FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

FEUP

Gestão Integrada SNMP de um Servidor de *E-mail*

Sunny Narendra

Dissertação realizada no âmbito do
Mestrado Integrado em Engenharia Electrotécnica e de Computadores
Major Telecomunicações

Orientador: Professor João Manuel Couto das Neves

Junho de 2009

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES 2008/2009

A Dissertação intitulada

"GESTÃO INTEGRADA SNMP DE UM SERVIDOR DE E-MAIL"

foi aprovada em provas realizadas em 23/Julho/2009

o júri

Presidente Professor Doutor Eurico Manuel Elias Morais Carrapatoso
Professor Associado do Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Professor Doutor Joaquim Melo Henriques Macedo
Professor Auxiliar do Departamento de Informática da Escola de Engenharia da Universidade do Minho

Professor Doutor João Manuel Couto das Neves
Professor Auxiliar Convidado do Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projecto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extractos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são correctamente citados.

Sunny Narendra

Autor - **SUNNY NARENDRA**

Faculdade de Engenharia da Universidade do Porto

Resumo

Actualmente o correio electrónico (*e-mail*) é um dos serviços essenciais e mais utilizados na Internet, incluindo nas redes Intranet. Os utilizadores e os processos de fluxo de informação baseiam a comunicação no *e-mail*, por isso os servidores de *e-mail* devem fornecer um serviço rápido, seguro e com um elevado grau de fiabilidade.

Existem vários tipos de servidores que oferecem este tipo de serviço, mas onde a sua gestão e a monitorização nem sempre é fácil. Consequentemente, é uma preocupação constante para a maioria dos administradores de sistemas responsáveis pela gestão deste tipo de serviço. É necessário desenvolver soluções que permitam resolver os problemas de gestão que são confrontados com os servidores de *e-mail*, bem como melhorar o seu desempenho de funcionamento.

Neste trabalho, a primeira parte é dedicada à caracterização dos requisitos de desempenho de um servidor de *e-mail*, posteriormente são verificados quais é que seriam críticos para o seu funcionamento, e por último, a monitorização e a gestão desses requisitos com recurso ao protocolo de gestão (SNMP).

Abstract

Today electronic mail (e-mail) is one of the essential and most used services on the Internet, including the Intranet networks. The users and work-flow processes based on e-mail communication, therefore the mail servers must provide a fast, safe and highly reliable service.

There are various type of servers which offer this kind of service, but managing and monitoring them isn't always easy. Consequently it is a constant concern for most system administrators responsible for managing this type of service. It is necessary to develop methods which will overcome the management problems currently faced with the mail servers as well as improve its operating performance.

In this work, the first part concentrates on establishing the performance requirements of a mail server, then observations are made as to which of them would be critical to its functioning, and finally, monitoring and management of these requirements are made using the management protocol (SNMP).

Agradecimentos

Em primeiro lugar agradeço ao meu orientador, Professor João Neves, pela forma como orientou o meu trabalho, pelo encorajamento que me foi transmitido, e pelo valioso apoio prestado em todas as fases de execução desta dissertação.

Um agradecimento muito especial à minha família.

Por último, agradeço a todos aqueles que não foram mencionados e que me apoiaram na realização deste trabalho.

Sunny Narendra

Índice

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

API	Application Programming Interface
ARPANet	Advanced Research Projects Agency Network
ASCII	American Standard Code for Information Interchange
ASN.1	Abstract Syntax Notation One
BER	Basic Encoding Rules
CA	Certification Authority
DKIM	DomainKeys Identified Mail
DNS	Domain Name System
DoD	United States Department of Defense
DoS	Denial of Service
EGP	Exterior Gateway Protocol
ESMTP	Extended SMTP
IANA	Internet Assigned Numbers Authority
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
IMAP	Internet Message Access Protocol
IMAPS	IMAP protocol over TLS/SSL
IP	Internet Protocol
ISO	International Organization for Standardization
ISP	Internet Service Provider
ITU-T	Telecommunication Standardization Sector
LDAP	Lightweight Directory Access Protocol
LMTP	Local Mail Transfer Protocol
MAPI	Messaging Application Programming Interface
MDA	Mail Delivery Agent
MIB	Management Information Base
MIME	Multipurpose Internet Mail Extensions
MSA	Mail Submission Agent
MTA	Message Transfer Agent
MUA	Mail User Agent
MX	Mail Exchange
NAT	Network Address Translation
NMS	Network Management Station
OID	Object Identifier
OSI	Open Systems Interconnection
PDU	Protocol Data Unit

PGP	Pretty Good Privacy
PID	Process ID
POP3	Post Office Protocol, version 3
POP3S	POP3 protocol over TLS/SSL
QoS	Quality of Service
RFC	Request for Comments
S/MIME	Secure MIME
SASL	Simple Authentication and Security Layer
SMI	Structure of Management Information
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SPF	Sender Policy Framework
SSH	Secure Shell Protocol
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UA	User Agent
UDP	User Datagram Protocol

Capítulo 1

Introdução

O presente capítulo pretende fornecer uma visão global do trabalho desenvolvido durante a dissertação. Inicia-se com uma breve motivação do tema abordado, posteriormente são descritos os objectivos e, na última parte, é apresentada a estrutura da dissertação.

1.1 Motivação

As mensagens de correio electrónico, vulgarmente referenciadas por *e-mail* (da expressão inglesa *electronic mail*), já deixaram de ser novidade e são consideradas de elevada importância para a maioria das empresas. Para estas, o serviço de *e-mail* é um instrumento essencial nas transacções comerciais, no atendimento dos clientes e nas negociações com os fornecedores.

Como é conhecido por todos nós o serviço de *e-mail* apresenta muitas vantagens em relação ao tradicional correio postal (*surface mail*) e mesmo ao serviço de FAX, entre as quais se destacam a sua rapidez e o baixo custo de envio. As mensagens podem ser lidas e escritas em qualquer momento, independente dos fusos horários e do horário comercial. Geralmente estas podem ser entregues a um destinatário em poucos minutos e as respostas também podem ser rápidas. Assim um servidor de *e-mail* deve ser capaz de integrar com os diversos sistemas já existentes, dar resposta eficiente a um elevado volume de tráfego de *e-mail* e ser robusto contra falhas. Sendo a gestão o elemento fundamental para garantir os requisitos de desempenho necessários [1].

Actualmente existe muita diversidade de servidores de *e-mail*, desde soluções proprietárias às de domínio público, como também as funcionalidades oferecidas pelos mesmos serem diversas, criando assim desafios quanto à sua gestão. Logo é necessário desenvolver plataformas de gestão ou adaptar as existentes de modo a que sejam capazes de efectuar a gestão deste tipo de servidores e garantir os seus requisitos de desempenho. Devendo estas plataformas ter compatibilidade com os diversos tipos de servidores de *e-mail* existentes.

1.2 Objectivos

O objectivo desta dissertação consiste no estudo detalhado do funcionamento de um servidor de *e-mail*, avaliando os seus requisitos de desempenho e identificando os problemas associados à sua gestão. Pretende-se também que sejam propostas soluções para a sua monitorização e gestão com recurso ao protocolo de gestão de redes SNMP (*Simple Network Management Protocol*).

Por fim, para demonstrar o resultado da solução encontrada será desenvolvida uma interface de gestão *Web* + SNMP para a monitorização em "tempo-real" e a gestão das configurações de um servidor de *e-mail*.

1.3 Estrutura da Dissertação

De forma a melhorar a organização deste relatório, dividiu-se o mesmo em seis capítulos, sendo este o primeiro e é introdução.

No capítulo 2 procura-se descrever de uma forma sucinta o funcionamento do serviço de *e-mail*. São abordados alguns dos protocolos mais relevantes para o seu funcionamento, como também são definidos os seus requisitos de funcionamento e apresentados alguns dos mecanismos de segurança que actualmente são utilizados no serviço de *e-mail*. Na parte final, apresenta-se os servidores de *e-mail* que foram escolhidos como casos de estudo.

O capítulo 3 é destinado à introdução da gestão de redes e à apresentação do protocolo SNMP. Aqui é feito um estudo detalhado deste protocolo para a verificação do que pode ser aproveitado do mesmo para o problema em questão.

No capítulo 4 é realizada a caracterização do problema, bem como a identificação dos pontos críticos do funcionamento dos servidores de *e-mail*. A última parte deste capítulo é dedicada à realização de um estudo do actual estado da arte, procurando-se identificar as ferramentas e plataformas de gestão dos servidores de *e-mail* existentes actualmente.

A apresentação de um sistema de gestão é feita no capítulo 5. Aqui é descrito com detalhe as escolhas que foram tomadas para a concretização deste sistema, bem como a explicação do seu funcionamento e da sua lógica de implementação. O sistema implementado permite a monitorização e a gestão de alguns parâmetros dos servidores de *e-mail*, como também o envio de notificações quando necessárias.

Por último, no capítulo 6 é criado um possível cenário para a utilização do sistema de gestão que foi implementado, é também neste cenário que são realizados alguns dos testes para comprovar o seu correcto funcionamento. No fim são feitas as conclusões finais e apresentadas algumas propostas de trabalho futuro sobre o tema.

No anexo A é incluída a MIB que foi criada e no anexo B apresenta-se o programa utilizado para gerar tráfego de mensagens de *e-mail*.

Capítulo 2

E-mail e Servidores

Neste capítulo é efectuada uma breve introdução do serviço de *e-mail*, bem como do funcionamento dos servidores de *e-mail* com a identificação de alguns dos seus requisitos.

O primeiro sistema de troca de mensagens possibilitava a comunicação entre os múltiplos utilizadores de um computador do tipo *mainframe*. Com o aumento de trocas das mensagens de *e-mail* e do desenvolvimento de redes cada vez mais complexas e exigentes, este modelo veio a sofrer alterações no sentido de adaptação a essas mesmas necessidades. Por isso, ao longo do tempo este serviço foi modificado, passando pela rede de computadores ARPANet (*Advanced Research Projects Agency Network*), que fez muitas contribuições, até ao modelo da Internet que actualmente é usado.

O modelo de *e-mail* tem seguido as recomendações do X.400, norma especificada pelo ITU-T (*Telecommunication Standardization Sector*). Muitas empresas como a IBM, Microsoft, Hewlett Packard entre outras, oferecem diversas soluções proprietárias de serviço de *e-mail*, sendo muitas destas baseadas nesta norma.

2.1 Arquitectura

No modelo X.400 são participantes dois tipos de agentes: o UA (*User Agent*) e o MTA (*Message Transfer Agent*). [2]

UA (*User Agent*)

O UA é o cliente de *e-mail* que permite ao utilizador compor e ler mensagens de *e-mail*, envia mensagens através do MTA e recebe mensagens da caixa de *e-mail* do utilizador.

MTA (*Message Transfer Agent*)

Normalmente é a designação atribuída aos servidores de *e-mail*. A função deste é transportar as mensagens de *e-mail*.

Um mesmo MTA pode desempenhar duas funções, a de servidor ou a de cliente:

- **Servidor MTA** - Quando recebe as mensagens de *e-mail* dos UAs ou MTAs e as coloca na caixa de *e-mail* do respectivo destinatário;

- **Cliente MTA** - Quando recebe uma mensagem de *e-mail* para um utilizador que não se encontra no seu domínio e tem a necessidade de fazer encaminhamento para outro MTA do domínio do destinatário.

No modelo da Internet (RFC 2821), o UA é também referenciado por MUA (*Mail User Agent*) e o MTA é identificado por *Mail Transfer Agent*.

Recentemente têm aparecido agentes que particularizam algumas funcionalidades dos MTAs. Estes estão presentes em diversas normas como: *Internet Mail Architecture* (draft-crocker-email-arch-00) e *Email Submission Operations: Access and Accountability Requirements* (RFC 5068).

MSA (*Mail Submission Agent*)

É uma aplicação que recebe as mensagens de *e-mail* dos UAs, efectua um pré-processamento e entrega aos MTAs.

MDA (*Mail Delivery Agent*)

É uma aplicação que recebe as mensagens de *e-mail* dos MTAs e as coloca na caixa de *e-mail* do respectivo destinatário. Este agente pode fazer a filtragem das mensagens, bem como o redireccionamento para outros endereços de *e-mail*.

2.2 Alguns Conceitos

De seguida são apresentados alguns conceitos relacionados com o serviço de *e-mail*.

Formato das Mensagens de *E-mail*

Uma mensagem de *e-mail* é constituídas por:

- **Envelope** - Que inclui informações sobre o transporte da mensagem;

- **Conteúdo** - Que inclui um cabeçalho (*header*) e a mensagem propriamente dita (*body*).

No cabeçalho estão presentes as informações sobre o remetente, destinatário e outros campos. Tal como ilustrado na figura 2.1.

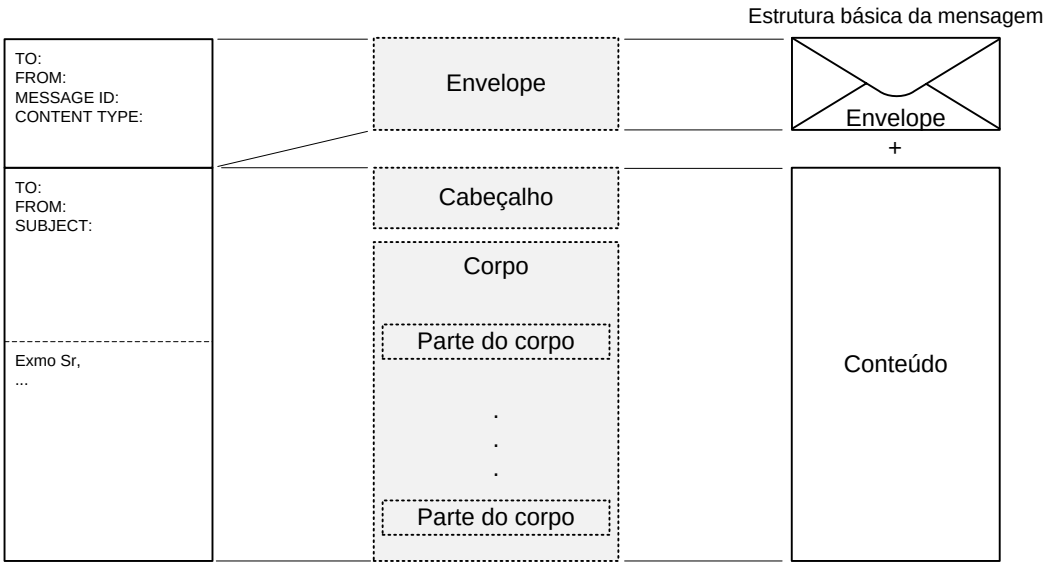

Figura 2.1: Formato de uma mensagem de *e-mail*

Endereço do Remetente

Como no correio postal, as mensagens de *e-mail* no mínimo têm dois tipos de endereços: um no envelope e outro no cabeçalho da mensagem. [3]

O endereço do remetente no envelope (designado de `return-path`) é utilizado durante o transporte da mensagem. Em caso de falha na entrega, o servidor devolve a mensagem para este endereço.

O endereço do remetente no cabeçalho da mensagem é incluído no campo `From` ou `Sender`, sendo este visível nos UAs. Normalmente os servidores de *e-mail* não dão muita importância a este endereço.

Mailbox

O termo *mailbox* refere-se ao repositório onde a mensagem é depositada, ou seja a caixa de *e-mail*. Os formatos mais comuns são: *mbox*, *maildir* e *mbx*. [4] [5]

No ***mbox***, as mensagens são guardadas num único ficheiro. Existe um marcador especial para a separação das mensagens dentro do ficheiro. Apenas permite um único acesso em modo de leitura/escrita, sendo por isso necessário recorrer ao mecanismo de *locking* de ficheiros.

O ***maildir*** apresenta uma estrutura em directórios onde cada mensagem é um ficheiro, deste modo não é necessário o *locking* de ficheiros. Este tipo de formato tem desvantagem na pesquisa de mensagens, pois necessita de abrir vários ficheiros.

O formato ***mbx*** é uma nova versão do formato *mbox*, continua na mesma a ser necessário o mecanismo de *locking* de ficheiros. A grande diferença reside no facto de cada mensagem num único ficheiro ser precedida por um registo que é obtido a partir da mensagem em questão, permitindo assim uma pesquisa mais rápida e eficiente de uma determinada mensagem.

Alias e Listas de Distribuição

Um *alias* é um endereço que serve apenas para reenviar as mensagens de *e-mail* que recebe para um outro endereço. A caixa de *e-mail* deste terá um outro endereço chamado principal, mas tanto o principal como o *alias* representam a mesma entidade.

As listas de distribuição surgiram para a agregação de um grupo de subscritores ou utilizadores. Têm o objectivo de facilitar a difusão de mensagens periódicas a um conjunto de utilizadores que tenham algo em comum.

Serviço de DNS

O DNS (*Domain Name System*) é um serviço que efectua a tradução de um domínio num endereço IP e vice-versa. É constituído por zonas, sendo que uma zona é uma sub-árvore resultante de uma partição hierárquica. [5]

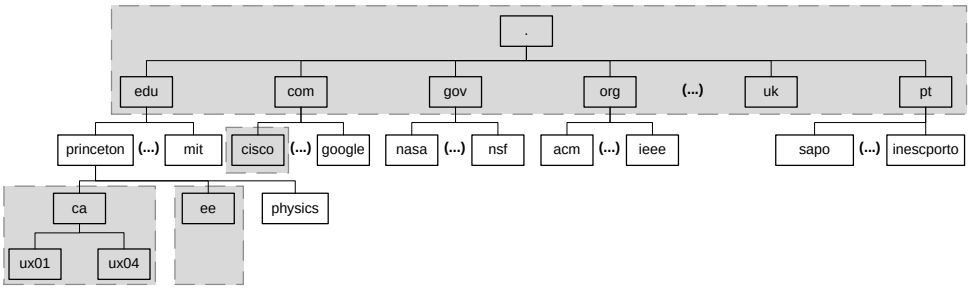

Figura 2.2: Árvore de DNS

Os servidores de nomes (*name servers*) são programas servidores que possuem registos designados de *resource records* de uma ou mais zonas e que periodicamente actualizam as suas bases de dados recorrendo a ficheiros em disco local ou a outros servidores de nomes.

Os diferentes *resource records* oferecem vários tipos de informações, os importantes para o serviço de *e-mail* são:

Registo	**Descrição**
`A` (*Address*)	Endereço IPv4 de um nó (*host*). Para o IPv6 é utilizado o registo `AAA`
`CNAME` (*Canonical Name*)	Nome canónico para um determinado *host*.
`MX` (*Mail Exchange*)	Os registos MXs apontam para os servidores de *e-mail*. Cada registo MX contém informações sobre o servidor e a prioridade para o envio de *e-mails* nesse domínio. Podem existir mais registos do tipo MX, que correspondem a servidores de *e-mail* com uma prioridade mais baixa e que funcionam como servidores secundários. Em caso do servidor primário deixar de funcionar, são os servidores secundários que recebem os *e-mails*.
`PTR` (*Pointer*)	Apontador para um domínio de nomes.
`TXT` (*Text*)	Uma descrição textual informativa, com máximo de 255 caracteres.
`SPF`	Do mesmo tipo do registo TXT. Apenas para as versões recentes de servidores de DNS.
`MINFO`	Informação sobre uma *mailbox* ou *mail list*.

Tabela 2.1: Registos do DNS

O registo TXT é muito utilizado pelo SPF, Sender ID e DKIM que são mecanismos de autenticação do remetente. Este tópico será desenvolvido na secção 2.6.2.1.

2.3 Protocolos de Transferência

O transporte de mensagens de *e-mail* pode ser realizado com o protocolo SMTP (*Simple Mail Transfer Protocol*) que foi inicialmente descrito no RFC 821 [6]. Existem algumas extensões deste protocolo, como o ESMTP (*SMTP Service Extensions*) - RFC 1869, que oferecem mais funcionalidades e conservam em geral toda a sintaxe do SMTP.

O acesso às caixas de *e-mail* pode ser realizado de diversas formas:

- Directamente, através de comandos executados localmente no servidor;
- Através do protocolo POP3 (*Post Office Protocol, version 3*) descrito no RFC 1939;
- Através do protocolo IMAP (*Internet Message Access Protocol*) descrito no RFC 1730.

2.3.1 SMTP

O protocolo SMTP administra a transmissão de mensagens de *e-mail* através das redes, o transporte é efectuado sobre o protocolo TCP (*Transmission Control Protocol*) no porto 25, que define um conjunto de comandos, respostas e procedimentos para a troca de mensagens entre os MTAs e UAs.

2.3.1.1 Sintaxe do SMTP

Os comandos e as respostas do SMTP têm uma sintaxe rígida. Todos comandos são feitos no sentido do cliente para servidor, ou seja, somente o cliente requer os comandos, enquanto o servidor os executa. As respostas começam todas com um código de três dígitos numéricos.

A sintaxe do SMTP é composta por caracteres do código ASCII, que segue o padrão de 7 bits. Não se deve transmitir com 8 bits sem o acordo de ambas as partes (cliente e servidor). Se um destes enviar sem o conhecimento do outro, poderão ocorrer situações em que ignorem o 1º bit mais significativo.

Na figura 2.3 é apresentada uma interacção típica entre o UA e o MTA.

2.3.1.2 Extensões do Protocolo

É quase impossível fazer alterações no protocolo SMTP devido à sua enorme base de servidores de *e-mail* que utilizam este protocolo. Para contornar os problemas que vão surgindo, são criadas extensões ao protocolo. No entanto, a necessidade de compatibilidade com os servidores já existentes, baseados no protocolo base definido no RFC 821, faz com que estas extensões sejam meramente opcionais, ou seja, apenas são usadas quando ambas as partes intervenientes as suportem.

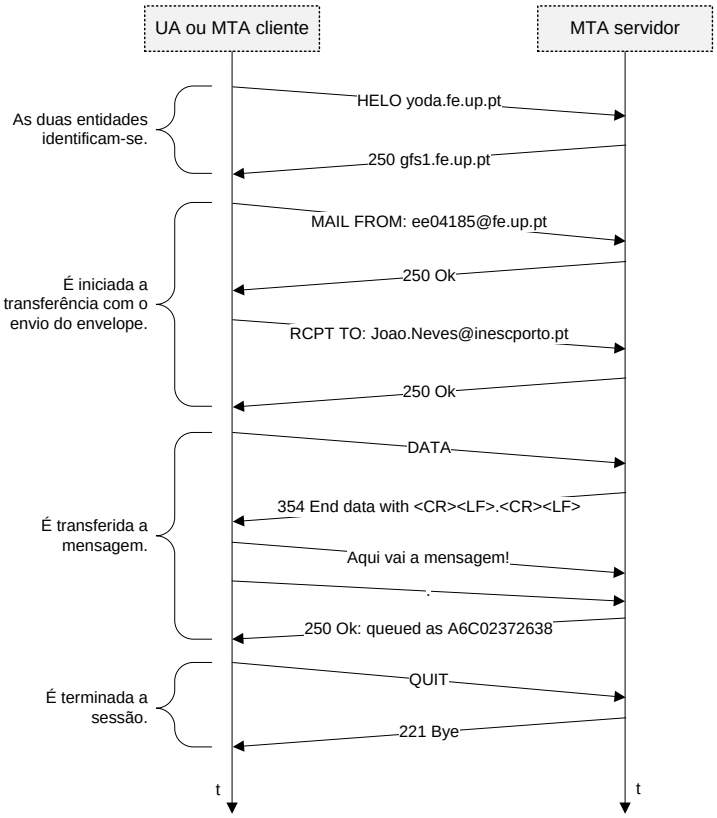

Figura 2.3: Interacção típica entre o UA e o MTA

2.3.1.3 MIME (*Multipurpose Internet Mail Extensions*)

O MIME veio essencialmente resolver o problema do envio de mensagens com caracteres especiais, representados por 8 bits e o envio de ficheiros em anexos.

A lógica consiste na conversão dessas mensagens e ficheiros na representação de 7 bits ASCII, por parte do UA. O MTA recebe a mensagem em 7 bits, tal como definido na norma. Quando esta é lida pelo UA do destinatário, é feita a conversão inversa.

2.3.2 POP3/IMAP

2.3.2.1 POP3 (*Post Office Protocol version 3*)

O protocolo POP3 permite aceder à caixa de *e-mail* que se encontra no servidor, sendo as mensagens descarregadas para o computador local. Normalmente elimina as mensagens do servidor, ficando estas apenas disponíveis no computador local.

No protocolo está definido um conjunto de comandos e respostas que permitem obter a lista das mensagens, obter o tamanho das mensagens, transferir mensagens seleccionadas, eliminar mensagens, etc. Os dados trocados no POP3, como a mensagem ou mesmo a senha do utilizador, circulam em claro, colocando graves riscos de segurança e privacidade. Em alternativa, pode-se recorrer ao POP3S (*POP3 protocol over TLS/SSL*) que usa encriptação na camada de transporte.

2.3.2.2 IMAP (*Internet Message Access Protocol*)

O IMAP é outro protocolo que também permite o acesso às caixas de *e-mail*, mas ao contrário do POP3 as mensagens ficam sempre no servidor, a não ser que sejam explicitamente eliminadas. Com este protocolo é possível que apenas sejam descarregados os cabeçalhos das mensagens de *e-mail*, não sendo necessária a recepção de todo o conteúdo.

O IMAP permite que se torne mais simples o acesso a partir de diferentes computadores e a serviços do tipo *Webmail*. Mas em contrapartida, consome mais recursos do servidor de *e-mail*, pois necessita de existir uma ligação sempre activa entre o cliente e o servidor, sendo todas operações executadas pelo servidor.

Como o POP3S, o IMAP também pode funcionar sobre canais seguros, IMAPS (*IMAP protocol over TLS/SSL*).

2.4 Tipos de Servidores

De um modo geral, podem-se identificar três tipos de servidores de *e-mail*: o MTA *relay*, o MTA local e o MTA acesso às caixas.

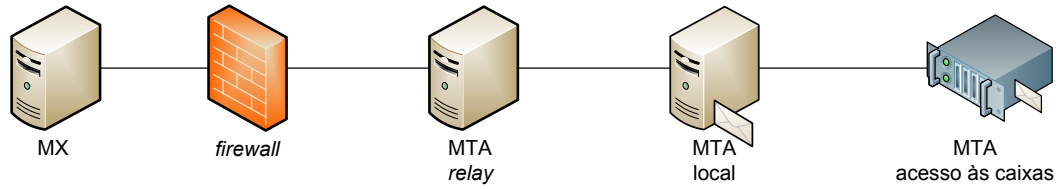

Figura 2.4: Tipos de servidores de *e-mail*

2.4.1 MTA *Relay*

A função deste tipo de servidor é basicamente fazer o encaminhamento das mensagens de *e-mail*, ou seja, receber as mensagens de um MTA ou de um UA e voltar a enviar para outro MTA.

Dentro desta categoria de MTA *relay* existem duas distinções, um que só faz a transferência de mensagens de *e-mail* exclusivamente no seu domínio e outro que faz também a transferência para outros domínios, este último na figura 2.4 está designado de MX (*Mail Exchange*).

Um MTA *relay* sem autenticação (MTA *relay* aberto) pode ser utilizado para o envio de mensagens não solicitadas (*spam*) devido ao facto de fornecerem anonimato. Quando isto acontece, no MTA *relay* consomem-se recursos. Deste modo, é importante que um MTA *relay* restrinja ao máximo os endereços IP e os domínios que podem usá-lo como *relay* ou recorrer a outros mecanismos de autenticação.

O MTA *relay* pode também funcionar como *gateway*, que tem como função a interligação de diferentes ambientes de transporte de *e-mails*. As diferenças podem ser a nível de transporte ou no formato dos *e-mails*.

2.4.2 MTA Local

Nos sistemas de *e-mail* existe pelo menos um servidor deste tipo, o MTA local, que tem como função fazer o atendimento das mensagens de *e-mail* dos utilizadores, quer para o envio como para a recepção.

2.4.3 MTA Acesso às Caixas

Este tipo de servidor é basicamente utilizado pelos UAs para o acesso à sua caixa de *e-mail* (*mailbox*). Interage aos comandos dos UAs, respondendo em conformidade a estes. Poderá eventualmente também receber as mensagens de *e-mail* para o envio.

Neste servidor normalmente a consulta das mensagens de *e-mail* é feita recorrendo ao protocolo POP3 ou ao protocolo IMAP.

2.5 Requisitos

De forma a melhor caracterizar os requisitos dos servidores de *e-mail*, dividiu-se estes em não funcionais e em funcionais.

2.5.1 Requisitos Não Funcionais

Um servidor de *e-mail* para além de satisfazer os requisitos funcionais, deve também satisfazer um conjunto de requisitos não funcionais. [5] [7] Os mais importantes são: a segurança e a fiabilidade, como descritos de seguida:

- **Segurança** - Um servidor de *e-mail* deve ser seguro contra os ataques, de modo a não comprometer o sistema. A segurança deve ser efectuada em multi-níveis. Só aos processos importantes é que devem ser atribuídos mais privilégios, os outros devem ser executados com os privilégios sempre mais baixos;

- **Fiabilidade** - Um servidor de *e-mail* deve ser fiável. Não deve perder nenhuma mensagem, mesmo em períodos de maior carga (*stress*).

Existem outros requisitos como:

- **Desempenho** - Um servidor de *e-mail* deve ser eficiente mesmo em casos de elevado fluxo de mensagens, procurando sempre aproveitar da melhor forma todos os recursos disponibilizados pelo sistema operativo;

- **Robustez** - Em caso de falta de recursos (por exemplo: memória, disco, etc.), o servidor de *e-mail* deve baixar o seu desempenho de uma forma suave. Diminuir apenas a qualidade de serviço (QoS - *Quality of Service*), mas deve garantir o funcionamento mínimo;

- **Disponibilidade** - Um servidor de *e-mail* deve estar sempre disponível, em caso de falhas devem existir servidores secundários que o possam substituir. Deve saber defender dos ataques do tipo DoS (*Denial of Service*);

- **Extensibilidade** - Um servidor de *e-mail* deve suportar os protocolos de transferência existentes, mas também deverá ser facilmente extensível para suportar novos protocolos. Do mesmo modo, fácil também de implementar novas funcionalidades;

- **Manutenção** - Um servidor de *e-mail* deve ser implementado de modo a que a sua manutenção seja facilmente realizada.

2.5.2 Requisitos Funcionais

Em termos de requisitos funcionais, são vários aos quais os servidores de *e-mail* devem satisfazer, existindo alguns que são essenciais, tais como aqueles que são descritos de seguida.[5] [8] [9]

- **Suporte ao protocolo de transporte dos *e-mails*** - Os servidores de *e-mail* no mínimo devem suportar o protocolo SMTP (versão base). Normalmente as extensões mais utilizadas são relativas à componente de segurança e privacidade.

- **Suporte ao protocolo de acesso às caixas de *e-mail*** - Os servidores de *e-mail* devem permitir aos UAs a consulta das caixas de *e-mail*, com suporte às operações básicas como: listagem, eliminar ou reenvio de mensagens. Normalmente é a implementação dos protocolos POP3/IMAP.

- **Processamento dos *e-mails*** - O processamento é composto por várias fases como:

  - Verificação dos endereços - Se os endereços respeitam a norma;
  - Reescrita dos endereços - É efectuada quando existem contas do tipo *aliases* ou outras operações pré-definidas;
  - Encaminhamento - É determinado qual o próximo salto (*next hop*). Isto é, se a mensagem atingiu o destino final, então é depositada na caixa de *e-mail*, caso contrário é enviada para o próximo servidor que é obtido da consulta do serviço de DNS;
  - Entrega - É feita ao servidor que é obtido no processo anterior.

- **Gestão das filas de espera** - O servidor de *e-mail* pode ter várias filas de espera contendo mensagens de *e-mail* a serem enviadas ou recentemente recebidas, tipicamente com diferentes prioridades.

  Em caso de ocorrência de algum erro no envio de uma mensagem, o servidor de *e-mail* deve colocar a mensagem novamente na fila de espera, para que seja reenviada depois de um determinado intervalo de tempo.

Quando se envia uma mensagem a múltiplos destinatários, nem sempre todos os servidores de *e-mail* dos destinatários estão disponíveis, sendo a mensagem colocada na fila de espera para uma entrega futura.

Existem várias métricas que um administrador deve ter em conta para a gestão das filas de espera, tais como: intervalo de tempo para o reenvio e a eliminação das mensagens, o número de retransmissões, a capacidade da fila, etc.

- **Comunicação com recursos externos** - Podem existir diversos tipos de recursos externos aos servidores de *e-mail*, tais como:
  - Base de dados - Que pode conter contas dos utilizadores e configurações de funcionamento dos servidores;
  - Serviço de directórios - Normalmente é implementado com o LDAP (*Lightweight Directory Access Protocol*), que permite armazenar as contas dos utilizadores e as suas caixas de *e-mail*, pode também ser utilizado para outro tipo de funcionalidades que não pertencem ao serviço básico de *e-mail*, tais como contactos e calendários;
  - Serviço de DNS - A sua importância já foi abordada anteriormente.
- **Garantir a segurança e a privacidade** - Por causa da simplicidade do SMTP o serviço de *e-mail* sofre dos mesmos problemas do correio postal, relacionados com a autenticação do remetente e a confidencialidade das mensagens.

2.6 Segurança

A arquitectura de segurança OSI, segundo as recomendações X.800 definidas pelo ITU-T, foca os seguintes aspectos [10]:

- **Ataque** - Acção que compromete as informações/recursos de um sistema;
- **Mecanismo** - Processo que detecta, previne ou recupera de um ataque;
- **Serviço** - Conjunto de mecanismos para atingir um determinado nível de segurança.

2.6.1 Serviços de Segurança

Os serviços de segurança importantes para os sistemas de *e-mail* são:

- **Autenticação** - A origem dos dados é quem diz ser;
- **Integridade dos dados** - Os dados recebidos são os mesmos que foram enviados por uma entidade autorizada;
- **Não repúdio** - Assegura que a fonte que enviou os dados não possa negar a sua autoria;
- **Confidencialidade** - A protecção dos dados contra o acesso não autorizado.

2.6.2 Mecanismos de Segurança

Os mecanismos de segurança que vão satisfazer alguns dos serviços de segurança anteriormente apresentados são:

- **Encriptação** - A utilização de algoritmos matemáticos para tornar ilegível a informação que é transmitida. A encriptação é obtida por exemplo com PGP, S/MIME, SASL, TLS, etc.

  A encriptação pode ser conseguida pela criptografia de chave pública:

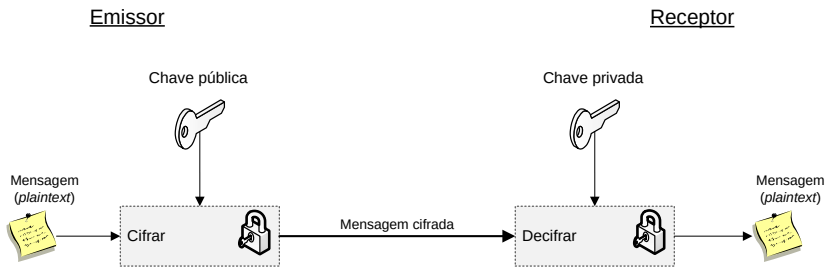

Figura 2.5: Encriptação das mensagens

  A mensagem é cifrada com a chave pública do destinatário. Sendo essa mensagem apenas decifrada com a respectiva chave privada. Como só o destinatário é que tem a posse da chave privada, apenas ele é que pode decifrar a mensagem.

- **Assinatura digital** - A assinatura é uma transformação de criptografia que é obtida dos dados e que permite à entidade receptora provar a origem e a integridade dos dados, e protege contra a falsificação. Conseguida por exemplo com PGP, S/MIME, DKIM, etc.

  O processo consiste na geração de uma assinatura com base numa função de *hash* obtida da mensagem a transmitir. Posteriormente a etiqueta resultante (resumo) é cifrada com a chave privada do remetente permitindo a confirmação por parte do destinatário da origem da mensagem (utilizando para tal, a chave pública do remetente).

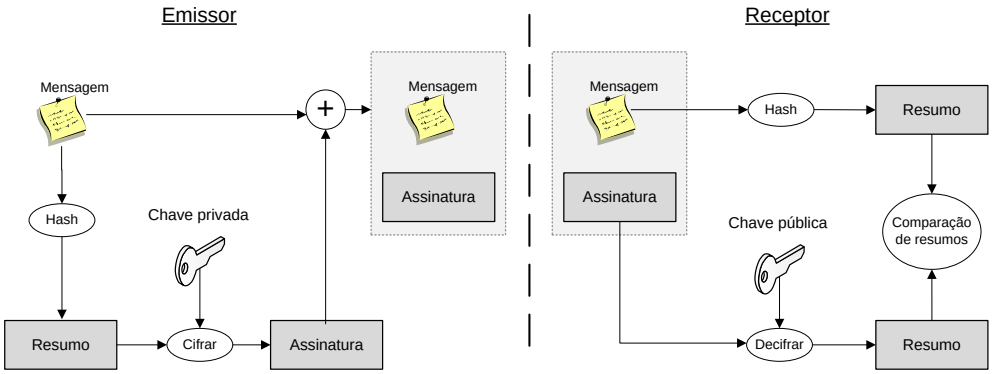

Figura 2.6: Assinatura digital

- **Integridade dos dados** - Garante que os dados não sofrem alteração desde o envio até à recepção.

- **Notário** (Entidade Certificadora) - A utilização de uma terceira entidade confiável para garantir certas propriedades dos dados trocados. Como por exemplo autoridade certificadora (CA)

- **Encaminhamento seguro** - Garantir que a comunicação não possa ser entendida por outros. Como por exemplo com a utilização do TLS ou SSL.

Na tabela 2.2 pode-se verificar quais são os mecanismos de segurança necessários para garantir os determinados serviços de segurança.

Serviços de Segurança	**Mecanismos de Segurança**
Autenticação	Encriptação e assinatura digital.
Integridade dos dados	Encriptação, assinatura digital e integridade dos dados.
Não repúdio	Assinatura digital, integridade dos dados e notário.
Confidencialidade	Encriptação e encaminhamento seguro.

Tabela 2.2: Serviços e mecanismos de segurança

2.6.2.1 Descrição dos Exemplos de Mecanismos de Segurança

SASL (*Simple Authentication and Security Layer*) tem como objectivo fornecer um mecanismo de autenticação dos clientes perante o servidor, sendo que estes clientes podem ser UAs ou outros MTAs. [5]

O SASL permite ao cliente negociar com o servidor o esquema de segurança. No caso de optarem por SMTP com segurança, primeiro é estabelecido um canal seguro TLS e depois o cliente autentica-se através das credenciais *username/password*.

De um modo geral, pode-se dizer que o SASL oferece a autenticação dos clientes e a encriptação nos dados transmitidos.

TLS (*Transport Layer Security*) permite estabelecer um canal seguro de comunicação entre o cliente e o servidor. O TLS e o seu antecessor SSL (*Secure Sockets Layer*) fornecem encriptação dos dados ponto a ponto e faz uso de certificados digitais para as autenticações. Sendo a encriptação dos dados garantida pela criptografia de chave pública.

PGP (*Pretty Good Privacy*) e S/MIME (*Secure/Multipurpose Internet Extension*) permitem assinar e cifrar a mensagem, com recurso à criptografia de chave pública. A assinatura e a mensagem cifrada são enviadas em anexo, devidamente assinaladas com o respectivo tipo de conteúdo.

O sistema de *e-mail* é um modelo do tipo *store and forward*, onde as mensagens passam de servidor em servidor (via o SMTP) até chegarem ao seu destino final, que pode acontecer em alguns minutos, horas ou até mesmo dias. A protecção dos dados ponto a ponto nestes casos não

é aplicável, pois não é possível ter um canal de comunicação sempre disponível. E é aqui que se exige que haja uma segurança extremo a extremo, conseguida pelo PGP e S/MIME.

Estes dois mecanismos são muito semelhantes, permitem obter os resultados relativamente idênticos, diferindo nas tecnologias utilizadas. A grande diferença entre PGP e S/MIME, é que no PGP a chave pública é utilizada à base de confiança, não tem nenhuma autoridade certificadora (CA - *Certification Authority*) que faça a sua autenticação, como os certificados X.509 que são utilizados pelo S/MIME. [10]

SPF (*Sender Policy Framework*) faz a autenticação do servidor de onde a mensagem é enviada, no entanto não confirma se o remetente é mesmo quem diz ser. Os servidores autorizados a enviar *e-mail* de um domínio são definidas num registo de DNS do tipo `TXT` ou `SPF`. É utilizado para combater a falsificação de endereços de retorno dos *e-mails* (`return-path`). [3]

Permite ao administrador de um domínio definir e publicar uma política SPF, onde são designados os endereços das máquinas autorizadas a enviar mensagens em nome deste domínio. Enquanto o administrador de um servidor de *e-mail* deve estabelecer critérios de aceitação de mensagens em função da verificação das políticas SPF publicadas para cada domínio.

O SPF pode ser considerado uma primeira barreira que evita a recepção completa do *e-mail* por parte do servidor, deste modo evitando o consumo de processamento.

Sender ID é um protocolo que foi derivado do SPF pela Microsoft, que visa a autenticação de um dos endereços do cabeçalho. Utiliza um algoritmo designado de PRA (*Purported Responsible Address*) definido no RFC 4407 para determinar quem é que originou e quem é responsável pela mensagem, enquanto o SPF faz a autenticação do endereço do envelope. Os registos do DNS utilizados pelo Sender ID têm uma sintaxe semelhante ao SPF. [3]

DKIM (*DomainKeys Identified Mail*) e o seu antecessor DomainKeys são também métodos de autenticação de *e-mails*. O DomainKeys (RFC 4870) desenvolvido pela Yahoo foi tornado obsoleto pelo DKIM (RFC 4871) que é uma especificação do IETF. [11]

Oferece um mecanismo de autenticação baseado em criptografia de chave pública, que garante a autenticidade conseguida com a assinatura do cabeçalho e do corpo da mensagem.

Ao contrário do que acontece no SPF e Sender ID, onde pelo seu serviço de DNS é publicado um registo do tipo `TXT` com a indicação dos servidores de *e-mail* autorizados a enviar *e-mails*, no caso do DKIM é publicada a sua chave pública.

O DKIM apresenta algumas desvantagens, como por exemplo: se a mensagem durante o transporte for ligeiramente alterada, a assinatura passa a ser inválida, podendo o servidor de destino rejeitar a mensagem; e devido às operações criptográficas que realiza, resulta numa carga computacional adicional nos servidores de *e-mail*.

2.7 Mecanismos de Filtragem

Actualmente existe uma vasta gama de mecanismos de filtragem para combater a recepção de mensagens não solicitadas e utilização abusiva dos servidores de *e-mail*. As mensagens não solicitadas, vulgarmente designadas por *spam*, são mensagens idênticas enviadas para vários destinatários e sem o prévio consentimento dos mesmos. Às vezes o volume destas mensagens é enorme que pode mesmo tornar o sistema inutilizável, sendo equivalente a um ataque do tipo DoS (*Denial of Service*). [5]

Como exemplos de mecanismos de filtragem temos:

- Filtros por assunto/conteúdo;
- Filtros anti-vírus;
- Listas negras em tempo-real (RBL: *Real-time BlackList*). Aqui por cada mensagem que o servidor de *e-mail* recebe, efectua a consulta a estas listas para verificar se o servidor remetente se encontra na lista. Normalmente a verificação é feita pelo endereço IP;
- Filtros que utilizam o teorema de *Bayes* para a classificação das mensagens;
- Filtros colaborativos. Aqui aplica-se o conceito da computação distribuída, isto é, os resultados das filtragens são partilhados por vários servidores de *e-mail*.

2.8 Casos de Estudo

Para a compreensão das funcionalidades dos servidores de *e-mail*, escolheu-se o Sendmail [8] [12], o Postfix [5] [13] e o Microsoft Exchange Server [9] para casos de estudo, por serem os mais utilizados a nível mundial [14]. As duas primeiras soluções são gratuitas e de código aberto enquanto a última é proprietária. De seguida apresenta-se uma breve descrição de cada uma.

2.8.0.2 Sendmail

O Sendmail é reconhecido pelo seu desempenho, segurança e estabilidade mas também pela sua complexidade de configuração.

Este servidor é composto por várias partes, que inclui programas, ficheiros, directórios e serviços. A lógica de funcionamento baseia-se na configuração de ficheiros que define a localização e o comportamento dessas mesmas partes.

2.8.0.3 Postfix

O Postfix é um servidor com bastante robustez, desempenho e que apresenta maior facilidade na manutenção e configuração. Além disso, Postfix é capaz de emular várias funções do Sendmail, evitando assim modificações nas aplicações que utilizam o Sendmail.

Outra característica importante do Postfix é a sua construção modular, facilitando a manutenção do código e permitindo a implementação de novas funcionalidades mais facilmente.

2.8.0.4 Microsoft Exchange Server

O Microsoft Exchange Server é um servidor de *e-mail* com software colaborativo desenvolvido pela Microsoft. Apenas é desenvolvido para plataformas da família Windows Server, não sendo possível a sua instalação noutros sistemas operativos, como a família *Unix*.

As características principais deste servidor são: entrega de mensagens, calendarização, contactos e tarefas, suporte para a mobilidade e acesso via *Web*.

Para além da sua compatibilidade com o protocolo de transporte normalizado (SMTP), este apresenta uma API (*Application Programming Interface*) para a comunicação entre os sistemas da Microsoft, designada por MAPI (*Messaging Application Programming Interface*).

2.9 Conclusão

Numa primeira fase, para minimizar os problemas de segurança, recorreu-se a sistemas de *e-mail* com autenticação SASL, a sistemas de filtragem anti-*spam* e anti-vírus e a autenticação dos *e-mails* com assinaturas digitais (conseguido por PGP e S/MIME). Actualmente a resolução tem-se verificado com a reformulação do protocolo base (SMTP), que consiste na publicação dos servidores que estão autorizados a enviar *e-mails* (com a utilização dos registos do SPF ou Sender ID), ou pela assinatura do cabeçalho e integridade dos dados (com a utilização do DKIM).

Todos os mecanismos apresentados não são 100% eficazes, resolvem apenas partes dos problemas. Sendo que todos estes mecanismos são opcionais, de modo a haver compatibilidade com o protocolo de transporte base (SMTP).

Os mecanismos de segurança podem ser implementados tanto pelos UAs ou pelos MTAs. Quando é efectua pelo UA, não existe qualquer tipo de alteração no seu transporte, deste modo esta operação é transparente para o MTA.

Capítulo 3

Gestão e SNMP

Este capítulo inicia-se com uma breve descrição sobre a gestão de redes, posteriormente é efectuada a apresentação do protocolo SNMP (*Simple Network Management Protocol*).

3.1 Gestão de Redes

Actualmente as redes têm-se tornado cada vez mais complexas, heterogéneas e exigentes. Não é possível que a sua gestão apenas seja feita com esforços/recursos humanos, mas sim passa pela criação de ferramentas que automatizem a sua gestão.

Em geral, a gestão de redes emprega o uso de diversas ferramentas e técnicas com o objectivo de garantir o bom funcionamento de uma rede, realizando uma gestão eficiente dos seus recursos. Não se limita apenas à componente de Hardware mas também permite a gestão de Software.

A ISO (*International Organization for Standardization*) definiu as seguintes áreas funcionais para organizar os requisitos de gestão das redes [15]:

- **Gestão de Falhas** - Identificação da falha, o seu isolamento e posteriormente a sua resolução;
- **Gestão da Configuração** - Actualização e manutenção das versões de software e das configurações dos recursos da rede;
- **Gestão da Contabilização** - Contabilização da utilização dos recursos da rede associados aos utilizadores. Com a possibilidade desta informação ser utilizada para a taxação pela utilização do recurso;
- **Gestão do Desempenho** - Monitorização e controlo para melhorar o desempenho da rede;
- **Gestão da Segurança** - Protecção da informação, controlo de acesso aos recursos e o registo de eventos.

3.2 Protocolo de Gestão

3.2.1 Arquitectura

Na arquitectura do modelo de gestão de redes estão presentes os seguintes elementos [16]:

- **Estação de gestão** - A estação de gestão conhecida por NMS (*Network Management Station*) é responsável por fazer *polling* e receber *traps* dos agentes da rede. A acção de *polling* consiste no envio de pedidos aos agentes para solicitar informações que estes têm e os *traps* são enviados pelos agentes para notificar a estação de gestão de eventos importantes. Esta inclui uma base de dados de informações onde estão presentes os dados necessários para a gestão dos agentes;
- **Agente** - O agente representa o equipamento gerido e deve responder aos pedidos efectuados pela estação de gestão. Pode eventualmente enviar para a estação de gestão informações de eventos importantes que não tenham sido solicitados. Recebe também da estação de gestão, pedidos para a alteração dos seus valores;
- **Informação de gestão** - Para que os equipamentos de uma rede possam ser geridos, estes são vistos como um objecto. Essencialmente, cada objecto é uma variável que caracteriza um aspecto do equipamento gerido. Sendo o conjunto destes objectos designado de MIB (*Management Information Base*). A função das MIBs é de guardar as informações dos agentes, onde as estações de gestão têm acesso a esta informação. Os objectos são normalizados para determinadas classes de equipamentos, como *routers*, impressoras, etc. Por exemplo, uma estação de gestão pode modificar alguma configuração no agente pela alteração dos valores presentes no respectivo objecto do agente;
- **Protocolo de gestão** - A comunicação entre a estação de gestão e o agente é efectuada através do protocolo de gestão, que inclui as seguintes operações:
  - *Get* - Permite à estação de gestão obter o valor dos objectos do equipamento gerido;
  - *Set* - Permite à estação de gestão definir valores nos objectos do equipamento gerido;
  - *Trap* - Permite ao agente notificar a estação de gestão de eventos importantes.

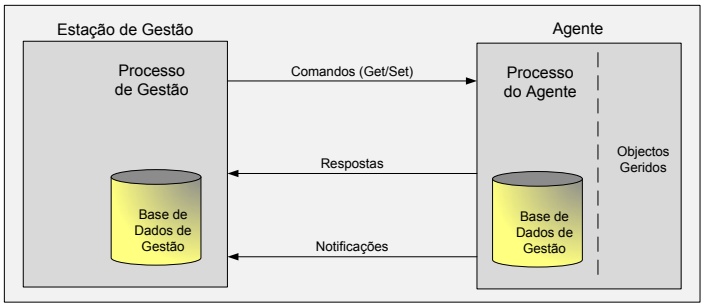

Figura 3.1: Arquitectura do protocolo de gestão

3.2.2 Informação de Gestão

Como já referido, a informação de gestão é composta por um conjunto de objectos (MIBs) que caracterizam os equipamentos da rede.

As MIBs (RFC 3418) são definidas e construídas de acordo com a estrutura SMI (*Structure of Management Information*). A estrutura SMI define os nomes aos objectos e especifica a estrutura dos dados utilizados. A definição de cada objecto inclui os seguintes atributos [15]:

- **Nome** - Identifica o objecto com um identificador, designado por OID (*Object Identifier*);
- **Tipo e Sintaxe** - Define o tipo de dados do objecto utilizando a linguagem ASN.1 (*Abstract Syntax Notation One*), que especifica como é que esses dados são representados;
- **Codificação** - O objecto é codificado com recurso ao codificador BER (*Basic Encoding Rules*). Este codificador define a forma como os objectos são codificados e descodificados de modo a serem transmitidos através do meio de transporte.

3.2.2.1 Nomes dos Objectos

Os objectos de gestão são organizados hierarquicamente numa estrutura em árvore. Cada objecto é identificado pelo seu OID, que corresponde a um nó da árvore. Os OIDs são representados por uma sequência de números separados por ponto, ou de forma a ser mais perceptível por uma sequência de nomes separados por ponto.

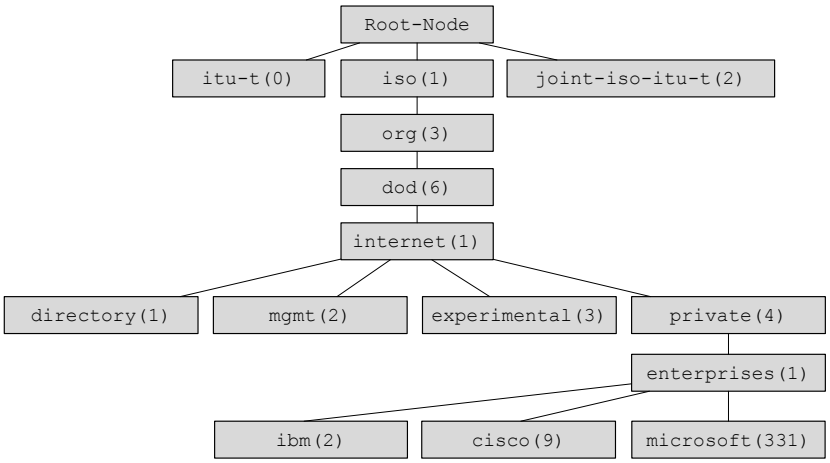

Figura 3.2: Árvore de objectos (SMIv1)

O primeiro nó que aparece na árvore é a raiz, que tem como seus nós filhos `itu-t(0)`, `iso(1)` e `joint-iso-itu-t(2)`. O ramo `itu-t(0)` é gerido pela ITU-T (*Telecommunication Standardization Sector*), o `iso(1)` é gerido pela ISO (*International Organization for Standardization*) e o `joint-iso-itu-t(2)` é gerido pelas ambas as entidades (ITU-T e ISO). O ramo que é utilizado pelo protocolo SNMP é o `iso(1)`, neste a ISO definiu o ramo das organizações `org(3)` e dentro deste está o Departamento de Defesa dos Estados Unidos da América (DoD), `dod(6)` que contém o ramo da Internet, `internet(1)`.

O ramo `internet(1)` tem como o seu OID o `1.3.6.1`, onde estão definidos os seguintes nós: o `directory(1)` reservado para uso futuro para o sistema de directórios OSI (X.500), o `mgmt(2)` define um conjunto de objectos normalizados na Internet, o `experimental(3)` reservado para testes e desenvolvimento, por último o `private(4)` que contém objectos definidos por outras organizações.

O nó `private(4)` contém apenas o ramo `enterprises(1)`, que permite aos fabricantes de Hardware e Software definirem os seus próprios objectos, que possibilita a cada fabricante organizar o seu objecto da forma que deseja.

A atribuição dos números no ramo `enterprises(1)` é feita pela IANA (*Internet Assigned Numbers Authority*). Por exemplo à Cisco foi atribuído o número 9, tal como ilustrado na figura 3.2, sendo o seu OID definido da seguinte forma:

`1.3.6.1.4.1.9` ou `iso.org.dod.internet.private.enterprises.cisco`

Com o aparecimento de uma segunda versão do SNMP, surge uma nova versão da SMI (SMIv2), que expande a árvore de objectos, adicionando novas funcionalidades às existentes. Um dos ramos adicionados nesta mudança de versão foi o `mib-2(1)`, ilustrado na figura 3.3. A MIB-II é uma actualização da MIB-I (presente na SMIv1) e é muito importante para a gestão dos equipamentos, sendo que todos os equipamentos que suportem o SNMP devem também suportar MIB-II.

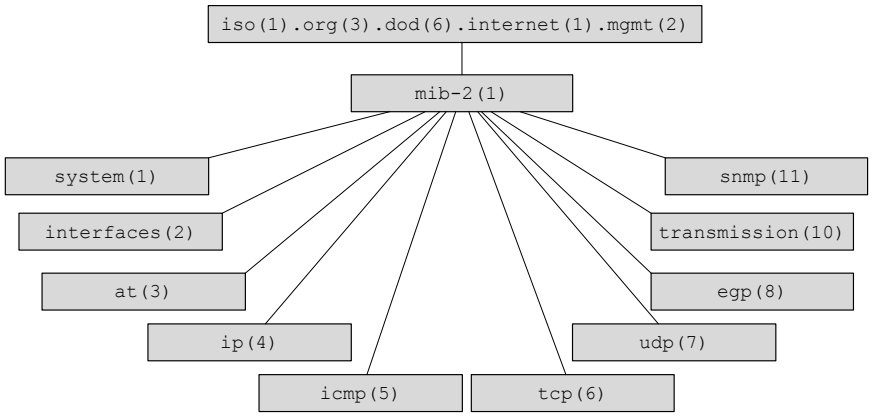

Figura 3.3: Árvore com alguns objectos da MIB-II

Os objectos presentes no ramo `mib-2(1)` são:

- `system(1)` - Define uma lista de objectos pertencentes à operação de um sistema, como o tempo de funcionamento, contacto e nome do sistema;
- `interfaces(2)` - Permite guardar as informações das interfaces de rede, tais como o seu estado, o número de pacotes enviados e recebidos, a taxa de transmissão, entre outros;
- `at(3)` - O *address translation* é utilizado para fazer a tradução do endereço IP em endereço físico;
- `ip(4)` - Contém estatísticas relativas aos pacotes IP, incluindo a tabela de *routing*;

- `icmp(5)` - Contém estatísticas sobre as mensagens ICMP (*Internet Control Message Protocol*);
- `tcp(6)` - Informações sobre os algoritmos, parâmetros e estatísticas do protocolo TCP;
- `udp(7)` - Inclui estatísticas do tráfego UDP (*User Datagram Protocol*);
- `egp(8)` - Inclui estatísticas do tráfego EGP (*Exterior Gateway Protocol*);
- `transmission(10)` - Informação relativa aos meios de transmissão;
- `snmp(11)` - Inclui estatísticas do tráfego SNMP.

3.2.2.2 Tipos de Dados dos Objectos

Tipos de Dados Base

Os tipos de dados base definidos nas duas versões de SMI são [15] [17]:

Tipo de dado	Descrição
`INTEGER` (SMIv1, SMIv2) `Integer32` (SMIv2) `Unsigned32` (SMIv2)	O `INTEGER` é inteiro de 32 bits, que na versão SMIv1 é utilizado para representar inteiros com sinal, sem sinal e enumerados. Na segunda versão (SMIv2) apenas é utilizado para tipos enumerados, pois nesta versão são definidos mais dois tipos novos: `Integer32` e `Unsigned32`. Exemplo de um tipo enumerado: `INTEGER{up(1) down(2) testing(3)}`. Por vezes, apenas é permitido um conjunto limitado de valores. Um inteiro limitado a 8 bits seria: `INTEGER (0..255)`.
`Counter` (SMIv1) `Counter32` (SMIv2) `Counter64` (SMIv2)	São todos contadores sem sinal. Os dois primeiros são de 32 bits e o último de 64 bits. Sendo que o de 64 bits deve-se evitar a sua utilização. Num contador, quando se atinge o valor máximo, o valor da contagem volta novamente a zero. Quando o agente gerido é reiniciado o valor inicial do contador deve ser zero. É utilizado para contagem por exemplo do número de pacotes enviados ou recebidos por uma *interface* de rede, do número de mensagens recebidas por um servidor de *e-mail*, etc.
`Gauge` (SMIv1) `Gauge32` (SMIv2)	Ambos são inteiros de 32 bits sem sinal. Ao contrário do que se verificou nos `Counter`s, aqui o valor pode aumentar ou diminuir. É utilizado por exemplo para a monitorização do número de processos actualmente em execução num sistema.
`TimeTicks` (SMIv1, SMIv2)	É um número inteiro de 32 bits sem sinal, utilizado para quantificar o tempo em centésimos de segundo.
`OCTET STRING` (SMIv1, SMIv2)	Define uma *string* de zero ou mais octetos. Geralmente utilizada para representar expressões textuais, como também para representar endereços físicos. Sendo muitas vezes utilizada para definição de convenções textuais. A condição `SIZE` permite limitar o número de octetos usados. Como por exemplo, zero a 255 octetos: `OCTET STRING SIZE(0..255)`; exactamente 8 octetos: `OCTET STRING SIZE(8)`, exactamente 8 ou exactamente 11 octetos: `OCTET STRING SIZE(8|11)`.
`OBJECT IDENTIFIER` (SMIv1, SMIv2)	Define uma *string* de números separados por ponto que representam um objecto na árvore de gestão. Por exemplo, 1.3.6.1.4.1.9 representa o OID da Cisco.
`IpAddress` (SMIv1, SMIv2)	Representa um endereço de rede IPv4 (32 bits).
`NetworkAddress` (SMIv1)	Semelhante ao anterior, mas permite representar outros tipos de endereços de rede.
`Opaque` (SMIv1, SMIv2)	Permite outro tipo de codificação ASN.1 numa `OCTET STRING`, mas raramente é utilizado.
`BITS` (SMIv2)	É baseado no tipo `OCTET STRING` e é utilizado para enumeração de *named* bits.
`SEQUENCE` (SMIv1, SMIv2)	Define uma lista que contém zero ou mais tipos de dados ASN.1.
`SEQUENCE OF` (SMIv1, SMIv2)	Define um objecto que é construído com elementos do tipo `SEQUENCE`. Tanto este como o tipo anterior são utilizados para a construção de tabelas.

Tabela 3.1: Tipos de dados base

Tipos de Dados Derivados

A versão SMIv2 permite definir convenções textuais, que são tipos de dados obtidos a partir dos tipos anteriormente descritos. De seguida apresentam-se algumas convenções textuais que são mais utilizadas. [17]

Tipo de dado	Descrição
`DisplayString`	É normalmente usado para dados textuais. Deve conter apenas caracteres NVT (*Network Virtual Terminal*) ASCII.
`TruthValue`	Valor booleano. *true(1), false(2).*
`PhysAddress`	Endereço físico que é representado como uma `OCTET STRING`.
`MacAddress`	Endereço MAC que é representado em seis octetos.
`TestAndIncr`	Assegura que duas estações de gestão não modifiquem o mesmo objecto ao mesmo tempo.
`AutonomousType`	Um OID que permite definir uma sub-árvore de uma MIB.
`VariablePointer`	É um apontador para um objecto.
`RowPointer`	É um apontador para uma linha de uma tabela.
`RowStatus`	Permite adicionar ou eliminar linhas numa tabela.
`TimeStamp`	Intervalo de tempo entre a inicialização do sistema e o instante em que é gerado um evento.
`TimeInterval`	Intervalo de tempo em centésimos de segundo.
`DateAndTime`	Uma `OCTET STRING` utilizada para representar informação sobre a data e a hora.
`StorageType`	O tipo de memória utilizada pelo agente.
`TDomain`	Descreve o tipo de serviço de transporte.
`Taddress`	Define o endereço do serviço de transporte.

Tabela 3.2: Tipos de dados derivados

3.2.2.3 Definição dos Objectos

A definição dos objectos é feita com recurso às macros, que são mecanismos que permitem a extensão da linguagem ASN.1. O SNMP utiliza este mecanismo para fornecer informações completas sobre os objectos, como o nome do objecto, o tipo de dados, o tipo de acesso, o seu estado e uma descrição informal. [15]

A SMI possui a macro `OBJECT-TYPE` que permite definir um objecto gerido. Adicionalmente a versão SMIv2 permite incluir mais campos, estes estão assinalados a negrito na figura 3.4.

SMIv1

```
<Nome do objecto> OBJECT-TYPE
   SYNTAX <Tipo de dados>
   ACCESS <read-only|read-write|write-only|not-accessible>
   STATUS <mandatory|optional|deprecated|obsolete>
   DESCRIPTION
       "Descrição textual do objecto."
   ::= { <Um OID único que define o este objecto> }
```

SMIv2

```
<Nome do objecto> OBJECT-TYPE
   SYNTAX <Tipo de dados>
   MAX-ACCESS <read-only|read-write|read-create|not-accessible|accessible-for-notify>
   STATUS <current|obsolete|deprecated>
   UNITS <As unidades do objecto em causa>
   REFERENCE "Referência a um documento"
   DEFVAL { <Valor padrão> }
   DESCRIPTION
       "Descrição textual do objecto."
   ::= { <Um OID único que define o este objecto> }
```

Figura 3.4: Definição dos objectos

Na versão SMIv2, o campo `ACCESS` foi modificado para `MAX-ACCESS` e os outros campos adicionados são facultativos.

3.2.2.4 Estrutura de uma MIB

Geralmente as MIBs estão organizadas da maneira descrita na figura 3.5. Existindo algumas diferenças nas duas versões. [17]

MIB (SMIv1)	MIB (SMIv2)	Descrição
`MY-MIB-NAME DEFINITIONS ::= BEGIN`	`MY-MIB-NAME DEFINITIONS ::= BEGIN`	Nome da MIB
`IMPORTS syntax`	`IMPORTS syntax`	Importa definições e objectos de outras MIBs
	`MODULE-IDENTITY statement`	Informação documental e permite também definir um objecto
`Data sub-typing definitions`	`TEXTUAL-CONVENTION definitions`	Definição de dados derivados
`Node definitions`	`Node definitions`	Definição dos nós
`Scalar Data Objects`	`Scalar Data Objects`	Definição de objectos escalares

`Table Object`
`Entry Object`
`Table SEQUENCE statement`
`Table Data Objects`
	`Table Object`
	`Entry Object`
	`Table SEQUENCE statement`
	`Table Data Objects`
		Definição de tabelas

`TRAP-TYPE Objects`	`NOTIFICATION-TYPE Objects`	Definição dos eventos/notificações

	`OBJECT-GROUP lists`
	`NOTIFICATION-GROUP lists`
	`MODULE-COMPLIANCE grouping lists`
		Definição dos objectos de conformidade

`END`	`END`	Fim

Figura 3.5: Estrutura de uma MIB

As MIBs construídas na versão SMIv2 são mais fáceis de ler do que as definidas na versão anterior, pois fornecem mais informações

A conversão das MIBs definidas em diferentes versões de SMI é possível se forem tomadas algumas precauções:

- É quase sempre possível converter SMIv1 em SMIv2, a menos de algumas alterações na sua sintaxe.
- A conversão da MIB de SMIv2 para SMIv1, é possível se forem seguidas algumas recomendações:
  - Não utilizar o tipo de dados `Counter64`, pois este tipo não é traduzido na versão SMIv1;
  - O tipo `NOTIFICATION-TYPE` deve ser definido de modo a ser possível a sua tradução em `TRAP-TYPE`;
  - Ter atenção aos nomes dos objectos e ao tipo enumerados.

As MIBs devem ser construídas de acordo com a estrutura SMIv2, só em casos de necessidade de compatibilidade com o agente é que se deve usar a versão SMIv1.

3.2.3 Protocolo SNMP

O SNMP - *Simple Network Management Protocol* [18] é um protocolo de gestão que foi desenvolvido pelo IETF (*Internet Engineering Task Force*) com o objectivo de tornar mais simples e prática a gestão das redes. É um protocolo que está situado no nível da aplicação do modelo de camadas TCP/IP. A troca de dados (datagramas) entre a estação de gestão e o agente é efectuada através do protocolo de transporte UDP (*User Datagram Protocol*). O porto UDP definido para os agentes foi o 161 e o das estações de gestão para a recepção das notificações o 162.

Este protocolo de transporte não garante a entrega dos datagramas nem oferece nenhum mecanismo de retransmissão em caso de erros, o que até pode ser benéfico pois não gera tráfego adicional e desse modo não perturba muito o funcionamento das redes geridas.

O controlo de erros e retransmissão pode ser feito na camada de aplicação, neste caso pelo SNMP, que pode implementar um mecanismo de retransmissão após um *timeout*. Mas mesmo assim não garante a entrega das mensagens do tipo *trap* (notificações), pois estas não são confirmadas.

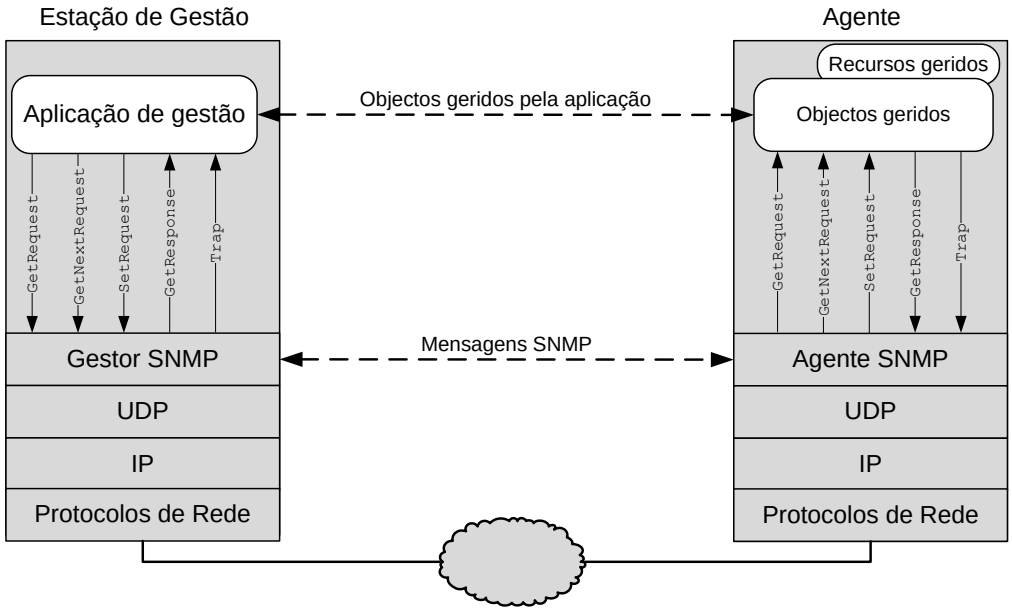

Figura 3.6: Modelo de camadas TCP/IP e SNMP

Actualmente a maioria dos equipamentos é capaz de ser gerida através deste protocolo, tendo mesmo suporte por parte de alguns dos sistemas proprietários.

Ao longo dos tempos foram introduzidas melhorias na especificação base do SNMP, levando ao aparecimento de três versões: SNMPv1, SNMPv2 e SNMPv3.

Nas versões SNMPv1 e SNMPv2 a autenticação entre as estações de gestão e os agentes está associada a um conceito de comunidade. Os nomes de comunidades (*community string*) são como se fossem *passwords* para o acesso aos objectos do agente e para o envio de notificações autenticadas à estação de gestão. O acesso aos objectos do agente pode ser configurado de dois modos diferentes: um que permite apenas efectuar operações de leitura (*read-only*) e outro que

permite o acesso de leitura e escrita (*read-write*). Para cada um destes modos de acesso existe uma *community string*.

Mesmo havendo este mecanismo de autenticação, esta *community string* é enviada em claro pela rede o que compromete dos dados dos agentes geridos.

3.2.3.1 SNMPv1

O SNMPv1 é a primeira versão do protocolo e inclui as seguintes operações:

- `GetRequest` - A estação de gestão solicita o valor de um determinado objecto ao agente. Numa mensagem `GetRequest` pode-se pedir o valor de múltiplos objectos;
- `GetNextRequest` - É semelhante à operação anterior, mas aqui é devolvido o OID e o valor do objecto seguinte;
- `SetRequest` - Permite que a estação de gestão possa modificar valores dos objectos presentes nos agentes, sendo que esses objectos têm de ter permissões de leitura e escrita (*read/write*). Tal como o `GetRequest`, uma mensagem pode ter múltiplos OIDs/valores;
- `GetResponse` - Enviado pelo agente para dar resposta à operação `GetRequest` e `GetNextRequest` ou para confirmar a operação `SetResponse`;
- `Trap` - Operação efectuada pelo agente gerido, enviando para a estação de gestão informação de ocorrências importantes.

Mensagens de Erros - SNMPv1

As mensagens de erros definidas na versão SNMPv1 estão presentes na tabela 3.3 e estas são enviadas pelo agente à estação de gestão em casos de ocorrência de erros.

Mensagem	Descrição
`noError(0)`	Não existe nenhum tipo de erro.
`tooBig(1)`	Não é possível enviar a resposta numa só mensagem.
`noSuchName(2)`	O OID não existe.
`badValue(3)`	O valor do OID não é consistente.
`readOnly(4)`	Raramente utilizado e é equivalente ao erro `noSuchName(2)`.
`genErr(5)`	Quando nenhum dos erros anteriores se aplica, então é enviado este que é um erro genérico.

Tabela 3.3: Mensagens de erros - SNMPv1

Traps Genéricos - SNMPv1

Os *traps* genéricos definidos na versão SNMPv1 são:

Mensagem	Descrição
`coldStart(0)`	É gerado quando o agente é reiniciado. Nesta situação todos os objectos de gestão presentes no agente serão também reinicializados.
`warmStart(1)`	Semelhante ao anterior, mas aqui nenhum objecto de gestão é inicializado.
`linkDown(2)`	Enviado quando a interface de rede passa para o estado inactivo.
`linkUp(3)`	Enviado quando a interface de rede passa para o estado activo.
`authenticationFailure(4)`	Indica que alguém está a tentar autenticar-se com uma *Community String* incorrecta.
`egpNeighborLoss(5)`	É gerado quando no protocolo EGP (Exterior Gateway Protocol) existe a perda de um vizinho.
`enterpriseSpecific(6)`	Indicação de que o *trap* é específico de uma MIB privada.

Tabela 3.4: *Traps* genéricos - SNMPv1

A versão SNMPv1 apresenta algumas limitações, como por exemplo a autenticação através da *community string* ser trivial e não garantir segurança, as mensagens SNMP são transportadas apenas em UDP, não é possível comunicação entre estações de gestão (apenas estação de gestão e o agente), nem é adequado para a transferência de grandes quantidades de dados e as mensagens *trap* não são confirmadas.

3.2.3.2 SNMPv2

A versão SNMPv2 teve como objectivo de incluir melhoramentos na versão SNMPv1. Entre as melhorias introduzidas pode-se destacar: a adição de novos tipos de dados, erros e macros; a extensão da árvore de OIDs; as operações que facilitem a transferência de grandes quantidades de dados; e são introduzidas convenções textuais, que permitem a leitura mais inteligível para os humanos de um módulo de uma MIB.

O SNMPv2 adiciona à versão anterior as seguintes operações:

- `GetBulkRequest` - Equivalente à operação `GetRequest`, sendo que agora é possível a transferência de grandes quantidades de dados. Ideal para a transferência de tabelas;
- `InformRequest` - Permite a comunicação entre as estações de gestão;
- `SNMPv2-Trap` - Semelhante ao `Trap` da versão SNMPv1, difere no formato do PDU *(Protocol Data Unit)*;
- `Report` - Previsto mas nunca implementado.

Mensagens de Erros - SNMPv2

As mensagens de erros definidas na versão SNMPv2 são:

Mensagem	Descrição
`noAccess(6)`	É enviada quando a alteração do valor do objecto não é permitida.
`wrongType(7)`	O valor do objecto que se está a modificar é diferente do tipo que é definido.
`wrongLength(8)`	O valor do objecto que se está a modificar não respeita as restrições de limite.
`wrongEncoding(9)`	A codificação do valor do objecto está incorrecta.
`wrongValue(10)`	O valor do objecto a modificar está incorrecto.
`noCreation(11)`	Tentativa de criação/alteração de um objecto que não existe na MIB.
`inconsistentValue(12)`	O objecto encontra-se num estado inconsistente, não aceita pedidos de modificação do seu valor.
`resourceUnavailable(13)`	Não é possível a alteração do valor do objecto, devido à falta de recursos.
`commitFailed(14)`	Quando nenhum dos erros/falhas anteriores se aplica, então é enviado este que é um erro genérico.
`undoFailed(15)`	A alteração do valor do objecto falhou e o agente não consegue colocar o valor que tinha antes.
`authorizationError(16)`	Operação não autorizada, normalmente deve-se à *community string* não estar correcta.
`notWritable(17)`	O objecto não permite a alteração do seu valor.
`inconsistentName(18)`	O objecto encontra-se num estado inconsistente, não é possível a modificação do seu valor.

Tabela 3.5: Mensagens de erros - SNMPv2

3.2.3.3 SNMPv3

A terceira versão do SNMP é uma extensão das anteriores versões na área de administração e na de segurança. Tem como objectivos ter uma arquitectura modular que possibilita fácil expansão e a interligação com outros protocolos, bem como resolução do problema de segurança identificado nas versões anteriores.

O SNMPv3 em relação ao protocolo mantém as mesmas operações existentes no SNMPv2. No que diz respeito às entidades, onde antes se tinha a estação de gestão e o agente, agora nesta nova versão apenas se tem uma entidade composta por duas peças, um motor SNMP (SNMP *Engine*) e uma ou mais aplicações SNMP (SNMP *Applications*).

3.2.3.4 Motor SNMP

A função do motor SNMP é a de enviar, receber, autenticar e encriptar as mensagens e controlar o acesso aos objectos geridos.

O motor SNMP apresenta os seguintes componentes:

- *Dispatcher* - Determina a versão do protocolo SNMP de cada mensagem recebida, se suportar essa versão então envia a mensagem para o respectivo modelo de processamento de mensagem *(Message Processing Subsystem)*;
- *Message Processing Subsystem* - É responsável pela preparação das mensagens a serem enviadas e pela extracção dos dados das mensagens recebidas;
- *Security Subsystem* - É responsável pela autenticação e a encriptação das mensagens;
- *Access Control Subsystem* - É responsável pelo controlo de acesso a um objecto gerido.

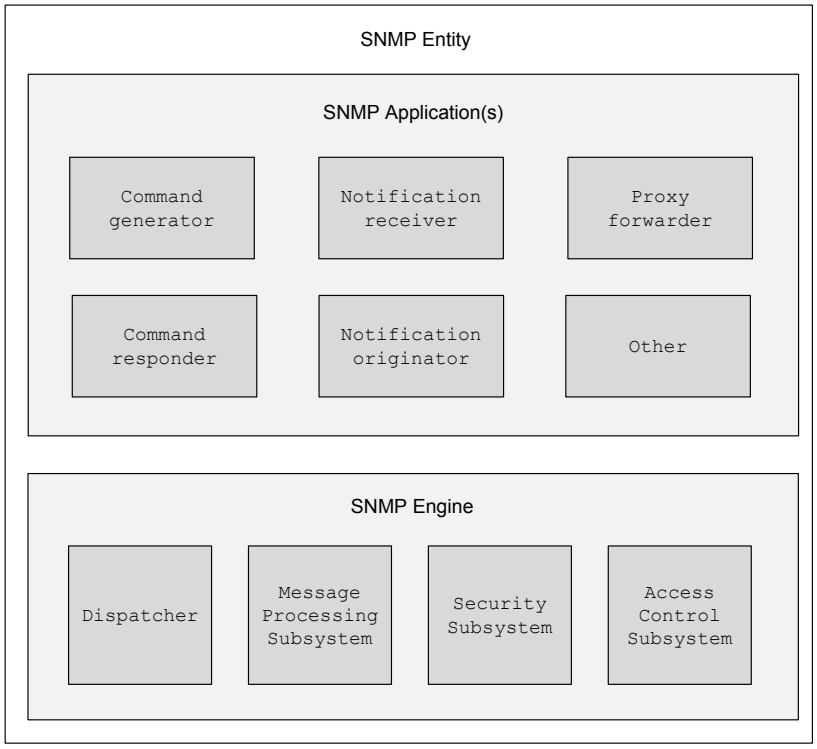

Figura 3.7: Entidade SNMPv3

3.2.3.5 Aplicações SNMP

As aplicações SNMP geram e respondem às mensagens SNMP, geram e recebem as notificações, e encaminham mensagens entre as entidades SNMP. Existem as seguintes aplicações:

- *Command generator* - Gera os pedidos (`GetRequest`, `GetNextRequest`, `GetBulkRequest` e `SetRequest`) e processa as respostas dos pedidos que faz;
- *Command responder* - Responde aos pedidos (`GetRequest`, `GetNextRequest`, `GetBulkRequest` e `SetRequest`);
- *Notification originator* - Gera mensagens de `Trap`;
- *Notification receiver* - Recebe as mensagens `Trap` e `inform`;
- *Proxy forwarder* - Encaminha as mensagens entre entidades.

3.3 Conclusão

Em relação às várias versões do SNMP pode-se dizer que SNMPv1 é a versão base, a versão SNMPv2 fornece mais funcionalidades e é mais eficiente do que a versão anterior, e a versão SNMPv3 adiciona mecanismos de segurança.

Capítulo 4

Caracterização do Problema e o Estado da Arte

Neste capítulo é construído um modelo genérico de um servidor de *e-mail* para facilitar a identificação dos principais problemas. É ainda incluído um tópico sobre as soluções que resolvem alguns dos problemas identificados, bem como uma análise comparativa entre algumas das plataformas de gestão existentes.

4.1 Modelização de um Servidor de *E-mail*

Um servidor de *e-mail*, relativamente à arquitectura de software pode seguir uma das seguintes metodologias [7]:

- Sistema monolítico, que tipicamente utiliza um único programa no atendimento das mensagens de *e-mail*. O Sendmail foi um dos servidores que era composto deste modo;
- Programas mais pequenos que executam funções ou tarefas específicas. Muitos destes programas são *daemons*, processos em *background* que se executam indefinidamente. Pode-se ter um processo principal, normalmente designado por mestre que controla os outros processos. Por exemplo esta é a metodologia que é adoptada pelo Postfix. É com base neste tipo de arquitectura que é abordado o modelo apresentado de seguida.

Do ponto de vista funcional, um servidor de *e-mail* mais simples deverá ter três grandes módulos: um que é responsável por receber os *e-mails*, um que faz o processamento e um que é responsável pela entrega dos *e-mails*, tal como se pode visualizar na figura 4.1. O modelo apresentado tem como base os servidores de escolhidos como casos de estudo.

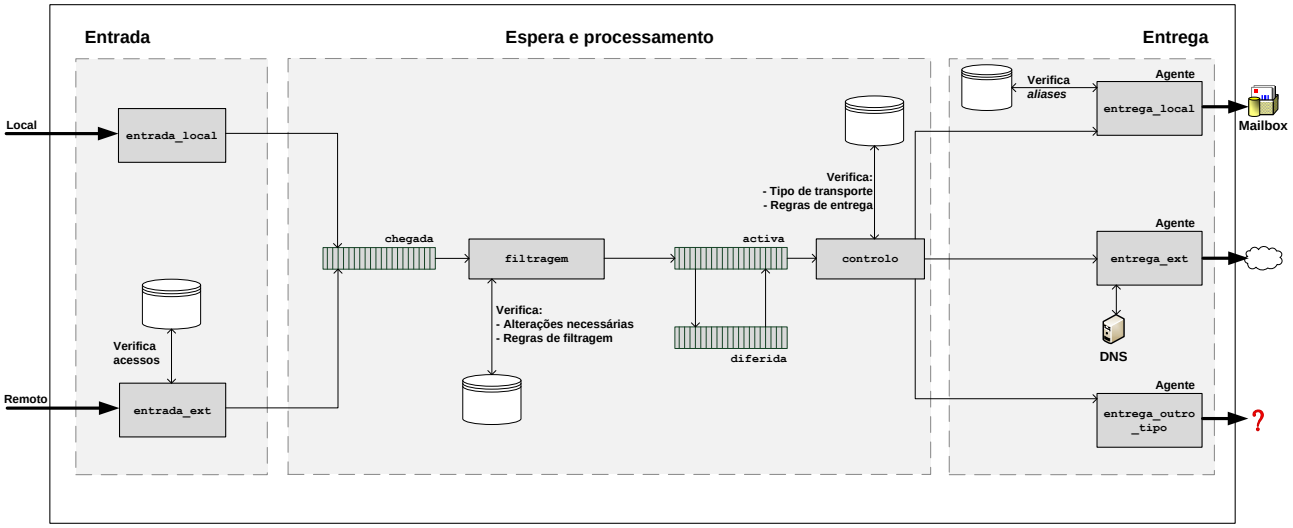

Figura 4.1: Modelo de um servidor de *e-mail*

4.1.1 Entrada dos *E-mails*

As mensagens de *e-mail* podem entrar no servidor de uma das seguintes maneiras:

- A mensagem pode ser aceite localmente, recebida do próprio servidor;
- A mensagem é recebida de domínios autorizados;
- O servidor de *e-mail* gera mensagens quando envia notificações de *e-mails* não entregues (*undeliverable*), ou quando existe adiamento na entrega (*deferred delivery attempts*).

Neste módulo estão presentes dois processos, o `entrada_local` e o `entrada_ext`. O `entrada_local` recebe as mensagens com origem no próprio servidor. Enquanto `entrada_ext` recebe as mensagens de outros servidores, normalmente via o protocolo de transporte SMTP.

Tanto processo `entrada_local` como o `entrada_ext`, colocam as mensagens na fila de espera para que sejam atendidas por outro processo. O `entrada_ext` antes de colocar a mensagem na fila, verifica se a origem é um domínio ou um remetente autorizado (controlo de acessos).

4.1.2 Espera e Processamento

Neste módulo as mensagens são colocadas em filas de espera e é efectuado o seu processamento. O gestor de filas de espera é um elemento crucial nos servidores de *e-mail*, pois é este o responsável pelo escalonamento das mensagens.

Alguns servidores de *e-mail* têm como requisito que todas as mensagens que entram ou saem do sistema sejam colocadas nas filas de espera. Cada fila indica também o estado de processamento da mensagem.

Os processos que estão neste módulo são:

- `filtragem` - É activado quando existem mensagens novas na fila de `chegada`. Responsável pela filtragem dos *e-mails* em vários níveis:
  - Processa os cabeçalhos dos *e-mails* e faz a formatação dos mesmos;
  - Verifica o formato dos cabeçalhos, se estão de acordo com o formato definido. Adiciona alguns campos como `From`, `Message_ID` e `Date`;
  - Pode fazer a reescrita dos endereços, em função dos valores pré-definidos de mapeamento (presentes em tabelas ou base de dados).

  Depois deste pré-processamento a mensagem é colocada na fila `activa`.

- `controlo` - Responsável pela gestão das filas de espera e pela distribuição das mensagens aos diversos agentes de entrega. Este espera pela chegada de novas mensagens na fila `activa` e depois efectua a sua distribuição. O gestor de filas de espera é responsável pelo escalonamento das filas, que garante uma distribuição rápida e justa da entrega dos *e-mails* para todos os destinatários dentro dos recursos que dispõe.

Entre o processo `filtragem` e `controlo` podem existir várias filas de espera, no modelo apresentado tem-se apenas duas filas.

No modelo apresentado estão presentes três filas de espera. A fila de `chegada` recebe as mensagens dos processos de entrada. Na fila `activa` as mensagens estão a ser processadas pelo `controlo` para fazer a entrega ao agente. Na fila `diferida` são colocadas as mensagens, em que a sua entrega não foi possível devido a falhas temporárias, as quais devem ser novamente submetidas depois de um tempo pré-definido ou submetidas "manualmente" (a pedido do utilizador).

4.1.3 Entrega dos *E-mails*

A entrega pode ser atendida por três processos diferentes: `entrega_local`, `entrega_ext` e `entrega_outro_tipo`.

É entregue ao processo de `entrega_local`, caso a *mailbox* do destinatário se encontre no mesmo servidor, aqui a mensagem é depositada na respectiva *mailbox*. Mas antes de proceder à entrega, este verifica se existem *aliases* para o utilizador em questão, em caso afirmativo, a mensagem é enviada para o respectivo endereço.

É atendida pelo processo de `entrega_ext`, se a entrega tiver como destino outro servidor, neste caso o servidor em questão passa a desempenhar a função de cliente.

Pode ainda ser atendida pelo processo `entrega_outro_tipo`, que permite fazer a entrega a outros tipos de transportes ou a programas, como por exemplo o LMTP (*Local Mail Transfer Protocol*). O LMTP, permite a transferência das mensagens de *e-mail* de um serviço de correio local para outro, sem a necessidade de existir uma unidade de armazenamento em comum. Assim permite a entrega a unidades de armazenamento não normalizadas. Baseado e uma versão simplificada do protocolo SMTP.

4.2 Identificação dos Principais Problemas

4.2.1 Pontos Críticos

Um servidor de *e-mail* deve satisfazer todos os seus requisitos funcionais, de modo a garantir os requisitos não funcionais e minimizar os pontos críticos que interferem no seu bom funcionamento. Na figura 4.2 estão presentes alguns dos principais pontos críticos que se podem identificar no modelo de servidor de *e-mail* anteriormente visto.

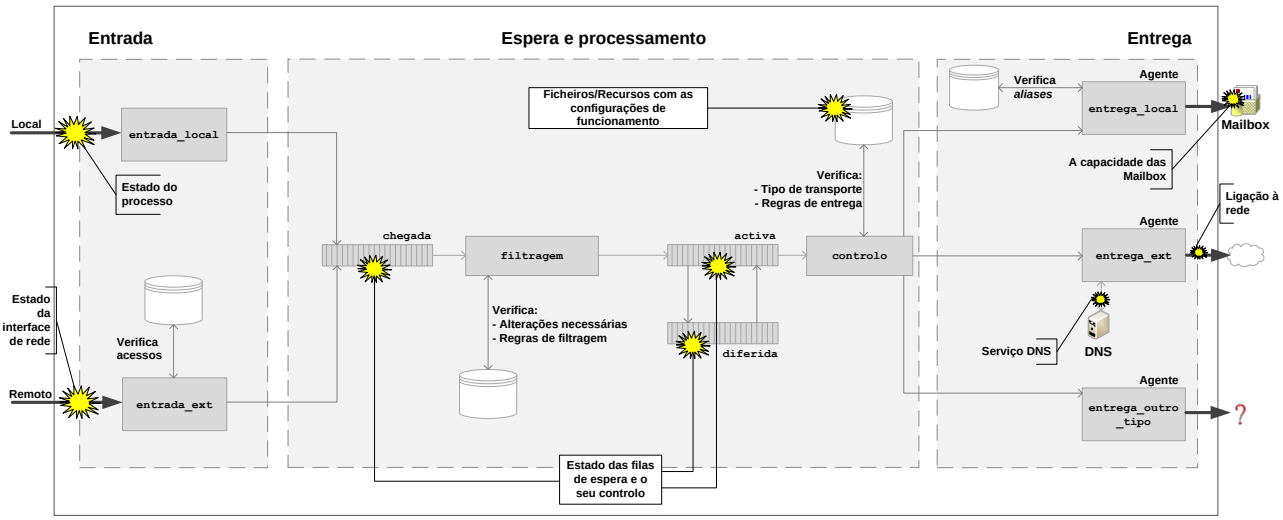

Figura 4.2: Principais pontos críticos

Podemos classificar estes pontos críticos como factores directos e indirectos do desempenho:

- **Factores directos** - São aqueles que têm origem no servidor de *e-mail*, portanto por todos estes pontos o servidor é responsável pelos mesmos;
- **Factores indirectos** - Estes são aqueles em que o servidor de *e-mail* não tem controlo sobre os mesmos. Normalmente relacionados com o sistema operativo e a rede.

4.2.1.1 Factores Directos

Filas de espera - As filas vão garantir alguns dos requisitos não funcionais como a fiabilidade, desempenho e robustez. O escalonamento das mensagens é em parte influenciado pelo funcionamento e o estado das filas. Deve-se monitorizar: o estado das filas, a sua capacidade, bem como a taxa de entrada e saída das mensagens.

Criação de novos processos - Por exemplo para o atendimento de novos *e-mails*. Pode-se determinar se o servidor está congestionado. Normalmente, os servidores têm um limite de criação de novos processos. Aqui deve ser monitorizado: o estado dos processos, o número de processos e sub-processos activos, tempo de execução dos processos, etc.

Configurações de funcionamento - As configurações, por exemplo podem ser: as permissões de acesso aos recursos do servidor, as configurações dos vários filtros, etc.

Armazenamento dos *e-mails* - Depende do formato das caixas de *e-mail* (*mbox*, *maildir*, *mbx* ou outro formato). Deve-se ter em atenção a capacidade das caixas de *e-mail* e aos resultados da escrita/leitura dos *e-mails*.

Actualização do envelope - Deve-se monitorizar se existem problemas na actualização do envelope dos *e-mails*.

4.2.1.2 Factores Indirectos

Processos e recursos externos - Todos os processos e recursos que poderão interagir com o servidor de *e-mail*, como por exemplo filtros de *spam*. Verificar o estado de execução dos processos, o número de processos activos, o tempo de execução dos processos, etc. Deve-se monitorizar alguns dos recursos disponibilizados por terceiros, ou pelo menos verificar que existe a conectividade com estes.

Partilha de recursos - É necessário ter em atenção alguns dos recursos partilhados por parte dos processos, para evitar problemas de concorrência no seu acesso. Em princípio o sistema operativo deve ser capaz de efectuar o escalonamento de recursos partilhados.

Interfaces de rede - As interfaces de rede devem ser monitorizadas, pois estas permitem a ligação com a rede.

Contas dos utilizadores - Contas que podem ser criadas e mantidas pelo sistema operativo ou por outro tipo de recuso externo. Deve-se monitorizar o número total destas contas e os erros ocorridos na entrega de mensagens a estas contas.

Outro tipo de transporte - Caso existam outros tipos de transporte ou outras interfaces de entrega, estas também devem ser monitorizadas.

4.2.2 Gestão

A monitorização permite ter uma visão do funcionamento do sistema. Por sua vez, a recolha dos dados permite gerar informações estatísticas que podem ser utilizadas para realizar a optimização do sistema que é conseguida pela gestão.

4.2.2.1 Actuação Sobre Alguns dos Pontos Críticos

Aos pontos críticos que foram analisados, podemos aplicar algumas operações de gestão. Nas filas de espera, por exemplo pode-se mudar as mensagens de uma fila para a outra, como também listar as mensagens presentes numa determinada fila, ou ainda efectuar uma nova tentativa de envio, devido à existência de um problema temporário. Ter o controlo do estado do servidor,

como dos principais processos que estão em execução. Pode-se também fazer a gestão de algumas configurações do sistema de filtragem.

4.2.2.2 Gestão das Configurações

A gestão das configurações tem como objectivo a modificação das configurações de funcionamento dos servidores de *e-mail* em função dos resultados obtidos pela monitorização. Aqui deve-se apenas incluir as configurações importantes, pois a maioria dos servidores de *e-mail* têm já ferramentas que permitem a sua configuração via *Web*.

4.3 Soluções Existentes

A gestão de redes possui diversas soluções para a monitorização e gestão de equipamentos, mas no que diz respeito à gestão e monitorização dos servidores de *e-mail* estas são muito escassas e não têm compatibilidade com a maioria dos servidores de *e-mail*. Das soluções existentes, umas limitam-se a apresentar os *logs* de registo do serviço de *e-mail* em formato gráfico e outras que apenas fornecem a monitorização e não sendo possível a sua gestão.

As soluções existentes podem-se separar em soluções comerciais e em software *Open Source*.

4.3.1 Soluções Comerciais

Algumas empresas têm vindo a desenvolver soluções para os servidores de *e-mail*, tais como:

- AdventNet ManageEngine - Applications Manager [19]
- IceWarp - eMail Server Management [20]
- Azaleos - ViewXchange & ManageXchange [21]

A primeira solução é dedicada ao Microsoft Exchange Server que permite a monitorização de várias informações como o estado dos serviços, das filas de espera, das configurações, etc. Pode-se ligar com outros servidores, sem ser o Microsoft Exchange Server, onde apenas é possível a monitorização do serviço de *e-mail*. A segunda solução é apenas compatível com o servidor de *e-mail* da própria empresa. E a última solução, monitoriza e controla apenas o Microsoft Exchange Server via um protocolo proprietário.

Estas soluções apenas suportam determinados tipos de servidores de *e-mail* e algumas têm protocolo de gestão específico. A título de exemplo, aqui apenas foram apresentadas três soluções, mas existem muitas outras semelhantes a estas.

4.3.2 Software *Open Source*

Existem também ferramentas de domínio público que permitem satisfazer alguns dos requisitos de gestão e monitorização dos servidores de *e-mail*. Estas podem ser organizadas em três grupos:

1. Gestão e monitorização, com recurso ao SNMP.

   - Plataformas de Gestão

   São diversas as plataformas que têm como principal objectivo a gestão e monitorização de equipamentos da rede e de alguns serviços, como o caso do serviço de *e-mail*. Estas serão descritas com detalhe na secção 4.4.

2. Monitorização de alguns parâmetros dos servidores de *e-mail*, sem recurso ao SNMP.

   - Mailgraph [22]
   - Queuegraph [23]
   - Webmin [24]

   A primeira apenas compatível com os servidores Sendmail e Postfix, que limita-se a gerar gráficos das mensagens recebidas/enviadas, em espera e rejeitadas com base nos registos dos *logs*. A segunda é específica para o Postfix, também gera gráficos como a anterior mas estes são referentes às diferentes filas de espera. A última, Webmin, permite a fazer a gestão de configurações de diversos sistemas Unix, onde se pode destacar o Sendmail, Postfix, entre outros servidores de *e-mail*, como também de diversos sistemas de filtragem (como por exemplo: SpamAssassin, Procmail, etc.) que são utilizados juntamente com os servidores de *e-mail*.

3. Monitorização de alguns parâmetros dos servidores de *e-mail*, com recurso ao SNMP.

   - Implementações das MIBs

   Existem algumas tentativas de implementações das MIBs. Estas serão analisadas no capítulo seguinte.

   Um exemplo que se aproxima deste trabalho esta descrito em [25]. Aqui implementa-se uma estação de gestão e um agente SNMP para a gestão do servidor de *e-mail* (servidor escolhido foi o Sendmail). A ideia consiste em aproveitar uma MIB já existente, *Mail Monitoring* MIB [26], com algumas extensões para habilitar as operações *trap*.

   Mesmo esta solução não permite o controlo sobre as configurações do servidor de *e-mail*.

4.4 Plataformas de Gestão

De uma vasta variedade de plataformas que se podem encontrar para a gestão de redes, as mais utilizadas são as seguintes: OpenNMS [27], ZABBIX [28], GroundWork OpenSource [29] e Cacti [30]. Estas plataformas são todas *Open Source* e permitem a monitorização de equipamentos através do protocolo SNMP. Relativamente à gestão não são muito flexíveis, pois em algumas delas a operação de gestão apenas é possível através de um agente que é específico para cada

	OpenNMS	**ZABBIX**	**GroundWork OpenSource**	**Cacti**
Agente	SNMP e outros	SNMP e ZABBIX	SNMP e Nagios	SNMP
Auto descoberta	Sim	Sim	Sim (limitado)	Não
SNMP Versões	Todas	Todas	Todas	Todas
SNMP Operações GET	Sim	Sim	Sim	Sim
SNMP Operações SET	Não	Não	Sim (com criação de *scripts*)	Não
SNMP Operações TRAP	Sim	Sim (via outros módulos)	Sim (via Nagios)	Sim (via outros módulos)
Criação de *add-ons/plugins*	Facilmente desenvolvidos em Java	Exportação e importação de *templates* (XML)	Através do Nagios	Através de PHP/XML
Alertas e notificações	Sim	Sim	Sim	Sim
Poller	Java	C	C	PHP
Interface *Web*	Java	PHP	PHP/Java	PHP
Armazenamento dos dados	PostgresSQL, RRDtool	MySQL, Oracle, PostgresSQL, SQLite	MySQL, RRDtool	MySQL, RRDtool

Tabela 4.1: Plataformas de gestão - Quadro comparativo

plataforma, sendo que essa comunicação não é feita por SNMP. Na tabela 4.1 é apresentado um quadro comparativo de diferentes plataformas.

O GroundWork é uma plataforma que foi desenvolvida com recurso a diferentes ferramentas existentes, entre as quais se encontram: o Nagios, o Cacti, o RRDtool, etc. O Nagios é uma das ferramentas mais utilizadas na monitorização de redes.

Relativamente ao tipo de agente, algumas delas para além do agente SNMP são compatíveis com outros agentes, como é o caso do OpenNMS, ZABBIX e GroundWork.

São todas compatíveis com as várias versões do protocolo SNMP e às operações `Get`. Em relação às operações `Set`, só algumas o permitem através da criação de *scripts*/módulos externos.

No que diz respeito à expansão das plataformas, isto é, criação de *add-ons/plugins* o Nagios apresenta vantagem por criação de simples *scripts* em diversas linguagens de programação.

Existem muitas funcionalidades que são comuns entre estas ferramentas, como o exemplo de recolha de dados de um agente SNMP e posteriormente a apresentação do seu gráfico. Sendo que a apresentação dos gráficos vária, algumas limitam a apresentar o gráfico, outras que têm possibilidade de fazer *zoom*, criar gráficos históricos, listar todos os valores, etc.

Nenhuma destas soluções permite o acesso à configuração do servidor de *e-mail*. Estas apenas fornecem monitorização de qualquer valor que seja fornecido pelo equipamento gerido e que se limitam a gerar gráficos com os valores recolhidos.

4.5 Conclusão

São diversas as soluções que resolvem parcialmente o problema da monitorização. Mas no que diz respeito ao controlo das configurações do servidor de *e-mail*, estas são específicas para determinados tipos de servidores de *e-mail*.

Das soluções apresentadas algumas delas serão utilizadas na implementação.

Capítulo 5

Implementação

Neste capítulo é efectuada uma análise de algumas MIBs, é descrita a base de implementação e é apresentada a solução concretizada.

5.1 MIBs

No caso do servidor de *e-mail*, as MIBs devem possuir objectos que possam satisfazer os seguintes requisitos:

- Monitorização dos pontos críticos;
- Gestão sobre os pontos críticos e as principais configurações;
- Detecção de falhas e o envio de notificações.

5.1.1 MIBs Existentes

De seguida são apresentadas algumas MIBs candidatas, juntamente com a descrição dos objectos que podem ser utilizados das mesmas.

5.1.1.1 MIB-II

Como já referido anteriormente, a MIB-II é uma MIB muito importante para a gestão dos equipamentos, neste caso para gestão da máquina (*host*) onde o servidor de *e-mail* (MTA) está inserido. Nesse sentido, determinou-se quais os objectos que se deveria implementar.

Dos vários ramos presentes na MIB-II, os que se aplicam a este caso são: `system(1)`, `interfaces(2)`, `ip(4)` e `tcp(6)`.

O ramo `system(1)` define uma lista de objectos pertencentes à operação de um sistema, como o tempo de funcionamento, contacto e nome do sistema. Na tabela 5.1 estão os objectos que vão ser utilizados deste ramo.

O próximo ramo `interfaces(2)` é destinado às interfaces de rede do equipamento gerido, onde estão presentes os seguintes objectos: `ifNumber(1)` e `ifTable(2)`. O primeiro indica

iso(1).org(3).dod(6).internet(1).mgmt(2).mib-2(1).system(1)	
Objecto	**Descrição**
sysDescr(1)	Informação textual do sistema a monitorizar, por exemplo: nome do sistema, versão do programa, etc.
sysObjectID(2)	Fornece a indicação de um OID que identifica o sistema, normalmente é uma MIB privada/proprietária.
sysUpTime(3)	Quantifica o tempo desde a última inicialização do sistema.
sysContact(4)	Contacto do responsável pelo sistema. Sendo possível a alteração do valor pela estação de gestão.
sysName(5)	Nome para o sistema. Sendo possível a alteração do seu valor pela estação de gestão.
sysLocation(6)	Local onde se encontra o agente. Sendo possível a alteração do seu valor pela estação de gestão.

Tabela 5.1: Ramo *system* da MIB-II

o número total de interfaces de rede que o sistema possui. O segundo é uma tabela que contém diversas informações sobre as interfaces de rede, onde cada entrada na tabela representa uma interface. Para cada interface de rede, pode-se obter diversas informações, como por exemplo: a sua identificação, o tipo de interface de rede, o seu estado operacional e administrativo, e diversos contadores com informações relativas aos pacotes transmitidos pela interface de rede.

Do ramo ip(4) é importante destacar a tabela ipRouteTable(21) que contém informações sobre a tabela de encaminhamento da interface de rede.

Do ramo tcp(6) a tabela tcpConnTable(13) fornece informações sobre as ligações TCP. Esta tabela pode ser importante para verificar por exemplo as ligações TCP no porto 25, que é utilizado pelo SMTP. Sendo que cada entrada na tabela corresponde a cada ligação TCP.

iso(1).org(3).dod(6).internet(1).mgmt(2).mib-2(1).tcp(6).tcpConnTable(13)	
Objecto	**Descrição**
tcpConnState(1)	O estado da ligação TCP. Que pode ter um dos seguintes valores: closed(1), listen(2), synSent(3), synReceived(4), established(5), finWait1(6), finWait2(7), closeWait(8), lastAck(9), closing(10), timeWait(11) ou deleteTCB(12). Sendo possível a alteração do valor pela estação de gestão.
tcpConnLocalAddress(2)	O endereço IP local para a ligação TCP. Para referir qualquer endereço IP, é utilizado o endereço 0.0.0.0.
tcpConnLocalPort(3)	O porto local utilizado na ligação TCP.
tcpConnRemAddress(4)	O endereço IP remoto da ligação TCP.
tcpConnRemPort(5)	O porto remoto utilizado na ligação TCP.

Tabela 5.2: Tabela *tcpConnTable* da MIB-II

5.1.1.2 MTA-MIB

A MTA-MIB (*Mail Monitoring MIB*) [26] é uma MIB desenvolvida pelo IETF, que contém diversos objectos para a monitorização dos MTAs.

Esta MIB é uma extensão da *Network Services Monitoring MIB* (RFC 2788), também normalizada pelo IETF, que permite monitorizar os serviços de rede, não sendo específica para um tipo de serviço.

A MTA-MIB encontra-se na posição 28 da MIB-II:

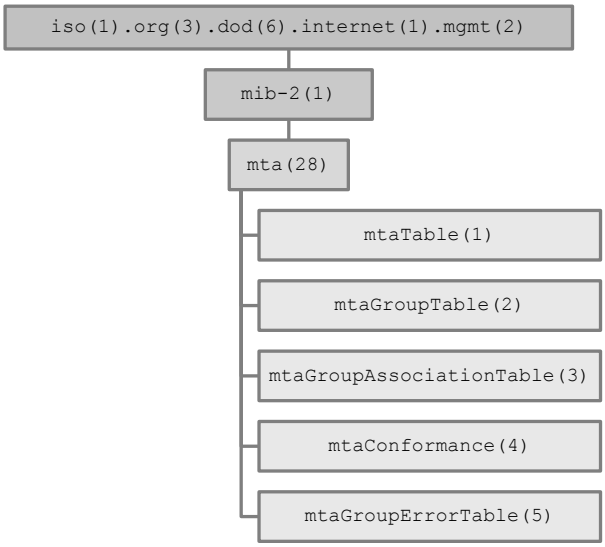

Figura 5.1: Principais ramos da MTA-MIB

São definidas quatro tabelas nesta MIB.

- A primeira delas contém informações para cada MTA, não sendo específica para nenhuma parte em particular do MTA.
- A segunda fragmenta o MTA em componentes designados de grupos.

  O conceito de grupo é utilizado para quebrar as várias actividades de um MTA. Por exemplo, permite separar em componentes: a recepção, o armazenamento nas filas de espera e a entrega.

  Cada grupo contém vários objectos necessárias para a monitorização de todos os aspectos das operações dos MTAs. Nem todos os objectos presentes nesta tabela são obrigatórios.
- A terceira tabela fornece um meio para correlacionar os objectos da *Network Services Monitoring MIB* com grupos específicos de diferentes MTAs.
- Por último, a quarta tabela fornece meios de detecção de alguns erros ocorridos durante o funcionamento do MTA.

As primeiras duas tabelas são obrigatórias caso a MTA-MIB seja implementada, as outras duas são opcionais.

Na tabela 5.3 está apresentado o primeiro ramo desta MIB (`mtaTable(1)`), onde é possível encontrar vários contadores com informações relativas às mensagens recebidas, armazenadas e enviadas pelo MTA.

No segundo ramo está presente a tabela `mtaGroupTable(2)` que possui todos os objectos do ramo anterior, mas aqui a contabilização é feita a nível de grupos. Adicionalmente a `mtaGroupTable(2)` contém vários objectos que permitem caracterizar os grupos.

iso(1).org(3).dod(6).internet(1).mgmt(2).mta(28).mtaTable(1).mtaEntry(1)	
Objecto	**Descrição**
mtaReceivedMessages(1)	O número total de mensagens recebidas desde a inicialização do MTA. Recebidas de outros MTAs, UAs ou aplicações.
mtaStoredMessages(2)	O número total de mensagens actualmente armazenadas no MTA, que inclui as mensagens que estão à espera de serem entregues a outros MTAs, UAs ou aplicações.
mtaTransmittedMessages(3)	O número total de mensagens transmitidas desde a inicialização do MTA.
mtaReceivedVolume(4)	O volume total de mensagens recebidas desde a inicialização do MTA, contabilizado em kilo-octetos.
mtaStoredVolume(5)	O volume total de mensagens actualmente armazenadas no MTA, contabilizado em kilo-octetos.
mtaTransmittedVolume(6)	O volume total de mensagens enviadas desde a inicialização do MTA, contabilizado em kilo-octetos.
mtaReceivedRecipients(7)	O número total de destinatários especificados em todas as mensagens recebidas desde a inicialização do MTA.
mtaStoredRecipients(8)	O número total de destinatários especificados em todas as mensagens actualmente armazenadas no MTA.
mtaTransmittedRecipients(9)	O número total de destinatários especificados em todas as mensagens enviadas desde a inicialização do MTA.
mtaSuccessfulConvertedMessages(10)	O número total de mensagens que foram convertidas com sucesso de um formato para outro, desde a inicialização do MTA.
mtaFailedConvertedMessages(11)	O número total de mensagens em que a sua conversão de um formato para outro não foi realizada com sucesso, desde a inicialização do MTA.
mtaLoopsDetected(12)	Contador de números de vezes que detecta situações de *message loop*.

Tabela 5.3: Tabela *mtaTable* da MTA-MIB

5.1.1.3 WINDOWS-NT-PERFORMANCE-EXCHANGE

O Microsoft Exchange Server possui uma MIB para a monitorização do seu desempenho. Esta MIB é muito optimizada para a arquitectura do próprio servidor, no entanto pode-se aproveitar algumas objectos da mesma.

O nome desta MIB é WINDOWS-NT-PERFORMANCE-EXCHANGE [31], encontra-se definida no interior do ramo privado atribuído à Microsoft (`1.3.6.1.4.1.311`) e apresenta os seguintes ramos:

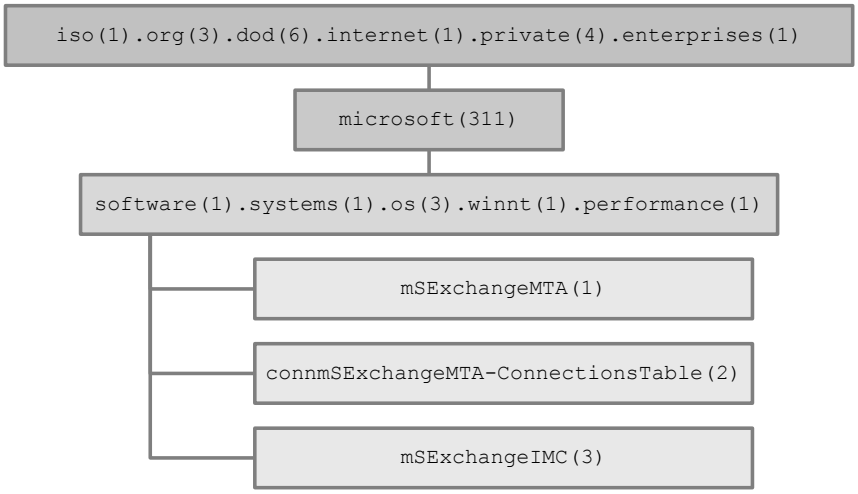

Figura 5.2: Principais ramos da WINDOWS-NT-PERFORMANCE-EXCHANG

Esta MIB está organizada em três ramos principais.

- O primeiro ramo (`mSExchangeMTA(1)`) contém vários contadores, incluindo alguns da MTA-MIB. Possui diversas operações específicas do Microsoft Exchange Server, como por exemplo a monitorização de conectores próprios do servidor. Também contém vários objectos que permitem a monitorização de algumas das operações efectuadas com os ficheiros, como por exemplo: a escrita, os acessos, etc.

- O segundo ramo (`connmSExchangeMTA-ConnectionsTable(2)`) é composto por uma tabela que permite monitorizar as várias ligações estabelecidas com outras entidades. Em termos de objectos, estes caracterizam cada ligação estabelecida, sendo que muitos dos contadores aqui presentes são os mesmos do ramo anterior.

- O último ramo (`mSExchangeIMC(3)`) é destinado à monitorização de um conector, o IMC (*Internet Mail Connector*) que fornece serviços de *gateway* SMTP.

5.1.2 Criação de uma MIB

Quando as MIBs normalizadas não são capazes de satisfazer todos os requisitos, é necessário complementar com objectos de outras MIBs.

Um método utilizado pelos fabricantes de Hardware e Software é implementar as suas próprias MIBs de modo a resolverem as limitações das MIBs normalizadas. Pode-se seguir uma das seguintes vias: expandir uma MIB já existente ou criar uma MIB de raiz.

A metodologia adoptada foi a criação de uma MIB de raiz, utilizando os objectos definidos nas duas principais MIBs: MTA-MIB e WINDOWS-NT-PERFORMANCE-EXCHANGE. As razões que conduziram à tomada desta decisão estão enumeradas de seguida.

- Em relação à MTA-MIB, esta possui objectos que apenas permite fazer a monitorização, não sendo possível a gestão do MTA.

- O conceito de grupos utilizado na MTA-MIB não é de fácil implementação. Pois, tal como descrito no RFC da definição da MIB, os grupos por vezes são confundidos com os grupos também existentes no SNMP. Da pesquisa efectuada às mensagens nas listas de *e-mail* de algumas ferramentas SNMP, verificou-se que este conceito é por vezes entendido incorrectamente como grupos de servidores de *e-mail*.

- As implementações de agentes da MTA-MIB são muito poucas. Apenas se encontrou no agente do servidor da Sun e no Sendmail, mesmo assim estas não respondem a todos os objectos. [32]

- Relativamente à MIB do Microsoft Exchange Server, apenas podem ser aproveitados alguns dos contadores aí presentes, visto que os outros objectos são específicos da arquitectura do próprio servidor.

MAILSERVER-MIB é o nome com que é identificada a MIB criada, que inclui alguns dos objectos presentes nas MIBs anteriormente referidas com a adição de mais alguns novos objectos. Esta MIB pode ser consultada no anexo A.

Como já referido no capítulo 3, as novas MIBs não normalizadas têm de ser definidas no ramo: `iso(1).org(3).dod(6).internet(1).private(4).enterprises(1)`, assim esta MIB foi colocada no ramo do INESC Porto [1] que possui o número `33536` atribuído pela IANA.

A MAILSERVER-MIB é definida e construída de acordo com a estrutura SMIv2. Mas foram tomados todos os cuidados para que a conversão de SMIv2 para SMIv1 se concretize com sucesso.

Não existe nenhum método óptimo para organizar as MIBs, mas sim é importante que antes de as criar se tenha um plano daquilo que se quer e durante toda a MIB deve-se manter o mesmo estilo de organização. [17] O método aqui seguido consistiu em organizar a MIB por funcionalidades. Assim, definiu-se sete ramos no interior desta MIB, como se pode constatar na figura seguinte:

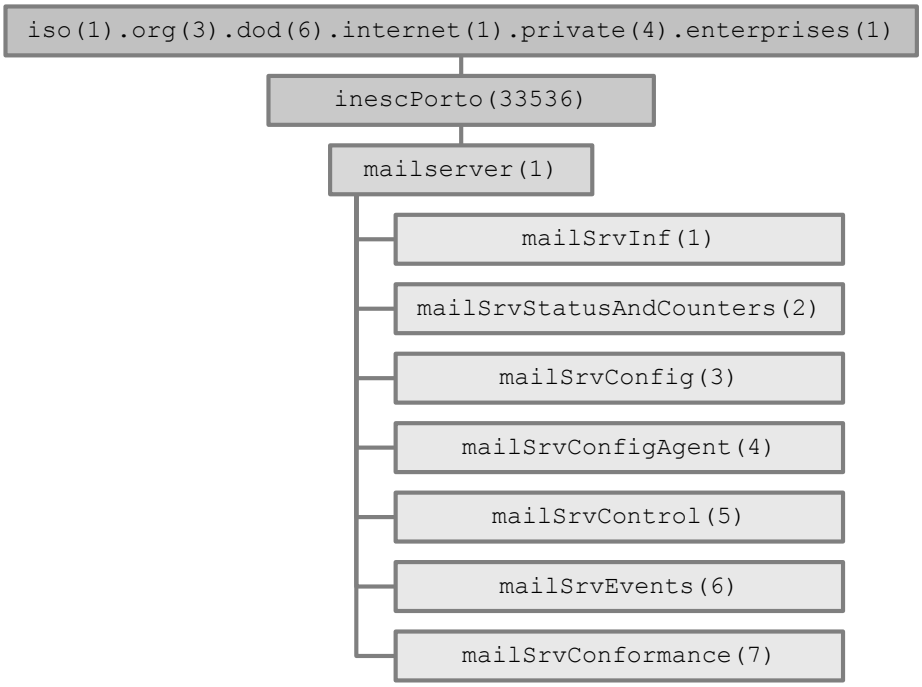

Figura 5.3: Principais ramos da MAILSERVER-MIB

A razão da escolha de cada um dos objectos teve como objectivo satisfazer alguns dos requisitos de funcionamento dos servidores de *e-mail*, como também foram incluídos alguns que permitem a avaliação do seu desempenho. Apenas foram escolhidos os parâmetros que não podem ser obtidos de uma consulta directa. Por exemplo o registo de MX de um determinado domínio pode

[1] INESC Porto - Instituto de Engenharia de Sistemas e Computadores do Porto

ser obtido directamente da consulta do DNS, por isso este tipo de informação não foi incluída na construção da MIB.

De seguida é feita uma descrição de cada ramo, bem como a indicação dos objectos presentes em cada um deles.

5.1.2.1 mailSrvInfo(1)

O ramo `mailSrvInfo(1)` contém informações gerais sobre o MTA, que permitem a sua identificação e caracterização.

- `infoDescr(1)`
  Descrição textual do MTA, que deve incluir a sua identificação e a versão. Deve-se incluir a versão, pois esta informação por vezes não é obtida directamente dos servidores, devido a questões de segurança.

- `infoUpTime(2)`
  Quantifica o tempo desde a última inicialização do MTA.

- `infoContactPostmaster(3)`
  O contacto do responsável pelo MTA e a descrição do modo como este é contactável.

- `infoServices(4)`
  Descrição textual dos serviços oferecidos pelo MTA. Como por exemplo a lista dos anti-vírus.

- `infoDomainsAccept(5)`
  Descrição textual dos domínios para os quais o MTA vai fazer a entrega local ou entrega para outro MTA do seu domínio.

- `infoDomainsRelay(6)`
  Descrição textual dos domínios para os quais o MTA vai fazer *relay* dos *e-mails*.

- `infoDefaultTransport(7)`
  Descrição do transporte padrão para a entrega.

- `infoMailboxType(8)`
  Descrição to tipo de armazenamento utilizado.

5.1.2.2 mailSrvStatusAndCounters(2)

O segundo ramo (`mailSrvStatusAndCounters(2)`) é o mais importante de todos, pois a maioria dos objectos para a monitorização encontram-se aqui.

- `statusConnectionsInbound(1)`
  O número actual de ligações SMTP estabelecidas do exterior para o MTA. Permite verificar se está a receber muitas ligações SMTP.

- `statusConnectionsOutbound(2)`
  O número actual de ligações SMTP estabelecidas do MTA para o exterior. Permite verificar se está a estabelecer muitas ligações SMTP com outros servidores.

- `statusProcessesTable(3)`
  Tabela que contém objectos que permitem a monitorização dos processos.

`1.3.6.1.4.1.33536.mailserver(1).mailSrvStatusAndCounters(2).statusProcessesTable(3).statusProcessesEntry(1)`

Objecto	Descrição
`processesIndex(1)`	Valor identificador do processo monitorizado. Cada processo terá um valor diferente.
`processesName(2)`	O nome do processo em questão.
`processesState(3)`	O estado actual do processo. Pode ter um dos seguintes valores: `notrunning(0)` ou `running(1)`.
`processesCpu(4)`	A capacidade do CPU utilizada pelo processo.
`processesMem(5)`	A memória física utilizada pelo processo.
`processesNumSubProc(6)`	O número de sub-processos criados pelo processo monitorizado. Isto é, o número de processos filhos do processo monitorizado.

Tabela 5.4: Tabela *statusProcessesTable* da MAILSERVER-MIB

- `statusQueuesTable(4)`
  Tabela que contém objectos que permitem a monitorização das filas de espera.

`1.3.6.1.4.1.33536.mailserver(1).mailSrvStatusAndCounters(2).statusQueuesTable(4).statusQueuesEntry(1)`

Objecto	Descrição
`queuesIndex(1)`	Valor identificador da fila de espera monitorizada. Cada fila terá um valor diferente.
`queuesName(2)`	O nome da fila de espera em questão.
`queuesCurrMsgs(3)`	O número total de mensagens actualmente na fila de espera.
`queuesCurrVolume(4)`	O volume total de mensagens actualmente na fila de espera.
`queuesMsgsIn(5)`	O número total de mensagens que entraram na fila de espera desde a inicialização do MTA.
`queuesMsgsOut(6)`	O número total de mensagens que saíram da fila de espera desde a inicialização do MTA.
`queuesOldestMessageId(7)`	O identificador da mensagem mais antiga presente na fila de espera.

Tabela 5.5: Tabela *statusQueuesTable* da MAILSERVER-MIB

- `statusConnectionsMade(5)`
  O número total de ligações SMTP estabelecidas desde a inicialização do MTA.

- `statusConnectionsLostInbound(6)`
  O número total de ligações SMTP recebidas que foram perdidas desde a inicialização do MTA.

- `statusConnectionsFailureOutbound(7)`
  O número total de ligações SMTP que foram estabelecidas com insucesso outros MTAs desde a inicialização do MTA.

- `statusMsgsTotalAccepted(8)`
  O número total de mensagens recebidas desde a inicialização do MTA.

- `statusMsgsTotalRejected(9)`
  O número total de mensagens rejeitadas desde a inicialização do MTA.

- `statusMsgsTotalSent(10)`
  O número total de mensagens enviadas desde a inicialização do MTA.

- `statusMsgsTotalSpam(11)`
  O número total de mensagens em que o filtro considerou como sendo *spam*, contabilizadas desde a inicialização do MTA.

- `statusMsgsTotalVirus(12)`
  O número total de mensagens em que o filtro considerou sendo de vírus, contabilizadas desde a inicialização do MTA.

- `statusMsgsTotalBounced(13)`
  O número total de mensagens que foram devolvidas ao MTA desde a inicialização do MTA.

- `statusVolumeTotalAccepted(14)`
  O volume total de mensagens recebidas desde a inicialização do MTA.

- `statusVolumeTotalSent(15)`
  O volume total de mensagens enviadas desde a inicialização do MTA.

- `statusLogFileVolume(16)`
  O volume actual do ficheiro de *log* do MTA. O aumento muito rápido deste valor, pode significar o aumento da actividade do MTA, isto é, está a processar mais mensagens.

- `statusMailSrvError(17)`
  O número total de erros ocorridos desde a inicialização do MTA.

- `statusMailSrvWarning(18)`
  O número total de avisos emitidos desde a inicialização do MTA.

- `statusMsgsTotalRecipientsIn(19)`
  O número total de destinatários especificados em todas as mensagens recebidas desde a inicialização do MTA.

- `statusMsgsTotalRecipientsOut(20)`
  O número total de destinatários especificados em todas as mensagens enviadas desde a inicialização do MTA.

- `statusSuccessfulConvertedMsgs(21)`
  O número total de mensagens que foram convertidas com sucesso de um formato para outro, desde a inicialização do MTA.

- `statusFailedConvertedMsgs(22)`
  O número total de mensagens em que a sua conversão de um formato para outro não foi realizada com sucesso, desde a inicialização do MTA.

- `statusFingerprintingSpamOS(23)`
  O *OS Fingerprinting* é um método que permite determinar o sistema operativo que estabelece ligação com o MTA. Aqui é colocada a informação relativa ao sistema operativo que envia mais mensagens do tipo *spam*. Esta informação pode ser utilizada na configuração de filtros de *e-mail*, como por exemplo atribuir uma classificação elevado para o sistema operativo que envia mais mensagens *spam*.

- `statusFingerprintingHamOS(24)`
  Semelhante ao anterior mas aqui é colocada a informação relativa ao sistema operativo que envia mais mensagens que não são *spam*.

5.1.2.3 mailSrvConfig(3)

No ramo `mailSrvConfig(3)` estão incluídas várias operações de gestão sobre as configurações do MTA. Apenas foram incluídas as configurações importantes, pois existem ferramentas *Web* que permitem já fazer este tipo de operações, tal como as apresentadas na secção do estado da arte.

- `configTimeoutReq(1)`
  O intervalo de tempo para a espera de cada pedido, depois deste tempo a sessão SMTP é terminada.

- `configMaxMsgSize(2)`
  O tamanho máximo permitido para cada mensagem, incluindo os anexos.

- `configMaxRecipients(3)`
  O número máximo de destinatários permitidos por cada mensagem. Quando o servidor estiver como MTA *relay*, pode-se diminuir este valor para não permitir o envio de mensagens *spam* para múltiplos destinatários.

- `configTimeoutBeforeError(4)`
  O tempo de espera antes de enviar resposta de rejeição (normalmente os erros identificados com códigos 4xx ou 5xx).

  Aumento deste tempo, pode de algum modo ajudar a diminuir mensagens *spam*, pois os servidores que enviam este tipo de mensagens normalmente não costumam esperar muito tempo pelas respostas. Pois têm outros servidores alvo na sua enorme lista.

- `configMaxParallelDeliveries(5)`
  O número máximo de entregas permitidas em simultâneo. Em caso das mensagens nas filas

de espera apresentarem uma ocupação elevada, pode-se aumentar este número para permitir que sejam processadas mais mensagens das filas.

- `configHopCountLimit(6)`
  Permite limitar o número de saltos que uma mensagem pode realizar. Ultrapassado este limite, a mensagem é considerada como perdida e é identificada como *mail loop*. Este valor deve ser alterado em função do objecto `eventMailSrvLoopDetection(3)`.

- `configMainQueueLimit(7)`
  O número máximo de mensagens permitidas na fila de espera principal, permitindo assim ter o controlo da capacidade da fila.

5.1.2.4 mailSrvConfigAgent(4)

O `mailSrvConfigAgent(4)` permite efectuar alterações no funcionamento do agente.

- `configInternalPollFreq(1)`
  Permite definir com que frequência é que o agente actualiza os seus dados internos. Isto é benéfico pois previne que o agente não esteja constantemente a actualizar os seus dados e a consumir recursos do MTA.

- `configEventSeverityThreshold(2)`
  Permite definir um limiar para a gravidade das notificações enviadas pelo agente. Definiu-se cinco níveis de gravidade: *normal, warning, minor, major* e *critical*. Embora não sejam utilizados todos os níveis, estes poderão ser importantes para implementações futuras.

5.1.2.5 mailSrvControl(5)

O controlo do funcionamento do MTA foi definido o ramo `mailSrvControl(5)`.

- `controlStatus(1)`
  Aqui é definido o estado actual do MTA, sendo possível a alteração do seu estado para um dos seguintes valores: `notrunning(0)` ou `running(1)`.

- `controlReload(2)`
  Este objecto permite recarregar as configurações do MTA. Depois da modificação de alguns objectos do ramo `mailSrvConfig(3)` é necessário proceder à actualização do MTA através deste objecto.

- `controlQueueDeleteMsg(3)`
  Permite apagar uma ou todas as mensagem da fila de espera principal. É necessário o ID que foi atribuído à mensagem na fila de espera.

- `controlQueueRequeueMsg(4)`
  Permite colocar uma ou todas as mensagem na fila de espera principal. É necessário o ID que foi atribuído à mensagem na fila de espera.

5.1.2.6 mailSrvEvents(6)

No ramo `mailSrvEvents(6)` estão definidos os vários tipos de eventos que podem ser gerados pelo MTA.

- `eventList(0)`
  Contém a lista das notificação possíveis.
  - `eventMailSrvStart(1)`
    Notificação que indica o arranque do MTA.
  - `eventMailSrvShutdown(2)`
    Notificação que indica o terminar do MTA.
  - `eventMailSrvLoopDetection(3)`
    Notificação destina à detecção de mensagens perdidas (*mail loop*).
  - `eventMailSrvStorageOverflow(4)`
    Notificação enviada quando o limite para o armazenamento das mensagens ultrapassa um limite pré-definido. Este limite é por cada *mailbox*.
  - `eventMailSrvConfigChanged(5)`
    Notificação que indica a alteração dos ficheiros de configurações do MTA.
  - `eventMailSrvGeneric(6)`
    Notificação para qualquer tipo de uso, anteriormente não referida.
- `eventDescriptors(1)`
  Contém a lista dos objectos que permite descrever as notificações.
  - `eventSeverity(1)`
    Indica a gravidade da notificação.
  - `eventDescription(2)`
    Descrição textual para a notificação.

5.1.2.7 mailSrvConformance(7)

Por último o `mailSrvConformance(7)`, apenas define os objectos que devem ser obrigatórios para estar em conformidade com a MAILSERVER-MIB.

- `mtaReqConfGroup(1)`
  Os grupos obrigatórios na implementação do agente.
- `mtaNotificationGroup(2)`
  As notificações obrigatórias na implementação do agente.
- `mtaModCompliances(3)`
  A lista dos grupos de conformidade que foram definidos.

5.2 Ferramentas e Tecnologias Escolhidas

5.2.1 MTA

O MTA eleito para a verificação da MIB construída foi o Postfix. As principais razões da sua escolha devem-se ao facto de ser um dos MTAs com maior utilização (como já foi visto anteriormente) e a facilidade que oferece na sua configuração. Está devidamente documentado e por já se ter tido experiência de utilização com o mesmo.

Apesar dos servidores de *e-mail* apresentarem diversos mecanismos nativos de filtragem, actualmente muitos deles recorre a outros sistemas de filtragem externos. Um sistema de filtragem que é muito utilizado é o amavisd-new [33], que oferece uma interface entre o MTA e os diferentes tipos de filtros existentes. A comunicação entre o MTA e o amavisd-new é efectuada através dos protocolos LMTP ou SMTP, ou ainda com utilização de programas específicos.

São vários os filtros utilizados pelos amavisd-new, como por exemplo o SpamAssassin que tenta identificar as mensagens *spam* usando uma variedade de mecanismos de filtragem, entre os quais inclui a verificação dos servidores remetentes por SPF e a verificação das assinaturas DKIM.

O amavisd-new utiliza as pontuações atribuídas às mensagens pelos diferentes filtros para determinar se a mensagem é *spam* e o que fazer caso seja.

Dada a enorme utilização do amavisd-new, não apenas no Postifix mas também noutros MTAs, este também é incluído na solução.

5.2.2 Agente

Relativamente ao agente, optou-se por fazer uma extensão de um agente já existente, o Net-SNMP [32].

O Net-SNMP é pacote de software que oferece um agente SNMP (snmpd), uma aplicação que permite receber as notificações SNMP (snmptrapd) e uma variedade de ferramentas para realizar as várias operações do protocolo SNMP.

Com a extensão do agente Net-SNMP os detalhes técnicos do SNMP são camuflados atrás da API de programação de alto nível (neste caso a linguagem de programação, Perl), levando apenas a preocupar-se essencialmente com os serviços a desenvolver e não com detalhes protocolares.

5.2.3 Estação de Gestão

Das várias soluções para a plataforma de gestão apresentadas no capítulo do estado da arte, escolheu-se o ZABBIX.

A principal razão da escolha de uma plataforma já implementada deve-se ao facto de permitir também a integração com outros sistemas de monitorização por SNMP, entre outras razões que se podem destacar:

- Devido à sua escalabilidade que permite a monitorização de uma elevada quantidade de equipamentos;
- Monitorização distribuída que permite configuração centralizada e acesso centralizado a toda a informação;
- Oferece mecanismos de monitorização em tempo real;
- Permite verificar condições e gerar alertas;

Como o ZABBIX não permitia algumas operações, houve a necessidade de fazer algumas extensões nesta plataforma, as quais serão descritas mais à frente.

5.2.4 Linguagens de Programação e Base de Dados

As linguagens de programação escolhidas foram Perl (agente e estação de gestão) e PHP (estação de gestão). A linguagem Perl (*Practical Extraction And Report Language*) oferece uma extensa biblioteca de funções, bem como permite um uso fácil de expressões regulares. A linguagem PHP (*Hypertext Preprocessor*) por ser uma das mais utilizadas para o desenvolvimento *Web* e onde existe uma vasta e muito bem documentada biblioteca de funções.

A base de dados escolhida foi PostgreSQL, uma das mais populares e por ter um bom índice de desempenho.

5.3 Sistema de Gestão

5.3.1 Arquitectura

Na figura 5.4 pode-se visualizar a arquitectura do sistema de gestão.

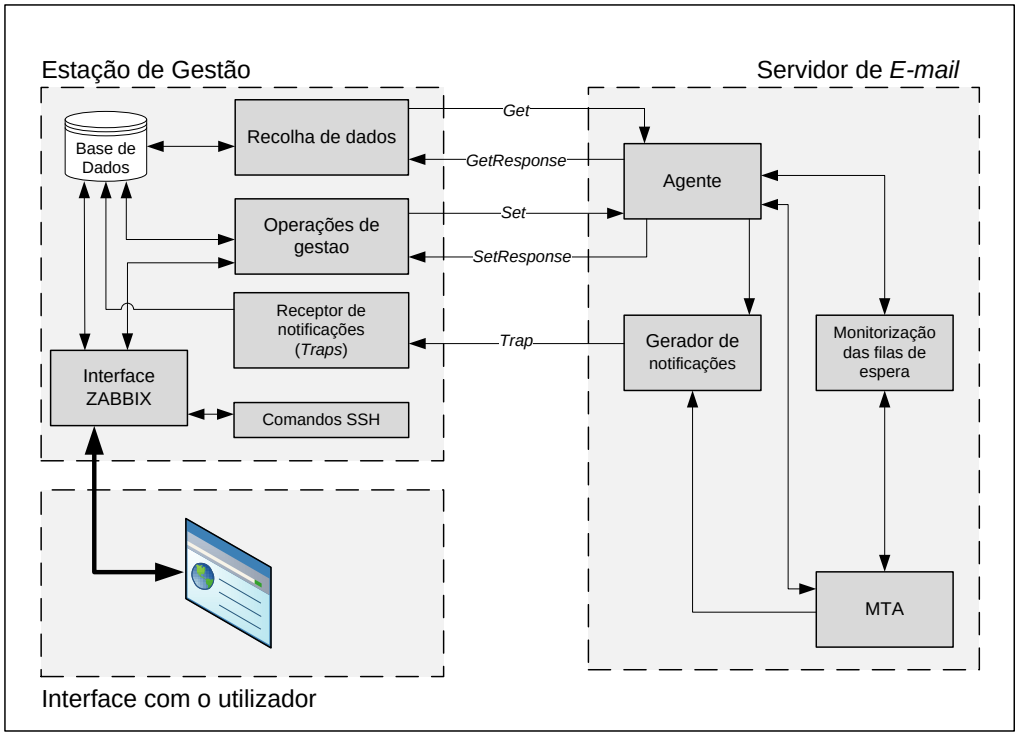

Figura 5.4: Arquitectura do sistema de gestão

O sistema de gestão é composto por três blocos essenciais: a estação de gestão, o agente gerido (servidor de *e-mail*) e a interface com o utilizador.

Na estação de gestão estão presentes os módulos que permitem a recolha e o tratamento dos dados, a recepção de notificações dos agentes, efectuar operações de gestão e um módulo para interagir com o utilizador.

No servidor de *e-mail*, estão os módulos que permitem actualizar a informação da MIB, responder aos pedidos SNMP e permitem também o envio de notificações.

5.3.2 Módulos do Servidor de *E-mail*

5.3.2.1 Agente

Este módulo inclui o agente do Net-SNMP mais o agente do MTA. A comunicação entre estes está esquematizada na figura 5.5.

No início de cada pedido existe a troca de mensagens PING e PONG, que serve para determinar a disponibilidade do agente MTA. Após receber a resposta, pode-se enviar os seguintes

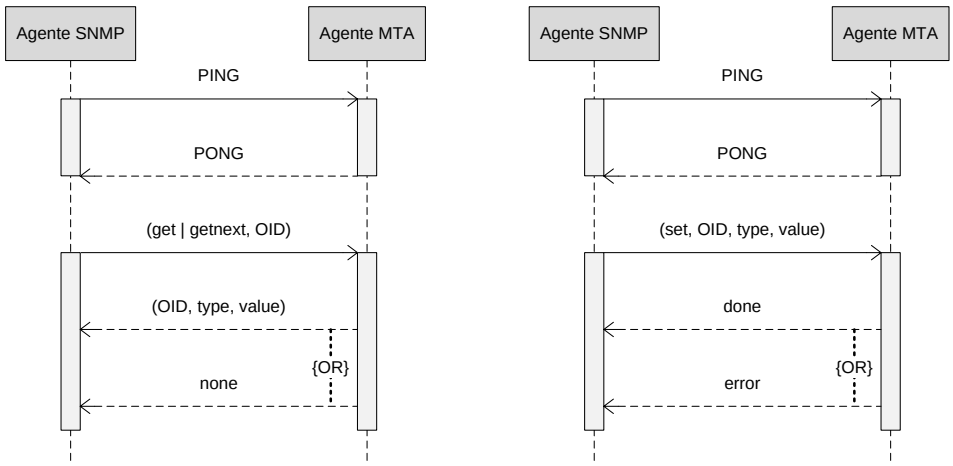

Figura 5.5: Comunicação entre os agentes

pedidos: `Get`, `GetNext` ou `Set`, seguido do OID pretendido. No caso da operação `Set`, adicionalmente é necessário enviar o tipo e o novo valor pretendido para o OID.

A resposta ao `Get` e `GetNext` inclui o OID, o tipo de dados e o valor do OID pedido, ou NONE caso o agente não implemente esse OID. A resposta ao `Set` pode ser positiva se a alteração tenha sido concretizada com sucesso (DONE), caso contrário é enviada uma mensagem de erro SNMPv2.

O agente Net-SNMP vai permitir monitorizar a máquina onde o MTA está inserido. Isto é conseguido pela MIB-II, que no agente Net-SNMP já vem implementada. O agente MTA vai implementar a MIB criada, MAILSERVER-MIB.

O agente MTA inclui os seguintes ficheiros:

- **`agent.pl`**
  O programa do agente;

- **`agent.conf`**
  Contém as configurações do funcionamento do agente, que podem ser: a localização dos ficheiros, a lista dos processos a monitorizar, a lista das filas de espera a monitorizar e algumas informações estáticas do MTA. Também pode-se definir um de três níveis passíveis para *debug*.

```
# cat /etc/snmp/agent/agent.conf

MYHOME = /etc/snmp/agent
agent_log_file = MYHOME/agent.log
agent_pid_file = MYHOME/agent.pid
(...)
mailserver_log_file = /var/log/mail.log
agent_debug_level = 2
baseOID = .1.3.6.1.4.1.33536.1

# List of processes to monitor
```

```
# ex: proc = <full_path_to_a_pid_file>
proc = /var/run/amavis/amavisd.pid
proc = /var/spool/postfix/pid/master.pid

# List of queues to monitor
# ex: queue = <full_path_to_a_queue_dir>
queue = /var/spool/postfix/incoming
queue = /var/spool/postfix/active
queue = /var/spool/postfix/deferred
queue = /var/spool/postfix/hold
queue = /var/spool/postfix/corrupt

# Description of the mailserver
infoDescr                  = MTA - Postfix v2.5.5-1.1
# Postmaster contact
infoContactPostmaster      = sunny@debian.lan
# Description of the services/processes offered by mailserver
infoServices               = amavisd-new, avast

# GET/SET OPERATIONS
configInternalPollFreq       = 20
configEventSeverityThreshold = 1
```

- **`agent.log`**
  O registo de todas as operações realizadas pelo agente consoante o nível de *debug* definido.

```
# cat /etc/snmp/agent/agent.log
(...)
Mon Jun 22 04:20:34 WEST 2009 [INFO ] : PING and PONG
Mon Jun 22 04:20:34 WEST 2009 [INFO ] : Operation: GET, OID: .1.3.6.1.4.1.33536.1.3.1.0
Mon Jun 22 04:20:34 WEST 2009 [INFO ] : Response get > OID: .1.3.6.1.4.1.33536.1.3.1.0,
                                              type: integer, value: 1800
(...)
```

- **`agent.pid`**
  Aqui é colocado o valor actual do PID (*Process ID*) do agente. Este valor é depois utilizado por um *script* (`postfix_status.pl`) para poder comunicar com o agente (comunicação inter-processos). Por exemplo, quando o Postfix muda de estado é executado esse *script* que comunica com o agente para colocar todos os contadores a zero. Isto foi conseguido pela alteração do programa de inicialização do Postfix:

```
# cat /etc/init.d/postfix
(...)
    start)
    log_daemon_msg "Starting Postfix Mail Transport Agent" postfix
    RUNNING=$(running)
    /etc/snmp/agent/postfix_status.pl start 2>/dev/null
(...)

    stop)
    RUNNING=$(running)
    log_daemon_msg "Stopping Postfix Mail Transport Agent" postfix
    /etc/snmp/agent/postfix_status.pl stop 2>/dev/null
(...)
```

De seguida são apresentadas os principais métodos do agente:

- **`getConfigOptions`**
  Permite obter as configurações de funcionamento do agente, que estão definidas no ficheiro `agent.conf`. Caso alguma configuração não esteja nesse ficheiro, então é utilizado um valor pré-definido.

- **`updateGlobalData`**
  Método responsável pela actualização da maior parte dos dados dos OIDs. Entre as actualizações, as mais importantes são:

  - `parserPostfixMain_cf` é responsável pela actualização dos valores das configurações de funcionamento do MTA. Basicamente são quase todos os objectos do ramo `mailSrvConfig(3)`;
  - `updateCounters` permite actualizar a maior parte dos contadores que estão no ramo `mailSrvStatusAndCounters(2)`. A actualização é conseguida pela filtragem do *log* de registos do MTA. Como existem muitas ferramentas que efectuam filtragem de *logs*, nesse sentido utilizou-se um programa da ferramenta LogWatch que permite filtrar os *logs* do Postfix. Por questões de eficiência, a leitura do ficheiro de *log* é efectuada através de apontadores, ou seja, a última posição do ficheiro lida é guardada na memória. Caso na próxima leitura não consiga posicionar-se nessa posição, a leitura é feita do início do ficheiro e é enviada uma notificação à estação de gestão para dar conhecimento desta ocorrência de erro;
  - `updateQueuesCounters` funciona de forma semelhante ao anterior, mas aqui o ficheiro de *log* é referente aos acessos à fila de espera. São actualizados os objectos `queuesMsgsIn(5)` e `queuesMsgsOut(5)`;
  - `updateAmavisCounters` actualiza os objectos `statusMsgsTotalSpam(11)`, `statusMsgsTotalVirus(12)`, `statusFingerprintingSpamOS(23)` e `statusFingerprintingHamOS(24)` consoante um ficheiro de estatísticas criado pelo amavisd-new.

  Como os contadores utilizados são de 32 bits é necessário garantir que não ultrapassem esse limite, isto é feito da seguinte forma:

```
my $limit32bit = (2**32)-1;

foreach my $key (%{$counters}) {
    if( defined($counters->{$key}) ) {
        $counters->{$key} = int($counters->{$key}) % ($limit32bit);
    }
}
```

- **`returnResponse`**
  É determinado qual a resposta aos pedidos. O seu funcionamento geral é descrito no fluxograma da figura 5.6.

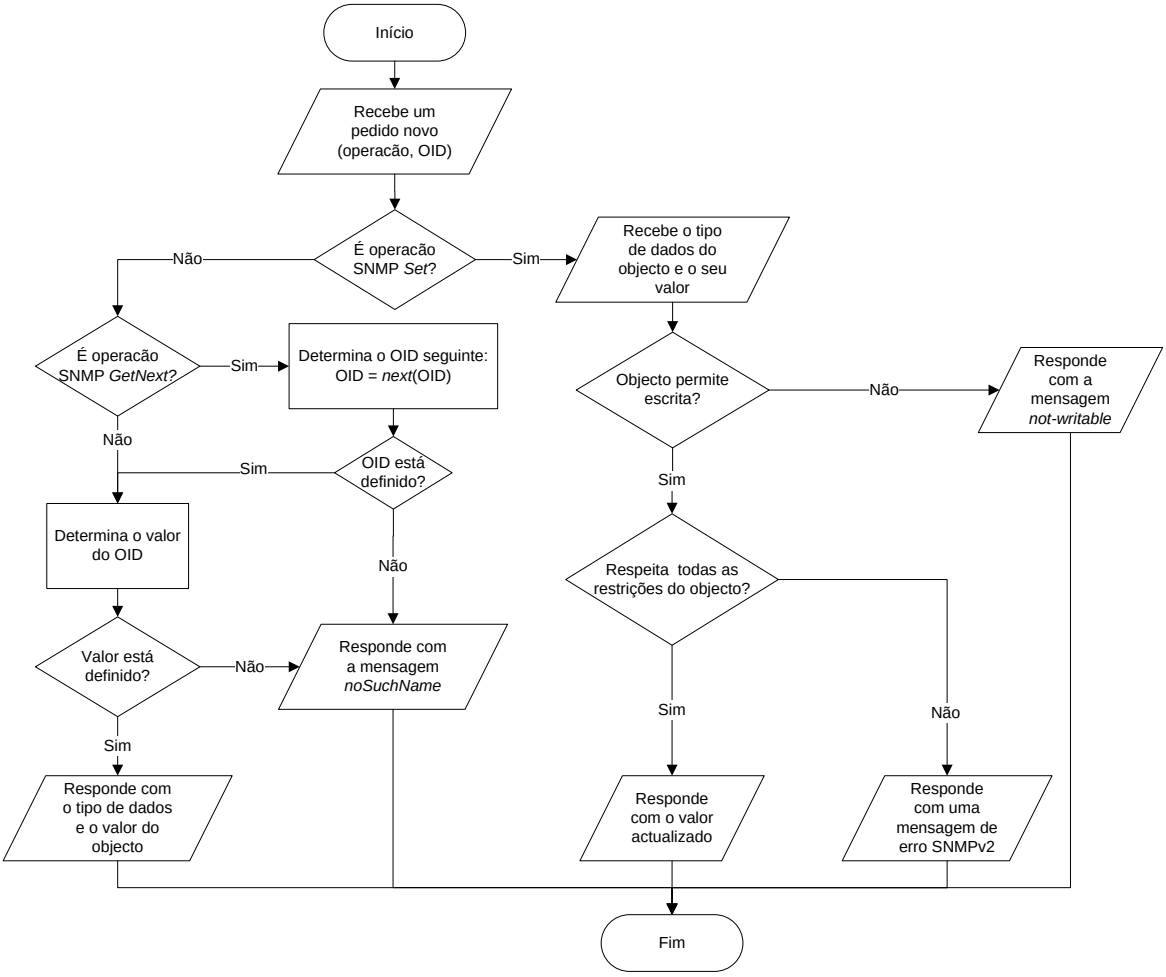

Figura 5.6: Fluxograma - Operações SNMP

- **`getNextOIDbyOID_statusProcessesTable`**, **`getNextOIDbyOID_statusQueuesTable`** **`getTypeValByOID_statusProcessesTable`** e **`getTypeValByOID_statusQueuesTable`**
  Os dois primeiros permitem obter o OID seguinte dos objectos das tabelas: processos e filas de espera. Enquanto os outros dois permitem obter o seu tipo de dados e o valor.

- **`update_infoUpTime`**
  É actualizado o objecto que responde ao tempo de execução do servidor de *e-mail*.

5.3.2.2 Gerador de Notificações

Este módulo tem como função o envio de todas as notificações do agente para a estação de gestão. Por cada notificação, são enviados os seguintes dados:

`OID-trap`	`infoUpTime`	`eventSeverity`	`eventDescription`

O `OID-trap` indica o OID da notificação. Os outros objectos já foram descritos anteriormente.

5.3.2.3 Monitorização das Filas de Espera

Tem como objectivo monitorizar as filas de espera, mais concretamente o número de mensagens que entram e saem da fila.

Por todas as mensagens que atravessam a fila de espera é criado um registo de *log* de acessos à fila, onde posteriormente esta informação é utilizada pelo módulo do agente. Isto é conseguido por um mecanismo de monitorização de sistemas de ficheiros, o *inotify*.

Nesta implementação apenas se testou com uma fila de espera (activa).

5.3.3 Módulos da Estação de Gestão

5.3.3.1 Recolha de Dados

Este módulo já faz parte da plataforma ZABBIX e não houve necessidade de fazer alterações no seu funcionamento. Na figura 5.7 pode-se visualizar um breve esquema do seu funcionamento.

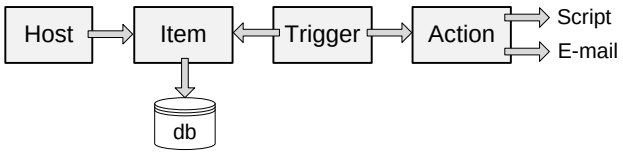

Figura 5.7: Recolha de dados no ZABBIX

Os *hosts* são os equipamentos geridos, que podem ter diversos objectos (*items*) a serem monitorizados pela plataforma. Ao actualizar o valor de um objecto na base de dados, pode-se definir um *trigger* que é uma condição a verificar. Caso a condição se verifique, então pode-se gerar uma acção (*action*). Sendo que a acção pode resultar na execução de um *script* ou o envio de uma mensagem de *e-mail*.

5.3.3.2 Operações de Gestão

O objectivo deste módulo é fazer a extensão da plataforma ZABBIX para permitir efectuar operações do tipo `Set`.

Na sua implementação teve-se o cuidado de garantir que este não fosse apenas específico para os servidores de *e-mail*, mas sim para qualquer outro agente SNMP que implemente as operações `Set`. Faz ao máximo o aproveitamento dos dados já existentes na base de dados, apenas foi necessário criar uma tabela na base de dados:

```
CREATE TABLE items_set (
    snmp_community_set varchar(64) DEFAULT '' NOT NULL,
    hostid integer NOT NULL REFERENCES hosts(hostid) ON DELETE CASCADE,
    itemid integer NOT NULL UNIQUE REFERENCES items(itemid) ON DELETE CASCADE,
    snmp_value_set varchar(255) NULL,
    snmp_timeout_set integer NOT NULL default '30',
    snmp_retries_set integer NOT NULL default '2'
) with OIDS;
```

Como a *community string* geralmente é diferente nas operações de escrita (`Set`), foi necessário guardar uma nova pois a que o ZABBIX tinha só permitia operações de leitura (`Get`).

O `hostit` e `itemit` são chaves estrangeiras que permitem associar a operação `Set` ao agente e ao objecto, respectivamente.

Os valores `snmp_value_set`, `snmp_timeout_set` e `snmp_retries_set` dizem respeito à operação `Set`. O primeiro indica o novo valor para o objecto, o segundo indica o intervalo de tempo que a estação de gestão espera pela resposta do agente e o último valor indica o número de tentativas caso o agente não responda.

Apenas é efectuada a modificação do valor pedido e é mostrado o resultado ao utilizador. Não se procede à actualização do valor na base de dados para evitar problemas de sincronismo com os outros módulos do ZABBIX. Ao utilizador é dito que passado *x* tempo o valor do objecto será actualizado na base de dados.

5.3.3.3 Receptor de Notificações

O ZABBIX apenas permite receber notificações do seu agente nativo. Sendo que estas não são notificações SNMP.

Este módulo tem como objectivo estabelecer uma interface entre as notificações SNMP e o ZABBIX. Ou seja, todas as notificações SNMP são recebidas por este módulo e é determinado origem do evento. Posteriormente é enviado para ZABBIX via uma aplicação própria da plataforma ZABBIX (zabbix_sender). A fluxograma da figura 5.8 permite exemplificar melhor o seu funcionamento.

5.3.3.4 Comandos SSH

Na plataforma de gestão criou-se uma *interface* para permitir executar comandos no servidor de *e-mail* através de uma ligação SSH (*Secure Shell Protocol*).

Este tem como principal objectivo obter a lista e a causa das mensagens não entregues que estão presentes nas filas de espera, para depois ser possível executar acções de gestão nas filas por SNMP. Pode-se também utilizar para a consulta de diversos registos de DNS.

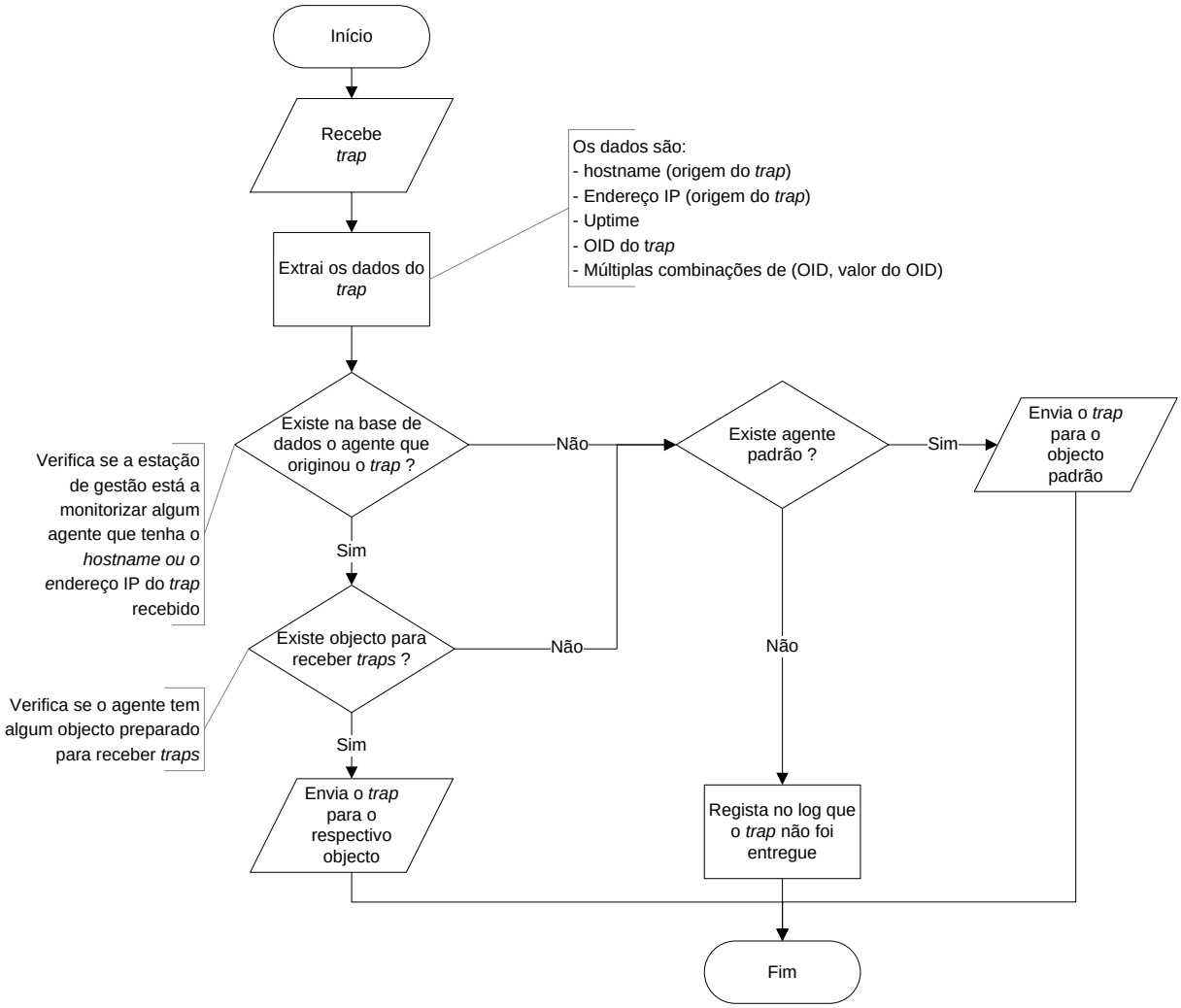

Figura 5.8: Fluxograma - Receptor de notificações SNMP

5.4 Segurança

A versão do SNMP utilizada na implementação e nos testes foi a SNMPv2. Mas o agente é compatível com a versão SNMPv3. Por razões de segurança, a versão SNMPv3 deve ser escolhida num ambiente de produção.

Na figura 5.9 é apresentado um teste que foi realizado para comprovar que o envio da *community string* e dos dados na versão SNMPv2 não oferecem segurança.

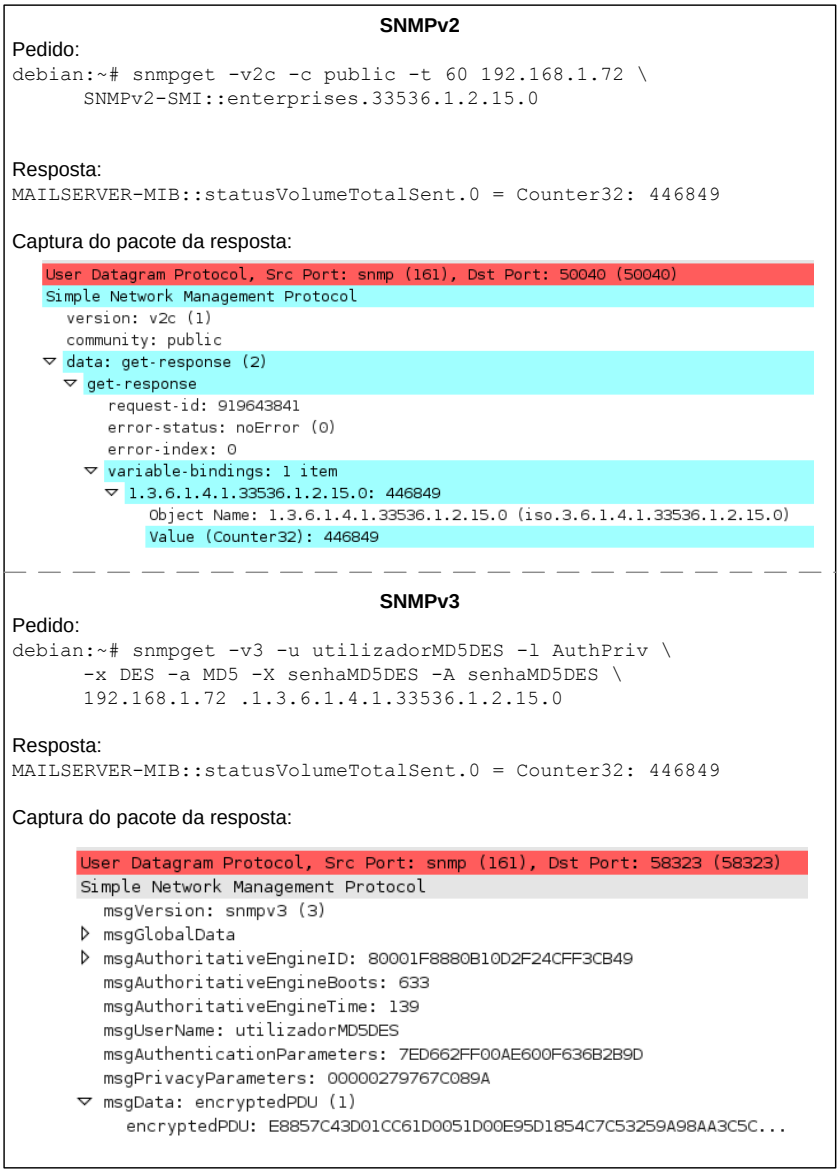

Figura 5.9: Teste de segurança com SNMPv2 e SNMPv3

Na captura dos pacotes pode-se ver que todos os dados transmitidos na versão SNMPv2 vão em claro, enquanto na versão SNMPv3 tanto os dados como a senha de acesso são encriptados. A senha é encriptada pelo algoritmo MD5 e os dados pelo algoritmo DES.

5.5 Conclusão

A análise das MIBs existentes revelou que estas apenas ofereciam operações de monitorização, não sendo possível a sua gestão. Com base nos requisitos e nas MIBs analisadas criou-se uma nova MIB (MAILSERVER-MIB).

O agente foi desenvolvido para o Postfix, sendo facilmente adaptável para outros MTAs da arquitectura Unix, tais como o Sendmail e qmail. Adaptação conseguida pela alteração de alguns comandos e a localização dos ficheiros. Para o Microsoft Exchange Server, pode-se seguir a lógica do agente e implementar um que seja adequado para a arquitectura Windows.

Apenas foram efectuados testes com os objectos da MAILSERVER-MIB. Os objectos da MIB-II não foram testados, pois tanto o agente Net-SNMP como a plataforma ZABBIX têm compatibilidade com esta MIB.

A solução apresentada não efectua a gestão das caixas de *e-mail*, nem monitoriza os acessos às mesmas. Existem também outras operações que aqui não foram consideradas e que permitem melhorar o desempenho dos servidores de *e-mail*, como por exemplo: a rotatividade dos logs, o balanceamento de carga feito por DNS *round-robbin*, a replicação dos dados, a utilização de DNS *cache*, etc.

O sistema de gestão apresentado permite monitorizar e efectuar algumas operações a gestão no servidor de *e-mail*. Deve-se ter em conta à frequência de *polling* dos pedidos SNMP, de modo a não influenciar o desempenho dos servidores de *e-mail*.

Capítulo 6

Resultados e Conclusões

Neste último capítulo é efectuada a apresentação dos resultados, é realizada uma síntese do trabalho realizado e são apresentadas algumas propostas de trabalho futuro.

6.1 Resultados

A realização dos testes teve a seguinte configuração de rede:

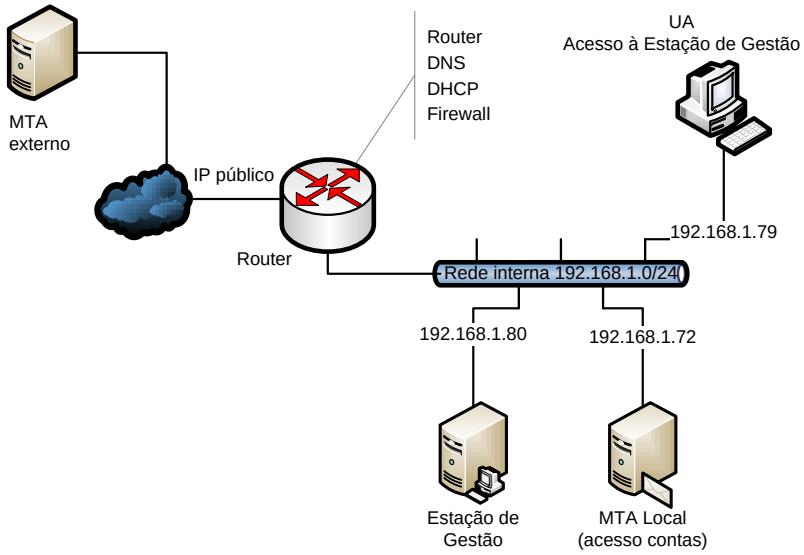

Figura 6.1: Configuração da rede

A configuração base é a de uma Intranet ligada à Internet através de um *Router*. Nesta configuração o ISP (*Internet Service Provider*) atribui um único IP público dinâmico por ligação, pelo que, para permitir o acesso simultâneo de vários *hosts* torna-se necessário o recurso ao NAT (*Network Address Translation*). Este é implementado pelo *Router*, como também implementa uma *firewall* e outros serviços de rede. Embora esta configuração não tenha um MTA *relay*, por motivos de segurança este deve ser incluído para evitar a troca directa de mensagens entre os MTAs externos e o local.

Tanto a implementação como os testes foram realizados com recurso às máquinas virtuais, para o caso escolheu-se o software VirtualBox. Foram criadas duas máquinas virtuais: uma para o servidor de *e-mail* (MTA local, que inclui também as contas dos utilizadores) e outra para a estação de gestão. Sendo o UA a máquina física (*host*).

Na tabela 6.1 são apresentadas as versões de software utilizadas na solução concretizada como nos testes.

Nome	**Descrição**	**Versão**
Postfix	MTA	`2.5.5-1.1`
ZABBIX	Plataforma de gestão	`1.6.4`
Net-SNMP	Agente SNMP	`5.4.1`
amavisd-new	Sistema de filtragem	`2.6.2`
p0f (*Passive OS Fingerprinting Tool*	Detecção do sistema operativo	`2.0.8-1`
Sun VirtualBox	Software de virtualização	`2.2.4 r47978`
Debian GNU/Linux	Sistema operativo	`5.0.1 (lenny)`
Kernel	-	`2.6.26-2-686`

Tabela 6.1: Versões de software utilizadas na solução

De forma a gerar tráfego de mensagens de *e-mail*, fez-se um *script* em Python, este pode ser consultado no anexo B, para enviar diversos tipos de mensagens, em tempo e quantidades aleatórias. Foram definidos 10 tipos de mensagens diferentes, cada tipo de mensagem é enviada no mínimo 10 vezes e no máximo 20 vezes consecutivamente, e depois do envio de cada conjunto de mensagens é feita uma pausa aleatória que varia de 10 a 60 segundos. Para aproximar estes testes de um cenário real, fez-se outro *script* semelhante ao anterior, mas aqui apenas são enviadas mensagens "normais", de modo que o sistema de filtragem deixe passar este tipo de mensagens.

6.1.1 Na Ferramenta Net-SNMP

De seguida são apresentados alguns dos resultados obtidos, após a execução do comando `snmpwalk` (que corresponde a sucessivos `snmpgetnext`). Primeiro são apresentados os resultados de cada ramo e depois é realizada uma pequena descrição dos mesmos.

6.1.1.1 Ramo mailSrvInfo(1)

```
-------------------------------------------------------------------------------
MAILSERVER-MIB::infoDescr.0 = STRING: MTA - Postfix v2.5.5-1.1
MAILSERVER-MIB::infoUpTime.0 = Timeticks: (1917500) 5:19:35.00
MAILSERVER-MIB::infoContactPostmaster.0 = STRING: sunny@debian.lan
MAILSERVER-MIB::infoServices.0 = STRING: amavisd-new, avast
MAILSERVER-MIB::infoDomainsAccept.0 = STRING: debian.lan, localhost, ee04185.selfip.com
MAILSERVER-MIB::infoDomainsRelay.0 = STRING: gmail.com, yahoo.com
MAILSERVER-MIB::infoDefaultTransport.0 = STRING: smtp
MAILSERVER-MIB::infoMailboxType.0 = STRING: Maildir/
-------------------------------------------------------------------------------
```

Neste ramo é possível verificar as informações gerais do MTA. Que inclui a versão do MTA, como o seu tempo de execução (por exemplo, depois de se reiniciar o MTA este tempo é colocado a zero). Pode-se verificar que apresenta o sistema de filtragem `amavisd-new` e `avast`. É ainda apresentada a informação relativa aos domínios, ao transporte (neste caso é o SMTP) e o formato da caixa de *e-mail* (*Maildir*).

6.1.1.2 Ramo mailSrvStatusAndCounters(2)

```
-----------------------------------------------------------------------------------------
MAILSERVER-MIB::statusConnectionsInbound.0 = Gauge32: 1
MAILSERVER-MIB::statusConnectionsOutbound.0 = Gauge32: 0
MAILSERVER-MIB::statusConnectionsMade.0 = Counter32: 129
MAILSERVER-MIB::statusConnectionsLostInbound.0 = Counter32: 0
MAILSERVER-MIB::statusConnectionsFailureOutbound.0 = Counter32: 249
MAILSERVER-MIB::statusMsgsTotalAccepted.0 = Counter32: 1131
MAILSERVER-MIB::statusMsgsTotalRejected.0 = Counter32: 0
MAILSERVER-MIB::statusMsgsTotalSent.0 = Counter32: 447
MAILSERVER-MIB::statusMsgsTotalSpam.0 = Counter32: 106
MAILSERVER-MIB::statusMsgsTotalVirus.0 = Counter32: 0
MAILSERVER-MIB::statusMsgsTotalBounced.0 = Counter32: 212
MAILSERVER-MIB::statusVolumeTotalAccepted.0 = Counter32: 2232007
MAILSERVER-MIB::statusVolumeTotalSent.0 = Counter32: 343685
MAILSERVER-MIB::statusLogFileVolume.0 = Gauge32: 148820157
MAILSERVER-MIB::statusMailSrvError.0 = Counter32: 12
MAILSERVER-MIB::statusMailSrvWarning.0 = Counter32: 0
MAILSERVER-MIB::statusMsgsTotalRecipientsIn.0 = Counter32: 0
MAILSERVER-MIB::statusMsgsTotalRecipientsOut.0 = Counter32: 0
MAILSERVER-MIB::statusSuccessfulConvertedMsgs.0 = Counter32: 0
MAILSERVER-MIB::statusFailedConvertedMsgs.0 = Counter32: 0
MAILSERVER-MIB::statusFingerprintingSpamOS.0 = STRING: none
MAILSERVER-MIB::statusFingerprintingHamOS.0 = STRING: Linux
-----------------------------------------------------------------------------------------
```

Este ramo contém diversos contadores e algumas informações sobre o estado actual das ligações e informações sobre o sistema operativo, através do p0f (*Passive OS Fingerprinting*).

No momento em que foi efectuado este pedido, o UA tinha estabelecido uma ligação SMTP com o servidor de *e-mail*, não existindo nenhuma ligação SMTP do servidor para o exterior. Também são contabilizadas todas as ligações que são efectuadas, como por exemplo o servidor já estabeleceu 129 ligações.

A discussão sobre os diferentes tipos de mensagens será feita na secção 6.1.2. O ficheiro de *log* apresenta o tamanho de 141.92 MBytes.

Os objectos `statusMsgsTotalRecipientsIn`, `statusMsgsTotalRecipientsOut`, `statusSuccessfulConvertedMsgs` e `statusFailedConvertedMsgs` não foram implementados mas o agente já está preparado para responder a esses OIDs.

Apesar de se terem contabilizadas já 106 mensagens do tipo *spam*, o objecto `statusFingerprintingSpamOS` não conseguiu determinar qual o sistema operativo que enviou estas mensagens, logo neste campo é colocado o valor *none*. O sistema operativo que envia mais mensagens não *spam* é o Linux.

6.1.1.3 Ramo mailSrvConfig(3)

```
-----------------------------------------------------------------------------------------
MAILSERVER-MIB::configTimeoutReq.0 = INTEGER: 1800 seconds
MAILSERVER-MIB::configMaxMsgSize.0 = INTEGER: 10485760 octets
MAILSERVER-MIB::configMaxRecipients.0 = INTEGER: 100
MAILSERVER-MIB::configTimeoutBeforeError.0 = INTEGER: 1 seconds
MAILSERVER-MIB::configMaxParallelDeliveries.0 = INTEGER: 20
MAILSERVER-MIB::configHopCountLimit.0 = INTEGER: 50
MAILSERVER-MIB::configMainQueueLimit.0 = INTEGER: 20000
-----------------------------------------------------------------------------------------
```

Ramo que contém algumas configurações importantes do servidor de *e-mail*, onde é possível a modificação dos seus valores.

O tempo de espera para cada pedido está definido para 30 minutos (1800 segundos), sendo este o valor padrão para o Postfix. O tamanho máximo da mensagem está configurado para 10 MBytes (10485760 octetos), o número máximo de destinatários por mensagem é de 100, o intervalo de tempo antes de enviar uma mensagem de erro é de 1 segundo, podem ocorrer 20 entregas em simultâneo, o número de saltos permitidos para uma mensagem é de 50 saltos (depois deste valor é enviada uma notificação à estação de gestão) e a fila de espera pode conter no máximo 20000 mensagens.

6.1.1.4 Ramo mailSrvConfigAgent(4)

```
-----------------------------------------------------------------------------------------
MAILSERVER-MIB::configInternalPollFreq.0 = INTEGER: 120 seconds
MAILSERVER-MIB::configEventSeverityThreshold.0 = INTEGER: normal(1)
-----------------------------------------------------------------------------------------
```

O agente actualiza os contadores de 120 em 120 segundos.

Como o `configEventSeverityThreshold` está com o valor *normal*, assim deve-se enviar todo o tipo de notificações à estação de gestão.

6.1.1.5 Ramo mailSrvControl(5)

```
-----------------------------------------------------------------------------------------
MAILSERVER-MIB::controlStatus.0 = INTEGER: running(1)
MAILSERVER-MIB::controlReload.0 = INTEGER: now(1)
MAILSERVER-MIB::controlQueueDeleteMsg.0 = STRING: none
MAILSERVER-MIB::controlQueueRequeueMsg.0 = STRING: none
-----------------------------------------------------------------------------------------
```

O objecto `controlStatus` indica qual o estado do MTA (neste caso, está em execução). Para actualizar as configurações do MTA, deve-se mudar para `1` o valor do objecto `controlReload`.

A eliminação de uma mensagem da fila de espera é conseguida pelo envio do ID da mensagem na fila para o objecto `controlQueueDeleteMsg`. A mensagem pode ser colocada outra vez na fila de espera activa, por um mecanismo semelhante ao anterior, mas o objecto em questão é `controlQueueRequeueMsg`.

6.1.1.6 Tabela statusProcessesTable

```
-----------------------------------------------------------------------------------------------
debian:/etc/snmp/agent# snmptable -v2c -c public -t 60 192.168.1.72 .1.3.6.1.4.1.33536.1.2.3
SNMP table: MAILSERVER-MIB::statusProcessesTable

    processesName processesState processesCpu processesMem processesNumSubProc
      amavisd-new        running          400 18840 KBytes                   2
 amavisd.pidsdsdw     notrunning            0     0 KBytes                   0
           master        running           18  1816 KBytes                   4
-----------------------------------------------------------------------------------------------
```

Nesta tabela são monitorizados três processos, para efeitos de testes foi adicionado o processo amavisd.pidsdsdw que não esta em execução. É possível verificar o estado de cada processo, o CPU e a memória utilizada por cada um dos processos, e o número que sub-processos que são criados pelo processo monitorizado.

O processo `master` corresponde ao processo principal do Postfix. Está a utilizar 1.8% do CPU, 1816 KBytes da memória e tem 4 sub-processos activos.

6.1.1.7 Tabela statusQueuesTable

```
-----------------------------------------------------------------------------------------------
debian:/etc/snmp/agent# snmptable -v2c -c public -t 60 192.168.1.72 .1.3.6.1.4.1.33536.1.2.4
SNMP table: MAILSERVER-MIB::statusQueuesTable

 queuesName queuesCurrMsgs queuesCurrVolume queuesMsgsIn queuesMsgsOut    queuesOldestMessageId
   incoming              0                0            0             0 incoming_out_msgID_FALTA
     active            163           342297         2812          2609   active_out_msgID_FALTA
   deferred             12            18935            0             0 deferred_out_msgID_FALTA
       hold              0                0            0             0     hold_out_msgID_FALTA
    corrupt              0                0            0             0  corrupt_out_msgID_FALTA
-----------------------------------------------------------------------------------------------
```

Nesta tabela são monitorizados 5 filas de espera. Para cada fila de espera é possível verificar o número e volume actual das mensagens presentes, o número de mensagens que entraram e saíram, e o ID da mensagem mais antiga na fila. Este último objecto (`queuesOldestMessageId`) não foi implementado.

6.1.2 Na Plataforma de Gestão

Agora são apresentadas algumas imagens da interface da plataforma do gestão. De notar que apenas se apresenta algumas funcionalidades da plataforma ZABBIX, uma consulta mais detalhada de todas as funcionalidades pode ser feita em [28].

6.1.2.1 Lista dos Objectos Monitorizados

Na figura seguinte pode-se visualizar a lista dos últimos valores actualizados. É possível verificar a data de actualização, o seu valor actual e a diferença do valor actual e o último valor guardado. Na última coluna é incluído um *link* que permite criar o gráfico com os valores que estão armazenados na base de dados, ou simplesmente verificar esses mesmos valores num formato textual.

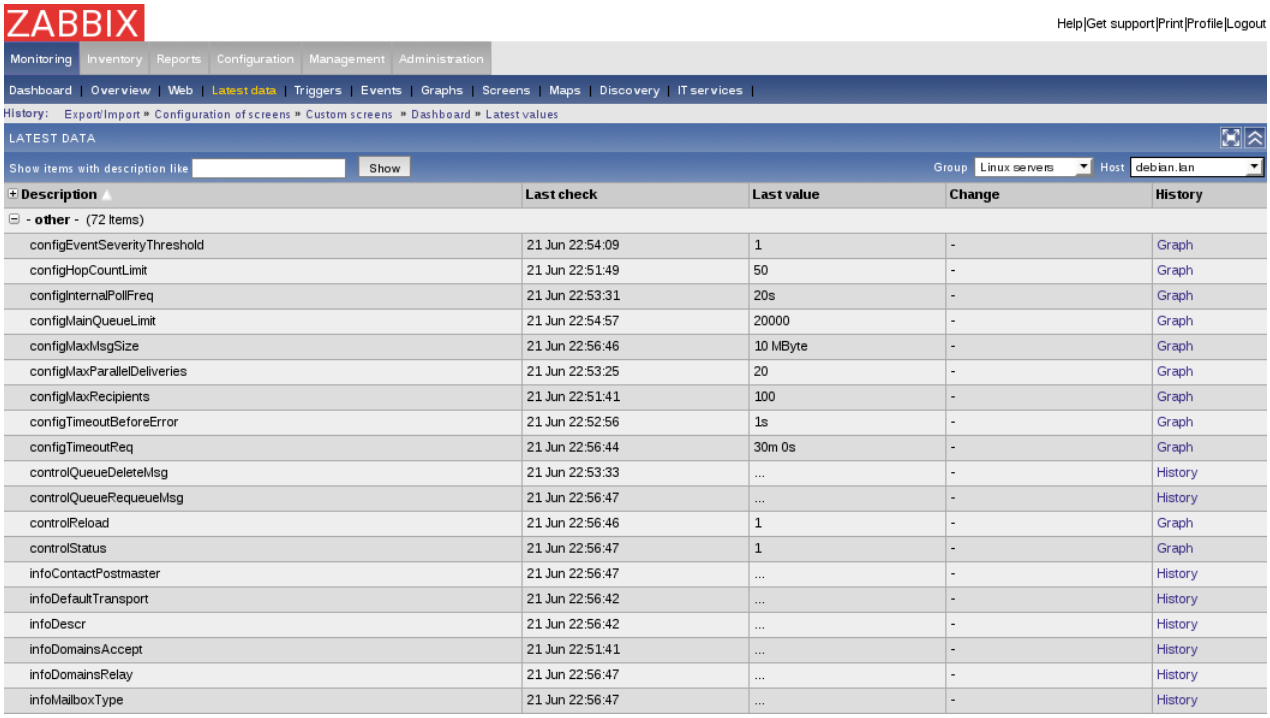

Figura 6.2: Últimos valores actualizados

6.1.2.2 Tabelas

Os valores do objectos das tabelas (processos e filas de espera) podem ser visualizados num formato de tabela. Pode-se definir quantos valores é que se pretende apresentar, no exemplo estão os últimos cinco valores para os processos e dois valores para as filas de espera.

Na tabela dos processos (figura 6.3), pode-se observar as mesmas informações que foram verificadas anteriormente, igualmente o segundo processo aparece com o estado *Down*. É notório a variação da utilização do CPU com a variação do número de sub-processos.

Na tabela das filas de espera (figura 6.4), com os objectos `queuesMsgsIn` e `queuesMsgsOut` é possível criar gráficos de débito de entrada e saída das mensagens das filas de espera, o processo como estes são criados é explicado mais à frente. Por exemplo a fila de espera `active`, na última leitura tem 180 mensagens que em termos de volume corresponde a 370900 Bytes.

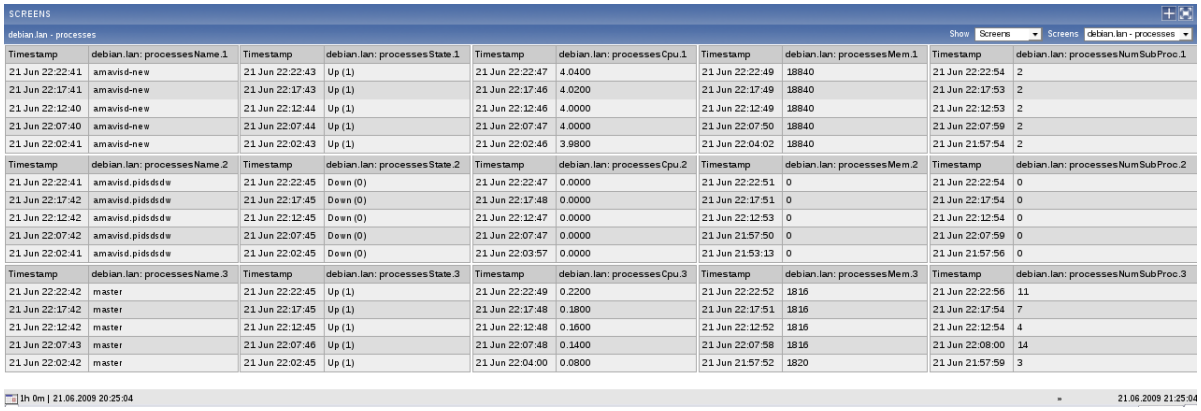

Figura 6.3: Tabela dos processos monitorizados

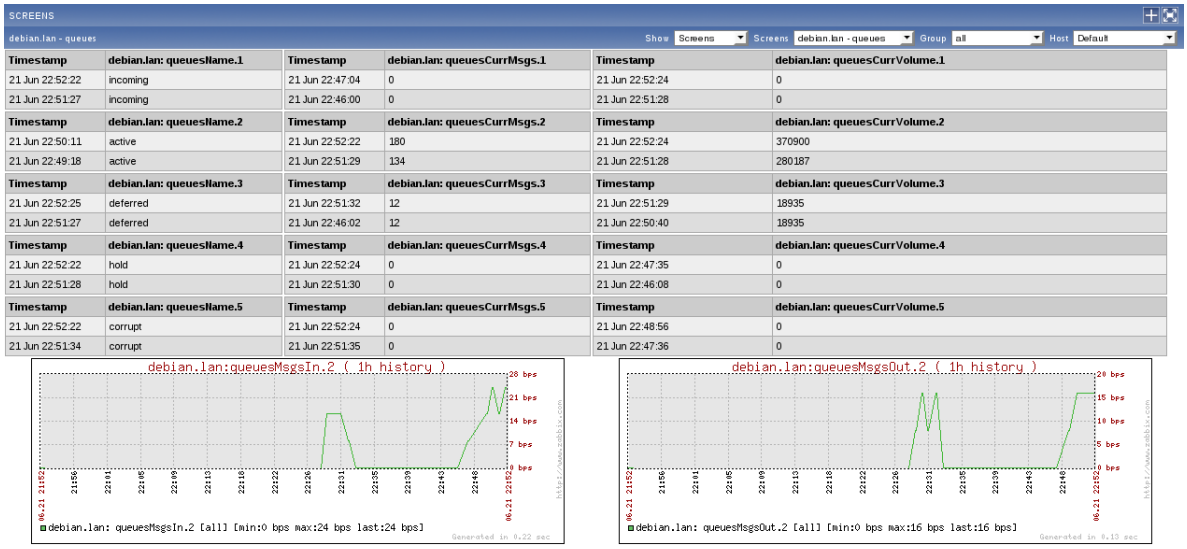

Figura 6.4: Tabela das filas de espera monitorizadas

6.1.2.3 Gráficos

Os gráficos podem ser criados com diferentes escalas temporais e diferentes formatos. Em relação à escala temporal, podem ser criados gráficos com intervalo de amostragem definido pelo utilizador. Nos dois gráficos seguintes estão presentes os diferentes tipos de contadores de mensagens apresentados na MIB.

No primeiro gráfico é possível verificar a variação dos diferentes tipos de mensagens que são recebidas pelo servidor de *e-mail*. A descontinuidade de alguns valores deve-se ao facto de se ter desactivado aqueles objectos durante aquele período.

No segundo gráfico pode-se visualizar uma informação estatística dos diferentes tipos de mensagens. Quase 57.16% corresponde às mensagens que são aceites pelo servidor e 21.72% representam as mensagens que são enviadas para outros servidores. Neste cenário, as mensagens foram enviadas para os seguintes servidores públicos: gmail.com e yahoo.com. Os 11.23% representam as mensagens que foram devolvidas, isto deve-se ao facto de o servidor da yahoo.com não permitir a recepção de mensagens de servidores que tenham o seu endereço IP inserido numa gama de IPs dinâmicos. Cerca de 9.89% das mensagens registadas são *spam*. Não tendo sido registada nenhuma mensagem de vírus, nem nenhuma mensagem foi rejeitada.

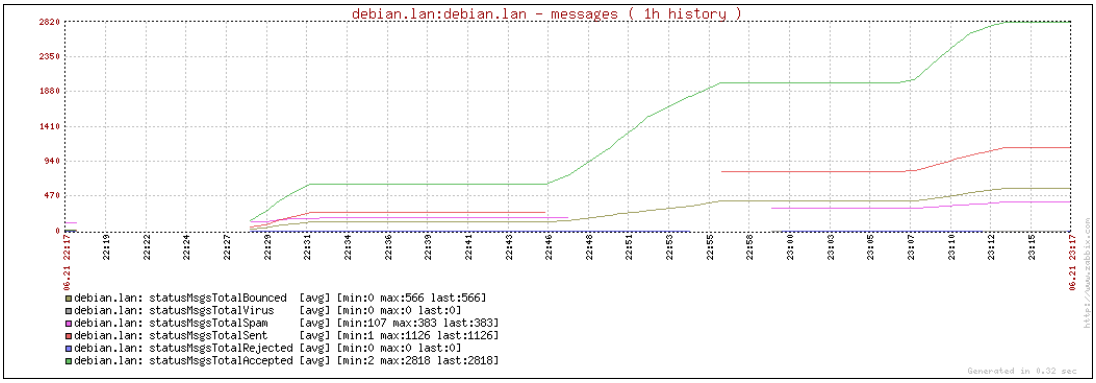

Figura 6.5: Mensagens processadas no período de amostragem 1 hora

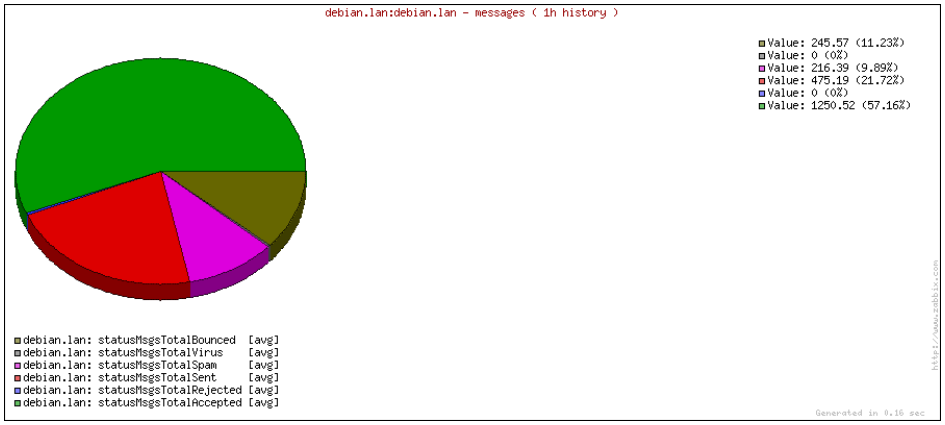

Figura 6.6: Estatística das mensagens processadas

6.1.2.4 Débito

Com os valores dos contadores de volume recebido e enviado pode-se criar gráfico com o débito das mensagens recebidas e enviadas.

O ZABBIX aplica a fórmula 6.1 para obter a variação das últimas duas amostras em função do intervalo de tempo entre essas mesmas amostras.

$$bps: \quad \frac{Valor_{actual} - Valor_{anterior}}{Tempo_{actual} - Tempo_{anterior}} * 8 \tag{6.1}$$

Cada valor deve estar em Bytes, o tempo em segundos e o resultado é apresentado em bps (*bit per second*).

No gráfico seguinte pode-se verificar que o volume das mensagens recebidas é maior do que o volume das mensagens que são enviadas.

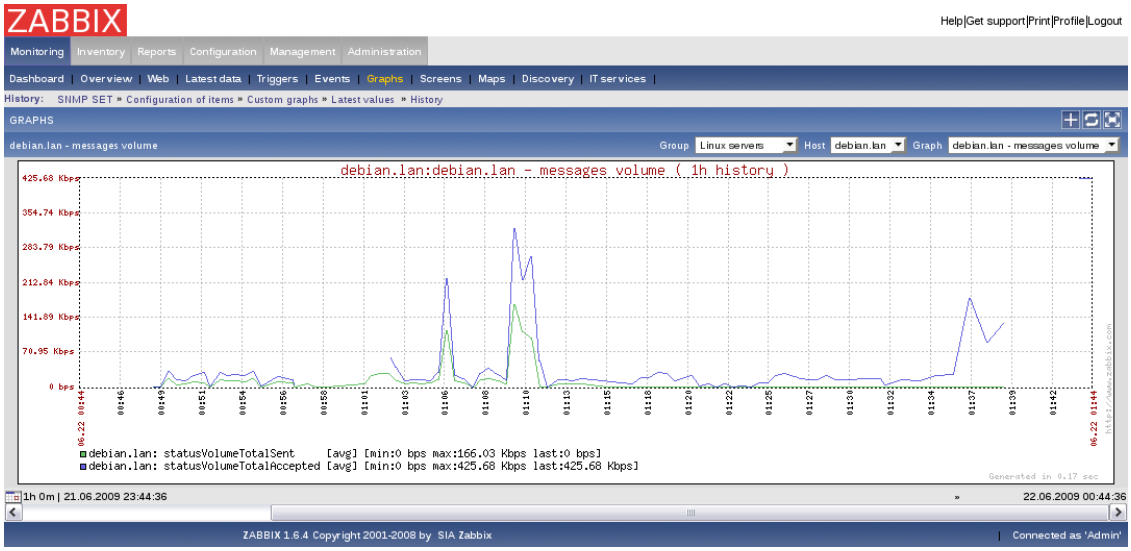

Figura 6.7: Volume das mensagens recebidas e enviadas

6.1.2.5 Notificações Recebidas

Na figura seguinte estão presentes todos os *traps* recebidos pela estação de gestão. Por exemplo na última notificação recebida, em primeiro temos a indicação do tempo (data e hora) da recepção, depois segue-se um conjunto de valores que inclui: o instante de tempo do funcionamento do MTA em que a notificação foi gerada (2 minutos e 3 segundos), a gravidade da notificação (*warning*), e por último a sua descrição (o ficheiro de configuração main.cf foi modificado).

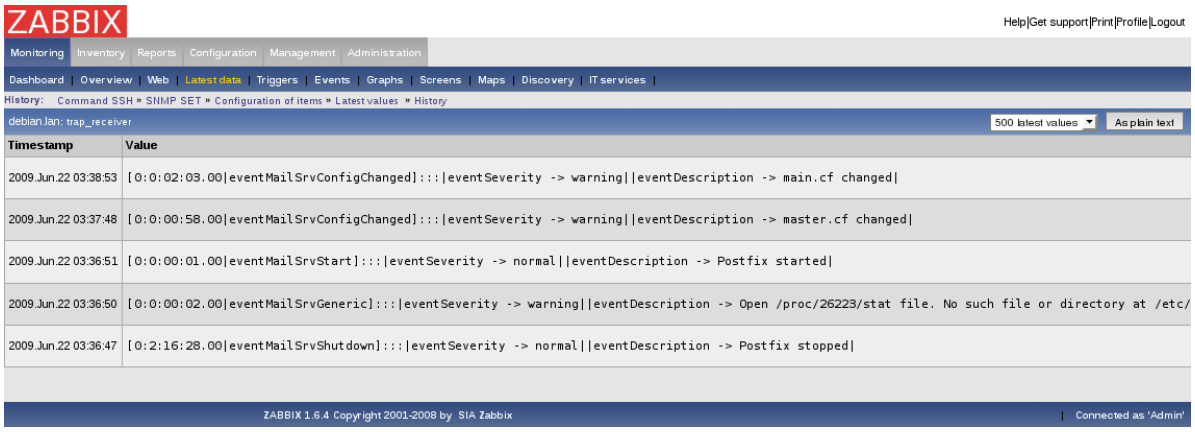

Figura 6.8: Notificações recebidas pela estação de gestão

6.1.2.6 Operações SNMP-SET

A operação SNMP-SET permite modificar os valores dos objectos do agente gerido. Para permitir este tipo de operações, foi criada uma nova secção na barra de menu do ZABBIX, que foi designada de `Management`. Dentro desta secção foram definidas duas sub-secções: `SNMP SET` e `Command SSH`, a primeira permite modificar o valor dos objectos presentes no agente e a segunda permite executar comandos no agente através do protocolo SSH.

Figura 6.9: Barra de menu do ZABBIX

De seguida é apresentado um cenário de utilização destas novas funcionalidades introduzidas no ZABBIX.

A lista das mensagens presentes nas filas de espera pode ser obtida pelo comando: `postqueue -p`, para executar este comando é necessário definir o utilizador e a sua senha no sistema.

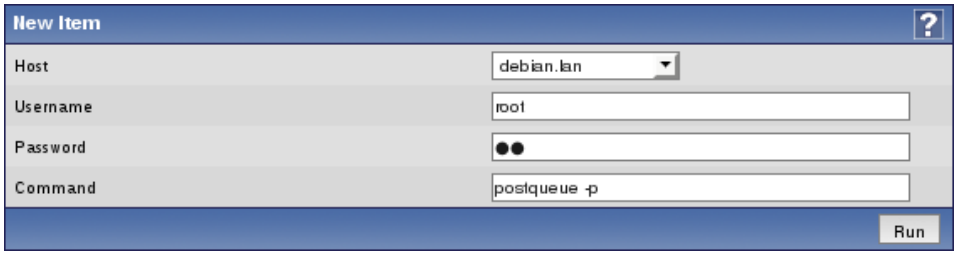

Figura 6.10: Comando SSH - Pedido

A resposta ao comando SSH foi a seguinte:

Figura 6.11: Comando SSH - Resposta

É possível verificar que existem duas mensagens na fila de espera, por exemplo a segunda mensagem que tem o ID `8FC0744CA` não foi entregue devido ao endereço IP deste servidor de *e-mail* se encontrar na lista negra, tal como é dito na descrição do erro.

De seguida vai-se enviar novamente esta mensagem, para tal é necessário mudar o valor do objecto `controlQueueRequeueMsg`, sendo primeiro necessário definir um novo objecto associado à operação SET.

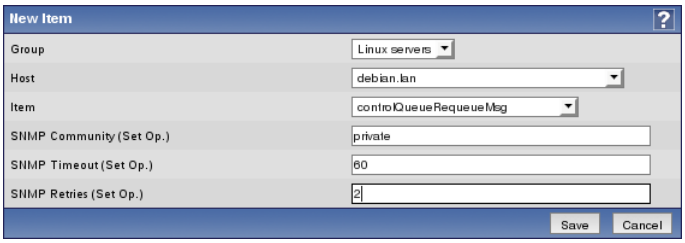

Figura 6.12: Operação SNMP-SET - Novo objecto

A operação foi efectuada com sucesso e o valor na base de dados da estação de gestão será actualizada dentro de 19 segundos:

Figura 6.13: Operação SNMP-SET - Operação efectuada com sucesso

6.1.2.7 *Triggers*

Os *triggers* permitem monitorizar os valores recebidos na estação de gestão. Se ultrapassada uma condição é gerado um evento configurado pelo utilizador.

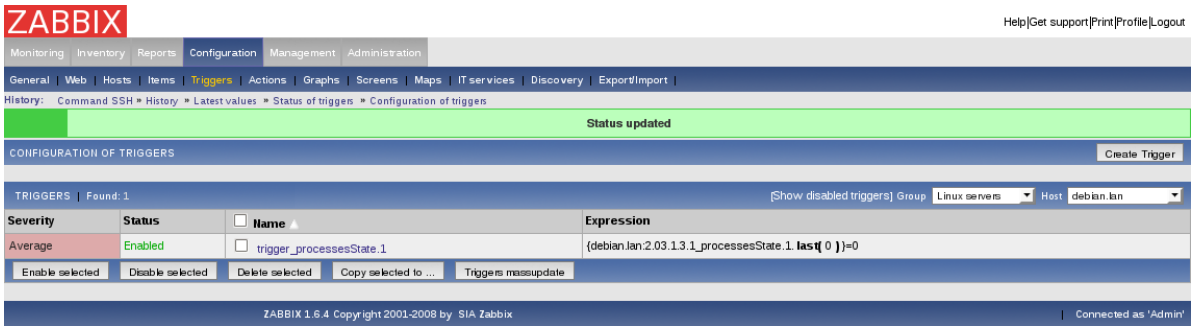

Figura 6.14: *Trigger* adicionado

No exemplo a condição definida foi: se o objecto (neste caso o estado do processo 1) mudar para o valor 0, então é emitido um aviso.

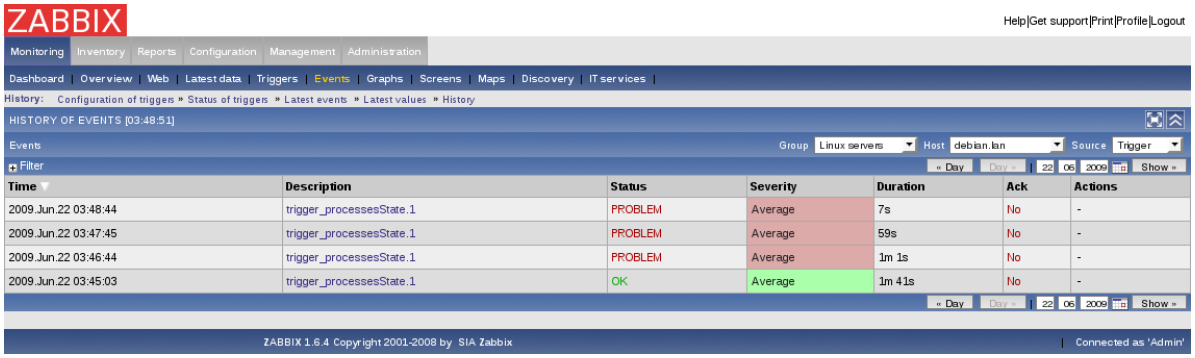

Figura 6.15: *Trigger* activado

Como se pode observar na figura anterior, o *trigger* foi activado três vezes. Pode-se mudar o modo como o trigger funciona depois de activado uma vez.

6.1.2.8 Monitorização de Serviços

O ZABBIX permite a monitorização de serviços, que pode ser utilizado para monitorizar o serviço de SMTP.

A monitorização pode ser realizada de dois modos diferentes: verificar apenas a disponibilidade, onde o resultado é 0 se o serviço não responder ou 1 se o serviço estiver a responder correctamente; ou verificar a disponibilidade e quantificar o tempo entre o pedido e o envio da resposta do servidor. Este último caso, está ilustrado na figura seguinte.

debian.lan: smtp_service_check_time

Timestamp	Value
2009.Jul.20 03:06:07	0.0062
2009.Jul.20 03:05:36	0.0071
2009.Jul.20 03:05:06	0.0024
2009.Jul.20 03:04:36	0.0035
2009.Jul.20 03:04:06	0.0013
2009.Jul.20 03:03:36	0.0016
2009.Jul.20 03:03:16	0.0014

Figura 6.16: Monitorização de serviços - SMTP

Na primeira coluna é possível verificar a data em que o teste foi realizado e na segunda coluna é colocado o intervalo de tempo, em segundos.

Este mecanismo de teste pode ser utilizado para monitorizar o serviço de DNS.

6.2 Conclusões Finais

O trabalho realizado demonstra que é possível fazer a monitorização e a gestão dos servidores de *e-mail* com o protocolo SNMP, existindo algumas limitações que o protocolo já prevê. É o caso da implementação do agente não ser universal, mas é por esta razão é que existem as MIBs para ultrapassar essas limitações.

A fase inicial do trabalho focou-se no estudo do funcionamento dos servidores de *e-mail*, com o objectivo de determinar os seus requisitos de desempenho e identificar os problemas associados à sua gestão.

Numa fase seguinte foi efectuada uma análise de diferentes ferramentas que pudessem resolver os problemas identificados.

Criou-se um modelo de um servidor de *e-mail* na tentativa de encontrar os pontos críticos associados à sua gestão.

Depois de encontrados os pontos críticos, foi efectuada uma análise às MIBs existentes, da qual construiu-se uma MIB nova com base nas existentes.

Por último foi efectuada a implementação do protótipo, tendo-se comprovando que é possível efectuar a monitorização e a gestão dos servidores de *e-mail* com recurso ao protocolo SNMP.

Culminou com a criação de algumas funcionalidades para a plataforma de gestão ZABBIX, estas podem ser consideradas como *add-on* para esta plataforma. De modo a facilitar a adição de novos servidores de *e-mail* no ZABBIX, criou-se um *template* com todos os objectos definidos na MAILSERVER-MIB.

6.3 Trabalho Futuro

Algumas propostas de trabalho futuro:

- A monitorização e controlo de acessos dos clientes. Visto que este problema foi identificado, mas na solução apresentada não foi resolvido;
- A gestão dos utilizadores e das suas caixas de *e-mail*, como também a gestão dos diversos sistemas de filtragem e das listas de distribuição, integrada no contexto de gestão por SNMP. Deve-se apenas incluir algumas informações relevantes, pois já existem ferramentas *Web* que permitem a sua gestão;
- A implementação do agente para outros servidores de *e-mail*.

Anexo A

MAILSERVER-MIB

De seguida segue a versão completa da MIB implementada. Esta satisfaz todas as regras da estrutura SMIv2 e automaticamente pode ser convertível na versão SMIv1.

```
MAILSERVER-MIB DEFINITIONS ::= BEGIN

-- { IMPORTS
    IMPORTS
    OBJECT-TYPE, MODULE-IDENTITY, OBJECT-IDENTITY, NOTIFICATION-TYPE,
    enterprises, Counter32, TimeTicks, Integer32, Gauge32
        FROM SNMPv2-SMI

    DisplayString
        FROM SNMPv2-TC

    OBJECT-GROUP, MODULE-COMPLIANCE, NOTIFICATION-GROUP
        FROM SNMPv2-CONF;
--} end of IMPORTS

-- { MODULE-IDENTITY statement
    inescPorto MODULE-IDENTITY
        LAST-UPDATED            "200904241600Z"     -- YYYY MM DD HH MM Z
        ORGANIZATION            "http://www.fe.up.pt/"
        CONTACT-INFO            "Sunny Narendra
                                 E-mail:  ee04185@fe.up.pt
                                 Web: http://www.fe.up.pt/~ee04185/"
        DESCRIPTION             "The MIB module describing
                                 Message Transfer Agents (MTAs)."
        -- List of versions
        REVISION                "200904241600Z"
        DESCRIPTION             "The fist version of the MAILSERVER-MIB."
    ::= { enterprises 33536 }
    mailserver                  OBJECT IDENTIFIER ::= { inescPorto 1 }
-- } end of MODULE-IDENTITY statement

-- { TEXTUAL-CONVENTION definitions

-- } end of TEXTUAL-CONVENTION definitions

-- { Node definitions
    mailSrvInfo                 OBJECT IDENTIFIER ::= { mailserver 1 }
    mailSrvStatusAndCounters    OBJECT IDENTIFIER ::= { mailserver 2 }
    mailSrvConfig               OBJECT IDENTIFIER ::= { mailserver 3 }
    mailSrvConfigAgent          OBJECT IDENTIFIER ::= { mailserver 4 }
    mailSrvControl              OBJECT IDENTIFIER ::= { mailserver 5 }
    mailSrvEvents               OBJECT IDENTIFIER ::= { mailserver 6 }
    mailSrvConformance          OBJECT IDENTIFIER ::= { mailserver 7 }
-- } end of Node definitions

```

```
-- { Objects (Scalar and tables)

    -- { mailSrvInfo(1)

    -- General information about the MTA and
    -- static configuration data.

    infoDescr                   OBJECT-TYPE
        SYNTAX                  DisplayString
        MAX-ACCESS              read-only
        STATUS                  current
        DESCRIPTION
            "A textual description of the MTA. This value should include
            the full name and version identification of the MTA system.
            It is mandatory that this only contain printable ASCII characters."
        ::= { mailSrvInfo 1 }

    infoUpTime                  OBJECT-TYPE
        SYNTAX                  TimeTicks
        MAX-ACCESS              read-only
        STATUS                  current
        DESCRIPTION
            "The time (in hundredths of a second) since the
            MTA was last re-initialized."
        ::= { mailSrvInfo 2 }

    infoContactPostmaster       OBJECT-TYPE
        SYNTAX                  DisplayString
        MAX-ACCESS              read-only
        STATUS                  current
        DESCRIPTION
            "The textual identification of the contact person for this
            managed MTA, together with information on how to contact
            this person."
        ::= { mailSrvInfo 3 }

    infoServices                OBJECT-TYPE
        SYNTAX                  DisplayString
        MAX-ACCESS              read-only
        STATUS                  current
        DESCRIPTION
            "The textual description of services that
            MTA offers. For example:
                - Content filter, Anti-virus, Anti-spam"
        ::= { mailSrvInfo 4 }

    infoDomainsAccept           OBJECT-TYPE
        SYNTAX                  DisplayString
        MAX-ACCESS              read-only
        STATUS                  current
        DESCRIPTION
            "What domains this MTA will deliver locally, instead of
            forwarding to another MTA."
        ::= { mailSrvInfo 5 }

    infoDomainsRelay            OBJECT-TYPE
        SYNTAX                  DisplayString
        MAX-ACCESS              read-only
        STATUS                  current
        DESCRIPTION
            "What domains this MTA will forward to another MTA."
        ::= { mailSrvInfo 6 }

    infoDefaultTransport        OBJECT-TYPE
        SYNTAX                  DisplayString
        MAX-ACCESS              read-only
        STATUS                  current
        DESCRIPTION
            "The default mail delivery transport."
        ::= { mailSrvInfo 7 }

    infoMailboxType             OBJECT-TYPE
        SYNTAX                  DisplayString
        MAX-ACCESS              read-only
        STATUS                  current
        DESCRIPTION
            "The current type of mail storage."
```

```
123         ::= { mailSrvInfo 8 }
124     -- } end of mailSrvInfo(1)
125
126     -- { mailSrvStatusAndCounters(2)
127
128     -- Provides state information, counters and dynamic configuration data.
129
130     statusConnectionsInbound    OBJECT-TYPE
131         SYNTAX                  Gauge32
132         MAX-ACCESS              read-only
133         STATUS                  current
134         DESCRIPTION
135             "The number of inbound SMTP requests that MTA is
136             currently processing. Default port is 25."
137         ::= { mailSrvStatusAndCounters 1 }
138
139     statusConnectionsOutbound   OBJECT-TYPE
140         SYNTAX                  Gauge32
141         MAX-ACCESS              read-only
142         STATUS                  current
143         DESCRIPTION
144             "The number of outbound SMTP requests that MTA is
145             currently processing. Default port is 25."
146         ::= { mailSrvStatusAndCounters 2 }
147
148     -- table start ((
149     statusProcessesTable        OBJECT-TYPE
150         SYNTAX                  SEQUENCE OF StatusProcessesEntry
151         MAX-ACCESS              not-accessible
152         STATUS                  current
153         DESCRIPTION
154             "Processes table."
155         ::= { mailSrvStatusAndCounters 3 }
156
157         statusProcessesEntry    OBJECT-TYPE
158           SYNTAX                StatusProcessesEntry
159           MAX-ACCESS            not-accessible
160           STATUS                current
161           DESCRIPTION
162             "An entry in statusProcessesTable."
163           INDEX                 { processesIndex }
164           ::= { statusProcessesTable 1 }
165
166         StatusProcessesEntry ::= SEQUENCE {
167           processesIndex        Integer32,
168           processesName         DisplayString,
169           processesState        INTEGER,
170           processesCpu          Gauge32,
171           processesMem          Gauge32,
172           processesNumSubProc   Gauge32
173           }
174
175         processesIndex          OBJECT-TYPE
176           SYNTAX                Integer32 (0..2147483647)
177           MAX-ACCESS            not-accessible
178           STATUS                current
179           DESCRIPTION
180             "Processes table index."
181           ::= { statusProcessesEntry 1 }
182
183         processesName           OBJECT-TYPE
184           SYNTAX                DisplayString
185           MAX-ACCESS            read-only
186           STATUS                current
187           DESCRIPTION
188             "The textual identification of this process."
189           ::= { statusProcessesEntry 2 }
190
191         processesState          OBJECT-TYPE
192           SYNTAX                INTEGER { notrunning(0), running(1) }
193           MAX-ACCESS            read-only
194           STATUS                current
195           DESCRIPTION
196             "The current state of this process.
197             Values: not_running(0), running(1)."
198           ::= { statusProcessesEntry 3 }
199
```

```
200         processesCpu            OBJECT-TYPE
201           SYNTAX                Gauge32
202           MAX-ACCESS            read-only
203           STATUS                current
204           DESCRIPTION
205             "The number of centi-seconds of the total system's CPU
206             resources consumed by this process.  Note that on a
207             multi-processor system, this value may increment by
208             more than one centi-second in one centi-second of real
209             (wall clock) time. Same as hrSWRunPerfCPU (RFC 1514)."
210           ::= { statusProcessesEntry 4 }
211
212         processesMem            OBJECT-TYPE
213           SYNTAX                Gauge32
214           UNITS                 "KBytes"
215           MAX-ACCESS            read-only
216           STATUS                current
217           DESCRIPTION
218             "The total amount of real system memory allocated to
219             this process. Same as hrSWRunPerfMem (RFC 1514)."
220           ::= { statusProcessesEntry 5 }
221
222         processesNumSubProc     OBJECT-TYPE
223           SYNTAX                Gauge32
224           MAX-ACCESS            read-only
225           STATUS                current
226           DESCRIPTION
227             "The total number of sub processes created by this process
228             i.e., number of child processes."
229           ::= { statusProcessesEntry 6 }
230       -- )) end table
231
232       -- table start ((
233       statusQueuesTable           OBJECT-TYPE
234           SYNTAX                  SEQUENCE OF StatusQueuesEntry
235           MAX-ACCESS              not-accessible
236           STATUS                  current
237           DESCRIPTION
238               "Queues table."
239           ::= { mailSrvStatusAndCounters 4 }
240
241         statusQueuesEntry       OBJECT-TYPE
242           SYNTAX                StatusQueuesEntry
243           MAX-ACCESS            not-accessible
244           STATUS                current
245           DESCRIPTION
246             "An entry in statusQueuesTable."
247           INDEX                 { queuesIndex }
248           ::= { statusQueuesTable 1 }
249
250         StatusQueuesEntry   ::= SEQUENCE {
251           queuesIndex           Integer32,
252           queuesName            DisplayString,
253           queuesCurrMsgs        Gauge32,
254           queuesCurrVolume      Gauge32,
255           queuesMsgsIn          Counter32,
256           queuesMsgsOut         Counter32,
257           queuesOldestMessageId DisplayString
258           }
259
260         queuesIndex             OBJECT-TYPE
261           SYNTAX                Integer32 (0..2147483647)
262           MAX-ACCESS            not-accessible
263           STATUS                current
264           DESCRIPTION
265             "Queues table index."
266           ::= { statusQueuesEntry 1 }
267
268         queuesName              OBJECT-TYPE
269           SYNTAX                DisplayString
270           MAX-ACCESS            read-only
271           STATUS                current
272           DESCRIPTION
273             "The textual identification of this queue."
274           ::= { statusQueuesEntry 2 }
275
276         queuesCurrMsgs          OBJECT-TYPE
```

```
277           SYNTAX                Gauge32
278           MAX-ACCESS            read-only
279           STATUS                current
280           DESCRIPTION
281             "The total number of messages currently stored in this queue."
282           ::= { statusQueuesEntry 3 }
283
284         queuesCurrVolume        OBJECT-TYPE
285           SYNTAX                Gauge32
286           MAX-ACCESS            read-only
287           STATUS                current
288           DESCRIPTION
289             "The total volume of messages currently stored in this queue."
290           ::= { statusQueuesEntry 4 }
291
292         queuesMsgsIn            OBJECT-TYPE
293           SYNTAX                Counter32
294           MAX-ACCESS            read-only
295           STATUS                current
296           DESCRIPTION
297             "The total number of messages that enter in this queue
298             since MTA initialization."
299           ::= { statusQueuesEntry 5 }
300
301         queuesMsgsOut           OBJECT-TYPE
302           SYNTAX                Counter32
303           MAX-ACCESS            read-only
304           STATUS                current
305           DESCRIPTION
306             "The total number of messages that leave this queue
307             since MTA initialization."
308           ::= { statusQueuesEntry 6 }
309
310         queuesOldestMessageId   OBJECT-TYPE
311           SYNTAX                DisplayString
312           MAX-ACCESS            read-only
313           STATUS                current
314           DESCRIPTION
315             "Message ID of the oldest message in this queue."
316           ::= { statusQueuesEntry 7 }
317       -- )) end table
318
319       statusConnectionsMade       OBJECT-TYPE
320           SYNTAX                  Counter32
321           MAX-ACCESS              read-only
322           STATUS                  current
323           DESCRIPTION
324               "The total number of connections (SMTP) made
325               since MTA initialization."
326           ::= { mailSrvStatusAndCounters 5 }
327
328       statusConnectionsLostInbound OBJECT-TYPE
329           SYNTAX                  Counter32
330           MAX-ACCESS              read-only
331           STATUS                  current
332           DESCRIPTION
333               "The total number of inbound connections (SMTP) lost
334               since MTA initialization."
335           ::= { mailSrvStatusAndCounters 6 }
336
337       statusConnectionsFailureOutbound OBJECT-TYPE
338           SYNTAX                  Counter32
339           MAX-ACCESS              read-only
340           STATUS                  current
341           DESCRIPTION
342               "The total number of outbound connections (SMTP) failed
343               since MTA initialization."
344           ::= { mailSrvStatusAndCounters 7 }
345
346       statusMsgsTotalAccepted     OBJECT-TYPE
347           SYNTAX                  Counter32
348           MAX-ACCESS              read-only
349           STATUS                  current
350           DESCRIPTION
351               "The total number of messages accepted
352               since MTA initialization."
353           ::= { mailSrvStatusAndCounters 8 }
```

```
354
355     statusMsgsTotalRejected     OBJECT-TYPE
356         SYNTAX                  Counter32
357         MAX-ACCESS              read-only
358         STATUS                  current
359         DESCRIPTION
360             "The total number of messages rejected
361             since MTA initialization."
362         ::= { mailSrvStatusAndCounters 9 }
363
364     statusMsgsTotalSent         OBJECT-TYPE
365         SYNTAX                  Counter32
366         MAX-ACCESS              read-only
367         STATUS                  current
368         DESCRIPTION
369             "The total number of messages sent
370             since MTA initialization."
371         ::= { mailSrvStatusAndCounters 10 }
372
373     statusMsgsTotalSpam         OBJECT-TYPE
374         SYNTAX                  Counter32
375         MAX-ACCESS              read-only
376         STATUS                  current
377         DESCRIPTION
378             "The total number of messages marked as spam
379             since MTA initialization. This can be done with external
380             filter."
381         ::= { mailSrvStatusAndCounters 11 }
382
383     statusMsgsTotalVirus        OBJECT-TYPE
384         SYNTAX                  Counter32
385         MAX-ACCESS              read-only
386         STATUS                  current
387         DESCRIPTION
388             "The total number of messages marked as virus
389             since MTA initialization. This can be done with external
390             filter."
391         ::= { mailSrvStatusAndCounters 12 }
392
393     statusMsgsTotalBounced      OBJECT-TYPE
394         SYNTAX                  Counter32
395         MAX-ACCESS              read-only
396         STATUS                  current
397         DESCRIPTION
398             "The total number of messages that a bounce message
399             (or Delivery Status Notification (DSN),
400             Non-Delivery Report/Receipt (NDR),
401             Non-Delivery Notification (NDN) message)
402             received since MTA initialization."
403         ::= { mailSrvStatusAndCounters 13 }
404
405     statusVolumeTotalAccepted   OBJECT-TYPE
406         SYNTAX                  Counter32
407         MAX-ACCESS              read-only
408         STATUS                  current
409         DESCRIPTION
410             "The total volume of messages accepted since MTA
411             initialization, measured in octets."
412         ::= { mailSrvStatusAndCounters 14 }
413
414     statusVolumeTotalSent       OBJECT-TYPE
415         SYNTAX                  Counter32
416         MAX-ACCESS              read-only
417         STATUS                  current
418         DESCRIPTION
419             "The total volume of messages sent since MTA
420             initialization, measured in octets."
421         ::= { mailSrvStatusAndCounters 15 }
422
423     statusLogFileVolume         OBJECT-TYPE
424         SYNTAX                  Gauge32
425         MAX-ACCESS              read-only
426         STATUS                  current
427         DESCRIPTION
428             "The total volume of the log file, measured in octets."
429         ::= { mailSrvStatusAndCounters 16 }
430
```

```
431     statusMailSrvError          OBJECT-TYPE
432         SYNTAX                  Counter32
433         MAX-ACCESS              read-only
434         STATUS                  current
435         DESCRIPTION
436             "The total number of error occurred
437             since MTA initialization."
438         ::= { mailSrvStatusAndCounters 17 }
439
440     statusMailSrvWarning        OBJECT-TYPE
441         SYNTAX                  Counter32
442         MAX-ACCESS              read-only
443         STATUS                  current
444         DESCRIPTION
445             "The total number of warning occurred
446             since MTA initialization."
447         ::= { mailSrvStatusAndCounters 18 }
448
449     statusMsgsTotalRecipientsIn OBJECT-TYPE
450         SYNTAX                  Counter32
451         MAX-ACCESS              read-only
452         STATUS                  current
453         DESCRIPTION
454             "The total number of recipients specified in all messages
455             received since MTA initialization."
456         ::= { mailSrvStatusAndCounters 19 }
457
458     statusMsgsTotalRecipientsOut OBJECT-TYPE
459         SYNTAX                  Counter32
460         MAX-ACCESS              read-only
461         STATUS                  current
462         DESCRIPTION
463             "The total number of recipients specified in all messages
464             transmitted since MTA initialization."
465         ::= { mailSrvStatusAndCounters 20 }
466
467     statusSuccessfulConvertedMsgs OBJECT-TYPE
468         SYNTAX                  Counter32
469         MAX-ACCESS              read-only
470         STATUS                  current
471         DESCRIPTION
472             "The number of messages that have been successfully
473             converted from one form to another since MTA
474             initialization."
475         ::= { mailSrvStatusAndCounters 21 }
476
477     statusFailedConvertedMsgs   OBJECT-TYPE
478         SYNTAX                  Counter32
479         MAX-ACCESS              read-only
480         STATUS                  current
481         DESCRIPTION
482             "The number of messages for which an unsuccessful
483              attempt was made to convert them from one form to
484              another since MTA initialization."
485         ::= { mailSrvStatusAndCounters 22 }
486
487     statusFingerprintingSpamOS  OBJECT-TYPE
488         SYNTAX                  DisplayString
489         MAX-ACCESS              read-only
490         STATUS                  current
491         DESCRIPTION
492             "Passive OS Fingerprinting attempts to discover what type of
493             system a message is sent from. This value should be the
494             name/version of the OS that sent most spam messages, or
495             'none' if there are no spam messages received."
496         ::= { mailSrvStatusAndCounters 23 }
497
498     statusFingerprintingHamOS   OBJECT-TYPE
499         SYNTAX                  DisplayString
500         MAX-ACCESS              read-only
501         STATUS                  current
502         DESCRIPTION
503             "Passive OS Fingerprinting attempts to discover what type of
504             system a message is sent from. This value should be the
505             name/version of the OS that sent most ham messages, or
506             'none' if there are no ham messages received."
507         ::= { mailSrvStatusAndCounters 24 }
```

```
508
509     -- } end of mailSrvStatusAndCounters(2)
510
511     -- { mailSrvConfig(3)
512
513     -- Provides information about the MTA configurations. Mostly of them
514     -- can be changed.
515
516     configTimeoutReq            OBJECT-TYPE
517         SYNTAX                  Integer32
518         UNITS                   "seconds"
519         MAX-ACCESS              read-write
520         STATUS                  current
521         DESCRIPTION
522             "Time-out for the request. After this time the connection
523             is closed."
524         ::= { mailSrvConfig 1 }
525
526     configMaxMsgSize            OBJECT-TYPE
527         SYNTAX                  Integer32
528         UNITS                   "octets"
529         MAX-ACCESS              read-write
530         STATUS                  current
531         DESCRIPTION
532             "Maximum message size, including attachments."
533         ::= { mailSrvConfig 2 }
534
535     configMaxRecipients         OBJECT-TYPE
536         SYNTAX                  Integer32
537         MAX-ACCESS              read-write
538         STATUS                  current
539         DESCRIPTION
540             "Maximum number of recipients that the MTA accepts
541             per message delivery request."
542         ::= { mailSrvConfig 3 }
543
544     configTimeoutBeforeError    OBJECT-TYPE
545         SYNTAX                  Integer32
546         UNITS                   "seconds"
547         MAX-ACCESS              read-write
548         STATUS                  current
549         DESCRIPTION
550             "The delay before sending a reject response (codes: 4xx or 5xx)."
551         ::= { mailSrvConfig 4 }
552
553     configMaxParallelDeliveries OBJECT-TYPE
554         SYNTAX                  Integer32
555         MAX-ACCESS              read-write
556         STATUS                  current
557         DESCRIPTION
558             "The maximum number of deliveries that MTA will perform to
559             the same destination simultaneously."
560         ::= { mailSrvConfig 5 }
561
562     configHopCountLimit         OBJECT-TYPE
563         SYNTAX                  Integer32
564         MAX-ACCESS              read-write
565         STATUS                  current
566         DESCRIPTION
567             "A message that contains more 'Received:' headers than this
568             will bounce. An extremely large number of this header may
569             indicate a mail loop or a misconfigured mail server
570             somewhere in the path of this message. This option
571             correlates to the  hopcount_limit directive and
572             defaults should be 50. This value rarely needs to be altered
573             from its default."
574         ::= { mailSrvConfig 6 }
575
576     configMainQueueLimit        OBJECT-TYPE
577         SYNTAX                  Integer32
578         MAX-ACCESS              read-write
579         STATUS                  current
580         DESCRIPTION
581             "The maximum number of messages in the main queue."
582         ::= { mailSrvConfig 7 }
583     -- } end of mailSrvConfig(3)
584
```

```
585     -- { mailSrvConfigAgent(4)
586
587     -- Provides information about the Agent configurations. Some of them
588     -- can be changed.
589
590     configInternalPollFreq      OBJECT-TYPE
591         SYNTAX                  Integer32
592         UNITS                   "seconds"
593         MAX-ACCESS              read-write
594         STATUS                  current
595         DESCRIPTION
596             "How often the Agent polls the MTA to
597             update the cache data/information.
598             This values in seconds."
599         ::= { mailSrvConfigAgent 1 }
600
601     configEventSeverityThreshold OBJECT-TYPE
602         SYNTAX                  INTEGER {
603                                     normal(1),
604                                     warning(2),
605                                     minor(3),
606                                     major(4),
607                                     critical(5)
608                                 }
609         MAX-ACCESS              read-write
610         STATUS                  current
611         DESCRIPTION
612             "User settable to control the severity
613             threshold for traps to be sent.
614             For example, if set = major(4), only the
615             major and critical traps will be sent."
616         ::= { mailSrvConfigAgent 2 }
617
618     -- } end of mailSrvConfigAgent(4)
619
620     -- { mailSrvControl(5)
621
622     -- Control over some running mode of the MTA.
623     -- Control data can be risky!
624
625     controlStatus               OBJECT-TYPE
626         SYNTAX                  INTEGER { running(1), notrunning(2) }
627         MAX-ACCESS              read-write
628         STATUS                  current
629         DESCRIPTION
630             "This value provides the status of the MTA, and this
631             value/status can be changed."
632         ::= { mailSrvControl 1 }
633
634     controlReload               OBJECT-TYPE
635         SYNTAX                  INTEGER { now(1) }
636         MAX-ACCESS              read-write
637         STATUS                  current
638         DESCRIPTION
639             "Reload configuration of the MTA. After
640             changing values on mailSrvConfig(3), the
641             information of the MTA must be updated through
642             this object."
643         ::= { mailSrvControl 2 }
644
645     controlQueueDeleteMsg       OBJECT-TYPE
646         SYNTAX                  DisplayString
647         MAX-ACCESS              read-write
648         STATUS                  current
649         DESCRIPTION
650             "Delete the message from the main queue.
651             Give the ID of the message on the queue, or 'ALL' to
652             delete all the messages."
653         ::= { mailSrvControl 3 }
654
655
656     controlQueueRequeueMsg OBJECT-TYPE
657         SYNTAX                  DisplayString
658         MAX-ACCESS              read-write
659         STATUS                  current
660         DESCRIPTION
661             "Re-queue the message to the main queue.
```

```
662             Give the ID of the message on the queue, or 'ALL' to
663             re-queue all the messages."
664         ::= { mailSrvControl 4 }
665     -- } end of mailSrvControl(5)
666
667 -- } end of Objects
668
669 -- { NOTIFICATION-TYPE Objects
670     -- { mailSrvEvents(6)
671
672     -- Different types of traps/notifications send by MTA.
673
674     -- node 1
675     eventDescriptors            OBJECT-IDENTITY
676         STATUS                  current
677         DESCRIPTION
678             "Event descriptor node."
679         ::= { mailSrvEvents 1 }
680
681     eventSeverity               OBJECT-TYPE
682         SYNTAX                  INTEGER {
683                                     normal(1),
684                                     warning(2),
685                                     minor(3),
686                                     major(4),
687                                     critical(5)
688                                 }
689         MAX-ACCESS              accessible-for-notify
690         STATUS                  current
691         DESCRIPTION
692             "This value is sent with each trap as an indication
693             of the intended severity of the event."
694         ::= { eventDescriptors 1 }
695
696     eventDescription            OBJECT-TYPE
697         SYNTAX                  DisplayString
698         MAX-ACCESS              accessible-for-notify
699         STATUS                  current
700         DESCRIPTION
701             "Text which may provide the user with further
702             diagnostic information."
703         ::= { eventDescriptors 2 }
704
705
706     -- node 0
707     eventList                   OBJECT-IDENTITY
708         STATUS                  current
709         DESCRIPTION
710             "Notifications objects are organized under this node."
711         ::= { mailSrvEvents 0 }
712
713     eventMailSrvStart           NOTIFICATION-TYPE
714         OBJECTS                 { eventSeverity, eventDescription }
715         STATUS                  current
716         DESCRIPTION
717             "This trap could in principle be sent when the MTA start."
718         ::= { eventList 1 }
719
720     eventMailSrvShutdown        NOTIFICATION-TYPE
721         OBJECTS                 { eventSeverity, eventDescription }
722         STATUS                  current
723         DESCRIPTION
724             "This trap is sent when the MTA terminates."
725         ::= { eventList 2 }
726
727     eventMailSrvLoopDetection   NOTIFICATION-TYPE
728         OBJECTS                 { eventSeverity, eventDescription }
729         STATUS                  current
730         DESCRIPTION
731             "This trap is sent when the MTA detect mail loops."
732         ::= { eventList 3 }
733
734     eventMailSrvStorageOverflow NOTIFICATION-TYPE
735         OBJECTS                 { eventSeverity, eventDescription }
736         STATUS                  current
737         DESCRIPTION
738             "If the space/quote of any user mail storage overflows
```

```
739             then this trap is sent. In the eventDescription
740             the user information is shown."
741         ::= { eventList 4 }
742
743     eventMailSrvConfigChanged   NOTIFICATION-TYPE
744         OBJECTS                 { eventSeverity, eventDescription }
745         STATUS                  current
746         DESCRIPTION
747             "If the configuration file changes
748             then this trap is sent. In the eventDescription
749             the file information is shown."
750         ::= { eventList 5 }
751
752     eventMailSrvGeneric         NOTIFICATION-TYPE
753         OBJECTS                 { eventSeverity, eventDescription }
754         STATUS                  current
755         DESCRIPTION
756             "If the none of above traps apply
757             then this is sent."
758         ::= { eventList 6 }
759
760     -- } end of mailSrvEvents(6)
761 -- } end of NOTIFICATION-TYPE Objects
762
763 -- { Conformance groups
764     -- { mailSrvConformance(7)
765
766     -- Describes the requirements for conformance to the MAILSERVER-MIB
767
768     mtaReqConfGroup             OBJECT-GROUP -- need ref at (**)
769         OBJECTS                 {
770                                 -- 1
771                                 infoDescr,
772                                 infoUpTime,
773                                 infoContactPostmaster,
774                                 infoServices,
775                                 infoDomainsAccept,
776                                 infoDomainsRelay,
777                                 infoDefaultTransport,
778                                 infoMailboxType,
779                                 -- 2
780                                 statusConnectionsInbound,
781                                 statusConnectionsOutbound,
782                                 -- table
783                                 processesName,
784                                 processesState,
785                                 processesCpu,
786                                 processesMem,
787                                 processesNumSubProc,
788                                 -- table
789                                 queuesName,
790                                 queuesCurrMsgs,
791                                 queuesCurrVolume,
792                                 queuesMsgsIn,
793                                 queuesMsgsOut,
794                                 queuesOldestMessageId,
795
796                                 statusConnectionsMade,
797                                 statusConnectionsLostInbound,
798                                 statusConnectionsFailureOutbound,
799                                 statusMsgsTotalAccepted,
800                                 statusMsgsTotalRejected,
801                                 statusMsgsTotalSent,
802                                 statusMsgsTotalSpam,
803                                 statusMsgsTotalVirus,
804                                 statusMsgsTotalBounced,
805                                 statusVolumeTotalAccepted,
806                                 statusVolumeTotalSent,
807                                 statusLogFileVolume,
808                                 statusMailSrvError,
809                                 statusMailSrvWarning,
810                                 statusMsgsTotalRecipientsIn,
811                                 statusMsgsTotalRecipientsOut,
812                                 statusSuccessfulConvertedMsgs,
813                                 statusFailedConvertedMsgs,
814                                 statusFingerprintingSpamOS,
815                                 statusFingerprintingHamOS,
```

```
                                -- 3
                                configTimeoutReq,
                                configMaxMsgSize,
                                configMaxRecipients,
                                configTimeoutBeforeError,
                                configMaxParallelDeliveries,
                                configHopCountLimit,
                                configMainQueueLimit,
                                -- 4
                                configInternalPollFreq,
                                configEventSeverityThreshold,
                                -- 5
                                controlStatus,
                                controlReload,
                                controlQueueDeleteMsg,
                                controlQueueRequeueMsg,
                                --
                                eventSeverity,
                                eventDescription
                                }
        STATUS                  current
        DESCRIPTION             "A collection of objects providing basic
                                monitoring and control of MTAs."
        ::= { mailSrvConformance 1 }


    mtaNotificationGroup        NOTIFICATION-GROUP
        NOTIFICATIONS           {
                                -- 6 (notitications)
                                eventMailSrvStart,
                                eventMailSrvShutdown,
                                eventMailSrvLoopDetection,
                                eventMailSrvStorageOverflow,
                                eventMailSrvConfigChanged,
                                eventMailSrvGeneric
                                }
        STATUS                  current
        DESCRIPTION             "Notifications sent by an MAILSERVER-MIB agent."
        ::= { mailSrvConformance 2 }


    mtaModCompliances           MODULE-COMPLIANCE
        STATUS                  current
        DESCRIPTION             "The compliance statement for SNMP entities."
        MODULE  -- this module
        MANDATORY-GROUPS        { mtaReqConfGroup, mtaNotificationGroup }  -- (**)
        ::= { mailSrvConformance 3 }
    -- } end of mailSrvConformance(7)

-- } end of conformance groups

END
```

Anexo B

Programa Para Gerar *E-mails*

Este programa permite enviar diversos tipos de mensagens, em tempo e quantidades aleatórias. Foi utilizado para gerar tráfego de mensagens de *e-mail*.

```
#!/usr/bin/python

# ------------------------------------------------------------------------
# mail_generator.py
# ------------------------------------------------------------------------
# Copyright 2009 Sunny Narendra <ee04185@fe.up.pt>
# ------------------------------------------------------------------------

import smtplib
import sys
import time
import random

mail_server = '192.168.1.72'
mail_from = 'billgates@debian.lan'
rcpt_to = 'sunny@debian.lan'

subject = 'test'

list = ["samples/sample-42-mail-bomb.txt",
                "samples/sample-nonspam.txt",
                "samples/sample-spam.txt",
                "samples/sample-virus-nested.txt",
                "samples/sample-badh.txt",
                "samples/sample-spam-GTUBE-junk.txt",
                "samples/sample-virus-simple.txt",
                "samples/sample-executable.txt",
                "samples/sample-spam-GTUBE-nojunk.txt",
                "samples/sample-virus-executable.txt"]

total = 0
while 1:
        # random message from the list
        r_num1_msg = random.randint(0, 9)
        # random number of messages
        r_num1_maxmsg = random.randint(10, 20)
        # random sleep time
        r_num1_time = random.randint(10, 60)

        # read
        file = open(list[r_num1_msg], 'r')
        msg = ''
        for line in file:
                msg = msg + line
        file.close()
```

```
46
47          # update
48          data = "From: %s\r\nTo: %s\r\nSubject: %s\r\n" % (mail_from, rcpt_to, subject)
49          data = data + msg
50
51          # connect
52          server = smtplib.SMTP(mail_server, 25)
53
54          # send
55          print list[r_num1_msg]
56          i = int(r_num1_maxmsg)
57          while i > 0:
58                  print i
59                  server.ehlo()
60                  server.sendmail(mail_from, rcpt_to, data)
61                  i = i - 1
62                  total = total + 1
63          server.close()
64
65          # sleep
66          print "total: ",total
67          print "sleep :", r_num1_time
68          time.sleep(r_num1_time)
```

Referências

[1] A. Deokjai Choi; Taesang Choi; Tang. Issues in enterprise e-mail management. páginas 78–82, Apr 1996.

[2] Recommendation F.400/X.400 (06/99). http://www.itu.int/rec/T-REC-F.400-199906-I/en/. [Online; último acesso 25-06-2009].

[3] OpenSPF - Sender Policy Framework. http://www.openspf.org/Introduction. [Online; último acesso 05-05-2009].

[4] Benchmarking mbox versus maildir. http://www.courier-mta.org/mbox-vs-maildir/. [Online; último acesso 24-07-2009].

[5] Kyle Dent. *Postfix: The Definitive Guide*. O'Reilly Media, Inc., 1st edição, Dezembro 2003.

[6] J. Postel. Simple Mail Transfer Protocol. RFC 821 (Standard), Agosto 1982. Obsoleted by RFC 2821.

[7] Munawar Hafiz. Security architecture of mail transfer agents. Em *Master's thesis*, páginas 2–3, University of Illinois at Urbana-Champaign, 2005.

[8] Bryan Costales, Claus Assmann, George Jansen, e Gregory Shapiro. *Sendmail, 4th Edition*. O'Reilly Media, Inc., 4 edição, Outubro 2007.

[9] Microsoft Exchange Web Page. http://www.microsoft.com/exchange/. [Online; último acesso 27-05-2009].

[10] William Stallings. *Cryptography and Network Security*. Prentice Hall, 4 edição, Novembro 2005.

[11] DomainKeys Identified Mail (DKIM) . http://www.dkim.org. [Online; último acesso 05-05-2009].

[12] Sendmail Web Page. http://www.sendmail.org/. [Online; último acesso 27-05-2009].

[13] Postfix Web Page. http://www.postfix.org/. [Online; último acesso 27-05-2009].

[14] MailRadar - Email Oriented Community for Linux Sysadmins. http://www.mailradar.com/. [Online; último acesso 29-06-2009].

[15] Douglas Mauro e Kevin Schmidt. *Essential SNMP*. O'Reilly Media, Inc., 2 edição, Setembro 2005.

[16] William Stallings. *SNMP, SNMPv2, SNMPv3 and RMON 1 and 2*. Addison Wesley, 3 edição, Fevereiro 1999.

[17] Larry Walsh. *SNMP MIB Handbook*. Wyndham Press, Março 2008.

[18] J.D. Case, M. Fedor, M.L. Schoffstall, e J. Davin. Simple Network Management Protocol (SNMP). RFC 1157 (Historic), Maio 1990.

[19] ManageEngine Applications Manager - Mail Server Monitoring. http://www.manageengine.com/products/applications_manager/help/monitors/mailserver-monitoring.html. [Online; último acesso 05-02-2009].

[20] IceWarp - eMail Server Management. http://www.icewarp.com/products/icewarp_email_server_software/management.php. [Online; último acesso 05-02-2009].

[21] Azaleos - ViewXchange and ManageXchange. http://www.azaleos.com/products.html. [Online; último acesso 05-02-2009].

[22] Mailgraph - a RRDtool frontend for Mail statistics. http://mailgraph.schweikert.ch/. [Online; último acesso 05-06-2009].

[23] Queuegraph - a RRDtool frontend for Postfix queue-statistics. http://www.arschkrebs.de/postfix/queuegraph/. [Online; último acesso 05-06-2009].

[24] Webmin: Web-based interface for system administration for Unix. http://www.webmin.com/. [Online; último acesso 05-06-2009].

[25] Jae-Young Kim e J.W.-K. Hong. Design and implementation of a web-based Internet/Intranet mail server management system. Em *Communications, 1999. ICC '99. 1999 IEEE International Conference on*, volume 1, páginas 641–645 vol.1, 1999.

[26] N. Freed e S. Kille. Mail Monitoring MIB. RFC 2789 (Proposed Standard), Março 2000.

[27] OpenNMS : Open-source Network Management (Mail Transport Monitor). http://www.opennms.org/index.php/Mail_Transport_Monitor. [Online; último acesso 05-06-2009].

[28] ZABBIX. http://www.zabbix.com/. [Online; último acesso 05-06-2009].

[29] GroundWork OpenSource. http://www.groundworkopensource.com/. [Online; último acesso 05-06-2009].

[30] Cacti : RRDtool-based Graphing Solution. http://www.cacti.net/. [Online; último acesso 05-06-2009].

[31] WINDOWS-NT-PERFORMANCE-EXCHANGE. http://www.oidview.com/mibs/311/WINDOWS-NT-PERFORMANCE-EXCHANGE.html. [Online; último acesso 05-02-2009].

[32] Net-SNMP. http://net-snmp.sourceforge.net/. [Online; último acesso 16-06-2009].

[33] amavisd-new. http://www.ijs.si/software/amavisd/. [Online; último acesso 29-06-2009].