

Télécommunications

Réseaux et Services : SNMP

06.07.2001

Auteur : Markus Jatton, prof. Télécom, eivd

1.0 Buts de l'exercice

Il s'agit de se familiariser avec les possibilités du protocole SNMP, qui permet relativement simplement de remplir des tâches de gestion élémentaires, et d'écrire une petite application de gestion de réseau utilisant SNMP.

- Durée : 8-12 heures
- Prérequis : un minimum de connaissances en Java
- Outils : JDK1.3 ou mieux, Stack SNMP de Westhawk, Linux, browser MIB SNMP (si disponible), navigateur Web, Tableur, RFC 1156, 1213, 2274

2.0 snmpwalk

L'utilitaire `snmpwalk` est installé sur les machines Linux du laboratoire B03, et permet d'interroger, à l'aide de requêtes SNMP GET NEXT la MIB (Management Information Base) d'un périphérique ou d'un élément de réseau quelconque. `snmpwalk` donne les noms des identificateurs de la MIB en se basant sur la RFC 1156, dont il existe d'ailleurs une version plus récente connue sous le nom de MIB-2, ou RFC 1213 (références, voir annexe). Ces deux RFC peuvent être trouvées sur Internet. La cible de notre examen sera le MSS (<MSSI-PADD>).

`man snmpwalk` permet de comprendre le fonctionnement de l'utilitaire, ainsi que la commande `snmpwalk` sans paramètres.

2.1 Structure de la MIB

Une MIB est partagée en groupes, qui classifient les objets gérés par domaines d'intérêt. La majeure partie des objets d'une MIB est définie de manière standard, par les RFC citées plus haut. Bien qu'il soit possible à tout un chacun d'étendre la définition d'une MIB en y ajoutant ses propres objets, définis en utilisant la syntaxe ASN.1, les objets définis de manière standard dans RFC 1156 ou RFC 1213 sont dans la majeure partie des cas suffisants pour remplir les besoins du gestionnaire de réseau.

2.2 Quelques groupes intéressants

Les MIB indiquées peuvent être trouvées sur Internet; les premières pages décrivent les divers groupes d'objets contenus dans la MIB; noter que certains groupes sont mandatoires (system), alors que d'autres peuvent être optionnels. Il en va de manière similaire des objets contenus dans un groupe. Si vous définissez votre propre groupe, il est évident qu'il devra avoir un statut "optional".

2.2.1 System

Quelques variables donnant des informations globales sur l'agent de ce système. Mandatory.

2.2.2 Interfaces

Nombre d'interfaces sur lesquels des paquets IP peuvent être transmis ou reçus. Mandatory.

2.2.3 Interface Table

Pour chaque interface, il y a une entrée IFEntry dans IFTable le décrivant. Mandatory.

2.2.4 Address Translation

Correspondance entre adresses physiques et adresses réseaux pour chaque interface (typiquement, IP - MAC). Mandatory.

2.2.5 IP

Variables concernant plus précisément le protocole IP. Mandatory.

2.2.6 IP Address

Informations spécifiques aux adresses IP de la machine; il y a une entrée dans ipAddrTable par adresse IP. Mandatory.

2.2.7 IP Routing

Pour chaque route connue dans ce système, il y a une entrée de type `IpRouteEntry` dans `ipRoutingTable` le décrivant. `Mandatory`.

2.2.8 ICMP

Statistiques d'entrées-sorties ICMP. `Mandatory`.

2.2.9 TCP

Informations sur le protocole TCP; les informations relatives à des connexions particulières sont volatiles. `Mandatory` pour les systèmes implémentant TCP.

2.2.10 Autres tables

Les autres tables concernent des protocoles particuliers, qui sont `Mandatory` si le protocole est implémenté (UDP, EGP, EGPNighbor, etc...).

2.3 Les versions du protocole

2.3.1 Version 1

La version 1 est la version de base. En dépit de sa relative simplicité, elle reste la plus utilisée pour les tâches de gestion de réseau. Sa description complète peut être trouvée à la référence suivante :

<http://www.snmp-products.com/tutorials/snmpv1.html>

La RFC 1156, qui décrit la MIB 1 peut être trouvée sur le site suivant (et sur d'autres très probablement) :

<http://community.roxen.com/developers/idocs/rfc/rfc1156.html>

2.3.2 Version 2

La version 2 ajoute de nouveaux types de données, une meilleure stratégie de protection contre les accès en écriture, une protection contre les accès simultanés par de multiples managers, et corrige quelques incohérences de la version 1. Elle est toutefois compatible avec la version 1.

La RFC 1213, qui décrit la MIB 2 peut être trouvée sur le site suivant (et sur d'autres très probablement) :

```
http://www.cis.ohio-state.edu/cgi-bin/rfc/  
rfc1213.html
```

2.3.3 Version 3

SNMP v3 étend les concepts de sécurité en introduisant le cryptage de l'information, et l'authentification. Elle est décrite par RFC 2274 :

```
http://www.cis.ohio-state.edu/cgi-bin/rfc/  
rfc2274.html
```

3.0 Manipulation

Dans un premier temps, on listera les variables système du MSS, avec `snmpwalk <MSSIPADD> public system`.

3.1 Accès à toute la MIB public

Utiliser `snmpwalk` pour lister le contenu de la MIB complet sur l'écran. Quelles sont les entrées principales de l'arbre constituant la MIB (du moins celles identifiables) ?

3.2 Interprétation

En utilisant RFC 1156, expliquer comment on peut découvrir la correspondance entre une adresse MAC et une adresse IP. Ecrire un script en langage Shell Linux (bash) permettant de simplifier cette conversion (sans toutefois l'automatiser).

3.3 Protocoles de routage

Quels sont les protocoles de routage utilisés par le MSS (distance vector, link state, ...?).

3.4 Trafic

Enumérer les diverses informations qui vous permettraient de connaître le trafic global écoulé par un élément de réseau, ainsi que la distinction de ce trafic par interface.

4.0 Le stack de Westhawk

Il existe plusieurs piles de protocole SNMP en Java, qui bâtissent sur java.net pour implémenter un stack indépendant du système d'exploitation. Le plus célèbre de ces stacks est sans doute celui de advent.net. C'est aussi le plus ancien. Pour des raisons purement didactiques, il nous a semblé préférable d'utiliser un autre stack pour ce laboratoire; ceci ne signifie absolument pas que ce dernier doit être considéré comme meilleur ! Il se trouve simplement que le stack de Westhawk semble mieux adapté au programme d'enseignement du langage Java que nous connaissons à l'eivd.

4.1 La société Westhawk

Westhawk est une entreprise basée au Royaume-Uni, qui a mis sur pied un stack agréable à utiliser, en langage Java.

Ce stack a été créé pour la génération de *petites* applications. En conséquence, il n'intègre pas d'explorateur de MIB et permet d'être intégré même à des applets, pour lesquelles de très nombreux exemples sont joints à la distribution.

Une description du packaging figure avec la distribution de Westhawk, ainsi que l'ensemble de la documentation (parfois succincte) des classes en format javadoc (cf. répertoire javadoc).

Il existe, indépendamment du stack Westhawk, une foule d'explorateurs de MIB (chercher "MIB browser" dans un moteur de recherche, et localiser par exemple le browser de MG-SOFT pour Windows pour obtenir une idée des performances de ces explorateurs). L'explorateur de MIB permet de faciliter l'accès à des entrées individuelles de la MIB. Il devient ainsi assez simple d'interroger individuellement certaines valeurs de la MIB.

4.2 Pourquoi un stack ?

On peut se demander, après avoir utilisé un browser de MIB tel que celui présenté, pourquoi il est encore nécessaire de programmer au niveau du stack SNMP. De fait, la MIB est extrêmement vaste, et une application générique, aussi bien conçue soit-elle, ne peut en aucun cas résoudre tous les cas de figure imaginable. Il est de ce fait nécessaire de programmer à bas niveau pour la mise sur pied d'applications spécifiques.

4.3 PDUs et contexte

Le stack utilisé fonctionne sur la base de deux notions élémentaires :

4.3.1 Contexte SNMP

Le contexte SNMP est un objet fixant l'environnement général du protocole, et est indispensable pour la construction d'une PDU.

4.3.2 PDU

La PDU est l'unité de protocole SNMP (GET, GET-NEXT, SET, etc...). Une PDU peut être grossièrement construite avec la séquence suivante :

```
try {
    context = new SmpContext(host, port, socketType);
    context.setCommunity(community);

    pdu = new OneGetNextPdu(context);
    pdu.addObserver(this);
    pdu.addOid(ipInReceives);
    pdu.send();
}
```

La classe `TcomGetNextTemplate.java` implémente le mécanisme de base de production de PDU, et d'envoi de requêtes SNMP constituées d'un seul OID à la fois.

La classe `TcomGetVecTemplate.java` implémente le mécanisme de base de production de PDU, et d'envoi de requêtes SNMP constituées de plusieurs OID à la fois.

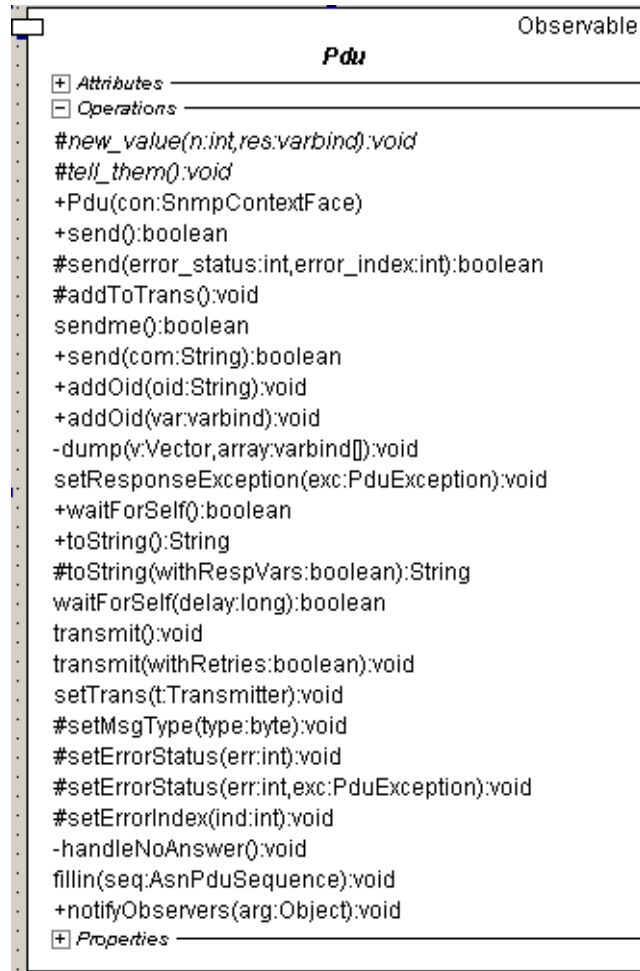
4.3.3 Caractéristiques remarquables de ce programme

Un examen attentif de la classe indique les caractéristiques suivantes :

- La pdu a la possibilité de s'émettre elle-même (`pdu.send()`)

FIGURE 1.

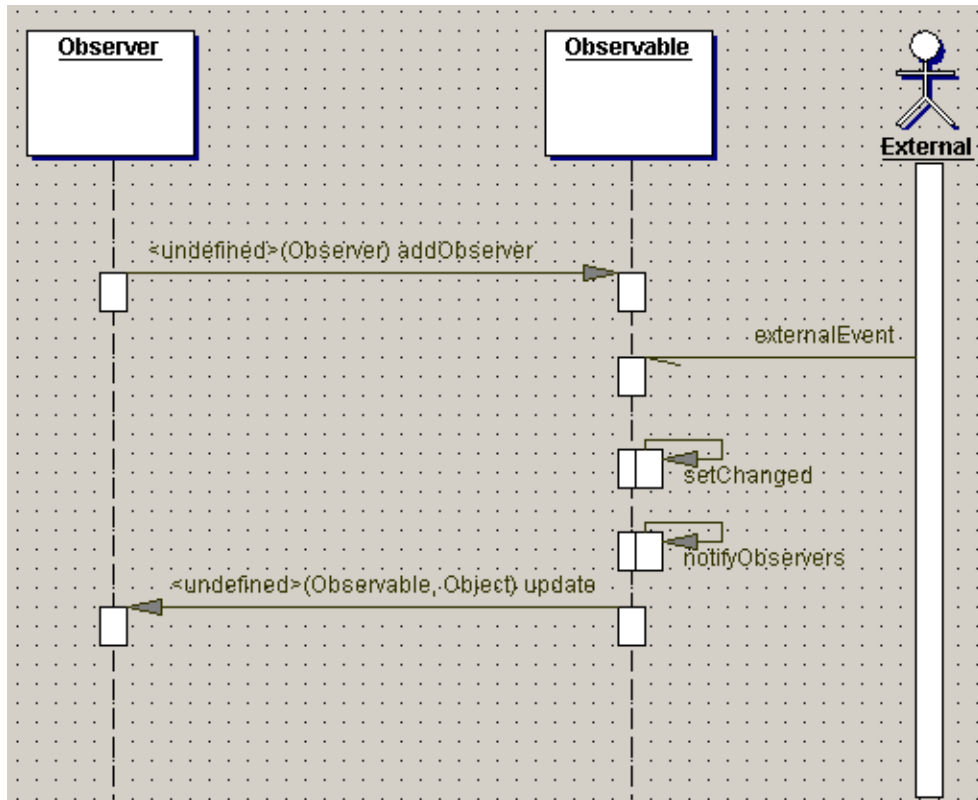
Pdu



- La relation entre PDU et application est du type Observateur Observé (méthode `update(Observable, Object)`).

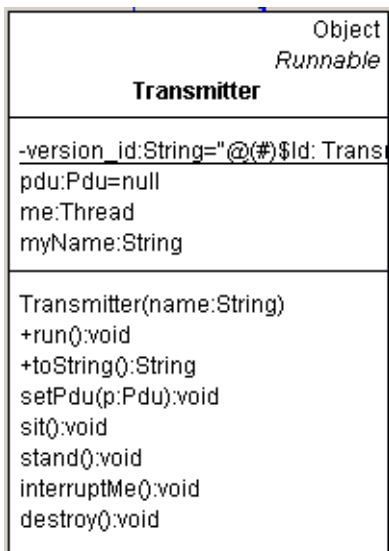
FIGURE 2.

Modèle Observateur-Observé



- La PDU crée son propre Thread d'émission, pour éviter à l'application d'avoir à effectuer le time-out sur le paquet UDP elle-même. (Exception : PDU dites "bloquantes").

FIGURE 3. Transmetteur associé à une PDU



- Les nombres ASN.1 de 32 bit (COUNTER32) peuvent être considérés comme des entiers, du point de vue Java. Mais comme Java ne connaît pas les entiers non signés (`unsigned int` en C), tout nombre de plus de 31 bit est considéré comme négatif; c'est en particulier le cas pour des quantités du type `Counter32`. Deux possibilités pour contourner ce problème : au lieu de forcer le type à `int`, on peut utiliser la classe ASN.1 du stack Westhawk, ce qui constitue la méthode correcte de faire. On peut bien sûr bidouiller aussi en détectant les quantités négatives de types `Counter32`, et en les convertissant dans un `long`.

4.4 snmpwalk en Java

De manière à exercer un peu l'utilisation du stack de Westhawk, on implémentera une version simplifiée de `snmpwalk` en Java, en utilisant et en complétant la classe fournie avec le laboratoire, `Snm-pWalk.class`. Que manque-t-il à cette classe pour implémenter réellement `snmpwalk` ?

5.0 Mise sur pied d'une application de gestion de réseau

5.1 Classe de sortie en format compatible tableur

La classe `ExcelIo` fait partie de la distribution de ce laboratoire, et vous permet de générer des fichiers qui peuvent par la suite être lus simplement par un tableur comme Excel, avec sa fonction d'importation. Il est ensuite possible de manipuler les valeurs avec les fonctions d'Excel, et de générer des graphiques qui pourront à leur tour être importés dans un logiciel de présentation comme Powerpoint. Noter qu'il s'agit là de la principale raison d'être de la gestion de réseau : persuader son chef qu'il faut plus d'équipements ! Blague à part, la gestion de réseau est effectivement avant tout une aide à la décision d'investissement dans un réseau de téléinformatique.

L'ouverture d'un fichier dans un format lisible par un tableur peut se faire de la façon suivante :

```
xio = new ExcelIo();
if (!xio.open("traffic.dat")) {
    System.out.println("Could not open traf-
fic.dat");
    System.exit(0);
}
```

Ensuite on utilisera l'une des méthodes `put()` pour écrire dans le fichier. Un enregistrement doit se terminer par `eor()`, comme dans :

```
int delta; long packets;
Calendar c = new GregorianCalendar();
xio.put(new String(c.get(Calendar.HOUR) + ":" +
    c.get(Calendar.MINUTE) + ":" +
    c.get(Calendar.SECOND)));
xio.put(packets); xio.put(delta); xio.eor();
```

Le fichier est fermé par `xio.close()`.

6.0 Microprojets

Ces diverses questions représentent des applications de gestion de réseau; il n'est pas nécessaire (et probablement pas possible, ou alors vous êtes très fort) de les implémenter toutes dans le temps imparti pour ce laboratoire; aussi chaque groupe implémentera-t-il l'un de ces microprojets, et il le présentera aux autres groupes en fin de labora-

toire, au cours d'une brève présentation de cinq à dix minutes, accompagnée d'une démonstration.

La difficulté de ces miniprojets est inégale; il en sera bien sûr tenu compte, et que ceux qui ont hérité d'un sujet facile ne se contentent pas d'une approche minimaliste ! Chacun est au contraire encouragé à imaginer d'autres applications, ou à implémenter un petit interface graphique pour accroître l'utilisabilité de l'outil réalisé.

Certains de ces microprojets sont liés; c'est voulu, et les étudiants ont bien sûr toute latitude pour mettre en commun leurs connaissances.

On utilisera le stack SNMP de Westhawk fourni en annexe. On essayera dans la mesure du possible de concevoir des applications qui ne soient pas bloquées pendant l'échange de PDU SNMP entre manager et agent, mais qui puisse au besoin faire autre chose en parallèle.

6.1 Mesure de trafic

Etablir une statistique des paquets reçus par unité de temps par l'ensemble des interfaces du MSS, de manière à obtenir, sur une feuille Excel ou dans un tableur quelconque, un graphique donnant la fonction de répartition des paquets reçus sur une base de 5 minutes, et le taux d'erreurs correspondant. L'intervalle de temps de la statistique pourra être aussi grand que l'utilisateur peut le désirer.

6.2 Détection des interfaces

Pour une machine définie par son adresse IP, on désire obtenir une liste de :

- tous les interfaces présents
- pour chaque interface, la taille maximale des datagrams admise
- l'état de cet interface
- l'adresse physique correspondant à cet interface

6.3 Contrôle des interfaces

On désire que, pour chaque interface de la machine mesurée, on obtienne dans un tableur :

- le nombre de bytes reçus
- le nombre de paquets erronés reçus
- le nombre de bytes émis

- le nombre de paquets qui auraient dû être transmis, mais dont la transmission a échoué en raison d'erreurs.

On évitera d'envoyer une PDU pour chaque requête d'OID, mais on essaiera de n'émettre qu'une PDU par interface, qui rassemble toutes les informations en un seul paquet UDP. (Indication : voir la classe `Getpdu_vec`, en n'oubliant pas de commencer par interroger le nombre d'interfaces !)

6.4 Adresse MAC - Adresse IP

Concevoir un utilitaire qui permette, à partir d'une adresse MAC, de trouver l'adresse IP correspondante. On essaiera de combiner le résultat avec le microprojet suivant.

6.5 Adresse IP - Adresse MAC

Concevoir un utilitaire qui permette, à partir d'une adresse IP, de trouver l'adresse MAC correspondante. On essaiera de combiner le résultat avec le microprojet précédent.

6.6 Machines gérables par SNMP

Concevoir un utilitaire qui permette, à partir d'un **domaine** d'adresses IP, de déterminer :

- Les machines actives (Note : ceci n'est pas sans autre possible par SNMP : ICMP peut peut-être aider ? Hélas, ICMP n'est pas utilisable en Java; il va donc falloir inventer quelque chose).
- Les machines actives et gérables par SNMP.
- Pour chacune de ces machines, le temps depuis la dernière mise en service et le type de machine.

6.7 Information de routage

Concevoir un utilitaire qui permette, à partir de l'adresse IP d'un élément de réseau, de lister sous une forme aisément interprétable les tables de routage de cet élément.

La sortie inclura toute information de quelque intérêt pour chaque route détectée.

6.8 TCP

Pour une machine donnée par son adresse IP, concevoir un utilitaire énumérant toutes les connexions TCP actives à un instant donné, sous forme (Adresse source / Port source) - (Adresse destination / Port destination) - état (closed, listen, synSent, synReceived, established, finWait1, finWait2, closeWait, lastAck, closing, timeWait); l'état de la connexion, en particulier, devrait apparaître sous forme lisible (en clair !).

6.9 Datagrams UDP

Concevoir et implémenter une petite application qui permette de faire, en permanence, une statistique des paquets UDP reçus par une machine désignée par son adresse IP, ainsi que la proportion de ces paquets n'ayant pas trouvé d'application sur le port UDP auquel ils se destinaient. Une représentation graphique en temps "quasi réel" (disons un "time-lag" de 5 minutes) serait souhaitable.

6.10 Anomalies IP

Les erreurs (surtout les destructions de paquets) peuvent souvent donner une indication quant à la cause de fonctionnements médiocres d'un réseau. Concevoir une petite application qui présente de manière confortable les informations significatives au niveau IP; savoir quelles sont ces informations significatives devrait vous renvoyer à votre cours de téléinformatique...

6.11 Anomalies des interfaces

Pour une machine donnée par son adresse IP, indiquer pour tous les interfaces présents dans cette machine, le nombre de paquets transmis ainsi que la proportion de paquets ayant été jetés (*discarded*).

6.12 Contrôle des interfaces

On désire que, pour chaque machine dans une liste d'adresses IP donnée, on puisse lister les interfaces supportant un protocole de bas niveau basé sur

- csma-cd
- isdn
- lapb

7.0 Annexes

Pour la distribution CD, le CD contient le stack SNMP de Westhawk, ainsi que les principales RFC (seule RFC 1156 est nécessaire pour ce laboratoire)

Pour la distribution dans le cadre des laboratoires réguliers de tcom, le contenu du CD est simplement placé dans un répertoire partagé du serveur Linux principal.