



Universidade do Minho

MIETI - 4.º Ano
Gestão de Redes
Trabalhos Práticos 1 e 2

Jorge Bastos
a68456@alunos.uminho.pt



Fevereiro, 2016

Conteúdo

1	Ficha de Trabalho Prático Nº1	6
1.1	Objetivos	6
1.2	Requisitos	6
1.3	Parte 1.A - <i>SNMP</i> no <i>Cisco IOS</i>	7
1.3.1	QUESTÃO TP1.A	7
1.3.1.1	<i>snmp-server view</i>	7
1.3.1.2	<i>snmp-server community</i>	8
1.3.1.3	Resultado Final	8
1.4	Parte 1.B - <i>Net-SNMP</i>	9
1.4.1	QUESTÃO TP1.B	9
1.4.1.1	Questão i. “Qual o valor e significado da instância do objeto com o OID lexicograficamente a seguir a 1.3.6.1.2.1.7.2 da sua estação de trabalho?”	10
1.4.1.2	Questão ii. “Como poderia calcular o número de pacotes IP que entram num router e já não saem (i.e., têm esse router como destino final)?”	10
1.4.1.3	Questão iii. “Quais as partições do sistema de ficheiros da sua estação de trabalho que a instrumentação do agente SNMP consegue identificar?”	11
1.5	Considerações Finais	12
2	Ficha de Trabalho Prático Nº2	13
2.1	Objetivos	13
2.2	Requisitos	13
2.3	Ferramenta de Monitorização do número de pacotes IP	13
2.3.1	Desenvolvimento da Aplicação	14
2.3.1.1	Arquitetura da solução	14
2.3.1.2	Implementação	15
2.3.2	Aplicação	15

2.3.3	Questão 1 - Justifique bem os objetos escolhidos e o intervalo de tempo por defeito escolhido para o <i>polling</i>	16
2.4	Considerações finais	17

Lista de Figuras

1.1	Utilização da primitiva <i>GetNext()</i>	10
1.2	Obtenção dos valores de <i>ipInReceives</i> e <i>ipForwDatagrams</i> . . .	11
1.3	Resultado do comando <i>snmpwalk -v 2c -c public localhost hrStorageType</i>	12
2.1	Interface gráfica da aplicação final.	15
2.2	Output da aplicação final.	16

Lista de Tabelas

1.1	Sintaxe do comando snmp-server view	8
1.2	Sintaxe do comando snmp-server community	8

Introdução

No âmbito da unidade curricular de Gestão de Redes, do curso de Engenharia de Telecomunicações e Informática, foi proposto aos alunos a realização de três trabalhos práticos. Este relatório apenas diz respeito à TP1 e TP2 uma vez que são os únicos trabalhos individuais.

O principal objetivo destes trabalhos práticos, assentam em consolidar os conhecimentos adquiridos na unidade curricular, nomeadamente o INMF (*Internet Standard Management Framework*) que engloba componentes como o protocolo SNMP (*Simple Network Management Protocol*) e as MIBs (*Management Information Bases*).

“Talent hits a target no one else can hit; Genius hits a target no one else can see.”

- Arthur Schopenhauer

Capítulo 1

Ficha de Trabalho Prático Nº1

1.1 Objetivos

- “Familiarização com a arquitetura e filosofias do modelo de gestão preconizado pelo *Internet-standard Network Management Framework* (INMF), dando especial relevo ao *Simple Network Management Protocol* (SNMP) e às *Management Information Bases* (MIBs).
- Os alunos devem ainda consolidar os conceitos inerentes aos seguintes aspetos desta arquitetura de gestão:
 1. Arquitetura do sistema, i.e., quais as entidades intervenientes e qual a sua função; diferenças entre o conceito de objeto de gestão e instância de objeto de gestão;
 2. A evolução das normas integrantes, desde a primeira versão de 1991 até ao INMFv3;
 3. Utilidade da norma *Structure of Management Information* (SMI);
 4. Os mecanismos de segurança integrados.”

1.2 Requisitos

- “Acesso a sistema com *Linux* com, pelo menos, um pacote *freeware* instalado com suporte a SNMP (versão 2, no mínimo): Net-SNMP, CMU-SNMP, SCOTTY, etc.”

1.3 Parte 1.A - *SNMP* no *Cisco IOS*

“O IOS da Cisco suporta, naturalmente, agentes SNMP. A ativação destes agentes é configurada nos routers Cisco através de um conjunto de comandos específicos para gestão de redes (ou melhor, equipamentos de rede) através do SNMP.”

1.3.1 QUESTÃO TP1.A

Nesta questão foi pedido aos alunos para consultarem a bibliografia disponível sobre o *Cisco IOS* e descobrir os comandos necessários para conseguir efetivar os seguintes requisitos:

- “Permissão de leitura pública dos valores das instâncias de todos os objetos da MIB-II”
- “Permissão de alteração das instâncias dos objetos do grupo *ip* para a comunidade *gestaoderedes*, a partir da rede local.”

Após uma leitura cuidada da bibliografia cedida, afirmo que os comandos necessários são respetivamente:

- *snmp-server view*
- *snmp-server community*

1.3.1.1 *snmp-server view*

Após o estudo da bibliografia do *CISCO IOS* percebi que tinha que recorrer a “view’s” para permitir a uma determinada comunidade obter acesso a um ramo de OIDs de uma árvore SNMP. Este comando permite criar ou atualizar uma “view”. Quanto à sua sintaxe pode ser expressado da seguinte forma,

- **snmp-server view** view-name oid-tree **included** — **excluded**

Na Tabela 1.1 podemos ver em detalhe o que significa cada uma das partes que constitui o comando.

Ainda podemos remover uma *view* introduzindo o seguinte comando,

- *no snmp-server view view-name*

Tabela 1.1: Sintaxe do comando `snmp-server view`

view-name	Identifica o nome da view que está a ser criada ou atualizada.
oid-tree	Especifica o OID da subtree para ser incluída ou excluída da view.
included — excluded	Indica se o tipo de view é incluído ou excluído.

Tabela 1.2: Sintaxe do comando `snmp-server community`

string	Atua como password e permite o acesso ao protocolo SNMP.
view-name	Nome da view definida anteriormente. (opcional)
ro — rw	Especifica o tipo de acesso para leitura ou para leitura-escrita, caso seja utilizado a opção de rw. (opcional)
number	Inteiro de 1 a 99 que especifica uma tabela de IP's que estão autorizados a utilizar a string de comunidade para obter acesso ao agente SNMP. (opcional)

1.3.1.2 *snmp-server community*

Quanto ao segundo comando, é utilizado para configurar a *community string* de acesso ao protocolo SNMP. Tal como o comando anterior, podemos eliminar uma community string sendo para isso necessário introduzir “no” antes do comando. Este apresenta a seguinte sintaxe:

- **snmp-server community** string [**view view-name**] [**ro — rw**] [**number**]

A Tabela 1.2 discrimina as opções que podem ser tomadas.

1.3.1.3 Resultado Final

Por forma a responder corretamente ao enunciado desta questão, os comandos finais são:

1. *snmp-server view view1 mib-2 included*
2. *snmp-server community public view view1 ro*
3. *snmp-server view view2 ip included*
4. *snmp-server community gestaoderedes view view2 rw*

O primeiro comando cria a vista de nome *view1* com acesso ao grupo *mib-2*. Já o segundo, cria a comunidade *public* que tem acesso apenas de leitura à *view1*. O terceiro comando cria uma vista de nome *view2* com o grupo *ip*. Por fim, cria-se uma comunidade de nome *gestaoderedes* com acesso de escrita e leitura à *view2*.

1.4 Parte 1.B - *Net-SNMP*

“O pacote de *software* SNMP que deve instalar disponibiliza um agente SNMP para Unix. Pretende-se nesta secção que configure e ative um agente SNMP na sua estação de trabalho. Para isso, estude as páginas do manual (*manpages*) do *daemon snmpd* e do ficheiro de configuração *snmpd.conf* e ative o agente na porta 6666.

Experimente o software instalado, obtendo, nomeadamente, a seguinte informação de monitorização (valores de variáveis da MIB-II):

- i Os valores das instâncias de todas as variáveis do grupo system da MIB-II da sua estação de trabalho e de um qualquer encaminhador IP (um qualquer router da rede da Universidade do Minho ou um da sua rede doméstica);
- ii Informações dos interfaces de rede da sua estação de trabalho e de um qualquer encaminhador IP (um qualquer router da rede da Universidade do Minho ou um da sua rede doméstica).”

1.4.1 QUESTÃO TP1.B

Esta questão é constituída por 3 perguntas.

- i “Qual o valor e significado da instância do objeto com o OID lexicograficamente a seguir a 1.3.6.1.2.1.7.2 da sua estação de trabalho?”
- ii “Como poderia calcular o número de pacotes IP que entram num router e já não saem (i.e., têm esse router como destino final)?”
- iii “Quais as partições do sistema de ficheiros da sua estação de trabalho que a instrumentação do agente SNMP consegue identificar?”

Para conseguir responder a estas perguntas, foi necessário utilizar um *software* de browsing de SNMP MIBs. O *software* escolhido foi o *qtmib-an SNMP MIB browser for Linux platforms*. É construído como um *front-end* para o *net-snmp* e permite ao utilizador qualquer dispositivo habilitado para SNMP. Implementa SNMPv1 e SNMPv2c, e é lançado sob a licença GPL v2. É um *software* que se demonstrou bastante *user friendly* daí ter-se tornado na minha primeira opção. Uma desvantagem apresentada é que não possui a descrição dos OID- *object identifier*, sendo que para contornar esse obstáculo utilizei o comando *snmptranslate*.

1.4.1.1 Questão i. “Qual o valor e significado da instância do objeto com o OID lexicograficamente a seguir a 1.3.6.1.2.1.7.2 da sua estação de trabalho?”

Para conseguir responder a esta questão fiz uso da primitiva *GetNext()* para que fosse possível obter o OID lexicograficamente a seguir a 1.3.6.1.2.1.7.2. Na Figura 1.1 podemos ver o decorrer desta operação bem como, o seu resultado.



Figura 1.1: Utilização da primitiva *GetNext()*.

Obtive então como resultado a identificação do OID pelo seu número e o valor que contém. A resposta a esta pergunta é então o OID 1.3.6.1.2.1.7.2.0 *udpNoPorts* (representa o número total de datagramas UDP para os quais não existia nenhuma aplicação na porta de destino.) , com o valor de 484.

1.4.1.2 Questão ii. “Como poderia calcular o número de pacotes IP que entram num router e já não saem (i.e., têm esse router como destino final)?”

Como não consegui obter acesso ao router, utilizei o *ip* do *localhost* (127.0.0.1). Para calcular o número pretendido para esta questão, tive que identificar os OIDs necessários. Podendo então assumir que a resposta assenta na diferença do valor entre dois *object identifier*.

O primeiro é o *ipInReceives* (1.3.6.1.2.1.4.3) e devolve o número total de datagramas que entraram incluindo os que entraram por erro. Já o segundo

OID é o *ipForwDatagrams* (1.3.6.1.2.1.4.6.0) que devolve o número de datagramas que entraram mas foram encaminhados para outra entidade ou seja, esta entidade não era o seu destino IP final, como um resultado do qual foi feita uma tentativa de encontrar uma rota para encaminhá-los para esse destino final.

A Figura 1.2 discrimina a obtenção destes dois OIDs.

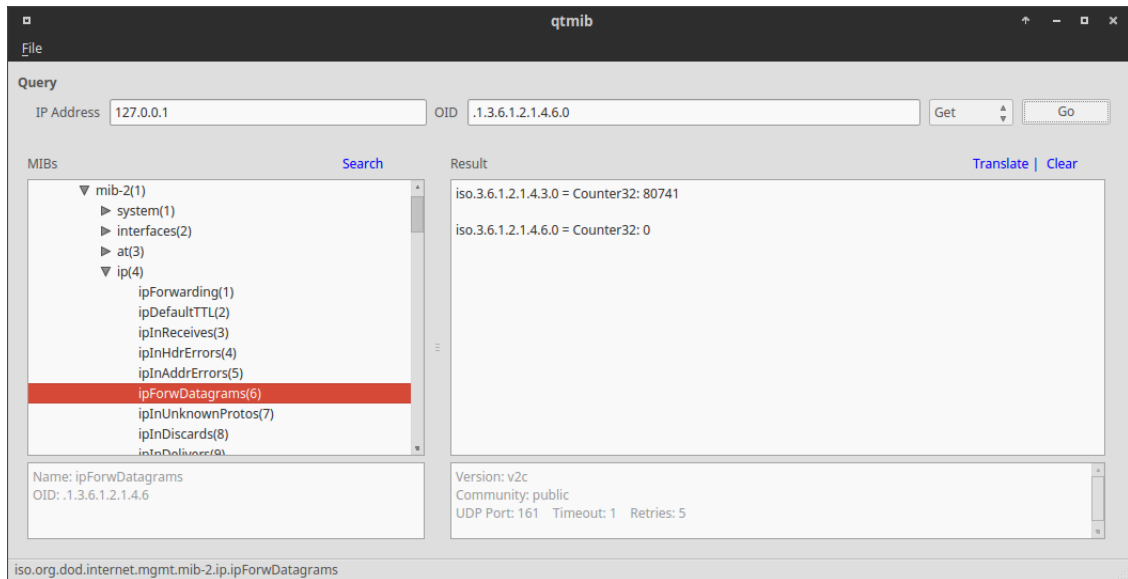


Figura 1.2: Obtenção dos valores de *ipInReceives* e *ipForwDatagrams*.

Após isto, o resultado final pretendido é de 80741 como descrito na operação (1.1).

$$ipInReceives - ipForwDatagrams = 80741 - 0 = 80741 \quad (1.1)$$

1.4.1.3 Questão iii. “Quais as partições do sistema de ficheiros da sua estação de trabalho que a instrumentação do agente SNMP consegue identificar?”

Para responder a esta questão, investiguei o grupo *host* e obtive os valores das instâncias que contém as informações sobre a monitorização do sistema de ficheiros.

O *hrStorageType* listar todo o tipo de dispositivos presentes na máquina, permitindo assim saber as partições do sistema de ficheiros da minha estação de trabalho. Obtive então a resposta a esta questão através do comando *snmpwalk -v 2c -c public localhost hrStorageType*, situação ilustrada na Figura 1.3.

```
jorge@PC-K52F ~ $ snmpwalk -v 2c -c public localhost hrStorageType
HOST-RESOURCES-MIB::hrStorageType.1 = OID: HOST-RESOURCES-TYPES::hrStorageRam
HOST-RESOURCES-MIB::hrStorageType.3 = OID: HOST-RESOURCES-TYPES::hrStorageVirtualMemory
HOST-RESOURCES-MIB::hrStorageType.6 = OID: HOST-RESOURCES-TYPES::hrStorageOther
HOST-RESOURCES-MIB::hrStorageType.7 = OID: HOST-RESOURCES-TYPES::hrStorageOther
HOST-RESOURCES-MIB::hrStorageType.8 = OID: HOST-RESOURCES-TYPES::hrStorageOther
HOST-RESOURCES-MIB::hrStorageType.10 = OID: HOST-RESOURCES-TYPES::hrStorageVirtualMemory
HOST-RESOURCES-MIB::hrStorageType.31 = OID: HOST-RESOURCES-TYPES::hrStorageFixedDisk
HOST-RESOURCES-MIB::hrStorageType.34 = OID: HOST-RESOURCES-TYPES::hrStorageFixedDisk
HOST-RESOURCES-MIB::hrStorageType.41 = OID: HOST-RESOURCES-TYPES::hrStorageFixedDisk
HOST-RESOURCES-MIB::hrStorageType.42 = OID: HOST-RESOURCES-TYPES::hrStorageFixedDisk
HOST-RESOURCES-MIB::hrStorageType.43 = OID: HOST-RESOURCES-TYPES::hrStorageFixedDisk
HOST-RESOURCES-MIB::hrStorageType.44 = OID: HOST-RESOURCES-TYPES::hrStorageFixedDisk
HOST-RESOURCES-MIB::hrStorageType.46 = OID: HOST-RESOURCES-TYPES::hrStorageFixedDisk
```

Figura 1.3: Resultado do comando *snmpwalk -v 2c -c public localhost hrStorageType*.

1.5 Considerações Finais

Neste primeiro trabalho prático consegui responder com sucesso a todas as questões, cumprindo com todos os objetivos arquitetados. Apesar da compreensão e resolução deste TP serem bastante acessíveis, a principal dificuldade foi a configuração de todas as ferramentas para que o mesmo pudesse ser realizado.

Capítulo 2

Ficha de Trabalho Prático Nº2

2.1 Objetivos

- “Familiarização com a arquitectura e filosofias do modelo de gestão preconizado pelo *Internet-standard Network Management Framework* (INMF), dando especial relevo ao *Simple Network Management Protocol* (SNMP) e às *Management Information Bases*(MIBs).
- Saber aplicar APIs SNMP para construção de ferramentas de monitorização.”

2.2 Requisitos

- “Sistema com um agente SNMPv2c instalado (preferencialmente o NET-SNMP) e pacote de desenvolvimento numa linguagem de programação que disponibilize APIs para construção dum gestor SNMPv2c.”

2.3 Ferramenta de Monitorização do número de pacotes IP

Este segundo trabalho prático resume-se a “criar um programa para monitorização (o mais possível em tempo real) do número de pacotes IP que entram e saem”, (tendo a decisão dos valores estatísticos utilizados e mostrados) “num sistema dum qualquer *host* na rede local (endereço e porta IP configuráveis pelo utilizador).O programa deve disponibilizar uma qualquer forma de visualização (com actualização o mais rápida e efetiva/útil possível) para que seja fácil ao gestor de rede ver a evolução dos valores”.

2.3.1 Desenvolvimento da Aplicação

Para a realização desta aplicação utilizei um protocolo de gestão de redes denominado por SNMP que permitiu descobrir o estado da rede e proceder à sua monitorização.

Foi ainda utilizada a API SNMP4J que permite obter todas as funcionalidades do SNMP em código Java.

2.3.1.1 Arquitetura da solução

Foram criadas duas classes Java denominadas por GUI e Main. A classe GUI é a principal e contém:

- I os dados do *ip* da MIB-II;
- II o calculo necessário para efetuar o trabalho com sucesso;
- III a definição do intervalo de *pooling*;
- IV a interface gráfica.

Já a classe Main, apenas faz uso da classe GUI.

Para a correta implementação foi necessário recorrer a três objetos.

- *ipInReceives* (1.3.6.1.2.1.4.3.0) - “The total number of input datagrams received from interfaces, including those received in error.”;
- *ipOutRequests* (1.3.6.1.2.1.4.10.0) - “The total number of IP datagrams which local IP user-protocols (including ICMP) supplied to IP in requests for transmission. Note that this counter does not include any datagrams counted in *ipForwDatagrams*.”;
- *ipForwDatagrams* (1.3.6.1.2.1.4.6.0) - “The number of input datagrams for which this entity was not their final IP destination, as a result of which an attempt was made to find a route to forward them to that final destination. In entities which do not act as IP Gateways, this counter will include only those packets which were Source-Routed via this entity, and the Source- Route option processing was successful.”.

Com a ajuda destes OIDs, o valor total dos pacotes IP que entram e saem, será calculado a partir da operação (2.1).

$$Total = ipForwDatagrams + ipOutRequests \quad (2.1)$$

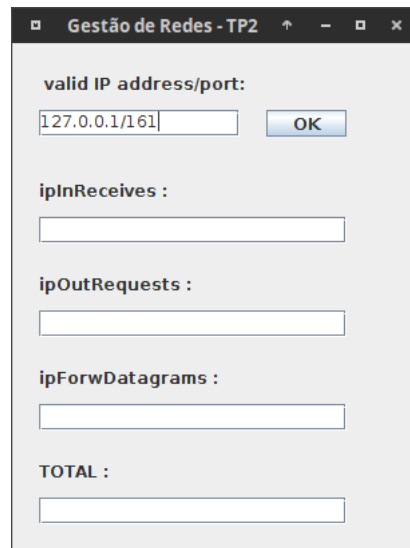
2.3.1.2 Implementação

O SNMP4J permitiu-me elaborar o código que identifica o agente SNMP a que corresponde o IP e a porta (introduzido pelo utilizador) e ainda o código que permite obter a primitiva *snmpGet()*. Após isso, criei um método que faz o get das OIDs referenciadas na secção anterior e imprime o seu valor na consola. Tudo isto acontece ciclicamente com intervalos de 1 segundo.

Quanto à interface gráfica, criei uma janela através da API Java Swing, que permite ao utilizador escolher a o endereço de IP e a respetiva porta, e ainda demonstra os valores correspondentes aos objetos referenciados bem como, o total calculado na operação (2.1). Infelizmente, dada a velocidade de recolha de dados, as áreas de texto da interface não estão a funcionar corretamente, uma vez que o Java Swing não consegue atualizar tão rápido.

2.3.2 Aplicação

Ao iniciar a aplicação o utilizador depara-se com a janela da Figura 2.1. Nesta janela o cliente deve introduzir o endereço IP desejado, bem como a respetiva porta.



The image shows a Java Swing window titled "Gestão de Redes - TP2". Inside the window, there is a label "valid IP address/port:" followed by a text input field containing the text "127.0.0.1/161" and an "OK" button. Below this, there are four more text input fields, each preceded by a label: "ipInReceives:", "ipOutRequests:", "ipForwDatagrams:", and "TOTAL:". All these input fields are currently empty.

Figura 2.1: Interface gráfica da aplicação final.

Após a confirmação, é então disposto o output (Figura2.2) que corresponde a quatro valores.

1. *ipInReceives*;

2. *ipOutRequests*;
3. *ipForwDatagrams*;
4. Total, resultado da operação (2.1).

```

/opt/java/jdk1.8.0_65/bin/java ...
127.0.0.1/161
| ipInReceives: 39573898 | ipOutRequests: 39517254 | ipForwDatagrams: 0 | total: 39517254 |
| ipInReceives: 39573898 | ipOutRequests: 39517254 | ipForwDatagrams: 0 | total: 39517254 |
| ipInReceives: 39573898 | ipOutRequests: 39517254 | ipForwDatagrams: 0 | total: 39517254 |
| ipInReceives: 39573898 | ipOutRequests: 39517254 | ipForwDatagrams: 0 | total: 39517254 |
| ipInReceives: 39573898 | ipOutRequests: 39517254 | ipForwDatagrams: 0 | total: 39517254 |
| ipInReceives: 39573898 | ipOutRequests: 39517254 | ipForwDatagrams: 0 | total: 39517254 |
| ipInReceives: 39573934 | ipOutRequests: 39517290 | ipForwDatagrams: 0 | total: 39517290 |
| ipInReceives: 39573934 | ipOutRequests: 39517290 | ipForwDatagrams: 0 | total: 39517290 |
| ipInReceives: 39573934 | ipOutRequests: 39517290 | ipForwDatagrams: 0 | total: 39517290 |
| ipInReceives: 39573934 | ipOutRequests: 39517290 | ipForwDatagrams: 0 | total: 39517290 |
| ipInReceives: 39573934 | ipOutRequests: 39517290 | ipForwDatagrams: 0 | total: 39517290 |
| ipInReceives: 39573934 | ipOutRequests: 39517290 | ipForwDatagrams: 0 | total: 39517290 |
| ipInReceives: 39573979 | ipOutRequests: 39517336 | ipForwDatagrams: 0 | total: 39517336 |
| ipInReceives: 39573979 | ipOutRequests: 39517336 | ipForwDatagrams: 0 | total: 39517336 |
| ipInReceives: 39573979 | ipOutRequests: 39517336 | ipForwDatagrams: 0 | total: 39517336 |
| ipInReceives: 39573979 | ipOutRequests: 39517336 | ipForwDatagrams: 0 | total: 39517336 |
| ipInReceives: 39573979 | ipOutRequests: 39517336 | ipForwDatagrams: 0 | total: 39517336 |
| ipInReceives: 39573979 | ipOutRequests: 39517368 | ipForwDatagrams: 0 | total: 39517368 |
| ipInReceives: 39574011 | ipOutRequests: 39517368 | ipForwDatagrams: 0 | total: 39517368 |
| ipInReceives: 39574011 | ipOutRequests: 39517368 | ipForwDatagrams: 0 | total: 39517368 |
| ipInReceives: 39574011 | ipOutRequests: 39517368 | ipForwDatagrams: 0 | total: 39517368 |
| ipInReceives: 39574011 | ipOutRequests: 39517368 | ipForwDatagrams: 0 | total: 39517368 |
| ipInReceives: 39574045 | ipOutRequests: 39517402 | ipForwDatagrams: 0 | total: 39517402 |
| ipInReceives: 39574045 | ipOutRequests: 39517402 | ipForwDatagrams: 0 | total: 39517402 |
| ipInReceives: 39574045 | ipOutRequests: 39517402 | ipForwDatagrams: 0 | total: 39517402 |
| ipInReceives: 39574045 | ipOutRequests: 39517402 | ipForwDatagrams: 0 | total: 39517402 |
| ipInReceives: 39574045 | ipOutRequests: 39517402 | ipForwDatagrams: 0 | total: 39517402 |
| ipInReceives: 39574045 | ipOutRequests: 39517402 | ipForwDatagrams: 0 | total: 39517402 |
Process finished with exit code 130

```

Figura 2.2: Output da aplicação final.

2.3.3 Questão 1 - Justifique bem os objetos escolhidos e o intervalo de tempo por defeito escolhido para o *polling*

Os objetos escolhidos (referidos na secção anterior), foram aqueles que na minha perspetiva permitiam obter a solução correta. O *ipInReceives* foi utilizado não na procura da resposta à TP2 pois, não entra na operação (2.1) mas sim porque representa um extra que penso que faz sentido demonstrar.

Quanto ao intervalo de tempo escolhido para o *polling* (1 segundo) deveu-se à consideração de determinados aspetos. Visto tratar-se de um programa

de monitorização em tempo real, o tempo não deve ser superior a 5 segundos, isto porque, quanto maior for o intervalo de *pooling*, pior será a noção do cliente, prejudicando a sua experiência de utilização.

2.4 Considerações finais

Concluído este segundo trabalho prático, posso afirmar que cumpri com as etapas objetivadas, tendo o desconhecimento do tempo de atualização da área de texto do Java Swing se revelado como a principal lacuna.

Conclusão

Neste relatório é descrita a solução apresentada para resolução dos trabalhos propostos.

Posso afirmar que todos os objetivos propostos foram alcançados. Este trabalho permitiu assim a consolidação dos conceitos, mecanismos e protocolos subjacentes ao modelo de gestão preconizado pelo INMF, realçando o protocolo SNMP e as MIBs.