMI. Can be overridden by the per-host specifications provided by the configuration file/service.

IPMI_CMD *(optional)* — The path to the `ipmitool` executable image (by default set to: **/usr/bin/ipmitool**).

deBug — The debug mask (default: 0), which is the bitwise OR of the following values:

- **0x01** — print update period;
- **0x02** — print retrieved data;
- **0x04** — print command queue;
- **0x08** — print output of `ipmitool` commands;
- **0x10** — print function start/end;
- **0x20** — print published services.
- **0x40** — print sent `ipmitool` commands.

4.5 Configuration String Format

The ASCII Configuration String — which can be loaded by a Configuration File (`-c option`) or retrieved through the Configuration Manager — consists of several *records* (one record for each controlled node), separated by the '**;**' (semicolon) or by the '**\n**' (newline) ASCII character.

Each record consists in turn of several *fields*, separated by the ',' (comma) ASCII character.

Blank lines as well as comment lines (lines starting with the '**#**' character) are skipped.

The string '**NULL**' can be used to omit a field in the configuration string.

The record has the format:

hostName,userName,passWord,port,authType,privLvl,oemType

where:

hostName *(mandatory)* — is the hostname of the controlled node.

userName *(optional)* — is the username used on the controlled node.

passWord *(optional)* — is the password used on the controlled node.

port *(optional)* — is the UDP port contacted on the controlled node.

authType *(optional)* — is the authentication type to use during IPMIv1.5 lan session activation on the controlled node. Supported types are: `NONE`, `PASSWORD`, `MD2`, `MD5`, or `OEM`.

privLvl *(optional)* — is the privilege level used on the controlled node. Allowed level are: `CALLBACK`, `USER`, `OPERATOR`, `ADMINISTRATOR`. Default level: `ADMINISTRATOR`.

oemType *(optional)* — is the OEM type to support on the controlled node. This usually involves minor hacks in place in the code to work around quirks in various BMCs from various manufacturers. Supported types are: `supermicro`, `intelwv2`, `intelplus`, `icts`, `ibm`.

4.5.1 Sample configuration strings

Example 1:

```
# Sample configuration file
# hostName,userName,passWord,port,authType,privLvl,oemType
farm0101
farm0102,myuser
farm0102,noUser
farm0103,noUser,mypassword
farm0104,myuser,mypassword,1623
farm0105,myuser,mypassword,1623,MD2,OPERATOR,intelwv2
farm0106,NULL,NULL,NULL,MD2,OPERATOR,intelwv2
```

Example 1:

```
farm0101;farm0102,myuser;farm0103,noUser,mypassword
```

4.6 Published DIM Commands and Services

4.6.1 CMD: /⟨NODE_NAME⟩/power_switch

- Command String Synopsis:
  on|off|cycle|soft_off|hard_reset|pulse_diag
- Description:
  Performs an IPMI chassis control command to change the power state of the node ⟨**NODE_NAME**⟩.
- Command String Arguments:

  on — Power-up the node.

  off — Power down the node into soft off (S4/S5 state)[b]. This command does not initiate a clean shutdown of the operating system prior to powering down the system.

  cycle — Provides a power off interval of at least 1 second. No action should occur if the node power is in S4/S5 state, but it is recommended to check power state first and only issue a power cycle command if the system power is on or in lower sleep state than S4/S5.

  soft_off — Initiate a soft-shutdown of the OS via ACPI. This can be done in a number of ways, commonly by simulating an overtemperture or by simulating a power button press. It is necessary to have OS support for ACPI and some sort of daemon watching for soft power events.

  hard_reset — Pulse the system reset signal.

  pulse_diag — Pulse a diagnostic interrupt (NMI) directly to the processor(s). This is typically used to cause the operating system to do a diagnostic dump (OS dependent).

4.6.2 SVC: /⟨NODE_NAME⟩/power_status

- Description:
  Returns the power state of the node as an integer: **0**=off, **1**=on.

4.6.3 SVC: /⟨NODE_NAME⟩/power_status_timestamp

- Description:
  Returns the timestamp corresponding to the last node power state detected, as an integer (**time_t**) measured in seconds since the Epoch (00:00:00 UTC, January 1, 1970).

[b] S0/G0=working, S1=processor/chip set clocks stopped, S2=stopped clocks with processor/cache context lost, S3=suspend-to-RAM, S4=suspend-to-disk, S5/G2=soft off, G3=mechanical off [17].

4.6.4 SVC: /⟨NODE_NAME⟩/sensors/current/names

- Description:
  Returns an array of NULL-terminated strings, containing the list of the current names.

4.6.5 SVC: /⟨NODE_NAME⟩/sensors/current/units

- Description:
  Returns an array of NULL-terminated strings, containing the list of the units as to measure the current, in the same sequence used in the /⟨NODE_NAME⟩/sensors/current/names service.

4.6.6 SVC: /⟨NODE_NAME⟩/sensors/current/input

- Description:
  Returns an array of floats, containing the measured current in the units specified in the /⟨NODE_NAME⟩/sensors/current/units service in the same sequence as in the /⟨NODE_NAME⟩/sensors/current/names service.

4.6.7 SVC: /⟨NODE_NAME⟩/sensors/current/status

- Description:
  Returns an array of NULL-terminated strings, containing the list of the alarm status of the current sensors, in the same sequence as in the /⟨NODE_NAME⟩/sensors/current/names service. The threshold used to evaluate the alarm status are those established by the PC manufacturer and stored in the IPMI firmware.

4.6.8 SVC: /⟨NODE_NAME⟩/sensors/fan/names

- Description:
  Returns an array of NULL-terminated strings, containing the list of the fan names.

4.6.9 SVC: /⟨NODE_NAME⟩/sensors/fan/units

- Description:
  Returns an array of NULL-terminated strings, containing the list of the units used to measure the fan speeds, in the same sequence as in the /⟨NODE_NAME⟩/sensors/fan/names service.

4.6.10 SVC: /⟨NODE_NAME⟩/sensors/fan/input

- Description:
  Returns an array of integers, containing the measured fan speeds in the units specified in the /⟨NODE_NAME⟩/sensors/fan/units service, in the same sequence as in the /⟨NODE_NAME⟩/sensors/fan/names service.

4.6.11 SVC: /⟨NODE_NAME⟩/sensors/fan/status

- Description:
  Returns an array of NULL-terminated strings, containing the list of the alarm status of the fan speeds sensors[c], in the same sequence as in the /⟨NODE_NAME⟩/sensors/fan/names service. The threshold used to evaluate the alarm status are those established by the PC manufacturer and stored in the IPMI firmware.

[c] Possible alarm status strings are: **cr**=critical, **nc**=non-critical, **nr**=non-recoverable, **ns**=not specified (e.g. node switched off), **ok**=ok, **us**=unspecified.

4.6.12 SVC: /⟨NODE_NAME⟩/sensors/temp/names

- Description:
  Returns an array of NULL-terminated strings, containing the list of the temperature names.

4.6.13 SVC: /⟨NODE_NAME⟩/sensors/temp/units

- Description:
  Returns an array of NULL-terminated strings, containing the list of the units used to measure the temperatures, in the same sequence as in the /⟨NODE_NAME⟩/sensors/temp/names service.

4.6.14 SVC: /⟨NODE_NAME⟩/sensors/temp/input

- Description:
  Returns an array of floats, containing the measured temperatures in the units specified in the /⟨NODE_NAME⟩/sensors/temp/units service in the same sequence as in the /⟨NODE_NAME⟩/sensors/temp/names service.

4.6.15 SVC: /⟨NODE_NAME⟩/sensors/temp/status

- Description:
  Returns an array of NULL-terminated strings, containing the list of the alarm status of the temperature sensors, in the same sequence as in the /⟨NODE_NAME⟩/sensors/temp/names service. The threshold used to evaluate the alarm status are those established by the PC manufacturer and stored in the IPMI firmware.

4.6.16 SVC: /⟨NODE_NAME⟩/sensors/voltage/names

- Description:
  Returns an array of NULL-terminated strings, containing the list of the voltage names.

4.6.17 SVC: /⟨NODE_NAME⟩/sensors/voltage/units

- Description:
  Returns an array of NULL-terminated strings, containing the list of the units as to measure the voltages, in the same sequence used in the /⟨NODE_NAME⟩/sensors/voltage/names service.

4.6.18 SVC: /⟨NODE_NAME⟩/sensors/voltage/input

- Description:
  Returns an array of floats, containing the measured voltages in the units specified in the /⟨NODE_NAME⟩/sensors/voltage/units service in the same sequence as in the /⟨NODE_NAME⟩/sensors/voltage/names service.

4.6.19 SVC: /⟨NODE_NAME⟩/sensors/voltage/status

- Description:
  Returns an array of NULL-terminated strings, containing the list of the alarm status of the voltage sensors, in the same sequence as in the /⟨NODE_NAME⟩/sensors/voltage/names service. The threshold used to evaluate the alarm status are those established by the PC manufacturer and stored in the IPMI firmware.

4.6.20 SVC: /⟨NODE_NAME⟩/sensors_timestamp

- Description:
  Returns the timestamp corresponding to the last sensor state detected, as an integer (**time_t**) measured in seconds since the Epoch (00:00:00 UTC, January 1, 1970).

4.6.21 SVC: /⟨CTRL_PC_NAME⟩/power_manager/success

- Description:
  This dummy service always returns 1. It is used by PVSS-DIM [18] to check if the `ipmiSrv` process is running.

4.6.22 SVC: /⟨CTRL_PC_NAME⟩/power_manager/actuator_version

- Description:
  Returns the RCS Identification string of the Power Manager Server actuator function source file, containing version and last modification time.

4.6.23 SVC: /⟨CTRL_PC_NAME⟩/power_manager/server_version

- Description:
  Returns the RCS Identification string of the Power Manager Server main program source file, containing version and last modification time.

5 The Power Manager Command-Line Clients

5.1 The pwSwitch Command-Line Client

5.1.1 Synopsis

```
pwSwitch [-m node_pattern] [-f] [-d delay_time] on|off|cycle|soft_off|
         hard_reset|pulse_diag
pwSwitch [-h]
```

5.1.2 Description

The **pwSwitch** command issues the IPMI **chassis power** command **on**, **off**, **cycle**, **soft_off**, **hard_reset** or **pulse_diag**, described in section 4.6.1 at page 14, to the farm nodes whose hostname matches the wildcard pattern [19] **node_pattern**.

If the command-line option **-m node_pattern** is omitted, the command is issued to **all** the farm nodes (the command **pwSwitch on** is equivalent to the command **pwSwitch -m * on**).

If one or more wildcards are used for **node_pattern** (or the command-line option **-m node_pattern** is omitted) the **pwSwitch** application asks for confirmation before starting to dispath the command, unless the **-f** command line switch is specified:

```
[online@lhcbmonitor ~]$ pwSwitch -m "farm010?" off
Going to power off the following nodes:
    1: farm0100
    2: farm0101
    3: farm0102
    4: farm0103
    5: farm0104
```

```
    6: farm0105
    6: farm0106
    8: farm0107
    9: farm0108
   10: farm0109
Sure you want to power off all the nodes above [yn]?
```

5.1.3 Command Line Options

-h — Print a help message and exit.

-m node_pattern — Issue the IPMI **chassis power** commands *only* to the farm nodes whose hostname matches the wildcard pattern [19] **node_pattern**.

-f — Do not ask for confirmation even if one or more wildcards are used for **node_pattern**.

-d delay_time — Interpose a delay of **delay_time** seconds (floating point value) between the dispatch of the same command to two different nodes (default: no delay).

5.1.4 Environment

The program **pwSwitch** needs the two environment variables:

DIM_DNS_NODE — **hostname.domain** of DIM dns node.

LD_LIBRARY_PATH — Variable, in PATH format, which must contain the path to the shared libraries **libdim.so** and **libFMCutils.so**.

5.1.5 Warning

The *wildcards* [19] in the command line must be *escaped* (by means of a *back-slash* or by means of a couple of *double quotation marks*) in order to avoid their shell expansion.

5.1.6 Examples

```
pwSwitch on
pwSwitch off
pwSwitch cycle
pwSwitch -f cycle
pwSwitch -f -d 0.1 on
pwSwitch -m farm0301 on
pwSwitch -m "farm030*" off
pwSwitch -m farm030\* cycle
pwSwitch -m "farm030[3-7]?" on
pwSwitch -m "farm030[1357]" off
pwSwitch -m "farm03[1357]1" off
pwSwitch -m "farm03[1357]?" off
```

5.2 The pwStatus Command-Line Client

5.2.1 Synopsis

pwStatus[-m node_pattern]

pwStatus[-h]

5.2.2 Description

Returns the power state of the nodes whose hostname matches the wildcard pattern [19] **`node_pattern`**. If the command-line option **`-m node_pattern`** is not specified, the power state of **all** the farm nodes is returned.

5.2.3 Command Line Options

`-h` — Print a help message and exit.

`-m node_pattern` — Returns only the power status of the farm nodes whose hostname matches the wildcard pattern [19] **`node_pattern`**.

5.2.4 Environment

The program **`pwStatus`** needs the two environment variables:

`DIM_DNS_NODE` — **`hostname.domain`** of DIM dns node.

`LD_LIBRARY_PATH` — Variable, in `PATH` format, which must contain the path to the shared libraries **`libdim.so`** and **`libFMCutils.so`**.

5.2.5 Warning

The *wildcards* [19] in the command line must be *escaped* (by means of a *back-slash* or by means of a couple of *double quotation marks*) in order to avoid their shell expansion.

5.2.6 Examples

```
pwStatus
pwStatus -m farm0301
pwStatus -m "farm030*"
pwStatus -m farm030\*
pwStatus -m "farm030[3-7]"
pwStatus -m "farm030[1357]"
pwStatus -m "farm03[1357]1"
pwStatus -m "farm03[1357]?"
```

5.2.7 Output sample

```
[online@lhcbmonitor ~]$ pwStatus -m "farm030[3-7]"

Power status of Farm Nodes:
farm0303: on (Thu May 17 15:44:13 2007)
farm0304: on (Thu May 17 15:44:26 2007)
farm0305: on (Thu May 17 15:43:53 2007)
farm0306: off (Thu May 17 15:44:18 2007)
farm0307: on (Thu May 17 15:43:54 2007)
```

5.3 The `ipmiViewer` Command-Line Client

5.3.1 Synopsis

`ipmiViewer[-m node_pattern]`

`ipmiViewer[-h]`

5.3.2 Description

Returns the sensor measurements (together with units and alarm status[d]) of the nodes whose hostname matches the wildcard pattern [19] **`node_pattern`**. If the command-line option **`-m node_pattern`** is not specified, the sensor measurements of **all** the farm nodes is returned.

5.3.3 Command Line Options

`-h` — Print a help message and exit.

`-m node_pattern` — Returns only the sensor measurements of the farm nodes whose hostname matches the wildcard pattern [19] **`node_pattern`**.

5.3.4 Environment

The program **`ipmiViewer`** needs the two environment variables:

`DIM_DNS_NODE` — **`hostname.domain`** of DIM dns node.

`LD_LIBRARY_PATH` — Variable, in `PATH` format, which must contain the path to the shared libraries **`libdim.so`** and **`libFMCutils.so`**.

5.3.5 Warning

The *wildcards* [19] in the command line must be *escaped* (by means of a *back-slash* or by means of a couple of *double quotation marks*) in order to avoid their shell expansion.

5.3.6 Examples

```
ipmiViewer
ipmiViewer -m farm0301
ipmiViewer -m "farm030*"
ipmiViewer -m farm030\*
ipmiViewer -m "farm030[3-7]?"
ipmiViewer -m "farm030[1357]"
ipmiViewer -m "farm03[1357]1"
ipmiViewer -m "farm03[1357]?"
```

5.3.7 Output sample

```
[online@lhcbmonitor ~]$ ipmiViewer -m "farm030[3-4]"

NODE: farm0303
  Power status (Thu May 17 15:54:13 2007):
    ON
  Temperatures (Thu May 17 15:54:01 2007):
    Temp              42.0   degrees C        ok
    Temp              47.0   degrees C        ok
    Planar Temp       35.0   degrees C        ok
    VRD 0 Temp        29.0   degrees C        ok
    VRD 1 Temp        31.0   degrees C        ok
  Fan speeds (Thu May 17 15:54:01 2007):
```

[d] Possible alarm status strings are: **`cr`**=critical, **`nc`**=non-critical, **`nr`**=non-recoverable, **`ns`**=not specified (e.g. node switched off), **`ok`**=ok, **`us`**=unspecified.

```
    Fan 1            3450     RPM             ok
    Fan 2A           8100     RPM             ok
    Fan 2B           5850     RPM             ok
    Fan 3A           8025     RPM             ok
    Fan 3B           5700     RPM             ok
    Fan 4A           7950     RPM             ok
    Fan 4B           5775     RPM             ok
    Fan 5A           8100     RPM             ok
    Fan 5B           5925     RPM             ok
  Voltages (Thu May 17 15:54:01 2007):
    CMOS Battery       3.083 Volts           ok
  Currents (Thu May 17 15:54:01 2007):
    (No current measurements available on the node)

NODE: farm0304
  Power status (Thu May 17 15:54:24 2007):
    ON
  Temperatures (Thu May 17 15:54:11 2007):
    Temp              48.0   degrees C       ok
    Temp              46.0   degrees C       ok
    Planar Temp       37.0   degrees C       ok
    VRD 0 Temp        24.0   degrees C       ok
    VRD 1 Temp        33.0   degrees C       ok
  Fan speeds (Thu May 17 15:54:11 2007):
    Fan 1            3560     RPM             ok
    Fan 2A           8670     RPM             ok
    Fan 2B           5850     RPM             ok
    Fan 3A           8565     RPM             ok
    Fan 3B           5850     RPM             ok
    Fan 4A           7370     RPM             ok
    Fan 4B           5985     RPM             ok
    Fan 5A           8550     RPM             ok
    Fan 5B           5295     RPM             ok
  Voltages (Thu May 17 15:54:11 2007):
    CMOS Battery       3.081 Volts           ok
  Currents (Thu May 17 15:54:11 2007):
    (No current measurements available on the node)
```

6 The Power Manager PVSS Clients

The Power Manager PVSS Client consists of a panel, showing the node SMC alarm status and including 4 tabs: Power, Temperature, Fan and Voltage.

6.1 The Power Tab

The Power Tab (Fig. 5) reports the node *power status* and the *timestamp* corresponding to the reported status. It is also provided of *4 buttons*: the first pair issues a system software *halt* and *reboot* through the FMC Task Manager and the standard Linux `shutdown` command; the second pair hard switches the node *on* or *off* through the FMC Power Manager and IPMI.

6.2 The Sensor Tabs

The Sensor Tabs (the Temperature Sensor Tab is shown in Fig. 6) show the sensor names, the sensor reported values, the sensor state evaluated by IPMI, using the thresholds set by the PC manufacturer and stored in the IPMI firmware together with the corresponding timestamp.

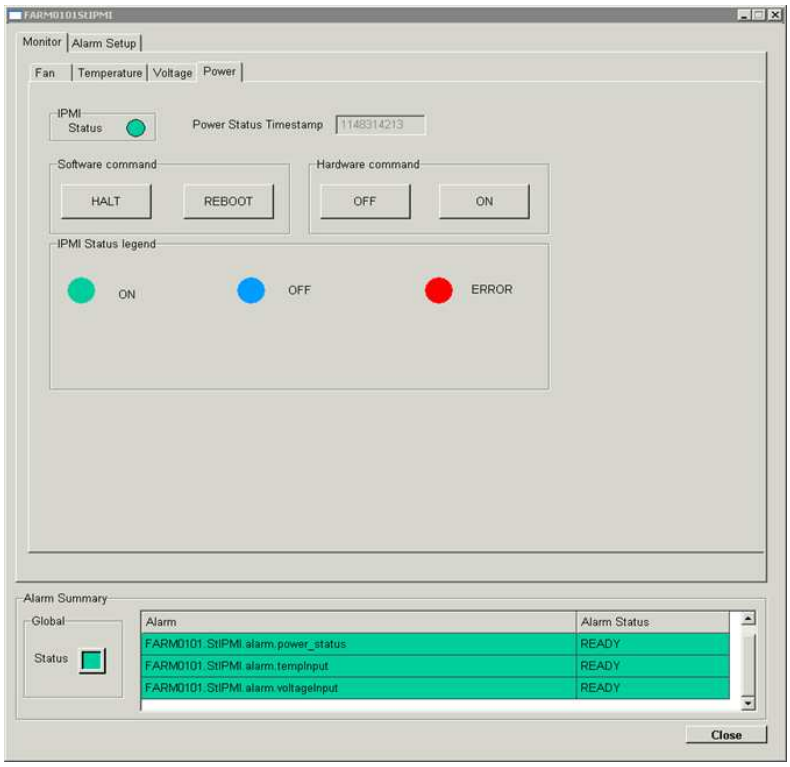

Figure 5 The Power Manager PVSS Client: the Power Tab.

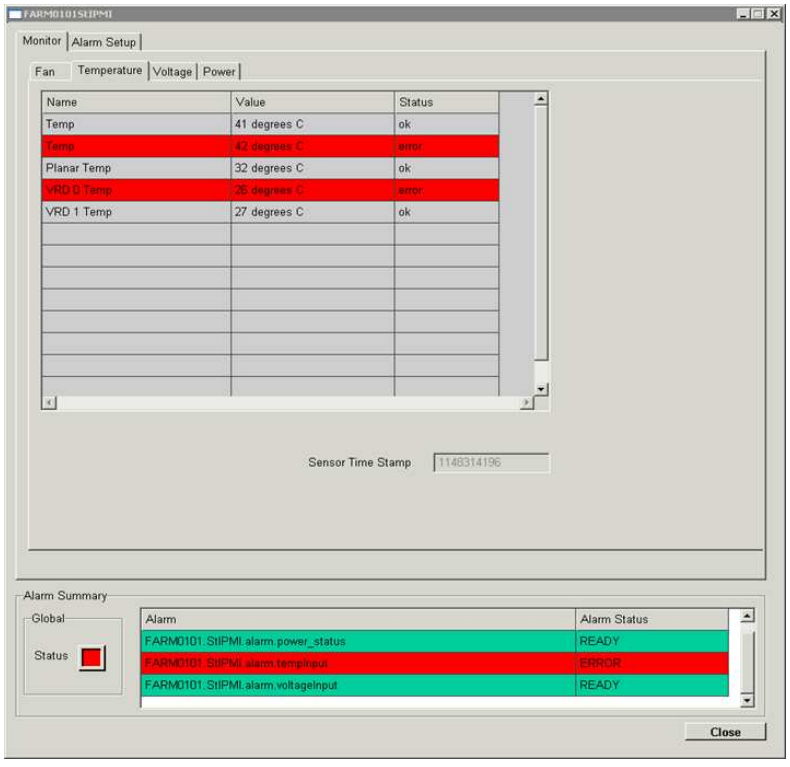

Figure 6 The Power Manager PVSS Client: the Temperature Sensor Tab.

7 Component versions

The current documentation refers to the Power Manager software contained in **FMC** version **3.7.0**. In particular, the versions of the Power Manager components, which can be retrieved with the Linux **`ident`** command applied to the executables, are shown in Table 1.

Table 1 Versions of the Power Manager components in FMC v. 3.7.0.

Executable component	Source(s)	Version
`ipmiSrv`	**`ipmiSrv.c`**	2.20
	`ipmiUtils.c`	1.31
`pwSwitch`	**`pwSwitch.C`**	1.3
`pwStatus`	**`pwStatus.C`**	1.5
`ipmiViewer`	**`ipmiViewer.C`**	1.10

8 Conclusion

We have realized a software tool — the Power Manager — able to switch on and off the nodes of the LHCb Event Filter Farm and to monitor their physical conditions: power status (on/off), temperatures, fan speeds and voltages, independently on the operating system actually running on the machines. It is composed of a server process running on a Control PC, and a client process to be used for sending commands to the server and retrieve the status information. The Power Manager can operate on farm nodes whose motherboards and network interface cards implement the IPMI specifications, version 1.5 or subsequent. It is also interfaced with the PVSS framework and fully integrated in the LHCb Experimental Control System.

9 References

[1] F. Bonifazi, D. Bortolotti, A. Carbone, D. Galli, D. Gregori, U. Marconi, G. Peco, and V. M. Vagnoni, "The Task Manager for the LHCb on-line farm," CERN, LHCb note 2004-099 DAQ, Nov. 24, 2004. [Online]. Available: http://doc.cern.ch/archive/electronic/cern/others/LHB/public/lhcb-2004-099.pdf

[2] Intelligent Platform Management Interface. [Online]. Available: http://www.intel.com/design/servers/ipmi/

[3] Intelligent Platform Management Interface Industry Promoters, Adopters and Contributors. [Online]. Available: http://www.intel.com/design/servers/ipmi/adopterlist.htm

[4] Intelligent Platform Management Interface Specification Second Generation v2.0. [Online]. Available: ftp://download.intel.com/design/servers/ipmi/IPMIv2_0rev1_0.pdf

[5] IPMI Intelligent Platform Management Bus Communication Protocol Specification v1.0. [Online]. Available: ftp://download.intel.com/design/servers/ipmi/ipmb1010ltd.pdf

[6] IPMI Intelligent Chassis Management Bus Bridge Specification v1.0. [Online]. Available: ftp://download.intel.com/design/servers/ipmi/ICMB_1013.pdf

[7] OpenIPMI. [Online]. Available: http://openipmi.sourceforge.net/

[8] R. Droms. RFC 2131 – Dynamic Host Configuration Protocol. [Online]. Available: http://tools.ietf.org/html/rfc2131

[9] IPMItool. [Online]. Available: http://ipmitool.sourceforge.net/index.html

[10] IPMI Management Utilities. [Online]. Available: http://ipmiutil.sourceforge.net/home.htm

[11] GNU FreeIPMI. [Online]. Available: http://www.gnu.org/software/freeipmi/

[12] C. Gaspar and M. Dönszelmann, "DIM – A Distributed Information Management System for the DELPHI experiment at CERN," in *Proceedings of the 8th Conference on Computer Applications in Nuclear, Particle and Plasma Physics*. Vancouver, BC, Canada: IEEE, Jun 1993, pp. 156–158. [Online]. Available: http://dim.web.cern.ch/dim/papers/DIM.PS

[13] pthread_cond_wait(3) – Linux man page. [Online]. Available: http://www.die.net/doc/linux/man/man3/pthread_cond_wait.3.html

[14] signal(7) – Linux man page. [Online]. Available: http://www.die.net/doc/linux/man/man7/signal.7.html

[15] longjmp(3) – Linux man page. [Online]. Available: http://www.die.net/doc/linux/man/man3/longjmp.3.html

[16] popen(3) – Linux man page. [Online]. Available: http://www.die.net/doc/linux/man/man3/popen.3.html

[17] Advanced Configuration and Power Interface Specification. [Online]. Available: http://www.acpi.info/DOWNLOADS/ACPIspec30b.pdf

[18] PVSS-DIM Integration. [Online]. Available: http://lhcb-online.web.cern.ch/lhcb-online/ecs/fw/FwDim.html

[19] glob(7) – Linux man page. [Online]. Available: http://www.die.net/doc/linux/man/man7/glob.7.html