IP 망 관리를 위한 SNMP 기반 EMS 개발

°민경주, 강석민, 김태석, 조익현, 권택근
충남대학교 컴퓨터공학과

An Implement of SNMP –based EMS for IP Network Management

Kyoung-Ju Min°, Suk-Min Kang, Tae-Suk Kim, Ic-Hyun Cho, Taeck-Geun Kwon
Dept. of Computer Engineering, Chungnam National Univ.
{kjmin, smkang, tskim, ihcho, tgkwon}@ce.cnu.ac.kr

요 약

본 논문에서는 Linux 상에서 SNMP (Simple Network Management Protocol) 에이전트(Agent)를 구현하고, 기존의 복잡한 명령어 체계를 향상하기 위해 shell을 통한 CLI (Command Line Interface)와 Java를 통한 EMS (Element Management Station) 시스템을 설계하고 구현하였다. 연구 개발된 에이전트는 매니저의 요청에 응답을 하고, 시스템 상의 오류와 같은 상황이 발생했을 때, 매니저에게 TRAP을 발생시키는 기능을 갖는다. 본 논문에서는 개발된 SNMP 에이전트의 개발 환경, 개발 내용과 실험내용에 대해 언급하였다.

1. 서 론

인터넷의 급속한 확산으로 인해 IP망의 이용이 점차로 증가 되어가고 있다. 이에 반해 IP 망의 관리는 기존의 핑(ping)과 같은 프로그램을 이용해 망에 있는 다른 호스트에게 간단한 형태의 메시지를 보내고 응답 받음으로써, 상대 호스트가 네트워크에서 살아 있다는 것을 아는 수준의 관리가 대부분이었다. 또한 기존의 IP 망 및 장비의 관리는 시스템에 문제가 발생했을 때 인력이 직접 장비를 점검하는 수준을 면하지 못하고 있고, 인건비를 줄이고 보다 효율적인 망 관리의 필요성이 부각되는데, 망 관리자의 요청에 적절한 행동을 하고, 문제 발생시 망 관리자에게 보고하는 SNMP 에이전트를 두어 망 관리를 효율적으로 할 필요가 대두되었다. 이때 SNMP 에이전트를 사용하기 위해서는 MIB (Management Information Base)를 매니저와 에이전트 양측에 두어, 망 관리 대상을 체계적으로 관리할 필요가 있다.

이러한 이유로 본 연구에서는 SNMP 에이전트를 구현하였는데, 사용자 인터페이스를 위해 platform에 독립적인 Java를 이용해 EMS (Element Management Station) 시스템을 개발하였고, SNMP에 적합한 Shell을 만들고, CLI (Command Line Interface)를 통해 콘솔(console)상에서도 간단한 명령어 체계를 이용해 관리 가능하도록 했다.

본 논문의 구성은 다음과 같다. 2장에서는 SNMP의 기본 용어 및 구조와 동작을 소개한다. 3장에서는 CNU-SNMP (Chungnam National Univ. – SNMP)구현에 관한 상세한 사항을 소개한다. 결론 및 향후 연구과제는 4장에서 기술한다.

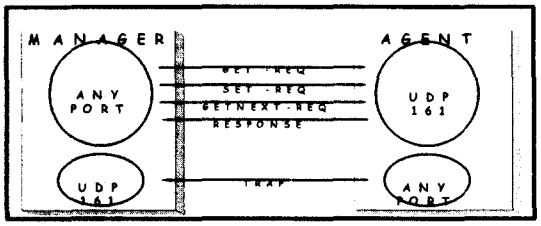
2. SNMP의 구조와 동작

SNMP는 망을 관리하는 관리자가 위치하는 매니저의 영역인 관리국(Management Station), 매니저의 요청에 대해 적절한 행동을 취하고, 매니저에게 응답을 하는 피관리 에이전트(Managed Agent), 망 관리 대상에 대한 개념적 구조체인 MIB, 매니저와 에이전트 사이의 통신을 위한 프로토콜인 NMS 네 개의 요소를 가지고 동작을 한다.

2.1 SNMP 관리 모델

SNMP의 주된 동작은 GET-REQUEST, SET-REQUEST, TRAP의 세 가지로 크게 분류될 수 있다.

매니저는 에이전트에게 관리에 필요한 대상에 대한 요청을 하게 되고, 이에 에이전트는 모듈에 정의된 절차를 걸친 결과를 매니저에게 응답하게 된다. 이때 필요한 행동이 GET-REQUEST이고, 에이전트는 이 REQUEST에 대해 RESPONSE를 보내게 된다. SET-REQUEST는 매니저가 에이전트의 설정을 바꾸기 위한 요청이다. 마지막으로 TRAP은 위의 두 가지와는 달리 에이전트가 매니저에게 보내는 메시지로, 시스템등에 문제가 발생하는 경우 관리자에게 보고하는 메시지이다. 이는 매니저와 에이전트 사이에 특정한 port를 통해서 이루어지는데, 이에 대한 내용은 [그림1]과 같다.



[그림 1] 매니저-에이전트 통신을 위한 포트설정

2.2 MIB (Management Information Base)

MIB는 망 관리 대상에 대한 개념적인 root로부터 시작하는 트리 형식의 구조체이다. 각각의 관리 대상은 하나 또는 그 이상의 대상 객체(object instance)로 구성되고, 특정 객체 식별자(object identifier =OID)에 의해 확인된다. 트리 형식의 구조에서 SNMP는 최종 node인 leaf만을 읽고 쓸 수가 있다. 이때, 트리에서 깊이(depth)가 증가함에 따라 '.'로 구분을

* 이 논문은 BK21 대전.충남 정보통신인력양성사업단의 RA 연구비 지원에 의한 것임.

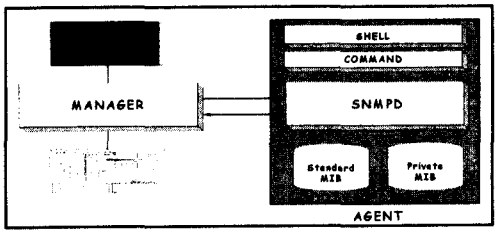
하게 된다. 가령 시스템의 위치를 나타내는 sysLocation은 'iso.org.dod.internet.mgmt.mib-2.system.sysLocation' 으로 식별하는데, 실제로는 '.1.3.6.1.2.1.1.6'과 같은 연속된 정수로 표현을 한다. 이때, system은 leaf가 아니기 때문에 읽기/쓰기가 허용이 되지 않고, leaf인 sysLocation만을 읽기/쓰기하게 된다.

3. CNU-SNMP의 설계 및 구현

CNU-SNMP의 구현 환경은 UCD-SNMP 2.4.1을 기반으로 CISCO 라우터를 위한 사설 MIB(Private MIB)를 참고하고, GUI는 FOUNDARY Networks사의 라우터를 참고하여, Linux Kernel 2.4.x에서 구현이 되었다.

3.1 매니저 – 에이전트 구성

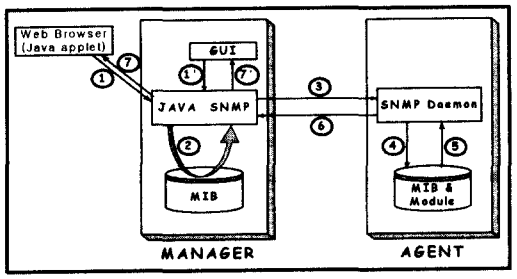
CNU-SMMP에서 구현한 SNMP의 매니저와 에이전트간 구성은 [그림 2]와 같다.



|그림 2| 매니저-에이전트 구성

[그림 2]에서 보는 것과 같이 매니저는 에이전트에게 GUI나 command를 통해 에이전트에게 정보를 요청하게 된다. CISCO ISO 12.0의 표준에 의한 CLI (Command Line Interface)를 통해 콘솔상에서 복잡한 명령어 체계를 통해서 메시지를 요청하게 되는 불편함을 없애기 위해, 명령어를 단순화시켜 콘솔상에서 관리가 가능하게 되었다. 이를 위해 별도의 Shell을 구현하게 되었다. 또한 사용자 관점에서 관리를 위해 Java를 이용한 GUI를 통해 관리함으로써, 효율적인 망 관리가 가능하게 되도록 설계하였다.

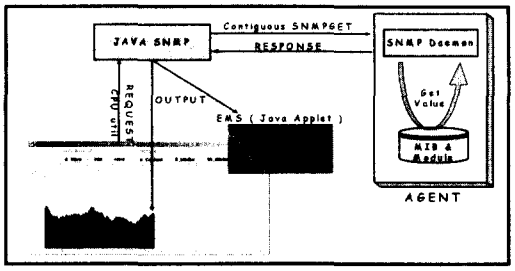
이러한 CNU-SNMP 에서 매니저-에이전트간 EMS 메시지 전송의 흐름을 단순화한 블록 다이어그램이 [그림 3]과 같다.



|그림 3| 매니저-에이전트간 EMS 동작 과정

[그림 3]과 같이 관리자는 GUI(①)나 Web Browser(①')를 통해 매니저인 Java SNMP에게 정보를 요청한다. Java SNMP는 MIB를 참고하여(②) 에이전트에게 요청을 하게 되고(③), 에이전트는 MIB나 MIB 모듈에 구현된 절차를 따라(④) 시스템 내에서 값을 가져와(⑤) Java SNMP에게 전달한다(⑥). Java SNMP는 GUI(⑦)나 웹 브라우저(⑦')에게 값을 보내면, 받은 값을 화면에 출력해 준다.

CNU-SNMP에서 구현된 모든 형태의 EMS 시스템은 위와 같은 절차를 걸쳐서 관리를 한다. 실례로, 에이전트의 CPU utilization을 가져와서 GUI나 웹 브라우저를 통해 관리자가 실제로 화면으로 보게 되는 절차를 보면 [그림 4]와 같다.

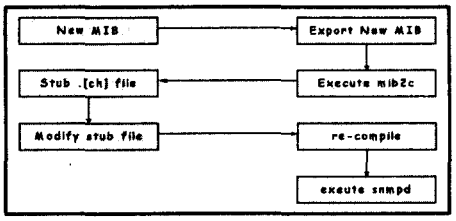


|그림 4| EMS 동작과정의 예(CPU utilization)

[그림 4]는 위에서 설명한 [그림 3]의 절차를 실제 수행하는 절차를 표현하고 있다. Linux 환경에서 CPU utilization은 "/proc/loadavg"에는 CPU 이용정도를 %단위로 표현해 주고 있으며 이 파일에는 CPU utilization의 snapshot, 5분 평균, 15분 평균등에 대한 정보가 있는데, CNU-CPU-MIB에 이러한 정보를 관리하는 객체가 정의되어있고, 이 객체에 대해 모듈에서 에이전트가 취해야 할 행동을 규정하고 있다.

3.2 사설(Private) MIB

본 논문에서 구현한 MIB와 MIB 모듈은 RFC 표준으로 제안된 표준MIB(Standard MIB)를 포함하고, 라우터 환경에 적용하기 위한 사설 MIB(Private MIB)를 중심으로 구현이 되었다. 앞으로 설명할 사설 MIB에 일부 표준 MIB의 내용을 포함하고 있다. MIB및 모듈의 추가는 다음의 [그림 5]의 절차를 걸쳐서 이루어지게 된다.

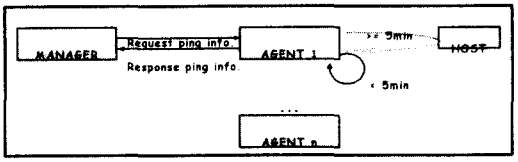


|그림 5| MIB 및 모듈 추가 절차

3.2.1 ping MIB

본 논문에서 구현한 ping MIB의 동작과정을 살펴보면 다음의 [그림 6]와 같다.

[그림 6]에서 보는 바와 같이 매니저는 에이전트에게 핑(ping)한 정보를 요청하게 된다. 이때 에이전트는 ping한 모든 호스트에 대한 정보를 자신이 파일로 관리를 한다. 그런



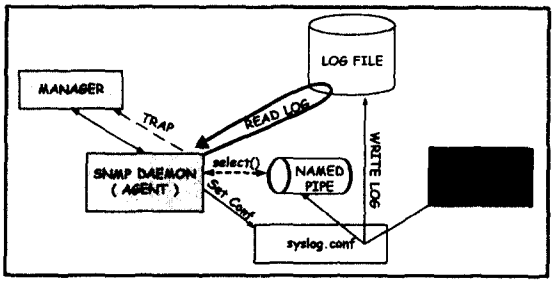
|그림 6| ping MIB의 동작

데, 무분별한 ping의 사용은 네트워크의 로드(load)를 증가시키기 때문에 로드를 줄이기 위한 방법으로, 요청한 값이 최근 5분 이내의 정보를 요청하게 되면, 에이전트는 자신의 파일에 저장된 내용을 가져와서 매니저에게 전달하게 된다.

반면 5분 이상이 된 정보는 네트워크의 변화를 충족시키지 못하기 때문에 호스트에게 ping을 하게 되고, 그 값을 파일에 적어 놓으면서, 매니저에게 값을 전달하게 된다. 이렇게 함으로써, 네트워크의 로드를 줄일 수 있다.

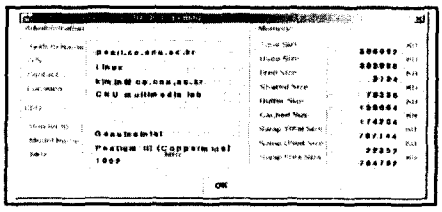
3.2.2 syslog MIB

syslog MIB는 시스템에서 발생하는 system log를 매니저가 관리하기 위해 만들어진 MIB및 모듈이다. 시스템 로그의 설정과, WRITE, READ 그리고 TRAP을 발생시키는 절차가 [그림 7]에 도시되어 있다.



[그림 7] syslog MIB의 동작

[그림 7]에서 살펴보는 바와 같이, 매니저는 에이전트에게 system log를 enable/disable 할 수 있고, 로그 파일의 위치와 severity level 등을 설정할 수 있다. 이때 syslog daemon은 설정된 configuration file을 참조하여 LOG FILE에 시스템등에 발생한 log를 기록하게 되는데, log가 발생하면 에이전트가 알 수 있도록 Named PIPE라 불리는 FIFO에도 로그를 남기게 된다. 에이전트는 named pipe를 select()함으로써 로그가 새로 남았음을 알 수 있고, 매니저에게 트랩을 발생시킨다. 매니저는 트랩 메시지를 확인하여, agent n으로부터 왔다는 사실과 시스템로그에 관한 트랩임을 알고, 에이전트에게 GUI나 EMS를 통해 로그에 관한 MIB정보를 가져오게 된다.



[그림 8] system information MIB

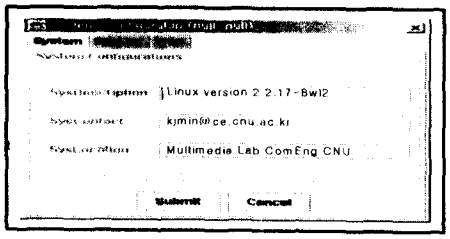
3.2.3 System Information MIB

system information MIB는 매니저가 에이전트의 시스템 정보를 얻기 위해 정의된 영역으로 Linux에서 /proc/meminfo, /proc/cpuinfo과, 표준 MIB의 system group을 참조하여 값을 얻어오는 형태로, CPU 모델이나, 총 메모리양, 사용된 메모리양 등을 나타내며, GUI의 모습은 [그림 8]과 같다.

3.3 Configuration & SET-REQUEST

시스템에 CNU-SNMP를 처음 setting하거나, 사용 중에, 값의 변화를 필요로 하는 경우가 자주 발생을 하게 된다. 이를 위해, Configuration의 기능이나, SET-REQUEST를 하게 되는데, 이를 GUI를 이용해 설정을 하는 경우가 [그림 9]과 같다.

CNU-SYSLOG-MIB에서 로그 파일에 남길 로그메세지의 severity의 설정이나, 로그메세지 관리에 대한 enable / disable 등을 설정하는 syslog configuration, 어떤 종류의 시스템 상황에 대해 트랩을 발생 시킬 것인가를 결정하는 TRAP configuration 등의 설정도 [그림 9]와 유사한 방식으로 결정할 수 있다.



[그림 9] System Configuration & SET-REQUEST

3.4 Command Line Interface

3.3에서 설명한 바와 같이, GUI나 EMS를 이용하지 않고 SNMP를 이용해 에이전트를 관리한다는 콘솔에서 복잡한 명령어 체계를 익히고, 트리 형식의 MIB전반에 대한 이해를 가지고 있어야 하는 불편이 있다. 이에 본 논문에서는 CISCO CLI 12.0을 기준으로 단순화된 명령어 SET을 만들고, 이를 위해 shell을 만듦으로써 콘솔상에서 관리도 가능하도록 하였다. 구현한 명령어 쉘은 [표 1]과 같다.

[표 1] shell 명령어 set

	명 령 어		명 령 어
1	Show snmp	8	snmp-server engineID
2	Show snmp engineID	9	snmp-server group
3	Show snmp group	10	snmp-server view
4	Show snmp user	11	snmp-server location
5	snmp-server chassis-id	12	snmp-server packetsize
6	snmp-server community	13	snmp-server user
7	snmp-server contact	14	no snmp-server

4. 결론 및 향후 연구 과제

지금까지 GUI , CLI 및 EMS와 연동 가능한 CNU-SNMP의 간단한 구현사례를 들어 최신의 표준 MIB 및 확장된 사설 MIB를 지원, 경제성 및 빠른 개발환경을 제공하는 Linux 기반 구현, 다른 OS로의 확장 가능성을 고려한 Java를 통한 Platform에 독립성, EMS와의 연동으로 웹을 통한 관리 및 쉘을 통한 단순화된 CLI를 통해 효과적인 망 관리가 가능함을 보였다.

향후 과제로는, EMS를 통한 WEB에서 보다 효율적인 관리, 환경에 맞는 사설 MIB및 모듈의 추가/구현, RTOS(Real Time OS)와 같은 다른 OS로의 porting에 대한 연구 진행 중이다.

참고 문헌

[1] W.Stallings, "SNMP and SNMPv2: the infrastructure for network management," IEEE Communications Magazine, Vol:36 Issue:3, pp.37-43, March 1998.

[2] B.Wijnen, R. Presuhn, K. McCloghеie, "View-based Access Control Model for the Simple Network Management(SNMP),"RFC 2275, January 1998.

[3] http://net-snmp.sourceforge.net/tutorial/

[4] ftp://ftpeng.cisco.com/pub/mibs/v2

[5] http://www.adventnet.com/products/javaagent/index.html

[6] http://www.mg-soft.com/agent.html

[7] http://www.foundrynetworks.com/services/documentation/