Projet Java: SS Manager

Rapport de la version 2

L .BALBIANI F .EL JAMAL M.TEIKITUHAAHAA

Ce rapport présente les spécifications pour la version 1 du projet SNMP Java de la 2ème année d'ingénieure STRI.

1. Cahier des charges

Voici ce qui est demandé pour la version 1 :

A cette étape on mettra en place une base de données (fichier, BD…) d'information de gestion sur le modèle d'une MIB SNMP.
On ajoutera l'implantation de l'opération « get-next » qui permet de parcourir cette base de données. Cette étape demande d'implanter une sécurisation de l'application répartie sur le modèle de SNMP v1 (communautés) ou de SNMP v3 (comptes utilisateurs). Deux types d'accès sont à prévoir : en lecture seule et en lecture/écriture.

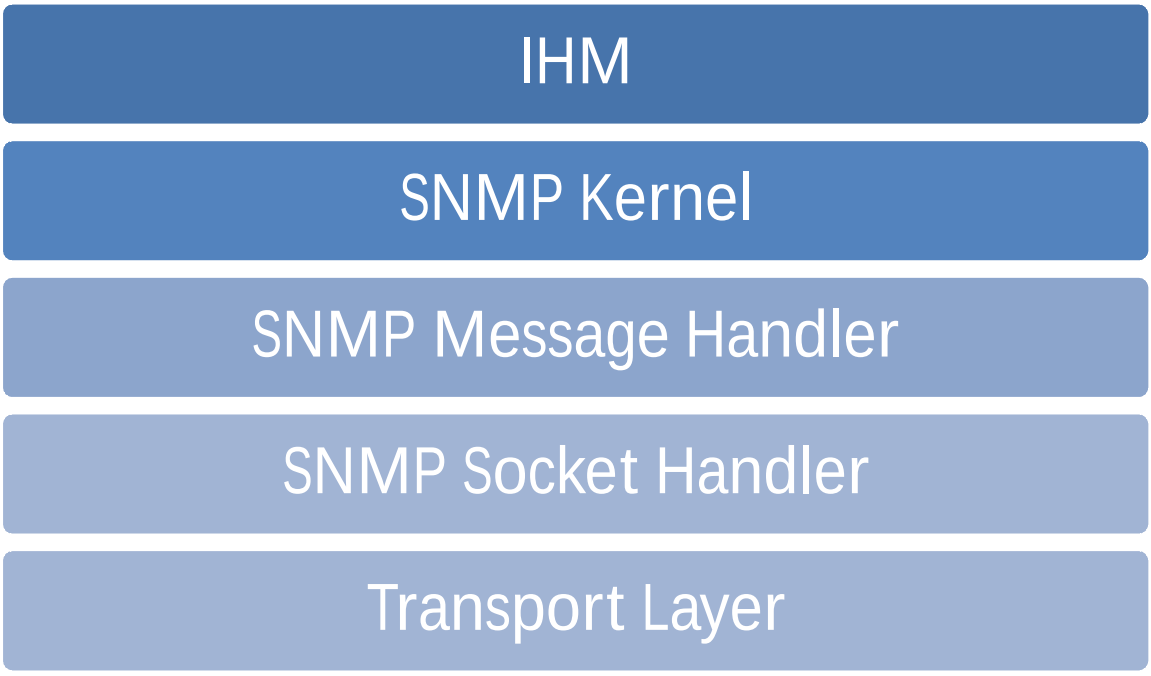
On programmera donc un manager et un Agent SNMP capable d'envoyer ou de recevoir des requêtes GET-NEXT, et respectant la communauté.

2. Architecture du programme

Pour la version 2, on respecte le même modèle en couche que dans la version 1.

1. Modèle en couche

Ce modèle en couche permet de représenté plus facile les ensembles de fonctionnalités de chaque classe. Elle sera valide tout au long du projet.



Les ouches suivantes n'ont les fonctionnalités ci-dessous que pour cette première version. Il se pourrait que par la suite, celles-ci viennent à être modifiées, ou d'autre rajoutée.

IHM

Cette couche représente l'interface graphique du Manager/Agent et a les fonctionnalités suivantes :

- Affichages des résultats des requêtes ;
- Affichages des notifications des agents (TRAP SNMP) ;
- Envoi de requêtes ponctuelles.

SNMP Kernel

C'est le noyau du programme et il est constitué soit du Manager ou de l'Agent SNMP pour cette version. Dans les prochaines versions, on tentera du fusionner ces entités qui devront partager les ressources communes comme les sockets, et faire en sorte de bien router les messages SNMP vers les bonnes entités.

SNMP Message Handler

Cette couche a les fonctionnalités suivantes :

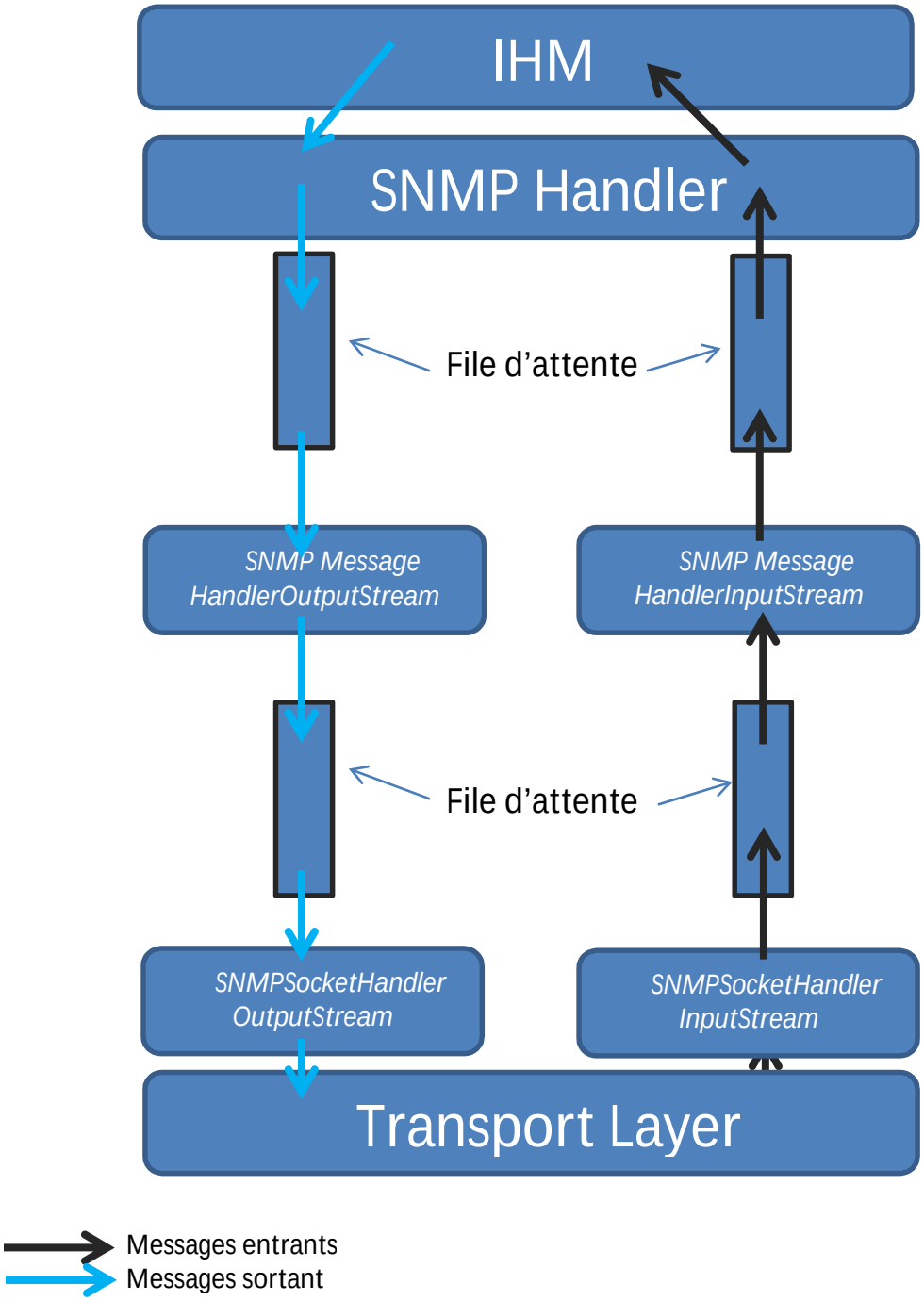
- Conversion des Messages SNMP en *DatagramPacket* et ou en *SNMPMessage* (voir chapitre 3 sur les Diagrammes) ;
- Gestion des retransmissions des Datagrammes (pour les versions suivantes).

SNMP Socket Handler

Cette couche a les fonctionnalités suivantes :

- Récupération ou envoi des Datagrammes vers la couche Transport (UDP). Elle permettra de gérer le taux d'envoi et le taux de réception des requêtes.
- Gestion des sockets.

2. Architecture du Manager version 1



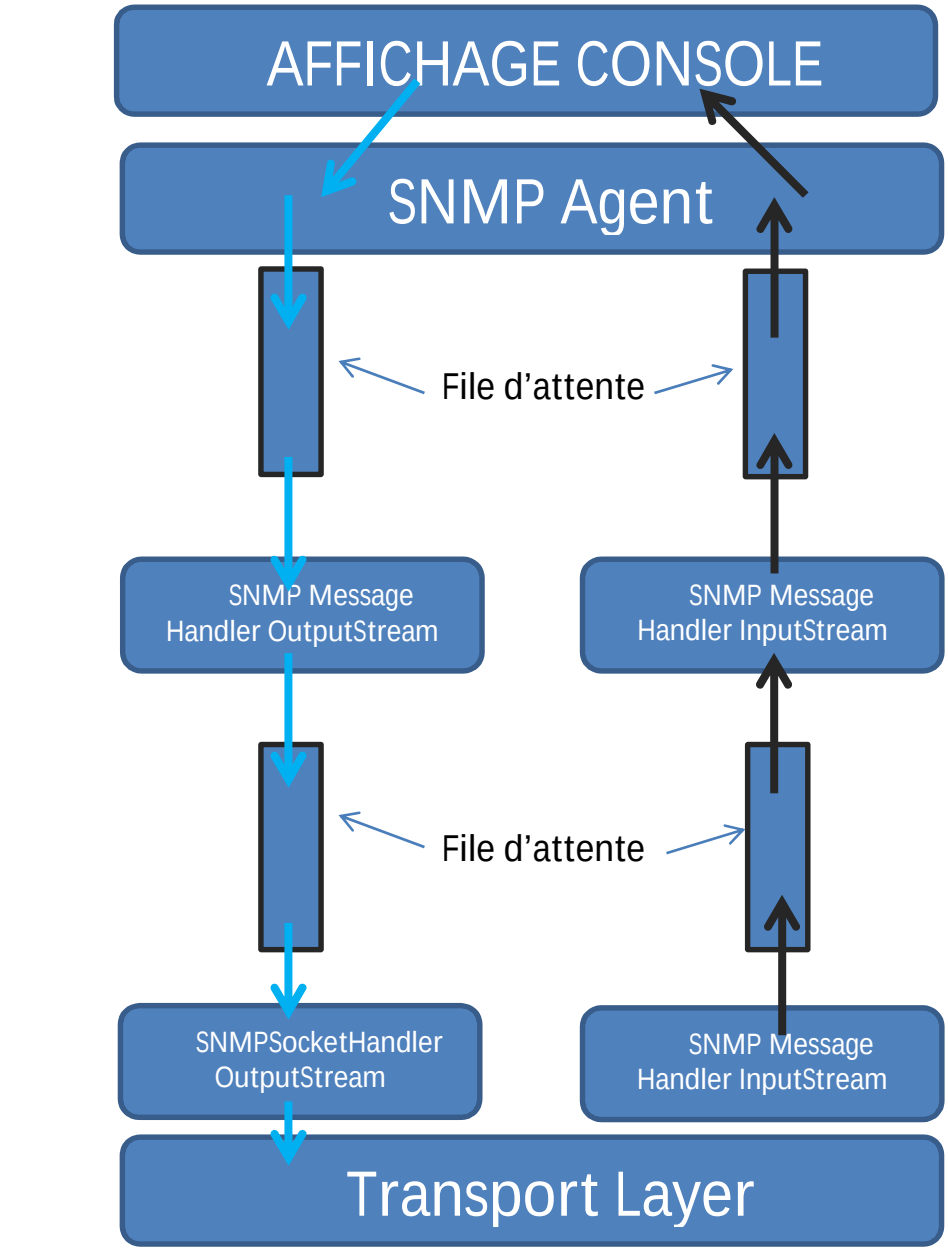
Description :

On identifie bien les couches présentées précédemment. Elles sont liées entre elles avec des files d'attentes de type FIFO. Lorsqu'une PDU SNMP arrive, elle est réceptionnée par la couche *SNMPSocketHandler* puis transférer à la couche supérieure via une première file d'attente, pour y être converti en *SNMPMessage.* Une fois la conversion terminée, elle ce dernier est placé dans une seconde file d'attente pour être traité par un Manager.

Le même processus se passe pour les messages sortants.

Le Manager enverra ses messages sur un numéro de port quelconque et écoutera les TRAP SNMP sur le port 162.

Architecture de l'Agent SNMP



Messages entrants
Messages sortants

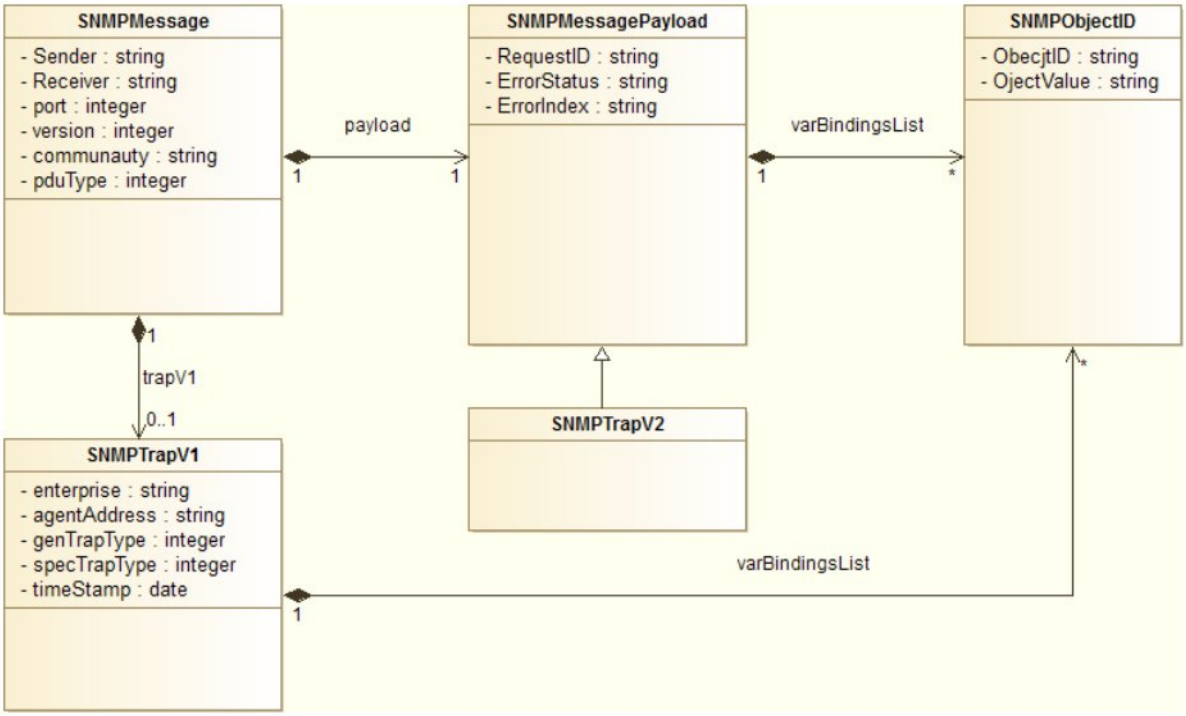
Description :

L'agent utilise suit le même processus de traitement des messages que le manager. Il écoutera les requêtes SNMP sur le port numéro 161, et émettra des TRAP SNMP à destination de son Manager sur le port 162.

Diagrammes

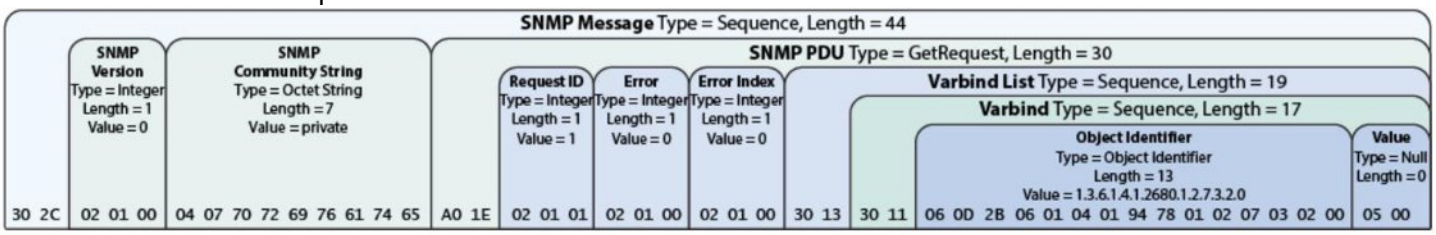
Dans les diagrammes de classe suivant, on ne mettra que les attributs importants. Les méthodes seront développées par chacun selon ses besoins.

SNMP Messages



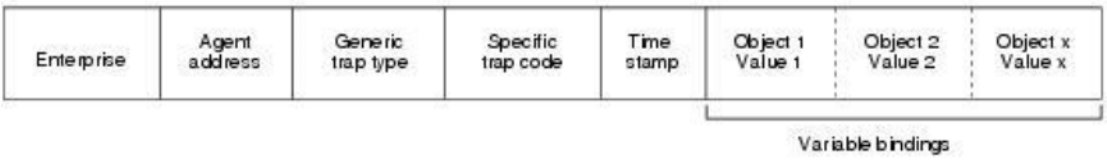
Il s'agit de l'une des parties les plus difficiles du programme : les gestions des messages SNMP. En effet ces derniers sont au format BER (Type length Value). Il s'agit de pouvoir extraire à partir d'un Datagramme, les informations dont nous avons besoins, et de pouvoir les constituer.

Voici un exemple de PDU SNMP :



Les PDU SNMP de la version 1 et version 2c sont identiques. Seul les TRAP SNMP changent.

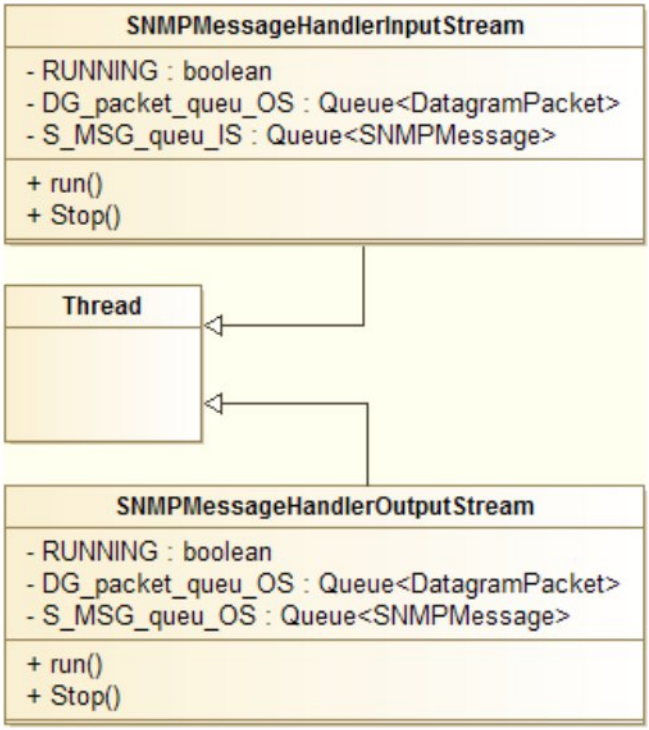
La TRAP SNMP v1 est de la forme suivante :



La TRAP SNMP v2 est de la forme d'un SNMP Message normale avec un PDU Type différent et ayant comme Variable *sysUpTime.0* et *snmpTrapOID.0.* (https://tools.ietf.org/html/rfc3416#page-22)

NB : les types des PDU sont disponibles en annexe. De plus les Traps ne seront pas géré dans cette version. Néanmoins, les classes seront programmées.

SNMP Message Handler

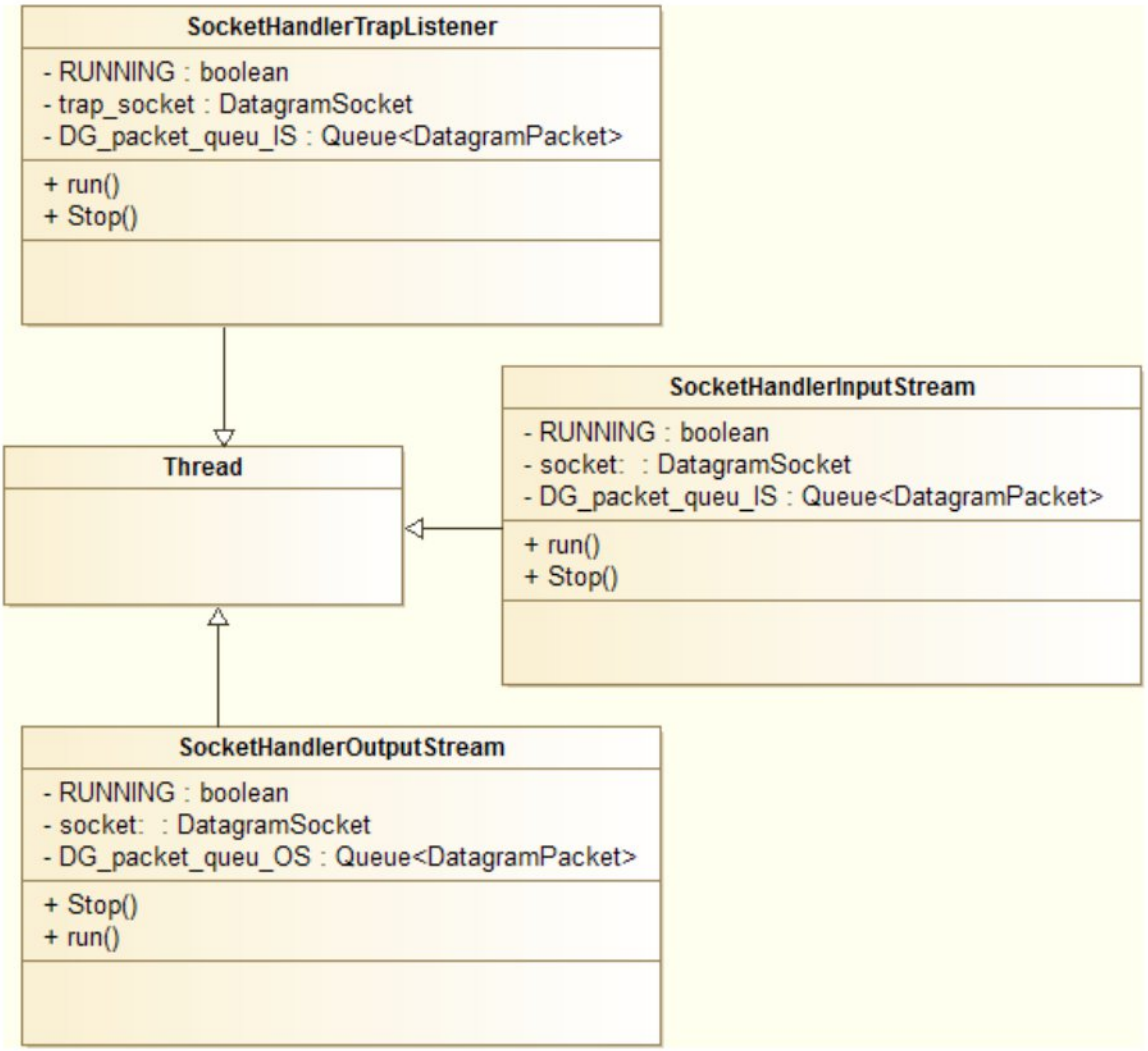


Les *SNMPMessageHandler* sont des Thread qui ne gèrent que la conversion des messages SNMP et des *DatagramPacket*, dans cette première version. La méthode run () contient le code qui sera exécuté par le Thread, et la méthode Stop () permet de stopper le Thread en mettent l'attribut RUNNING à false permettant ainsi au Thread de quitté la boucle infini et de se terminer.

Voici l'algorithme général d'un Thread :

```
// méthode run()
DEBUT THREAD
  WHILE(RUNNING)
  DEBUT
   SI LA FILE D'ATTENTE A LIRE N'EST PAS VIDE ALORS
      DEBUT SI
        ON RECUPERE LE MESSAGE A TRAITER
        ON LE CONVERTI
        ON LE TRANSMET A L'AUTRE FILE D'ATTENTE
      FIN SI
   SINON
        ON ENDORS LE THREAD PENDANT 100ms
  FIN
FIN THREAD
```

SNMP Socket Handler



Les *SNMPSocketHandler* sont des Thread qui ne gèrent l'envoi et la réception des DatagramPacket. La méthode run() contient le code qui sera exécuté par le Thread, et la méthode Stop() permet de stopper le Thread en mettent l'attribut RUNNING à false permettant ainsi au Thread de quitté la boucle infini et de se terminer.

Voici l'algorithme général d'un Thread :

```
// méthode run() pour SocketHandlerOutputStream
DEBUT THREAD
  WHILE(RUNNING)
  DEBUT
   SI LA FILE D'ATTENTE A LIRE N'EST PAS VIDE ALORS
      ON RECUPERE LE DATAGRAMPACKET
      ON LE TRANSMET SUR LE SOCKET
   SINON
         ON ENDORT LE THREAD PENDANT 100ms
  FIN
FIN THREAD
```

```
// méthode run()
// Pour SocketHandlerInputStream et SocketHandlerTrapListener

DEBUT THREAD
  WHILE(RUNNING)
  DEBUT
   LIRESOCKET () // Fonction Bloquante
      ON RECUPERE LE DATAGRAMPACKET
      ON LE TRANSMET A LA FILE D'ATTENTE
  FIN
FIN THREAD
```

SNMP Kernel (version 1)

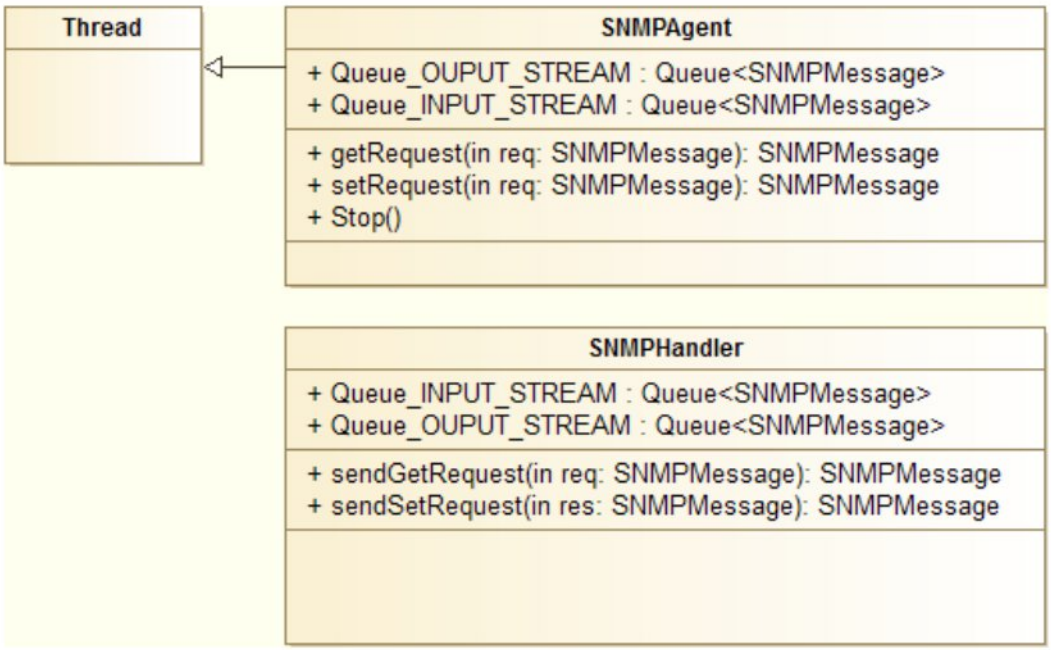


Diagramme de Séquence - Manager: Envoi du requête GET, GET-NEXT et SET (l'envoie d'une requête Set suit le mé

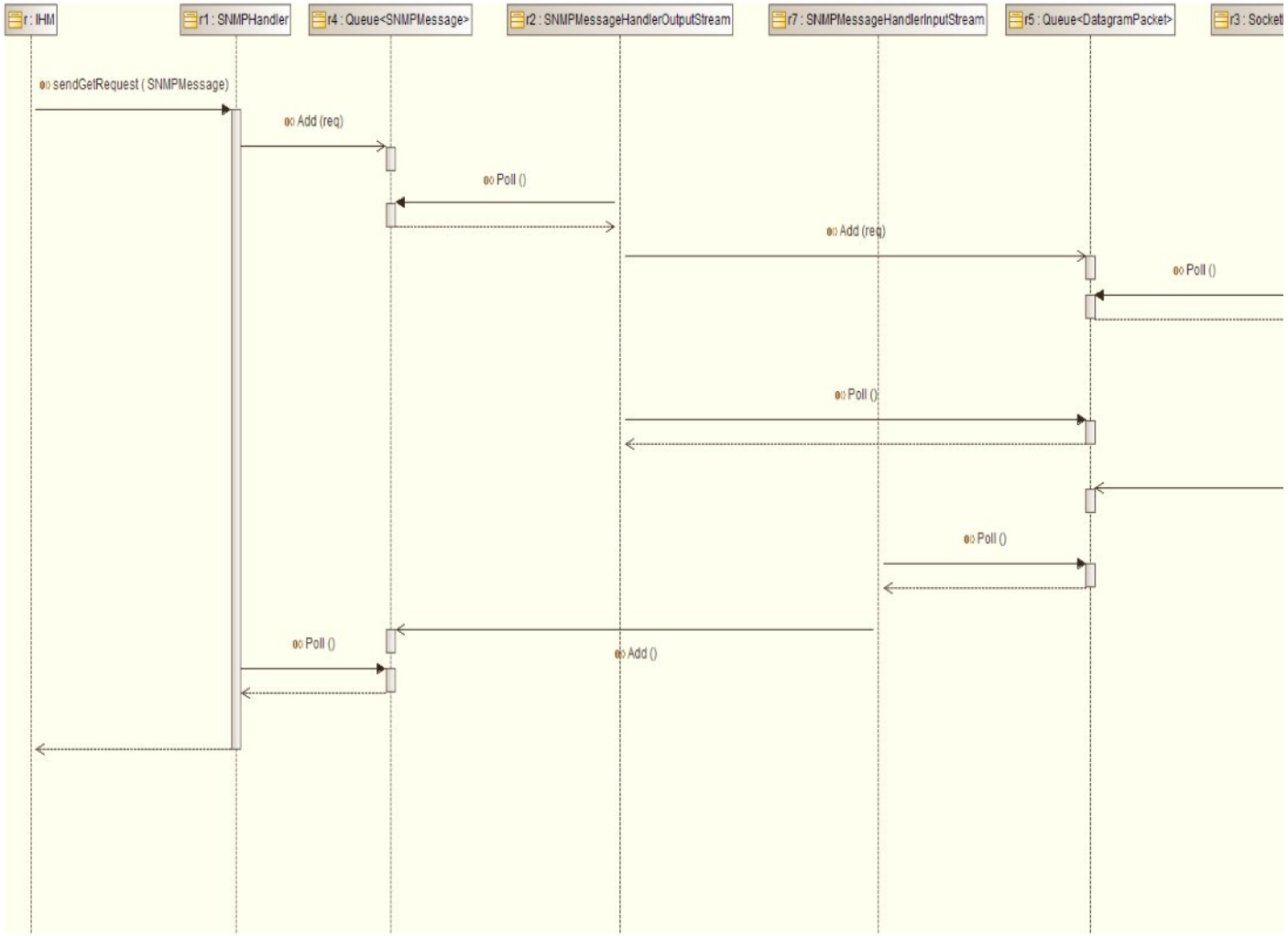
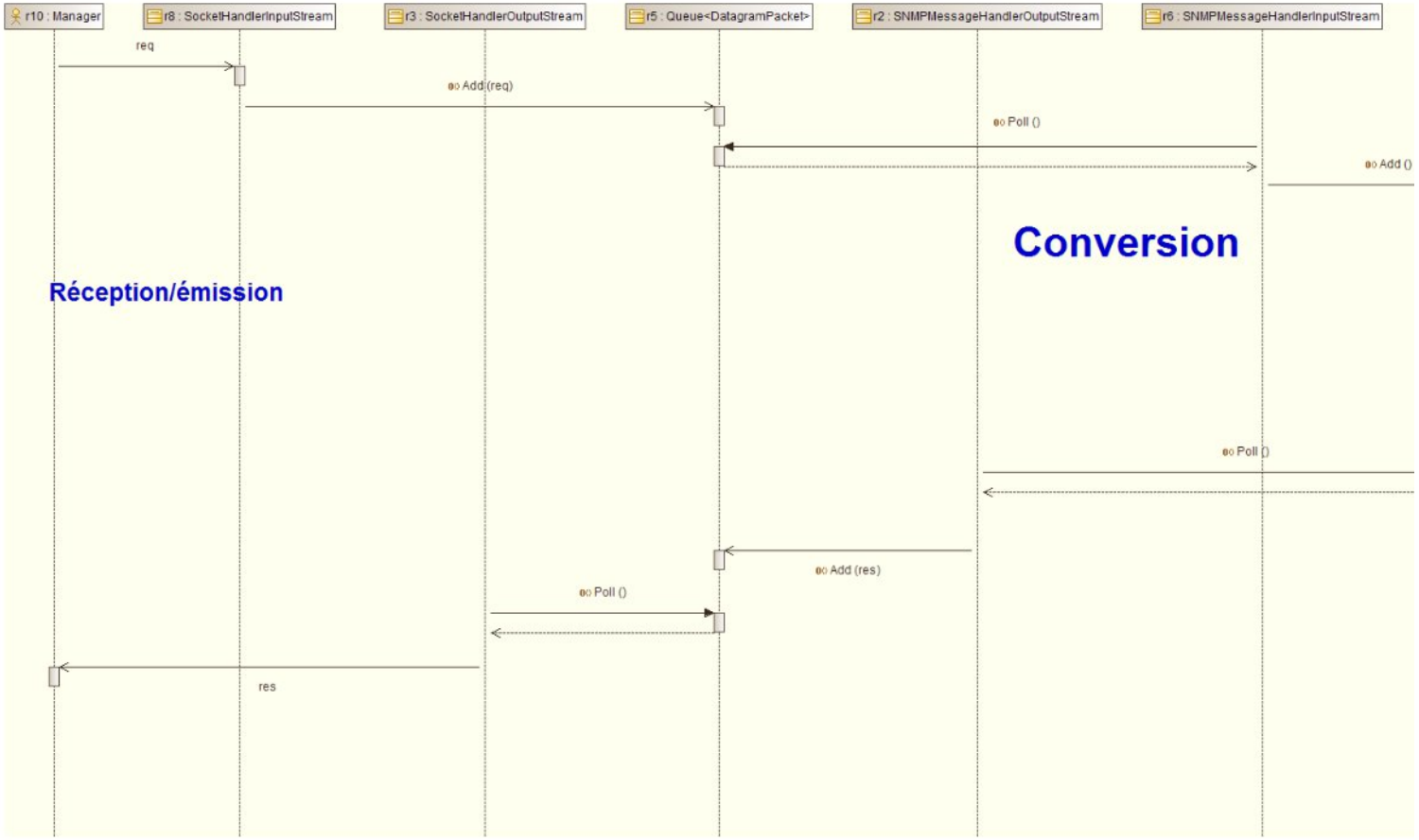


Diagramme de Séquence – Agent: Réception d'une requête GET, SET et GET-NEXT.

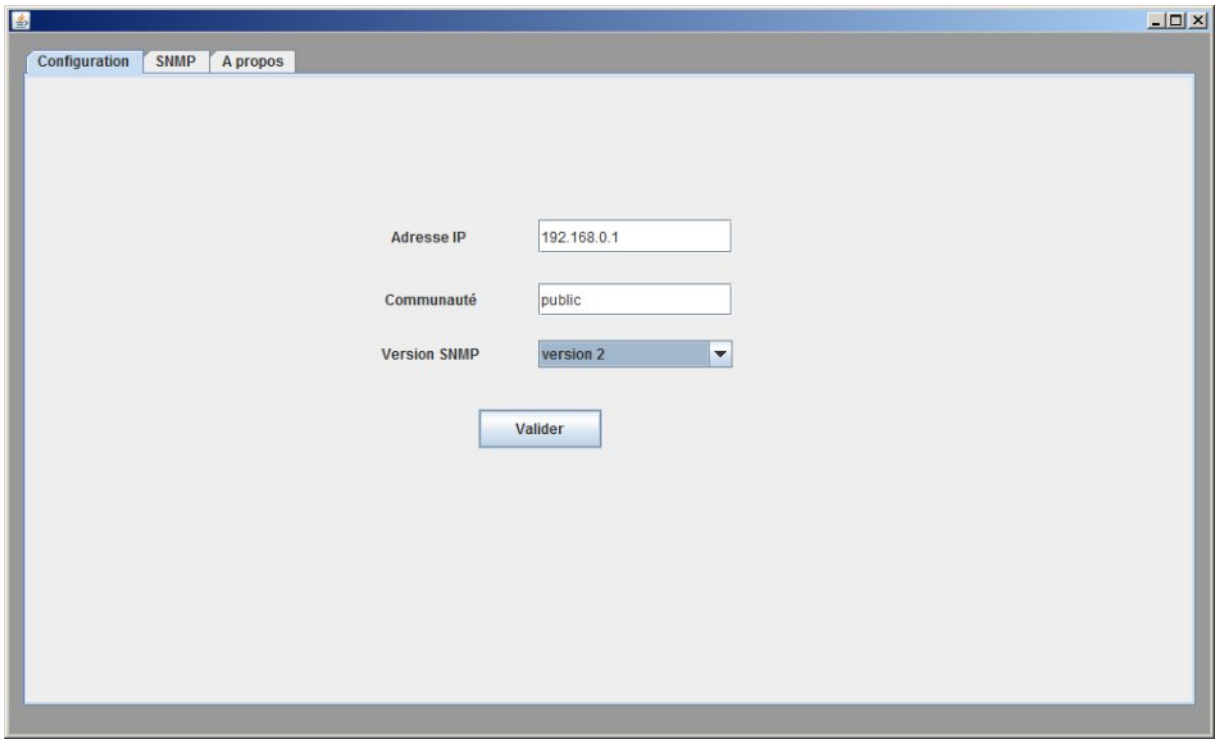


Lorsque l'agent reçoit un requête GET-NEXT, il incrémente l'avant dernière valeur de l'oid pouis le transmet à la fo requêtes GET.

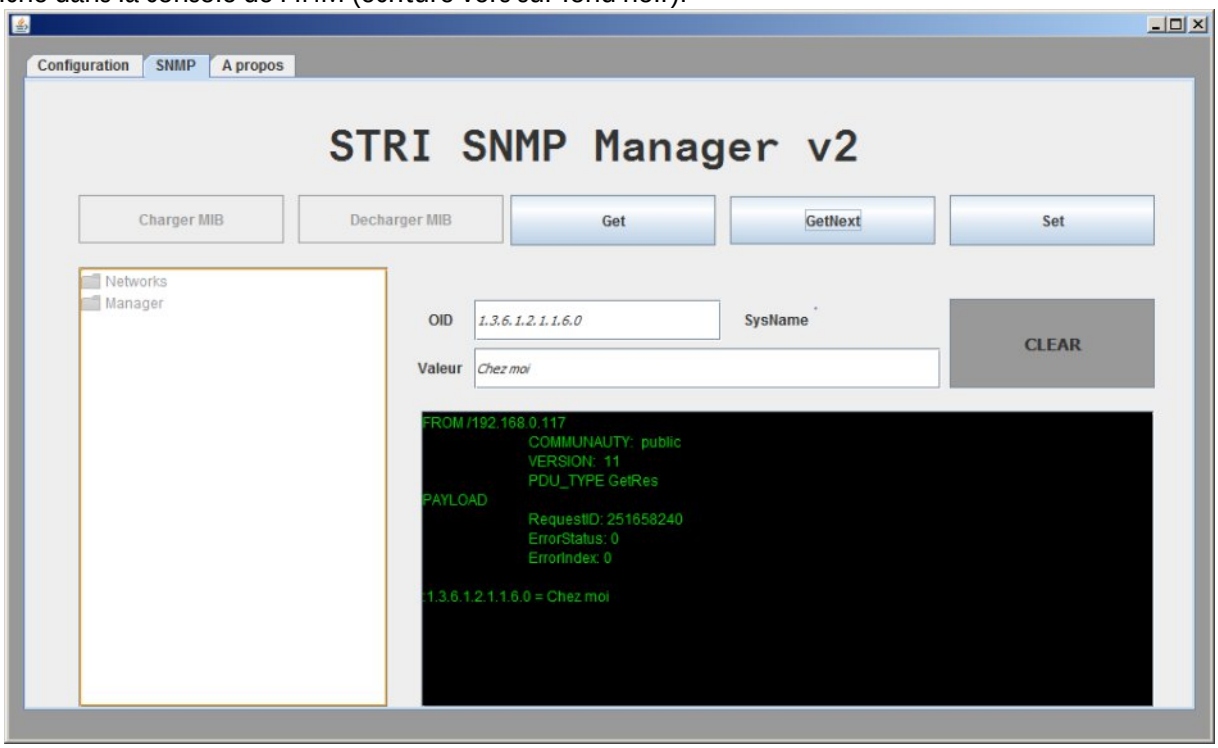
Exemple : 1.3.6.1.2.1.1.5.0 sera incrémenté et deviendra une requête GET avec comme OID 1.3.6.1.2.1.1.6.0 avar traitement de la requête GET.

Description de l'IHM du Manager:

L'utilisateur entre les paramètres (IP, communauté, et le choix de sa version) puis clique sur le bouton valider. S'il ne le fait pas, Les boutons pour envoyer les requêtes restent bloquées.



Lorsqu'un utilisateur souhaite envoyer une requête, il doit remplir le champ OID (et remplir sa valeur en cas d'une requête SET). Une fois l'OID entré, l'IHM se charge de résoudre le nom de l'objet en interrogeant la mib SNMP. Lors de la réception de la réponse, le contenu détaillé de la réponse est affiché dans la console de l'IHM (écriture vers sur fond noir).



Annexe

Liens git hub du projet : https://github.com/xZven/SS_Manager

Type BER :

Primitive Data Types	Identifier	Complex Data Types	Identifier
Integer	0x02	Sequence	0x30
Octet String	0x04	GetRequest PDU	0xA0
Null	0x05	GetResponse PDU	0xA2
Object Identifier	0x06	SetRequest PDU	0xA3

Fichier annexe :
STRI SNMP Manager (à ouvrir avec modelio)
Code source dans un Projet Maven.