Instituto Superior de Engenharia do Porto

UMA APLICAÇÃO DE GESTÃO DE REDES BASEADA NO PARADIGMA DOS AGENTES MÓVEIS

Joaquim Ricardo Azevedo Teixeira

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática

Área de Especialização em
Arquitecturas Redes e Sistemas

Orientador: Doutora Maria João Monteiro Ferreira Viamonte

Júri:

Presidente:

Doutor Luís Miguel Moreira Lino Ferreira, Professor Adjunto, Instituto Superior de Engenharia do Porto

Vogais:

Doutor Alexandre Manuel Tavares Bragança, Professor Adjunto, Instituto Superior de Engenharia do Porto

Doutora Maria João Monteiro Ferreira Viamonte, Professor Adjunto, Instituto Superior de Engenharia do Porto

Porto, Novembro de 2011

À Minha Esposa

Ao David e à Mariana

Agradecimentos

Gostaria de agradecer a todos aqueles que, de alguma forma, contribuíram para que este trabalho fosse realizado. Agradeço especialmente:

À minha orientadora, Doutora Maria João Monteiro Ferreira Viamonte, por toda a ajuda, pela disponibilidade que sempre teve, pelos conselhos que me deu, pelo exemplo de humildade e dedicação que é, e acima de tudo pelo trabalho extraordinário que efectuou na revisão desta dissertação;

Ao GECAD (Grupo de Investigação em Engenharia do Conhecimento e Apoio à Decisão), por todos os meios disponibilizados para a realização deste trabalho. Uma palavra de apreço especial para o Doutor Carlos Ramos e para a Doutora Zita Vale, por todo o apoio que me têm transmitido ao longo dos meus anos no GECAD;

Ao Bruno Canizes, Carlos Freitas, Doutora Goreti, Doutora Isabel Praça, Eng. Sérgio Ramos, Filipe Fernandes, Hugo Morais, Hussein Khodr, Ivo Pereira, João Laranjeira, João Soares, Marco Silva, Pedro Faria, Ricardo Costa, Tiago Soares, Tiago Sousa, Tiago Pinto, entre outros, por valorizarem a minha presença no GECAD, tornando-o mais do que um mero ambiente de trabalho;

Ao corpo Docente e não Docente do Instituto Superior de Engenharia do Porto;

Aos meus Amigos Ana Isabel, Cristina, Gilberto e Zeferino, pelos bons momentos passados ao longo do último ano;

Aos meus dois Sobrinhos, David e Mariana, e à Beatriz Pinheiro. Vocês são únicos, lembram-me a criança que ainda tenho dentro de mim, mas acima de tudo porque vocês acham tudo em nada, enquanto muitos adultos não acham nada em tudo;

Aos meus Sogros, que me ajudaram de várias formas nos últimos anos;

À minha Esposa, por todo o Amor com que me presenteia, pedindo desde já a sua compreensão por não lhe ter dado a devida atenção e carinho no período em que decorreu este trabalho;

Aos meus avós paternos (que bom foi ter onde obter apoio e comida caseira ao jantar). Aos maternos, o meu avô pelo que não consigo expressar em palavras e a minha avó, que não está comigo. Dizem que a saudade se dilui com o tempo, é mentira;

Por último, aos meus pais e ao meu irmão, pela dedicação, determinação, força e coragem que sempre me transmitiram, não só durante a realização deste trabalho, mas em todas as etapas da minha vida, nomeadamente na persecução da minha formação académica.

Resumo

A massificação da utilização das tecnologias de informação e da Internet para os mais variados fins, e nas mais diversas áreas, criou problemas de gestão das infra-estruturas de informática, ímpares até ao momento. A gestão de redes informáticas converteu-se num factor vital para uma rede a operar de forma eficiente, produtiva e lucrativa. No entanto, a maioria dos sistemas são baseados no *Simple Network Management Protocol* (SNMP), assente no modelo cliente-servidor e com um paradigma centralizado. Desta forma subsiste sempre um servidor central que colecta e analisa dados provenientes dos diferentes elementos dispersos pela rede. Sendo que os dados de gestão estão armazenados em bases de dados de gestão ou *Management Information Bases* (MIB's) localizadas nos diversos elementos da rede. O actual modelo de gestão baseado no SNMP não tem conseguido dar a resposta exigida. Pelo que, existe a necessidade de se estudar e utilizar novos paradigmas de maneira a que se possa encontrar uma nova abordagem capaz de aumentar a fiabilidade e a performance da gestão de redes. Neste trabalho pretende-se discutir os problemas existentes na abordagem tradicional de gestão de redes, procurando demonstrar a utilidade e as vantagens da utilização de uma abordagem baseada em Agentes móveis. Paralelamente, propõe-se uma arquitectura baseada em Agentes móveis para um sistema de gestão a utilizar num caso real.

Palavras Chave

Gestão de Redes, *Simple Network Management Protocol*, Informação de Gestão, Elementos de Rede, Agente e Agentes móveis.

Abstract

Network management is a vital factor in a successfully, effectively and also profitably way of operating a network. Nowadays businesses become progressively more dependent on networking services and keeping those services running constantly several hours a day or sometimes even during the entire day seven days a week, becomes synonymous with keeping the business running. A large number of network management systems as represented by Simple Network Management Protocol (SNMP) are based on the client-server centralized paradigm, where a central station collects and analyses data retrieved from physically distributed network elements. In those systems, management data is stored in a standard structure maintained on the elements to be managed, such as Management Information Base (MIB) Objects Tree in SNMP. There is a daemon agent at each network element which periodically fetches and returns management data in response to inquiries from the manager. With the rapid development of network scale, the current SNMP network management model has not been able to meet requirements. Therefore, a new kind of management system and paradigm must be provided to enhance the reliability and performance of the network management. This thesis discusses problems existing in traditional network management and in order to demonstrate the applicability of Mobile Agents to network management, this work will investigate the related problems and will suggest and implement an architecture for an Agent-Based Network Management System based on Mobile Agents for a real network.

Keywords

Network management, Simple Network Management Protocol, Management Information, Network Elements, Agents and Mobile Agents.

Índice

Índice de Figuras

Índice de Tabelas

Notação e Glossário

SNMP	*Simple Network Management Protocol*
MIB	*Management Information Base*
RMON	*Remote Monitoring*
GECAD	Grupo de Investigação em Engenharia do Conhecimento e Apoio à Decisão
OSI	*Open Systems Interconnection*
FCAPS	*Fault, Configuration, Accounting, Performance and Security*
TMN	*Telecommunications Management Network*
SMI	*Structure of Management Information*
OID	*Object Identifier*
CMIP	*Common Management Information Protocol*
CORBA	*Common Object Request Broker Architecture*
API	*Aplication Programming Interface*
SMA	Sistema Multi-Agente
OAA	*Open Agent Architecture*
FDA	*Fault Detection Agent*
IA	*Intervention Agent*

1 Introdução

1.1 Enquadramento

Actualmente os sistemas de gestão de redes baseados em SNMP assentam no modelo cliente-servidor, onde existe uma aplicação gestora central responsável por obter informação de gestão dos elementos de rede que se pretendem gerir. Os sistemas de gestão baseados neste paradigma necessitam que seja transferida uma grande quantidade de informação entre os vários elementos de rede e a aplicação gestora. O que conduz a um consumo de banda elevado e pode provocar o chamado *bottleneck* junto da aplicação de gestão [1-4].

A solução para este problema passa por abordagens descentralizadas de gestão, tais como *Remote Monitoring* (RMON) e a gestão por delegação (MbD). Uma terceira abordagem, a utilização de Agentes móveis para a distribuição e delegação de tarefas de gestão foi também alvo de investigação nas últimas décadas [4-10].

Para alcançarmos uma gestão mais eficiente, pode-se então usar uma abordagem baseada em Agentes móveis capazes de efectuar uma gestão descentralizada. O Agente móvel pode ser usado para consultar e recolher dados das MIB's, de modo a fazer a monitorização do fluxo da rede num ambiente distribuído. São atribuídas tarefas a um determinado Agente (que podem ser tarefas de monitorização ou mesmo tarefas de configuração) e este pode ser enviado a um determinado *host* para levar a cabo as funções de gestão, reportando posteriormente os resultados da sua actividade à aplicação gestora. Basicamente a solução assenta em distribuir o mecanismo de gestão de maneira a ultrapassar as limitações de uma arquitectura centralizada. Na área da gestão de redes tem vindo a ser feita muita investigação com o propósito de avaliar a aplicabilidade das tecnologias baseadas em Agentes, assim como se tem assistido ao crescente estudo de várias plataformas para o desenvolvimento de sistemas Multi-Agente para serem utilizados na gestão de redes [3, 4, 11-20].

Um das abordagens possíveis é a utilização de Agentes móveis. Estes podem ser utilizados com a finalidade de aliviar a sobrecarga na banda de rede, através da utilização da delegação de autoridade, por parte da entidade responsável pela gestão para o Agente móvel em questão. Estes são flexíveis, podem ser criados à medida dos requisitos do utilizador e lançados a partir de um gestor. Podem visitar cada elemento de uma rede de acordo com uma tabela contendo o itinerário, fazer a computação e se necessário analisar, processar e comprimir os dados de gestão localmente retornando-os posteriormente para o gestor de rede [3, 4, 21-23].

Esta abordagem permite, mover a inteligência para os nós onde os dados de gestão residem, sendo que as próprias decisões de gestão podem ser tomadas localmente, evitando a transferência de grandes quantidades de dados desde os nós remotos até ao gestor central [4, 24].

1.2 Objectivos e Principais Contribuições

A gestão de redes tem colocado aos especialistas na área vários desafios, nomeadamente a gestão eficiente dos recursos disponibilizados com vista à manutenção e gestão da rede. Para tal, este trabalho tem como objectivo principal o estudo da aplicabilidade dos Agentes móveis na gestão de redes.

É proposta uma arquitectura com base no paradigma dos Agentes móveis para o cenário real da rede do Grupo de Investigação em Engenharia do Conhecimento e Apoio à Decisão (GECAD). A arquitectura proposta procura ultrapassar os problemas existentes nas arquitecturas centralizadas.

Paralelamente, demonstra-se que novos paradigmas podem coexistir com o paradigma tradicional, permitindo obter uma nova abordagem para a gestão de redes.

1.3 Organização da Dissertação

Nesta secção é apresentado o guia de leitura do presente documento. Esta dissertação é composta por seis capítulos, sendo cada capítulo descrito resumidamente de seguida:

- Neste primeiro capítulo, *Introdução*, é efectuado o enquadramento do tema deste trabalho, Agentes móveis para gerir um sistema de rede distribuído, sendo também apontados os principais objectivos do trabalho, bem como as suas contribuições;
- No segundo capítulo, *Introdução à Gestão de Redes*, pretende-se introduzir alguns conceitos fundamentais da gestão de redes. Neste capítulo é apresentada a importância da gestão de redes, as suas Áreas Funcionais, o Modelo Arquitectural, o Modelo de Informação e o Modelo Relacional;
- No terceiro capítulo, *Gestão Baseada em Agentes*, inicia-se uma primeira abordagem às definições de Agente, tipos e classificação, Sistemas Multi-Agente, assim como o conceito de Agentes móveis. Procura-se descrever características e vantagens sobre a arquitectura tradicional, linguagens de programação, entre outros;
- O quarto capítulo, *Cenário Real – Rede do GECAD* apresenta o cenário real em que este trabalho se enquadra, desde toda a componente física da rede, passando por cablagem estruturada, equipamentos activos e principais servidores, até aos serviços e processos indispensáveis ao bom funcionamento da rede. É também objecto de detalhe a solução de gestão actual, suas características e requisitos;

- No quinto capítulo, *Arquitectura Proposta*, apresenta-se a arquitectura proposta, baseada no paradigma dos Agentes móveis. É dada uma visão geral da arquitectura, o tipo de Agentes móveis utilizados bem como a comunicação entre eles. Uma segunda parte deste capitulo está voltada para a implementação da solução, descrevendo que plataformas e API's foram usadas, bem como outro tipo de especificações;
- As conclusões deste trabalho são apresentadas no sexto e último capítulo, *Conclusões*. Neste capítulo são descritas as principais limitações deste trabalho, sendo apresentadas direcções para trabalho futuro relacionado com a gestão baseada em Agentes móveis.

2 Introdução à Gestão de Redes

2.1 Introdução e Definições

Gestão de redes refere-se a todas as actividades que permitem manter uma rede em funcionamento, em conjunto com toda a tecnologia requerida para suportar essas mesmas actividades. Uma grande porção das tarefas exigidas para atingirmos os objectivos de uma gestão correcta passa simplesmente pela monitorização da rede, sabendo retirar conclusões de forma a perceber tudo aquilo que acontece. Como é natural existem outros aspectos não menos importantes, que serão detalhados posteriormente.

Existe então um conjunto de métodos, procedimentos e ferramentas que têm como objectivo a operação, administração, manutenção e o fornecimento de serviços em sistemas de rede. A Operação consiste em manter a rede (e serviços a esta afectos) a funcionar sem problemas. Inclui a monitorização para detectar eventuais falhas de forma expedita, de preferência antes do utilizador final ser afectado. A Administração envolve todos os recursos da rede e a forma como estão atribuídos, contendo tarefas que permitam manter todos os recursos sob controlo. Quanto à Manutenção, o nome já indica de forma implícita alguns dos conceitos que estão subjacentes, isto é, possibilita a realização de reparações, actualizações (sejam de *hardware* ou *software*) e ainda todas as operações necessárias a que a rede funcione correctamente quando sofre alterações significativas.

As próximas três figuras (figura 1, 2 e 3) ilustram alguns dos conceitos base necessários para compreender melhor a gestão de uma rede [2, 25, 26].

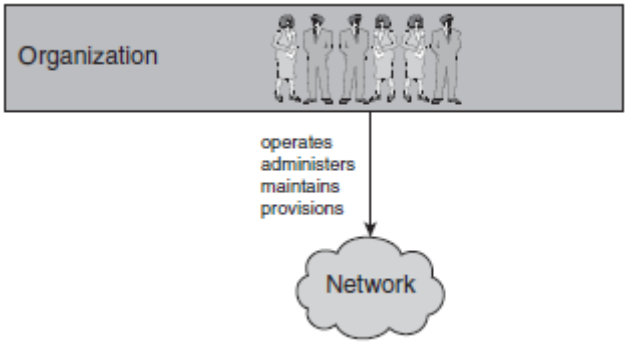


Figura 1- Uma Rede e a sua Organização (retirada de [2])

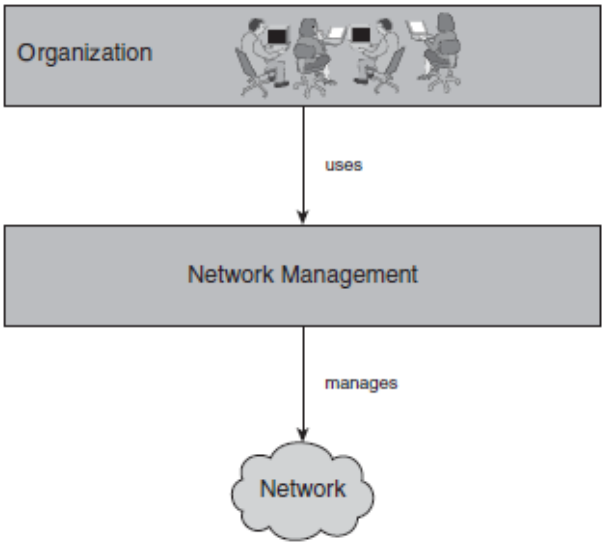


Figura 2- O papel da Gestão de Redes (retirada de [2])

Na figura 3 podemos ver todos os sistemas e aplicações que suportam as actividades e procedimentos referidos anteriormente.

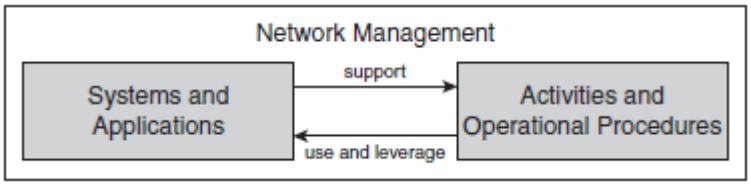


Figura 3 – Componentes da Gestão de Redes (retirada de [2])

Na proxima figura, figura 4, está patente o conceito de multidisciplinaridade, ou seja, para que possamos fornecer um determinado serviço podemos ter vários componentes envolvidos da rede empenhados nessa função. É comum fazer-se uma separação em termos de gestão de acordo com o ilustrado na figura [2].

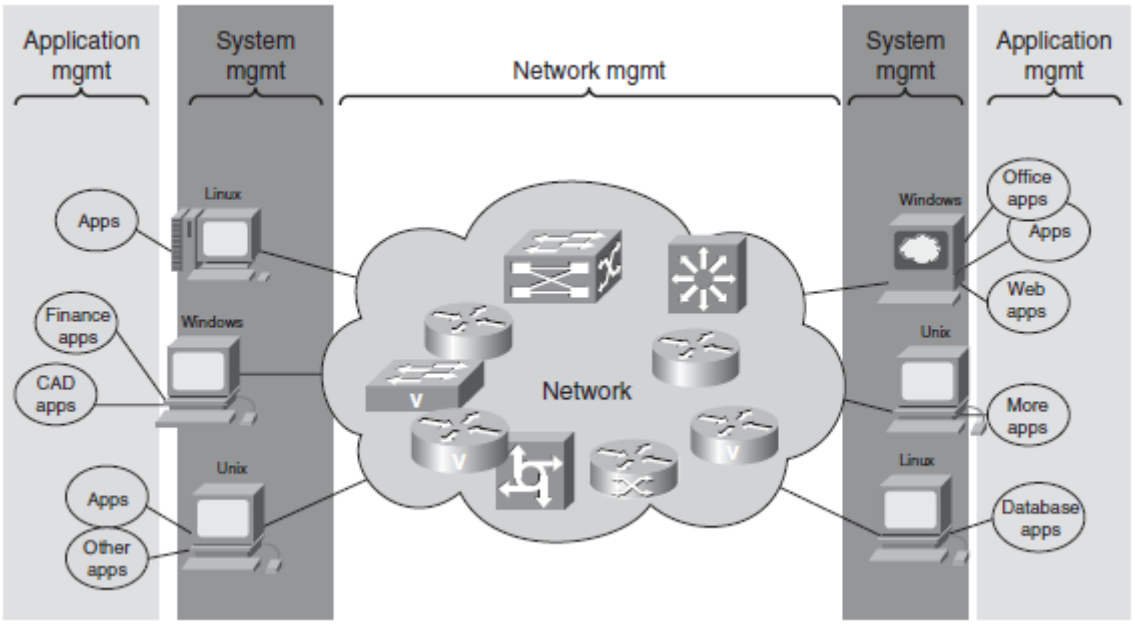


Figura 4 – Gestão da Rede, Sistemas e aplicações (retirada de [2])

2.2 Importância da Gestão de Redes

Nos dias de hoje a informação tem-se revelado como um elemento fulcral para as empresas, de forma a obterem vantagens competitivas sobre terceiros. Esta competitividade é a base para a sobrevivência no mercado, ou, para aqueles que sabem tirar partido das vantagens, um destacamento da concorrência. Pelo que, as empresas têm investido de forma agressiva nas tecnologias de informação, diminuindo distâncias, tempos de resposta entre transacções com parceiros de negócio e aumentando a capacidade de tomar decisões de forma mais célere e segura [1].

Esta correlação entre os negócios e as redes torna a sua gestão crítica, uma vez que temos toda uma envolvente de recursos que são vitais para os utilizadores, exigindo hoje em dia uma permanente disponibilidade, 24 horas por dia, 7 dias por semana. Para além de todos aqueles serviços mais comuns (DNS, DHCP, WEBSERVER, VPN), existem várias aplicações, Sistemas Operativos e arquitecturas, sendo que sem uma gestão forte, o negócio pode ficar comprometido. É então necessário obter uma boa performance em toda a rede, segurança e grande disponibilidade, contra a qual encontramos quase sempre uma redução de custos exigida pelas organizações. Ainda assim, e ao contrário do que existe (ou não) na maioria das empresas, uma simples má configuração, instalação de aplicações ou até dispositivos (ex. *router wireless*) não autorizados pode comprometer toda a infra-estrutura, caso não exista uma monitorização apropriada [1, 2].

A produtividade nas empresas aumenta quando as aplicações estão disponíveis e em funcionamento

com um tempo de resposta ideal. Para ser credível, a quantificação destes benefícios tem de ser conservadora e realística. Uma forma de fazer isto é olhar para as últimas dez a quinze interrupções que afectaram vários utilizadores e, caso a caso, fazer a pergunta: "Se eu tivesse o meu Sistema de monitorização, esta interrupção poderia ter sido evitada ou resolvida de forma mais célere?"
Para qualquer uma destas interrupções ou falhas, uma forma de quantificar o impacto que existiu na produtividade da empresa pode ser calculado através da seguinte fórmula [2]:

- **Mão-de-obra perdida** (Duração da interrupção **x** Número de pessoas afectadas) **x Percentagem da Produtividade Afectada = Impacto na Produtividade da Empresa**

A percentagem da produtividade afectada não vai ser 100% porque na maioria dos casos existem tarefas de substituição que são postas em curso para evitar que determinado recurso esteja parado.

Alternativamente, podemos usar a chamada regra 50-50 para calcular o impacto na produtividade da empresa. Esta regra teoriza que um sistema de monitorização bem planeado e administrado resultará no afastamento de 50% das interrupções do ano transacto e reduzirá o tempo destas no ano corrente em 50% [2].

O que acontecerá analisando o impacto discutido nos parágrafos anteriores na produtividade das próprias pessoas ligadas aos departamentos de tecnologias de informação? Os resultados que normalmente surgem têm a ver com o facto de uma maior quantidade de trabalho ser realizada com o mesmo número de recursos humanos disponíveis, por oposição à necessidade de reduzir esse mesmo número. Naturalmente, surge a indigência de justificar o investimento que é inevitável no aumento de trabalho realizado pelo grupo ligado à administração de sistemas informáticos. Esse tipo de cálculos pode ser realizado usando métodos parecidos com os apresentados no cálculo que o impacto das interrupções tinha na produtividade da empresa em geral. Basicamente, é necessário analisar junto da equipa de Tecnologias de Informação (IT) em cada elemento, qual a quantidade de trabalho feita de forma reactiva vs planeada durante as interrupções, permitindo a inferência do nível de melhoramento de produtividade se existisse um sistema de monitorização.

Falhas acontecem sendo indispensável detectar, diagnosticar e corrigir. O nível de serviços que foi previamente garantido aos utilizadores finais deve ser monitorizado e assegurado (por exemplo uma determinada largura de banda). Toda a panóplia de serviços, sua implementação e disponibilização aos utilizadores finais deve ser gerida, incluindo quando estes são solicitados [2].

Depois das considerações feitas até aqui podemos ter a tentação de refutar toda esta importância subjacente à gestão de redes, uma vez que implementá-la não é fácil. É necessário saber o que existe e em que local, como tudo está interligado e conhecer fisicamente, e ao pormenor, cada um dos elementos de rede, e seus segmentos. No entanto, se um problema surge num dispositivo

importante para o bom funcionamento da rede, torna-se indispensável o administrador ser avisado, e se possível serem efectuadas acções interventivas de forma a corrigir a falha antes de esta afectar os utilizadores. A simples ligação de um dispositivo com um IP que entre em conflito com outra máquina pode trazer problemas, assim como, a disponibilização involuntária de um serviço similar a um já existente na organização (ex. DHCP) [1, 2, 6].

Grandes tempos de resposta na implementação de novos serviços, assim como a manutenção dos existentes, com uma qualidade acima da média são factores que podem traduzir vantagens competitivas. Um dos objectivos da gestão de redes é concretizar as operações de forma eficiente, tornando o trabalho dos administradores mais produtivo. Se existirem ferramentas de teste e diagnóstico consegue-se identificar e resolver problemas de forma mais rápida, para além de que se for criado, um ou mais processos autónomos, para as dificuldades mais rotineiras, libertamos os administradores de rede para outras funções. É sempre essencial analisar a performance da rede, assim como os potenciais estrangulamentos, para que munidos desse tipo de informação seja possível alocar recursos onde eles são mais necessários, minimizando o investimento na infra-estrutura. Outra das vantagens com a implementação deste tipo de ferramentas reside no facto de não ser obrigatório estarmos perante um *expert* na área para fazer uma primeira despistagem e até solucionar um eventual problema [2, 25].

Outro aspecto operacional que não está relacionado directamente com os custos mas que é importante é a qualidade. Devemos considerar a qualidade das comunicações, assim como a dos serviços que são fornecidos. São incluídas propriedades como a quantidade de banda disponível e o atraso em tempos de resposta, que por sua vez é um factor muito importante na forma como os utilizadores respondem utilizando serviços pela rede. Um utilizador acaba sempre por se interrogar: "Poderei confiar neste serviço? É de qualidade elevada?"; "Está disponível na maior parte do tempo?"; "Existem cópias de segurança dos meus dados?"; "É fiável o local onde estão armazenados esses dados?".

As vantagens da gestão de redes não estão só ligadas ao custo e à qualidade. Com a solução adequada podemos obter oportunidades de mercado que não surgiriam se a gestão não existisse. Por exemplo, se um cliente tiver a capacidade de seguir online os seus pagamentos e até configurar determinados recursos, estamos perante uma mais-valia para a empresa. Mas para isso ser possível é necessário a existência de um sistema de gestão da rede eficiente que faça de sustentáculo para esse tipo de capacidades, ou seja, um serviço que poderia não ser equacionado como plausível, passa a sê-lo devido à existência de um sistema de gestão eficiente [2].

Ser proactivo é vital, gerindo uma rede em vez de reagir a determinados acontecimentos depois de estes estarem consumados. É certo um permanente crescimento das redes actuais, a sua dispersão

geográfica, em conjunto com a diversidade de tecnologias existentes. Se na presença de uma solução de gestão encontramos dificuldades, tais como garantir a resposta rápida e eficaz, flexibilidade, escalabilidade, e coerência na gestão independentemente do fabricante dos equipamentos, podemos imaginar o que acontece caso não se disponha de uma aplicação de gestão [6].

2.3 Áreas Funcionais da Gestão de Redes

O modelo OSI (*Open Systems Interconnection*) define cinco áreas funcionais ao qual vulgarmente se dá o acrónimo de FCAPS (*Fault, Configuration, Accounting, Performance* e *Security*). É apresentado de seguida um diagrama sobre estas áreas, onde cada uma se propõe a uma determinada função exemplo[3, 22, 27-29]:

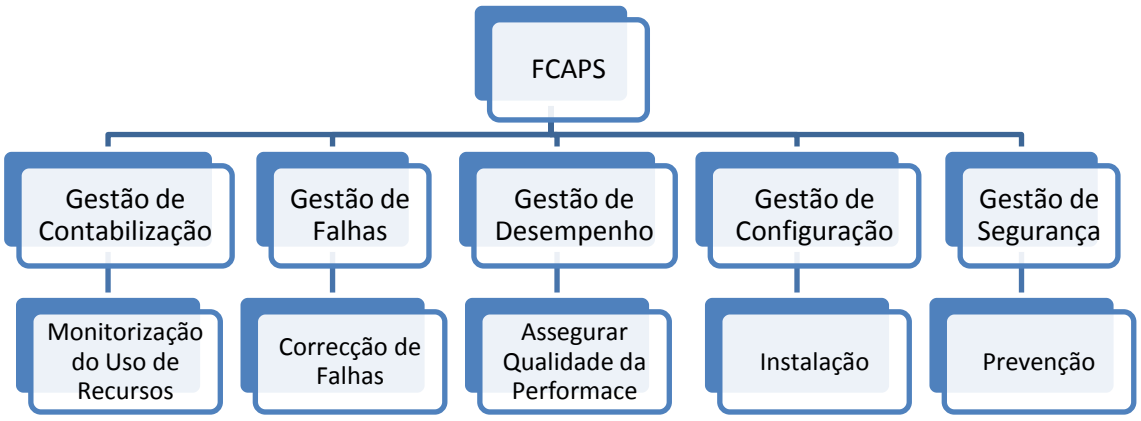


Figura 5 – Modelo OSI: Cinco Áreas Funcionais

A **gestão de falhas** abrange a detecção, isolamento e correcção de anomalias causadas no ambiente em questão. As falhas podem ser provocadas por falha material ou por algum problema de software mostrando-se como eventos particulares que devem ser diagnosticados, tendo em vista o funcionamento contínuo da rede e respectivos serviços. As falhas podem ser persistentes ou transitórias, sendo que certos eventos como pacotes descartados ou *frames* erróneas são comuns, mesmo numa rede robusta. Este tipo de erros deve ser registado e monitorizado mas não requerem qualquer tipo de acção interventiva, ao contrário das falhas persistentes. Algumas falhas persistentes podem ser corrigidas de forma automática, ao nível dos equipamentos de rede, apesar de muitas carecerem da intervenção ao nível do administrador. Sendo assim, qualquer entidade de gestão de falhas ao nível de equipamento de rede deve ter algum tipo de suporte para detecção das falhas persistentes e correcção automática destas. Por fim, todas as funções necessárias para

executar a manutenção do registo de erros, reagir perante os mesmos, assim como testes de diagnóstico e posteriores correcções encontram-se também no grupo de gestão de falhas.

A **gestão de configuração** corresponde ao conjunto de funções de gestão que permite obter controlo, supervisão e colecta de informações sobre mudanças de configuração em cada um dos objectos geridos. Este tipo de configuração é usado para monitorizar os recursos existentes (inclusive os que se encontram em falha) e seus respectivos detalhes.

A gestão de contabilização permite identificar e atribuir custos à utilização de recursos. Desta forma podemos saber quantos e quais os recursos a ser consumidos e a que custo. A gestão de contabilização permite também estabelecer limites e associar planos de tarifas aos recursos (ex. criar controlos de espaço em disco através de quotas).

A gestão de desempenho permite avaliar o comportamento dos recursos e objectos geridos de forma a garantir a qualidade de serviço acordada com os utilizadores finais. A eficiência das actividades de comunicação e a existência de funções para recolher informação estatística, manter e examinar registos do estado do sistema são também propriedades deste grupo. Todo este conjunto de características permitem ainda reunir toda a informação sob a forma de um histórico, de maneira a que se possa analisar e planear o sistema, optimizando-o através de tarefas de gestão de desempenho.

A gestão de segurança deve minimizar o acesso não autorizado ou acidental a funções de controlo na rede, assegurando dessa forma uso legítimo, confidencialidade, integridade dos dados, entre outros. Consoante os direitos de acesso deve personalizar os elementos de rede a nível de funcionalidades, retirando-as ou disponibilizando-as. Tem como objectivo suportar a aplicação de políticas de segurança através de meios tais como a criação, destruição e controlo de serviços e mecanismos de segurança, a distribuição de informação relevante de segurança, e o relato de eventos relacionados com esta.

Apresenta-se de seguida uma tabela, tabela 1, com um resumo das diferentes funções que cada uma das áreas funcionais deve cobrir para uma gestão da rede eficiente [3, 22, 27-29]:

Gestão de Falhas	*Gestão de Configuração*	*Gestão de Contabilização*	*Gestão de Performance*	*Gestão de Segurança*
Detecção de Falhas	Inicialização de Recursos	Monitorização de Serviços	Garantir Nível de performance	Acesso a Recursos de forma Selectiva
Correcção de Falhas	Configuração Remota	Monitorização do Uso de Recursos	Recolha de Dados	*Logs* de Acesso
Isolamento de Falhas	Inicialização de Tarefas e sua Monitorização	Custo por Serviço	Geração de Relatórios	Privacidade dos Dados
Recuperação da Rede	Suporte para Instalação Automática de Software	Combinação de Custos para Vários Recursos	Análise de Dados de Performance	Verificação de Permissões de Acesso
Geração, Análise e Tratamento de Alarmes	Gestão de Inventário	Utilização de Quotas	Planeamento/Ajuste do Sistema	Geração de Alarmes
Testes de Diagnóstico	Mudança de Configurações	Verificação de Adulteração de Tarifas	Manutenção de *Logs* e Dados Históricos	Tratamento de Falhas de Segurança
Análise, Tratamento e *Logging de Erros*	Término de Recursos	Auditorias	Análise de Taxas de Erros	-

Tabela 1 – Áreas Funcionais: Resumo

2.4 Modelo Arquitectural da Gestão de Redes

O modelo arquitectural descreve a estrutura geral de todas as entidades envolvidas em tarefas de gestão, as suas interfaces e os métodos de comunicação. O exemplo mais referido deste modelo é a arquitectura TMN (*Telecommunications Management Network*), que prevê uma rede de gestão conceptual, interligada com a rede de telecomunicações em diversos pontos para enviar e receber dela informação, bem como controlar a sua operação [1, 6, 22].

Apesar da existência de vários elementos na gestão de redes existem dois que são fundamentais: o gestor de redes e o objecto gerido. O gestor de redes corresponde ao software e hardware responsável por captar a informação de gestão e processá-la. Hoje em dia, encontramos imensas aplicações (componente de software) deste tipo, capazes de realizar a função de gestor, muitas delas completamente *freeware*, excedendo em muitas situações a versatilidade de algumas das soluções pagas disponíveis no mercado [1, 6, 22].

Quanto ao objecto gerido, corresponde a uma abstracção de um determinado recurso a ser gerido. A palavra ideal é abstracção uma vez que o recurso pode estar relacionado com as comunicações, processamento de dados, recurso físico (ex: espaço em disco), ou outros. Não devemos entender

por recurso uma determinada máquina, sendo que é muito difícil tratar os dispositivos geridos directamente em termos de gestão. Para gerir um determinado recurso num determinado dispositivo necessitamos de aceder à MIB, presente nesse mesmo dispositivo, com a ajuda de um Agente que comunica e interpreta a informação de gestão necessária. Existindo comunicação entre o gestor e o objecto gerido, é obrigatória a presença de um protocolo de gestão, responsável por definir o conjunto de acções de comunicação entre um e outro, para além de garantir que consigamos obter o significado e interpretação de cada mensagem enviada/recebida [1, 6, 22].

Resumindo, a arquitectura de gestão tem de conseguir espelhar a forma como os objectos geridos e o gestor interagem e comunicam, entre outros, para fins de gestão.

2.4.1 Arquitecturas Centralizadas

Este tipo de abordagem é a mais comum nos vários modelos de gestão, entre os quais se destacam o OSI, TMN, SNMPv1, SNMPv2 e SNMPv3. Quando a arquitectura é centralizada, figura 6, surge como o resultado da existência de uma topologia Gestor-Agente, em que se prevê a subsistência de apenas uma entidade responsável pela gestão em conjunto com os restantes objectos geridos [3, 6, 22].

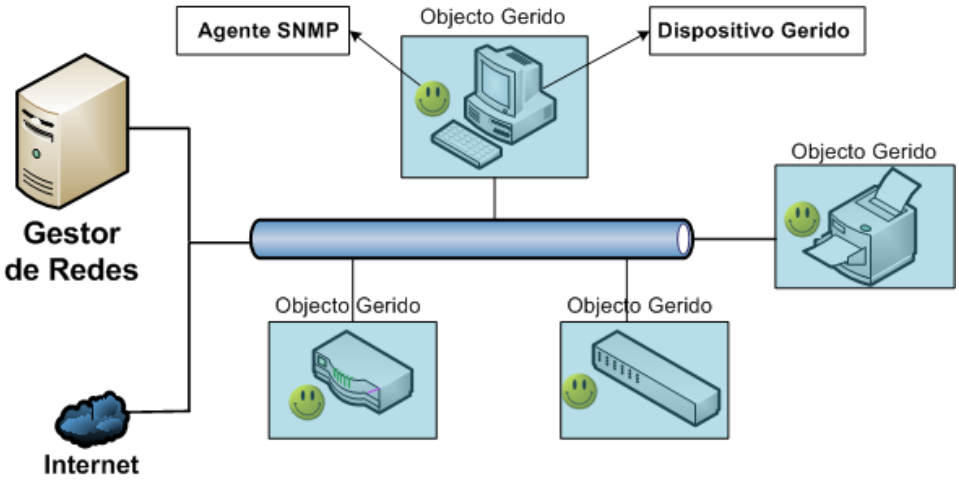


Figura 6 – Exemplo de Arquitectura Centralizada (baseada em [6])

Cada um dos objectos geridos necessita de interagir com o gestor de rede, sendo que essa ligação é feita através do uso de Agentes capazes de consultar e compreender a respectiva MIB do objecto gerido. Estes Agentes encontram-se articulados com cada um dos objectos geridos e têm como função, agora de forma mais específica, permitir a gestão dos dispositivos e/ou serviços com eles relacionados. Os Agentes são conhecidos por serem "estúpidos", uma vez que não possuem qualquer tipo de inteligência, limitando-se a recolher dados de gestão e em última instância a efectuar algum tipo de configuração a mando do gestor de redes. Este último centraliza em si toda a

informação de gestão, assim como todas as instruções, comandos a executar, capacidade de processamento e funcionalidades [3, 6, 22].

O facto de esta arquitectura surgir como a mais corrente tem a ver com a grande vantagem de ser relativamente simples de implementar. Já as desvantagens prendem-se com a não escalabilidade, flexibilidade limitada e dificuldades em obter com facilidade várias possibilidades de reconfiguração. Ainda dentro das desvantagens encontramos uma muito importante e muitas vezes menosprezada: a grande quantidade de tráfego de gestão que atravessa a rede, podendo em alguns casos causar um *bottleneck* quer no segmento em que se encontra o gestor de redes, quer nos elementos de rede com que este interage.

2.4.2 Arquitecturas Descentralizadas

A gestão centralizada não é a mais adequada aos dias de hoje, uma vez que tal como foi referido anteriormente, as redes atingem com relativa facilidade uma complexidade demasiado grande com centenas ou mesmo milhares de dispositivos. A gestão distribuída ganha assim um maior relevo à medida que a necessidade de escalabilidade vai aumentando [3, 6, 7, 22, 30].

Numa etapa inicial, ou se preferirmos na maioria dos casos, quando este tipo de gestão é introduzida, parte das tarefas são delegadas pelos diversos Agentes espalhados pela rede, tornando-os "inteligentes". Quer o grau de inteligência, quer o grau de distribuição, dependem muito daquilo que as pessoas responsáveis pela administração da rede pretendem ao ver a gestão acompanhada das funcionalidades como um todo. A arquitectura distribuída permite assim manipular toda aquela grande quantidade de informação de gestão de uma forma mais eficiente, sendo que os dados gerados dessa manipulação e transferidos para a rede para efeitos de comunicação serão menores [3, 6, 7, 22, 30].

A escalabilidade, tolerância a falhas e a capacidade da delegação de tarefas nos Agentes permitem mover uma porção do processamento e inteligência para o local trazendo valor acrescentado. Sistemas exigentes com grande complexidade exigem uma elevada interacção, que esta abordagem proporciona, uma vez que estão mais perto da informação e resolvem o problema do *polling* (consultas aos Agentes), obrigando a que os cálculos sejam realizados localmente. A forma mais simples e conhecida de descentralização é a monitorização remota ministrada por sondas remotas RMON (*Remote Network Monitoring*). Estas sondas analisam o tráfego baseadas num conjunto de premissas (configuradas remotamente) num determinado segmento de rede, podendo realizar de forma autónoma e independente uma série de estudos e tomar acções interventivas. Este tipo de acções pode, por exemplo, ser um envio de notificações para o gestor de redes aquando da detecção de algum problema [3, 6, 7, 22, 30].

As arquitecturas distribuídas podem ser divididas em arquitecturas hierárquicas e arquitecturas cooperantes. Nas arquitecturas hierárquicas a delegação de tarefas pelos diferentes níveis é feita de forma vertical desde os níveis hierárquicos superiores até aos inferiores. Já nas cooperantes, essa delegação de tarefas é horizontal entre entidades pares encontrando-se no mesmo nível hierárquico. Uma arquitectura hierárquica de gestão, figura 7, pode ser representada como uma árvore, onde se encontram os Agentes no nível mais baixo e onde se verifica a existência da aplicação de gestão no nível de topo. Esta delega tarefas nos níveis inferiores, actuando apenas muito raramente de forma directa nos Agentes que se encontram nos níveis mais baixos. Os gestores de nível intermédio adquirem as aplicações que lhe foram delegadas, e é-lhes imputada a execução de tarefas de gestão sobre os níveis inferiores com o controlo do nível superior. Por último, os Agentes presentes no nível inferior apenas contêm os objectos geridos sobre os quais são executadas as tarefas de gestão [6, 22].

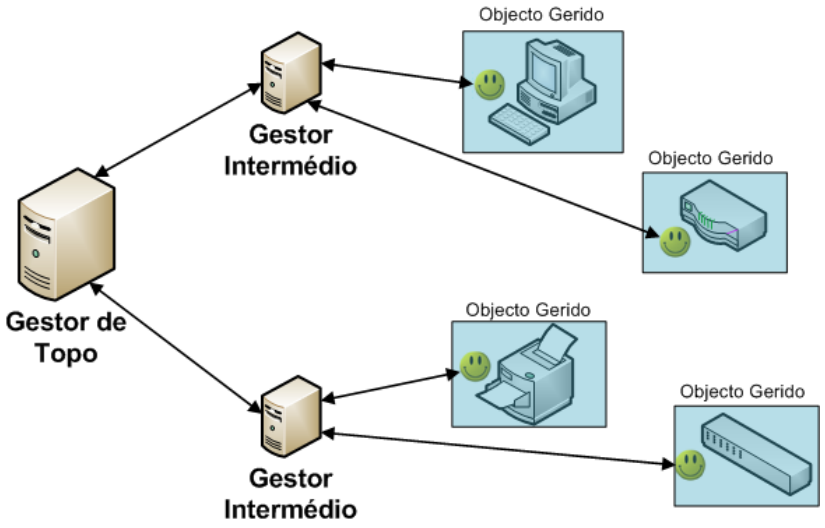


Figura 7 – Arquitectura Hierárquica (baseada em [6])

Nas arquitecturas de gestão cooperativa, os Agentes têm autonomia para cooperar uns com os outros trabalhando por objectivos de gestão. Um exemplo desta arquitectura está ilustrado na próxima figura, figura 8, em que o gestor de redes é responsável por lançar um programa na rede que percorre vários elementos, através de um itinerário previamente definido, recolhendo e processando a informação de gestão e regressando no fim ao ponto inicial, acompanhado dos resultados das suas acções. Esta abordagem é a sugerida e estudada neste trabalho. Encontramos o termo **Agente móvel**, ainda que a palavra **Agente**, neste caso não tem a conotação habitual, pois é um termo ligado à inteligência artificial [6, 22].

Figura 8 – Arquitectura com Agentes móveis (baseada em [6])

O gestor de topo é responsável por iniciar a criação do Agente móvel, lançando-o na rede com um determinado itinerário. O Agente móvel inicia a sua travessia, percorrendo os diversos elementos de rede, onde executará tarefas de gestão, sejam de verificação ou de configuração. É gerado um relatório com as intervenções efectuadas, apresentado aquando do regresso do Agente ao gestor de topo (em alguns casos poderá acabar a travessia sem que o gestor seja a última paragem), contendo toda a informação útil ao administrador de redes [6, 22].

2.5 Modelo de Informação de Gestão

Tal como referido anteriormente, existem recursos que são geridos, em que cada um deles possui uma representação lógica. O modelo de informação trata deste aspecto em particular, isto é, deve traçar pormenorizadamente a forma de aceder à informação de gestão, o seu significado e a maneira de descrever nova informação de gestão [1, 2, 6, 22, 26, 28, 31].

A MIB é um repositório conceptual de dados que contem uma perspectiva de gestão sobre o dispositivo que está a ser gerido. Os dados contidos nesse repositório constituem a gestão da informação. As operações de gestão são dirigidas contra esta "base de dados", por exemplo as portas de rede de um *switch* podem ser representadas numa tabela, em que cada porta tem uma entrada correspondente na tabela. As colunas no quadro conceptual contêm atributos que correspondem a propriedades reais do dispositivo. Exemplos de tais atributos são o tipo de protocolo de comunicação suportado pela porta e o número de pacotes transmitidos [1, 2, 6, 22, 26, 28, 31].

A MIB não deve ser confundida com uma base de dados real, sendo que é uma forma de vermos o dispositivo e não uma base de dados onde informação se encontra puramente armazenada, correspondendo a um género de *proxy* para o elemento de rede que se encontra a ser gerido. O *RFC*

- *Request For Comment 1066* expôs a primeira versão da MIB, a MIB I. Este *standard* define a base de informação fundamental para monitorizar e controlar redes informáticas baseadas na pilha TCP/IP. A evolução surgiu com uma segunda MIB, bastante usada neste trabalho: o RFC 1213 propôs a MIB II que facilita informações gerais de gestão acerca de um determinado equipamento gerido [1, 2, 6, 22, 26, 28, 31].

A estrutura de informação de gestão, ou SMI como é mais conhecida (*Structure of Management Information*), especifica todas as regras que definem e permitem construir/alterar uma MIB. Esta estrutura inclui os tipos de dados, a sua representação, as relações entre os objectos geridos e a descrição destes objectos está de acordo com a SMI Internet (*RFC 1155* e *RFC 2578*), empregando um subconjunto da sintaxe de ASN.1 (*Abstract Syntax Notation One*) [X.208]. Nesta estrutura os objectos estão organizados numa hierarquia em árvore, sendo reconhecidos por um OID (*Object Identifier*). Um OID é constituído por uma sequência de números inteiros traçada a partir dos nós das árvores e separada por pontos. A próxima figura, figura 9, ilustra a árvore da MIB II, em que o OID internet corresponde a *iso.org.dod.internet* ou *1.3.6.1* [1, 2, 6, 22, 26, 28, 31, 32].

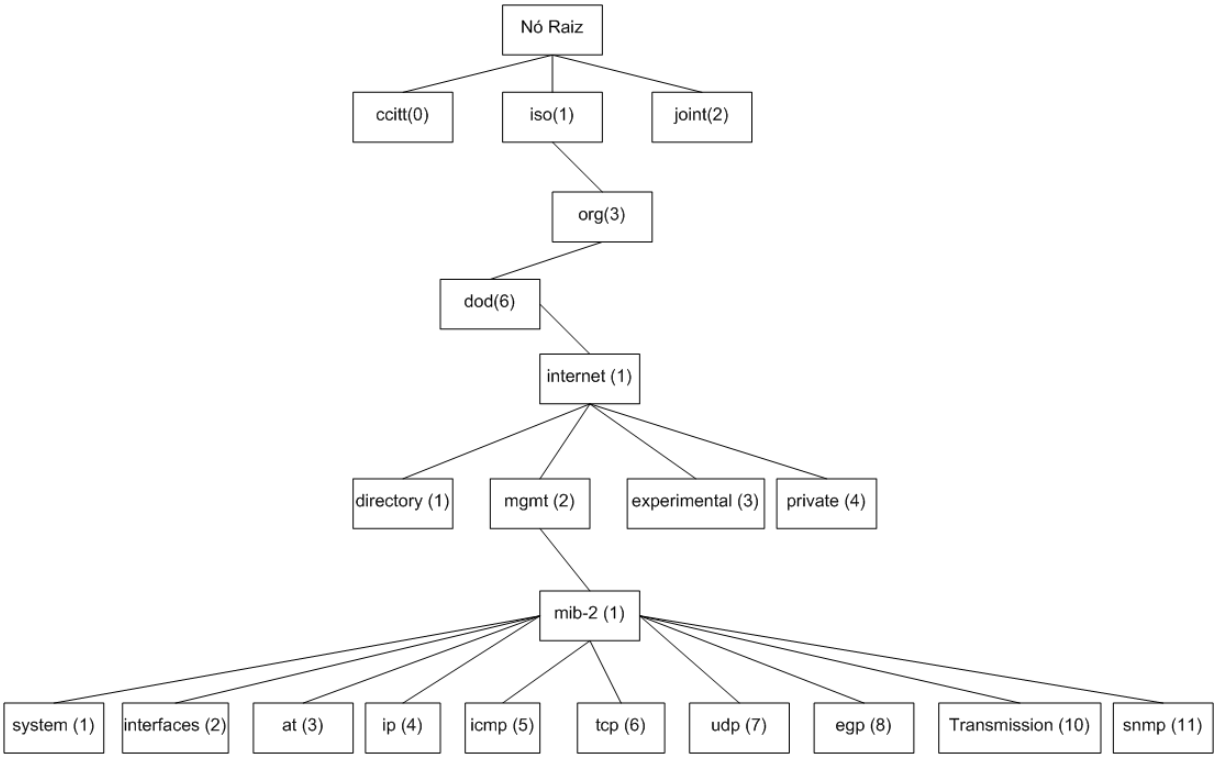


Figura 9 – MIB II

Existem várias ferramentas, nomeadamente *freeware*, que permitem consultar de forma mais intuitiva as MIBs. Exemplo de uma ferramenta é o software "iReasoning Mib Browser", trata-se de um instrumento muito útil para efectuar consultas às bases de dados de informação de gestão. A

próxima figura, figura 10, ilustra o interface gráfico aquando de uma operação sobre uma MIB em específico [33, 34].

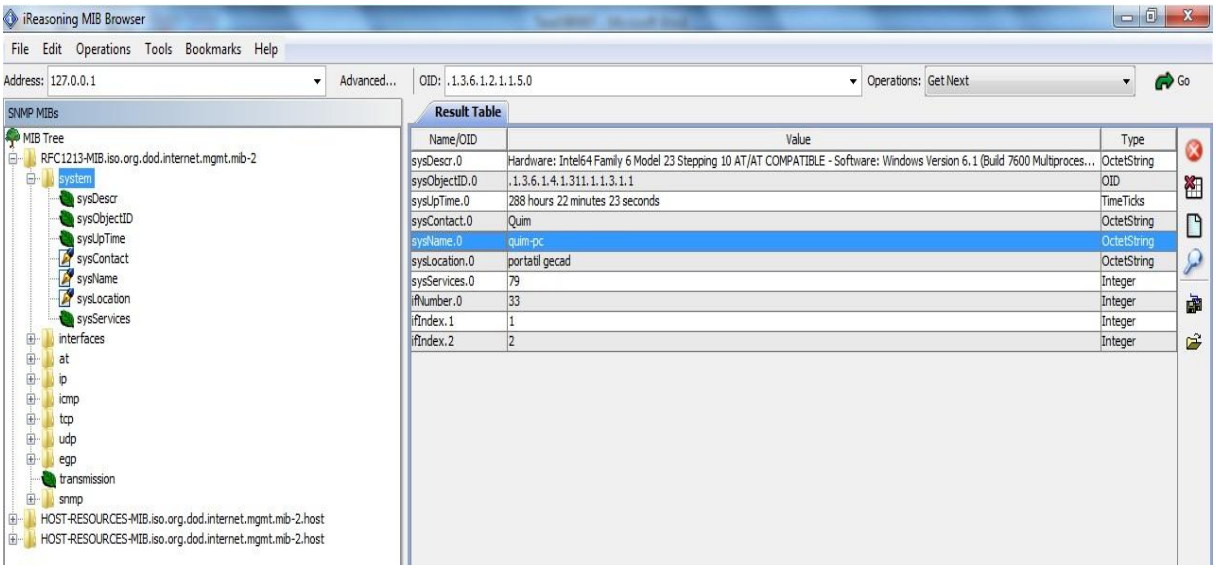


Figura 10 – iReasoning Mib Browser

Da observação da figura anterior, podemos ver o endereço IP da máquina, neste caso o endereço de *loopback* 127.0.0.1. Ao seu lado localiza-se a OID da MIB sobre a qual queremos efectuar uma operação de gestão. Subsistem ainda duas zonas principais: a árvore da MIB à esquerda, onde podemos explorar todas as OID's existentes e ao centro a tabela com os resultados das operações de gestão, nomeadamente as consultas.

Name/OID	Value	Type
sysDescr.0	Hardware: Intel64 Family 6 Model 23 Stepping 10 AT/AT COMPATIBLE - Software: Windows Version 6.1 (Build 7600 Multiproces...	OctetString
sysObjectID.0	.1.3.6.1.4.1.311.1.1.3.1.1	OID
sysUpTime.0	288 hours 22 minutes 23 seconds	TimeTicks
sysContact.0	Quim	OctetString
sysName.0	quim-pc	OctetString
sysLocation.0	portatil gecad	OctetString
sysServices.0	79	Integer
ifNumber.0	33	Integer
ifIndex.1	1	Integer
ifIndex.2	2	Integer

Figura 11 – Exemplo Tabela Resultado Mib Browser

A figura anterior, figura 11, ilustra um exemplo de uma consulta de uma série de OID's, através de uma operação de gestão apelidada de "GET-BULK". Esta operação devolveu a tabela acima representada, sendo que permitiu verificar os seguintes parâmetros:

- ✓ A primeira coluna corresponde à OID em causa, que contém também uma semântica própria;
- ✓ A segunda coluna ilustra o valor contido na OID (ex: o PC neste caso está ligado à 288 horas, 22 minutos e 23 segundos);

- ✓ A terceira e última coluna equivale ao tipo de dados do valor da OID respectiva (vemos inteiros, *strings* específicas deste modelo de gestão, entre outros).

2.6 Modelo Relacional de Gestão

O modelo relacional de gestão descreve a forma como o sistema gestor e o sistema gerido comunicam entre si. De seguida apresentam-se os protocolos SNMP em redes IP com base no modelo Internet e o CMIP (*Common Management Information Protocol*) baseado no modelo OSI para redes de telecomunicações, uma vez que são os mais utilizados. Existem outros modelos relacionais como as tecnologias de objectos distribuídos que serão abordados nesta secção [1, 3, 6, 28].

2.6.1 Protocolos de Gestão OSI e Internet

Tal como foi referido, o CMIP foi desenvolvido sobre a pilha protocolar OSI, definindo como a informação de gestão de redes é transmitida entre as aplicações de gestão e os Agentes. Devemos ter em atenção que o CMIP não descreve funcionalidades do sistema de gestão de redes, mas apenas a estrutura e forma como a troca de informação de gestão é realizada e como deve ser utilizada e interpretada. Apresentam-se algumas das vantagens deste protocolo de gestão sobre o SNMP [1, 3, 6, 28]:

- ✓ Variáveis CMIP transportam informação, mas também podem ser usadas para realizar tarefas;
- ✓ CMIP é um protocolo que apresenta boas capacidades a nível de segurança;
- ✓ Permite realizar mais acções num único pedido do que o SNMP;
- ✓ Tem uma maior competência na criação de relatórios sobre situações anómalas na rede.

Existe a especificação do CMIP para redes IP, sendo designada de **CMOT** (*CMIP over TCP*). Apresentam-se de seguida as desvantagens deste protocolo, que demonstram o porquê de não conseguir superar o SNMP até aos dias de hoje em termos de redes IP [1, 3, 6, 28]:

- ✓ Requer uma grande quantidade de recursos do sistema apenas para a gestão, condicionando e pondo em causa o normal funcionamento do sistema quando esses recursos não abundam, ou quando precisam de ser direccionados para outros fins;
- ✓ Para implementar e gerir este tipo de protocolo são necessários recursos humanos especializados, sendo difícil a sua programação devido à sua complexidade;
- ✓ Pouco funcional quando comparado com o SNMP devido à falta de adaptação ao ambiente de elevada heterogeneidade existente nas redes IP.

Passamos ao estudo do protocolo SNMP, sendo necessário referir que este teve um papel preponderante neste trabalho e na construção da ferramenta de gestão de redes com Agentes móveis (Capítulo 5). Este protocolo foi desenvolvido nos finais dos anos 80, encontrando-se na sua terceira versão (SNMPv3) e tendo como objectivo efectuar toda a comunicação de informação de gestão entre o Gestor e os Agentes.

O SNMP é um protocolo de gestão definido ao nível da aplicação e usado para obter informações de Agentes espalhados numa rede baseada na pilha TCP/IP. Esses dados são obtidos através de solicitações (*polling*), utilizando os serviços de protocolo UDP (*User Datagram Protocol*) para enviar e receber mensagens pela rede. Existem vantagens em usar UDP, tais como o facto de ser mais simples implementar Agentes SNMP em equipamentos de rede menos sofisticados (não é preciso manter ligações e gerir retransmissões). Em casos de problemas de rede, a probabilidade de entrega em teoria é maior, havendo no entanto desvantagens: retransmissões têm de ser geridas pelas aplicação e a quantidade de informação que se pretende enviar por comandos é limitada, principalmente quando comparado com o CMIP. A próxima figura, figura 12, ilustra a arquitectura tradicional de gestão baseada no protocolo SNMP [1, 3, 6, 22, 28, 35].

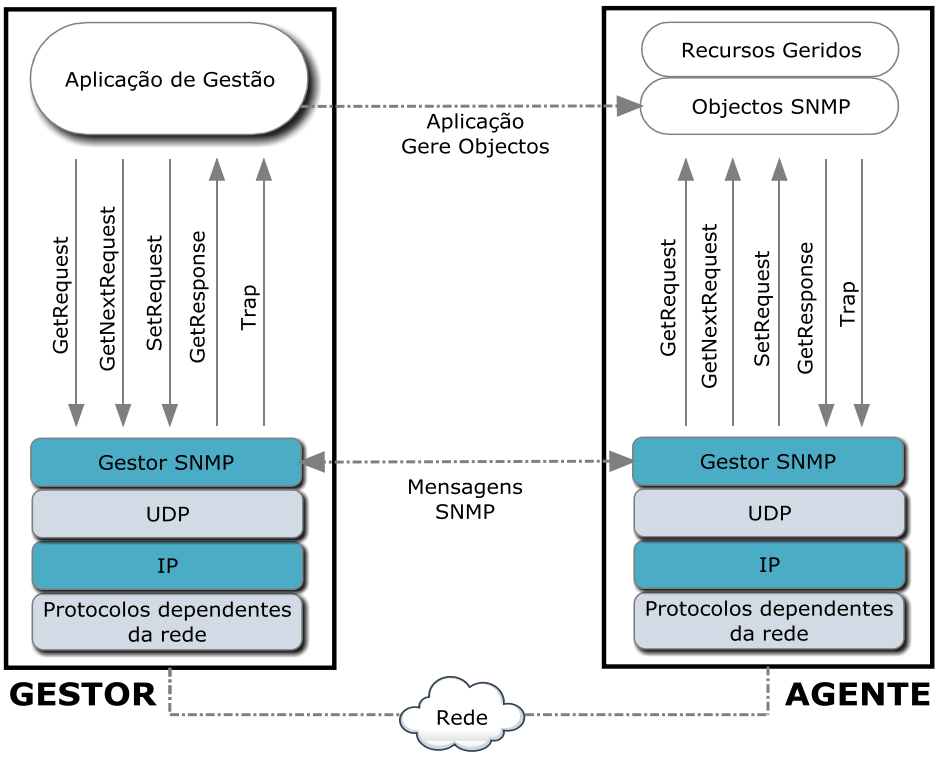


Figura 12 – Arquitectura SNMP Tradicional (baseado em [35])

Analisando a figura anterior encontramos todos os elementos mais comuns a uma rede IP com gestão através de mensagens protocolares SNMP. De um lado o Gestor, do qual faz parte a

respectiva aplicação de gestão e do outro o Agente. De uma forma natural deparamo-nos com as camadas que compõem o modelo, sendo que quer no Gestor, quer no Agente, existem uma série de operações de gestão utilizadas assiduamente por ambos. Como o diagrama indica, existem operações protocolares síncronas desencadeadas pelo Gestor (o chamado *polling*) que são as "**GetRequest**", "**GetNextRequest**" e "**SetRequest**". No SNMPv2 e v3 existe já a operação protocolar "**GetBulkRequest**", sendo que todas estas mensagens são respondidas com "**GetResponse**" por parte do Agente. No caso dos GET's deve-se especificar a OID das respectivas instâncias da MIB: o comando "GET-REQUEST (sysDescr, null)" devolve um erro indicando que a OID não existe; a forma certa de o executar seria "GET-REQUEST (sysDescr.0, null)" do qual se receberia um [GET-RESPONSE (sysDescr.0, "servidorHttp")]. O comando "GETNEXT-REQUEST" devolve a próxima instância (ordenadas por OID), sendo muito útil para percorrer tabelas e fazer travessias completas de uma MIB [1, 3, 6, 22, 28, 35].

Existem ainda as operações protocolares assíncronas, visíveis na última imagem através do comando "**Trap**", geradas pelo Agente em situações excepcionais pelo porto UDP 162. Este envio surge quando o Agente está com problemas e por iniciativa própria (assincronamente) contacta o gestor. Na construção, preparação, envio e recepção de uma mensagem SNMP são executados os seguintes procedimentos [1, 3, 6, 22, 28, 35]:

1. Construir a PDU (*Protocol Data Unit*) utilizando a notação ASN.1. A PDU pode ser um "GetRequest", "SetRequest", etc;
2. Enviá-la opcionalmente para um serviço de autenticação em conjunto com os endereços de origem e destino e com o nome da comunidade;
3. Construir a mensagem SNMP, adicionando à PDU (autenticada ou não) o campo da versão e o nome da comunidade;
4. Codificar a PDU, utilizando o esquema de codificação BER (*Basic Encoding Rules*), para que seja enviada para o serviço de transporte;
5. Aquando da recepção, construir a mensagem SNMP em ASN.1 a partir de BER;
6. Validar a versão do protocolo SNMP utilizado, retirando a PDU SNMP da mensagem;
7. Enviá-la opcionalmente para um serviço de autenticação em conjunto com os endereços de origem e destino e com o nome da comunidade;
    a. Se a autenticação falhar a PDU é ignorada e é gerado um "Trap";
    b. Se a autenticação não falhar, enviar a PDU em ASN.1.
8. Analisar os campos da PDU, verificando se o seu preenchimento está correcto;
9. Por último, executar a acção pedida, tendo em conta o perfil de comunidade identificado.

No protocolo SNMP a segurança sempre foi e continua a ser um alvo de grande discussão, isto porque a primeira versão continha permissões baseadas apenas numa palavra passe (*comunity string*), enviada a cada mensagem de uma forma imprudente (*clear text*). A versão SNMPv3 foi a única que conseguiu introduzir melhorias no que toca a segurança, aproveitando o trabalho desenvolvido em versões anteriores como a v2, v2*, v2c e v2u, tornando este tipo de mensagens

encriptadas. Com estes aperfeiçoamentos consegue-se combater na terceira versão alguns ataques de segurança, tais como: modificação da informação, espionagem, alteração da sequência das mensagens, uma vez que o protocolo não é orientado à ligação, e ataques de *mascarade*, em que uma entidade realiza operações para as quais não tem permissão. Permanecem ainda alguns ataques que não estão contemplados, nomeadamente os que são feitos com vista à negação de serviço impedindo a comunicação entre Agente e Gestor.

A figura seguinte, figura 13, ilustra os princípios fundamentais da arquitectura, abordando-a numa perspectiva de entidade e módulos. Estes dois conceitos devem ser diferenciados e explicados: a arquitectura de gestão SNMP é constituída por um conjunto de entidades SNMP distribuídas, que cooperam entre si, podendo funcionar como Gestor, Agente ou ambos; cada uma das entidades é estabelecida tendo por base módulos, que interagem entre si para providenciar serviços [1, 3, 6, 22, 28, 35, 36].

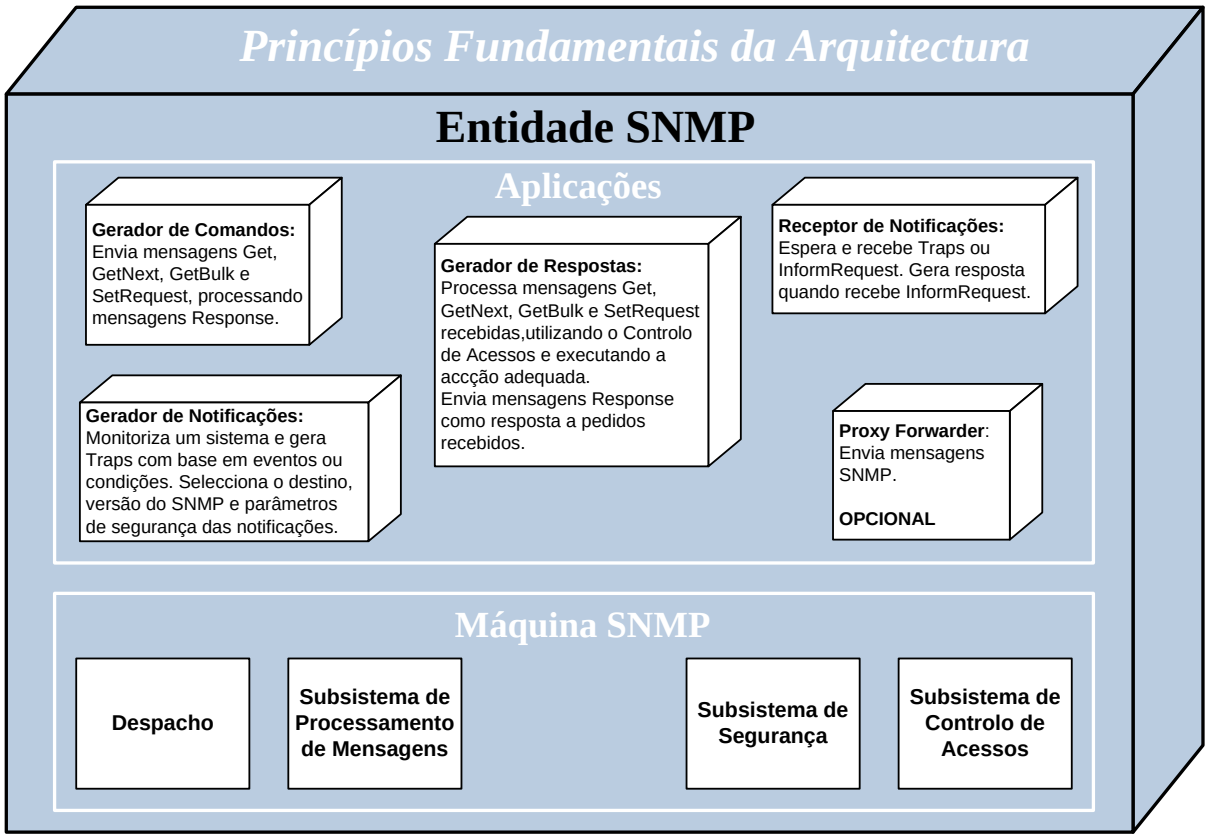


Figura 13 – Entidade SNMP (baseado em [36])

O Despacho (*Dispatcher*) recebe e envia mensagens para o respectivo modelo de processamento de mensagens, sendo que após a passagem destas por estes dois módulos do motor SNMP, o subsistema de segurança encarrega-se de autenticar as mensagens decifrando e codificando ainda as mensagens privadas. O subsistema de controlo de acesso determina se as permissões de acesso aos

objectos geridos são válidas. Como podemos ver na figura anterior, existem cinco tipos de aplicações SNMP:

- ✓ Gerador de comandos que monitorizam e manipulam dados de gestão;
- ✓ Gerador de respostas que facilitam o acesso à informação de gestão;
- ✓ Gerador de notificações iniciam mensagens assíncronas;
- ✓ Receptor de notificações processam as mensagens assíncronas;
- ✓ *Proxy Forwarders* encaminham mensagens entre entidades.

2.6.2 Gestão Usando Objectos Distribuídos

A gestão é uma aplicação distribuída, como tantas outras, podendo assim utilizar os mecanismos de suporte à comunicação entre sistemas distribuídos já existentes. As tecnologias de objectos distribuídos permitem criar arquitecturas cliente-servidor flexíveis, em que a lógica do negócio e os dados são encapsulados em objectos, que se podem localizar em qualquer parte do sistema distribuído. Uma forma alternativa de realizar a comunicação entre o sistema gestor e os objectos geridos é através da invocação remota de métodos de objectos. A arquitectura CORBA (*Common Object Request Broker Architecture*) é um exemplo de uma arquitectura que se alicerça numa tecnologia deste tipo. Como concorrente do CORBA, a Microsoft lançou outra tecnologia de objectos distribuídos, denominada de *Commom Object Model* (COM), que se tornou no *standard* de facto [12, 25, 26, 37].

O objectivo do CORBA passa por usar a própria tecnologia distribuída, retirando um aproveitamento desse facto para gerir os ambientes distribuídos. Esta arquitectura possibilita acabar com a tradicional abordagem que utiliza um modelo para "operação" da rede e outro para gestão da mesma, sendo que todo este interesse assenta no facto desta tecnologia se situar no nível da aplicação e ser facilmente usada como tecnologia de interoperabilidade. É notório que é mais fácil construir e integrar sistemas de gestão ao nível API (*Aplication Programming Interface*) do que ao nível protocolar [12, 25, 26, 37].

A arquitectura do CORBA passa por conter alguns elementos fundamentais: os objectos (Agentes, Gestores e Recursos de rede), serviços específicos de gestão dos objectos e mecanismos de invocação remota desses mesmos serviços. Trata-se de uma gestão distribuída, mais simples e com uma Framework com suporte em tempo real de gestão e comunicação entre objectos distribuídos. Aparece por último o ORB (*Object Request Broker*), que define a interacção entre as aplicações clientes que requerem serviços e as aplicações servidoras, que os executam. Os métodos evocados pelos clientes nos servidores são-no feitos de forma transparente, em que o ORB que recebe o pedido é responsável por localizar o objecto, transferir os parâmetros, invocar o método e transferir

o resultado. Os clientes têm apenas noção da existência de um componente, que é a interface que o objecto do servidor define, não sabendo a localização do objecto, o Sistema Operativo em que o objecto é executado nem a forma como o servidor implementa o mesmo objecto [12, 25, 26, 37].

Pode-se referir as seguintes vantagens na arquitectura CORBA [12, 25, 26, 37]:

- ✓ Uma só arquitectura de gestão e comunicação;
- ✓ Suporte inerente de objectos distribuídos;
- ✓ Esconde os detalhes da comunicação entre objectos (transparência);
- ✓ Interface entre IDL (*Interface Definition Language*) e diferentes linguagens de programação;
- ✓ Coordenação entre os Agentes é parte integrante da arquitectura CORBA.

Como qualquer outra arquitectura possui desvantagens [12, 25, 26, 37]:

- ✓ Não existe informação de gestão definida para a arquitectura CORBA;
- ✓ É uma arquitectura complexa (gestão internet);
- ✓ Agentes de gestão baseados em CORBA são difíceis de encontrar no Mercado;
- ✓ Necessidade de serviços adicionais para a gestão de rede;
- ✓ Suporte tempo-real é dedicado e pesado.

No entanto, teve um impacto importante mas pouco expressivo enquanto "arquitectura completa" para gestão de sistemas distribuídos, ao contrário do que conseguiu junto dos sistemas de telecomunicações (principalmente por meio do consórcio TINA – *Telecommunications Information Networking Architecture*), onde teve relevância. Continua a ser a tecnologia mais relevante como mecanismo de interoperabilidade na integração de sistemas heterogéneos, apesar do peso crescente da aplicação de tecnologias como Java RMI e DCOM à gestão de redes [12, 25, 26, 37].

2.6.3 Outros Protocolos de Gestão

Apesar de tudo o que foi referido até aqui, devemos sublinhar que existem mais protocolos de gestão de redes, entre os quais encontramos por exemplo o bem conhecido WBEM (*Web-Based Enterprise Management*) promovido por várias empresas importantes como é o caso da Microsoft. Este protocolo possui um modelo de informação de gestão que é o CIM (*Common Information Model*), permitindo o envio de mensagens CIM-XML sobre HTTP ou HTTPS. Estas mensagens são geradas com a ajuda de um modelo de codificação baseado em XML (*Extensible Markup Language*), como o nome atrás indica e enviadas para um servidor (WBEM *Server*), capaz de as descodificar e compreender os pedidos e acções a executar [6, 37-39].

Outro tipo de gestão muito citado e usado é a gestão por políticas, em que o gestor da rede define políticas de gestão de alto nível. Um exemplo poderá ser o facto de apenas um utilizador ter acesso a um determinado servidor, ou o servidor "xy" ter prioridade no acesso à internet em relação às outras máquinas. Estas políticas são sucessivamente convertidas em políticas/directivas de mais baixo nível, até ao ponto em que se traduzem em configurações específicas dos equipamentos da rede. A alteração das políticas de alto nível traduz-se automaticamente na alteração das configurações de baixo nível [6, 37-39].

Ainda que não seja simples traduzir directamente "políticas de alto nível" em configurações de equipamentos, já é viável traduzir "políticas de nível intermédio". Torna-se possível, por exemplo, definir uma configuração abstracta de um conjunto de *routers*, que depois é "traduzida" automaticamente na configuração específica de cada um dos modelos dos equipamentos instalados. Dessa forma, quando um *router* for substituído, o novo vai obter automaticamente uma configuração equivalente, mesmo que seja de outra marca/modelo. O IETF (*Internet Engineering Task Force*) definiu recentemente protocolos para definir e "transportar" políticas (modelo COPS - *Common Open Policy Service*), sendo que muitos fabricantes já fornecem equipamentos "COPS-ENABLED" [6, 37-39].

2.7 Conclusões

A análise e resolução dos problemas de gestão podem ter várias aproximações que vão desde o tratamento dos problemas isoladamente até uma visão integrada da infra-estrutura, actuando sobre ela como um todo.

Nesta última abordagem, as arquitecturas de gestão têm um papel fundamental, constituindo a base para o desenvolvimento de sistemas de gestão abertos, aplicáveis nos ambientes heterogéneos existentes nas actuais infra-estruturas informáticas.

3 Gestão Baseada em Agentes

3.1 Definição de Agente

Um Agente normalmente é definido como uma entidade real ou virtual, que emerge num ambiente onde pode tomar decisões de forma autónoma. É capaz de perceber esse mesmo ambiente, comunica-se com outros Agentes e consegue ganhar autonomia como consequência da sua observação, conhecimento adquirido e interacções com os restantes Agentes da comunidade. O Agente pode ser visto como uma entidade cognitiva, activa e autónoma que consegue operar sobre o Mundo e Agentes que o rodeiam, sem algo ou alguém para o aconselhar e controlar. Para além destas características, podemos completar com uma capacidade sensorial e competências para agir e reagir sobre o meio ambiente [40-43].

Algumas definições são mais centradas no Agente vendo-o como uma entidade individual, mais próxima de algo ou alguém (por exemplo, um Agente que efectua buscas ou compras na Internet em nosso nome). Esta abordagem tem vindo a ser associada ao conceito de Agente inteligente. Outras definições são mais orientadas para as competências dos Agentes sociais. Nestes casos, o foco não é o Agente em si, mas a comunidade dos Agentes e portanto, as características mais valorizadas são a argumentação, negociação e capacidade de resolução de conflitos. Pode-se dizer que esta abordagem é centrada no conceito de Sistemas Multi-Agente (*Multi-Agent System)* [40-43].

3.2 Tipos e Classificação de Agentes

Os Agentes podem ser classificados de acordo com um conjunto de características possíveis. É difícil encontrar um Agente que apresente todas as características que irão ser apresentadas de seguida, mas é claro que algumas são fundamentais na definição de conceito de Agente.

A próxima tabela, tabela 2, apresenta todas as características principais que podem ser usadas para descrever um Agente [40-43].

No capítulo 5 serão apresentadas e discutidas as características dos Agentes que possuem especial interesse para o contexto deste trabalho, gestão de redes baseadas em Agentes móveis.

Propriedades	*Significado*
Capacidade Sensorial	Possui "sensores" de forma a reunir informação sobre o ambiente
Reactividade	Sente e age, reagindo às constantes mudanças do ambiente
Autonomia	Decide e controla as suas próprias acções
Pro-Actividade	É orientado a objectivos, indo além do simples reagir ao ambiente
Persistência	Existe durante longos períodos de tempo
Capacidades Sociais	Comunica e coopera com outros Agentes ou mesmo pessoas, concorre, negoceia, etc
Aprendizagem	Capaz de mudar o comportamento baseando-se em experiências anteriores
Mobilidade	Capaz de mover-se de um computador para outro
Flexibilidade	As suas tarefas não precisam de estar obrigatoriamente pré-determinadas
Agilidade	Capacidade de rapidamente tirar proveito de novas oportunidades
Carácter	Comportamento emocional e personalidade credível
Inteligência	Capacidade de raciocinar de forma autónoma, para planear as suas acções, corrigir os seus erros, para reagir a situações inesperadas, adaptar e aprender

Tabela 2 – Classificação de Agente

3.3 Sistemas Multi-Agente

Um Sistema Multi-Agente (SMA) é um sistema computacional em que dois ou mais Agentes interagem e/ou laboram em conjunto, de forma a executar determinadas tarefas ou alcançar um conjunto de objectivos definidos previamente. Estes sistemas são no fundo sociedades de Agentes, sendo que o verdadeiro desafio está em conseguir coordenar esses Agentes, ou seja, gerir as interacções e as dependências das suas actividades. Estes tipos de sistemas são naturalmente distribuídos, o que obriga a que haja uma grande preocupação com a coordenação [40-43].

No que diz respeito a este aspecto, diversas metodologias de coordenação foram propostas por diferentes autores, dividindo-se em dois grandes grupos: metodologias para domínios com Agentes competitivos, absorvidos com o seu bem próprio, e metodologias para domínios cooperativos, em que os Agentes têm uma preocupação especial pelo bem do conjunto. Quando nos deparamos com um domínio em que actuam Agentes competitivos, a coordenação por negociação é aquela mais estudada, em que é notória a importância que os mercados electrónicos e os leilões assumem. Já no segundo caso as metodologias estudadas apontam para a construção de equipas de Agentes e neste contexto, assumem particular relevância aquelas que permitem definir uma organização estrutural da sociedade de Agentes, a definição de papéis e tarefas, a alocação destas aos diversos Agentes, entre outros, de forma a criar um "ambiente" cooperativo [40-43].

Para que um determinado Agente possa actuar como parte do sistema, é fundamental a existência de uma infra-estrutura que permita a comunicação e/ou interacção entre os Agentes que compõe o SMA [40-43].

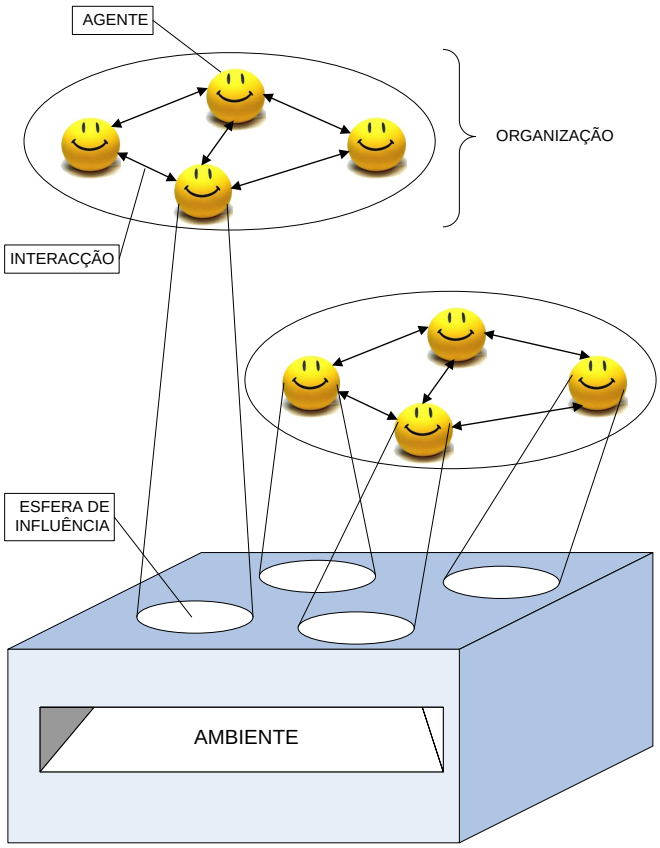


Figura 14 – Sistema Multi-Agente (baseada em [43])

Como podemos observar na figura anterior, figura 14, existem diversos Agentes na comunidade, cada um com diferentes capacidades de percepção e acção no mundo e a sua respectiva esfera de influência. Dessa forma, cada Agente consegue influenciar diferentes partes do ambiente, se bem que estas podem coincidir, dependendo das relações existentes entre os Agentes. Por exemplo, num SMA que tenha como objectivo gerar horários de vários departamentos numa Universidade, a esfera de influência vai recair sobre os horários dos docentes, turmas e salas desses departamentos. Com relativa facilidade pode dar-se o caso de dois Agentes interagirem entre si, no caso de efectuarem alocações em docentes, turmas ou salas que lhes sejam comuns [40-43].

Os SMA decorreram do campo originalmente denominado por Inteligência Artificial Distribuída, constituindo presentemente o âmago deste campo. Com esta circunstância compreendemos a motivação principal destes sistemas, que se encontra relacionada com o facto da maioria dos problemas encontrados serem inerentemente distribuídos de uma ou várias formas. Seguem-se outras motivações, também estas não menos importantes [40-43]:

- ✓ Problemas com dimensões demasiado elevadas para serem resolvidas apenas com a presença e utilização de um único Agente;

- ✓ Simplificar a interface que permite a cooperação entre homem e máquina, onde ambas as partes entram como Agentes do sistema;
- ✓ Problemas geográfica e/ou funcionalmente distribuídos ganham uma solução natural;
- ✓ Existem problemas que o são de forma ainda mais acentuada porque os recursos tais como informações, conhecimento, peritos, entre outros, se encontram dispersos. Esta abordagem permite soluções muito mais eficientes tirando partido da já existente distribuição de elementos e aproveitando-a, de forma a resolver um dado problema;
- ✓ A rentabilidade dos recursos existentes para resolver um determinado problema aumenta potencialmente, sendo este distribuído;
- ✓ Os SMA's são ideais quando existe uma distribuição espacial dos intervenientes de forma natural.

Num SMA defrontamo-nos com diversos Agentes, que podem estar a trabalhar em conjunto para atingir um objectivo geral, ou cada um com o seu propósito, sendo que no entanto de uma forma global estão relacionados. Assim, para atingirem o seu bem próprio, os Agentes necessitam de interagir uns com os outros, provocando o culminar de uma situação em que o bem do conjunto acaba também por ser privilegiado. Esta cooperação está subjacente nas aplicações de gestão de redes, uma vez que podemos ter um Agente responsável por verificar se os serviços baseados no protocolo http se encontram a funcionar em pleno em diversos servidores (tentando reanimar o serviço caso esteja em baixo), e um outro "apenas" com a função de analisar o espaço em disco. Esta é uma situação em que se tenta atingir um bem comum: a gestão de redes e a "saúde" dos servidores, mas em que cada um dos Agentes possui objectivos individuais. Inclusive, noutros casos a interacção entre estes dois Agentes poderia ser necessária, até o cruzamento dos dois relatórios gerados por eles, para que o administrador de rede ou um terceiro Agente possa analisar a situação de uma forma global e tentar intervir onde os outros dois executaram o seu plano [40-43].

3.3.1 Comunicação em SMA's

A comunicação em SMA's é geralmente diferente das mais utilizadas nas restantes áreas das Ciências da Computação, sendo de alto nível. Trata-se de linguagens próximas das utilizadas por humanos, devido à existência de todas as características já referidas, nomeadamente as suas habilidades sociais [40-43].

No âmbito deste trabalho um Agente é visto como uma entidade autónoma com habilidades de percepção, processamento e capacidade para actuar num determinado ambiente. Um Agente é capaz de abranger conhecimento e capacidade de raciocinar com base neste, definindo a cada instante qual a melhor acção a executar, de acordo com os seus objectivos. Tem de ser capaz de

comunicar com outros Agentes ou mesmo humanos, tirando partido da sua habilidade social para interagir no ambiente em causa. Por último, é possuidor de mobilidade assim como de inteligência [40-43].

Todas estas características e capacidades são muito vantajosas, mas inúteis se não existir a capacidade para comunicar. Normalmente, um Agente possui na sua arquitectura um módulo dedicado à comunicação. Este subdivide-se nas componentes de percepção e de acção onde a primeira é destinada à recepção de mensagens e segunda ao envio de mensagens. Para que o Agente possa fazer uso da sua inteligência, o módulo de comunicações está directamente ligado ao módulo da inteligência, permitindo ter acesso às mensagens recebidas, que juntamente com a capacidade de compreender o ambiente, permite definir quais as mensagens a enviar e acções a tomar [40-43].

3.3.2 Arquitecturas do Subsistema de Comunicação

O módulo de comunicação pode adoptar uma de duas arquitecturas existentes [40-43]:

- ✓ **Comunicação directa**, nesta arquitectura os Agentes comunicam entre si sem existir a necessidade de um terceiro elemento na cadeia. Antes de se iniciar a comunicação é necessário que os Agentes forneçam uns aos outros dados sobre aquilo que conseguem fazer (capacidades) e/ou respectivas necessidades. No entanto, esta abordagem obriga a que os Agentes tenham em simultâneo a responsabilidade de gerir as comunicações e gerir os seus objectivos. A figura seguinte, figura 15, ilustra este tipo de comunicação.

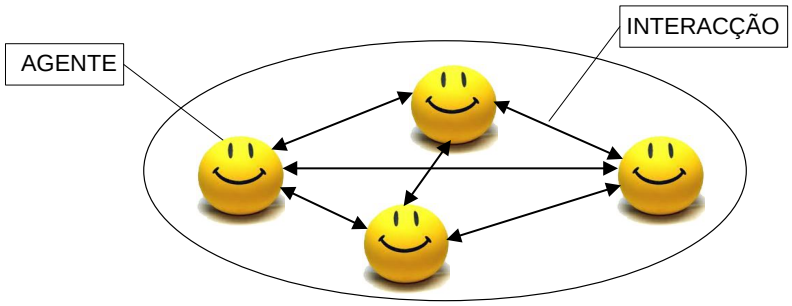


Figura 15 – Comunicação Directa (baseada em [43])

- ✓ **Comunicação assistida**: nesta arquitectura surge um Agente designado por "Agente facilitador", podendo existir mais do que um no mesmo SMA. Os restantes Agentes apoiam-se no Agente facilitador de forma a comunicarem entre si. Esta abordagem pode ainda funcionar de duas formas distintas: uma onde todas as comunicações são feitas via Agente facilitador, o Agente *i*, quando quer comunicar com o Agente *j*, envia a sua mensagem para o facilitador, que tratará de a encaminhar; ou então a variante onde todos

os Agentes se registam junto do Agente facilitador, especificando as suas necessidades e/ou capacidades, assim a comunicação apenas tem que passar pelo Agente facilitador numa fase inicial, uma vez que nas iterações seguintes esta já pode ser efectuada de forma directa (Agente-Agente).

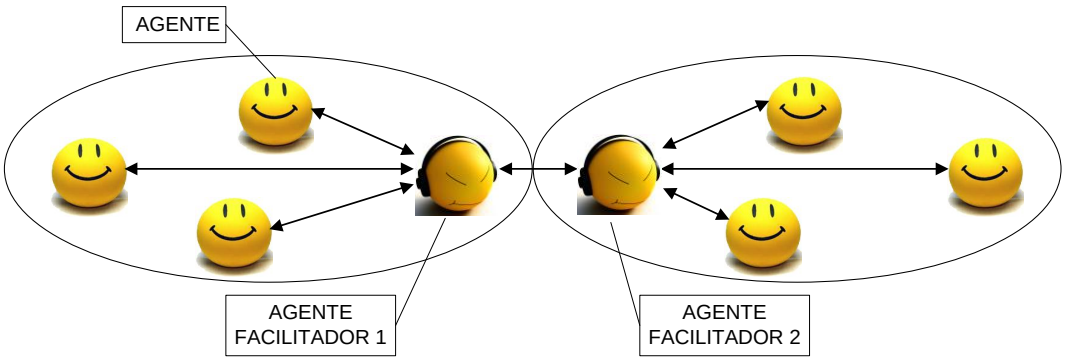


Figura 16 – Comunicação Assistida (baseada em [43])

A figura anterior, figura 16, ilustra um SMA com comunicação assistida onde existem dois Agentes facilitadores para dois domínios diferentes. Os Agentes comunicam com o respectivo Agente facilitador do seu domínio e estes são capazes de comunicar entre si (Facilitador-Facilitador). Um dos problemas existentes neste tipo de arquitectura é o número de mensagens dirigidas ao Agente facilitador, podendo resultar numa situação de "gargalo" mais conhecida no âmbito das redes de comunicação como *bottlenecks* nos facilitadores, impedindo estes de agir, e conduzindo à falha do sistema [40-43].

3.3.3 Considerações Finais em SMA's

A utilização de SMA para a resolução de problemas traz inúmeras vantagens: o paralelismo, permitindo configurar o sistema de forma a serem atribuídas diferentes tarefas a Agentes distintos, conduzindo a uma diminuição no tempo de execução das mesmas; Ao mesmo tempo, como não existe um ponto único de falha (são utilizados vários Agentes) no sistema, aumenta a fiabilidade; Por último destacamos dois aspectos muito importantes: a capacidade de simplificação, pois obtemos um desdobramento e divisão de tarefas em subtarefas e por último a redução do volume de comunicação, passando a ser realizada a alto nível devido ao facto do processamento estar junto da fonte de informação [40-43].

3.4 Agentes móveis

3.4.1 Introdução

Nos últimos anos temos vindo a assistir a um aumento exponencial do número de pessoas e dispositivos com acesso à internet, que por sua vez tem vindo a disponibilizar também uma crescente panóplia de serviços e de informação, conduzindo à necessidade de serem criadas aplicações com potencialidades de troca de informação entre redes heterogéneas de uma forma eficaz. Por esta razão foram criados vários paradigmas de desenvolvimento de *software* para computação distribuída entre os quais podemos destacar os paradigmas cliente-servidor, *Remote Procedure Calling* (RPC) e Agentes móveis.

3.4.2 Características dos Agentes móveis

Os Agentes móveis são processos de *software* construídos de forma a dotá-los de capacidade para se moverem autonomamente desde uma localização física na rede para uma outra localização, a qualquer momento. Quando nos referimos a "qualquer momento", devemos ter em conta algumas considerações, tais como a próxima localização e o estado de execução do Agente em conjunto com o código que é transportado por este. Desta forma, é possível recriar, após o "salto" final na rede, no sentido de alcançar o *host* destino, esse mesmo estado, e o Agente pode prosseguir com a sua execução. Este tipo de Agentes são uma forma específica dos paradigmas de *mobile code* e *software agents*, mas ao contrário do *Remote Evaluation* e *Code on Demand*, os Agentes móveis tomam a iniciativa e decidem migrar entre *hosts* da rede em qualquer altura da sua execução, agindo de forma proactiva [3, 12, 13, 44-46].

O Agente móvel realiza as suas tarefas quando e onde achar mais adequado, não necessitando de estar previamente instalado em todas as localizações/clientes. De outra forma os conceitos e vantagens de autonomia e mobilidade e a própria continuação da execução do Agente desperdiçavam-se [3, 12, 13, 44-46].

Recordando a nossa definição de Agentes móveis e considerando-os como um novo paradigma para sistemas distribuídos, completando as mais tradicionais técnicas, podemos identificar três características essenciais dos Agentes móveis [12, 22, 47-50]:

- ✓ Os Agentes móveis são normalmente utilizados em ambientes muito amplos, com redes heterogéneas e em que nenhuma suposição pode ser feita no que diz respeito à fiabilidade dos equipamentos, assim como da segurança das ligações de rede;

- ✓ A migração dos Agentes é feita para aceder a recursos disponíveis apenas noutros servidores da rede e não apenas para balanceamento de carga, como acontece nos sistemas de objectos móveis;
- ✓ Os Agentes móveis são capazes de migrar mais do que uma vez, sendo essa característica muitas vezes apelidada de capacidade *multi-hop*. Depois de um Agente visitar um determinado servidor, pode avançar no seu trajecto a caminho de outros servidores continuando o cumprimento das suas tarefas. Já nos paradigmas *Remote Evaluation e Code on Demand* código móvel é transferido apenas uma vez.

Abordando esta definição de Agente móvel de uma forma mais técnica, podemos afirmar que o termo código refere-se a um determinado tipo de representação da execução de programas de computadores. Nas linguagens script como PERL, poderia ser o código fonte; no caso de JAVA seria o código no formato "JAVA byte" a ser interpretado pela *JAVA Virtual Machine*. O termo "dados" corresponde a todas as variáveis do Agente (nas linguagens orientadas a objectos estamos a falar do conjunto de todos os atributos do respectivo objecto). Por fim, o termo "estado" refere-se a toda a informação sobre o estado da execução do Agente, tudo aquilo que permite guardar o que este estava a fazer num determinado ponto da sua execução de modo a poder, mais tarde, retomar esse estado [12, 14, 49, 51].

Não devemos confundir os atributos com o estado de execução, uma vez que um Agente possui uma agenda de tarefas a executar que vão alterar os seus atributos, tais como o local em que se encontra, etc; se usarmos apenas os seus atributos numa migração estamos a transferir apenas os seus valores correspondentes. Devemos no entanto, ter atenção ao facto de ser muito difícil, principalmente em Agentes móveis construídos em JAVA, aceder ao seu estado de execução, porque implica modificar a *JAVA Virtual Machine*. O programador pode contudo optar por guardar explicitamente o estado do Agente nos seus atributos associados, sendo os valores desses atributos transferidos para o próximo nó. A responsabilidade do tratamento do estado de execução fica assim a cargo do programador [12, 14, 49, 51].

3.4.3 Arquitectura Cliente-Servidor vs Agentes móveis

A arquitectura cliente-servidor é o paradigma mais comum naquilo que concerne a computação distribuída no momento. Neste paradigma todos os elementos são estacionários em relação à execução. É feito de forma usual um pedido de um cliente a um servidor solicitando a execução de um serviço; o servidor processa o serviço requisitado, utilizando os recursos existentes no próprio servidor; sendo posteriormente o resultado devolvido para o cliente. Este pedido contém por norma o nome do serviço a requisitar acrescido de alguns parâmetros. Nos últimos anos uma nova técnica

tem vindo a ganhar força (*Web Services*), permitindo gozar deste paradigma a uma escala ao nível da Internet, bem acima daquilo que outras arquitecturas como CORBA ou RMI, que surgiram primeiro e mais voltadas para a Intranet. A próxima figura, figura 17, fornece um exemplo deste tipo de implementação, sendo o servidor o *Host Y* e o cliente o *Host X* [12, 37, 45, 52, 53].

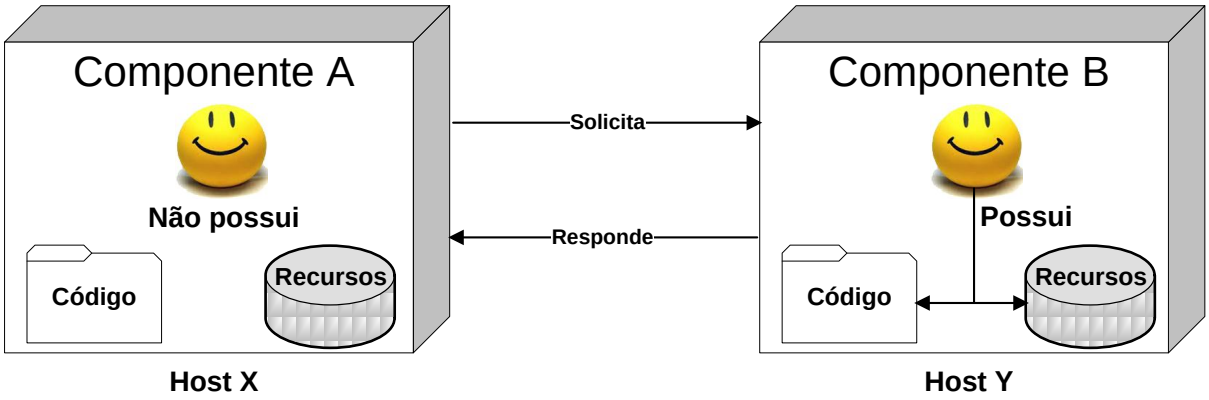


Figura 17 – Arquitectura Cliente-Servidor

Uma arquitectura com Agentes móveis tem de ser vista como um novo e diferente paradigma do que vimos anteriormente. Como tal, encontramos um exemplo base da implementação dessa arquitectura com um Agente móvel e dois *hosts* na próxima figura, figura 18 [12, 37, 45, 52, 53].

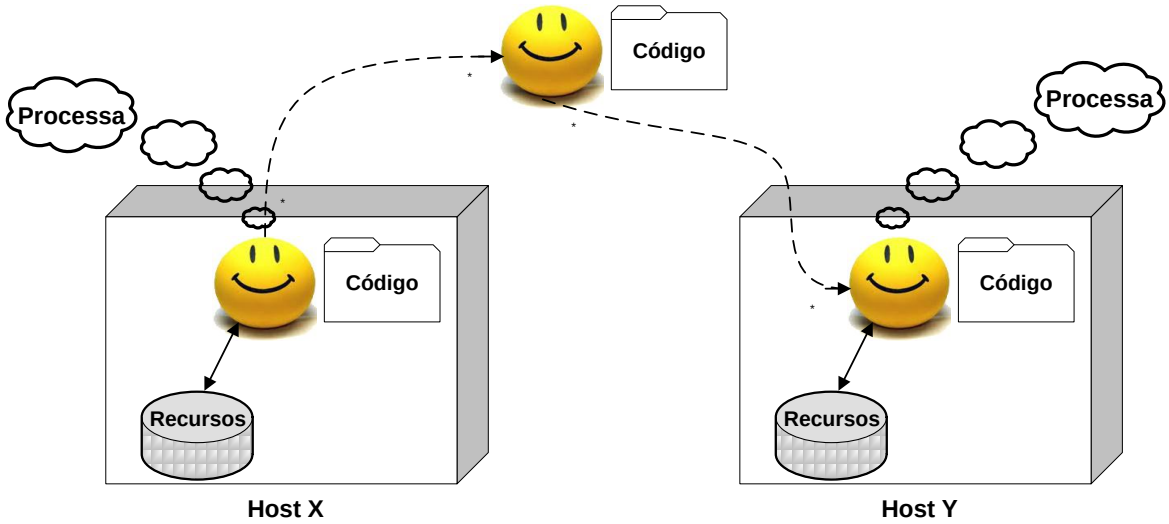


Figura 18 – Arquitectura Agentes móveis

O Agente apresentado no *host* x faz-se acompanhar de código e acede a recursos para executar uma ou mais tarefas. De seguida, devido à necessidade de outros e/ou mais recursos, apesar de possuir o código, necessita de se mover para o *host* y, onde continuará a sua execução. Outro tipo de necessidade pode surgir levando a que o Agente se mova, como por exemplo, o simples facto de ter alcançado os objectivos traçados inicialmente, estando portanto disponível para migrar para outras

máquinas na rede que eventualmente necessitem da sua presença, quer para o mesmo tipo de tarefas, quer para a realização de outras [12, 37, 45, 52, 53].

Se a componente de inteligência lhe estiver associada, o Agente é capaz de agir mediante a percepção que possui do ambiente que o rodeia e de acontecimentos que se vão dando à medida que a sua execução avança. Tanto pode decidir terminar uma tarefa, como adaptar-se de forma a cumpri-la de forma diferente do habitual, caso surjam problemas, ou até migrar para um novo destino [12, 37, 45, 52, 53].

3.4.4 Vantagens do Uso de Agentes móveis

Actualmente não podemos afirmar, com toda a certeza, que o paradigma dos Agentes móveis se possa afirmar de forma independente e estabelecer-se da forma como o paradigma cliente-servidor conseguiu. No entanto, no contexto deste trabalho o que se pretendeu foi estudar novas abordagens que pudessem coexistir com as aplicações tradicionais de gestão de redes, no sentido de tentar ultrapassar alguns dos problemas existentes (Cap. 4 e Cap. 5).

Para tal, procurou-se demonstrar a aplicabilidade do paradigma de Agentes móveis na gestão de redes. Do estudo efectuado destaca-se [3, 12, 13, 47, 48, 54-60]:

- ✓ **Redução do tráfego na rede**: nos sistemas distribuídos existem protocolos de comunicação que são utilizados gerando múltiplas interacções, produzindo um elevado tráfego na rede. Os Agentes móveis conseguem, como já referido, guardar determinados parâmetros, continuando posteriormente a sua execução em outro *host*, onde as várias tarefas vão ocorrer localmente e não com o auxílio de recursos da rede. Este procedimento diminui o número de ligações a estabelecer necessárias para completar as tarefas. Para além disso, o volume de dados a transferir decresce consideravelmente, porque estamos a efectuar o processamento localmente. A próxima figura, figura 19, ilustra estes aspectos.

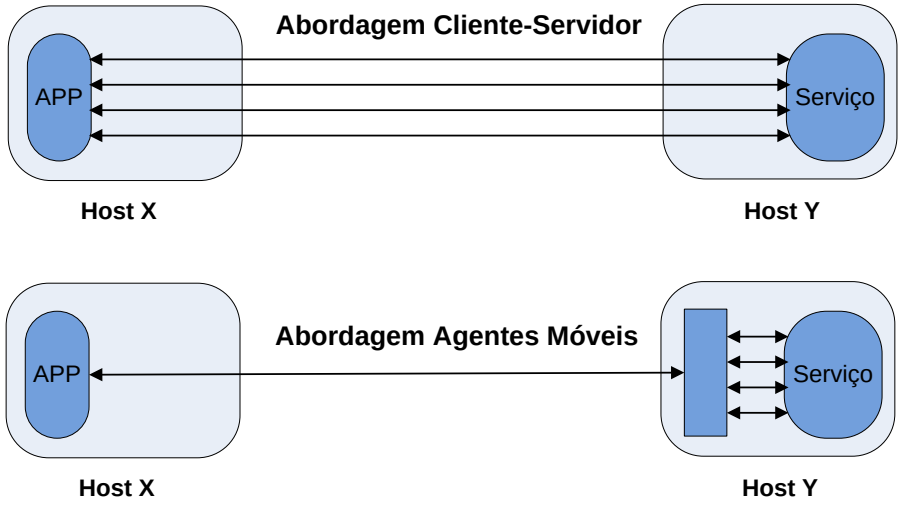


Figura 19 – Abordagem Cliente-Servidor vs Agentes móveis

- ✓ **Próximo da ausência de latências na rede**: sendo o sistema distribuído, e adicionando o facto de na maioria dos casos serem necessárias respostas imediatas, quando elevamos o número de comunicações é natural que surjam latências na rede. Nas soluções baseadas em Agentes móveis é possível enviá-lo para os pontos da rede onde é necessário efectuar tarefas de gestão. Assim, as latências de rede são reduzidas quase a zero;
- ✓ **Encapsulamento de protocolos**: na existência das inerentes trocas de informação num sistema distribuído, é necessário que cada *host* esteja preparado para interpretar os dados recebidos e enviados, estando dotado de protocolos específicos para tal. No entanto, esses protocolos vão evoluindo, passando a incluir novas funcionalidades, alteram-se e em cada *host* passa a ser necessário actualizá-los. Essa tarefa é muito difícil e trabalhosa para os administradores de sistemas, que muitas das vezes não a concretizam. Os Agentes móveis, por seu lado, conseguem mover-se para qualquer local, estabelecendo os seus próprios canais de comunicação baseados num protocolo implementado nos mesmos;
- ✓ **Execução assíncrona e autónoma**: os Agentes móveis podem ser configurados para levarem a cabo um conjunto de tarefas num determinado local para onde os Agentes são deslocados. Dessa forma, estamos a dar-lhe autonomia e estes podem operar assincronamente, independentes do processo que os criou originalmente. O Agente móvel pode voltar a conectar-se mais tarde, após a completa execução das tarefas, ou numa altura que considere ideal, de acordo com as suas capacidades autónomas. A figura seguinte, figura 20 ilustra de forma explícita e simples esta vantagem;

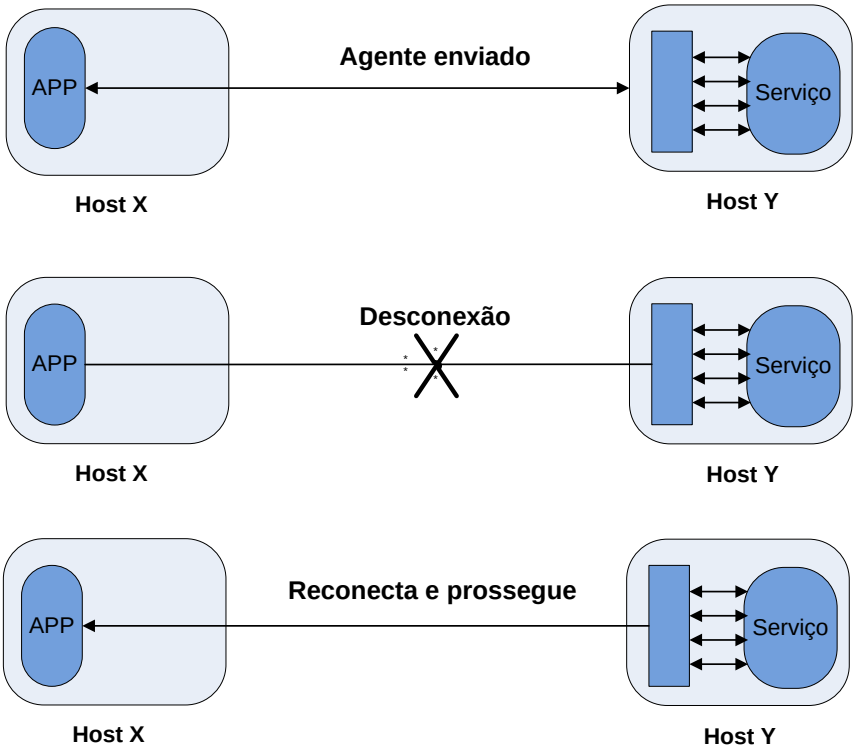


Figura 20 – Execução Assíncrona

- ✓ **Dinamismo**: os Agentes móveis podem fazer uso da sua autonomia e pró actividade, podendo tomar decisões com base na percepção que vão obtendo do ambiente em que estão inseridos. Se um determinado problema surgir, vários Agentes podem ainda distribuir-se de forma inteligente, conseguindo resolver um determinado problema que necessite da sua presença em vários *hosts*, ou seja, através de uma solução distribuída;
- ✓ **Heterogeneidade**: no que diz respeito à computação em rede, esta é naturalmente heterogénea, quer a nível do *hardware*, quer a nível do *software*. Os Agentes móveis são independentes dos computadores e camadas de transporte, dependendo apenas dos seus ambientes de execução. Consegue-se assim obter condições favoráveis para integrar redes heterogéneas;
- ✓ **Robustez e tolerância a falhas**: o dinamismo associado aos Agentes móveis permite que os Agentes reajam a situações adversas de forma autónoma, permitindo obter um sistema robusto e tolerante a falhas. Imaginando um *host* que se encontra a encerrar e em que se encontram Agentes a executar no momento, estes poderão ser informados, guardar o seu estado e migrar para um outro local, continuando a sua execução.

3.4.5 Mobilidade

O comportamento típico de um Agente móvel passa por migrar de um *host* para outro. Durante o procedimento o *host* actual (onde reside o Agente) pode ser chamado de emissor (*sender*) e o *host* destino (para onde o Agente se quer mudar) o receptor (*receiver*). Durante o processo de migração ambos necessitam de comunicar, transferindo dados pela rede, sobre o Agente que quer migrar. Pelo que é necessário um protocolo de comunicação, podendo ser também apelidado de protocolo de migração. Alguns sistemas simplificam esta vertente, através do uso de comunicação assíncrona, comparável a um *email*, enquanto que outros optam por desenvolver protocolos de comunicação mais complexos para aplicar sobre TCP/IP. Existem pelo menos seis passos essenciais neste processo [12, 13, 48, 52, 61].

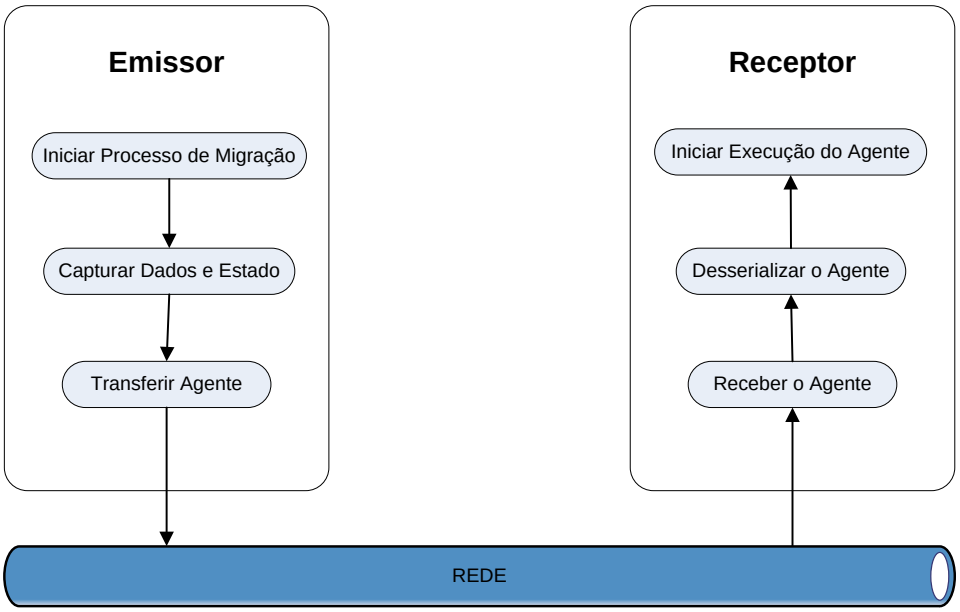


Figura 21 – Processo de Migração de um Agente (baseada em [12])

Como podemos ver na figura anterior, figura 21, o processo de migração começa de um forma típica, através de um comando que despoleta a ordem para o Agente migrar, pelo qual este anuncia a sua intenção e o seu destino. A primeira tarefa que o sistema no *host* deve iniciar é a suspensão imediata da *thread* responsável pela execução do Agente, garantindo que nenhuma outra, incluindo os filhos dessa mesma *thread*, fica activa. O segundo passo corresponde à captura do estado de execução e dos dados do Agente; todas as variáveis do Agente são serializadas, ou seja, os valores destas no momento, são escritas para uma representação externa e persistente, podendo-se tratar de um bloco de memória ou de um ficheiro, por exemplo. O estado do Agente também é guardado para que o ponto de suspensão seja conhecido, obtendo no final um Agente serializado, resultante de todo este processo e que corresponde a um *stream* de bytes contendo os dados e a informação do

estado. O terceiro e último passo da primeira fase é a migração propriamente dita, com a coadjuvação de um protocolo de migração. Os últimos três passos são executados no *host* que vai receber o Agente, começando pela recepção deste último. O *host* receptor verifica se o Agente pode ser recebido ou não, baseando-se não só no próprio Agente mas também na sua proveniência, assegurando que Agentes desconhecidos ou não considerados fiáveis não entram. A seguir o Agente tem de ser transformado numa cópia exacta daquilo que existia previamente no *host* emissor, na altura em que surgiu o comando a dar ordem para migrar. A terminar, uma *thread* nova deve ser lançada de modo a que o Agente possa estar operacional de novo e pronto a executar as tarefas no receptor [12, 13, 48, 52, 61].

Se colocarmos todos estes passos num plano mais simples, podemos fazer a comparação com um utilizador que navega na Internet. Quando este visita uma página Web, não está a visualizar directamente no servidor os seus conteúdos. É feito um *download* e uma cópia desses conteúdos são visualizados no cliente, existindo assim uma migração comparável ao referido relativamente aos Agentes móveis. São distinguidos dois tipos principais no que diz respeito à mobilidade [12, 13, 48, 52, 61]:

- ✓ **Mobilidade Fraca**: o Agente decide transferir-se guardando apenas a componente estática do seu estado de execução, como os valores dos atributos dos objectos estáticos, ao contrário por exemplo da *stack*, registos da máquina, entre outros, que fazem parte de uma componente dinâmica. A acompanhar a transferência emerge naturalmente o código que acompanha o Agente;
- ✓ **Mobilidade Forte**: neste tipo de mobilidade surge logicamente a transferência da componente dinâmica, que obriga a que o Agente prossiga a sua execução na máquina destino na instrução imediatamente a seguir à instrução que precedeu a transferência.

Uma primeira análise, sem grandes reflexões, indica a escolha pela mobilidade forte, uma vez que guarda a totalidade do estado do Agente móvel antes da transferência. No entanto, a mobilidade mais usada tem sido a fraca, porque os programas que implementam este tipo de mobilidade possuem uma eficiência mais elevada. Outra razão é a linguagem que os programadores optam por utilizar, nomeadamente JAVA, muito usada neste tipo de sistemas com Agentes móveis e que não possui mecanismos para armazenar a componente dinâmica, a não ser que se modifique a *Java Virtual Machine*. Contudo, a mobilidade fraca permite tirar partido de uma construção de um sistema baseado em Agentes móveis sem que isso se transforme numa forte limitação [12, 14, 49, 51].

3.4.6 Agentes móveis e a Linguagem de Programação

A escolha da linguagem para desenvolver um Agente terá efeito na arquitectura do Agente produzido. As linguagens fornecem um conjunto de funcionalidades para a implementação dos Agentes. As características que devem ser observadas na escolha são [3, 12, 14, 48, 49, 61, 62]:

- ✓ **Orientação a objectos**: os Agentes são objectos e a comunicação de Agentes ocorre através da invocação de métodos que constituem a sua interface pública;
- ✓ **Independência de plataforma**: são usados em computadores heterogéneos e em diferentes sistemas de Agentes distribuídos. A linguagem deve oferecer facilidades que permitam esta interação;
- ✓ **Capacidade de comunicação**: para tal, a linguagem deve possibilitar que se implementem componentes orientados à comunicação; através deles é que os Agentes se comunicam entre si, com recursos internos e externos num ambiente de rede;
- ✓ **Segurança**: deve ser oferecido pela linguagem um alto grau de funcionalidade, a partir dos modelos de segurança específicos da linguagem ou através da integração de modelos externos (protocolos de criptografia, *firewalls*);
- ✓ **Manipulação de código**: as aplicações podem requerer que o código do programa seja manipulado em tempo de execução. Para tal, são necessários mecanismos de identificação do código do Agente.

Mesmo tendo todas estas características, esta linguagem deve ser reactiva, multitarefa e permitir armazenamento de dados persistentes. Assim, uma linguagem bastante popular na construção de Agentes é o Java, que é desenvolvida pela *Sun Microsystems*, orientada a objectos e baseada em redes.

A próxima tabela, tabela 3, apresenta as plataformas mais conhecidas no universo dos Agentes móveis. Normalmente não existem aplicações baseadas em Agentes móveis, mas sim várias aplicações que beneficiam deste paradigma, tais como aplicações de *e-commerce*, obtenção de informação distribuída, serviços de telecomunicações e rede, gestão de redes informáticas, processamento paralelo, entre outros [3, 22, 44, 48, 51, 52, 55, 63, 64].

Toolkit	*Organização*	*Endereço Web*
ADK	Tryllian	*www.tryllian.com*
Aglets	Open Source	*aglets.sourceforge.net*
Ajanta	University of Michigan	*www.cs.umn.edu/Ajanta/*
Concordia	Mitsubishi	*www.merl.com/projects/concordia/*
D' Agents	Dartmouth College	*agent.cs.dartmouth.edu*
Grasshopper	IKV	*www.grasshopper.de*
Mole	University of Stuttgart	*mole.informatik.uni-stuttgart.de*
Odyssey	General Magic	dev-heba.wikispaces.com/file/view/odyssey.doc
Semoa	Fraunhofer Society	*www.semoa.org*
Tacoma	Uni Tromso	*www.cs.uit.no/forskning/DOS/Tacoma/*
Tracy	University of Jena	*www.mobile-agents.org*
Voyager	ObjectSpace/Recursion	*www.recursionsw.com*

Tabela 3 – Plataformas de Agentes móveis

As *toolkits* que são referidas na tabela anterior resultam de investigação na área dos Agentes móveis, das quais vamos destacar algumas individualmente [3, 22, 44, 48, 51, 52, 55, 63, 64]:

1. **Odyssey** é uma biblioteca de desenvolvimento de Agentes móveis desenvolvida pela General Magic, sendo que esta empresa já teria desenvolvido um primeiro *toolkit* de nome "Telescript". No entanto, este teve pouco sucesso devido à sua natureza proprietária dando lugar ao aparecimento da Odyssey, com uso de linguagens e protocolos de rede abertos;
2. **Concordia** foi desenvolvida pela Mitsubishi e é conhecida por ser rica em funcionalidades, necessitando apenas da implementação padrão da *Java Virtual Machine*, tendo um ambiente constituído por um servidor e Agentes. Possui mecanismos de segurança e tolerância a falhas;
3. A **Voyager** também é muito conhecida, no entanto não oferece mecanismos de segurança contra Agentes não autorizados. Assenta numa plataforma ORB (*Object Request Broker*) com suporte a Agentes;
4. **Aglets**, uma das mais populares *toolkits* de Agentes móveis, desenvolvida inicialmente pela *IBM Tokyo Research Laboratories* e mais tarde pela comunidade *Open Source*. A

grande vantagem reside no facto de ser extremamente bem desenhada e ao mesmo tempo possuir facilidade de utilização. Contém um sistema de segurança bastante robusto.

Todas estas abordagens têm pontos em comum, como por exemplo, o facto de serem implementadas em JAVA, utilizarem a sua máquina virtual e ainda técnicas de serialização de objectos para envio de informação. Qualquer uma destas *toolkits* fazem uso de um Agente intermediário, *proxy* ou facilitador, como já vimos na secção anterior. A execução e aceitação de Agentes em todas as abordagens é feita num ambiente controlado, para que se possa implementar alguma segurança base, onde apenas Agentes da mesma *toolkit* são aceites dentro de um determinado sistema. As grandes diferenças entre todas estas *toolkits* reside nos mecanismos de transporte de Agentes e na forma como estes interagem entre si, para além da implementação da segurança, que sofre abordagens diferentes [3, 22, 44, 48, 51, 52, 55, 63, 64].

3.4.7 JAVA nos Agentes móveis

A linguagem de programação Java tornou-se num *standard* no que diz respeito aos Agentes móveis, muito por culpa das suas muitas capacidades que permitem diminuir o esforço em construir *toolkits* completas. Na maioria dos sistemas existe uma restrição no que diz respeito à linguagem de programação a usar para o desenvolvimento dos Agentes e a linguagem usada para os Agentes comunicarem, obrigando a que seja a mesma. Não é comum os Agentes serem programados em linguagens distintas, no entanto, a plataforma usada neste trabalho, possui como umas das características principais a capacidade de no mesmo sistema, ser possível encontrarmos Agentes capazes de comunicar entre si e com o facilitador, ainda que programados em linguagens diferentes [12, 65-67].

Quase todos os *toolkits* desenvolvidos nos últimos anos foram-no através da linguagem Java, tanto para o *toolkit* para Agentes móveis, como para os próprios Agentes em si. A característica mais importante que fez de Java uma linguagem voltada para a Internet foi a sua portabilidade. Os programas escritos em Java são compilados para uma arquitectura independente num formato *byte-code* e de seguida este é executado usando a *Java Virtual Machine*. Como as máquinas virtuais estão disponíveis para quase todo o tipo de *hardware* e Sistemas Operativos, os programas em Java têm a enorme vantagem de poderem ser executados em quase todos os sistemas de computação. A portabilidade é um requisito muito importante nos sistemas de Agentes móveis, uma vez que estes têm de migrar numa rede de sistemas computacionais heterogéneos [12, 65-67].

O que se encontra no formato *byte-code* é então executado na máquina virtual, o que protege o sistema operativo de acessos directos não autorizados por parte dos programas em Java. O controlo de segurança é simplificado, uma vez que a existência de um formato intermediário de código

permite uma mais fácil procura de violações de segurança do que no código nativo. A própria linguagem suporta o desenvolvimento de aplicações "*seguras*", porque, ao contrário da linguagem C por exemplo, um dos componentes da *Java Virtual Machine*, o *byte code verifier*, filtra o código que viola semântica base de Java antes da execução. Mesmo durante o período de *runtime*, um gestor de segurança controla todas as operações potencialmente inseguras, como acesso a ficheiros ou conexões de rede. Como já foi referido, a linguagem Java suporta programação orientada a redes, usando *sockets* ou *remote method invocation* (RMI), que se trata de uma versão do RPC orientado a objectos. Java RMI é tão poderoso que podemos criar uma *toolkit* muito simples de Agentes móveis com apenas cem linhas de código [12, 65, 66].

Nos próximos parágrafos são apresentadas algumas das características que fazem da linguagem Java a eleita no desenvolvimento de Agentes móveis [3, 12, 14, 49, 61, 65]:

- ✓ **Independência de Plataforma**: Java foi desenhada para permitir a operação em redes heterogéneas. Tal como referido anteriormente, a programação em Java é compilada num formato especial que será lido pela máquina virtual, instalada no sistema e independente de qualquer arquitectura. Podemos assim desenvolver Agentes móveis sem a preocupação do tipo de sistema computacional em que estes irão ser instalados;
- ✓ **Execução segura**: aquando da sua criação, a linguagem Java foi pensada para utilização não só na Intranet mas também na Internet, transformando a segurança num ponto fulcral do seu desenho. Com a simples utilização desta linguagem, ganha-se automaticamente alguma segurança no que toca à aceitação de Agentes não reconhecidos, uma vez que estes, devido à sua arquitectura, não conseguem mexer no código do *host* ou aceder a informação para a qual não têm autorização;
- ✓ **Carregamento de classes dinâmico**: com este mecanismo, a máquina virtual consegue carregar e definir classes em *runtime*. Cada Agente está protegido por um *namespace*, permitindo a sua execução de forma independente e segura uns dos outros;
- ✓ **Programação com múltiplas *threads***: cada um dos Agentes é executado de forma independente dos outros, cada um na sua própria thread, mesmo que seja no mesmo local. Este tipo de programação consegue fazer com que os Agentes possam ganhar autonomia. Para além da programação *multithread*, encontramos um conjunto de primitivas de sincronização disponíveis e que são vitais para a interacção entre os Agentes;
- ✓ **Serialização de objectos**: o facto de os Agentes poderem ser serializados e desserializados é a chave para a mobilidade. A linguagem Java possui de raiz esta possibilidade, permitindo a representação de um objecto numa forma suficientemente detalhada, que possa ser reconstruída posteriormente. A nova forma que o objecto toma depois de

serializado tem de permitir que este consiga reconhecer a classe a partir da qual o seu estado foi guardado, restaurando-o numa nova instância.

Apesar de todas as vantagens referidas e explicadas relativamente ao uso da linguagem Java, devemos ter em conta também algumas limitações que podem levantar problemas na fase de implementação de um sistema com Agentes móveis. Uma das limitações tem a ver com o facto de esta linguagem possuir um suporte inadequado para o controlo de recursos. O Java não está dotado de nenhum sistema que permita controlar os recursos que um determinado objecto Java consome. Ataques do tipo *Denial of Service* (DoS) são possíveis, uma vez que um Agente pode consumir memória de forma ilimitada, através da execução de um ciclo infinito, retirando todos os recursos necessários ao funcionamento normal do sistema no qual está a operar [3, 12, 14, 49, 61, 65].

Cada um dos Agentes precisa de ter noção de quantos e quais os Agentes que existem na sua comunidade, no seu sistema e quais estão a tentar a aceder aos seus objectos. Para tal, são utilizados os métodos públicos presentes nos objectos Java, de modo a que um outro objecto possa-lhes fazer referência. Isto é vantajoso mas tem um senão: não existe protecção de referências. A solução viável para este problema passa pela implementação de Agentes intermediários, os chamados *proxies* ou facilitadores [3, 12, 14, 49, 61, 65].

Uma outra dificuldade já referida, embora superficialmente nesta dissertação, corresponde ao suporte, ou neste caso, a não existência dele, para guardar e carregar um estado de execução em pleno. Informações como o estado do contador de programa ou a *frame stack* são regiões permanentemente proibidas a programas em Java. Torna-se impossível a um Agente retomar a sua exacta computação no *host* para onde se acaba de mover (considerando a mobilidade forte). Esta tarefa é exequível se optarmos por mexer na máquina virtual de Java, de outra forma o Agente baseia-se em atributos internos e registo de eventos externos para informar o novo *host* de que ponto de o Agente retomar a sua execução [3, 12, 14, 49, 61, 65].

3.4.8 Conclusão

Ao longo das últimas secções foi apresentado o estado actual do estado da arte relativamente a Agentes móveis. A nossa análise procurou dar ênfase à tolerância a falhas, configuração, segurança, entre outros. No entanto, a utilização de Agentes móveis também permite atingir metas no que diz respeito a todas as áreas funcionais da gestão de redes. Os Agentes móveis continuam a ser objecto de pesquisa e investigação, apesar de não serem muito utilizados em redes/aplicações de grande escala. Este facto deve-se à existência de acomodação à gestão tradicional, além de que em muitos casos, em empresas, institutos ou universidades, não existir de todo gestão com base em aplicações.

Outro motivo prende-se com a necessidade de desenvolver este tipo de sistemas para que todas as plataformas o possam utilizar, criando ambientes em que haja interoperabilidade. Pretende-se que não se note diferenças ou exclusões entre Sistemas Operativos, fabricantes de activos de rede ou mesmo aplicações de software. Apesar de tudo, em alguns domínios, como as redes activas, observamos a existência de um crescimento do uso ou tentativa de incorporar técnicas de Agentes móveis: um exemplo é o dos pacotes que circulam na rede carregarem código que será executado em *switch's*, permitindo a sua configuração dinamicamente. Outras aplicações têm surgido na área das telecomunicações e mesmo na área de e-commerce.

4 Cenário Real - Rede do GECAD

4.1 Introdução

O GECAD possui duas áreas de trabalho localizadas em dois edifícios, separados fisicamente por uma distância considerável, dentro do campus do Instituto Superior de Engenharia do Porto.

A primeira área está localizada no quarto piso (por inteiro) do edifício I. Sendo constituído por dois laboratórios (brevemente serão três), com postos de trabalho, uma sala de reuniões, uma sala de ambientes inteligentes de decisão, um espaço onde se encontram vários equipamentos de rede, servidores e *workstations*, e o gabinete da directora. Todas as áreas excepto o gabinete da directora são iguais em termos de área como iremos ver, com mais detalhe, mais à frente neste documento. A figura seguinte, figura 22, permite ver o posicionamento do GECAD no campus do ISEP.

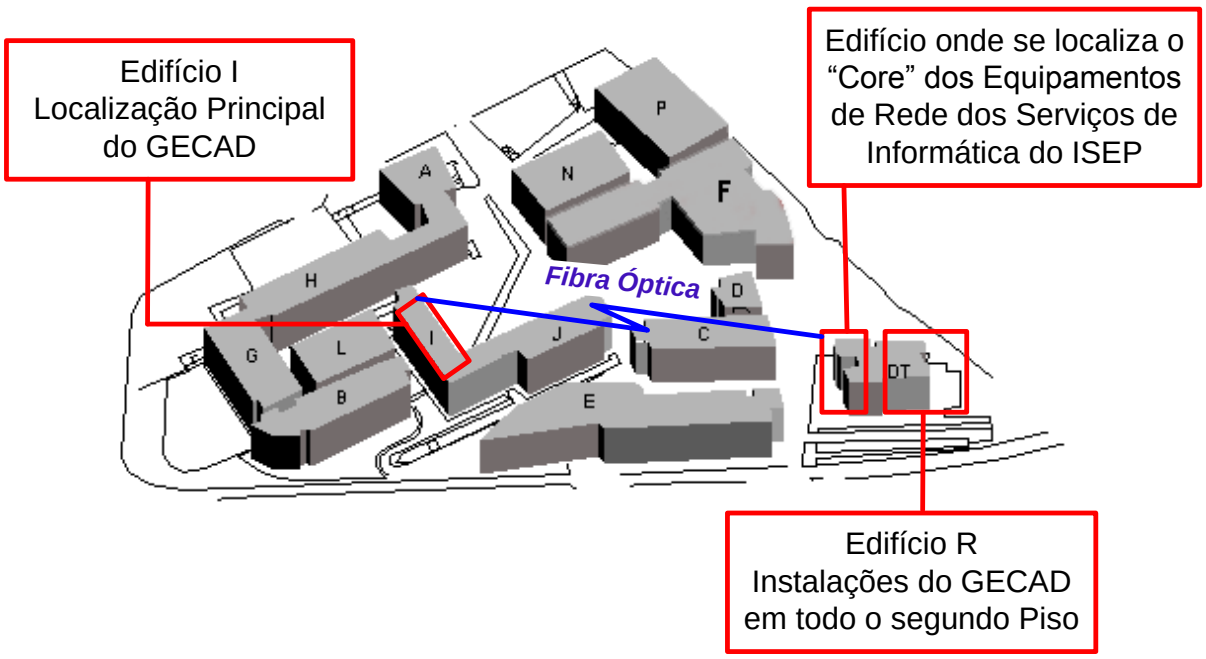


Figura 22 – Posicionamento do GECAD no ISEP

A segunda área está localizada no edifício R do ISEP, no segundo piso.

Junto deste, separado apenas por uma travessia, encontra-se uma outra edificação onde se localizam os principais equipamentos de rede dos serviços de informática do ISEP. A conexão entre o edifício I e o edifício R foi facilitada pelo facto de já existir uma ligação de fibra óptica entre o edifício I e o *Datacenter* do ISEP (DSI). O caminho restante é concluído em cabo UTP CAT5e, permitindo velocidades até 1Gbps.

A rede LAN do GECAD consiste assim num sistema de cablagem estruturada, tipo Ethernet, a 10/100/1000 Mbps, composta por três segmentos periféricos: Bastidor I307; Bastidores 1 e 2 da I403; e Bastidor R210. A arquitectura que interliga estes segmentos é assegurada por um *switch* "LinkSys SRW2048" no bastidor I307 (de realçar que é Bastidor de Edifício), que distribui com cabo UTP CAT5e (*Backbone* de Edifício) para os Bastidores I403 e para o R210, através de fibra óptica (*Backbone* de Campus). No entanto, não é o *switch* do GECAD que possui ligação com fibra, mas sim um *Router Switch* do ISEP que se encontra ligado ao *LinkSys* através de um chicote UTP. Este *Router Switch* é da total responsabilidade dos serviços de informática do ISEP. A cablagem horizontal é composta pela utilização de cabos UTP CAT5 que permitem velocidades até 1000Mbps. De uma forma resumida, todas as tomadas de rede que se encontram nas instalações do GECAD, edifício I, (piso quatro), seguem até ao bastidor no Piso três do mesmo edifício, na sala I307.

No que diz respeito ao edifício R, todos os pontos de rede convergem para a sala R210, onde se localiza um *switch* do GECAD para o suporte aos equipamentos dos utilizadores.

4.2 Arquitectura e Topologia da Rede

Relativamente a equipamentos, convém referir que ligados à rede do GECAD há: cerca de vinte cinco computadores portáteis, cinquenta PC's e quatro impressoras, das quais três são multifunções. Há, ainda a registar, treze servidores. O equipamento activo será descrito numa das secções seguintes deste capítulo (secção 4.4.1).

Em seguida iremos descrever o Domínio existente na rede de dados do GECAD. O nome do domínio é "gecad.isep.ipp.pt*"* e o nome NETBIOS é "GECAD". A classe de endereços escolhidos é da classe C, privados, permitindo obter a seguinte configuração: endereços na gama "192.168.2.1-192.168.2.199" para servidores, estações de trabalho, equipamento activo e impressoras; os endereços a partir de "192.168.2.200" até "192.168.2.253" são usados para clientes VPN. Neste momento os segmentos de endereços precisam de algum trabalho uma vez que denotam alguma desarrumação, não existindo claramente a típica separação de servidores, equipamento activo e estações de trabalho.

A máscara de rede é a "255.255.255.0", e a próxima figura, figura 23, permite ver o número de *subnets* existente (apenas uma), o número total de controladores de domínio, assim como, quais são os *Global Catalog Servers*. A conexão entre os dois servidores indica a replicação de dados em termos de *Active Directory*, que é feita entre eles.

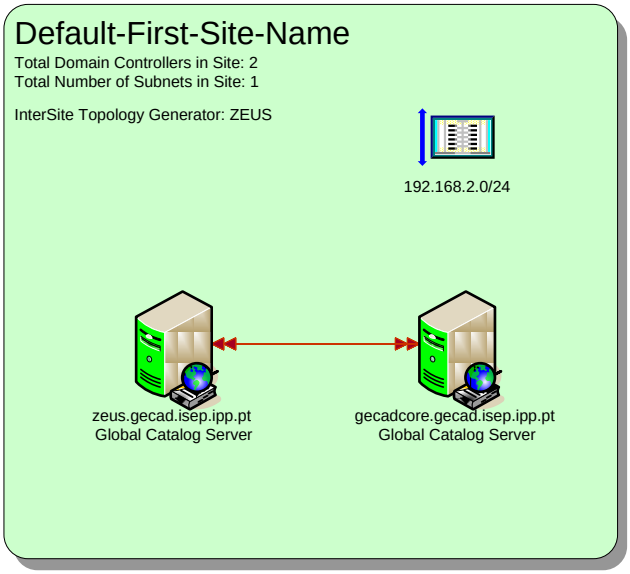


Figura 23 – Topologia de Replicação entre Controladores de Domínio

O serviço de DNS está integrado no *Active Directory*, tendo o GECAD dois servidores de nomes. Em todos os casos de uma implementação mais recente deste serviço, por cada domínio, dentro de uma determinada floresta, é criada uma partição para guardar os registos de forma integrada, figura 24.

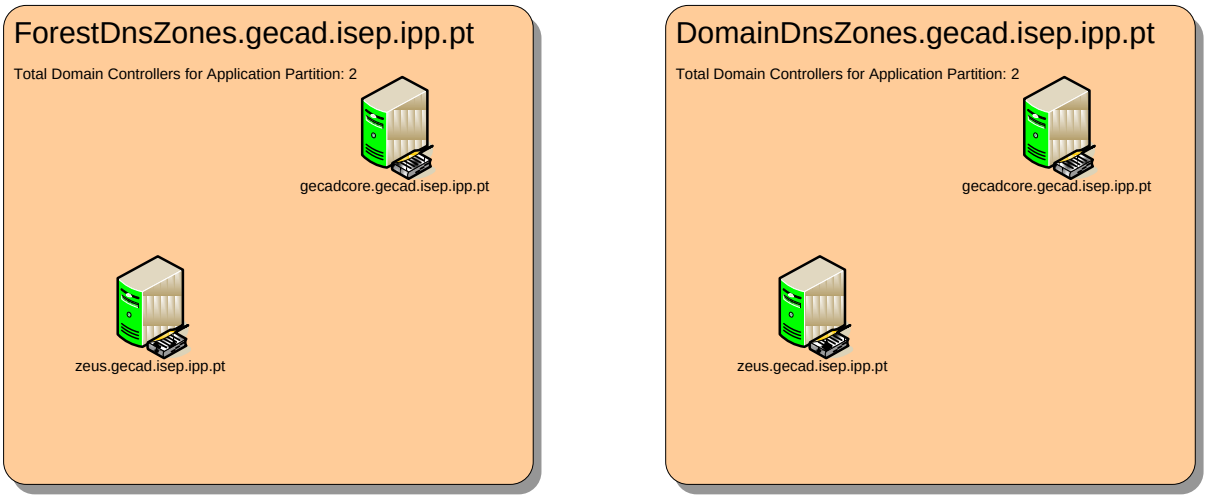


Figura 24 – DNS - Floresta e Domínios do GECAD

Pelo facto de apenas existir um domínio dentro da floresta, apenas se visualiza na figura anterior, figura 24, os mesmos dois controladores de domínio, sendo estes responsáveis por manter tanto as partições aplicacionais da floresta como do domínio em si. As duas máquinas em questão são os servidores "GECADCORE" e "ZEUS" e o nível funcional é o do *Windows Server 2003*. No entanto, e embora os papéis principais já estarem atribuídos ao "GECADCORE" que possui o

Sistema Operativo da Microsoft, *Windows Server 2008*, é necessário garantir uma elevação ao nível funcional para *Windows Server 2008*, passando a ter em todos os controladores de domínio este Sistema Operativo instalado.

Na próxima figura, figura 25, pode-se observar o descrito com detalhe, inclusive o número de utilizadores registados no *Active Directory*, assim como, as atribuições dos vários papéis relativos à gestão do domínio, entre outras informações úteis.

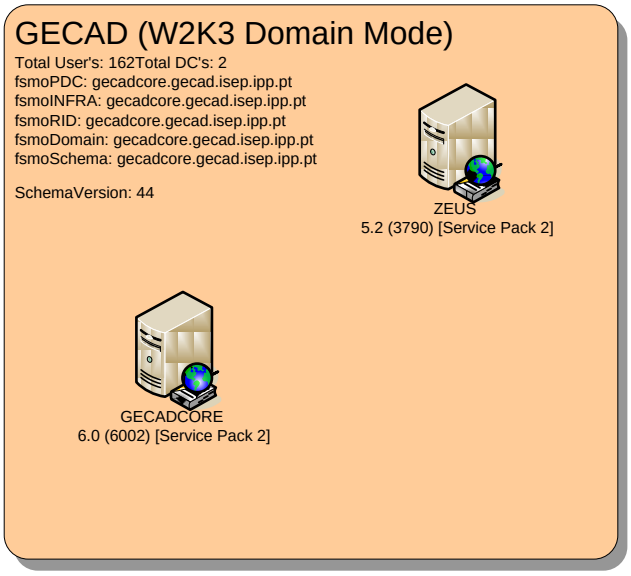


Figura 25 – Domínio GECAD

Na realidade o número total de utilizadores é muito elevado. Não correspondendo directamente ao número de PC's acima apresentado. Isto deve-se ao facto de um grande número significativo dos elementos do GECAD estar fisicamente em outras localizações acedendo aos serviços e conteúdos do GECAD através de redes Virtuais (VPN).

4.3 Meio Físico

Um estudo mais aprofundado do meio físico que contém todos os equipamentos ligados à rede do GECAD, permite compreender melhor o meio físico. O edifício I possui cinco potenciais salas de trabalho. Duas são efectivamente usadas exclusivamente através de postos destinados aos investigadores da unidade que trabalham de forma regular, salas I404 e I405. As figuras seguintes, figura 26 e figura 27, apresentam as plantas respectivas com a disposição actual dos equipamentos.

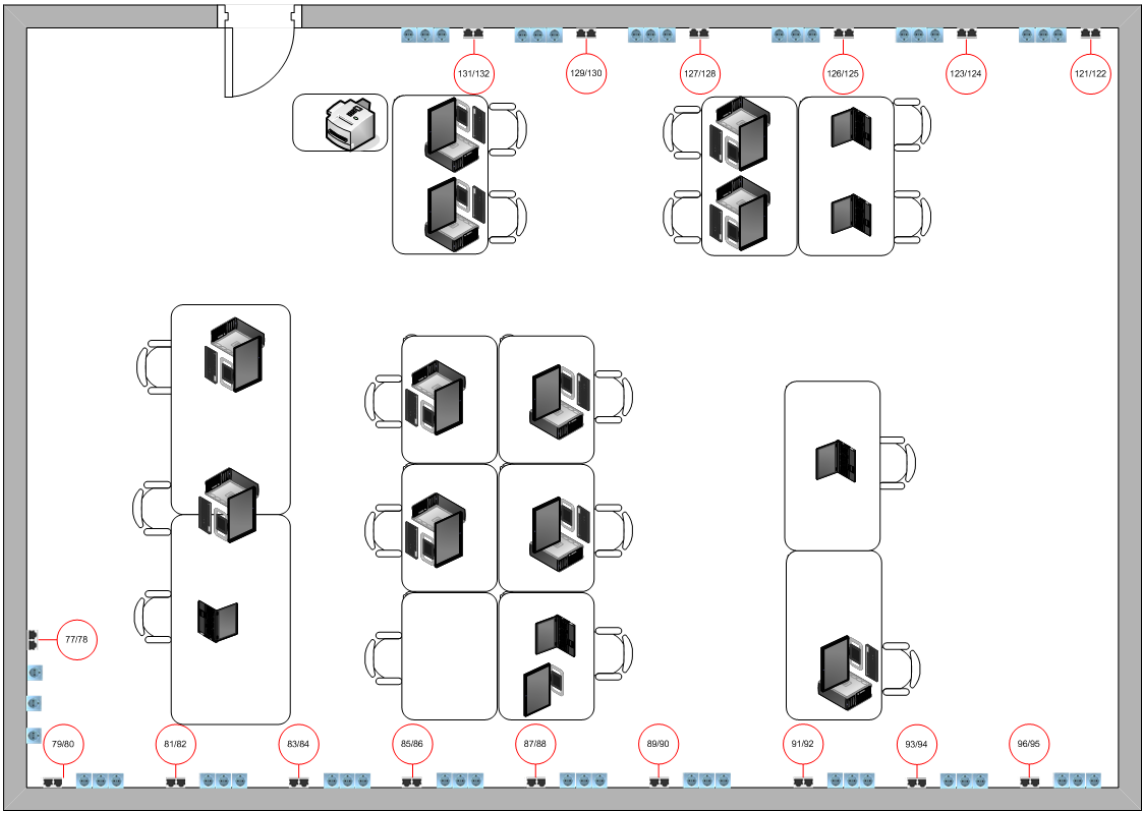


Figura 26 – Sala I405

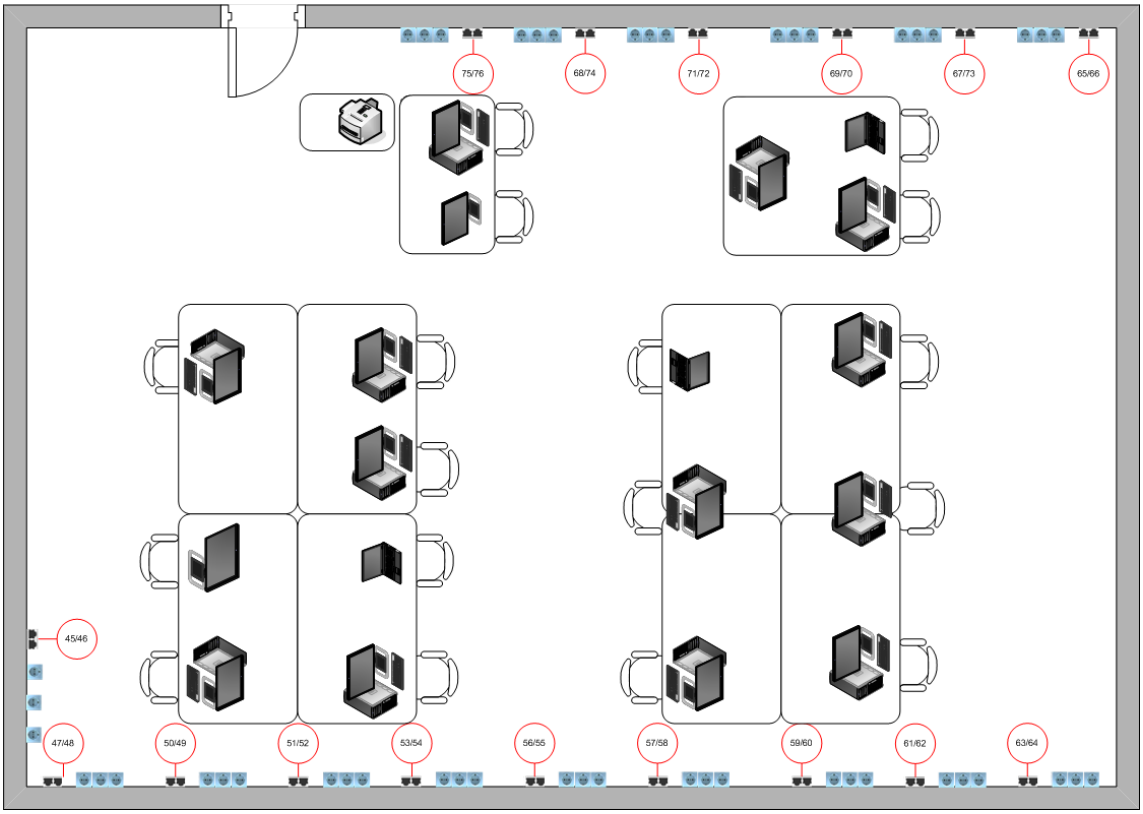


Figura 27 – Sala I404

Da observação das figuras anteriores, figura 26 e figura 27, é possível concluir que a cablagem horizontal percorre as salas apenas através de calhas técnicas nas paredes laterais, o que condiciona bastante a acomodação dos cabos e dos activos de rede. Sendo que este facto obriga ao uso de *switch's*, a maioria sem gestão, de forma a facilitar a chegada da rede aos computadores, o que poderia ser feito de forma mais directa se existissem pontos de rede em todas as paredes. Desta forma, segmenta-se por vezes em demasia algumas secções do GECAD, tornando mais difícil a gestão da rede, uma vez que os pacotes têm de dar mais um salto através dos activos sem gestão, sem SNMP e mais difíceis de monitorizar.

Ambas as salas possuem uma impressora de rede, *Desktop's* e *Workstations*, para os investigadores que necessitam de executar tarefas que requerem maior capacidade de processamento. As diferentes bancadas estão também protegidas por UPS's que permitem a conexão a computadores via USB.

Convém, no entanto referir que durante o desenrolar deste trabalho está-se a preparar a colocação de mais postos de trabalho na sala I405, uma vez que ainda existe espaço disponível, e mais dois na sala I404, completando a capacidade da mesma.

A sala de ambientes inteligentes de decisão (I406) é um espaço diferente dos anteriores no que ao seu uso diz respeito. É bastante usada para reuniões formais e para demonstrações do trabalho desenvolvido no GECAD. Esta sala (figura 28) é caracterizada por conter seis postos de trabalho, cada um com um monitor interactivo, sensível ao toque. Para além deste equipamento, existe um plasma de 62'', também este sensível ao toque, ligado à Workstation que vemos na imagem anterior, ideal para apresentações que decorrem naturalmente neste tipo de ocasiões. A sala está, ainda equipada com um vidro interactivo, conectado a um pequeno PC e um sistema Vídeo/Áudio constituído por três câmaras e várias colunas integradas no tecto.

O grande objectivo da existência deste espaço em conjunto com o equipamento descrito é tornar esta sala, tal como o nome indica, num laboratório de ambientes inteligentes de decisão. Com a utilização de software auxiliar, desenvolvido por investigadores do GECAD e presente nos postos de trabalho, assim como na coadjuvação do sistema de vídeo/áudio, as reuniões realizadas tomam um sentido mais protegido e inteligente quando chega a tomada de decisão.

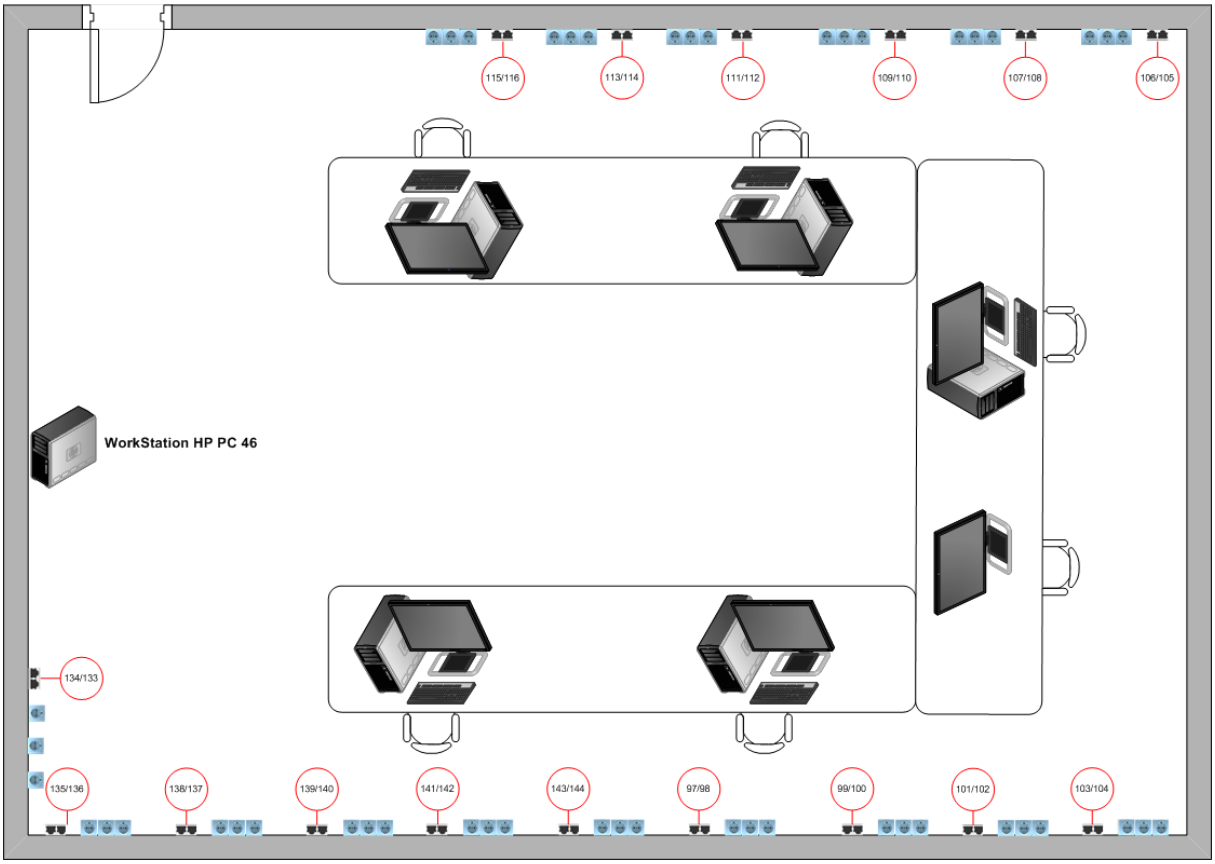


Figura 28 – Sala I406 – Laboratório De Ambientes Inteligentes de Decisão

O próximo laboratório a ser apresentado é aquele que possivelmente estará ligado de uma forma mais íntima à gestão de redes, como podemos constatar na figura 29.

Como ilustrado existe uma divisória que permite isolar os futuros postos de trabalho de forma acústica e térmica, do pequeno espaço que é composto pelos vários equipamentos que sustentam as actividades do GECAD em termos informáticos.

Encontramos dois bastidores que serão avaliados em mais detalhe e também diversos servidores no formato "Tower".

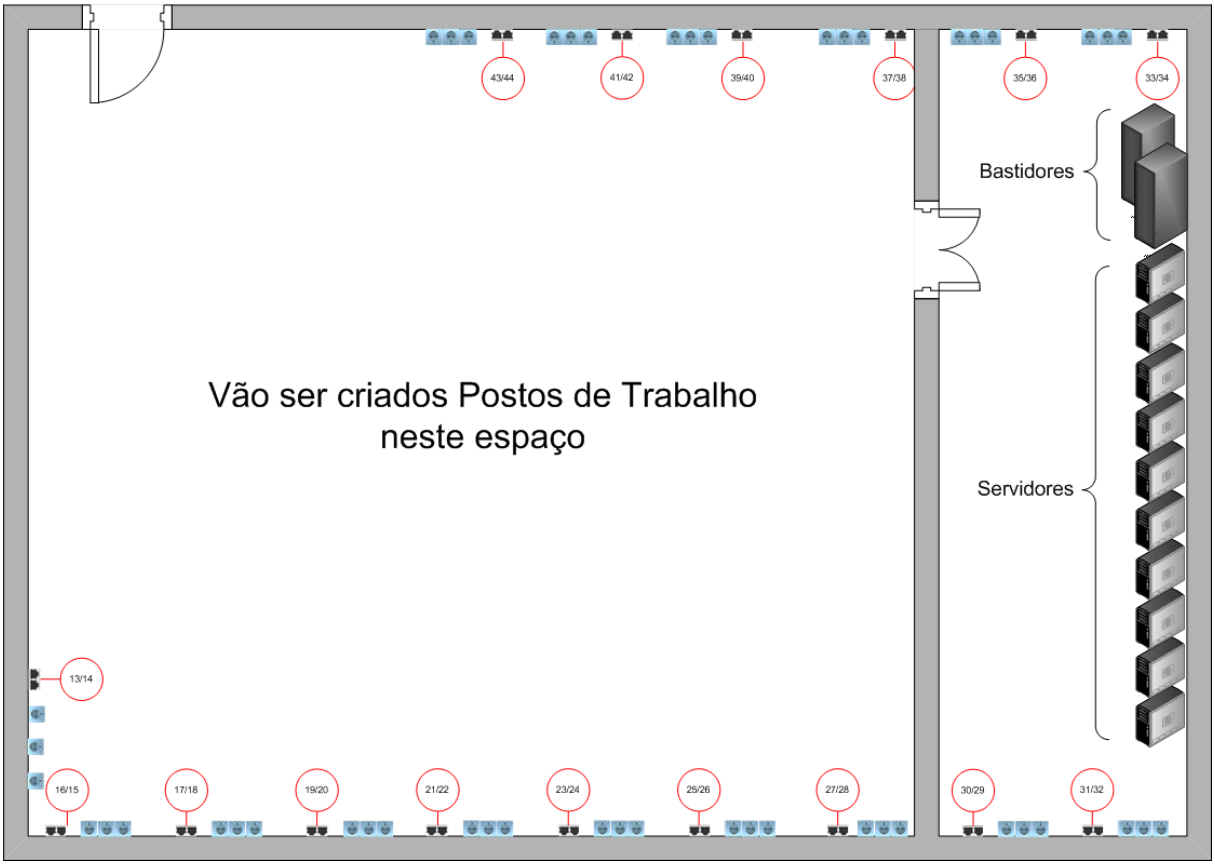


Figura 29 – Sala I403

À primeira vista a presença de tantos servidores e Workstations quando comparado com o número de utilizadores, postos de trabalho e laboratórios poderia ser considerada anormal, do ponto de vista de alguém que possua experiencia a avaliar este tipo de ambientes. No entanto, muitos dos projectos necessitam de *clusters* de PC's, plataformas de processamento massivo ou até de vários servidores dedicados, com os diferentes serviços associados, a cada um deles.

De facto, a parede na qual se encontram os bastidores e servidores na imagem 29 mede 7.40m, o que é um valor considerável.

A figura seguinte, figura 30, mostra com detalhe e pormenor o que está instalado.

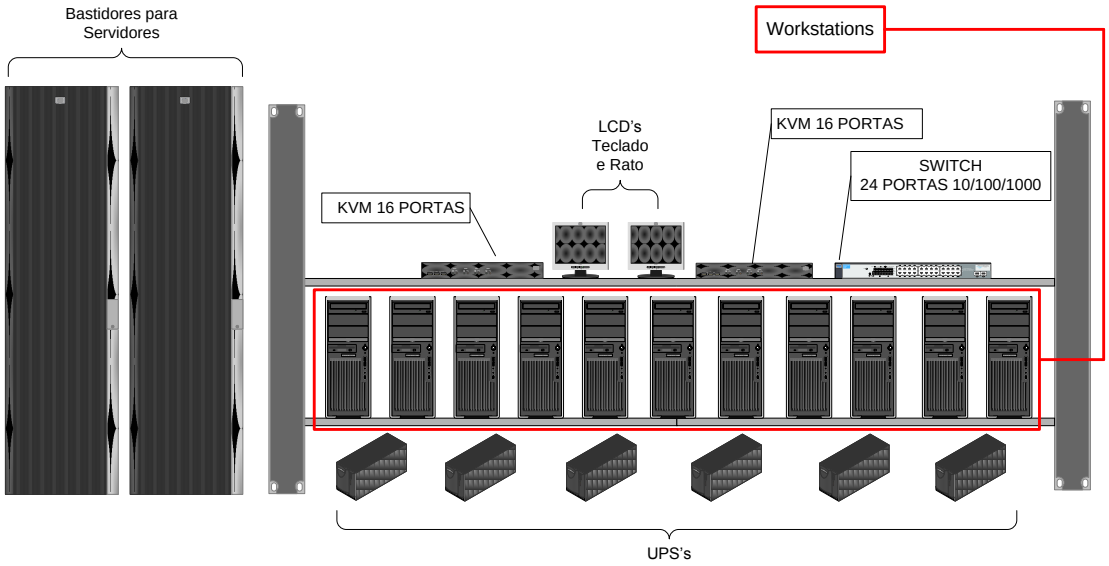


Figura 30 – Infra-estrutura Informática Sala I403

O bastidor mais à direita está em pleno uso das suas capacidades, uma vez que possui um distribuidor de energia, uma KVM, servidores, um *switch*, entre outros. Já o seu semelhante vai entrar em funcionamento a partir de Novembro de 2011, pois existe a necessidade de expandir e melhorar as condições para os investigadores do GECAD, na plataforma de computação massiva que foi adquirida em 2008. Junto destes começa uma estante que contém várias workstations e servidores de vários projectos, com diferentes fins, consoante as necessidades. São ilustrados doze servidores na estante, mas esta possui mais alguns *Desktop's* (perfazendo dezanove computadores no total) que são usados esporadicamente, com serviços não tão fulcrais como as máquinas que são mostradas. Por cima temos o *switch* que suporta todo o equipamento, mesmo as UPS's que permitem a conexão à rede via Ethernet. A terminar estão dois monitores, teclados e ratos que permitem o controlo das máquinas imediatamente abaixo, através da ligação às duas KVM's existentes, suportando cada uma, dezasseis PC's.

Actualmente o centro de investigação GECAD não se limita ao edifício I, conforme já referido anteriormente. O segundo piso do edifício R, como já foi mostrado anteriormente, foi recentemente remodelado para acolher os nossos investigadores, de forma a dar resposta às necessidades de expansão e crescimento da unidade de investigação. A figura 31 dá-nos uma perspectiva geral daquilo que está ao dispor do GECAD.

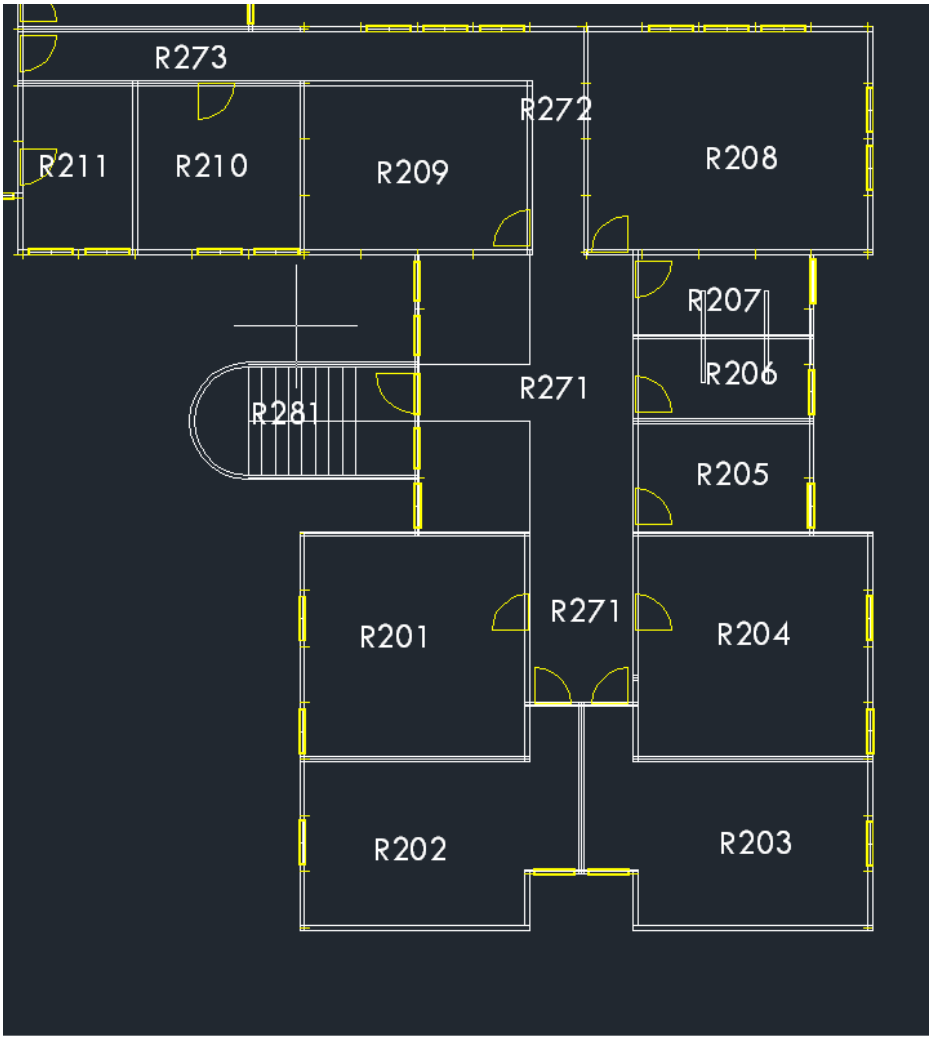


Figura 31 – Edifício R

As áreas representadas com a nomenclatura R271, R272, R273 e R281 correspondem a zonas comuns, mais precisamente corredores e escadas. A sala R203 trata-se de um espaço para reuniões, onde não existem postos de trabalho e a R204 possui dois gabinetes com dois postos de trabalho. R201 e R202 são os laboratórios de energia e electrónica respectivamente, onde existem entre os dois, cerca de cinco postos de trabalho com *Desktop's*, acrescidos de alguns espaços para portáteis. R205 é uma sala de apoio, sem equipamentos conectados à rede e os espaços imediatamente ao lado são os WC's.

Seguem-se as salas R208 e R209, ambas destinadas a acolher bancadas com postos de trabalho para vários investigadores. No entanto, apenas uma está em funcionamento com cerca de doze postos (R208), um *switch* que suporta os equipamentos localizados neste espaço e duas UPS's que protegem os mesmos. A sala 209 será brevemente mobilada e acolherá mais algumas pessoas.

Resta por último referir a sala R210, onde se encontra um bastidor de edifício, com activos de rede do ISEP e os activos de rede do GECAD para suportar a infra-estrutura.

4.4 Componentes da Rede

4.4.1 Bastidores e Respectivo Equipamento Activo

Nesta secção iremos abordar com mais detalhe os segmentos periféricos da rede, assim como os equipamentos activos que permitem o bom funcionamento desta. Vamos iniciar a análise pelo bastidor localizado no Piso três do edifício I, responsável pelos pontos de rede do GECAD, entre outros.

Os primeiros conjuntos de *patch panels* não pertencem a pontos de rede do GECAD, mas sim aos do terceiro piso do edifício I. É correcto dizer que perto de metade do bastidor não é da responsabilidade da nossa unidade de investigação, tal como a imagem ilustra. Junto a este bastidor encontram-se mais dois, também pertencentes ao ISEP e responsáveis por assegurar a comunicação para os pontos de rede do piso um e dois. Daí resulta o facto destes três bastidores serem considerados de edifício, não existindo a segmentação por piso. Olhando de forma mais atenta para a segunda metade do bastidor apresentado na figura 32 encontramos quatro *patch panels*, cada um com quarenta e oito portas passíveis de serem usadas, acrescidos de dois organizadores de cabos, sendo que estes já competem e referem-se aos pontos de rede e cablagem do piso quatro do mesmo edifício, isto é, laboratórios do GECAD.

Inicialmente a infra-estrutura estava assegurada através da utilização de três *switch's HP Procurve 1800G*, com gestão, de vinte e quatro portas a 10/100/1000 Mbps e ainda com o *Linksys SRW2048*, este já com quarenta e oito portas a 10/100/1000 Mbps e com funcionalidades acrescidas em relação aos HP. Devido à recente expansão das instalações com a introdução do segundo piso do edifício R, um dos *HP's* (com o IP "192.168.2.23") foi necessário para ser colocado nessa localização, para permitir distribuir a rede pelos pontos existentes nas salas que já estão em funcionamento.

Esta situação conduziu à imagem actual que perdura, por enquanto, uma vez que a partir de finais de Novembro de 2011 este bastidor terá de novo três *switch'*s "HP Procurve 1800G". Ao deslocar um dos *switch's* ficamos apenas com dois de vinte e quatro portas (IP's "192.168.2.21" e "192.168.2.22") mais o "Linksys" de quarenta e oito ("192.168.2.20"), não conseguindo, no entanto, ter alguns pontos importantes ligados, resultante desta mudança.

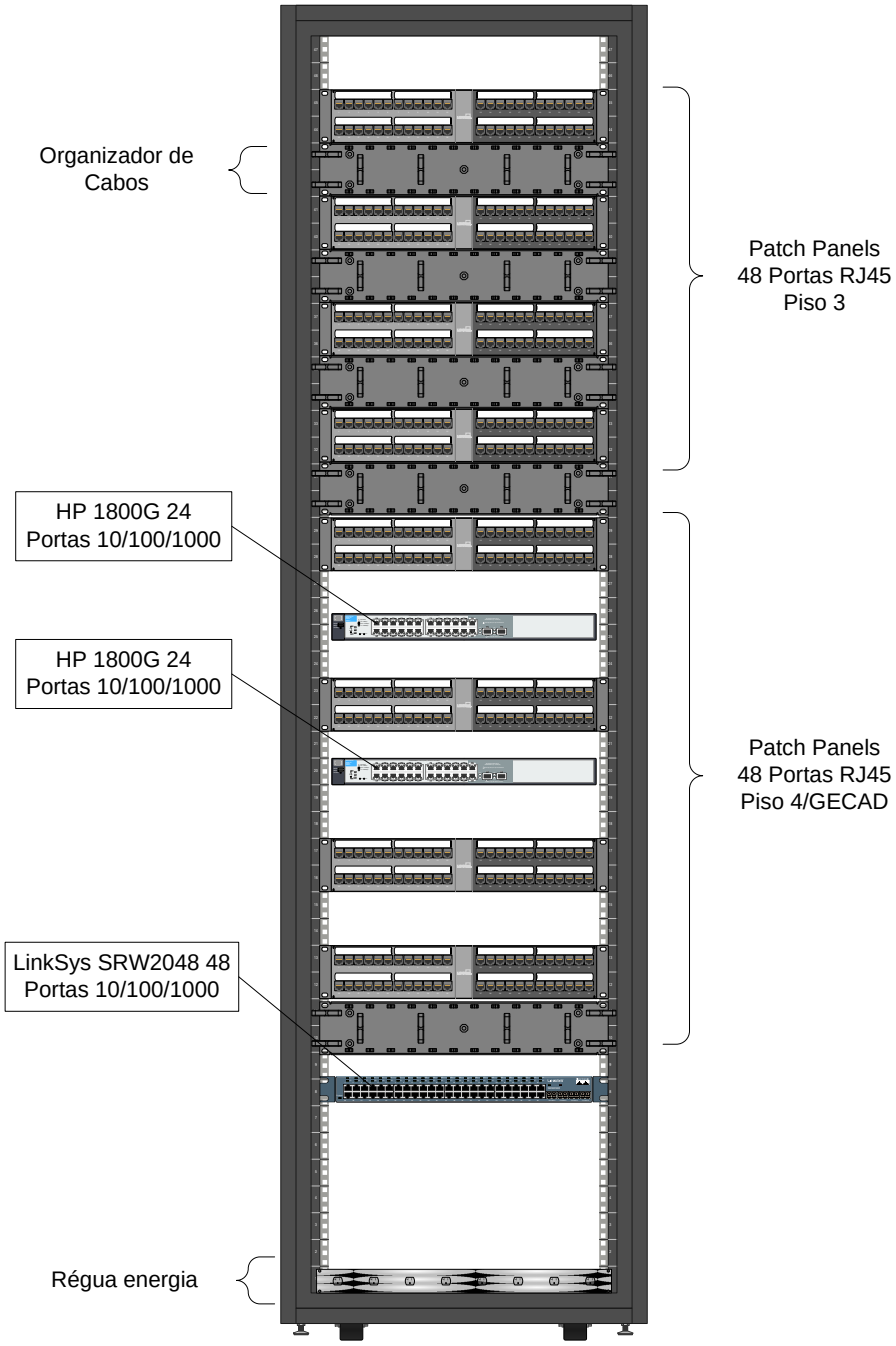


Figura 32 – Bastidor I307

Um bastidor muito importante para o suporte da nossa infra-estrutura informática localiza-se na sala I403, ou seja, já nas próprias instalações do GECAD, no edifício I. É caracterizado por conter vários servidores, incluindo a nossa plataforma de computação massiva, voltada para processamento extremo. Na figura 33, observando o bastidor de forma ascendente, encontramos em baixo uma UPS que o sustenta e que permite monitorização e gestão, seguido de um servidor WEB de um projecto em específico da unidade. Imediatamente acima localizam-se dois de três servidores

pertencentes à plataforma de computação massiva do GECAD, usados pelos vários utilizadores em aplicações como MATBLAB, GAMS, PSCAD, entre outros como aplicações mais rotineiras. Um dos objectivos passa também por fornecer aos utilizadores que possam ser possuidores de máquinas mais antigas, melhores condições de trabalho.

Estes servidores possuem, cada um, dois processadores XEON X5450 a 3.0 GHz, com 12MB cache L2 e oito núcleos no total. Os seus nomes são "appserver" e "servusers", sendo que o que os diferencia neste momento passa pelos 4GB e 6GB RAM, Sistema Operativo Windows Server 2008 32 bits e 64 bits respectivamente. O "appserver" usufrui de quatro discos SAS a 10000k 146 GB, estando estes configurados em RAID 1, dois a dois. Um par para o Sistema Operativo e outro para os dados dos utilizadores. Já o "servusers"apenas tem dois discos em RAID 1 usados para ambos os efeitos.

Mais acima encontramos um outro servidor, um router e um *switch*. O servidor corresponde ao ponto onde está armazenado o site principal da unidade de investigação, assim como alguns serviços que iremos abordar mais tarde, o router Cisco 1801 é responsável pela comunicação com o exterior, fazendo de gateway para os clientes e o *switch* existente suporta a comunicação e ligação com a rede dos equipamentos que vemos neste bastidor. Devemos ainda destacar o PDU (*Power Distribution Unit*), que se encontra conectado à UPS. De uma forma mais simples, a UPS vai buscar a energia à rede eléctrica, a uma tomada normal, seguindo-se a conexão com o PDU. Ou seja: em vez de termos todos os equipamentos ligados directamente à UPS, estes estão ligados a uma série de extensões, cada uma com seis fichas fêmeas de três pinos, típicas daquele tipo de ligações de uma fonte de alimentação de um PC normal (onde se encontra o macho). O PDU, depois de receber a energia da UPS, distribui-a por essas extensões que permitem a conexão dos equipamentos presentes no bastidor. É de relembrar que a UPS é muito importante no apoio a toda esta parte da infra-estrutura, estando conectada à rede via Ethernet, permitindo uma gestão mais apurada e aperfeiçoada no caso de falhas de energia, uma vez que gera notificações e *logs* importantes.

A KVM, reunida com o conjunto teclado rato e monitor de gaveta, geram uma capacidade de controlo sobre os principais servidores do bastidor, assim como, de outros dois que se encontram imediatamente à direita deste. Em cada uma das máquinas existe um adaptador que reúne os conectores de rato, teclado e VGA, afluindo num cabo UTP tradicional de rede por IP que termina na KVM. Por último, há o servidor "gecadcore" que reúne todas as condições que permitem ao administrador de rede administrar o domínio GECAD sob a plataforma Windows Server 2008. Este possui outros serviços vitais para o bom funcionamento da rede, como DNS e DHCP, entre outros. Durante este trabalho o servidor que albergava as funções principais de controlador de domínio foi

substituído por este mais recente, sendo que o cerne da implementação prática assim como a aplicação de gestão que está em uso no momento passaram a coexistir também neste servidor. A figura seguinte, figura 33, permite identificar com mais detalhe tudo aquilo que é relevante neste bastidor, assim como identificar quais as acções de gestão de redes a executar.

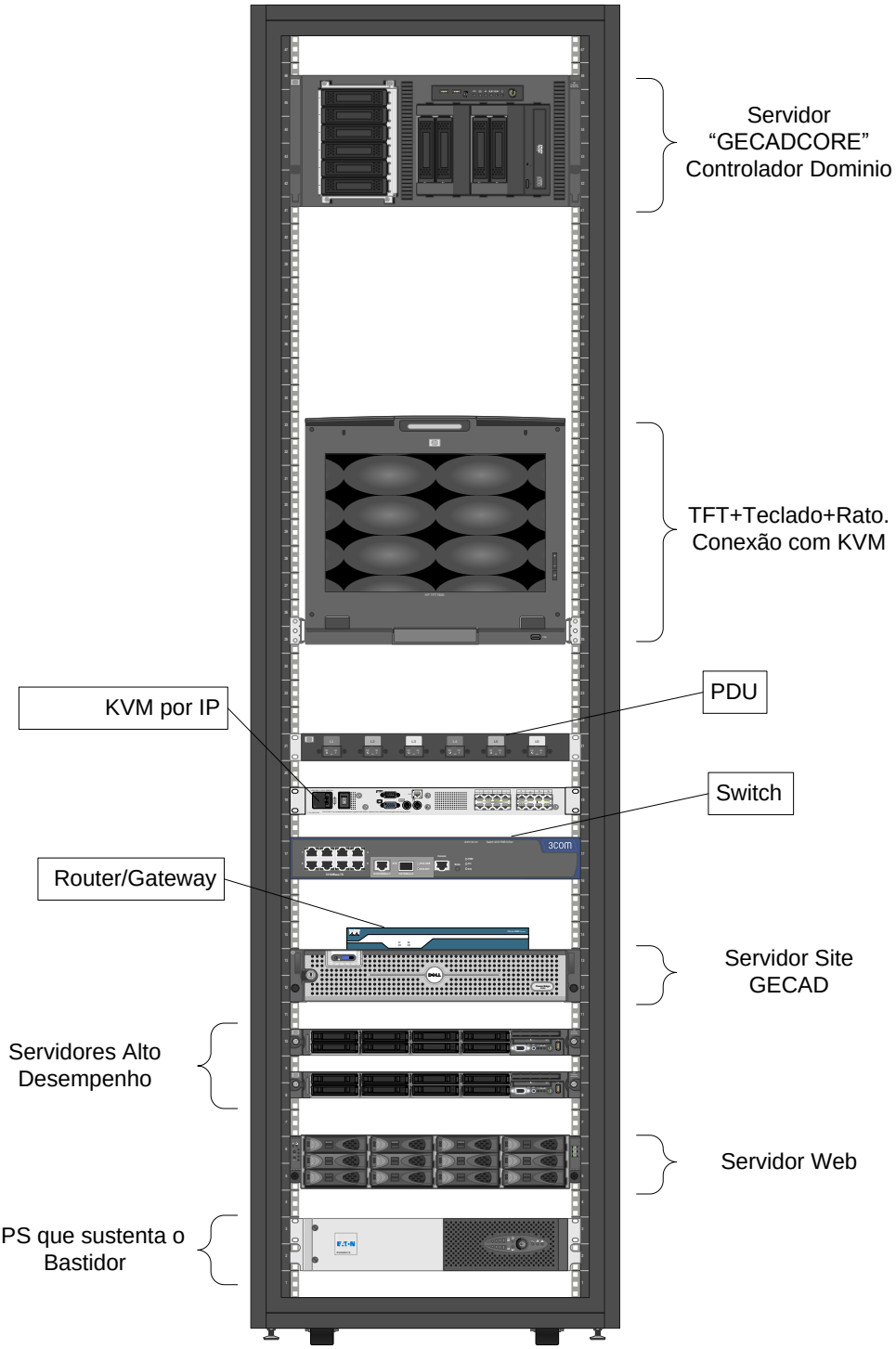


Figura 33 – Bastidor I403

A seguinte figura, figura 34, ilustra o modo como os dois bastidores, referidos anteriormente, se encontram conectados e de que forma a rede assenta e toma como base estes dois pontos fulcrais.

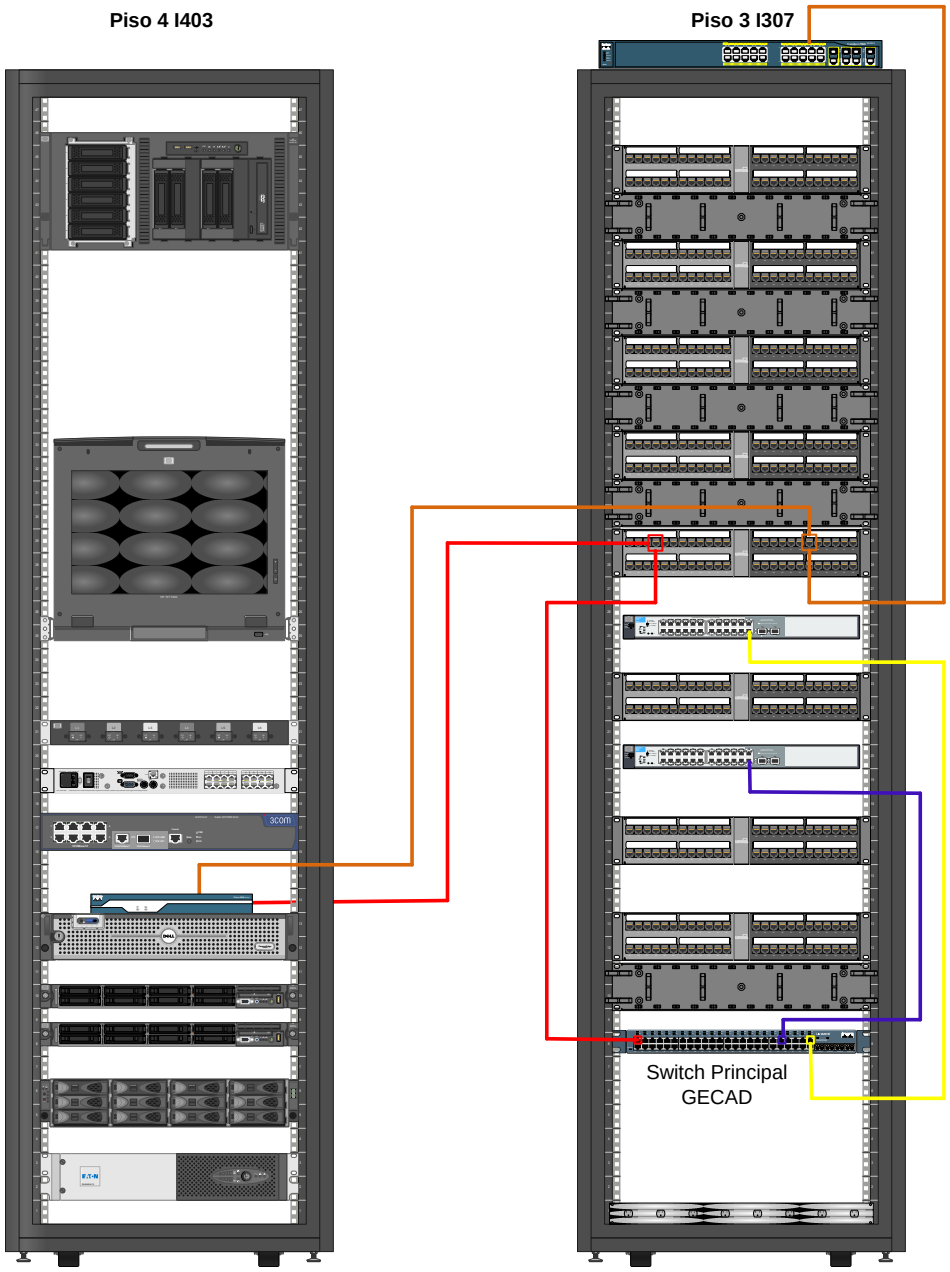


Figura 34 – Conexão entre I403 e I307

Como a figura ilustra, o router/gateway do GECAD possui duas conexões, uma à rede LAN e uma outra WAN à rede do ISEP. Uma vez que a sala onde se encontra o *switch* principal da unidade não possui as melhores condições para a subsistência dos equipamentos de rede, para além de ser distante das instalações do GECAD, optou-se por colocar o router no piso quatro, sendo necessário

um caminho de dados um pouco mais complexo para as suas ligações. De facto, a sala I307 é demasiado quente, tendo já colocado em risco equipamentos em casos extremos de temperatura, principalmente no Verão. Questões que podem decorrer facilmente da observação da figura são:

- Porque não colocar o router ligado ao próprio *switch* do bastidor em que se encontra;
- Porque não colocar o router no bastidor da sala I307, o que permitiria que este ficasse ligado directamente às duas redes.

As respostas a estas questões prendem-se com o facto de o acesso a esta sala ser mais complicado e moroso, havendo algum distanciamento entre esta e as instalações da unidade de investigação. Se o router estivesse na I307 e fosse necessário algum tipo de intervenção que exigisse proximidade física ao equipamento, seria muito mais fácil trabalhar no local onde ele se encontra neste momento, para além de que o processo de gestão de uma rede não passa só pela parte lógica mas também pela parte física e respectiva preservação dos equipamentos, nas melhores condições de trabalho.

Com as cores amarela e lilás temos a representação das ligações dos dois *switch's* HP de vinte e quatro portas 10/100/1000 Mbps ao *switch* principal ajudando a suportar a rede neste edifício. Tal como estas conexões existem, convergindo para um equipamento central da rede, também o router/gateway está ligado ao mesmo *switch* para comunicação com a rede interna. O que foi feito para diminuir os saltos na rede quando são efectuados pedidos ao router, passou por encaminhar dois chicotes UTP até duas tomadas de rede muito próximas do bastidor, tomadas essas cujo caminho de dados desemboca no bastidor da sala I307, e como representado na imagem, uma das ligações segue do *patch panel* para o *switch* principal e outra para o router do ISEP, permitindo o acesso externo.

Estando o router na sala I403, o único caminho/forma mais adequado para comunicar com o exterior era este, no entanto, podia-se ter optado por ligar o router à rede interna, directamente ao *switch* 3COM que se encontra no bastidor junto dele. Isto não foi feito porque se olharmos para o trajecto de um pedido de um utilizador à internet, por exemplo, os pacotes de rede teriam de seguir desde a estação de trabalho, passar para o piso três até ao *switch* HP correspondente (um dos dois existentes), onde a máquina está ligada, voltando de novo para o piso de cima e entrando no router, não sem antes passar, pelo *switch* 3COM em causa, mais próximo de si fisicamente. No retorno, o pedido era encaminhado para o mesmo *switch*, mas teria ainda de dar um salto adicional para o segundo *switch* no piso três e só daí partir para a estação de trabalho destino. Desta forma, conseguimos diminuir o número de saltos, apesar da distância percorrida pelos pacotes ser maior em metros, o que se torna irrisório devido à capacidade e características da cablagem.

A área de trabalho do GECAD no edifício R possui também um bastidor, como se pode observar na figura seguinte, figura 35.

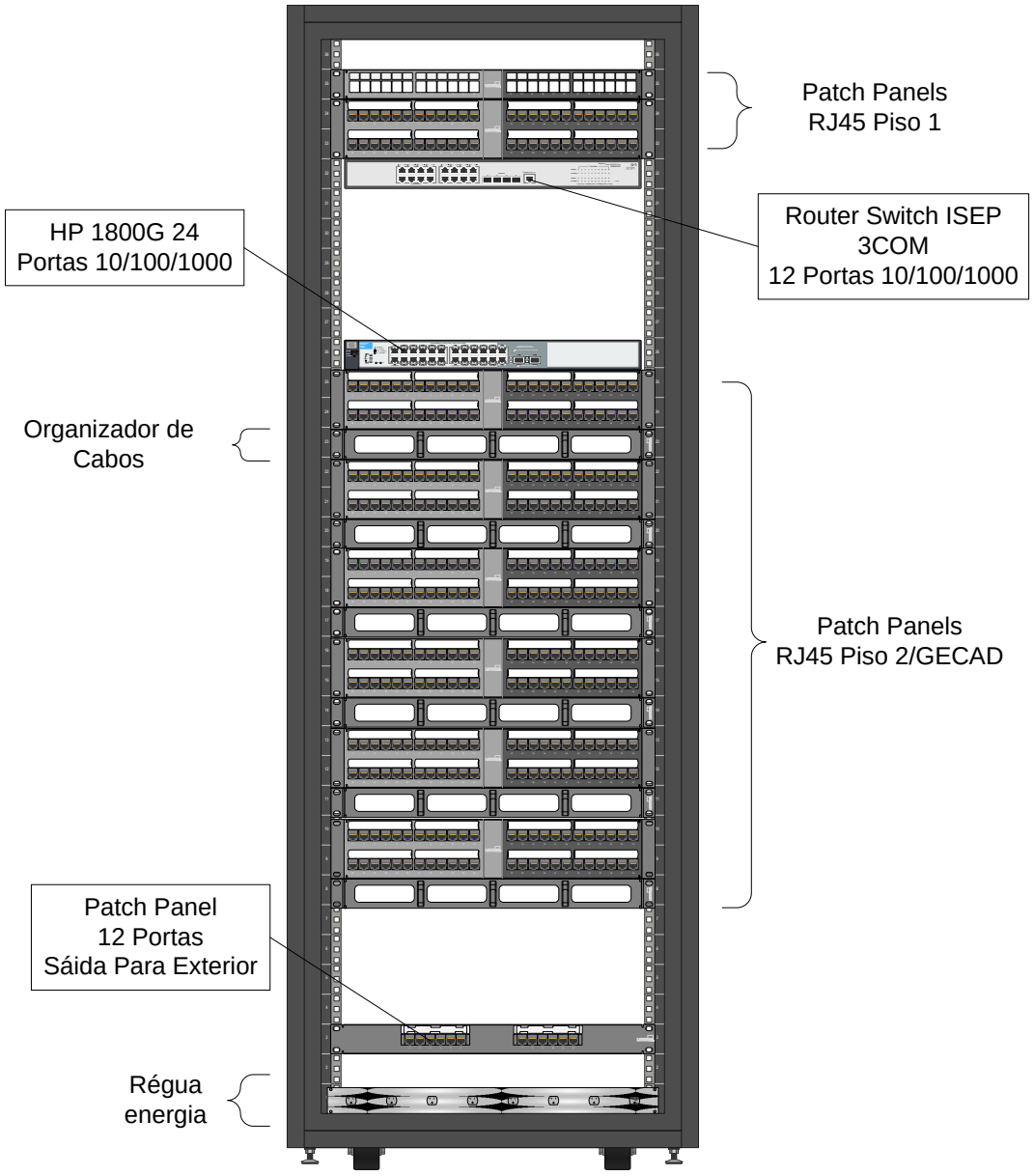


Figura 35 – Bastidor R210

O *Router Switch* do ISEP permite a interligação da rede do GECAD entre o edifício I e o R, pelo que ligar um PC num edifício ou num outro é indiferente, uma vez que a rede é fisicamente a mesma, proporcionando uma gestão mais flexível e directa. Neste bastidor existem seis *patch panels* de quarenta e oito portas e um outro, apenas com doze, das quais cinco são usadas para comunicação com o exterior do edifício R. O *switch* HP que é visível na figura 35 é responsável por suportar a infra-estrutura neste edifício, sendo neste momento suficiente, uma vez que não

possuímos mais de vinte e três pontos de rede activos. Uma das portas deste *switch* é usada naturalmente para conectar-se ao router *switch* do ISEP, de forma a obter conexão com o edifício I. Brevemente será necessário aumentar esta capacidade, sendo que o GECAD já se encontra precavido através da compra de *switch's* idênticos, aumentando assim o número de elementos de rede a monitorizar. Por outro lado, no bastidor que se encontra na sala I307, também se está a atingir a saturação, o que provavelmente conduzirá a alterações no futuro próximo que podem passar por efectuar um acréscimo de equipamento do mesmo tipo no edifício R.

Detalhando um pouco mais e fazendo uma análise mais profunda do que existe e deve ser objecto de gestão neste trabalho, no que concerne a equipamento activo temos o exposto na figura seguinte, figura 36.

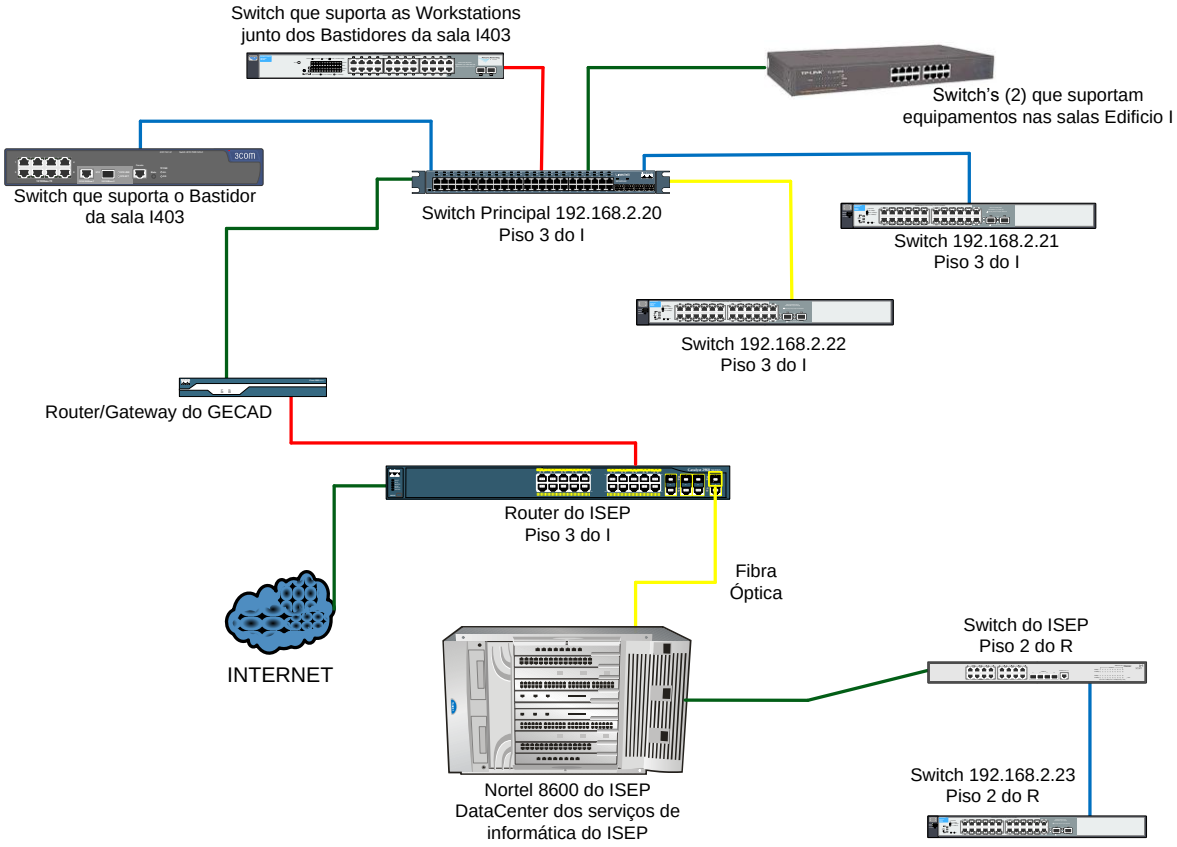


Figura 36 – Diagrama Equipamento Activo de Rede

A figura pode ser dividida em dois grupos de análise. Dividindo horizontalmente o diagrama ilustrado, através do uso do router do ISEP no piso três do edifício I, na parte superior, obtemos um conjunto de equipamentos todos conectados ao *switch* principal do GECAD, com o IP "192.168.2.20". Este *switch* é o ponto comum e ganha especial relevância quando avaliamos as

conexões de toda a rede, uma vez que, existindo uma falha neste aparelho, as comunicações podem ser colocadas em causa.

Excepto nos equipamentos do ISEP, como iremos analisar, não existe qualquer configuração de VLAN's adicionais em nenhum dos *router's* ou *switch's* da nossa unidade de investigação. A criação de VLAN's de forma a segmentar a rede e a isolar certos elementos da mesma, ou das salas de trabalho por completo foi, para já, posta de parte. Se considerarmos a disposição física apresentada na secção 4.3, quer das mesas de trabalho, quer das calhas técnicas em cada uma das salas, e se adicionalmente acrescentarmos aquilo que tem vindo a ser explicado nesta mesma secção, chegamos à conclusão que existem desvantagens na criação de uma ou mais VLAN's, ou mesmo incapacidade para tal. As características e capacidades do *switch* principal (com algumas limitações em termos de *layer* 3) e a componente humana têm peso nesta questão. Em algumas redes é crucial isolar salas de trabalho, por exemplo no próprio campus do ISEP e nas suas salas de aula. Não se deve colocar em causa toda a rede, quando esta é de grandes dimensões sobretudo se alguma pessoa, bem ou mal intencionada, faz modificações num router ou *switch* existente nas salas. Basta um cabo de rede mal ligado para a rede entrar em *loop* podendo conduzir à paragem da mesma.

Do ponto de vista físico, notamos anteriormente que todas as salas da unidade de investigação possuem as calhas técnicas apenas lateralmente, e do lado das portas esta é mais curta e apresenta menores pontos de energia e de rede. Este é um grande condicionalismo quando pensamos em acrescentar equipamentos activos nas próprias salas. Além disso, se criarmos VLAN's a isolar cada uma das salas, por exemplo, é necessário um aumento de equipamento activo capaz de lidar com este tipo de situações, que não existe actualmente, nomeadamente *switch's* com capacidades plenas a nível de *layer* 3.

A solução que subsiste na rede passa por ter os dois *switch's* com os IP's "192.168.2.21" e "192.168.2.22", colocados fisicamente na sala I307 e conectados directamente com um cabo UTP "CAT5e" ao *switch* principal do GECAD. Estes três equipamentos permitem gestão, isto é, não se limitam a estender a capacidade de ligação de mais clientes, também facultam um GUI através de um browser para configurar vários parâmetros, para além da possibilidade de o fazer por telnet ou ligação a uma porta COM de um PC. Os três *switch's* agregam os elementos de rede existentes nas instalações do GECAD no edifício I, através de comunicação entre as suas portas e as tomadas de rede presentes nas salas, com um caminho de dados construído desde o piso três ao piso quatro. Como os pontos de rede algumas vezes são insuficientes ou encontram-se fora de alcance, nas salas de trabalho existem alguns *switch's* de dezasseis portas 10/100/1000 Mbps, sem IP ou qualquer tipo de gestão, como elemento auxiliar.

Ainda na parte superior da figura 36, estão dois *switch's* sem gestão ou IP. O primeiro, com apenas oito portas 10/100/1000 Mbps suportando todos os elementos de rede que existem no bastidor já descrito com o auxílio da figura 33. O segundo, com vinte e quatro portas 10/100/1000 Mbps, também sem gestão, que fisicamente ao lado deste bastidor ajuda a conectar todas as Workstations e *Desktops* que possuem serviços importantes para determinados projectos. Resta apenas referir o Router/Gateway Cisco 1801 com o IP "192.168.2.1", quatro portas para ligação à LAN e duas portas WAN. Este router permite, entre outros serviços passíveis de configurar, encaminhamento de pacotes para o meio exterior (conexão WAN), para que se possa fornecer aos utilizadores o acesso à internet. É utilizado também para permitir acesso por VPN à nossa rede através de qualquer ponto na internet, com a coadjuvação do servidor "GECADCORE", que valida as credenciais enviadas por todos aqueles que necessitarem de aceder a determinados recursos a qualquer momento do dia, fora do ambiente físico do GECAD.

Abordando agora a parte inferior da figura 36, separada pelo router do ISEP da Cisco que é o ponto de partida que nos permite ter comunicação com o exterior, é de notar que é também através dele que se inicia a interligação com o edifício R. Numa sala própria para o efeito, num edifício adjacente ao R, o ISEP possui o cerne da sua infra-estrutura de rede, no qual se encontra um *Router* "Nortel 8600", ligado ao Cisco do edifício I através de fibra óptica. Tanto num router como no outro, foi reservada uma porta com uma VLAN para a nossa gama de rede "192.168.2.0", de forma a utilizar os dois equipamentos para comunicação entre os dois edifícios, mas sempre internamente na rede do GECAD. De seguida temos a comunicação entre o "Nortel" do ISEP e um *switch* que existe já nas nossas instalações no segundo piso do edifício R, também ele com uma VLAN específica da nossa gama de endereços. Um último "salto" é dado até que se atinge o ponto final neste caminho de dados, isto é, o *switch* do GECAD com o IP "192.168.2.23" com vinte e quatro portas 10/100/1000 Mbps e capacidade de gestão a nível de *layer* 2.

4.4.2 Principais Servidores e Respectivos Serviços

Após uma análise em que foi detalhada a rede em termos de equipamento activo, é necessário fazer um levantamento daquilo que existe em termos de elementos de rede, tais como servidores, workstations, ou desktops mais poderosos e respectivos serviços alojados. Já foi feita uma breve descrição de alguns, devido à sua localização no bastidor aplicacional descrito na secção anterior e que se encontra na sala I403. No entanto, devem ser encontrados e descritos de forma detalhada todos os elementos importantes para o normal funcionamento da rede, de forma a identificar requisitos em termos de gestão nesta área.

Não nos referimos apenas a controladores de domínio acrescentando um ou outro servidor importante, uma vez que, dada a natureza do trabalho desenvolvido no GECAD, todos os projectos são diferentes, têm as suas especificações e acima de tudo, possuem os seus servidores com diferentes serviços alocados a cada um. Para além destas particularidades, deve-se ter em conta aquilo que muitos dos projectos têm em comum, em termos de consumo de recursos, ou seja, existem por exemplo servidores aplicacionais no GECAD voltados para a computação massiva, logo é de esperar que esses computadores estejam a ser usados por diferentes pessoas, com diferentes programas de simulação, ou outros, a serem utilizados. A próxima figura ilustra a rede do GECAD sob outro ponto de vista, contendo tudo aquilo que é objecto de atenção por parte de quem administra a rede e que ainda não foi analisado detalhadamente.

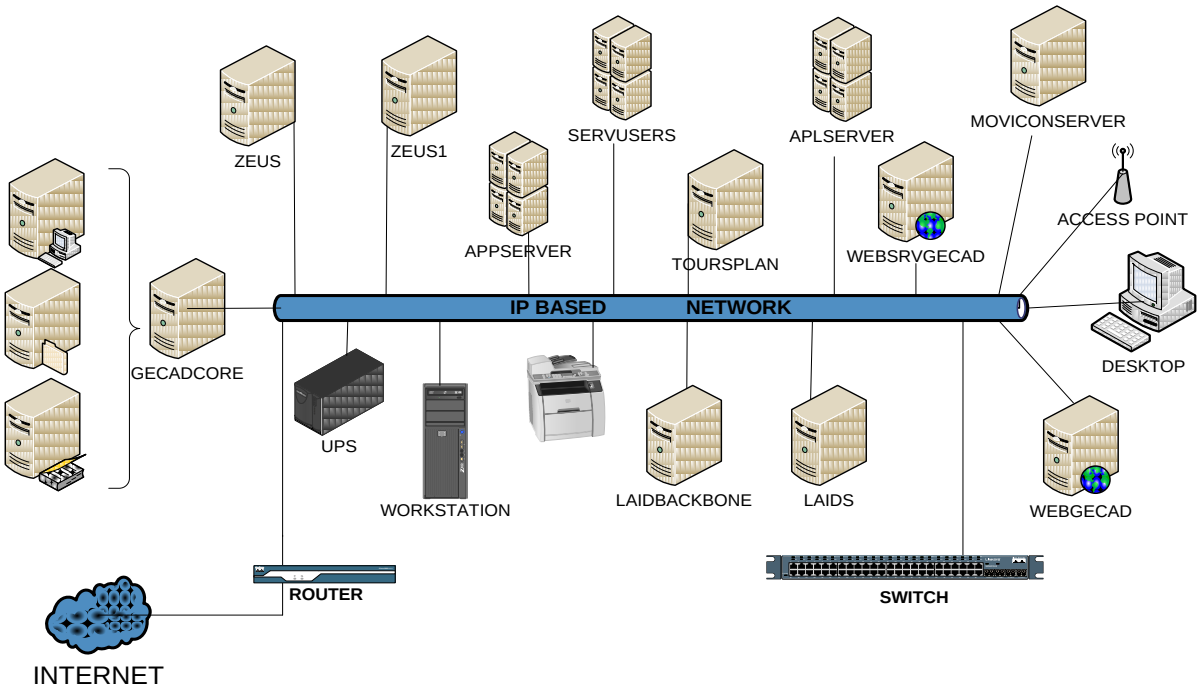


Figura 37 – Diagrama Equipamentos na Rede GECAD

O GECAD está dotado de dois controladores de domínio, como já foi referido na secção 4.2, sendo estes os elementos de rede com o nome "GECADCORE" e "ZEUS". Na figura anterior podemos observar que o servidor "GECADCORE" acumula três funções principais: serviços primordiais de *Active Directory*, servidor de ficheiros e ainda serviços de gestão, uma vez que é nesta máquina que se localiza o centro da monitorização da rede da unidade de investigação. Deve ainda ser feita referência ao facto de possuir ainda as funções de servidor de DNS e DHCP, sendo responsável pela cooperação dada ao router de forma a permitir ligações por VPN seguras. O seu parceiro de replicação de *Active Directory*, de nome "ZEUS", encontra-se em fase de despromoção da maioria dos serviços que albergava, estando no entanto, em pleno funcionamento como segundo servidor

de DNS e servidor de ficheiros, sendo responsável por alojar as pastas de rede dos utilizadores (os quais serão migrados no decorrer deste trabalho para o GECADCORE).

Segue-se um terceiro servidor de nome "ZEUS1", antigo controlador de domínio secundário, posição agora ocupada pelo "ZEUS", sendo bastante utilizado em funções de servidor de ficheiros, para áreas temporárias, isto é, existe um conjunto de espaço disponível para os utilizadores usarem da forma que entenderem, nomeadamente bastante eficiente para troca de ficheiros entre eles. Também esta área será migrada para o "GECADCORE", de forma a libertar esta máquina para funções mais ligadas a projectos que necessitem de recursos mais eficientes ao nível do processamento.

O GECAD está ainda munido de três servidores aplicacionais com propósitos de computação massiva: "APPSERVER", "SERVUSERS" e "APLSERVER". Estes servidores destacam-se dos restantes pelo simples facto de conterem todo o tipo de software que quase todos os investigadores necessitam para trabalhar. Quando surgem necessidades especiais de processamento, normalmente associadas a simulações em programas como MATLAB, GAMS, PSCAD, LINGO, POWERWORLD, CLEMENTINE, entre outros, estas máquinas são bastante poderosas e podem ser usadas de forma concorrente por vários utilizadores. Para além dos programas citados, existe a necessidade de disponibilizar outras ferramentas mais comuns, como por exemplo, editores de texto, que podem inclusive ser utilizadas em simultâneo. Estamos a falar de servidores em que cada um contém dois processadores com oito cores a 3.0GHz, 12MB de cache L2, 6GB RAM no "APLSERVER", 4GB no "APPSERVER" e 8GB no "SERVUSERS". Estas três máquinas possuem discos SAS a 10000 rotações à excepção do SERVUSERS que contém discos SATA de 2,5''. Num futuro próximo irá ser adicionado um servidor com capacidades semelhantes mas com dois processadores da AMD, os OPTERON, seis discos 300GB SAS a 10000 rotações e 16GB RAM. Como ainda não se encontra totalmente configurado não aparece ainda no diagrama de rede da figura anterior.

"MOVICONSERVER" e "TOURSPLAN" são servidores dedicados a dois projectos independentes, onde ambos subsistem com algumas aplicações específicas dos próprios projectos, em que o segundo baseia-se em serviços WEB disponíveis ao público, resultantes da investigação realizada e o primeiro está ligado aos sistemas de energia, fornecendo a alguns utilizadores internos a capacidade de trabalhar num software apropriado para os fins pretendidos.

"WEBGECAD" e "WEBSRVGECAD", como os nomes indicam, são responsáveis por vários serviços ligados à WEB, entre os quais HTTP, HTTPS, e FTP. O primeiro servidor aloja e disponibiliza ao público o site do GECAD, encontrando-se vários conteúdos adicionais apenas para os investigadores que se encontram ligados à nossa unidade de investigação. Muitos projectos

possuem aplicações voltadas para a WEB, que foram desenvolvidas ao longo do tempo e que são alojadas neste servidor. O "WEBSRVGECAD" possui algumas funcionalidades idênticas às do servidor anterior, mas a uma escala menor.

Os últimos dois servidores que aparecem no diagrama da figura 37 são o "LAIDS" e o "LAIDBACKBONE". O GECAD possui, ainda, uma linha de investigação em sistemas inteligentes, sendo que dentro desta, existem projectos ligados aos ambientes inteligentes de apoio à decisão. Vários conteúdos resultantes da investigação realizada estão disponibilizados nestes servidores através de serviços http e serviços multimédia. Para além de todas as aplicações desenvolvidas no âmbito desta linha de investigação, há que considerar ainda as várias ferramentas necessárias para o desenvolvimento das mesmas.

O *router* e o *switch* que se encontram representados servem apenas para lembrar a existência de equipamento activo que já foi descrito de forma pormenorizada na secção 4.4.1, pelo que são necessárias funções de gestão que serão apresentadas e discutidas neste capítulo e posteriormente no capítulo 5.

As impressoras e as UPS's surgem como elementos muito importantes para qualquer utilizador do GECAD, quer pelas necessidades de impressão gerais, quer pela protecção e disponibilidade (UPS) que é oferecida a numerosos elementos de rede, principalmente a servidores e ao equipamento activo. Como tal, muita informação útil para gestão pode ser retirada das cinco impressoras de rede que o GECAD tem, assim como das UPS's, que conectadas à rede, podem ser configuradas para ajudar um administrador a perceber as possíveis falhas eléctricas existentes, ou passíveis de virem a ocorrer.

Muitos dos investigadores do GECAD, para além dos servidores dos respectivos projectos ou da unidade em geral, possuem Desktops ou mesmo Workstations (máquinas mais poderosas com capacidades mais similares às de um servidor). Nesse sentido poderá ser no futuro necessário incorporar muitos destes elementos de rede, e respectivos serviços, e processos, no âmbito da gestão, o que não foi considerado neste trabalho. Este facto prende-se com os objectivos subjacentes ao trabalho. No entanto, a proposta apresentada neste trabalho com base em mobilidade irá contribuir para ultrapassar este problema no futuro.

De referir ainda, a forma como está organizada toda a estrutura que é responsável por manter os ficheiros dos vários utilizadores seguros, disponíveis 24 sobre 24 horas de forma rápida e eficiente. O servidor ZEUS contem várias áreas para uso dos utilizadores, onde existe um controlo de quotas rigoroso em todas elas. Quando um utilizador acede à sua área de ficheiros na rede do GECAD possui pelo menos 5GB disponíveis, podendo de forma justificada ser alargado esse espaço. As

áreas de trabalho estão assentes em discos em RAID 1 para prevenir falhas mecânicas, acrescendo um serviço adicional que permite possuir versões anteriores dos ficheiros. Permitindo que qualquer ficheiro tenha, a cada doze horas, uma versão anterior salvaguardada. Também no ZEUS é possível fazer uso de áreas temporárias criadas essencialmente pelos próprios investigadores, facilitando as trocas de ficheiros entre eles e também o armazenamento momentâneo, se necessário.

Para além da área de 5GB (ou mais) que todos os utilizadores possuem, também nos servidores aplicacionais (APLSERVER, APPSERVER e SERVUSERS) cada utilizador é proprietário de espaço. Também aqui surge a existência de quotas atribuídas de acordo com as necessidades, sendo que todos os servidores aplicacionais mantêm os dados em discos montados em RAID, algumas das configurações com discos *spare*, que entram em funcionamento aquando da falha mecânica de um dos parceiros.

Relativamente a Backups, todos os dias, a partir das 00h00 até às 08h00 são realizadas cópias integrais das partições do sistema de todos os servidores mais importantes, para além, naturalmente, de todos os dados dos utilizadores do GECAD. Estas cópias são colocadas num disco externo conectado por USB ao "APPSERVER", à excepção da cópia dos dados das áreas localizadas no ZEUS, que estão a ser alojadas no GECADCORE. De forma regular, a unidade central que recebe as cópias de segurança é copiada integralmente para outro disco externo com mais capacidade, que não fica nas instalações do GECAD, salvaguardando os dados em caso de um acidente ou falha que comprometa tudo o que se encontra a nível de dados armazenados em servidores nesta sala.

Após ter sido feito o levantamento do estado da arte e o estudo pormenorizado da rede do GECAD, que incluiu a identificação minuciosa de todos os serviços, processos, aplicações e elementos de rede a serem monitorizados decidiu-se utilizar uma ferramenta conhecida dos administradores de rede, o *Network Mapper* (NMAP - http://nmap.org/). Esta ferramenta permitiu efectuar um levantamento completo da rede que inclui todos os elementos de rede importantes, portas abertas, serviços a funcionar e respectivas versões na maioria dos casos. Assim, para além do objectivo fulcral, tornou-se possível identificar falhas de segurança com vista a serem corrigidas, permitindo melhorar a rede, antes de se implementar uma solução de gestão. Na figura 38 é possível ver alguns dos resultados do levantamento executado na *subnet* "192.168.2.0". Com um simples *click* num dos *hosts* à esquerda podemos encontrar as portas, protocolo, estado, serviço e versão correspondentes. Sem sequer aceder directamente a algum dos *hosts* conseguimos identificar o essencial. Apesar de tudo e como é natural, para assegurar de que tudo o que é essencial era monitorizado, mais tarde todos os elementos de rede foram analisados da forma mais rigorosa.

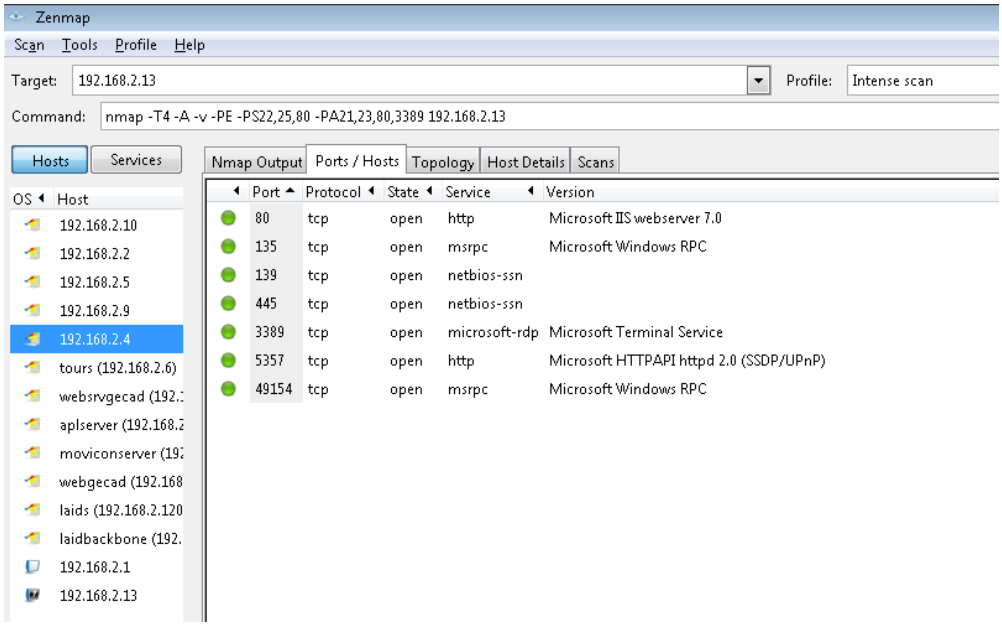


Figura 38 – Varrimento do NMAP

A figura 39 permite apresentar os vários *hosts*, assim como um em específico, com o IP, endereço físico, Sistema Operativo, serviços e traçar rotas.

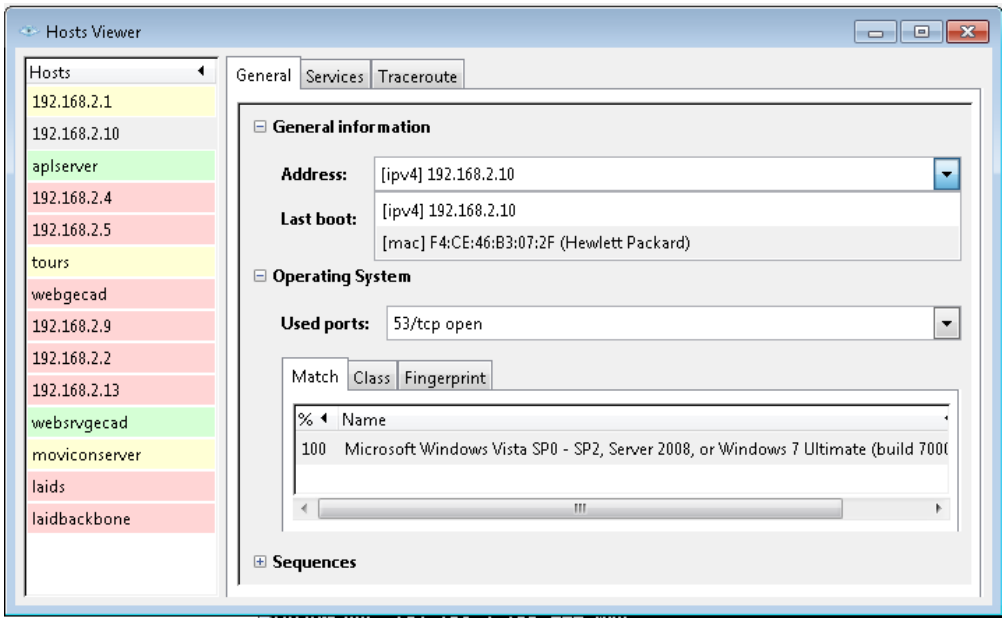


Figura 39 – Vista dos vários Hosts com o NMAP

De facto é notável e assustadora (se virmos a questão pelo lado da segurança), a quantidade de detalhes que foram obtidos em poucos minutos sobre todos os elementos de rede. Na figura 40 encontra-se isolado um dos *hosts*, com informação que vai desde o estado, até ao número de portas

abertas, o tempo desde que se encontra ligado, a última vez que sofreu um boot, endereços IP e físico, Sistema Operativo, entre outros detalhes.

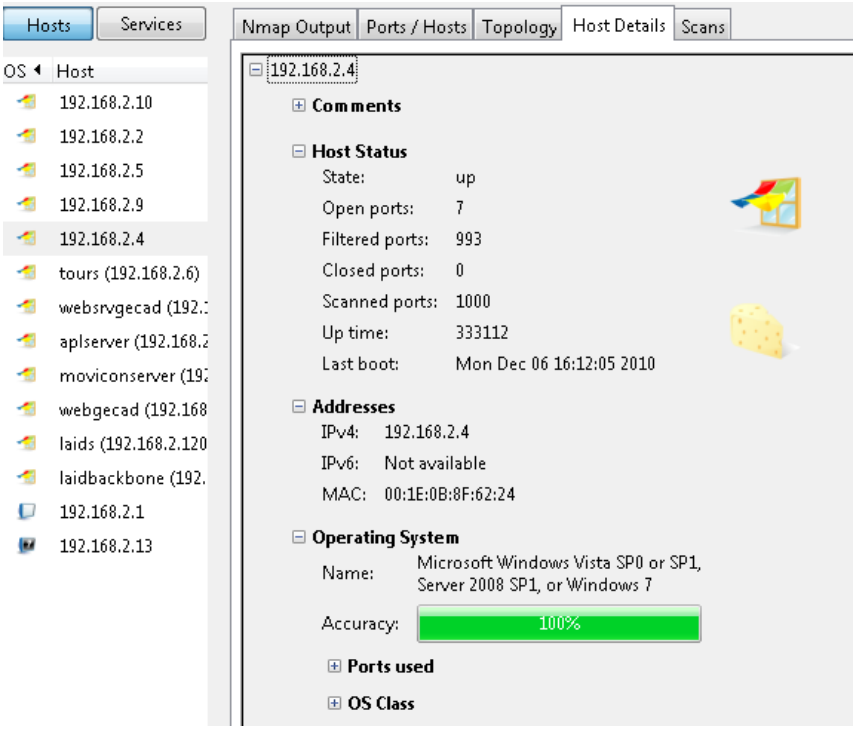


Figura 40 – Detalhes de um Host no NMAP

Com a existência destes dados foi possível identificar o conjunto de elementos a monitorizar, a importância de cada elemento, assim como identificar a relevância de cada um dos serviços disponibilizados.

4.5 Ferramenta de Gestão Actual

4.5.1 Introdução

A gestão da rede do GECAD era feita através de uma ferramenta de monitorização. No entanto, as necessidades actuais implicam que exista uma solução de gestão mais sofisticada.

A rede do GECAD necessitava de uma solução de gestão que permitisse garantir o pleno funcionamento da rede descrita. A partir do momento em que este trabalho começou, tornou-se necessário encontrar uma ferramenta robusta, que suportasse o paradigma tradicional, e que mais tarde facilitasse a coexistência com um segundo paradigma, os Agentes móveis. Para tal, foi necessário encontrar uma ferramenta que permitisse aceder às interfaces e ao código fonte, uma vez que se pretendia adicionar o paradigma dos Agentes móveis. A ferramenta eleita é o "GroundWork Monitor". Trata-se de uma ferramenta *Open Source* que integra vários outros projectos *Open*

Source, como por exemplo o "Nagios", "Apache" e "NMAP". O motor da monitorização é o "Nagios" e todas estas ferramentas estão inseridas num Sistema Operativo Linux [68-70].

4.5.2 Ferramenta GroundWork Monitor – Características

A solução de gestão "GroundWork Monitor" utiliza uma arquitectura aberta, baseada em standards, de forma a alcançar uma plataforma de monitorização de sistemas e rede estável, bastante funcional, escalável e comprovada a todos os níveis. Através da existência de várias soluções conhecidas *Open Source*, de monitorização de rede, foi conseguida uma harmoniosa combinação de todas estas ferramentas, incluindo uma configuração de armazenamento dos dados, juntando-as numa solução global, capaz de ser instalada e configurada mesmo em empresas que possuam redes vastas, ao nível das WAN. Uma das grandes razões que fez com que esta ferramenta fosse a eleita prende-se com o facto de ser possível moldá-la ao nosso gosto ou necessidades específicas da nossa rede. A figura 41 fornece uma visão geral da arquitectura desta solução, dividindo-a em três grandes camadas: portal, normalização e instrumentação. Estas camadas fornecem a estrutura que permite uma interacção fluida entre os diversos projectos *Open Source* que estão presentes em cada uma delas [68, 71-73].

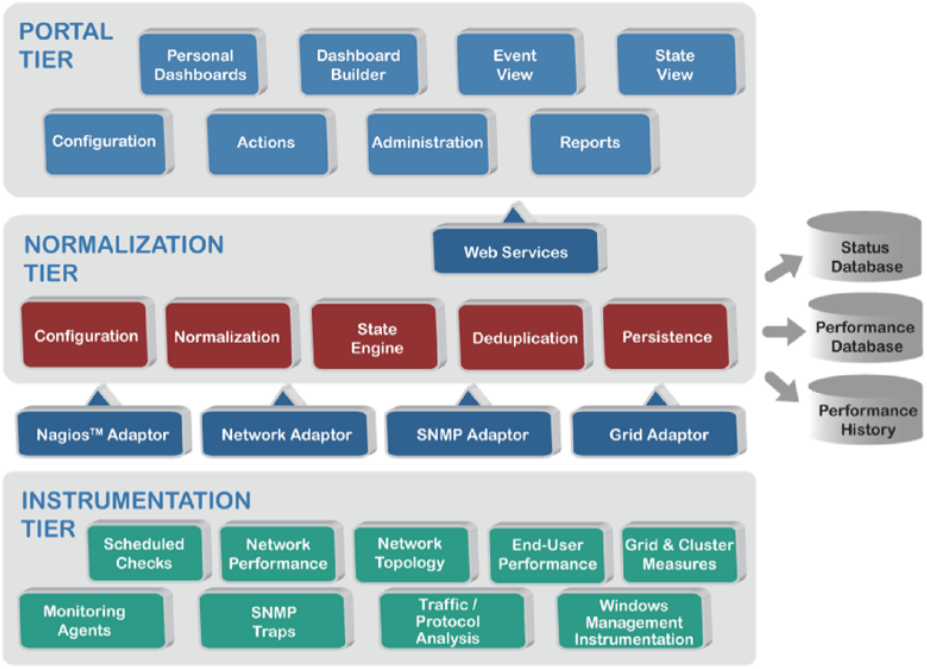


Figura 41 – Visão Geral GroundWork Monitor (retirada de [71])

Deve ser realçado um aspecto muito importante: existem vários projectos *Open Source* que apenas são incluídos na solução de gestão numa versão *Enterprise*, paga por licença, sendo que nada nos impede de moldar a nossa própria solução de gestão com esses mesmos projectos. Para além disto,

podem surgir outros benefícios naturais, que resultam da compra deste software, distinguindo os clientes que fazem uso de uma solução paga, daqueles que a usam sem custos.

De seguida iremos analisar com detalhe as três camadas que a constituem. **A camada de instrumentação** contém todos os aspectos relacionados com serviços de recolha de dados, de maneira a assegurar uma cobertura alargada sobre hardware, Sistemas Operativos, rede, aplicações e serviços. Esta camada é altamente flexível e ajustável para podermos fazer acertos que vão de encontro às nossas necessidades específicas. Vamos de seguida analisar e fazer uma breve descrição do que cada bloco representa dentro desta camada [68, 70-73]:

- ***Scheduled Checks*** permite obter dados de informação sobre disponibilidade e performance para a mais vasta selecção de elementos de rede, através do uso do motor de agendamento do "Nagios", assim como do seu sistema de plug-in;
- ***Monitoring Agents*** contém Agentes e *proxy's* baseados em ferramentas de monitorização que incluem o "NRPE" e "NSClient", que servem para obter dados de gestão de máquinas com Sistemas Operativos da Microsoft. Para além destes encontramos também o "GroundWork Linux Child" e o "GroundWork Windows Child Server" que permitem uma monitorização de ambientes distribuídos, fornecendo uma interface capaz de facultar o controlo de cada um dos servidores de gestão de forma centralizada, sem ser necessário administrá-los de forma individual;
- ***Network Performance*** é responsável por conseguir obter dados de performance de rede utilizando para tal o projecto "Cacti". Este projecto não vem implementado nalgumas das soluções mas é possível configurá-lo, pois também é *Open Source*, com o grande objectivo de guardar toda a informação relevante para criar gráficos e preenchê-los com dados existentes numa base de dados "MySQL". A interface é completamente orientada a PHP e existe ainda suporte a SNMP, para todos aqueles que preferem ou estão habituados a criar gráficos de tráfego com MRTG (*Multi Router Traffic Grapher*);
- ***Network Traffic Analysis*** possui um *add-on* muito bom que fornece análises detalhadas de tráfego usando *NetFlow* e *sFlow*;
- ***Network Discovery*** através do *add-on* do projecto "NeDi", descobre a topologia da rede e os elementos a esta conectada. Este tipo de abordagem automatizada oferece várias vantagens, nomeadamente o facto de várias vertentes no sistema serem automaticamente configuradas;
- ***SNMP Traps*** é o módulo que torna capaz a execução de *polling* por SNMP e processamento das chamadas *traps* aos elementos de rede e respectivos Sistemas Operativos;

- ***End-user Performance*** corresponde a uma parte do sistema "GroundWork Monitor" cujo objectivo é proporcionar a cooperação com software proprietário da "e-Valid", mais concretamente o "iScoute", com o objectivo de seguir os processos de negócio e a performance das aplicações a partir da perspectiva do utilizador final;
- ***Windows Management Instrumentation*** consegue providenciar uma cobertura profunda a nível de monitorização para Desktops, Servidores e aplicações da plataforma Windows;
- ***Grids and Clusters*** é um módulo colocado na solução de gestão "GroundWork Monitor" para que esta possa servir de *add-on* a uma outra conhecida ferramenta *Open Source* de gestão, o "Ganglia". Este está mais especializado em monitorização de clusters e a combinação dos dois sistemas permite recolher e analisar estatísticas sobre a performance a vários níveis no Sistema Operativo dos elementos de rede.

A camada de Normalização coloca todos os formatos e representações de dados de diferentes fontes num standard, permitindo aos motores de estado e persistência a realização de tomadas de decisão e inferências baseados em informação originária de várias proveniências. Na figura 41 estão representados alguns dos blocos mais importantes deste portal [68, 70-73]:

- ***State Engine*** providencia um agendamento eficiente, lógica de repetição e supressão de mensagens ilegítimas baseando-se no ambiente de monitorização previamente configurado. Tem de suportar tomada de decisões de alto nível combinando os eventos que chegam de diferentes elementos de rede com as diferentes alturas ou tempos em que estes chegam. Daqui resultam mudanças lógicas de estado que estão ao alcance e podem ser verificadas pelo staff operacional;
- ***De-Duplication*** consolida informação relacionada, une e sumariza resultados de medições efectuadas ao nível da performance para a construção de relatórios e tendências históricas, assegurando que a aplicação consiga ser capaz de fazer a monitorização de milhares de elementos de rede;
- ***Persistence Engine*** é construído sobre "MySQL" e "RRDs" (*Round-Robin Databases*) facilitando armazenamento para informação de estados, medições de performance actuais e informações de performance históricas.

A camada de nome Portal fornece autenticação e controlo de acessos por *role* ou funções, isto é, mediante as permissões de determinado utilizador, uma vez que pode ser mais do que um a administrar todo o sistema, é dado o acesso à interface da solução de gestão "GroundWork Monitor". A interface contém diversas vistas sobre a rede monitorizada: baseada em estados, baseada em eventos, em relatórios e oferecendo ainda a capacidade de ajustar e configurar as diferentes capacidades do portal para cada utilizador. Esta camada facilita também o acesso às

interfaces de utilizador dos projectos *Open Source* subjacentes à solução de gestão no seu todo, reduzindo assim a necessidade de limitar aquele tipo de utilizadores que já estão familiarizados com essas ferramentas. Sobressaem, no entanto, na figura 41 os seguintes módulos que merecem destaque [68-73]:

- ***Personal Dashboards*** permite aos utilizadores finais criar vistas personalizadas sobre informações de gestão consideradas vitais e obtidas a partir de consultas, vistas sobre elementos de rede ou até vistas de aplicações específicas organizadas nos chamados *Dashboards*. Podemos ver um *Dashboard* que contém vários elementos importantes na figura 42;
- ***Event View*** permite a consulta dos eventos recentes de forma sequencial no tempo com grandes capacidades de filtragem. Encontramos vistas detalhadas assim como uma API introduzida na solução de gestão de forma a ser possível a integração de software de gestão dedicado a incidentes ou *ticket management*;
- ***State View*** correlaciona o estado actual e performance de servidores, aplicações, serviços e restantes elementos de rede com a capacidade de receber indicações e estar ciente dos problemas existentes, efectuar análises para despiste e reagendar as verificações. Tudo isto é possível com a interacção com uma única página;
- ***Configuration*** como o próprio nome indica, corresponde a todo o tipo de configurações que podem ser carregadas para o sistema através de um único interface baseado em Web. É possível aplicá-las manualmente pelos administradores, programar essas mesmas configurações usando a API, ou obtê-las de forma automática com ferramentas que possuam regras para *auto discovery*;
- ***Administration*** é um módulo que está directamente voltado para assegurar um acesso apropriado e controlado por parte dos utilizadores ao sistema de gestão, gerindo para cada um as suas permissões e respectivas funções. Pode ser integrado nas implementações de *Active Directory* ou outras que utilizem "LDAP". Permite ainda gerir o portal em si, ou seja, é através deste módulo que nos é facultado o acesso a toda uma panóplia de utilidades para criarmos novos *Dashboards*, modificar os existentes ou até criar um outro portal adicional desde a raiz;
- ***Reports*** inclui relatórios prontos a serem gerados e visualizados a qualquer momento, sobre a disponibilidade do sistema de gestão, dos elementos de rede, de serviços, de um conjunto de variáveis, que agrupadas, fornecem dados importantes para um planeamento eficiente das medidas a tomar no futuro. Um projecto integrado na solução é muito importante para atingir estes objectivos, que é o "BIRT Reports Designer". Trata-se de uma ferramenta

Open Source, baseada em "Eclipse" e voltada para a criação e personalização de relatórios. Na figura 48 podemos observar um exemplo das potencialidades da ferramenta. Trata-se da verificação do número de interrupções por serviço que aconteceram, num determinado período de tempo.

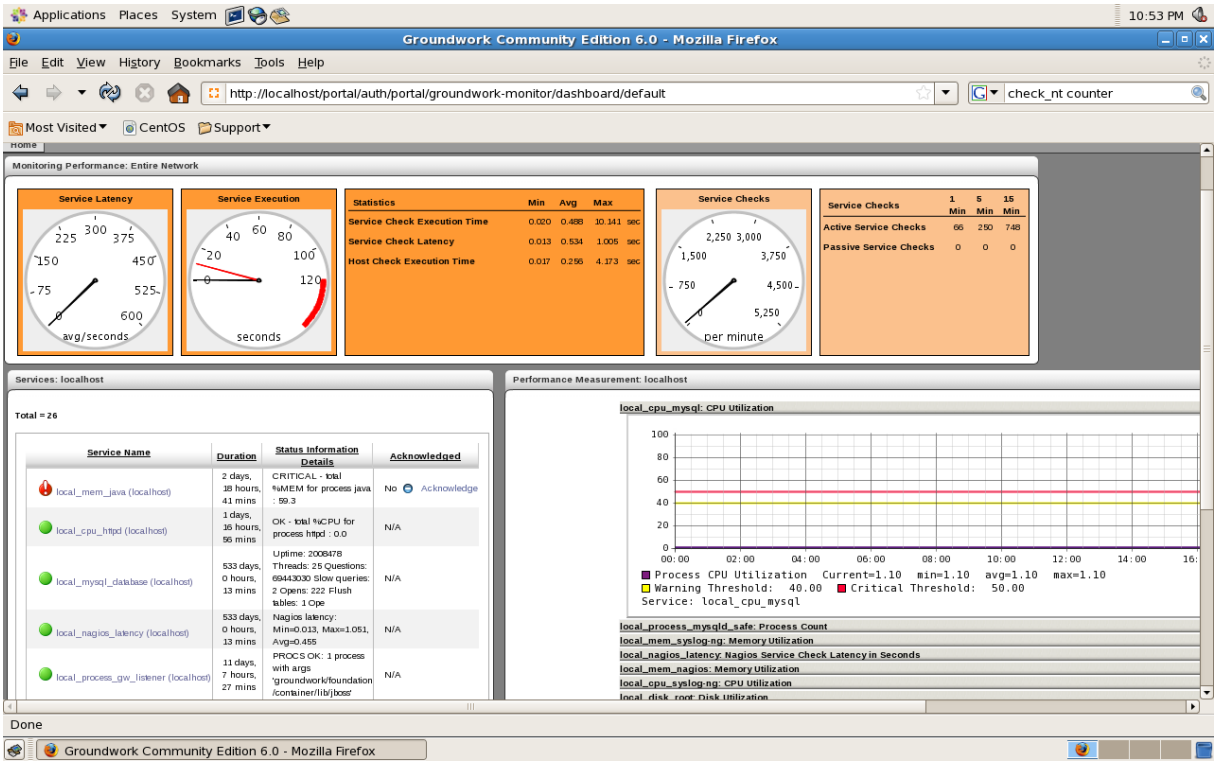


Figura 42 – Exemplo de um Dashboard

De facto, encontramos uma lista do estado de serviços de um determinado *host* à esquerda, um gráfico de performance da utilização do CPU à direita, do mesmo *host*, alguns dados sobre verificações de serviços no centro, e em cima à direita o número total de verificações de serviços ao fim de um, cinco e quinze minutos. Podemos criar novos *Dashboards* ou alterar os existentes, de forma a ir de encontro às nossas necessidades.

A próxima figura, figura 43, dá-nos uma perspectiva do ecrã principal a nível de verificação de estados na solução de gestão utilizada pelo GECAD. Todas as características mencionadas na descrição do módulo *State View* são aqui encontradas: gráficos que expressam o número total de grupos de elementos de rede num estado aceitável ou degradado, o mesmo em relação a todos os *hosts* e a todas as verificações efectuadas, assim como uma árvore do lado esquerdo com todos os elementos.

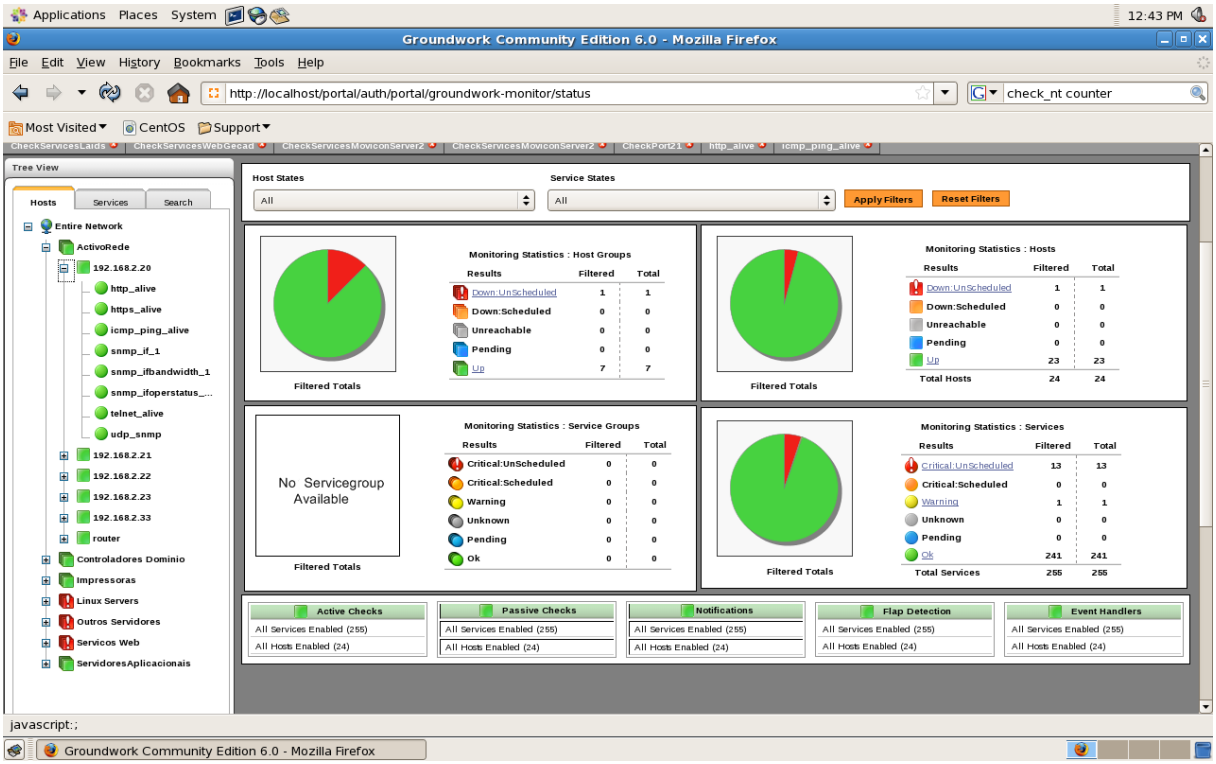


Figura 43 – Vista de Estados

Podemos escolher um dos elementos de rede, o que resulta na expansão do mesmo, mostrando todas as verificações que lhe estão atribuídas; um simples *click* adicional numa das verificações permite visualizar os seus detalhes e se necessário reagendar o *timing* da próxima tentativa de *polling* para a verificação em específico.

Na próxima figura, figura 44, vemos como a configuração do "Nagios" foi embebida numa página Web, muito mais agradável e amiga no que diz respeito às dificuldades e tempo perdido que se pode obter configurando manualmente os ficheiros do motor da gestão. Destaque para um simples *wizard* quando queremos adicionar um *host*, fora do âmbito da procura por *Auto Discovery*, assim como, é possível visualizar no lado esquerdo da figura a facilidade com que se pode efectuar a selecção de um *host* existente, dentro de um determinado grupo de *host's*, acedendo assim aos seus detalhes, e possibilitando a sua configuração.

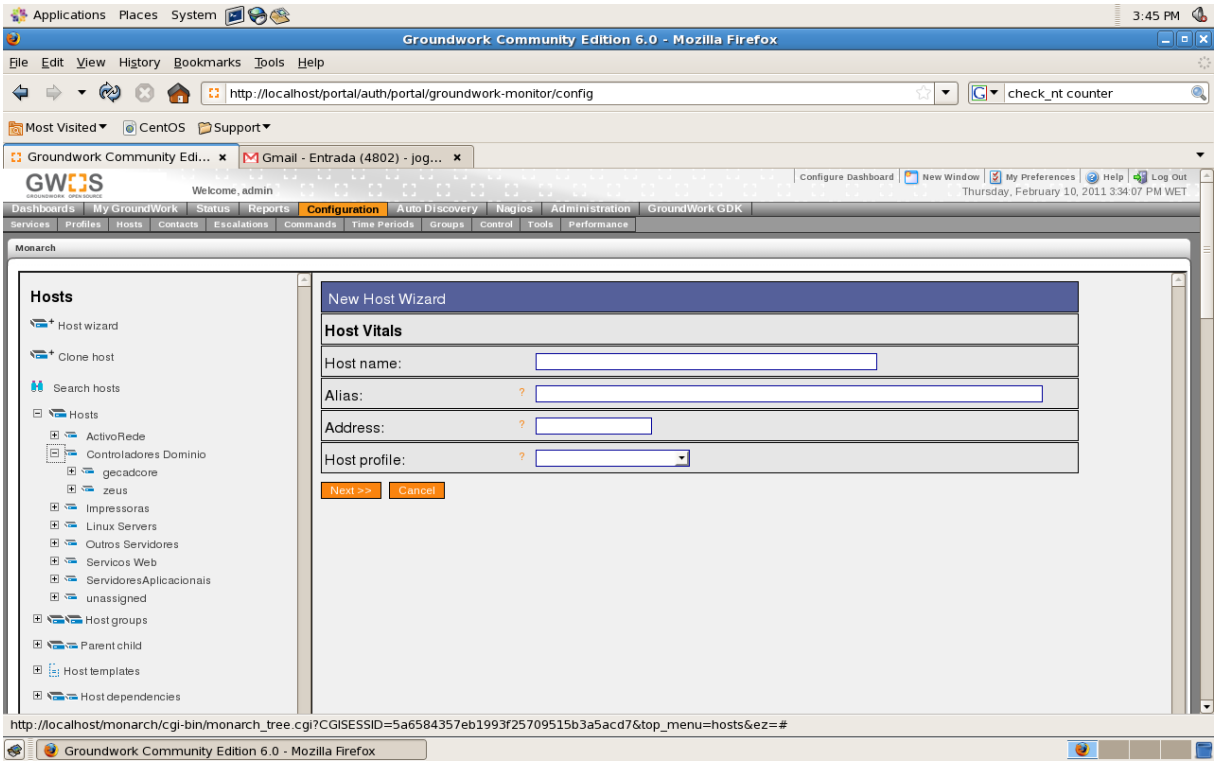


Figura 44 – Módulo de Configuração (elementos de rede)

Como podemos verificar na próxima figura, figura 45, a configuração (do módulo *Configuration*) está relacionada com os serviços. Os serviços para o "Nagios", são verificações, consultas, *pollings*, *checks*. Pelos serviços passa toda a monitorização de uma rede, e da análise cuidada da próxima figura é possível observar toda a terminologia do "Nagios", através da configuração dos respectivos ficheiros.

No "Nagios", cada serviço tem por detrás um determinado comando que permite executar a verificação que pretendemos. Um comando pode conter parâmetros, sendo que esses podem ser configurados aquando da atribuição do próprio comando a um serviço. No entanto, se necessário, podemos criar novos comandos, modificar os existentes ou eliminar os que consideramos desnecessários. Tudo isto com a mesma facilidade demonstrada nas configurações anteriores e patente na figura 46.

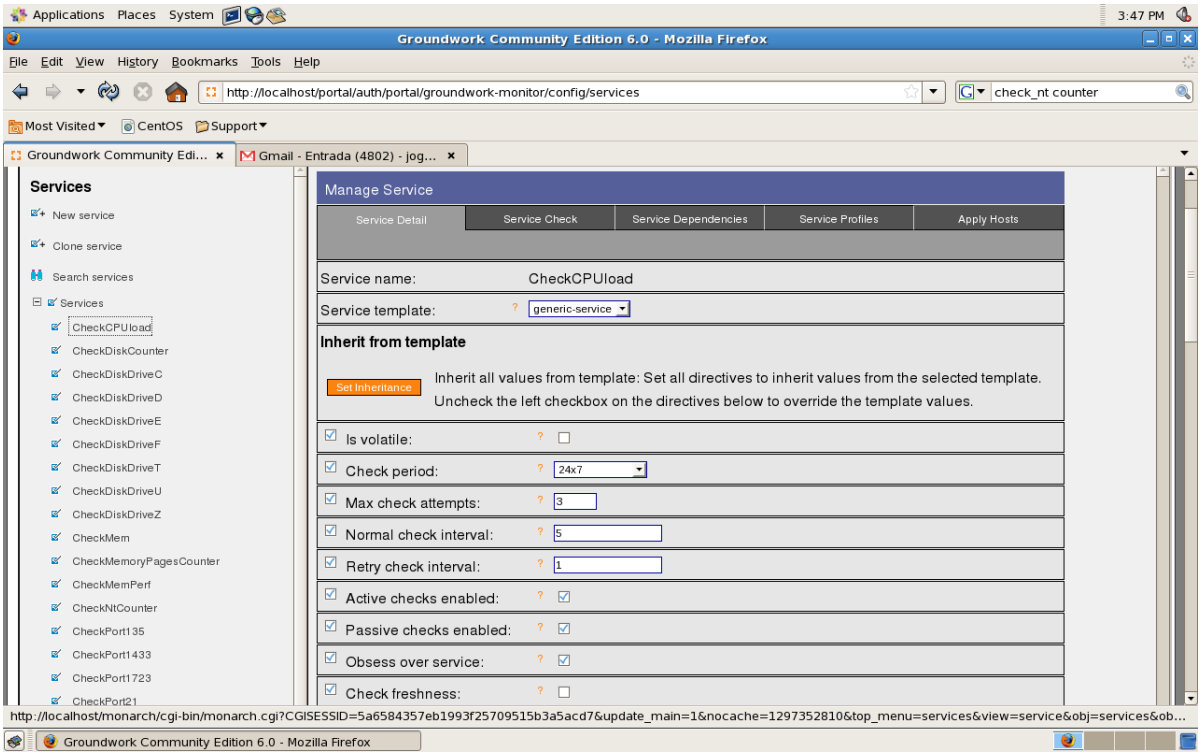


Figura 45 – Módulo de Configuração (serviços)

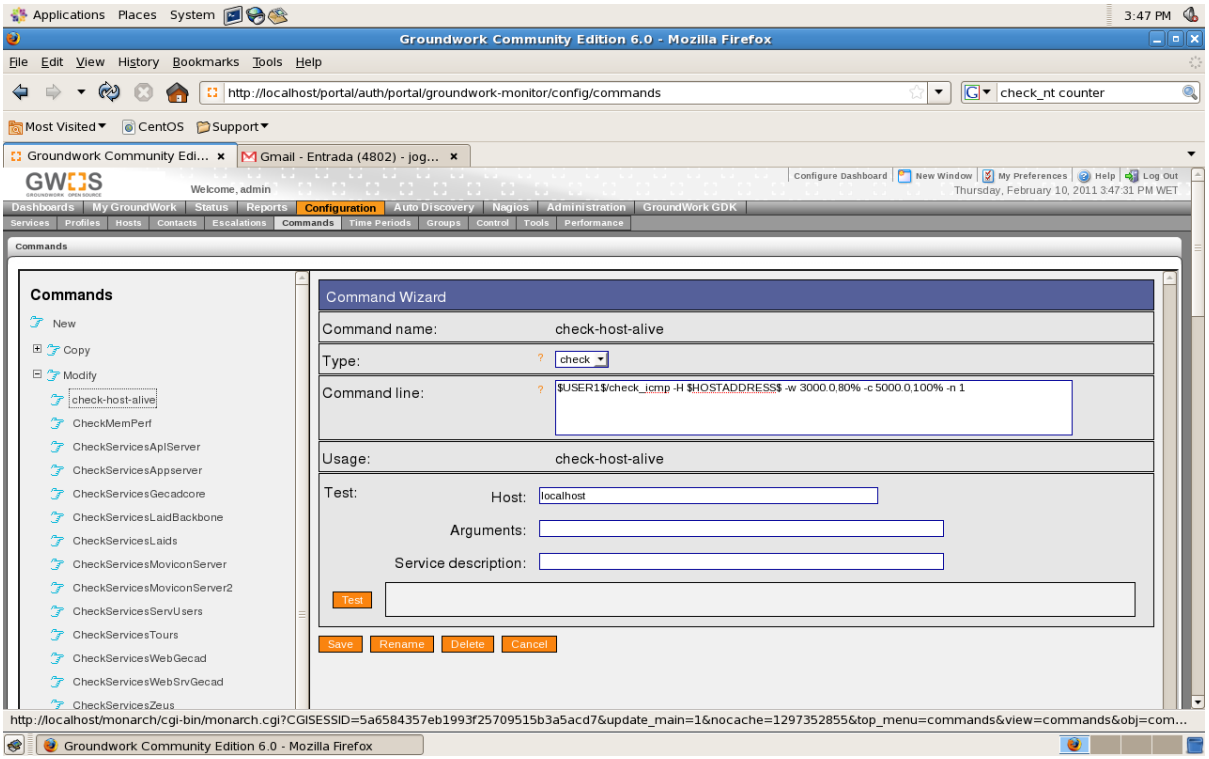


Figura 46 – Módulo de Configuração (comandos)

Convém salientar que existe a possibilidade de testar neste ambiente qualquer comando ou serviço criado, permitindo de imediato verificar se o seu comportamento é o pretendido.

Para além dos *host's*, grupos de *host's*, serviços e respectivos grupos, comandos, perfis e gráficos de performance, que são construídos a partir de dados obtidos do *polling* contínuo aos elementos de rede, até à configuração dos principais ficheiros do "Nagios" é possível editar pelo uso deste módulo e da mesma forma que os anteriores. A edição é feita através da simples navegação pelas diversas páginas Web indicadas. Estas modificações são executadas através do componente de controlo do módulo *configuration*, sendo a edição de dois ficheiros do "Nagios" a base destas funcionalidades. Os ficheiros são o "cgi.cfg" e o "nagios.cfg", onde são definidas várias configurações base para que o próprio "Nagios" possa ser compilado e executado. Como podemos ver na figura 47, à esquerda, a edição embebida na página Web correspondente a cada ficheiro pode ser seleccionada. Após estarem concluídas as configurações podemos fazer um teste à mesma, na opção "*preflight test*", que nos devolve os possíveis erros existentes na nossa configuração. Isto avalia todos os ficheiros, incluindo os de comandos, serviços, entre outros já mencionados, retornando dados importantes no caso da existência de problemas, ou retornando o OK para fazermos um *commit* à nossa configuração, isto é, compilar todos os ficheiros e reiniciar todos os serviços ligados ao "Nagios", aplicando assim as novas configurações.

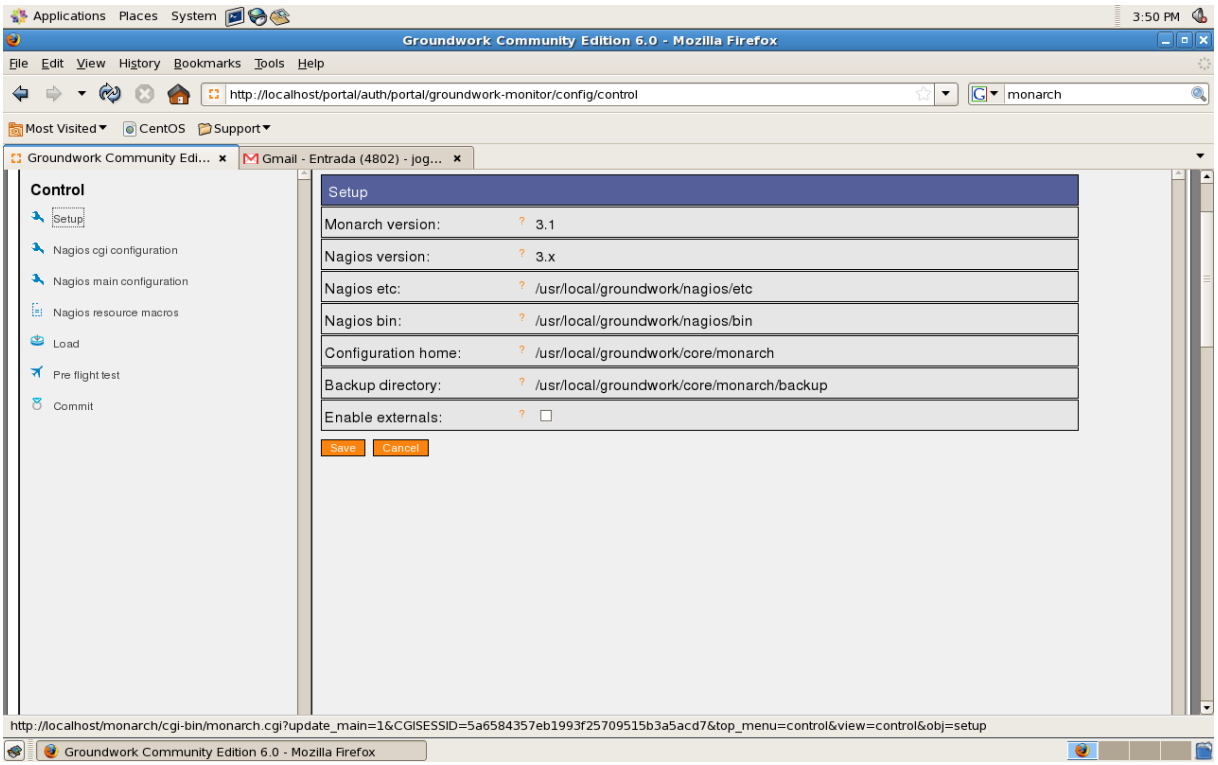


Figura 47 – Módulo de Configuração (controlo)

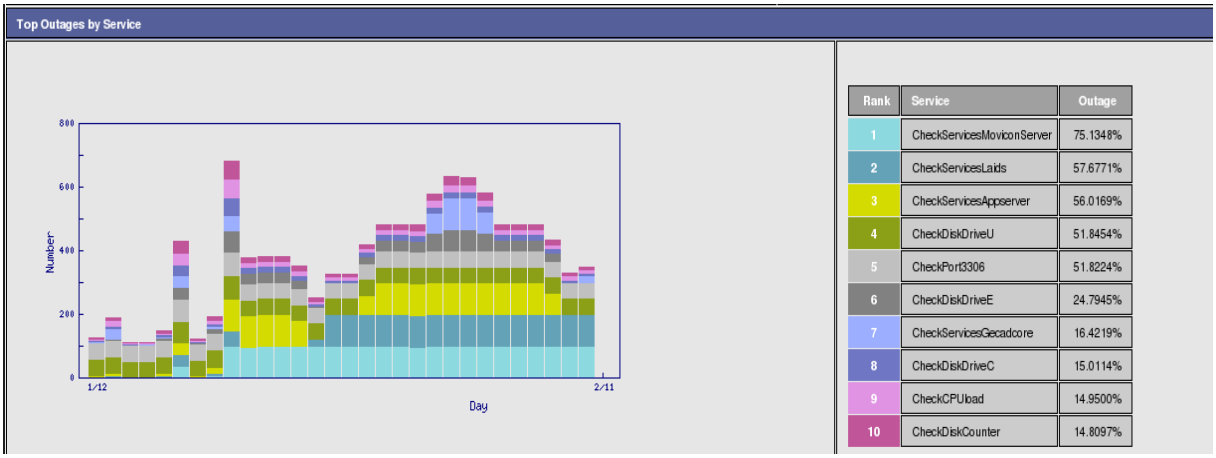


Figura 48 – Exemplo de Relatório GroundWork Monitor

Motor de Gestão

Tal como já referido, a camada Portal, entre outras características, permite o acesso às interfaces dos projectos *Open Source* associados à solução global. Um desses projectos consiste no motor da funcionalidade de monitorização, o "Nagios". Embutido no "GroundWork Monitor", tem grandes responsabilidades em todo o processo de gestão. Qualquer administrador de rede, ou utilizador, que esteja habituado à sua interface pode encontrá-la acessível tal como a conhece.

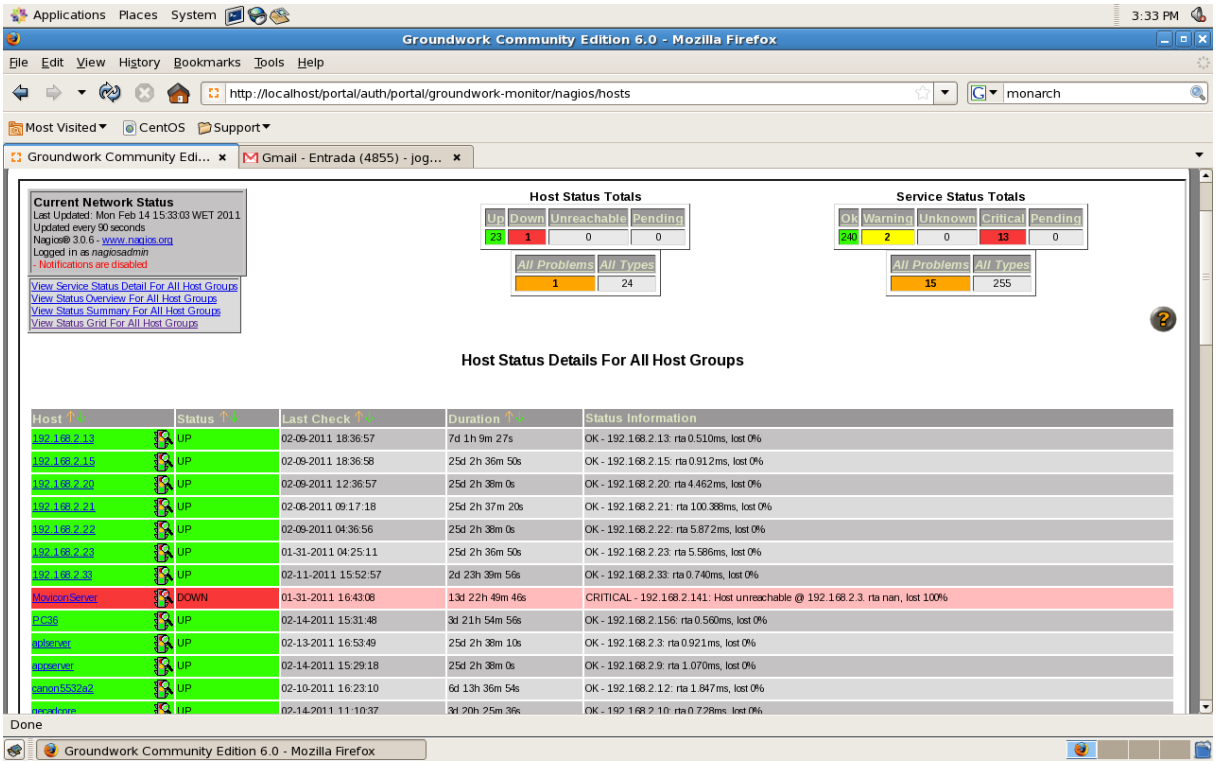


Figura 49 – Visão Geral do Nagios

O "Nagios" é uma aplicação de monitorização de rede, *Open Source*, que permite, englobado na solução de gestão "GroundWork Monitor", monitorizar *host's* e serviços, alertando quando existem problemas, assim como quando estes são resolvidos. É o ***core* da gestão**, e responsável pela monitorização dos serviços de rede, tais como, SMTP, POP3, HTTP, NNTP, ICMP, SNMP, entre outros. É através dele que conseguimos monitorizar recursos nos elementos de rede, sejam computadores ou activos de rede na maioria dos Sistemas Operativos, incluindo a plataforma Windows, amplamente usada no GECAD. Na figura 50 vemos todos os elementos de rede associados aos seus grupos, em conjunto com o seu estado [69].

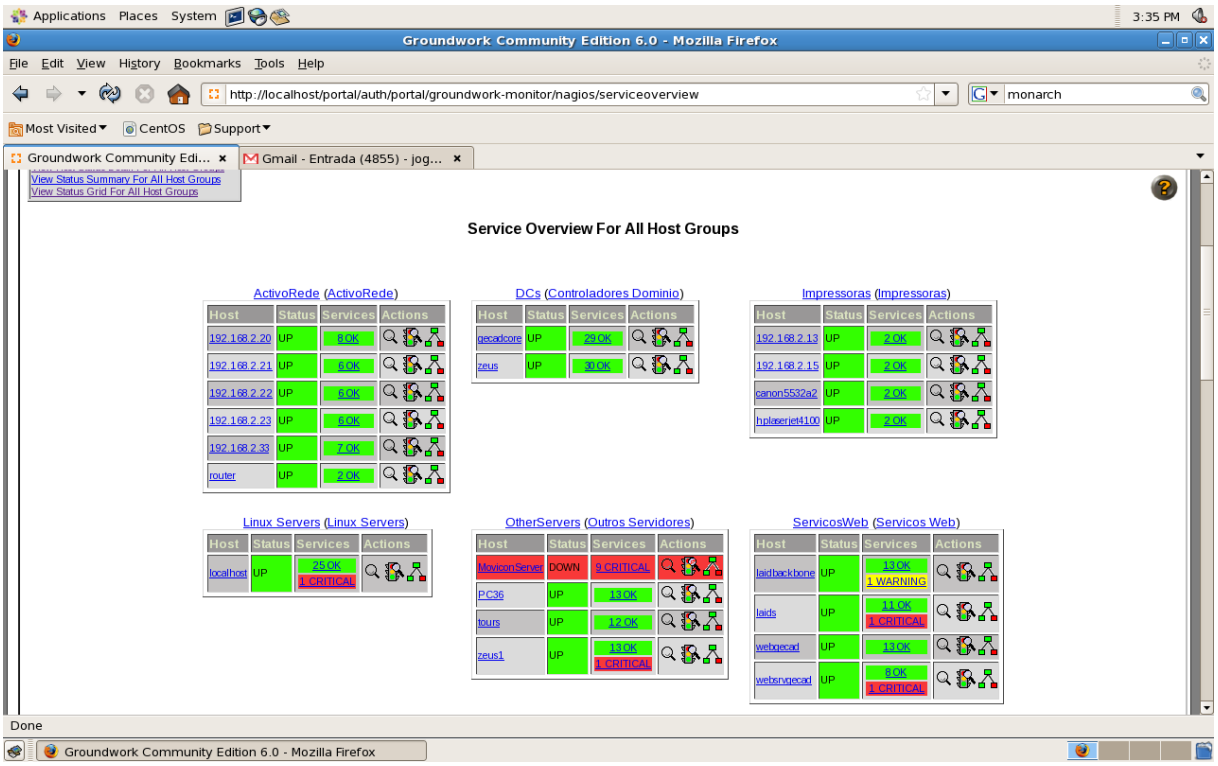


Figura 50 – Nagios – Visão geral dos serviços por Hostgroup

A capacidade de notificar um administrador de rede (através de email, pager, SMS, ou qualquer outro definido por *plugin*) quando um serviço ou equipamento apresentam problemas é um dos pontos fortes deste motor de gestão, uma vez que a notificação é também enviada após a resolução do problema. Paralelamente, pode-se configurar para que existam também notificações quando determinados parâmetros considerados críticos são atingidos. Quando notamos que certas situações se repetem ou são pré-determinadas podem ser definidos tratadores de eventos que executam tarefas mediante certos acontecimentos, acrescentando pro-actividade às características do "Nagios".

O suporte para implementação de monitorização redundante está presente, assim como, uma excelente interface Web para visualização do status da rede ou outros como vimos nas figuras 49 e 50. No caso da solução de gestão da rede do GECAD, essa mesma interface praticamente não é usada, dado que tudo o que diz respeito ao "Nagios" está embebido e acessível no "GroundWork Monitor" [69].

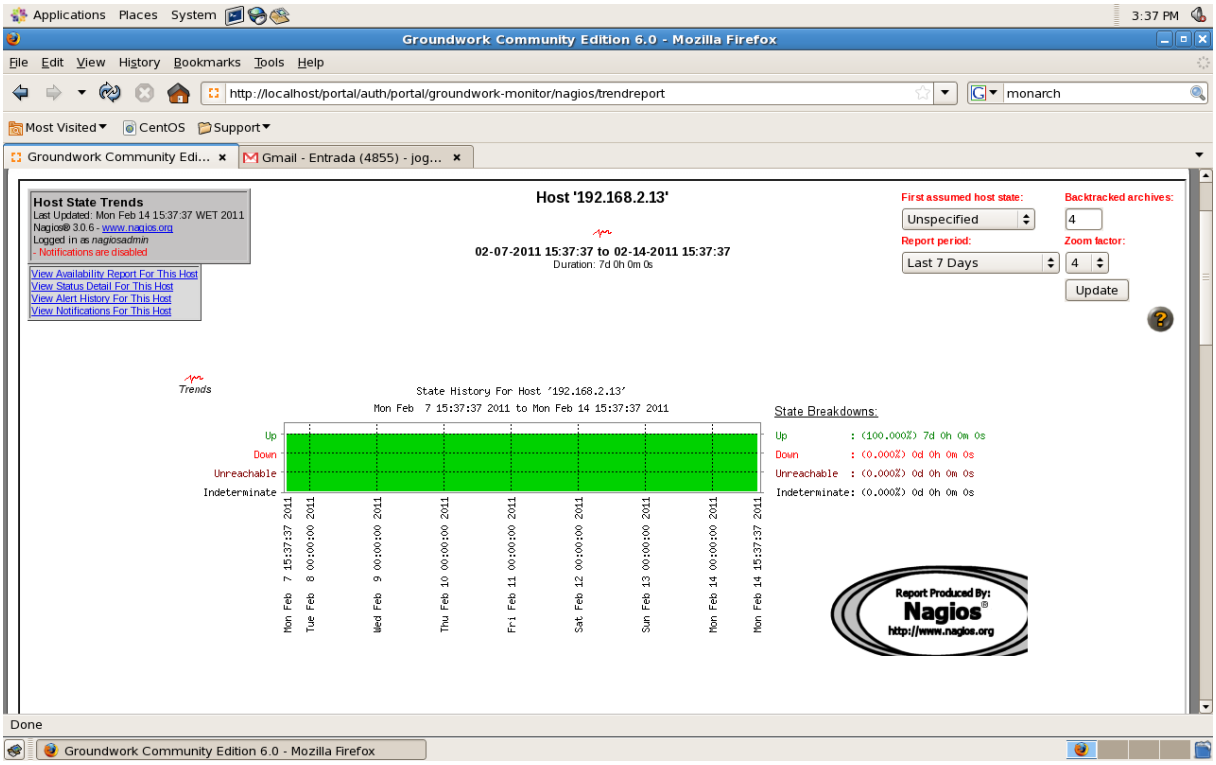


Figura 51 – Nagios – Estado de Um host ao longo do tempo

A figura anterior, figura 51 apresenta o que é possível obter no "GroundWork Monitor".

4.5.3 Ferramenta - Configuração

A aplicação de gestão está configurada com o objectivo de coexistir com o paradigma dos Agentes móveis. No entanto, as tarefas que vão ser suportadas por Agentes móveis foram inicialmente implementadas na solução de gestão tradicional por serem críticas. De forma a encontrar e classificar eficientemente todos os elementos relevantes, foi usada a funcionalidade de *Auto Discovery* para a *subnet* do GECAD ("192.168.2.0"). A figura 52 mostra uma visão geral hierárquica da configuração efectuada.

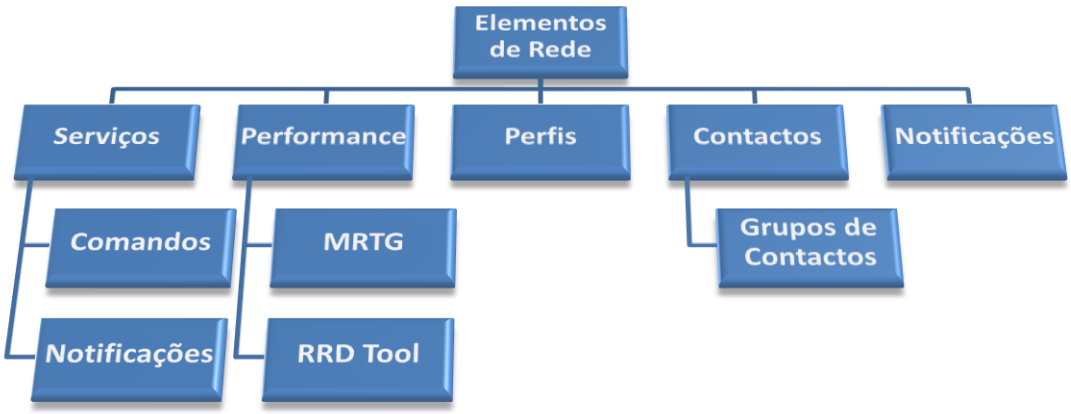


Figura 52 – Visão Geral da Configuração

É evidente o destaque dado aos elementos de rede, uma vez que o segundo nível já tem os serviços, a performance, perfis, contactos e notificações para os próprios elementos de rede. A cada elemento de rede é associado um conjunto de serviços, ou verificações (*checks* como o "Nagios" se lhes refere), que são responsáveis pela monitorização de determinado recurso, tal como a utilização do CPU, da memória, dos discos, entre outros. Por trás de cada serviço existe um comando que é executado, ou seja, os serviços estão associados aos elementos de rede e os comandos aos respectivos serviços. Daí resulta a hierarquia encontrada na figura 52, onde temos comandos associados aos serviços e notificações associadas a serviços.

Pode-se definir contactos e grupos de contactos para notificar sempre que ocorra um determinado problema. O tipo de notificações possível é bastante abrangente e está directamente relacionado com o que está associado. Podem existir notificações que são enviadas reportando-se directamente sobre um elemento de rede, sendo os estados do elemento de vários tipos, mas usualmente enviam-se notificações quando este altera para *down* (em baixo), *unreachable* (impossível de contactar) e *recovery* (recuperado). Devemos ter em atenção que na maioria dos casos o estado *unreachable* não se deve confundir com o facto do elemento de rede estar em baixo, isto porque o mesmo pode estar ligado, com todos os serviços activos e a funcionar em pleno, mas devido a algum problema na rede entre a estação gestora e o elemento, não é possível chegar a este. Na configuração da solução de gestão da rede do GECAD estes são os estados que originam notificações por email para os administradores da estação gestora. Relativamente aos serviços, os estados possíveis são os seguintes: *unknown* (desconhecido) *critical* (crítico) *recovery* (recuperado) e *warning* (alerta). Crítico pode-se referir a um CPU que tem demasiada carga do que o aconselhado e definido aquando da criação do serviço, como pode ser um determinado processo que deveria estar a correr

no elemento de rede, mas não o está. Um estado de alerta, fazendo uma pequena analogia, pode-se verificar na mesma situação de sobrecarga do CPU, mas apenas até um ponto em que ainda não se considera crítico, por exemplo, 80% de utilização é um alerta mas 85% já poderá ser crítico.

Alguns administradores de rede optam por acrescentar às típicas notificações apresentadas, uma outra, que corresponde ao facto de um determinado elemento de rede ou serviço estar num estado de *flapping*, ou seja, num período, as alterações de estados são demasiado frequentes, o que indica que poderá existir algum problema. Neste caso também são enviadas notificações.

Na estação gestora de rede do GECAD, estão definidos perfis de elementos de rede e perfis de serviços. Os perfis foram criados para, de acordo com o elemento de rede ou serviço, ser-lhe atribuído automaticamente uma série de configurações. Na prática, por exemplo, se adicionamos um elemento de rede com capacidades de responder a *pollings* SNMP, devemos atribuir-lhe um perfil de acordo com as suas capacidades. Esse perfil irá conter desde logo uma série de serviços ligados à capacidade de comunicar por SNMP.

Outro aspecto fulcral na gestão da rede do GECAD que ainda não foi abordado e que está representado na figura anterior trata-se da performance. Normalmente o "GroundWork Monitor" é capaz de obter dados de performance dos vários elementos de rede e dos seus respectivos serviços. Alguns são conseguidos com facilidade, mediante apenas alguns dados de resposta, enquanto outros estão inerentes a uma série de cálculos que necessitam de ser efectuados após o envio de vários *pollings*, de forma a consultar vários objectos nas MIB's. Dois conceitos são essenciais, e por isso surgem no diagrama apresentado na figura 52, relativamente à configuração de performance: "MRTG" e "RRDTool". "RRDTool" é um sistema de base de dados *round-robin* criado por Tobias Oetiker desenvolvido para armazenar dados sobre o estado de redes de computadores, entre outros [74]. RRD é a forma abreviada de nos referirmos a *Round Robin Database*, sendo que esta ferramenta nos permite não só o armazenamento, mas também a criação, de gráficos que permitem dar uma ideia visual dos dados que se encontram guardados. Estes dados podem ser utilizados ou exibidos por outros sistemas como o "Cacti", que também pode ser incorporado na solução de gestão "GroundWork Monitor". Relativamente ao GECAD, a ferramenta "RRDTool" está a ser utilizada para a produção de gráficos aquando da recepção de dados que o permitam, isto é, são necessários dados específicos, de performance, que após o seu armazenamento são processados pela ferramenta de modo a construir as vistas que entendermos. Existem vários alvos interessantes que foram abordados para configuração, nomeadamente os tempos de resposta a pacotes ICMP, HTTP, HTTPS, assim como utilização de CPU e memória em todos os elementos de rede, considerados fundamentais. A figura 53 apresenta um exemplo de um

gráfico gerado pela ferramenta "RRDTool", em que o objectivo é observar o consumo de memória num determinado elemento de rede, durante as últimas vinte e quatro horas.

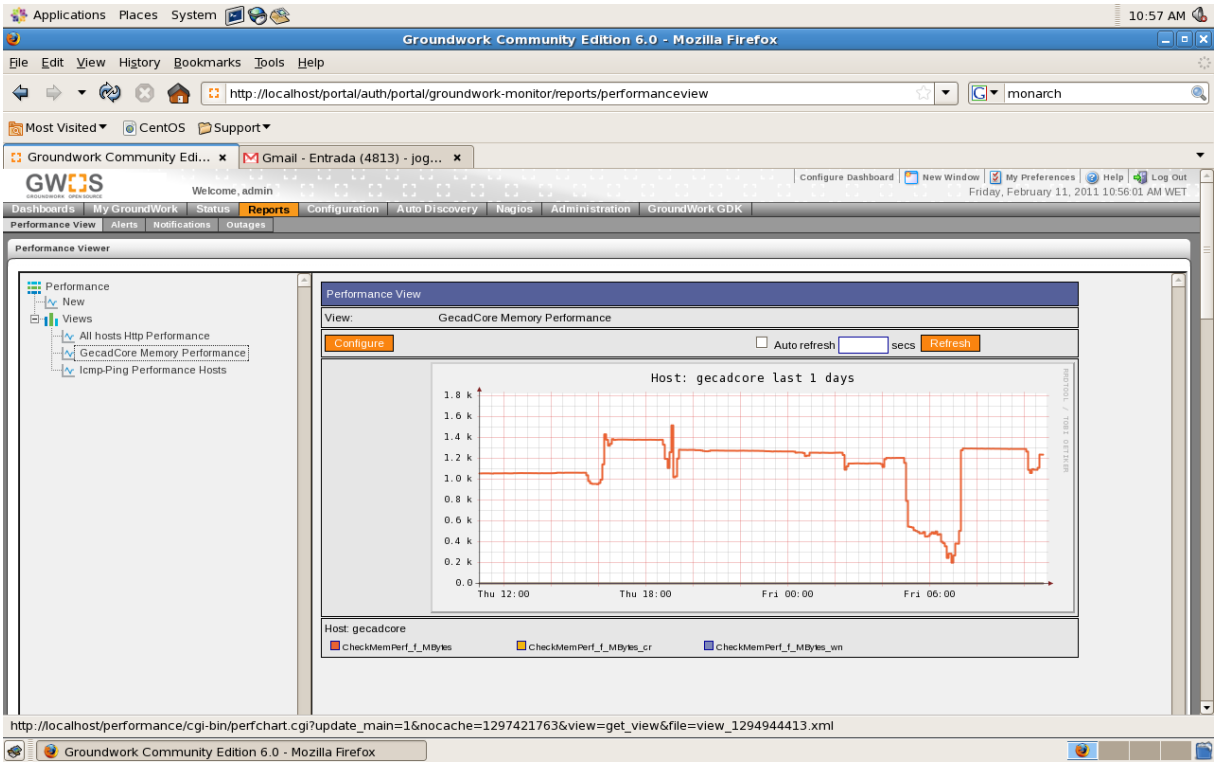


Figura 53 – Gráfico RRDTool

Na árvore à esquerda, mediante a selecção, existem mais de cinquenta gráficos configurados, que são mantidos a partir de dados obtidos dos elementos de rede permitindo mostrar informação sobre o uso de vários recursos. O *Multi Router Traffic Grapher* (MRTG) e o "Cacti" são outro tipo de ferramentas que estão em fase de testes. A diferença entre eles reside no facto de o "MRTG" ser mais voltado para activos de rede enquanto o "Cacti" consegue ser mais completo e apresentar resultados visualmente mais ilustrativos [75].

A abordagem inicial da configuração do "GroundWork Monitor" passou por separar os elementos de rede que não interessavam para qualquer tipo de monitorização dos restantes, todos resultados da pesquisa efectuada em *auto discovery*. Uma vez encontrados os elementos de rede essenciais, foi feita uma divisão destes por grupos. Foram criados vários grupos, e a cada um, atribuídos os respectivos elementos de rede que melhor se enquadravam, figura 54.

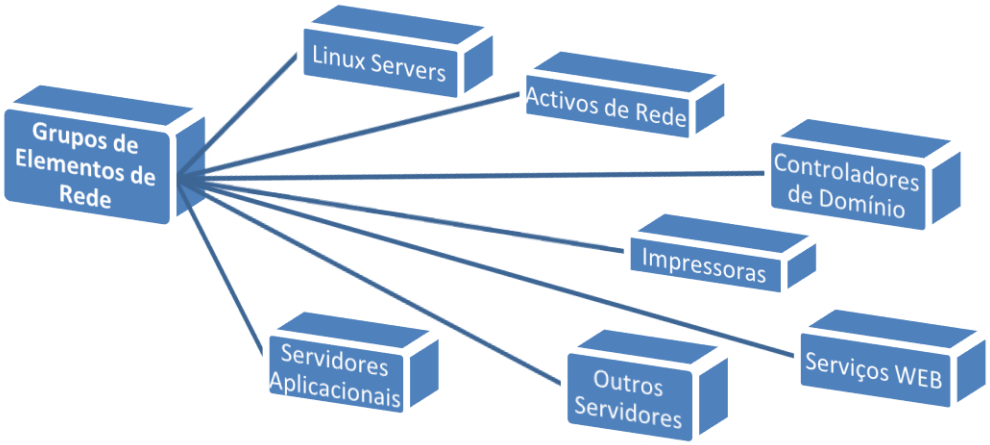


Figura 54 – Grupos de Elementos de Rede

O resultado são sete grupos de elementos de rede: Linux Servers, Activos de rede, Controladores de domínio, Impressoras, Serviços Web, Servidores Aplicacionais e Outros Servidores. No GECAD o único elemento de rede com Sistema Operativo baseado em Linux, e importante na monitorização, é a máquina responsável por albergar a estação gestora. Quanto aos activos de rede são todos os *routers*, *switch's* e *AP's* existentes e que suportam gestão, isto é, aqueles aos quais podemos atribuir um endereço IP e que se pode gerir através de um simples browser, ou mesmo por linha de comandos. Na nossa unidade de investigação existem sete *switch's* sem gestão, isto é, equipamentos que não aderem ao uso de SNMP, ou outros métodos, que permitam monitorizar os mesmos e verificar o seu bom funcionamento em tempo real. Sete *switch's* é um número bastante elevado, pois vários problemas podem acontecer na rede derivados da falha num destes aparelhos. Quanto aos restantes activos de rede, são aproveitadas as suas capacidades de SNMP assim como a simples comunicação por TCP/IP para obter dados relevantes para a gestão.

A rede possui dois controladores de domínio que naturalmente são responsáveis por manter vários serviços fulcrais para o bom funcionamento da mesma, assim como, para algumas tarefas que os utilizadores necessitam de executar. Normalmente, associam-se a este tipo de elementos de rede funções de DHCP, DNS, servidor de ficheiros, entre outras, muito próprias dos controladores de domínio e que exigem uma monitorização constante, eficiente e proactiva. Assim sucede no nosso caso, existindo um controlo bastante elevado por parte da solução de gestão sobre estes dois elementos.

O quarto grupo corresponde às impressoras, existentes nos laboratórios do edifício I e R e que totalizam cinco equipamentos deste tipo a ser activamente monitorizados. Neste grupo em particular existe a natural necessidade de saber se os elementos de rede estão, ou não, disponíveis, mas muito mais poderia ser feito (aproveitando as informações que é possível obter por SNMP) do

que aquilo que está efectivamente a ser controlado. Isto deve-se ao facto de existirem outras prioridades, a nível de gestão, e este trabalho propor um sistema de gestão de redes baseado no paradigma dos Agentes móveis.

No quinto grupo encontramos os Serviços Web, onde se localizam todos os servidores mais importantes que são responsáveis por albergar conteúdos disponíveis sob a forma de páginas Web. Alguns destes fazem uso desse tipo de alcance e oportunidades que um simples browser oferece para dar a conhecer a quem interessar os respectivos projectos, enquanto outros correspondem mesmo a aplicações relativas a projectos em específico, isto é, uma dada aplicação voltada para a Web retrata e representa os objectivos que esse mesmo projecto possui. Neste grupo está o servidor responsável por albergar o site do GECAD, e outros três com aplicações específicas.

O GECAD possui três servidores, em breve quatro, que são utilizados para fins de computação massiva. Estes elementos de rede são vitais para muitos dos utilizadores que necessitam de recursos adicionais para executar determinadas tarefas. Como tal, o grupo dos Servidores Aplicacionais é constituído precisamente por estes três servidores, que só podem ser acedidos remotamente por utilizadores da unidade de investigação, sendo que cada um deles tem a mesma prioridade no acesso aos recursos (CPU, RAM, etc). Existe um balanceamento de carga efectuado a vários níveis cujo objectivo é garantir que vários utilizadores possam estar conectados com as mesmas possibilidades na utilização de recursos.

O último grupo possui "Outros Servidores" que não se encaixam em nenhuma das categorias anteriores mas que também são importantes para uma boa gestão. Neste grupo está o "MoviconServer", responsável por albergar várias conexões (perto de trinta até ao momento) de utilizadores fictícios para efeitos de testes, numa aplicação específica de um projecto. Estão incluídos, a Workstation de nome "PC36", assim como, um servidor de nome "ZEUS1" (antigo controlador de domínio), que contém vários backups oriundos de diversas fontes. Por último, um servidor de nome TOURS faz também parte deste grupo, uma vez que, está dedicado a um projecto, actuando como servidor de desenvolvimento, hospedando e disponibilizando as aplicações resultantes desse mesmo desenvolvimento.

Para o "Nagios" atingir os objectivos de monitorização estabelecidos, contribuem os serviços aos quais estão associados determinados comandos. A próxima figura, figura 55, ilustra a forma como é feita a monitorização de um serviço [69, 76, 77].

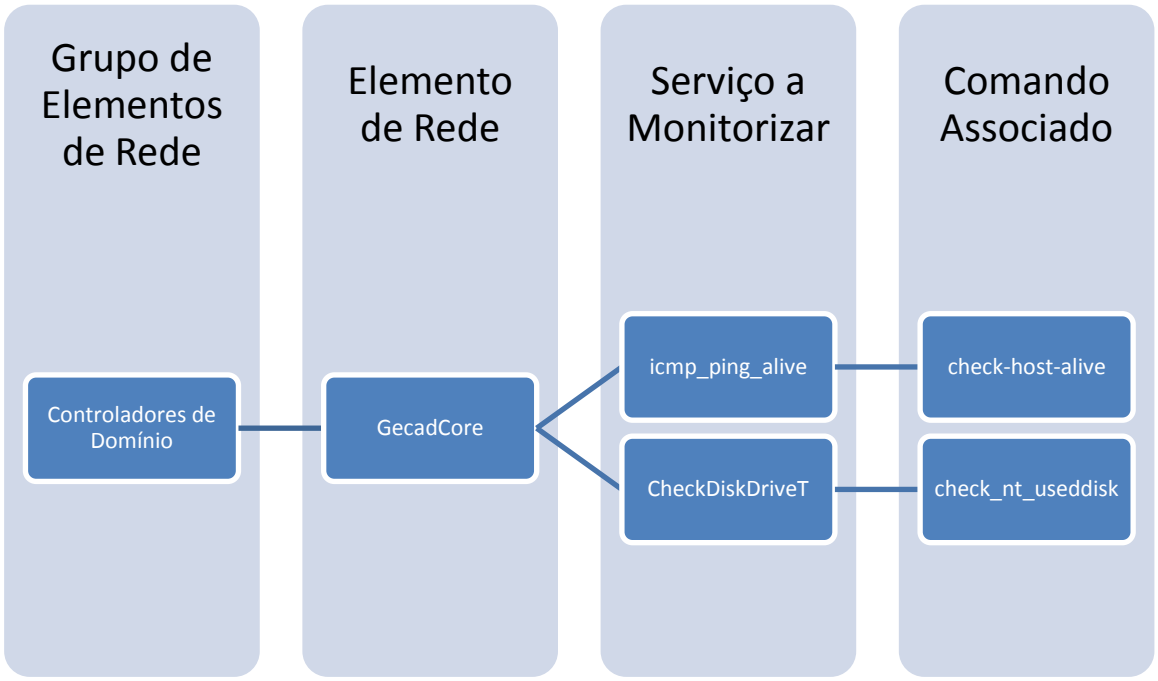


Figura 55 – Exemplo de Monitorização

Basicamente o elemento de rede tem associados a si serviços (no caso ilustrado são dois) que possuem uma designação e um comando associado. O servidor "gecadcore" tem vários discos e partições, sendo que uma delas tem a letra "T" a representá-la junto do Sistema Operativo. Daí a escolha do nome "CheckDiskDriveT" para nomear o serviço que deve tornar-se responsável por periodicamente verificar o espaço ocupado/livre dessa unidade, retornando valores em bytes e em percentagem. Isto é conseguido através da execução de um comando que contém as directivas essenciais para conseguir cumprir a verificação junto do elemento de rede. Neste caso específico, é utilizado um *plugin* que reside na máquina gestora, permitindo comunicação entre este e um pequeno cliente que reside no elemento de rede, obtendo-se os resultados dessa consulta da entidade gestora ao elemento em causa. Já o serviço "icmp_ping_alive" é usado para verificar se o elemento de rede está a funcionar e com capacidades de comunicação. São enviados pacotes ICMP, com a ajuda do comando associado, devendo ter-se em conta as possíveis *firewall's* existentes, para não confundir um elemento de rede, com problemas de um outro que funciona perfeitamente mas bloqueia pedidos ICMP. A partir daqui conseguimos saber se existe comunicação, qual o nível de resposta aos pacotes enviados, e gerar gráficos baseados nesse mesmo nível avaliando a performance.

Os dois exemplos anteriores não utilizam o protocolo SNMP. A solução de gestão utiliza vários protocolos com vista à obtenção de informação que suporte a gestão. Foram também desenvolvidos comandos que satisfazem as necessidades específicas da rede do GECAD. A aplicação de gestão

adoptada tem esta facilidade, permitindo a criação de novos comandos, ou até mesmo alterar os disponibilizados.

A tabela seguinte apresenta o número de elementos de rede e respectivos serviços existentes em cada grupo.

Grupo de Elementos de Rede	Número de Elementos de Rede	Número de Serviços
Activos de Rede	6	35
Controladores de Domínio	2	59
Impressoras	4[1]	8
Linux Servers	1	26
Serviços WEB	4	48
Servidores Aplicacionais	3	31
Outros Servidores	4	48
Total	**24**	**255**

Tabela 4 – Elementos de rede/Serviços

O número de elementos de rede em conjunto com os serviços monitorizados já é um valor elevado, tendo em conta a dimensão do GECAD. As características da rede do GECAD e as ferramentas já instaladas e configuradas representam um cenário muito interessante para propor uma abordagem de gestão baseada em Agentes móveis (capítulo 5).

O número total de serviços é de duzentos e cinquenta e cinco, estando longe daquilo que é consultado na realidade junto dos elementos de rede. Isto acontece devido ao facto de muitos dos serviços existentes concentrarem monitorizações de mais do que um componente nos elementos de rede. Por exemplo, o facto de um servidor possuir o "SQL SERVER" instalado, significa que pelo menos um processo terá de ser monitorizado, de forma a controlar o estado da execução. No entanto, quando o "SQL SERVER" é iniciado, são lançados vários processos correspondentes ao software, que também necessitam de monitorização. Obtemos um total de vinte e quatro elementos de rede e duzentos e cinquenta e cinco serviços na solução de gestão, mas a realidade é bem diferente, pois existe muito mais computação, consultas e acções que são levadas a cabo. Resumindo, não é por se tratar apenas de um serviço que o seu trabalho reside somente na execução de uma tarefa.

[1] Falta uma Impressora adicionada à rede perto do fim dos trabalhos totalizando 5 impressoras

4.5.4 Evolução da solução de Gestão – Requisitos

O exposto nas secções anteriores deste capítulo conduz à formulação das seguintes questões:

- "Que falhas existem na actual solução de gestão";
- "Como podemos melhorar a solução de gestão actual";
- "Em que medida os Agentes móveis podem acrescentar valor".

Para conseguirmos encontrar resposta a estas questões é necessário fazer alguma análise, que nos conduzirá aos requisitos da solução. Se utilizarmos apenas o paradigma tradicional na estação gestora, sem o auxílio de Agentes móveis, retiramos desde logo a possibilidade de deslocar parte do processamento e acrescentar pró actividade nos elementos de rede.

O "GroundWork Monitor", como qualquer aplicação tradicional de gestão, possui tarefas que envolvem transferência de dados em massa, tais como tabelas SNMP extensas. Estas tabelas são obtidas usando de forma repetida a operação "GET-NEXT". Por cada "GET-NEXT" precisamos de esperar pela resposta para que possamos passar ao próximo "GET", sendo que por cada "GET" apenas nos é devolvida uma linha da tabela. Se a tabela for grande, este procedimento tem impacto nos recursos da rede provocando latência. Existe uma melhoria, através de uma operação mais complexa, o "GET-BULK" que introduz melhorias, transferindo várias linhas simultaneamente. No entanto obriga a que os utilizadores saibam o número máximo de dados que necessitam, caso contrário, apenas servirá para receber dados a mais do que os realmente desnecessários.

Surge então um problema, o efeito gargalo junto do servidor que gere a rede através da nossa aplicação, e também o consumo excessivo de recursos na rede. O recurso a uma solução baseada em Agentes móveis permite mover o processamento para os elementos de rede, processando localmente as operações "GET-NEXT". Este é **um requisito** fundamental, a capacidade de termos na nossa aplicação Agentes móveis que falem SNMP e que consigam fazer não só as operações "GET-NEXT", mas também que sejam capazes de filtrar aquilo que realmente nos interessa, efectuando processamento no elemento de rede de forma a distribuir a carga que o processo de gestão provoca.

O **segundo requisito** passa por conseguir aumentar a eficiência junto de processos de gestão que envolvem dados sobre recursos como CPU, RAM e espaço em disco. Para que a ferramenta actual consiga dados estatísticos tem de repetidamente consultar os elementos de rede, sem fazer qualquer tipo de intervenção no final. Com Agentes móveis podemos mover o processamento e também tomar medidas de imediato sempre que são detectados problemas.

O **terceiro requisito** corresponde à necessidade de obter informações que permitam criar gráficos ilustrativos do comportamento de determinados recursos num dado elemento de rede. No entanto, uma solução de gestão tradicional, para obter tais informações, recorre ao *polling* e após a recepção das correspondentes respostas gera os gráficos. Uma solução baseada em Agentes móveis permite eliminar os *pollings*, uma vez que é enviado o Agente específico para o elemento de rede, onde executa as funções para as quais foi configurado, retornando apenas quando obteve todos os dados necessários, com as vantagens evidentes desta abordagem.

O **quarto requisito** surge da necessidade de recolher dados sobre os activos de rede que comunicam via SNMP, de forma mais eficiente. Este tipo de tarefas pode ser realizado tendo em conta uma distribuição do *polling* efectuado, isto é, desistir da abordagem centralizada que é executada pela aplicação "GroundWork Monitor". Os Agentes podem mover-se para um segmento de rede, para um determinado elemento (PC, Servidor, etc), que esteja o mais próximo possível do activo de rede em questão, executando todos os pedidos e análises iniciais a partir desse ponto, aliviando assim a aplicação principal.

O **quinto requisito** está relacionado com o objectivo de fazer uma avaliação correcta e completa do estado de determinados elementos de rede, que se encontram ligados a *switch's* sem gestão. Se ocorre uma falha nestes activos de rede, a aplicação actual não consegue distinguir se o problema se encontra no elemento de rede que não é atingível, ou se está no *switch* em questão. Um Agente móvel preparado para tal, consegue recriar o trajecto necessário para chegar ao elemento de rede do qual não obtemos resposta, validando ou não, cada um dos pontos por onde passa.

O **sexto requisito** procura garantir a recuperação de falhas ligadas às caches de DNS e ARP, que causam muitos problemas a nível de comunicação nos elementos de rede quando surgem. Para tal, necessitamos de Agentes móveis que se desloquem para o local, façam um diagnóstico e consigam reparar as ligações.

Por último, o **sétimo requisito** está relacionado com todos os processos vitais para o bom funcionamento da rede, que se localizam em vários elementos desta e que é necessário acompanhar com regularidade. Neste ponto a carga que recai sobre a aplicação tradicional existente é muito grande, porque tem de fazer constantes *pollings* para averiguar dez, quinze ou vinte processos por elemento de rede. Geralmente, se algum estiver parado, não é tomada qualquer acção ou quando o é, exige resposta, análise, envio do comando a executar para recuperação e nova resposta com o resultado. Os Agentes móveis podem agir com o objectivo de processar todas essas consultas em paralelo e localmente, em cada um dos elementos de rede, procurando ser autónomos, recuperando serviços ainda no local e apenas depois informar dos resultados.

4.6 Conclusão

Nas últimas secções foram apresentadas as características da rede do GECAD, começando pela sua topologia, meio físico e componentes existentes. Por último, foi apresentada a solução de gestão utilizada. Foram ainda identificados os requisitos para melhorar a solução actual e que servem de base à arquitectura proposta no próximo capítulo com base em Agentes móveis.

Por tudo o que foi exposto, pode-se concluir que as áreas funcionais de gestão que são importantes incluir no modelo de gestão a seguir para a rede do GECAD são:

- Gestão de Falhas;
- Gestão do Desempenho;
- Gestão de Contabilização;

As acções de monitorização estão na base das três áreas funcionais referidas. Por outro lado, para realizar a tarefa de monitorização são necessários *pollings*, os quais acarretam consequências já amplamente discutidas neste trabalho. Pelo exposto, consideramos que uma solução de gestão baseada no paradigma dos Agentes móveis é adequada.

5 Arquitectura Baseada em Agentes móveis

5.1 Introdução

Neste capítulo é apresentada uma arquitectura para um sistema de gestão, baseada no paradigma dos Agentes móveis, para o grupo de investigação GECAD. São apresentados com detalhe todos os elementos que constituem o sistema e as respectivas interacções.

5.2 Arquitectura

5.2.1 Visão Geral

Como podemos verificar pela observação da figura 56 existem três elementos fundamentais na arquitectura proposta: o Núcleo, os Elementos de Rede e os Agentes móveis, todos eles assentes numa rede IP utilizada para o suporte de todo o sistema.

O **Núcleo** é o centro de controlo do sistema. Nele está incluída a ferramenta de gestão tradicional, já apresentada no quarto capítulo, secção 4.5 (Gestor de Redes). O núcleo é responsável pela criação e instanciação dos Agentes móveis, assim como da apresentação dos resultados obtidos por eles. A grande maioria dos elementos de rede possui capacidades SNMP. Tais capacidades também foram adicionadas aos Agentes móveis. É responsabilidade do núcleo a unificação dos serviços e capacidades de todos os elementos que constituem o sistema.

O **Agente Facilitador do OAA** é quem coordena a comunidade de Agentes. Cada Agente da comunidade regista as suas capacidades ou funcionalidades no Facilitador do OAA. Quando um determinado serviço é requisitado por um Agente, o pedido é feito ao Agente Facilitador. O Agente Facilitador é o Agente que conhece todos os Agentes disponíveis, assim como os serviços disponibilizados por cada um. O pedido de uma tarefa pode, ou não, ser dividido em subtarefas para serem executadas por Agentes diferentes e distribuídos.

O **Elemento de Rede** pode ser um *router*, um *switch*, um servidor ou um simples PC, no fundo, todos aqueles componentes existentes na rede e que o administrador necessite de monitorizar e/ou configurar em tempo real.

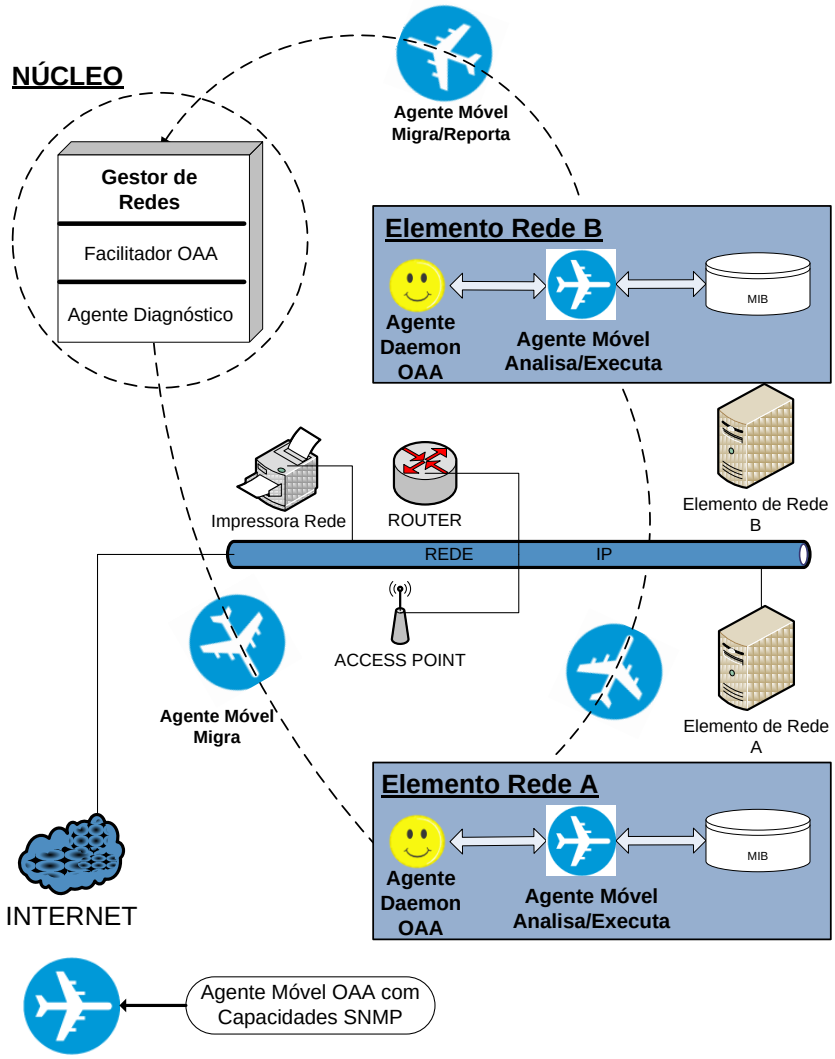


Figura 56 – Visão Geral da Arquitectura

A maioria das funções de gestão são executadas através do protocolo SNMP. No entanto, algumas das funções de gestão executadas pelos Agentes móveis não usam o protocolo SNMP.

O **Agente de Diagnóstico** é capaz de criar e controlar Agentes móveis, monitorizá-los, entre outras tarefas relacionadas com eles. Quando o sistema inicializa este componente, ele encarrega-se de fazer uso da sua capacidade para criar e lançar um Agente móvel de acordo com as preferências do utilizador. Após o lançamento dos Agentes, o seu estado tem de ser monitorizado em conjunto com as acções que estes realizam. No entanto, não devemos confundir este tipo de controlo com as funções do Agente facilitador, responsável por saber quem pode executar o quê. Para além das funções referidas, o Agente de diagnóstico recebe os relatórios com a informação produzida pelos Agentes móveis, tendo que interagir com o seu superior hierárquico, o gestor de redes.

Os **Agentes móveis** presentes na figura 56 possuem descrição comportamental, informação sobre o seu estado e atributos próprios. Estes foram criados de forma a serem capazes de se tornarem

location aware para que possam decidir para onde e quando se devem mover. Mover um destes Agentes implica o envio do seu estado e código, estando esta fase patente nos trajectos a ponteado na mesma figura. Podemos identificar também outras fases da vida do Agente móvel, em que este executa e/ou analisa, demonstrando inteligência, pois decide para onde se irá mover de seguida e se necessita de executar alguma acção.

Uma análise sintética à visão geral da arquitectura, figura 56, permite verificar que esta tem por base uma rede IP e é constituída por vários elementos, tais como, um *Access Point*, um *router*, um servidor, etc. Para gerir estes elementos o núcleo foi dotado de funções de gestão. Os Agentes móveis pertencentes a esta solução fazem parte de uma comunidade de Agentes denominada OAA, que será detalhada na secção 5.3.2, e possuem capacidades SNMP, de modo a poderem efectuar tarefas de gestão junto dos elementos de rede. Se observarmos com detalhe os elementos A e B, notamos que estes Agentes móveis analisam informação e executam funções de gestão através de interacções com as MIB's respectivas. Para tal, os mesmos têm que ter a capacidade de comunicar via SNMP. Por último, o **Agente Daemon OAA** é um Agente específico: este Agente não é móvel, no entanto, pertence à mesma comunidade. O propósito deste Agente é receber os Agentes móveis providenciando funcionalidades que lhes permitem repor o estado e atributos.

5.2.2 Tipo de Agentes

No sistema proposto existem vários Agentes na comunidade, com diferentes características, que devem ser explicados para que seja possível compreender a nossa proposta para a solução de gestão. A figura 57 ilustra o Modelo Multi-Agente proposto para levar a cabo as funcionalidades de gestão pretendidas. Do modelo fazem parte o Agente facilitador, o Agente diagnóstico, os Agentes daemon OAA, e que não são móveis. Os Agentes que possuem capacidades móveis são o Agente de detecção/correcção de falhas e o Agente de intervenção.

O Agente de detecção/correcção de falhas é programado para percorrer um determinado itinerário na rede com o objectivo de monitorizar o seu funcionamento. É também capaz de resolver eventuais problemas detectados, ou seja, leva a "inteligência" para junto dos elementos de rede.

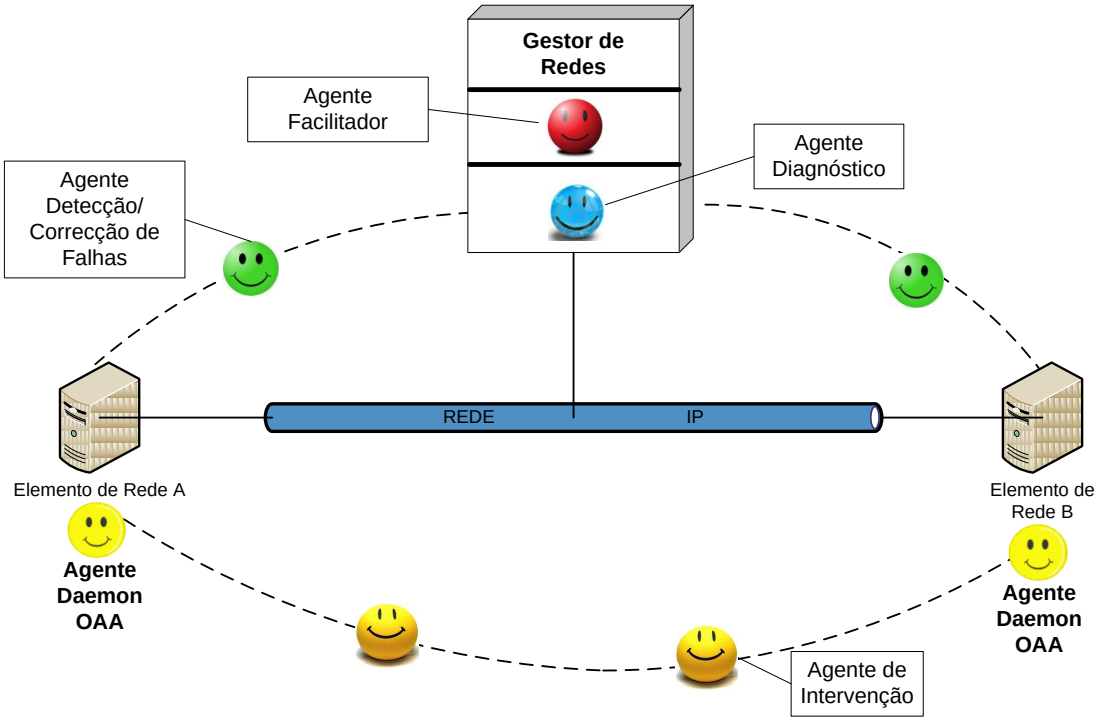


Figura 57 – Agentes Existentes

O Agente de intervenção é chamado pelo Agente de detecção/correcção de falhas sempre que este não consegue solucionar os problemas detectados. Este modelo permite desenhar Agentes que se movem pela rede com funções de detecção/correcção de falhas mais "leves", uma vez que são dotados de capacidades apenas para resolver os problemas mais frequentes, sendo os Agentes de intervenção mais sofisticados, e apenas são movidos quando solicitado ou em condições especiais.

Um exemplo desta funcionalidade é o Agente de detecção/correcção de falhas ser criado com o objectivo de percorrer um itinerário composto por três elementos de rede e consultar o disco dos mesmos para averiguar se existe, ou não, falta de espaço nas diferentes partições. Pretende-se ainda que o mesmo seja capaz de efectuar uma análise ao conteúdo dos mesmos no sentido de eliminar ficheiros que não são utilizados. Quanto ao Agente de intervenção, pode ser criado para resolver um problema similar, o consumo elevado de banda por parte do adaptador de rede de um elemento: inicialmente quem detecta o problema é um Agente de detecção/correcção de falhas, reportando a situação, tal como faz uma aplicação de gestão tradicional, emitindo um relatório com alertas. Mas o Agente de intervenção, através da utilização da chamada a programas auxiliares ou comandos do Sistema Operativo, consegue efectuar um estudo mais elaborado, com critérios de avaliação e acções mais incisivas, no entanto, mais incisivas. Podendo inclusive resolver o problema, se estiver preparado para tal.

5.2.3 Comunicação Entre Agentes

Nesta secção pretende-se demonstrar quais as principais interacções existentes entre os diferentes tipos de Agentes presentes no sistema. A figura 58 mostra os principais Agentes envolvidos no processo de gestão, assim como as relações existentes entre eles.

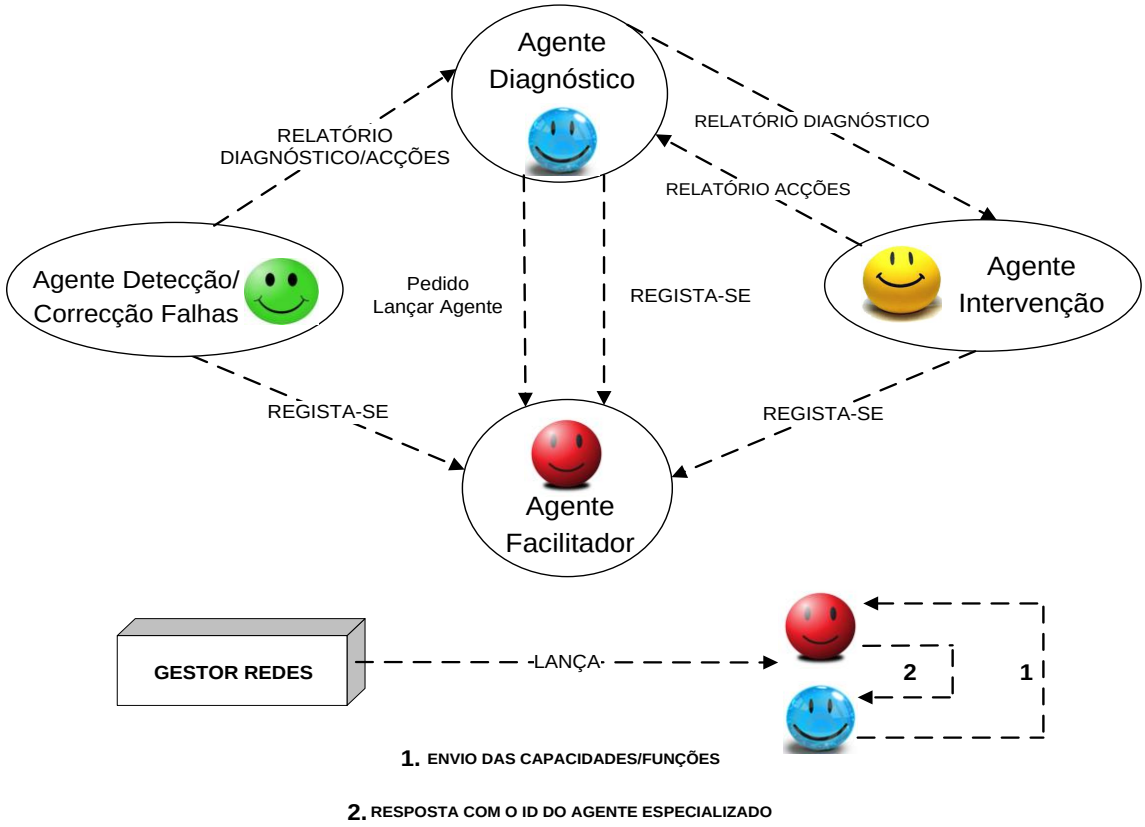


Figura 58 – Comunicação entre Agentes

O Agente de diagnóstico e o Agente facilitador são lançados pelo gestor de redes, que inicia também a aplicação correspondente aos Agentes móveis. Sempre que um Agente na comunidade é criado, regista-se junto do facilitador, informando-o do seu nome e respectivas capacidades. Pelo que o primeiro Agente a ser instanciado é o facilitador. Em seguida é lançado o Agente de diagnóstico, que se junta à comunidade.

Tem que existir pelo menos um Agente de cada tipo para que o sistema possa arrancar. Uma vez em pleno funcionamento, é possível mover vários Agentes do mesmo tipo, sendo o Agente de diagnóstico responsável por fazer o pedido de criação do respectivo Agente, comunicando essas mesmas intenções ao Agente facilitador.

Na figura anterior, figura 58, vemos também várias interacções entre o Agente de diagnóstico e os Agentes de intervenção, assim como com os Agentes de detecção/correcção de falhas. Quando um

Agente de detecção/correcção de falhas é lançado, tem um determinado itinerário e várias tarefas de gestão a executar. Como tal, é necessário no final dessas tarefas, compilar um relatório com as acções levadas a cabo e quais os resultados obtidos. Esse relatório é entregue ao Agente de diagnóstico, que analisando e interpretando o mesmo encarrega-se de verificar se existem necessidades adicionais.

A interacção com o Agente de intervenção é diferente, pois este necessita de mais informação para trabalhar, isto é, precisa que o Agente de diagnóstico lhe transfira um relatório, com um ou mais problemas existentes na rede, resultantes de operações realizadas pelos Agentes de detecção/correcção de falhas (fluxo de dados entre Agente de detecção/correcção de falhas e Agente de diagnóstico na figura 58). Também, caso não se verifique a situação anterior, e como são Agentes dotados de funções mais elaboradas, é necessário que o Agente de diagnóstico entregue um relatório diferente, criado pelo próprio, com dados concretos sobre quais os alvos de tarefas de gestão, para que o Agente de intervenção em questão possa mover-se, efectuar os trabalhos e devolver os seus resultados.

5.3 Implementação

A plataforma OAA em conjunto com a API "WebNMS SNMP" constitui a base da implementação do sistema. Iremos explicar a função de cada uma delas na solução de gestão baseada em Agentes móveis. A terceira secção apresenta a comunidade de Agentes, onde são descritos com todos os pormenores a forma como se harmonizam a plataforma OAA com a "WebNMS SNMP" e com a linguagem JAVA, de modo a criar todos os Agentes e suas respectivas tarefas de gestão.

A componente de interface é muito importante, pois envolve o utilizador do sistema de gestão. Pelo que, esta vai ser abordada não só pelo seu aspecto ou funcionalidade, mas também porque possui um papel mais activo, como iremos ver nas secções seguintes. Existem depois várias especificações de toda esta implementação que apesar de ajudar na compreensão do que foi feito, não se encaixam em nenhuma secção em particular e serão discutidas exactamente antes de se avançar para a parte final deste capítulo, com alguns dos exemplos práticos do sistema de gestão.

5.3.1 Plataforma OAA

A plataforma OAA (*Open Agent Architecture*) é a base de implementação da comunidade de Agentes. Trata-se de uma plataforma amplamente utilizada no nosso grupo de investigação para o desenvolvimento de aplicações Multi-Agente. Pelo que, existe muito conhecimento adquirido na utilização do mesmo. No entanto, esta plataforma possui características que a tornaram a escolha adequada para desenvolver o sistema de gestão baseado em Agentes móveis, tais como [78-80]:

- **Aberta** – os Agentes podem ser escritos em várias linguagens (Prolog, Java, ANSI C/C++, LISP) e Sistemas Operativos. Neste caso, as barreiras de linguagem e do Sistema Operativo são mínimas;
- **Distribuída** – os Agentes podem ser distribuídos por múltiplas máquinas ligadas em rede. Esta é uma vantagem muito importante pois, no caso concreto deste Sistema, permite que cada tarefa a ser executada esteja dividida em várias subtarefas, onde estas subtarefas são executadas por vários Agentes e em várias máquinas;
- **Extensível** – é possível adicionar e remover Agentes em tempo de execução, permitindo a criação de cenários flexíveis e robustos;
- **Móvel** – podem ser executados interfaces com o utilizador baseados em OAA (*Open Agent Architecture*) em *Personal Digital Assistants* (PDAs).

A plataforma OAA é composta por dois tipos distintos de Agentes: os facilitadores do OAA e os clientes. Normalmente, existe apenas um facilitador por aplicação, mas é possível ter múltiplos facilitadores. O facilitador do OAA é o Agente responsável pelas tarefas relacionadas com a coordenação e a comunicação. O facilitador do OAA tem de certo modo um comportamento similar ao de um *router*, no sentido em que é responsável pela distribuição de dados e mensagens pelo conjunto de Agentes clientes. A primeira comunicação entre dois Agentes é obrigatoriamente feita através do facilitador, depois já podem comunicar directamente se necessário. O outro tipo de Agente é o Agente cliente, que quando criado, cria uma ligação com o facilitador informando-o dos serviços que fornece [78-80].

A plataforma OAA disponibiliza uma linguagem de comunicação, *Inter-Agent Communication Language* (ICL), que é partilhada pelos Agentes, independentemente da linguagem em que tenham sido criados e do próprio Sistema Operativo do computador em que os Agentes se encontrem. A linguagem ICL é próxima da *Knowledge Query Manipulation Language* (KQML), também uma linguagem de comunicação entre Agentes de software [78-80].

Constituição de um Agente

Os Agentes providenciam serviços a outros Agentes. Estes serviços podem também ser chamados de capacidades ou *solvables* que é o nome técnico utilizado na aplicação e na plataforma OAA. Os Agentes têm vários *solvables* que são as suas capacidades. Os *solvables* podem ser considerados tarefas que os Agentes realizam mediante um pedido difundido para a comunidade de Agentes. Os pedidos à comunidade de Agentes são feitos com a seguinte estrutura [78-80]:

Pedido (solvable, parâmetros, respostas)

Os pedidos podem ser feitos internamente na comunidade de Agentes (feitos entre Agentes) ou externamente (da componente de interface para a comunidade de Agentes). No Pedido, o argumento *solvable* é a variável que vai ter o *solvable* (capacidade) e que vai ser requisitado à comunidade de Agentes. Os Agentes que realizarem este *solvable* vão responder com uma tarefa ou acção que pode, ou não, implicar uma resposta. A variável parâmetros é uma lista que vai conter os parâmetros do pedido. Pode ser considerado como um filtro, o pedido é apenas enviado para o, ou os, Agentes da lista de parâmetros. Por último, a variável respostas vai conter as respostas, do, ou dos, Agentes que realizaram o *solvable* requisitado, e queiram enviar alguma resposta ou resultado a quem fez o pedido [78-80].

Para ser possível a criação de um Agente na comunidade de Agentes é necessário definir os *solvables* do Agente e adicioná-los a uma lista de *solvables*. O Agente depois de definir os *solvables* deve-se registar no facilitador do OAA, de modo a que estes se tornem públicos para a comunidade [78-80].

Comunidade de Agentes

A comunidade de Agentes é constituída por um conjunto de Agentes especialistas em várias áreas distintas, a trabalhar em conjunto, para resolver tarefas para o utilizador. Apesar de ser possível criar apenas um Agente com o objectivo de cumprir autonomamente todas as tarefas, é preferível ter vários Agentes com capacidades distintas, para ser possível realizar tarefas em simultâneo. O benefício de um sistema baseado em Agentes só é completamente compreendido quando for estendido por vários Agentes especializados. Cada um dos Agentes da comunidade contém *solvables* que podem ser considerados como capacidades, funcionalidades ou mesmo serviços. A comunidade de Agentes deve ter o maior número de Agentes com capacidades o mais variadas possível, para poder ser mais flexível e para poder responder a um maior número de pedidos. Quando é feito um pedido para a comunidade de Agentes, é seleccionado quais os Agentes que têm essa capacidade (*solvable*) e solicitado que estes resolvam o pedido [78-80].

Facilitador

O Agente Facilitador do OAA é quem coordena a comunidade de Agentes. Cada Agente da comunidade de Agentes regista as suas capacidades ou funcionalidades no Facilitador do OAA. Quando são requisitados serviços por um Agente, em vez deste pedir a um Agente específico que realize essa tarefa, pode apenas fazer o pedido e depois o Agente Facilitador do OAA é que decide qual, ou quais, são os Agentes disponíveis e capazes de responder. O pedido de uma tarefa pode, ou não, ser dividido em subtarefas para serem executadas por Agentes diferentes e distribuídos [78-80].

5.3.2 API WebNMS SNMP

Com a possibilidade de criar uma comunidade de Agentes através da plataforma OAA, para realizar actividades de gestão de redes, foi necessário dotar alguns desses Agentes com capacidades SNMP. Para tal, foi incorporada a API "WebNMS SNMP", uma vez que permite o rápido desenvolvimento e implementação de soluções de gestão de redes integradas, baseadas em SNMP. A acompanhar esta API existe um conjunto de pequenas ferramentas úteis que ajudam a completar o desenvolvimento das nossas aplicações, sendo o "MibBrowser" uma delas, já apresentado na secção 2.5.

A API suporta, com o recurso à sua biblioteca SNMP em JAVA, o desenvolvimento de aplicações *standalone*, baseadas em Web, ou mesmo distribuídas (sob a forma de EJB, CORBA ou RMI). No caso deste trabalho obtém-se uma solução distribuída através dos Agentes móveis. É possível implementar as operações básicas SNMP GET, SNMP GETNEXT, SNMP GETBULK, e SNMP SET recorrendo à biblioteca JAVA SNMP. Esta possibilidade permite que os Agentes da comunidade OAA possam ler e escrever nas MIB's, no sentido de efectuar acções de gestão [81-83].

A arquitectura contém várias camadas de API's, que fornecem aos utilizadores (programadores) diferentes níveis de acesso, de acordo com os seus objectivos, no que ao desenvolvimento das suas aplicações diz respeito. Por exemplo, um programador que possua algumas dificuldades sobre conceitos SNMP pode utilizar directamente as API's de alto nível para o seu ambiente de desenvolvimento. Já os mais experientes poderão usar a arquitectura de baixo nível para o desenvolvimento das suas aplicações. As API's de baixo nível SNMP, MIB e SAS podem ser usadas directamente ou a partir de *beans* fornecidos pela API de alto nível.

Em qualquer um dos casos, as aplicações dos programadores são capazes de falar com as API's SNMP através das API's distribuídas, a partir de onde as aplicações dos utilizadores podem assentar. A próxima figura, figura 59, ilustra a arquitectura API "WebNMS SNMP" [81-83].

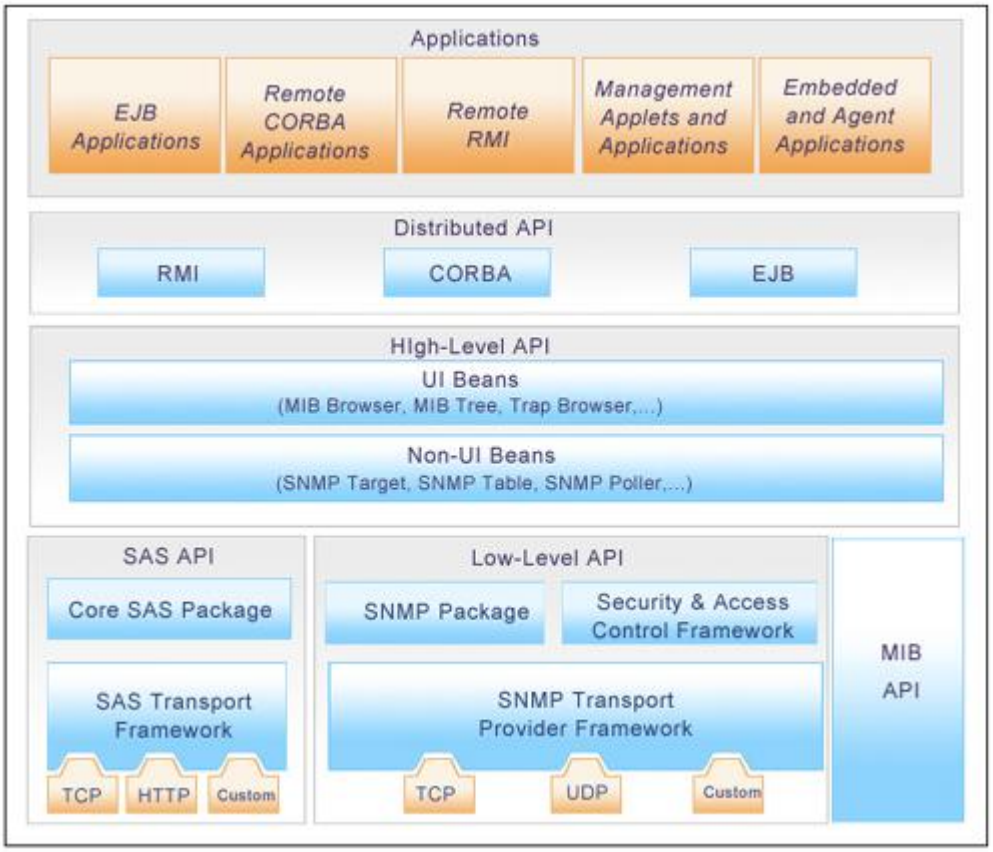


Figura 59 – Arquitectura API WebNMS (retirada de [83])

API SNMP de baixo nível

Esta API implementa as funções principais de SNMP. Inclui classes que facilitam a comunicação entre entidades SNMP oferecendo segurança nas mensagens e privacidade para as aplicações e *applets*. Contem também classes que podem ser utilizadas em applets de gestão, executáveis a partir de um browser. A API SNMP de baixo nível fornece a referência para a implementação de entidades SNMPv3 com *User-based Security Model* (USM) e *View-based Access Control Model* (VACM). USM tem por objectivo a autenticação, encriptação e desencriptação de pacotes SNMP e VACM tem a seu cargo a administração do acesso aos dados das MIB's [81-83].

MIB API

A MIB API transmite a informação sobre os dados disponíveis num determinado Agente SNMP. Esta API permite aos programas desenvolvidos em JAVA tirar partido ao máximo da informação contida nos ficheiros MIB. Facilita também o carregamento e descarregamento das MIB's nas aplicações e nas applets, para além de suportar uma série de funções que possibilitam alcançar as propriedades do objecto gerido. Os componentes são construídos através do uso de tipos de dados primitivos SNMP, disponíveis na API SNMP de baixo nível [81-83].

SAS API

Esta API fornece suporte para as applets construídas em JAVA contornarem as restrições de segurança existentes e provocadas pelos browsers. SAS permite à applet o envio e a recepção de pacotes SNMP para qualquer dispositivo gerido a partir do elemento de rede origem da applet. O servidor SAS necessita de ser instalado em conjunto com o servidor WEB em que a applet reside [81-83].

API SNMP de alto nível

Consiste em *beans User Interface* (UI) e *Non user Interface* que são construídos utilizando funções SNMP fornecidas pela API de baixo nível e pela MIB API. Estes componentes *bean* podem ser utilizados por qualquer ferramenta de construção de *beans* ligada à linguagem JAVA, ou manipulando directamente no próprio código [81-83].

API distribuída

Esta API facilita a construção de aplicações capazes de processar operações SNMP de forma distribuída, quer através de paradigmas que utilizem RMI, CORBA ou EJB. Neste trabalho foi proposta uma arquitectura e implementado um sistema onde as operações de gestão SNMP são executadas de uma forma distribuída, recorrendo a Agentes móveis e comunicações via Sockets [81-83].

Para que os Agentes OAA fossem capazes de executar as tarefas de gestão, tal e qual, como pretendido, foi fundamental perceber que funcionalidades esta API fornecia e que benefícios eram obtidos.

A tabela 5 apresenta as funcionalidades da API e o porquê da sua escolha, pois foram identificados os benefícios que essas mesmas funcionalidades disponibilizam [81-83].

Funcionalidades	Benefícios
Standards Abertos e Multi-Plataforma	Construída sobre tecnologias Standard ligadas à Internet, como JAVA, garantindo maior produtividade e interoperabilidade.
Suporte *Multilingue*	Comunicação baseada em SNMPv1, SNMPv2c e SNMPv3.
Segurança SNMPv3	Suporta os seguintes algoritmos de encriptação: HMAC-SHA-96, HMAC-MD5-96, CBC-DES e AES a 128 bits.
Análise Robusta da MIB: Suporte aos formatos SMIv1 e SMIv2	Consegue analisar as definições da MIB de qualquer fornecedor OEM. Independentemente da estrutura da informação da MIB estar em formato SMIv1 ou v2, a análise é feita de forma eficiente.
Carregamento de MIB	Opção para carregar definições de MIB a partir de um ficheiro pré compilado, de um ficheiro serializado, ou mesmo a partir de uma base de dados de forma a melhorar a performance.
SNMP *Beans*	Componentes *bean* de alto nível para facilitar o desenvolvimento das aplicações.
Suporte a IPv6	Fornece suporte para a conectividade SNMP em IPv6 com os elementos de rede, para além da já típica comunicação sobre IPv4.
Suporte a Base de Dados	Proporciona escalabilidade uma vez que é possível guardar as definições da MIB e os dados de configuração SNMPv3 em qualquer tipo de base de dados relacional, como MySQL ou ORACLE.
SNMP MIB Browser	Permite testar, monitorizar e gerir vários elementos de rede por SNMP através de uma rede. Os administradores de redes e sistemas conseguem carregar MIB's standard ou proprietárias e aceder aos dados de configuração sobre software e hardware.

Tabela 5 – Funcionalidades e Benefícios da API WebNMS

5.3.3 Implementação da Comunidade de Agentes

A comunidade de Agentes criada para a aplicação é constituída por catorze Agentes móveis, onde todos são subclasses da classe Agente (Agent):

- AgentRAM;
- AgentCPU;
- AgentPROC;

- AgentNE;
- AgentDISK;
- AgentNetwork403;
- AgentNetwork404;
- AgentNetwork405;
- AgentNetwork406;
- AgentSwitch20;
- AgentSwitch21;
- AgentSwitch22;
- AgentSwitch23;
- AgentMRTG.

Como já foi discutido anteriormente, a plataforma de Agentes OAA também tem um Agente que é lançado automaticamente sempre que a plataforma é iniciada, que é o OAA *Facilitator*. Este Agente não vai ser explicado nesta secção de forma detalhada como os outros, uma vez que em termos de implementação pouco ou nada é feito junto do código que rege este Agente, sendo no entanto, um Agente necessário e fulcral para o bom funcionamento da plataforma e da aplicação construída.

Agent

A classe "Agent" é a super classe de todos os Agentes. Esta classe tem como variáveis de instância a *myoaa* que é a variável que fica com a ligação à plataforma OAA que é do tipo *Liboaa*, tem a *libcomString* que é a *string* de apoio à criação da ligação à plataforma, o nome do Agente e os *solvables* existentes, o *AgentName* e *agentSolvable* respectivamente. A variável *agentName* vai ser o nome do Agente único em toda a plataforma e o *agentSolvables* vão ser as capacidades do Agente. Os métodos desta classe são o *ConnectFac()* que é o método que vai registar os Agentes na plataforma OAA (Comunidade de Agentes) e o *oaaDoEventCallback* que é o método que vai ser chamado quando é necessário resolver algum *solvable* de um determinado Agente. Todas estas variáveis de instância e métodos vão ser herdados pelos Agentes da aplicação já que os Agentes vão ser subclasses desta classe como podemos ver no modelo apresentado na figura 60.

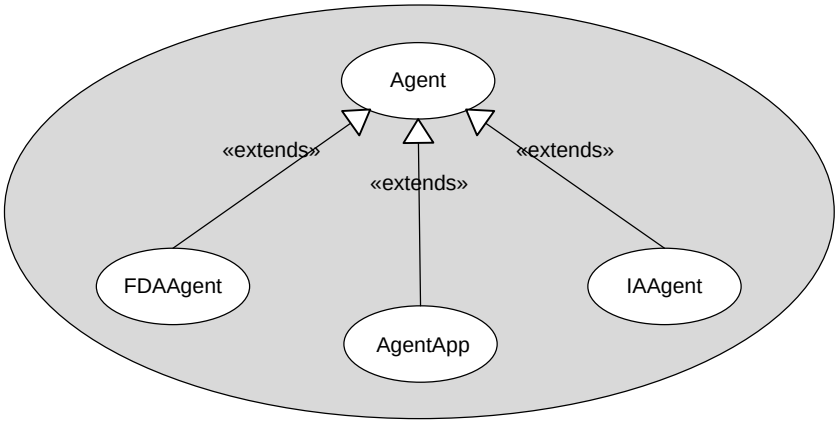


Figura 60 – Modelo da Comunidade de Agentes – Topo

Neste modelo estão presentes três Agentes de topo que herdam variáveis de instância e métodos do Agente principal, o "Agent". Os Agentes de detecção/correcção de falhas (aqui conhecidos como FDAAgent de *Fault Detection Agent*) e os de intervenção (aqui conhecidos como IAAgent de *Intervention Agent*) irão ser detalhados de seguida, para ser possível perceber o porquê da existência deste tipo de herança.

AgentApp

O "AgentApp" é o Agente interface. Este é o Agente de interface (ponte de comunicação) entre a comunidade de Agentes e a componente de interface e/ou vice-versa. O "AgentApp" é um Agente diferente, já que está fixo à componente de interface (que engloba a aplicação de Gestão de Agentes), no entanto, tem os *solvables* e resolve-os como qualquer outro Agente.

A comunicação entre as duas componentes é feita nos dois sentidos, tanto no sentido componente interface - comunidade de Agentes, como no sentido comunidade de Agentes componente de interface. Quando a comunicação é feita no sentido componente de interface comunidade de Agentes são chamados métodos que estão definidos nesta classe ("AgentApp"). Estes métodos vão enviar pedidos com *solvables* que serão realizados por Agentes com essas capacidades (*solvables*). Quando a comunicação é feita no sentido inverso, comunidade de Agentes componente de interface, são enviados pedidos de *solvables* por Agentes da comunidade de Agentes que serão realizados pelo "AgentApp". O "AgentApp" tem os *solvables* (capacidades) para tratar todos os pedidos para a comunicação feita neste sentido. A comunicação neste sentido é geralmente feita quando existe um Agente que faz uma tarefa autonomamente e que quer comunicar os seus resultados.

O "AgentApp" foi ainda dotado de características que lhe permitem comportar-se como o Agente de diagnóstico, explicado na secção 5.2.3, uma vez que ele está vinculado à interface com o utilizador e o mesmo pode pedir a criação de vários tipos de Agentes, para além da execução de

tarefas de gestão através dos Agentes existentes na comunidade. Se um Agente possuir algum tipo de relatório e o quiser apresentar ao utilizador final, este tipo de dados passa pelo "AgentApp". Este Agente terá como variáveis de instância o "frontEnd" que identifica a janela de comunicação com o utilizador e o "id" que identifica o Agente. Terá um *solvable* que é o writeMSG. Este *solvable* recebe como parâmetro o "id" do Agente que iniciou a comunicação com a comunidade e a mensagem que o Agente da comunidade pretende fazer passar para a interface. De seguida será referida a comunicação no sentido componente de interface comunidade de Agentes. Esta classe "AgentApp" tem alguns métodos auxiliares que visam corresponder a pedidos feitos para a comunidade de Agentes. Um dos mais importantes é o seguinte:

- *requestReport*() - tem como objectivo o envio de um pedido para a comunidade de Agentes para algum Agente resolver o *solvable presentReport*. Este método será activo quando o utilizador desejar ser informado visualmente na interface da aplicação, do relatório produzido no final das actividades de um determinado tipo de Agente; isto porque a informação guardada pelos Agentes, à medida que percorrem um itinerário e executam tarefas de gestão não é necessariamente, nem obrigatoriamente passada ao utilizador a nível da interface.

Na figura 61, é apresentado o modelo da comunidade de Agentes na perspectiva dos Agentes de detecção/correcção de falhas. O FDAAgent é uma classe que representa todas as tarefas de gestão que a aplicação poderá executar com o auxílio de Agentes de detecção/correcção de falhas. Como é possível visualizar na figura, esta classe é classe-mãe de outras cinco classes.

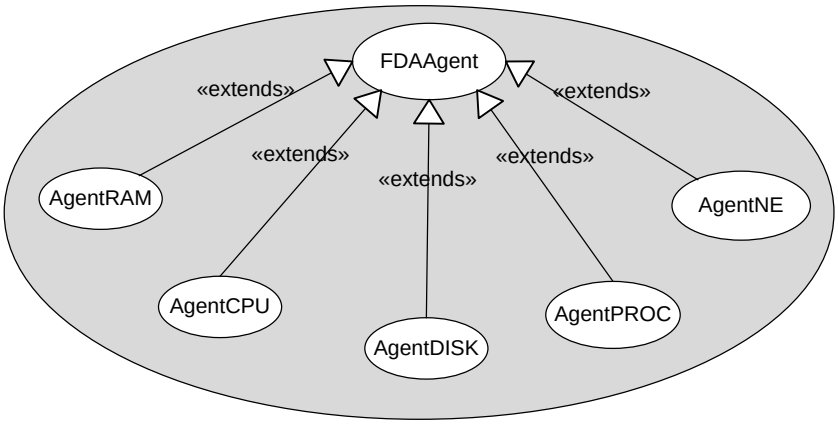


Figura 61 – Modelo da Comunidade de Agentes – Detecção e Correcção de Falhas

A próxima figura, figura 62, ilustra a fracção da comunidade de Agentes que nos leva aos Agentes de intervenção (IAAgent). O IAAgent é uma classe que representa todas as tarefas de gestão que a aplicação poderá executar com o auxílio de Agentes de intervenção. Esta classe é classe-mãe de outras nove classes.

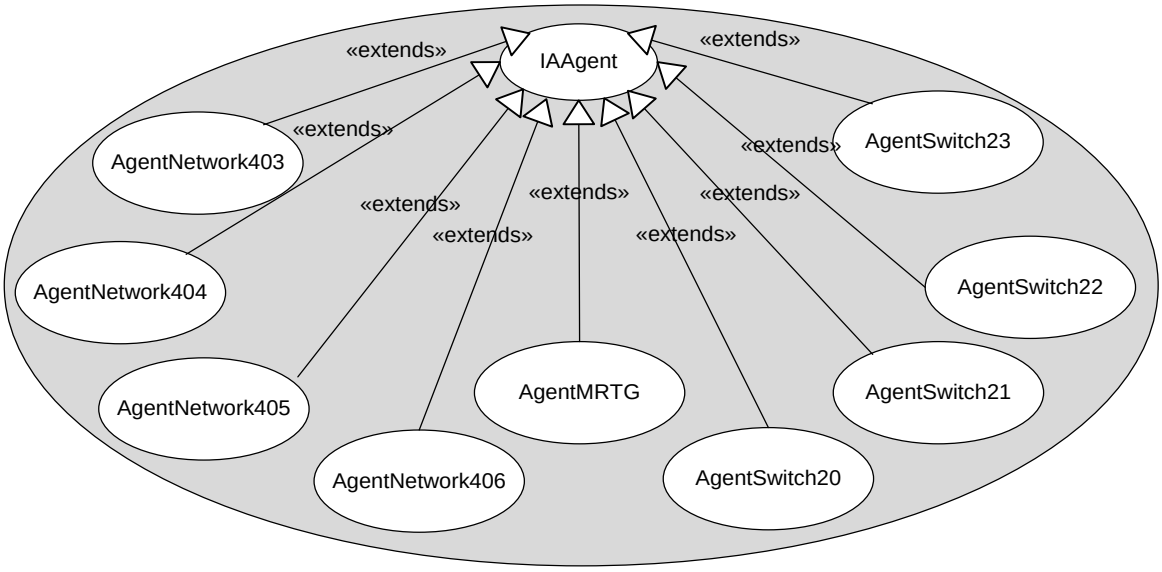


Figura 62 – Modelo da Comunidade de Agentes – Intervenção

De seguida irá ser descrita a estrutura de cada um dos vários tipos de Agentes que encontramos não só na figura 61 mas também os restantes, mencionados no inicio desta secção. Serão indicadas as variáveis de instância mais importantes e descrita a estrutura de cada Agente. De realçar, que todos os Agentes estendem uma classe de nome *NE*, responsável por guardar informações básicas, mas fundamentais a cada passagem pelos elementos de rede. Sendo assim, encontramos as seguintes variáveis de instância comuns a todos os Agentes:

- *ArrayList* com objectos do tipo *NE*, em que são armazenadas informações comuns a todos os elementos de rede, que façam parte do itinerário de um determinado Agente:
  - ✓ *descricao*, onde é armazenada uma descrição sobre o elemento de rede, presente na MIB*;*
  - ✓ *ip*, contendo o IP do elemento de rede*;*
  - ✓ *upTime*, onde encontramos há quanto tempo o elemento de rede se encontra em funcionamento*;*
  - ✓ *name*, que possui sempre o nome que identifica o elemento perante a rede de dados;
  - ✓ *date*, com informação sobre dia, hora, minutos e a que segundo foram iniciadas as tarefas de gestão.

AgentRAM

Este Agente é uma subclasse da classe FDAAgent, estando responsável por monitorizar a quantidade de RAM utilizada num determinado elemento de rede, ou se desejável, num conjunto de elementos de rede pertencentes a um determinado segmento. Por cada salto que o Agente dê, toma

nota de várias componentes, nomeadamente a descrição do elemento de rede, há quanto tempo se encontra ligado, a data e hora a que as tarefas de gestão se iniciaram e o IP. É capaz de calcular quantidades em bytes e utilizações em percentagens da memória, assim como ser proactivo, reagindo e tomando decisões perante determinados eventos, como o facto de encontrar muito ou pouco consumo de RAM.

Este Agente em particular contém ainda algumas variáveis de instância, das quais se destacam:

- *ArrayList* com objectos do tipo *RAM*, onde são armazenadas todas as informações relevantes, à medida que cada um dos elementos de rede são percorridos:
  - ✓ *valorTotal*, responsável por identificar a quantidade de RAM existente no elemento de rede;
  - ✓ *valorUtilizado*, com informação em bytes, da quantidade de RAM utilizada;
  - ✓ *percentUtilizado*, em que após o cálculo dos valores mencionados, é também calculado o valor utilizado em percentagem, para ajudar do ponto de vista histórico e visual.
  - ✓ *valorLiberto*, calculado após a execução do Agente no elemento de rede.

O Agente tem um *ArrayList* de objectos *MIB* devido à existência dos vários elementos de rede que vai consultar, isto é, pode ser útil ou necessário verificar uma determinada OID de uma MIB no elemento de rede "A", mas no elemento de rede "B" a consulta ser efectuada sobre um outro OID.

No que a *solvables* diz respeito, o Agente possui três:

- *presentReport*, em que toda a informação que foi agregada durante o seu trajecto é compilada e apresentada visualmente na interface com o utilizador;
- *checkRAM*, que possui dois parâmetros (*idAgApp* e *segment*), em que o primeiro identifica o "id" do "AgentApp" que iniciou a comunicação entre a componente de interface e a comunidade de Agentes. O segundo identifica o segmento de rede para onde o Agente se deve dirigir. Após o pedido para resolver este *solvable* ser recebido, o "AgentRAM" inicia a migração segundo um itinerário baseado no segmento de rede que lhe foi passado e executa as tarefas de gestão associadas;
- *sayHello* que possui um parâmetro (*Agt*), que o Agente utilizada para se "apresentar" na interface ao utilizador, isto é, se quisermos saber se um Agente deste tipo está presente na comunidade, podemos simplesmente pedir-lhe para dizer "*hello, my name is …"*.

Esta classe é também constituída por vários métodos auxiliares que têm como objectivo ajudar na execução dos *solvables* do Agente. Os métodos são os seguintes:

- *checkNextHop*, verifica a possibilidade, ou não, de o Agente conseguir efectuar a migração para o próximo elemento de rede na sua lista, iniciando uma análise simples, que visa obter dados sobre a conectividade para o seu próximo "salto";
- *move*, que é chamado quando desejamos que o Agente em questão se desloque na rede para o próximo elemento, através do uso de serialização e comunicação com *sockets*, como será possível detalhar na secção 5.3.6;
- *getValues*, método que visa recolher dados fulcrais sobre memória RAM junto da MIB;
- *usage*, responsável por calcular a quantidade de RAM que está a ser utilizada no momento da consulta. Este método recorre a processos mais elaborados, uma vez que não é possível obter os dados da MIB de uma forma directa, envolvendo cálculos em inúmeras vezes. A quantidade é devolvida em bytes;
- *percentage*, calcula a percentagem de utilização da quantidade de RAM, com recurso ao valor adquirido pelo método anterior;
- *freeRAM*, está preparado para executar comandos no Sistema Operativo que visam libertar RAM que se encontra alocada mas não está a ser utilizada, entre outros;
- *usageDetails*, tem como objectivo, quando o Agente nota que o consumo de memória é excessivo, retirar da MIB informação sobre que processos estão em execução, sendo que por cada um é mapeada a quantidade de RAM que estão a consumir;
- *requestMRTG*, visa chamar um "AgentMRTG" se o utilizador assim o entender, por forma a este estudar, durante um período de tempo pré definido, a quantidade de memória RAM que é consumida no elemento de rede, conseguindo no fim do estudo gerar um gráfico localmente, transferindo-o pela rede para que este seja criado na estação gestora;
- *compileReport*, baseando-se em todas as operações que foram efectuadas, faz um resumo do que foi consultado, em que elementos de rede e as acções que foram tomadas, criando um relatório final da actividade do Agente.

AgentCPU

Este Agente é uma subclasse da classe FDAAgent, estando responsável por monitorizar a quantidade de CPU utilizada num determinado elemento de rede, ou se desejável, num conjunto de elementos de rede pertencentes a um segmento da mesma. Tal como o anterior, este também tem uma estrutura que lhe permite guardar os mesmos dados iniciais, a cada presença num elemento de rede. Das variáveis de instância, destacam-se:

- *ArrayList* com objectos do tipo *CPU* onde são armazenadas todas as informações relevantes, à medida que cada um dos elementos de rede é percorrido. Este objecto possui as seguintes variáveis de instância:
  - ✓ objecto do tipo *CPUdetails* de nome *carga*, em que depois da consulta na MIB, é armazenado o consumo, em percentagem, por cada núcleo do processador em causa. Em todos os elementos de rede encontraremos processadores com vários núcleos, e é vital saber a utilização que cada um possui.
  - ✓ *media*, com o consumo em média do CPU, depois de analisados todos os núcleos;
  - ✓ *percentUsado*, similar ao anterior mas com o cálculo da percentagem, baseando-se também na informação obtida a partir do objecto *CPUdetails*;

O Agente tem também um *ArrayList* chamado *MIB* pelas mesmas razões já descritas no AgentRAM.

No que a *solvables* diz respeito, o Agente possui três:

- *presentReport* e *sayHello* já descritos nesta secção;
- *checkCPU*, que possui uma estrutura similar com os mesmos dois parâmetros do *solvable checkRAM* do "AgentRAM" (*idAgApp* e *segment*).

Também vários dos métodos auxiliares do Agente são iguais aos do anterior, mas apenas iremos abordar com detalhe os que são novos, ou os que têm um funcionamento e objectivos ligeiramente diferentes. Os métodos são os seguintes:

- *checkNextHop*;
- *move*;
- *getValues*, método que visa recolher dados fulcrais sobre o CPU junto da MIB, entre eles as percentagens de utilização por núcleo;
- *usageDetails*, tem como objectivo, quando o Agente nota que o consumo é excessivo, retirar da MIB informação sobre processos que estão em execução, sendo que por cada um é mapeada a quantidade de CPU que está a consumir, sendo posteriormente elaborado um relatório para o administrador de redes;
- *requestCpuStop*, que envia um pedido para a interface com o utilizador, inquirindo o mesmo sobre se deseja terminar algum processo e qual;
- *freeCPU*, está preparado para executar comandos no Sistema Operativo que visam parar processos desnecessários (apenas se o utilizador especificar quais);
- *requestMRTG*, visa chamar um AgentMRTG, por forma a este estudar, durante um período de tempo pré definido, o comportamento de todos os núcleos do(s) processador(es), gerando localmente um gráfico, transferindo-o pela rede para a estação gestora;

- *compileReport*.

AgentDISK

Este Agente é uma subclasse da classe FDAAgent, tendo por objectivos a análise e correcção de situações deficitárias no que a espaço em disco concerne, nos elementos de rede em que tal se aplique. Como os dois Agentes anteriores, guarda sempre informações gerais por cada elemento de rede em que execute tarefas de gestão. Possui as seguintes variáveis de instância:

- *ArrayList* com objectos do tipo *DISK* onde são armazenadas todas as informações relevantes, à medida que cada um dos elementos de rede é percorrido. São de destacar neste objecto as seguintes variáveis de instância:
  - ✓ *ArrayList* com objectos do tipo *DiskDetails*. Este objecto tem como fim reunir dados por cada partição que o disco contenha. Aqui são armazenados, por cada partição consultada, vários componentes, tais como: descrição da partição, o seu tamanho, a quantidade utilizada e a percentagem de utilização.

O Agente tem também um *ArrayList* chamado *MIB* pelas mesmas razões já descritas nos Agentes anteriores.

Os *solvables* são três:

- *presentReport*, em que toda a informação que foi agregada durante o seu trajecto é compilada e apresentada visualmente na interface com o utilizador;
- *checkDISK*, que possui uma estrutura similar com os mesmos dois parâmetros dos *solvables* descritos nos Agentes anteriores (*idAgApp* e *segment*);
- *sayHello*.

Esta classe é também constituída por vários métodos auxiliares que têm como objectivo ajudar na execução dos *solvables* do Agente. Os métodos são os seguintes:

- *checkNextHop*;
- *move*;
- *getValues*, método que visa recolher dados fulcrais sobre todas as partições do disco em causa, junto da MIB;
- *usage*, responsável por calcular a quantidade de espaço alocado no momento da consulta. Este método recorre a processos mais elaborados, uma vez que não é possível obter os dados da MIB de uma forma directa, envolvendo cálculos em inúmeras situações. A quantidade é devolvida em bytes, para cada partição;
- *percentage*, calcula as percentagens de utilização com recurso aos valores adquiridos pelo método anterior;

- *freeDISK*, executa comandos no Sistema Operativo que visam libertar espaço, recorrendo a limpeza de ficheiros temporários, determinadas extensões que se sabem ser de ficheiros que podem ser descartados, entre outros;
- *compileReport*.

AgentPROC

Este Agente é uma subclasse da classe FDAAgent, possuindo um papel decisivo na monitorização de processos em execução nos elementos de rede. Este Agente tem de conseguir identificar se um determinado conjunto de processos afecto a um elemento de rede está, ou não, no estado desejável. Se encontrar processos parados, quando o desejável era que estivessem em execução, deve reagir, tentando repor o normal funcionamento do processo em questão. O processamento é todo local, isto é, a verificação de cada um dos processos para cada um dos elementos de rede é feita nestes últimos. Para além da estrutura similar aos Agentes anteriores vamos analisar os detalhes deste Agente em particular:

- *ArrayList* com objectos do tipo *Processes* onde são armazenadas todas as informações relevantes, à medida que cada um dos elementos de rede é percorrido. Este objecto possui ainda as seguintes variáveis de instância:
  - ✓ *ArrayList* com objectos do tipo *ProcessesDetails* de nome *details*. Tal como, no *AgentDISK*, o primeiro *ArrayList* contém informação útil sobre cada elemento de rede percorrido, mas para além disso, é necessário outro, que nos informe da descrição de cada um dos processos, estado, e OID's a consultar por cada elemento de rede. Na prática, o segundo *ArrayList* contém informação detalhada processo a processo para cada elemento de rede.

O Agente tem também um *ArrayList* chamado *MIB* pelas mesmas razões já descritas nos Agentes anteriores.

Encontramos três *solvables*:

- *presentReport* e *sayHello* já descritos nesta secção;
- *checkPROC*, que possui uma estrutura similar com os mesmos dois parâmetros do *solvable* dos Agentes anteriores (*idAgApp* e *segment*).

Seguem-se os métodos auxiliares do Agente:

- *checkNextHop*;
- *move*;
- *getValues*, método que visa recolher dados sobre cada um dos processos junto da MIB;

- *compileState*, tem como objectivo identificar para cada um dos processos no elemento de rede em causa o seu estado actual;
- *repair*, método para tentar a recuperação de um processo que deveria estar em funcionamento, mas que por alguma razão não está em execução;
- *compileReport*.

AgentNE

Este Agente é uma subclasse da classe FDAAgent, com o objectivo de corrigir problemas de conectividade nos vários servidores do GECAD. Muitas das vezes é possível não apenas corrigir como evitá-los, uma vez que, tanto a cache de DNS como a de ARP encontram-se armazenadas na RAM, o que pode trazer problemas devido ao tempo que este tipo de elementos de rede passa conectado à rede. Para além da estrutura similar aos Agentes anteriores vamos analisar os detalhes deste Agente em particular:

- *ArrayList* com objectos do tipo *networkAdapter* onde são armazenadas todas as informações relevantes, à medida que cada um dos elementos de rede é percorrido. Este objecto possui ainda as seguintes variáveis de instância:
  - ✓ *description*, com uma descrição da placa de rede em questão;
  - ✓ *type*, com o tipo de ligação;
  - ✓ *physicalAddress*, contendo o endereço físico;
  - ✓ *speed*, com a velocidade da interface;
  - ✓ *inDiscards*, pacotes descartados à entrada da interface, mesmo não contendo qualquer erro;
  - ✓ *outDiscards*, pacotes descartados à saída da interface, mesmo não contendo qualquer erro;
  - ✓ *inErrors*, número de pacotes com erros, o que preveniu a sua passagem a uma camada protocolar mais alta;
  - ✓ *outErrors*, quantidade de pacotes que devido a erros, não puderam ser transmitidos.

O Agente tem também um *ArrayList* chamado *MIB* pelas mesmas razões já descritas nos Agentes anteriores.

Encontramos três *solvables*:

- *presentReport* e *sayHello* já descritos nesta secção;
- *checkNE*, que possui uma estrutura similar com os mesmos dois parâmetros do *solvable* dos Agentes anteriores (*idAgApp* e *segment*).

Seguem-se os métodos auxiliares do Agente:

- *checkNextHop*;
- *move*;
- *getValues*, método que recolhe algumas informações relacionadas com o adaptador de rede;
- *requestMRTG*, tem como objectivo, mediante algumas premissas, pedir a um Agente do tipo MRTG que faça um "estudo" sobre a carga a que a placa de rede tem sido exposta, durante um determinado período de tempo, até que seja possível materializar visualmente os dados na estação gestora;
- *repair*, método com a finalidade de limpar todo o tipo de informação relacionado com DNS e ARP, renovando as ligações de rede, sem colocar em causa a conectividade;
- *compileReport*.

AgentSwitch

Cada um dos Agentes com o prefixo *AgentSwitch* representa uma subclasse da classe *IAAgent*, estando responsáveis por monitorizar vários parâmetros das ligações de rede do seu *switch* correspondente. Na prática, o *AgentSwitch20*, dado de seguida como exemplo, está encarregado de monitorizar o elemento de rede (nestes casos é um *switch*) que possui o IP "192.168.2.20". Nenhum destes Agentes tem a seu cargo a monitorização de mais do que um elemento de rede, não deixando de ser móveis, uma vez que se deslocam para um local físico na rede o mais próximo possível do alvo, trabalhando localmente num PC ou Workstation, conectado directamente ao *switch*. Tal como os antecessores, todos estes Agentes guardam a descrição do elemento de rede, há quanto tempo se encontra ligado, a data e hora a que as tarefas de gestão se iniciaram e o IP. Começam a trabalhar a partir destes dados com objectivos bem definidos: criar um relatório o mais completo possível, onde é incluída informação obtida em cada uma das portas do *switch*.

Este Agente em particular contém ainda algumas variáveis de instância, das quais se destacam:

- *ArrayList* com objectos do tipo *Switch*, onde são armazenadas algumas informações básicas mas relevantes:
  - ✓ *ArrayList* com objectos do tipo *connectionDetails*, onde por cada *switch* vemos todas as características de todas as interfaces:
    - *ifDescr*, contendo uma descrição da interface;
    - *ifSpeed*, identificando a capacidade actual da interface em termos de velocidade;
    - *ifOperStatus*, diz-nos se a interface está ligada a um elemento de rede, ou não;

- *ifIinOctects*, com o número total de octetos recebidos na interface;
- *ifInDiscards*, representa o número total de pacotes descartados à entrada da interface, apesar de não existirem erros na sua construção;
- *ifInErrors*, número total de pacotes descartados à entrada da interface, devidos a erros;
- *ifOutOctects*, número total de pacotes transmitidos a partir da interface;
- *ifOutDiscards*, representa o número total de pacotes descartados à saída da interface apesar de não existirem erros na sua construção;
- *ifOutErrors*, com o número total de pacotes rejeitados à saída da interface devido a erros;

✓ *ArrayList* MIB, com as informações base das OID's que temos de consultar.

No que a *solvables* diz respeito, o Agente possui três:

- *presentReport* e *sayHello* já descritos nesta secção;
- *checkSwitch*, que possui dois parâmetros (*idAgApp* e *switch*), em que o primeiro identifica o *id* do "AgentApp" que iniciou a comunicação entre a componente de interface e a comunidade de Agentes. O segundo identifica o *switch* para onde o Agente se deve dirigir;

Esta classe é também constituída por vários métodos auxiliares que têm como objectivo ajudar na execução dos *solvables* do Agente. Os métodos são os seguintes:

- *checkNextHop*;
- *move*;
- *getValues*, método que visa recolher dados junto da MIB, para cada uma das interfaces do *switch*;
- *analise*, método que tem como objectivo verificar os dados mais importantes recebidos da MIB, executando um diagnóstico preliminar, nomeadamente junto da informação que diz respeito à não transmissão de pacotes devido a erros ou falta de memória;
- *requestMRTG*, visa chamar um "AgentMRTG" se o "AgentSwitch" encontrar demasiados problemas na analise preliminar, por forma a este estudar, durante um período de tempo pré definido, a evolução de alguns dos parâmetros que necessitem de atenção;
- *compileReport*.

AgentNetwork

Cada um dos Agentes com o prefixo *AgentNetwork* representa uma subclasse da classe *IAAgent*, estando responsáveis por verificar problemas de conectividade nos *switch's* da rede que não possuem IP, sem qualquer tipo de acesso para gestão. Estes Agentes deslocam-se, mediante o seu segmento, para um elemento de rede directamente conectado a este tipo de *switch's*. No fundo,

tentam recriar o trajecto que um pacote de rede faria se tivesse como origem ou destino um equipamento ligado ao *switch* em questão. O objectivo é a avaliação da conectividade deste tipo de equipamentos, pois muitas vezes um servidor ou workstation não está alcançável, conduzindo à interpretação de que o problema está no próprio *switch* em que se encontram ligados.

Este Agente tem algumas variáveis de instância, das quais se destacam:

- *ArrayList* com objectos do tipo *networkSegment*, onde são armazenadas algumas informações básicas mas relevantes:
  - ✓ *ip*, identificando o endereço IP do elemento de rede em que actualmente se encontra;
  - ✓ *result*, com o resultado do teste de conectividade para o próximo salto.

No que a *solvables* diz respeito, o Agente possui três:

- *presentReport* e *sayHello* já descritos nesta secção;
- *checkNetwork*, que possui dois parâmetros (*idAgApp* e *segment*), em que o primeiro identifica o *id* do "AgentApp" que iniciou a comunicação entre a componente de interface e a comunidade de Agentes. O segundo identifica o segmento de rede onde o Agente deve focar a sua atenção;

Esta classe é também constituída por vários métodos auxiliares que têm como objectivo ajudar na execução dos *solvables* do Agente. Os métodos são os seguintes:

- *checkNextHop*;
- *move*;
- *testConectivity*, método para efectuar testes de conectividade para com o próximo elemento de rede destino e guardar os resultados;
- *compileReport*.

AgentMRTG

O "AgentMRTG" é um dos Agentes subclasse da classe IAAgent. Com a excepção da situação em que são utilizados em testes ou envios premeditados através da interface da aplicação, este Agente está normalmente à espera que sejam solicitados os seus serviços. Em particular devemos destacar o "AgentCPU", o "AgentRAM", "AgentNE" e "AgentSwitch", uma vez que são os que podem necessitar do auxílio do "AgentMRTG", embora para diferentes fins. Este Agente guarda a descrição do elemento de rede, há quanto tempo se encontra ligado, a data e hora a que as tarefas de gestão se iniciaram e o IP. Recolhidos estes detalhes e consoante o seu alvo e tipo de funções, iniciam o processo de análise que culmina na criação de gráficos, para que se possa dar uma ideia

mais concisa e ao mesmo tempo sugestiva, devido ao formato visual, daquilo que foi objecto de estudo.

Este Agente contém variáveis de instância, das quais se destacam:

- *ArrayList* com objectos do tipo *MRTG*, onde são armazenadas algumas informações básicas mas relevantes. Dependendo do tipo de análise que o Agente necessite de efectuar, terá naturalmente, atributos de acordo com esse estudo:
  - ✓ No caso do objecto em questão ser um CPU, elementos importantes como a sua carga são importantes para a construção do gráfico;
  - ✓ Na RAM, vai analisar e guardar a quantidade em uso, em percentagem e/ou bytes;
  - ✓ Para os pedidos efectuados pelo "AgentNE", vai calcular a taxa de ocupação dos adaptadores de rede do elemento em causa;
  - ✓ Finalmente, no caso do pedido ter sido enviado por um "AgentSwitch", a sua tarefa é mais complexa, pois poderá ter de estudar a carga em várias portas, para além da quantidade de pacotes com erros descartados à entrada ou saída de cada uma delas;
  - ✓ *ArrayList* MIB, com as informações base das OID's que temos de consultar.

O Agente possui os seguintes *solvables*:

- *presentReport* e *sayHello* já descritos nesta secção;
- *studyCPU*, que possui dois parâmetros (*idAgApp* e *ip*), onde o primeiro identifica o *id* do "AgentApp" que iniciou a comunicação entre a componente de interface e a comunidade de Agentes, ou entre um Agente ("AgentCPU", "AgentRAM", "AgentNE" ou "AgentSwitch") e o "AgentMRTG" directamente. O segundo identifica o *ip* para onde o Agente se deve dirigir;
- *studyRAM*, com a mesma estrutura do *checkCPU*;
- *studyNE*, com os dois parâmetros habituais (*idAgApp* e *ip*), acrescidos de um terceiro que indica com mais detalhe quais os objectivos do estudo pedido;
- *studySwitch*, com os dois parâmetros habituais (*idAgApp* e *ip*), mas com um terceiro parâmetro adicional (*tasks*), indicando que informações deve retirar da MIB, definindo o seu objecto de estudo (pode ser a carga de uma porta de um *switch*, podem ser todas, como podemos desejar que o Agente investigue a quantidade de erros ao longo do tempo em algumas das portas).

Esta classe é também constituída por vários métodos auxiliares que têm como objectivo ajudar na execução dos *solvables* do Agente. Os métodos são os seguintes:

- *checkNextHop*;

- *move*;
- *getValues*, método que visa recolher dados junto da MIB, para cada um dos objectos de estudo;
- *createFiles*, método que tem como objectivo a criação de *logs* e de um conjunto de ficheiros em que são armazenados os gráficos;
- *zipFiles*, comprime o directório em que se encontram os ficheiros anteriores, para de seguida os transportar consigo;
- *showGraphic*, faz uso dos ficheiros que criou, para num simples browser ou numa segunda janela da aplicação, mostrar o conteúdo, neste caso, o(s) gráfico(s) correspondente(s);
- *compileReport.*

5.3.4 Implementação da Interface

A aplicação de Gestão de Agentes consiste numa aplicação simples que permite ao administrador do sistema lançar todos os Agentes necessários para a execução de tarefas de gestão disponíveis. Nesta secção irá ser descrita a componente de interface com o utilizador. Serão ilustradas algumas situações de interacção, principalmente as que possibilitam a utilização das funcionalidades do sistema. A próxima figura, figura 63, mostra a interface.

Figura 63 – Interface do Sistema

A interface foi pensada com base em parâmetros de simplicidade que permitissem uma compreensão rápida e eficaz do objectivo da mesma.

No lado esquerdo da janela encontram-se disponíveis todas as opções que permitem criar qualquer tipo de Agentes suportados pela aplicação. Encontramos nove *Combo Boxes*, onde é dada a possibilidade ao administrador de redes de escolher quais os Agentes que deseja criar, de modo a que estes se juntem à comunidade. Surge de seguida um botão com o texto "*Set All to None*" pela simples razão de que a determinada altura, podemos desejar criar apenas um Agente para um determinado segmento ou teste. Dessa forma, não temos de escolher a opção "*None*" manualmente para todas as caixas com as opções de criação dos Agentes.

De seguida, e após a determinação dos Agentes que se pretendem criar, resta dar a instrução "*Create Agents*". Esta opção avalia as escolhas efectuadas e inicia o processo de criação de todos os Agentes.

Podem ser criados novos Agentes em qualquer altura. Junto do botão "*Create Agents*", encontram-se outros dois botões importantes, um responsável por solicitar o envio dos Agentes para a rede e um que permite saber quais os Agentes registados junto do facilitador, ou seja, na comunidade.

Os botões de teste existem para permitir avaliar o sistema aqui proposto e implementado. A próxima figura, figura 64, ilustra quais as opções disponíveis para a criação dos Agentes em cada *Combo Box*, exceptuando os "AgentNetwork" e "AgentSwitch". De realçar que foram definidos três segmentos de rede em que os restantes Agentes actuam. O segmento 1 é composto pelos elementos de rede "Zeus", "Zeus1" e "Gecadcore", máquinas ligadas ao controlo do domínio. O segundo está ligado aos servidores aplicacionais, o terceiro suporta os servidores orientados a aplicações WEB. Quando escolhemos um segmento, o Agente em causa terá de passar por todos os elementos de rede do respectivo segmento.

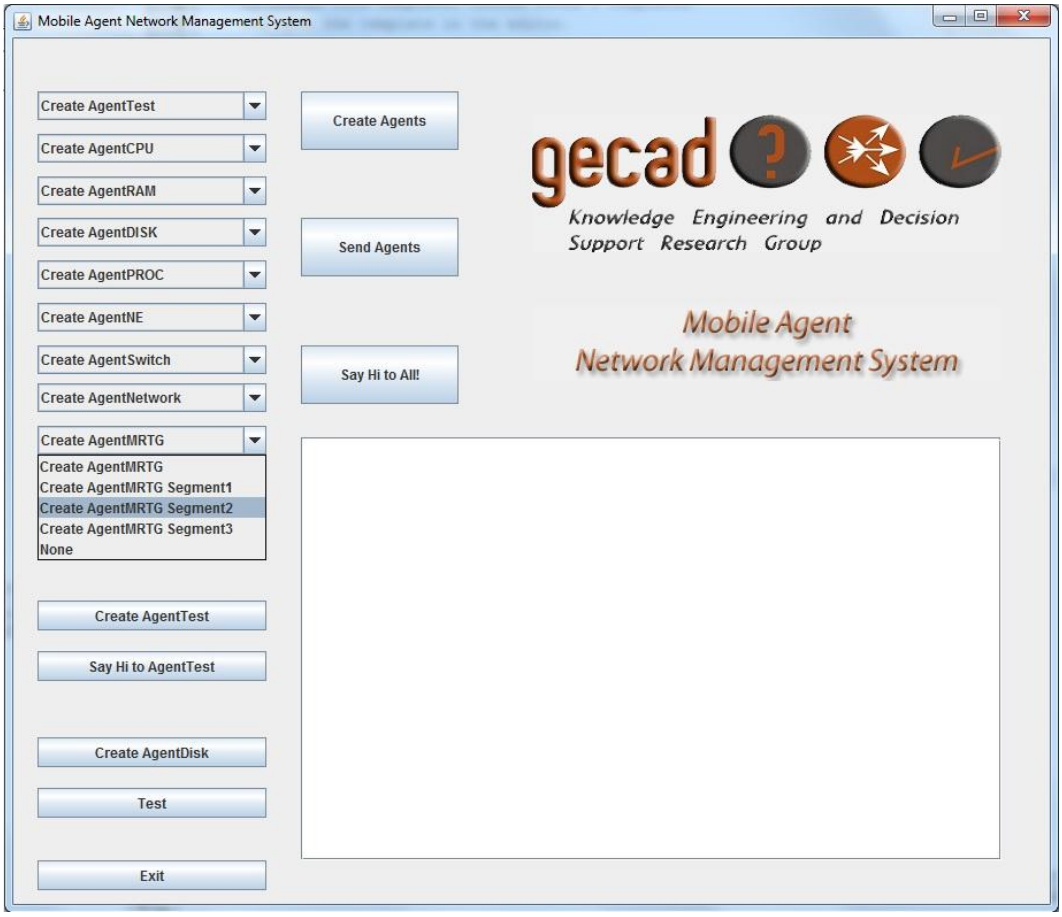


Figura 64 – Interface do sistema – Opções

Qualquer *Combo Box* possui cinco opções:

- criar um Agente para enviar para um único elemento de rede;
- criar um Agente com o destino segmento de rede 1;
- criar um Agente com o destino segmento de rede 2;
- criar um Agente com o destino segmento de rede 3;
- em branco, se não pretendemos adicionar à comunidade nenhum Agente deste tipo.

Relativamente aos *AgentSwitch*, possuem também cinco opções, uma para cada *switch*, acrescido da opção *None*. Os "AgentNetwork", possuem da mesma forma, quatro opções com vista a cada uma das salas de trabalho (I403, I404, I405 e I406) para além da opção *None*.

A próxima figura, figura 65, ilustra algumas das mensagens que podem surgir na caixa de output principal. Como podemos ver, vários Agentes foram criados, o sistema notifica o facto com uma mensagem de sucesso, ou falha. Pressionando o botão "*Say Hi to All*" conseguimos obter resposta de todos os Agentes da comunidade, identificando-se cada um deles com o nome e capacidades, com que se registaram junto do Agente facilitador.

Figura 65 – Interface do Sistema - Output

5.3.5 Detalhes da implementação

A solução por nós proposta assenta no desenvolvimento de uma aplicação de gestão de redes baseada em Agentes móveis, o que implicou dividir o problema, uma vez que se misturam paradigmas. Para tal, foi implementado um sistema que está dividido em três camadas que se apresentam de seguida:

- camada de interface, que representa a interacção do sistema com o utilizador através de interfaces gráficos;
- camada lógica, compreende toda a implementação que visa resolver as funcionalidades do sistema;
- camada de acesso a dados, representa todo o acesso a dados que serão fundamentais para o funcionamento do sistema.

A próxima figura, figura 66, ilustra esta divisão por camadas do sistema.

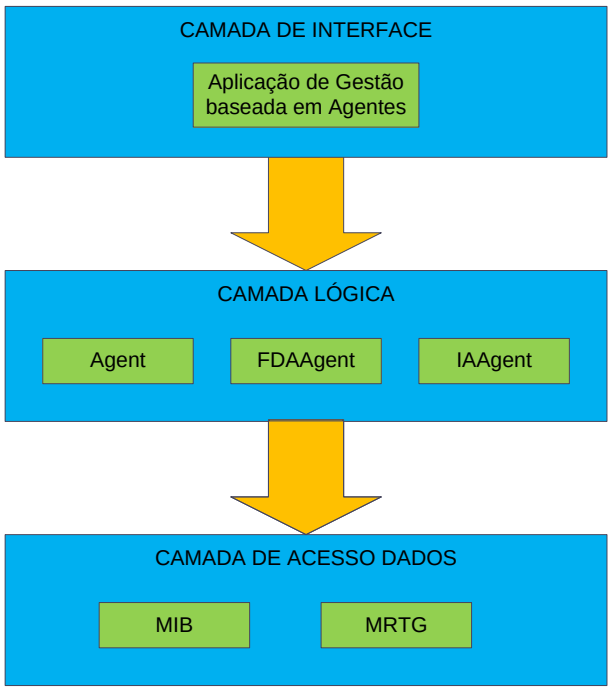


Figura 66 – Camadas do Sistema

A camada de interface está presente na aplicação de gestão de Agentes, uma vez que todo o tipo de interacções com o utilizador são feitas através desta. Na camada lógica, todas as funcionalidades do sistema são executadas por Agentes, sendo na figura anterior representados pelas suas classes-mãe, discutidas na secção 5.3.4. A camada de acesso a dados contém as MIB's, de onde os Agentes obtêm a informação para posteriormente efectuarem as suas funções [84]. Esta camada contém também os dados MRTG, isto é, todo aquele tipo de informação que é construída, armazenada e transportada pela rede com vista à criação de gráficos MRTG.

A divisão do problema está também patente quando analisamos os pacotes de classes existentes. Existem quatro grandes conjuntos ou pacotes de classes:

- *gecad.imanm.agents*;
  - ✓ onde encontramos as classes mãe IAAgent e FDAAgent, que por sua vez contêm todas as classes que correspondem aos Agentes móveis, em conjunto com todas as suas capacidades para executar tarefas de gestão;
- *gecad.imanm.gui*;
  - ✓ toda a interacção com o utilizador passa por aqui, seja através da interface principal, ou de classes responsáveis por apresentar dados noutras janelas;
- *gecad.imanm.networkElements*;
  - ✓ os Agentes são móveis e como tal, existem vários atributos e variáveis de instância, que necessitam de ser guardados no momento em que partem para o seu destino, assim como, quando se deslocam entre elementos de rede. Por cada tipo de

Agente, existe uma classe auxiliar dentro deste pacote, responsável por apoiar cada um deles no armazenamento e transporte de dados;

- *gecad.imanm.util*;
  - ✓ tudo o que é relevante para testes encontra-se neste pacote de classes. Sendo também algumas dessas classes encarregues de tarefas que apesar de incompletas em termos de desenvolvimento, já se encontram em uso na aplicação.

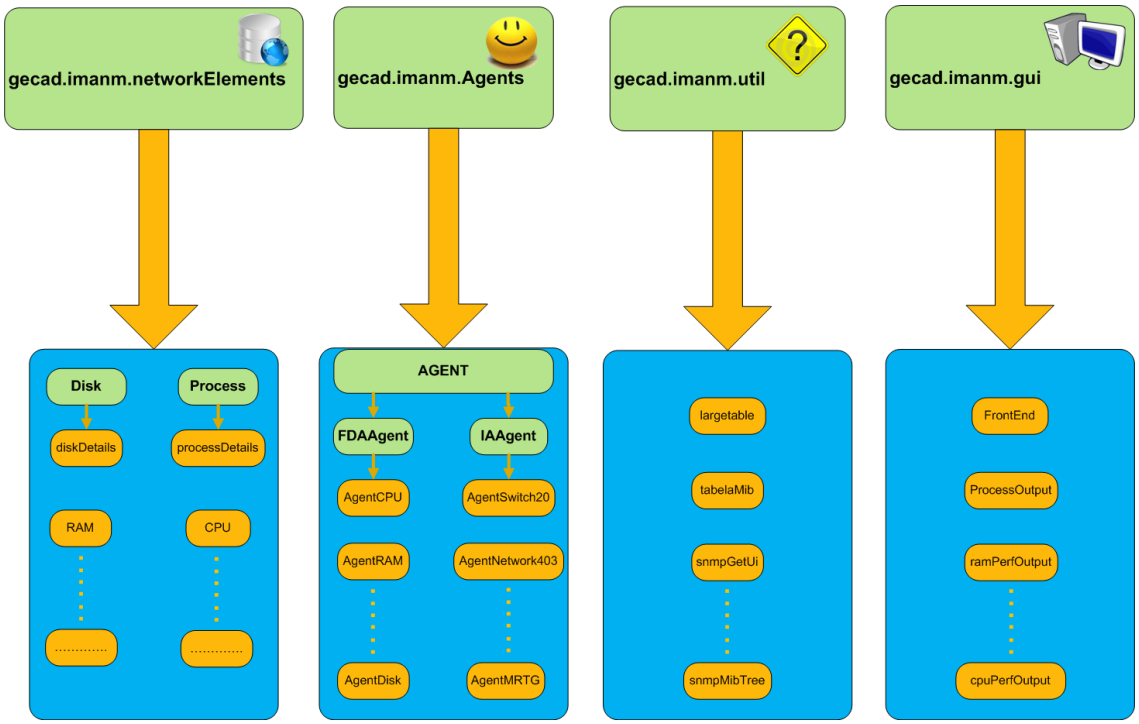


Figura 67 – Pacotes de Classes

A figura 67 ilustra a forma como estão organizados os pacotes de classes referidos. No pacote dos Agentes encontramos naturalmente todos os Agentes móveis, que por sua vez se servem do auxílio das classes presentes no pacote *gecad.imanm.networkElements*. As classes a verde representam classes-mãe, de onde partem setas a laranja para as que herdam os seus atributos. De seguida vamos ver um exemplo que explica este facto no diagrama anterior, no pacote *networkElements*: quando um Agente chega a um disco de um determinado elemento de rede, este disco pode ter várias partições. A classe mãe a verde, de nome *Disk*, simboliza todos os dados importantes que um "AgentDisk" necessita de guardar aquando da execução de tarefas de gestão. No entanto, por cada disco, temos de guardar dados da partição x, y ou z. Daí a existência de uma classe *diskDetails*, onde são armazenados todos os dados por partição e que herda certos atributos da classe-mãe.

No pacote *gecad.imanm.util* estão algumas das classes que auxiliaram e/ou auxiliam a construção da aplicação. A classe *largetable*, por exemplo, consegue apresentar uma janela ao utilizador para este fazer *pooling* por SNMP a um elemento de rede, para uma determinada tabela com um OID

específico. Já no pacote *gecad.imanm.gui* encontramos a classe *FrontEnd*, da janela principal, assim como outras, responsáveis por apresentar outro tipo de dados e interacção ao utilizador. Um exemplo é a classe *ProcessOutput*, que apresenta uma tabela com todos os dados de todos os processos em execução num determinado elemento de rede.

Para o funcionamento correcto do protótipo desenvolvido, são necessárias duas premissas iniciais: o Agente facilitador tem de estar em execução na estação gestora e os Agentes *Daemon* têm, por sua vez, de estar em execução nos elementos de rede que estejam configurados como elementos de rede na aplicação de gestão. Isto pode ser observado na próxima figura, figura 68, que descreve o funcionamento geral do sistema. Após estas duas condições essenciais estarem cumpridas, o protótipo pode ser lançado, bem como o Agente de diagnóstico, criado imediatamente e de forma automática após o lançamento da Aplicação. Como em qualquer criação de um Agente, o de diagnóstico não é excepção e tem de se registar junto do facilitador, indicando-lhe o nome e quais os *solvables* que consegue resolver.

Garantindo que o registo anterior foi bem sucedido acedemos à interface da aplicação. Nesta, o objectivo inicial passa pela criação dos Agentes que o utilizador acha necessários para a execução das tarefas de gestão pretendidas. Para tal, e como vemos no diagrama da figura 68, na Interface são escolhidos os tipos de Agentes, quais os segmentos alvo e eventualmente se irão ser efectuados testes, após o que se segue a criação dos Agentes móveis.

A parte final de todo este processo culmina numa tomada de decisão, onde um ou mais Agentes de detecção/correcção de falhas podem ter chegado à conclusão que precisam da ajuda de um Agente de intervenção. Independentemente da existência, ou não, dessa necessidade, em ambos os casos existe um retorno de resultados à estação gestora. A diferença é que no caso em que vão ser solicitados Agentes de intervenção, antes do retorno dos resultados, é efectuado um pedido de auxílio que é remetido para um Agente existente. Caso este não esteja presente na comunidade, é efectuado um outro pedido para tentar a criação automatizada de um Agente capaz de resolver o *solvable* em causa. Finalmente, os resultados podem ser apresentados na Interface, guardados em ficheiros de *log*, ou ambos, dependendo das situações, com a continuação da execução.

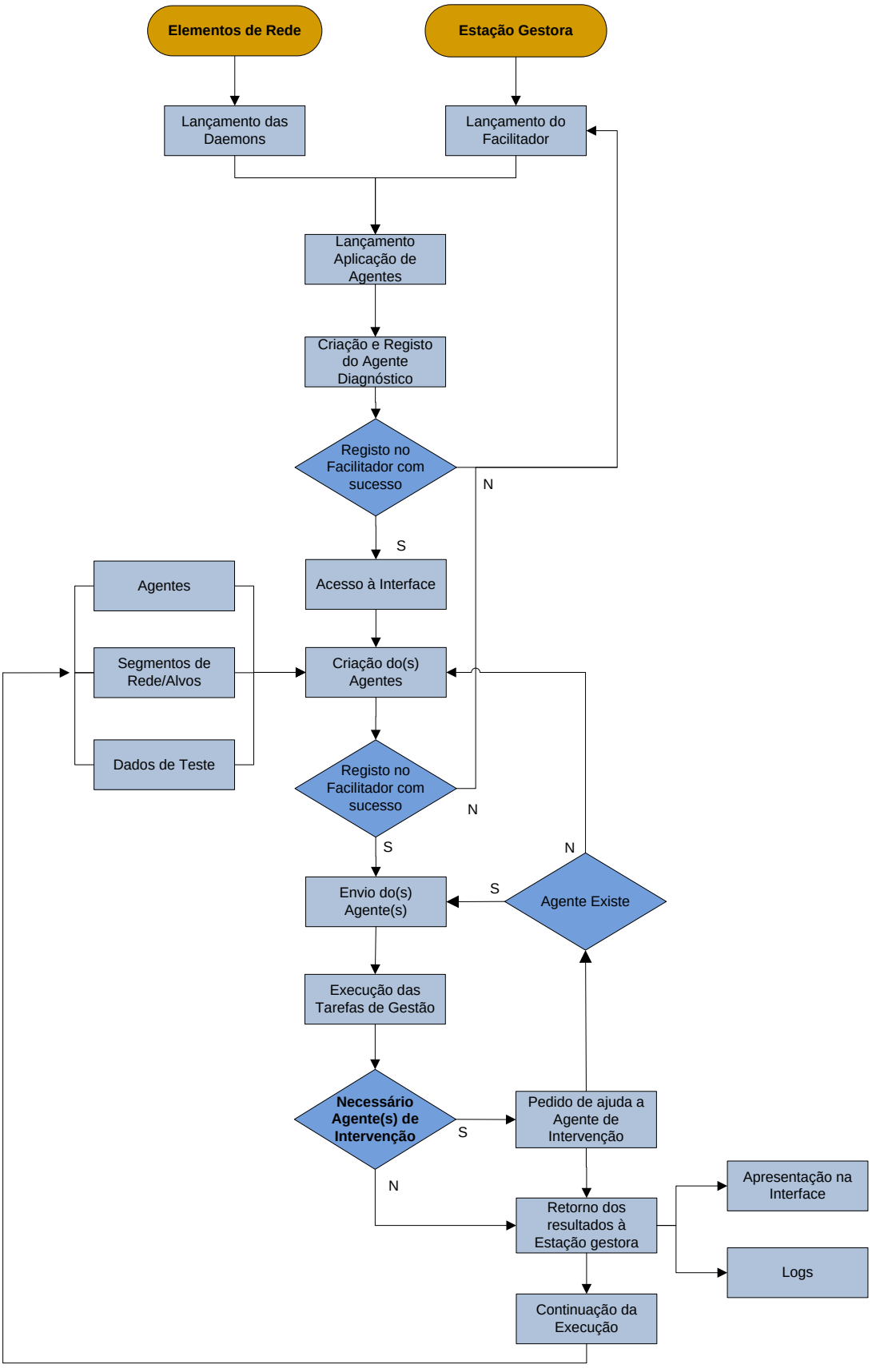


Figura 68 – Funcionamento Geral do Sistema

Iremos abordar de seguida o funcionamento dos Agentes móveis, colocando o enfoque desta análise no seu ciclo de vida e que alterações surgem durante o seu curso. Para acompanhar e perceber melhor todo este processo, apresentamos a figura 69.

Um pedido para a criação de um Agente parte da interface. Seguindo-se uma de duas situações possíveis: o utilizador manualmente cria o Agente na interface, ou é efectuado um pedido para a criação do mesmo de forma automatizada, no caso de ser preciso resolver um *solvable* urgente, por um Agente que ainda não exista na comunidade. O novo Agente tem de ser criado pelo Agente de diagnóstico que está por sua vez ligado à Interface, como explicado na secção 5.3.4.

O primeiro teste passa por verificar a conectividade com o Agente facilitador. Em caso de não ser possível a conexão, deve-se verificar a execução do facilitador e tentar de novo. Se existir conectividade, o Agente regista-se, fornecendo o seu nome, localização sob a forma de IP, porta de conexão e os *solvables* que consegue resolver.

A partir desse momento, estando registado na comunidade de Agentes, aguarda por um pedido que se enquadre nas suas funções, isto é, um pedido para resolver um *solvable* que seja do seu domínio. Quando é efectuado um pedido para a comunidade, como vemos na figura 69, o facilitador consulta os dados que lhe foram fornecidos por todos os Agentes e identifica este Agente como sendo capaz de resolver o *solvable* em causa. Após essa identificação ser efectuada, o Agente migra para o primeiro elemento de rede. Quando chega ao primeiro elemento de rede executa duas funções em simultâneo: as tarefas de gestão e remover os *solvables* que é capaz de resolver do seu estado. Este comportamento deve-se ao facto de cobrir a possibilidade de um outro Agente enviar um pedido para a comunidade com o mesmo *solvable*. O Agente capaz de responder ao pedido já iniciou a migração para um determinado elemento de rede com vista à execução de tarefas de gestão, pelo que está indisponível para executar outro pedido.

Se o Agente tiver outro destino (elemento de rede) no seu itinerário, migra para o próximo elemento de rede e executa as tarefas de gestão que lhe estão atribuídas para o mesmo, até que dá o salto final para a estação gestora, onde apresenta os resultados e repõe os *solvables* que continha, ficando nesse momento disponível para responder a futuros pedidos.

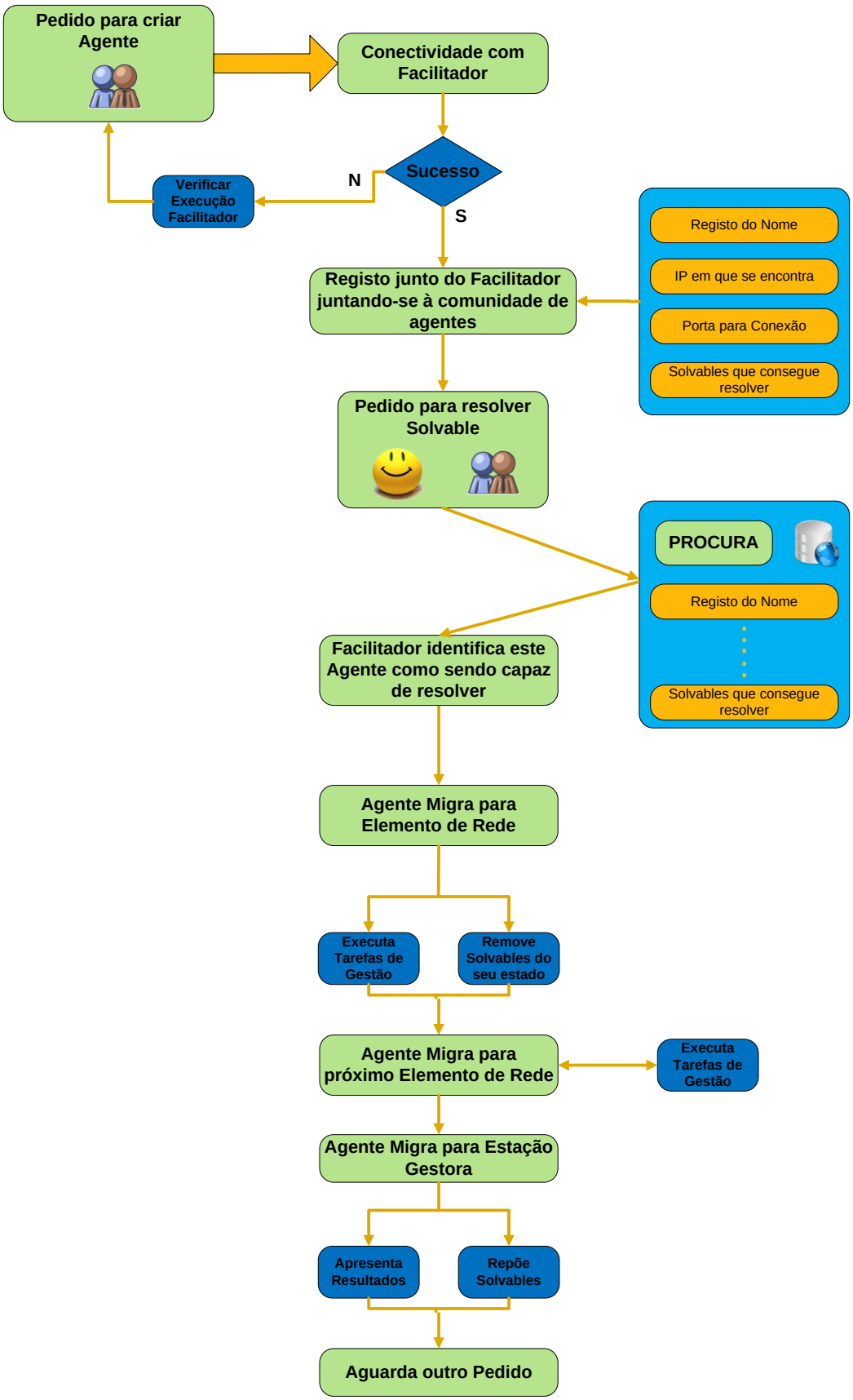


Figura 69 – Funcionamento dos Agentes móveis

Existem algumas noções relativamente aos Agentes móveis que ainda não foram detalhadas: a forma encontrada para se moverem, como interagem com as *Daemons* que se encontram à sua espera nos diversos elementos de rede e como são capazes de ler ou escrever valores nas MIBs. Nos próximos parágrafos iremos abordar estas questões.

Um dos requisitos essenciais para a implementação do sistema aqui proposto foi permitir que os Agentes tivessem a capacidade de comunicar via SNMP. Tal capacidade permite que os Agentes consigam ler os valores das MIB's para executar posteriormente as acções indicadas. Nas MIB's encontram-se vários tipos de dados: *Int*, *String*, *OctectString*, *TimeTicks*, *Gauge*, *Counter32*, entre outros. A API usada para obter estes fins sugere na documentação o simples uso de uma sucessão de comandos básicos para obter os dados desejados [34, 82]:

…….

Criar e Iniciar **target** do Tipo **SnmpTarget**

//Definir o alvo

target.setTargetHost("localhost")

//Definir a OID

target.setObejctId (".1.3.6.1.2.1.1.0")

String Result ← target.SnmpGet() **OU String Result** ← target.SnmpGet().toString

…….

A verdade é que tanto no uso desta API como noutras, esta abordagem falha em várias situações, nomeadamente quando queremos obter dados que não sejam do tipo *Int*. Na maioria dos casos, apenas "lixo" é retornado, independentemente da versão de SNMP usada, ou de outros factores que possam ser alterados. Existiam duas opções: a primeira seria programar num nível mais baixo, desperdiçando as capacidades da API de alto nível; a segunda passava pela criação de um algoritmo que garantisse a integridade do *GET* ou *GETNEXT*, de qualquer tipo de dados que queiramos obter, utilizando a API de alto nível.

Partindo do pressuposto que as MIB's foram carregadas devidamente no código da aplicação e estão prontas para uso é possível interagir com estas. A próxima figura, figura 70, ilustra o algoritmo que foi criado para efectuar uma consulta a qualquer tipo de dados de uma determinada OID. Podemos ver no início a declaração de dois *arrays* de bytes, dois inteiros, duas *strings*, sendo a *string* de nome "multi" a que é retornada no fim. Por último, é criada uma variável *target*, do tipo *SnmpTarget*, responsável por albergar o objecto de gestão, quer na forma de elemento de rede, quer na forma de OID a consultar.

```
... ... ... ... ... ... ... ... ... .
VARIÁVEIS
i,multi_aux: int
bytes, bytes2: byte[]
result, multi: String
target: Snmp Target
INICIO
Elemento de rede target ← "localhost"
OID target ← ".1.3.6.1.2.1.1.0"
bytes ← Operação GET sobre OID codificando o retorno em bytes
result ← String (bytes)
Tentar
        multi_aux ← Integer.parseInt(target.snmpGet().toString())
        multi ← multiaux+""
   Excepção NumberFormatException
      Tentar
      result ← result sem espaços em branco
      bytes2 ← new byte[result.length() / 2]
        Para i← 0 até i← < bytes2.length fazer
        bytes2[i] ← (byte) Integer.parseInt(result.substring(2 * i, 2 * i + 2), 16)
        fim
        multi ← new String(bytes2);
        multi ← multi.substring(0, multi.length() - 1)
           SE multi.isEmpty() == true
           multi ← result
           Fim SE
      Excepção NumberFormatException2
         String charray ← ""
         Para i← 0 até i← < bytes.length fazer
         charray += ((char) bytes[i])
         fim
         multi ← charray
      Fim Excepção
   Fim Excepção
retorna multi
Fim
```

Figura 70 – Algoritmo para ler dados da MIB

A abordagem inicial foi a de ler os dados da MIB num formato em bytes e criar uma *string*, baseada nesse mesmo conjunto de bytes. O algoritmo contém três formas ou tentativas para ler um determinado valor de uma OID. Começa por admitir que está a tentar obter algo relacionado com valores numéricos que são passíveis de ser extraídos de forma directa, colocando o valor numa *string* final. Caso falhe, existe uma excepção que é lançada, responsável por tratar todos os restantes tipos de dados que não se enquadrem na primeira abordagem.

Na segunda tentativa, ainda relacionada com valores numéricos, são removidos os espaços da *string* formada através do *array* de bytes, *string* essa que se encontra no formato hexadecimal. Quando tal acontece, a "fórmula" para conseguir obter um resultado válido passa por criar um novo *array* de bytes baseado na *string* existente (neste caso "result"). Mas esse novo *array* deve ser uma representação fiel da *string* hexadecimal dada, sendo assim, o tamanho do novo array é igual a "result.lenght()/2". Este *array* é preenchido byte a byte para que seja obtido um conjunto de bytes que representem a *string* de forma fiável. Logo de seguida resta-nos criar uma *string* final a partir desse novo *array* de bytes e utilizá-la da forma que entendermos pois estaremos perante uma representação normal e perceptível aos nossos olhos daquilo que consta no valor da OID.

Por último, se nenhuma das opções anteriores funcionar, estamos perante um caso em que apenas podemos, byte a byte, construir a *string* final com um *charray*, em que por cada byte iremos ler um *char*, até que todo o *array* de bytes inicial esteja percorrido. No fim, teremos também uma representação fiel do valor que queremos da MIB, no formato de uma *string*. Com este algoritmo, é possível ler qualquer dado de qualquer MIB utilizando as operações de leitura comuns.

A próxima figura, figura 71, ilustra com detalhe os últimos três componentes que faltam apresentar nesta secção: a migração de um Agente, em conjunto com a *Daemon* que o espera no elemento de rede destino e a classe auxiliar que serve para guardar o seu estado antes de se mover. Tudo se inicia quando existe um pedido para resolver um *solvable* lançado na comunidade e é encontrado o Agente que o pode fazer. Esse Agente possui um itinerário, pois no mínimo tem de se deslocar para um elemento de rede, executar tarefas de gestão, e dar um salto final para a estação gestora. Após a identificação feita pelo facilitador de que este Agente consegue resolver o pedido inicial, é iniciado o processo de migração.

Mediante o tipo de Agente que procura mover-se, irá necessitar de ajuda, servindo-se das classes auxiliares para guardar os atributos e o seu estado. Por exemplo, se este for um AgentRAM, irá procurar apoio de uma classe desse tipo. É então criado e instanciado um objecto da classe auxiliar ao qual se segue a verificação da conectividade com a *Daemon* do elemento de rede para onde o Agente se deseja mover. Em caso de não existir sucesso, o Agente deve verificar o seu itinerário, notificar a aplicação acerca do problema de comunicação e passar ao próximo elemento de rede. Se o teste for bem sucedido é feita a serialização do estado e atributos do Agente em causa. É de realçar que as *Daemons*, como indicado no diagrama no lado direito, estão sempre em execução, à espera de um pedido de conexão.

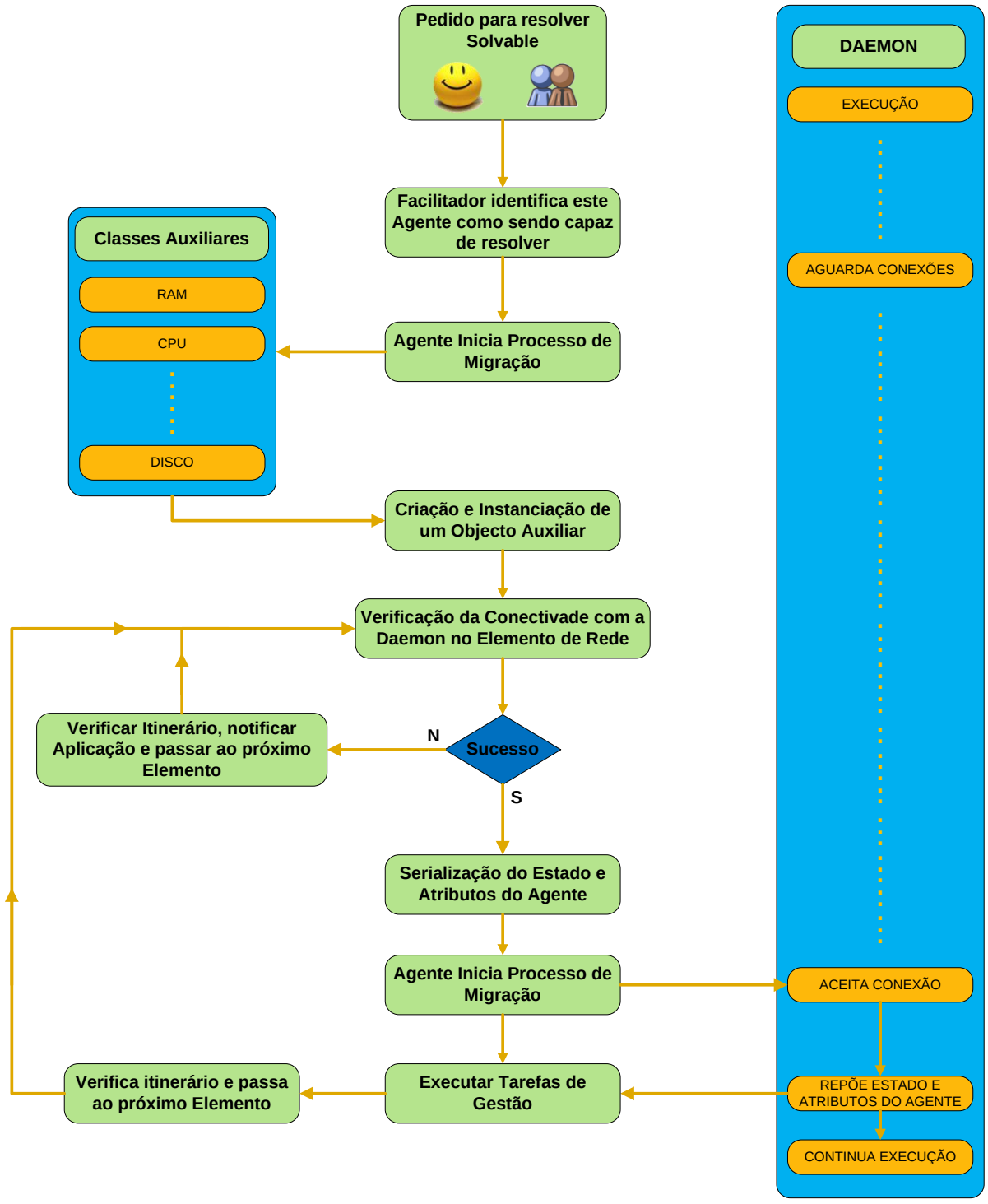


Figura 71 – Funcionamento da Migração, Daemon e Classes Auxiliares no Sistema

O Agente inicia então o processo de migração, a conexão com a *Daemon* é estabelecida, os dados são transmitidos e a *Daemon* é responsável por repor o estado e os atributos do Agente, continuando posteriormente a sua execução e aguardando por novas conexões. São efectuadas localmente as tarefas de gestão programadas e aquando do seu término, o Agente verifica o

itinerário, passando ao próximo elemento de rede, que pode bem ser a estação gestora, ou seja, a paragem final, enquanto não existir um novo pedido.

5.3.6 Exemplos de Utilização dos Agentes

Nesta secção são apresentados exemplos de Agentes móveis desenvolvidos.

Relativamente a recursos como CPU, RAM e espaço em disco, existem vários serviços do Nagios responsáveis por monitorizar cada um deles junto dos vários elementos de rede. Sempre que uma determinada partição num servidor está perto de ficar sem espaço, é marcado um alerta no serviço (*warning* a cor amarela), caso o espaço se encontre perto do limite mais alto em percentagem (imposto pelo administrador) o alerta passa de *warning* para *critical*, assinalado a vermelho. O *polling* é efectuado, o administrador recebe um email, e no entanto, pode ser tarde demais, uma vez que não estaríamos perante o primeiro computador que parasse perante falta de espaço.

AgentDisk

Actualmente só para consultas de espaço em disco em servidores vitais, existem trinta e oito serviços (um serviço pode analisar várias partições) que verificam esta situação periodicamente. Se algum dos Agentes se deparar com falta de espaço em disco, mediante um conjunto de regras, actua imediatamente, ou seja, liberta o espaço que for possível e só depois deixa o elemento de rede para dar o próximo salto. Se esta situação ocorrer, por exemplo, no servidor ZEUS1, que contém um espaço elevado para partilha de documentos, o Agente pode eliminá-los. Os utilizadores estão cientes (através de um anúncio presente nas pastas) de que aquele espaço é temporário. Isto tem de ser feito no entanto por data, pois não queremos que o nosso Agente elimine os ficheiros mais recentes. Se no próximo salto o Agente se encontra no servidor SERVUSERS, que é um servidor aplicacional, com pastas dedicadas aos utilizadores, mas apenas com uma partição, a de sistema, deve tomar outras medidas, nomeadamente eliminar ficheiros temporários.

A abordagem seguida para o desenvolvimento do **AgentDisk** foi pensada segundo o que foi descrito no parágrafo anterior. No GECAD, a grande maioria dos Sistemas Operativos são o Windows 7 e o Windows Server 2003 e 2008, relativamente a elementos de rede que precisam de monitorização. Como tal, foram definidos quatro directórios alvo que contêm ficheiros temporários ou desnecessários:

- *C: \Windows\Temp*;
- *C:\Users\%Username%\Appdata\Local\Microsoft\Windows\Temporary Internet Files*;
- *C:\Users\%Username%\Appdata\Local\Microsoft\Office\Recent*;
- *C:\Users\%Username%\Appdata\Local\Temp*;

Foram também identificadas diversas extensões de ficheiros que são desnecessárias, mas que não fazem parte do domínio de execução do "AgentDisk", pois esse tipo de acções requer um cuidado extra de análise aos ficheiros e torna o processo muito moroso, pois toda a partição tem de ser examinada.

Para efeitos de testes foi incorporado um algoritmo de limpeza com controlo de erros para eliminar todos os ficheiros e subdirectórios existentes nas pastas referidas. Antes de executar o algoritmo, é necessário saber se o elemento de rede necessita de uma limpeza, sendo analisada a partição, tamanho, tamanho utilizado e respectiva percentagem de uso, de forma a podermos definir valores de alerta. No exemplo apresentado de seguida, figura 72, um "AgentDisk" é criado, trabalha num determinado elemento de rede e regressa à estação gestora, fornecendo o nome do elemento, a descrição do sistema, há quanto tempo se encontra ligado e o tempo exacto em que as tarefas de gestão se iniciaram.

Figura 72 – Exemplo do trabalho de um AgentDisk

O exemplo apresentado é de um computador que não necessita de uma limpeza de disco, pois a percentagem de utilização situa-se nos 27%. No entanto, o valor para alerta foi modificado, forçando a execução das tarefas. O "AgentDisk" começa por carregar as MIBs de que necessita para fazer uma primeira análise e guardar dados. São executados vários cálculos, porque o espaço em disco não pode ser lido da MIB directamente, uma vez que apenas temos disponíveis as unidades de alocação para um determinado tipo de *storage* e o espaço utilizado sob a forma de unidades de alocação. Após obter os dados iniciais, o Agente calcula a percentagem de utilização da partição, efectuando de seguida uma leitura sobre a mesma, definindo se deve intervir e de que forma.

O passo seguinte passa por definir os directórios que queremos atacar e correr o algoritmo de limpeza para cada um deles. O Agente aguarda um tempo pré definido para que a limpeza seja efectuada e os dados sejam actualizados na MIB. Depois desse tempo de espera, volta a consultar e a calcular o espaço em disco que se encontra alocado. Com os dois valores na sua posse (valor utilizado na análise e valor após limpeza) consegue efectuar um último cálculo, referente à quantidade de espaço que conseguiu libertar, neste caso cerca de 3GB ou 3000MB.

AgentRAM

O mesmo pode ser feito em relação a recursos como o CPU e a RAM. Mover pelo menos o processamento para os elementos de rede no caso do CPU e tomar algumas medidas no caso da RAM. No GECAD muitas vezes os servidores e Workstations estão ligados dias a fio, conduzindo a que a RAM comece a fragmentar, aumentando e não sendo liberta à medida que alguns processos deixam de subsistir. Este inconveniente torna-se mais evidente quando a maioria dos utilizadores faz uso de simulações com software próprio ou possuem soluções de virtualização, em que ambas as tarefas necessitam da presença de uma quantidade de RAM elevada. Como tal, podemos através do uso do AgentRAM, em conjugação com comandos existentes nos próprios Sistemas Operativos, libertar RAM facilmente, passando de um *polling* constante sem resultados práticos a não ser os alertas, para a existência de uma reacção imediata, pronta e que pode ser decisiva.

Para tal, foi ainda tido em conta a forma como os Sistemas Operativos Microsoft efectuam a gestão de memória, uma vez que são amplamente utilizados onde o Agente AgentRAM irá intervir. A funcionalidade de nome *Superfetch* utiliza uma quantidade substancial de memória RAM, que coloca em cache. No entanto, o Sistema Operativo apenas utiliza memória para colocar em cache quando não existe qualquer outro uso para esta. Se uma aplicação necessitar de uma quantidade de RAM elevada, o Windows liberta a memória que está em cache, para tal. Mas quais as razões porque isto é assim e porque é este facto relevante? A resposta é simples: toda a memória que não está a ser usada para nenhum fim é considerado desperdício. Dessa forma, enchendo a memória

com dados do disco que se podem vir a tornar úteis é muito melhor do que não utilizar a memória de todo. Caso esses dados sejam precisos, já estão carregados, tornando as operações muito mais rápidas. A figura 73 é um exemplo do separador *Performance* do *Windows Task Manager*, em que foi realçado o valor *free [85-87]*.

Esta breve explicação dada sobre a gestão da memória assim como a figura 73, prendem-se com o facto de o AgentRAM trabalhar sobre o valor *free* como iremos ver de seguida. Quanto mais baixo for o valor que está presente no campo assinalado a vermelho, melhor para o Sistema Operativo. No exemplo que é dado até é demasiado alto, uma vez que 847 MB é um valor considerável, mesmo para quem tem 6GB de memória. Resumindo, os valores que realmente interessam são os que se encontram nos campos *Available* e *Cached*, sendo importante não descurar o *free*, na tentativa de o manter o mais baixo possível.

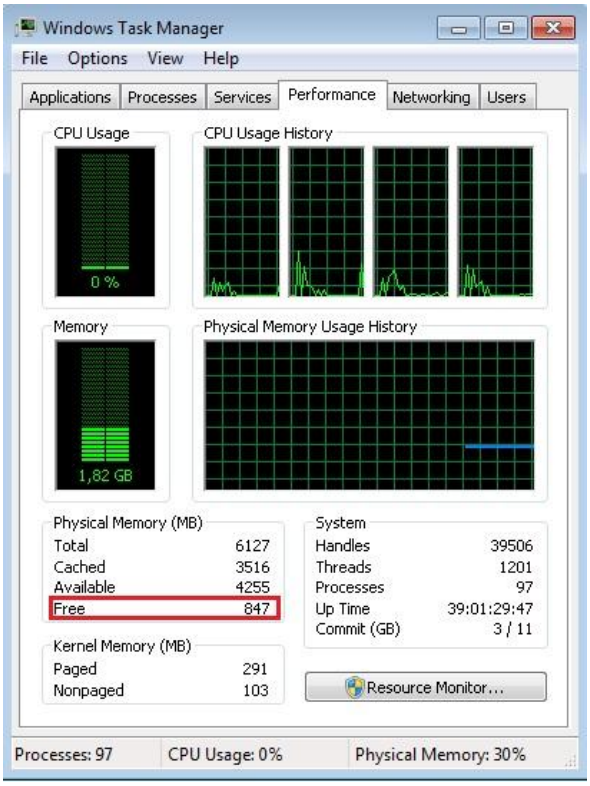


Figura 73 – Performance no Windows Task Manager

Após a migração, a primeira tarefa que um AgentRAM executa, passa por efectuar cálculos similares aos do "AgentDisk" para obter dados sobre a quantidade de RAM existente no elemento de rede, quanto está em uso e a percentagem de utilização, para além de informação sobre o nome da máquina, descrição do sistema, *uptime* e tempo exacto em que se iniciaram as tarefas de gestão.

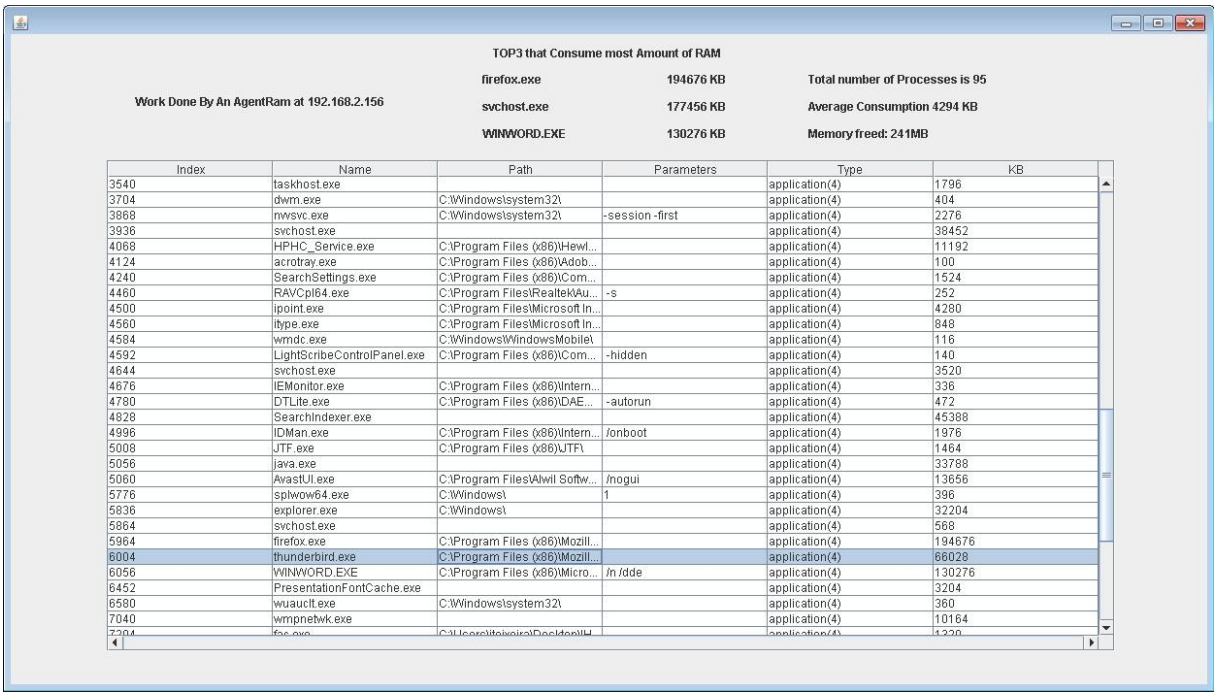
A segunda grande tarefa passa por automaticamente chamar um programa externo, de nome "MemBoost" [88]. O "MemBoost" é lançado e instantaneamente, com o auxílio da classe "Robot",

em Java, é enviado um comando "ALT-F4", de forma a colocar o programa minimizado na *tray*, sem incomodar possíveis utilizadores no momento da execução das tarefas do Agente.

Depois o AgentRAM aguarda sensivelmente um minuto e meio, deixando o "MemBoost" executar as optimizações que entender na memória, de seguida o próprio Agente encarrega-se de terminar o processo correspondente ao programa, enviando um pedido ao Sistema Operativo. Podemos ver de seguida as linhas de código que o permitem:

```
ProcessBuilder builder2 = new ProcessBuilder(new String[] { "cmd.exe", "/C", "taskkill /F /IM memBoost.exe"});
    try {
       Process newProcess2 = builder2.start();
    } catch (IOException ex) {
       Logger.getLogger(AgentTest.class.getName()).log(Level.SEVERE, null, ex);
    }
```

Após o término da execução do "MemBoost", o AgentRAM debruça-se sobre o valor *free* referido anteriormente. Para evitar que este valor se mantenha alto, o Agente executa um comando fornecido pela Microsoft que visa tratar esse valor e toda a memória que está presa em processos que se encontram num estado IDLE, desperdiçando-a: "*rundll32.exe advapi32.dll,ProcessIdleTasks*". Em último lugar, se foi necessária intervenção, é porque a taxa de utilização de memória RAM se encontrava elevada. Sendo assim, o Agente anexa um conjunto de dados adicional, reúne-os com o seu estado, para apresentar aquando da sua chegada à estação gestora. Podemos ver um exemplo na próxima figura, figura 74.



Index	Name	Path	Parameters	Type	KB
3540	taskhost.exe			application(4)	1796
3704	dwm.exe	C:\Windows\system32\		application(4)	404
3868	nwsvc.exe	C:\Windows\system32\	-session -first	application(4)	2276
3936	svchost.exe			application(4)	38452
4068	HPHC_Service.exe	C:\Program Files (x86)\Hewl...		application(4)	11192
4124	acrotray.exe	C:\Program Files (x86)\Adob...		application(4)	100
4240	SearchSettings.exe	C:\Program Files (x86)\Com...		application(4)	1524
4460	RAVCpl64.exe	C:\Program Files\Realtek\Au...	-s	application(4)	252
4500	ipoint.exe	C:\Program Files\Microsoft In...		application(4)	4280
4560	itype.exe	C:\Program Files\Microsoft In...		application(4)	848
4584	wmdc.exe	C:\Windows\WindowsMobile\		application(4)	116
4592	LightScribeControlPanel.exe	C:\Program Files (x86)\Com...	-hidden	application(4)	140
4644	svchost.exe			application(4)	3520
4676	IEMonitor.exe	C:\Program Files (x86)\Intern...		application(4)	336
4780	DTLite.exe	C:\Program Files (x86)\DAE...	-autorun	application(4)	472
4828	SearchIndexer.exe			application(4)	45388
4996	IDMan.exe	C:\Program Files (x86)\Intern...	/onboot	application(4)	1976
5008	JTF.exe	C:\Program Files (x86)\JTF\		application(4)	1464
5056	java.exe			application(4)	33788
5060	AvastUI.exe	C:\Program Files\Alwil Softw...	/nogui	application(4)	13656
5776	splwow64.exe	C:\Windows\	1	application(4)	396
5836	explorer.exe	C:\Windows\		application(4)	32204
5864	svchost.exe			application(4)	568
5964	firefox.exe	C:\Program Files (x86)\Mozill...		application(4)	194676
6004	thunderbird.exe	C:\Program Files (x86)\Mozill...		application(4)	66028
6056	WINWORD.EXE	C:\Program Files (x86)\Micro...	/n /dde	application(4)	130276
6452	PresentationFontCache.exe			application(4)	3204
6580	wuauclt.exe	C:\Windows\system32\		application(4)	360
7040	wmpnetwk.exe			application(4)	10164

Figura 74 – Exemplo do Output de um AgentRAM

Trata-se de uma tabela com todos os processos que ocupam memória RAM no elemento de rede em questão, incluindo por cada um o índice, nome, local onde foi executado, com que parâmetros, o tipo e a quantidade de RAM que está a consumir. É também calculado o TOP3 dos processos que utilizam a maior parte da memória, assim como o número de processos e a média de consumo. Convém realçar o facto de que a curta execução do Agente (um minuto e meio) conduziu à libertação de 241 MB de RAM.

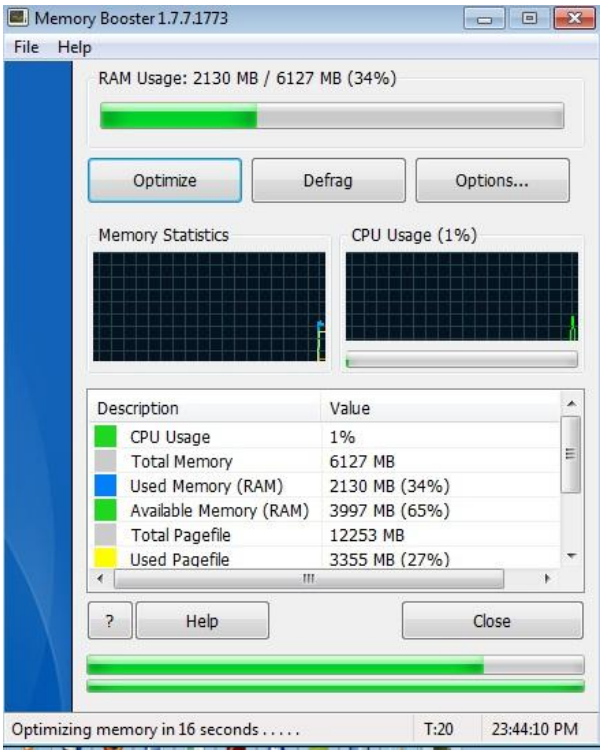


Figura 75 – MemBoost

A figura 75 ilustra o interface do "MemBoost", que executa optimizações de forma automática, em períodos de tempo configurados pelo utilizador, ao mesmo tempo que nos fornece uma visão mais completa do uso da memória.

AgentMRTG

Um AgentMRTG pode ser lançado a partir da interface ou requerido por um dos Agentes de detecção/correcção de falhas, quando estes encontram eventuais problemas. O exemplo que iremos analisar de seguida é um dos mais simples, que tem como objectivo analisar a carga no adaptador de rede de um determinado elemento de rede. O AgentMRTG poderá ser solicitado por outro tipo de Agentes cujas análises se venham a tornar mais complexas, como analisar a carga nas portas 1, 4 e 23 de um *Switch* e procurar por erros de entrada ou saída em pacotes.

A próxima figura, figura 76, é um exemplo do output que um AgentMRTG pode trazer até à estação gestora. Tal como o nome indica, com o MRTG (*Multi Router Traffic Grapher*) pretende-

se obter dados sob a forma de gráficos. Para construir esses gráficos são criados ficheiros no disco do elemento de rede, que após a análise são comprimidos num único ficheiro e transportados com os restantes atributos do Agente.

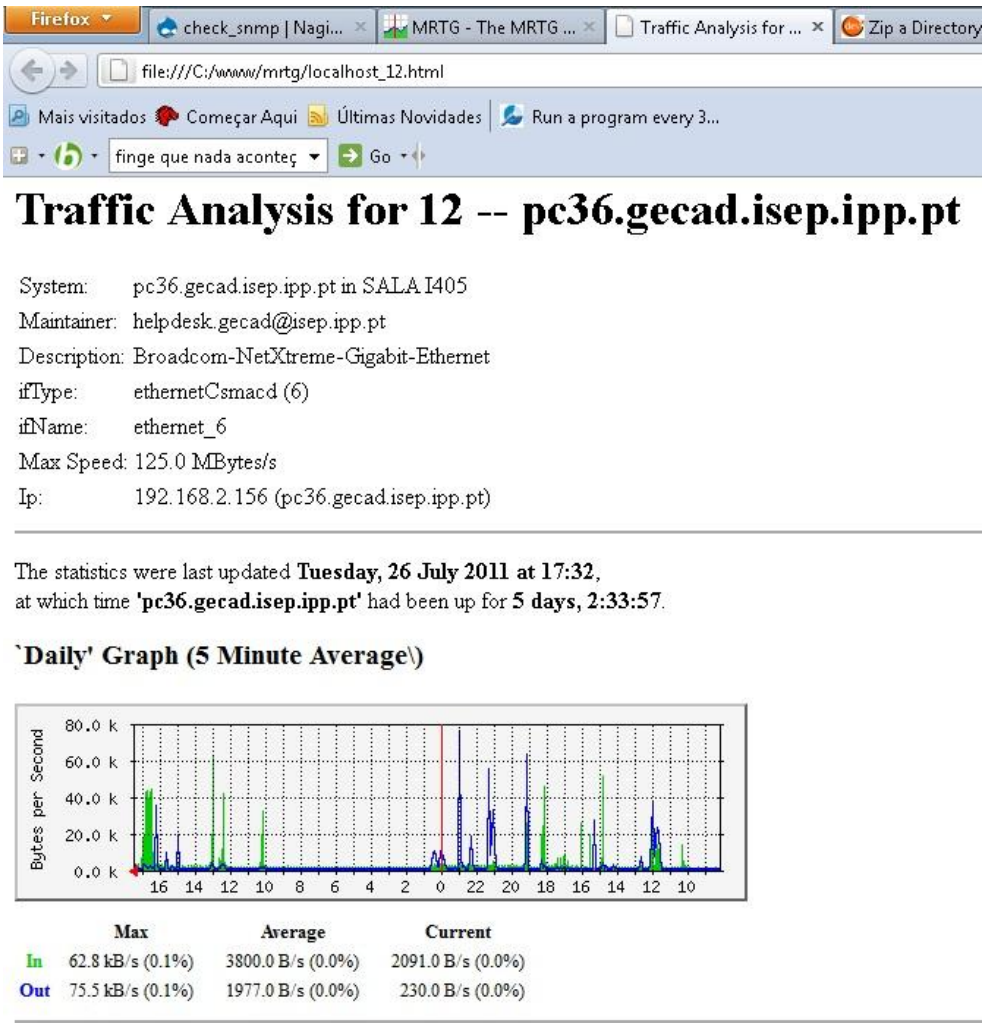


Figura 76 -Output AgentMRTG

É de realçar que este é um gráfico com dados sobre a carga de input e output no adaptador de rede que é construído com médias a cada cinco minutos, fornecendo uma visão geral sobre as últimas vinte e quatro horas, daí o nome "*Daily Graph*".

A figura 76 apresenta ainda os dados num browser, o que não é obrigatório, existindo outras alternativas. Na própria aplicação de Agentes, tal como no Output de uma tabela do AgentRAM, podemos ter uma janela com o gráfico, ou apenas o ficheiro de *log* que é criado em conjunto com as imagens que contêm os gráficos. Quando o Agente chega à estação gestora não trás apenas a imagem ou a página Web em que consegue mostrar o gráfico construído. Com este, vem também um ficheiro de *log* com registos, semelhante ao da figura 77.

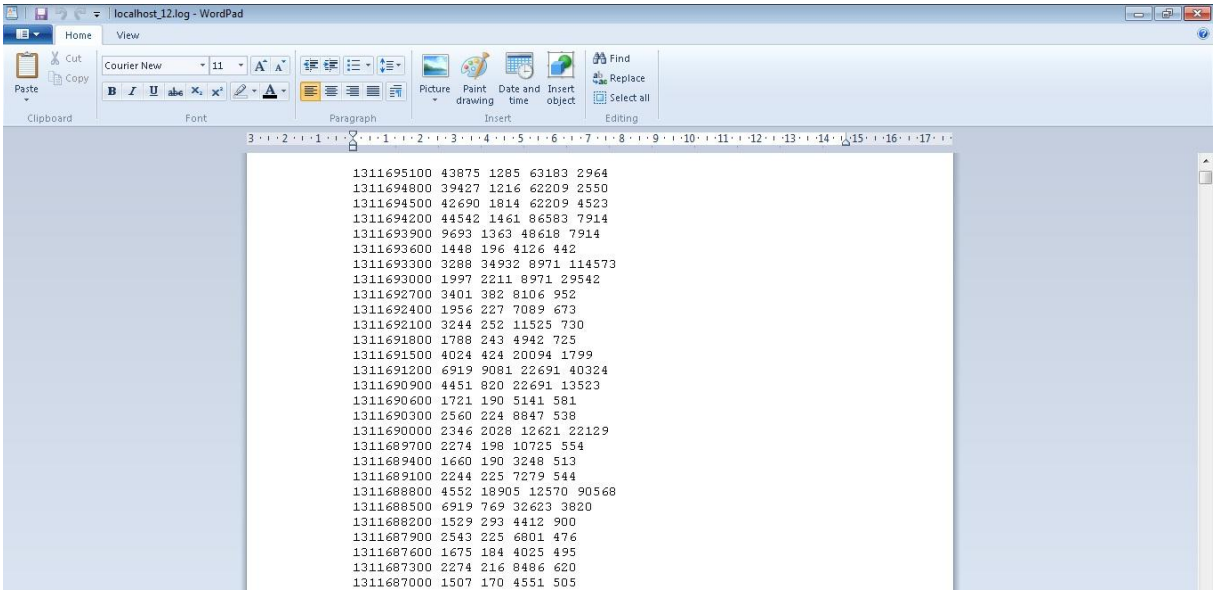


Figura 77 – Exemplo de ficheiro Log AgentMRTG

São apresentados cinco valores por linha, isto é, cinco colunas. A primeira coluna representa um *Unix Timestamp*, um registo de tempo, para o ponto no tempo em que a informação da linha é relevante. A segunda coluna corresponde ao valor médio da taxa de transferência de dados de input em bytes por segundo, enquanto a terceira coluna representa esse valor médio, mas para dados de output. Já a quarta coluna corresponde à maior taxa de transferência de dados *input* em *bytes* por segundo, para o intervalo corrente. Este valor é calculado a partir de todos os *updates* que ocorreram no intervalo corrente. Se esse intervalo for de uma hora e os *updates* acontecerem a cada cinco minutos, o valor equivale à maior taxa de transferência em cinco minutos verificada durante a hora. A quinta e última coluna contêm valores com a mesma lógica de raciocínio da quarta mas para dados de output.

Com as OID's correctas, podemos monitorizar quase tudo, transformando visualmente num gráfico. De seguida, figura 78, encontra-se código exemplo para monitorizar a carga do CPU num *switch*.

```
# Router CPU load %

Target[cpu.1]:1.3.6.1.4.1.9.2.1.58.0&1.3.6.1.4.1.9.2.1.58.0:public@10.10.10
.1
 RouterUptime[cpu.1]: public@10.10.10.1
 MaxBytes[cpu.1]: 100
 Title[cpu.1]: CPU LOAD
 PageTop[cpu.1]: <H1>CPU Load %</H1>
 Unscaled[cpu.1]: ymwd
 ShortLegend[cpu.1]: %
 XSize[cpu.1]: 380
 YSize[cpu.1]: 100
 YLegend[cpu.1]: CPU Utilization
 Legend1[cpu.1]: CPU Utilization in % (Load)
 Legend2[cpu.1]: CPU Utilization in % (Load)
 Legend3[cpu.1]:
 Legend4[cpu.1]:
 LegendI[cpu.1]:
 LegendO[cpu.1]:  Usage
 Options[cpu.1]: gauge
```

Figura 78 – Exemplo Monitorização CPU MRTG

Estes foram os três grandes exemplos práticos que servem para ilustrar as funcionalidades acrescentadas pelos Agentes móveis à aplicação de gestão tradicional. Naturalmente existem os restantes Agentes como o AgentCPU, que processa um output parecido com a tabela do AgentRAM, mas que nos diz a carga de processamento que cada processo em execução está a consumir em unidades "*centisecond*".

Um outro problema frequente nos elementos de rede (com excepção dos activos de rede, em que a frequência é menor) é o aparecimento de falhas com origem na cache de DNS ou na cache da ARP. Desta forma, o AgentNE é o ideal porque periodicamente repara as caches de DNS e ARP, de modo a que as ligações de rede se mantenham "limpas"o mais possível, durante um maior período de tempo. Este tipo de informação de caches é guardado na memória RAM, o que acrescido à duração de forma consecutiva que estes servidores passam conectados à rede, pode originar problemas num espaço de tempo mais curto do que o habitual. Como tal, o AgentNE pode facilmente entrar num elemento de rede com o propósito de limpar todo este tipo de informação e renovar as ligações de rede, sem colocar em causa a conectividade e prevenindo parte das falhas que podemos vir a encontrar e que anteriormente não eram passíveis de ser resolvidas autonomamente.

Por último, o AgentPROC também é muito importante: é possível saber se um determinando conjunto de processos está em execução. Fundamentalmente, há um excesso (não do ponto de vista de gestão mas de recursos consumidos na estação gestora) de alguns serviços que efectuam consultas em elementos de rede. Dando um exemplo prático, um serviço do "Nagios" que seja responsável por verificar se dez processos no servidor GECADCORE estão em funcionamento. Para tal, foram consideradas três opções:

1. Deixamos de verificar os dez processos de uma vez só, dividindo o trabalho por vários serviços do "Nagios", mas aumentando o número de *pollings* efectuados no total ao elemento de rede;
2. Simplesmente passamos de dez para menos verificações, escolhendo as essenciais;
3. A grande questão é que não é fácil determinar quais são os essenciais, sendo todos os serviços existentes no "Nagios", uma vez que antes de serem configurados foram alvo de um estudo, logo podemos entrar em descuidos desnecessários ao remover consultas e monitorizações essenciais. Com o AgentPROC, podemos mover o Agente para os elementos de rede, com as tarefas exigidas previamente determinadas, e no caso de algum processo estar parado, este pode, no próprio elemento de rede, tentar a recuperação do mesmo, numa forma de actuar parecida com a da Microsoft que muitas das vezes é ineficiente.

5.4 Conclusões e Resultados

O sistema proposto existe implementado na forma de protótipo, disponibilizando todas as funcionalidades propostas. No entanto, para se usufruir em qualquer elemento de rede da solução baseada em Agentes móveis é necessário que o mesmo esteja dotado das seguintes condições:

- Agente *Daemon OAA* presente no elemento de rede, pronto para receber os Agentes móveis;
- Configuração efectuada através de um pequeno ficheiro de texto que indica qual a máquina e a porta para conexão ao facilitador.

Não é necessária a instalação da plataforma OAA em todos os elementos de rede, o que tornaria a configuração morosa e pesada. Apenas temos de libertar duas portas para conexão na firewall dos respectivos elementos, para além de garantir que estão aptos a ser geridos por SNMP.

Antes de a implementação estar completa, foram efectuados alguns testes na aplicação de gestão tradicional. Esta indicava-nos que existiam duzentos e cinquenta e cinco serviços atribuídos ao motor de gestão, com perto de oitocentas verificações, a cada quinze minutos, entre as quais as da carga de CPU e RAM, para além de averiguar se determinados processos estavam, ou não, em execução nos elementos de rede. A próxima figura, figura 79, foi gerada pela aplicação tradicional, indicando a quantidade de alarmes por semana e por serviço.

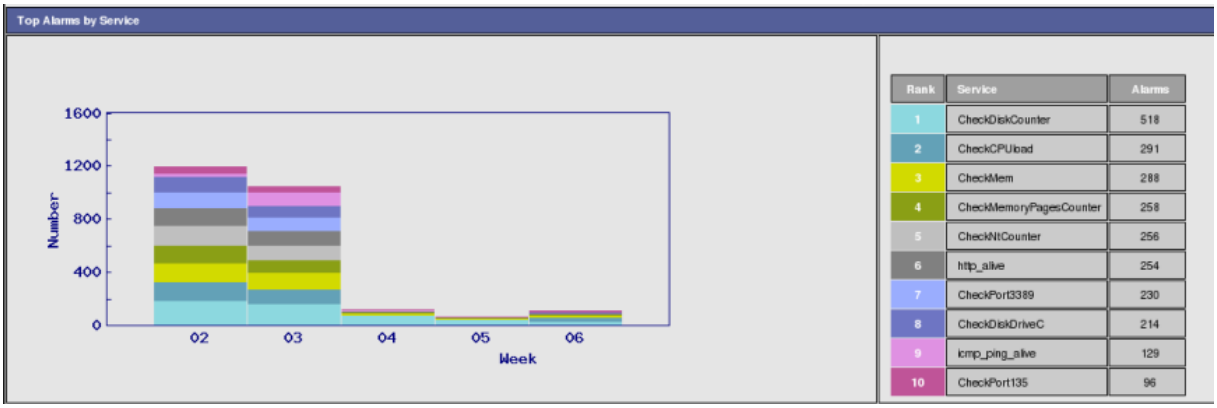


Figura 79 – Alarmes na Aplicação de Gestão Tradicional

Como a instalação da ferramenta estava no início, nota-se uma quantidade demasiado elevada de alarmes em diferentes serviços do motor de gestão nas semanas dois e três. Um cenário mais realista do que realmente se verificava antes de se iniciar a implementação da aplicação com base em Agentes móveis, está ilustrada nos números das semanas quatro, cinco e seis. A tabela ao lado, indica o nome do serviço e o número de alarmes que gerou para o espaço de tempo de todo o gráfico.

Quando o sistema estava implementado iniciou-se uma fase de testes que permitiram encontrar pequenos erros que foram de imediato corrigidos, assim como pequenos ajustes no comportamento dos Agentes. De seguida simulou-se o comportamento do motor de gestão tradicional com a ajuda dos Agentes móveis. Foram retiradas da aplicação tradicional, apenas as verificações de carga de CPU e RAM, processos, e espaço em disco nas partições de sistema. Todas as outras funções de gestão ficaram intactas. Podemos apresentar alguns números ilustrativos das mais-valias obtidas com a adopção da solução proposta: passamos de duzentos e cinquenta e cinco serviços no "Nagios" para cento e noventa e dois serviços, e desses cento e noventa e dois, muitos reflectiam-se em *pollings* pesados para a aplicação, que tinha de verificar num único serviço se dez processos estavam em execução num determinado elemento de rede; de cerca de oitocentas verificações a cada quinze minutos, passamos para quinhentas e setenta e seis. Isto sem tocar em nada ligado a MRTG ou verificações de outras partições no disco para além da de sistema, entre muitos outros. É óbvio que para além da diminuição do *polling*, o processamento foi deslocado para os elementos de rede e conseguimos intervir e reagir a eventos no caso da RAM, espaço em disco nas partições de sistema e no que diz respeito aos processos em execução, tal como demonstrado na secção anterior.

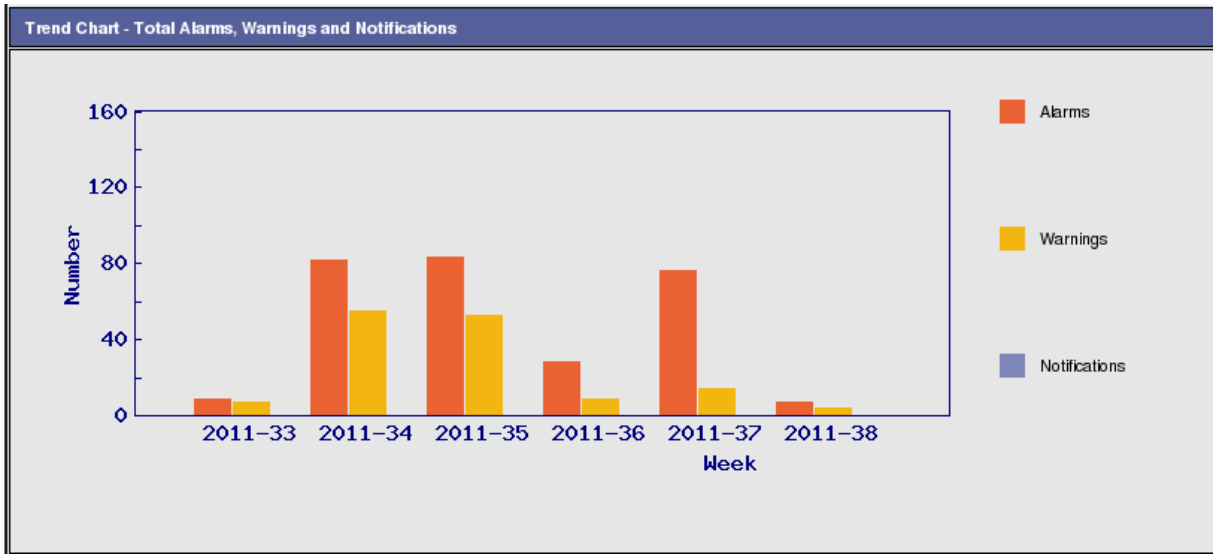
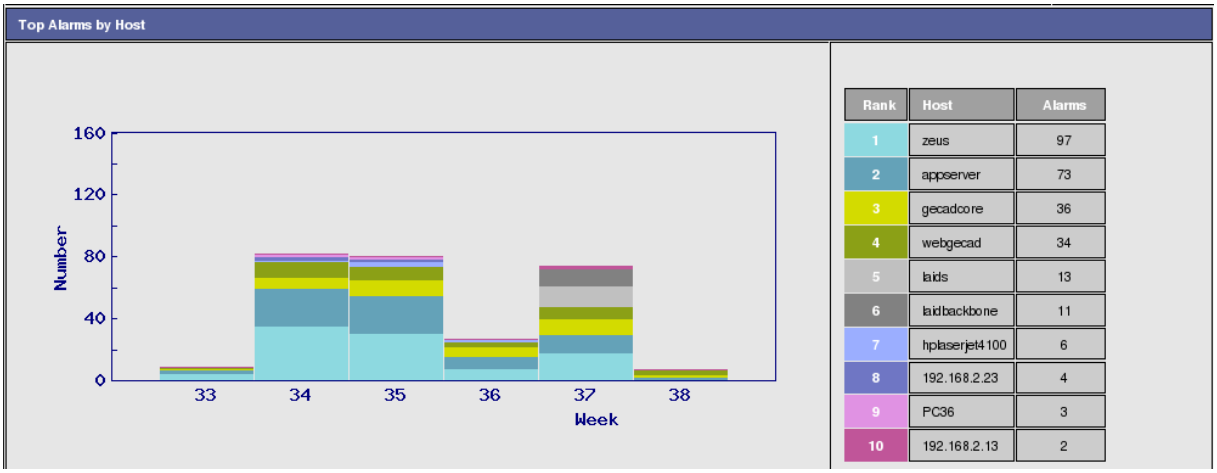


Figura 80 – Alarmes após implementação dos Agentes móveis

A figura anterior apresenta dados sobre os alarmes para as semanas trinta e três há trinta e oito. A trinta e três deve ser ignorada, bem como a trinta e seis, onde existiu uma falha grave do hardware da máquina onde se encontra a estação gestora sob o paradigma tradicional. Antes da simulação, a situação tradicional era a das semanas trinta e quatro e trinta e cinco. Após a implementação dos Agentes móveis vemos uma evidente queda dos *warnings* e alguma diminuição dos alarmes. Na própria semana trinta e oito, apesar de apenas estarem processados os dados de Domingo, Segunda e meio dia de Terça, a tendência é claramente para baixar estes números.

Devemos, apesar de tudo, explicar em que contexto se aplica os dados que acabamos de rever, nomeadamente os alarmes gerados. É natural que retirando alguns serviços ao motor de gestão, este diminua a quantidade de alarmes gerados, principalmente os chamados *warnings*, que são contabilizados quando o serviço não está num estado crítico, mas também, não se encontra saudável. No entanto, esta diminuição só foi possível através da implementação dos Agentes móveis. Existem serviços que mesmo mantidos sob a alçada da aplicação tradicional, por exemplo, as verificações na RAM, deixariam de gerar quantidades enormes de alarmes e avisos, apesar de "entupirem" o servidor com *pollings* e a caixa de correio do administrador. Isto porque os Agentes móveis que encontrem elementos de rede com a RAM elevada tratam de intervir no imediato, conseguindo baixar a mesma em mais de 250MB em apenas um minuto e meio de trabalho. Nalguns casos, essa baixa, no mesmo período de tempo, atingiu médias de 500MB. Com estes resultados, os alarmes e notificações baixariam, porque simplesmente não existiria razão para gerar um alarme. Sendo assim, cessa a necessidade de manter certos serviços a cargo da aplicação tradicional.

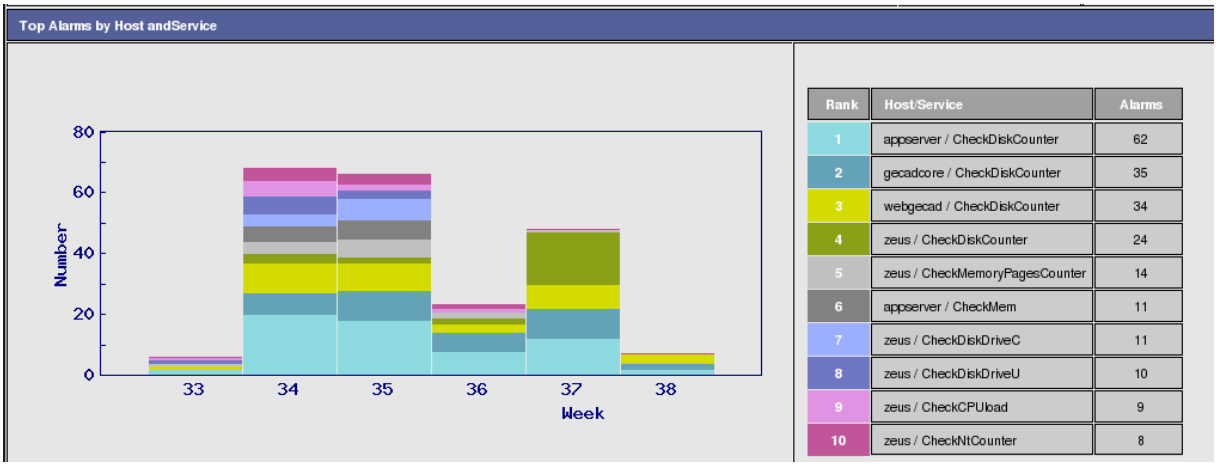


Rank	Host	Alarms
1	zeus	97
2	appserver	73
3	gecadcore	36
4	webgecad	34
5	laids	13
6	laidbackbone	11
7	hplaserjet4100	6
8	192.168.2.23	4
9	PC36	3
10	192.168.2.13	2

Figura 81 – Top dos alarmes por elemento de rede

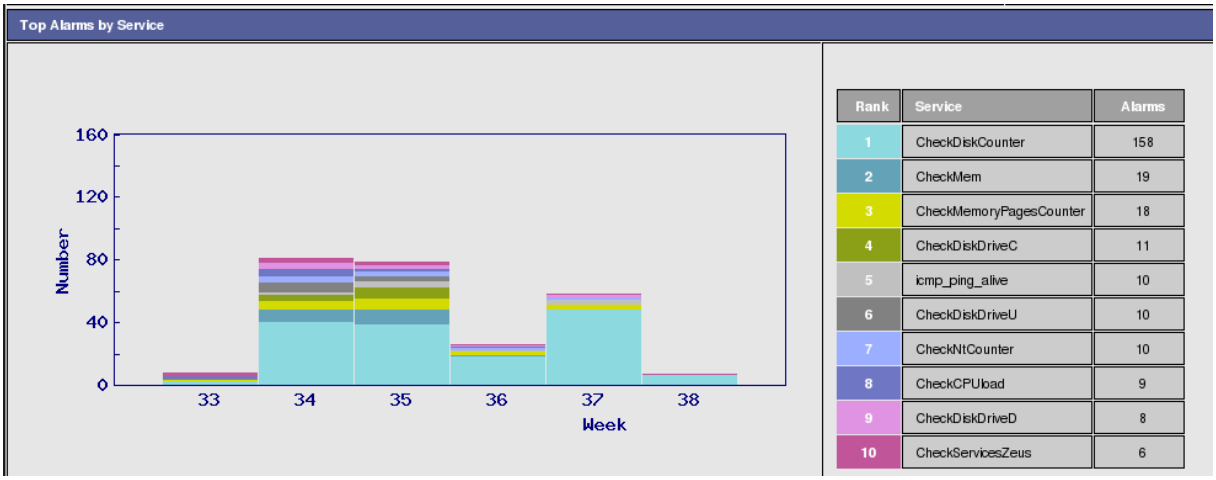
A figura anterior ilustra também um decréscimo em termos de alarmes no total, e por elemento de rede. Na semana trinta e sete, as faixas a cinzento-escuro e cinzento-claro, como indicado na tabela ao lado, pertencem aos elementos de rede "laids" e "laidbackbone". Estes dois elementos de rede estiveram em baixo para manutenção apenas durante seis a sete horas de trabalho. Se não fosse o caso, imaginando o gráfico sem essas duas faixas, é ainda mais notória a queda nos números.

As figuras 82 e 83 fornecem novas perspectivas acerca da redução do número de alertas por elemento de rede e serviço no primeiro caso, e apenas por serviço no segundo caso.



Rank	Host/Service	Alarms
1	appserver / CheckDiskCounter	62
2	gecadcore / CheckDiskCounter	35
3	webgecad / CheckDiskCounter	34
4	zeus / CheckDiskCounter	24
5	zeus / CheckMemoryPagesCounter	14
6	appserver / CheckMem	11
7	zeus / CheckDiskDriveC	11
8	zeus / CheckDiskDriveU	10
9	zeus / CheckCPUload	9
10	zeus / CheckNtCounter	8

Figura 82 – Top alarmes por elemento de rede e serviço



Rank	Service	Alarms
1	CheckDiskCounter	158
2	CheckMem	19
3	CheckMemoryPagesCounter	18
4	CheckDiskDriveC	11
5	icmp_ping_alive	10
6	CheckDiskDriveU	10
7	CheckNtCounter	10
8	CheckCPUload	9
9	CheckDiskDriveD	8
10	CheckServicesZeus	6

Figura 83 – Top de alarmes por serviço

É de realçar que serviços ligados à memória RAM (“CheckMem”) e serviços ligados ao espaço em disco nas partições do sistema (“CheckDiskDriveC”) aparecem em lugares cimeiros nas tabelas, apesar da extinção dos mesmos nas semanas trinta e sete e trinta e oito, o que indica, tal como referido anteriormente, que se tratam de serviços prioritários e sobre os quais era importante tomar medidas. O que foi conseguido de duas formas: colocando o foco de processamento localmente e executando acções de correcção logo após as análises nos elementos de rede, através da solução baseada em Agentes móveis.

A solução proposta garante uma grande distribuição da gestão. Os Agentes móveis responsáveis por executar tarefas de gestão ligados à RAM, CPU, processos e espaço em disco fazem-no localmente no elemento de rede, consumindo os recursos que cada elemento tem para oferecer, retornando os resultados no fim, tendo-os transportado pela rede. Foi assim diminuído algum do *polling* excessivo que caracteriza este tipo de aplicações tradicionais.

O trabalho junto dos activos de rede foi melhorado graças ao facto de ser possível migrar um Agente móvel para um elemento de rede que esteja directamente conectado ao activo em questão. Desta forma, conseguimos duas coisas: a primeira, retirar o processamento e o *polling* da aplicação central; a segunda, tanto o AgentSwitch como o AgentMRTG podem trabalhar em conjunto para efectuar análises que se complementem fornecendo dados mais completos sobre o equipamento.

Para terminar, existem cerca de trinta gráficos de performance na solução tradicional cujo trabalho pode ser substituído por Agentes MRTG, fazendo os estudos em cada elemento de rede e recriando os gráficos apenas na estação gestora, numa fase posterior. É possível também a criação de novos gráficos com código próprio da ferramenta, sobre o comportamento de qualquer tipo de OID que se enquadre nos moldes da avaliação da performance e que queiramos estudar.

6 Conclusões e Trabalho Futuro

Este capítulo apresenta a síntese do trabalho desenvolvido e apresentado nesta tese. São também apresentados os objectivos alcançados, as limitações e perspectivas de desenvolvimento futuro.

6.1 Resumo

Este trabalho discute os problemas existentes na abordagem tradicional de gestão de redes, procurando demonstrar a utilidade e as vantagens da utilização de uma abordagem baseada em Agentes móveis. Paralelamente, foi proposta uma arquitectura baseada em Agentes móveis para um sistema de gestão a utilizar num cenário real, o GECAD.

A utilização de Agentes móveis na distribuição e delegação de tarefas de gestão não difere do paradigma tradicional no que aos propósitos diz respeito. Neste trabalho, em qualquer um dos paradigmas, conseguimos efectuar a observação de tendências de fluxo de dados e diagnóstico de situações anómalas na rede. A forma como essas tarefas de gestão são executadas é que é diferente. Permitindo, uma distribuição das tarefas de gestão por Agentes móveis com características reactivas. O leque de objectivos das tarefas de gestão é alargado com os Agentes móveis, sendo possível criar uma estrutura mais eficiente, que distribui o mecanismo de gestão de maneira a que possamos ultrapassar barreiras impostas pela arquitectura centralizada.

De modo a cumprir os objectivos do trabalho, foi feito, num primeiro momento, um estudo sobre o tema *Gestão de Redes*. Este estudo prévio contempla todo o capítulo dois da dissertação e permitiu a familiarização com conceitos fundamentais para abordar novas temáticas. São identificadas as áreas funcionais, bem como os dois modelos das arquitecturas de gestão: centralizadas e descentralizadas. Ficou bem patente a diferença entre um modelo e o outro, uma vez que foram explicados os vários cenários que é possível encontrar em cada uma das arquitecturas. Também os modelos de gestão foram estudados, nomeadamente o modelo de informação e o relacional. Com o modelo de informação de gestão, conseguimos saber em pormenor como aceder a cada recurso gerido e correspondente representação lógica, ou seja, a informação de gestão e a forma como esta está estruturada. O modelo relacional de gestão descreve a forma como o gestor e o sistema gerido comunicam.

De seguida foi feito o estudo relativo à aplicação do paradigma dos Agentes móveis na gestão de redes (Capítulo 3). Para tal, foi definido o que é um Agente de software e que características deve ter para ser considerado um Agente móvel. Foram também estudados os sistemas Multi-Agente, uma vez que, as características e os requisitos necessários, para o desenvolvimento de um sistema

de gestão para o cenário real, levam ao uso de capacidades distribuídas, cujo projecto, implementação, operação e configuração, permitam a integração de competências numa rede dinâmica. Estas capacidades podem ser aproveitadas com sucesso, através de uma solução baseada em sistemas Multi-Agente. A comunicação nestes sistemas foi abordada, bem como as arquitecturas com comunicação directa ou assistida, esta última a escolhida para a aplicação que viria a ser desenvolvida. Após o conhecimento do que é um sistema Multi-Agente, era fulcral terminar com a introdução aos Agentes móveis, que numa comunidade de Agentes com tarefas de gestão, representariam a base deste trabalho. Foram exploradas as suas características e vantagens sobre a arquitectura tradicional cliente-servidor. Foram também estudados alguns aspectos práticos, como por exemplo, quais as linguagens suportadas para o seu desenvolvimento, assim como que plataformas existem disponíveis.

Paralelamente, foi efectuado um estudo exaustivo do cenário real, isto é, a rede do GECAD. O quarto capítulo está dividido em duas partes. A primeira é constituída pelas secções 4.1 até à secção 4.4, onde é feita uma abordagem à arquitectura e topologia da rede, sendo descritos elementos fundamentais como áreas de trabalho existentes, cablagem, números totais por tipo de equipamentos, classe de endereços, domínio, número de utilizadores, entre outros. Seguiu-se uma análise da rede, ao meio físico da mesma. As salas de trabalho são descritas em pormenor, sendo dados exemplos de como a noção de onde está o quê é muito importante, salientando as nossas limitações e o que podemos fazer para aproveitar ao máximo aquilo que possuímos. Realce para a descrição da sala de trabalho I403, onde se encontra quase toda a base da infra-estrutura informática da rede do GECAD. A primeira parte do capítulo quatro termina com a secção 4.4, onde são detalhados todos os componentes da rede, desde bastidores e formas como estes estão conectados, até aos equipamentos activos de rede, sem descurar os principais servidores e respectivos serviços. É com base nesta análise que é permitido: configurar a ferramenta de gestão tradicional, e propor e desenvolver uma aplicação baseada em Agentes móveis. A segunda parte do capítulo contempla a descrição da ferramenta de gestão actual, com todas as suas características e respectivo motor de gestão. Todos os detalhes da configuração são descritos na secção 4.5.3. Terminamos o estudo com o levantamento dos requisitos para uma nova solução de gestão a implementar baseada em Agentes móveis. Na secção 4.5.4 encontramos um conjunto de melhorias ou correcção de actuais falhas da solução de gestão baseada no paradigma tradicional, que conduziram à solução proposta: o desenvolvimento de um protótipo que cumpra esses requisitos, provando que os dois paradigmas se podem complementar de forma harmoniosa na rede do GECAD, e que uma abordagem com Agentes móveis é uma mais-valia.

Em seguida, foi implementada a solução proposta para a rede do GECAD (capítulo 5). Foi necessário criar um protótipo de acordo com os requisitos referidos na secção 4.5.4 e que no mínimo conseguisse efectuar tarefas de gestão após a migração para um determinado elemento de rede. Após a elaboração da arquitectura, foram descritos os tipos de Agentes da mesma e as suas interacções. Depois, partiu-se para uma fase onde se pretendia escolher as tecnologias a utilizar. Sabendo que a elaboração do projecto baseava-se na implementação de uma comunidade de Agentes, a escolha recaiu na plataforma OAA. Trata-se de uma plataforma muito utilizada por diferentes investigadores do GECAD. No entanto, era necessário dotar os Agentes da comunidade de capacidades SNMP. Para tal, foi escolhida a API "WebNMS SNMP", cujas bibliotecas foram importadas para o projecto de modo a que pudesse ser utilizada. Foi implementada a comunidade de Agentes (descrita na secção 5.3.4), assim como a Interface. A terminar, são dados alguns exemplos práticos, nomeadamente sobre o "AgentDisk", o "AgentRAM" e o "AgentMRTG". Estes Agentes permitem exemplificar as capacidades de um Agente móvel que execute tarefas de gestão, reconhece determinados eventos e consegue reagir aos mesmos com a autonomia desejada.

Por último, depois de elaborado o sistema, foram feitos vários testes aos vários módulos do mesmo, com o objectivo de encontrar eventuais falhas existentes, no sentido de evoluir a solução.

6.2 Objectivos Alcançados

O primeiro objectivo deste trabalho foi efectuar um estudo aprofundado da área de gestão de redes. Foram estudados conceitos e noções fundamentais, desde a importância que a gestão de redes assume em qualquer ambiente informático, passando pelas áreas funcionais existentes, até ao Modelo Arquitectural, Modelo de informação de gestão e Modelo relacional de gestão.

O segundo objectivo foi estudar um novo paradigma para a gestão de redes baseado em Agentes móveis. A definição de Agente, os tipos e classificações foram abordados, de maneira a que fosse possível compreender o funcionamento de um sistema Multi-Agente. Com estes conceitos bem definidos, introduziu-se os Agentes móveis. Tornou-se muito importante reconhecer as suas características, comparar paradigmas de gestão (Tradicional e Agentes móveis), identificar as vantagens dos Agentes móveis, aprofundar o conceito de mobilidade nos mesmos.

Outro objectivo consistiu em efectuar uma análise detalhada sobre um cenário real, a rede do GECAD. Analisar a arquitectura e topologia da rede, o meio físico e os componentes da mesma tornou-se crucial para implementar uma solução de gestão segundo o paradigma tradicional.

Paralelamente, foi necessário estabelecer quais os requisitos necessários para aplicar o paradigma dos Agentes móveis à solução tradicional. Foi criada uma arquitectura, definido o tipo de Agentes

que a constituiriam e como seria efectuada a comunicação entre eles. Para implementar a comunidade de Agentes criando um sistema Multi-Agente, era necessário escolher uma plataforma. Após estudo e análise, a plataforma escolhida foi a OAA, surgindo também a necessidade de enriquecer os Agentes da comunidade com capacidades SNMP. Com esse objectivo, a API "WebNMS SNMP" foi importada para o projecto, para que quando fosse necessário dotar um Agente de aptidões SNMP, tal fosse possível de executar. O passo seguinte passou por implementar e descrever cada um dos Agentes da comunidade, bem como a criação de uma interface gráfica. Foi proposta a divisão da gestão da rede em três segmentos, cobertos pelos vários Agentes móveis, que foram também divididos por tarefas e tipo de intervenção. Existem catorze Agentes móveis na solução de gestão implementada, cada um com diferentes objectivos, apresentados no capitulo 5, assim como quais os seus atributos e principais métodos. Foram criados diagramas para a compreensão do funcionamento geral do sistema, do ciclo de vida dos Agentes móveis e do comportamento aquando da sua migração.

De entre todos os Agentes, existem alguns com uma maturidade maior a nível de desenvolvimento. Isso ficou demonstrado nos exemplos apresentados na secção 5.3.7. Todos os Agentes têm as bases necessárias para actuar em conformidade com o que lhes foi destinado, mas no caso dos exemplos referidos na mesma secção, a capacidade de intervenção é maior. Isto sucedeu porque a solução com Agentes móveis foi pensada e desenvolvida de forma a retirar o máximo rendimento da mesma no menor espaço de tempo possível. As atenções foram assim dirigidas inicialmente para aquele tipo de tarefas que aliviavam ou melhoravam a solução de gestão tradicional no imediato.

Ficou também demonstrado ao longo do capítulo 5 a capacidade enriquecedora da solução de gestão com Agentes móveis relativamente a uma solução tradicional. A gestão da rede pode ser partilhada entre os dois paradigmas, o que conduz a uma distribuição da mesma, com todas as vantagens que daí advêm. Tarefas que eram executadas pela solução de gestão tradicional podem agora ser executadas pela aplicação de Agentes móveis, de forma mais eficiente e com distribuição do processamento, sendo obtidas as melhorias esperadas, referidas na secção 5.4.

6.3 Limitações e Trabalho Futuro

Este trabalho procura demonstrar a aplicabilidade do paradigma dos Agentes móveis na gestão de redes, distribuindo as competências.

Foi proposta uma arquitectura e desenvolvido e implementado um sistema para um cenário real. No entanto, os Agentes podem ser melhorados, dotando-os de maior maturidade, maior autonomia e uma maior capacidade de responder a cenários, cada vez mais complicados. Os interfaces gráficos

do sistema, embora cumpram com o necessário, poderão sofrer alterações. Pelo que, no futuro, poderão ser feitas melhorias na interacção com o utilizador e apresentar resultados e/ou relatórios visualmente mais atractivos. Podem, também ser feitas melhorias em relação ao modo como esses resultados são tratados após a sua obtenção. Nem sempre é necessário que tomem uma forma visual, no entanto, devem ser guardados em ficheiros para futura análise de eventos. Isto acontece no protótipo, principalmente pela quantidade de testes que necessitamos de fazer, mas pode ser melhorado de forma a tratar todos os resultados de uma forma mais uniforme e consistente.

Relativamente aos Agentes móveis, o "AgentRam", "AgentDisk" e o "AgentMRTG" já estão numa fase bastante avançada, com bons resultados e uma autonomia excelente, podendo sempre haver melhorias, no sentido de alargar as possibilidades de intervenção em cenários mais complexos. Quanto aos restantes podem ser alvo das seguintes alterações:

- O "AgentCPU" consegue detalhar, tal como o "AgentRAM", que processos estão a consumir o quê em termos de processamento. No futuro, o trabalho pode e deve ser orientado no sentido de conseguir libertar recursos (processamento) de forma autónoma e inteligente. É ainda assim muito complicado encontrar um equilíbrio no que ao CPU diz respeito, quando precisamos de mais processamento e as opções existentes não passem por mais do que matar processos desnecessários;
- O "AgentPROC" consegue devolver tabelas similares aos seus colegas "AgentCPU" e "AgentRAM", indicando que processos estão a correr no sistema. Isto é vital, uma vez que na solução de gestão tradicional, muitos dos pollings efectuados serviam para verificar se determinado processo estaria, ou não, a correr no elemento de rede. Se não está, o Agente tenta reactivá-lo, usando os meios que a Microsoft providencia e que algumas vezes não são tão eficazes como desejaríamos. Este Agente pode ser evoluído no futuro, no sentido de garantir quase a 100% que qualquer processo é recuperado;
- O "AgentNE" poderá ser alvo de modificações que visem melhorar o estudo sobre a conectividade do adaptador de rede do elemento, uma vez que já é possível guardar informação de erros de entrada e saída, mas com o auxílio do "AgentMRTG" pode ser analisada ainda mais informação;
- O "AgentNetwork" possui funções muito específicas e será muito difícil acrescentar algo mais;
- O "AgentSwitch" possui capacidade para guardar vários atributos dos equipamentos activos de rede, mas o trabalho poderá dirigir-se ainda mais sobre esses dados, para que possa ser criada, mantida e estudada uma base de dados para estudar tendências de fluxo na rede, entre outros.

Basicamente, existe muito trabalho realizado, mas muito por explorar. Apenas a possibilidade de, para além da mobilidade e de todas as suas características, poder munir estes Agentes de inteligência adicional e capacidades de aprendizagem, é um factor muito atractivo. Trata-se de uma área emergente que permite que muita investigação possa ser conduzida.

Bibliografia

[1] (2010). "Gerenciamento SNMP", Available: http://pt.scribd.com/doc/7231009/Gerenciamento-SNMP

[2] A. Clemm, "Network management fundamentals". Indianapolis, Ind.: Cisco Press, 2007.

[3] P. J. Martins, "Comparação dos Paradigmas Cliente/Servidor e Agentes Móveis: Um estudo em Gerência de Redes", MsC, Universidade Federal de Santa Catarina, 2002.

[4] H. Luo, "Agent-based Network Management System", MsC, Department of Computer Science, Beijing Institute of Technology, Beijing, 2000.

[5] G. Goldszmidt and Y. Yemini, "Distributed management by delegation", *Proceedings of the 15th International Conference on Distributed Computing Systems*, pp. 333-340, 1995.

[6] F. E. B. d. M. Nogueira, "Gestão de Redes Um Serviço de Valor Acrescentado", MsC, Departamento de Engenharia Electrotécnica e de Computadores, Faculdade de Engenharia da Universidade do Porto, Porto, 2004.

[7] D. Perkins, "RMON: remote monitoring of SNMP-managed LANs": Prentice Hall PTR, 1999.

[8] T. Fioreze, *et al.*, "Comparing web services with SNMP in a management by delegation environment", *Integrated Network Management IX*, pp. 601-614, 2005.

[9] J. Schonwalder and J. Quittek, "Secure management by delegation within the Internet Management Framework", *Integrated Network Management Vi*, pp. 687-700, 1999.

[10] S. F. Bush and A. B. Kulkarni, "Active networks and active network management: a proactive management framework": Kluwer Academic/Plenum Publishers, 2001.

[11] M. Bernichi and F. Mourlin, "Network Management with mobile agent toolkit", *Third 2008 International Conference on Convergence and Hybrid Information Technology, Vol 2, Proceedings*, pp. 1012-1017, 2008.

[12] P. Braun and W. Rossak, "Mobile agents : basic concepts, mobility models, and the Tracy toolkit". San Francisco, CA: Elsevier : Morgan Kaufmann ; Heidelberg : Dpunkt.verlag, 2005.

[13] D. Chess, *et al.*, "Mobile agents: Are they a good idea?", *Mobile Object Systems*, vol. 1222, pp. 25-45, 1997.

[14] D. Gavalas, *et al.*, "A mobile agent platform for distributed network and systems management", *Journal of Systems and Software*, vol. 82, pp. 355-371, Feb 2009.

[15] B. Liu, *et al.*, "Agent cooperation in multi-agent based network management", *Proceedings of the 8th International Conference on Computer Supported Cooperative Work in Design, Vol 2*, pp. 283-287, 2004.

[16] S. Makki and S. V. Wunnava, "Application of mobile agents in managing the traffic in the network and improving the reliability and quality of service", *IMECS 2006: International Multiconference of Engineers and Computer Scientists*, pp. 32-35, 2006.

[17] F. A. C. e. S. Mouta, "Ambiente de Desenvolvimento de Sistemas Multiagente para Controlo e Supervisão de Processos em Aplicações Distribuídas", PhD, Faculdade de Engenharia Universidade do Porto, Porto, 1996.

[18] Y. Shi, *et al.*, "A mobile agent- and policy-based network management architecture", in *Computational Intelligence and Multimedia Applications, 2003. ICCIMA 2003. Proceedings. Fifth International Conference on*, 2003, pp. 177-181.

[19] S. R. Stuart Wagner, Keith Landgraf, "Agent Technology for Network Management", I. Telcordia Technologies, Ed., ed, 2002.

[20] L. Tang and B. Pagurek, "A comparative evaluation of mobile agent performance for network management", *Ninth Annual Ieee International Conference and Workshop on the Engineering of Computer-Based Systems, Proceedings*, pp. 258-267, 2002.

[21] I. Satoh, "Building reusable mobile agents for network management", *Ieee Transactions on Systems Man and Cybernetics Part C-Applications and Reviews*, vol. 33, pp. 350-357, Aug 2003.

[22] A. M. Souza, "Gerenciamento de Redes Baseado em Agentes Móveis", Tecnólogo em Processamento de Dados, Centro de Ciências Formais e Tecnologia, Universidade Tiradentes, Aracaju, Brasil, 2001.

[23] H. Li and L. Meng, "Reconstruction Mechanism Based on Distributed Intelligent Agents for Network Management", *2008 7th World Congress on Intelligent Control and Automation, Vols 1-23*, pp. 5143-5147, 2008.

[24] E. Y. L. Timon C. Du, An-Pin Chang, "Mobile Agents in Distributed Network Management", *Communications of the ACM*, vol. 46, 2003.

[25] S. Aidarous and T. Plevyak, "Managing IP networks: challenges and opportunities": John Wiley, 2003.

[26] S. B. Morris, "Network management, MIBs and MPLS". Upper Saddle River, N.J. ; London: Prentice Hall PTR, 2003.

[27] N. Greenfield, "FCAPS Management for the Smart Grid High-level Summary", A. I. S. Engineering, Ed., ed, 2009.

[28] M. A. Miller, "Managing internetworks with SNMP", 2nd ed. ed. New York: M&T Books, 1997.

[29] F. S. Systems, "FCAPS White Paper", ed, 2005.

[30] C. A. Koon-Seng Lim, Rolf Stadler, "Decentralizing Network Management", *IEEE electronic Transactions on Network and Service Management*, 2009.

[31] M. J. Viamonte, "SMI e MIBs", ISEP, Ed., ed. Porto, 2008.

[32] A. S. TANENBAUM, "Redes de Computadores": CAMPUS, 2003.

[33] "MIB Browser", Available: http://ireasoning.com/mibbrowser.shtml

[34] AdventNet. (2007). "ADVENTNET SNMP UTILITIES 5", Available: http://www.webnms.com/snmputilities/help.html

[35] M. J. Viamonte, "SNMP – Simple Network Management Protocol", ISEP, Ed., ed. Porto, 2008.

[36] M. J. Viamonte, "SNMPv2 e SNMPv3", ISEP, Ed., ed. Porto, 2008.

[37] M. J. Viamonte, "Novas Arquitecturas de Gestão", ISEP, Ed., ed. Porto, 2008.

[38] P. R. B. d. A. Pereira, "Políticas Activas Para Gestão de Redes", PhD, Universidade Técnica de Lisboa - Instituto Superior Técnico, Lisboa, 2003.

[39] J. S. Strassner, "Policy-based network management : solutions for the next generation". Amsterdam ; London: Morgan Kaufmann Publishers, 2004.

[40] C. R. António Silva, "Sistemas Baseados em Agentes", ISEP, Ed., ed. Porto, 2008.

[41] A. S. Carlos Ramos, "Agent-Based Systems", ISEP, Ed., ed. Porto, 2008.

[42] L. M. Helder Coelho, "Agent Workshop", F. d. C. d. U. d. Lisboa, Ed., ed. Lisboa.

[43] L. P. Reis, "Coordenação em Sistemas Multi-Agente: Aplicações na Gestão Universitária e Futebol Robótico", PhD, FEUP, Porto, 2003.

[44] "AgentesInteligentes e Sistemas Multi-agente", Available: http://www.slidefinder.net/a/agentes_inteligentes_sistemas_multi_agente/11074443

[45] S. Pleisch, "State of the Art of Mobile Agent Computing - Security, Fault Tolerance, and Transaction Support", IBM Research, Zurich Research Laboratory, Ruschlikon, Switzerland1999.

[46] X. Yang, "Mobile Agent Computing in Electronic Bussiness: Potentials, Designs and Challenges", Phd, Griffith University, Australia, 1995.

[47] A. R. Cardoso, "Integrando Agentes Móveis com Sistemas Legados para Gerenciamento de Redes ATM Heterogêneas", MsC, Universidade Federal de Campina Grande, Campina Grande 2002.

[48] D. B. Lange and M. Oshima, "Programming and deploying Java mobile agents with Aglets": Addison-Wesley, 1998.

[49] L. Zhao, "Java-Based Mobile Agents", S. o. E. a. C. Science, Ed., ed. Wellington, New Zealand.

[50] S. Fernandes, "Introdução aos Agentes Autónomos, Agentes Móveis", INESC, Ed., ed, 2000.

[51] R. R. Gudwin. (2009). "Java e Agentes Móveis", Available: http://www.dca.fee.unicamp.br/~gudwin/courses/IA009/Aulas.html

[52] E. Lourenço, "Agentes Móveis", U. F. d. Paraná, Ed., ed. Curitiba, 2006.

[53] T. Sundsted. (1998). "Agents on the move", Available: http://www.javaworld.com/javaworld/jw-07-1998/jw-07-howto.html

[54] J. K. Chun, *et al.*, "Network management based on PC communication platform with SNMP and mobile agents", *22nd International Conference on Distributed Computing Systems Workshop, Proceedings*, pp. 222-227, 2002.

[55] L. D. L. Gibeon Soares de Aquino, "Agentes Móveis", C. d. I. d. UFPE, Ed., ed. Recife, Brasil.

[56] R. R. Gudwin. (2009). "Agentes Móveis", Available: http://www.dca.fee.unicamp.br/~gudwin/courses/IA009/Aulas.html

[57] J. Hu, *et al.*, "A Network Management Model Based on Mobile Agent System", *2008 International Conference on Apperceiving Computing and Intelligence Analysis (Icacia 2008)*, pp. 279-283, 2008.

[58] X. Huang, *et al.*, "Study of the Active Network Management System Model based on Agent", *2008 4th International Conference on Wireless Communications, Networking and Mobile Computing, Vols 1-31*, pp. 11165-11168, 2008.

[59] S. K. Huy Hoang To, Bala Srinivasan, "Mobile Agents for Network Management: When and When Not!", presented at the 2005 ACM Symposium on Applied Computing, 2005.

[60] C. P. Iwan Adhicandra, Ebrahim Shaghouei, "Using Mobile Agents to Improve Performance of Network Management Operations", presented at the PostGraduate Networking Conference, Liverpool John Moores University, 2003.

[61] A. Koliousis and J. Sventek, "A trustworthy mobile agent infrastructure for network management", *2007 10th Ifip/Ieee International Symposium on Integrated Network Management (Im 2009), Vols 1 and 2*, pp. 383-390, 2007.

[62] F. C. Knabe, "Language Support for Mobile Agents", Phd, School of Computer Science, Carnegie Mellon University, Pittsburgh, 1995.

[63] A. D. Group, "The Aglets 2.0.2 User's Manual", ed, 2009.

[64] D. Horvat, *et al.*, "Mobile agents and Java mobile agents toolkits", *Telecommunication Systems*, vol. 18, pp. 271-287, 2001.

[65] A. Marco Avvenuti, "MobileRMI: a toolkit for enhancing Java Remote Method Invocation with mobility", presented at the 6th ECOOP Workshop on Mobile Object Systems: Operating System Support, Security and Programming Languages, France, 2000.

[66] W. Grosso, "Java RMI". Sebastopol, CA ; Farnham: O'Reilly, 2002.

[67] J. Byassee. (2002, 2009). "Unleash mobile agents using Jini", Available: http://www.javaworld.com/javaworld/jw-06-2002/jw-0628-jini.html

[68] "GroundWork Monitor Community Edition (GWMCE)", Available: http://www.gwos.com/forums/index.php?/forum/43-groundwork-monitor-community-edition-gwmce/

[69] E. Galstad. (2009). "Nagios Version 3.x Documentation", Available: http://library.nagios.com/library/products/nagioscore/manuals/

[70] I. GroundWork Open Source, "GroundWork Monitor Community Edition 6.0 Preview Install Guide – VMware Virtual Appliance", ed, 2009.

[71] "Technology and Architecture", Available: http://www.gwos.com/technology/

[72] "GroundWork Monitor Theory of Operations", Available: http://proddoc.groundworkopensource.com/Bookshelf_RoboHelp/Welcome_to_GroundWork_Monitor/Theory_of_Operations.htm

[73] C. Thomas, "GroundWork Monitor Architecture Overview", I. GroundWork Open Source, Ed., ed. San Francisco, 2010.

[74] "RRDtool", Available: http://oss.oetiker.ch/rrdtool/

[75] "MRTG - The Multi Router Traffic Grapher", Available: http://oss.oetiker.ch/mrtg/index.en.html

[76] "Monitoring Windows Machines", Available: http://nagios.sourceforge.net/docs/3_0/monitoring-windows.html

[77] "Configure SNMP Agent in Windows 2000/XP/2003", Available: http://www.windowsreference.com/networking/configure-snmp-agent-in-windows-2000xp2003/

[78] (2008). "OAA® 2.3.2 Documentation", Available: http://www.ai.sri.com/oaa/distribution/v2.3/2.3.2/

[79] "OAA® 2.3.2 Downloads", Available: http://www.ai.sri.com/oaa/distribution/v2.3/2.3.2/download/windows/oaa2.3.2_02.zip

[80] G. Marreiros, "Agentes de Apoio à Argumentação e Decisão em Grupo", PhD, Universidade do Minho, Braga, 2007.

[81] "WebNMS SNMP API 4", Available: http://www.webnms.com/snmp/index.html

[82] (2009). "Developing Applications Using SNMP API", Available: http://www.webnms.com/snmp/help.html

[83] (2010). "WebNMS SNMP API 4 Datasheet", Available: http://www.webnms.com/snmp/snmpapi_datasheet.html

[84] ""A Free SNMP MIB Search Engine for SNMP MIBs"", Available: http://www.mibdepot.com/index.shtml

[85] (2009). "Clear Windows 7 Cache", Available: http://www.addictivetips.com/windows-tips/clear-windows-7-cache/

[86] P. Bright. (2010). "Behind the Windows 7 memory usage scaremongering", Available: http://arstechnica.com/microsoft/news/2010/02/behind-the-windows-7-memory-usage-scaremongering.ars

[87] M. E. Russinovich, *et al.*, "Windows internals", 5th ed. ed. Redmond, Wash.: Microsoft Press, 2009.

[88] "MemBoost", Available: http://memboost.50g.com/index.html#