[Aus: "Heimat des Dilettantismus", "http://www.henrys.de/daniel/"]

SNMP
The Simple Network Management Protocol

When we were young, we wanted to become a pilot, astronaut or a fire engine driver.
-Our mothers would laugh at us.

When we were in our pre-teens, we wanted to become programmers.
-Our fathers would laugh at us.

When we became programmers, we wanted to become network administrators.
- The world (our girlfriends included) would laugh at us.

The above is not a figment of our imagination but facts relating to our daily life. When we started off as programmers we proved our fathers wrong. Sorry, we could not do the same regarding our mothers. This left us with the world and our girlfriends to deal with. It took us some time to devise ways to prove them wrong. But before we let you into our secret of how to do it, lets try and justify their laughs.

Network administrators of yore were perfect nerds, living in ivory towers(actually dingy low-lit rooms , disconnected from the world). In short, they were people to be avoided, basically belonging to class called social rejects. The rationale behind this was very simple. Network administrators had no control over their networks. The designation 'Network administrator' sounded glamorous but that was where the glamour ended. All he was supposed to do was to add/remove users to the network , add/remove network paraphernalia (or was it peripherals?). If he got bored of doing all this, he would set up the network all over again and life went on. (By the way, the world and his girlfriend had left him)

The Empire Strikes back

When he coined the title, George Lucas did not even dream that it may become a mission statement for many a network administrators on this planet. The network administrator it was high time to take control of the life (life had become synonymous with routers, bridges and other peripherals.) To a layman, getting control over his life meant knowing his wife, family, his pay-check, his dues(however unpleasant they may seem.) But to a network administrator, it meant knowing the traffic on his routers, the load on the server etc. But the big question was 'How does he do it? Simple SNMP ???

Sorry, we did not mean to shock you. We thought that you would have got impatient reading this stuff and so we decided to get down to business. SNMP is an acronym for Simple Network Management Protocol. It belongs to the rare breed of computer jargon which justify their name, i.e it is actually simple. As seeing is believing let us check out SNMP.

Imagine a situation where a company has 50 routers , 20 servers and all sorts of network hardware that runs the TCP/IP suite (A network administrators utopian dream!!). In such a situation questions like 'What is the load on the server?', 'What time does this loading take place ?' or 'When to expand the lease line?' etc formed part of the network administrators nightmare (almost lead to his downfall). To add to his sorrow, he had to work with diverse piece of equipment from diverse companies running equally diverse protocols. Variety is the spice of life you may say but this is the case of food (for thought) getting too spicy. The only thing that was common between the network elements was that they ran the TCP/IP suite. Therefore TCP/IP formed the lowest common denominator as it was the only thing we could standardize on. Beyond this standardization was almost zero, in some cases it approached the negative index. The Network Administrator had to figure out a way to manage his network which was running the TCP/IP suite. Therefore this gained the generic term network management under TCP/IP. Here is how the Network Administrators went about their work.

The first thing he did was to built himself another ivory tower (a cleaner one this time). This he named it as the 'Network Management Station'. He then went about stating the guidelines for the protocol to be called 'Simple Network Management Protocol (SNMP). The guidelines were as follows [in no order of importance]

1. The protocol should be cross-platform compatible (No we DON'T use JAVA). Put simply, it meant that the rules for talking to a router from CISCO should not differ from those used to talk to a router from 3COM. (All trademarks acknowledge)
2. The language used (if any) and the output generated should not increase the network traffic. Rationale:- Choking of network by the traffic should not lead the network administrator being fired.
3. It should be fast. Faster network always gave the network administrator a 'high'. Well that is one of the reasons as to why a network administrator did not buy a Ferrari.
4. The system was to work as if it were operating in a client-server atmosphere with the software running at the network management station acting as the client and the software running at the other network elements (jargonsss.... but basically stuff like routers, bridge , etc.) acting as the server. Perfect case of 'cart drawing the horse". (Taming of the shrew (or the network)). The factors or the variables affecting the server were to be stored in a database.

But before he did all this, he implemented the most important item on his agenda. He thought life was getting too simple due to all the 'simple' stuff floating around. He decided to rename everything that he set his eyes on. So the client software became a manager (to reflect his new status), the server software became the agent. (it would have been a better idea to call them spies because all they do was to pass on information about the network element). The database seemed to remind people of likes of ORACLE, SYBASE etc. So he decided to rename it as the Management Information Base or the MIB. The variables were formatted using a data definition language called the ASN.1, but more on this later. The variable itself was renamed as an object. The final objective of the entire exercise was to present a case of name changed to protect identity. We recommended you take a break (coffee or otherwise) because it is going to be a hell of a day today.

The Network Administrator had by now renamed most (if not all the) factors (name unchanged) involved in making up a network . Rule of nature state that every simple answer gives rise to complex questions. So the question was how does the Network Administrator know which element he was to communication with? He, therefore went around drawing a 'tree diagram' of the network. He arranged the factors involved as part of the tree. The tree diagram was used to

1. define administrative relationships
2. organize network management data
3. assign a unique identifier to every network management variable.

As he was still in the mood to rename things, so he decided to call the tree as the Structure of Management Information (SMI) . A sample of the tree diagram (or the SMI) is shown below.

The elements of the SIM were called as Groups or modules. These groups were nothing but a collection of objects (the good old variable) relating to that group. For example the Ethernet module contained further groups of Ethernet related stuff like DIX, 802.3 and so on. These were further subdivided to obtain groups or objects as the case may be. As each module/group and objects were numbered, accessing them became a child's play (not necessarily the network administrator's child) The modules at the end of any node was called the leaf node (network administrators are the most eco-friendly people on the planet)

The elements in the SMI have been assigned numbers. This makes traversing through the SMI easier. You always travel from the roots up (or down). For example to access the interfaces 'leaf', we traverse through root - iso - org - dod - internet - mgmt - mib - interfaces. As numbering has been enabled to access the same interface all we say is 1.3.6.1.2.1.2 (eco-friendly again - less waste paper).

Pressing the accelerator

By now, or so we hope you will be able to visualize the setup of the entire SNMP structure. If you don't then please hit the Page Up and go through it all over again. If you did understand then it is time for some action, Let's put all the pieces together. The manager can seek any data from the agent at any point of time. To do this the manager sends a UDP packet to the agent. Actually any other protocol can be used instead of UDP. Despite being unreliable UDP won the battle for the post of the messenger because of it's ability to have a very small packet size. This helped fulfill the second guideline of the specifications. The manager speaks to the agent on port 161 whereas the agent responds by communicating with the manager on port 162.

The SNMP version 1 has only 5 (unbelievable but true) commands in it's instruction set. They are listed as below

1. Get_Request - used to request the value of 1 or more MIB variable
2. Get_Next_Request - used to read the next value , works in a sequential way
3. Set_Request - used to update one or more MIB value
4. Get_Response - Returns an answer to the above 3 commands
5. Trap - used to report significant events on the network. Such events include 'cold or warm restart' or a failed link.

By the way, it reminds us to tell you that the instruction set is no longer called an instruction set but an SNMP Protocol Data Unit or a PDU. In the above PDU, the first 3 are generated by the manager. They are self explanatory . Hence lets not waste disk space talking about them . The other two are generated by the client. The differentiation comes in the fact that 'Get_Response' works in the polling mode i.e it forms the reply to a query from the manager, whereas 'trap' is generated by the agent only under certain condition independent of the presence or absence of a query i.e it is interrupt driven. For the uninitiated, lets put it in a simpler format. Polling (Get_Response) is akin to you asking your date for kiss. Interrupt driven (Trap) is your date giving you a kiss without you asking for it. Well we must admit, keeping your trap shut all the time does not do you any good !!!

SNMP version 2 is the latest version of SNMP commercially available. For once developers have retained the earthly flavor of the previous version. They struck to their task and have kept the SNMP as simple as possible. SNMP v2 is just a superset of SNMP v1. The 5 basic instruction of the SNMP v1 . PDU have been retained though some names have been changed (moods at play again). Get_Response whereas trap has been renamed as SNMPv2 - trap. This however does not affect them. Functionality remain the same. The 'Get' function have also been strengthened. In SNMP v1 the get_request and get_next request would fail in case one of the read processes failed. Generation of error signals was virtually non-existant. However in SNMPv2 request failure leads to appropriately error messages generation. Besides these 5 instructions, SNMPv2 includes 2 new instruction

1. inform request
2. get_bulk_request

Inform request is used by a network administrator to talk to his brethren i.e. network administrator at other network management stations. Birds of a feather flock together. This enables a large complex network to be further subdivided into smaller network, each with his own network management station. Get_Bulk_Request is used to transfer large block of data from the MIB to the Manager in a single transfer operation. That's all that consists the PDU of SNMPv2.

Lets turn our attention to the objects that constitute the MIB of a device . An object is actually a variable which holds information about a particular device. An object is known by the following

1. A unique name, called object identifier (remember the mood)
2. Attributes :- This include
   - - data type
   - details required for the correct implementation.
   - - status information
3. Read/Write option available

On deciding the datatypes to be used in SNMP, the designers remembered their motto of keeping SNMP simple. Therefore only elementary datatypes like integer or octet strings are available under SNMP. The only other datatype was the structure which unfortunately doomed to be renamed as 'Sequence'. These were the specification for SNMP. The answer was simple , use a Data Definition Language like ASN.1

One of the greatest mystery of our time have been to know how long the server at our ISP's end has been functioning. This mystery is deepened by some cryptic message from the help-desk of our ISP, saying that the server will be shutdown for maintenance. We decided to pip them at the post by finding out how much time the server was up (or down) This can be done using the object identifier called 'sysuptime' in the object called 'system'. The object identifier for sysuptime is 1.3.6.1.2.1.3. The program given below uses a basic windows socket program. How to master socket programming has been best explained in our tutorial on sockets programming so check it out. Here we will concentrate only on the bits & bytes that are sent across and received.

We sent a UDP packet on port 161 to the server at VSNL (our ISP). The IP address of the server is 202.54.1.18, but you can change it to any IP address you like. Since WindowsNT 4.0 has an in-built SNMP service, this program can also be tried out using NT. The initial parts of the program involves a simple Windows socket program and hence is not explained here. For details on such programs, have a

look at our tutorial on sockets programming. The only section of the code that might intrigue you with it's presence are the functions 'abc' and 'abc1'. These are our in-house helper functions and are used to write the trapped bytes to the hard disk. The latter part of the program deals with the process of sending a data packet to the server. The various bytes have been explained below.

SNMP.C

```
#include <windows.h>
#include <stdio.h>
unsigned char kk[1000],ll[1000];
void abc(char *p)
{
FILE *fp=fopen("c:\\snmp\\z.txt","a+");
fprintf(fp,"%s\n",p);
fclose(fp);
}
int ii,dw,jj;
void abc1(unsigned char p)
{
FILE *fp=fopen("c:\\snmp\\z.txt","a+");
fprintf(fp,"%d:   %x..%d..%c\n",jj,p,p,p);
fclose(fp);
}
WNDCLASS a;HWND b;MSG c;char aa[200];SOCKET s;struct hostent h;
WSADATA ws;DWORD e;char bb[100];struct sockaddr_in sa;
long _stdcall zzz (HWND,UINT,WPARAM,LPARAM);
int _stdcall WinMain(HINSTANCE i,HINSTANCE j,char *k,int l)
{
a.lpszClassName="a1";
a.hInstance=i;
a.lpfnWndProc=zzz;
a.hbrBackground=GetStockObject(WHITE_BRUSH);
RegisterClass(&a);
b=CreateWindow("a1","SNMP manager",WS_OVERLAPPEDWINDOW,1,1,10,20,0,0,i,0);
ShowWindow(b,3);
while ( GetMessage(&c,0,0,0) )
DispatchMessage(&c);
return 1;
}
long _stdcall zzz (HWND w,UINT x,WPARAM y,LPARAM z)
{
if ( x == WM_LBUTTONDOWN)
{
e=WSAStartup(0x0101,&ws);
sprintf(aa,"WSAStartup e = %ld",e);
```

```
s = socket(PF_INET,SOCK_DGRAM,0);
sprintf(aa,"socket s = %ld",s);
sa.sin_family=AF_INET;
sa.sin_addr.s_addr = inet_addr("202.54.1.18");
sa.sin_port=htons(161);
kk[0]=0x30;
kk[1]=0x25;

kk[2]=2;
kk[3]=1;
kk[4]=0;

kk[5]=4;
kk[6]=6;
kk[7]='p';
kk[8]='u';
kk[9]='b';
kk[10]='l';
kk[11]='i';
kk[12]='c';

kk[13]=0xa1;
kk[14]=0x18;

kk[15]=2;
kk[16]=1;
kk[17]=1;

kk[18]=2;
kk[19]=1;
kk[20]=0;

kk[21]=2;
kk[22]=1;
kk[23]=0;

kk[24]=0x30;
kk[25]=0xd;

kk[26]=0x30;
kk[27]=0xb;
```

```
kk[28]=6;
kk[29]=7;
kk[30]=0x2b;
kk[31]=6;
kk[32]=1;
kk[33]=2;
kk[34]=1;
kk[35]=1;
kk[36]=3;
kk[37]=5;
kk[38]=0;

e=sendto(s,kk,39,0,(struct sockaddr *)&sa,sizeof(sa));
sprintf(aa,"SendTo %ld",e);
dw = sizeof(sa);
ii=recvfrom(s,ll,1000,0,(struct sockaddr *)&sa,&dw);
sprintf(aa,"Recv from %d",ii);
abc(aa);
for (jj=0;jj<ii;jj++)
abc1(ll[jj]);
MessageBox(0,"hi","hi",0);
}
if ( x == WM_DESTROY)
PostQuitMessage(0);
return DefWindowProc(w,x,y,z);
}
```

The first byte we send across to the server (it can also be a router) at the other end is 0x30. This stands for The Universal Sequence and is usually the first byte in BER/DER compliant protocols.

If we break up the byte into it's individual bits, we get

The first two bits from the left, when off, imply that the query is universal, i.e. it applies to all fields. The next bit from the left is on and that means that the query is a constructed one. The value of the fourth bit is 16 and that's Sequence in the RFC. So it's a Universal Sequence.

The next byte in the array holds the size of the total packet, excluding the first two bytes.

The next byte is 0x02, which means that the information that follows is an Integer, i.e. a number. The byte in kk[3] is the length of the data and kk[4] is the actual value. The 0x00 there stands for the version number which is zero.

The byte that comes next is 0x04 which means that the following data is an octet (8 bit) string. The byte after that holds the length of the string and next six bytes hold the string itself. The string 'public' stands for the community we belong too; that's just a fancy way of saying it's our password!

The byte in kk[13] is 0xa1. If we were to break this up into it's constituent bytes, we'd get

1	**0**	1	0	0	0	0	1

When the first three bytes are 101, it means that the data to follow is Context Specific i.e. you have to look up the documentation and look for it as it's context changes with every protocol. The last four bits hold the number, which is 1 in this case. So it's Context Specific 1. This stands for Get Next Request in the documentation.

The next byte holds the length.

The next three bytes hold the Request ID, which we've set to one. Kk[15] is 0x02 which means that the data is an Integer. The next byte is the length and the data itself is 0x01.

The next three bytes contain the error status of the present connection. Since we've just begun talking, it is set to zero.

Right after that we have another three bytes which hold the error index. It too is set to zero.

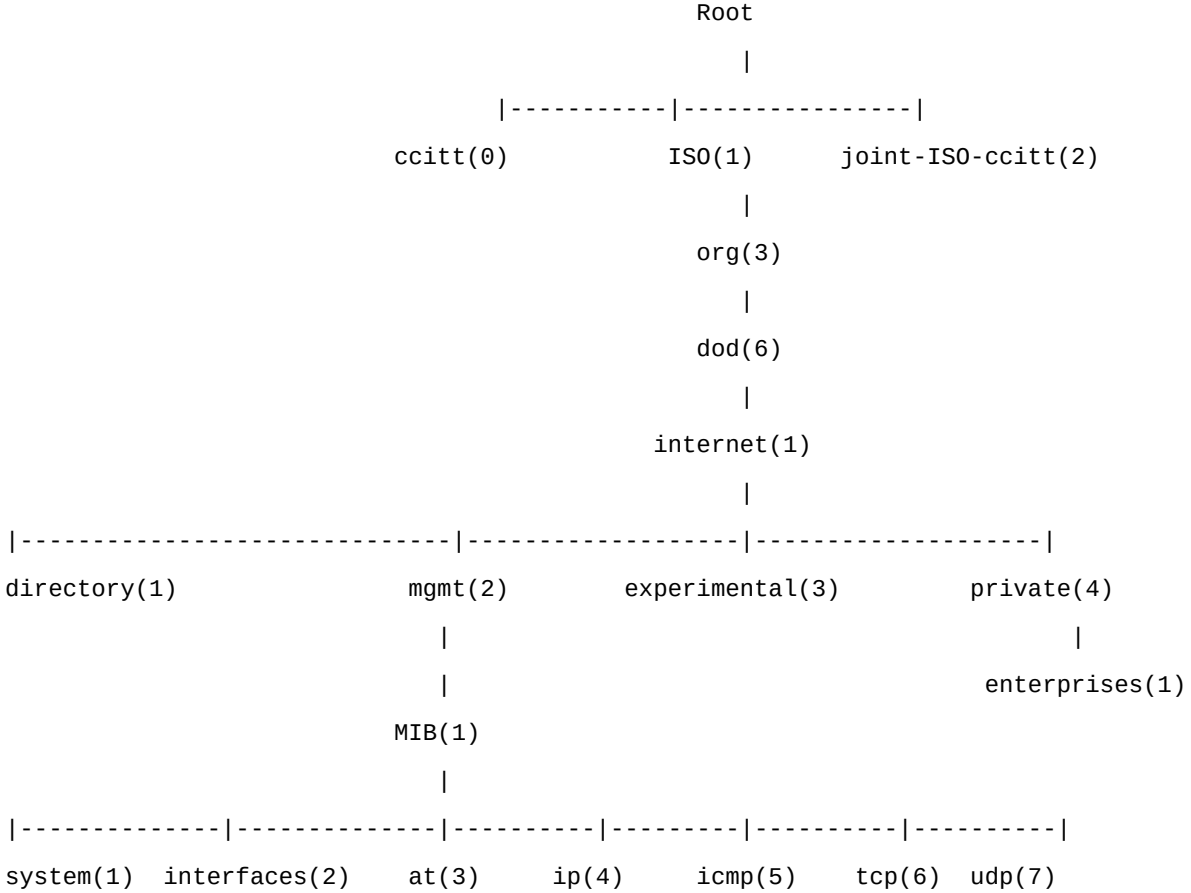
Now, at kk[26] we have another structure (or sequence if you prefer) starting up. The byte after that is the length of the data that follows.

Immediatly after the first structure, we have another one. This format is maintained so that if you want to ask two questions at the same time, you can. Simply add another 0x30 after the end of this one and more data.

After the length byte we have the byte 0x06 which stands for the Object data type. The byte after that is the length.

The next seven bytes are very important.

The ISO, in it's infinite wisdom, has chopped up the Internet into a tree like hierarchy.



Right at the top is the root, then comes the ISO (1), then org under ISO (1.3), then the Department of Defense or the dod under org (1.3.6). Right after that we have the Internet (1.3.6.1) under the Internet we have mgmt (1.3.6.1.2) and under mgmt we have MIB or the Management Information Database (1.3.6.1.2.1). Since we're using SNMP which is used to manage a network system, we have system (1.3.6.1.2.1). So the hierarchy we're under is 1.3.6.1.2.1. The last 3, in kk[36], is Sysuptime under System.

So the bytes from kk[30] to kk[36] hold the hierarchy of our query. What you may find a little confusing is that the first byte, kk[30], is 0x2b instead of 1.3 as it should be. Actually, this is the ISO being little clever (for once!). To save transmitting an extra byte, they decided to multiply the first 1 in 1.3.6.1.2.1.3 with 40 and add it to the second number i.e. 3. So we get 1*40+3=43 or 0x2b!! Confusing, but neat.

The last two bytes in the packet are 0x05 and 0x00 which collectively stand for a NULL value. This is because we can't have an answer in a query!

The output file provides the key to understanding SNMP. The contents of the output file is reproduced below to save you the trouble of actually running the programs and finding the results. The column on the left are the actual contents of the output file whereas the column on the right hand side represent the analysis of the bytes that were 'collected' by us.

Z.TXT Recv from 44 - Inserted by the program and signifies that length of packet is 44 bytes

0	30	48	0	**Structure**
1	2a	42	*	Len - 42
2	2	2		**Version number** (int)
3	1	1		Len -1
4	0	0		value - 0
5	4	4		**Community** (Octet String)
6	6	6		Len -6
7	70	112	p	
8	75	117	u	
9	62	98	b	
10	6c	108	l	
11	69	105	i	
12	63	99	c	public
13	a2	162	˜	**Context Specific** 2
14	1d	29		Len -29
15	2	2		**ID** (int)
16	1	1		Len -1
17	1	1		value -1
18	2	2		**Error Status** (int)
19	1	1		Len -1
20	0	0		Value -0
21	2	2		**Error Index** (int)
22	1	1		Len -1
23	0	0		Value -1
24	30	48	0	**Strucutre**
25	12	18		Len -18
26	30	48	0	**Structure**
27	10	16		Len - 16
28	6	6		**Object**

29	8	8		Len -8
30	2b	43	+	**Sysuptime**
31	6	6		Len -6
32	1	1		
33	2	2		
34	1	1		
35	1	1		
36	3	3		
37	0	0		1.2.1.1.3.0
38	43	67	C	**Time Ticks**
39	4	4		Len -4
40	0	0		the time in milliseconds
41	bc	188	ź	
42	c8	200	Č	
43	a4	164	¤	

Lets examine the output now.

The first byte we see is 0x30 which stands, as before, for the start of a structure (or a sequence). The next byte after that is the length of the packet to follow. The next three bytes hold an Integer whose value is zero. This is the version number we sent across in our query. Now come eight bytes for a string (strings always start with a 0x04) which holds our password.

Byte number 13 is 0xa2. If broken up into it’s individual bits it looks like this

1	0	1	0	0	0	1	0

Since the first three bits are 101 it means that this is ’Context Specific’; the last four bits hold the number which is two in this case. So it is Context Specific 2 which means that this is our asnwer.

The byte after that is the length of the data to follow.

The next three bytes contain an Integer which is the ID of the packet (it’s the same as our ID i.e. it’s one). The three bytes after that hold the Error status, which is still zero and the following three bytes hold the error index, which is also zero.

Now come two bytes for a structure and it’s length. The next two bytes also contains a structure and it’s length.

Then we have the number 0x06 which as mentioned before represents an Object. It is immediately followed by it’s length.

Now comes the tree like hierarchy our query belongs to and then another 6 bytes that hold our answer. The last zero means that the bytes following it hold the answer to our query about SysUpTime.

Byte 38 contains the number 0x43 which stands for Time Ticks. The string that follows holds the time, in milliseconds, that the server 202.54.1.18 has been up.

We have not taken the trouble of calculating the corresponding time in days and hours for a simple reason that the answer obtained from the server was to large to fit into the calculator provided with Windows95. We hope that this program infuses in you a desire to use the various objects provided with SNMP. To do this just replace the object identifier used in the above program with the desired identifier. Then you can check the output file for the answer to your query. Do let us know about your success and travails with SNMP. Also provide us with your feedback as to how you liked this tutorial.

The above tutorial is a joint effort of

Mr. Vijay Mukhi
Ms. Sonal Kotecha
Mr. Arsalan Zaidi
Mr. Vinesh Kurup

Back to the main page

Vijay Mukhi's Computer Institute
VMCI, B-13, Everest Building, Tardeo, Mumbai 400 034, India
Tel : 91-22-496 4335 /6/7/8/9 Fax : 91-22-307 28 59
e-mail : vmukhi@giasbm01.vsnl.net.in
http://www.neca.com/~vmis