Using SNMP for creating Distributed Diagnostic Tools

O. Reisacher*, P. Ribeiro, H. P. Christiansen**

SL Controls Group

CERN, Geneva, Switzerland

*Left CERN in Feb. 1995

**Currently at UID, Linkoping, Sweden

Abstract

In this paper we describe how SNMP (Simple Network Management Protocol) can be extended to do control system diagnostics. Our solution consists of a SNMP agent for LynxOS and a configurable MIB (Management Information Base) browser. We have reused diagnostic modules from the existing diagnostic system and integrated our development into a commercial network management product.

Introduction

The control system for the LEP and SPS accelerators at CERN is a distributed computer system of over 300 nodes. This system must be operational during long periods of the year on a 24 hours a day basis. A number of diagnostic tools have been developed over the years to support the intervention of the on-call team, and they were also integrated into the Control Room operations environment. For the management of the large Local Area Network covering over 200 sq. km, a commercial product, HP OpenView is used. The project aimed at integrating these different tools into a common framework was started in 1993.

This paper is divided into five parts. We start with the description of the existing distributed diagnostic architecture and the motivations for this project. A brief description of the current network management standards follows. The description of an SNMP agent for LynxOS along with its enterprise MIB is followed by the presentation of a configurable MIB browser. The last part covers the integration of this framework into the HP OpenView environment.

1. Existing architecture and motivations for this project

The LEP and SPS control system is a distributed system that can be divided into three interconnected logical layers :

1. Back-Ends

At this level run the applications for the daily operation of the accelerators. These include not only all the application software related to normal machine operation, but also all the work related to anticipating control system failures and bringing the system back to a normal state after a breakdown. At this level, for system management, we have a home-made tool called xcluc that browses an information repository where all the periodic reports from the agents arrive. Xcluc presents the state of the different network nodes, enables access to the report information stored in the repository and also provides a number of primitives to interact with the different nodes. These systems are mainly X-Terminals running on HP 9000-7xx servers.

2. Front-Ends

The front-ends run a number of applications controlling the activity of a certain number of entities described in 3. These entities also act as proxies for the message-oriented acquisition and control scheme. A home-made diagnostic tool, clic (from the sound made when one takes a photograph) runs periodically acquiring a certain number of states pertaining to important sub-systems in this environment. These systems are PC or VME boards running the LynxOS operating system.

3. Equipment Control Assemblies (ECA)

These systems are doing I/O multiplexing acquisition and control. They are connected to the upper layer (the front-ends) by a message oriented protocol on top of a field bus. The status of the different ECA stations on the bus is periodically controlled by the clic application mentioned on 2.

The main motivation for this project was to develop an integrated system and network tool for operations and diagnostics. It was desired to reuse existing code from the clic application to but standardize the system information model and transport protocol aspects. As HP OpenView (OV) is used for the network management aspects, the basic idea was to transform the existing tools so that they could be easily interfaced to this package. Accordingly we took a number of decisions :

- use SNMPv1 with MIB II, supported by OV

- use the HP SNMP library

- develop a number of applications to support the integration

2. Standards

A number of Network Management standards have been developed over the past few years. These standards have a scope that goes beyond the area explored by this paper, system management. All the different architectures and standards for designing network management share the basic model that splits the Managing System and the Managed System into two separate sub-systems. Each of these sub-systems communicates with a Manager Kernel or an Agent and these communicate with each other in a client-server relationship via a Management Protocol. The applications, their environment and the system they run on are called a managing system. The network equipment and software is represented as a collection of managed objects. Those managed objects are supported by an agent. The managed objects, their agent and the computer system they run on are called the managed system. This management framework provides application programming interfaces (APIs) to the applications and objects. Of the three main management families, SNMP, CMIP (OSI) and DME (OSF), we have chosen SNMP to use as a framework for this project.

The management information model (SMI) is the most important part of a network management structure, as it defines the kinds of interactions between management applications and managed objects

supported by the model. The managed objects are often active entities, their behavior being independent of any management system.

The SNMP family

The following refers to what is termed as SNMP version 1 as described in [RFC 1155, 1157, 1212]. There are three distinct sets in this family of standards :

- Management Information Base (MIB)

This specifies the variables maintained by the framework (this is the information that can be queried and set by the manager (MIB II, [RFC 1213])). This is in practice a text file describing the variables supported by a SNMP agent. For the sake of simplicity, a limited number of syntactical elements of the base syntax ASN.1 (Abstract Syntax Notation 1), are used. To avoid variations on the possible syntactical constructs defining the same object,a set of macros in ASN.1 notation is used [RFC 1212].

- Structure of Management Information (SMI)

Athis is a set of common structures and an identification scheme used to reference the variables in the MIB, described in [RFC 1155].

- The Management Protocol

This is a protocol layer on top of the User Datagram Protocol (UDP), implementing five basic types of transaction

SNMP GET

SNMP GETNEXT

SNMP SET

SNMP RESPONSE

SNMP TRAP

The philosophy behind SNMP is that the agent and the object implementation should be simple. In this context, the SNMP SMI only supports a limited set of interactions between the application and the object. Operations to Get and Set attributes (called MIB variables) are supported. Attributes can be gathered into tables and groups, but operations on tables or groups are not directly supported. Create, Delete and various Actions are all implemented as a side effect of setting a MIB variable. An individual MIB variable is named within an agent by an object identifier. Object identifiers are a sequence of integers, representing a path through a tree. Individual MIB variables in a system are represented by a leaf on this tree. The nodes above the leaf level represent tables, groups or MIB variable types. The GetNext operation is the only means to make an inventory, similar to a directory tree listing, of all the MIB variables supported by an agent.

3. SNMP agent for LynxOS and its enterprise MIB

The SNMP agents available for commercial systems are often closed to modification. Any extension capability that is available is often of limited scope. The first version of the SNMP agent for LynxOS was based on the Carnegie-Mellon University (CMU) source code.

Along with the extensible SNMP agent prototype, compliant to the MIB II specification, a ASN.1 parser has been developed. Each extension, new object, is described as a MIB object with the ASN.1 syntax and entered in the database. This description is run through the parser which generates the hooks to the C code to be called whenever an access to the new object is requested. The new code should be linked with the other SNMP agent modules to produce the extended agent (see figure 2). Using this method, the SNMP agent was first made to support the network management MIB. The private part of the MIB was then integrated in a stepwise fashion.

Figure 1 shows the main building blocks of the described SNMP agent. The system surveillance is performed every trap_interval seconds. If an error condition is detected a TRAP is sent to the manager. Trap_interval is a MIB object that can be set. Another agent module handles the SNMP requests originated by the manager. The extensions to the MIB describe the system variables that should be monitored and also the variables specifying the limits against which the system variables are tested. These variables are specific to the system to be managed.

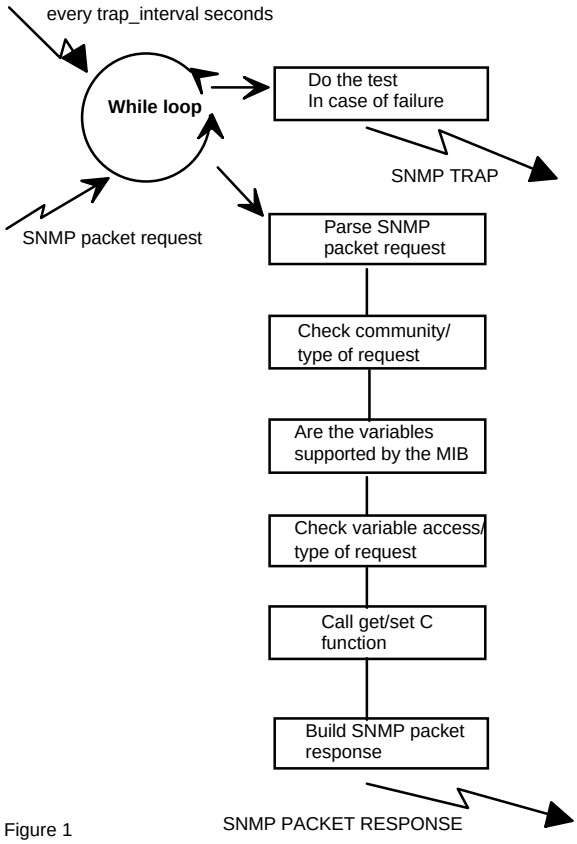

Figure 1

The instrumentation of operating system parameters is not a trivial task. One of the main concerns of this project was to provide this information using a minimum of system resources. As our LynxOS systems

run diskless, we had to implement and incorporate into the SNMP agent functionality similar to the one provided by system applications like **df**, **ps** and **netstat**. The agent is not **forking**, doing dynamic memory allocation or interacting with files once it is up and running. The reason is that this is the only way to be efficient when the node is in a bad state and needs diagnostic actions.

The following is an example of adding an object to the private MIB subtree. This example code is written for LynxOS, where many system parameters can be obtained using the **info** system call.

The ASN.1 definition for the MIB :

```
max_inode OBJECT-TYPE
        SYNTAX INTEGER
        ACCESS read-only
        DESCRIPTION
                "Read-only integer value corresponding to the system file table size"
::= { filesystem 1 }
```

The C function that implements the described object :

```
#include "snmp_impl.h"
#include "pdu.h" /* VarBind */
#include "error.h" /* ERROR_MALLOC, NO_ERROR */
int get_file_table_size ( var )
VarBind *var;
{
        var->value.integer = ( long *) malloc (sizeof (long));
        if (var->value.integer == NULL) return (ERROR_MALLOC);
        * (var->value.integer) = info (I_NFILES);
        var->value.len = 1;
        return (NO_ERROR);
}
```

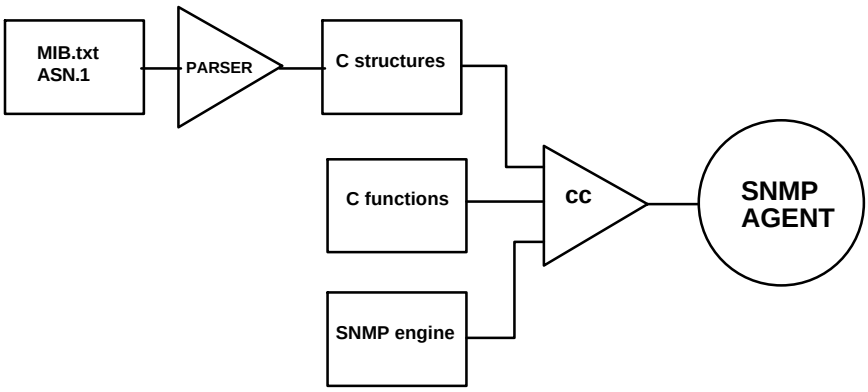

Figure 2

4. The development of a configurable MIB browser

The development of this product was decided at a late stage of the project. Apparently in contradiction with the project aim of integrating applications at the top level, this product was justified by the need for a simplified and user configurable view of a limited number of variables and tables.

The OpenView browser has many very powerful capabilities, such as the possibility to dynamically load/unload an MIB file. It also has a menu-based application builder, but it suffers from a number of drawbacks such as being too close to SNMP theory, producing an almost unreadable output of table variables and producing a complex set of windows,where the destination entity for the manager queries is not easily known all the time.

The new browser was developed using the Hewlett-Packard SNMP library and a Motif user interface.

Following the requirement specifications, the browser is capable of reproducing a session by saving the previous session layout in a configuration file, on a one per user basis. This enables a user to create a reproducible session profile by adding to the current session the objects pertaining to the required status view.

A clear distinction is made between columns in a table object and plain objects so that the presentation follows the object type.

All the SNMP related information is kept away from the user interface.

The browser can be configured to run in a single window mode.

The type of the data displayed can be changed with a “translator” function.

5. The integration of this framework into HP OpenView environment

The HP OpenView system allows the creation of submaps and different views. This has been exploited in this project. In addition to the view that IP Map offers, a secondary view, close to the existing repository browsing tool **xcluc,** has been created.

The usage of the IP Map application can continue as before. This view will reflect the combined network and system states of the nodes as reported by the extended SNMP agents.

With the API provided with HP OpenView, a simple interface to display parts of the extended MIB can be quickly built. The experience acquired by doing this will make extensions to similar interfaces concerning other objects in the MIB an even easier job.

The API also provides interfaces to the HP OpenView database to get hold of the received traps and also OSF/Motif functions that allow several applications to work with, and control, the visual aspects of a common submap hierarchy, thus cooperating with the IP Map application.

Conclusions

This project proved the feasibility of replacing the home-made distributed diagnostic tools by a SNMP-based framework. Among other benefits, we will be able to integrate the view of network and system diagnostics, use a standard protocol for the manager agent communications, provide a standard hierarchy for the managed data and reduce the maintenance effort to one agent.

We found some limitations and these included limited support at the data types level, very slow access time under SNMPv1 for table object retrieval, and that further studies were needed to be done on security and reliability issues.

References

Managing networks and systems with SNMP. O. Reisacher 1995, http://www.cern.ch/CERN/Divisions/SL/SpsLepControls/snmp_sysman.html

Installation and Management of the SPS and LEP control system computers, Alastair Bland, CERN, paper presented at ICALEPCS’93, Berlin, Germany

SNMP, SNMPv2 and CMIP, William Stallings, Addison Wesley 1993