N92-16571

USING A CLIPS EXPERT SYSTEM TO AUTOMATICALLY MANAGE TCP/IP NETWORKS AND THEIR COMPONENTS

Ben M. Faul

TRW Systems Engineering & Development Division
Carson, California

Abstract. This paper describes an expert system that can directly manage networks components on a TCP/IP network. Previous expert systems for managing networks have focused on managing network faults after they occur. However this proactive expert system can monitor and control network components in near real time. The ability to manage directly network elements from CLIPS is accomplished by the integration of the Simple Network Management Protocol (SNMP) and an Abstract Syntax Notation (ASN) parser into the CLIPS artificial intelligence language.

INTRODUCTION

Networking is one of the fastest growing segments of the computer market. Networks can be as simple as several PCs on a LAN to a corporate-wide area network composed of hundreds of machines to a global network comprised of hundreds of local area networks and hundreds of thousands of machines.

The emergence of network-based applications and even operating systems demands the network components operate as efficiently and effectively as possible.

Managing a network can be an arduous task, as there are numerous components that comprise the network, originationg from many vendors. In addition, the components are usually dispersed over a large geographic area. But, even if the network components were co-located, most of the network devices don't even have an operator's console.

All of these factors add up to a nightmare when things go wrong in the network. Traditional system operation concepts deal with problems as they arise. In a network, just locating a downed component can be a major task. All the while, applications and users are idle while technicians scour the campus looking for the problem. Several CLIPS applications have been described previously that aid in the isolation and diagnosis of problems [Leigh A.] by using a question and answer session with a human.

The next logical step in managing a network, is to look for, and solve, problems under expert system control. This is a natural application for an expert system like CLIPS. However, to utilize CLIPS as a solution, the expert system shell must incorporate several new features it does not currently have.

NEW CLIPS FEATURES TO FACILITATE NETWORK MANAGEMENT

The US Government and commercial vendors recognized the need for developing a vendor-independent mechanism for managing network components, largely because of the network management chaos that erupted <u>after</u> networking became so prevalent.

Typical automated network management systems rely on a specific vendor's diagnostic hardware. One has even been written in CLIPS. [Hansen & Flores]. However, vendor-dependent network management solutions have only limited application in a network comprised of elements from different vendors.

To answer this need for vendor-independent network management the US Government's Network Working Group developed the Simple Network Management Protocol or, SNMP [Case, Fedor, Schoffstall, & Davin].

By integrating SNMP into the CLIPS language, expert systems can then be built that can take direct control of network elements; thus obviating the need for most human interaction.

SNMP Architectural Model

Implied in the SNMP architecture is a collection of network management stations and network elements. The network management station executes the applications that monitor and control network elements. Network elements are devices on the network such as hosts, routers, gateways, terminal concentrators, PCs, etc. that communicate on the network. The SNMP is used to communicate the management information to the network elements. The CLIPS program will be the manager for the network. The various hardware components on the network will be the network elements that CLIPS will manage.

The first goal of the SNMP integrated into CLIPS is to explicitly minimize the number and complexity of functions used by the manager program. This will result in the reduction of new language constructs in CLIPS; hence maintaining the portability and integrity of the language.

Another goal of the SNMP/CLIPS integration is to provide a paradigm for monitor and control that can accommodate unanticipated aspects of network management. As time and network products progress, the CLIPS manager will be extendible at the expert system shell level. This will tend to eliminate further extensions to the language itself.

The third and most important goal is that the resultant system will be independent of the architecture and mechanisms of the particular network elements. Achievement of this goal allows the CLIPS network manager to control network elements from any vendor.

Representation of Management Information

The information communicated using SNMP is represented using the ASN.1 language [ISO Standard 8824]. Use of the ASN.1 language internal to SNMP is key to its machine independence and eventual conformance with GOSIP mandates.

The information communicated using ASN.1 is called the management information base (MIB). There is a standard MIB, that all the conforming network products recognize.

An example ASN.1 variable in the MIB is represented in Table 1.

```
        sysUpTime OBJECT-TYPE
            SYNTAX TimeTicks
            ACCESS read-only
            STATUS mandatory
            ::= { system 2 }
```

Table 1. An ASN.1 Definition

The example in Table 1 will be referenced in the following paragraphs to illustrate CLIPS network management via the SNMP.

Protocol Operations

The SNMP functions integrated into CLIPS operate as inspections or modifications of variables that correspond to entries in the MIB. The manager specifies the MIB variable to view or alter, and the managee (also called "agent") does the appropriate get variable or set variable action. Notice in the above example that the variable is <u>read-only</u>. Thus the manager may view, but may not modify this variable.

Using the MIB, the variables become accessible in a machine-independent form. The CLIPS manager does not care what the network element's internal representation is of the variable or how it is derived and maintained.

Primarily, the CLIPS manager works by polling the network element agents for the appropriate information.

There are no imperative commands in the protocol. The manager merely sets a MIB variable to some value. The network element agent then decides what to do with the value. The example given in Request for Comments (RFC) 1157 [Case, Fedor, Schoffstall, & Davin] is that of a "reboot command". Rather than explicitly implementing a REBOOT command, this action might be invoked by simply setting a parameter indicating the number of seconds until the system reboots.

Identification of Object Instances

The variables (or names) of all object types in the MIB are defined explicitly in the Internet-standard MIB [Rose M.], known as the MIB-II of RFC 1158. The entries in this standardized MIB make it possible for CLIPS to manage TCP/IP network elements in a vendor-independent fashion. Referencing RFC 1158, the CLIPS developer can access any of the defined variables on any network element that supports the SNMP.

Each instance of any object type defined in the MIB is identified with a variable name. The MIB is organized in a hierarchical fashion, thus making it easy to "walk the MIB" to obtain aggregate information. An example of walking a portion of the MIB is to obtain all the information under the variable name "system".

In SNMP the objects are identified with fully qualified variable names in "x.y" format, where "x" is the name of a non-aggregate object defined in the MIB and "y" is the object identifier that is specific to the desired instance. This naming strategy admits exploitation of contiguous lexicographic retrieval of related variables, which makes MIB walking possible.

For example, the fully qualified ASN.1 name that represents how long a network element has been up and running, "upTime", is:

```
iso.org.dod.internet.mgmt.mib.system.sysUpTime.0
 1   3   6     1       2    1     1       2     0
```

The numbers underneath the definition show the integer representation that defines the variable. The CLIPS programmer references variables by the text ASN.1 name, the ASN.1 parser converts the name into the array of integers that correspond to the name for actual transmission over the SNMP. Notice from Table 1 the last statement was the assignment ::= {system 2}. Working backwards one can see that "mib ::= { mgmt 1 }", "mgmt ::= {internet 2}", and so on.

It is possible to obtain the system up time from a remote network element by asking for this variable. Alternatively, all system variables could be retrieved from the network element by requesting "iso.org.dod.internet.mgmt.mib.system".

As a short cut, all the variables are presumed to be preceded with "iso.org.dod.internet.mgmt.mib". Thus a request for system up time merely becomes "system.sysUpTime.0".

There are about one hundred variables defined in the MIB. For purposes of example in this paper, and to eliminate confusion, only a portion of the internet MIB and variables are presented. The MIB variables used in the code fragments presented in this paper are defined in Table 2.

```
system      OBJECT IDENTIFIER ::= { mib 1  }
ip          OBJECT IDENTIFIER ::= { mib 4  }

sysDescr OBJECT-TYPE          -- Describes the system
      SYNTAX       octet-string
      ACCESS       read-only
      STATUS       mandatory
      ::= { system 1 }

sysContact OBJECT-TYPE        -- Who to contact if problem
      SYNTAX       octet-string
      ACCESS       read-write
      STATUS       mandatory
      ::= { system 4 }

sysLocation OBJECT-TYPE       -- Where is this hardware
      SYNTAX       octet-string
      ACCESS       read-write
      STATUS       mandatory
      ::= { system 6 }

ipInDiscards OBJECT-TYPE      -- How many packets discarded
      SYNTAX       counter       -- because of internal errors
      ACCESS       read-only     -- in the system.
      STATUS       mandatory
      := { ip 8 }
```

Table 2. Example MIB Definition

The variables defined implement self-explanatory functions, with the possible exception of ip.ipInDiscards.0. This variable is a counter in the network element that tallies network packets that were destroyed because the network element couldn't process them. Usually this means that there wasn't enough buffer space to process the message. Obviously, if the number of discards goes up, the more problematic the operation of the network will become. It is this variable, ip.ipInDiscards, that will be examined in the further examples.

INTEGRATION OF SNMP WITH CLIPS

The SNMP protocol and an ASN.1 parser have to be built in order to access the MIB in the remote network entities. Fortunately, neither the SNMP protocol nor the ASN.1 parser are particularly hard to come by in the "C" language. There are two public sources of SNMP, one is from Carnegie Mellon University, and the other is from the Massachusetts Institute of Technology. Both are distributed without charge, if you follow their liberal licensing agreement.

The Carnegie Mellon University SNMP was chosen because of the author's familiarity with other CMU efforts and was thus comfortable with their code.

The SNMP code from CMU implements an ASCII database of MIB variables for SNMP, an ASN.1 parser, the SNMP protocol over TCP/IP, and a set of applications programs that allow one to access the MIB variables on the network elements. The SNMP system is designed to run under the UNIX operating system. For the management station, the CLIPS system was built on an Everex 80386 system running the SCO Open Desktop (UNIX System V.32) operating system.

The first step in the process was to build the application programs and use them to access the MIB variables of network elements.

There were two germane applications: "snmpget" and "snmpwalk". The snmpget allows one to get a variable from the network element. The syntax is "snmpget <host-name> <access-control> <asn.1 name>. The access control parameter is the "password" that allows one to access the variables. For read-only purposes "public" will do. Table 3 shows what the snmpget command will return if issued against a network element named "gandalf".

```
$snmpget gandalf public system.sysDescr.0
IBM RS6000, version 1.1 AIX operating system
$
```

Table 3. The SNMPGET Command

However, there was no "snmpset" provided so one was coded using the CMU snmplib.a application library. Then it was possible to both view and modify the variables (if allowed). In Table 3, the system contact name on the remote element will be viewed, then changed.

```
$snmpget gandalf public system.sysContact.0
Edward Williams, 213-555-1212
$
$snmpset gandalf private system.sysContact.0 "Ben Faul 213-555-1212"
$
```

Table 4. View Then Set a Variable

From now on, any one requesting the system contact name of element gandalf will get the new contact name "Ben Faul".

Once familiar with the application programs and the SNMP library, integration with the CLIPS expert system was an easy process. The three application programs were then modified to be incorporated into CLIPS. The UNIX syntax was preserved in the language to allow the application programs' documentation to be used with CLIPS.

Using standard CLIPS implementation methodologies the language was extended to include the following new constructs: (snmp-get), (snmp-getnext), (snmpset).

Surprisingly, these are the only new constructs required in the language to facilitate SNMP access.

Get An Instance Variable

To get the value of a variable on the network element, the (snmp-get) command is used. The form of the command is:

```
(snmp-get <host> <variable-name> <access>)
```

The <host> parameter defines the symbolic name of the element being queried. The <variable-name> is the ASN.1 name of the variable. The <access> parameter is used for authenticating privilege to view or modify the variable. Usually for this command the <access> variable is set to "public".

The snmp-get returns a multi-field variable, The first field is the return code. The second field is the type (integer or string), which is used to field the returned variable in the third field (if integer) or fourth field (if string). If an error occurs a -1 is returned in the type field, with an error string contained in the fourth field.

Get The Next Instance Variable

For getting contiguous variables in the MIB, the snmp-get-next command is provided. The form of the command is:

```
(snmp-get-next <host> <variable-name> <access>).
```

The <host> parameter defines the symbolic name of the element being queried. The <variable-name> is the ASN.1 name of the variable fragment. The <access> parameter is used for authenticating privilege to view or modify the variable. Usually for this command the <access> variable is set to "public".

The snmp-get-next returns a multi-field variable, The first field is the return code. The second field is the type (integer or string), which is used to field the returned variable in the third field (if integer) or fourth field (if string). The fifth field is a string that identifies the fully qualified name being returned. If an error occurs a -1 is returned in the type field, with an error string contained in the fourth field.

Each successive call to the snmp-get-next returns the next variable that was contiguous with the previous variable, until the MIB variables in that fragment is exhausted. In this manner one may examine the MIB by "walking down the branches". Indeed, the entire MIB may be returned by successive calls using the ASN.1 variable: "iso.org.dod.internet.mgmt.mib"

Set An Instance Variable

To set a variable in a network element, the (snmp-set) command is used, assuming that one has authority to do so. The form of the command is:

```
(snmp-set <host> <variable-name> <access> <new-value>).
```

The <host> parameter defines the symbolic name of the element being queried. The

<variable-name> is the ASN.1 name of the variable. The <access> parameter is used for authenticating privilege to view or modify the variable. Usually for this command the <access> variable is set to "private".

The snmp-set returns a multi-field variable, The first field is the return code. If the code is 1 then the replacement was successful. Otherwise, 0 indicates an error occurred with the third field containing an applicable error string.

The architecture of the SNMP/CLIPS integration, and how it is applied to manage a sample network is described in Figure 1.

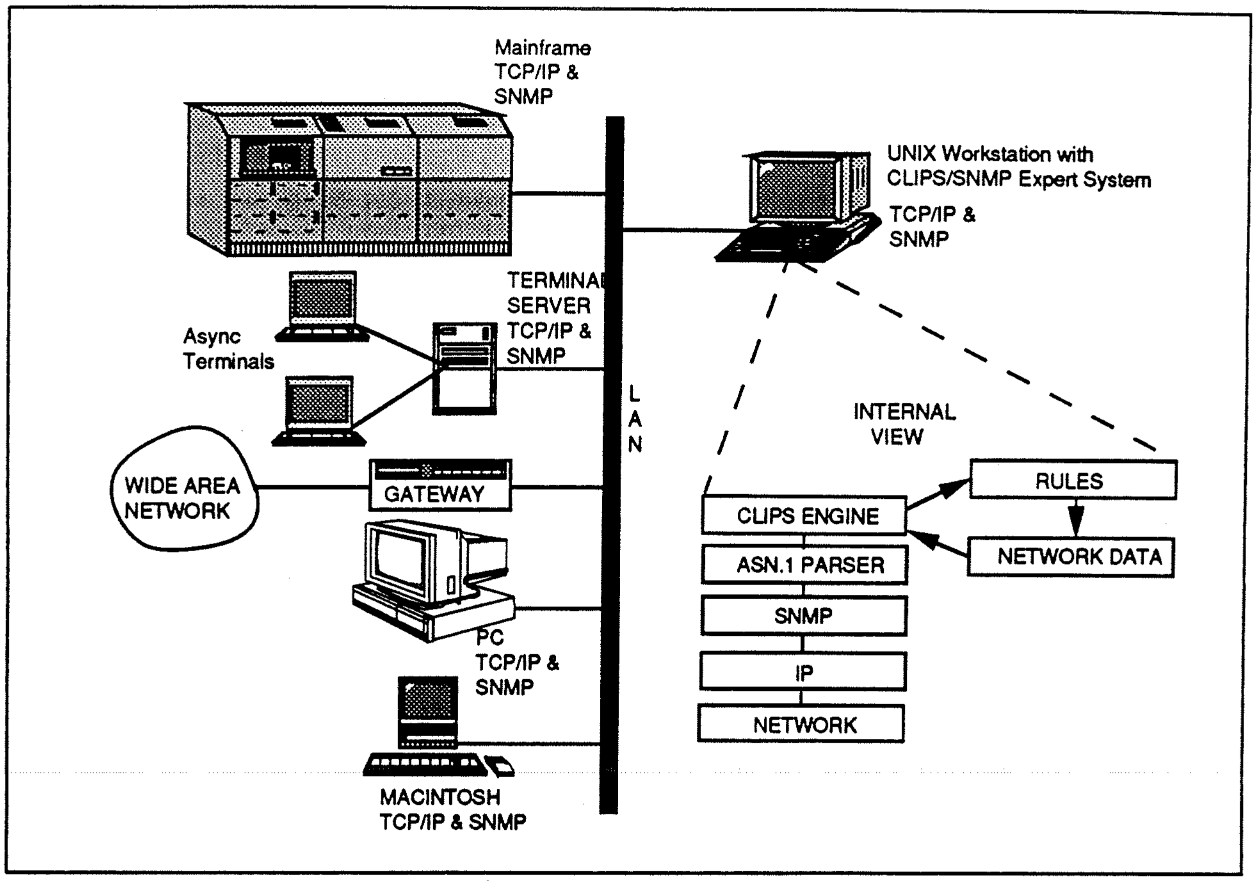

Figure 1, SNMP/CLIPS Integrated Architecture

The network elements, such as the gateway, terminal concentrator, and host interface are all controlled, automatically from the host running the CLIPS expert system. The expert system gathers SNMP data from the network elements and stores it in the data file. The data file is used as feedback to the expert system to change the network element parameters based on trend analysis and knowledge contained within the expert system.

EXAMPLE CLIPS NETWORK MANAGEMENT SYSTEM

The previous sections dealt with building the infrastructure in CLIPS to enable it to do the work of network management. The concepts of proactive network management will be discussed in this section and displayed in the code fragment in Table 5, below.

Network Initialization

The network manager expert system must obtain the element names of the various network entities to be managed on the network. These are simply stored in an ASCII database from which the expert system reads upon initialization.

A more flexible approach would be to utilize a database for the network elements, but that would complicate the design for example purposes.

Network Status Information

The network manager expert system polls the network elements for various performance parameters. The information returned is stored in an ASCII flat file for future reference.

Problem Detection

Problem detection is accomplished by applying rules against the ASCII flat file that holds network status information. Upon finding a potential problem, the expert system asserts the facts determined by the current and historical status as determined from analyzing the ASCII file.

Problem Correction

The network manager expert system's problem correction rules attempt to solve the problem by class of failure. In the case of overloaded gateways, routing tables may be adjusted to alleviate traffic in this element. In the case of an interface card reporting many transmission errors, a diagnostic printout showing the location of the unit and the type of error condition may be printed.

ILLUSTRATIVE EXAMPLE

To see how well SNMP and CLIPS work together, consider the CLIPS code fragment presented in Table 5.

```
        (bind $?discard-data
              (snmp-get ?element "public" "ip.ipInDiscards.0")
        (bind ?rc (nth 1 $?discard-data))
;
; If no network error then process, else note the error
;
        (if (ne 1 ?rc)
        then
              (printout t "Error getting data from " ?element)
              (printout t "     Error = " (nth 4 $?discard-data))
              (assert network-error ?element))
        else
;
; Determine if packet discard threshold has been exceeded
;
              (if (> (nth 3 $?discard-data) ?max-allowed)
              then
                    (bind $?contact-data
                          (snmp-get ?element "public" "system.sysContact.0"))
                    (bind $?location-data
                          (snmp-get ?element "public" "system.sysLocation.0"))
                    (printout t "Packet discards exceeded for " ?element)
                    (printout t "   Likely problem is memory exceeded")
                    (printout t "      Check unit at location "
                          (nth 4 $?location-data))
                    (printout t "      Notify "
                          (nth 4 $?contact-data))
                    (printout t "   Re-route in progress)
                    (assert reroute-to ?element)))
```

Table 5. CLIPS/SNMP Sample Code

As can be seen in Table 5, the addition of the SNMP capability to CLIPS does not greatly influence the character and flavor of the language. However, the addition of the SNMP access allows the full power of the language to be used to influence the performance of a network of SNMP compliant commercial products.

CONCLUSIONS

The application of CLIPS to SNMP has proved to be quite successful. The network manager expert system is capable of detecting faults and does a credible job of proactive management.

To enhance the proactive management capabilities, the expert system should utilize an SQL database to store the polled data. This would permit more extensive trend analysis to be done.

The SNMP library from CMU utilizes a synchronous network connection. This means that the (snmp-get) hangs until the message is returned to CLIPS or the read times out. To manage a very large number of elements efficiently within CLIPS, non-synchronous network I/O will be required.

The user interface of the network manager is a simple X Window System/MOTIF GUI that uses very simple icons to represent the different entities. A full featured X Window System interface that is truly object-oriented is a must for building a serious network management product.

REFERENCES

Case, J., Fedor, M., Schoffstall, M., Davin M. (1990) A Simple Network Management Protocol, *DARPA RFC 1157.*

Hansen, R.F., Flores, L.M., (1990) JESNET Expert Assistant, *First CLIPS Conference Proceedings, Houston, pp.140-146.*

ISO Standard 8824 (1987), "Specification of Abstract Syntax Notation One (ASN.1), *International Organization for Standardization*

Leigh, A.B. (1990) The Network Management Expert System Prototype for Sun Workstations. *First CLIPS Conference Proceedings, Houston, pp. 148-154.*

Rose, M. (ed) (1990) Management Information Base for Network Management of TCP/IP based Internets: MIB-II, *DARPA RFC 1158*