GATEBOX

Bruno Sobral Oliveira

Vítor Brandão Silva

Orientador FEUP – Prof. José Ruela

Orientador INESC – Rui Moreira

Ciência.Inovação 2010

Universidade do Porto
Faculdade de Engenharia
FEUP

INESCPORTO
INSTITUTO DE ENGENHARIA DE SISTEMAS E COMPUTADORES DO PORTO

Faculdade de Engenharia da Universidade do Porto
Departamento de Engenharia Electrotécnica e de Computadores
Rua Roberto Frias, s/n, 4200-465 Porto, Portugal

Julho de 2006

GateBox

Bruno Sobral Oliveira

Vítor Brandão Silva

Trabalho realizado no âmbito da disciplina de Projecto, Seminário ou Trabalho de Fim de Curso, do 2º semestre, do 5º ano, da Licenciatura em Eng. Electrotécnica e de Computadores, Ramo Sistemas Telecomunicações, Electrónica e Computadores da Faculdade de Engenharia da Universidade do Porto.

Faculdade de Engenharia da Universidade do Porto
Departamento de Engenharia Electrotécnica e de Computadores
Rua Roberto Frias, s/n, 4200-465 Porto, Portugal

Julho de 2006

Resumo

Apresenta-se neste documento o projecto GateBox, cujo objectivo se centra na produção de um servidor Linux, projectado para operar como router/gateway. Este servidor, designado por máquina GateBox, deverá oferecer conectividade à Internet assim como um agregado de serviços de rede (DHCP, DNS, FTP, SAMBA, SSH, ...) ao conjunto de utilizadores que se encontrem sob alçada do servidor.

Num outro nível esta projecto visa a criação, em termos conceptuais, de uma rede GateBox através da distribuição de máquinas GateBox na WAN. O sistema encontra-se centralizado por motivos de gestão e administração numa máquina GateBox central, dita remota ou de administração.

Cada máquina GateBox será proprietária de um IP público, mantendo conectividade à Internet através de ligações a dois ou mais ISPs (necessariamente diferentes), procurando assim garantias de qualidade de serviço.

Na concepção do projecto existe o conceito de diferenciação de acessos, na rede de clientes GateBox, através da aplicação de classes de serviço. Este projecto embora não implemente os mecanismos que garantem a aplicação prática destas classes, dá o suporte necessário à sua integração.

A visão conceptual de utilizadores como clientes e a discriminação do acesso por classe de serviço obriga a regras robustas de autenticação. A autenticação é feita através de um servidor RADIUS a correr na máquina GateBox, sendo a ligação cliente–GateBox efectuada via PPPoE. Após autenticação cada cliente recebe um IP privado em concordância com a classe de serviço subscrita.

Ao nível da gestão implementou-se o protocolo SNMP – Simple Network Management Protocol – em toda a rede GateBox visando um controlo do estado de cada GateBox. A detecção e actuação autónoma de anomalias é outra das características do protocolo implementado proporcionando assim um aumento da fiabilidade das ligações.

A gestão e administração das GateBoxes são efectuadas na máquina remota. Neste servidor central desenvolveu-se uma interface Web, pensada no gestor da rede, onde é possível a gestão individualizada das GateBoxes e a administração dos clientes registados na rede GateBox.

Tendo em vista os clientes GateBox as palavras de ordem são facilidade de utilização/configuração e fiabilidade. Ao gestor da rede oferece-se um sistema de gestão diversificado e completo com uma forte componente de administração remota.

Índice

Lista de Figuras

Lista de Tabelas

Capítulo 1

1. Introdução

1.1 Enquadramento

O trabalho descrito neste relatório insere-se no âmbito da disciplina de Projecto, Seminário ou Trabalho de Fim de Curso (PSTFC) da Licenciatura em Engenharia Electrotécnica e de Computadores (LEEC), Ramo Sistemas Telecomunicações, Electrónica e Computadores (TEC). O projecto decorreu nas instalações do INESC[1] Porto sob proposta dos investigadores INESC Rui Moreira e Jorge Mamede e sob supervisão do professor José Ruela da Faculdade de Engenharia da Universidade do Porto (FEUP).

1.2 Motivação

A proposta GateBox revelou-se extremamente atraente aos olhos dos autores deste documento pelo seu carácter completo e inovador. Completo porque proporciona um desenvolvimento integral de um sistema Linux capaz de disponibilizar serviços que abrangem desde o routing até à autenticação. Inovador pela arquitectura global da rede GateBox e pela diferenciação de classes de serviço. O leitor ficará familiarizado com estes conceitos ao tomar contacto com os restantes capítulos deste documento.

1.3 Objectivos

Este projecto foi lançado com o propósito de produção de um servidor Linux como router/gateway capaz de disponibilizar diversos serviços de rede. Este servidor deverá servir um grupo limitado de utilizadores providenciando-lhes conectividade à Internet com

---

[1] Instituto de Engenharia de Sistemas e Computadores

garantias de qualidade de serviço[2] e de segurança. Estando presente o paradigma do utilizador-cliente é igualmente necessário implementar mecanismos de autenticação dos mesmos. Consta ainda dos objectivos deste projecto a criação de uma interface Web para a gestão e administração do servidor Linux.

1.4 Estrutura do Relatório

Este documentado encontra-se repartido em 7 capítulos dos quais o primeiro é constituido por esta introdução ao trabalho.

No segundo capítulo são apresentados os traços gerais deste projecto, identificando-se os problemas que se colocam na implementação do produto, ou seja a máquina GateBox. È ainda exposto o cenário real e de teste do produto aqui criado.

No terceiro capítulo é dado início à implementação pratica da máquina GateBox. São aqui apresentados os serviços básicos que integram o servidor Linux numa fase inicial.

No quarto capítulo é discutida a problemática da autenticação dos clientes apresentado-se as respectivas soluções.

O quinto capítulo debruça-se sobre o protocolo usado para executar a gestão da rede de máquinas GateBox, posteriormente designada por rede GateBox.

O derradeiro capítulo tira algumas conclusões sobre a comparação entre as proposições iniciais e os resultados obtidos.

[2] A qualidade de serviço é garantida com a integração do projecto FlowManager, desenvolvido em estreita colaboração com o projecto GateBox

Capítulo 2

2. Descrição do Problema

2.1 Caracterização do cenário

Pretende-se com este trabalho desenvolver um router/gateway, intitulado GateBox, que disponibilize vários serviços de rede, a um grupo de clientes que se encontrem na sua rede interna. A GateBox fornece aos seus utilizadores conectividade à Internet e pode disponibilizar vários serviços de rede. Cada GateBox possui um IP público com duas ou mais ligações com ISPs diferentes, efectuando um balanceamento de carga, garantindo melhor qualidade de serviço. A gestão e administração das GateBoxes é efectuada por uma máquina remota possuindo um IP público.

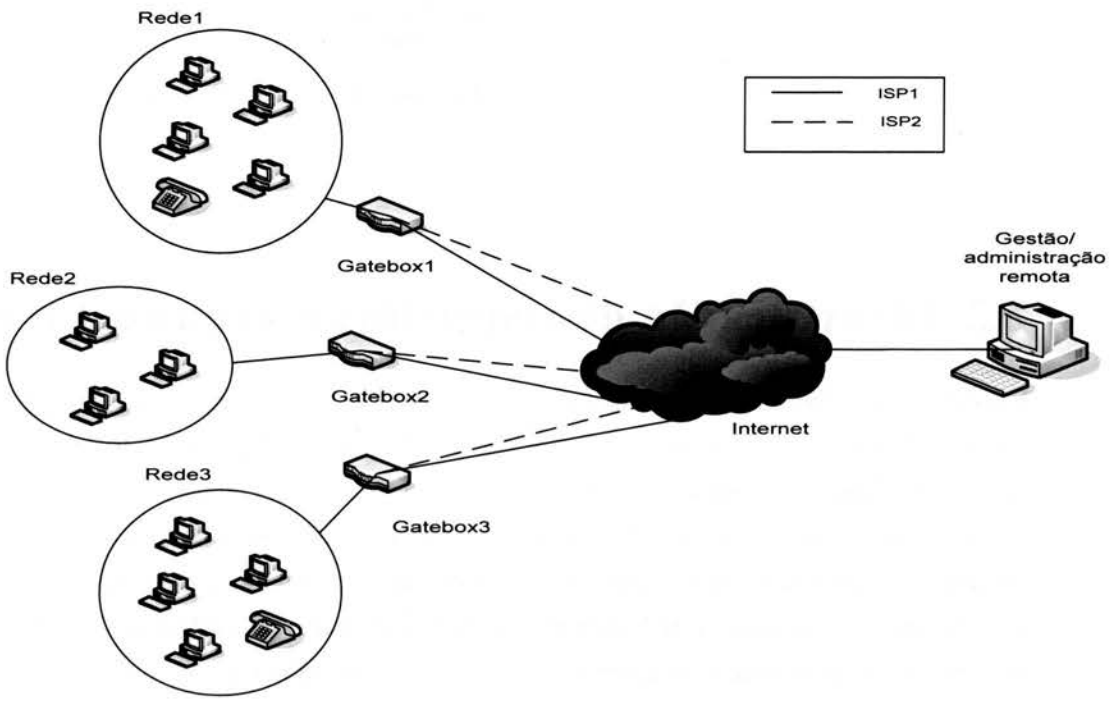


Figura 1 - Cenário real

Para simular e desenvolver o sistema descrito em cima, foram efectuadas algumas alterações. A GateBox encontra-se ligada à rede do INESC Porto, possuindo assim um IP externo dinâmico. Por isso, é necessário realocar a máquina de gestão e administração remota para a rede interna, onde é possível atribuir-lhe um IP fixo, podendo assim comunicar com a GateBox através do seu IP da rede interna (192.168.1.1). A GateBox possui duas ligações à rede do INESC, de modo a simular as múltiplas ligações aos ISPs.

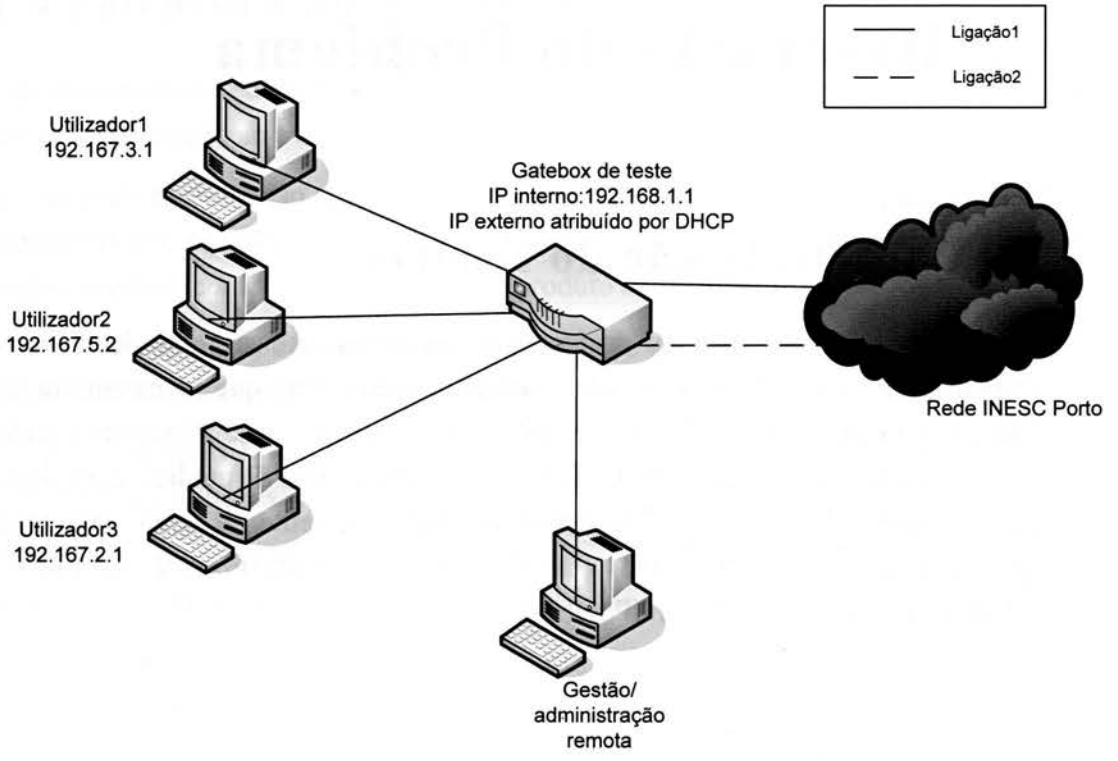


Figura 2 – Cenário de teste

2.2 Identificação de requisitos e arquitectura

Existem cinco classes de serviços (1 baixa prioridade e 5 máxima prioridade) para os utilizadores. Estas classes definem a prioridade e largura de banda do tráfego download e upload. Utilizadores com classes mais altas terão maior prioridade e largura de banda face aos utilizadores de classes inferiores. Mas se nenhum utilizador de classe elevada estiver a utilizar os recursos da rede e apenas se encontrar um utilizador de classe baixa a utilizá-los, ele terá toda a largura de banda disponível. Esta parte da gestão de tráfego por classes de serviço, corresponde a outro projecto chamado FlowManager.

No âmbito deste trabalho é necessário a existência de um servidor de autenticação local instalado em cada GateBox, que permita apenas o acesso à Internet e a serviços, a utiliza-

dores autenticados no sistema. Para o FlowManager ser integrado com a GateBox é necessário efectuar a atribuição de IPs por classe na autenticação, ou seja, quando um utilizador por exemplo de classe 5 se autentica no sistema, irá receber um IP 192.167.5.1, onde o terceiro octeto do IP representa a sua classe de serviço.

As GateBoxes deverão possuir os serviços básicos instalados e funcionais, como o DHCP, DNS, SSH , FTP.... entre outros.

A gestão das GateBoxes será efectuada essencialmente utilizando o protocolo SNMP-Simple Network Management Protocol. Para a gestão e administração remota do sistema é necessário uma Interface web, onde se encontram todas essas funcionalidades integradas.

Capítulo 3

3. Configuração do sistema básico

Neste capítulo apresentam-se os procedimentos iniciais realizados com o objectivo de implementar a máquina GateBox como um servidor capaz de fornecer serviços básicos de rede. O trabalho aqui apresentado foi desenvolvido numa fase inicial do projecto e pode não reflectir o estado final do projecto. Procurou-se nesta fase proporcionar um conjunto de serviços básicos que permitissem que a GateBox oferecesse conectividade a um conjunto de clientes. Inclui-se ainda nesta fase a configuração dos sistemas de partilha de ficheiros SAMBA e FTP.

3.1 Dynamic Host Configuration Protocol

O **DHCP, Dynamic Host Configuration Protocol**, é um protocolo de serviço TCP/IP que oferece configuração dinâmica de terminais, com concessão de endereços IP de host e outros parâmetros de configuração para clientes de rede. Este protocolo é o sucessor do BOOTP que, embora mais simples, tornou-se limitado para as exigências actuais. O DHCP surgiu como standard em Outubro de 1993. O RFC 2131 contém as especificações mais actuais (Março de 1997). O último standard para a especificação do DHCP sobre IPv6 (DHCPv6) foi publicado a Julho de 2003 como RFC 3315.

Resumidamente, o DHCP opera da seguinte forma:

- Um cliente envia um pacote *broadcast* (destinado a todas as máquinas) com um pedido DHCP;
- Os servidores DHCP que capturarem este pacote irão responder (se o cliente se enquadrar numa série de critérios) com um pacote com configurações onde constará, pelo menos, um endereço IP, uma máscara de rede e outros dados opcionais, como o gateway, servidores de DNS, etc.

O DHCP usa um modelo cliente-servidor, no qual o servidor DHCP mantém a gestão centralizada dos endereços IP usados na rede.

Para implementar o DHCP na GateBox é necessário instalar o pacote DHCP disponível no repositório do Ubuntu. Configurou-se o servidor DHCP para actualizar dinamicamente o servidor DNS (ver capítulo 2). Nos ficheiros de configuração do DHCP, podemos definir como se quer que o servidor DNS seja actualizado. A vantagem é que quando um novo cliente se liga e obtém um endereço IP, o nome da sua máquina é automaticamente inserido no sistema DNS. Por razões de segurança é necessária uma chave para permitir a comunicação entre os servidores DHCP e DNS.

Começar por editar o ficheiro de configuração *dhcpd.conf* que se encontra na pasta */etc/dhcp3/*, com a seguinte configuração:

```
authoritative;                  # No other DHCP servers on this subnet
ddns-update-style interim;      # Supported update method - see man dhcpd.conf

include "/etc/bind/rndc.key";

 zone gatebox.pt. {              # Zona a ser actualizada
    primary 127.0.0.1;
    key rndckey;
 }

 zone 1.168.192.in-addr.arpa. { # Zona de retorno a ser utilizada
    primary 127.0.0.1;
    key rndckey;
 }

 subnet 192.168.1.0 netmask 255.255.255.0 {

  range                         192.168.1.2 192.168.1.254;
  default-lease-time            600;
  max-lease-time                7200;

  option routers                192.168.1.1;    # Default gateway
  option subnet-mask            255.255.255.0;
  option broadcast-address      192.168.1.255;
  option domain-name            "gatebox.pt";
  option domain-name-servers    192.168.1.1;
}
```

O ficheiro */etc/bind/rndc.key* tem de ser primeiro gerado com o comando:

```
$ rndc-confgen -a
```

Para correr o servidor DHCP basta executar o seguinte comando:

```
$ /etc/init.d/dhcp3-server start
```

3.2 Domain Name Server

O **DNS** (***Domain Name System***) é um sistema de gestão de nomes hierárquico e distribuído operando segundo duas definições:

- Examinar e actualizar a sua base de dados.
- Resolver nomes de servidores em endereços de rede (IPs).

O sistema de distribuição de nomes de domínio foi introduzido em 1984 e com ele os nomes de hosts residentes numa base de dados pôde ser distribuído entre servidores múltiplos, baixando assim a carga em qualquer servidor que promova a administração no sistema de nomeação de domínios. Ele baseia-se em nomes hierárquicos e permite a inscrição de vários dados digitados além do nome do host e seu IP. Em virtude da base de dados DNS ser distribuída, o seu tamanho é ilimitado e o desempenho não degrada tanto quando se adiciona mais servidores nele.

Para implementar o servidor DNS na GateBox é necessário instalar o pacote BIND9 disponível nos repositórios do Ubuntu. O comando *rndc-confgen*, usando a opção "– a", pode gerar automaticamente as chaves necessárias para o *bind*. É gerado o seguinte ficheiro */etc/bind/rndc.key*:

```
key "rndc-key" {
        algorithm hmac-md5;
        secret "JIjUPfT2GZZ172o5IdcK1Q==";
 };
```

É necessário configurar o ficheiro */etc/bind/named.conf*. Começar por acrescentar:

```
include "/etc/bind/rndc.key";
controls {
    inet 127.0.0.1 allow { localhost; } keys { rndc-key; };
};
```

Acrescentar no ficheiro */etc/bind/named.conf.local* as duas zonas seguintes:

```
zone "gatebox.pt" {
       type master;
       file "/etc/bind/db.gatebox.pt";
       allow-update { key "rndc-key"; };
       notify yes;
};
zone "1.168.192.in-addr.arpa" {
       type master;
       file "/etc/bind/db.10.168.192";
       allow-update { key "rndc-key"; };
       notify yes;
};
```

Se um determinado nome não for encontrado no servidor DNS da GateBox é necessário por no ficheiro */etc/bind/named.conf.options* os IPs de servidores DNS alternativos, como por exemplo:

```
forwarders {
          192.135.246.1;
          194.117.24.1;
}
```

Em seguida criar os dois ficheiros de zona na pasta */etc/bind/*:

- ***db.gatebox.pt***

```
$TTL 86400 ;1 day

gatebox.pt IN SOA porteiro.gatebox.pt. hostmaster.gatebox.pt. (
                  2006033045 ;serial
                  21600      ;refresh (6 horas)
                  3600       ;retry (1 hora)
                  604800     ;expire (1 semana)
                  3600       ;minimum (1hora)
                  )
                  NS porteiro.gatebox.pt
                  MX 10 porteiro.gatebox.pt

$ORIGIN gatebox.pt.
porteiro  A       192.168.1.1
www       CNAME   porteiro
ftp       CNAME   porteiro
```

- ***db.10.168.192***

```
$TTL 86400 ;1 day

1.168.192.in-addr.arpa IN SOA porteiro.gatebox.pt. hostmas-
ter.gatebox.pt. (
                  2006033045 ;serial
                  21600      ;refresh (6 horas)
                  3600       ;retry (1 hora)
                  604800     ;expire (1 semana)
                  3600       ;minimum (1hora)
                  )
                  NS porteiro.1.168.192.in-addr.arpa.

$ORIGIN 1.168.192.in-addr.arpa.
1   PTR   porteiro
```

Para que o servidor DNS seja capaz de efectuar a actualização dos ficheiros de zona quando recebe pedidos do servidor DHCP, tem que ter permissões para escrever nos ficheiros de zona. É necessário mudar as permissões com os comandos:

```
$ cd /var/named
$ chown bind:bind *
```

No ficheiro */etc/resolv.conf* colocar o seguinte:

```
  search gatebox.pt
  nameserver 192.168.1.1
```

Para correr o servidor DNS basta corer o comando:

```
$ /etc/init.d/bind9 start
```

3.3 Servidor Apache com Secure Socket Layer

O **servidor Apache** (Apache Server) é o servidor web livre mais bem sucedido. Foi criado em 1995 por Rob McCool, então funcionário do NCSA (National Center for Supercomputing Applications), Universidade de Illinois. Na última pesquisa efectuada pelo site www.Netcraft.com, em Dezembro de 2005, foi contactado que a utilização do Apache supera 60% de servidores activos no mundo.

Secure Socket Layer (SSL) é uma forma de encriptação de dados, usada para garantir comunicação segura entre dois computadores (note-se que a comunicação só é segura entre o cliente e o servidor; não há garantias quanto a segurança entre eventuais comunicações posteriores de dados sensíveis entre o servidor e outro, eventual repete-se, computador). A tecnologia usada é a tecnologia RSA, de chave dupla, pública e privada.

Após a instalação do servidor Apache2, disponível nos repositórios do Ubuntu, começa-se por gerar o certificado SSL com o comando:

```
$ apache2-ssl-certificate
```

Agora é necessário configurar o SSL, começando por activar o modulo para o servidor apache2, com os comandos:

```
$ ln -s /etc/apache2/mods-available/ssl.conf /etc/apache2/mods-
enabled/
$ ln -s /etc/apache2/mods-available/ssl.load /etc/apache2/mods-
enabled/
```

É necessário especificar ao apache2 onde se encontra o certificado gerado, para isso basta editar o ficheiro */etc/apache2/mods-enabled/ssl.conf* e adicionar a seguinte linha, antes do */IfModule* (perto do fim do ficheiro):

```
SSLCertificateFile /etc/apache2/ssl/apache.pem
```

O próximo passo consiste em especificar no apache2 para escutar pedidos SSL no default port (443), para isso basta editar o ficheiro */etc/apache2/ports.conf* e adicionar no fim do ficheiro a seguinte linha:

```
add Listen 443
```

Falta criar agora o SSL site para o apache2, adicionando um host virtual no SSL site, editando o ficheiro */etc/sites-available/default* e colocar:

```
<VirtualHost localhost:443>
        DocumentRoot /var/www/
        SSLEngine On
        SSLCertificateFile /etc/apache2/ssl/apache.pem
</VirtualHost>
```

No final basta reiniciar o servidor apache com o comando:

```
$ /etc/init.d/apache2 restart
```

Ao abrir a página https://127.0.0.1/ deverá aparecer:

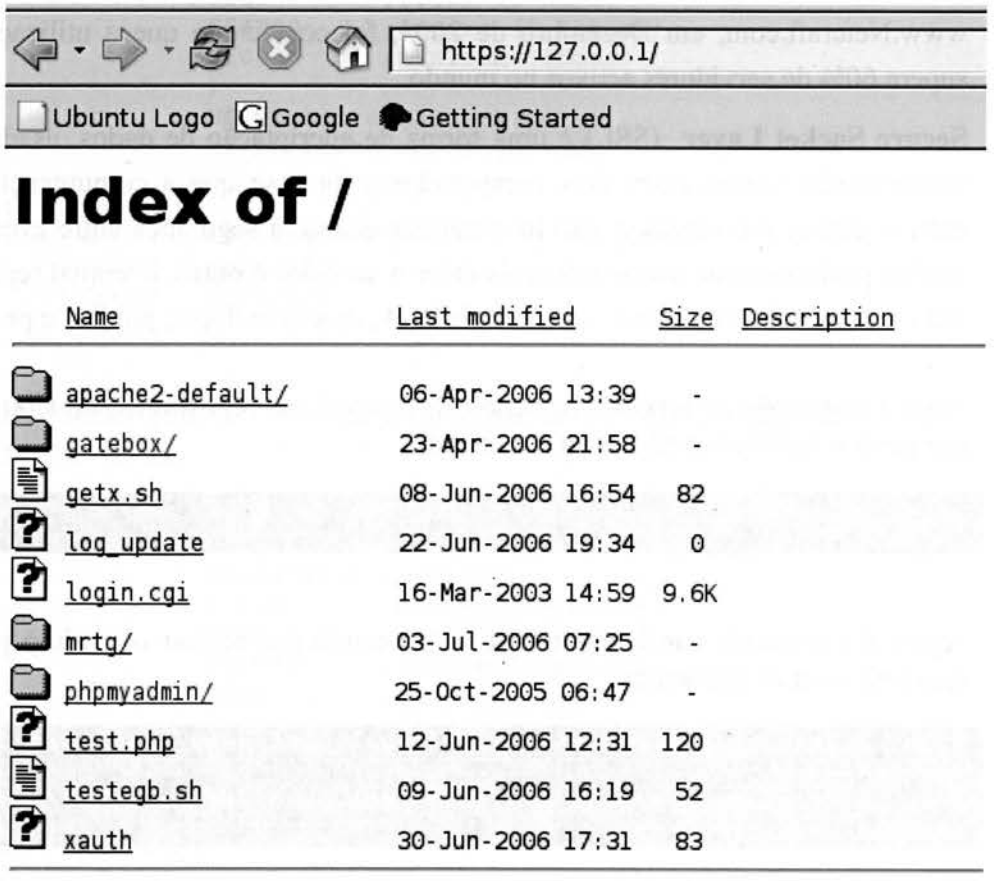


Figura 3 – Localhost em HTTPS

3.4 Secure Shell

Secure Shell ou **SSH** é, simultaneamente, um programa de computador e um protocolo de rede que permite a conexão com outro computador na rede, de forma a executar comandos de uma unidade remota. Possui as mesmas funcionalidades do TELNET, com a vantagem da conexão entre o cliente e o servidor ser encriptada.

O SSH faz parte da lista de protocolos TCP/IP que torna segura a administração remota de um servidor Unix.

Para implementar o servidor SSH na GateBox usa-se o pacote OpenSSH disponível nos repositórios do Ubuntu. Após instalado basta correr *sshd* na shell para iniciar o servidor SSH.

Para facilitar as operações de administração das GateBoxes é necessário implementar chaves públicas no ssh para não pedir a password para cada ligação que se faz com cada GateBox. É possível criar uma chave de autenticação RSA para estar apto a efectuar o login numa GateBox sem ter que introduzir a senha.

Na máquina da administração correr o comando ***ssh-keygen*** e pressionar enter quando pedir uma palavra-chave. Isto irá gerar uma chave privada e uma chave pública. Estas chaves encontram-se em *~/.ssh/id_rs* e *~/.ssh/id_rsa.pub*. Depois é necessário adicionar o conteúdo do ficheiro da chave pública (*id_rsa.pub*) dentro do ficheiro *~/.ssh/authorized_keys2* na GateBox (o ficheiro deverá ter o modo 600).

Agora é possível aceder à GateBox da máquina de administração por SSH sem ser pedida a password.

3.5 Samba

O Samba é uma aplicação que utiliza o protocolo SMB[3] proporcionando compatibilidade com redes que utilizam este protocolo principalmente redes NetBios da Microsoft. A disponibilização deste serviço permite que clientes ligados a uma máquina GateBox possam partilhar ficheiros entre si de forma simples e rápida.

Inicialmente é feita a instalação dos seguintes pacotes, via apt-get:

```
$ apt-get install samba samba-common samba-doc libcupsys2-gnutls10
libkrb53 winbind smbclient
```

Instalados os pacotes é necessário configurar o servidor Samba. A configuração é feita através do ficheiro */etc/samba/smb.conf*. O *smb.conf* divide-se fundamentalmente em três partes: configuração do servidor Samba (parâmetros na secção [global]), configuração dos diretórios/pastas pessoais dos utilizadores (parâmetros na secção [homes]) e as demais secções que correspondem aos directórios compartilhados. Lista-se de seguida o conteúdo do ficheiro:

[3] SMB – Server Message Block

```
[global]
   workgroup = MSHOME
   netbios name = porteiro
   server string = %h server (Samba, Ubuntu)

   passdb backend = tdbsam
   security = user
   username map = /etc/samba/smbusers
   name resolve order = wins bcast hosts
   domain logons = yes
   preferred master = yes
   wins support = yes

   # Set CUPS for printing
   printcap name = CUPS
   printing = CUPS

   # Default logon
   logon drive = Z:
   ;logon script = scripts/logon.bat
   logon path = \\porteiro\profile\%U


   # Useradd scripts
   add user script = /usr/sbin/useradd -m %u
   delete user script = /usr/sbin/userdel -r %u
   add group script = /usr/sbin/groupadd %g
   delete group script = /usr/sbin/groupdel %g
   add user to group script = /usr/sbin/usermod -G %g %u
   add machine script = /usr/sbin/useradd -s /bin/false/ -d
/var/lib/nobody %u
   idmap uid = 15000-20000
   idmap gid = 15000-20000


   # sync smb passwords woth linux passwords
   passwd program = /usr/bin/passwd %u
   passwd chat = *Enter\snew\sUNIX\spassword:* %n\n
*Retype\snew\sUNIX\spassword:* %n\n .
   passwd chat debug = yes
   unix password sync = yes

   # set the loglevel
   log level = 3


[printers]
   comment = All Printers
   path = /var/spool/samba
   printable = yes
   guest ok = yes
   browsable = yes


[netlogon]
   comment = Network Logon Service
   path = /home/samba/netlogon
   admin users = Administrator
   valid users = %U
   read only = no
   browsable = no
```

```
[profile]
   comment = User profiles
   path = /home/samba/profiles
   valid users = %U
   create mode = 0600
   directory mode = 0700
   writable = yes
   browsable = no


[allusers]
  comment = All Users
  path = /home/samba/shares/allusers
  valid users = @users
  force group = users
  create mask = 0660
  directory mask = 0771
  writable = yes
  browsable = no


[public]
   comment = Public Folder
   path = /home/samba/public
   public = yes
   writable = yes
   browsable = yes
   create mask = 0777
   directory mask = 0777
   force user = nobody
   force group = nogroup

[gatebox]
   comment = GateBox Public Folder
   path = /home/master/GateBox
   public = yes
   writable = yes
   browsable = yes
   create mask = 0777
   directory mask = 0777
   valid users = userftp
```

Relativamente à secção global destaque-se as seguintes opções:

netbios name: O "netbios name" deve ser o mesmo que o nome da máquina (hostname). No caso da máquina de testes esta recebeu o hostname "porteiro".

workgroup: MSHOME especifica o domínio Windows que as máquinas Windows usam.

logon drive: H: é a letra que identifica a unidade Samba que irá aparecer no Explorador do Windows.

Após a secção printers segue-se a definição das pastas locais:

- netlogon
- profile

- allusers
- public
- gatebox

Para tal é necessário criar as respectivas directorias:

```
$ mkdir /home/samba
$ mkdir /home/samba/netlogon
$ mkdir /home/samba/profiles
$ mkdir /var/spool/samba
$ chmod 777 /var/spool/samba/
$ chown -R root:users /home/samba/
$ chmod -R 771 /home/samba/
$ mkdir /home/samba/public
$ mkdir /home/samba/gatebox
```

A pasta *gatebox* foi definida como o espaço de partilha de ficheiros. Para efeito de testes este directório encontra-se foi definido no mesmo disco do sistema operativo. No entanto é interessante a colocação deste espaço num disco externo dedicado ao armazenamento de ficheiros dos clientes GateBox.

O acesso à pasta *gatebox* encontra-se limitado ao utilizador *userftp* (este utilizador foi criado para o servidor FTP, apresentado na próxima secção, tendo sido reutilizado para o acesso Samba). O utilizador *userftp* é necessariamente um utilizador criado no sistema operativo. O nome *userftp* não transparece para os utilizadores do serviço Samba. Em alternativa, faz-se o mapeamento no ficheiro *smbusers* do nome *userftp* para *gatebox*. Deste modo para que um utilizador tire partido do Samba usará o nome de utilizador gatebox e uma password atribuída pelo administrador GateBox.

3.6 File Transfer Protocol

O protocolo File Transfer Protocol (Protocolo de Transferência de Arquivos) ou FTP é um protocolo bastante usado na Internet para transferência de ficheiros de forma rápida e versátil. O FTP é baseado no TCP [RFC959,1985] mas é anterior à pilha de protocolos TCP/IP, sendo posteriormente adaptado para o TCP/IP. É o padrão da pilha TCP/IP para transferir ficheiros sendo um protocolo genérico independente de hardware e do sistema operativo.

A implementação de um servidor FTP na GateBox foi pensada com o propósito de trazer conteúdos disponíveis na Internet para a rede local. Deste modo, seria criado, tal como no caso Samba, um espaço de armazenamento na máquina GateBox para acesso via FTP. Este acesso, dado localizar-se localmente, facultaria um acesso muito rápido e fiável aos conteúdos.

Para se implementar o servidor FTP usou-se o pacote proftpd, instalado via apt-get:

```
$ apt-get install proftpd
```

A configuração do pacote proftpd.conf é feita via */etc/proftpd.conf*:

```
AllowOverwrite on
AuthAliasOnly on

UserAlias gatebox userftp

ServerName              "porteiro"
ServerType              standalone
DeferWelcome                  on

MultilineRFC2228        on
DefaultServer                 on
ShowSymlinks                  off

TimeoutNoTransfer       600
TimeoutStalled                600
TimeoutIdle             1200

#DisplayLogin                    welcome.msg
DisplayFirstChdir               .message
ListOptions                   "-l"

RequireValidShell             off

TimeoutLogin 20

RootLogin               off

# It's better for debug to create log files ;-)
ExtendedLog                   /var/log/ftp.log
TransferLog                   /var/log/xferlog
SystemLog               /var/log/syslog.log

#DenyFilter             \*.*/

# choose to use /etc/ftpusers file
UseFtpUsers off

# Allow to restart a download
AllowStoreRestart       on

# Uncomment this if you are using NIS or LDAP to retrieve passwords:
PersistentPasswd        off

# Uncomment this if you would use TLS module:
#TLSEngine              on

# Uncomment this if you would use quota module:
#Quotas                       on

# Uncomment this if you would use ratio module:
#Ratios                       on
```

```
# Port 21 is the standard FTP port, so don't use it for security reasons
#(choose here the port you want)
Port                    1980
#Port                   21

# To prevent DoS attacks, set the maximum number of child processes
# to 30.  If you need to allow more than 30 concurrent connections
# at once, simply increase this value.  Note that this ONLY works
# in standalone mode, in inetd mode you should use an inetd server
# that allows you to limit maximum number of processes per service
# (such as xinetd)
MaxInstances                8

# Set the user and group that the server normally runs at.
User                    nobody
Group                   nogroup

# Display a message after a successful login
AccessGrantMsg "welcome !!!"
# This message is displayed for each access good or not
ServerIdent                 on       "FTP Gatebox"

# Set /home/FTP directory as home directory
DefaultRoot /home/FTP

# Lock all the users in home directory, ***** really important *****
DefaultRoot ~

MaxLoginAttempts    5

#VALID LOGINS
<Limit LOGIN>
AllowUser userftp
#AllowUser frbr
DenyALL
</Limit>

<Directory /home/FTP>
Umask 022 022
AllowOverwrite off
      <Limit MKD STOR DELE XMKD RNRF RNTO RMD XRMD>
      DenyAll
      </Limit>
</Directory>

<Directory /home/FTP/download/*>
Umask 022 022
AllowOverwrite off
      <Limit MKD STOR DELE XMKD RNEF RNTO RMD XRMD>
      DenyAll
      </Limit>
</Directory>

<Directory> /home/FTP/upload/>
Umask 022 022
AllowOverwrite on
      <Limit READ RMD DELE>
            DenyAll
      </Limit>
```

```
        <Limit STOR CWD MKD>
                AllowAll
        </Limit>
</Directory>

MasqueradeAddress porteiro.gatebox.pt

# These ports should be safe...
PassivePorts 60000 65535

UseReverseDNS off
IdentLookups off
```

Para se ter acesso ao serviço FTP é necessário definir um utilizador no sistema operativo. Para tal criou-se o utilizador *userftp* (também usado no Samba) com directório pessoal em */home/FTP*. Para aumentar o nível de segurança definiu-se este utilizador com a shell */bin/false*. Este utilizador será apenas usado para acesso FTP e não para uso do sistema operativo pelo que não necessita de uma shell válida.

O alojamento dos ficheiros fica assim restringido ao directório pessoal do utilizador *userftp*. Dentro da directoria criaram-se directórios para upload e download, definindo-se os respectivos privilégios de acesso (apenas leitura no download, e leitura e escrita do upload):

```
$ mkdir /home/FTP/download
$ chmod 755 /home/FTP/download
$ mkdir /home/FTP/upload
$ chmod 777 /home/FTP/upload
```

Tal como no samba o utilizador *userftp* é conhecido por *gatebox*. Este mapeamento é feito através da directiva UserAlias.

O acesso ao servidor FTP faz-se através da porta 1980, como se mostra no seguinte comando:

```
$ ftp porteiro.gatebox.pt:1980
```


```
$ ftp porteiro.gatebox.pt:1980
```


A autenticação no servidor faz-se com o nome de utilizador gatebox e com a password do utilizador *userftp*.

Capítulo 4

4. Autenticação

4.1 Problema da autenticação

O problema da autenticação surge na necessidade de cada utilizador que se subscreve ao serviço, necessitar de se autenticar para obter acesso à Internet. Para isso é necessário existir um servidor de autenticação com uma base de dados e um cliente de autenticação para cada utilizador. Na base de dados irão estar armazenados todos os dados necessários para autenticação de cada utilizador e ainda outro tipo de informação de accounting.

A autenticação será feita localmente, ou seja, cada GateBox terá um servidor de autenticação local onde se encontram todos os utilizadores associados a ela.

Para desenvolver um sistema deste tipo decidimos que a melhor opção para implementar o servidor de autenticação local seria um servidor RADIUS com uma base de dados em mySQL. O que é o RADIUS e a sua integração na GateBox encontra-se no capítulo 4.2.

Após a implementação do servidor RADIUS na GateBox tentou-se arranjar uma solução para autenticar os utilizadores nesse servidor. Havia outro problema inerente ao FlowManager que era necessário atribuir IPs aos utilizadores autenticados conforme a sua classe de serviço, por exemplo, um utilizador de classe 2 terá um IP 192.167.2.1 e um utilizador de classe 5 terá um IP 192.167.5.1, onde o terceiro octeto corresponde à classe de serviço do utilizador.

Para ultrapassar estes dois problemas começamos por tentar modificar o servidor DHCP já instalado de forma a autenticar utilizadores no servidor RADIUS e atribuir IPs por classes de serviço. Após algumas experiências chegamos à conclusão que o servidor DHCP não apresentava os requisitos necessários para ultrapassar o problema.

Foi então que surgiu a ideia da implementação de um servidor **PPPoE** (Point to Point Protocol over Ethernet), que comunica com o servidor RADIUS. Com o PPPoE já foi possível autenticar os utilizadores no RADIUS e atribuir IPs por classes de serviços. No capítu-

lo 3.3 encontra-se todos os passos da integração do servidor PPPoE com RADIUS na Gatebox.

Todos os utilizadores que não se autenticam não possuem qualquer tipo de conectividade à Internet nem com a GateBox.

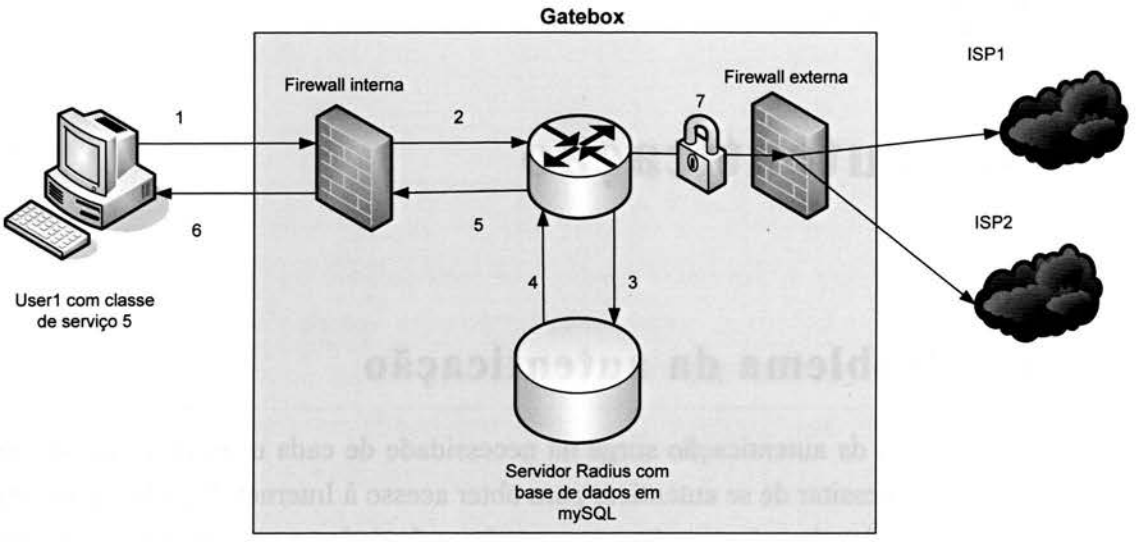


Figura 4 – Cenário de autenticação

Na Figura 6 encontra-se um exemplo de um utilizador a autenticar-se na GateBox:

- 1e 2) O utilizador liga o cliente PPPoE e envia o seu username e password para o servidor PPPoE na GateBox;
- 3) O servidor envia a informação recebida para o servidor RADIUS;
- 4) O servidor RADIUS notifica o servidor PPPoE que o user1 encontra-se autenticado e envia o IP associado a esse user (192.167.5.1);
- 5 e 6) O servidor PPPoE envia a resposta do servidor RADIUS e o IP para o cliente PPPoE do utilizador que se autenticou;
- 7) É criado um túnel entre o cliente e a GateBox por PPPoE e a ligação 7 é aberta para o utilizador, ficando com acesso à Internet.

4.2 Servidor RADIUS

4.2.1 O que é o RADIUS

Remote Authentication Dial In User Service (RADIUS) é um protocolo AAA (authentication, authorization and accounting) para aplicações tais como, acesso à rede e mobilidade IP. É vocacionado para funcionar em situações locais ou remotas.

Ao introduzir o username e password no cliente de autenticação, esta informação é transferida para o servidor RADIUS, onde este vai verificar se a informação recebida é válida usando protocolos de autenticação como o PAP (Password Authentication Protocol), CHAP (Challenge Handshake Authentication) ou EAP. Se a informação for válida o servidor permite o acesso ao utilizador à rede e atribui um endereço IP.

O servidor RADIUS é também notificado sempre que uma sessão se inicia ou termina, podendo assim registar os períodos de conexão de cada utilizador, onde esta informação poderá ser usada para dados estatísticos.

4.2.2 Implementação do Servidor Radius

O pacote open source usado para implementação do servidor RADIUS foi o **freeradius v1.1.0** e a base de dados em **mySQL**.

Após instalação do pacote freeradius e do servidor mySQL é necessário configurá-los e construir a base de dados.

Os ficheiros de configuração do freeradius encontram-se na pasta */etc/freeradius*. Primeiro alterou-se as últimas linhas do ficheiro principal de configuração *radiusd.conf* para funcionar com SQL:

```
....
authorize {
preprocess
chap
mschap
suffix
sql
}
authenticate {
Auth-Type PAP {
pap
}
Auth-Type CHAP {
chap
}
```

```
Auth-Type MS-CHAP{
mschap
}
}
preacct {
pre process
acct_unique
suffix
}
accounting {
acct_unique
detail
unix
sql
radutmp
}
session {
radutmp
}
.....
```

É necessário agora editar o ficheiro "sql.conf" e introduzir nele o username e password para o radius poder comunicar com a base de dados no servidor mySQL, sendo apenas necessário editar as seguintes linhas:

```
....
server = "máquina onde se encontra o servidor mySQL"
login = "username da base de dados"
password = "password da base de dados"
radius_db = "nome da base de dados radius"
....
```

Para testar que o servidor RADIUS encontra-se bem instalado e configurado basta correr o servidor em debug mode, com o comando "freeradius –X". Se o comando terminar como se encontra demonstrado em baixo o servidor encontra-se operacional.

```
...
Listening on authentication *:1812
Listening on accounting *:1813
Listening on proxy *:1814
Ready to process requests
```

Agora é necessário criar a base de dados RADIUS no servidor mySQL. Primeiro cria-se a base de dados:

```
$ mysql -u root -p
```

(entra no servidor mysql como root)

```
$ mysql>CREATE DATABASE radius
```

(cria base de dados com nome radius)

As tabelas necessárias para o servidor RADIUS encontram-se na pasta /usr/share/doc/freeradius/db_mysql.sql, basta só copiá-las para a base de dados criada no ponto anterior com o seguinte comando:

```
$ mysql -u root -p senhadorootnomysql radius < db_mysql.sql
```

A base de dados RADIUS possui vários tipos de tabelas, cada uma com funcionalidades diferentes, onde as mais relevantes encontram-se a seguir representadas:

- **usergroup**, associa um username a um grupo de utilizadores.
- **radcheck**, põe uma entrada para cada utilizador com o valor da sua password no atributo 'Password'.
- **radreply**, cria para cada utilizador respostas especificas para um determinado username.
- **radgroupreply**, cria atributos que são retornados para todos os membros de um grupo.
- **radacct,** encontra-se toda a informação de accounting.

É agora necessário criar alguns users na base de dados RADIUS para testar o sistema.

Na tabela **radcheck** deverá ser acrescentado o seguinte:

id	UserName	Attribute	Op	Value
1	user1	Password	==	pass1
2	user2	Password	==	pass2

Ao "user1" ficou-lhe atribuída a password "pass1" e ao "user2" a password "pass2".

Na tabela **usergroup** associamos o user1 e o user2 ao grupo gatebox:

id	UserName	GroupName
1	user1	gatebox
2	user2	gatebox

Na tabela **radgroupcheck** definimos a autenticação como sendo local para o grupo Gate-Box:

id	GroupName	Attribute	Op	Value
1	gatebox	Auth-Type	:=	Local

Na tabela **radgroupreply** defenimos as propriedades das ligações:

id	GroupName	Attribute	Op	Value
1	gatebox	Framed-Compression	:=	Van-Jacobsen-TCP-IP
2	gatebox	Framed-Protocol	:=	PPP
3	gatebox	Service-Type	:=	Framed-User
4	gatebox	Framed-MTU	:=	1500

Encontra-se completa a base de dados para iniciar os testes finais com o servidor RADIUS para verificar se um utilizador é válido ou não.

É necessário ainda editar o ficheiro */etc/freeradius/clients.conf,* para que seja o nosso "RAS" (Resolve Address Server) virtual e acrescentar no final o seguinte:

```
...
  client 192.167.0.0/24 {     (a nossa sub-rede)
         secret = deec
         shortname = gatebox
         nastype = other
  }
...
```

Se o teste for feito na própria máquina basta substituir 192.167.0.0/24 por 127.0.0.1.

Podemos agora correr o servidor RADIUS em debug com o comando "freeradius –X" e executar o seguinte comando para testar:

```
$ radtest  user1  pass1  192.168.1.1 0  deec
```

Onde user1 é o username, pass1 a password do user, 192.168.1.1, o IP da máquina onde se encontra o servidor RADIUS e deec a password que se encontra no ficheiro *clients.conf.*

Após a execução do comando deverá aparecer algo de género como resposta:

```
   Sending Access-Request of id 25 to 192.168.1.1:1812
         User-Name = "user1"
         User-Password = "pass1"
         NAS-IP-Address = 192.168.1.1
         NAS-Port = 0
  rad_recv: Access-Accept packet from host 192.168.1.1:1812, id=25,
length=44
         Service-Type = Framed-User
         Framed-Protocol = PPP
         Framed-Compression = Van-Jacobson-TCP-IP
         Framed-MTU = 1500
```

O servidor RADIUS encontra-se operacional. No capítulo seguinte iremos falar da integração do PPPoE com o servidor RADIUS.

4.3 Point to Point Protocol over Ethernet - PPPoE

4.3.1 O que é o PPPoE

O **protocolo ponto-a-ponto (point-to-point protocol**, em inglês), também conhecido como **PPP**, foi desenvolvido e padronizado através da RFC 1548 (1993) com o objectivo de transportar todo o tráfego entre 2 dispositivos de rede através de uma conexão física única. Embora seja um protocolo, o PPP encontra-se na lista de interfaces. Na prática, a interface PPP é implementada através de conexões físicas. Actualmente é possível usar conexões PPP sobre Ethernet (PPPoE).

PPPoE (Point-to-Point Protocol over Ethernet) é um protocolo para conexão de utilizadores que se encontram numa rede Ethernet à Internet. É utilizado nas conexões de um ou múltiplos utilizadores numa rede LAN à Internet através de uma linha DSL, de um dispositivo wireless (sem fio) ou de um modem de cabo broadband comum. O protocolo PPPoE deriva do protocolo PPP. O PPPoE estabelece a sessão e realiza a autenticação com o fornecedor de acesso à Internet.

4.3.2 Integração do servidor PPPoE na Gatebox

O objectivo de um servidor PPPoE na GateBox é permitir aos utilizadoras autenticarem-se com o servidor RADIUS local e obterem um IP fixo que está associado ao seu username. O servidor PPPoE cria túneis entre a GateBox e os clientes autenticados, rejeitando o acesso a qualquer cliente que não se encontre autenticado.

O pacote de instalação para integrar o servidor PPPoE na GateBox foi o RP-PPPoE v3.8 disponível em http://www.roaringpenguin.com/penguin/pppoe/rp-pppoe-3.8.tar.gz.

Antes de compilar e instalar o rp-pppoe primeiro é necessário instalar o PPP daemon disponível nos pacotes do Ubuntu.

Após compilar e instalar o rp-pppoe no sistema é necessário configurá-lo e efectuar algumas alterações no servidor RADIUS.

Começar por remover todos os utilizadores que se encontrem nos ficheiros */etc/ppp/pap-secrets* e */etc/ppp/chap-secrets*. Incluir as seguintes linhas no ficheiro */etc/ppp/pppoe-server-options*

```
require-pap    (ou require-chap no caso de usarmos o procolo chap)
login
lcp-echo-interval 10
lcp-echo-failure 2
ms-dns ipdnsprimario
ms-dns ipdnssecundario
plugin radius.so
plugin radattr.so
```

É necessário editar o ficheiro */etc/radiusclient/servers* e colocar o endereço do servidor RADIUS e a senha dele. Editar também o ficheiro */etc/radiusclient/radiusclient.conf* e inserir em authserver e em acctserver o endereço do servidor RADIUS, na GateBox desenvolvida para os testes temos:

```
authserver  192.168.1.1:1812
acctserver  192.168.1.1:1813
```

Editar também o ficheiro */etc/freeradius/naslist* e colocar o IP do servidor RADIUS e colocar o tipo como "other".

O servidor PPPoE fica assim configurado e para o pôr a correr basta correr o comando:

```
$ pppoe-server -C Fornecedor -L enderecoiplocal -I ethX
```

(onde ethX = interface de rede que utilizará o PPPoE)

Se for pretendido atribuir classes de IPs aos clientes basta criar um ficheiro */etc/ppp/ips* e colocar a gama de endereços, como por exemplo "192.167.1.0-255". Basta agora colocar este ficheiro como argumento no comando anterior:

```
$ pppoe-server -C Fornecedor -L enderecoiplocal -p /etc/ppp/ips -I ethX
```

Assim o servidor PPPoE vai atribuir IPs dinâmicos aos utilizadores autenticados na gama 192.167.1.1-192.167.1.255.

Mas como já foi referido no capítulo 3.1, surge a necessidade de atribuir IPs aos clientes que se autenticam segundo a sua classe de serviço. A melhor solução encontrada foi atribuir IPs fixos a cada utilizador, por exemplo, um utilizador de classe 5 terá um IP 192.167.5.1 e um de classe 3 terá um IP 192.167.3.1.

Para implementar a atribuição de IPs por classes de serviço basta na base de dados RADIUS acrescentar na tabela **radreply** uma associação entre um utilizador e o seu IP, a seguir segue-se um exemplo da tabela:

id	UserName	Attribute	Value	Op
1	user1	Framed-IP-Address	192.167.5.1	:=
2	user2	Framed-IP-Address	192.167.3.1	:=

Sempre que o utilizador user1 se autenticar no sistema irá receber sempre o IP 192.167.5.1, que é um IP com classe 5, assim o FlowManager sabe automaticamente que se trata de um cliente com alta prioridade.

Após alguns testes reparou-se que o servidor RADIUS estava a permitir duas ou mais ligações simultâneas a um username, o que não é viável para o cenário da GateBox onde cada conta só pode ter uma ligação através do PPPoE e não várias em simultâneo à GateBox. Para ultrapassar este problema basta acrescentar na base de dados RADIUS na tabela **radgroupcheck** a linha com id 2:

id	GroupName	Attribute	Op	Value
1	gatebox	Auth-Type	:=	Local
2	gatebox	Simultaneous-Use	:=	1

Agora todos os utilizadores que se encontrem no grupo gatebox estão restringidos a uma ligação em simultâneo por conta.

Com o servidor PPPoE e o servidor RADIUS a funcionarem, os utlizadores conseguem-se autenticar facilmente no servidor RADIUS e toda a informação de accounting é armazenada na base de dados na tabela radacct.

4.3.3 Configuração dos clientes PPPoE

Para configurar a ligação de um cliente à GateBox por PPPoE no sistema operativo Windows é bastante directo. No painel de controlo seleccionar em "*Ligações de rede*", seleccionar "*Criar uma nova ligação*". Quando pedir para seleccionar o tipo de ligação de rede escolher "*Ligar à Internet*" e clicar em seguinte. Na secção "A preparar" seleccionar a opção "Configurar a minha ligação manualmente" e clicar em seguinte. Na secção "Ligação à Internet" seleccionar a opção "*Ligar utilizando uma ligação de banda larga que requer um nome de utilizador e palavre-passe*" e clicar em seguinte. No nome do ISP colocar "*Gatebox*" e clicar em seguinte, depois basta apenas introduzir o username e password. A configuração do cliente PPPoE fica assim concluída. A interface de ligação é igual à representada na Figura 5, basta clicar no ligar e se o username e password introduzidos forem válidos o utilizador fica autenticado com o servidor RADIUS.

Figura 5 – Cliente PPPoE em Windows

Na Figura 6 encontram-se os detalhes da ligação PPPoE em cliente Windows.

Figura 6 – Detalhes da ligação PPPoE

Para clientes em sistema operativo **Linux,** uma das várias possibilidades é compilar e instalar o pacote open-source RP-PPPoE v3.8 disponível em http://www.roaringpenguin.com/penguin/pppoe/rp-pppoe-3.8.tar.gz. Após compilado e instalado, basta correr o comando pppoe-setup na consola e configurar. Após configuração corre-se o comando pppoe-start e é efectuada a autenticação com o servidor RADIUS.

Capítulo 5

5. Gestão do sistema usando SNMP

O SNMP foi o protocolo seleccionado para executar as tarefas de gestão do sistema. Nas próximas secções serão descritas as principais características deste protocolo assim como a implementação de funcionalidades para expandir a capacidade do agente e assim responder aos requisitos de gestão das máquinas GateBox.

5.1 Gestão da rede e motivação para o SNMP

Enquadrando-se as máquinas GateBox (routers) num sistema integrado onde se inclui uma máquina central responsável pela administração das demais coloca-se a necessidade de introduzir mecanismos de gestão. Estes mecanismos de gestão deverão operar tanto a um nível local como a um nível inter-máquinas garantindo-se que o hardware e o software que compõe a rede são continuamente analisados na senda de falhas que comprometam a eficiência do sistema. Deste modo o sistema de gestão deverá questionar em cada máquina diversos parâmetros relativos ao hardware e software tais como interfaces de rede e processos do sistema operativo informando a máquina central do seu estado ou actuando de forma autónoma como resposta à detecção de uma anormalidade.

A escolha do sistema de gestão capaz de cumprir os propósitos referidos recaiu sobre o protocolo SNMP (do inglês *Simple Network Management Protocol* - Protocolo de Gestão Simples de Rede). Este protocolo, desde que foi desenvolvido em 1988 [RFC1067,1988], assumiu-se como o padrão para a gestão de redes dado tratar-se de uma solução simples, requerendo pouco código na sua implementação permitindo que os fabricantes construíssem facilmente agentes do SNMP para os seus produtos.

O SNMP é expansível permitindo que os fabricantes adicionem facilmente novos módulos de gestão aos produtos já existentes fazendo ainda a separação entre a arquitectura de gestão da arquitectura dos dispositivos físicos proporcionando flexibilidade na interacção do agente com dispositivos de diversos fabricantes. Por último refira-se que, ao contrário de outros padrões, o SNMP não é uma mera especificação no papel mas sim uma implementação que se encontra largamente disponível actualmente.

5.2 SNMP – Protocolo de gestão simples de rede

5.2.1 Arquitectura

O sistema de gestão de redes adopta um formato similar ao convencional **modelo cliente-servidor**, usando no entanto os termos "gestor" para a aplicação cliente e "agente" para a aplicação servidora.

O **gestor** possui ao seu dispor um conjunto de aplicações que permitem a interacção com os diversos agentes de modo a recolher informação ou executar determinadas acções de gestão. Poderá ainda receber eventos assíncronos originados no agente (traps), os quais são tipicamente gerados devidos à ocorrência de falhas.

O **agente** é a entidade que reside nos dispositivos a serem geridos fazendo a interface entre a informação disponível no dispositivo e o tipo de dados reconhecido pelo protocolo. Os dispositivos geridos podem assumir a forma de routers, switches, hubs, terminais e mesmo impressoras. No contexto do sistema GateBox a gestão de dispositivos centra-se exclusivamente nos routers.

Em cada dispositivo/router encontra-se definido um conjunto de **objectos** a serem geridos pelo agente. Estes objectos correspondem a diversas características da máquina, tanto a nível de software como ao nível físico, como é o exemplo de informação sobre o espaço disponível em disco, o número de octetos que atravessaram uma determinada interface de rede ou um parâmetro de configuração de um serviço do Sistema Operativo. Estes objectos encontram-se organizados de acordo com uma base de dados virtual designada por **Management Information Base (MIB)**. A forma como se encontra estruturada a informação nestas bases de dados virtuais encontra-se desenvolvida secção 5.2.3.

È ainda oportuno referir a existência de **sub-agentes**, os quais expandem a funcionalidade do agente principal. Estes sub-agentes são usados implementar secções específicas da estrutura MIB como por exemplo o conjunto de dados relativos às interfaces de rede. A criação destes sub-agentes encontra-se descrita na secção Desenvolvimento do agente.

Tal como descrito nos parágrafos anteriores a estrutura do sistema de gestão adopta o modelo ilustrado na Figura 7. Note-se no entanto que o modelo pode-se inverter dado que cada máquina poderá funcionar como gestor/agente. Destaca-se no entanto a posição do gestor como entidade central à qual os diversos agentes reportam o seu estado. Em ambos os casos, agente e gestor, implementam na totalidade o protocolo SNMP.

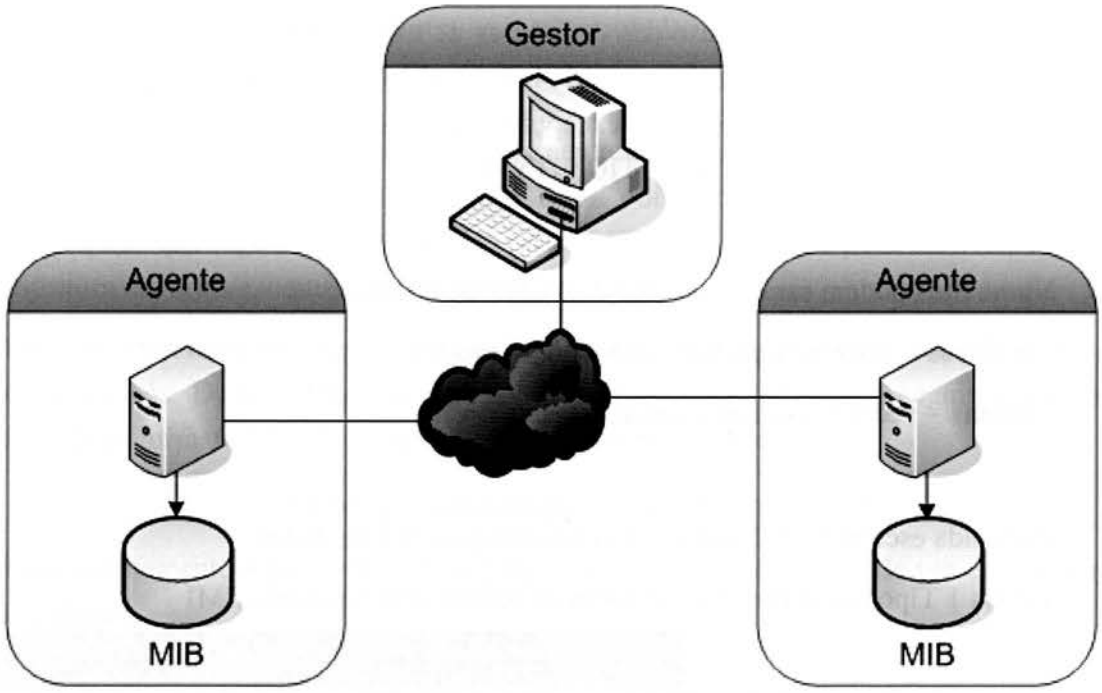


Figura 7 – Arquitectura do sistema de gestão(modelo cliente-servidor)

5.2.2 Estrutura da informação – SMI[4] e MIB

Os objectos acedidos pelo SNMP devem ser definidos numa estrutura de informação hierárquica designada por MIB (Management Information Base). A definição dos objectos na MIB é efectuada com recurso a uma sub-entidade do **ASN.1** (do inglês: ***Abstract Sintax Notation.1)*** designada por "**Structure Management Information**" ou SMI.

A **SMI** define o esquema da base de dados virtual que contém os objectos geridos. Como referido, segue o modelo definido pela notação simbólica ASN.1 a qual implementa um método de descrição de estruturas de dados abstractas [RFC1155].

Esta estrutura de gestão encontra-se dividida em 3 secções: definição de módulos, definição de objectos e definição de notificações.

- As **definições de módulos** são usadas para descrever módulos de informação. A declaração destes módulos é efectuada através da macro ASN.1, MODULE-IDENTITY.
- Os objectos são descritos usando-se as **definições de objectos**. Para tal é aplicada a macro OBJECT-TYPE.

[4] SMI – Structure Management Information

- Finalmente é feito uso das **definições de notificações** quando se pretende descrever transmissões assíncronas de informação (usadas tipicamente no envio de alertas devido à ocorrência de excepções num dado objecto - traps). Neste caso é invocada a macro NOTIFICATION-TYPE.

Numa SMI podem ser encontradas dois tipos de objectos:

Escalares – definem uma instância de um objecto;

Tabelas – vectores bi-dimensionais de escalares;

Para cada escalar a SMI, define um conjunto possível de dados:

Tabela 1 Tipos de dados dos escalares de acordo com o modelo SMI

	SMIv1	**SMIv2**
Tipos simples	INTEGER	INTEGER
	OBJECT STRING	OBJECT STRING
	OBJECT IDENTIFIER	OBJECT IDENTIFIER
	-	Integer32
Tipos de aplicação global	Unsigned32	Unsigned32
	Gauge32	Gauge32
	COUNTER	Counter32
		Counter64
	TIMETICKS	TimeTicks
	IPADDRESS	IpAddress
	OPAQUE	Opaque
	NETWORK ADDRESS	-
Pseudo-tipos	-	BITS

Para cada objecto é ainda necessário especificar o tipo de acesso, que se resume aos seguintes modos:

read-only – apenas possível leitura do conteúdo;

read-write – leitura e escrita na variável gerida

accessible-for-notify – associada a operações de lançamento de eventos assíncronos.

not-accessible – a variável encontra-se implementada mas não pode ser acedida.

Para que o objecto se encontre completamente definido é necessário identificá-lo univocamente na estrutura hierárquica das MIBs. È pois agora oportuno apresentar o modelo hierárquico que rege a distribuição das várias MIBs. Este modelo pode ser visto como uma estrutura em árvore, como esquematizado na Figura 8:

Pode-se observar neste esquema a existência do nó *root* a partir do qual derivam todos os restantes nós. Deste modo para endereçar o nó *gatebox* teríamos a seguinte disposição, similar ao formato URI[5] (o nó root é identificado pelo ponto inicial):

```
.iso.org.dod.internet.private.enterprises.gatebox
```

Equivalentemente é possível a identificação do nó gatebox através de um endereçamento numérico:

```
.1.3.6.1.4.1.8073
```

As duas representações anteriores recebem o nome de **Identificadores de Objectos** ou **OIDs** (Object IDs) e permitem uma identificação exclusiva dos diversos nós.

Até este momento foram feitas referências aos nós que constituem o modelo em arvóre. No entanto os conteúdos acedidos pelo SNMP encontram-se exclusivamente nas folhas. Estas extremidades são de facto as **instâncias** do objecto (nó ascendente).

O acesso ás instâncias faz-se atrelando um índice ao OID do objecto. O índice é um sufixo especificado pelo protocolo que permite identificar uma instância em particular. Exemplifica-se de seguida o acesso a estas instâncias, dependendo do tipo de objecto: escalar ou tabela.

No caso de um escalar apenas existe uma instância por objecto. O acesso faz-se especificando o índice 0. Assim, supondo que o nó gatebox é de facto uma folha (não possui descendentes), o acesso ao seu conteúdo é feito do seguinte modo:

```
.1.3.6.1.4.1.8073.0
```

No caso das tabelas além do índice é necessário especificar a coluna a que pertence a instância em particular. Assim, se o mesmo nó gatebox fosse o identificador de um objecto tabular, o acesso à quinta instância na coluna 2 é feito deste modo:

```
.1.3.6.1.4.1.8073.2.5
```

[5] URI – Universal Resource Identifier

O nó *gatebox* encontra-se registado como um descendente do nó *enterprises*, o qual tipicamente alberga MIBs de diversos fabricantes, as quais implementam estruturas de gestão de informação destinadas aos seus produtos.

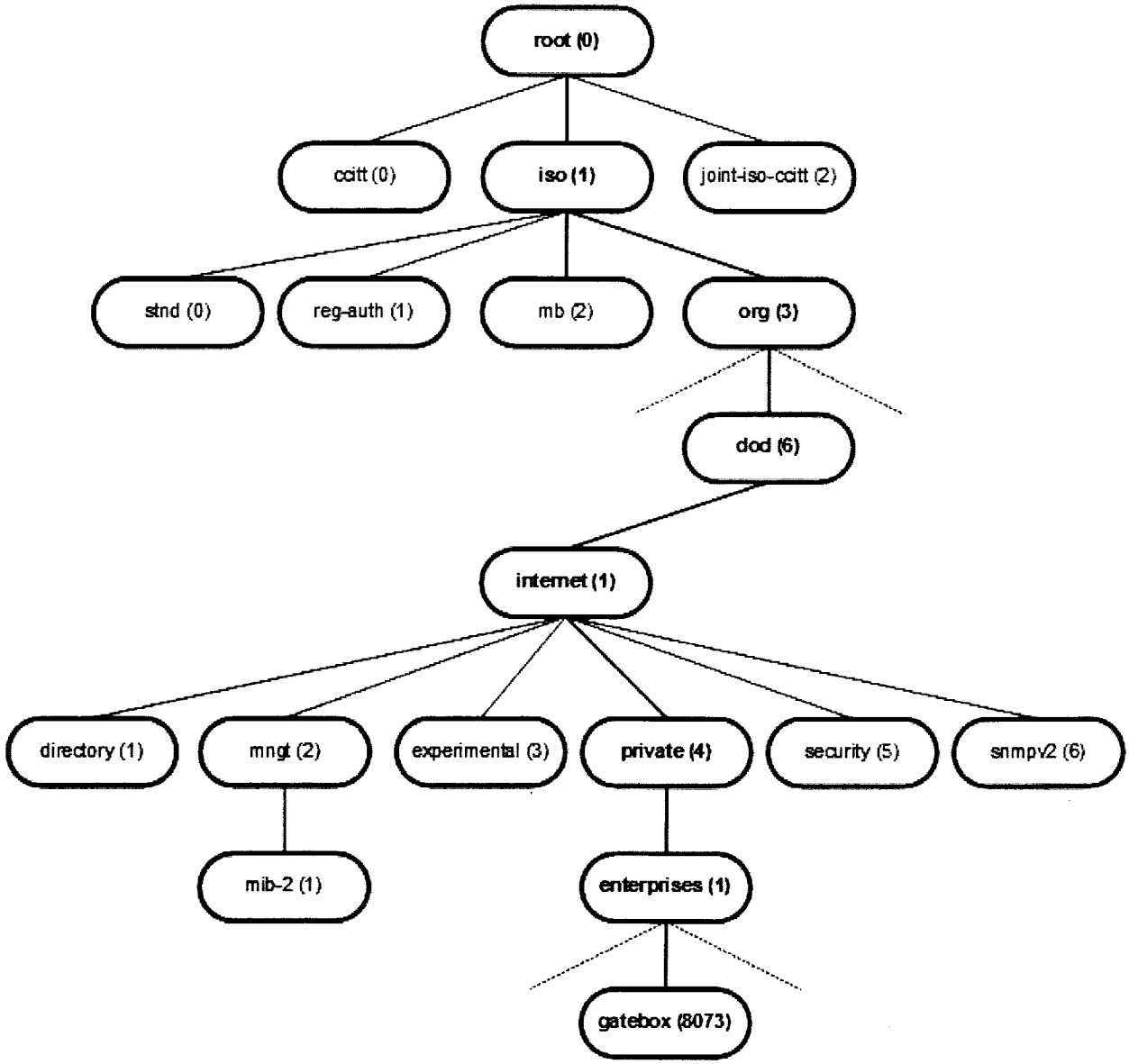


Figura 8 – Porção da árvore MIB onde se pode observar a localização do nó "gatebox".

O registo dos diversos nós é definido nas várias MIBs que compõem esta estrutura.

Para tornar clara a implementação de uma MIB através do modelo SMI observe-se a seguinte declaração formal:

```
GATEBOX-IP-MIB DEFINITIONS ::= BEGIN

IMPORTS
      GateBox                                    FROM GATEBOX-MIB
      MODULE-IDENTITY, OBJECT-TYPE,
            Integer32                 FROM SNMPv2-SMI
      MODULE-COMPLIANCE, OBJECT-GROUP           FROM SNMPv2-CONF;

gateBoxIp MODULE-IDENTITY
    LAST-UPDATED "200605040000Z"            -- 04 May 2006, midnight
    ORGANIZATION "INESC"
    CONTACT-INFO "Editor:   Bruno Oliveira & Vitor Brandão
                            Campus da FEUP
                              Rua Dr. Roberto Frias, 378
                              4200 - 465 Porto
                              Portugal

              email:    admin@gatebox.pt"

    DESCRIPTION  "The GateBox MIB"
    ::= { GateBox 4 }


gbIPv4Flags   OBJECT IDENTIFIER ::= { gateBoxIp 1 }

gbIpAutoconfig OBJECT-TYPE
    SYNTAX     INTEGER {
                    not(0),     -- ...
                    HostReceivedIpConfiguration(1)  -- ...
               }
    MAX-ACCESS read-write
    STATUS     current
    DESCRIPTION
           "[ /proc/sys/net/ipv4/ip_autoconfig ]
           This file contains the number one if the host received its IP
configuration by RARP, BOOTP, DHCP or a similar mechanism. Otherwise it
is zero.
"
    ::= { gbIPv4Flags 1 }

-- (...)

END
```

Esta declaração implementa a MIB GATEBOX-IP-MIB responsável por registar o nó gateBoxIp tal como definido pela directiva MODULE-IDENTITY:

```
gateBoxIp MODULE-IDENTITY
```

Este nó é um descendente do nó gatebox já apresentado. A identificação do nó ascendente e a posição que ocupa sob esse nó (4) é efectuada através da sintaxe:

```
::= { GateBox 4 }
```

Definido o módulo é necessário povoar esta nova estrutura com os objectos a gerir. Neste exemplo é criado um objecto (*gbIpAutoconfig*) onde é mapeado o valor presente no ficheiro "/proc/sys/net/ipv4/ip_autoconfig". Por uma questão de organização definiu-se o nó pai *gbIPv4Flags*. Este nó, é definido como um descendente do nó gateBoxIp:

```
gbIPv4Flags   OBJECT IDENTIFIER ::= { gateBoxIp 1 }
```

A criação do objecto *gbIpAutoconfig* é efectuada através da macro OBJECT-TYPE:

```
gbIpAutoconfig OBJECT-TYPE
```

Tal como nos casos anteriores é necessário especificar o nó pai e qual a posição (1) sob esse nó:

```
    ::= { gbIPv4Flags 1 }
```

De acordo com as declarações acima apresentadas o nó *gbIpAutoconfig* possui a seguinte OID:

```
.iso.org.dod.internet.private.enterprises.gatebox.gateBoxIp.gbIPv4Flags.gbIpAutoconfig
```

ou,

```
.1.3.6.1.4.1.8073.4.1.1
```

A implementação deste objecto só ficará completa após se caracterizar o tipo de dados associado e o tipo de acesso:

```
SYNTAX      INTEGER {
                    not(0),     -- ...
                    HostReceivedIpConfiguration(1)   -- ...
                }
    MAX-ACCESS read-write
    STATUS     current
    DESCRIPTION
            "[ /proc/sys/net/ipv4/ip_autoconfig ]
            This file contains the number one if the host received its IP
configuration by RARP, BOOTP, DHCP or a similar mechanism. Otherwise it
is zero.
```

Temos neste exemplo um objecto orientado para o manuseamento de inteiros (INTEGER) e com acesso para leitura e escrita (read-write). O acesso a este objecto para acções de leitura (get) e escrita (set) é descrito na próxima secção.

5.2.3 Protocolo e comandos básicos

O protocolo de gestão SNMP foi pensado com o propósito de ser um protocolo simples. Tendo este objectivo em mente os criadores do SNMP construíram-no sobre o protocolo de transporte **UDP**[6], o qual permite maior velocidade de processamento e uma maior simplicidade no formato das mensagens de protocolo. Ao invés, o protocolo TCP[7] apresenta-se como um protocolo complexo consumindo excessivamente recursos como memória e processador.

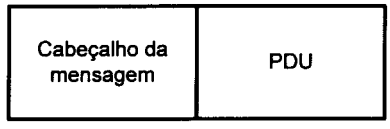
A simplicidade do protocolo UDP apresenta um custo relativamente à fiabilidade já que não existem mecanismos protocolo para confirmar que os dados chegaram ao destino correctamente. No entanto aplicações que usam um modelo pergunta-resposta são favorecidas com este protocolo tirando partido da resposta como reconhecimento positivo para a pergunta.

Sendo as comunicações de gestão de redes caracterizadas por breves trocas de mensagem no formato pergunta-resposta, o UDP apresenta-se como o candidato ideal para implementar a camada de transporte.

Relativamente às **portas UDP**, o agente SNMP escuta na porta 161 enquanto o gestor usa a porta 162. O gestor pode enviar pedidos a partir de qualquer porta (porta de origem) para a porta 161 no agente (porta de destino). A resposta do agente será reportada à porta de origem. As mensagens de traps são recebidas na porta 162, sendo originadas no agente em qualquer porta.

As mensagens trocadas entre gestor e agente adoptam um formato em concordância com a versão do SNMP a ser usada. Actualmente o SNMP conta com três versões (v1, v2c e v3), as quais são exploradas na próxima secção.

Nas versões 1 e 2, as mensagens SNMP são compostas por um cabeçalho e uma PDU.



[6] UDP – User Datagram Protocol

[7] TCP – Transmission Control Protocol

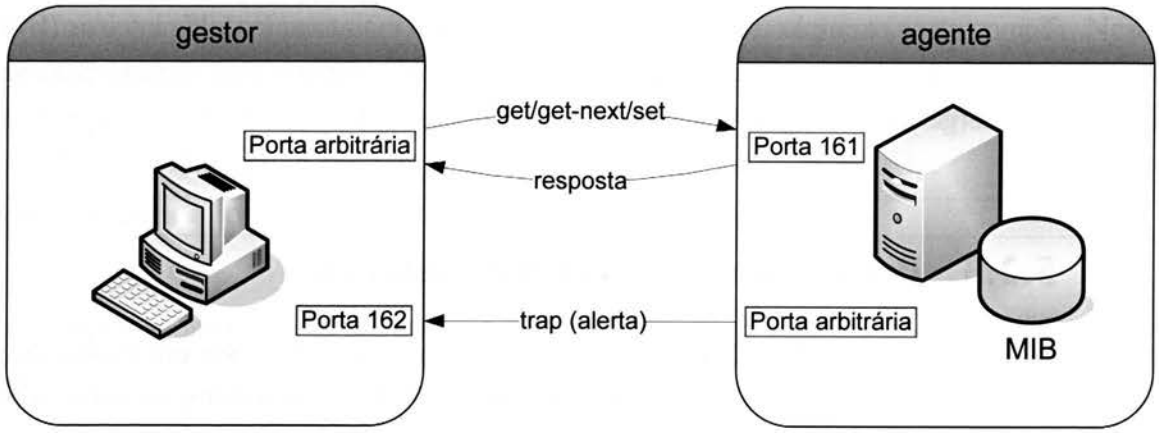


Figura 9- Uso das portas UDP na comunicação entre gestor e agente

O cabeçalho da mensagem possui 2 blocos, permitindo a especificação da versão SNMP e a da comunidade. A comunidade pode ser vista como um domínio administrativo e só se encontra presente nas duas primeiras versões.

Na PDU é especificado o tipo de PDU, os dados relativos às instâncias, um identificador de resposta e ainda dois campos de registo de erros. O tamanho da PDU é variável de acordo com o tipo de dados (ver Tabela 1) e quantidade de instâncias transportadas. O tamanho total da mensagem encontra-se limitado pela próprio tamanho máximo da mensagem UDP – 64KB.

O tipo da PDU define a operação SNMP a executar. Na sua senda por simplicidade, o protocolo SNMP inclui um reduzido número de operações ou comandos. Na versão 1 estão disponíveis cinco operações:

- **GetRequest** - Usada para ler os valores presentes nas variáveis.
- **GetNextRequest** – Usada para ler os valores das variáveis de modo sequencial. È deste modo apropriado para ler variáveis dispostas numa coluna de uma tabela. O primeiro elemento é acedido através de um comando GetRequest sendo aplicado o comando GetNextRequest aos elementos subsequentes.
- **Set** – Usada para actualizar o valor de uma variável.
- **GetResponse** – Resposta ao envio de uma mensagem GetRequest, GetNextRequest e Set.
- **Trap** – Lançada pelo agente SNMP para notificar o gestor SNMP da existência de um erro. Estas mensagens não são solicitadas.

Em versões posteriores foram adicionadas as operações:

- **GetBulkRequest** – Iterador usado para transferir grandes quantidades de informação do agente para o gestor eficientemente.

- **Inform** – Operação Trap com reconhecimento da recepção.

5.2.4 SNMP versões 1, 2 e 3

Em 1988 é lançado o RFC 1067 propondo o protocolo de gestão de redes SNMP. Esta proposta aparece suportada pelos RFCs 1065 e 1066, propostos no mesmo ano, que definem a estrutura de informação descrita no capítulo 5.2.2. Surge assim a **1ª versão** do SNMP. Esta implementação é caracterizada por uma segurança insuficiente.

A autenticação dos utilizadores é apenas feita com recurso à especificação do nome da **comunidade**, a qual é transportada nas mensagens SNMP sem qualquer tipo de encriptação. A cada comunidade está associado um tipo de acesso (leitura ou leitura/escrita). Assim quando um gestor envia uma mensagem identifica-se, definindo a comunidade a que pertence.

A **versão 2** inclui melhorias relativamente à eficiência, confidencialidade e segurança. A operação GetBulkRequest é apresentada nesta versão permitindo o pedido de grandes quantidades de informação num único pedido. No entanto a revisão do sistema de segurança, tornou o protocolo excessivamente complexo não tendo sido bem aceite. Como alternativa surge um modelo simplificado, baseado no modelo de comunidades, designado por SNMPv2c. Esta proposta tornou-se assim o padrão para a versão 2.

Finalmente, a **versão 3** do protocolo SNMP traz consigo claras melhorias relativamente à segurança tornado o uso do SNMP muito mais atractivo. O modelo de comunidades é deixado cair, optando-se pela definição de utilizadores SNMPv3. Para cada utilizador é possível a definição de uma passphrase para os algoritmos de autenticação (MD5) e encriptação (DES).

Dado a sua mais valia em termos de segurança, a versão 3 é usada em exclusivo no âmbito deste projecto.

5.3 Desenvolvimento do agente

5.3.1 Net-SNMP – ferramenta para implementação do SNMP

Concluídos os desenvolvimentos teóricos coloca-se agora a necessidade de colocar em prática o protocolo SNMP já descrito. O pacote **Net-SNMP**[8] disponibiliza um conjunto de aplicações.que permitem implementar as versões 1, 2c e 3 do protocolo SNMP.

Esta implementação disponibiliza aplicações que correm na linha de comandos permitindo a recolha de informação (**snmpget**, snmpgetnext, **snmpwalk**), a alteração de conteúdos (**snmpset**), a geração de traps (**snmptrap**) e ainda um comando de conversão entre o formato numérico e textual de OIDs assim como a visualização de excertos das MIBs (**snmptranslate**).

Estão ainda disponíveis dois daemons[9]:

- **snmpd**: implementa o agente SNMP respondendo a pedidos sobre a informação gerida. Suporta ainda os protocolos SNMP Multiplexing (SMUX) e Agent Extensibility (AgentX).
- **snmptrapd**: recebe notificações SNMP. Estas notificações podem ser registadas, enviadas para um gestor SNMP ou passadas para uma aplicação externa.

O Net-SNMP fornece ainda uma biblioteca para o desenvolvimento de aplicações SNMP tirando-se partido das **API**s disponibilizadas em C e Perl.

O pacote Net-SNMP foi seleccionado para implementar o protocolo SNMP dado tratar-se de sofware livre e aberto, com excelentes capacidades de expansão do agente a ainda pela vasta comunidade que o usa, a qual mantém uma forte actividade em várias mailing lists de suporte a este software.

Para a implementação do agente SNMP foi usada a **versão 5.3.0.1** (source code) a partir do site: http://sourceforge.net/project/showfiles.php?group_id=12694&package_id=11571&release_id=385258

[8] http://net-snmp.sourceforge.net/

[9] Programa de computador que corre em background sem necessitar de intervenção directa do utilizador

5.3.2 Definição das MIBs privadas

A implementação de uma MIB segundo o modelo SMI foi já apresentada na secção 5.2.2. Nesta secção pretende-se mostrar a estrutura de informação criada para albergar os diversos objectos a serem geridos num sistema GateBox.

Como referido na secção 5.2.2, as MIBs privadas, construídas pelas respectivas empresas, são colocadas sob o nó enterprises (ver Figura 8). Tendo o cuidado de se fazer um reconhecimento dos nós já ocupados seleccionou-se o nó 8073 para albergar o nó gatebox e a respectiva estrutura de informação.

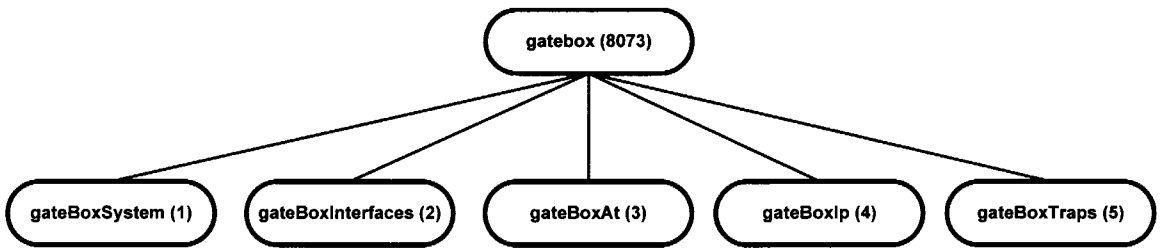


Figura 10- Nó gatebox e nós descendentes

Como observado na figura anterior encontram-se definidos cinco nós descendentes do nó gatebox. Destes nós destacaremos os nós gateBoxSystem, gateBoxInterfaces e GateBoxIp.

5.3.2.1 GateBoxSystem

Este nó é responsável pelo armazenamento de informação relativa a recursos do sistema como o disco e a memória comportando ainda uma estrutura responsável pela execução de determinadas acções mediante activação no lado do gestor,

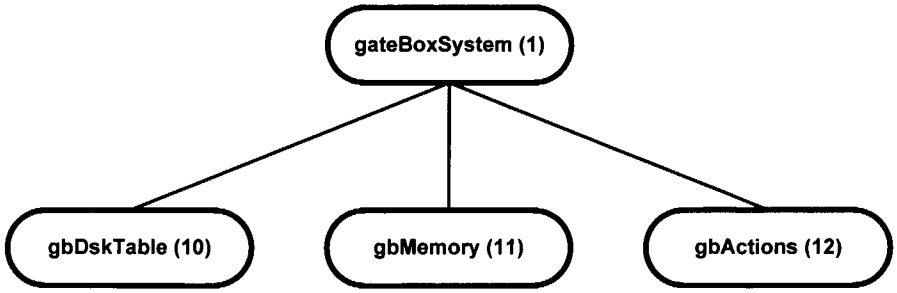


Figura 11- Árvore gateBoxSystem

O nó **gbDskTable** comporta os objectos que caracterizam os discos e partições presentes na máquina. O facto de possuir vários items (discos e/ou partições), os quais possuem idênticas características, justifica a utilização de **tabelas**.

Tal como referido na secção 5.2.2, uma tabela não é mais do que um vector bi-dimensional de escalares. A indexação de cada item é feita com recurso ao objecto gbDskIndex, o qual permite especificar qual dos itens a aceder.

Os nós **gbMemory** e **gbActions** são compostos por escalares simples já que neste caso existem múltiplos itens (a memória é contabilizada como um todo independentemente dos módulos presentes).

Com recurso ao comando snmptranslate é possível visualizar a estrutura de dados criada sob a alçada do nó gateBoxSystem:

```
+-gateBoxSystem(1)
|  |
|  +-gbDskTable(10)
|  |  |
|  |  +-gbDskEntry(1)
|  |     |  Index: gbDskIndex
|  |     |
|  |     +- -R-- Integer32 gbDskIndex(1)
|  |     |        Range: 0..65535
|  |     +- -R-- String    gbDskPath(2)
|  |     |        Textual Convention: DisplayString
|  |     |        Size: 0..255
|  |     +- -R-- String    gbDskDevice(3)
|  |     |        Textual Convention: DisplayString
|  |     |        Size: 0..255
|  |     +- -R-- Integer32 gbDskMinimum(4)
|  |     +- -R-- Integer32 gbDskMinPercent(5)
|  |     +- -R-- Integer32 gbDskTotal(6)
|  |     +- -R-- Integer32 gbDskAvail(7)
|  |     +- -R-- Integer32 gbDskUsed(8)
|  |     +- -R-- Integer32 gbDskPercent(9)
|  |     +- -R-- Integer32 gbDskPercentNode(10)
|  |     +- -R-- Integer32 gbDskErrorFlag(100)
|  |     +- -R-- String    gbDskErrorMsg(101)
|  |              Textual Convention: DisplayString
|  |              Size: 0..255
|  |
|  +-gbMemory(11)
|  |  |
|  |  +- -R-- Integer32 gbMemIndex(1)
|  |  +- -R-- String    gbMemErrorName(2)
|  |  |       Textual Convention: DisplayString
|  |  |       Size: 0..255
|  |  +- -R-- Integer32 gbMemTotalSwap(3)
|  |  +- -R-- Integer32 gbMemAvailSwap(4)
|  |  +- -R-- Integer32 gbMemTotalReal(5)
|  |  +- -R-- Integer32 gbMemAvailReal(6)
|  |  +- -R-- Integer32 gbMemTotalSwapTXT(7)
|  |  +- -R-- Integer32 gbMemAvailSwapTXT(8)
|  |  +- -R-- Integer32 gbMemTotalRealTXT(9)
```

```
    |   |  +- -R-- Integer32 gbMemAvailRealTXT(10)
    |   |  +- -R-- Integer32 gbMemTotalFree(11)
    |   |  +- -R-- Integer32 gbMemMinimumSwap(12)
    |   |  +- -R-- Integer32 gbMemShared(13)
    |   |  +- -R-- Integer32 gbMemBuffer(14)
    |   |  +- -R-- Integer32 gbMemCached(15)
    |   |  +- -R-- Integer32 gbMemSwapError(100)
    |   |  +- -R-- String    gbMemSwapErrorMsg(101)
    |   |           Textual Convention: DisplayString
    |   |           Size: 0..255
    |   |
    |   +-gbActions(12)
    |      |
    |      +- -RW- Integer32 gbDoReboot(1)
    |      +- -RW- Integer32 gbDoGateboxRestart(2)
    |      +- -RW- String    gbDoUpdateDb(3)
    |      |        Textual Convention: DisplayString
    |      |        Size: 0..255
    |      +- -RW- Integer32 gbDoGateboxFirewall(4)
    |      +- -RW- Integer32 gbDoSshdRestart(5)
    |      +- -RW- Integer32 gbDoFreeradiusRestart(6)
```

Nesta estrutura é visível a disposição hierárquica, o tipo de dados de cada objecto e o modo de acesso associado – leitura (R) e escrita (RW).

A declaração formal desta MIB está disponível no Anexo.A1 – gateBoxSystem.

5.3.2.2 GateBoxInterfaces

Este ramo da árvore lida com as interfaces de rede presentes num máquina GateBox. Tal como no caso dos discos, a presença de várias interfaces obriga à utilização do formato tabular. Esta tabela encontra-se alojada no nó gbIfTable.

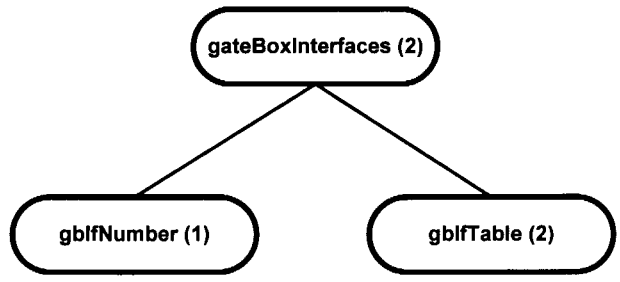


Figura 12 – Árvore gateBoxInterfaces

O comando snmptranslate permite observar a organização da informação nesta fracção da árvore gatebox:

```
    +-gateBoxInterfaces(2)
    |  |
    |  +- -R-- Integer32 gbIfNumber(1)
    |  |
```

```
    |  +-gbIfTable(2)
    |     |
    |     +-gbIfEntry(1)
    |        |  Index: gbIfIndex
    |        |
    |        +-gbIfIndex(1)
    |        |
    |        +- -R-- String    gbIfDescr(2)
    |        |        Textual Convention: DisplayString
    |        |        Size: 0..255
    |        +- -R-- EnumVal   gbIfType(3)
    |        |        Textual Convention: IANAifType
    |        |        Values: other(1), regular1822(2), (...), doc-
sCableMCmtsDownstream(229), adsl2(230)
    |        +- -R-- Integer32 gbIfMtu(4)
    |        +- -R-- Gauge     gbIfSpeed(5)
    |        +- -R-- String    gbIfPhysAddress(6)
    |        |        Textual Convention: PhysAddress
    |        +- -RW- EnumVal   gbIfAdminStatus(7)
    |        |        Values: up(1), down(2), testing(3)
    |        +- -R-- EnumVal   gbIfOperStatus(8)
    |        |        Values: up(1), down(2), testing(3), unknown(4),
dormant(5), notPresent(6), lowerLayerDown(7)
    |        +- -R-- TimeTicks gbIfLastChange(9)
    |        +- -R-- Counter   gbIfInOctets(10)
    |        +- -R-- Counter   gbIfInUcastPkts(11)
    |        +- -R-- Counter   gbIfInNUcastPkts(12)
    |        +- -R-- Counter   gbIfInDiscards(13)
    |        +- -R-- Counter   gbIfInErrors(14)
    |        +- -R-- Counter   gbIfInUnknownProtos(15)
    |        +- -R-- Counter   gbIfOutOctets(16)
    |        +- -R-- Counter   gbIfOutUcastPkts(17)
    |        +- -R-- Counter   gbIfOutNUcastPkts(18)
    |        +- -R-- Counter   gbIfOutDiscards(19)
    |        +- -R-- Counter   gbIfOutErrors(20)
    |        +- -R-- Gauge     gbIfOutQLen(21)
    |        +- -R-- ObjID     gbIfSpecific(22)
```

Note-se que se na declaração gbIfType se omitiram os tipos possíveis por uma questão de clareza.

```
Values: other(1), regular1822(2), (...), docsCableMCmtsDownstream(229),
adsl2(230)
```

No anexo A.2 pode-se observar a descrição formal desta estrutura de dados.

5.3.2.3 GateBoxIp

Finalmente, no nó gateBoxIp encontram-se implementadas alguns objectos criados com o propósito de aceder aos ficheiros presentes em */proc/sys/net/ipv4/*. Estes ficheiros permitem definir algumas opções globais no kernel do sistema operativo relativas ao protocolo IP. Os objectos correspondentes a cada ficheiro estão localizados sob a alçada do nó gbIPv4Flags.

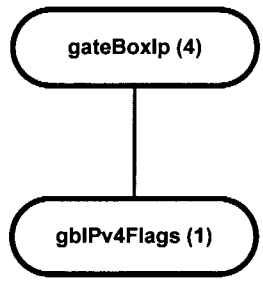


Figura 13 – Àrvore gateBoxIP

Para cada ficheiro encontra-se definido um escalar, onde se encontra armazenado o conteúdo do ficheiro correspondente. Os objectos implementados representam apenas uma amostra dos ficheiros a controlar:

```
+-gateBoxIp(4)
|  |
|  +-gbIPv4Flags(1)
|     |
|     +- -RW- EnumVal   gbIpAutoconfig(1)
|     |        Values: not(0), HostReceivedIpConfiguration(1)
|     +- -RW- Integer32 gbIpDefaultTtl(2)
|     |        Range: 1..255
|     +- -RW- INTEGER   gbIpDynaddr(3)
|     +- -RW- EnumVal   gbIpForward(4)
|              Values: forwarding(1), notForwarding(2)
```

O anexo A.3 permite observar a descrição formal deste ramo, inluindo na descrição dos objectos alguns comentários relativos à localização e função dos items a controlar.

5.3.3 Implementação de módulos

Nesta secção pretende-se apresentar a implementação dos módulos de software usados para preencher o conteúdo dos vários objectos descritos na secção anterior. Estes módulos de software são construídos tendo como base um esqueleto de código, em linguagem C, produzido pela ferramenta Net-SNMP. O código produzido varia de acordo com o tipo de objecto criado: escalares ou tabelas.

Nas próximas sub-secções mostram-se os procedimentos que permitiram a implementação dos objectos, apresentando-se quando oportuno excertos do código final. Os ficheiros mais relevantes podem ser consultados na secção B dos Anexos

5.3.3.1 gbDskTable

Como referido, a ferramenta Net-SNMP fornece um esqueleto de código para a implementação de objectos. A aplicação mib2c é a responsável por gerar código a partir da porção MIB especificada.

No caso do formato tabular o comando mib2c é invocado tendo como argumento o template mib2c.mfd.conf. A invocação do comando completa-se com a especificação do nó a gerir:

```
$ mib2c -c mib2c.mfd.conf gbDskTable
```

O comando anterior gera os seguintes ficheiros, contendo código base para edição:

```
gbDskTable.c
gbDskTable.h
gbDskTable_data_access.c
gbDskTable_data_access.h
gbDskTable_data_get.c
gbDskTable_data_get.h
gbDskTable_data_set.c
```

```
gbDskTable_data_set.h
gbDskTable_enums.h
gbDskTable_interface.c
gbDskTable_interface.h
gbDskTable_oids.h
gbDskTable_Makefile
gbDskTable_subagent.c
```

O modelo de código gerado faz uso intensivo das seguintes estruturas de dados:

- Data Context
- MIB Index
- Row Request Context

A estrutura ***data context*** implementa o armazenamento dos requeridos pela tabela gbDskTable. Observe-se de seguida a declaração da estrutura, feita no ficheiro gbDskTable.h:

```
/* data context
 * This structure contains storage for all the columns defined in the gbDskTable.
 */
typedef struct gbDskTable_data_s {

// gbDskPath(2)/DisplayString/ASN_OCTET_STR/char(char)//L/A/w/e/R/d/H
   char    gbDskPath[255];
   size_t      gbDskPath_len; /* # of char elements, not bytes */

// gbDskDevice(3)/DisplayString/ASN_OCTET_STR/char(char)//L/A/w/e/R/d/H
   char    gbDskDevice[255];
   size_t      gbDskDevice_len; /* # of char elements, not bytes */

// gbDskMinimum(4)/INTEGER32/ASN_INTEGER/long(long)//l/A/w/e/r/d/h
   long    gbDskMinimum;

// gbDskMinPercent(5)/INTEGER32/ASN_INTEGER/long(long)//l/A/w/e/r/d/h
  long    gbDskMinPercent;

// gbDskTotal(6)/INTEGER32/ASN_INTEGER/long(long)//l/A/w/e/r/d/h
  long    gbDskTotal;
```

```
// gbDskAvail(7)/INTEGER32/ASN_INTEGER/long(long)//l/A/w/e/r/d/h
  long    gbDskAvail;

// gbDskUsed(8)/INTEGER32/ASN_INTEGER/long(long)//l/A/w/e/r/d/h
   long    gbDskUsed;

// gbDskPercent(9)/INTEGER32/ASN_INTEGER/long(long)//l/A/w/e/r/d/h
   long    gbDskPercent;

// gbDskPercentNode(10)/INTEGER32/ASN_INTEGER/long(long)//l/A/w/e/r/d/h
   long    gbDskPercentNode;

// gbDskErrorFlag(100)/INTEGER32/ASN_INTEGER/long(long)//l/A/w/e/r/d/h
   long    gbDskErrorFlag;

// gbDskErrorMsg(101)/DisplayString/ASN_OCTET_STR/char(char)//L/A/w/e/R/d/H
   char    gbDskErrorMsg[255];
   size_t      gbDskErrorMsg_len; /* # of char elements, not bytes */

} gbDskTable_data;
```

Note-se nesta declaração a correspondência entre os tipos de dados do código gerado (linguagem C) e o formato dos dados presentes na MIB gbDskTable, lidos pela ferramenta mib2c.

Após a definição da coluna modelo da tabela é necessário criar um índice que permita referenciar cada coluna que integre essa tabela. Como referido anteriormente o número de colunas será igual ao número de itens geridos, ou seja o número de partições existentes na máquina.

A seguinte declaração cria a variável *gbDskIndex* onde será especificado o índice actual.

```
/*
 * This structure is used to represent the index for gbDskTable.
 */
typedef struct gbDskTable_mib_index_s {

//  gbDskIndex(1)/INTEGER32/ASN_INTEGER/long(long)//l/A/w/e/R/d/h
   long    gbDskIndex;

} gbDskTable_mib_index;
```

Por último apresenta-se a estrutura ***row request***. Esta estrutura é usada durante o processamento de um pedido SNMP contendo as estruturas descritas anteriormente assim como alguns itens adicionais.

O código que implementa esta estrutura encontra-se igualmente no ficheiro *gbDskTable.h*:

```
/*
 * When your functions are called, you will be passed a
 * gbDskTable_rowreq_ctx pointer.
 */
typedef struct gbDskTable_rowreq_ctx_s {

    /* this must be first for container compare to work */
    netsnmp_index        oid_idx;
    oid                  oid_tmp[MAX_gbDskTable_IDX_LEN];

    gbDskTable_mib_index        tbl_idx;
```

```
    gbDskTable_data              data;
    /*
     * flags per row. Currently, the first (lower) 8 bits are reserved
     * for the user. See mfd.h for other flags.
     */
    u_int                        rowreq_flags;
    /*
     * storage for future expansion
     */
    netsnmp_data_list             *gbDskTable_data_list;

} gbDskTable_rowreq_ctx;
```

Após a implementação do código base é necessário implementar as funções de acesso aos dados. Estas funções são responsáveis pela leitura dos dados disponíveis e pela definição dos índices. O acesso aos dados está implementado no ficheiro gbDskTable_data_access.c.

No kernel Linux a informação relativa ao disco está disponível através do comando *df* [die.net, 2006]. A aquisição de dados do sistema faz-se assim pela interpretação da informação gerada pelo comando. Para ser possível analisar o seu conteúdo é feito o redireccionamento do "output" para o ficheiro auxiliar *df.netsnmp*.

No ficheiro df.netsnmp fica assim disponível para consulta os seguintes dados:

```
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/hda2             19534436   4029452  15504984  21% /
tmpfs                   258204         0    258204   0% /dev/shm
tmpfs                    10240      2608      7632  26% /dev
```

A geração do ficheiro *df.netsnmp* e a sua interpretação podem ser visualizadas no seguinte excerto da função gbDskTable_container_load implementada no ficheiro gbDskTable_data_access.c:

```
    /*
     * open our data file.
     */
    system("df > /home/vbs/.snmp/docs/df.netsnmp");
    filep = fopen("/home/vbs/.snmp/docs/df.netsnmp", "r");
    //filep = fopen("/proc/net/dev", "r");
    if(NULL ==  filep) {
        return MFD_RESOURCE_UNAVAILABLE;
    }
    /*
     * create socket for ioctls
     */
    fd = socket(AF_INET, SOCK_DGRAM, 0);
    if(fd < 0) {
        fclose(filep);
        return MFD_RESOURCE_UNAVAILABLE;
    }
    /*
     * ignore header line
     */
    fgets(line, sizeof(line), filep);

    while( 1 ) {
```

```
        char   *stats, *filesystem;
        static const char     *scan_line = "%lu %lu %lu %lu %*s %s";
        static char     scan_expected = 5;
        int    scan_count;
        unsigned long size, used, avail, use_pc;
        char   mounted_on[255];
    /*
     * get a line (skip blank lines)
     */
    do {
        if (!fgets(line, sizeof(line), filep)) {
            fclose(filep);
            filep = NULL;
        }
    } while (filep && (line[0] == '\n'));

    /*
     * check for end of data
     */
    if(NULL == filep)
        break;
    /*
     * parse line into variables
     */
   filesystem = line;

            while (*filesystem && *filesystem == ' ')
            filesystem++;
              if ((!*filesystem) || ((stats = strchr(filesystem, ' ')) == NULL))
{
            snmp_log(LOG_ERR,
                     "interface data format error 1, line ==|%s|\n",
                     line);
            DEBUGMSGTL(("gbIfTable", "found '%s'\n", filesystem));
            continue;
        }
        /*
         * If we've met this filesystem before, use the same index.
         * Otherwise find an unused index value and use that.
         */
        *stats++ = 0; /* null terminate name */
              gbDskIndex = se_find_value_in_slist("filesystems", filesystem);
        if (gbDskIndex == SE_DNE) {
            gbDskIndex = se_find_free_value_in_slist("filesystems");
            if (gbDskIndex == SE_DNE)
                gbDskIndex = 1;       /* Completely new list! */
            se_add_pair_to_slist("filesystems",
                                 strdup(filesystem), gbDskIndex);
            DEBUGMSGTL(("gbDskTable","new gbDskIndex %d for %s\n",gbDskIndex,
filesystem));
        }
        if ((NULL == filesystem) || (0 == gbDskIndex)) {
            rc = MFD_END_OF_DATA;
            break;
        }

        rowreq_ctx = gbDskTable_allocate_rowreq_ctx();
        if (NULL == rowreq_ctx) {
            snmp_log(LOG_ERR, "memory allocation failed\n");
            return MFD_RESOURCE_UNAVAILABLE;
        }
        if(MFD_SUCCESS != gbDskTable_indexes_set(rowreq_ctx
                                 , gbDskIndex
               )) {
            snmp_log(LOG_ERR,"error setting index while loading "
                     "gbDskTable data.\n");
            gbDskTable_release_rowreq_ctx(rowreq_ctx);
```

```
            continue;
        }

        size = used = avail = use_pc = 0;
        mounted_on[0] = 0;

        scan_count = sscanf(stats, scan_line, &size, &used, &avail, &use_pc,
mounted_on);
        if(scan_count != scan_expected) {
            snmp_log(LOG_ERR, "error scanning interface data (expected %d, got
%d)\n", scan_expected, scan_count);
            rc = MFD_ERROR;
            break;
        }
```

Cada coluna da tabela gbDskTabela refere-se a uma partição do sistema de ficheiros. A distribuição das partições por colunas e a respectiva actualização dos dados é feita com o auxílio da função *se_find_value_in_slist* que permite saber se a última partição analisada já se encontra registada na tabela.

```
gbDskIndex = se_find_value_in_slist("filesystems", filesystem);
```

Colectados os dados fornecidos pelo kernel actualizam-se os campos da estrutura *data context* para reflectir os dados actuais presentes no sistema:

```
    /*
     * Populate gbDskTable data context here.
     *
     * setup/save data for gbDskPath
     * gbDskPath(2)/DisplayString/ASN_OCTET_STR/char(char)//L/A/w/e/R/d/H
     */
    rowreq_ctx->data.gbDskPath_len = strlen(mounted_on)* sizeof(mounted_on[0]);
    memcpy( rowreq_ctx->data.gbDskPath, mounted_on, rowreq_ctx-
>data.gbDskPath_len );
    /*
     * setup/save data for gbDskDevice
     * gbDskDevice(3)/DisplayString/ASN_OCTET_STR/char(char)//L/A/w/e/R/d/H
     */
     rowreq_ctx->data.gbDskDevice_len = strlen(filesystem)*
sizeof(filesystem[0]);
    memcpy( rowreq_ctx->data.gbDskDevice, filesystem, rowreq_ctx-
>data.gbDskDevice_len );
    /*
     * setup/save data for gbDskMinimum
     * gbDskMinimum(4)/INTEGER32/ASN_INTEGER/long(long)//l/A/w/e/r/d/h
     */
    rowreq_ctx->data.gbDskMinimum = size*MIN_PERCENT;
    /*
     * setup/save data for gbDskMinPercent
     * gbDskMinPercent(5)/INTEGER32/ASN_INTEGER/long(long)//l/A/w/e/r/d/h
     */
    rowreq_ctx->data.gbDskMinPercent = MIN_PERCENT*100;
    /*
     * setup/save data for gbDskTotal
     * gbDskTotal(6)/INTEGER32/ASN_INTEGER/long(long)//l/A/w/e/r/d/h
     */
    rowreq_ctx->data.gbDskTotal = size;
    /*
     * setup/save data for gbDskAvail
     * gbDskAvail(7)/INTEGER32/ASN_INTEGER/long(long)//l/A/w/e/r/d/h
     */
```

```
    rowreq_ctx->data.gbDskAvail = avail;
    /*
     * setup/save data for gbDskUsed
     * gbDskUsed(8)/INTEGER32/ASN_INTEGER/long(long)//l/A/w/e/r/d/h
     */
    rowreq_ctx->data.gbDskUsed = used;
    /*
     * setup/save data for gbDskPercent
     * gbDskPercent(9)/INTEGER32/ASN_INTEGER/long(long)//l/A/w/e/r/d/h
     */
    rowreq_ctx->data.gbDskPercent = use_pc;
    /*
     * setup/save data for gbDskPercentNode
     * gbDskPercentNode(10)/INTEGER32/ASN_INTEGER/long(long)//l/A/w/e/r/d/h
     */
    rowreq_ctx->data.gbDskPercentNode = 0;
    /*
     * setup/save data for gbDskErrorFlag
     * gbDskErrorFlag(100)/INTEGER32/ASN_INTEGER/long(long)//l/A/w/e/r/d/h
     */
    if ( use_pc > (1-MIN_PERCENT)*100 )
      rowreq_ctx->data.gbDskErrorFlag = 1;
    else
      rowreq_ctx->data.gbDskErrorFlag = 0;
    /*
     * setup/save data for gbDskErrorMsg
     * gbDskErrorMsg(101)/DisplayString/ASN_OCTET_STR/char(char)//L/A/w/e/R/d/H
     */
    /*
     * make sure there is enough space for gbDskErrorMsg data
     */
      sprintf(rowreq_ctx->data.gbDskErrorMsg, "Atenção! Apenas %lu%% de espaço
livre em disco.\n",100-use_pc);
      rowreq_ctx->data.gbDskErrorMsg_len = strlen(rowreq_ctx-
>data.gbDskErrorMsg);
       /*
        * insert into table container
        */
       CONTAINER_INSERT(container, rowreq_ctx);
       ++count;
    }
```

Estes dados podem agora ser acedidos através das rotinas presentes no ficheiro *gbDskTable_data_get.c*, que fazem a interface entre o pedido de dados SNMP e a estrutura de armazenamento dos mesmos.

Este módulo é então compilado gerando-se a aplicação gbDskTable responsável pela gestão das variáveis da tabela gbDskTable. A aplicação é então lançada, anexando-se ao agente SNMP, através do método **AgentX**[10]. Este é o método usado para o controlo de objectos tabulares. Este método diverge do caso escalar, apresentado na próxima secção, onde o módulo de software é integrado no próprio agente. Correndo-se o comando snmpwalk para percorrer todas as instâncias da tabela obtém-se o seguinte resultado:

```
$ snmpwalk localhost gbDskTable
GATEBOX-SYSTEM-MIB::gbDskIndex.1 = INTEGER: 1
```

[10] AgentX (Agent eXtensibility) – Protocolo que permite que vários sub-agentes se conectem ao agente SNMP trocando informação sobre os objectos por eles geridos.

```
GATEBOX-SYSTEM-MIB::gbDskIndex.2 = INTEGER: 2
GATEBOX-SYSTEM-MIB::gbDskPath.1 = STRING: /
GATEBOX-SYSTEM-MIB::gbDskPath.2 = STRING: /dev/shm
GATEBOX-SYSTEM-MIB::gbDskDevice.1 = STRING: /dev/hda2
GATEBOX-SYSTEM-MIB::gbDskDevice.2 = STRING: tmpfs
GATEBOX-SYSTEM-MIB::gbDskMinimum.1 = INTEGER: 976721
GATEBOX-SYSTEM-MIB::gbDskMinimum.2 = INTEGER: 12910
GATEBOX-SYSTEM-MIB::gbDskMinPercent.1 = INTEGER: 5
GATEBOX-SYSTEM-MIB::gbDskMinPercent.2 = INTEGER: 5
GATEBOX-SYSTEM-MIB::gbDskTotal.1 = INTEGER: 19534436
GATEBOX-SYSTEM-MIB::gbDskTotal.2 = INTEGER: 258204
GATEBOX-SYSTEM-MIB::gbDskAvail.1 = INTEGER: 15504984
GATEBOX-SYSTEM-MIB::gbDskAvail.2 = INTEGER: 258204
GATEBOX-SYSTEM-MIB::gbDskUsed.1 = INTEGER: 4029452
GATEBOX-SYSTEM-MIB::gbDskUsed.2 = INTEGER: 0
GATEBOX-SYSTEM-MIB::gbDskPercent.1 = INTEGER: 21
GATEBOX-SYSTEM-MIB::gbDskPercent.2 = INTEGER: 0
GATEBOX-SYSTEM-MIB::gbDskPercentNode.1 = INTEGER: 0
GATEBOX-SYSTEM-MIB::gbDskPercentNode.2 = INTEGER: 0
```

5.3.3.2 gbActions

Os objectos descendentes do nó gbActions foram criados tendo como objectivo a execução de determinadas rotinas no agente alvo. Estas rotinas estão definidas em scripts sh e pretendem executar acções como a reinicialização do sistema, a actualização da base de dados Radius e a reinicialização do servidor Radius. A implementação destas funcionalidades, como alternativa ao método de administração adoptado no capítulo 6, permite que exista comunicação com as máquinas GateBox mesmo em caso de falha do servidor ssh. Este é um dos aspectos que reflecte a mais valia da integração do protocolo SNMP como serviço de gestão, e administração, da rede GateBox.

Os objectos implementados podem ser observados recorrendo-se novamente ao comando snmptranslate para imprimir a estrutura de dados:

```
    |  +--gbActions(12)
    |     |
    |     +-- -RW- Integer32 gbDoReboot(1)
    |     +-- -RW- Integer32 gbDoGateboxRestart(2)
    |     +-- -RW- String    gbDoUpdateDb(3)
    |     |        Textual Convention: DisplayString
    |     |        Size: 0..255
    |     +-- -RW- Integer32 gbDoGateboxFirewall(4)
    |     +-- -RW- Integer32 gbDoSshdRestart(5)
    |     +-- -RW- Integer32 gbDoFreeradiusRestart(6)
```

Neste caso os objectos são do tipo escalar, ao invés do modelo tabular apresentado no nó gbDskTable. Para se criar a base de código C capaz de suportar esta implementação recorreu-se novamente ao comando mib2c especificando-se o template mib2c.scalar.conf.

```
mib2c -c mib2c.scalar.conf gbActions
```

Da execução do comando resultam os ficheiros gbActions.c e gbActions.h. Estes ficheiros são compilados no agente não sendo necessário lançar uma aplicação externa como no caso anterior (gbDskTable). Para cada escalar define-se a respectiva OID e regista-se o handler que irá lidar com os pedidos de acesso à variável. Este procedimento é exemplificado de seguida com o objecto gbGateboxForward.

Inicialmente identifica-se o objecto a controlar através da OID correspondente:

```
oid gbDoGateboxRestart_oid[]  = { 1,3,6,1,4,1,8073,1,12,2 };
```

Esta OID é então passada como argumento à função *netsnmp_register_scalar*, em conjunto com o *handler handle_gbDoGateboxRestart* e o modo de acesso à variável (leitura e escrita).

```
netsnmp_register_scalar(
    netsnmp_create_handler_registration(
                "gbDoGateboxRestart",
                 handle_gbDoGateboxRestart,
                 gbDoGateboxRestart_oid,
                 OID_LENGTH(gbDoGateboxRestart_oid),
                 HANDLER_CAN_RWRITE
    ));
```

O escalares criados na estrutura gbActions foram pensados para serem utilizados como triggers já que têm como propósito a execução de um ficheiro scríptico associado. Assim, os escalares definidos como inteiros possuem sempre o valor 0. A variável gbDoUpdateDb foi definida com o formato string, possuindo como valor normal NULL, ou seja uma string vazia. Para que ocorra uma acção é necessário fazer uma operação SET como o valor 1 para os escalares inteiros, ou com uma string no escalar gbDoUpdateDb.

Sempre que uma operação SNMP for invocada para a variável gbDoGateboxRestart é chamado o respectivo handler:

```
int
handle_gbDoReboot(
                    netsnmp_mib_handler *handler,
                    netsnmp_handler_registration *reginfo,
                    netsnmp_agent_request_info   *reqinfo,
                    netsnmp_request_info         *requests)
                   );
```

A variável C *val* é usada para armazenar o conteúdo do escalar, a qual possui normalmente, como referido, o valor 0:

```
int val = 0;
```

A operação invocada é reconhecida inquirindo-se a variável *reqinfo->mode*:

```
    switch(reqinfo->mode) {
```

Para desencadear uma acção é necessário correr o comando set com o valor 1:

```
$ snmpset localhost gbDoGateboxRestart = 1
```

Quando o comando set é invocado é executada a seguinte secção do código:

```
        case MODE_SET_ACTION:
            DEBUGMSGTL(("gbDoGateboxRestart","MODE_SET_ACTION\n"));
            val = *requests->requestvb->val.integer;
            if (val == 1) {
             DEBUGMSGTL(("gbDoGateboxRestart ","System is going to reboot\n"));
             ActionDoGateboxRestart();
            }
            else {
             DEBUGMSGTL(("gbDoGateboxRestart ","System will NOT be re-
booted\n"));
            }
            break;
```

Para se conhecer o inteiro introduzido na linha de comandos inquire-se o campo *integer* da estrutura *val*:

```
            val = *requests->requestvb->val.integer;
```

A variável é então comparada com o valor sendo chamada a função *ActionDoReboot* no caso afirmativo. Esta função encarrega-se da chamada do ficheiro sh *gatebox_restart.sh*:

```
      system("/home/master/sh/gatebox_restart.sh");
```

A chamada de um ficheiro externo, não se integrando o seu conteúdo no ficheiro *gbActions.c*, justifica-se pela possibilidade de se efectuarem alterações sem necessidade de se recompilar o agente.

5.3.3.3 gbIfTable

A implementação do objecto gbIfTable é em tudo similar ao objecto gbDskTable, já descrita em detalhe na secção 5.3.3.1.

Neste caso os dados encontram-se disponíveis no ficheiro */proc/net/dev* o qual contém informação referente às interfaces de rede presentes na máquina, como se pode observar de seguida:

```
Inter-|   Receive                                                |
Transmit
 face |bytes    packets errs drop fifo frame compressed multicast|bytes
packets errs drop fifo colls carrier compressed
    lo: 4126090   32904    0    0    0     0          0          0
4126090   32904    0    0    0     0       0          0
  eth0:29637626  345302    1    0    0     0          0          0
782213721  539560    0    0    0 129727       0          0
```

```
  eth1:   36072     284    1    0    0     0          0          0
102444     544    4    0    1     0       3          0
  eth2:46741901   66596    1    0    0     0          0          0
2391763   24763    6    0    2     0       4          0
  eth3:111436915  120485    1    0    0     0          0          0
8078180   53594    9    0    3     0       6          0
  sit0:       0       0    0    0    0     0          0          0
0       0    0    0    0     0       0          0
  imq0: 7699151   63594    0    0    0     0          0          0
7699151   63594    0    0    0     0       0          0
  imq1:143993391  109226    0    0    0     0          0          0
143993391  109226    0    0    0     0       0          0
  ppp0:16269823  312757    0    0    0     0          0          0
718479842  490180    0    0    0     0       0          0
  ppp1:  210863    1279    0    0    0     0          0          0
914079    1445    0    0    0     0       0          0
```

A execução do comando snmpwalk coloca em evidência os dados disponibilizados por este módulo:

```
$snmpwalk localhost gbIfTable
GATEBOX-INTERFACES-MIB::gbIfIndex.1 = INTEGER: 1
GATEBOX-INTERFACES-MIB::gbIfIndex.2 = INTEGER: 2
GATEBOX-INTERFACES-MIB::gbIfIndex.3 = INTEGER: 3
GATEBOX-INTERFACES-MIB::gbIfIndex.4 = INTEGER: 4
GATEBOX-INTERFACES-MIB::gbIfIndex.5 = INTEGER: 5
GATEBOX-INTERFACES-MIB::gbIfIndex.6 = INTEGER: 6
GATEBOX-INTERFACES-MIB::gbIfIndex.7 = INTEGER: 7
GATEBOX-INTERFACES-MIB::gbIfIndex.8 = INTEGER: 8
GATEBOX-INTERFACES-MIB::gbIfIndex.9 = INTEGER: 9
GATEBOX-INTERFACES-MIB::gbIfType.1 = INTEGER: softwareLoopback(24)
GATEBOX-INTERFACES-MIB::gbIfType.2 = INTEGER: ethernetCsmacd(6)
GATEBOX-INTERFACES-MIB::gbIfType.3 = INTEGER: ethernetCsmacd(6)
GATEBOX-INTERFACES-MIB::gbIfType.4 = INTEGER: ethernetCsmacd(6)
GATEBOX-INTERFACES-MIB::gbIfType.5 = INTEGER: ethernetCsmacd(6)
GATEBOX-INTERFACES-MIB::gbIfType.6 = INTEGER: tunnel(131)
GATEBOX-INTERFACES-MIB::gbIfType.7 = INTEGER: other(1)
GATEBOX-INTERFACES-MIB::gbIfType.8 = INTEGER: other(1)
GATEBOX-INTERFACES-MIB::gbIfType.9 = INTEGER: ppp(23)
GATEBOX-INTERFACES-MIB::gbIfPhysAddress.1 = STRING: 0:0:0:0:0:0
GATEBOX-INTERFACES-MIB::gbIfPhysAddress.2 = STRING: 0:80:c8:f8:45:1d
GATEBOX-INTERFACES-MIB::gbIfPhysAddress.3 = STRING: 0:80:c8:f8:45:1e
GATEBOX-INTERFACES-MIB::gbIfPhysAddress.4 = STRING: 0:2:3f:34:aa:f0
GATEBOX-INTERFACES-MIB::gbIfPhysAddress.5 = STRING: 0:80:c8:f8:45:20
GATEBOX-INTERFACES-MIB::gbIfPhysAddress.6 = STRING: 0:0:0:0:45:20
GATEBOX-INTERFACES-MIB::gbIfPhysAddress.7 = STRING: 0:0:0:0:0:0
GATEBOX-INTERFACES-MIB::gbIfPhysAddress.8 = STRING: 0:0:0:0:0:0
GATEBOX-INTERFACES-MIB::gbIfPhysAddress.9 = STRING: 0:0:0:0:0:0
GATEBOX-INTERFACES-MIB::gbIfInOctets.1 = Counter32: 3855624
GATEBOX-INTERFACES-MIB::gbIfInOctets.2 = Counter32: 16197913
GATEBOX-INTERFACES-MIB::gbIfInOctets.3 = Counter32: 36072
GATEBOX-INTERFACES-MIB::gbIfInOctets.4 = Counter32: 4834509
GATEBOX-INTERFACES-MIB::gbIfInOctets.5 = Counter32: 62176032
GATEBOX-INTERFACES-MIB::gbIfInOctets.6 = Counter32: 0
GATEBOX-INTERFACES-MIB::gbIfInOctets.7 = Counter32: 5671264
GATEBOX-INTERFACES-MIB::gbIfInOctets.8 = Counter32: 53986088
GATEBOX-INTERFACES-MIB::gbIfInOctets.9 = Counter32: 7083217
GATEBOX-INTERFACES-MIB::gbIfInUcastPkts.1 = Counter32: 31172
GATEBOX-INTERFACES-MIB::gbIfInUcastPkts.2 = Counter32: 163701
```

```
GATEBOX-INTERFACES-MIB::gbIfInUcastPkts.3 = Counter32: 284
GATEBOX-INTERFACES-MIB::gbIfInUcastPkts.4 = Counter32: 36658
GATEBOX-INTERFACES-MIB::gbIfInUcastPkts.5 = Counter32: 84531
GATEBOX-INTERFACES-MIB::gbIfInUcastPkts.6 = Counter32: 0
GATEBOX-INTERFACES-MIB::gbIfInUcastPkts.7 = Counter32: 29751
GATEBOX-INTERFACES-MIB::gbIfInUcastPkts.8 = Counter32: 45181
GATEBOX-INTERFACES-MIB::gbIfInUcastPkts.9 = Counter32: 133164
GATEBOX-INTERFACES-MIB::gbIfOutOctets.1 = Counter32: 3855624
GATEBOX-INTERFACES-MIB::gbIfOutOctets.2 = Counter32: 353912194
GATEBOX-INTERFACES-MIB::gbIfOutOctets.3 = Counter32: 100869
GATEBOX-INTERFACES-MIB::gbIfOutOctets.4 = Counter32: 1187214
GATEBOX-INTERFACES-MIB::gbIfOutOctets.5 = Counter32: 6692468
GATEBOX-INTERFACES-MIB::gbIfOutOctets.6 = Counter32: 0
GATEBOX-INTERFACES-MIB::gbIfOutOctets.7 = Counter32: 5671264
GATEBOX-INTERFACES-MIB::gbIfOutOctets.8 = Counter32: 53986088
GATEBOX-INTERFACES-MIB::gbIfOutOctets.9 = Counter32: 297497890
GATEBOX-INTERFACES-MIB::gbIfOutUcastPkts.1 = Counter32: 31172
GATEBOX-INTERFACES-MIB::gbIfOutUcastPkts.2 = Counter32: 250618
GATEBOX-INTERFACES-MIB::gbIfOutUcastPkts.3 = Counter32: 538
GATEBOX-INTERFACES-MIB::gbIfOutUcastPkts.4 = Counter32: 7494
GATEBOX-INTERFACES-MIB::gbIfOutUcastPkts.5 = Counter32: 36270
GATEBOX-INTERFACES-MIB::gbIfOutUcastPkts.6 = Counter32: 0
GATEBOX-INTERFACES-MIB::gbIfOutUcastPkts.7 = Counter32: 29751
GATEBOX-INTERFACES-MIB::gbIfOutUcastPkts.8 = Counter32: 45181
GATEBOX-INTERFACES-MIB::gbIfOutUcastPkts.9 = Counter32: 203203
```

5.3.3.4 gbIPv4Flags

O nó gbIPv4Flags congrega sob si um conjunto de escalares tal como no exemplo gbActions. Na secção 5.3.2.3 é feita uma referência aos escalares monitorizados. Como já referido, cada escalar implementado encontra-se associado a cada um dos ficheiros presentes na directoria */proc/sys/net/ipv4/*.

5.3.4 Monitorização do sistema

Um aspecto fulcral do SNMP é a sua capacidade em analisar continuamente o sistema reportando deste modo a ocorrência de eventuais falhas. Ao longo deste capítulo forma feitas referências à operação trap presente no protocolo SNMP. Esta operação é utilizada como forma de alertar o gestor para a ocorrência de uma anomalia num dos agentes. O lançamento deste alerta só é realmente atraente se for efectuado de forma automática. No entanto, no âmbito da ferramenta Net-SNMP, esta funcionalidade não foi possível de implementar de forma eficaz. Deste modo foi desenvolvido um modo alternativo capaz de lançar alertas na ocorrência de uma falha. Este método será descrito nos próximos parágrafos.

A ferramenta Net-SNMP permite, via ficheiros de configuração snmpd.conf e *snmpd.local.conf*, que se especifiquem determinadas características do sistema para monitorização pelo agente. Entre estas características incluem-se os processos activos, o espaço

em disco, a memória utilizada e a carga do processador. Um excerto do ficheiro *snmpd.local.conf* é exposto de seguida:

```
##############################################################################
###
# SECTION: Monitor Various Aspects of the Running Host
#
#   The following check up on various aspects of a host.

# proc: Check for processes that should be running.
#     proc NAME [MAX=0] [MIN=0]

proc     apache2 0 0
procfix  apache2 /root/.snmp/sh/procfix.sh apache2
proc     flowmanager 0 0
procfix  flowmanager /root/.snmp/sh/procfix.sh flowmanager
proc     freeradius  0 0
procfix  freeradius /root/.snmp/sh/procfix.sh freeradius
proc     named 0 0
procfix  named /root/.snmp/sh/procfix.sh named
proc     pppoe-server 0 0
procfix  pppoe-server /root/.snmp/sh/procfix.sh pppoe-server
proc     proftpd 0 0
procfix  proftpd /root/.snmp/sh/procfix.sh proftpd
proc     smbd 0 0
procfix  smbd /root/.snmp/sh/procfix.sh smbd
proc     sshd 0 0
procfix  sshd /root/.snmp/sh/procfix.sh sshd

# disk: Check for disk space usage of a partition.
#    disk PATH [MIN=100000]

disk  / 10%
disk /windows 10%

# load: Check for unreasonable load average values.
#   Watch the load average levels on the machine.
#
#    load [1MAX=12.0] [5MAX=12.0] [15MAX=12.0]

load  80 70 60
swap 1000
```

Neste exemplo pretende-se a monitorização de processos, do disco, da ocupação do processador e da memória swap.

A especificação dos **processos** a monitorizar é realizada com a directiva *proc* seguida do nome do processo alvo e do nº mínimo e máximo de processos que deverão estar activos (a correr). Caso se especifique um nº mínimo 0 e um nº máximo 0, como observado nos exemplos, assume-se que pelo menos 1 processo deverá estar a correr não havendo limite máximo. No caso de nenhum processo estar a correr assume-se que ocorreu uma falha ou anomalia, situação em que deverá ser tomada uma acção para corrigir o problema. Para tal é útil a especificação na directiva *procfix* do ficheiro sh que deverá ser invocado para corrigir a falha.

A análise do **disco**, ou mais correctamente de uma partição, é declarada através da directiva *disk*. Os argumentos passados são a partição a analisar e o respectivo tamanho mínimo de espaço disponível. No caso de o espaço descer abaixo desse valor ocorre uma anomalia.

O uso de **processador** é analisado especificando-se a directoria *load*. Esta recebe três argumentos, que indicam a carga máxima durante 1, 5 e 15 minutos a partir da qual deverá ser considerada uma anomalia.

A **memória swap** também se encontra a ser monitorizada indicando-se após à directoria swap o limite mínimo de memória disponível.

Para que se conheça o estado dos processos monitorizados é necessário o acesso à tabela prTable. Esta tabela permite-nos aferir do estado de cada processo monitorizado. Nesta tabela destaca-se o objecto prErrorFlag que permite reconhecer a existência de uma anomalia caso uma das suas instâncias possua o valor 1.

```
$ snmpwalk localhost prTable
UCD-SNMP-MIB::prIndex.1 = INTEGER: 1
UCD-SNMP-MIB::prIndex.2 = INTEGER: 2
UCD-SNMP-MIB::prIndex.3 = INTEGER: 3
UCD-SNMP-MIB::prIndex.4 = INTEGER: 4
UCD-SNMP-MIB::prIndex.5 = INTEGER: 5
UCD-SNMP-MIB::prIndex.6 = INTEGER: 6
UCD-SNMP-MIB::prIndex.7 = INTEGER: 7
UCD-SNMP-MIB::prIndex.8 = INTEGER: 8
UCD-SNMP-MIB::prNames.1 = STRING: apache2
UCD-SNMP-MIB::prNames.2 = STRING: flowmanager
UCD-SNMP-MIB::prNames.3 = STRING: freeradius
UCD-SNMP-MIB::prNames.4 = STRING: named
UCD-SNMP-MIB::prNames.5 = STRING: pppoe-server
UCD-SNMP-MIB::prNames.6 = STRING: proftpd
UCD-SNMP-MIB::prNames.7 = STRING: smbd
UCD-SNMP-MIB::prNames.8 = STRING: sshd
UCD-SNMP-MIB::prMin.1 = INTEGER: 0
UCD-SNMP-MIB::prMin.2 = INTEGER: 0
UCD-SNMP-MIB::prMin.3 = INTEGER: 0
UCD-SNMP-MIB::prMin.4 = INTEGER: 0
UCD-SNMP-MIB::prMin.5 = INTEGER: 0
UCD-SNMP-MIB::prMin.6 = INTEGER: 0
UCD-SNMP-MIB::prMin.7 = INTEGER: 0
UCD-SNMP-MIB::prMin.8 = INTEGER: 0
UCD-SNMP-MIB::prMax.1 = INTEGER: 0
UCD-SNMP-MIB::prMax.2 = INTEGER: 0
UCD-SNMP-MIB::prMax.3 = INTEGER: 0
UCD-SNMP-MIB::prMax.4 = INTEGER: 0
UCD-SNMP-MIB::prMax.5 = INTEGER: 0
UCD-SNMP-MIB::prMax.6 = INTEGER: 0
UCD-SNMP-MIB::prMax.7 = INTEGER: 0
UCD-SNMP-MIB::prMax.8 = INTEGER: 0
UCD-SNMP-MIB::prCount.1 = INTEGER: 6
UCD-SNMP-MIB::prCount.2 = INTEGER: 2
UCD-SNMP-MIB::prCount.3 = INTEGER: 1
UCD-SNMP-MIB::prCount.4 = INTEGER: 1
UCD-SNMP-MIB::prCount.5 = INTEGER: 1
UCD-SNMP-MIB::prCount.6 = INTEGER: 1
```

```
UCD-SNMP-MIB::prCount.7 = INTEGER: 2
UCD-SNMP-MIB::prCount.8 = INTEGER: 2
UCD-SNMP-MIB::prErrorFlag.1 = INTEGER: 0
UCD-SNMP-MIB::prErrorFlag.2 = INTEGER: 0
UCD-SNMP-MIB::prErrorFlag.3 = INTEGER: 0
UCD-SNMP-MIB::prErrorFlag.4 = INTEGER: 0
UCD-SNMP-MIB::prErrorFlag.5 = INTEGER: 0
UCD-SNMP-MIB::prErrorFlag.6 = INTEGER: 0
UCD-SNMP-MIB::prErrorFlag.7 = INTEGER: 0
UCD-SNMP-MIB::prErrorFlag.8 = INTEGER: 0
UCD-SNMP-MIB::prErrMessage.1 = STRING:
UCD-SNMP-MIB::prErrMessage.2 = STRING:
UCD-SNMP-MIB::prErrMessage.3 = STRING:
UCD-SNMP-MIB::prErrMessage.4 = STRING:
UCD-SNMP-MIB::prErrMessage.5 = STRING:
UCD-SNMP-MIB::prErrMessage.6 = STRING:
UCD-SNMP-MIB::prErrMessage.7 = STRING:
UCD-SNMP-MIB::prErrMessage.8 = STRING:
UCD-SNMP-MIB::prErrFix.1 = INTEGER: 0
UCD-SNMP-MIB::prErrFix.2 = INTEGER: 0
UCD-SNMP-MIB::prErrFix.3 = INTEGER: 0
UCD-SNMP-MIB::prErrFix.4 = INTEGER: 0
UCD-SNMP-MIB::prErrFix.5 = INTEGER: 0
UCD-SNMP-MIB::prErrFix.6 = INTEGER: 0
UCD-SNMP-MIB::prErrFix.7 = INTEGER: 0
UCD-SNMP-MIB::prErrFix.8 = INTEGER: 0
UCD-SNMP-MIB::prErrFixCmd.1 = STRING: /root/.snmp/sh/procfix.sh apache2
UCD-SNMP-MIB::prErrFixCmd.2 = STRING: /root/.snmp/sh/procfix.sh flowman-
ager
UCD-SNMP-MIB::prErrFixCmd.3 = STRING: /root/.snmp/sh/procfix.sh freera-
dius
UCD-SNMP-MIB::prErrFixCmd.4 = STRING: /root/.snmp/sh/procfix.sh named
UCD-SNMP-MIB::prErrFixCmd.5 = STRING: /root/.snmp/sh/procfix.sh pppoe-
server
UCD-SNMP-MIB::prErrFixCmd.6 = STRING: /root/.snmp/sh/procfix.sh proftpd
UCD-SNMP-MIB::prErrFixCmd.7 = STRING: /root/.snmp/sh/procfix.sh smbd
UCD-SNMP-MIB::prErrFixCmd.8 = STRING: /root/.snmp/sh/procfix.sh sshd
```

A análise periódica desta estrutura permite detectar eventuais falhas. A variável prErrorFlag é a responsável por indicar a ocorrência de erros. Esta variável detém normalmente o valor 0 passando a 1 quando o nº de processos activos não se encontra dentro dos limites definidos no ficheiro snmpd.local.conf (ou snmpd.conf).

A inquirição periódica desta estrutura faz-se com recurso ao processo cron. Este processo Unix permite que as tarefas definidas nos seus ficheiros de configuração sejam executadas periodicamente. Adicionou-se deste modo, nos seus ficheiros de configuração, uma instrução para que cada seja executado em cada minuto um scipt perl capaz criado para interrogar a tabela acima representada.

O script perl criado tem como função detectar quais os processos que não estão a operar correctamente. Detectando uma falha é executado o ficheiro de correcção correspondente,

definido com a directiva procfix. O resultado deste processo de monitorização é uma acção autónoma e rápida na resolução de problemas internos, levando a que as máquinas sejam minimamente afectadas pela quebra de um processo.

Nos restantes casos monitorizados (disco, memória e processador), toma-se a acção de notificação do gestor de redes através do envio de um email. Para tal recorre-se ao comando sendmail.

5.4 Interface com os agentes

A interface entre o gestor de rede e o sistema de gestão é efectuada na máquina a correr em modo gestor (cliente). A partir desta máquina, dita central, será possível questionar os vários agentes relativamente aos objectos que albergam. Esta interacção será efectuada através dos comandos SNMP já referidos. Pensando-se numa interface centralizada, capaz de reunir operações de gestão e administração, desenvolveu-se uma **interface web** a qual permite a execução de diversas acções implementadas nos agentes assim como a observação do valor dos objectos mais relevantes presentes em cada router. Esta interface web encontra-se extensivamente descrita no próximo capítulo.

Capítulo 6

6. Administração remota

6.1 Cenário para administração remota

Surge agora a necessidade de poder monitorizar, controlar e configurar todas as GateBoxes a partir de um sistema centralizado remoto, uma vez que os serviços após estarem instalados é necessário garantir o seu funcionamento contínuo.

O que se pretende é que exista um sistema centralizado remoto onde se encontra toda a informação dos clientes (como por exemplo nome, morada, cidade, classe de serviço...), das Gateboxes (como por exemplo id, morada, IP...) e outras informações necessárias.

Para um determinado número de GateBoxes, por exemplo numa região, existe um computador remoto responsável pela administração e gestão dessas máquinas.

Para implementar um sistema de administração e gestão remota desenvolveu-se uma interface web em php na máquina de administração, onde se integrou nela todas as funcionalidades e serviços necessários.

Este cenário encontra-se representado na Figura 14.

Os seguintes capítulos falam exclusivamente do desenvolvimento e das funcionalidades dessa interface.

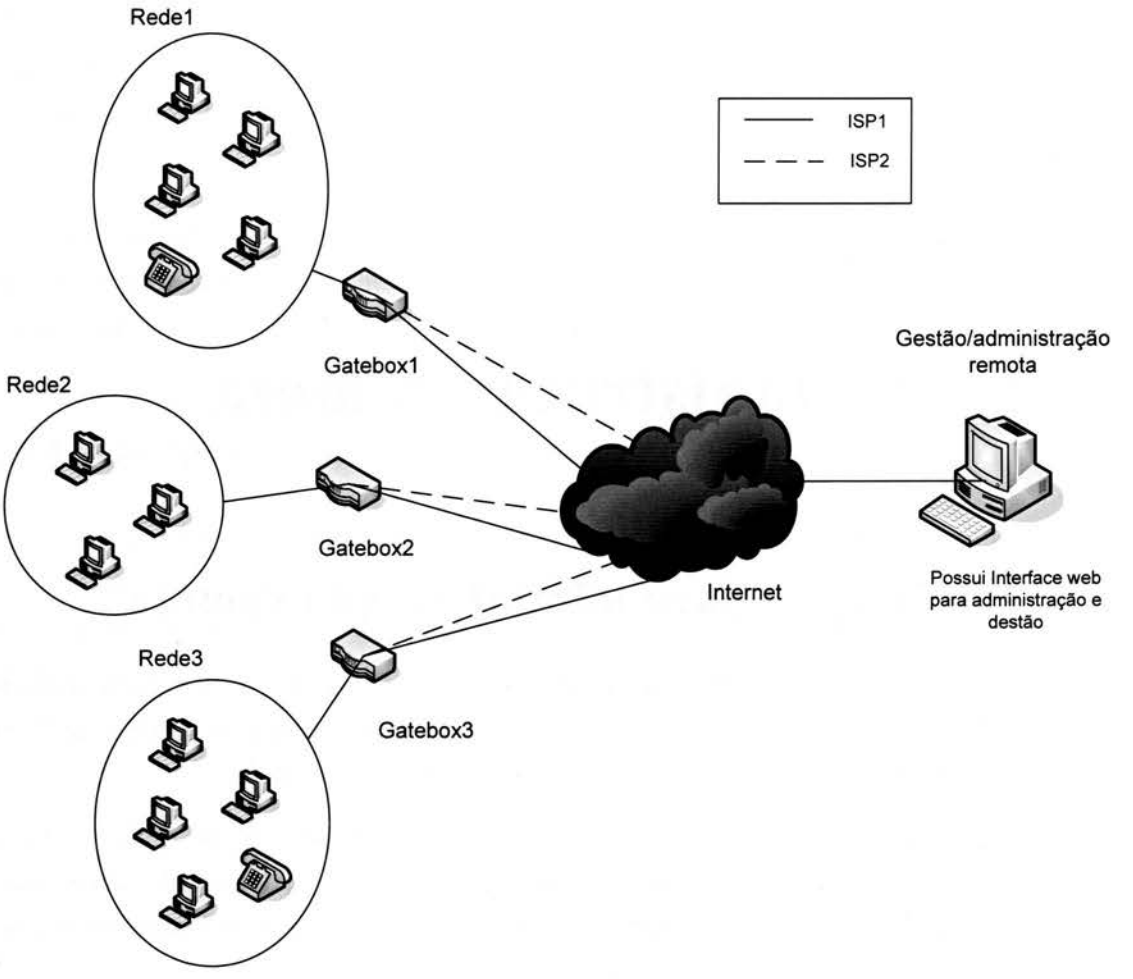


Figura 14 – Cenário para administração remota real

6.2 Aspectos iniciais da Interface Web

A interface web desenvolvida para gestão e administração remota em php foi desenvolvida num PC normal que iria funcionar como administrador remoto. Esta aplicação só é acedível apenas a partir da máquina onde se encontra instalada.

Antes de se desenvolver a interface, criou-se uma base de dados em mysql onde iria estar armazenada toda a informação. Essa base de dados é composta por três tabelas (nota: a informação contida nas tabelas das figuras é apenas para dar exemplos do tipo de informação que a base de dados contém):

- A tabela **users** onde está contida toda a informação dos clientes, como por exemplo o username e password, a GateBox a que esse utilizador está associado (coluna id_gatebox), o seu IP e a sua classe (ver Figura 15);

username	pass	pnome	unome	nacionalidade	cidade	morada	admin	id_gatebox	ip_user	classe
admin	admin	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	1	*NULL*	*NULL*	*NULL*
user1	n2gqz8ba97s1	Bruno	Oliveira	Portuguesa	Porto	Rua Dinis	*NULL*	1	192.167.5.1	class5
user2	7w64m1skrgdt	Vitor	Silva	Portuguesa	Porto	Rua Dinis	*NULL*	1	192.167.5.2	class5
user3	yx4nfar9p0g3	Luis	Silva	Portuguesa	Porto	Rua Dinis	*NULL*	1	192.167.5.3	class5

Figura 15 – Tabela Users

- A tabela **gatebox** onde está contida toda a informação de cada gatebox (ver Figura 16.);

id	cidade	morada	ip	mac
1	Porto	Inesc	192.168.1.1	00:00:1C:81:42:90
2	Maia	Rua de Cima	192.168.1.111	00:00:1C:81:42:91

Figura 16 – Tabela gatebox

*A tabela **serviços** que contém as várias classes de serviços disponíveis (ver Figura 17);*

nome	descricao
class1	Internet connection with lowest priority
class2	Internet connection with low priority
class3	Internet connection with medium priority
class4	Internet connection with high priority
class5	Internet connection with highest priority

Figura 17 – Tabela serviços

Estando a base de dados criada desenvolveu-se a interface web para permitir inserir, remover, editar e visualizar a informação contida nesta base de dados e através desta informação e dos serviços já instalados (como por exemplo o SNMP) efectuar a gestão e monitorização do sistema.

Ao introduzir o endereço ***https://127.0.0.1/gatebox/*** na máquina de administração surge a página de login representada na Figura 18. Para aceder ao conteúdo da interface web é necessário efectuar o login nesta página, só os utilizadores com atributo admin activo da tabela users da base de dados, é que conseguem aceder.

Figura 18 – Página de login

Se o login for efectuado com sucesso, entra-se na página principal (apresentada na Figura 19), que contém informação relativa às classes que existem e o número de utilizadores e GateBoxes que existem no sistema. No lado esquerdo encontra-se o menu que permite aceder às três diferentes secções, que são: a secção de edição dos serviços (Services), a secção das GateBoxes (Gateboxes) e a página dos utilizadores (Users).

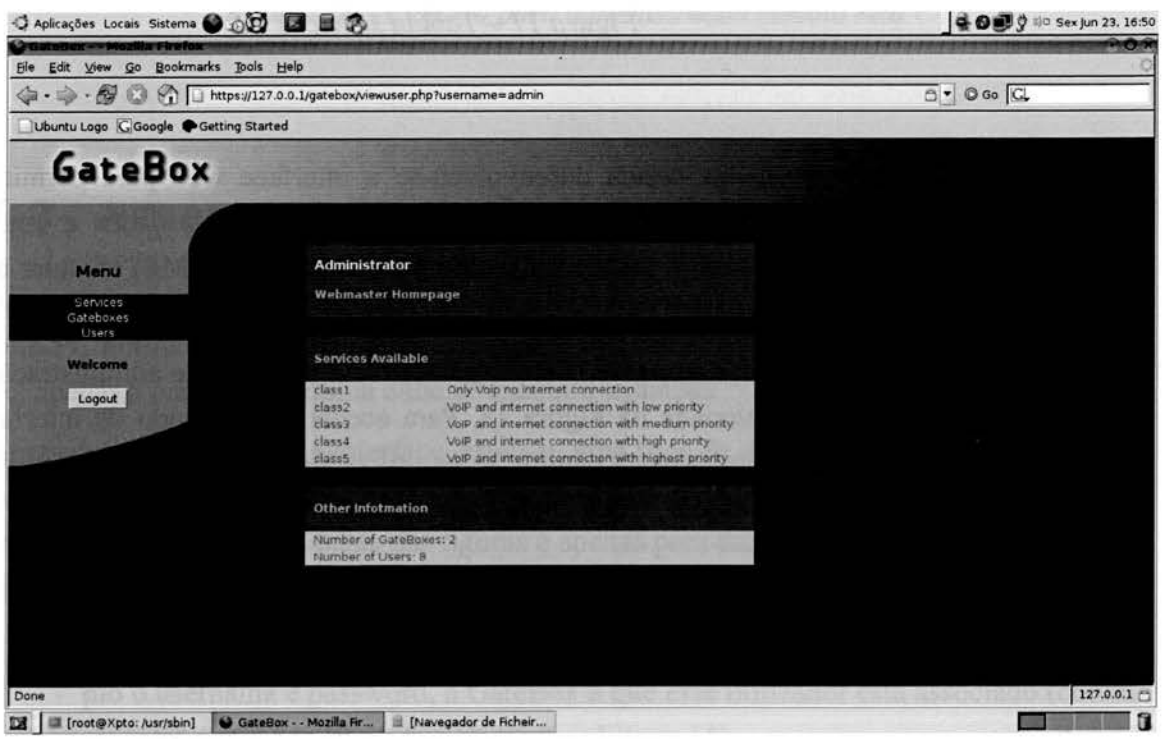


Figura 19 – Página inicial

Os próximos capítulos dizem respeito às diversas funcionalidades e desenvolvimentos efectuados em cada uma das três secções.

6.3 Secção dos serviços na Interface Web

Na secção de serviços encontram-se as classes de serviço disponíveis, onde é possível acrescentar uma nova, editar alguma já existente ou remover uma classe de serviço. Na Figura 20 está apresentada esta secção.

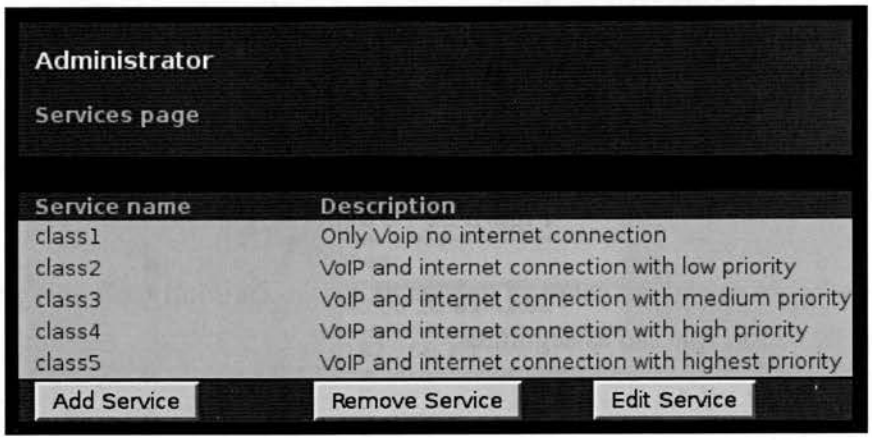


Figura 20 – Secção dos serviços

Nesta secção existem três opções:

- "***Add Service***" que permite adicionar uma nova classe de serviço na base de dados;

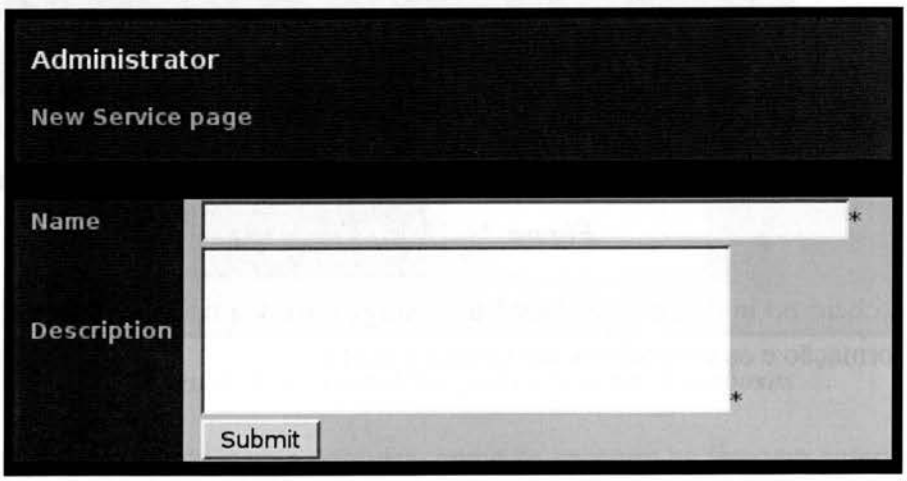


Figura 21 – Adicionar uma nova classe de serviço

- "***Remove Service***" permite remover da base de dados uma das classes já existentes;
- "**Edit Service**" permite editar a descrição de uma classe.

6.4 Secção das Gateboxes na Interface Web

Na secção das Gateboxes encontram-se as várias opções para a sua gestão e administração. Na Figura 22 encontram-se as várias opções, onde as linhas em branco são links para as páginas correspondentes.

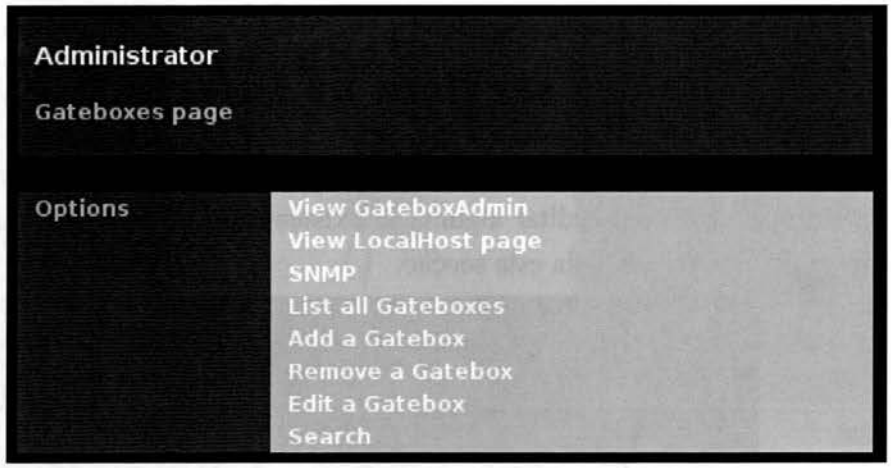


Figura 22 – Página das Gateboxes

Primeiro irá ser falado sobre as opções da gestão da informação das Gateboxes contida na base de dados, que são as seguintes:

- "***List all Gateboxes***" lista todas as gateboxes existentes naquele sistema (Figura 23).

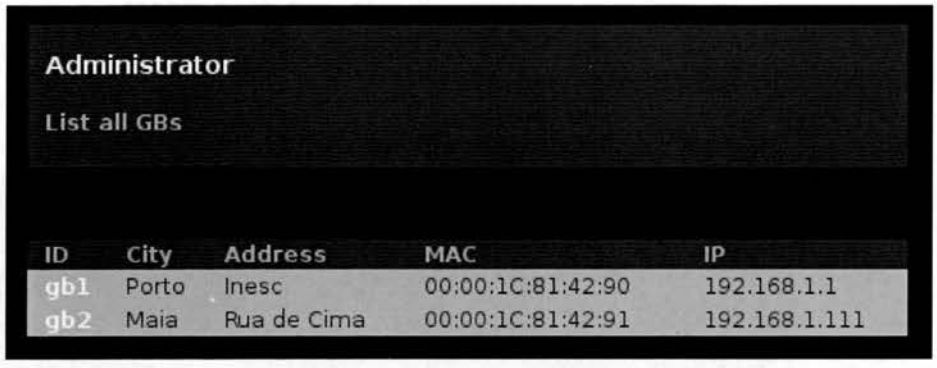


Figura 23 – Listagem de todas as GBs

Ao clicar no id de uma das GateBoxes surge a página dessa GateBox, que contém toda a informação e os utilizadores associados a ela ().

- "***Add a Gatebox***" permite adicionar uma nova Gatebox no sistema:

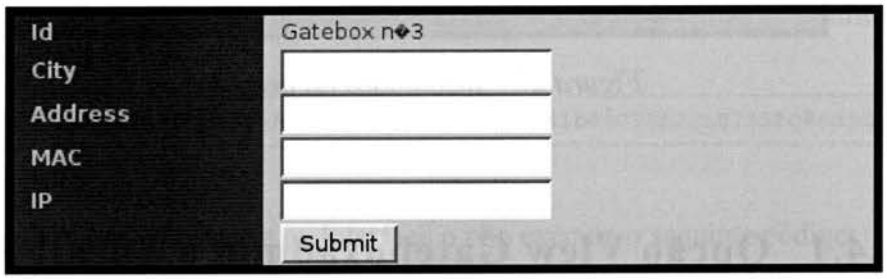


Figura 24 – Formulário para adicionar uma nova Gatebox

- "**Remove a Gatebox**" permite remover uma Gatebox e todas as suas associações da base de dados;
- "***Edit a Gatebox***" permite editar a informação de uma determinada Gatebox, a informação antiga é mostrada nos campos respectivos para ajudar o administrador a efectuar as alterações (Figura 25);

Figura 25 – Formulário para editar uma Gatebox

- "***Search***" permite pesquisar uma determinada gatebox na base de dados (Figura 26).

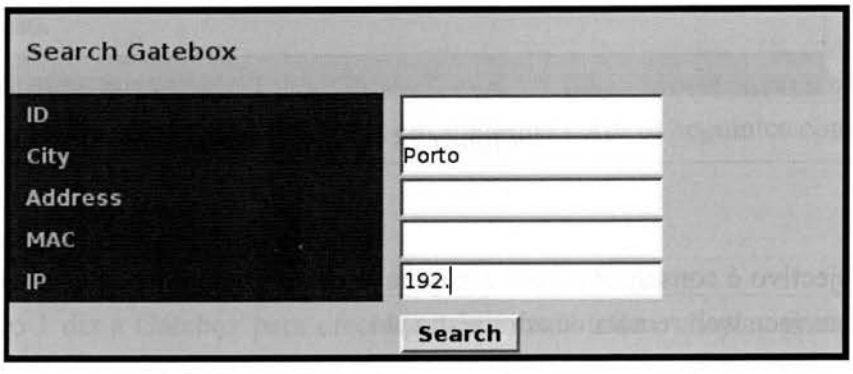


Figura 26 – Formulário para pesquisar Gateboxes

Como exemplo ao efectuar uma pesquisa, como se encontra na imagem anterior, irão aparecer todas as Gateboxes que estejam instaladas na região do Porto e com 192.* como IP. Os resultados da pesquisa encontram-se na Figura 27.

ID	City	Address	MAC	IP
gb1	Porto	Inesc	00:00:1C:81:42:90	192.168.1.1

Figura 27 – Resultados da pesquisa da Figura 26

6.4.1 Opção View Gateboxadmin e Localhost

Em cada Gatebox existe instalada uma página local para configuração e gestão dos serviços e recursos da Gatebox, chamada Webadmin. Esta é uma aplicação disponível nos pacotes do Ubuntu, bastando apenas instalar e configurar a aplicação para os serviços necessários (ver Figura 28). Para aceder a esta página só é possivel através da máquina em que se encontra instalada, introduzindo o endereço ***https://'endereçolocal':10000/*** .

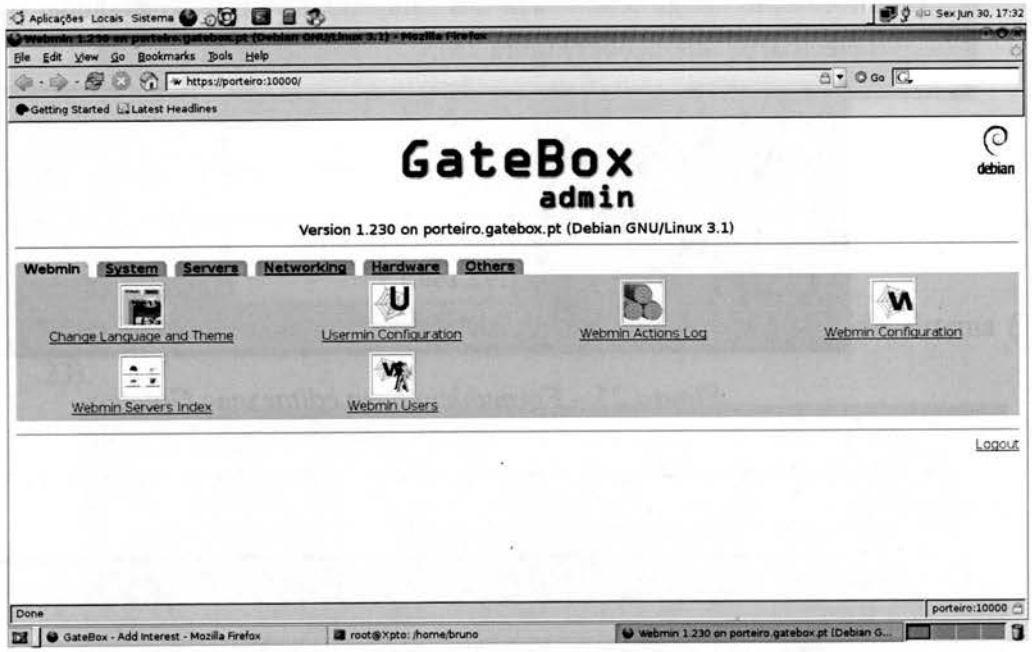


Figura 28 – Página Webadmin

O objectivo é conseguir aceder à página Webadmin de uma determinada Gatebox, através da Interface web remota de administração.

Para isso é necessário utilizar as funcionalidades do Xauth e ssh. Com Xauth é impossível abrir uma aplicação gráfica num computador remoto por ssh (ver o capítulo 3.3) e ver o display gráfico dessa aplicação na máquina local. Assim consegue-se aceder à Gatebox por ssh, abrir a aplicação desejada e com o xauth a aplicação é aberta na máquina local.

Na máquina de administração existe um script em shell chamado ***getx.sh*** que é necessário pô-lo a correr automaticamente quando se liga o PC. Este script corre o comando "***xauth list***", que contém a chave do display da máquina que é necessário colocar na máquina

remota para conseguir fazer o export e reencaminha o output para um ficheiro de texto chamado xauth. Este comando retorna a seguinte string onde a chave encontra-se à frente do MAGIC-COOKIE-1:

```
localhost.localdomain:0  MIT-MAGIC-COOKIE-1 cfab61b4c07562ca8735fc86d9d403dd
```

Quando se selecciona esta opção na Interface o php executa o seguinte código:

```
....
(1)  $filename = '/var/www/xauth';
(2)  $fp = fopen( $filename, "r" );
(3)  $content = fread( $fp, filesize( $filename ) );
(4)  $rest = substr($content, 30);
(5)  $command = 'ssh root@192.168.1.1 xauth add 192.167.5.1:0 '.$rest;
(6)  $output = shell_exec($command);
(7)  $command = 'ssh root@192.168.1.1 /home/master/sh/gbadmin.sh 192.167.5.1';
(8)  $output = shell_exec($command);
....
```

Nas instruções 1, 2 e 4 é aberto e copiado o contúdo para uma variável do ficheiro que contém a password do display da máquina de administração. Na instrução 4 é retirada apenas a password da string.

Na instrução 5 é construído o comando ssh, que cria um túnel encriptado entre a máquina de administração e a Gatebox seleccionada (neste exemplo a Gatebox com IP 192.168.1.1) e adiciona no xauth da Gatebox o IP da máquina do administrador e a password do seu display. Na linha 6 esse comando é executado com a função **shell_exec ()** , que permite executar comandos da shell através do php. Este comando ssh executado é instântaneo, ou seja, cria um túnel encriptado entre o administrador e a Gatebox, executa o comando e fecha a ligação.

Na linha 7 e 8 é executado na Gatebox o comando *'/home/master/sh/gbadmin.sh 192.167.5.1'*, sendo *gbadmin.sh* um script em shell que corre os seguintes comandos:

```
(1)  export DISPLAY=$1:0
(2)  firefox https://porteiro:10000/
```

O commando 1 diz à Gatebox para efectuar o export de todo o display gráfico de aplicações para a máquina com IP 192.167.5.1 (IP passado como argumento), sendo este o endereço da máquina da administração. No comando 2 executa o firefox com o link da página do Webadmin, abrindo assim a página na máquina de administração (fig.28).

Para visualizar a página local do apache de uma gatebox, o processo é o mesmo do que o descrito em cima, só que em vez de de correr o script gbadmin.sh corre o script viewlocal.sh, que executa no firefox 'http://127.0.0.1/'. Acedendo ao localhost de uma determinada Gatebox pode-se aceder a várias interfaces web, como por exemplo, o phpadmin que

permite aceder e gerir as base de dados em mysql e o MRTG, que é uma interface web que permite saber a estatística de tráfego por interface utlizando o SNMP.

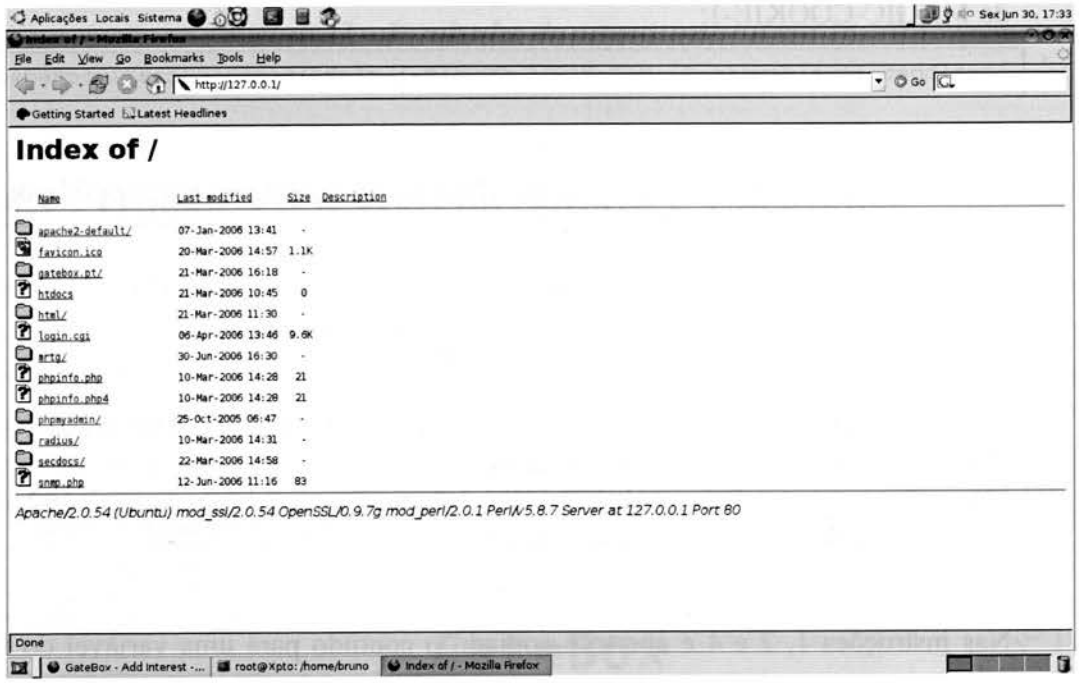


Figura 29 – Página local no servidor apache da GateBox

Houve alguns problemas quando se tentou executar no começo comandos ssh através da função php shell_exec(), devido às permissões. Uma vez que a Interface web se encontra no servidor apache qualquer comando que ele corra a partir do php é executado pelo utilizador do apache que é o www-data. Para funcionar basta copiar o ficheiro com a chave pública do ssh, que é o id_rsa e o id_rsa.pub, que se encontram na pasta ***/~/.ssh***, para a pasta ***/var/www/.ssh***.

6.4.2 Opção SNMP

Um dos obectivos da Interface Web era integrar nela as funcionalidades do SNMP referidas no capítulo 5. Por isso foi desenvolvido um módulo para ler variáveis e executar comandos SNMP. A secção do SNMP apresenta as opções apresentadas na Figura 30.

Figura 30 – Opções SNMP

A primeira opção permite ler diferentes variáveis de uma determinada Gatebox, como por exemplo, a memória em RAM, o espaço em disco, os processos... entre outros.

Se por exemplo o administrador pretender saber a memória total livre na Gatebox1, na opção Read seleccionar 'Memory' (Figura 30) e clicar em ok. Depois surge outra página onde se pode seleccionar a Gatebox que se pretende saber qual a memória total livre, o comando a executar (get, getnext ou walk), a instância e a variável, tal como se encontra na Figura 31.

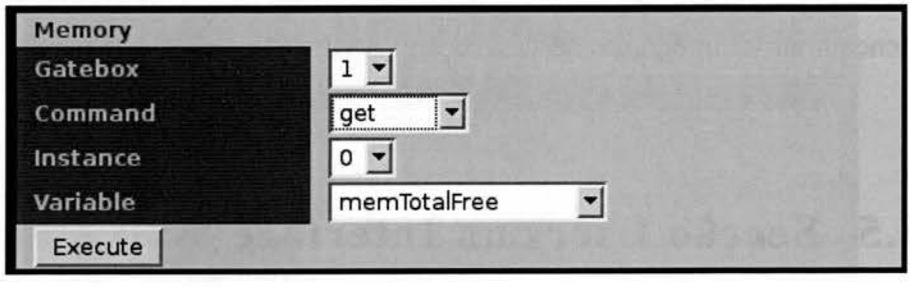


Figura 31 – Escolha de parâmetros

Ao clicar no *execute* da Figura 312 surge a resposta da Gatebox ao valor pedido da memória total livre por SNMP, como se encontra na Figura 32.

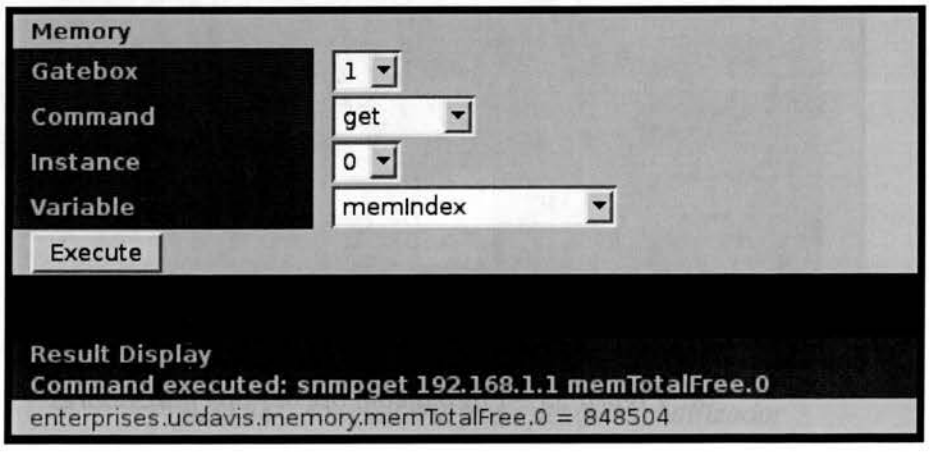


Figura 32 – Resultado do comando get

Os comandos SNMP são executados com funções SNMP integrados no php, bastando instalar o pacote com a biblioteca SNMP para o php. Para executar por exemplo o comando 'snmpget 192.168.1.1 memTotalFree.0', no php utiliza-se a seguinte função:

```
snmp3_get($host,$user,$level,$auth_protocol,$auth_key,$priv_protocol,$priv_key
,$oid1)
```

As outras opções presentes na Figura 320 permitem através do protocolo SNMP fazer o restart de uma determinada Gatebox, restart aos serviços (existe um script em shell em cada Gateboxpara efectuar automaticamente o restart de todos os serviços instalados) e reiniciar alguns dos serviços essenciais. O desenvolvimeto do código SNMP para estes comandos encontra-se no capítulo 5.3.4, o php executa apenas a função snmp3_set() com os dados que o utlizador escolheu.

```
snmp3_set(string host, string sec_name, string sec_level, string auth_protocol,
string auth_passphrase, string priv_protocol, string priv_passphrase, string
object_id, string type, mixed value)
```

Todas as funcionalidades desenvolvidas através do protocolo SNMP descritas no capítulo 5 encontram-se integradas nesta seccção SNMP da interface web desenvolvida.

6.5 Secção Users na Interface Web

Na secção dos utilizadores encontram-se as várias opções para a sua gestão e administração. Na Figura 333 encontram-se as várias opções, onde as linhas em branco são links para as páginas correspondentes.

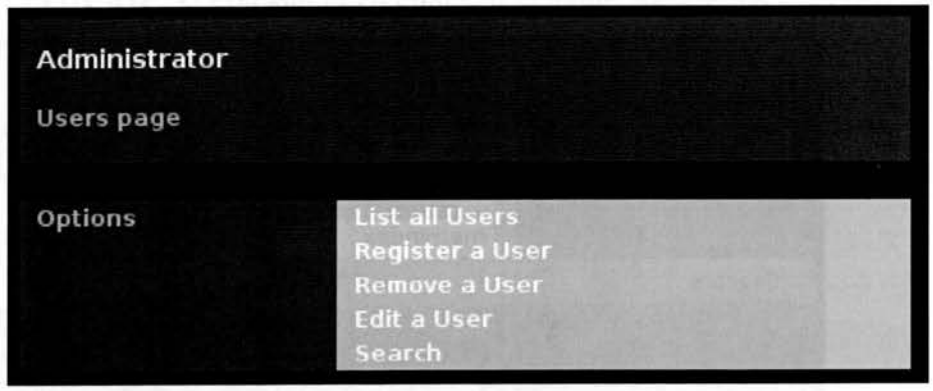


Figura 33 – Página com opções dos utilizadores

Primeiro irá ser falado sobre as opções da gestão da informação dos utilizadores contida na base de dados, que são as seguintes:

- "***List all Users***" lista todos os utilizadores existentes naquele sistema (Figura 344).

Username	Name	City	Gatebox nº	User IP
user1	Bruno Oliveira	Porto	1	192.167.5.1
user2	Vitor Silva	Porto	1	192.167.5.2
user3	Luis Silva	Porto	1	192.167.5.3

Figura 34 – Lista de todos os utilizadores

Ao clicar no username de um dos Utilizadores surge a página desse utilizador, que contém toda a informação e opções de administração para aquele utilizador (ver Figura 355).

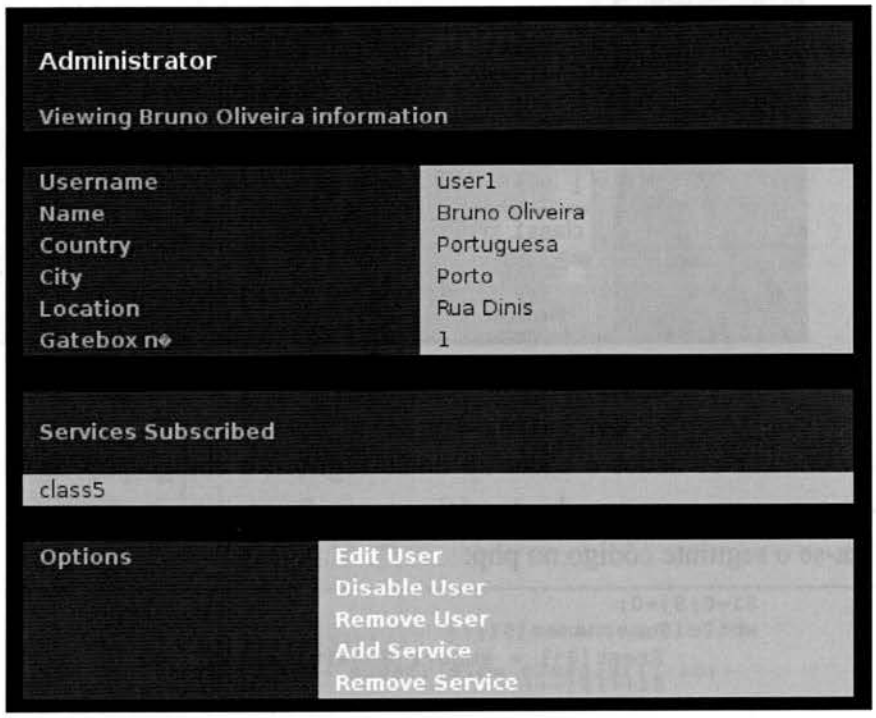


Figura 35 – Informação detalhada e opções de um utilizador

- "***Search***" permite pesquisar um determinado utilizador na base de dados (Figura 366).

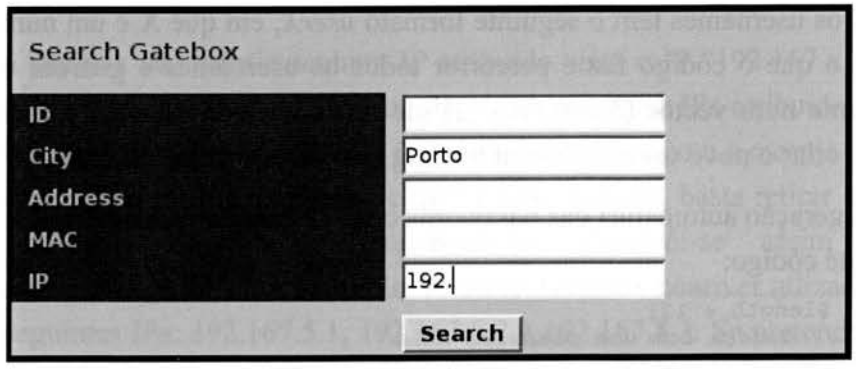


Figura 36 – Formulário para pesquisar utilizador

No próximo capítulo fala-se sobre o registo e alteração de utilizadores no sistema.

6.5.1 Registo de utilizadores no sistema

Para registar um novo utilizador é necessário introduzir todos os dados dele na base de dados do sistema na tabela 'users' que se encontra na máquina de administração e introduzir a informação de autenticação (username, password,IP) na base de dados RADIUS da Gatebox correspondente. Logo que se regista o utilizador é informação tem de ser armazenada na base de dados de administração e na base de dados da GateBox respectiva.

Para o registo de novos utilizadores, é utilizada a interface apresentada na Figura 377.

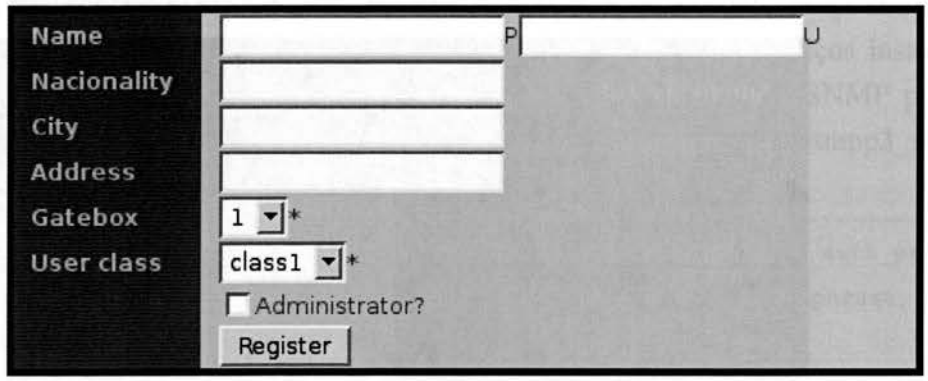


Figura 37 – Formulário para registar utilizadores

Após introduzir os dados e clicar no botão register, o código php da interface web gera um username e uma password automáticamente. Para gerar os usernames automáticamente utiliza-se o seguinte código no php:

```
                $i=0;$j=0;
(1)             while($usernames[$i]){
                        $rest[$j] = substr($usernames[$i], 4);
                        $i++;$j++;
                }

(2)             $max_num=max($rest);
                $max_num++;
(3)             $username= "user".$max_num;
```

Todos os usernames têm o seguinte formato *userX*, em que X é um número, então basicamente o que o código faz é percorrer todos os usernames e guardar o valor X de cada username num vector ($rest). No (2) ele retira o valor máximo e incrementa, bastando depois criar o novo username com a string *user* mais o valor máximo.

Para a geração automática das passwords com 12 caracteres aleatórios, a interface utiliza o seguinte código:

```
        $length = 12;
        // começa com uma password nula
        $password = "";

        // possiveis caracteres
        $possible = "0123456789abcdfghjkmnpqrstvwxyz";

        // inicia contador
```

```
        $i = 0;

        // adiciona caracteres aleatórios a $password até $length ser atingido
        while ($i < $length) {

                // selecciona um caracter dos possíveis aleatóriamente
                $char = substr($possible, mt_rand(0, strlen($possible)-1), 1);

                // se o caracter existir não adiciona na password
                if (!strstr($password, $char)) {
                        $password .= $char;
                        $i++;
                }
        }
```

Para a geração automática do IP do utilizador à que ter em conta a classe de serviço escolhida e a Gatebox onde se está a registar o utlizador, uma vez que é necessário ter em conta os IPs que já existem atribuidos naquela Gatebox para não haver IPs iguais. Para isso a interface web utliza o seguinte código:

```
(1)     if(!$ips_gb){
                $ip_user = "192.167.".$x.".1";
        }
(2)     else{
                $users_ips=Users::getbyGbAndClass($gatebox,$class);

                $i=0;$j=0;
                while($users_ips[$i]){
                        if($users_ips[$i]->ip_user){
                                $ips_class[$j]=$users_ips[$i]->ip_user;
                                $i++;$j++;
                        }
                        else
                                $i++;
                }

                $i=0;$j=0;
                while($ips_class[$i]){
                        $rest[$j] = substr($ips_class[$i], 10);
                        $i++;$j++;
                }
                $max_num=max($rest);
                $max_num++;

                $ip_user = "192.167.".$x.".".$max_num;
        }
```

Se naquela gatebox não existir nenhum IP atribuído, gera o IP "192.167.x.1", onde x é a classe de serviço escolhida (1, 2, 3, 4 ou 5). Se existirem já IPs atribuídos o código vai buscar à base de dados apenas os IPs da Gatebox escolhida e com a mesma classe introduzida. Após ter a lista de IPs da mesma classe nessa Gatebox, basta retirar o quarto digito máximo da lista de IPs e incrementá-lo. Constrói-se assim o ip com "192.167.classe.max_num". Por exemplo, se numa Gatebox contiver utlizadores de classe 5 com os seguintes IPs: 192.167.5.1, 192.167.5.2 e 192.167.5.3. Se pretender-mos adicionar um novo utilzador de classe 5 nessa Gatebox o Interface php gera o IP 192.167.5.4.

Após a geração do username, password e endereço IP para o novo utilizador, é necessário colocar esta informação na base de dados RADIUS onde se pretende registar o utlizador. Isto é feito utlizando o SSH (secure shell), onde o php executa o seguinte comando SSH que cria um túnel encriptado com a Gatebox escolhida e corre um script em perl, onde lhe passa como argumentos o username, password e endereço IP gerados.

```
$command= "ssh -t root@".$ip_gatebox." /home/master/Desktop/update_db.pl
".$username." ".$password." ".$ip_user;
```

O script em perl chama-se update_db.pl e encontra-se em todas as Gatebox, recebe como parâmetros o username, password e IP do novo utlizador por ssh e adiciona na base de dados RADIUS nas tabelas respectivas (ver capítulo 4.1 do RADIUS).

```
#Fscript perl de registo de utilizadores na base de dados RADIUS
#!/usr/bin/perl
use DBI;

my $driver = 'mysql';
my $database = 'radius';          # nome da base de dados
my $username = master;            # username da base de dados
my $password = deec;              # password da base de dados

my $user=$ARGV[0];
my $pass=$ARGV[1];
my $ip=$ARGV[2];

# we set up a connection string specific to this database
my $dsn = "DBI:$driver:database=$database";

# efectua a ligação à base de dados
my $dbh = DBI->connect($dsn, $username, $password);

#-----------------------------------------------------------
open(log_update,">/home/master/log_update") || die "Could not open file";

if(!$dbh){
      print log_update "erro";
}
else{
      print log_update "";
}
close(log_update);
#-----------------------------------------------------------
# prepara a query SQL para a tabela radcheck do RADIUS
my $sql_statement = "insert into radcheck(id,UserName,Attribute,op,Value) val-
ues('','".$user."','Password','==','".$pass."');" ;
my $sth = $dbh->prepare($sql_statement)
      || die "Could not prepare: " . $dbh->errstr();

# executa a query
$sth->execute() || die "Could not execute: " . $dbh->errstr();

# termina a query sql
$sth->finish();

#-----------------------------------------------------------
#-----------------------------------------------------------
# prepara a query SQL para a tabela radreply do RADIUS
my $sql_statement = "insert into radreply(id,UserName,Attribute,op,Value) val-
ues('','".$user."','Framed-IP-Address',':=','".$ip."');" ;
```

```
my $sth = $dbh->prepare($sql_statement)
      || die "Could not prepare: " . $dbh->errstr();

# executa a query
$sth->execute() || die "Could not execute: " . $dbh->errstr();

# termina a query sql
$sth->finish();

#-------------------------------------------------------
#-------------------------------------------------------
# prepara a query SQL para a tabela usergroup do RADIUS
my $sql_statement = "insert into usergroup(id,UserName,GroupName) val-
ues('','".$user."','gatebox');" ;
my $sth = $dbh->prepare($sql_statement)
      || die "Could not prepare: " . $dbh->errstr();

# executa a query
$sth->execute() || die "Could not execute: " . $dbh->errstr();

# termina a query sql
$sth->finish();

#-------------------------------------------------------

# termina ligação com base de dados
$dbh->disconnect();
```

Se o registo na base de dados RADIUS da GateBox foi efectuado com sucesso, a Interface web de administração regista também o utilizador com toda a informação na base de dados central da administração

Capítulo 7

7. Conclusões e Perspectivas de Desenvolvimento

A realização deste projecto, em colaboração com o projecto FlowManager, produziu a máquina GateBox. Esta máquina congrega em si vários serviços de rede, permitindo a sua integração num cenários real de distribuição de Internet, como exemplificado no capítulo 2.

Ao longo deste documento percorreram-se as diversas fases que levaram à criação da máquina e da rede GateBox. O arranque do projecto foi caracterizado pela investigação das tecnologias existentes capazes de responder aos requisitos do sistema a produzir. Diversas propostas foram equacionados tendo vir a ser aprovadas ou rejeitadas em função da sua aplicabilidade no sistema final GateBox. Assim, após uma antevisão teórica estas tecnologias foram colocadas em prática obtendo-se resultados (quase) imediatos sobre a sua taxa de sucesso no sistema GateBox. Aliás a vertente prática é evidente ao longo da exposição deste trabalho, facto que permite obter resultados claros sobre as opções tomadas no decorrer do projecto.

Concluída a implementação das máquinas GateBox e da respectiva congénere administrativa (GateBox central para gestão e administração da rede GateBox) é possível admitir que os objectivos propostos foram atingidos na sua globalidade. Não obstante ter-se atingido as metas propostas, a implementação da máquina e rede GateBox não se encontra fechada podendo ser adicionadas novas funcionalidades e expandidas as já existentes. Uma delas será um Gateway para suportar VoIP e permitir chamadas gratuitas entre utlizadores na mesma GateBox. Introduzir na GateBox suporte para distribuição de conteúdos multimédia, usando por exemplo o Samba e o FTP para integrar VoD (Vídeo on Demand).

A nível pessoal destaque-se a evidente mais valia que este projecto produziu ao nível do conhecimento sobre redes de computadores para os autores deste trabalho. Destaque-se ainda o enriquecimento profissional pela integração num ambiente jovem e extremamente dinâmico, que é predicado do INESC Porto.

Referências Bibliográficas

[RFC959, 1985] *RFC959 –File Transfer Protocol (FTP)*, 1985 [Em Linha]. Disponível em http://www.ietf.org/rfc/rfc959.txt . [Consultado em 10/03/2006]

[RFC1067, 1988] *RFC1067 –A Simple Network Management Protocol*, 1988 [Em Linha]. Disponível em http://www.ietf.org/rfc/rfc1067.txt . [Consultado em 29/06/2006]

[RFC1155, 1990] *RFC1155 –Structure and Identification of Management Information for TCP/IP-based Internets*, 1990 [Em Linha]. Disponível em http://www.ietf.org/rfc/rfc1155.txt . [Consultado em 29/06/2006]

[die.net, 2006] *RFC1155 –df(1) - Linux man page*, 2006 [Em Linha]. Disponível em http://www.die.net/doc/linux/man/man1/df.1.html. [Consultado em 29/06/2006]

Apêndice A - Definição das MIBs gateBox

A.1 gateBoxSystem

```
      GATEBOX-SYSTEM-MIB DEFINITIONS ::= BEGIN

IMPORTS
    GateBox FROM GATEBOX-MIB
    OBJECT-TYPE, NOTIFICATION-TYPE, MODULE-IDENTITY,
    Integer32, Opaque, enterprises, Counter32 FROM SNMPv2-SMI
    TEXTUAL-CONVENTION, DisplayString, TruthValue FROM SNMPv2-TC;

gateBoxSystem MODULE-IDENTITY
    LAST-UPDATED "200605240000Z"            -- 24 May 2006, midnight
    ORGANIZATION "INESC"
    CONTACT-INFO "Editor:   Bruno Oliveira & Vitor Brandão
                            Campus da FEUP
                                        Rua Dr. Roberto Frias, 378
                                        4200 - 465 Porto
                                        Portugal

              email:    admin@gatebox.pt"

    DESCRIPTION  "The GateBox Interfaces MIB"
    ::= { GateBox 1 }

-- Current GateBox core mib table entries:
--   diskTable        OBJECT IDENTIFIER ::= { gateBoxSystem10 }
--   loadTable        OBJECT IDENTIFIER ::= { gateBoxSystem11 }

gbDskTable OBJECT-TYPE
    SYNTAX  SEQUENCE OF gbDskEntry
    MAX-ACCESS    not-accessible
    STATUS  current
    DESCRIPTION
      "Disk watching information.  Partions to be watched
       are configured by the snmpd.conf file of the agent."
    ::= { gateBoxSystem 10 }

gbDskEntry OBJECT-TYPE
    SYNTAX     gbDskEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
        "An entry containing a disk and its statistics."
    INDEX      { gbDskIndex }
    ::= { gbDskTable  1 }

gbDskEntry ::= SEQUENCE {
    gbDskIndex          Integer32,
    gbDskPath           DisplayString,
    gbDskDevice         DisplayString,
    gbDskMinimum        Integer32,
    gbDskMinPercent     Integer32,
```

```
    gbDskTotal          Integer32,
    gbDskAvail          Integer32,
    gbDskUsed           Integer32,
    gbDskPercent        Integer32,
    gbDskPercentNode    Integer32,
    gbDskErrorFlag      Integer32,
    gbDskErrorMsg       DisplayString
}

gbDskIndex OBJECT-TYPE
    SYNTAX  Integer32 (0..65535)
    MAX-ACCESS    read-only
    STATUS  current
    DESCRIPTION
      "Integer reference number (row number) for the disk mib."
    ::= { gbDskEntry 1 }

gbDskPath OBJECT-TYPE
    SYNTAX  DisplayString
    MAX-ACCESS    read-only
    STATUS  current
    DESCRIPTION
      "Path where the disk is mounted."
    ::= { gbDskEntry 2 }

gbDskDevice OBJECT-TYPE
    SYNTAX  DisplayString
    MAX-ACCESS    read-only
    STATUS  current
    DESCRIPTION
      "Path of the device for the partition"
    ::= { gbDskEntry 3 }

gbDskMinimum OBJECT-TYPE
    SYNTAX  Integer32
    MAX-ACCESS    read-only
    STATUS  current
    DESCRIPTION
      "Minimum space required on the disk (in kBytes) before the
         errors are triggered.  Either this or gbDskMinPercent is
         configured via the agent's snmpd.conf file."
    ::= { gbDskEntry 4 }

gbDskMinPercent OBJECT-TYPE
    SYNTAX  Integer32
    MAX-ACCESS    read-only
    STATUS  current
    DESCRIPTION
      "Percentage of minimum space required on the disk before the
         errors are triggered.  Either this or gbDskMinimum is
         configured via the agent's snmpd.conf file."
    ::= { gbDskEntry 5 }

gbDskTotal OBJECT-TYPE
    SYNTAX  Integer32
```

```
    MAX-ACCESS    read-only
    STATUS  current
    DESCRIPTION
      "Total size of the disk/partion (kBytes)"
    ::= { gbDskEntry 6 }

gbDskAvail OBJECT-TYPE
    SYNTAX  Integer32
    MAX-ACCESS    read-only
    STATUS  current
    DESCRIPTION
      "Available space on the disk"
    ::= { gbDskEntry 7 }

gbDskUsed OBJECT-TYPE
    SYNTAX  Integer32
    MAX-ACCESS    read-only
    STATUS  current
    DESCRIPTION
      "Used space on the disk"
    ::= { gbDskEntry 8 }

gbDskPercent OBJECT-TYPE
    SYNTAX  Integer32
    MAX-ACCESS    read-only
    STATUS  current
    DESCRIPTION
      "Percentage of space used on disk"
    ::= { gbDskEntry 9 }

gbDskPercentNode OBJECT-TYPE
    SYNTAX  Integer32
    MAX-ACCESS    read-only
    STATUS  current
    DESCRIPTION
      "Percentage of inodes used on disk"
    ::= { gbDskEntry 10 }

gbDskErrorFlag OBJECT-TYPE
    SYNTAX  Integer32
    MAX-ACCESS    read-only
    STATUS  current
    DESCRIPTION
      "Error flag signaling that the disk or partition is under
       the minimum required space configured for it."
    ::= { gbDskEntry 100 }

gbDskErrorMsg OBJECT-TYPE
    SYNTAX  DisplayString
    MAX-ACCESS    read-only
    STATUS  current
    DESCRIPTION
      "A text description providing a warning and the space left
       on the disk."
    ::= { gbDskEntry 101 }
```

```
gbMemory OBJECT IDENTIFIER ::= { gateBoxSystem 11 }

gbMemIndex OBJECT-TYPE
    SYNTAX  Integer32
    MAX-ACCESS    read-only
    STATUS  current
    DESCRIPTION
      "Bogus Index.  This should always return the integer 0."
    ::= { gbMemory 1 }

gbMemErrorName OBJECT-TYPE
    SYNTAX  DisplayString
    MAX-ACCESS    read-only
    STATUS  current
    DESCRIPTION
      "Bogus Name. This should always return the string 'swap'."
    ::= { gbMemory 2 }

gbMemTotalSwap OBJECT-TYPE
    SYNTAX  Integer32
    MAX-ACCESS    read-only
    STATUS  current
    DESCRIPTION
      "Total Swap Size configured for the host."
    ::= { gbMemory 3 }

gbMemAvailSwap OBJECT-TYPE
    SYNTAX  Integer32
    MAX-ACCESS    read-only
    STATUS  current
    DESCRIPTION
      "Available Swap Space on the host."
    ::= { gbMemory 4 }

gbMemTotalReal OBJECT-TYPE
    SYNTAX  Integer32
    MAX-ACCESS    read-only
    STATUS  current
    DESCRIPTION
      "Total Real/Physical gbMemory Size on the host."
    ::= { gbMemory 5 }

gbMemAvailReal OBJECT-TYPE
    SYNTAX  Integer32
    MAX-ACCESS    read-only
    STATUS  current
    DESCRIPTION
      "Available Real/Physical gbMemory Space on the host."
    ::= { gbMemory 6 }

gbMemTotalSwapTXT OBJECT-TYPE
    SYNTAX  Integer32
    MAX-ACCESS    read-only
```

```
    STATUS  current
    DESCRIPTION
      "Total virtual gbMemory used by text."
    ::= { gbMemory 7 }

gbMemAvailSwapTXT OBJECT-TYPE
    SYNTAX  Integer32
    MAX-ACCESS    read-only
    STATUS  current
    DESCRIPTION
      "Active virtual gbMemory used by text."
    ::= { gbMemory 8 }

gbMemTotalRealTXT OBJECT-TYPE
    SYNTAX  Integer32
    MAX-ACCESS    read-only
    STATUS  current
    DESCRIPTION
      "Total Real/Physical gbMemory Size used by text."
    ::= { gbMemory 9 }

gbMemAvailRealTXT OBJECT-TYPE
    SYNTAX  Integer32
    MAX-ACCESS    read-only
    STATUS  current
    DESCRIPTION
      "Active Real/Physical gbMemory Space used by text."
    ::= { gbMemory 10 }

gbMemTotalFree OBJECT-TYPE
    SYNTAX  Integer32
    MAX-ACCESS    read-only
    STATUS  current
    DESCRIPTION
      "Total Available gbMemory on the host"
    ::= { gbMemory 11 }

gbMemMinimumSwap OBJECT-TYPE
    SYNTAX  Integer32
    MAX-ACCESS    read-only
    STATUS  current
    DESCRIPTION
      "Minimum amount of free swap required to be free
       or else gbMemErrorSwap is set to 1 and an error string is
       returned gbMemSwapErrorMsg."
    ::= { gbMemory 12 }

gbMemShared OBJECT-TYPE
    SYNTAX  Integer32
    MAX-ACCESS    read-only
    STATUS  current
    DESCRIPTION
      "Total Shared gbMemory"
    ::= { gbMemory 13 }
```

```
gbMemBuffer OBJECT-TYPE
    SYNTAX  Integer32
    MAX-ACCESS    read-only
    STATUS  current
    DESCRIPTION
      "Total Buffered gbMemory"
    ::= { gbMemory 14 }

gbMemCached OBJECT-TYPE
    SYNTAX  Integer32
    MAX-ACCESS    read-only
    STATUS  current
    DESCRIPTION
      "Total Cached gbMemory"
    ::= { gbMemory 15 }

gbMemSwapError OBJECT-TYPE
    SYNTAX  Integer32
    MAX-ACCESS    read-only
    STATUS  current
    DESCRIPTION
      "Error flag.  1 indicates very little swap space left"
    ::= { gbMemory 100 }

gbMemSwapErrorMsg OBJECT-TYPE
    SYNTAX  DisplayString
    MAX-ACCESS    read-only
    STATUS  current
    DESCRIPTION
      "Error message describing the Error Flag condition"
    ::= { gbMemory 101 }


gbActions OBJECT IDENTIFIER ::= { gateBoxSystem 12 }

gbDoReboot OBJECT-TYPE
    SYNTAX  Integer32
    MAX-ACCESS    read-write
    STATUS  current
    DESCRIPTION
      "If DoReboot is set to 1 then system is rebooted"
    ::= { gbActions 1 }

gbDoGateboxRestart OBJECT-TYPE
    SYNTAX  Integer32
    MAX-ACCESS    read-write
    STATUS  current
    DESCRIPTION
      "If GateboxRestart is set to 1 then the GateBox is rebooted"
    ::= { gbActions 2 }

gbDoUpdateDb OBJECT-TYPE
    SYNTAX  DisplayString
    MAX-ACCESS    read-write
    STATUS  current
```

```
    DESCRIPTION
      "Update DataBase <string>"
    ::= { gbActions 3 }

gbDoGateboxFirewall OBJECT-TYPE
    SYNTAX  Integer32
    MAX-ACCESS    read-write
    STATUS  current
    DESCRIPTION
      "GateboxFirewall <string>"
    ::= { gbActions 4 }

gbDoSshdRestart OBJECT-TYPE
    SYNTAX  Integer32
    MAX-ACCESS    read-write
    STATUS  current
    DESCRIPTION
      "If gbDoSshdRestart is set to 1 then the SSH daemon is restarted"
    ::= { gbActions 5 }

gbDoFreeradiusRestart OBJECT-TYPE
    SYNTAX  Integer32
    MAX-ACCESS    read-write
    STATUS  current
    DESCRIPTION
      "If gbDoFreeradiusRestart is set to 1 then the freeradius daemon is
restarted"
    ::= { gbActions 6 }

END
```

A.2 gateBoxInterfaces

```
      GATEBOX-INTERFACES-MIB DEFINITIONS ::= BEGIN

IMPORTS
      GateBox                                    FROM GATEBOX-MIB
    MODULE-IDENTITY, OBJECT-TYPE, Counter32, Gauge32, Counter64,
    Integer32, TimeTicks, mib-2,
    NOTIFICATION-TYPE                          FROM SNMPv2-SMI
    TEXTUAL-CONVENTION, DisplayString,
    PhysAddress, TruthValue, RowStatus,
    TimeStamp, AutonomousType, TestAndIncr     FROM SNMPv2-TC
    MODULE-COMPLIANCE, OBJECT-GROUP,
    NOTIFICATION-GROUP                         FROM SNMPv2-CONF
    snmpTraps                                  FROM SNMPv2-MIB
    IANAifType                                 FROM IANAifType-MIB;

gateBoxInterfaces MODULE-IDENTITY
    LAST-UPDATED "200605040000Z"            -- 04 May 2006, midnight
    ORGANIZATION "INESC"
    CONTACT-INFO "Editor:   Vitor Brandão
                            Campus da FEUP
                                        Rua Dr. Roberto Frias, 378
                                        4200 - 465 Porto
                                        Portugal

              email:    vsilva@inescporto.pt"

    DESCRIPTION  "The INESC GateBox Interfaces MIB"
    ::= { GateBox 2 }


gbIfNumber  OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
            "The number of network interfaces (regardless of their
            current state) present on this system."
    ::= { gateBoxInterfaces 1 }

gbIfTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF gbIfEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "A list of interface entries.  The number of entries is
            given by the value of ifNumber."
    ::= { gateBoxInterfaces 2 }

gbIfEntry OBJECT-TYPE
    SYNTAX      gbIfEntry
```

```
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "An entry containing management information applicable to a
            particular interface."
    INDEX   { gbIfIndex }
    ::= { gbIfTable 1 }

gbIfEntry ::=
    SEQUENCE {
        gbIfIndex                 InterfaceIndex,
        gbIfDescr                 DisplayString,
        gbIfType                  IANAifType,
        gbIfMtu                   Integer32,
        gbIfSpeed                 Gauge32,
        gbIfPhysAddress           PhysAddress,
        gbIfAdminStatus           INTEGER,
        gbIfOperStatus            INTEGER,
        gbIfLastChange            TimeTicks,
        gbIfInOctets              Counter32,
        gbIfInUcastPkts           Counter32,
        gbIfInNUcastPkts          Counter32,  -- deprecated
        gbIfInDiscards            Counter32,
        gbIfInErrors              Counter32,
        gbIfInUnknownProtos       Counter32,
        gbIfOutOctets             Counter32,
        gbIfOutUcastPkts          Counter32,
        gbIfOutNUcastPkts         Counter32,  -- deprecated
        gbIfOutDiscards           Counter32,
        gbIfOutErrors             Counter32,
        gbIfOutQLen               Gauge32,    -- deprecated
        gbIfSpecific              OBJECT IDENTIFIER -- deprecated
    }

gbIfIndex OBJECT-TYPE
    SYNTAX      InterfaceIndex
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
            "A unique value, greater than zero, for each interface.  It
            is recommended that values are assigned contiguously
            starting from 1.  The value for each interface sub-layer
            must remain constant at least from one re-initialization of
            the entity's network management system to the next re-
            initialization."
    ::= { gbIfEntry 1 }

gbIfDescr OBJECT-TYPE
    SYNTAX      DisplayString (SIZE (0..255))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
            "A textual string containing information about the
            interface.  This string should include the name of the
            manufacturer, the product name and the version of the
```

```
            interface hardware/software."
    ::= { gbIfEntry 2 }

gbIfType OBJECT-TYPE
    SYNTAX      IANAifType
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
            "The type of interface.  Additional values for ifType are
            assigned by the Internet Assigned Numbers Authority (IANA),
            through updating the syntax of the IANAifType textual
            convention."
    ::= { gbIfEntry 3 }

gbIfMtu OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
            "The size of the largest packet which can be sent/received
            on the interface, specified in octets.  For interfaces that
            are used for transmitting network datagrams, this is the
            size of the largest network datagram that can be sent on the
            interface."
    ::= { gbIfEntry 4 }

gbIfSpeed OBJECT-TYPE
    SYNTAX      Gauge32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
            "An estimate of the interface's current bandwidth in bits
            per second.  For interfaces which do not vary in bandwidth
            or for those where no accurate estimation can be made, this
            object should contain the nominal bandwidth.  If the
            bandwidth of the interface is greater than the maximum value
            reportable by this object then this object should report its
            maximum value (4,294,967,295) and ifHighSpeed must be used
            to report the interace's speed.  For a sub-layer which has
            no concept of bandwidth, this object should be zero."
    ::= { gbIfEntry 5 }

gbIfPhysAddress OBJECT-TYPE
    SYNTAX      PhysAddress
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
            "The interface's address at its protocol sub-layer.  For
            example, for an 802.x interface, this object normally
            contains a MAC address.  The interface's media-specific MIB
            must define the bit and byte ordering and the format of the
            value of this object.  For interfaces which do not have such
            an address (e.g., a serial line), this object should contain
            an octet string of zero length."
    ::= { gbIfEntry 6 }
```

```
gbIfAdminStatus OBJECT-TYPE
    SYNTAX  INTEGER {
                up(1),       -- ready to pass packets
                down(2),
                testing(3)   -- in some test mode
            }
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
            "The desired state of the interface.  The testing(3) state
            indicates that no operational packets can be passed.  When a
            managed system initializes, all interfaces start with
            ifAdminStatus in the down(2) state.  As a result of either
            explicit management action or per configuration information
            retained by the managed system, ifAdminStatus is then
            changed to either the up(1) or testing(3) states (or remains
            in the down(2) state)."
    ::= { gbIfEntry 7 }

gbIfOperStatus OBJECT-TYPE
    SYNTAX  INTEGER {
                up(1),         -- ready to pass packets
                down(2),
                testing(3),    -- in some test mode
                unknown(4),    -- status can not be determined
                               -- for some reason.
                dormant(5),
                notPresent(6),    -- some component is missing
                lowerLayerDown(7) -- down due to state of
                                  -- lower-layer interface(s)
            }
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
            "The current operational state of the interface.  The
            testing(3) state indicates that no operational packets can
            be passed.  If ifAdminStatus is down(2) then ifOperStatus
            should be down(2).  If ifAdminStatus is changed to up(1)
            then ifOperStatus should change to up(1) if the interface is
            ready to transmit and receive network traffic; it should
            change to dormant(5) if the interface is waiting for
            external actions (such as a serial line waiting for an
            incoming connection); it should remain in the down(2) state
            if and only if there is a fault that prevents it from going
            to the up(1) state; it should remain in the notPresent(6)
            state if the interface has missing (typically, hardware)
            components."
    ::= { gbIfEntry 8 }

gbIfLastChange OBJECT-TYPE
    SYNTAX      TimeTicks
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
```

```
            "The value of sysUpTime at the time the interface entered
            its current operational state.  If the current state was
            entered prior to the last re-initialization of the local
            network management subsystem, then this object contains a
            zero value."
    ::= { gbIfEntry 9 }

gbIfInOctets OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
            "The total number of octets received on the interface,
            including framing characters.

            Discontinuities in the value of this counter can occur at
            re-initialization of the management system, and at other
            times as indicated by the value of
            ifCounterDiscontinuityTime."
    ::= { gbIfEntry 10 }

gbIfInUcastPkts OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
            "The number of packets, delivered by this sub-layer to a
            higher (sub-)layer, which were not addressed to a multicast
            or broadcast address at this sub-layer.

            Discontinuities in the value of this counter can occur at
            re-initialization of the management system, and at other
            times as indicated by the value of
            ifCounterDiscontinuityTime."
    ::= { gbIfEntry 11 }

gbIfInNUcastPkts OBJECT-TYPE
    SYNTAX  Counter32
    MAX-ACCESS  read-only
    STATUS      deprecated
    DESCRIPTION
            "The number of packets, delivered by this sub-layer to a
            higher (sub-)layer, which were addressed to a multicast or
            broadcast address at this sub-layer.

            Discontinuities in the value of this counter can occur at
            re-initialization of the management system, and at other
            times as indicated by the value of
            ifCounterDiscontinuityTime.

            This object is deprecated in favour of ifInMulticastPkts and
            ifInBroadcastPkts."
    ::= { gbIfEntry 12 }

gbIfInDiscards OBJECT-TYPE
```

```
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
            "The number of inbound packets which were chosen to be
            discarded even though no errors had been detected to prevent

            their being deliverable to a higher-layer protocol.  One
            possible reason for discarding such a packet could be to
            free up buffer space.

            Discontinuities in the value of this counter can occur at
            re-initialization of the management system, and at other
            times as indicated by the value of
            ifCounterDiscontinuityTime."
    ::= { gbIfEntry 13 }

gbIfInErrors OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
            "For packet-oriented interfaces, the number of inbound
            packets that contained errors preventing them from being
            deliverable to a higher-layer protocol.  For character-
            oriented or fixed-length interfaces, the number of inbound
            transmission units that contained errors preventing them
            from being deliverable to a higher-layer protocol.

            Discontinuities in the value of this counter can occur at
            re-initialization of the management system, and at other
            times as indicated by the value of
            ifCounterDiscontinuityTime."
    ::= { gbIfEntry 14 }

gbIfInUnknownProtos OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
            "For packet-oriented interfaces, the number of packets
            received via the interface which were discarded because of
            an unknown or unsupported protocol.  For character-oriented
            or fixed-length interfaces that support protocol
            multiplexing the number of transmission units received via
            the interface which were discarded because of an unknown or
            unsupported protocol.  For any interface that does not
            support protocol multiplexing, this counter will always be
            0.

            Discontinuities in the value of this counter can occur at
            re-initialization of the management system, and at other
            times as indicated by the value of
            ifCounterDiscontinuityTime."
    ::= { gbIfEntry 15 }
```

```
gbIfOutOctets OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
            "The total number of octets transmitted out of the
            interface, including framing characters.

            Discontinuities in the value of this counter can occur at
            re-initialization of the management system, and at other
            times as indicated by the value of
            ifCounterDiscontinuityTime."
    ::= { gbIfEntry 16 }

gbIfOutUcastPkts OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
            "The total number of packets that higher-level protocols
            requested be transmitted, and which were not addressed to a
            multicast or broadcast address at this sub-layer, including
            those that were discarded or not sent.

            Discontinuities in the value of this counter can occur at
            re-initialization of the management system, and at other
            times as indicated by the value of
            ifCounterDiscontinuityTime."
    ::= { gbIfEntry 17 }

gbIfOutNUcastPkts OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      deprecated
    DESCRIPTION
            "The total number of packets that higher-level protocols
            requested be transmitted, and which were addressed to a
            multicast or broadcast address at this sub-layer, including
            those that were discarded or not sent.

            Discontinuities in the value of this counter can occur at
            re-initialization of the management system, and at other
            times as indicated by the value of
            ifCounterDiscontinuityTime.

            This object is deprecated in favour of ifOutMulticastPkts
            and ifOutBroadcastPkts."
    ::= { gbIfEntry 18 }

gbIfOutDiscards OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
```

```
            "The number of outbound packets which were chosen to be
            discarded even though no errors had been detected to prevent
            their being transmitted.  One possible reason for discarding
            such a packet could be to free up buffer space.

            Discontinuities in the value of this counter can occur at
            re-initialization of the management system, and at other
            times as indicated by the value of
            ifCounterDiscontinuityTime."
    ::= { gbIfEntry 19 }

gbIfOutErrors OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
            "For packet-oriented interfaces, the number of outbound
            packets that could not be transmitted because of errors.
            For character-oriented or fixed-length interfaces, the
            number of outbound transmission units that could not be
            transmitted because of errors.

            Discontinuities in the value of this counter can occur at
            re-initialization of the management system, and at other
            times as indicated by the value of
            ifCounterDiscontinuityTime."
    ::= { gbIfEntry 20 }

gbIfOutQLen OBJECT-TYPE
    SYNTAX      Gauge32
    MAX-ACCESS  read-only
    STATUS      deprecated
    DESCRIPTION
            "The length of the output packet queue (in packets)."
    ::= { gbIfEntry 21 }

gbIfSpecific OBJECT-TYPE
    SYNTAX      OBJECT IDENTIFIER
    MAX-ACCESS  read-only
    STATUS      deprecated
    DESCRIPTION
            "A reference to MIB definitions specific to the particular
            media being used to realize the interface.  It is

            recommended that this value point to an instance of a MIB
            object in the media-specific MIB, i.e., that this object
            have the semantics associated with the InstancePointer
            textual convention defined in RFC 2579.  In fact, it is
            recommended that the media-specific MIB specify what value
            ifSpecific should/can take for values of ifType.  If no MIB
            definitions specific to the particular media are available,
            the value should be set to the OBJECT IDENTIFIER { 0 0 }."
    ::= { gbIfEntry 22 }

END
```

A.3 gateBoxIp

```
      GATEBOX-IP-MIB DEFINITIONS ::= BEGIN

IMPORTS
      GateBox                                    FROM GATEBOX-MIB
      MODULE-IDENTITY, OBJECT-TYPE,
            Integer32                FROM SNMPv2-SMI
      MODULE-COMPLIANCE, OBJECT-GROUP           FROM SNMPv2-CONF;

gateBoxIp MODULE-IDENTITY
    LAST-UPDATED "200605040000Z"            -- 04 May 2006, midnight
    ORGANIZATION "INESC"
    CONTACT-INFO "Editor:   Vitor Brandão
                            Campus da FEUP
                                        Rua Dr. Roberto Frias, 378
                                        4200 - 465 Porto
                                        Portugal

              email:    vsilva@inescporto.pt"

    DESCRIPTION  "The INESC GateBox MIB"
    ::= { GateBox 4 }



gbIPv4Flags   OBJECT IDENTIFIER ::= { gateBoxIp 1 }

gbIpAutoconfig OBJECT-TYPE
    SYNTAX     INTEGER {
                    not(0),     -- ...
                    HostReceivedIpConfiguration(1)  -- ...
               }
    MAX-ACCESS read-write
    STATUS     current
    DESCRIPTION
           "[ /proc/sys/net/ipv4/ip_autoconfig ]
           This file contains the number one if the host received its IP
configuration by RARP, BOOTP, DHCP or a similar mechanism. Otherwise it is
zero.
"
    ::= { gbIPv4Flags 1 }

gbIpDefaultTtl OBJECT-TYPE
    SYNTAX     Integer32 (1..255)
    MAX-ACCESS read-write
    STATUS     current
    DESCRIPTION
           "[ /proc/sys/net/ipv4/ip_default_ttl ]
           The default value inserted into the Time-To-Live field of the IP
header of datagrams originated at this entity, whenever a TTL value is not
supplied by the transport layer protocol."
```

```
    ::= { gbIPv4Flags 2 }

gbIpDynaddr OBJECT-TYPE
    SYNTAX      INTEGER
    MAX-ACCESS read-write
    STATUS      current
    DESCRIPTION
            "[ /proc/sys/net/ipv4/ip_dynaddr ]
            You need to set this if you use dial-on-demand with a dynamic
interface address. Once your demand interface comes up, any local TCP sock-
ets which haven't seen replies will be rebound to have the right address.
This solves the problem that the connection that brings up your interface
itself does not work, but the second try does."
    ::= { gbIPv4Flags 3 }

gbIpForward OBJECT-TYPE
    SYNTAX      INTEGER {
                     forwarding(1),     -- acting as a router
                     notForwarding(2)   -- NOT acting as a router
                }
    MAX-ACCESS read-write
    STATUS      current
    DESCRIPTION
            "[ /proc/sys/net/ipv4/ip_forward ]
            The indication of whether this entity is acting as an IP router
in respect to the forwarding of datagrams received by, but not addressed
to, this entity.  IP routers forward datagrams.  IP hosts do not (except
those source-routed via the host)."
    ::= { gbIPv4Flags 4 }

END
```