Facultatea de Electronica, Telecomunicatii si Tehnologia Informatiei

Managementul retelelor mari folosind SNMP

Cristian Iorga,

Master ISC, grupa 471A

Cuprins

Network Management Systems (NMS)

Managementul retelelor (**network management**) se refera la mentenanta si administrarea retelelor mari de calculatoare si de telecomunicatii la nivel superior. Managementul retelelor consta in executia unui set de functii necesare pentru controlul, planificarea, alocarea, coordonarea si monitorizarea resurselor retelei.

Cresterea continua atat a retelelor service-providerilor cat si a retelelor de tip “enterprise” (guvernamentale, banci, companii mari etc) a fost insotita de o cerere crescuta pentru instrumente software avansate de gestionare a retelelor, independente de vendor-ul echipamentelor ce alcatuiesc reteaua. Managerii retelelor enterprise se confrunta cu o misiune descurajatoare datorita diversitatii mari a elementelor de retea (network elements): routere multi-vendor, swtch-uri, echipamente radio, linii inchiriate, servere (Unix, Linux, Windows), diverse aplicatii critice (de exemplu aplicatii de facturare a serviciilor) avand in spate baze de date diferite (Oracle, SQL Server etc.) si a tehnologiilor: Ethernet, MPLS, ATM, DWDM etc . Toate elementele enumerate anterior reprezinta motorul companiilor iar disponibilitatea acestora este cruciala. Instrumentele software existente care usureaza gestionarea retelelor mari, trebuie dezvoltate constant pentru a se adapta la noile tehnologii si produse ce vor fi integrate in retea. De aceea managerii retelei trebuie sa aiba o intelegere clara atat a modului in care pot fi gestionate noile echipamente si produse pentru a comunica cerintele catre dezvoltatorii software-ului de management precum si a modului in care instrumentele de management sunt proiectate si functioneaza. Similar, pentru a putea dezvolta instrumente eficiente din punct de vedere al costului de exploatare, dezvoltatorii trebuie sa aiba o intelegere clara a provocarilor cu care se confrunta managerii in gestionarea retelelor si cum aceste provocari determina costul management-ului retelei.

Software-ul folosit pentru managementul retelor mari (date/telecom) poarta numele generic de “Network Management System”, pe scurt NMS. Acestea sunt solutii com plexe, de multe ori distribuite, avand o disponibilitate foarte ridicata (high availability). Avantajele oferite de NMS-uri sunt urmatoarele:

- descoperirea automata a elementelor retelei
- vedere grafica a retelei

- managementul defectelor (**fault management**)
- managementul performantelor (**performance management**)
- configurarea de la distanta a elementelor de retea
- corelarea alarmelor (**root cause analisys**)
- trimiterea de notificari prin sms/email cand sunt probleme
- crearea de statistici/rapoarte cu privire la problemele din retea

In 1996 ITU-T a introdus FCAPS, un model pentru gestionarea retelelor. FCAPS este un acronim pentru: F*ault, Configuration, Administration, Performance, Security* care reprezinta categoriile de management acoperite sub termenul de "network management"
Fault Management – scopul fault managementului este de recunoaste, izola, corecta si inregistra caderile care apar in retea. In momentul in care apare o problema o notificare este trimisa catre o consola de monitorizare sau/si catre o lista de persoane prin mail sau sms. Caderile sunt inregistrate si pe baza lor se fac statistici si predictii.
Configuration Management – in retelele mari, gestionarea configuratiilor este foarte importanta. Gestionarea configuratiilor isi propune:

- strangerea si stocarea configuratiilor de la echipamentele de retea
- simplificarea configurarii echipamentelor
- urmarirea schimbarilor facute in configuratii
- automatizarea configurarilor (de exemplu maparile dintre adrese si numele DNS)

Administration Management – scopul este de a administra un set de utilizatori autorizati, prin definirea utilizatorilor, parolelor si a drepturilor de acces, si de a administra operatii ale echipamentelor cum ar fi backup-ul si sincronizarea.
Performance Management – permite managerilor sa determine eficienta curenta a retelei Gestionarea perfomantelor adreseaza chestiuni ca latimea de banda, utiliarea retelei, rata erorilor, timpi de raspuns etc. Prin colectarea datelor legate de performanta, pot fi depistate tendinte ce indica probleme in buna functionare a retelei inainte de a fi afectate serviciile oferite.
Security Management – este procesul prin care se controleaza accesul la resursele retelei. Presupune actiuni ca autorizare, autentificare, criptare, folosirea de sisteme de detectie si prevenire a intruzionilor etc.

Arhitectura unui sistem NMS este prezentata in figura 1.

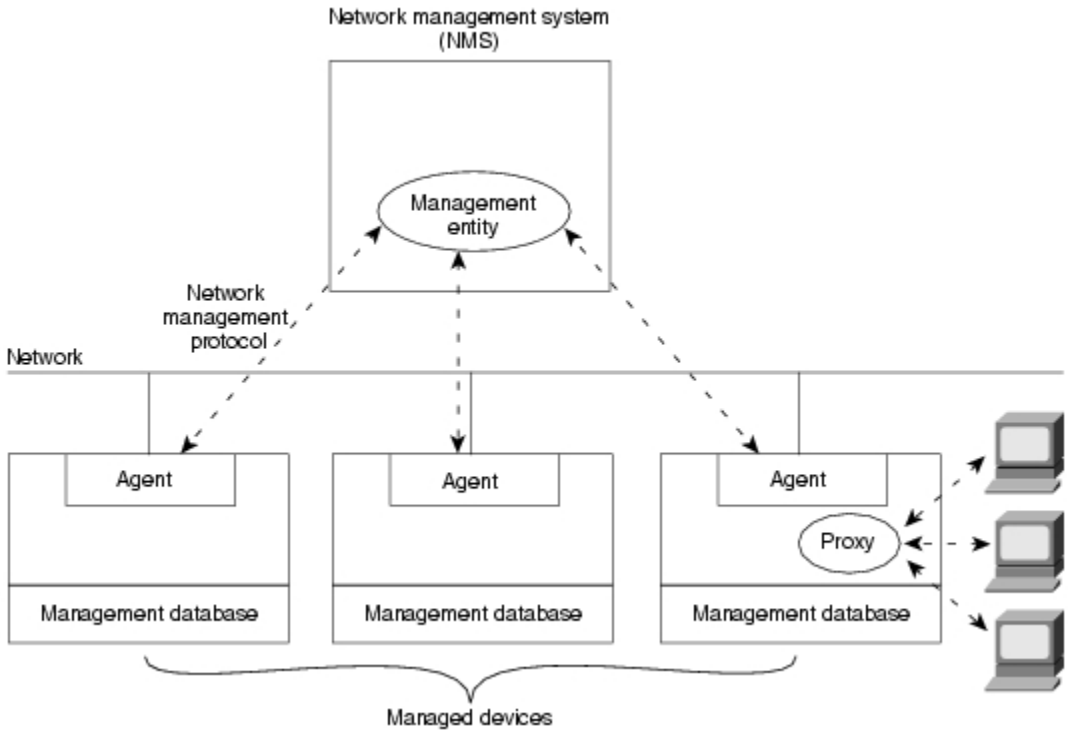

Figure 1Arhitectura unui NMS

Exista un numar mare de protocoale pentru gestionarea retelelor (**network management protocols**). Protocoale cunoscute sunt: SNMP - Simple Network Management Protocol, CMIP – Common Management Information Protocol, WBEM – Web-Based Enterprise Management, TL1 – Transaction Language 1, JMX – Java Management Extension etc.

Protocolul SNMP este in acest moment cel mai utilizat protocol pentru gestionarea retelelor. Acesta este un protocol “simplu” si eficient, ajungand prin versiunea 3 la maturitate. In continuare va fi descrisa evolutia acestui protocol, functionarea lui precum si modalitati de utilizare.

Protocolul SNMP

Introducere

SNMP este un protocol standardizat de catre IETF (Internet Engineering Task Force), avand drept utilizare usurarea monitorizarii si managementului echipamentelor din retele IP. Fiind suportat de foarte multe tipuri de echipamente si "simplitatea" sa, au facut ca acesta sa fie cel mai utilizat protocol pentru "network management".
Prima versiune, SNMPv1, a aparut in 1988 ca o nevoie de a avea un standard pentru managementul retelelor a caror dimensiunea crestea exponential. SNMPv1 a fost initial descris in 3 RFC-uri (Request For Comments): 1065 (**Structure and Identification of Management Information for TCP/IP-based internets**), 1066 (**Management Information Base for Network Management of TCP/IP-based internets**) si 1067 (**A Simple Network Management Protocol**) care au fost inlocuite cu RFC-urile 1155, 1156 si 1157. Autentificarea in acest protocol se face pe baza de comunitati, care sunt de fapt niste parole care sunt transmise in text clar pentru ca aplicatiile de management sa poata accesa dispozitivele SNMP. Aceasta este o problema de securitate majora, dar in ciuda acestui fapt SNMPv1 este inca versiunea principala suportata de multi vendori. Versiunea a doua protocolului, SNMPv2c, este definit de RFC-urile 3416, 3417, si 3418. Litera "c" de la sfarsitul versiunii indica ca protocolul este bazat pe comunitati. Ultima versiune a protocolului, SNMPv3, aduce cea mai importanta contributie in ceea ce priveste securitatea: autentificare puternica si comunicatie privata intre entitatile gestionate. In 2002 s-a facut tranzitia pentru versiunea 3 de la un standard "draft" la un standard complet. SNMPv3 este descris in RFC-urile 3410, 3411, 3412, 3413, 3414, 3415, 3416, 3417, 3418 si 2576.

In lumea SNMP exista doua tipuri de entitati: managerul – este un server pe care ruleaza un software de management (NMS) si care este responsabil pentru primirea trap-urilor (datagrame informationale trimise asincron, nu ca efect al unei cereri) si interogarea agentilor (polling); agentul – este un software care ruleaza intr-un element de retea gestionat. Poate fi un proces separat (un daemon in Unix de exemplu) sau poate fi incorporat in sistemul de operare (de exemplu in IOS la routerele Cisco). Agentul pastreaza un istoric al aspectelor operationale ale unui dispozitiv, raspunde cu informatiile necesare la cererile NMS-ului iar in cazul in care apar evenimente care influenteaza buna functionare a dispozitivului, trimite trap-uri catre NMS. Interactiunea dintre manager si agent este reprezentata in figura 2. Fiecare agent detine o lista a obiectelor gestionate si a starii lor (de exemplu un agent intr-un router gestioneaza toate interfetele, fiecare interfata fiind un obiect gestionat). Structura informatiei de management (**Structure of Management Information – SMI**) ofera o cale de a defini obiectele gestionate si comportamentul lor. Baza informatiei de management (Management Information Base (MIB) poate fi vazuta ca o baza de date de obiecte gestionate pe care un agent le urmareste. Orice stare sau informatie statistica care poate fi accesata de NMS este definita intr-un MIB. SMI-ul ofera calea de a defini obiectele iar MIB-ul este definitia obiectelor folosind sintaxa SMI. Un agent poate implementa mai multe MIB-uri, dar toti agentii implementeaza MIB-II (RFC 1213). Scopul principal al

acestui MIB este de a defini informatii de management generale pentru TCP/IP cum ar fi numarul de octeti trimisi/receptionati, viteza interfetelor, MTU (Maximum Transfer Unit) etc. Un numar semnificativ de MIB-uri au fost propuse ca standard pentru a gestiona multe lucruri cum ar fi ATM (RFC 2515), Frame Relay (RFC 2115), RDBMS (RFC 1697), DNS Server (RFC 1611) etc.Pe langa aceste MIB-uri standard, fiecare vendor poate creea MIB-uri proprietare pentru a putea monitoriza elemente specifice tipului sau de echipament.

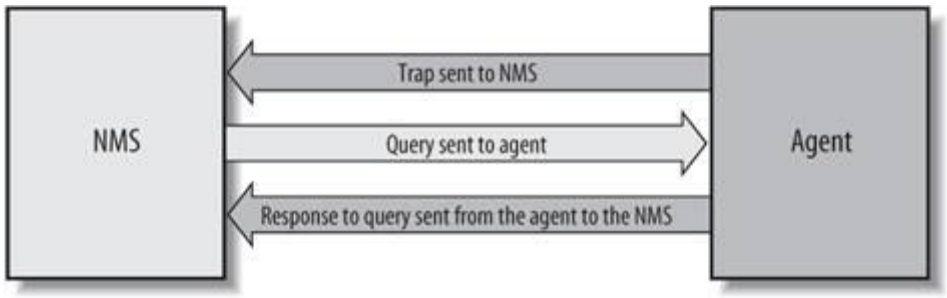

Figure 2 Interactiunea NMS-Agent

Datagrama UDP a protocolului SNMP

SNMP-ul utilizeaza Uniform Datagram Protocol (UDP) ca protocol de transport pentru a pasa datele intre agenti si manageri. UDP-ul (RFC 768) a fost ales in ciuda TCP-ului (Transmission Control Protocol) datorita lipsei unei conexiunii intre cele doua entitati care comunica, in acest fel resursele folosite fiind reduse. Dezavantajul folosirii UDP-ului este ca se pot pierde datagrame ce contin trap-uri, fara a exista posibilitatea sa se notifice acest lucru.
SNMP-ul foloseste portul UDP 161 pentru a trimite si a primi cereri si portul 162 pentru a primi trap-uri de la dispozitivele gestionate. Comunicatia dintre NMS si agent are loc in felul urmator (vezi Figura 3): la nivelul aplicatie se decide tipul actiunii ce va fi intreprinse: agentul trimite un trap catre NMS, NMS-ul trimite o cerere catre un agent sau agentul. Nivelul aplicatie ofera servicii catre un end-user (in cazul nostru un operator), de exemplu afisarea starii unui port a unui switch. Nivelul UDP permite la doua gazde sa comunice; header-ul UDP contine printre alte informatii si portul destinatie al dispozitivului catre care se trimite trap-ul/cererea (161 pentru cerere, 162 pentru trap-uri). Nivelul IP incearca sa livreze pachetele la destinatie iar nivelul fizic este responsabil de transmiterea informatiilor pe mediul fizic.

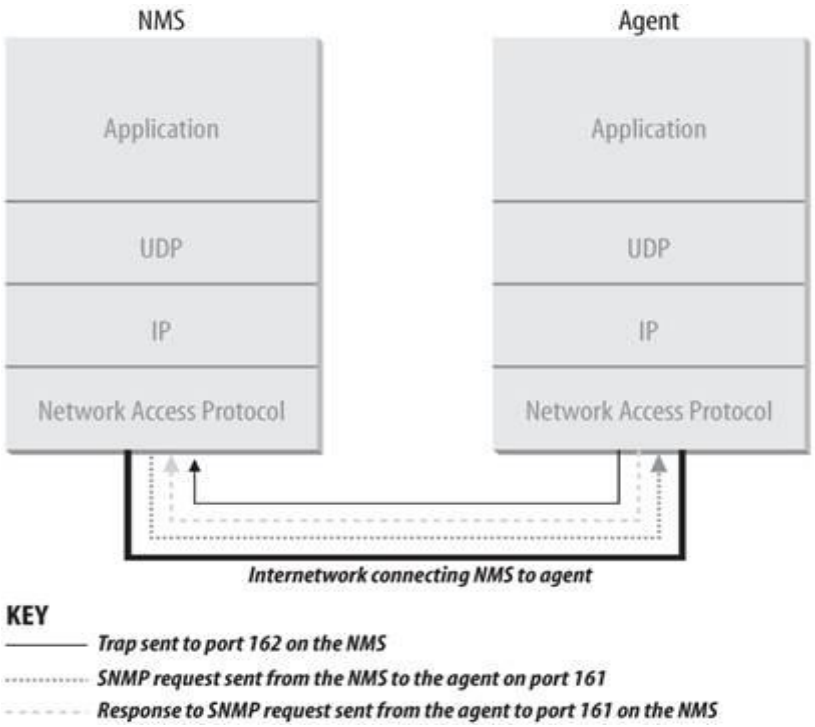

Figure 3 Modelul de comunicatie TCP/IP

Comunitati SNMP

SNMPv1 and SNMPv2 folosesc notiunea de comunitate ca mecanism de securitate pentru comunicatia dintre NMS si agent. Acestea nu sunt decat simple parole care au marele dezavantaj de a fi transmise in clar la versiunile 1 si 2. Exista 3 tipuri de comunitati: **read-only, read-write si trap**. Primul tip de comunitate, asa cum si numele sugereaza, permite doar citirea datelor nu si modificarea acestora (de exemplu citirea numarului de octeti transmisi/primiti pe o interfata). Comunitatea read-write permite atat citirea datelor cat si scrierea acestora (de exemplu resetarea unui router se poate face prin setarea unei variabile in agent). Comunitatea “trap” permite receptionarea trap-urilor (notificari asincrone) de la agent. Majoritatea vendorilor livreaza echipamentele cu comunitati setate implicit si anume “public” pentru comunitatea read-only si “private” pentru comunitatea “read-write”. Acestea trebuie schimbate cu atat mai mult daca informatiile de management vor traversa si alte retele. Pentru a reduce riscurile atacurilor bazate pe SNMP, firewall-urile pot fi setate sa permita trafic pe portul UDP 161 (cereri SNMP) doar daca au ca origine adresa IP a NMS-ului. In mod asemanator agentii pot fi configurati sa comunice doar cu adresa IP a NMS-ului. Cand agentul primeste cereri SNMP de la adrese IP straine sau cand o autentificare esueaza, acesta poate fi configurat sa trimita trap-uri.

Trap-uri SNMP

Un trap este o notificare asincrona trimisa de un agent SNMP catre un NMS folosind protocolul UDP (port 162). De fapt un trap este un grup de date care este definit intr-un MIB si este unic identificat printr-un ID. Trap-urile sunt de doua categorii: generice si specifice unui vendor. Trap-urile generice sunt in numar de 7 si sunt urmatoarele:

coldStart (0) – indica faptul ca agentul a fost rebootat; toate variabilele de management vor fi resetate

warmStart (1) – indica faptul ca agentul s-a reinitializat. Nici una din variabilele de management nu vor fi resetate

linkDown (2) – trimis cand o interfata a unui dispozitiv este cazuta.

linkup (3) – trimis cand o interfata a unui dispozitiv se ridica

authenticationFailure (4) – indica o incercare de interogare a agentului cu o comunitate gresita.

egpNeighborLoss (5) – indica ca un vecin EGP a cazut.

enterpriseSpecific (6) – indica ca trap-ul este vendor specific. Pentru a procesa acest trap correct, NMS-ul trebuie sa decodeze numarul trap-ului specific care este o parte a mesajului SNMP.

Unul din punctele forte ale SNMP-ului este posibilitatea de orice vendor de echipamente sa isi construiasca propriul MIB si sa monitorizeze orice particularitate a echipamentului sau crede de cuviinta. Un trap specific este identificat in mod unic de ID-ul vendorului si de numarul trap-ului care este dat de catre vendor.

Trap-urile sunt receptionate de catre NMS care, in functie de complexitatea sa, poate sa afiseze un mesaj pe o consola de monitorizare, poate trimite un mail/sms catre o persoana responsabila de evenimentele din retea sau poate chiar sa intreprinda actiuni ca de exemplu resetarea unui echipament (tot prin intermediul SNMP). In figura 4 este fereastra alarmelor (trap-urilor) afisata de HP OpenView Network Node Manager, unul dintre NMS-urile de top existente pe piata.

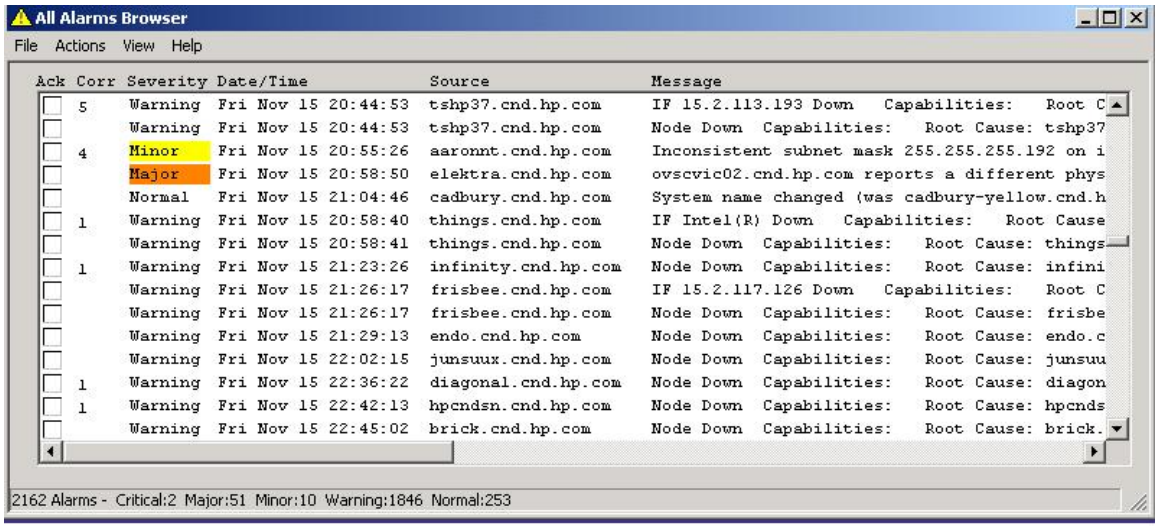

Figure 4 Consola cu alarme (trapuri) in HP OV Network Node Manager

The Structure of Management Information (SMI)

Pentru a intelege ce tip de informatii un echipament poate furniza, trebuie sa intelegem cum sunt datele reprezentate in contextul SNMP-ului. Structura informatiei de management (SMI) defineste exact cum obiectele gestionate sunt denumite si ce tipuri de date au asociat. SMIv1 este descrisa in RFC 1155 dar in cursul timpului au aparut imbunatatiri concretizate in SMIv2 (RFC 2578).
RFC 1155 contine descrierea unui model informational pentru managementul retelelor impreuna cu un set generic de tipuri de date utilizat pentru descrierea informatiei de management.

Definirea obiectelor gestionate poate fi impartita in trei atribute:

Numele – numele, sau identificatorul obiectului (object identifier – OID), defineste in mod unic un obiect gestionat. Numele apar in doua forme, una numerica si una care poate fi usor citita
.
Tipul si sintaza – tipul de date al obictului gestionat este definit utilizand un subset al Abstract Syntax Notation One (ASN.1). ASN.1 este un standard care descrie structuri de date pentru reprezentarea, codarea, transmiterea si decodarea datelor. Acest standard ofera un set de reguli formale pentru descrierea structurii obiectelor care sunt independente de tehnicile de codare specifice unei masini. ASN.1 a fost definit in 1984 (colaborare intre OSI si ITU-T) ca parte a standardului CCITT X.409, in 1988 a fost definit de propriul standard X.208 iar in 1995 dupa revizii substantiale este acoperit de seria de standarde X.680. ASN.1 defineste sintaxa abstracta a informatiei dar nu restrictioneaza modul in care informatia este codata. Variate reguli de codare ofera sintaxa de transfer a valorilor datelor a caror sintaxa este descrisa in ASN.1: BER (Basic Encoding Rules), CER (Cannonical Encoding Rules), XER (XML Encoding Rules) etc. ASN.1 este folosit in protocoale ca SMTP, H323 (VoIP), SNMP etc.

Codarea – o instanta a unui obiect gestionat este codata intr-un sir de octeti folosind Basic Encoding Rules (BER). BER defineste cum obiectele sunt codate si decodate pentru a fi transmise peste un mediu de transport cum este Ethernet-ul. Regulile de codare sunt denumite in mod generic **sintaxa de transfer** in contextual ASN.1 Aceasta sintaxa defineste elemente cum ar fi reprezentarea pentru tipuri de date de baza, structura informatiei de lungime si mijloacele pentru definirea tipurilor de date compuse bazate pe mai multe tipuri de date primitive. Fiecare element de date este codat ca un identificator de tip, o descriere a lungimii, datele efective si, acolo unde este necesar, un marcaj sfarsit de continut. Acest tip de marcaj sunt denumite in mod comun **codari TLV** (type-length-value).
De notat ca pe langa protocoalele enumerate mai sus, ASN.1 impreuna cu BER este folosit in majoritatea serviciilor de telefonie celulara pentru transmiterea mesajelor de control peste retea.

Denumirea OID-urilor

Obiectele gestionate sunt asezate intr-o ierarhie sub forma de arbore. Aceasta structura este baza pentru schema de nume a SNMP-ului.
Un identificator de obiect (OID) este dintr-o serie de numere intregi pe baza nodurilor arborelui, si despartite prin punct. Exista si o forma ce poate fi citita usor (asemanator adreselor IP), aceasta fiind o insiruire de nume separate prin punct.In figura 5 este reprezentata structura de arbore pertinenta pentru SNMP.

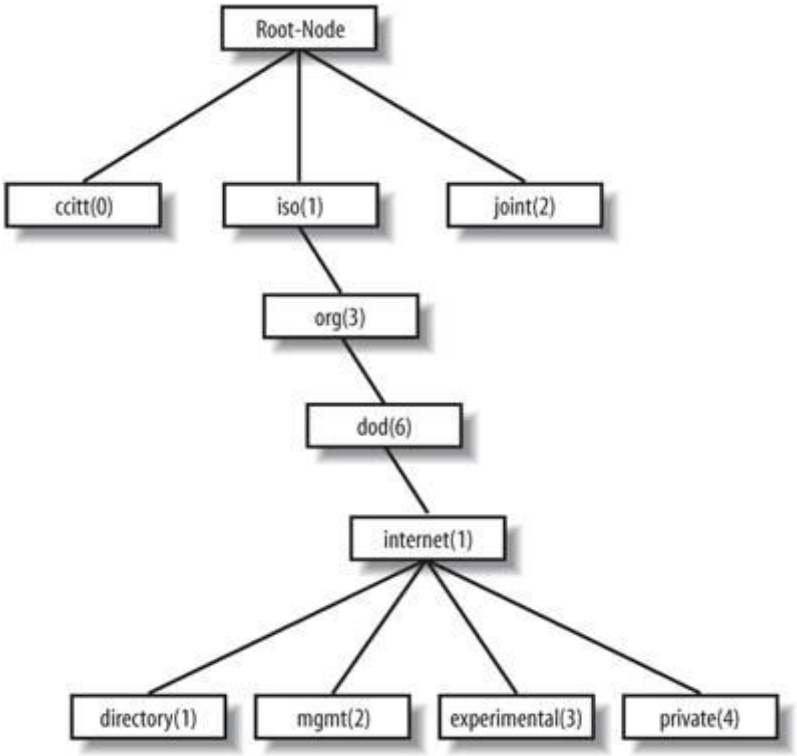

Figure 5 Structura arborescenta a schemei de nume in SNMP

In acest arbore primul nod este denumit **nod-radacina**, orice parte ce contine descendenti este denumita **subarbore** iar extremitatile sunt denumite **frunze**. Exemplu de subarbori sunt ccitt(0), iso(1), si joint(2). Pentru SNMP este interesant subarborele iso(1).org(3).dod(6).internet(1) care are OID-ul 1.3.6.1 (sau iso.org.dod.internet). Reprezentarea numerica este modul in care un obiect gestionat este reprezentat intr-un agent. Ramura **directory** nu este utilizata in present, ramura **mgmt** defineste un set de obiecte Internet standard ce sunt gestionate. Ramura **experimental** este rezervata pentru teste. Obiectel de sub ramura **private** sunt definite de organizatii/vendori sau persone individuale. Iata cum arata definitia subarborelui **internet** si a celorlati patru subarbori:

```
internet      OBJECT IDENTIFIER ::= { iso org(3) dod(6) 1 }
directory     OBJECT IDENTIFIER ::= { internet 1 }
mgmt          OBJECT IDENTIFIER ::= { internet 2 }
experimental  OBJECT IDENTIFIER ::= { internet 3 }
private       OBJECT IDENTIFIER ::= { internet 4 }
```

Prima linie declara **internet** ca OID 1.3.6.1 si este definit ca subarbore al iso.org.dod (1.3.6). Operatorul ::= este operatorul **definitie**. Celelalte ramuri sunt declarate in mod similar ca subarbori pentru **internet**.
Subarborele private contine ramura **enterprises**, utilizata de vendori pentru a defini obiecte private pentru orice tip de hardware sau software. Definitia in SMI este:

```
enterprises   OBJECT IDENTIFIER ::= { private 1 }
```

Asignarea si gestionarea numerelor enterprise pentru fiecare organizatie, vendor, institutie sau persoana individuala este facuta de IANA (Internet Assigned Numbers Authority). De exemplu **numarul enterprise** privat pentru Cisco este 9 ceea ce duce la OID-ul de baza 1.3.6.1.4.1.9

Tipuri de date in SMIv1

SMIv1 defineste cateva tipuri de date care sunt de o importanta fundamentala pentru pentru managementul retelelor. Aceste tipuri definesc ce fel de informatie un obiect gestionat poate tine. Aceste tipuri de date sunt asemantoare cu cele din limbajele de programare, si sunt descrise in continuare:

INTEGER – un numer pe 32 de biti, folosit adesea pentru a specifica tipuri enumerate in contextual unui singur obiect gestionat. De exemplu starea operationala a unei interfete dintr-un router poate fi: up, down sau testing si este reprezentata de valorile 1,2 sau 3. Valoarea 0 nu trebuie folosita, conform specificatiilor din RFC 1155

OCTET STRING – un sir de 0 sau mai multi octeti, folosit pentru a reprezenta texte.

Counter – Un numar pe 32 de biti cu valoarea minima 0 si valoare maxima 2^{32}-1. Cand valoarea maxima este atinsa, incepe din nou de la 0. Este utilizat pentru a tine informatii cum ar fi numarul de octeti trimisi sau primiti pe o interfata, numarul de erori pe o interfata etc. Cand un agent este restartat toate variabilele de tip Counter ar trebui setate la 0.

OBJECT IDENTIFIER – un sir de numere despartite prin punct, care reprezinta obiectul gestionat in arborele obiectelor.

NULL – Nu este utilizat curent in SNMP

SEQUENCE – Defineste o lista ce contine alte tipuri de date ASN.1

SEQUENCE OF – Defineste un obiect gestionat care este alcatuit dintr-o secventa (SEQUENCE) de tipuri ASN.1

IpAddress – Reprezinta o adresa IPv4 pe 32 de biti. Nici SMIv1 si nici SMIv2 nu fac referire la adrese IPv6 pe 128 de biti.

NetworkAddress – La fel ca IpAddress

Gauge – Un numar intreg pe 32 de biti cu valori intre 0 si 2^{32}-1, care spre deosebire de tipul **Counter**, acesta poate si creste si descreste dar nu poate trece de valoarea maxima. Viteza unei interfete este masurata cu acest tip de date.

TimeTicks – Un numar pe 32 de biti cu valoarea minima 0 si valoare maxima 2^{32}-1 care masoara timpul in sutimi de secunda.

Opaque – Permite orice alta codare ASN.1 sa fie cuprinsa intr-un OCTET STRING

Scopul acestor tipuri de obiecte este acela de a defini obiectele gestionate. Putem san e gandim la un MIB ca la o specificatie care defineste obiectele gestionate pe care un vendor le suporta. MIB-urile sunt distribuite in format text si pot fi usor modificate cu un editor de text.
In continuare se va analiza structura MIB-II petru a intelege mai bine cele precizate anterior.

```
RFC1213-MIB DEFINITIONS ::= BEGIN

          IMPORTS
                  mgmt, NetworkAddress, IpAddress, Counter, Gauge,
                  TimeTicks
                          FROM RFC1155-SMI
                  OBJECT-TYPE
                          FROM RFC1212;

          mib-2      OBJECT IDENTIFIER ::= { mgmt 1 }
-- groups in MIB-II

          system       OBJECT IDENTIFIER ::= { mib-2 1 }
          interfaces   OBJECT IDENTIFIER ::= { mib-2 2 }
          at           OBJECT IDENTIFIER ::= { mib-2 3 }
          ip           OBJECT IDENTIFIER ::= { mib-2 4 }
          icmp         OBJECT IDENTIFIER ::= { mib-2 5 }
          tcp          OBJECT IDENTIFIER ::= { mib-2 6 }
          udp          OBJECT IDENTIFIER ::= { mib-2 7 }
          egp          OBJECT IDENTIFIER ::= { mib-2 8 }
          transmission OBJECT IDENTIFIER ::= { mib-2 10 }
          snmp         OBJECT IDENTIFIER ::= { mib-2 11 }
```

Prima linie defineste numele MIB-ului si anume RFC1213-MIB. Sectiunea IMPORTS permite importul tipurilor de date si a OID-urilor din alte fisiere MIB. Acest MIB importa din RFC1155-SMI descris anterior, urmatoarele mgmt, NetworkAddress, IpAddress, Counter, Gauge, TimeTicks iar din RFC1212 (Concise MIB Definition) importa OBJECT-TYPE. OID-urile urmeaza structura ierarhica descrisa anterior, de exemplu grupul **transmission** are OID-ul 1.3.6.1.2.10.

System (OID 1.3.6.1.2.1.1) – defineste o lista de obiecte legate de sistem , cum ar fi numele sistemului, timpul de cand functioneaza sistemul etc.

Interfaces (OID 1.3.6.1.2.1.2) – pastreaza un status al fiecarei interfete intr-un echipament gestionat (este *up/down*, numar de oceti transmisi/primiti, erori etc)

At (OID 1.3.6.1.2.1.3) – address translation, este invechit si folosit doar pentru compatibilitate

Ip – (OID 1.3.6.1.2.1.4) – pastreaza aspecte legate de protocolul IP inclusiv rutarea IP

Icmp – (OID 1.3.6.1.2.1.5) – urmareste lucruri legate de ICMP, cum ar fi erori ICMP, discardari etc.

Tcp – (OID 1.3.6.1.2.1.6) – unul din lucrurile pe care le urmareste este starea conexiunii tcp (closed, listen, synSent etc.)

Udp – (OID 1.3.6.1.2.1.7) – este utilizat pentru statistici UDP

Egp – (OID 1.3.6.1.2.1.8) – pastreaza statistici legate de EGP si tabela vecinilor EGP

Transmission – (OID 1.3.6.1.2.1.10) – sub acest subarbore se gasesc alte MIB-uri care descriu medii de transmisiuni (eex. FDDI, RS232, FrameRelay, ATM, etc.)

Snmp (OID 1.3.6.1.2.1.11) – masoara performante legate de SNMP, numar de pachete SNMP intrate/iesite/discardate etc

```
Dupa definirea OID-urilor urmeaza definirea obiectelor dupa urmatorul
format:


<name> OBJECT-TYPE
    SYNTAX <datatype>
    ACCESS <either read-only, read-write, write-only, or not-
accessible>
    STATUS <either mandatory, optional, or obsolete>
    DESCRIPTION
        "Textual description describing this particular managed
object."
    ::= { <Unique OID that defines this object>

Un exemplu de obiect este ifTable, care reprezinta un tabel al
interfetelor de retea dintr-un echipament:

ifTable OBJECT-TYPE
              SYNTAX  SEQUENCE OF IfEntry
              ACCESS  not-accessible
              STATUS  mandatory
              DESCRIPTION
        "A list of interface entries. The number of entries is given by
the value of ifNumber."
              ::= { interfaces 2 }
```

```
ifEntry OBJECT-TYPE
              SYNTAX  IfEntry
              ACCESS  not-accessible
              STATUS  mandatory
              DESCRIPTION
                  "An interface entry containing objects at the
subnetwork
                   layer and below for a particular interface."
              INDEX   { ifIndex }
              ::= { ifTable 1 }

          IfEntry ::=
              SEQUENCE {
                  ifIndex
                      INTEGER,
                  ifDescr
                      DisplayString,
                  ifType
                      INTEGER,
                  ifMtu
                      INTEGER,
                  ifSpeed
                      Gauge,
                  ifPhysAddress
                      PhysAddress,
                  ifAdminStatus
                      INTEGER,
                  ifOperStatus
                      INTEGER,
                       ...
```

Sintaxa pentru **ifTable** este o secventa **ifEntry**, ceea ce inseamna ca este un tabel continand coloanele definite in in ifEntry. Obiectul este **non-accesible**, ceea ce inseamna ca nu putem interoga un agent pentru aceasta valoare. Statutul este **mandatory** ceea ce inseamna ca agentul trebuie sa implementeze acest obiect pentru a se conforma specificatiilor MIB-II.
Numele secventei **IfEntry** incepe cu litera mare spre deosebire de numele obiectului **ifEntry**. Aceasta este modalitatea prin care se defineste o secventa. O secventa este o lista de obiecte coloana si tipurile de date SMI associate. **Index-ul** este o cheie unica utilizata la definirea unui singur rand in ifTable. Unicitatea este asigurata de catre agent. Index-ul pentru IfEntry este definit astfel:

```
ifIndex OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "A unique value for each interface. Its value ranges between
         1 and the value of ifNumber. The value for each interface
         must remain constant at least from one reinitialization of the
         entity's network management system to the next
reinitialization."
       ::= { ifEntry 1 }
```

Extensii ale SMI in SMIv2

SMIv2 extinde arborele SMI de obiecte prin adaugarea ramurii **snmpV2** la subarborele **internet** (vezi figura 6).

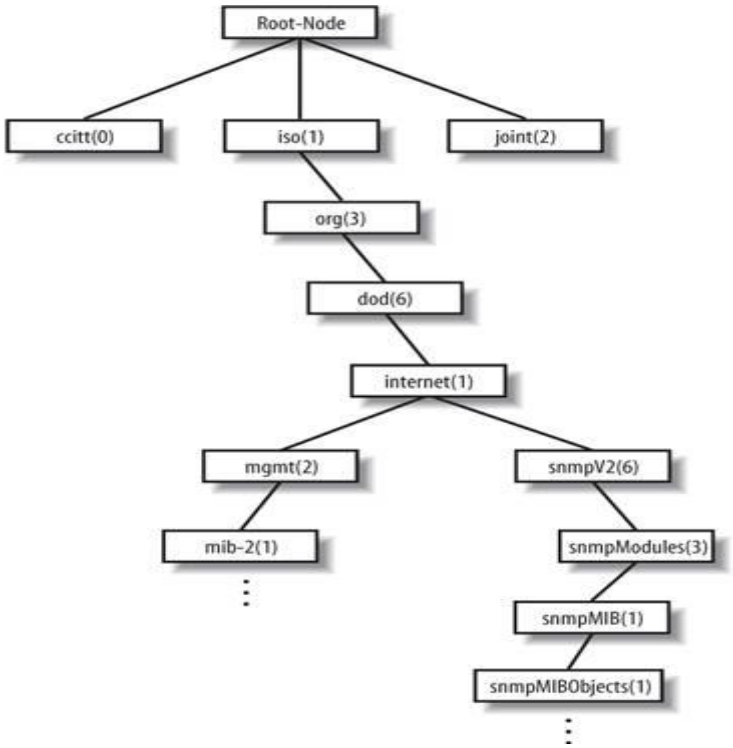

Figure 6 Extinderea structurii arborescente in SMIv2

SMIv2 adauga noi tipuri de date:

Integer32 – la fel ca INTEGER
Counter32 – identic cu Counter
Gauge32 – identic cu Gauge
Unsigned32 – reprezinta valori in inetrvalul 0 2^{32}-1
Counter64 - similar cu Counter32, dar valoarea maxima este 2^{64}-1
BITS – este un tip de date ce permite definirea de nume pentru campuri individuale de biti intr-o valoare OCTET-STRING

Definitia unui obiect in SMIv2 s-a schimbat fata de prima versiune. Exista acum campuri optionale care permit un control sporit asupra modului in care un obiect este accesat, o mai buna descriere etc. Noua sintaxa pentru definirea obiectelor este urmatoarea:

<name> OBJECT-TYPE
  SYNTAX <datatype>
  UnitsParts <Optional>
  MAX-ACCESS

STATUS
DESCRIPTION
"Textual description describing this particular managed object."
AUGMENTS { <name of table> }
::= { <Unique OID that defines this object> }

UnitsParts – o descriere textuala a unitatilor utilizate sa reprezinte obiectul (secunde, milisecunde etc.)

MAX-ACCESS – in SNMPv2 tipul de acces al obiectului este MAX-ACCESS, ale carui valori pot fi **read-only, read-write, read-create, not-accesible** si **accessible-for-notify**

STATUS – aceasta clauza a fost extinsa sa permita urmatoarele cuvinte cheie: **current, obsolete** si **deprecated.** In SNMPv2 **current** este echivalent cu **mandatory** in SNMPv1.

AUGMENTS – aceasta clauza permite extindera unui tabel prin adaugarea unor coloane reprezentate de alte obiecte.

Este util atunci cand se construiesc MIB-urile sa se defineasca noi tipuri similare celor definite in SMI. Spre deosebire de tipurile definite in SMI, fiecare din aceste tipuri are un nume diferit, o sintaxa similara, dar o semantica mai precisa. Noile tipuri se numesc **conventii textuale** si sunt folosite pentru usurarea citirii MIB-urilor.
In SMIv2, cateva **conventii textuale** importante sunt:

DisplayString – un sir de caractere ASCII NVT (Network Virtual Terminal)

PhysAddress – adresa fizica (sau adresa MAC) reprezentata ca OCTET STRING

TruthValue – defineste valorile booleene **true (adevar)** si **false(fals)**

VariablePointer – este un pointer la instanta unui obiect

TimeStamp – masoara timpul scurs intre timpul cand un echipament este functional si momentul producerii unui eveniment

DateAndTime – reprezinta informatia de data sit imp

StorageType – denota tipul de memorie pe care un agent il utilizeaza, si poate avea valorile: other(1), volatile(2), nonVolatile(3), permanent(4) si readOnly(5)

Operatii SNMP

Sistemele de gestionare a retelelor (NMS) si agentii SNMP, schimba informatii SNMP prin intermediul unor operatii. Fiecare din operatiile SNMP are o datagrama (PDU) standard. Aceste operatii sunt:

- get
- getnext
- getbulk (SNMPv2 si SNMPv3)
- set
- getresponse
- trap
- notification (SNMPv2 si SNMPv3)
- inform (SNMPv2 si SNMPv3)
- report (SNMPv2 si SNMPv3)

Aceste operatii sunt implementate de majoritatea NMS-urilor sau pot fi gasite sub forma de pachet de tool-uri in linie de comanda. Un astfel de pachet este **NetSNMP Tools** care este un proiect *open source*, si care ofera un set de programe in linie de comanda ce permit executarea operatiilor mai sus mentionate.

Operatia *get*

Operatia *get* este initiate de catre NMS care trimite cererea catre agent. Agentul primeste cererea si o proceseaza cat mai bine posibil. Dupa ce agentul aduna informatiile cerute de NMS, trimite catre acesta un getresponse. Sunt situatii (de exemplu in cazul routerelor care au un processor foarte incarcat) in care cererea nu poate fi procesata si este discardata.

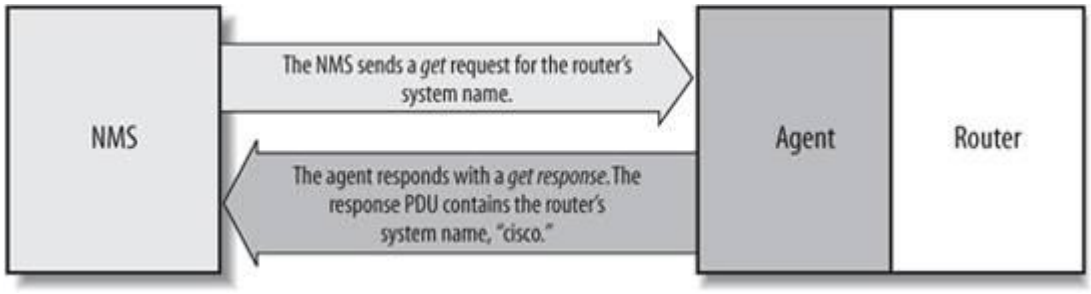

Figure 7 Exemplu al unei operatii GET

Intrebarea care apare este de unde stie agentul ce informatii are nevoie NMS-ul? Unul din parametrii unei operatii GET este o legatura de variabila (variable binding sau pe scurt varbind). **Varbind** este o lista de obiecte ale unui MIB ce permite agentului sa stie ce infomatii doreste NMS-ul. **Varbinds** pot fi privite ca perechi OID=valoare.

Un exemplu de operatie GET folosind programul *snmpget* din pachetul NetSNMP arata astfel:

```
C:\>snmpget -v 1 -c secret 86.104.xxx.xxx 1.3.6.1.2.1.1.1.0

SNMPv2-MIB::sysDescr.0 = STRING: Hardware: x86 Family 6 Model 9
Stepping 5 AT/AT COMPATIBL
E - Software: Windows 2000 Version 5.1 (Build 2600 Uniprocessor Free)
```

Parametrii acestei comenzi sunt: v – versiunea protocolului SNMP, c – comunitatea, adresa IP a hostului pe care se gaseste agentul SNMp si de la care dorim sa aflam infomatii si OID-ul care de fapt este informatia pe care o dorim. Daca mergem pe arborele SNMP, observam ca OID-ul **1.3.6.1.2.1.1.1** corespunde ...mgmt.mib-2.system.sysDescr. Ceea ce pare ciudat este 0-ul de la sfarsitul OID-ului. In SNMP, obiectele MIB sunt definite prin conventia x.y unde x este OID-ul propriu-zis iar y este un identificator de instanta. Pentru obiecte scalare, y este totdeauna 0. Pentru obiecte tabel, identificatorul de instanta permite selectarea unui rand din tabel (de exemplu y=1 selecteaza randul unu al tabelului).

Operatia GETNEXT

Operatia GETNEXT permite trimiterea unor secvente de comenzi pentru a primi un grup de valori dintr-un MIB. Comanda GETNEXT traverseaza un subarbore in ordine lexicografica (vezi figura 8). Deoarece OID-ul este o secventa de intregi, este usor pentru un agent sa inceapa de la nodul radacina si sa parcurga arborele pana cand ajunge la informatia dorita (cautare depth-first). In momentul in care NMS-ul primeste informatia de la agent emite o alta comanda GETNEXT. Aceasta operatie este repetata pana cand se primeste o eroare, ceea ce semnifica ca s-a ajuns la sfarsitul MIB-ului.

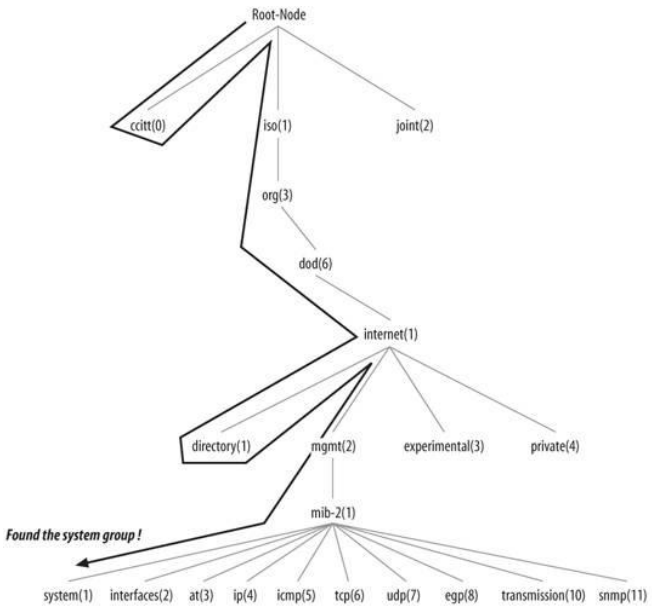

Figure 8 Parcurgerea arborelui SNMP in operatia GETNEXT

Rezultatul partial al unei operatii GETNEXT, obtinut cu comanda *snmpwalk* din pachetul NetSNMP este ilustrat mai jos

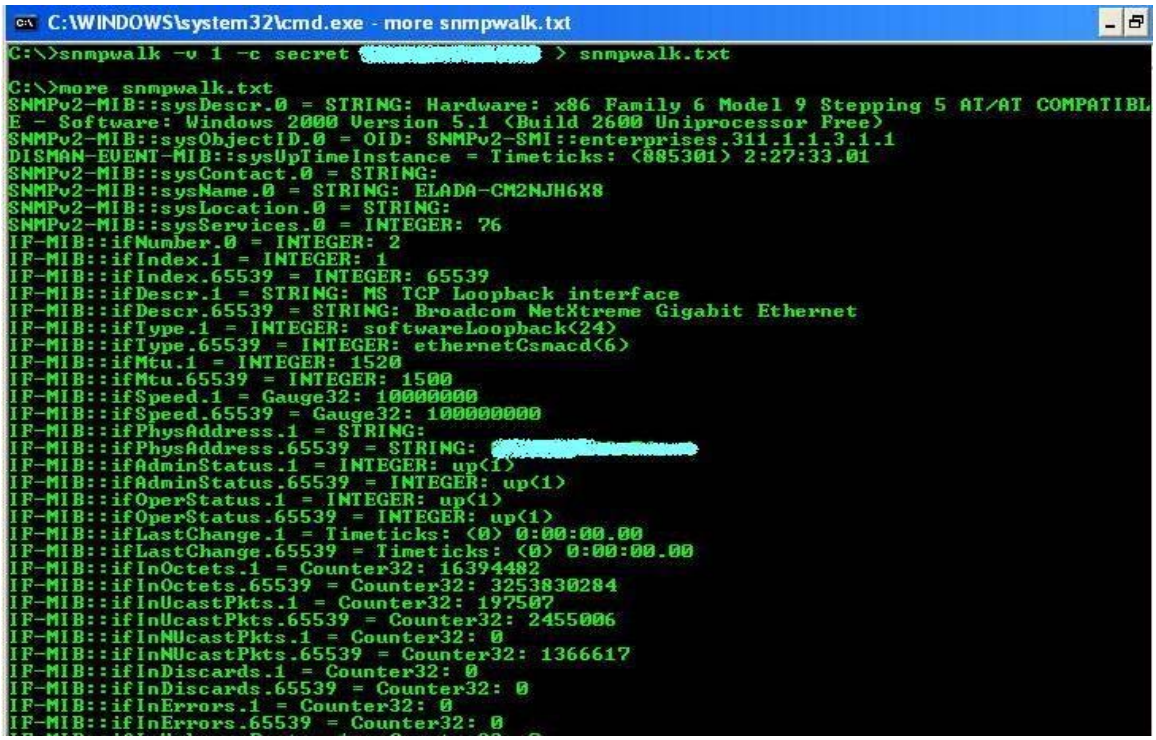

Figure 9 Rezultatul unei operatii GETNEXT obtinuta prin comanda snmpwalk

Operatia GETBULK

Aceasta operatie este definita in SNMPv2, si permite unei aplicatii de management sa primeasca o sectiune mare a unui tabel intr-o singura operatie. Si operatia GET poate cere mai multe obiecte odata dar marimea mesajele de raspuns sunt limitate de capacitatile agentului. Operatia GETBULK spune agentului sa trimita cat de mult poate, raspunsuri incomplete fiind astfel posibile. Operatie GETBULK primeste 2 parametri, *nonrepeaters* – spune ca primele N obiecte pot sa fie primite cu o operatie *getnext* si max-repeaters – numarul maxim de repetari ale operatiunii *genext* pentru a primi restul obiectelor.
In pachetul NetSNMP Tools, operatia este implementata de comanda snmpgetbulk in care se specifica versiunea de SNMP, parametrii mentionati anterior si obiectele dorite:

```
C:\>snmpbulkget -v2c -c secret -Cn1 -Cr3 86.104.152.159 sysDescr
ifInOctets ifOutOctets
SNMPv2-MIB::sysDescr.0 = STRING: Hardware: x86 Family 6 Model 9
Stepping 5 AT/AT COMPATIBL E - Software: Windows 2000 Version 5.1
(Build 2600 Uniprocessor Free)
IF-MIB::ifInOctets.1 = Counter32: 17470828
IF-MIB::ifOutOctets.1 = Counter32: 17470828
IF-MIB::ifInOctets.65539 = Counter32: 149404043
IF-MIB::ifOutOctets.65539 = Counter32: 1386103169
IF-MIB::ifInUcastPkts.1 = Counter32: 207056
IF-MIB::ifOutUcastPkts.1 = Counter32: 207056
```

Operatia SET

Operatia SET este folosita pentru a schimba valorile obiectelor gestionate sau pentru a crea un nou rand intr-un tabel. Este posibil ca mai multe obiecte sa fie setate in acelasi timp. In pachetul NetSNMP Tools, operatia SET este implementata prin comanda *snmpset*, asa cum se vede din exemplul de mai jos

```
C:\usr\bin>snmpget -v 2c -c secret 86.104.xxx.xxx 1.3.6.1.2.1.1.6.0
SNMPv2-MIB::sysLocation.0 = STRING:

C:\usr\bin>snmpset -v 2c -c secret 86.104.xxx.xxx 1.3.6.1.2.1.1.6.0 s
"Undeva in Bucuresti!"
SNMPv2-MIB::sysLocation.0 = STRING: Undeva in Bucuresti!

C:\usr\bin>snmpget -v 2c -c secret 86.104.152.159 1.3.6.1.2.1.1.6.0
SNMPv2-MIB::sysLocation.0 = STRING: Undeva in Bucuresti!
```

Cu ajutorul operatiei SET se pot face operatii cum ar fi reload-ul unui router, inchiderea unei interfete etc., de aceea trebuie avut grija cand se foloseste.

Operatia NOTIFICATION

Pentru a standardiza formatul datagramelor din SNMPv1 (PDU-urile pentru GET si SET au un format diferit), SNMPv2 defineste **NOTIFICATION-TYPE**, care are formatul datagramei identic atat pentru GET cat si pentru SET. RFC 2863 (The Interface Group MIB) defineste tipul generic notificare pentru evenimentul *linkDown*, astfel:

```
linkDown NOTIFICATION-TYPE
    OBJECTS { ifIndex, ifAdminStatus, ifOperStatus }
    STATUS  current
    DESCRIPTION
            "A linkDown trap signifies that the SNMP entity, acting in
            an agent role, has detected that the ifOperStatus object
      for one of its communication links is about to enter the down
            state from some other state (but not from the notPresent
            state).  This other state is indicated by the included
value of ifOperStatus."
    ::= { snmpTraps 3 }
```

Operatiile SNMP INFORM si SNMP REPORT

SNMPv2 ofera un mechanism, numit **inform**, care permite confirmarea primirii trapurilor. Cand este trimis un **inform** (un trap cu o cerere de confirmare), NMS-ul trimite confirmarea catre agent.
SNMP **report** este parte a SNMPv3 si intentia sa este de a permite motoarelor SNMP sa comunice intre ele.

SNMPv3

Desi singurele modificari majore aduse de SNMPv3 sunt in legatura cu securitatea protocolului, introducerea de noi conventii textuale, concepte si terminologii fac sa para ca au fost facute schimbari majore. In SNMPv3 notiunile de agent si manager nu mai sunt folosite ci au fost inlocuite de notiunea comuna de **entitate**, fiecare entitate fiind alcatuita dintr-un motor SNMP (*snmp engine*) si aplicatii SNMP.

RFC-urile care compun SNMPv3 sunt urmatoarele: RFC 3411 Architecture for SNMP Frameworks, RFC 3412 Message Processing and Dispatching, RFC 3413 SNMP Applications, RFC 3414 User-based Security Model (USM), RFC 3415 View-based Access Control Model (VACM), RFC 3416 Protocol Operations for SNMPv2, RFC 3417 Transport Mappings for SNMPv2, RFC 3418 MIB for SNMPv2, RFC 2576 Coexistence Between SNMP Versions, RFC 2570 Introduction to SNMPv3, RFC 2786 Diffie-Hellman USM Key Management.

Motorul SNMPv3

Motorul SNMP este compus din 4 parti: **Dispatcher-ul, Message Processing Subsystem , Security Subsystem, Access Control Subsystem**.
Rolul dispatcher-ului este de a trimite si a primi mesajele. Determina versiunea fiecarui mesaj primit, si apoi il trimite catre subsistemul de prelucrare a mesajelor.
Subsistemul de prelucrare a mesajelor pregateste mesajele sa fie trimise si extrage datele din mesajele receptionate si poate contine module pentru fiecare din cele 3 versiuni snmp in vederea prelucrarii cererilor.
Subsistemul de securitate ofera servicii de autentificare si criptare. Autentificarea este una bazata pe utilizator si foloseste algoritmii MD5 (Message-Digest 5) sau SHA (Secure Hash Algorithm) pentru a nu trimite parolele in clar. Mesajele SNMP sunt criptate/decriptate folosind algoritmul DES (Data Encryption Standard).
Subsistemul de control al accesului este utilizat pentru a gestiona accesul la obiectele din MIB-uri.

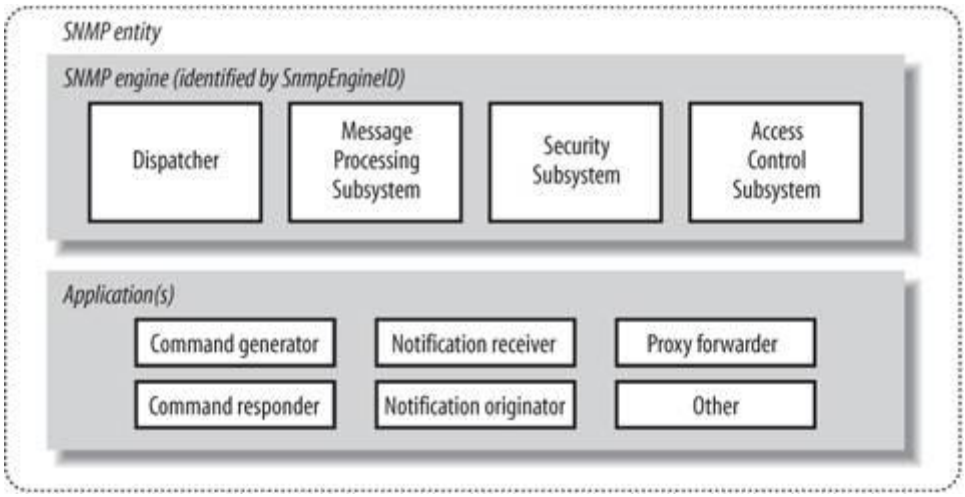

Figure 10 Structura unei entitati in SNMPv3

Aplicatii SNMPv3

Command generator – genereaza comenzile *get, getnext, getbulk* si *set*. Aceasta aplicatie este implementata de un NMS.

Command responder – raspunde la comenzile comenzile *get, getnext, getbulk* si *set*. Este implementata de o entitate pe echipamentele gestionate (sau de un agent SNMP pentru versiunile 1 si 2).

Notification originator – genereaza trap-uri si notificari SNMP. Este implementata de o entitate existenta in echipamentul gestionat.

Notification receiver – primeste trap-uri si notificari si este implementat de NMS.

Proxy forwarder – permite pasarea mesajelor intre entitati

Prin RFC 3411, SNMPv3 permite definirea unor aplicatii aditionale, ceea ce reprezinta un avantaj fata de versiunile anterioare.

Cateva conventii textuale importante care au aparut in versiunea a treia sunt:

snmpEngineID – un identificator unic pentru un motor SNMP.

snmpSecurityModel – modelul de securitate utilizat; poate avea valorile SNMPv1, SNMPv2 sau USM (User-Based Security Model).

snmpSecurityLevel – nivelul de securitate la care mesajele pot fi trimise sau nivelul de securitate cu care operatiile sunt procesate. Poate avea valorile noAuthNoPriv (fara autentificare si fara criptare), authNoPriv (cu autentificare dar fara criptare), authPriv (cu autentificare si criptare).

snmpTagValue – un sir de octeti ce contine o eticheta, ce poate avea valori ca *router, host, etc.*

Pachete de dezvoltare a aplicatiilor SNMP

Pentru dezvoltarea aplicatiilor de management care se bazeaza pe protocolul SNMP, sunt disponibile o serie de pachete de dezvoltare dintre care cele mai cunoscute sunt NetSNMP Toolkit, HP OpenView SNMP API si WinSNMP API. Fiecare dintre aceste pachete ofera pe langa aplicatiile ce implementeaza operatiile obisnuite din SNMP (get, getnext, walk etc), agenti SNMP capabili sa interpreteze cererile SNMP si o serie de API-uri ce pot fi folosite pentru scrierea propriilor agenti si dezvoltarea de aplicatii de management. Pachetele NetSNMP si HP OpenView SNMP API permit dezvoltarea aplicatiilor atat pe platforme Unix/Linux cat si Windows iar pachetul WinSNMP API este exclusiv pentru platforme Windows. In continuare se vor prezenta doar primele doua pachete precum,

NetSNMP Toolkit

Pachetul NetSNMP este un set de instrumente si librarii (API-uri) pentru construirea aplicatiilor de *"network and systems management"* si care are distribuit "*free"*. Atat sursele cat si fisierele executabile sunt disponibile pentru un numar apreciabil de platforme: Solaris, HPUX, AIX, Irix, OsX, Linux, Win32.
Setul de aplicatii NetSNMP implementeaza SNMPv1, SNMPv2c si SNMPv3 utilizand atat IPv4 cat si IPv6 si include:

- aplicatii in linie de comanda pentru:
  - captarea informatiilor de la dispozitive capabile SNMP, fie folosind cereri simple (**snmpget, snmpgetnext**) sau cereri multiple (**snmpwalk, snmptable**)
  - manipularea informatiilor de configurare pe dispozitive capabile SNMP (**snmpset)**
  - captarea unor colectii de informatii: **snmpdf** (echivalentul comenzii **df** din Unix, afiseaza organizarea spatiului pe dscuri), **snmpnetstat** (afiseaza informatii legate de retea penru un host), **snmpstatus** (afiseaza informatii ca adresa IP a hostului, timpul de cand functioneaza, numarul de pachete primite/trimise pe o interfata etc)
  - conversia intre valorile numerice si textuale ale OID-urlor (**snmptranslate**)
- o aplicatie grafica pentru navigarea in MIB-uri (**tkmib**)
- o aplicatie pentru primirea notificarilor SNMP **(snmptrapd**); aceste notificari pot fi filtrate si inregistrate in fisiere log, pot fi pasate catre un NMS superior sau catre o alta apliatie
- un agent ce raspunde la interogari SNMP (**snmpd**), ce include suport pentru un numar mare de MIB-uri oferind in acelasi timp posibilitatea extinderii acestuia
- o librarie pentru dezvoltarea de aplicatii SNMP folosind atat API-uri C cat si Perl

NetSNMP Perl API

Modulul *Net::SNMP* ofera o interfata SNMP obiect orientata de nivel inalt. O aplicatie Perl poate fi folosita sa receptioneze informatii de la echipamente ce suporta protocolul SNMP sau sa seteze parametri pe acestea, folosind toate cele trei versiuni ale protocolului. Fiecare obiect *Net::SNMP* ofera o mapare unu la unu intre un obiect Perl si un agent sau manager SNMP. Odata ce obiectul a fost creat acesta poate utilizat pentru a realiza operatiunile permise de SNMP.
Un obiect *Net:SNMP* poate fi creat astfel incat sa aiba proprietatile *blocant* sau *neblocant* Implicit metodele utilizate sa trimita mesaje SNMP nu se intorc in programul apelant pana cand schimbul de datagrame a avut loc cu succes sau pana cand a trecut o perioada de timp acceptata. Acest comportament da obiectului starea *blocata* deoarece executia codului este intrerupta pana cand metoda apelata intoarce un rezultat.
Argumentul *nonblocking* cu o valoare *true* poate fi pasat constructorului obiectului SNMP pentru ai da acestuia un comportament neblocant. O metoda apelata de un obiect neblocant aseaza intr-o coada mesajele SNMP si se intoarce imediat in programul appellant permitand continuarea executiei codului. Mesajele din coada nu sunt trimise pana cand nu se intra intr-o bucla de evenimente prin apelarea metodei *snmp_dispatcher()*. Cand mesajele sunt trimise, orice raspuns la mesaje invoca subrutina definita de utilizator (subrutina callback) atunci cand mesajul a fost pus in coada de asteptare. Din aceasta bucla se iese atunci cand toate mesajele au fost scoase din coada de asteptare

Obiecte blocante

Starea blocanta este starea implicita a obiectelor Net:::Snmp. Pentru obiectele care initiaza o sesiune SNMP cerand un raspuns (interogare SNMP), se returneaza o referinta la o structura *hash* continand rezultatul interogarii.Daca o eroare a intervenit, valoarea intoarsa de metoda este “undefined”. Pentru a determina cauza erorii metoda *error()* poate fi utilizata. Structura *hash* este construita din lista de variabile *VarBindList* continuta in mesajul de raspuns, folosind perechi O*bjectName-ObjectSyntax d*in aceasta lista. Valorile cheilor *hash*-ului constau din OID-uri in format numeric corespunzatoare fiecarui O*bjectName* din lista. Valorile fiecarei intrari a *hash-ului* este setata egala cu valoarea ObjectSyntax corespunzatoare din *VarBindList.* Referinta la aceasta structura *hash* poate fi obtinuta si folosind metoda *var_bind_list()*.

Obiecte neblocante

Cand un obiect cu comportament *neblocant* este creat, invocarea unei metode asociate obiectului intoarce imediat fie valoare *adevar* fie valoarea *undefined* in cazul unei erori. Continutul listei de variabile *VarBindList* continuta in mesajul raspuns poate fi receptionata tot cu un apel catre metoda *var_bind_list()*.

Apelul metodelor

Apelul metodelor cu argumente ce au valori text se face conform sintaxei:

```
$object->method(-argument => $value);
```

Cand obiectele sunt create cu comportament *neblocant*, sunt necesare argumente suplimentare in apelurile metodelor obiectului dintre care cel mai important este **-callback**: acest argument asteapta o referinta catre o subrutina sau un vector al carui prim element este o referinta catre o subrutina. Aceasta subrutina este executata atunci cand se primeste un mesaj SNMP de raspuns, cand o eroare a intervenit sau cand numarul de reincercari a fost deposit.

Cea mai importanta metoda este constructorul obiectelor Net::SNMP, **session()**, cu ajutorul caruia sunt create obiectele. Sintaxa acestei metode este urmatoarea:

```
($session, $error) = Net::SNMP->session(
                        [-hostname      => $hostname,]
                        [-port          => $port,]
                        [-localaddr     => $localaddr,]
                        [-localport     => $localport,]
                        [-nonblocking   => $boolean,]
                        [-version       => $version,]
                        [-domain        => $domain,]
                        [-timeout       => $seconds,]
                        [-retries       => $count,]
                        [-maxmsgsize    => $octets,]
                        [-translate     => $translate,]
                        [-debug         => $bitmask,]
                        [-community     => $community,]   # v1/v2c
                        [-username      => $username,]    # v3
                        [-authkey       => $authkey,]     # v3
                        [-authpassword  => $authpasswd,]  # v3
                        [-authprotocol  => $authproto,]   # v3
                        [-privkey       => $privkey,]     # v3
                        [-privpassword  => $privpasswd,]  # v3
                        [-privprotocol  => $privproto,]   # v3
                      );
```

Intr-un context de scalar este returnata o referinta catre obiectul creat daca nu au fost erori iar intr-un context de lista este returnata o referinta la obiectul creat si un mesaj de eroare gol daca nu au fost erori sau este returnata valoarea “undefined” si mesajul erorii in cazul in care au aparut probleme la crearea obiectului.
In mod implicit modulul Net::SNMP foloseste ca protocol de transport UDP/IPv4. Pe langa acesta se mai poate folosi si UDP/IPv6, TCP/IPv4 si TCP/IPv6, acestea setandu-se prin intermediul atributului **–domain**.
Cu ajutorul argumentului –**hostname** se seteaza adresa echipamentului destinatie (implicit are valoarea *localhost*)
Argumentul **–port** contine numarul portului adresei destinatie, implicit fiind 161

Implicit adresa sursa si portul sursa sunt asignate dinamic de masina pe care ruleaza modulul Net::SNMP, acestea putand fi schimbate prin intermediul argumentelor **localaddr** si **–localport.**

Versiunea protocolului SNMP este specificata de argumentul **–version**, care poate lua atat valori numerice (1, 2 sau 3) cat si textuale (snmpv1, snmpv2c, snmpv3). Acest argument, in functie de valoarea sa face ca anumite attribute sa fie obligatoriu de definit. In cazul versiunilor unu si doi, modelul de securitate este bazat pe comunitati si singurul argument ce mai trebuie setat este **–community**. Daca versiunea protocolului este 3 atunci mecanismul USM (User-based Security Model) este utilizat. In functie de nivelul de securitate (implicit *NoAuthNoPriv* care cere doar ca argumentul **–username** sa fie setat) se seteaza alte argumente: **-authkey** si **–authpassword** pentru nivelul *AuthNoPriv* si **–privkey**, **-privpassword** pentru nivelul *AuthPriv*. Doi algoritmi de hashing pot fi folositi pentru autentificare: MD5 si SHA ce se seteaza cu ajutorul argumentului **–authprotocol**, iar pentru criptare se foloseste doar algoritmul DES.

Alte doua metode importante sunt **get_request()** si **set_request()** care au aceeasi sintaxa:

```
$result = $session->get_request(
                      [-callback        => sub {},]     # non-blocking
                      [-delay           => $seconds,]   # non-blocking
                      [-contextengineid => $engine_id,] # v3
                      [-contextname     => $name,]      # v3
                       -varbindlist      => \@oids,
                   );
```

Aceste doua metode realizeaza operatiile de *get* si *set* Mesajul SNMP este construit folosind lista de OID-uri in format zecimal pasat metodei ca o referinta la un vector folosind argumentul **–varbindlist**. Fiecare OID este palsat intr-o singura datagrama *GetRequest* in aceeasi ordine ca a vectorului ce contine OID-urile. Un **context snmp** este o colectie de informatii de management la care o entitate snmp are acces. Parametrii **–contextengineid** si –**contextname** identifica in mod unic un conext snmp.

Metoda **trap()** este folosita pentru a trimite trap-uri SNMPv1 catre un manager:

```
$result = $session->trap(
                      [-delay           => $seconds,]   # non-blocking
                      [-enterprise      => $oid,]
                      [-agentaddr       => $ipaddress,]
                      [-generictrap     => $generic,]
                      [-specifictrap    => $specific,]
                      [-timestamp       => $timeticks,]
                      -varbindlist      => \@oid_value,
                    );
```

Valoare atributului **–enterprise** este un OID in format zecimal, valoarea implicita fiind 1.3.6.1.4 (iso.org.dod.internet.private.enterprises)
Adresa IP a interfetei de pe care se transmite trap-ul este data de argumentul **–agentaddr**, in mod implicit este adresa routei *default* (0.0.0.0).

Legarea variabilelor este setata prin **–varbindlist** care este o referinta la un *array* ce contine grupuri de trei elemente: OID-ul in format zecimal, tipul obiectului (din cele definite in ASN.1) si valoarea obiectului.
Pentru a trimite trap-uri cand se folosesc obiecte SNMP cu versiunile 2c si 3, se foloseste metoda **snmpv2_trap()**:

```
$result = $session->snmpv2_trap(
                      [-delay          => $seconds,]   # non-blocking
                       -varbindlist    => \@oid_value,
                      );
```

Un exemplu de lista de variabile legate este dat mai jos:

```
sysUpTime.0 - ('1.3.6.1.2.1.1.3.0', TIMETICKS, $timeticks)
snmpTrapOID.0 - ('1.3.6.1.6.3.1.1.4.1.0', OBJECT_IDENTIFIER, $oid)
```

In continuare este listingul unui mic script Perl care ilustreaza cele discutate anterior, comentariile fiind sugestive si suficiente pentru intelegerea lui.

```
#!/usr/local/bin/perl

# Acest script demonstreaza cum poate fi folosit NetSNMP Perl API
pentru culegerea informatiilor de management
# Echipamentul interogat este un UPS APC

use Net::SNMP;

#  Incarca toate MIB-urile disponibile in calea C:\usr\share\snmp\mibs
(sau /usr/share/snmp/mibs daca e in Unix)
# Pentru UPS-urile APC, MIB-ul folosit este powernet361.mib
$ENV{'MIBS'}="ALL";

# Definesc variabilele necesare conectarii
$AGENT = "10.55.101.102";
$COMMUNITY = "public";
$PORT = "161";

# Folosesc metoda session() pentru crearea unui obiect Net::SNMP
my ($session, $error) = Net::SNMP->session(
      -hostname  => $AGENT,
      -community => $COMMUNITY,
      -port      => $PORT
   );

# Definesc array-ul cu OID-urile a caror  valori le doresc
@APC_VarList = (['1.3.6.1.4.1.318.1.1.1.1.1.1.0'],  #0      Modelul UPS
                ['1.3.6.1.4.1.318.1.1.1.2.2.1.0'],  #1      Capacitatea
                ['1.3.6.1.4.1.318.1.1.1.2.2.2.0'],  #2      Temperatura
                ['1.3.6.1.4.1.318.1.1.1.4.2.3.0'],  #3      Incarcarea
                ['1.3.6.1.4.1.318.1.1.1.4.2.1.0'],  #4      Tensiunea
                ['1.3.6.1.4.1.318.1.1.1.4.2.2.0'],  #5      Frecventa l
                ['1.3.6.1.4.1.318.1.1.1.4.1.1.0']); #6      Starea

# Folosesc metoda get_next_request() pentru a interoga agentul SNMP;
argumentul este referinta la array-ul ce contine OID-urile necesare

@APC_INFO = $session->get_next_request(
```

```
                        -varbindlist      => \@APC_VarList,
                    );

# Termin sesiunea de SNMP
$session->close;

# Elimin ghilimelele din numele UPS-ului
$APC_INFO[0] =~ s/\"//g;

# Afisez rezultatele ; folosesc constructie de tipul "here document"
# (print <<END Text to display END)
print <<END;
APC SNMP Agent: ${SNMP_TARGET}
Model: ${APC_INFO[0]}
Capacitate baterie: ${APC_INFO[1]}
Temperatura : ${APC_INFO[2]}

Stare UPS: ${APC_INFO[6]}
Incarcare: ${APC_INFO[3]}

Paramatri: ${APC_INFO[5]}VAC @ ${APC_INFO[6]}Hz
END
```

Iesirea acestui program este urmatoarea:

```
APC SNMP Agent: 10.55.101.102
Model: Silcon DP380E          "The UPS model name (e.g. 'APC Smart-UPS
600')."
Capacitate baterie: 100       "The remaining battery capacity expressed
in percent of full capacity."
Temperatura : 19        "The current internal UPS temperature expressed
in Celsius."

Stare UPS: onLine       "The current state of the UPS.  If the UPS is
unable to determine the state of the UPS this variable is set to
unknown(1)."
Incarcare: 13                 "The current UPS load expressed in
percent of rated capacity."

Parametri: 229VAC @ 50Hz      "The output voltage of the UPS system in
VAC." "The current output frequency of the UPS system in Hz."
```

In dreptul fiecarei variabile am adaugat descrierea luata din MIB pentru a intelege valorile returnate.
Acest script foloseste un obiect NetSNMP blocant, ce asteapta un raspuns din partea agentului SNMP inainte de a trece mai departe cu interpretarea codului.

Acest script este unul foarte simplu, insa scopul este doar de a ilustra modul in care se pot obtine informatii de management folosind NetSNMP Perl API. Desigur, se pot dezvolta aplicatii mult mai complexe in care sa se culeaga o multime de informatii de la echipamente, introducerea acestora intr-o baza de date, trasarea de grafice pe baza informatiilor captate, declansarea unor actiuni cum ar fi trimiterea de email-uri atunci cand se depaseste un anumit prag pentru un parametru, captarea si afisarea alarmelor in timp real, pe scurt cam tot ceea ce face un Network Management System comercial.

OVSNMP API

Sistemul de *network management* oferit de HP, HP OpenView Network Node Manager, este unul dintre cele mai competitive NMS-uri de pe piata la ora actuala avand peste 100.000 de copii vandute. Acesta ofera posibilitati de gestionare pentru retele teoretic oricat de complexe, descoperirea automata a topologiei retelei (nivelurile 2 si 3), corelarea evenimentelor din retea etc. HP OV Network Node Manager ofera un set de instrumente de dezvoltare (NNM SDK) ce contine API-uri pentru :

- dezvoltare aplicatii de management pe baza SNMP
- OpenView tracing si logging
- Controlul proceselor OpenView
- Dezvoltarea aplicatiilor OpenView pentru Windows

OpenView SNMP API suporta atat SNMPv1 cat si SNMPv2c printr-o interfata comuna. Mecanismul de transport suportat este UDP/IP, insa versiunea Windows ofera suport si pentru UDP/IPX.
Suportul pentru construirea agentilor SNMP este limitat

Arhitectura OVSNMP API

Subsistemul *Event Handler* se ocupa de pasarea mesajelor intre aplicatii HP OpenView. Motorul ECS (Event Correlation Services) este parte integrala din subsistemul de evenimente, si integreaza logica de corelare a evenimentelor, afectand evenimentele din subsistemul de evenimente disponibile spre subscriere. API-urile OVSNMP si WinSNMP pot partaja notificarile primite pe portul standard SNMP, permitand ca aplicatii dezvoltate folosind cele doua API-uri sa coexiste; totusi evenimentele OpenView vor fi livrate doar aplicatiilor OVSNMP.

OVSNMP API ofera suport special pentru aplicatii care comunica cu un agent prin intermediul uni proxy. O astfel de aplicatie poate adresa un mesaj direct agentului, API-ul OVSNMP recunoaste ca agentul are un proxy si trimite mesajul catre acel proxy care comunica mai departe cu agentul. Informatia utilizata pentru a recunoaste agenti proxy este stocata in fisiere de configurare care trebuiesc actualizate de catre administratori. Aplicatiile dezvoltate pot avea in vedere doar agenti SNMPv1, doar agenti SNMPv2 sau pot fi generice prin folosirea unei interfete comune intre cele doua versiuni de protocoale care face translatii ale operatiilor in functie de versiunea gentului folosit. Totusi trebuie avut in vedere ca mesajele de eroare nu au mapare unu la unu in cele doua versiuni si de accea trebuie tratate separat. Implicit apelul unei functii se face folosind stilul SNMPv1. Pentru a utiliza API-urile OVSNMP in stilul SNMPv2 intai se creaza o structura de sesiune cu ajutorul functiei OVsnmpOpen(). In structura OVsnmpSession se seteaza bitul OVSNMP_V2API in campul *session_flags*.
Daemon-ul *ovtrapd* se leaga la portul 162, receptioneaza trap-urile si apoi le forwardeaza catre aplicatii locale

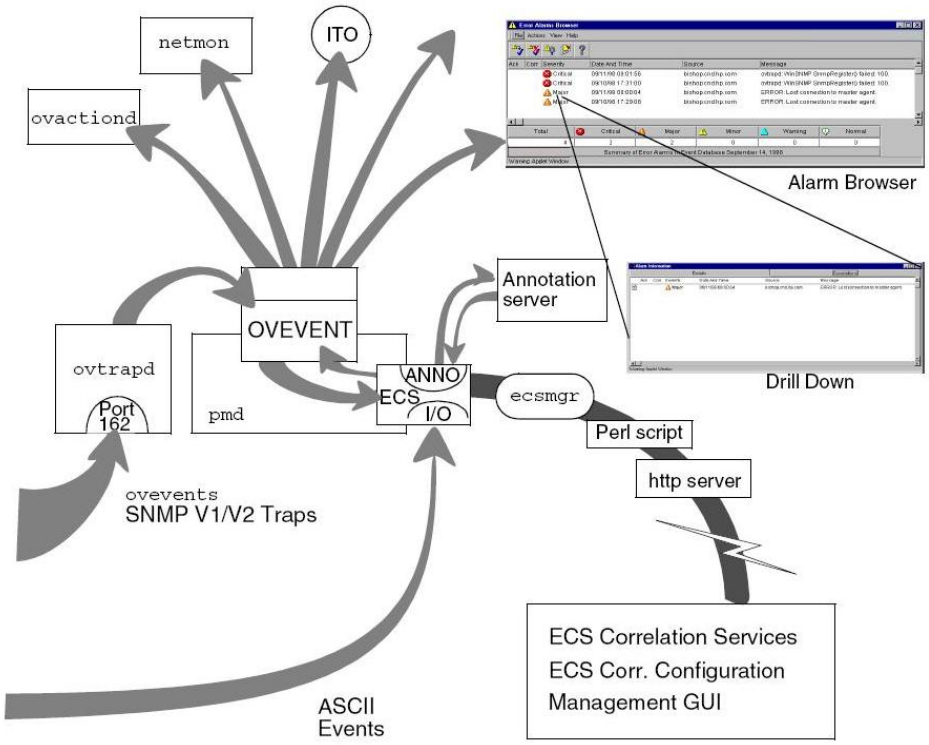

Figure 11 Arhitectura OVSNMP API

.

O aplicatie care subscrie la subsistemul de evenimente trebuie sa specifice unul din urmatoarele moduri : RAW – include toate evnimentele primite de subsistemul de evenimente excluzand cele generate de motorul ECS ; CORR – include toate evenimentele ce vin de la un anumit stream de evenimente ; ALL – include toate evenimentele primite de subsistemul de evenimente plus cele generate de ECS. Folosind functia `OVsnmpEventOpen()` o aplicatie poate stabili o sesiune de comunicare directa cu subsistemul de evenimente pentru a trimite si receptiona evenimente. Cu ajutorul filtrelor se specifica ce evenimente sa se receptioneze (RAW, CORR, ALL) . Fiecare eveniment are un identificator unic (OvsnmpUUID) ce poate fi extras cu ajutorul functiilor `OVsnmpEventSend()`, `OVsnmpExtractUUID()`.

La fel ca si NetSNMP API, exista doua moduri de programare : **blocant** si **neblocant**. In modul neblocant stiva OVSNMP are nevoie de un mecanism care sa informeze aplicatia cand un eveniment este receptionat iar lucrul acesta se face cu ajutorul functiei «***callback*** » Daca se folosesc cereri blocante, libraria SNMP ofera posibilitatetea retransmisiei pe baza de intreval de timp sau numar de reincercari. In cazul cererilor neblocante, retransmisia se face automat folosind bucle de evenimente.

In continuare se vor descrie cele mai importante functii, necesare pentru o dezvoltare minimala :

OVsnmpOpen() ; OVsnmpClose()

Stabileste o sesiune logica cu OVsnmpAPI cu scopul de a comunica cu un agent snmp. Primeste ca parametrii de intare *peername* si *remote_port*. *Peername* este un sir de caractere care identifica agentul SNMP catre care se adreseaza cererile si poate fi un nume de host sau adresa IP. *Remote_port* este un numar ce identifica portul UDP al agentului SNMP. Daca se specifica valoarea SNMP_USE_DEFAULT_REMOTE_PORT pentru *remote_port* atunci se foloseste valoarea 161. Sesiunea este terminata cu ajutorul functiei **OVsnmpClose()**

OVsnmpEventOpen()

Stabileste o sesiune logica cu OVsnmpAPI pentru a receptiona evenimente SNMP ; permite aplicarea filtrelor pentru a reduce numarul de evenimente.

OVsnmpCreatePdu()

Aloca si initializeaza o structura OVsnmpPdu utilizata in operatiile SNMP.

OVsnmpAddNullVarBind()

Aloca si initializeaza o structura OVsnmpVarBind si o ataseaza in datagramele de cereri snmp (get, getnext…)

OVsnmpFreePdu()

Dealoca o structura de date OVsnmpPdu

OVsnmpBlockingSend()
Trimite o cerere SNMP catre un agent si asteapta un raspuns

OVsnmpRecv()

Receptioneaza un raspuns snmp sau trap-uri si intoarce rezultatul. Functia este suspendata pana cand se primesc date.

OVsnmpSend ()
Trimite o cerere snmp dar nu asteapta un raspuns.

OVsnmpRead()

Primeste date de la sesiunile in asteptare. Informatia este returnata cu ajutorul functiei callback.

Concluzii

Nevoia de sisteme de management al retelelor este tot mai mare datorita cresterii acestora in dimensiune si complexitate. Protocolul SNMP este o optiune viabila pentru astfel de sisteme deoarece este standardizat, este relativ simplu permitand dezvoltarea de aplicatii *in-house* intr-un timp scurt si foarte multe echipamente il suporta. Desigur, acest protocol are si dezavantaje, cum ar fi utilizarea protocolului UDP pentru transport ceea ce nu garanteaza livrarea mesajelor sau lipsa securitatii pentru versiunile 1 si 2c dar acestea nu cantaresc foarte mult in luarea deciziei de utilizare a protocolului SNMP pentru gestionarea echipamentelor. Exista si protocoale care depasesc aceste neajunsuri si ofera in plus alte functionalitati (de exemplu protocolul Q3), dar complexitate lor este sporita si uneori nu se justifica.
Gama sistemelor de management al retelelor este foarte variata, plecand de la pachete software « free » ce contin un set de aplicatii minime necesare in gestionarea retelei prin SNMP precum si instrumente de dezvoltare a aplicatiilor (pachetul NetSNMP Tools) si pana la sisteme comerciale foarte complexe ce pot gestiona zeci de mii de noduri, oferind functionalitati deosebite cum ar fi « Root Cause Analysis », rapoarte si statistici privind incidentele, corelarea evenimentelor din retea etc.(pachetul HP OpenView Network Node Manager), dar toate aceste functionalitati avand un pret mai mare.
In momentul in care se ia decizia ca un nou echipament sa fie introdus in retea, trebuie raspuns la intrebarea : « Care sunt costurile de management ale acestui echipament ? » Daca acel tip de echipament stie sa comunice SNMP, iar informatiile care se pot obtine sunt cele dorite, atunci costurile pot fi minime, iar un sistem de management bazat pe SNMP este alegerea potrivita.

Bibliografie

[1] Stephen B. Morris, *Network Management, MIBs and MPLS: Principles, Design and Implementation*, Addison Wesley, 2003
[2] Douglas Mauro, Kevin Schmidt, *Essential SNMP, 2nd Edition*, O'Reilly, 2005
[3] David T. Perkins (Author), Evan McGinnis (Author), *Understanding SNMP MIBs*, Prentice Hall PTR, 1996
[4] RFC 1155 - *Structure of Management Information(SMI)*
[5] RFC 1157 - *Simple Network Management Protocol (SNMP)*
[6] RFC 1156 - *Management Information Base for network management of TCP/IP-based internets*
[7] RFC 2578 - *Structure of Management Information Version 2 (SMIv2*)
[8] RFC 2579 - *Textual Conventions for SMIv2*
[9] RFC 1212 - *Concise MIB definitions*
[10] ITU-T Recommendation X.692 (2002) | ISO/IEC 8825-3:2002, *Information technology – ASN.1 encoding rules: Specification of Encoding Control Notation (ECN)*.
[11] NetSNMP toolkit, http://.net-snmp.sourceforge.net/
[12] HP OpenView NNM, *SNMP Developer's Guide*, http://www.openview.hp.com/products/nnm/