Jorge Miguel Vidigueira Simões Gil

Monitorização, Alarmística e Gestão de Redes

Abordagem Open Source em Ambiente Empresarial

IPL
instituto politécnico
de leiria

Departamento de Engenharia Informática
ESTG - Leiria
Instituto Politécnico de Leiria
Setembro de 2012

Jorge Miguel Vidigueira Simões Gil

Monitorização, Alarmística e Gestão de Redes

Abordagem Open Source em Ambiente Empresarial

IPL
instituto politécnico
de leiria

Dissertação de Mestrado em Engenharia Informática - Computação Móvel

Supervisor: Prof. Doutor Mário João Gonçalves Antunes

Departamento de Engenharia Informática
ESTG - Leiria
Instituto Politécnico de Leiria
Setembro de 2012

Agradecimentos

Ao meu orientador, Prof. Doutor Mário João Gonçalves Antunes por todo o conhecimento, sugestões e tempo disponibilizado, que foram fundamentais durante a elaboração desta tese.

À família por me ter acompanhado durante todo o meu percurso académico e pelo suporte e motivação que me deram e continuam a dar.

Aos que me acolheram na Linkcom, me fizeram sentir à vontade desde o primeiro dia, que sempre mostraram disponibilidade e contribuiriam para o meu crescimento profissional, em particular ao Luís Fonseca e Pedro Quintela.

Aos meus colegas da academia de Leiria que me acompanharam e suportaram durante estes últimos anos.

Ao Instituto Politécnico de Leiria, por todas as condições disponibilizadas.

Aos amigos de sempre...

Resumo

O aumento das redes de computadores, quer em número de equipamentos, quer em heterogeneidade desses mesmos equipamentos levou a que a monitorização de um parque informático se tenha tornado mais difícil e complexa. Por um lado, pela dificuldade logística de monitorizar um número crescente de dispositivos, por outro pelos vários métodos necessários para recolher informação referente aos diferentes tipos de dispositivos. Para colmatar estas dificuldades torna-se assim necessário recorrer à utilização de aplicações que assegurem, de forma integrada, as tarefas de monitorização e alarmística em redes heterogéneas de grande dimensão.

Uma empresa que dependa da sua infra-estrutura de rede e dos serviços por ela prestados necessita de manter o seu bom estado de funcionamento de modo a garantir a competitividade do seu negócio. A utilização de aplicações de monitorização permite que uma empresa mantenha uma abordagem pró-activa no modo como visualiza a sua infra-estrutura de rede, garantindo que os problemas são conhecidos antes de provocar danos que provoquem a interrupção dos serviços e deteriorem a qualidade de serviço prestada.

A distribuição das tarefas de monitorização e alarmística, nomeadamente verificação de estados dos dispositivos e dos valores de desempenho de componentes como CPU ou disco, por várias aplicações torna-se uma desvantagem, pois é necessário multiplicar o esforço na manutenção das várias aplicações, bem como possuir conhecimento do modo de operação de cada uma. Para além desta dificuldade é necessário o cruzamento de informação entre as varias aplicações, o que nem sempre é uma tarefa simples ou intuitiva.

É com estas dificuldades que a empresa Linkcom se depara na sua actual solução de monitorização, composta pelas aplicações Nagios e Cacti, que desempenham funções distintas. Enquanto o Nagios monitoriza os estados dos dispositivos e efectua funções de alarmística, o Cacti efectua a monitorização de desempenho dos componentes dos dispositivos.

O trabalho desenvolvido no âmbito desta tese visa apresentar uma nova solução integrada de monitorização e alarmística para a Linkcom, baseada em tecnologia open-source, que

possa colmatar os problemas apresentados, através da centralização das funcionalidades executadas pelo Nagios e Cacti, numa única plataforma de gestão centralizada.

Para além da centralização das funcionalidades, a nova solução tem como objectivo facilitar a utilização e configuração, bem como possuir uma apresentação mais apelativa e acrescentar algumas funcionalidades de gestão de rede, nomeadamente a possibilidade de manter um registo documental actualizado dos componentes da rede e disponibilizar relatórios detalhados que permitam visualizar o comportamento da rede e dos seus componentes ao longo do tempo.

Esta tese apresenta uma análise comparativa entre várias aplicações open-source de monitorização e alarmística, nomeadamente Nagios, Cacti, Zabbix Zenoss e Shinken, que resulta na selecção e implementação da aplicação que melhor se adequa a este caso. Apresenta ainda as principais vantagens em relação à solução utilizada actualmente e a forma como pretende colmatar as suas limitações.

Abstract

The growth of computer networks, caused by the increasing number of devices and the heterogeneity of their systems and technologies, means that the network monitoring and management became a very difficult and complex task. This difficulty is caused by an-ever growing number of devices and their several methods of providing performance data. In order to reduce the complexity of network monitoring and management tasks, monitoring and alarmistic tools that can handle system performance monitoring on large heterogeneous networks are used.

A company that provides network services as its core business must have special attention to its network devices performance in order to maintain a reliable service delivery to its costumers. The use of a monitoring solution allows a complete overview of a network and its components, providing means to pro-actively manage the network and detect problems in an early stage and, therefore, avoid service failure.

The use of several tools for the network monitoring tasks may be an issue, due to the effort of their maintenance, as well as the need to know each tool way of operation. There is also the task of merge the data collected by all the tools, which isn't always a simple or intuitive task.

These are the problems that affect Linkcom, which has its network monitored by several monitoring tools, namely Nagios and Cacti. Each one of this tools has a specific task, Cacti monitors system and service performance, as Nagios performs state monitoring and alarmistic tasks.

This thesis aims to propose an open-source network monitoring and alarmistic solution to Linkcom network that offers less effort to maintain that the current solution. The new solution also aims to be more user-friendly and efficient by presenting all the data collected in just one platform. It will also have features that are not present in the current solution, such as network devices inventory and components and network performance reports.

In order to select the proper application to be implemented at Linkcom, several open-

source network monitoring tools, namely Nagios, Cati, Zabbix, Zenoss and Shinken, were analysed. These tools were tested against the required features and ease-of-use which resulted in the selection of the most suitable tool for Linkcom network monitoring.

Índice

Lista de Tabelas

Lista de Figuras

Acrónimos

CIM Common Information Model

CGI Common Gateway Interfaces

CLI Comand Line Interface

CMDB Configuration Management Database

FCAPS Fault, Configuration, Accounting, Performance, Security

GSM Global System for Mobile

IIS Internet Information Services

IPMI Intelligent Plataform Management Interface

ISO International Organization for Standardization

ITIL Information Technology Infrastructure Library

ITU International Telecommunication Union

ITU-T ITU- Telecomunication Standardization Sector

LDAP Lightweight Directory Access Protocol

MD Managed Devices

MIB Management Information Base

NMS Network Management System

OID Object IDentifier

PDU Protocol Data Unit

PIA Plugin Architecture

RRDTool Round Robin Database Tool

SLA Service-Level Agreement

SMS Short Message Service

SNMP Simple Network Management Protocol

TMN Telecomunication Management Network

UPS Uninterruptible Power Supply

WBEM Web-Based Enterprise Management

WMI Windows Management Instrumentation

WMIC Windows Management Instrumentation Commandline

Capítulo 1

Introdução

Este capítulo apresenta o enquadramento do tema da tese e respectiva motivação, identificando os objectivos propostos e a organização do documento.

1.1 Motivação e Enquadramento

A constante evolução das redes de computadores e dos seus serviços tem levado a que os parques informáticos das empresas e organizações sejam cada vez mais complexos e heterogéneos, proporcionando vários desafios na tarefa de manter e gerir a sua infraestrutura de rede e servidores. A par do crescimento do parque informático, cresce a dependência dos serviços por si prestados e, consequentemente, a necessidade de manter o seu correcto funcionamento ao longo do tempo. O uso de técnicas mais tradicionais, como a execução manual *scripts* de verificação de equipamentos ou serviços tornou-se uma tarefa bastante difícil e complexa devido ao elevado número de verificações a realizar numa grande quantidade de dispositivos da rede. No sentido de colmatar estas dificuldades foram desenvolvidas várias aplicações de gestão e monitorização de redes, tais como o Nagios [10] ou o Cacti [11], destinadas a monitorizar os vários componentes de dispositivos, como a utilização de CPU ou a ocupação de discos, estados dos serviços em execução nos servidores, tais como serviços de base de dados ou serviços web, assim como outros equipamentos activos de rede, como *switches* ou *firewalls*.

As aplicações de monitorização e gestão de redes disponíveis no mercado, na sua instalação padrão, não respondem a todas as necessidades da grande maioria das empresas, efectuando apenas a monitorização básica, como a disponibilidade dos equipamentos. Torna-se portanto, necessária a sua configuração de modo a que possam ser integradas na empresa e

possam responder às principais necessidades de monitorização.

A principal função destas aplicações consiste na monitorização de recursos da rede através de verificações periódicas aos dispositivos. Os dados recolhidos, para além de permitirem a análise da rede em tempo real, são também utilizados para efeitos de histórico e detecção de ocorrências na infraestrutura. Perante certas ocorrências a aplicação de monitorização pode ser configurada para se comportar de forma pró-activa e despoletar acções que permitam, por exemplo, reiniciar o serviço ou o dispositivo com o objectivo de solucionar o problema. As ocorrências detectadas são enviadas através de vários canais, como Short Message Service (SMS) ou e-mail, para o administrador de rede sob a forma de alarmes.

Empresas dependentes dos recursos da sua rede devem utilizar mecanismos de monitorização e alarmística eficientes que mantenham, de forma centralizada, informações actualizadas do estado dos seus recursos de rede e permitirem métodos de envio de alarmes de acordo com as ocorrências na rede monitorizada.

A Linkcom [12] é um exemplo de uma empresa em que a utilização de uma aplicação deste tipo é uma mais valias para a prestação dos seus serviços de gestão de redes aos seus clientes. A Linkcom foi criada em 2000, como integrante do Grupo AITEC, composto por empresas como a Link Consulting ou a Link Management Solutions. A Linkcom tem como objectivo prestar serviços especializados na área das tecnologias de informação. Os seus serviços destinam-se a pequenas e médias empresas, principalmente o fornecimento de soluções de sistemas de informação que permitam aumentar a sua eficiência, sob a forma de fornecimento de *hardware* e *software*, instalação e administração de sistemas, suporte técnico e alojamento de websites e aplicações.

Sendo uma empresa que presta serviços que recaem na sua infraestrutura de rede, necessita de manter o desempenho da infraestrutura em níveis competitivos, assegurando os níveis de qualidade de serviço contratados com os clientes. A monitorização da infraestrutura de uma rede permite assim ter uma visão global dos componentes da rede e respectivo estado, que se traduz numa rápida identificação dos problemas que surjam. Esta característica permite diminuir o tempo dedicado à identificação do problema, o que possibilita uma intervenção mais rápida na resolução do problema.

A solução de monitorização e alarmística implementada na Linkcom é composta por várias aplicações orientadas para diferentes tarefas, nomeadamente monitorização de disponibilidade, realizada pelo Nagios [10], e monitorização de desempenho dos diversos componentes dos dispositivos da rede, realizada pelo Cacti[11].

Esta solução cumpre globalmente todas as tarefas de monitorização e alarmística. No

entanto, obriga a uma multiplicação do esforço de gestão e configuração pelas diversas aplicações. Para além do esforço utilizado para a configuração é também necessário a formação de gestores de rede nas várias aplicações.

A informação recolhida pelas várias aplicações obriga a que seja necessário o cruzamento sempre que se pretende analisar problemas ou o estado e evolução da infra-estrutura de rede. O cruzamento desta informação pode-se tornar uma tarefa complexa, pois as abordagens utilizadas para apresentação de dados utilizadas pelas várias aplicações diferem entre si.

A configuração de aplicações, como o Nagios, é complexa e pouco intuitiva pois utiliza ficheiros de texto para a definição dos vários parâmetros da aplicação como, por exemplo, dispositivos, alertas ou utilizadores. Já o Cacti, apesar de possibilitar a sua configuração através da interface Web, não possui mecanismos que permitam o envio de alarmes referentes aos dados recolhidos.

A motivação para esta dissertação consiste no estudo e implementação de uma solução *open-source* integrada, de monitorização e alarmística, orientada a um ambiente empresarial. Pretende-se avaliar a implementação de uma solução de monitorização *open-source* e de baixo custo num ambiente empresarial. Pretende-se, igualmente, comparar o uso deste tipo de soluções com outras proprietárias e de custo elevado. O caso de estudo para esta tese foi a empresa Linkcom, que possuía uma solução de monitorização e alarmística com algumas limitações e dificuldades de análise de dados recolhidos, visto ser uma solução distribuída por aplicações diferentes, designadamente o Nagios e o Cacti. Com a nova solução de monitorização e alarmística pretende-se colmatar essas limitações bem como introduzir algumas funcionalidades de gestão, como documentação dos componentes monitorizados.

1.2 Objectivos

Com este trabalho pretende-se propor uma solução de monitorização e gestão de recursos de rede destinada a um ambiente empresarial tendo como caso de estudo a empresa Linkcom. Os principais objectivos a alcançar com este projecto são:

- identificar as limitações da solução de monitorização actualmente em funcionamento na Linkcom;
- analisar e testar as aplicações *open source* mais relevantes para a monitorização de rede, disponíveis no mercado;

- implementar uma nova solução de monitorização num ambiente de testes e identificar as vantagens, desvantagens e mais valias de uma utilização em ambiente empresarial;
- definir a estratégia de migração para o ambiente de produção;
- avaliar a solução em produção na Linkcom e mitigar as suas limitações.

O projecto foi integralmente desenvolvido na Linkcom e será futuramente integrado em ambiente de produção.

1.3 Organização do Documento

O presente documento está organizado em mais cinco capítulos:

- Capítulo 2: Revisão da Literatura - este capítulo apresenta o levantamento de algumas das principais aplicações para monitorização de rede. Serão igualmente abordadas as temáticas de modelos de gestão de redes e protocolos de monitorização
- Capítulo 3: Metodologia - neste capítulo encontra-se descrita a análise da utilização das várias aplicações numa rede de testes, bem como uma análise comparativa entre elas
- Capítulo 4: Modelo Proposto - este capítulo descreve os requisitos para a nova solução de monitorização e alarmística, a arquitectura proposta e os métodos utilizados para recolha de parâmetros dos diferentes dispositivos monitorizados
- Capítulo 5: Caso de Estudo - neste capitulo descreve-se a metodologia utilizada para as tarefas de migração para a nova solução, os detalhes da implementação da nova solução, dificuldades encontradas e métodos de mitigação de falsos positivos utilizados bem como a descrição de funcionalidades adicionadas. O capítulo termina com os resultados obtidos na implementação da nova solução.
- Capítulo 6: Conclusões - Por fim, efectua-se a síntese e análise do trabalho desenvolvido. Apresentam-se algumas sugestões a implementar no futuro, com intuito de melhorar a solução.

Capítulo 2

Revisão da Literatura

Este capitulo descreve a visão geral das aplicações, tecnologias e conceitos relacionados com este projecto. Pretende-se apresentar as recomendações de monitorização e modelos de gestão existentes, bem como as aplicações analisadas, as suas principais funcionalidades e como se distinguem entre si. Serão também apresentados os métodos mais utilizados nas tarefas de monitorização.

2.1 Modelos de Gestão

A gestão e monitorização de rede é uma actividade de extrema importância e é independente do tamanho ou do tipo de rede. Uma gestão correcta permite que os níveis de disponibilidade e de desempenho, tanto da rede como dos serviços que esta fornece, se mantenham em níveis aceitáveis de acordo com o nível de qualidade na prestação do serviço acordado.

O crescimento de uma rede implica que a complexidade da sua gestão também aumente, obrigando à utilização de aplicações que permitam a automatização das tarefas de gestão e controlo da rede. Este tipo de aplicações permite observar o estado actual da rede, bem como analisar a sua evolução e de todos os seus componentes, permitindo um planeamento de possíveis alterações que necessitem de ser realizadas.

Independentemente das aplicações que se utilizem, uma rede deve ser gerida tendo como referência modelos recomendados por organizações que definem *standards* referentes às tecnologias da informação, como o caso do International Organization for Standardization (ISO) ou International Telecommunication Union (ITU).

2.1.1 Modelo de Gestão FCAPS

A gestão de uma rede deve ter por base um dos modelos de referência de gestão de rede disponíveis. A divisão ITU- Telecomunication Standardization Sector (ITU-T), pertencente à ITU, juntamente com a ISO definiram um modelo de gestão para redes de comunicação denominado Telecomunication Management Network (TMN) [13].

O modelo conceptual de gestão TMN define cinco áreas para a gestão de rede, denominadas FCAPS. A divisão FCAPS [14] foi efectuada de modo a que cada área possua um âmbito e um conjunto de componentes bem definido. As suas áreas são as seguintes:

- **Gestão de Falhas** (Fault);
- **Gestão de Configuração** (Configuration);
- **Gestão de Contas** (Accounting);
- **Gestão de Desempenho** (Performance);
- **Gestão de Segurança** (Security).

2.1.1.1 Gestão de Falhas

O objectivo da categoria gestão de falhas consiste em manter uma rede funcional e sem falhas. Esta categoria encontra-se dividida em três fases: monitorização, diagnóstico e envio de notificações ou correcção de falhas de forma pro-activa.

Segundo esta categoria, deverá existir um meio de visualização de informação relativa a todas as notificações ou alarmes enviados, por exemplo através de representações gráficas de estados de componentes que permitam uma fácil interpretação da informação.

Outra das sugestões dadas pela norma é a existência de um histórico dos eventos, como forma de prever futuras falhas. Assim que os problemas estiverem resolvidos, todos os alarmes referentes devem ser removidos do sistema de modo a manter o estado da rede o mais actualizado possível.

2.1.1.2 Gestão de Configuração

De modo a monitorizar correctamente uma rede é necessária a sua correcta configuração, que implica a descoberta da rede, dos dispositivos que a compõem, dos seus recursos e dos

serviços fornecidos.

De modo a evitar a constante verificação da configuração da rede, a gestão de configuração sugere a existência de uma sincronização entre o sistema de monitorização e a própria rede. Para cumprir este objectivo é necessário que o sistema possua a capacidade de encontrar alterações na configuração dos dispositivos da rede.

Um sistema de gestão deve ainda proteger o sistema, através de cópias de segurança, de modo a facilitar a sua recuperação em caso de necessidade.

2.1.1.3 Gestão de Contas

Esta categoria define métodos de facturação de serviços fornecidos através da rede. Desta forma, a gestão de contas mede a utilização de recursos da rede, determinando o seu custo para o fornecedor de serviços e cobra um custo aos utilizadores pelos serviços disponibilizados.

A gestão de contas prevê ainda o suporte de auditorias e comunicação de fraudes, mediante a análise de comportamentos suspeitos [15].

2.1.1.4 Gestão de Desempenho

A gestão de desempenho tem como objectivo principal a recolha e análise dos dados estatísticos da rede com o intuito de monitorizar e corrigir o comportamento e eficácia dos equipamentos e da rede em si. Esta análise permite identificar se a qualidade dos serviços é a que se encontra acordada no Service-Level Agreement (SLA). Uma análise aos dados recolhidos por esta categoria permite ainda obter informação essencial ao planeamento de alterações da rede ou antever a ocorrência de problemas.

2.1.1.5 Gestão da Segurança

A categoria gestão de segurança sugere a definição de políticas de autenticação, controlo de acesso, confidencialidade, integridade e não repudio na comunicação entre sistemas, entre utilizadores e sistemas e entre utilizadores, juntamente com mecanismos de segurança aplicáveis a qualquer tipo de comunicação.

Estas medidas permitem garantir que apenas os utilizadores com nível de permissão adequado possam efectuar alterações a configurações de rede e/ou dispositivos.

Para além das medidas de segurança internas, devem ser tomadas outras medidas que afectam os acessos com o exterior. Um exemplo das medidas a tomar nestes casos prende-se com a limitação de portos e endereços IP acessíveis do exterior e a criação de listas de acesso ou monitorização de padrões de tráfego considerados suspeitos.

Devido à natureza deste projecto e tendo em conta a sua integração na Linkcom, o desenvolvimento desta tese foca principalmente o contexto das áreas de gestão de falhas e gestão de desempenho. Apesar de todas as áreas apresentadas serem igualmente importantes, as áreas de gestão de falhas e gestão de desempenho são aquelas em que uma aplicação de monitorização e alarmística melhor se integra e às quais dá melhor resposta. Esta característica não inviabiliza que a solução cumpra também algumas características das restantes áreas.

2.1.2 Simple Network Management Protocol (SNMP)

O Simple Network Management Protocol (SNMP) [16] é um protocolo desenhado para gestão de redes. Permite que os dispositivos de rede sejam geridos remotamente através do protocolo IP. A sua arquitectura consiste nos seguintes componentes:

a) **Network Management System (NMS)**: aplicação que efectua a recolha e o processamento de dados relativos à gestão de um dispositivo.

b) **Management Information Base (MIB)**: é um conjunto estruturado de vários objectos, sendo cada objecto composto por um nome, sintaxe e Object IDentifier (OID). Um objecto representa um parâmetro do dispositivo e o valor associado que pode ser obtido através da utilização do protocolo SNMP. A representação das MIB é feita através de um esquema em árvore na qual as informações de topo se ramificam até aos detalhes dos dispositivo, ou seja, os seus parâmetros de funcionamento. Os objectos representados numa MIB são independentes do protocolo, o que permite a criação de novas variáveis MIB para parâmetros específicos de um dispositivo.

c) **Agente SNMP**: módulo de software de gestão de rede, presente num dispositivo, responsável pelo tratamento da informação referente à entidade onde reside. A sua principal função é traduzir os dados armazenados na MIB para linguagem entendida pelo SNMP. O agente pode ser um programa independente executado no dispositivo (serviço) ou embutido no sistema operativo, como é o caso do IOS da Cisco.

d) **Managed Devices (MD)**: equipamentos que possuem um agente SNMP. Possuem a

informação de gestão a ser disponibilizada ao NMS. Qualquer equipamento de rede que possua um agente SNMP é um potencial MD.

Através desta arquitectura, representada na Figura 2.1, o SNMP disponibiliza vários tipos de informação sobre os vários componentes de um MD, tais como dados das interfaces de rede ou carga da CPU, entre outros.

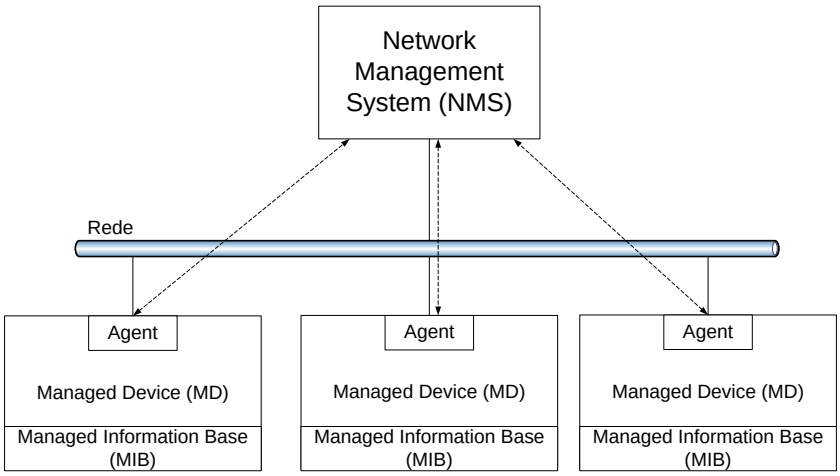

Figura 2.1: Arquitectura global do protocolo SNMP.

As primeiras versões do SNMP são consideradas não seguras, pois apenas se baseiam numa autenticação trivial, utilizando *community strings* como palavra-passe [17]. As *community strings* não oferecem segurança, uma vez que circulam em texto claro pela rede. O agente permite que sejam configuradas três tipos de *communities*: *read-only*, *read-write* e *trap*. A community *read-only* apenas permite que dados do agente sejam lidos, a comunidade *read-write* permite que os dados sejam lidos e alterados e a *trap* permite o envio de dados ao NMS de modo assíncrono. Neste caso, o MD pode enviar para o NMS o resultado de um evento que tenha ocorrido num dos seus componentes. Por exemplo, a alteração de estado de uma interface de rede.

O SNMPv3 [18] veio melhorar o sistema de segurança do SNMP através da utilização de autenticação assimétrica [19]. Este tipo de autenticação permite a troca segura de informação, uma vez que tanto a informação trocada, como a palavra-passe se encontram cifrados quando circulam pela rede.

2.1.2.1 MIB-II

A MIB-II [20] é a segunda versão da MIB destinada à gestão de redes TCP/IP. Foi criada de modo a ser uma MIB standard para este tipo de redes. Esta MIB possui grupos de objectos referentes ao sistema e interfaces de rede, bem como grupos referentes a tabelas de encaminhamentos e protocolos de comunicação, tais como TCP, UDP ou IP.

Deste modo os vários fabricantes de equipamentos implementem a sua própria MIB, destinada a parâmetros específicos de cada equipamento, mantendo os objectos standard da MIB-II (Figura 2.2).

A troca de informação entre o agente SNMP e o NMS é efectuada através do protocolo UDP. A decisão de utilizar o UDP, em vez do TCP, tem como base a redução do *overhead* criado na comunicação evitando sobrecarregar a rede. Os portos utilizados pelo SNMP são o 161 para troca de pedidos e respostas, e o 162 para a recepção de *traps*.

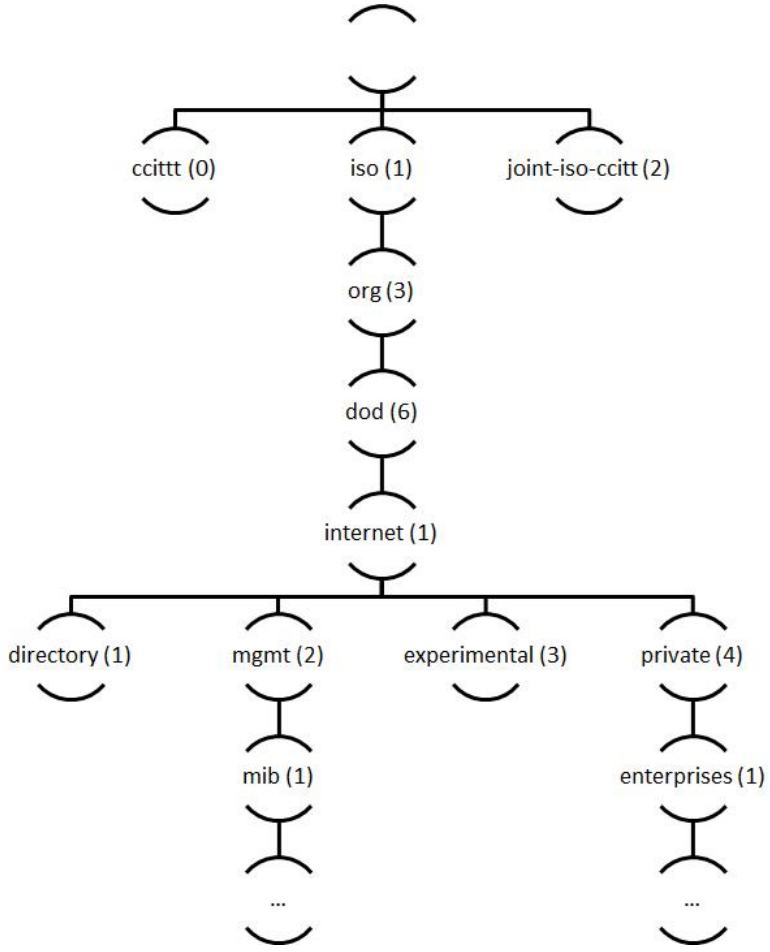

Figura 2.2: Estrutura em árvore da MIBII.

2.1.2.2 Mensagem SNMP

As principais mensagens trocadas entre NMS e o Agente SNMP permitem definir quais as informações a consultar ou alterar nos MD. As mensagens trocadas são as seguintes:

- `Get`: mensagem enviada pelo gestor (NMS) para o agente, efectuando um pedido do valor de uma variável através do envio do OID.
- `GetNext`: o gestor pede informação do objecto seguinte ao que é enviado na mensagem. Deste modo é possível recolher vários objectos contíguos.
- `Set`: o gestor envia a mensagem com o valor a colocar no objecto definido.
- `Response`: mensagem que contém o valor do objecto pedido anteriormente.
- `Trap`: o agente envia uma mensagem não solicitada ao gestor (NMS). É uma situação excepcional e é normalmente utilizada quando o agente detecta uma anomalia no dispositivo em causa.

As mensagens SNMP são enviadas no protocolo IP e são constituídas por três campos principais: versão do SNMP utilizada, *community string* e Protocol Data Unit (PDU). O PDU contém várias informações referentes aos objectos da MIB, divididas por diversos campos.

As mensagens SNMP mais trocadas são mensagens de `Get, Set, Response` e `Trap`. Uma representação de um pacote SNMP pode ser observado na Figura 2.3.

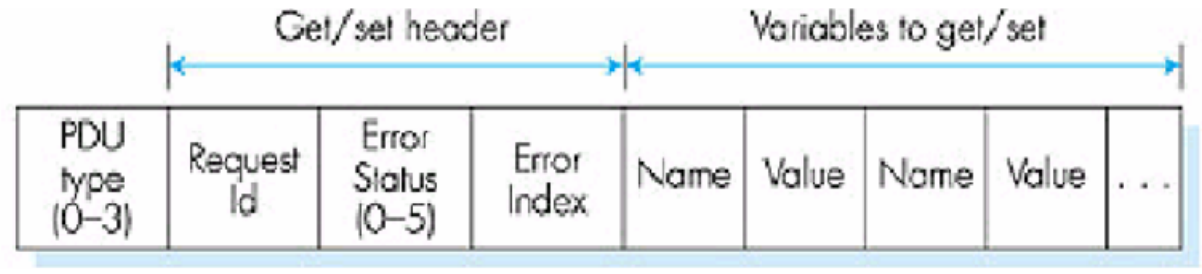

Figura 2.3: Estrutura de uma mensagem SNMP. [1]

Os principais campos que numa mensagem SNMP são:

- ***Request ID*** - Campo no formato inteiro que identifica uma mensagem SNMP. A resposta contém o mesmo *Request ID* de modo a que seja possível mapear as mensagens de pedido e resposta;
- ***Error Status*** - Identificador do tipo de erro ocorrido no processamento do pedido, com cinco classes de erros distintas. No caso de um pedido, este identificador assume sempre o valor de "classe 0"(sem erros);
- ***Error Index*** - Caso exista algum erro, guarda o ponteiro para o objecto que causou o erro;
- ***Variables*** - Conjunto dos objectos pedidos. Possui dois campos por objecto, o `OID` e o `Value`. No caso da utilização da versão 1 do SNMP este campo apenas contem um objecto por pedido.

A Figura 2.4 apresenta um pedido SNMP efectuado pelo NMS a um MD. Neste caso foi efectuado um pedido utilizando a versão 2 e o parâmetro pedido pode ser verificado através do campo ObjectName. A Figura 2.5 representa a resposta ao pedido efectuado anteriormente (Figura 2.4). Neste caso o campo Value encontra-se preenchido com o valor e tipo de dados do parâmetro pedido. Neste caso foi efectuado um pedido pelo item *SysDescr*.

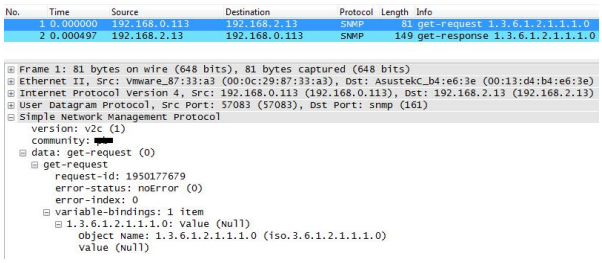

Figura 2.4: Pedido SNMP.

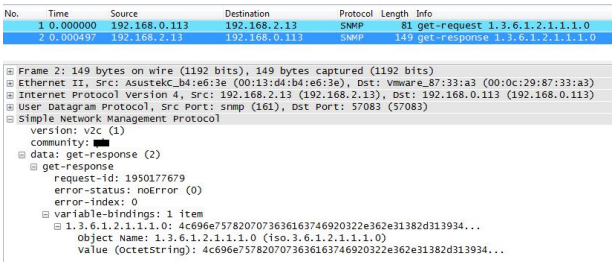

Figura 2.5: Resposta SNMP.

Quando é necessário recolher vários parâmetros contíguos, o SNMP permite a utilização do método *Get-Next*, como se pode observar na Figura 2.6. Este método consiste na execução de pedidos consecutivos pelo próximo OID disponível a um MD. Este método é normalmente utilizado para recolha de todos os OID da MIB de um equipamento ou para recolha de um subconjunto de uma MIB.

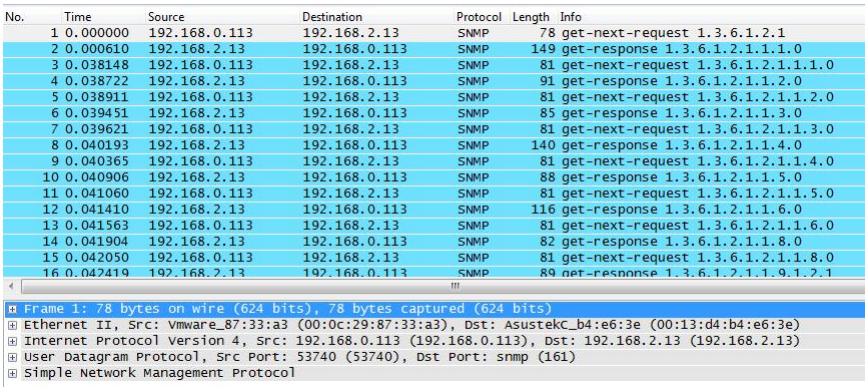

Figura 2.6: Exemplo de utilização do comando GetNext.

2.2 Windows Management Instrumentation (WMI)

O Windows Management Instrumentation (WMI) [21] é a implementação do Web-Based Enterprise Management (WBEM) efectuada pela Microsoft e é parte integrante dos sistemas operativos Windows desde a versão 2000. O WMI é um repositório de objectos, representado através do modelo Common Information Model (CIM), que disponibiliza informação acerca dos componentes de *hardware* e *software*.

Através do Windows Management Instrumentation Commandline (WMIC) é possível aceder às várias instâncias disponibilizadas pelo WMI de uma forma simplificada, o que permite obter vários dados acerca de um computador com Windows. Através da execução do `wmic` é possível definir scripts de recolha das várias instâncias e, deste modo, obter o estado dos vários componentes do computador com o intuito de monitorizar o seu estado.

2.3 Aplicações de Monitorização

Actualmente as aplicações de monitorização de redes desempenham um papel bastante importante nas organizações, uma vez que permitem a observação do estado dos dispositivos, bem como os serviços e recursos do parque informático, permitindo um melhor controlo sobre a rede. Os métodos utilizados por grande parte deste tipo de aplicações consistem em efectuar verificações regulares aos equipamentos da rede como ,por exemplo, routers, switches ou servidores, receber e tratar os resultados obtidos de modo a apresenta-los ao administrador, geralmente, através de uma interface Web [22].

As aplicações de monitorização de redes analisadas nesta tese baseiam-se na arquitectura cliente-servidor, na qual um servidor monitoriza um conjunto de dispositivos clientes. O

modo de recolha da informação por parte do servidor varia, mas normalmente pode ser dividido em dois grande grupos: com agentes (*agentfull*) e sem agentes (*agentless*).

O modo de monitorização com agentes baseia-se na utilização de aplicações que possuem uma relação activa com a aplicação de monitorização e que são instalas no dispositivo que se deseja monitorizar [23]. Normalmente estes agentes recolhem informação localmente e disponibilizam-na ao servidor de monitorização. A utilização de agentes é um processo moroso, pois requer a instalação e configuração de um agente em cada máquina que se deseja monitorizar. Por outro lado fornece informação com grande detalhe acerca dos parâmetros monitorizados.

A monitorização sem agente (*agentless*) refere-se à monitorização sem a utilização de um agente da aplicação. Este método permite utilizar protocolos *standard* para a monitorização, como é o caso do SNMP, SSH ou do WMI. Apesar de ser um método mais simples, pois não necessita da instalação de agentes, a informação recolhida pode ser menos detalhada.

O modo de monitorização a utilizar (*agentfull ou agentless*) deve ser escolhido tendo em conta aspectos como o detalhe de informação desejado ou o tipo de equipamento, pois existem equipamentos que apenas disponibilizam informação por SNMP como é o caso de *switches* ou *routers*.

As aplicações de monitorização disponibilizam, além de mecanismos de monitorização, a possibilidade de gerar relatórios sob os dados monitorizados. Esta funcionalidade permite analisar o comportamento dos componentes da rede ao longo do tempo, fornecendo uma base sólida para planeamento de alterações que sejam necessárias.

Existem aplicações de monitorização de rede que possuem funcionalidades de alarmística e permitem o envio de alertas perante anomalias nos serviços ou dispositivos monitorizados através de vários modos como e-mail, SMS ou Jabber.

De seguida apresentam-se e analisam-se as aplicações utilizadas no âmbito desta tese, designadamente Nagios, Cacti, Zabbix, Zenoss e Shinken. A descrição de cada aplicação engloba os seus requisitos, arquitectura, funcionalidades, modo de funcionamento e respectivo modo de configuração.

2.3.1 Nagios

O Nagios [10] é uma aplicação *open source* de monitorização de rede e de sistemas que monitoriza os estados dos recursos, apresentado-os como "disponível"ou "indisponível".

O Nagios recorre a *plugins* para efectuar as várias verificações. Esta característica torna o Nagios uma aplicação bastante flexível e modular para monitorização de sistemas e serviços.

Periodicamente o *deamon* do Nagios executa verificações aos dispositivos e serviços, através da execução de *plugins*, que obtêm a informação pretendida.

O Nagios suporta monitorização de dispositivos (PC, router, switch) e de serviços (HTTP, DNS). Cada um dos serviços monitorizados está associado a um dispositivo, no qual é executado. A informação monitorizada e respectivo histórico é apresentado numa página Web.

Caso seja detectado um valor fora do normal é gerada uma mensagem de notificação que pode ser enviada ao utilizador em diversos formatos, como e-mail ou SMS. Em alguns casos específicos, o próprio Nagios pode tentar solucionar o problema. Apesar dos *plugins* do Nagios efectuarem uma monitorização baseada em valores, o resultado é apresentado, ao administrador, sob a forma de um de quatro estados: `OK`, `WARNING`, `CRITICAL` e `UNKNOWN`. Cada um dos estados pode ser definido pelo administrador através da definição de valores limite (*threashold*) para os estados `WARNING` e `CRITICAL`. Apesar desta abordagem apresentar ao utilizador uma abstracção dos valores monitorizados, pode tornar-se uma desvantagem no caso de ser necessário gerar relatórios detalhados com os valores de desempenho monitorizados como, por exemplo, ex. gráficos de utilização de rede.

2.3.1.1 Requisitos

Os principais requisitos para o correcto funcionamento do Nagios são um computador com sistema operativo Linux ou um variante de UNIX e compilador de C. Como a grande maioria das verificações são efectuadas por rede é necessário a pilha protocolar TCP/IP correctamente configurada [2].

Caso se utilize os Common Gateway Interfaces (CGI) incluídos no Nagios é necessário um servidor Web (preferencialmente Apache) e a biblioteca *gd library*.

Os *plugins* do Nagios possuem requisitos específicos, que dependem da sua implementação como, por exemplo, bibliotecas de Perl.

2.3.1.2 Arquitectura

A arquitectura do Nagios encontra-se dividida em várias camadas, como se pode observar na Figura 2.7.

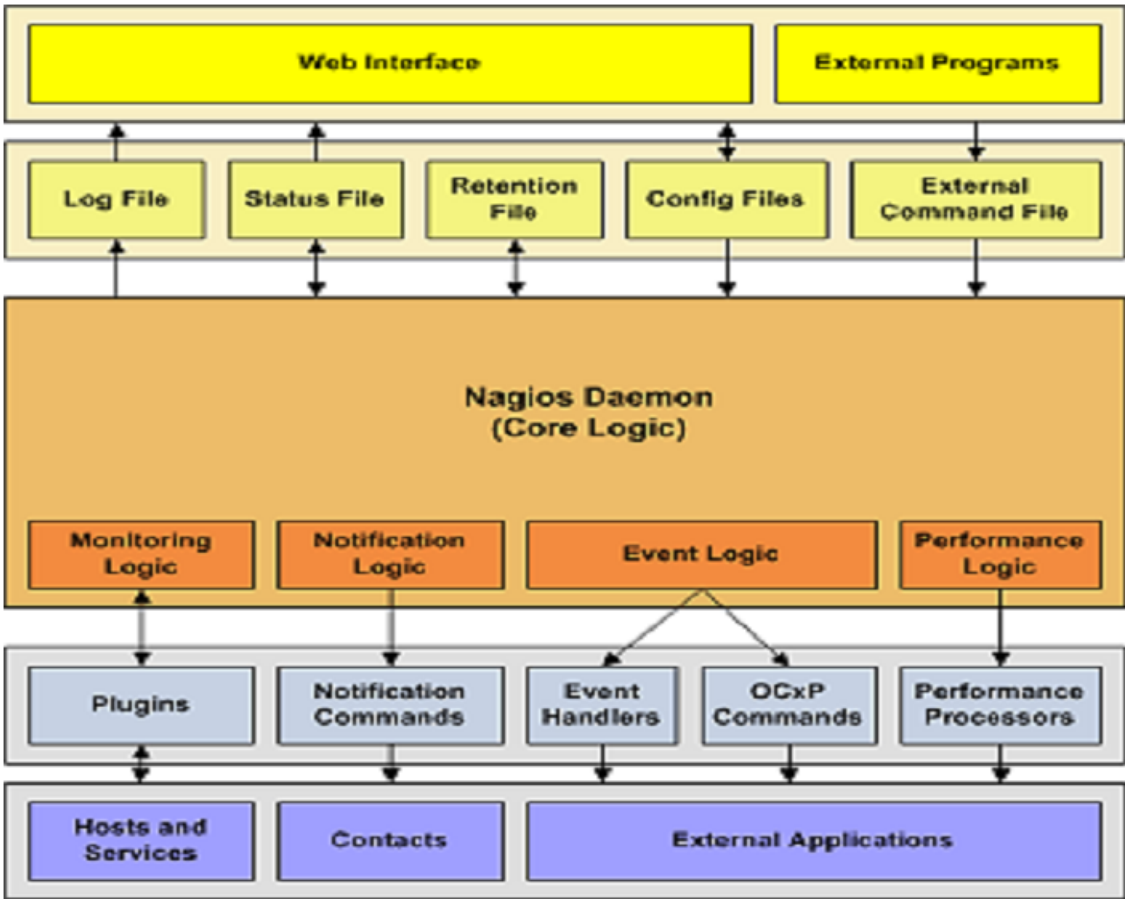

Figura 2.7: Arquitectura do Nagios [2].

A camada central (*Nagios Daemon*) constitui o principal serviço do Nagios e é responsável por toda a lógica de monitorização, notificação e de eventos. Logo abaixo encontra-se a camada de comunicação, na qual se destaca a presença de *plugins*, cujo objectivo principal é o de garantir a comunicação entre a camada central e os dispositivos monitorizados (*Hosts and Services*), representados na ultima camada. É nesta ultima camada que se encontram representados os contactos utilizados no envio das notificações. A camada imediatamente acima da central define a configuração do Nagios e é constituída por ficheiros de configuração do programa e ficheiros de *log* e *status*. Por fim, a camada superior é constituída pelos métodos de apresentação da informação monitorizada, nomeadamente a interface *Web* [24].

2.3.1.3 Funcionalidades

De entre as funcionalidades do Nagios, as que mais se destacam são as seguintes:

- **Monitorização do estado de dispositivos e serviços**: verificação regular do estado dos dispositivos, através de ICMP, e do estado dos serviços, através de verificações ao próprio serviço, como HTTP. Caso os valores recolhidos se encontrem fora do normal o estado é alterado para `WARNING` ou `CRITICAL`, consoante a gravidade do problema
- **Notificações**: permite avisar o utilizador, caso exista alguma anomalia com os objectos monitorizados, através do envio de alertas, através de e-mail ou SMS
- **Organização de Objectos**: o Nagios permite organizar os objectos monitorizados em grupos lógicos definidos pelo utilizador
- **Definição de Dependências**: as dependências são definidas manualmente pelo utilizador sob a forma de hierarquia entre os objectos monitorizados.
- **Definição de Templates**: o Nagios permite definir ficheiros de templates, que definem um conjunto de características e podem ser aplicados a vários objectos
- **Suporte para Plugins**: o Nagios possui uma grande quantidade de plugins que permite monitorizar um grande conjunto de dispositivos e serviços

2.3.1.4 Funcionamento

O Nagios utiliza um conjunto de *plugins* para efectuar as verificações de serviços e dispositivos. Os *plugins* desenvolvidos para o Nagios são responsáveis por recolher e analisar os dados monitorizados. Esta característica permite que os próprios utilizadores desenvolvam *plugins* capazes de satisfazer as suas necessidades.

Um exemplo do funcionamento do sistema de dependências do Nagios encontra-se representado na Figura 2.8. Neste caso é apresentada um rede na qual todos os dispositivos estão dependentes do switch 1. Em caso de falha no Switch 1, numa implementação sem dependências configuradas, o administrador iria receber alertas de todos os dispositivos ligados aquele switch. Se houverem dependências configuradas, apenas será enviado o alerta referente ao Switch 1.

O sistema de dependências pode ser também aplicado a serviços, como por exemplo, um serviço HTTP não estar disponível por necessitar de uma ligação a um servidor de SQL [3].

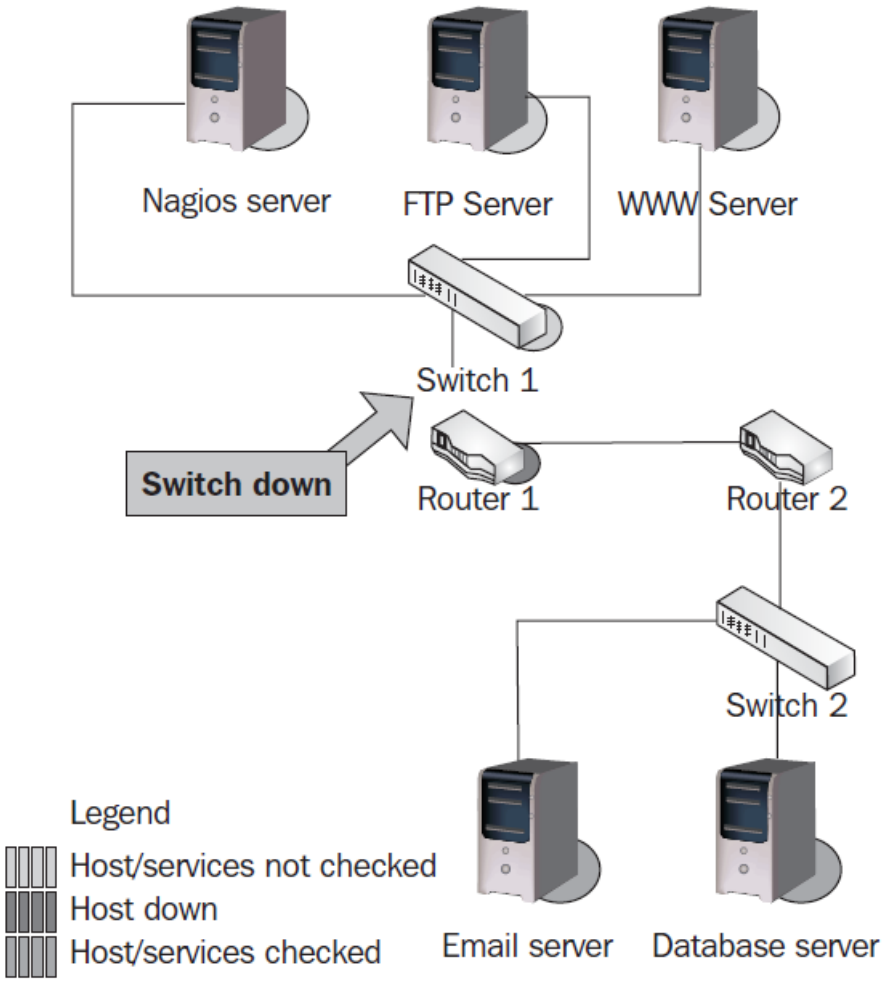

Figura 2.8: Dependências de Objectos no Nagios [3].

O sistema de estados do Nagios verifica se um objecto se encontra a funcionar e armazena o seu estado. Para esse fim, o Nagios utiliza *soft* e *hard states* para classificar o estado actual de um serviço.

Os *soft states* foram desenvolvidos de modo a não criar um alerta quando são identificados problemas durante um período muito curto de tempo, como é o caso de reiniciar o serviço HTTP. Em caso de alterações no estado, o Nagios, irá verificar o serviço novamente, várias vezes, de modo a garantir se o novo estado é o que reflecte a realidade. Após o novo estado ser reconhecido como o definitivo, é denominado de *hard state*. A Figura 2.9 demonstra a passagem dos estados perante os vários resultados obtidos, neste caso após a terceira verificação do estado, este passa de *soft* a *hard state*.

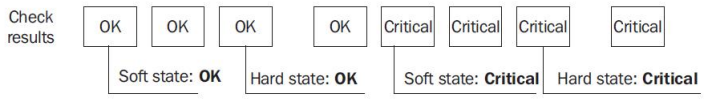

Figura 2.9: Alteração de Estados no Nagios [3].

2.3.1.5 Configuração

A configuração do Nagios é efectuada através da edição um conjunto de ficheiros de texto que podem ser divididos em ficheiros de configuração da aplicação e ficheiros de configuração de objectos.

Configuração da aplicação
O ficheiro de configuração principal do Nagios (`nagios.cfg`), que é utilizado na inicialização do programa. Possui uma sintaxe na qual as atribuições são efectuadas pelo esquema `<parâmetro> = <valor>`, como é exemplificado na Figura 2.10.

```
log_file=./var/nagios.log
cfg_file=./etc/objects/commands.cfg
resource_file=./etc/resource.cfg
```

Figura 2.10: Exemplo de sintaxe de configuração do Nagios.

Uma lista completa dos parâmetros que podem ser definidos e a sua função encontra-se no website do Nagios [10].

Configuração de Objectos
Normalmente a organização de objectos por ficheiro é efectuada pelo administrador. Por esse motivo, um bom planeamento da organização dos ficheiros de configuração permite facilitar a gestão dos equipamentos a monitorizar.

Os ficheiros de configuração de objectos permitem definir o nome e endereço do objecto, grupo a que pertence, dependências e tipos de verificações a efectuar.

2.3.2 Cacti

O Cacti [11] é uma aplicação *open source* de monitorização cuja principal funcionalidade é a análise do desempenho. É uma aplicação web e funciona como um *frontend* para o Round

Robin Database Tool (RRDTool), utilizando todas as suas potencialidades para a criação de gráficos de desempenho.

2.3.2.1 Requisitos

Para a instalação do Cacti é necessário um servidor web com suporte PHP, como o Apache ou o Internet Information Services (IIS) [25]. Os pacotes PHP necessários para o seu funcionamento são o Lightweight Directory Access Protocol (LDAP), SNMP, e MySQL.

O servidor de base de dados recomendado pelo fabricante do Cacti é o MySQL (versão 5) e a aplicação RRDTool guarda e desenha os gráficos baseados nos valores recolhidos.

O Cacti utiliza o SNMP para recolha de dados, o que implica que apenas recolhe informação disponibilizada pelas MIB do dispositivo através do agente SNMP.

2.3.2.2 Arquitectura

A arquitectura do Cacti encontra-se representado na Figura 2.11.

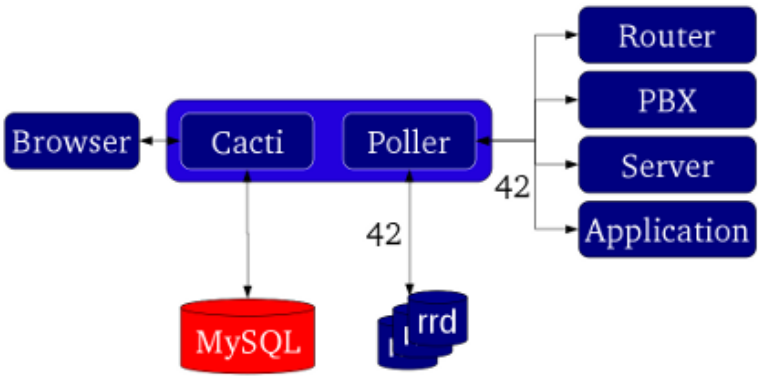

Figura 2.11: Arquitectura do Cacti [4].

A administração e utilização da aplicação é efectuada através do frontend web. Qualquer operação efectuada na aplicação é armazenado na base de dados MySQL.

O servidor web, para além de disponibilizar a interface, possui também um *poller*. A função do *poller* é efectuar recolha periódica de dados, normalmente através de SNMP, dos sistemas a monitorizar (servidores, switches, routers, etc.). Os resultados obtidos pelo *poller* são armazenados em ficheiros *rrd*, através dos quais o Cacti cria os gráficos [4].

2.3.2.3 Funcionalidades

Das funcionalidades do Cacti, destacam-se as seguintes:

- **Desenho de gráficos de desempenho**: esta á a principal funcionalidade do Cacti e é garantida pela aplicação RRDTool. Este facto possibilita que todas as formulas suportadas pelo RRDTool possam ser utilizadas no Cacti
- **Monitorização Agentless**: o *poller* do Cacti permite recolher informação referente aos dispositivos monitorizados sem recorrer a agentes
- **Suporte de Templates**: a definição de Templates permite definir um conjunto de características de recolha de dados (*DataSource*) e de definição de gráficos (*Graphics*) por tipo de dispositivo de modo a facilitar a sua configuração
- **Organização hierárquica**: os grupos de dispositivos monitorizados e respectivos gráficos são organizados numa hierarquia definida pelo administrador.

2.3.2.4 Funcionamento

A estrutura de funcionamento do Cacti pode ser dividida em três tarefas distintas demonstradas na Figura 2.12.

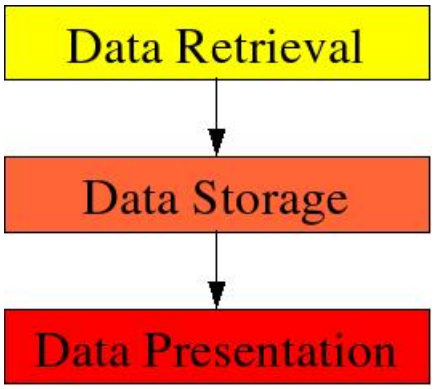

Figura 2.12: Estrutura de funcionamento do Cacti [4].

A primeira tarefa é a obtenção dos dados (*Data Retrieval*). Para esta função é utilizado o *poller*, que é executado a partir do escalonador de tarefas do sistema, como o *crontab* em sistemas UNIX.

O armazenamento de dados (*Data Storage*) no Cacti é efectuado recorrendo ao RRDTool. O RRDTool é um sistema que permite armazenar e apresentar dados organizados em janelas temporais, como é o caso de largura de banda, utilização do processador ou utilização de discos.

A apresentação final dos dados (*Data Presentation*) é efectuada através da aplicação de criação de gráficos do RRDTool [4]. Em conjunto com um servidor web, os gráficos podem ser acedidos através de qualquer browser em qualquer plataforma suportada.

2.3.2.5 Configuração

A configuração dos dispositivos a monitorizar, e respectivos gráficos é efectuada através da interface web. A interface permite adicionar novos objectos a monitorizar, os quais são associados a *templates*. Cada *template* já possui um conjunto de gráficos e *data sources* que serão aplicados ao novo objecto. O objecto pode ser editado de modo a, definir quais os gráficos do *template* que se deseja utilizar ou, adicionar novos gráficos e/ou *data sources*.

Tal como a configuração de objectos, também as definições e utilizadores da própria aplicação podem ser configurados a partir da interface web.

2.3.3 Zabbix

Nesta secção serão descritas as principais características do Zabbix [26]. É uma aplicação de monitorização que possui um agente como principal meio de recolha de dados. No entanto o Zabbix também suporta monitorização através de SNMP e Intelligent Plataform Management Interface (IPMI) [27].

2.3.3.1 Requisitos

Ao nível de software, é recomendada a utilização do servidor web Apache e um sistema de base de dados (MySQL ou PostgreSQL). De entre o *software* opcional, encontra-se o `NET-SNMP`, para verificações por SNMP e o `Iksemel`, para o envio de alertas através de Jabber.

2.3.3.2 Arquitectura

Pode-se observar pela arquitectura do Zabbix (Figura 2.13), que o elemento central é constituído pelo servidor Zabbix, a base de dados e o *frontend* Web.

Existem outros componentes opcionais que podem fazer parte de uma configuração do Zabbix, como é o caso do Zabbix *proxy*, que efectua monitorizações de dispositivos inacessíveis directamente pelo servidor Zabbix.

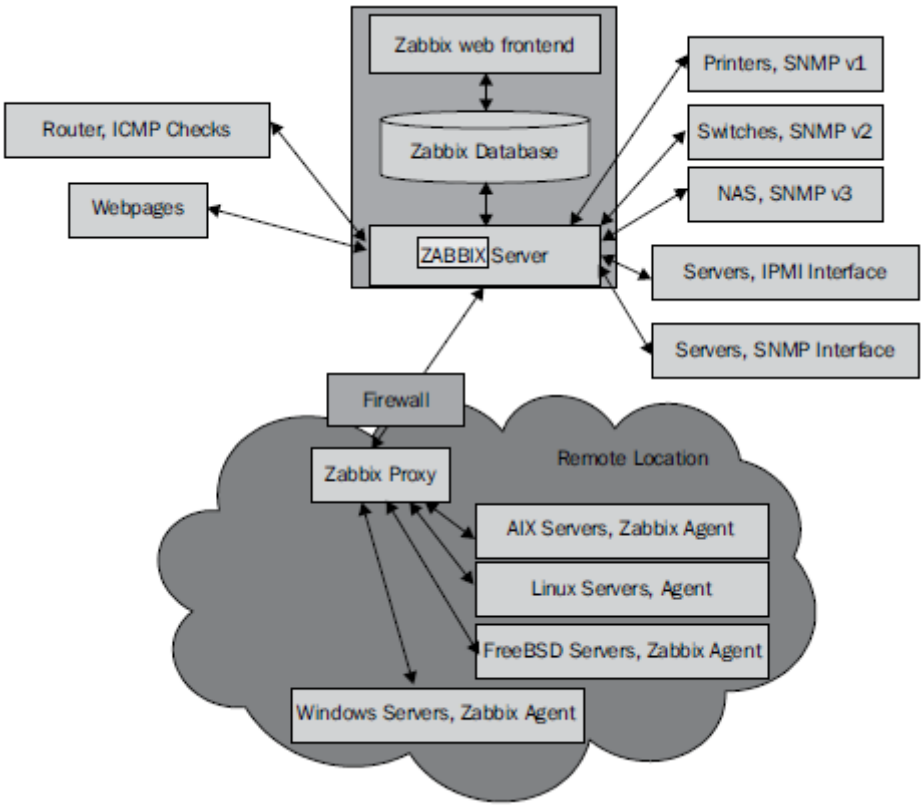

Figura 2.13: Arquitectura do Zabbix [5].

O servidor Zabbix está encarregue das verificações dos dispositivos monitorizados, do tratamento dos dados recolhidos e dos mecanismos de alerta. Apesar de não existirem problemas em executar estes três componentes num único servidor, estes podem-se separar de modo a tirar proveito de um servidor de base de dados e/ou de um servidor Web já existentes.

Os dispositivos são directamente monitorizados pelo servidor Zabbix através de diversos protocolos e métodos de comunicação como o SNMP, o IPMI ou agentes do Zabbix. Os agentes apenas podem ser instalados nos dispositivos suportados (PCs). O agente recolhe localmente informação dos componentes do sistema e envia-os ao servidor Zabbix.

O Zabbix permite monitorizar dispositivos que não se encontrem directamente acessíveis como, por exemplo, em localizações remotas ou protegidas por firewall através do Zabbix Proxy. O Zabbix proxy recolhe os dados de uma localização remota e envia-os ao servidor Zabbix. O Zabbix Proxy não é um componente obrigatório e pode ser utilizado para balanceamento de carga.

Apesar dos componentes do Zabbix poderem estar distribuídos, a sua configuração é centralizada, diminuindo, deste modo, a probabilidade de configurações incoerentes entres componentes [5].

2.3.3.3 Funcionalidades

As principais funcionalidades do Zabbix são as seguintes:

- **Vários modos de monitorização**: o Zabbix permite monitorizar a infra-estrutura de rede através de protocolos destinados a monitorização, como SNMP ou IPMI, ou através de um agente próprio.
- **Desenho de Gráficos de desempenho**: os dados recolhidos pela monitorização podem ser utilizados para definição de gráficos de desempenho
- **Alertas**: tal como o Nagios o Zabbix permite o envio de alertas baseado em vários parâmetros, como anomalia de valores, juntamente com o período de tempo em que a anomalia ocorreu. Deste modo é possível definir, com grande granularidade, os alertas a despoletar
- **Auto-discovery**: o Zabbix permite fazer uma procura na rede e adicionar automaticamente os dispositivos encontrados. Como contra-partida, esta funcionalidade apenas se aplica a dispositivos que possuam o agente Zabbix instalado
- **Suporte de Templates**: os templates que o Zabbix possui permitem definir os parâmetros a recolher, bem como os gráficos e *triggers* associados

2.3.3.4 Funcionamento

O Zabbix efectua verificações regulares aos itens monitorizados e compara o valor recolhido com um *trigger* definido. Caso o valor monitorizado active um *trigger*, o estado do item passa a `PROBLEM` e é criado um Evento. Quando um evento corresponde a todas as condições de uma Acção (`Action`), esta é executada. Uma Acção contem uma ou mais operações que

definem qual a resposta a dar, tal como o envio de um alerta ou a execução de um comando. Quando o valor monitorizado volta ao normal, o estado passa a `OK`.

A Figura 2.14 demonstra o processamento de *triggers* e execução de acções. Neste caso após uma alta carga de CPU durante 3 minutos, é enviado um e-mail ao utilizador.

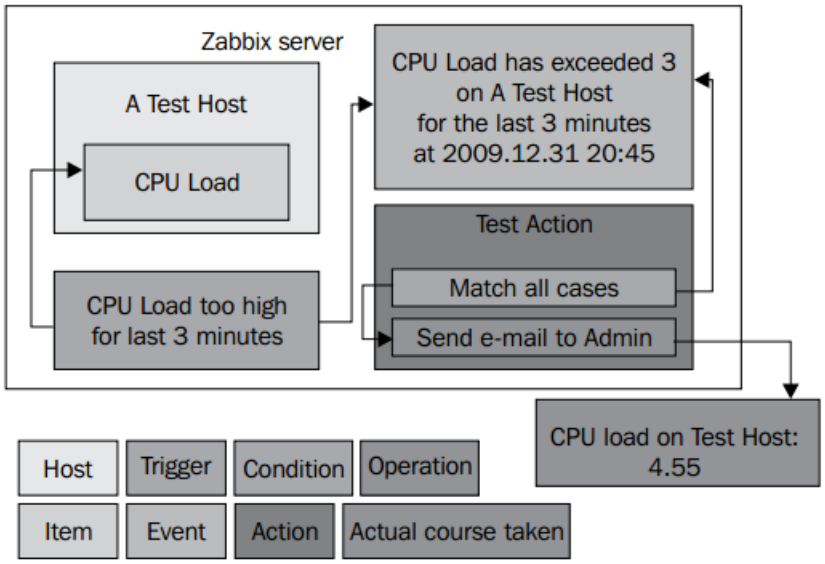

Figura 2.14: Processamento de *triggers* no Zabbix [5].

2.3.3.5 Configuração

Todas as opções de configuração são efectuadas através do *frontend* Web.

A configuração dos parâmetros a recolher, definição de gráficos e definição de *triggers* encontram-se definidos em templates.

A definição de acções a executar requerem a existência de *triggers* devidamente configurados. Estas podem ser baseadas em apenas um *trigger* ou num conjunto de *triggers*.

2.3.4 Zenoss

Zenoss [28] é uma aplicação *open-source* de monitorização de sistemas e serviços. O Zenoss Core é uma de duas versões da aplicação desenvolvidas pela Zenoss, Inc, e é mantida pela comunidade. A versão Zenoss Enterprise possui algumas funcionalidades acrescidas, que a tornam numa aplicação comercial e é mantida pela própria Zenoss, Inc.

O Zenoss Core não utiliza agentes próprios, efectuando a monitorização dos dispositivos através de SNMP, WMI ou execução de comandos remotos através de SSH.

2.3.4.1 Requisitos

O Zenoss apenas é compatível com sistemas Linux ou MacOS, de modo que a execução num sistema Windows, apenas é possível utilizando uma maquina virtual (*virtual appliance*).

Como requisitos mínimos de software, o Zenoss necessita do MySQL e bibliotecas SNMP, como o lib-net-SNMP [6].

2.3.4.2 Arquitectura

Tal como demonstra a Figura 2.15 , a arquitectura do Zenoss Core encontra-se dividida em 3 camadas principais: camada de utilizador (*User Layer*); camada de dados (*Data Layer*) e camada de recolha (*Collection Layer*) [6].

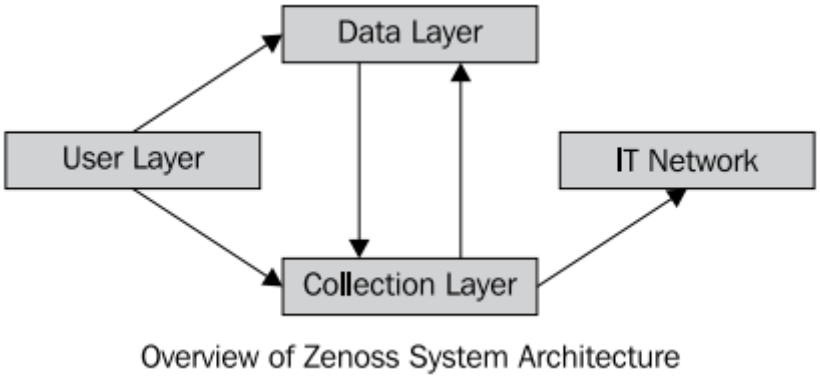

Figura 2.15: Arquitectura do Zenoss [6].

A camada de utilizador é utilizada pela interface web ou linha de comandos. A interface web é baseada na *framework* aplicacional Zope é um modo de input tanto para as camadas de dados como de recolha, de modo a permitir certas tarefas, como gerir dispositivos e eventos, monitorizar desempenho, criar relatórios e configurar a própria aplicação.

A camada de dados do Zenoss armazena dados em três tipos de bases de dados. Os dados recolhidos, pela camada de recolha, são enviados para o a aplicação ZenHub, que separa os dados e os armazena no local mais apropriado (Figura 2.16).

Quando os eventos atingem um limiar definido, são armazenados numa base de dados

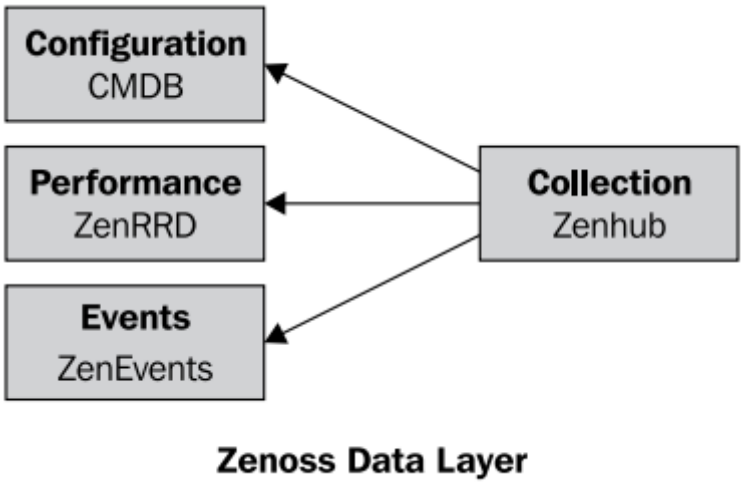

Figura 2.16: Camada de Dados do Zenoss (Data Layer) [6].

MySQL. Outros eventos, por sua vez, podem gerar acções, como o envio de e-mail ou SMS. Os dados de desempenho recolhidos são armazenados em ficheiros RRD através da RRDTool.

A terceira base de dados utilizada pelo Zenhub é denominada Configuration Management Database (CMDB) e armazena as configurações dos dispositivos monitorizados. A CMDB é standard de Information Technology Infrastructure Library (ITIL) para gestão em ambientes TI [29].

Na camada de recolha de dados (Figura 2.17) encontram-se os vários *deamons* destinados a recolher informação de desempenho e eventos, que são fornecidos ao Zenhub. Para uma fácil identificação, todos os *daemons* do Zenoss Core são designados através do prefixo "Zen".

2.3.4.3 Funcionalidades

As principais funcionalidades do Zenoss são as seguintes:

- **Identificação automática de componentes**: o Zenoss detecta automaticamente os componentes de cada dispositivo e inicia a sua monitorização. Uma verificação periódica e automática identifica alterações nos dispositivos e actualiza os seus componentes.
- **Monitorização agentless**: a monitorização dos dispositivos é efectuada sem recorrer

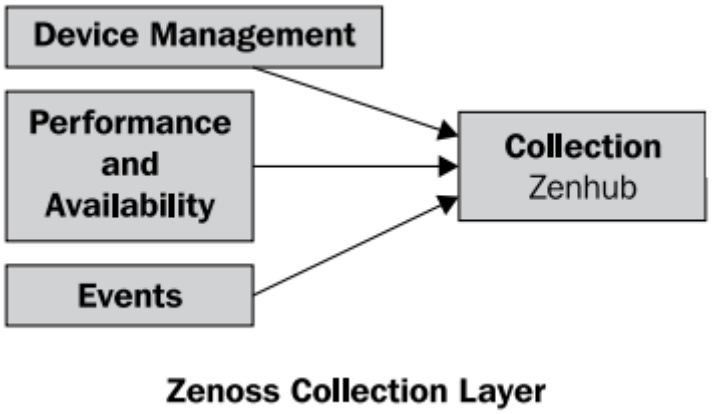

Figura 2.17: Camada Recolha de Dados do Zenoss (Collection Layer) [6].

a agentes da aplicação, utilizando apenas protocolos de monitorização como SNMP ou WMI.

- **Suporte de plugins**: o Zenoss possui vários *plugins* próprios, denominados *ZenPacks*, que permitem aumentar as funcionalidades da aplicação
- **Alarmes**: os alarmes permitem ser definidos de acordo com várias características para além dos valores anormais, tais como grupo de dispositivo, repetições do evento ou altura em que ocorreu. Os principais métodos de envio de alarme do Zenoss são e-mail e SMS

2.3.4.4 Funcionamento

O Zenoss Core baseia-se num conjunto de *deamons* de modo a executar as várias tarefas de recolha, tratamento e apresentação de dados.

Os *daemons* que constituem o Zenoss podem ser divididos em:

- Gestão de dispositivos;
- Performance e disponibilidade;
- Eventos.

Os *daemons* responsáveis pela gestão de dispositivos, efectuam funções de descoberta, configuração e modelação de dispositivos. A modelação dos dispositivos permite identificar quais os seus componentes e iniciar a respectiva monitorização.

A monitorização dos dispositivos e detecção de eventos é efectuada por diferentes processos da aplicação consoante o método de monitorização, nomeadamente, SNMP, execução de scripts, ICMP, verificação de processos e verificação de serviços.

A detecção e tratamento de eventos externos também é efectuada por diferentes processos da aplicação consoante o tipo de evento, nomeadamente, *syslog* (Linux), *traps* (SNMP) ou *eventlog* (Windows).

2.3.4.5 Configuração

A interface permite a adicionar e configurar dispositivos, gerir eventos, visualizar ou gerar relatórios sob os dados de monitorização recolhidos.

A interface web permite a gestão da própria aplicação e dos correspondentes *daemons*, bem como as configurações necessárias. Apenas algumas funcionalidades não podem ser configuradas e/ou executadas a partir da interface web e necessitam de ser executados por ambiente Comand Line Interface (CLI), um exemplo deste tipo de funcionalidades é o envio de relatórios para o e-mail (`reportmail`).

2.3.5 Shinken

Shinken [30] é uma aplicação de monitorização, desenvolvida em Python, baseada no Nagios. Por esse motivo é compatível com todos os plugins, interfaces e configurações destinados ao Nagios.

2.3.5.1 Requisitos

Tal como o Nagios, os requisitos necessários para executar o Shinken é um dispositivo com sistema Linux, compilador de C, comando *make*, biblioteca de C e implementação da pilha protocolar TCP/IP, um servidor Web, sendo o Apache a opção recomendada e instalação de Python juntamente com o módulo Pyro.

No caso da utilização de certos *plugins*, pode ser necessário instalar dependências, como é o caso do NET-SNMP ou open-SSL. Para a utilização da funcionalidade de *auto-discovery*, o Shinken necessita da instalação do Nmap [31].

2.3.5.2 Arquitectura

A grande diferença do Shinken em relação ao Nagios, é a sua arquitectura. Enquanto que o processo do Nagios executa todas as funcionalidades básicas da aplicação, no Shinken, estas funcionalidades são divididas por vários processos, tornando a arquitectura desta aplicação mais distribuída. Os processos criados com a distribuição das funcionalidades no Shinken, são executados como *deamons* independentes que comunicam entre si [32], como se pode observar na Figura 2.18.

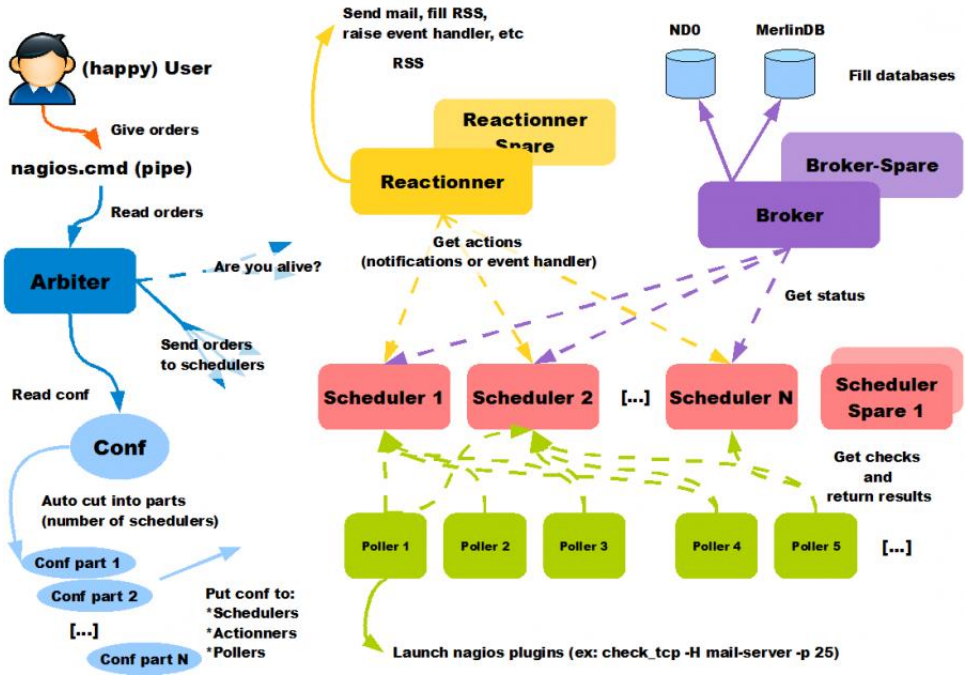

Figura 2.18: Arquitectura Shinken [7].

Os componentes da arquitectura do Shinken são:

- **arbiter:** *daemon* que está encarregue de ler a configuração, dividi-la e distribui-la pelos restantes componentes. Gerir a disponibilidade dos restantes *daemons* e reencaminhar os pedidos para um *daemon* de reserva. Esta arquitectura apenas permite que um **arbiter** seja executado.

- **Scheduler:** responsável por escalonar verificações aos dispositivos monitorizados, analisar os dados recolhidos e executar acções. Este *daemon* mantém uma fila de espera de verificações e eventos de outros componentes da arquitectura (pollers ou reactionners). A arquitectura do Shinken permite que existam várias instâncias de Schedulers em execução.

- **Poller:** executa verificações pedidas pelo scheduller. Após a execução da verificação o resultado é enviado ao Scheduler. Podem existir várias instâncias de Pollers.
- **Reactioner:** é responsável por notificações e lançar eventos com base nos dados fornecidos pelo scheduller.
- **Broker:** tem como objectivo recolher e gerir os dados dos Schedulers. Estas tarefas são efectuadas por módulos que variam consoante o formato que se deseja armazenar, como MySQL ou Oracle.

2.3.5.3 Funcionalidades

Como o Shinken é uma aplicação baseada no Nagios apresenta todas as suas funcionalidades. Tal como no Nagios, as funcionalidades podem ser aumentadas através da utilização de *plugins*, quer sejam desenvolvidos para o Shinken, quer sejam desenvolvidos para Nagios. As funcionalidades específicas do Shinken, incluem o sistema de *auto-discovery* e suporte nativo para monitorização de ambientes virtuais, como o VMWare ESX [33].

2.3.5.4 Funcionamento

Tal como foi referido o Shinken utiliza vários processos, cada um com a sua funcionalidade [34]. O Arbiter lê a configuração, divide-a e aplica-a aos vários Schedulers activos. Em simultâneo é verificado se todos os Schedulers se encontram em funcionamento e, caso algum tenha terminado inesperadamente é substituído por um Scheduler de reserva. (Figura 2.19)

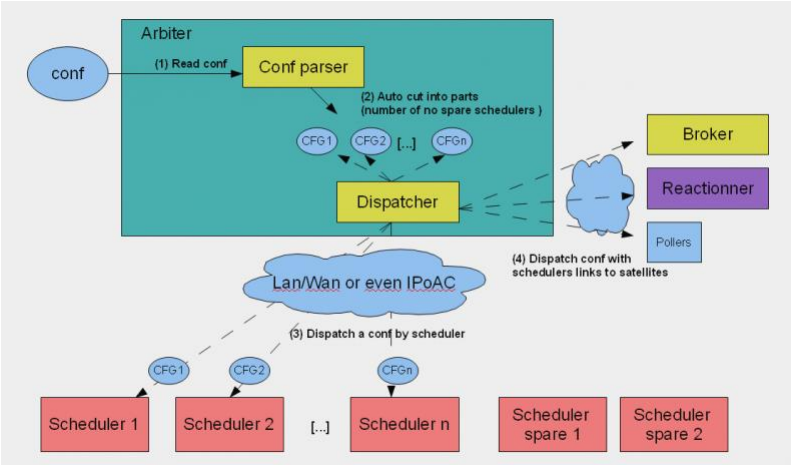

Figura 2.19: Método de Funcionamento do Shinken [8].

No caso do tratamento dos comandos do utilizador, como é o caso de agendamento de um período de manutenção ou necessidade de efectuar uma verificação, o Arbiter envia o comando apenas ao Scheduler responsável. No caso do comando ser identificado, pelo Arbiter, como um comando destinado a todos os dispositivos, este é enviado a todos os Schedulers que se encontrem activos. (Figura 2.20)

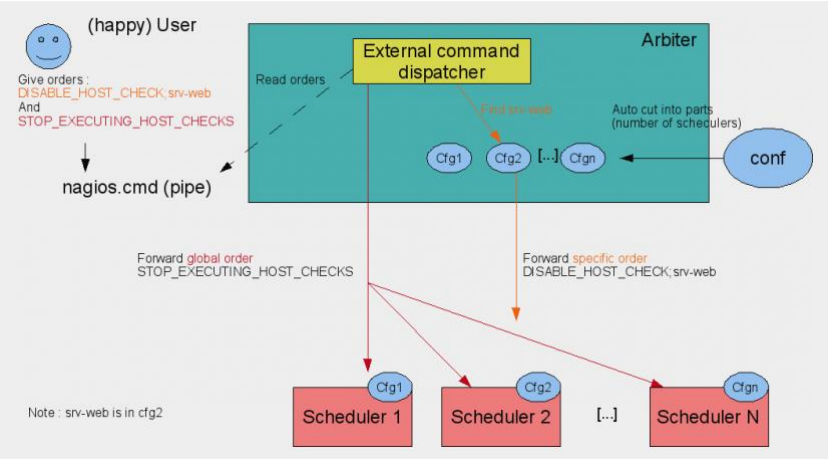

Figura 2.20: Execução comandos no Shinken [8].

2.3.5.5 Configuração

À semelhança do Nagios, a configuração do Shinken, é efectuada por vários ficheiros de configuração que permitem configurar a aplicação e os dispositivos monitorizados.

A principal diferença na configuração em relação ao Nagios é a existência de um ficheiro de configuração especifica do Shinken, que definem as configurações do Arbiter, Scheduler, Pollers, Brokers, e Reactioners do Shinken.

Os ficheiros de configuração de objectos possuem a mesma sintaxe do Nagios, pelo que podem ser configurados utilizando as mesmas directivas [35].

2.4 Sumário

Após a analise de todas as aplicações de monitorização pode-se observar que grande parte das funcionalidades são partilhas pelas aplicações, permitindo a monitorização de vários dispositivos, bem como funcionalidades de alarmística. Apesar de muitas das funcionalidades serem partilhas entre as aplicações analisadas, o seu modo de funcionamento e de

configuração são bastante distintos.

Enquanto que as aplicações Nagios e Sinken dependem de plugins para efectuar a recolha ou apresentação de dados ao utilizador, as restantes fazem a recolha de dados e apresentação sem necessidade de aplicações externas. No entanto permitem estender as suas funcionalidades através dessa abordagem.

No campo da configuração das aplicações também existe um grande diferença, principalmente entre o Nagios ou Shinken e as restantes aplicações. Enquanto que as configurações do Nagios ou Shinken são definidas através da edição de ficheiros de texto, o que requer um conhecimento da sintaxe e estrutura da configuração, as restantes aplicações fazem uso de *interfaces* gráficas para a sua configuração. Esta abordagem requer um menor esforço por parte do utilizador, uma vez que é mais intuitiva e fácil de utilizar. O modo de configuração pode ser um factor determinante na escolha de uma aplicação, uma vez que a sua utilização depende da interacção regular com os utilizadores.

A facilidade de utilização da própria aplicação, também é um factor importante, principalmente na interacção com o utilizador. Uma aplicação que possua tarefas automatizadas, como descoberta de dispositivos e identificação de componentes a monitorizar por cada dispositivo, permite que o utilizador se concentre mais na informação recolhida e menos na configuração dos parâmetros a recolher. Das aplicações analisadas, o Zabbix, o Zenoss e o Shinken, possuem funcionalidades de descoberta de dispositivos, apesar do Zabbix apenas detectar dispositivos que possuam o seu agente. No que toca à configuração automática de parâmetros, apenas o Zenoss possui essa funcionalidade, que permite definir quais os parâmetros a monitorizar de acordo com o tipo de equipamento (servidor, maquina virtual, switch, etc).

Outras das funcionalidades que podem ser desempenhadas por este tipo de aplicações é o inventário da infra-estrutura de rede. Esta funcionalidade é uma mais valia, pois permite ter uma ideia geral dos componentes de uma rede. Apesar de todas as aplicações recolherem as características dos vários componentes da rede, a aplicação que o faz de forma mais autónoma é o Zenoss, pois consegue identificar automaticamente as principais características de cada dispositivo, enquanto que as restantes aplicações necessitam da intervenção do utilizador para definir quais as características que devem ser recolhidas.

Capítulo 3

Análise Comparativa das Aplicações

Com o objectivo de testar as aplicações analisadas no capítulo anterior foi definido um cenário de testes. Este cenário pretende apresentar os principais dispositivos, respectivos sistemas operativos e organização da rede de produção na Linkcom.

A heterogeneidade dos sistemas que se encontram na rede da Linkcom foi tido em conta no desenho do cenário de testes, de modo a permitir testar as aplicações e o seu comportamento quando utilizadas num ambiente real. Nesse sentido foram avaliados vários tipos de equipamentos de rede, como switches, routers ou servidores. No que diz respeito aos sistemas operativos dos servidores foram utilizados Windows e Linux.

De seguida é descrita a arquitectura da rede de testes e a abordagem tomada para análise das aplicações.

3.1 Arquitectura da Rede de Testes

A arquitectura de rede utilizada durante o teste das aplicações possui um conjunto heterogéneo de equipamentos que visam representar a diversidade de equipamentos existentes na rede da Linkcom.

Os seguintes equipamentos foram utilizados na arquitectura da rede de testes:

- Servidor de monitorização;
- Servidores aplicacionais;
- Equipamentos activos de rede (Switch, Firewall, Router, etc);

- Outros dispositivos de rede, como UPS ou Sensor de Temperatura.

A Figura 3.1 ilustra a arquitectura de rede utilizada para os testes:

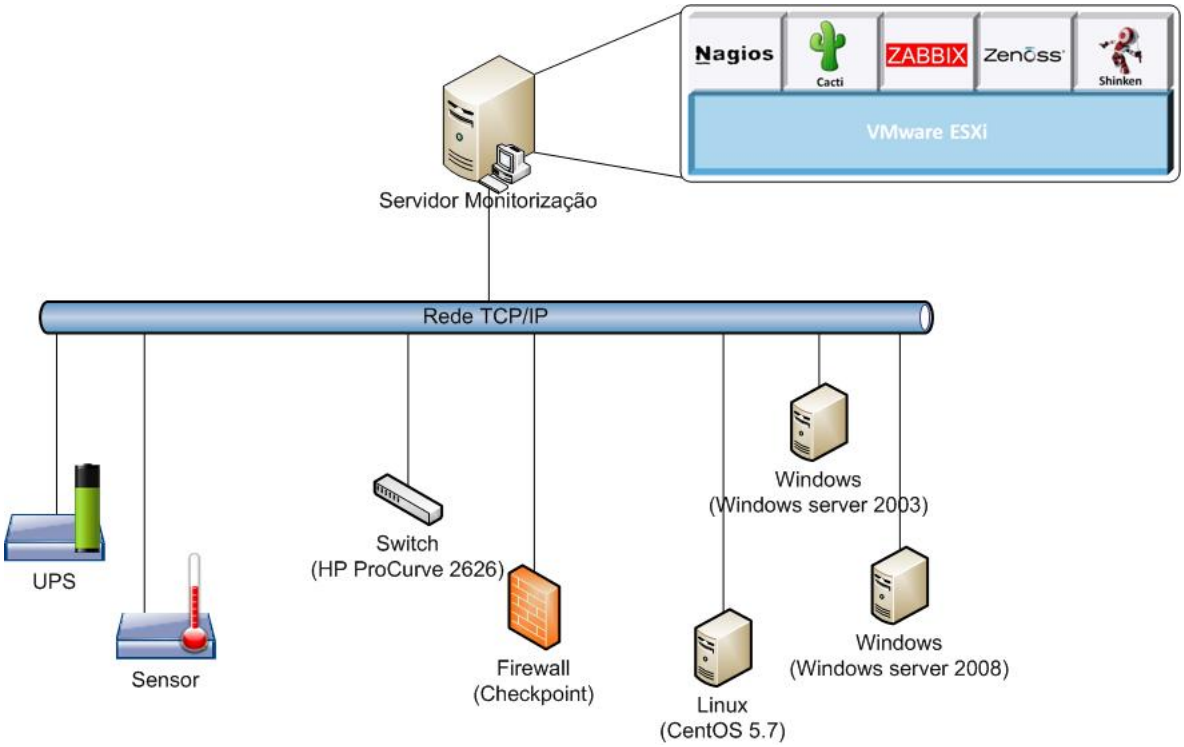

Figura 3.1: Arquitectura de Rede de testes.

O servidor de monitorização é um equipamento onde foi instalado o VMWare EXSi com várias instâncias destinadas a cada uma das aplicações de monitorização. Em cada uma das instâncias foi instalado o sistema operativo Linux CentOS [36]. A escolha desta distribuição prende-se com o facto de ser uma distribuição orientada para ambientes empresariais e ser a distribuição de Linux utilizada em ambiente de produção na Linkcom, diminuindo, desta forma, a discrepância entre o ambiente de testes e o ambiente de produção.

Os três servidores aplicacionais utilizados para os testes de monitorização possuem diferentes sistemas operativos, nomeadamente CentOS 5.7 e Windows Server, versões 2003 e 2008. A diversidade de sistemas operativos é utilizada para verificar os diferentes métodos de recolha de dados e os parâmetros que cada um dos sistemas operativos disponibiliza.

O switch HP Procurve 2626 e a firewall Checkpoint foram utilizados para representar equipamentos nos quais não é possível instalar software que permita disponibilizar informação relativa à monitorização destes dispositivos. Nestes casos os equipamentos apenas disponibilizam, através de SNMP, as informações presentes na respectiva MIB.

De seguida serão descritas as várias categorias que compõem a arquitectura de rede e as suas respectivas características.

3.1.1 Servidor de Monitorização

O servidor de monitorização é o computador onde se encontra instalada a aplicação de monitorização utilizada para os testes. Neste caso foram utilizadas as aplicações Nagios, Cacti, Zabbix, Zenoss e Shinken.

Para rentabilizar os recursos existentes, foi utilizado o sistema de virtualização VMWare ESXi [37] num computador com um processador Intel Core 2 Duo 6300 com velocidade de 1.86GHz e 4GB de RAM, como representado na Figura 3.2.

De modo a analisar as aplicações de forma independente foi criado uma máquina virtual para cada aplicação de monitorização (Figura 3.2). As máquinas virtuais possuem uma instalação do sistema operativo CentOS 5.7 com 1GB de RAM e 15 GB de espaço em disco. Cada aplicação foi testada isoladamente através de máquinas virtuais, permitindo assim analisar o comportamento e desempenho das várias aplicações em ambientes similares.

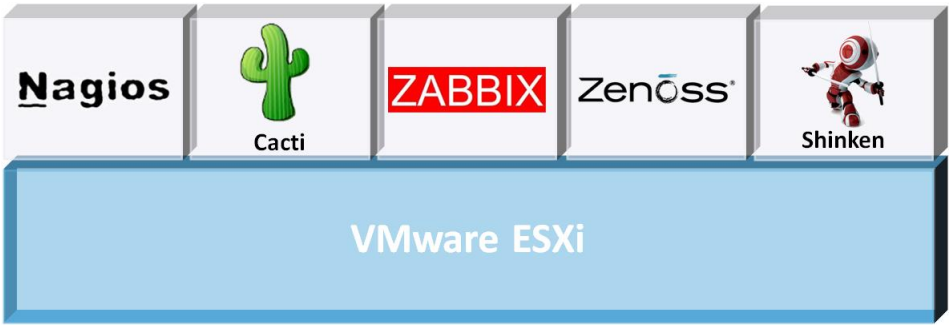

Figura 3.2: Máquinas virtuais no Servidor de Monitorização.

3.1.2 Servidores Aplicacionais

O conjunto de equipamentos que compõem o grupo "Servidores"fornecem essencialmente servidos de rede como, por exemplo, HTTP ou DNS, os seus sistemas operativos e componentes são heterogéneos de modo a poder testar o suporte das aplicações de monitorização para os diferentes sistemas. Neste caso foram utilizados 3 sistemas operativos e versões diferentes. Para os servidores Windows foram utilizadas as versões Server 2003 e Server 2008, quanto aos sistemas Linux foi utilizado o CentOS 5.7.

Os servidores deste grupo possuem o serviço SNMP ou WMI activos de modo que o servidor de monitorização possa recolher dados através deste protocolo.

Na classe servidores optou-se por monitorizar os seguintes parâmetros:

- Utilização CPU;
- Utilização memória RAM;
- Utilização disco/partições;
- Operações I/O de discos;
- Estado dos serviços;
- Disponibilidade.

Estes parâmetros foram seleccionados devido à natureza dos equipamentos e à utilização que fazem de cada um dos recursos monitorizados [38] A monitorização destes parâmetros permite verificar a utilização dos vários componentes do servidor em tempo real. Estas métricas permitem ainda decidir se é necessário substituir ou actualizar os componentes de *hardware e software* dos servidores.

3.1.3 Equipamentos Activos de rede

Os equipamentos activos de rede são compostos por dispositivos que têm um papel activo no funcionamento da rede, como por exemplo, switches, routers ou firewalls.

Neste tipo de dispositivos optou-se por monitorizar os seguintes parâmetros:

- Utilização do CPU;
- Utilização da memória RAM;
- Tráfego das interfaces;
- Total de endereços MAC reconhecidos por switches (MACCount);
- Disponibilidade.

Os parâmetros monitorizados nesta categoria de equipamento permitem identificar a carga de tráfego que circula na rede, bem como o número de endereços físicos (MAC) reconhecidos pelo equipamento.

3.1.4 Outros Dispositivos de Rede

Esta categoria, mais genérica, pretende englobar os dispositivos presentes na rede que não se enquadram em qualquer uma das anteriores por possuírem características únicas, como por exemplo, Uninterruptible Power Supply (UPS) e sensores térmicos e de humidade que foram utilizados neste cenário.

Assim foi possível recolher a seguinte informação acerca deste tipo de dispositivos:

UPS

- Carga actual da bateria;
- Estado da Bateria;
- Disponibilidade.

Sensor Térmico

- Temperatura;
- Humidade;
- Disponibilidade.

3.2 Templates

Os templates são utilizados para definir um conjunto de configurações que são aplicadas aos dispositivos monitorizados. De modo a facilitar a configuração de cada tipo de dispositivo, existe um template que define quais os parâmetros a recolher e respectivos métodos de recolha. Deste modo é possível definir um conjunto de características referentes a cada tipo de dispositivo e aplicar o template mais adequado a cada caso.

Este tipo de abordagem permite facilitar as tarefas de configuração de cada dispositivo, pois apenas é necessário utilizar o template adequado para que as tarefas de monitorização adequadas sejam aplicadas ao dispositivo.

Os templates foram definidos de modo a utilizar os agentes presentes nos diferentes tipos de equipamentos. Assim, foi utilizado o SNMP como principal modo de recolha de dados referentes aos vários equipamentos e o WMI como método de recolha de dados para os servidores Windows.

3.3 Utilização das Aplicações de monitorização

De modo a efectuar uma análise mais profunda às aplicações, estas foram instaladas e configuradas de modo a monitorizar os equipamentos que constam na arquitectura de rede descrita anteriormente na secção 3.1.

De seguida é apresentada uma análise da experiência de utilização das várias aplicações, tendo em conta os seguintes aspectos:

- Configuração;
- Gestão de utilizadores
- Monitorização de objectos
- Acesso via Web
- Gestão de Alertas e Eventos

3.3.1 Nagios

O Nagios é uma das aplicações que já se encontra em utilização na Linkcom, no entanto foi instalado num ambiente de testes de modo a efectuar uma comparação no mesmo cenário de todas as outras aplicaçoes.

Uma vez que a instalação do Nagios é efectuada a partir do código fonte são necessários conhecimentos básicos de Linux e da utilização de compiladores, mais propriamente o `gcc`. A documentação é clara e aplica-se à ultima versão da aplicação disponibilizada pelo fabricante, o que facilita a resolução de possíveis problemas que possam surgir.

A configuração da aplicação tem que ser efectuada através de ficheiros de configuração do próprio Nagios, o que implica conhecimento da sua sintaxe específica. Apesar de simples esta tarefa pode tornar-se repetitiva e entediante, no entanto podem ser utilizados scripts em Bash ou Perl de modo a automatizar algumas tarefas de configuração do Nagios.

Para além destas características, qualquer alteração às configurações do Nagios requerem uma verificação manual às configurações e execução do comando que as aplique.

Os utilizadores suportados pelo Nagios são configurados através do servidor web Apache, pois é este que efectua a autenticação dos utilizadores. A definição de contactos é configurada em ficheiros de configuração apropriados e inclui os meios de contacto, nomeadamente

e-mail e SMS e permite a atribuição de utilizadores a objectos individuais ou grupos de objectos.

A interface Web do Nagios é orientado para consulta dos objecto monitorizados e gestão de eventos. Como descrito no capítulo 2 não é possível editar as configurações dos objectos monitorizados, nem adicionar novos objectos através do frontend web.

Uma das principais vantagens deste frontend é a visualização geral dos estados dos objectos, permitindo identificar rapidamente qual ou quais os objectos a funcionar incorrectamente.

Apesar de não suportar algumas funções consideradas de configuração básicas, como modificação, remoção ou criação de objectos, é um frontend bastante intuitivo.

Tal como referido no capítulo 2, os alertas são configurados no objecto a monitorizar através da definição de valores de *threashold* para o estados `WARNING` e `CRITICAL`. O envio de alertas é automaticamente configurado para o contacto configurado para cada dispositivo ou grupo de dispositivos.

O suporte para o período de manutenção, disponível através da interface web, permite definir intervalos de tempo em que não são enviados alertas.

Um alerta pode despoletar a execução de um comando ou envio do alerta consoante a acção configurada.

3.3.2 Cacti

À semelhança do Nagios, o Cacti também se encontra em ambiente de produção. Os motivos para uma instalação num ambiente de teste prende-se com os motivos de comparação.

A instalação do Cacti em ambiente Linux não apresenta dificuldades e é suportada por uma boa documentação. São necessários conhecimentos de MySQL, para a configuração da base de dados e de servidor Web Apache, para instalação do servidor Web.

A configuração e execução de tarefas no Cacti são suportados pela interface web. Esta abordagem permite que a configuração do Cacti seja mais intuitiva quando comparada com o Nagios.

A configuração da monitorização dos dispositivos é efectuada através da utilização de templates.

Baseado na experiência própria, os templates de *data sources* e gráficos são os mais difíceis de criar, em contrapartida os templates de hosts resultam da selecção de um conjunto de

templates de gráficos de de *data sources*.

Os utilizadores do Cacti são configurados de modo manual ou através da integração com LDAP.

A definição de níveis de acesso utiliza um conjunto de perfis de acesso, nomeadamente anónimo, normal e administrador, que podem ser aplicados a utilizadores ou grupos. O acesso pode ainda ser personalizado através da definição personalizada de permissões de execução e visualização de várias tarefas.

No Cacti, cada objecto é configurado de forma unitária, seleccionando o template adequado. Os objectos podem ser personalizados através da adição de mais templates gráficos ou de *data sources*.

A interface web do Cacti permite ter um total controlo sob as configurações da aplicação, tornando-se mais *user-friendly* na configuração das várias opções disponibilizadas.

A interface peca nalgumas configurações, em que é necessário mais clicks que o ideal. Outra das características menos intuitiva é a organização das interfaces de gestão de gráficos, as quais são separadas por várias interfaces, tornando-se necessário navegar até outro interface para uma acção diferente sob o gráfico, como a sua remoção.

Nativamente, o Cacti não suporta detecção de alertas. Esta funcionalidade pode ser adicionada através do plugin "thold".

Este plugin permite definir valores de *threashold* para os valores monitorizados. O plugin acrescenta um interface de gestão de eventos e a possibilidade de envio de alertas. No entanto o envio de alertas apenas se encontra disponível por e-mail.

3.3.3 Zabbix

O Zabbix é uma aplicação que combina as funcionalidades do Nagios com as do Cacti, permitindo a monitorização de desempenho e disponibilidade e suporte a alertas.

A instalação do Zabbix foi efectuada através da compilação de código. É necessário ter em atenção a base de dados a instalar, pois várias são suportadas, bem como algumas funcionalidades extra, como o suporte Jabber.

A configuração de parâmetros da aplicação é efectuada através do frontend web. A sua total dependência do frontend web impossibilita que grande parte das funções sejam efectuadas a partir da linha de comandos, tal como execução de scripts de automatização de configura-

ções.

Por omissão existem três perfis de utilizadores (User,Admin e Super Admin), o que permite definir o controlo de acesso e execução de funções.

Os utilizadores podem ser criados manualmente ou através de integração com LDAP e podem ser agrupados, independentemente do tipo de utilizador.

O Zabbix suporta vários tipos de monitorizações nativamente. Desde métodos agentless, como SNMP ou IPMI, ou com recurso ao agente da aplicação. Qualquer um dos métodos é configurado através do frontend web e permite definir o intervalo de verificações para cada parâmetro individualmente.

Uma vez que é orientado para o agente, muitos dos templates que se encontram no Zabbix são destinados a essa abordagem. Esta característica implica que sejam criados templates para outros métodos de monitorização como é o caso do SNMP.

O frontend web do Zabbix permite efectuar todas as configurações da aplicação, por esse motivo contem bastante informação e é apresentada sem grande tratamento. Esta característica pode provocar alguma confusão e falte de clareza de algumas opções.

O frontend web é rápido e simples, no entanto algumas funcionalidades requerem um elevado número de "point & click".

As condições de definição de um alerta são altamente configuráveis através de comparações, períodos temporais, impacto do evento ou dependências entre parâmetros monitorizados. Para cada alerta deste tipo podem ser definidas várias acções, desde a execução de um comando ao envio de uma mensagem a um utilizador.

3.3.4 Zenoss

O Zenoss à semelhança do Zabbix possui funcionalidade de monitorização de desempenho e de estado. No entanto não utiliza agente próprio para a monitorização.

O Zenoss suporta instalação através de um aplicativo do género "All in One", em que apenas é necessário instalação dos pré-requisitos e execução do aplicativo de instalação.

A documentação é bastante clara e detalhada para a instalação e configuração da aplicação.

A configuração de parâmetros mais específicos, através do frontend web, pode ser pouco clara e requer algum conhecimento do modo de funcionamento do Zenoss Core.

A instalação de plugins (ZenPacks) é bastante simples e a verificação da compatibilidade é imediata.

A gestão de utilizadores permite configurar políticas de acesso e permissões às operações e objectos monitorizados pelo Zenoss Core.

As permissões são atribuídas através de *roles* (ZenUser, ZenManager e Manager), de modo a facilitar a configuração de vários níveis de acesso.

A adição de novos objectos pode ser efectuada através de auto-discovery ou manualmente.

Os templates do Zenoss permitem identificar os componentes existentes nos dispositivos, como quantidade de discos e partições ou interfaces de rede e efectuar a respectiva monitorização automaticamente.

A interface web apresenta uma imagem bastante apelativa e permite efectuar grande parte das configurações da aplicação.

O Zenoss possui uma *dashboard* personalizável que permite adicionar vários componentes como eventos activos ou mapa da localização dos servidores

Apesar de ser bastante apelativo, a interface web, possui alguma tarefas complexas, pelo que pode ser necessária alguma formação para a sua utilização.

A configuração de um eventos no Zenoss é baseada numa de três características: valores de threashold, estado de serviços ou logs dos sistemas.

A configuração de cada alerta permite definir facilmente uma das opções pré-configuradas de resposta (execução de comando ou envio de mensagens)

3.3.5 Shinken

O Shinken é uma aplicação de monitorização que apresenta várias características bastante semelhantes ao Nagios.

A instalação do Shinken não é complexa, no entanto, muita da documentação está desactualizada e alguma é herdada do Nagios. Foram sentidas algumas dificuldades ao seguir a documentação para instalação da aplicação, nomeadamente na instalação do módulo Pyro (Python), uma vez que a versão anunciada na documentação não é compatível com a instalação utilizada.

A principal diferença de configuração do Shinken em relação ao Nagios é a necessidade de

configurar os componentes específicos desta aplicação. A sintaxe de configuração é igual à do Nagios, no entanto existe a introdução de novos elementos, derivados da distribuição da arquitectura.

A interface web instalado por omissão é diferente do Nagios, no entanto as suas funcionalidades são bastante semelhantes, destaca-se o facto de suportar os vários componentes do Shinken e permitir a sua gestão para além de permitir a actualização em massa de alguns parâmetros de objectos.

3.3.6 Síntese

O Nagios revela-se uma aplicação bastante modular e personalizável. No entanto o seu modo de configuração revela-se bastante complexo e pode tornar-se repetitivo.

As suas monitorizações são apresentadas sob a forma de estados, tornando os valores de desempenho recolhidos indisponíveis, tornando impossível analisar tendências acerca dos recursos monitorizados.

O seu frontend Web apenas permite visualizar os dispositivos e gerir as várias verificações, no entanto não permite efectuar operações de gestão, nomeadamente adicionar, remover ou editar qualquer objecto.

O Cacti mostra como principal funcionalidade desenhar gráficos de desempenho, no entanto peca por não possui mecanismo de identificação de alertas.

Existe um conjunto de templates que permitem a monitorização dos principais recursos de uma rede, no entanto o desenho de novos templates é complexo e requer conhecimentos da aplicação RRDTool.

O desenho de gráficos demora três ciclos de verificações (15 minutos), o que torna o *debug* de gráficos um pouco demorado.

Como ponto negativo pode-se incluir o facto do Cacti não possui a funcionalidade de auto descoberta como funcionalidade padrão da aplicação, podendo tornar o trabalho do administrador bastante pesado, embora existam *plugins* de terceiros que acrescentem esta funcionalidade.

Ainda assim é uma aplicação que consegue monitorizar grande quantidade de parâmetros de *hardware*, principalmente os que são disponibilizados por SNMP.

O Zabbix apresenta funcionalidades de monitorização de disponibilidade, como o Nagios, e

de desempenho, como o Cacti. A sua operação é optimizada para a utilização de agente, motivo pelo qual o suporte e utilização de outros métodos de monitorização se tornam complexos de configurar.

A interface Web permite efectuar toda a configuração da aplicação. No entanto apresenta demasiada informação, podendo tornar-se pouco clara em relação a alguns itens.

Os alertas podem ser configurados com grande detalhe e suportam execução de várias acções, tais como execução de tarefas ou envio de alertas.

O Zenoss, disponibiliza monitorização de desempenho e de disponibilidade dos recursos. No entanto o método de operação do Zenoss é optimizado para monitorização sem utilização de agentes.

A interface Web permite uma configuração de alto nível de objectos e permite que todos os eventos sejam geridos através de uma *interface* user-friendly. Apesar de ser uma interface apelativa, apresenta alguma lentidão e algumas configurações podem ser complexas, o que torna algumas funções, mais específicas, pouco intuitivas.

Os alertas podem ser configurados com grande granulosidade de modo a criar alertas bastante específicos.

O desenho de templates não é complexo, mas é necessário conhecimento da organização da aplicação e dos protocolos/métodos a utilizar na monitorização. Já as operações de gestão de dispositivos são bastante mais fáceis e intuitivas.

Shinken é uma aplicação baseada no Nagios e possui todas as suas características e compatibilidades. A grande diferença em relação ao Nagios é a divisão da sua arquitectura central em módulos, dividindo-o por tarefas a executar e optimizando o seu funcionamento. Esta optimização traduz-se em menor peso para o sistema, quer em relação à CPU, quer em relação à memória utilizada.

Em contrapartida, grande partes das outras características do Shinken são similares às do Nagios, tais como impossibilidade de gerir objectos a partir da interface web e método de configuração da aplicação.

3.4 Análise Comparativa

Esta secção apresenta as conclusões obtidas após a análise das aplicações no ambiente de testes na Linkcom.

A Tabela 3.1 demonstra o suporte de funcionalidades por parte das aplicações analisadas.

Funcionalidades	Nagios	Cacti	Zabbix	Zenoss	Shinken
Auto-Discovery	✗*	✗*	✓	✓	✗*
Agente	✓	✓	✓	✗	✓
SNMP	✗*	✓	✓	✓	✗*
Syslog	✗*	✗	✓	✓	✗*
Scripts Externos	✓	✓	✓	✓	✓
Plugins	✓	✓	✗	✓	✓
Alertas	✓	✗	✓	✓	✓
Monitorização Distribuída	✓	✓	✓	✓	✓
Gráficos	✗*	✓	✓	✓	✗*
Mapas	✓	✗*	✓	✓	✓
Eventos	✓	✗*	✓	✓	✓

*Funcionalidade suportada através de plugin.

Tabela 3.1: Comparativo das funcionalidades disponibilizadas pelas aplicações.

Com base nesta análise pode-se verificar que grande parte das aplicações possuem as mesmas funcionalidades instaladas por omissão. Os pontos onde existem maiores diferenças é no modo de configuração de cada aplicação e no método de recolha de dados, que pode ser do tipo *agentfull* ou *agentless*.

Após esta análise, tornou-se claro que apenas o suporte de funcionalidades não é suficiente para tomar a decisão de escolha de uma aplicação, uma vez que muitas das funcionalidades podem ser adicionadas através de *plugins*, como por exemplo *auto-discovery* de equipamentos ou o desenho de gráficos.

No entanto, apesar de muitas das funcionalidades serem comuns a várias aplicações, a sua configuração pode ser bastante diferente. Com base nestes critérios foi elaborada a Tabela 3.2, que demonstra os métodos de configuração nas diferentes aplicações.

Funcionalidades	Nagios	Cacti	Zabbix	Zenoss	Shinken
Dispositivos e Serviços	Edição de ficheiro	Frontend Web	Frontend Web	Frontend Web	Edição de ficheiro
Alertas	Edição de ficheiro	N/A	Frontend Web	Frontend Web	Edição de ficheiro
Mapas	Automático	N/A	Manual	Automático	Automático
Gráficos	N/A	Frontend Web	Frontend Web	Frontend Web	N/A
Utilização de Modelos (Templates)	Edição de ficheiro	Frontend Web	Frontend Web	Frontend Web	Edição de ficheiro
Gestão de Plugins	Sistema de ficheiros	Frontend Web	N/A	Frontend Web	Sistema de ficheiros

Tabela 3.2: Análise comparativa do modo de configuração das aplicações.

Como se pode observar, muitas das configurações do Nagios e do Shinken são baseados em edição manual de ficheiros de configuração. Esta abordagem leva a um eventual aumento de erros de configuração e requer conhecimento da sintaxe de configuração das aplicações ou consulta constante à documentação. A organização hierárquica dos ficheiros no Nagios e Shinken são definidos manualmente, pelo que é essencial a existência de documentação descritiva da organização e hierarquia de ficheiros de configuração para referência futura.

A utilização do frontend web como principal meio de interacção por parte das restantes aplicações torna a configuração mais fácil e intuitiva, pelo que deixa de ser necessário uma consulta frequente à documentação.

A configuração de mapas da rede (esquemas de rede) é efectuada, na sua maioria manualmente, como é o caso do Nagios e Shinken, em que o mapa de rede é definido de acordo com as dependências configuradas. Já a abordagem tomada pelo Zenoss permite que o mapa seja criado automaticamente à medida que se vão adicionando novos dispositivos.

A única aplicação analisada que não possui um método automático de criação de mapas de rede é o Zabbix. Neste caso é necessário adicionar todos os componente manualmente. Esta tarefa pode tornar-se bastante repetitiva caso seja necessário representar um grande número de dispositivos no mapa.

À semelhança das configurações, a criação de templates do Nagios e do Shinken requerem que se conheça a sintaxe utilizada nos seus ficheiros de configuração. Já a criação de

templates do Cacti requer alguns conhecimentos das funções utilizadas pelo RRDTool. No caso do Zabbix, a criação de templates requer vários passos e, consequentemente, um uso excessivo do rato. Relativamente à criação de templates no Zenoss, esta possui poucos passos, no entanto é necessário conhecer a organização de templates na aplicação. Por esse motivo é um pouco mais complexa e necessita de algum conhecimento mais aprofundado da aplicação por parte do utilizador. Tanto no Zabbix como no Zenoss, a definição de templates encontra-se simplificada pelo *frontend* Web.

À excepção do Nagios e Shinken, em que é necessário configurar ficheiros de texto, a utilização e instalação de *plugins* é efectuada através do *frontend* Web, tornando imediata a sua integração nas aplicações.

Enquanto no Zabbix e no Zenoss o suporte para a instalação de *plugins* se encontra disponível por omissão, no caso do Cacti é necessária a instalação manual do *add-on* Plugin Architecture (PIA), de modo a suportar a instalação de *plugins* de terceiros.

Após a análise das aplicações, decidiu-se não utilizar nem o Nagios, nem o Shinken devido ao facto da sua configuração ser bastante complexa, repetitiva e propensa a erros, levando a que a gestão total da aplicação recaia sobre o administrador.

Quanto ao Cacti, foi excluído devido à falta de suporte para detecção de eventos e envio de alertas. Apesar desta funcionalidade estar disponível através de um *plugin*, a sua configuração é complexa e apenas permite o envio de alertas por e-mail.

Com base na análise efectuada às aplicações e pelos motivos indicados, chegou-se à conclusão que as aplicações que melhor se adequam aos requisitos definidos para a rede da Linkcom e que oferecem maior facilidade de utilização, configuração e instalação são o Zabbix e o Zenoss.

Numa primeira fase foram seleccionadas as aplicações Zabbix e Zenoss pois, baseado nos testes efectuados, seriam as que melhor se adequavam para a utilização num ambiente empresarial. No entanto, numa rede de testes, torna-se impossível definir qual seria o seu comportamento e quais as tarefas de manutenção numa rede com maior quantidade e diversidade de dispositivos.

Numa segunda fase, em que foi utilizada uma percentagem da rede da Linkcom, com maior quantidade de dispositivos, tornou-se claro que o Zenoss possuía uma larga vantagem na facilidade de utilização em relação ao Zabbix.

As vantagens na automatização das tarefas de monitorização demonstradas pelo Zenoss, tais como a identificação automática dos componentes (discos, partições, CPU, interfaces

de rede) e respectiva monitorização, bem como a detecção automática de alterações de componentes tornaram-se um factor de peso para a selecção desta aplicação. Como o Zabbix não possui estas funcionalidades é o administrador que tem que identificar quais os componentes a monitorizar, identificar os métodos de recolha de dados para cada componente e aplicar o template mais adequado, tornando-se desvantajoso em relação ao Zenoss.

Capítulo 4

Modelo Proposto

Neste capítulo é descrito o modelo de monitorização e alarmística que se encontra em utilização na Linkcom. São descritos os requisitos necessários para a nova solução de monitorização e alarmística e os métodos utilizados para recolha de dados de cada tipo de dispositivo, incluindo o protocolo utilizado para a monitorização e especificação dos respectivos parâmetros. Por fim será apresentada a arquitectura utilizada na nova solução de monitorização e alarmística.

4.1 Estado Actual da Monitorização e Alarmística

O cenário que se encontra actualmente em utilização na Linkcom consiste em dois sistemas distintos para as funções de monitorização e funções de alarmística.

O sistema de monitorização recolhe os valores de desempenho dos serviços e dispositivos monitorizados e apresenta-os sob a forma de gráficos. A aplicação utilizada para este fim é o Cacti.

O sistema de alarmística identifica os estados dos serviços e dispositivos monitorizados. Este sistema apenas identifica estados dos dispositivos e não recolhe valores referentes ao seu desempenho. Esta função é executada pelo Nagios.

A Figura 4.1 representa o esquema utilizado para a monitorização da rede actualmente em utilização na Linkcom. Como se pode observar as tarefas de monitorização são efectuadas através das aplicações Nagios e Cacti. Os dispositivos monitorizados por esta aplicações incluem switches, routers e servidores. Os dispositivos encontram-se localizados no *datacenter* local ou em localizações remotas, as quais são acedidas através da Internet. De

modo a garantir que a confidencialidade dos dados trocados com as localizações remotas as ligações são efectuadas através de VPN ou túnel SSH.

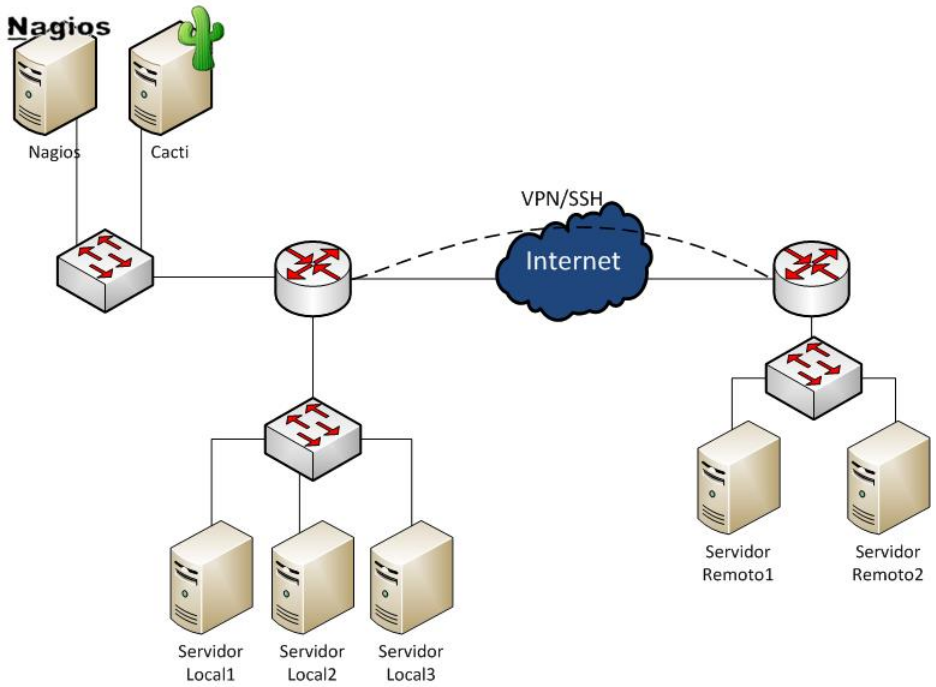

Figura 4.1: Esquema da solução actual de monitorização.

Ambas as aplicações em funcionamento na Linkcom desempenham as suas funções correctamente, no entanto encontram-se em plataformas separadas. Esta característica implica que a visualização da informação esteja dividida pelas aplicações, o que torna a sua análise complicada pois não existe uma plataforma que apresente a informação referente a ambas as aplicação de forma combinada e coerente.

A nova solução pretende integrar as funções das soluções já existentes em produção,nomeadamente monitorização de estados e monitorização de desempenho, numa única plataforma integradora.

4.2 Requisitos da Solução

A solução que se pretende apresentar deverá incluir a aplicação que melhor se adequa à Linkcom de modo a realizar, autonomamente, tarefas de recolha e processamento de dados relativos ao funcionamento dos equipamentos que compõem a rede.

A solução a implementar necessita de suportar as seguintes funcionalidades:

- **monitorização de hardware**- desempenho de CPU, utilização de discos ou utilização de memória;
- **monitorização de software**- verificação de serviços e desempenho de aplicações, como Microsoft Exchange ou servidores de base de dados;
- **detecção de eventos**- identificação de comportamentos anormais nos parâmetros monitorizados;
- **envio de alertas**- envio de anomalias através de SMS ou email;
- **criação relatórios**- histórico dos valores recolhidos, de alertas e eventos e lista de maquinas da rede;
- **suporte de perfis de utilizadores**- atribuição de dispositivos e definição de horário de disponibilidade;
- **frontend web**- iteração compatível com browser.

Um factor preferencial, mas não obrigatório, consiste no facto da solução suportar a monitorização sem a utilização de agentes. Como alternativa devem ser utilizados protocolos de monitorização standard, como o SNMP, e/ou outros meios de recolha de dados, como WMI ou execução de comandos através de sessões SSH. A utilização de agentes apenas deverá ser considerada em último recurso, uma vez que implica a instalação de um aplicativo em cada dispositivo a monitorizar. Esta abordagem torna-se uma grande desvantagem em redes de grande dimensão, pois é necessário despender uma grande quantidade de tempo e recursos na instalação de agentes em toda a infraestrutura de rede, a mesma situação também é verificada caso seja necessário actualizar os agentes anteriormente instalados.

4.3 Arquitectura de Monitorização Proposta

A figura 4.2 apresenta a arquitectura de monitorização proposta para esta solução de monitorização e alarmística. A arquitectura de monitorização foi dividida em 3 camadas, de acordo com a sua funcionalidade.

A camada 1 - Cliente, é composta pelos dispositivos monitorizados, os quais disponibilizam um conjunto de informação mediante o pedido do servidor de monitorização. Regra geral, os dispositivos monitorizados apenas disponibilizam informação a pedido pelo servidor de monitorização. No entanto existem dispositivos que podem enviar mensagens assíncronas para o servidor sob a forma de *traps* SNMP.

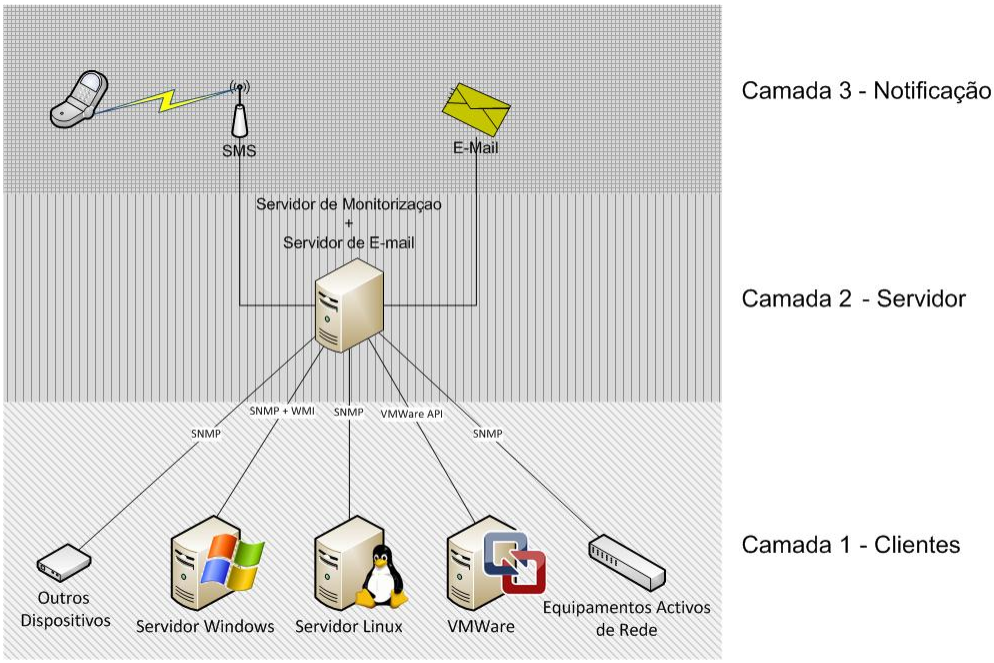

Figura 4.2: Modelo proposto.

Os dados são recolhidos através de vários métodos que incluem a utilização do protocolo SNMP, WMI ou VMWareAPI.

A camada 2 - Servidor é composta pelo servidor de monitorização. É a camada responsável pelo tratamento dos dados recolhidos dos dispositivos monitorizados. Posteriormente permite a comparação dos valores recolhidos com valores de referência e identificação de eventos que podem dar origem a alertas.

Esta camada também é responsável pela apresentação dos dados ao utilizador através de um frontend web. Todos os dados recolhidos são armazenados nesta camada de modo a permitir efectuar uma análise do desempenho ao longo do tempo.

A camada 3 - Notificação representa o modo de comunicação de alarmes com o utilizador. Os métodos utilizados para a comunicação são o envio de SMS e de e-mail. Ambos os métodos de comunicação encontram-se configurados no servidor de monitorização de modo a evitar a dependência de dispositivos que se possam encontrar indisponíveis.

4.4 Métodos de Recolha de Dados

Esta secção descreve os métodos e protocolos utilizados para a recolha de dados de cada um dos parâmetros a monitorizar pela solução de monitorização e alarmística.

Como foi referido, a recolha de dados é efectuada de diferentes modos de acordo com sistema operativo e natureza do dispositivo. Tal como descrito nos requisitos da aplicação de monitorização, estes métodos de recolha de dados foram seleccionados por serem aqueles que permitem recolher todos os parâmetros necessários sem a necessidade de instalação de agentes da aplicação. Os templates definidos para cada tipo de dispositivo terão em conta os métodos de recolha de dados definidos nesta secção.

4.4.1 Monitorização de Estados

A monitorização do estado de dispositivos é efectuada, maioritariamente, pelo protocolo ICMP, verificando se existe resposta às mensagens de `ping`. Caso exista resposta às mensagens enviadas é considerado que o dispositivo se encontra activo. Em dispositivos configurados para ignorar mensagens ICMP torna-se necessária a utilização de outros protocolos como SSH, WMI, SNMP ou verificar se os serviços em execução no dispositivo respondem a pedidos, como é o caso de telnet ou http.

A monitorização dos estados dos serviços que se encontrem em funcionamento nos dispositivos monitorizados é efectuado através da verificação de portos definidos para os serviços mais comuns, como http (porto 80) ou dns (porto 53).

4.4.2 Monitorização de Desempenho

A monitorização de desempenho de *hardware* recolhe dados referentes à utilização de recursos, como a utilização de CPU ou da memória RAM. Cada tipo de dispositivo possui diferentes componentes de *hardware* a monitorizar. No entanto existem componentes comuns a todos os dispositivos que utilizem as mesmas MIBs como a MIB-II, para redes TCP/IP ou Host Resource MIB [39], para monitorização de recursos do dispositivo. Estas MIBs permitem recolher informação comum a vários dispositivos, como é o caso da utilização de CPU, utilização da memória RAM, utilização do disco e informações sobre as *interfaces* de rede.

Deste modo, os dados, recolhidos por SNMP, relativos ao desempenho de hardware comum entre os vários servidores encontram-se representados na tabela 4.1, nomeadamente dados referentes ao armazenamento (Discos e RAM), como Descrição do Armazenamento, Espaço Total e Espaço Utilizado, utilização de CPU (Carga de CPU) e interfaces de rede (Tráfego Recebido, Tráfego Enviado, Erros de Recepção e Erros de Envio).

Parâmetro	**OID**	**Unidade**
Descrição do Armazenamento	.1.3.6.1.2.1.25.2.3.1.3	N/A
Espaço Total	.1.3.6.1.2.1.25.2.3.1.5	Unidades de Alocação
Espaço Utilizado	.1.3.6.1.2.1.25.2.3.1.6	Unidades de Alocação
Unidades de Alocação	.1.3.6.1.2.1.25.2.3.1.4	Bytes
Tipo de Armazenamento	1.3.6.1.2.1.25.2.3.1.2	N/A
Carga de CPU	.1.3.6.1.2.1.25.3.3.1.2	Percentagem (%)
Tráfego Recebido	.1.3.6.1.2.1.2.2.1.10	Bytes
Erros de Recepção	.1.3.6.1.2.1.2.2.1.14	Número de Erros
Tráfego Enviado	.1.3.6.1.2.1.2.2.1.16	Bytes
Erros de Envio	.1.3.6.1.2.1.2.2.1.20	Número de Erros

Tabela 4.1: Parâmetros comuns entre servidores

Cada um dos parâmetros presentes na tabela 4.1 pode devolver um vector, caso o dispositivo possua vários componentes do mesmo tipo, como é o caso dos vários tipo de armazenamento. Esta característica implica que se tenham que identificar quais os índices de cada componente a monitorizar, por exemplo,Memory buffers (6), Cached memory (7) ou partições como /boot (35), demonstrados na Figura 4.3.

Em alguns casos é necessário utilizar de parâmetros auxiliares, como é o caso de Unidades de Alocação, para o cálculo do valor absoluto de alguns parâmetros. Um destes casos é o Espaço Total em Bytes, pois a Unidade de Alocação pode variar consoante o tipo de armazenamento (disco rígido, memória RAM ou leitores ópticos).

Apesar dos parâmetros apresentados fornecerem dados acerca dos principais componentes de *hardware* comuns entre servidores de vários sistemas operativos, como Windows ou Linux, os sistemas operativos Linux fornecem informação mais detalhada proveniente das MIBs UCD [40]. Os parâmetros fornecidos pela MIB UCD são disponibilizados pelo agente `net-snmp`. Os parâmetros mais relevantes para este projecto encontram-se na Tabela 4.2.

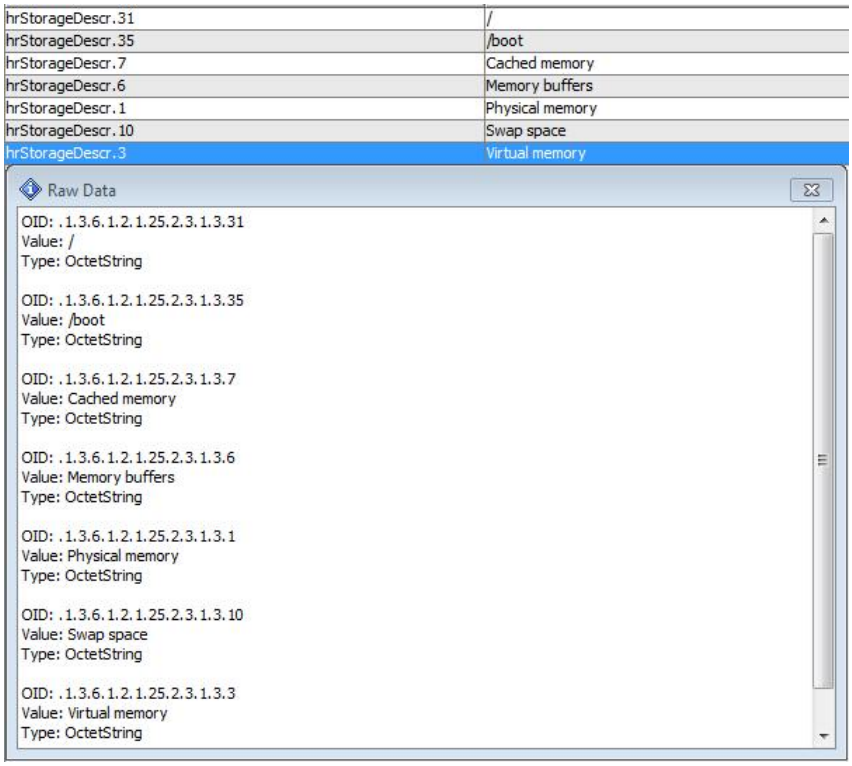

Figura 4.3: Exemplo de descrições de vários tipos de armazenamento (SNMP).

Parâmetro	OID	Unidade
Memória de Buffer	.1.3.6.1.4.1.2021.4.14.0	Bytes
Memória de Cache	.1.3.6.1.4.1.2021.4.15.0	Bytes
Memória Partilhada	.1.3.6.1.4.1.2021.4.13.0	Bytes
Memória Disponível	1.3.6.1.4.1.2021.4.6.0	Bytes
Memória Total	1.3.6.1.4.1.2021.4.5.0	Bytes
Operações de Leitura do Disco	.1.3.6.1.4.1.2021.13.15.1.1.3	Bytes
Operações de Escrita do Disco	.1.3.6.1.4.1.2021.13.15.1.1.5	Bytes
Carga de CPU 1min	.1.3.6.1.4.1.2021.10.1.3.1	Percentagem (%)
Carga de CPU 5min	.1.3.6.1.4.1.2021.10.1.3.2	Percentagem (%)
Carga de CPU 15min	.1.3.6.1.4.1.2021.10.1.3.3	Percentagem (%)
CPU User Time	.1.3.6.1.4.1.2021.11.50.0	Percentagem (%)
CPU Nice Time	.1.3.6.1.4.1.2021.11.51.0	Percentagem (%)
CPU System Time	.1.3.6.1.4.1.2021.11.52.0	Percentagem (%)
CPU Idle Time	.1.3.6.1.4.1.2021.11.53.0	Percentagem (%)

Tabela 4.2: Parâmetros disponibilizados pelo Net-SNMP.

Como se pode observar na tabela 4.2 os dados fornecidos pelas MIBs do Net-SNMP são mais detalhados que os fornecidos pelas MIBs genéricas (MIB-II e Host Resource MIB). No entanto apenas são disponibilizados pelos sistemas Linux que possuem o agente Net-SNMP instalado.

A ocupação da memória é apresentada consoante a sua utilização (*Buffer, Cache* e Partilhada). Para além desta separação, é apresentada a memória total e a memória que ainda se encontra disponível.

Em relação a informações acerca dos discos rígidos, o Net-SNMP permite visualizar a quantidade de informação trocada nas transacções dos discos (Operações de Leitura/Escrita do Disco). Como a informação obtida diz respeito à quantidade de bytes transferidos desde o arranque do dispositivo é necessário calcular a diferença entre duas verificações consecutivas de modo a que seja possível calcular a média de bytes transferidos por segundo (Bps).

No que diz respeito à informação de CPU, é possível obter a média de carga em 3 períodos de tempo (1 minuto, 5 minutos e 15 minutos). Para além da média de carga é possível ainda obter métricas detalhadas referentes aos vários tipos de utilização de processamento, nomeadamente *User, Nice, System e Idle*, que correspondem respectivamente aos processos dos utilizadores, aos processos cuja prioridade foi alterada em execução, processos do sistema e inactividade do processador.

4.4.3 Equipamentos Activos de Rede

Os equipamentos activos de rede, tais como switches ou routers, possuem os parâmetros de monitorização de interfaces comuns, de modo que na tabela 4.3 apenas se encontram os parâmetros específicos de monitorização deste tipo de equipamentos. Uma vez que os equipamentos de rede da empresa são da marca HP, a monitorização dos equipamentos de rede, utiliza a MIB específica deste fabricante.

Parâmetro	OID	Unidade
Carga de CPU	.1.3.6.1.4.1.11.2.14.11.5.1.9.6.1.0	Percentagem(%)
Memória Disponível	.1.3.6.1.4.1.11.2.14.11.5.1.1.2.1.1.1.6.1	Bytes
Memória Total	.1.3.6.1.4.1.11.2.14.11.5.1.1.2.1.1.1.5	Bytes
Tráfego Input	.1.3.6.1.2.1.31.1.1.1.6	Octetos
Tráfego Output	.1.3.6.1.2.1.31.1.1.1.10	Octetos

Tabela 4.3: Parâmetros SNMP de equipamentos de rede.

Os parâmetros apresentados nesta categoria são homónimos dos parâmetros recolhidos nos servidores, no entanto os identificadores (OID) utilizados são diferentes.

O tráfego de input e de output analisa a largura de banda utilizada por cada interface de rede do dispositivo. Normalmente estes parâmetros são compostos por um vector com todas as interfaces de rede do dispositivo, no qual cada índice do vector representa uma interface do dispositivo. A unidade destes parâmetros encontra-se em Octetos (Bytes), de modo que para apresentar a utilização em bits por segundo (bps) é necessário multiplicar o valor obtido por 8.

4.4.4 Dispositivos Windows

Como os parâmetros disponibilizados pelo agente SNMP dos dispositivos Windows não fornece todos os dados necessários para cumprir os requisitos definidos, tal como operações de I/O dos discos rígidos. Foi necessária uma nova abordagem para monitorizar este tipos de dispositivos.

Por esse motivo, foi utilizado o WMI para recolha de dados de monitorização dos dispositivos Windows.

A tabela 4.4 apresenta as classes utilizada para a recolha dos parâmetros que não são disponibilizados através de SNMP.

Parâmetro	Classe WMI
Discos Lógicos	Win32_PerfRawData_PerfDisk_LogicalDisk
Discos Físicos	Win32_PerfRawData_PerfDisk_PhysicalDisk
Memória	Win32_PerfRawData_PerfOS_Memory
Processador	Win32_Processor

Tabela 4.4: Parâmetros recolhidos por WMI.

Cada uma das classes apresentadas recolhe um conjunto de características referentes ao parâmetro apresentado. A classe referente aos discos lógicos permite obter dados relacionados com a sua utilização e operações de I/O dos sistemas de ficheiros. A classe referente aos discos físicos apresenta informação referente às operações de I/O. A classe relacionada com a memória disponibiliza informação sobre a quantidade de memória utilizada. Por fim, a classe referente ao processador disponibiliza informação acerca da utilização de CPU.

Um exemplo de uma consulta utilizado o WMI pode ser observada na Figura 4.4. Neste caso é efectuada uma consulta às operações de leitura (`DiskReadsPerSec`) e escrita (`DiskWritesPerSec`) de um disco físico.

```
SELECT DiskReadsPerSec,DiskWritesPerSec
FROM Win32_PerfRawData_PerfDisk_PhysicalDisk
WHERE Name="_Total"
```

Figura 4.4: Exemplo de consulta efectuada por WMI.

4.4.5 Dispositivos VMware

A principal solução de virtualização utilizada na Linkcom é baseada em aplicações VMWare. Apesar de ser possível recolher informação acerca destes dispositivos utilizando SNMP, as aplicações VMware permitem a utilização da API do produto para obter informações relevantes para a monitorização.

A API do VMware permite obter informações acerca do funcionamento do sistema físico bem como dos sistemas virtualizados, fornecendo informações de utilização de CPU, memória e transacções do disco.

É possível obter informação detalhada acerca das máquinas virtualizadas e respectiva utilização de recursos do sistemas e do servidor VMWare, como utilização de CPU ou de

Memória através da API do VMWare. Por este motivo decidiu-se utilizar a API para recolha de dados deste tipo de dispositivos.

4.4.6 Outros Dispositivos

Existem outros dispositivos que, por desempenharam funções únicas, não se encontram em nenhuma das categorias descritas.

Parte-se do principio que um dispositivo pertencente a esta classe possuam um agente SNMP que permita a sua monitorização, sendo necessário definir quais os parâmetros a recolher.

4.5 Envio de alarmes

Os métodos de alarmes a utilizar na nova solução de monitorização divide-se em dois meios de contacto: e-mail e SMS.

Cada um dos meios de contacto é configurado de modo que cada utilizador apenas seja alertado de acordo com as suas definições pessoais, como o horário de disponibilidade ou dispositivos pelos quais è responsável.

O envio de e-mail e SMS encontra-se dependente do próprio servidor de monitorização, garantindo que estes serviços se encontram sempre disponíveis, independentemente do estado da restante infra-estrutura.

Para o envio de e-mail será utilizado o servidor de email do *Zope*. O envio de SMS é efectuado através da aplicação gnokii [41] e está dependente de um módulo Global System for Mobile (GSM) directamente ligado ao servidor de monitorização.

4.6 Sumário

Neste capítulo foram apresentados os métodos de recolha de dados e a arquitectura utilizada na solução de monitorização e alarmística a implementar na Linkcom.

A recolha de dados é efectuada através de dois principais métodos, a utilização do protocolo SNMP e a utilização do WMI. Enquanto que o protocolo SNMP é utilizado para a maioria dos dispositivos, independentemente do seu género, os dispositivos Windows são monitorizados através da combinação do SNMP e do WMI e uma minoria de dispositivos,

como é o caso dos servidores VMWare, utiliza APIs próprias que permitem recolher dados relevantes para a monitorização deste tipo de dispositivos como, por exemplo, as instâncias de maquinas virtuais e respectivo estado ou utilização de CPU, memória e disco.

A arquitectura de monitorização encontra-se dividida em 3 camadas, cada uma representando uma função a desempenhar pela solução de monitorização e alarmística. Assim, a camada 1 desempenha tarefas de recolha de dados dos dispositivos monitorizados, a camada 2 trata e apresenta os dados recolhidos ao utilizador, bem como identifica eventos com base nos dados recolhidos, enquanto a camada 3 é responsável pela comunicação de eventos através de SMS e e-mail.

Capítulo 5

Caso de Estudo - Linkcom

Este capítulo descreve as decisões tomadas na definição do plano de migração para a nova solução de monitorização descrita no Capitulo 4, bem como a implementação do Zenoss para a monitorização em ambiente real.

5.1 Metodologia de monitorização

De modo a efectuar a migração para a nova solução foi definida a seguinte metodologia:

1. Identificar os dispositivos monitorizados pela solução de monitorização em produção;
2. Identificar os grupos lógicos definidos da solução de monitorização em produção;
3. Recolher contas de utilizadores e respectivos horários de disponibilidade da solução em ambiente de produção;
4. Adaptar configurações de utilizadores à nova solução de monitorização;
5. Recolher grupos de utilizadores e de dispositivos da solução em produção e adaptar à nova solução;
6. Adicionar, à nova solução, os dispositivos a monitorizar;
7. Configurar parâmetros específicos de certos dispositivos, como credenciais de acesso;
8. Iniciar a entrada em produção da nova solução de monitorização e alarmística;
9. Desactivar a solução antiga monitorização.

5.1.1 Migração de Dispositivos e Grupos

A recolha dos dispositivos a monitorizar passa pela identificação dos dispositivos monitorizados pelo Nagios e Cacti e respectivas características, como tipo de dispositivo (switch, router, servidor, etc), sistema operativo e período durante o qual são enviados alertas.

Uma vez que os dispositivos monitorizados se encontravam organizados por grupos lógicos, foi necessária a recolha destes grupos e dispositivos que os compõem, bem como os utilizadores responsáveis por cada um dos grupos de dispositivos.

Dispositivos

A migração de dispositivos é uma tarefa simples, pois apenas é necessário adicionar os dispositivos à nova solução de monitorização e aplicar o template que melhor se adequa a cada caso. A monitorização dos vários componentes dos dispositivos não necessita de configurações adicionais, salvo algumas excepções em que é necessária a monitorização de um componente especifico que não conste no template de monitorização ou alguma configuração referente às credenciais de acesso. Esta característica permite que uma grande quantidade de dispositivos seja adicionada e as configurações base sejam automaticamente aplicadas de acordo com o template utilizado.

Os templates da nova solução de monitorização têm como denominação "Classe". Cada Classe define os modos de monitorização, tais como SNMP ou WMI, e os parâmetros dos dispositivos que são monitorizados. As classes encontram-se numa organização hierárquica, de sistemas, de modo a manter uma estrutura coerente, apresentada na Figura 5.1.

Existem algumas classes definidas pela aplicação, nomeadamente "Ping"e "Discovered", que efectuam a monitorização de mais básica. A classe "Ping"apenas efectua monitorização do estado do dispositivo (ligado/desligado) através de ICMP. A classe "Discovery"funciona como repositório dos dispositivos adicionados pela funcionalidade *auto-discovery* da aplicação. Cabe ao utilizado mover os dispositivos para o local apropriado após a descoberta.

Neste caso concreto, foi criado um script em *python* que permite automatizar a tarefa de mover os dispositivos da classe "Dyscovery"para a classe mais apropriada (Anexo D).

Grupos de Dispositivos

A migração de grupos de dispositivos foi efectuada através da replicação da organização presente na actual solução de monitorização para a nova solução. No entanto, de modo a facilitar a atribuição de horários de notificações, nomeadamente o laboral e o de prevenção, na nova solução de monitorização, a migração de grupos de dispositivos irá sofrer algumas

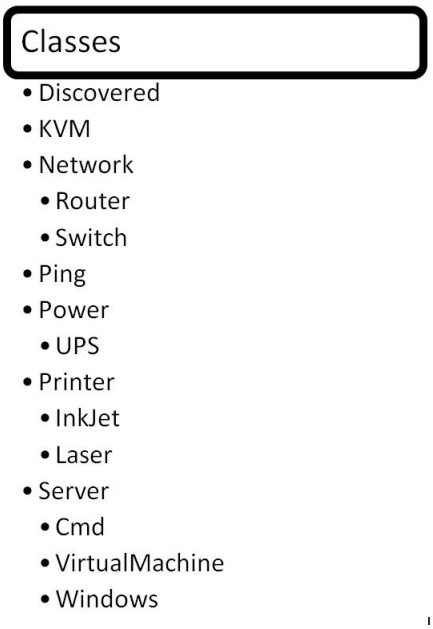

Figura 5.1: Organização dos dispositivos por Classes.

alterações na estrutura.

Os grupos que constam da solução actual dividem os dispositivos em grupos consoante a sua função. Alguns dos grupos que se encontram definidos na solução de monitorização são, por exemplo, APs, Rede Corporate (Equipamentos de Rede) ou Rede Corporate (Host), como demonstra a Figura 5.2.

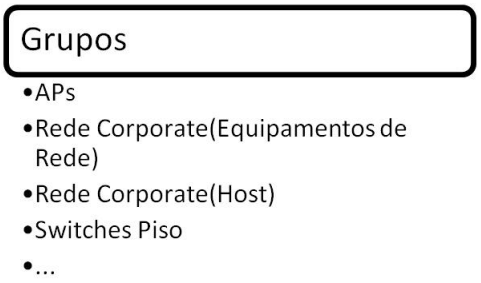

Figura 5.2: Grupos de dispositivos da solução actual.

Na nova solução são mantidos os nomes dos grupos, mas é definida uma nova hierarquia. Dessa forma foram definidos os grupos "Prevenção"e "Laboral", que efectuam as notificações de alertas 24 horas e 8 horas por dia respectivamente.

Foram replicados os grupos para a nova solução ficando, no entanto divididos de acordo com o período em que são enviadas notificações referentes aos dispositivos.

As alterações tornam-se possíveis devido ao suporte de subclasses por parte do Zenoss, que

permite organizar os grupos hierarquicamente. Assim, a definição de grupos no Zenoss mantém a estrutura básica dos grupos actuais, mas divide os grupos em dois de acordo com o período de notificações.

Assim os dois grupos base ("Prevencao"e "Workhours") são constituídos pelos grupos que já se encontram definidos. A Figura 5.3 demonstra a hierarquia de grupos utilizada na nova solução de monitorização.

Figura 5.3: Migração de Grupos: Prevenção.

Os dois grupos base possuem os mesmos subgrupos. No entanto os dispositivos presentes em cada um são diferentes. Esta abordagem facilita a atribuição de responsáveis a grupos de dispositivos, pois os principais tipos de utilizadores dividem-se em utilizadores de prevenção e os restantes. Neste caso os utilizadores de prevenção devem receber notificações a qualquer altura do dia, enquanto os restantes utilizadores apenas devem receber alertas durante o horário de expediente.

5.1.2 Migração de Utilizadores

A migração de utilizadores tem como objectivo manter a organização de utilizadores o mais semelhante possível à solução actual. A principal dificuldade encontrada é o facto do Zenoss possuir a limitação de apenas um contacto de cada tipo (SMS e email) por utilizador. Esta limitação não existe no Nagios.

Utilizadores

Após análise dos ficheiros de configuração de utilizadores do Nagios, verificou-se que vários utilizadores tinham o mesmo contacto de email, como se pode observar na configuração do Nagios apresentada na Figura 5.4, variando apenas o horário de notificação.

```
define contact {
contact_name gs_mail
(...)
Email foo@linkcom.pt
}

define contact {
contact_name gs_mail_8x5
(...)
Email foo@linkcom.pt
}

define contact {
contact_name prevenção
(...)
Email foo@linkcom.pt
}
```

Figura 5.4: Migração: Duplicação de Emails

A nova abordagem pretende remover a duplicação de contactos dos utilizadores através da associação de contactos a utilizadores e de períodos de notificações a grupos de utilizadores. Deste modo torna-se possível que um utilizador e respectivo contacto sejam atribuídos a vários grupos de utilizadores.

Grupos de utilizadores

A maioria dos utilizadores do Nagios foram alterados na migração para o Zenoss. Este facto deve-se, principalmente á duplicação de contactos existente nos utilizadores do Nagios, causado pela duplicação de contactos devido ao facto do horário de notificação ser diferente. Assim, os utilizadores da solução actual cujas únicas diferenças entre si são o método de contacto (email e SMS) ou o período de envio de notificações (24x7 ou 8x7) serão convertidos num único utilizador. Como os períodos de notificação foram transferidos para os grupos de utilizadores, não é necessário aplicar a duplicação de utilizadores para

diferentes períodos de notificação (Figura 5.5).

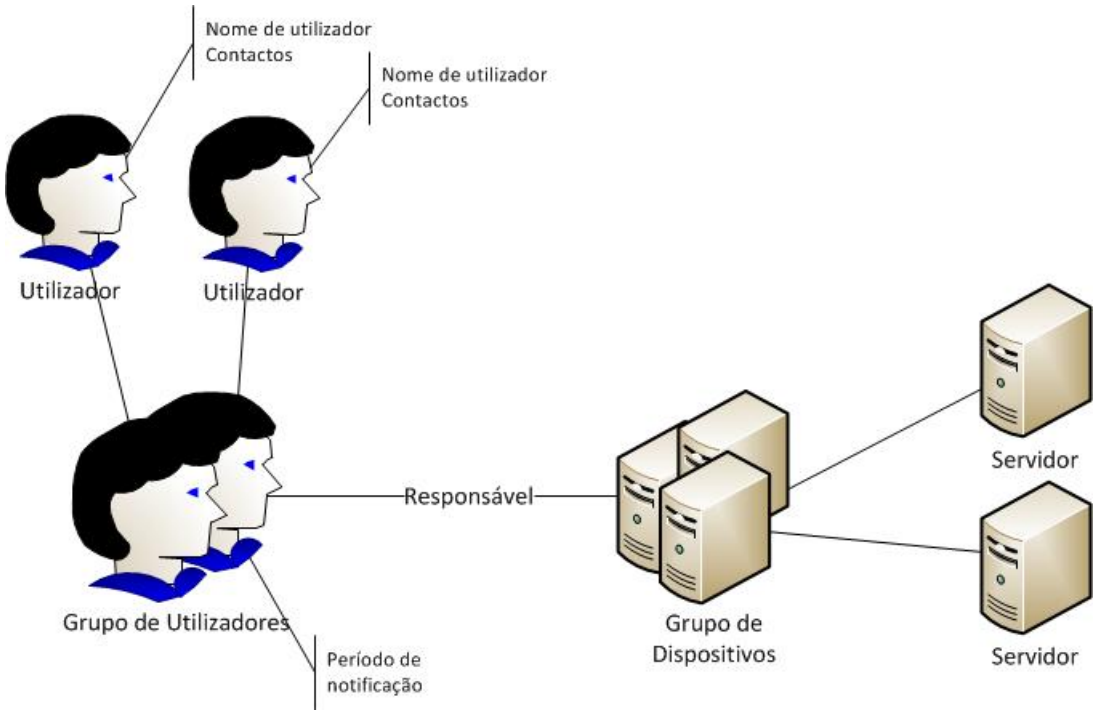

Figura 5.5: Organização de Utilizadores.

Os utilizadores são compostos pelo nome de utilizador e respectivos contactos. Os grupos de utilizadores apenas possuem informação do horário de notificação para alertas.

O modo utilizado para atribuição de utilizadores responsáveis por dispositivos implica que os utilizadores pertençam, no mínimo, um grupo de modo a poder definir o período de notificação. Por sua vez, os grupos de utilizadores são responsáveis por um ou vários dispositivos. Deste modo, em conjunto com os grupos de dispositivos definidos acima, é possível gerir facilmente utilizadores de acordo com o seu perfil de notificação (Prevenção ou Workhours). Caso seja necessário remover ou adicionar um utilizador a um conjunto de dispositivos basta associar o utilizador ao grupo responsável por esses dispositivos.

Em casos onde existem vários contactos de email por utilizador é necessário criar um novo utilizador com esse contacto, de modo a manter a configuração da nova solução de monitorização coerente com as definições da antiga.

5.2 Tarefas de migração

Esta secção descreve as tarefas de migração efectuadas na transição entre as soluções de monitorização.

5.2.1 Configuração da Aplicação

Antes de iniciar a migração dos dispositivos, utilizadores e respectivos grupos foi necessário preparar a aplicação, através da configuração dos seus vários componentes, de modo a garantir a correcta monitorização dos dispositivos serviços de rede, assim como, assegurar que os eventos sejam anunciados de forma correcta através dos meios de comunicação definidos (email e SMS).

Assim a configuração da aplicação foi iniciada através da definição dos parâmetros personalizados, necessários para completar a documentação dos dispositivos, nomeadamente Empresa, Funcionalidade, presença de UPS e opções de tipo de servidor (Fisico Core e Central, Fisico de Produção, VM Core e Central ou VM de produção).

Os grupos de dispositivos criados encontram-se organizados pela hierarquia representada na Figura 5.6.

Figura 5.6: Hierarquia de grupos de dispositivos.

A duplicação de subgrupos foi criada de modo a manter a coerência de grupos entre os dois grupos base (Workhours e Prevencao). Apesar de parecer uma divisão complexa, a gestão de dispositivos, tendo em conta o período de envio de alertas, torna-se mais fácil.

A aplicação foi instalada num servidor dedicado com um processador Intel Xeon E5410 a 2.33GHz com 16GB de memória RAM e disco rígido de 300 GB. Todas as especificações cumprem os requisitos mínimos recomendados para um conjunto entre 500 e 1000 dispositivos. Como a quantidade de dispositivos a monitorizar, no caso da Linkcom, são cerca de

200, não existe qualquer limitação por parte deste servidor a nível de *hardware*.

De modo a responder aos vários requisitos de monitorização dos diferentes dispositivos e acrescentar alguma funcionalidades foram instalados plugins (ZenPacks) que aumentam as funcionalidades e gama de dispositivos suportados pela solução, nomeadamente:

- Active Directory
- Microsoft Exchange
- Microsoft SQL Server (MSSQL)
- SNMP Performance Monitor
- Hyper-V monitoring
- VMWare ESXi Monitor

Os ZenPacks instalados permitem a monitorização mais detalhada de determinadas aplicações, nomeadamente Active Directory, Microsoft Exchange e Microsoft SQL Server, monitorização de um maior número de parâmetros de desempenho por SNMP, como ocupação dos sistemas de ficheiros, ocupação de memória ou utilização de CPU, e monitorização de maquinas virtuais de sistemas Microsoft (Hyper-V) e VMWare (VMWare ESXi).

Para o envio de alertas para os utilizadores foi necessário garantir que os serviços de email e SMS funcionassem correctamente. O serviço de email é suportado pela framework aplicacional Zope, que serve de base à aplicação Zenoss, pelo que é instalado juntamente com a aplicação de monitorização.

O serviço SMS não é suportado directamente pelo Zenoss, pelo que foi necessário utilizar o aplicação Gnokii para esse fim. Apesar de ser uma aplicação externa é possível utilizar a interface de comunicação com pager, incluída no Zenoss, como meio de interacção com o Gnokii. Assim foi desenvolvido um script que utiliza o Gnokii para envio de SMS's através dos parâmetros utilizados para envio de mensagens de pager pelo Zenoss. O script para envio de SMS encontra-se no Anexo A.

5.2.2 Migração de Dispositivos e Grupos

O levantamento dos dispositivos existentes na solução anterior permitiu identificar os diversos tipos de dispositivos (Tabela 5.1), grupos a que pertencem (Figura 5.7) e período de resposta a alertas, nomeadamente prevenção e laboral (Tabela 5.2).

	Quantidade
Servidores Linux	21
Servidores Windows	108
Switches	32
Routers	3
AP	10
Firewall	6
Servidores VMWare	11
Storage	3
Sem categoria	5
Total de dispositivos	199

Tabela 5.1: Quantidade de dispositivos monitorizados.

Os dispositivos com sistema Windows encontram-se em larga maioria na rede da Linkcom pelo que o WMI será o método mais utilizado para recolha de dados dos dispositivos. A maioria dos restante equipamentos são monitorizados por SNMP à excepção dos servidores VMWare e dispositivos sem categoria. Os servidores VMWare são monitorizados através da API do produto e os dispositivos sem categoria são monitorizados apenas por ICMP. Neste caso apenas é possível monitorizar a disponibilidade dos dispositivos.

Os grupos lógicos de dispositivos mantêm a sua estrutura na nova solução, sofrendo apenas pequenas alterações, nomeadamente subgrupos, como exemplificado no grupo Rede Corporate (figura 5.7).

Figura 5.7: Exemplo de subgrupos de dispositivos.

A existência de dois perfis de envio de alertas implicam a necessidade de configurar o período de alertas em cada um dos dispositivos. A nova abordagem aplica o período de alertas aos grupos dos dispositivos. Esta característica permite configurar facilmente o período de envio de alertas de dispositivos consoante o grupo a que foram adicionados.

	Dispositivos
Prevenção	41
Laboral (Workhours)	158

Tabela 5.2: Grupos de notificação.

A migração de dispositivos recorreu à utilização do componente `zenbatchload`. Este componente do Zenoss e permite adicionar dispositivos através de um ficheiro com formato pré-definido. O `zenbatchload` permite, assim, efectuar um conjunto de configurações relativas à organização ou aos parâmetros dos dispositivos, o que se torna uma mais valia quando é necessário adicionar uma grande quantidade de dispositivos com configurações diferentes. O modo de utilização do `zenbatchload` encontra-se descrito e exemplificado no Anexo B.

5.3 Plano de contingência

Antes da entrada em ambiente de produção da nova solução de monitorização e desactivação da solução actual, as duas soluções encontraram-se a funcionar em simultâneo. Achou-se que o período de tempo necessário para a análise do funcionamento seria de 2 semanas. Este período tem como objectivo testar o funcionamento da aplicação em ambiente real, de modo a identificar possíveis falsos positivos, a utilização de recursos de rede, os métodos de comunicação de alertas, nomeadamente envio de email e SMS, bem como efectuar as alterações que forem necessárias às configurações da aplicação.

Este período permite igualmente detectar eventuais problemas que possam surgir com o funcionamento prolongado da solução em ambiente real e, deste modo, ajustar os parâmetros adequados sem interferir no normal funcionamento das tarefas de monitorização e alarmística.

5.3.1 Falsos positivos

A grande maioria dos casos de falsos positivos identificados durante o período de contingência eram originados por falha de respostas aos pedidos de *pooling* durante um ciclo de verificações, dando origem a eventos de alerta. No ciclo seguinte os eventos eram suprimidos pela resposta correcta aos pedidos de *pooling*.

A solução utilizada para mitigar este tipo de alertas passa pela utilização da abordagem de *hard e soft states* do Nagios, no qual um alerta é criado após um número definido de falhas consecutivas. Assim foi aumentado o número de eventos consecutivos necessários para despoletar um alerta para o valor dois. Deste modo é necessário que o mesmo evento possua duas falhas consecutivas nas respostas aos pedidos de *pooling* para criar um alerta. Este valor foi utilizado de modo a evitar que uma falha momentânea na entrega dos pacotes provocasse um evento e consequente alerta.

Para além das falhas de resposta a pedidos de *pooling*, outro motivo de falsos positivos consiste na verificação das entradas de *syslog* e *eventlog* que adicionam um grande numero de eventos que muitas das vezes são informativos. O método utilizado para resolver o número de falsos positivos criados foi definir quais os tipos de *logs* que seriam considerados mais relevantes como, por exemplo, logs do Microsoft Exchange, servidor MSSQL ou servidor MySQL.

5.3.2 Impacto do tráfego gerado pela aplicação de monitorização

A nova solução de monitorização permite monitorizar um maior número de parâmetros por dispositivo, motivo pelo qual será esperado um aumento do tráfego efectuado pela nova solução de monitorização e alarmística. O impacto referente à utilização dos recursos da rede pela nova solução de monitorização pode ser verificado através da o tráfego efectuado pelas interfaces de rede utilizadas por cada uma das soluções de monitorização. A análise do tráfego efectuado por cada uma das aplicações encontra-se na figura 5.8. Neste caso a nova solução é representada pelo dispositivo (*device*) `localhost` e a solução actual pelo dispositivo `noc.aitec.pt`.

A análise ao tráfego efectuado pelas duas aplicações demonstra um ligeiro aumento de utilização de rede por parte da nova solução de monitorização. No entanto, não é o suficiente para provocar um congestionamento na rede, uma vez que a taxa de transferência (*throughput*) média aumentou 58,97 kbits/segundo em tráfego recebido e 42,01 kbits/segundo em tráfego enviado. À semelhança do Nagios, o escalonador do Zenoss tem a capacidade de

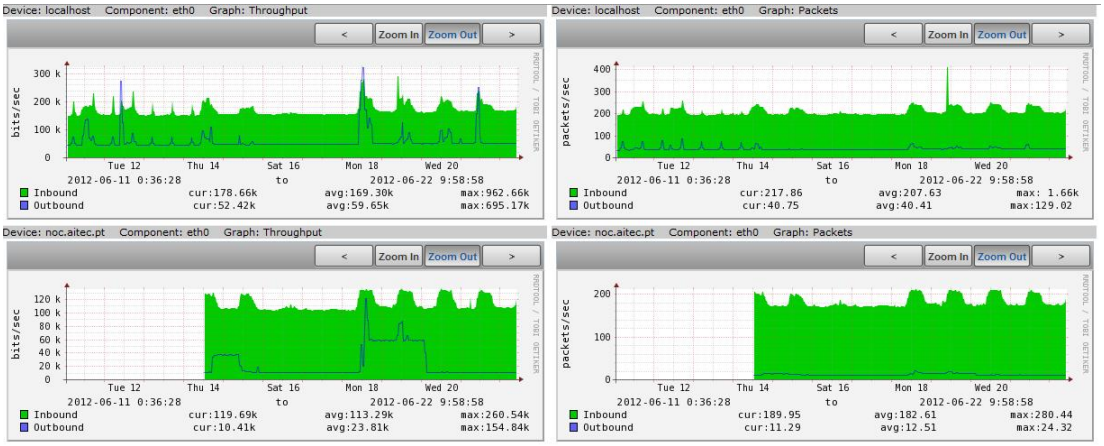

Figura 5.8: Utilização de rede pelo Zenoss e Nagios.

distribuir os ciclos de verificações, contribuindo assim para uma utilização de largura de banda constante e evitando os picos de tráfego.

5.3.3 Backups

O sistema de backup do Zenoss efectua uma cópia de todos os dados recolhidos e configurações do Zenoss, nomeadamente a base de dados dos eventos, a base de dados dos utilizadores e dispositivos, as configurações e os dados de desempenho (gráficos).

O componente do Zenoss utilizado para efectuar os backups é denominado `zenbackup`. O restauro de backups também é suportado pelo Zenoss através da utilização do aplicativo `zenrestore`.

Os backups regulares são efectuados com recurso ao escalonador de tarefas do Linux (`crontab`), que permite que o `zenbackup` seja executado em intervalos regulares. O backup integral da aplicação é efectuado semanalmente às 00:00 de Domingo para um directório local. Durante todos os dias da semana, em horário de baixo congestionamento da rede, é efectuado um backup dos dados referentes aos eventos e aos gráficos criados durante a monitorização. A decisão de apenas efectuar backups a este tipo de dados prende-se com o facto de serem dados que são recolhidos constantemente e por esse motivo podem ser necessários para análise, ao contrário das configurações que, regra geral, são alteradas apenas em casos específicos como, por exemplo, ao adicionar um novo dispositivo.

A implementação actual desta solução não prevê a existência de backups remotos. Esta decisão foi tomada devido à exportação regular de dispositivos e eventos presentes na aplicação para fins de histórico. A informação exportada permite ser utilizada para reposição

em caso de falha do sistema.

A aplicação não apresenta limitações no que diz respeito à criação de backups remotos pelo que, caso se venha a verificar essa necessidade é possível efectuar essa operação sem complexidade acrescida..

5.4 Funcionalidades Adicionadas

Para além das tarefas de monitorização e alarmística, a nova solução, implementada na Linkcom, acrescenta algumas funcionalidades de gestão de rede, tais como o inventário dos dispositivos monitorizados e respectivas características, propriedades de dispositivos personalizadas e a localização física de dispositivos.

5.4.1 Propriedades Personalizadas

A personalização das propriedades dos dispositivos permite adicionar um conjunto de novos campos com informação relevantes e que não se encontram definidos por omissão na aplicação de monitorização. É possível adicionar campos de vários tipos, como numérico, alfa-numérico ou booleano, que têm como objectivo definir propriedades especificas para os dispositivos. Esta funcionalidade é uma mais valia quando se deseja definir um conjunto de propriedades relevantes que não foram contempladas no desenvolvimento da aplicação.

Algumas das propriedades personalizadas que foram definidas nesta implementação da aplicação estão representadas na Figura 5.9.

Neste caso foram acrescentadas às propriedades do dispositivo algumas informações relevantes para o negócio, como o cliente, representado pelo campo "Empresa"ou se é um dispositivo físico ou virtual, representado pelo valor booleano dos campos "Fisico"ou "VM".

Como forma de exportar as características dos dispositivos monitorizados, juntamente com as propriedades personalizadas, foi criado um script que permite organizar toda esta informação em formato de folha de cálculo. Foi escolhido o formato de folha de cálculo de modo a ser compatível com o formato utilizado pela Linkcom. A descrição do script desenvolvido encontra-se no Anexo C.

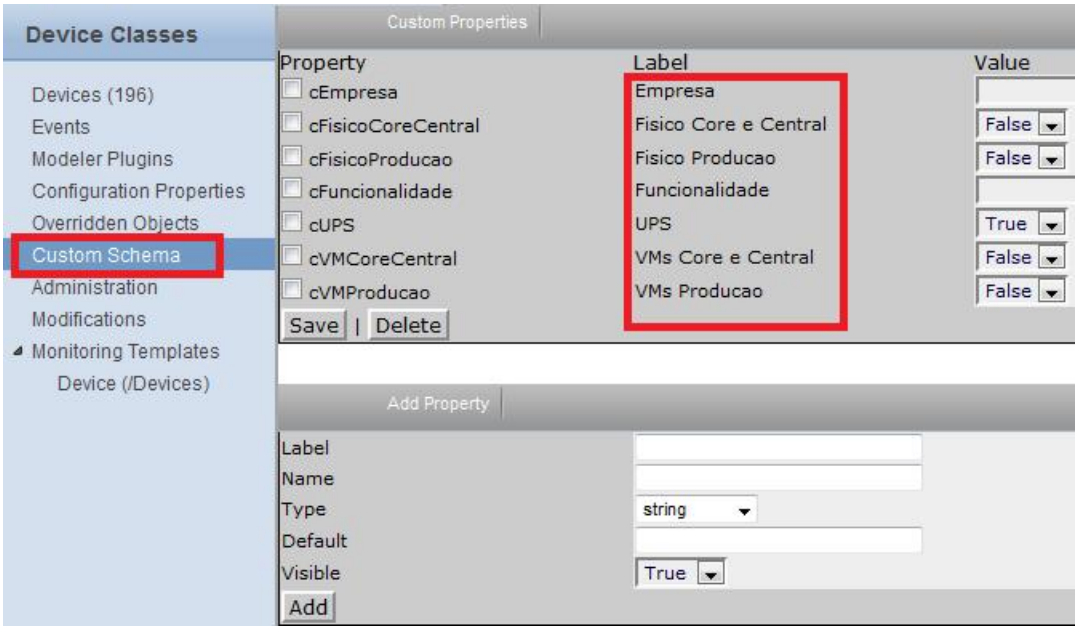

Figura 5.9: Propriedades personalizadas de um dispositivo.

5.4.2 Localização Física

A localização física de dispositivos, exemplificada na Figura 5.10, permite identificar facilmente o local onde se encontra o dispositivo, bem como o estado das ligações entre as várias localizações.

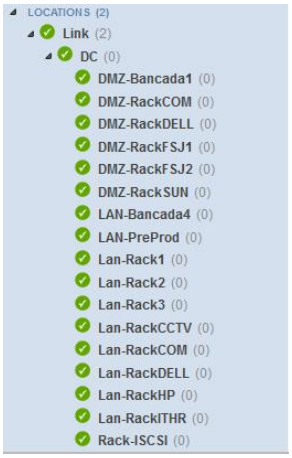

Figura 5.10: Localização de dispositivos.

Neste caso a localização é apresentada sob a forma de uma árvore, na qual a localização mais genérica é `Link`, e as mais especificas são todas as localizações que derivam de DC (*Data Center*).

No caso de existirem dispositivos em várias localizações cada localização é adicionada ao mapa disponibilizado pela aplicação, como exemplificado na Figura 5.11

Figura 5.11: Localizações representadas no mapa [9].

5.4.3 Monitorização de Aplicações

A monitorização de aplicações, nomeadamente Active Directory, Microsoft Exchange e Microsoft SQL Server, é efectuada através de WMI e permite a monitorização de vários parâmetros referentes a cada aplicação. Assim, cada aplicação possui parâmetros específicos a monitorizar.

1. Active Directory:
   - **Estatísticas da Active Directory**: Recolhe estatísticas de acesso à AD, nomeadamente pesquisa, leitura e escrita
   - **Sessões de clientes**: Apresenta as sessões activas de clientes autenticados por LDAP.
2. Microsoft Exchange:
   - **Utilizadores Outlook Wed Application (OWA)**: Mostra estatísticas de utilizadores ligados ao OWA, nomeadamente total de utilizadores ligados, utilizadores ligados através da interface light e utilizadores ligados através da interface premium.
   - **Tempo de resposta**: Apresenta tempo de resposta a pedidos efectuados ao Exchange

- **Estatísticas SMTP**: Mostra o número de ligações SMTP de entrada e saída

3. Microsoft SQL Server:

   - **Bases de Dados**: Apresenta as bases de dados existentes no servidor
   - **Tamanho das Bases de Dados**: Apresenta o tamanho total e o espaço utilizado por cada base de dados e respectivos logs
   - **Transacções**: Mostra o número de transacções por cada base de dados
   - **Transacções Pendentes**: Apresenta o número total de transacções pendentes

5.4.4 Monitorização Distribuída

A monitorização distribuída permite distribuir as tarefas de monitorização e gestão de eventos através da distribuição de instâncias do `Zenoss` em vários servidores. Esta funcionalidade permite monitorizar redes que não se encontram directamente acessíveis através do servidor Zenoss, dividir a carga de monitorização por vários servidores ou definir diferentes configurações para cada servidor como, por exemplo, ciclos de verificações.

Esta funcionalidade é suportada de forma nativa pelo Zenoss e é denominada "*Distributed Collector*", é composta por "*collectors*"e um "*hub*". Os "*collectors*"são um conjunto de servidores que monitorizam um conjunto de dispositivos ou uma rede e enviam os dados recolhidos ao "*hub*". O "*hub*"é um servidor que coordena e permite configurar todos os "*collectors*"e apresenta os dados recolhidos ao utilizador.

Os "*collectors*"podem ser utilizados para distribuição de carga, em casos onde a monitorização se torna uma tarefa pesada para um único servidor, ou para monitorização de locais remotos, nos casos em que é necessário monitorizar uma rede numa localização externa. A utilização de um "*collector*"para monitorização de locais remotos permite que apenas seja necessário configurar permissões de ligação, por exemplo numa firewall, para um único dispositivo que monitoriza a rede remota, evitando configurar permissões para todos os dispositivos remotos monitorizados.

Esta funcionalidade foi implementada na rede da Linkcom, no âmbito desta tese, de modo a monitorizar uma localização remota. Foi estabelecida uma ligação segura com a localização remota, na qual foi instalada um *collector*, como exemplificado na Figura 5.12. A monitorização remota é efectuada pelo *collector* (`ZenCollector`) e os dados são enviados para o *hub* (`ZenHub`) localizado na Linkcom.

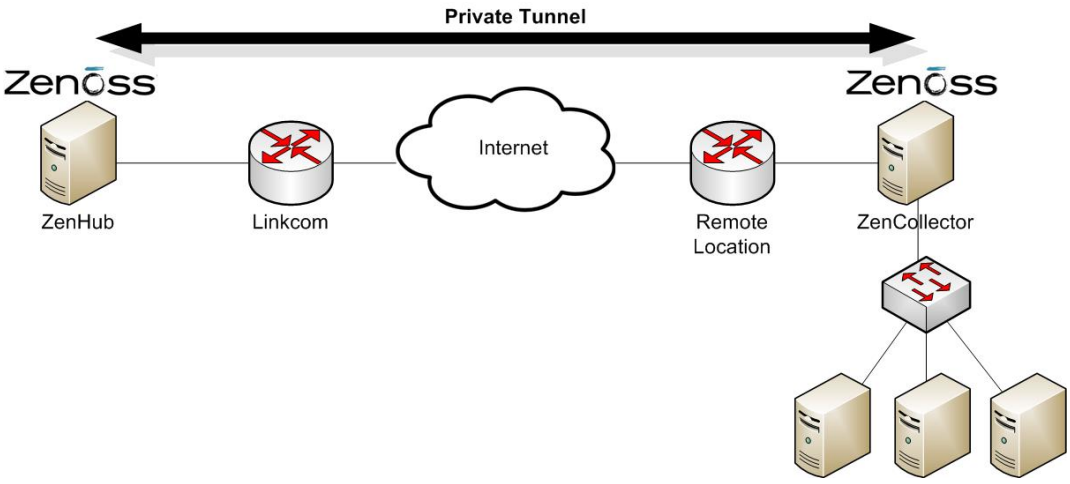

Figura 5.12: Esquema de monitorização distribuída

.

5.5 Resultados e Discussão

Após a migração dos dispositivos, dos utilizadores e dos respectivos grupos para a nova solução de monitorização, esperava-se um aumento de eventos, que se veio a verificar. Este aumento deve-se ao facto da nova solução de monitorização analisar um maior número de componentes, como Servidor MS SQL ou Active Directory e eventos do Syslog (Linux) e Eventlog (Windows), que levaram à apresentação de alertas que não se encontravam definidos na solução anterior, como ocupação dos discos, carga de CPU ou monitorização de *logs* dos sistemas. Outro dos motivos que levou ao aumento do número de eventos de alerta foi a existência de dispositivos que possuíam configurações erradas, como credenciais ou métodos de verificação, as quais tiveram que ser rectificadas de modo a eliminar esse conjunto de falsos positivos.

O aumento de parâmetros monitorizados permite ter uma visualização mais correcta do estado geral da rede, quando comparado com a solução de monitorização anterior. Uma vez que é possível visualizar um conjunto de parâmetros de forma integrada que não se verificava até à implementação da nova solução.

A organização simplificada dos grupos de dispositivos e utilizadores permite uma melhor gestão de períodos de disponibilidade dos utilizadores e a atribuição de responsáveis por grupos de dispositivos ou dispositivos individuais.

A possibilidade de adicionar propriedades personalizadas aos dispositivos permite definir um conjunto de informações, como as referentes ao negócio ou outras adicionais, que não se encontram por omissão na aplicação mas que são importantes para a documentação da infra-estrutura de rede.

A funcionalidade de exportação das características dos dispositivos permite, a qualquer momento, obter um documento no formato de folha de calculo actualizada com todos os dispositivos de rede monitorizados.

A interface mais apelativa do Zenoss e a facilidade de utilização são uma mais valia que permite uma utilização da solução mais simples e intuitiva comparativamente à utilizada anteriormente pelo Nagios e Cacti.

Capítulo 6

Conclusões

A gestão e monitorização de rede não depende apenas da solução implementada, mas do conhecimento das tecnologias envolvidas na infra-estrutura de rede. Uma vez que uma solução deste tipo apenas permite a identificação da origem dos problemas, é necessário uma equipa que possua o conhecimento dos sistemas monitorizados de modo a resolver os problemas identificados em tempo útil.

Ainda assim não se deve eliminar a utilização de uma aplicação que permita automatizar as tarefas de monitorização dos vários componentes da rede de modo a identificar, de forma rápida e detalhada, os problemas duma rede. Deste modo o tempo que os administradores de rede utilizavam para localizar o problema é libertado para a resolução de problemas.

A análise das aplicações tirou partido da existência de um cenário de testes, onde foi possível obter uma experiência mais prática das aplicações analisadas e respectivas funcionalidades, bem como do método de interacção com o utilizador. Esta abordagem também permitiu identificar quais as possíveis dificuldades na instalação e configuração das aplicações que, nem sempre, se encontram reflectidas na documentação. Por exemplo a utilização de versões especificas de dependências da aplicação, que podem diferir das versões actualmente disponibilizadas.

Um dos requisitos da nova solução de monitorização é a recolha de dados sem recurso a agentes de uma aplicação de monitorização, o que implica que tenham de ser utilizados protocolos e métodos de monitorização que os dispositivos suportem. O SNMP é considerado o protocolo standard de monitorização e por esse motivo encontra-se na maioria dos dispositivos. No entanto nem todos os dispositivos que se encontram na rede da Linkcom disponibilizam todas as informações necessárias através deste protocolo. Foi por isso necessária a utilização de métodos alternativos de monitorização, como o WMI, para dispositivos

Windows ou VMWare API para servidores VMWare.

De todas as aplicações analisadas a que melhor se comportava com a utilização de todos os métodos definidos para monitorização da rede da Linkcom é o Zenoss, garantindo a monitorizar uma rede heterogénea sem recorrer a agentes. Este foi um dos principais factores para a sua selecção como aplicação de monitorização e alarmística a implementar na Linkcom. Para além destas funcionalidades, o Zenoss consegue identificar quais os componentes dos dispositivos monitorizados automaticamente e possui uma interface apelativa e, na maioria das tarefas, intuitiva.

O Zenoss Core que é a versão gratuita da aplicação Zenoss Enterprise, cuja principal diferença, para além do suporte prestado, são as funcionalidades disponibilizadas por omissão após instalação. A grande maioria destas funcionalidades permite ser adicionada através da instalação de plugins, colmatando assim essa limitação. Apesar de ser uma aplicação *open-source* a documentação disponível é bastante clara e existe uma comunidade activa em torno deste produto, o que permite obter algum suporte para além da documentação da aplicação.

O Zenoss Core foi a aplicação que melhor se aplica ao caso concreto da Linkcom, no entanto pode ser utilizado em qualquer outra empresa com as mesmas características, em que se pretende a recolha detalhada de dados de desempenho e a monitorização deve ser efectuada sem recorrer a agentes.

Com a finalização desta tese pode-se concluir que, normalmente, uma solução de monitorização e alarmística *open-source* responde às necessidades e exigências do meio empresarial. Para além de ser uma alternativa escalável e de baixo custo, consegue-se obter um retorno de investimento mais imediato que com a utilização de aplicações proprietárias mais complexas em termos de configuração e com custos, normalmente, mais elevados.

6.1 Trabalho Futuro

Apesar da solução implementada permitir a monitorização de disponibilidade e desempenho dos vários dispositivos, existem algumas funcionalidades que poderão ser desenvolvidas constituindo uma mais valia e enriquecendo a solução final:

- **Detecção automática da classe do dispositivo** - Esta funcionalidade permitirá associar um dispositivo à classe correcta, permitindo a sua correcta monitorização assim que é adicionado. A utilização de todos os métodos possíveis de recolha de dados, como WMI, SNMP, VMWARE API, etc. permite identificar qual o tipo de dispositivo

(Linux, Windows, switch, etc.). Baseado nesta identificação, é possível criar um mecanismo que aplique aos dispositivos os templates adequados.

- **Criar um sistema de alertas de SLA** - Esta funcionalidade permitirá avisar os responsáveis caso a disponibilidade de um serviço se encontre em níveis considerados críticos, de modo a dar uma resposta mais urgente a um serviço especifico, no caso da existência de vários alertas em simultâneo.

- **Integrar na interface web funcionalidades desenvolvidas** - No âmbito desta dissertação foram desenvolvidas algumas funcionalidades, como exportação do inventário dos dispositivos ou classificação por classes de dispositivos descobertos, que apenas podem ser executados pela linha de comandos. A integração destas funcionalidades na interface permitiria a sua utilização sem a necessidade de as executar remotamente através da linha de comandos;

- **Efectuar ligação remota a dispositivos monitorizados** - permitir ligações remotas através de SSH, RDP ou telnet, directamente a partir da solução de monitorização. Esta funcionalidade é implementada através da utilização de aplicações de terceiros como Putty, para ligações SSH e telnet, ou Remote Desktop Connection para ligações RDP.

- **Solução de alta disponibilidade** - nesta fase não foi prevista a implementação de alta disponibilidade para o sistema de monitorização, no entanto, a implementação de um cluster de alta-disponibilidade é possível através do projecto Linux-HA (Linux-High Availability).

Bibliografia

[1] Raj Jain. Network management. page 9, 2009.

[2] Galstad E. *Nagios Core Version 3.x Documentation*. Nagios Community.

[3] Kocjan W. *Learning Nagios 3.0: A detailed tutorial to setting up, configuring,and managing this easy and effective system monitoring software*. Packt Publishing, 2008.

[4] Cacti Community. Cacti 0.8.7 manual (http://docs.cacti.net/ manual:087:2_basics.0_principles_of_operation). Internet. Consultado em 03-11-2011.

[5] Olups R. *Zabbix 1.8 Network Monitoring*. Packt Publishing Ltd, 2010. ISBN 978-1-847197-68-9.

[6] M. Badger. *Zenoss Core 3.x Network and System Monitoring*. Packt Publishing, April 2011. ISBN 978-1-849511-58-2.

[7] http://www.shinken-monitoring.org/wiki/the_shinken_architecture. Consultado em 11 de Outubro de 2011.

[8] http://www.shinken-monitoring.org/wiki/how_dispatching_works. Consultado em 11 de Outubro de 2011.

[9] Zenoss discussion in zenoss-user board (http://community.zenoss.org/message/48265), April 2010.

[10] Nagios website (http://www.nagios.org/). Consultado em 8 de Fevereiro de 2012.

[11] Cacti website (http://www.cacti.net/). Consultado em 8 de Fevereiro de 2012.

[12] Linkcom sistemas de informação http://www.linkcom.pt. Concultado em 23 de Maio de 2012.

[13] ITU-T. Specification documents m.3000, m.3400, 02 2000.

[14] Tmn management functions, 02 2000.

[15] Murthy Goyal, Pankaj; Mikkilineni; Ganti. *FCAPS in the Business Services Fabric Model*. 2009.

[16] Stallings W. Snmp, snmpv2, snmpv3, and rmon 1 and 2. Addison-Wesley, 1999.

[17] J. Case, M. Fedor, M. Schoffstall, and J. Davin. Snmpv1 rfc 1157, May 1990.

[18] U. Blumenthal and B. Wijnen. Snmpv3 rfc 3414, December 2002.

[19] IETF. User-based security model(usm) rfc2574. IETF.

[20] M. Rose. Mib-ii (rfc 2970). RFC, March 1991.

[21] A. Boshier. "windows management intrumentation: A simple, powerful tool forscripting windows management". *MSDN Magazine*, 2003.

[22] Clemm A. *Network Management Fundamentals: a guide to understanding how network management technology really works.* Cisco Press, 2007.

[23] B.W Schermer. *Schermer, B.W., Software agents, surveillance, and the right to privacy: A legislative framework for agent-enabled surveillance. Leiden University Press, 2007, p.140.* Leiden University Press, 2007.

[24] Turnbull J. *Pro Nagios 2.0.* Apress, 2006.

[25] Cacti Community. Cacti 0.8.7 manual (http://docs.cacti.net/manual:087:1_installation.2_install_windows Internet. Consultado em 03-11-2011.

[26] Zabbix website (http://www.zabbix.com/). Consultado em 8 de Fevereiro de 2012.

[27] Intel, Hewlett-Packard, NEC, and Dell. *IPMI - Intelligent Plataform Management Interface Specification Second Generation*. Intel and Hewlett-Packard and NEC and Dell, February 2004.

[28] Zenoss website (http://community.zenoss.org/). Consultado em 8 de Fevereiro de 2012.

[29] Jan Clifford, David e van Bon. *David Clifford, Jan van Bon (2008). Implementing ISO/IEC 20000 Certification: The Roadmap. ITSM Library. . ISBN 90-8753-082-X.* Van Haren Publishing, 2008.

[30] Shinken website (http://www.shinken-monitoring.org/). Consultado em 8 de Fevereiro de 2012.

[31] Galstad E., Guilbaud S., Jan O., and Gabès J. Official shinken documentation (http://www.shinken-monitoring.org/wiki/ official/gettingstarted-quickstart-gnulinux). Consultado em 2 de Janeiro de 2012.

[32] Shinken Community. The shinken architecture (http://www.shinken-monitoring.org/wiki/the_shinken_architecture). Consultado em 2 de Janeiro de 2012.

[33] Shinken Community. Shinken features (http://www.shinken-monitoring.org/features). Consultado em 2 de Janeiro de 2012.

[34] Shinken Community. Shinken dispatch (http://www.shinken-monitoring.org/wiki/how_dispatching_works). Consultado em 2 de Janeiro de 2012.

[35] Shinken Community. Configure shinken(http://www.shinken-monitoring.org/wiki/official/configuringshinken-config). Consultado em 02 de Janeiro de 2012.

[36] CentOS Project. http://www.centos.org/. consultado em 12-12-2011.

[37] Vmware esxi (http://www.vmware.com/products/vsphere/esxi-and-esx/overview.html). Consultado em 8 de Fevereiro de 2012.

[38] Rajkumar Buyya. Parmon: a portable and scalable monitoring system for clusters. 1999.

[39] P. Grillo S. Waldbusser. Host resource mib (rfc 2970). RFC, March 2000.

[40] Net-snmp application suite mibs: http://net-snmp.sourceforge.net/docs/mibs/, March 2012. Consultado em 19 de Março de 2012.

[41] Gnokii website (http://gnokii.org/). Consultado em 30 de Maio de 2012.

Anexo A

Script de envio de SMS

A solução implementada utiliza o Gnokii como meio de interacção com o modem GSM para envio de SMS. Não foi utilizado o script original de envio de mensagens do Zenoss, uma vez que este está optimizado para envio de mensagens de pager. No entanto foi adaptado para utilização do Gnokii e, assim, enviar SMSs.

O script final encontra-se a seguir:

```
#!/bin/bash

RETVAL=0
LOGFILE="/var/log/zenossSMSlog.log"

# Make sure there is a recipient defined
if [[ -z $RECIPIENT ]]; then
    echo `date +%x` `date +%r`: NO recipient recieved, please Check. >> ${LOGFIL
    exit 1
fi

# Retrieve the message. Zenoss stores in stdin until it is read. Exit if error

read -t2 MSG

# Create the command used to send the page
CMD=`echo ${MSG} | gnokii --sendsms ${RECIPIENT}`
LOG="echo ${MSG} | gnokii --sendsms ${RECIPIENT}"
```

```
# Log the command if $QPAGE_LOG_MESSAGES is defined

if [[ -n $LOGFILE ]]; then
    echo `date +%x` `date +%r`: $LOG >> ${LOGFILE}
fi

#Execute the command

$CMD
RETVAL=$?
exit $RETVAL
```

As alterações introduzidas foram principalmente a substituição do comando que enviava a mensagem para pager, através do qpage:

```
qpage -f ${QPAGE_SENDER} -p $ {RECIPIENT} ${MSG}
```

para o comando que utiliza o gnokii para envio de mensagens SMS:

```
`echo ${MSG} | gnokii --sendsms ${RECIPIENT}`
```

Nestes comandos a variável *RECIPIENT* representa o número de contacto e a variável *MSG*, representa a mensagem a ser enviada. Estas variáveis são enviadas para script pelo próprio Zenoss

Para além do envio de mensagens o script guarda, em ficheiro de log, a data em que os comandos foram executados.

Anexo B

Exemplo de utilização do `zenbatchload`

O aplicativo `zenbatchload` é parte integrante do Zenoss e é utilizado para adicionar novos dispositivos à aplicação quando se deseja um grande nível de especificação de configurações. O `zenbatchload` permite definir várias propriedades dos dispositivos a adicionar, tais como a classe a que serão adicionados, grupo a que pertencem ou qual o porto a utilizar para consultas SNMP.

A sintaxe utilizada para a execução da aplicação é a seguinte:

```
zenbatchload -i devicelist.txt
```

O ficheiro `devicelist.txt` deverá ter a estrutura exemplificada de seguida:

```
/Server/Linux
192.168.2.54 setGroups=['/myGroup']
192.168.2.58
(...)
/Network/Router/Firewall
194.117.36.62 setSnmpPort='999'
194.117.37.254
(...)
/Network/Mikrotik
194.117.39.8
194.117.39.11
```

```
(...)
/Network/Switch
192.168.1.35
192.168.1.36
(...)
"/Server/Virtual Machine Host/ESXi"
192.168.3.193
192.168.3.194
(...)
/Server/Windows
10.50.55.1
146.193.1.1
(...)
```

Neste exemplo são definidas as classes, tais como `/Server/Linux` ou `/Network/Switch`, e os dispositivos que serão adicionados a cada uma delas encontram-se imediatamente a seguir. Neste exemplo os dispositivos com endereço IP 192.168.2.54 e 192.168.2.58 serão adicionados à classe `/Server/Linux`. Existem outras propriedades que podem ser definidas e são apenas referentes a um dispositivo especifico. Neste exemplo pode-se observar que um dispositivo foi adicionado ao grupo `myGroup` e outro foi especificado porto SNMP para 999.

Anexo C

Script de exportação de dispositivos para folha de cálculo

O script apresentado visa extrair informações relevantes dos dispositivos monitorizados da infra-estrutura de modo a serem utilizados para manter uma listagem externa dos dispositivos e respectivas caracteristicas.

O ficheiro de criado é uma folha de calculo Excel onde são gravadas as caracteristicas dos vários dispositivos, juntamente com as respectivas propriedades personalizadas.

```
#!/usr/bin/env python

import Globals,re
import xlwt

from Products.ZenUtils.ZenScriptBase import ZenScriptBase
from Products.ZenUtils.Utils import convToUnits
from transaction import commit


class device:

def __init__(self, deviceClass, company, funcionalidade, deviceId, location, pro
if (deviceClass == 'WMI'):
self.deviceClass = 'Windows'
else:
```

```
self.deviceClass = deviceClass
self.company = company
self.funcionalidade = funcionalidade
self.deviceId = deviceId
self.location = location
self.prodState = prodState
self.cpu = cpuInfo
self.filesystem = filesystemInfo
self.filesystemTotalSize = filesystemTotalSize

self.ram = ram
self.priority = priority
self.fisicoCoreCentral = fisicoCoreCentral
self.vmCoreCentral = vmCoreCentral
self.fisicoProducao = fisicoProducao
self.vmProducao = vmProducao
self.ups = ups
self.IpAddr = deviceIp



self.hardwaremodel = "Fisico"

if "Virtual" in hwModel:
self.hardwaremodel = "VM"
if ".1.3.6.1.4.1.11.2.3.7.11" in hwModel:
self.hardwaremodel = "Switch"
#alterar a pesquisa em deviceClass por alternativa
if "ESX" in self.deviceClass:
self.hardwaremodel = "Host VM"
if "Hyper-V" in self.deviceClass:
self.hardwaremodel = "Host VM"



class networkInterface:
```

```
def __init__(self, name, iterableIP):
self.name = name
self.IP_list.append(iterableIP)

class hwCard:

def __init__(self, prodName, manufacturerName):
self.prodName = prodName
self.manufacturerName = manufacturerName

class cpu:

def __init__(self, cacheL2, index, manufacturer, clockRate):
self.cacheL2 = cacheL2
self.index = index
self.manufacturer = manufacturer
self.clock = clockRate

class filesystem:

def __init__(self, totalSize, filesystemId, usedSpace, freeSpace):
self.totalSize = totalSize
self.filesystemId = filesystemId
self.free = freeSpace
self.used = usedSpace

def sizeof_fmt(num):
for x in ['bytes','KB','MB','GB','TB']:
if num < 1024.0:
return "%3.1f%s" % (num,x)
num /= 1024.0


devices = []

dmd = ZenScriptBase(connect=True).dmd
```

```
for dev in dmd.Devices.getSubDevices():
auxFisicoCoreCentral = False
        auxFisicoProducao = False
        auxVMCoreCentral = False
        auxVMProducao = False
        auxUPS = False
        interface = True


osType = dev.getPrimaryPath()[5]
# deviceId = dev.id
deviceId = dev.name()
deviceIp = dev.manageIp
deviceState = dev.getProductionStateString()
rackSlot = dev.rackSlot
priority = dev.getPriorityString()
hwManufacturername = dev.getHWManufacturerName()
hwProductionName = dev.getHWProductName()
operativeSystem = dev.getOSProductName()
ram = sizeof_fmt(dev.hw.totalMemory)
location = dev.getLocationName()
#group list
auxGroup = []
for group in dev.getDeviceGroupNames():
auxGroup.append(group)

company = dev.cEmpresa
funcionalidade = dev.cFuncionalidade

if dev.cFisicoCoreCentral == True:
fisicoCoreCentral = True
if dev.cFisicoProducao == True:
fisicoProducao = True
if dev.cVMCoreCentral == True:
vmCoreCentral = True
if dev.cVMProducao == True:
vmProducao = True
```

```
if dev.cUPS == True:
UPS = True

owner = dev.hw.getOwner()

#device class
rawClass = dev.getDeviceClassPath().split('/',10)
deviceClass = rawClass[len(rawClass)-1]

# get interfaces from devices
auxNetworkInterfaces = []
auxNetworkIPs = []
if len(dev.os.interfaces()) > 0:
for network in dev.os.interfaces():
for aux in network.ipaddresses():
auxNetworkInterfaces.append(aux.name())
auxNetworkIPs.append(aux.getIpAddress())
else:
interface = False


#extra hardware
if len(dev.hw.cards()) > 1:
for a in dev.hw.cards():
auxCardManufacturer.append(p.getManufacturername())
auxCardProductionName.append(p.getProductname())


cpuinfo = []

cpuCount = 0
for c in dev.hw.cpus():
auxCpuL2Cache = sizeof_fmt((c.cacheSizeL2*1000))
auxCpuClockspeed = sizeof_fmt(c.clockspeed*1000)
auxCpuManufacturer = c.getModelName()
cpuCount += 1
cpuinfo.append(cpu(auxCpuL2Cache,cpuCount,auxCpuManufacturer,auxCpuClockspeed))
```

```
filesystemInfo = []

totalfilesize = 0
for f in dev.os.filesystems():
size = f.totalBlocks*f.blockSize
totalfilesize += size

#windows drives
if osType == "Windows":
auxFilesystemId = f.id
if osType == "WMI":
auxFilesystemId = f.id
#linux drives
if osType == "Linux":
auxFilesystemId = f.mount
#list of filesystems size
auxFilesystemSize = sizeof_fmt(size)
if osType != "Linux":
tmpUsedSpace = f.usedBytesString()
tmpFreeSpace = f.availBytesString()

filesystemInfo.append(filesystem(auxFilesystemSize, auxFilesystemId,tmpUsedSpace,tmpFr
auxFilesystemTotalSize = sizeof_fmt(totalfilesize)

hardwareModel = dev.hw.getModelName()


#create device
devices.append(device(deviceClass, company, funcionalidade, deviceId, location, device


wbk = xlwt.Workbook()
sheet_devices = wbk.add_sheet('devices')

#configuracao de estilos
```

```
sheet_devices.panes_frozen = True
sheet_devices.remove_splits = True
sheet_devices.horz_split_pos = 1
sheet_devices.horz_split_first_visible = 1

sheet_devices.row(0).write(0,'Tipo Maq')
sheet_devices.row(0).write(1,'Empresa')
sheet_devices.row(0).write(2,'Host')
sheet_devices.row(0).write(3,'IP')
sheet_devices.row(0).write(4,'Tipo')
sheet_devices.row(0).write(5,'CPU')
sheet_devices.row(0).write(6,'HDD [GB]')
sheet_devices.row(0).write(7,'Drive Letter')
sheet_devices.row(0).write(8,'Total')
sheet_devices.row(0).write(9,'Usado')
sheet_devices.row(0).write(10,'Livre')
sheet_devices.row(0).write(11,'RAM')
sheet_devices.row(0).write(12,'Prioridade')
sheet_devices.row(0).write(13,'Fisico Core e Central')
sheet_devices.row(0).write(14,'VMs Core e Central')
sheet_devices.row(0).write(15,'Fisico de Producao')
sheet_devices.row(0).write(16,'VMs de Producao')
sheet_devices.row(0).write(17,'UPS')



for i in range(0, len(devices)):
sheet_devices.row(i+1).write(0,devices[i].hardwaremodel)
sheet_devices.row(i+1).write(1,devices[i].company)
sheet_devices.row(i+1).write(2,devices[i].deviceId)
sheet_devices.row(i+1).write(3,devices[i].IpAddr)
sheet_devices.row(i+1).write(4,devices[i].deviceClass)
#tratamento do CPU
tmpCpuInfo = ''
for cpu in devices[i].cpu:
tmpCpuIndex = cpu.index
tmpCpuMan = cpu.manufacturer
```

```
tmpCpuClock = cpu.clock

tmpCpuInfo += "#" + str(cpu.index) + " " + tmpCpuMan + " "

sheet_devices.row(i+1).write(5,tmpCpuInfo)

#tratamento dos filesystems
tmpFilesizeList = ''
tmpFileDriveList = ''
tmpFreeSpaceList = ''
tmpUsedSpaceList = ''
for fs in devices[i].filesystem:

tmpFileSize = fs.totalSize
tmpFilesystemId = fs.filesystemId

tmpFilesizeList += (str(tmpFileSize) + "; ")
tmpFileDriveList += (str(tmpFilesystemId) + "; ")
tmpFreeSpaceList += (fs.free + "; ")
tmpUsedSpaceList += (fs.used + "; ")

sheet_devices.row(i+1).write(6,tmpFilesizeList)
sheet_devices.row(i+1).write(7,tmpFileDriveList)
sheet_devices.row(i+1).write(8,devices[i].filesystemTotalSize)
sheet_devices.row(i+1).write(9,tmpUsedSpaceList)
sheet_devices.row(i+1).write(10,tmpFreeSpaceList)

sheet_devices.row(i+1).write(11,devices[i].ram)
sheet_devices.row(i+1).write(12,devices[i].priority)
sheet_devices.row(i+1).write(13,str(devices[i].fisicoCoreCentral))
sheet_devices.row(i+1).write(14,str(devices[i].vmCoreCentral))
sheet_devices.row(i+1).write(15,str(devices[i].fisicoProducao))
sheet_devices.row(i+1).write(16,str(devices[i].vmProducao))
sheet_devices.row(i+1).write(17,str(devices[i].ups))



wbk.save('deviceList.xls')
```

Anexo D

Script de atribuição de classes automatizada

Com o intuito de automatizar a alteração dos dispositivos da classe "Discovery"para uma classe mais apropriada, como a classe "Server/Linux"para dispositivos com esse sistema operativo, foi criado um script em *python* para o efeito.

O script le a propriedade "OSProductName", que contém o nome do sistema operativo, e move o dispositivo para a classe correcta.

Este script encontra-se configurado para a realidade da Linkcom, pelo que são identificados quatro sistemas:

- **Windows**
- **Linux**
- **VmWare**
- **Procurve** [1]

```
discovered = dmd.Devices.Discovered

for device in discovered.getSubDevices():
prodkey = device.getOSProductName().lower()
#verify if the product is linux
```

[1] Refere-se aos switches HP Procurve

```
if prodkey.find('linux') != -1:
discovered.moveDevices('/Server/Linux',device.id)
print 'Linux match - moving device %s to class /Server/Linux' %device.id
#verify if the product is Windows
elif prodkey.find('windows') != -1:
discovered.moveDevices('/Server/Windows',device.id)
print 'Windows match - moving device %s to class /Server/Windows' %device.id
#verify if the product is VMware
elif prodkey.find('vmware') != -1:
discovered.moveDevices('/Server/Virtual Machine Host/ESXi',device.id)
print 'VMware match - moving device %s to class /Server/Visrtual Machine Host/ESXi' %d
#verify if the product is HP ProCurve
elif prodkey.find('procurve') != -1:
discovered.moveDevices('/Network/Switch/HP',device.id)
print 'ProCurve match - moving device %s to class /Network/Switch/HP' %device.id
```