L'ADMINISTRATION DE RESEAU

Patrice KADIONIK, Maître de Conférence à l'ENSEIRB
kadionik@enseirb.fr
http://www.enseirb.fr/~kadionik

Qu'est-ce que l'administration de réseau ? Cet article en présente les concepts dans un premier temps. Le protocole SNMP qui est le standard de fait dans ce domaine sera ensuite passé en revue.
La mise en pratique de SNMP est présentée au travers la mise en œuvre du package NET-SNMP.
L'heure est maintenant aux systèmes embarqués (sous Linux) et l'on parle beaucoup de connectivité IP. La méthode classique de contrôle à distance d'un système embarqué est d'y intégrer un serveur web. Cet article décrit comment on peut faire la même chose avec SNMP. Un module électronique connecté sur le port parallèle d'un PC est présenté en détail et servira de tests à cet effet. Le lecteur verra comment piloter ce module par SNMP sous Linux. Il verra aussi comment faire de même avec µClinux comme exemple de Linux embarqué.

1. L'ADMINISTRATION DE RESEAU

1.1. Introduction

On peut se poser à quoi correspond le concept d'administration de réseau. L'ISO (*International Standard Organization*) a cerné 5 axes :

- La gestion des anomalies (*Fault Management*). L'objectif de l'administration réseau est d'avoir un réseau opérationnel sans rupture de service (taux de disponibilité à 99,999 % par exemple soit quelques secondes d'indisponibilité par an), ce qui définit une certaine Qualité de Service (QoS) offerte par l'opérateur à l'abonné. On doit être en mesure de localiser le plus rapidement possible toute panne ou défaillance. Pour cela, on surveille les alarmes émises par le réseau, on localise un incident par un diagnostic des alarmes, on journalise les problèmes…

- La gestion de la configuration réseau (*Configuration Management*). Il convient de gérer la configuration matérielle et logicielle du réseau pour en optimiser l'utilisation. Il est

important que chaque équipement, chaque compteur… soit parfaitement identifié de façon unique à l'aide d'un nom ou identificateur d'objet OID (*Object Identifier*).

- La gestion des performances (*Performance Management*). Il convient de contrôler à tout moment le réseau pour voir s'il est en mesure d'écouler le trafic pour lequel il a été conçu.
- La gestion de la sécurité (*Security Management*). On gère ici les contrôles d'accès au réseau, la confidentialité des données qui y transitent, leur intégrité et leur authentification.
- La gestion de la comptabilité (*Accounting Management*). L'objectif est de gérer la consommation réseau par abonné en vue d'établir une facture.

En fait, on s'aperçoit qu'un administrateur système d'un réseau local d'une entreprise, d'un campus, d'une école administre aussi son réseau. Il le fait sans trop de problèmes mais les difficultés s'amoncellent dès que la taille du réseau devient importante. La solution est alors de rationaliser, de normaliser les choses et l'on a proposé des normes d'administration de réseau.
L'ISO a proposé dans les années 80 la norme CMIS/CMIP (*Common Management Information Service* ISO 9595, *Common Management Information Protocol* ISO 9596) comme protocole d'administration de réseau et définit un cadre général au niveau architecture (ISO 7498).
En parallèle, l'IAB (*Internet Activities Board*) approuve le protocole SNMP (*Simple Network Management Protocol*) comme solution à cours terme et CMOT (*CMIP Over TCP*) à plus long terme. Au début des années 90, SNMP, plus simple, devient alors standard de fait et est adopté par de nombreux constructeurs. C'est LE protocole d'administration de réseau des réseaux IP mais aussi des réseaux des opérateurs comme pour les réseaux ATM !

1.2. Les concepts de SNMP

❖ Introduction

Bien qu'issu du monde IP, le protocole SNMP (*Simple Network Management Protocol*) développé dans les années 80 reprend beaucoup des idées du protocole CMIS/CMIP de l'ISO. SNMP est défini dans la RFC 1157 (*Request For Comments*). Il utilise le concept d'application Client/Serveur bien connu dans le monde IP :

- Sur chaque équipement administrable s'exécute un programme serveur : l'agent SNMP. Cet agent gère les informations relatives à l'équipement et sont stockées dans une base de données propre : la MIB (*Management Information Base*).
- On retrouve une station d'administration côté client qui interagit avec un ou plusieurs agents SNMP : le manager SNMP. Le manager est généralement une application possédant une interface graphique élaborée pour plus de convivialité. On peut citer comme exemple de manager le produit commercial Openview de HP.

On retrouve donc 3 éléments importants dans SNMP :

- Une base de donnée (MIB) gérée par chaque agent SNMP et modifiable éventuellement par un manager SNMP. Les RFC 1156 et 1213 définissent la MIB-I (version 1) puis la MIB-II qui remplace la précédente, c'est à dire les objets que doit gérer tout agent SNMP.
- Une structure commune et un système de représentation des objets de la MIB : le SMI (*Structure of Management Information*, RFC 1155).
- Un protocole d'échange entre manager et agent : le protocole SNMP (RFC 1157). Le protocole SNMP a évolué pour intégrer les aspects de confidentialité et d'authentification.

Seule, la version 1 de SNMP est décrite ici. C'est d'ailleurs la seule version adoptée et utilisé par tous.

❖ Le protocole SNMP

SNMP est un protocole bâti au dessus de UDP/IP :

SNMP
UDP
IP
Liaison

Figure 1 : SNMP et la pile de protocoles IP

L'utilisation du protocole UDP est justifiée pour sa simplicité, sa rapidité et sa concision (8 octets d'entête contre 20 pour TCP), ce qui permet de remonter très rapidement des alarmes vers un manager. Malheureusement, c'est un protocole de transport en mode non connecté et non fiable, ce qui signifie que l'on pourra perdre des messages SNMP…

5 types de messages ou requêtes SNMP peuvent être échangés (SNMPv1) entre agent et manager :

- Obtention de la valeur courante d'un objet de la MIB géré par un agent : requête `get-request` (GET).
- Obtention de la valeur courante du prochain objet de la MIB géré par un agent à partir d'un objet courant : requête `get-next-request` (GETNEXT).
- Mise à jour de la valeur courante d'un objet de la MIB géré par un agent : requête `set-request` (SET).
- Renvoi de la valeur d'un objet de la MIB géré par un agent : requête `get-response`. C'est la réponse à un GET, GETNEXT ou SET. On voit que SNMP est un protocole de type commande/réponse sans états.
- Signal émis par un agent en direction d'un manager (pour remonter une alarme par exemple) : message `trap` (TRAP).

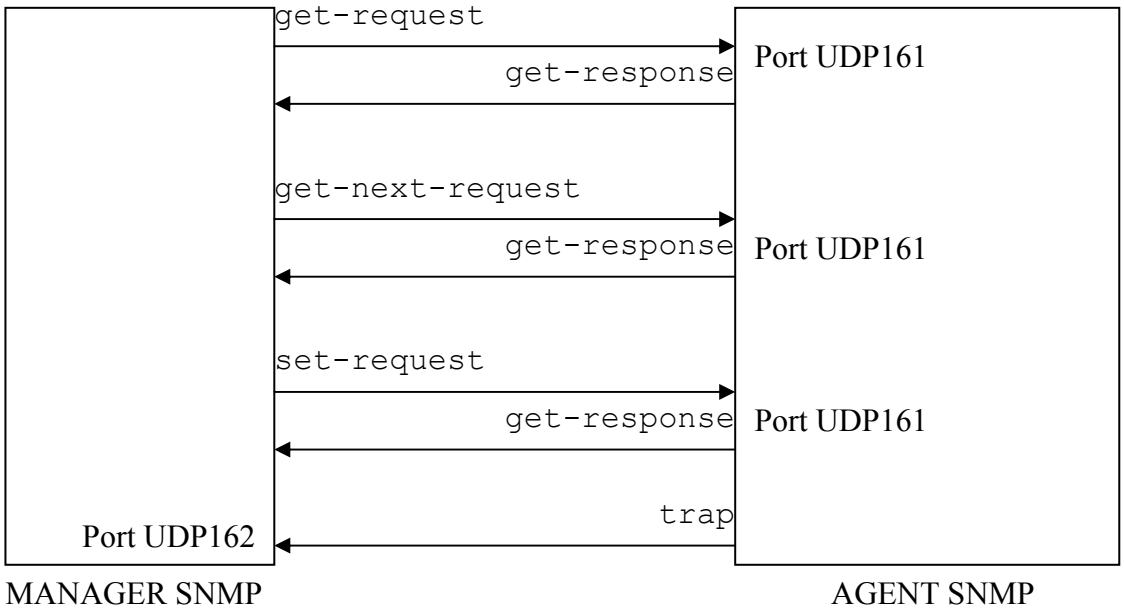

Figure 2 : requêtes SNMP et numéros de port UDP

SNMP utilise le port UDP 161 côté agent pour les requêtes SNMP GET, GETNEXT et SET et le port UDP 162 pour les TRAPs côté manager.
La structure générale d'une requête SNMPv1 `get-request`, `get-next-request`, `set-request` et `get-response` est la suivante :

entête IP 20 o.	entête UDP 8 o.	version (0)	communauté	type PDU (0 à 3)	Id requête	statut erreur (0 à 5)	index erreur	nom (type)	longueur	valeur	T	L	V	…

Figure 3 : datagramme IP d'une requête SNMP

La structure générale d'un `trap` SNMPv1 est la suivante :

entête IP 20 o.	entête UDP 8 o.	version (0)	communauté	type PDU (4)	entreprise	adresse agent	type TRAP (0 à 5)	code spéc.	time stamping	T	L	V	…

Figure 4 : datagramme IP d'un TRAP SNMP

Deux remarques sont à faire :

- Le champ communauté (*community*) est un chaîne de caractères qu'il faut voir comme un mot de passe de validation d'une requête SNMP par l'agent (GET, GETNEXT, SET) ou par le manager (TRAP). Si la communauté est incorrecte, la requête est rejetée. La communauté passe en clair sur le réseau. C'est une faille importante de sécurité de SNMPv1. Il faut attendre les versions ultérieures de SNMP pour combler cette lacune.
- Les paramètres d'une requête sont encodés suivant le codage TLV (Type, Longueur, Valeur) ou BER (*Basic Encoding Rules*) défini dans SMI. Ce codage est assez classique en télécommunications et on le retrouve par exemple utilisé pour la signalisation Q.931 du réseau RNIS à l'interface Usager/Réseau. Il faut noter que ce système d'encodage est gourmand en octets car un paramètre sur 1 octet sera encodé en utilisant 3 octets (l'entier 12 est encodé comme $02 $01 $0C) ! On peut en revanche passer un nombre variable de paramètres par ce système d'encodage.

❖ Structure SMI

La structure SMI décrit les règles de description de l'information et permet d'identifier de façon unique un objet de la MIB géré par un agent SNMP. Chaque objet possède donc un identificateur unique ou OID (*Object ID*).
SMI s'intéresse aussi à la représentation des données (et leur type) pour chaque objet de la MIB. Un objet de la MIB est déclaré et défini en langage ASN.1 (*Abstract Syntax Notation 1* : langage de représentation de donnée).
SNMP n'utilise qu'une petite partie du langage ASN.1. Au niveau des types, seuls quelques uns sont utilisés comme :

- INTEGER : valeur entière sur 32 bits en complément à 2.
- OCTET STRING : chaîne de caractères.
- IpAddress : adresse IP.
- PhysAddress : adresse MAC (6 octets pour un réseau de type Ethernet).
- Counter : entier de 32 bits non signé qui s'accroît de 0 à (2exp32 –1) puis revient à 0.
- TimeTicks : compteur de temps sur 32 bits non signé en 1/100 de s.
- …

❖ Structure de la MIB

La MIB est une base de donnée gérée par un agent SNMP regroupant les objets gérés en respectant les règles SMI. Elle possède une structure d'arbre similaire à celui employé dans le DNS (*Domain Name System)*. On retrouve une racine non nommée à partir de laquelle on référencie de façon absolue un objet par son OID (nœud de l'arbre).

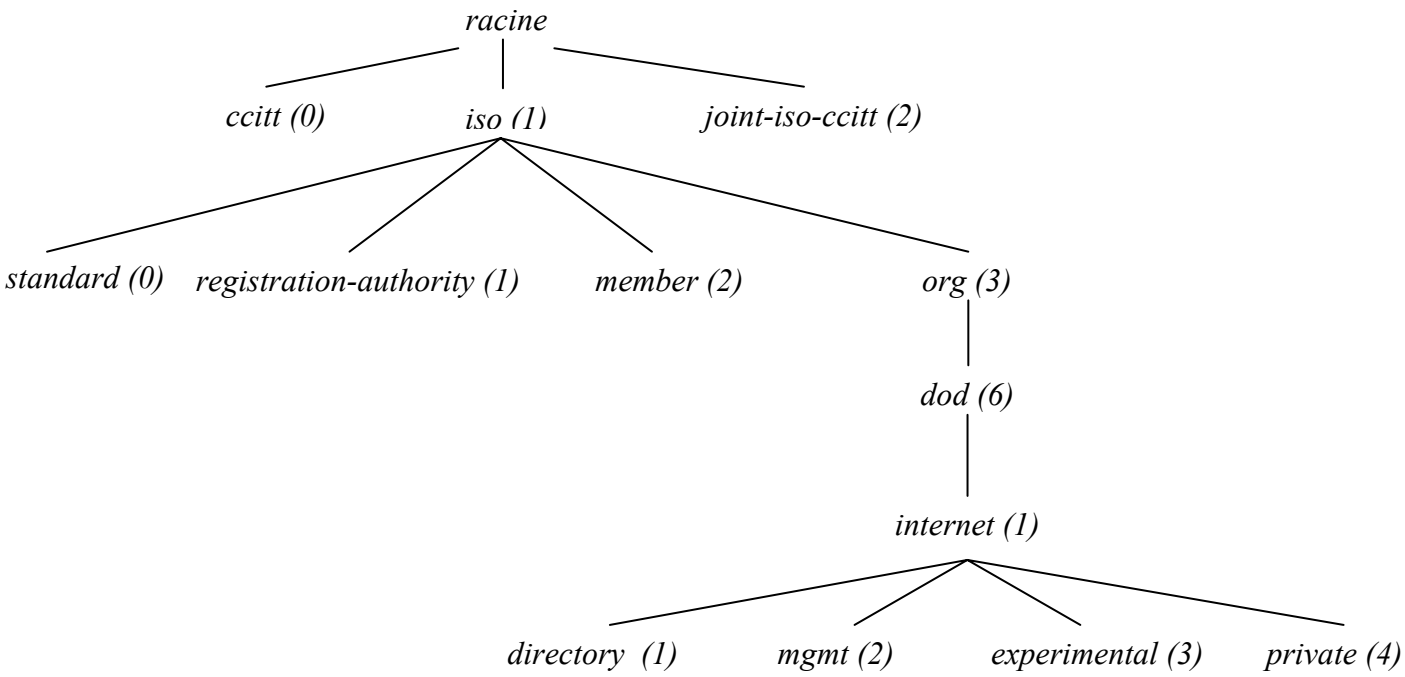

Figure 5 : structure de la MIB

Si l'on fait un zoom sur la branche `internet` de la MIB, on a :

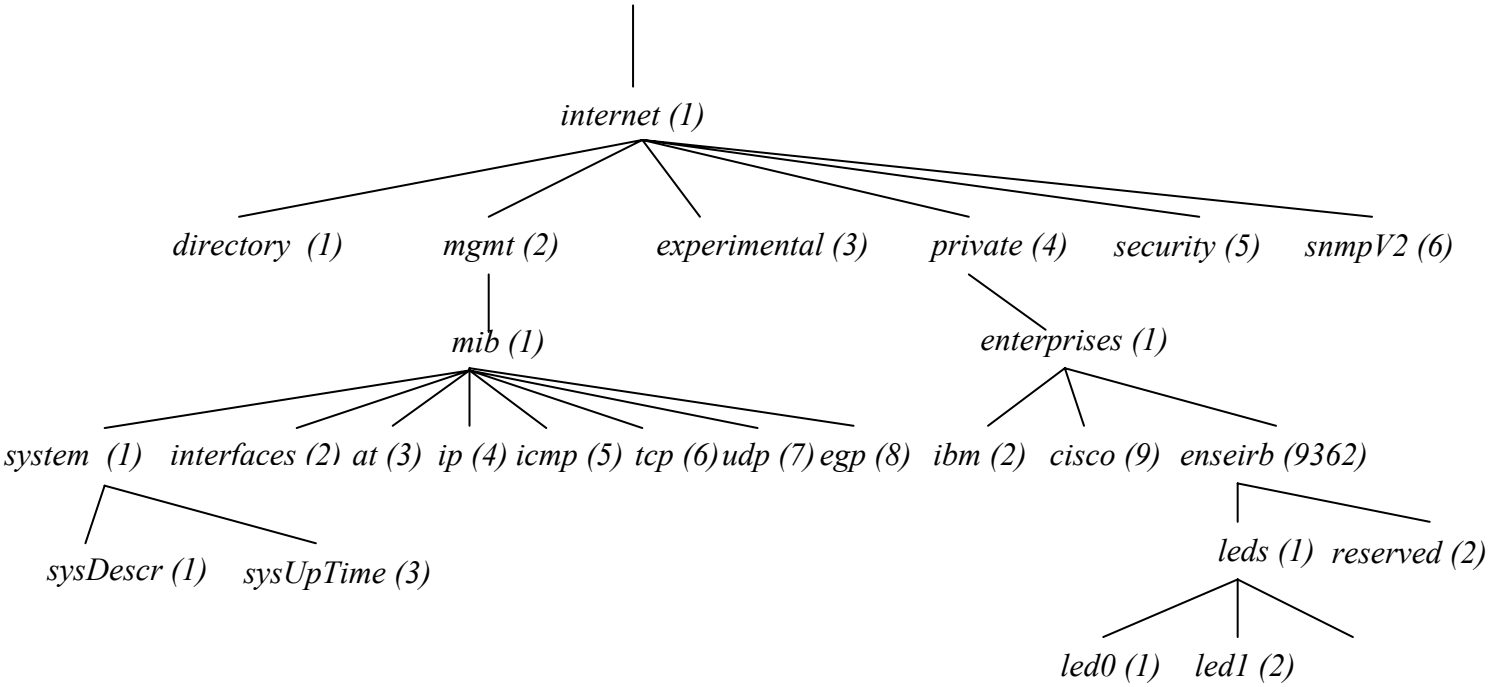

Figure 6 : structure de la MIB. Zoom de la branche internet

Chaque nœud de l'arbre possède un nom symbolique. Chaque objet pourra être identifié de façon symbolique ou en utilisant son OID.

Par exemple, l'objet `sysDescr` a comme nom symbolique `iso.org.dod.internet.mgmt.mib.system.sysDescr.0`. Son OID en respectant les règles SMI est 1.3.6.1.2.1.1.1.0.

On voit que les objets intéressants ont tous un OID commençant par 1.3.6.1.2.1. Il faut noter que c'est l'OID qui est transmis dans une requête SNMP et non le nom symbolique de l'objet.

La MIB contient un certain nombre d'objets standards : c'est la MIB standard. Si l'agent doit gérer des objets propres, ils font partie de la MIB privée. Ces objets sont placés dans la branche `iso.org.dod.internet.private.enterprises.nom_entreprise`. On « étend » ainsi la MIB de l'agent SNMP. Il convient pour cela de souscrire un numéro identificateur d'entreprise ou d'institution à l'IANA (*Internet Assigned Numbers Authority*). Le numéro assigné à l'ENSEIRB (Ecole Nationale Supérieure d'Electronique, Informatique et Radiocommunications de Bordeaux) est 9362.

L'objet `led0` a comme nom symbolique `iso.org.dod.internet.private.enterprises.enseirb.leds.led0.0`. Son OID en respectant les règles SMI est 1.3.6.1.4.1.9362.1.1.0.

On remarquera que l'accès à un objet simple (feuille de l'arbre) de la MIB se fait en rajoutant .0 comme suffixe à l'OID.

1.3. Les différentes versions de SNMP

En 1993, de nouvelles RFC (RFC 1441 à 1452) ont été rajoutées pour réviser SNMPv1 (RFC 1155 à 1157) pour définir SNMPv2c (*classic*). SNMPv2c a été mis à jour en 1996 (RFC 1901 à 1908) pour donner SNMPv2.
Les principales différences entre SNMPv1 et SNMPv2 sont :

- De nouvelles requêtes SNMP définies.
- Deux nouvelles branches dans la MIB : snmpV2 et snmpV2-M2M (*Manager to Manager*).
- Correction des failles de sécurité : pas de communauté en clair sur le réseau. Utilisation de procédés de chiffrement et d'authentification des requêtes SNMP.
- Mécanismes pour la configuration à distance.

Malheureusement, SNMPv2 a été gelé et n'est pas déployé (en expérimentation). Comble de malchance, il existe aussi SNMPv3 (RFC 2571 à 2575) en gestation…
Ces hésitations et différentes versions brouillent beaucoup le jeu. On retiendra donc que la seule version stable et officielle est SNMPv1 (avec ses défauts…) !

2. MISE EN ŒUVRE DE NET-SNMP SOUS LINUX

2.1. Introduction

La mise en œuvre de SNMP sous Linux se fera en utilisant le standard de fait dans le logiciel libre : le package NET-SNMP.

Le projet NET-SNMP appelé anciennement UCD-SNMP a été historiquement développé par l'université américaine *Carnegie Mellon University* (CMU) puis amélioré et maintenu maintenant par l'université américaine *University of California Davis* (UCD).

NET-SNMP est en fait un ensemble d'outils et de fonctionnalités :

- Une API (*Application Programming Interface*) d'accès à SNMP.
- Un agent SNMP extensible.
- Des commandes en ligne pour interroger des agents SNMP.
- Des commandes en ligne pour gérer et générer des TRAPs SNMP.
- Une version de la commande UNIX netstat utilisant SNMP.
- Un browser de MIB SNMP (tkmib) écrit en Tk.

NET-SNMP est porté sur différents systèmes et en particulier sur :

- Linux (noyaux 2.4 à 1.3).
- HP-UX (10.20 à 9.01 et 11.0).
- Ultrix (4.5 à 4.2).
- Solaris (2.8 à 2.3) et SunOS (4.1.4 à 4.1.2).
- NetBSD (1.5alpha à 1.0).
- FreeBSD (4.1 à 2.2).
- Win32.
- …

NET-SNMP supporte SNMPv1, SNMPv2 et SNMPv3 que ce soit côté agent SNMP comme du côté manager SNMP via les commandes en ligne NET-SNMP.

2.2. Installation de NET-SNMP

L'installation de NET-SNMP est des plus traditionnels sous Linux :

Décompression :
```
# cd
# tar xvzf net-snmp-5.0.2.tar.gz
# ln -s net-snmp-5.0.2 net-snmp
```

Configuration. On choisira d'utiliser par défaut SNMPv1 :
```
# cd ~/net-snmp
# ./configure
```

Compilation :
```
# make
```

Installation :
```
# make install
```

Les commandes Linux NET-SNMP sont recopiées dans le répertoire `/usr/local/bin` et `/usr/local/sbin` qu'il faudra rajouter à sa variable d'environnement `PATH`.

Les fichiers correspondants aux MIBs exploités côté manager SNMP par les commandes en ligne NET-SNMP sont sous `/usr/local/share/snmp/mibs`.

2.3. Configuration et lancement de l'agent SNMP NET-SNMP

L'agent SNMP NET-SNMP est l'exécutable `snmpd` sous `/usr/local/sbin`. Il possède un fichier de configuration général s'appelant `snmpd.conf` à copier sous `/usr/local/share/snmp`. Un exemple de fichier qu'il faudra bien sûr modifier en fonction de ce que l'on veut faire est fourni. C'est le fichier `EXAMPLE.conf` sous `~/net-snmp`.

On pourra consulter l'aide en ligne sur la structure de ce fichier (`# man snmpd.conf`).

Les champs les plus importants sont :

Déclaration d'une machine et d'un réseau IP ayant accès à l'agent SNMP, `local` et `mynetwork` de communauté `tst` :
```
#          sec.name  source          community
com2sec local     localhost       tst
com2sec mynetwork 192.9.201.0/24  tst
```

Déclaration de groupes d'accès aux objets de la MIB de l'agent SNMP pour `local` et `mynetwork` :
```
####
# Second, map the security names into group names:
#                 sec.model  sec.name
```

```
group MyRWGroup   v1          local
group MyRWGroup   v2c         local
group MyROGroup v1          mynetwork
group MyROGroup v2c         mynetwork
```

Accès en lecture/écriture aux objets de la MIB de l'agent pour l'accès `local` et lecture seulement pour l'accès `mynetwork` :

```
####
# Finally, grant the 2 groups access to the 1 view with different
# write permissions:
#                context sec.model sec.level match  read   write  notif
access MyROGroup ""      any       noauth    exact  all    none   none
access MyRWGroup ""      any       noauth    exact  all    all    none
```

Autorisation de la génération de TRAPs SNMP en direction d'un manager SNMP de la machine `localhost` de communauté `tst` :

```
####
# Traps and v2 traps enabled and sent to localhost
#
# command   host manager      community
trapsink    localhost   tst
trap2sink   localhost   tst
```

Renseignement de l'objet `sysLocation` de la branche `system` :

```
syslocation ENSEIRB, Bordeaux, France
```

Renseignement de l'objet `sysContact` de la branche `system` :

```
syscontact Kadionik <kadionik@enseirb.fr>
```

L'utilitaire `snmpconf` permet de créer le fichier `snmpd.conf` de façon interactive et conviviale sans en connaître exactement sa structure.

Création rapide du fichier `snmpd.conf` à la première utilisation :

```
# snmpconf -g basic_setup
```

Mode interactif :

```
# snmpconf
```

Après création du fichier `snmpd.conf` avec les valeur décrites précédemment et recopie sous `/usr/local/share/snmp`, il ne reste plus qu'à lancer l'agent `snmpd` :

```
# snmpd
```

On pourra lancer dans une autre fenêtre un sniffer réseau comme `tcpdump` pour voir les échanges agent SNMP/manager SNMP :

```
# tcpdump -vv -i lo
```

2.4. Tests de l'agent SNMP NET-SNMP

Le test de l'agent SNMP NET-SNMP se fait en utilisant un manager SNMP. Dans le cas du package NET-SNMP, on a accès à des commandes en ligne sous `/usr/local/bin` permettant d'émettre des requêtes SNMP.

Il convient d'abord de configurer son environnement Linux. Pour accéder aux objets de la MIB d'un agent sous forme symbolique et non sous forme décimale OID, il faut aller lire l'ensemble des fichiers MIB sous `/usr/local/share/snmp/mibs` :

```
#
# PATH
#
PATH=$PATH:/usr/local/bin:/usr/local/sbin

#
# MIBS : forces to read all MIB files under /usr/local/share/snmp/mibs
#
MIBS=ALL

#
# exporting all variables
#
export PATH MIBS
```

Les principales commandes utiles notées globalement `snmpxxx` sont :

- `snmpget` : envoi d'une requête SNMP GET pour obtenir une information sur un objet de la MIB d'un agent SNMP distant.
- `snmpset` : envoi d'une requête SNMP SET pour mettre à jour la valeur d'un objet de la MIB d'un agent SNMP distant.
- `snmpgetnext` : envoi d'une requête SNMP GETNEXT et donne aussi la valeur de l'objet suivant de la MIB d'un agent SNMP distant si toutefois il en existe un.
- `snmpwalk` : cette commande fonctionne comme `snmpgetnext` mais permet de balayer complètement une branche de la MIB d'un agent SNMP distant.
- `snmptranslate` : permet de convertir un objet d'une MIB représenté sous sa forme décimale OID en sa forme symbolique et réciproquement.

L'utilitaire `snmpconf` permet de créer le fichier `snmp.conf` de façon interactive et conviviale sans en connaître sa structure pour configurer par défaut l'usage des commandes `snmpxxx` précédentes (choix de la version du protocole SNMP par défaut, communauté…). Le fichier ainsi créé sera à recopier sous `/usr/local/share/snmp`.

Une fois l'agent SNMP lancé sur sa machine comme précédemment, il est ensuite possible de le tester avec les commandes `snmpxxx`.

Correspondance OID et nom symbolique :

```
# snmptranslate 1.3.6.1.2.1.1.3.0
SNMPv2-MIB::sysUpTime.0
```

Représentation sous forme graphique de la branche `system` de la MIB par analyse de l'ensemble des fichiers MIB sous `/usr/local/share/snmp/mibs` :

```
# snmptranslate -Tp -IR system
+--system(1)
   |
   +-- -R-- String    sysDescr(1)
   |        Textual Convention: DisplayString
   |        Size: 0..255
. . .
```

Récupération par SNMPv1 de la valeur courante de l'objet `sysUpTime` géré par l'agent SNMP de la machine `localhost` :

```
# snmpget -v 1 -c tst localhost system.sysUpTime.0
SNMPv2-MIB::sysUpTime.0 = Timeticks: (12908) 0:02:09.08
```

Récupération de la valeur courante de l'objet `sysUpTime` avec SNMPv2c :

```
# snmpget -v 2c -c tst localhost system.sysUpTime.0
SNMPv2-MIB::sysUpTime.0 = Timeticks: (13966) 0:02:19.66
```

Récupération de la valeur courante des objets `sysLocation` et `sysContact` (renseignés dans le fichier `snmpd.conf`) :

```
# snmpget -v 1 -c tst localhost system.sysLocation.0
SNMPv2-MIB::sysLocation.0 = STRING: ENSEIRB, Bordeaux, France
# snmpget -v 1 -c tst localhost system.sysContact.0
SNMPv2-MIB::sysContact.0 = STRING: Kadionik <kadionik@enseirb.fr>
```

Parcours de la branche `system` de la MIB de l'agent SNMP :

```
# snmpwalk -v 1 -c tst localhost system
SNMPv2-MIB::sysDescr.0 = STRING: Linux poire1 2.4.18-3 #1 Thu Apr 18
07:31:07 EDT 2002 i586
SNMPv2-MIB::sysObjectID.0 = OID: NET-SNMP-MIB::netSnmpAgentOIDs
SNMPv2-MIB::sysUpTime.0 = Timeticks: (28119) 0:04:41.19
SNMPv2-MIB::sysContact.0 = STRING: Kadionik <kadionik@enseirb.fr>
SNMPv2-MIB::sysName.0 = STRING: poire1
SNMPv2-MIB::sysLocation.0 = STRING: ENSEIRB, Bordeaux, France
. . .
```

Affection d'une nouvelle valeur à l'objet `sysLocation` de la MIB de l'agent avec SNMPv1 :

```
# snmpset -v 1 -c tst localhost system.sysLocation.0 s "coucou"
Error in packet.
Reason: (noSuchName) There is no such variable name in this MIB.
Failed object: SNMPv2-MIB::sysLocation.0
```

Affection d'une nouvelle valeur à l'objet `sysLocation.0` de la MIB de l'agent avec SNMPv2c :

```
# snmpset -v 2c -c tst localhost system.sysLocation.0 s "coucou"
Error in packet.
Reason: notWritable (that object does not support modification)
Failed object: SNMPv2-MIB::sysLocation.0
```

En comparant le résultat des 2 dernières commandes, on voit que SNMPv1 ne définit pas en retour de code d'erreur, ce que corrige SNMPv2c. Dans notre cas, on essaye de modifier un objet de la MIB de l'agent SNMP distant accessible en lecture seulement !

La trace tcpdump générée par l'exécution de la commande :

```
# snmpget -v 1 -c tst localhost sysUpTime.0
```

est la suivante :

```
# tcpdump -vv -i lo
tcpdump: listening on lo
15:17:54.733310 poire1.1092 > poire1.snmp:  [udp sum ok] |30|26|02|01{
SNMPv1 |04|03C=tst |a0|1c{ GetRequest(28) |02|04R=691700679
|02|01|02|01|30|0e |30|0c|06|08system.sysUpTime.0|05|00} }  (DF) (ttl 64,
id 0, len 68)
15:17:54.734564 poire1.snmp > poire1.1092:  [udp sum ok] |30|29|02|01{
SNMPv1 |04|03C=tst |a2|1f{ GetResponse(31) |02|04R=691700679
```

```
|02|01|02|01|30|11 |30|0f|06|08system.sysUpTime.0=|43|03127125} }   (DF)
(ttl 64, id 0, len 71)
```

On voit que la communauté (ici tst) qu'il faut considérer comme un mot de passe circule en clair sur le réseau avec SNMPv1 !

Il est à noter que l'on peut aussi générer des TRAPs SNMP en direction d'un manager SNMP d'une machine en utilisant la commande `snmptrap`. Le démon `snmptrapd` (sous `/usr/local/sbin`) peut être utilisé comme collecteur pour traiter les TRAPs SNMP. Il utilise le fichier de configuration `snmptrapd.conf` à recopier sous `/usr/local/share/snmp`. Ce fichier de configuration peut être généré en utilisant l'outil `snmpconf` comme précédemment. On verra l'utilisation des TRAPs SNMP lors du contrôle à distance par SNMP d'un système électronique.

3. MODULE ELECTRONIQUE DE TEST

Un petit module électronique se branchant sur le port parallèle d'un PC est proposé afin d'illustrer les possibilités de contrôle à distance par SNMP d'un système électronique.

Le schéma de principe de ce module est donné ci après.

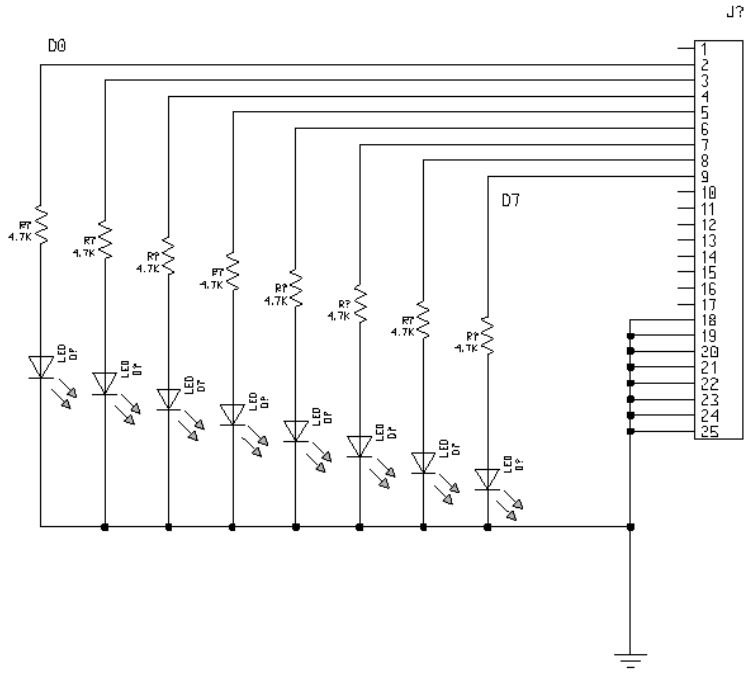

Figure 7 : schéma électrique du module électronique

Huit leds sont respectivement connectées sur les signaux de données D0 à D7 de l'interface parallèle.

Le typon pour la réalisation du circuit imprimé simple face est donné ci-après. On pourra récupérer le fichier PostScript correspondant à l'adresse WWW donnée à la fin de l'article pour une impression à l'échelle 1.

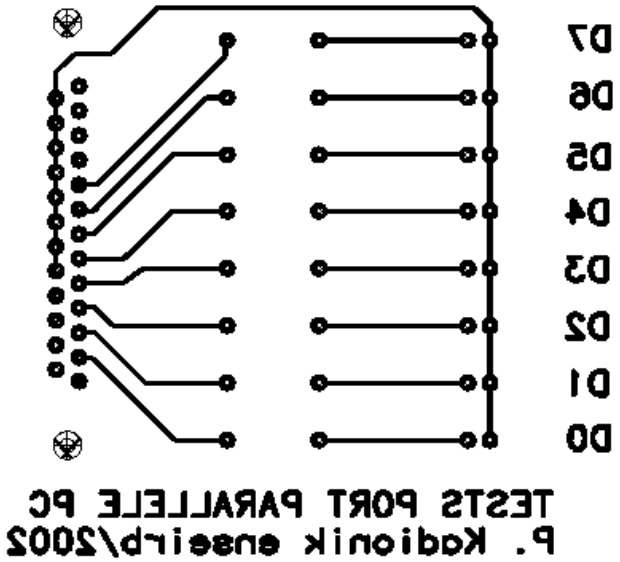

Figure 8 : typon du module électronique

Une plaque de circuit imprimé simple face suffit pour la réalisation. Il est à noter que le texte côté cuivre doit apparaître de façon lisible (à l'envers sur la figure). Pour les électroniciens, on pourra récupérer les fichiers Gerber et de perçage à la même adresse que précédemment.

Au niveau matériel, on a besoin de :

- 1 connecteur femelle 25 broches coudé à souder de type ITT.
- 8 résistances de 4,7 Kohms.
- 8 leds.
- 1 câble port parallèle droit fil à fil mâle/mâle pour la connection entre le module électronique et le PC.

Le schéma d'implémentation est donné ci-après.

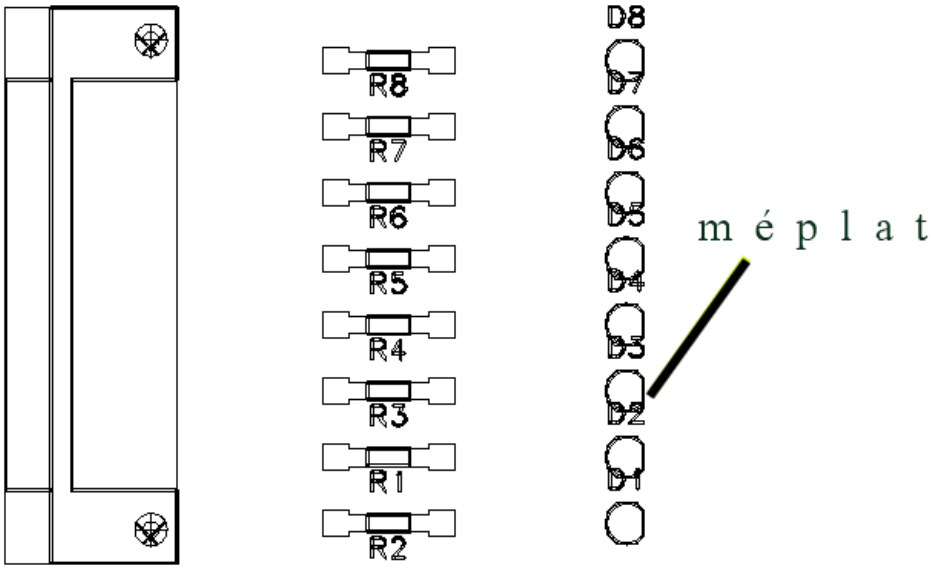

Figure 9 : layout du module électronique

La seule précaution est de ne pas souder les leds à l'envers, le méplat de chaque led devant être orienté à l'opposé du connecteur DB25.

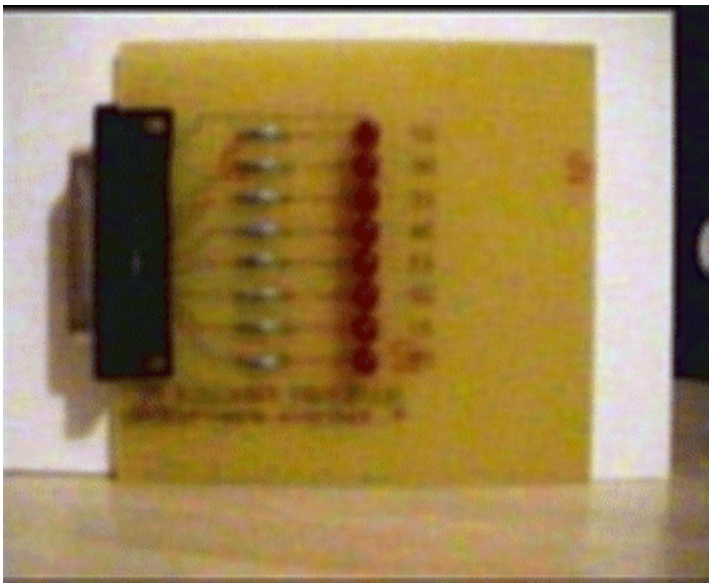

Figure 10 : photo du module électronique

Il n'y a pas de précautions particulières à respecter si ce n'est d'éviter les courts-circuits lors de la réalisation du circuit imprimé et l'auteur ne peut bien sûr être tenu responsable d'éventuelles fausses manipulations.

Une fois le module électronique connecté, il est possible de le tester simplement à l'aide des programmes de test fournis. Les fichiers sources `readleds.c` et `writeleds.c` seront compilés pour produire les exécutables `readleds` et `writeleds`.

La commande suivante éteint les 8 leds. On relit leur valeur courante :

```
# writeleds 0
Leds written = 0x00
```

```
# readleds
Leds read = 0x00
```

La commande suivante allume les 8 leds :

```
# writeleds ff
Leds written = 0xff
```

La commande suivante allume les leds paires (D0…) et éteint les leds impaires (D1…):

```
# writeleds 55
Leds written = 0x55
```

La commande suivante allume les leds impaires et éteint les leds paires :

```
# writeleds AA
Leds written = 0xaa
```

Le shell script `k2000.sh` permet de réaliser un chenillard à la « K2000 » ☺ :

```
# k2000.sh
```

Ces 3 programmes doivent être exécutés en tant que root pour pouvoir accéder au port parallèle.

4. CONTROLE PAR SNMP D'UN SYSTEME ELECTRONIQUE SOUS LINUX

Le but de cette partie est de montrer la façon d'étendre l'agent SNMP NET-SNMP pour le contrôle par SNMP du module électronique de test. La manière classique de contrôler à distance un système électronique est d'embarquer un serveur web comme décrit dans l'article « Linux embarqué : le projet µClinux » de Linux Magazine de février 2002. Embarquer SNMP en est une autre plus originale assurant une interopérabilité avec tous les logiciels d'administration de réseau (HP Openview…).

4.1. Ecriture du fichier MIB en ASN.1

La première chose à faire est de créer un fichier texte `MIB-ENSEIRB.txt` écrit en ASN.1 décrivant les nouveaux objets à intégrer dans la MIB privée de l'agent SNMP NET-SNMP. On a ainsi créé un objet pour chaque led à piloter du module électronique. Cette extension se fait toujours dans la branche `private` de l'arbre de la MIB. Il est à noter qu'il faut pour un professionnel avoir un identificateur unique (9362 pour l'ENSEIRB) attribué par l'IANA.
SNMP n'utilise qu'une partie du langage ASN.1 (recommandation UIT-T X.208) et l'on pourra s'inspirer des fichiers sous `/usr/local/share/snmp/mibs` en guise d'apprentissage du langage ASN.1.

```
ENSEIRB-MIB DEFINITIONS ::= BEGIN

IMPORTS
      enterprises                              FROM SNMPv2-SMI,
      MODULE-IDENTITY                          FROM SNMPv2-SMI
      MODULE-COMPLIANCE, OBJECT-GROUP          FROM SNMPv2-CONF;

--
```

```
-- A brief description and update information about this mib.
--
enseirb MODULE-IDENTITY
    LAST-UPDATED "0104010000Z"            -- 01 Apr 2001, midnight
    ORGANIZATION "ENSEIRB"
    CONTACT-INFO "
                  Author:     Patrice Kadionik
                              ENSEIRB, School of Electrical Engineering
                  postal:     PO Box 99
                              33402 TALENCE CEDEX
                              FRANCE
                  email:      kadionik@enseirb.fr
                  phone:      +33-5-56-84-65-00
                 "
    DESCRIPTION  "MIB for remote control by SNMP
                 "
    ::= { enterprises 9362 }

leds               OBJECT IDENTIFIER ::= {  enseirb 1 }
reserved           OBJECT IDENTIFIER ::= { enseirb 2 }

--
-- LED 0 connected to the PC parallel port
--
led0            OBJECT-TYPE
    SYNTAX      Integer32 (0..1)
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
      "Led 0 connected to bit 0 of the PC parallel port."
    DEFVAL { 0 }
    ::= { leds 1 }

. . .

--
-- LED 7 connected to the PC parallel port
--
led7            OBJECT-TYPE
    SYNTAX      Integer32 (0..1)
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
      "Led 7 connected to bit 7 of the PC parallel port."
    DEFVAL { 0 }
    ::= { leds 8 }
END
```

Le fichier `ENSEIRB-MIB.txt` sera ensuite recopié sous `/usr/local/share/snmp/mibs` pour pouvoir être exploité par les commandes `snmpxxx` au niveau symbolique.

On pourra tester le fichier créé et corriger les éventuelles erreurs signalées en utilisant simplement la commande :

```
# snmptranslate -Tp -IR enseirb
+--enseirb(9362)
   |
   +--leds(1)
   |  |
   |  +-- -RW- Integer32 led0(1)
```

```
|   |          Range: 0..1
|   +-- -RW- Integer32 led1(2)
|   |          Range: 0..1
|   +-- -RW- Integer32 led2(3)
|   |          Range: 0..1
|   +-- -RW- Integer32 led3(4)
|   |          Range: 0..1
|   +-- -RW- Integer32 led4(5)
|   |          Range: 0..1
|   +-- -RW- Integer32 led5(6)
|   |          Range: 0..1
|   +-- -RW- Integer32 led6(7)
|   |          Range: 0..1
|   +-- -RW- Integer32 led7(8)
|              Range: 0..1
|
+--reserved(2)
```

On a 8 objets dans la branche `enseirb.leds` du type entier 32 bits accessibles en lecture/écriture et pouvant prendre une valeur entière comprise entre 0 et 1, c'est à dire 0 ou 1 !
On choisit comme convention la valeur 0 pour éteindre la led correspondante et 1 pour l'allumer.

4.2. Extension de l'agent SNMP NET-SNMP

Il convient maintenant d'intégrer dans l'agent SNMP NET-SNMP la branche `enseirb` sous `iso.org.dod.internet.private.enterprises` (voir figure 6) et les actions associées d'allumage et d'extinction des leds du module électronique.

L'outil `mib2c` écrit en langage PERL a besoin de modules PERL spécifiques à installer :

```
# cd ~/net-snmp
# cd perl
# cd default_store
# perl Makefile.PL
# make
# make install
# cd ..
# cd SNMP
# perl Makefile.PL
# make
# make install
```

L'installation étant à présent complète, on va maintenant exploiter l'outil `mib2c`. `mib2c` permet de transformer un fichier MIB écrit en ASN.1 en fichiers C qui seront ensuite compilés avec l'agent SNMP pour en étendre sa MIB. On remarque que :

- Les fichiers générés sont des fichiers squelettes à compléter avec le code pour piloter le module électronique. De ce fait, à chaque utilisation de `mib2c`, les fichiers générés écrasent les précédents. Il faut donc veiller à bien les sauvegarder après modification.
- L'extension de la MIB privée de l'agent SNMP est intégrée dans l'agent comme toute la MIB qu'il gère d'ailleurs. L'agent SNMP ne lit donc pas de fichiers MIB à son lancement et notamment les fichiers sous `/usr/local/share/snmp/mibs`. Il faudra ainsi recompiler l'agent à chaque fois que l'on étend sa MIB…

On se place ensuite dans le répertoire `~/net-snmp/agent/mibgroup` avant d'exécuter `mib2c` :

```
# cd ~/net-snmp
# cd agent/mibgroup
# mib2c -c ../../local/mib2c.scalar.conf enseirb
writing to enseirb.h
writing to enseirb.c
running indent on enseirb.h
running indent on enseirb.c
```

Il y a génération des fichiers sources C `enseirb.c` et `enseirb.h` que l'on va modifier pour piloter le module électronique. `mib2c` prend comme paramètre un objet SNMP sous forme symbolique (d'où l'intérêt de recopier le fichier `MIB-ENSEIRB.txt` sous `/usr/local/share/snmp/mibs`), `enseirb` dans notre cas.

Il convient maintenant de modifier le fichier `enseirb.c` pour y intégrer le pilotage du module. Ce fichier contient l'intégration de la MIB ENSEIRB dans la MIB de l'agent SNMP. On notera la présence de fonctions handlers exécutées pour chaque objet rajouté dans la MIB de l'agent pour des requêtes SNMP GET et SET. Ce sont les fonctions appelées ici `do_led0()`, ..., `do_led7()`.

Une machine à états finis est mise en œuvre dans chaque handler pour le traitement des requêtes SNMP. On notera les états caractéristiques suivants :

- MODE_GET : traitement d'une requête SNMP GET. On retourne dans ce cas là la valeur courante de la led.
- MODE_SET_RESERVE1 : vérification sur le type, la valeur de l'objet de la MIB de l'agent à modifier par une requête SNMP SET.
- MODE_SET_RESERVE2 : allocation mémoire si besoin est (`malloc()`).
- MODE_SET_FREE : libération mémoire si besoin est (`free()`).
- MODE_SET_ACTION : modification de l'objet de la MIB. Dans notre cas, on va changer l'état de la led du module électronique et sauvegarder sa nouvelle valeur courante (0 : led éteinte, 1 led allumée).

La dernière étape est de recompiler l'agent NET-SNMP pour intégrer l'extension de sa MIB :

```
# cd ~/net-snmp
# ./configure --with-mib-modules=enseirb
# make
# make install
```

4.3. Tests de l'extension de l'agent SNMP NET-SNMP

Il faut d'abord lancer l'agent SNMP. On pourra utiliser différentes options pour générer des traces.
La commande suivante affiche sur la sortie standard les messages de debug du fichier `enseirb.c` introduits à l'aide de la macro `DEBUGMSGTL()` :

```
# snmpd -L -Denseirb
```

La commande suivante affiche en plus sur la sortie standard les requêtes SNMP reçues et émises par l'agent SNMP à la « `tcpdump` » :

```
# snmpd -Ld -Denseirb
```

Pour piloter les leds du module électronique par SNMP, on utilise les commandes `snmpxxx`.

Etat courant des 8 leds au lancement de l'agent SNMP :

```
# snmpwalk -c tst -v 1 localhost enseirb
ENSEIRB-MIB::led0.0 = INTEGER: 0
ENSEIRB-MIB::led1.0 = INTEGER: 0
ENSEIRB-MIB::led2.0 = INTEGER: 0
ENSEIRB-MIB::led3.0 = INTEGER: 0
ENSEIRB-MIB::led4.0 = INTEGER: 0
ENSEIRB-MIB::led5.0 = INTEGER: 0
ENSEIRB-MIB::led6.0 = INTEGER: 0
ENSEIRB-MIB::led7.0 = INTEGER: 0
```

Etat courant de la led 0 :

```
# snmpget -c tst -v 1 localhost led0.0
ENSEIRB-MIB::led0.0 = INTEGER: 0
```

Led 0 allumée :

```
# snmpset -c tst -v 1 localhost led0.0 i 1
ENSEIRB-MIB::led0.0 = INTEGER: 1
```

Mauvais paramètre. Doit être compris entre 0 et 1 :

```
# snmpset -c tst -v 1 localhost led0.0 i 2
led0.0: Value out of range (2)
```

Etat courant de la led 0 :

```
# snmpget -c tst -v 1 localhost led0.0
ENSEIRB-MIB::led0.0 = INTEGER: 1
```

Led 7 allumée :

```
# snmpset -c tst -v 1 localhost led7.0 i 1
ENSEIRB-MIB::led7.0 = INTEGER: 1
```

Etat final des 8 leds :

```
# snmpwalk -c tst -v 1 localhost enseirb
ENSEIRB-MIB::led0.0 = INTEGER: 1
ENSEIRB-MIB::led1.0 = INTEGER: 0
ENSEIRB-MIB::led2.0 = INTEGER: 0
ENSEIRB-MIB::led3.0 = INTEGER: 0
ENSEIRB-MIB::led4.0 = INTEGER: 0
ENSEIRB-MIB::led5.0 = INTEGER: 0
ENSEIRB-MIB::led6.0 = INTEGER: 0
ENSEIRB-MIB::led7.0 = INTEGER: 1
```

Le shell script `k2000snmp.sh` permet de réaliser un luxueux chenillard à la K2000 avec SNMP :

```
# k2000snmp.sh
```

4.4. Génération de TRAPs SNMP

Il est aussi possible d'émettre des TRAPs SNMP depuis l'agent NET-SNMP. Dans le fichier `enseirb.c`, lorsque la led0 passe d'allumée à éteinte, l'agent SNMP émet un TRAP de numéro spécifique 100 en direction des managers SNMP définis dans le fichier de configuration `snmpd.conf`. On utilise pour cela la fonction `send_easy_trap()` de l'API NET-SNMP.

La trace `tcpdump` générée est la suivante :

```
14:59:01.316389 poire1.1090 > poire1.snmptrap:  [udp sum ok] |30|27|02|01{
SNMPv1 |04|03C=tst |a4|1d{ Trap(29)  |06|09E:8072.4.0 |40|040.0.0.0|02|01
enterpriseSpecific|02|01[specific-trap(100)!=0] |43|0213784|30|00} }  (DF)
(ttl 64, id 0, len 69)
```

La commande `snmptrapd` (démon) permet de collecter les TRAPS SNMP reçus. L'exemple suivant permet de les afficher à l'écran :

```
# snmptrapd -P
2002-07-04 14:56:29 NET-SNMP version 5.0.1 Started.
2002-07-04 14:56:56 poire1 [udp:127.0.0.1:1089]:
      RFC1213-MIB::sysUpTime.0 = Timeticks: (1292) 0:00:12.92    SNMPv2-
MIB::snmpTrapOID.0 = OID: NET-SNMP-MIB::netSnmpNotifications.0.100
2002-07-04  14:56:56  0.0.0.0(via  udp:127.0.0.1:1090)  TRAP,  SNMP  v1,
community tst
      NET-SNMP-MIB::netSnmpNotifications Enterprise Specific Trap (100)
Uptime: 0:00:12.92
```

5. CONTROLE PAR SNMP D'UN SYSTEME EMBARQUE SOUS µCLINUX

On a choisi ici d'étendre l'agent NET-SNMP porté sous µClinux pour les processeurs Motorola ColdFire. Le projet µClinux est un Linux embarqué qui a été présenté dans le numéro de février 2002 de Linux Magazine. On utilise ici une carte d'évaluation Motorola ColdFire 5407C3.

L'extension de l'agent sous µClinux se fait comme sur Linux.
On désire ici piloter par SNMP les leds de la carte d'évaluation.
Pour cela, on développe un fichier MIB. On utilise le même fichier `MIB-ENSEIRB.txt` puisque la carte d'évaluation possède 8 leds.
On installe maintenant sous Linux NET-SNMP version 4.2.1 car la version de NET-SNMP portée sous µClinux est actuellement la version 4.2.1.

On utilise ensuite l'outil `mib2c` sous Linux pour créer les fichiers squelettes `enseirb.c` et `enseirb.h` qui seront modifiés en conséquence pour contrôler les 8 leds de la carte Motorola.
Les fichiers ainsi créés sont recopiés dans la distribution µClinux sous `~/µClinux/user/ucdsnmp/agent/mibgroup`.

La seule difficulté est ensuite de modifier les fichiers `Makefile` et header `.h` sous `~/µClinux/user/ucdsnmp` pour prendre en compte la compilation des fichiers `enseirb.c` et `enseirb.h`.

Il ne reste plus qu'à générer le noyau µClinux en incluant le package ucdsnmp, le télécharger dans la carte d'évaluation puis tester depuis Linux comme précédemment.

On voit ainsi un des intérêts majeurs du package NET-SNMP : il est facilement portable pour être intégré à un Linux embarqué et donc être embarqué sur un système électronique pour un contrôle à distance par SNMP…

6. SNMP ET LA SECURITE

On ne peut pas dire que SNMP soit un protocole sécurisé avec notamment SNMPv1. Il faut bien voir que la communauté (*community*) est à considérer comme un mot de passe qui circule en clair sur le réseau (voir les traces `tcpdump` précédentes).

Comme beaucoup de protocoles et notamment les protocoles Internet, on s'est surtout attaché historiquement à mettre d'abord en place un système de transmission de données en dehors de tout critère de sécurité et de confidentialité.

SNMPv1 n'est donc pas sécurisé. Malheureusement, c'est la seule version de protocole SNMP admise et supportée par tous les équipements.
En conséquence, si l'on n'a pas besoin d'un agent SNMP sur sa machine ou son système électronique, on n'activera pas ce service par défaut. Dans le cas contraire, il faudra bien vérifier son fichier de configuration (`snmpd.conf` pour NET-SNMP) et adopter la politique du « tout interdire sauf » et non pas du « tout autoriser sauf » pour ne pas laisser des trous de sécurité. Il faudra aussi changer les noms de communauté par défaut qui sont généralement `public` pour l'accès aux objets en lecture de la MIB de l'agent et `private` pour l'accès aux objets en écriture. Si ce n'est pas le cas, un cracker pourrait très facilement arrêter à distance par SNMP une machine, voire pire ! On a vu précédemment qu'il est envisageable de contrôler des systèmes électroniques par SNMP : éteindre une led ou éteindre une machine par SNMP revient à la même chose…

7. MANAGERS SNMP

Il existe des managers SNMP ayant une interface graphique conviviale pour administrer un réseau par SNMP. Il y a le grand classique Openview de HP dans le domaine commercial. Un manager intéressant dans la logiciel libre à citer est Loriot développé par Ludovic Lecointe. Ce manager SNMP fonctionne sous Win32 et est vraiment très complet. On peut citer les fonctionnalités suivantes :

- Recherche d'agents SNMP sur des plages d'adresses IP.
- Gestion du réseau sous forme graphique.
- Browser de MIB.
- Récupération des TRAPs SNMP.
- Envoi de requêtes SNMP GET, GETNEXT, SET…
- …

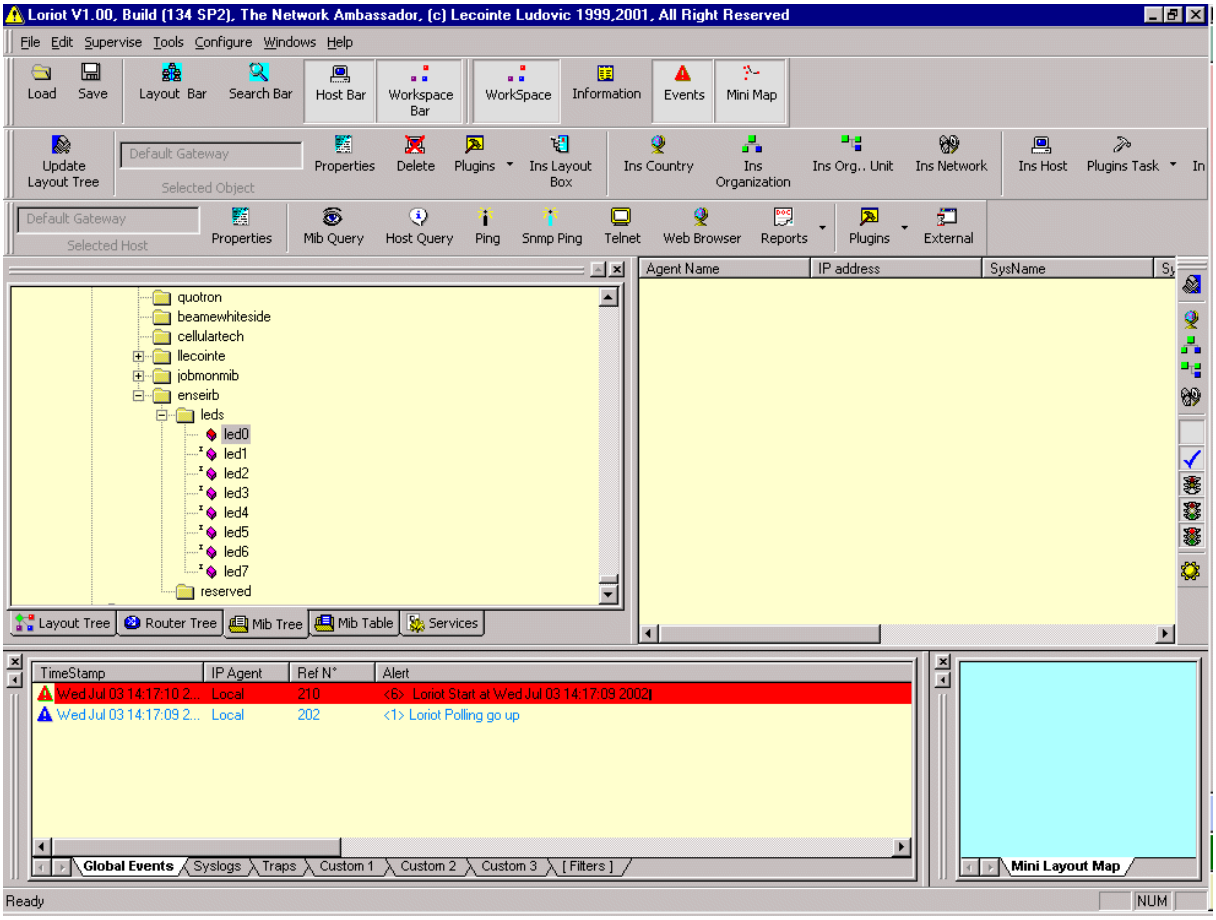

Figure 11 : manager SNMP Loriot

Il est fourni avec le package NET-SNMP un browser de MIB tkmib développé en Tk qui permet de visualiser sous forme graphique les fichiers MIB de `/usr/local/share/snmp/mibs`. On peut aussi émettre des requêtes de base SNMP.

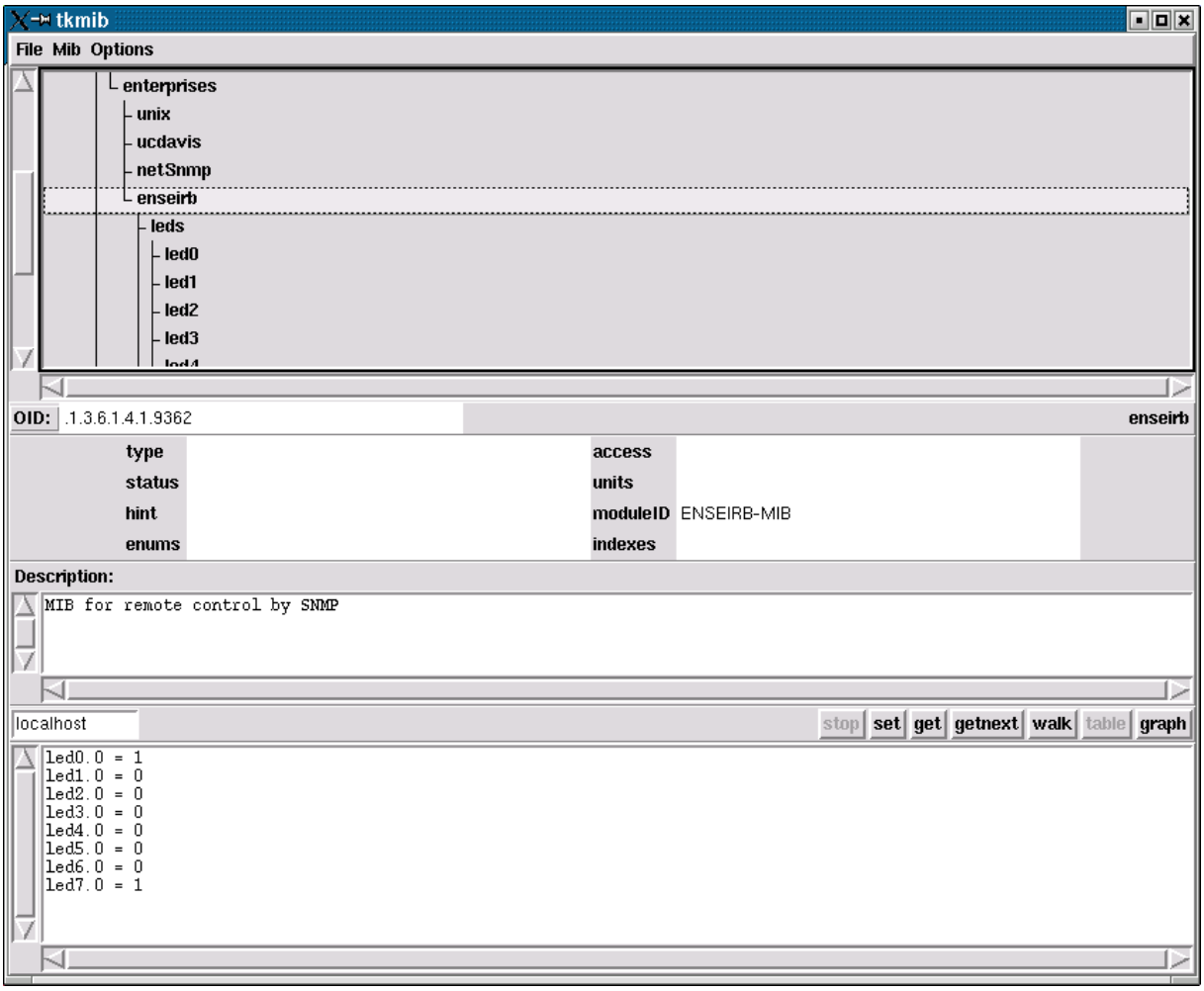

Figure 12 : browser de MIB tkmib de NET-SNMP

Il existe une solution plus complète sous Linux appelé Scotty. C'est en fait une extension Tcl pour l'administration de réseau.
Tkined est une interface graphique Tk utilisant Scotty permettant d'administrer un réseau comme Loriot. Il fait partie du package Scotty.

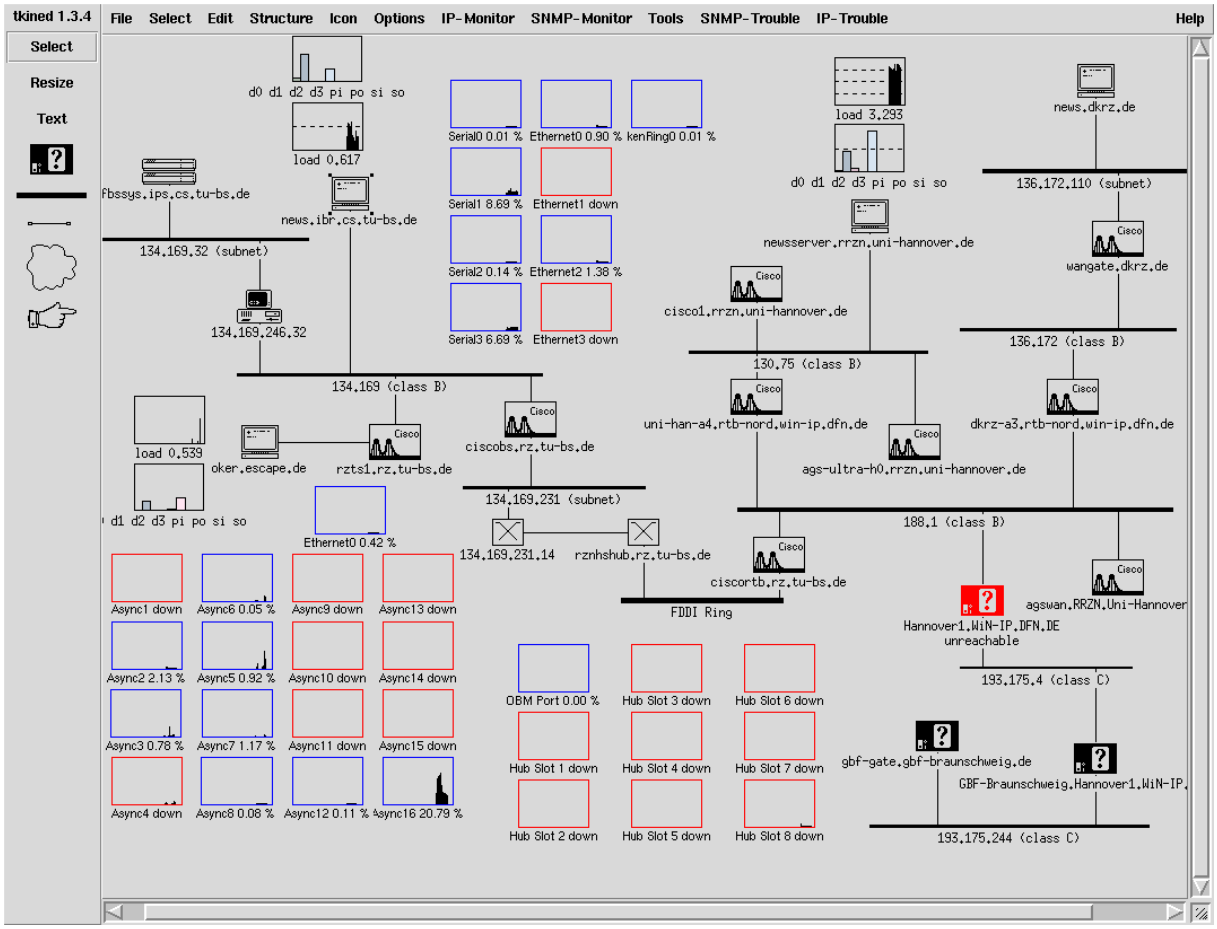

Figure 13 : manager SNMP Tkined

8. CONCLUSION

On a pu voir à travers cet article l'importance de l'administration de réseau qui est une des tâches qui incombe généralement à l'administrateur système pour un réseau local. Pour un grand opérateur, la tâche est très ardue car il lui est essentiel de garder son réseau en état de marche et d'intervenir au plus vite en cas de panne.

Le protocole SNMP a été développé pour faciliter l'administration de réseau. On a pu voir que grâce à des requêtes SNMP simples (GET, SET…) et la remontée d'informations par TRAPs SNMP, on pouvait maintenir son réseau. On a aussi vu la mise en œuvre du standard de fait dans ce domaine : le package NET-SNMP.

Dans le cas des systèmes embarqués, on a présenté comment étendre l'agent SNMP NET-SNMP pour contrôler à distance via SNMP un système électronique que ce soit sous Linux ou bien avec µClinux comme Linux embarqué. C'est une méthode un peu moins classique qu'utiliser un serveur web embarqué. Cela permet surtout d'intégrer le système électronique à un réseau et de l'administrer avec des managers comme HP Openview. Le manager SNMP est la pièce maîtresse dans l'environnement SNMP. Il est clair qu'il doit intégrer des fonctions d'expertise pour une aide à la prise de décision lorsqu'il y a beaucoup d'informations (alarmes, polling…) à traiter dans le cas d'un réseau de grande envergure.

SNMPv1 est la version du protocole SNMP supporté et adopté par tous. Il possède de gros trous de sécurité. En conséquence, on n'activera jamais un agent SNMP pour le plaisir et à la légère sous peine de voir un jour sa machine s'arrêter subitement sans explication !

L'ensemble des schémas du module électronique et les programmes utilisés dans cet article sont disponibles à l'adresse WWW suivante : http://www.enseirb.fr/~kadionik/embedded/snmp/annexe.html

9. REFERENCES

Littérature :

- Les RFC disponibles à http://www.rfc-editor.org/
  RFC1089. SNMP over Ethernet
  RFC1156. Management Information Base Network Management of TCP/IP based internets
  RFC1157. SNMP : Simple Network Management Protocol
  RFC1158. Management Information Base Network Management of TCP/IP based internets: MIB-II
  RFC1212. Concise MIB definitions.
  RFC1213. MIB-II : Management Information Base for network management of TCP/IP based internets
  RFC1215. Convention for defining traps for use with SNMP
  RFC1270. SNMP communications services
  RFC1303. A Convention for Describing SNMP-based Agents
  RFC1351. SNMP Administrative Model
  RFC1352. SNMP Security Protocols
  RFC1353. Definitions of Managed Objects for Administration of SNMP Parties
  RFC1442. Structure of Management Information for SNMP version 2
  RFC1443. Textual Conventions for SNMP version 2
  RFC1445. Administrative Model for SNMP version 2
  RFC1446. Security Protocols for SNMP version 2
  RFC1447. Party MIB for SNMP version 2
  RFC1448. Protocol Operations for SNMP version 2
  RFC1449. Transport Mappings for SNMP version 2
  RFC1450. Management Information Base for SNMP version 2

- Gestion des réseaux ouverts : SNMPv2. M. Rose. Editions Interéditions. LE livre de référence sur SNMP.
- Télécom 2. De l'ingénierie aux services. C. Servin. Editions Dunod.
- TCP/IP Illustré. Les protocoles. Volume 1. W. Stevens. Editions Vuibert.

- Administration de réseau. Cours en ligne. http://netman.cit.buffalo.edu/index.html
- Présentation de SNMP par guill.net. http://www.guill.net/reseaux/Snmp.html
- SNMP for dummies. Société NETCOM. http://www.netcom-sys.com/techdocs.html

Portails et adresses web sur SNMP :

- Le portail SimpleWeb.org. Tout sur SNMP. http://www.simpleweb.org/
- Le portail snmpinfo.com. http://www.snmpinfo.com/
- SNMP sous Linux. http://linas.org/linux/NMS.html
- La page Télécom de l'auteur. http://www.enseirb.fr/~kadionik/telecom/telecom.html

- La page Systèmes embarqués de l'auteur. http://www.enseirb.fr/~kadionik/embedded/embedded.html
- Liste des OID réservés de la branche private.enterprises. http://www.iana.org/assignments/enterprise-numbers
- Demander un OID dans la branche private.enterprises à l'IANA. http://www.iana.org/cgi-bin/enterprise.pl

Logiciels :

- Le package NET-SNMP. http://net-snmp.sourceforge.net/
- Les sources de tous les programmes et matériels décrits dans cet article. Testés avec RedHat 7.3 et µClinux version 20010602. http://www.enseirb.fr/~kadionik/embedded/snmp/annexe.html
- HOWTO sur l'extension de l'agent NET-SNMP v4.x.x. http://www.csc.liv.ac.uk/%7Edaves/Misc/UCD/guide.html
- Le manager SNMP Loriot sous Windows. http://www.llecointe.com/
- Le manager SNMP Scotty sous Linux. http://wwwhome.cs.utwente.nl/%7Eschoenw/scotty/