DESENVOLVIMENTO E INTEGRAÇÃO DE SERVIÇOS SNMP NUMA PLATAFORMA DE AGENTES MÓVEIS

Rodrigo Reis, Paulo Simões, Luis Silva, Fernando Boavida

CISUC – Dep. Eng. Informática
Universidade de Coimbra
Pólo II, Pinhal de Marrocos
3030 Coimbra, Portugal

E-mail: rodrigo@student.dei.uc.pt, {psimoes, luis, boavida}@dei.uc.pt

Sumário. No âmbito do Projecto JAMES foi desenvolvida na Universidade de Coimbra uma infra-estrutura para Agentes Móveis explicitamente orientada para a Gestão de Sistemas e Redes.

Neste artigo será apresentada a forma como a arquitectura SNMP foi integrada e implementada na plataforma do Projecto JAMES, de modo a que os Agentes Móveis possam aceder a "recursos SNMP" e constituir, eles próprios, serviços de gestão com interface SNMP.

1. Introdução

As aplicações de gestão de sistemas e redes (GSR) são maioritariamente baseadas num de dois protocolos: o SNMP [Rose94], mais orientado para a gestão de redes de dados, e o CMIP [ISO90], bastante disseminado em redes de telecomunicações. Estes protocolos enquadram-se em arquitecturas cliente-servidor estáticas e centralizadas, com uma entidade central responsável pelo processamento de todos os dados resultantes da comunicação com os elementos de rede. Esta centralização contribui para tornar as aplicações de gestão pouco flexíveis e pouco escaláveis.

Diversas abordagens alternativas têm sido exploradas no sentido de descentralizar as tarefas de GSR [Znaty97], destacando-se os conceitos percursores de gestão por delegação, as redes activas, o uso da tecnologia CORBA [OMG], a gestão baseada em tecnologia WEB, os agentes inteligentes e, mais recentemente, os Agentes Móveis (AM).

Resumidamente, um AM pode ser descrito como um pequeno programa capaz de migrar para uma máquina remota, onde executará determinadas funções, e posteriormente migrar para outras máquinas com o objectivo de aí cumprir outras tarefas. O principal atractivo desta abordagem é a distribuição do processamento pela rede, levando o código de encontro aos dados e não o inverso. Cada tarefa é executada na localização mais conveniente, podendo essa localização ser escolhida de forma dinâmica e flexível.

Com a potencial capacidade de permitir soluções com menor tráfego de rede, mais escaláveis, mais flexíveis e mais robustas, os AM são um paradigma em pleno crescimento em diversas áreas de aplicação [Pham98], tais como computação móvel, comércio electrónico, aplicações para a Internet, pesquisa de informação, trabalho cooperativo, gestão de redes e telecomunicações [Chess95] [Hermans] [Magedanz96] [Pham98].

Neste momento existem no mercado diversas implementações comerciais de agentes móveis, todas elas mais ou menos generalistas. Entre os principais competidores, são de destacar a IBM [Aglets], a Mitsubishi [Concordia], a General Magic [Odyssey], a ObjectSpace [Voyager], a AdAstra [JumpingBeans] e o IKV++ [Grasshopper].

Na nossa opinião, estes produtos são interessantes mas têm-se mostrado demasiado generalistas para dar uma resposta satisfatória na área de GSR. É por essa razão que uma das principais tarefas do projecto JAMES [Silva99a] é o desenvolvimento de uma infra-estrutura de AM explicitamente desenhada e afinada para aplicações de GSR. No âmbito desta focagem na gestão de redes é atribuída especial atenção à interoperabilidade com tecnologias já existentes, tais como o SNMP, o CORBA e o CMIP. Existem duas razões que tornam esta interoperabilidade crucial para o sucesso dos AM em GSR.

Em primeiro lugar, porque essas tecnologias permitem aceder de forma expedita à informação e aos serviços de gestão. Ainda que constituam uma solução atractiva para executar tarefas inteligentes mais próximo da informação de gestão, os AM não irão substituir os protocolos clássicos no interface com os serviços de gestão em ambientes heterogéneos. Pelo contrário, eles complementam esses protocolos oferecendo metáforas de programação mais poderosas e flexíveis. De qualquer forma, os protocolos clássicos continuarão a ser necessários, seja (i) porque alguns dos Elementos de Rede (*Network Elements*: NEs) não podem alojar Agentes Móveis, (ii) porque os serviços de gestão dos NEs não são acessíveis directamente pelos agentes móveis, ou (iii) porque os interfaces de acesso aos serviços de gestão não estão normalizados.

Em segundo lugar, porque as aplicações de gestão baseadas em AM necessitam de coexistir com os sistemas de gestão já instalados, uma vez que se destinam habitualmente a resolver determinados problemas específicos (implantação gradual) e não a fornecer soluções globais e completas. Para que os AM sejam uma ferramenta apropriada na criação ou extensão de novos serviços de gestão, é importante que esses novos serviços possam posteriormente ser utilizados pelas plataformas já instaladas. Normalmente, a linguagem "nativa" dessas plataformas é o SNMP, o CMIP ou o CORBA.

Poder-se-ia considerar que a integração de AM com os protocolos de gestão clássicos pode ser relegada para as próprias aplicações de gestão baseadas em AM, utilizando eventualmente *stacks* protocolares generalistas já existentes. Porém, a mobilidade inerente aos AM, os problemas de segurança e o controlo de recursos imposto pela utilização de aplicações baseadas em AM limitam seriamente o uso destes protocolos sem um suporte explícito por parte da infra-estrutura. Por esta razão, a plataforma JAMES inclui esse suporte para interoperabilidade com aplicações SNMP.

Neste artigo será descrita uma parcela do trabalho de integração do protocolo SNMP com a plataforma JAMES: o Interface de Programação SNMP (API SNMP) para comunicação entre AM e recursos SNMP. O resto do artigo está organizado da seguinte forma: a Secção 2 discute trabalho relacionado; a Secção 3 apresenta uma breve descrição do JAMES e da arquitectura de alto nível dos seus serviços SNMP; e a Secção 4 foca diversos aspectos do desenho e desenvolvimento do API SNMP. A Secção 5 conclui o artigo. Em [Silva99b] é apresentada uma descrição mais alargada de diversos aspectos da plataforma, incluindo a integração dos serviços SNMP e outras características associadas a GSR.

2. Trabalho Relacionado

2.1 Ao Nível das Plataformas de AM

O CORBA foi a primeira das tecnologias aplicadas em GSR a merecer atenção por parte da comunidade dos AM, o que não surpreende uma vez que o CORBA é útil num largo espectro de aplicações distribuídas, e não apenas em GSR.

Decorrem neste momento duas iniciativas de normalização na área, o MASIF (*Mobile Agent System Interoperability Specification* [Masif98]), focado na interoperabilidade entre infra-estruturas de AM, e o consórcio FIPA (*Foundation for Intelligent Physical Agents* [FIPA]), de âmbito bastante alargado mas com especial destaque na comunicação entre agentes inteligentes de sistemas distintos. Actualmente poucas plataformas suportam estas duas normas: apenas existe uma plataforma MASIF (o Grasshoper [Grasshoper], da IVK++) e são muito poucas as plataformas que suportam, mesmo de forma parcial, especificações FIPA. De qualquer modo, um número significativo de plataformas de AM já providencia suporte, mais ou menos rudimentar, para aceder a serviços distribuídos usando o CORBA.

Já o SNMP não é directamente suportado por nenhuma implementação comercial de AM, pertencendo as poucas experiências registadas a projectos de investigação.

Os projectos INCA [Nicklish98], da NEC CCRLE-Berlin, AMETAS/NetDoctor [Zapf99], da Universidade de Frankfurt, e DIANA [Marcus99], do EURECOM, fazem uso do SNMP como forma *ad-hoc* de permitir aos Agentes aceder a informação de gestão residente nos elementos de rede, sem contudo apresentar preocupações de suporte de mobilidade.

O projecto Discovery [Lazar97], da Universidade de Maryland, propõe o uso do modelo de informação SNMP para representar a informação interna do ambiente de execução dos agentes móveis. Para isso, as MIBs já existentes na máquina hospedeira são agregadas com as MIBs próprias da plataforma de AM, ficando directamente acessíveis pelos AM por meio de um API interno. Esta abordagem apresenta dois inconvenientes. Em primeiro lugar, limita a forma como se apresenta a informação interna do sistema de AM. Em segundo lugar, obriga a modificações nos serviços SNMP nativos da máquina hospedeira que afectam consideravelmente a portabilidade da plataforma.

O projecto Astrolog/MAGENTA [Sahai98], que usa agentes de *software* para suporte da mobilidade do operador da rede, apresenta uma abordagem próxima da abordagem do Discovery, com agentes SNMP, internos e monolíticos, para acesso à informação local e para passagem de informação aos nós coordenadores da infra-estrutura.

O projecto *Perpetuum Mobile Procura* (PMP [Bieszczad97]), da Universidade de Carleton, propõe o uso do DPI (Distributed Protocol Interface [Wijnen94]) como forma indirecta de aceder a agentes SNMP locais e de estender agentes SNMP com a instalação de serviços implementados em Java. Este é um dos poucos projectos onde o SNMP é usado no acesso a recursos de gestão e como mecanismo de instalação de novos serviços. Contudo, o uso de DPI afecta a portabilidade, pois implica que o agente SNMP nativo suporte este protocolo.

2.2 Ao Nível de Ferramentas SNMP em JAVA

O desenvolvimento de ferramentas de suporte ao protocolo SNMP em JAVA é recente, existindo poucas implementações disponíveis que sejam interessantes.

O *AdventNet SNMP* [AdventNet] é um conjunto de ferramentas de desenvolvimento de aplicações SNMP. Estando actualmente na versão 2.1, é uma opção popular para construção de aplicações baseadas em SNMP. Este pacote suporta SNMPv1 e SNMPv2c e inclui algumas funcionalidades do SNMPv3. A estrutura deste *software* é dividida num API de baixo nível, que contém a funcionalidade básica do SNMP, e num conjunto de APIs de alto nível, organizados em JavaBeans.

A iniciativa *Java Management Extensions* (JMX) [JMX], derivada do *Java Management API* (JMAPI), é um projecto da *Sun Microsystems* com o objectivo de oferecer um API normalizado para o desenvolvimento de aplicações de GSR. Esta iniciativa teve um longo período de inactividade, tendo sido retomada muito recentemente. Aparentemente, os objectivos continuam a ser os mesmos do JMAPI, e um conjunto de especificações foi disponibilizado para discussão pública. Nestas especificações está incluída a especificação de um API para desenvolvimento de aplicações que utilizam o protocolo SNMP. Futuramente serão disponibilizadas implementações das JMX; porém não é possível fazer uma previsão em relação à data de lançamento, pois o ritmo de trabalho desta iniciativa tem-se mostrado inconstante.

O *Java Dynamic Management Kit* (JDMK) [JDMK] é outro produto da *Sun Microsystems* orientado para o desenvolvimento de soluções de GSR. Sendo mais orientado para a construção de agentes de gestão nos próprios elementos de rede, baseia-se numa infra-estrutura modular guiada por serviços de gestão distribuídos por toda a rede. Este pacote oferece integração com vários protocolos de gestão (destacando-se o SNMP) e permite, por exemplo, comunicar com aplicações de gestão baseadas neste protocolo. Deve no entanto mencionar-se que os *agentes* do JDMK não são agentes móveis, ainda que possam ser dinamicamente instalados nos elementos de rede por mecanismos *push/pull*.

3. Arquitectura dos Serviços SNMP do JAMES

3.1 Arquitectura Geral da Plataforma JAMES

O projecto JAMES tem como principal objectivo explorar a tecnologia de AM em aplicações na área de GSR, e tem duas linhas de acção complementares. Por um lado, o desenvolvimento de uma plataforma de AM adaptada aos requisitos impostos por essas aplicações (coordenado pela Universidade de Coimbra). Por outro, o desenvolvimento de produtos de software que aproveitem essa plataforma dos pontos de vista técnico e comercial (coordenado pela Siemens SA e pela Siemens AG). Nesta subsecção, iremos apresentar uma descrição muito resumida da plataforma JAMES.

A plataforma JAMES fornece um ambiente de execução para AM. Na versão actual, existe uma distinção entre o ambiente instalado na máquina responsável pela gestão da infra-estrutura (designado por *JAMES Manager*) e o ambiente instalado nos outros nós da rede (NEs), referido como *JAMES Agency*. A Figura 1 apresenta a forma como a plataforma é organizada.

A máquina onde está instalado o *JAMES Manager* é responsável pela manutenção de toda a infra-estrutura de AM, fornecendo diversos interfaces entre os AM e os seus utilizadores, sejam eles operadores humanos ou aplicações. Estes interfaces permitem controlar, de forma remota, o lançamento e funcionamento dos AM e da própria plataforma.

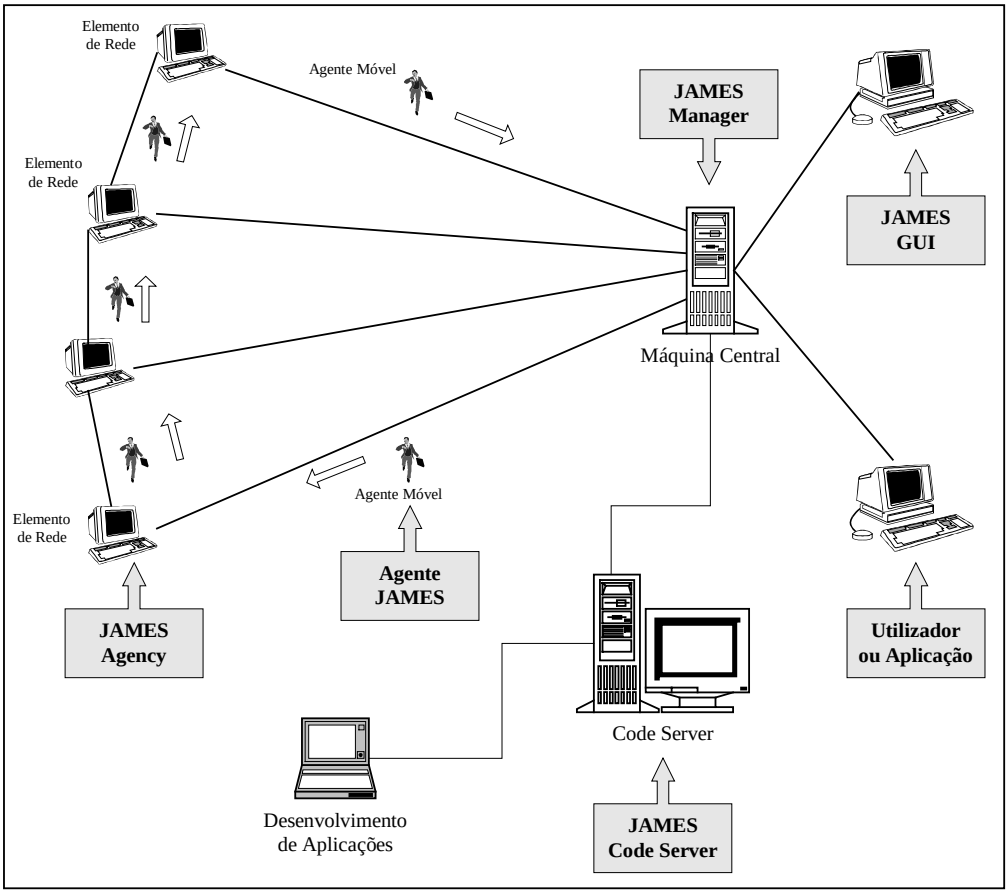

Figura 1: Organização Geral da Plataforma JAMES

Cada elemento da rede mantém uma *Java Virtual Machine* (JVM) onde é executada a *JAMES Agency*. Os AM podem migrar entre as várias *Agências* de forma a aceder a informação e serviços de gestão, executar tarefas, e enviar informação processada para os utilizadores. As *Agências* evitam a intrusão de agentes não autorizados usando diversos mecanismos de autenticação. A comunicação entre os diversos elementos da infra-estrutura, para controlo do sistema, troca da dados e migração de agentes, é efectuada por *sockets*. A migração dos AM pode ser efectuada de forma atómica.

As aplicações podem consistir apenas num ou vários AM, ou podem incluir módulos "estáticos" para coordenar as actividades dos AM e criar um interface com o utilizador final. Os AM são desenvolvidos em Java e usam o API do JAMES para controlar a sua mobilidade. Depois de desenvolvidos, em Java, os AM devem ser registados e armazenados no *JAMES Code Server*. Este servidor mantém assim uma relação de todos os AM autorizados, funcionando assim como ponto central na autorização e autenticação da cada instância em execução. O registo dos agentes é efectuado usando HTTP, e a comunicação entre o *Code Server* e o *JAMES Manager* usa *sockets*.

Por razões de escalabilidade, futuras versões do JAMES permitirão múltiplos *Code Server* e *JAMES Manager*.

A plataforma JAMES, ainda numa versão preliminar, inclui já as seguintes características:

- Serviço de actualização remota de AM e *agências*;
- Protocolo de migração atómica;
- Suporte de tolerância a falhas por mecanismos de *checkpoint-and-restart*;
- Itinerários reconfiguráveis de forma dinâmica durante a execução;
- Suporte para *disconnected computing*;
- Mecanismos de monitorização do sistema e dos AM;
- Controlo da utilização de recursos;
- Mecanismos de distribuição de código com suporte de *caching* e *prefetching*;
- Interface CORBA;
- Suporte SNMP;
- Suporte para os paradigmas "*simple agent*" "*migratory agent*" e "*master/worker model*".

A descrição de algumas destas características, que está fora do âmbito deste artigo, pode ser encontrada em [Silva99b].

3.2 Requisitos dos Serviços SNMP do JAMES

No desenho da infra-estrutura de integração do SNMP com a plataforma JAMES houve várias considerações a ter em conta, devido às características tanto do protocolo como da plataforma. A principal consideração foi, obviamente, a necessidade de suporte explícito para a mobilidade das aplicações (ou seja, dos AM). Por outro lado a solução final deveria ter um impacto mínimo na plataforma, em termos de desempenho, complexidade e portabilidade.

Pretende-se que a plataforma JAMES inclua uma infra-estrutura de serviços SNMP já integrados e disponíveis para o desenvolvimento de aplicações de GSR em várias áreas de aplicação. Esta infra-estrutura deve reduzir consideravelmente o tempo e o custo de desenvolvimento deste tipo de aplicações. De forma a preservar a portabilidade da plataforma, estes serviços devem ser integralmente desenvolvidos na linguagem Java. Como se pode ver na Figura 2, três serviços SNMP são considerados:

- um serviço que permite a interacção de Agentes Móveis com agentes-SNMP, actuando assim como gestores SNMP;
- suporte de comunicação entre gestores SNMP e os AM da plataforma;
- e um serviço de gestão permitindo que plataformas de gestão administrem a plataforma JAMES através do SNMP.

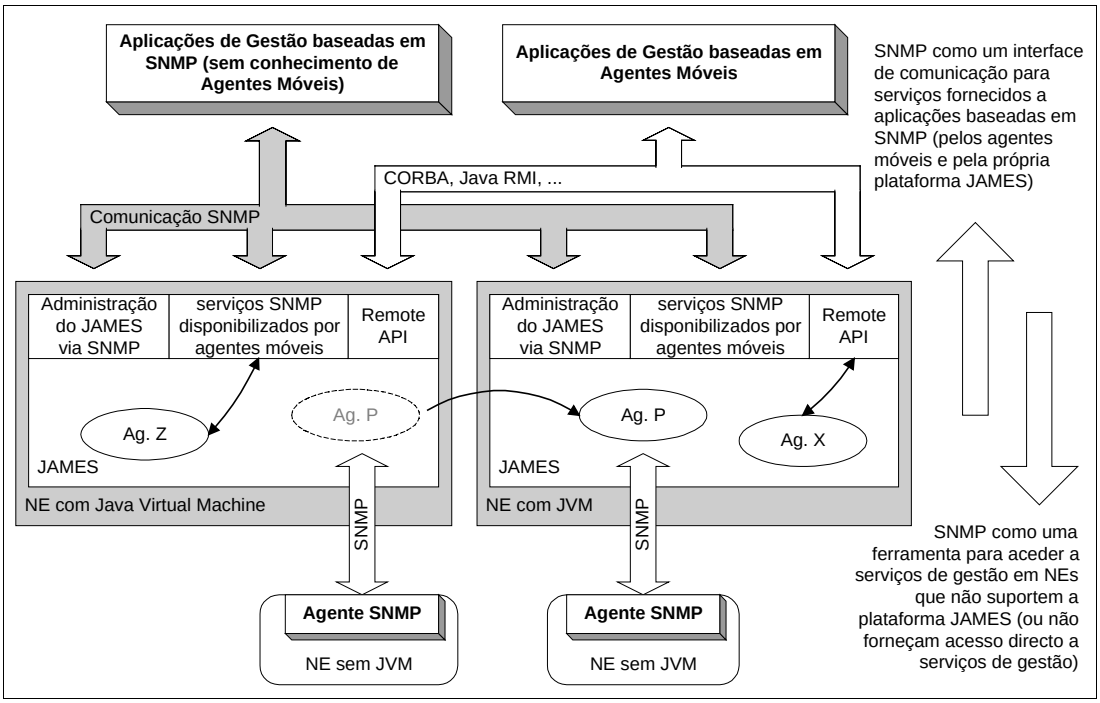

Figura 2: Infra-estrutura de Integração

Neste artigo é dada especial atenção ao chamado "API SNMP", integrado no primeiro serviço. O API SNMP fornece a um AM a possibilidade de interagir com um Agente SNMP. Com o intuito de construir um API que se adapte da melhor forma à plataforma JAMES e ao protocolo SNMP foram definidos os seguintes objectivos para o API SNMP:

- deve ser simples de utilizar, permitindo ao programador desenvolver aplicações sem se preocupar com detalhes intrínsecos ao protocolo SNMP;
- deve ser flexível, permitindo ao programador desenvolver aplicações funcionais e escaláveis;
- deve ser portável, de forma a poder ser utilizado em ambientes heterogéneos;
- deve aproveitar as funcionalidades da plataforma JAMES, de modo a poder desenvolver aplicações que façam uso dessas;
- não deve interferir nem comprometer a funcionalidade da plataforma JAMES.

3.3 Alternativas no Desenho dos Serviços SNMP

Uma das primeiras questões que se colocaram no desenho do API SNMP foi determinar se seria viável utilizar um dos *stacks* SNMP já existentes no mercado ou se seria necessário construir um novo *stack*. Optou-se pela construção de um *stack* de raiz pelas razões que se enunciam:

- os raros *stacks* SNMP disponíveis em Java (comerciais ou não) não apresentam todas as funcionalidades desejadas para a plataforma ao nível de suporte de mobilidade;
- esses *stacks* comerciais, cujo código-fonte não está disponível, não davam garantias de portabilidade (por meio de código 100% Java) e implicavam um acréscimo de complexidade não aceitável;
- adicionalmente, o uso de *stacks* comerciais de terceiros poderia limitar o valor comercial da plataforma, uma questão sensível para os parceiros industriais do Projecto JAMES;
- seria necessário adaptar o *stack* às necessidades da infra-estrutura: se bem que existissem *stacks* SNMP aceitáveis para o desenvolvimento de aplicações de GSR clássicas, estes não incluem funcionalidades que tirem partido da plataforma JAMES.

A construção de um *stack* SNMP de raiz, apesar de implicar um maior esforço de programação, permitiu construir uma solução que melhor se adapta e explora as potencialidades da plataforma JAMES.

Um outro aspecto importante a considerar foi a localização do *stack* SNMP no âmbito da plataforma JAMES. Várias foram as alternativas consideradas (no Agente Móvel, na Agência e num Agente Serviço), tendo cada uma delas as suas vantagens e inconvenientes, tal como se descreve de seguida.

***Stack* localizado no Agente (*Fat Agent*).** Nesta alternativa o *stack* fica localizado no Agente, como mostra a Figura 3, sofrendo a mobilidade do AM um impacto considerável. O *stack* SNMP está contido no AM, aumentando o seu tamanho e a sua latência na migração entre agências. Adicionalmente, a existência de notificações assíncronas obriga o agente a permanecer estacionário até recepção destas ou, em alternativa, a abandonar transacções SNMP em curso. Adicionalmente, o AM não pode receber *Traps* SNMP de forma transparente.

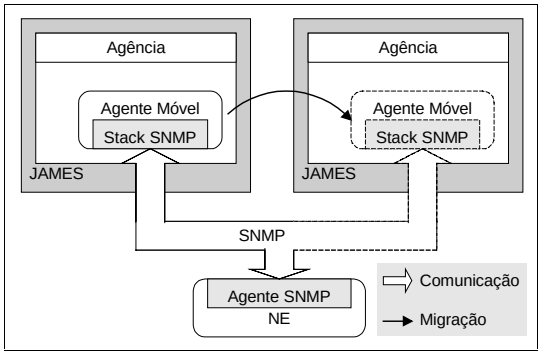

Figura 3: *Stack* no Agente (*Fat Agent*)

***Stack* localizado na Agência (*Fat Agency*).** O *stack* SNMP localizado na Agência (Figura 4) resolve os problemas de recepção de notificações assíncronas e de latência na migração dos MAs. Porém, surgem problemas de escalabilidade devidos à partilha de um único motor protocolar por todos os AM residentes na Agência. Passa ainda a ser necessário um protocolo de comunicação entre os AM e a agência, para efectuar as transacções SNMP.

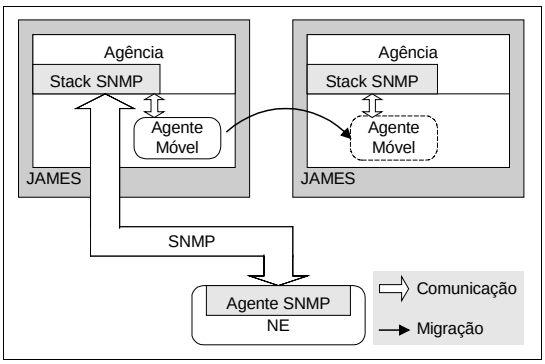

Figura 4: *Stack* na Agência (*Fat Agency*)

Agente-Serviço SNMP: "Agente Especial". Esta alternativa de integração é uma solução híbrida para o problema em questão. Existe um AM que contém o *stack* SNMP e que fornece este serviço aos outros agentes móveis da plataforma, como se pode ver na Figura 5. Habitualmente, este AM permanece estacionário e "serve" os outros AM. Estes podem migrar pela rede sem preocupações de mobilidade (o serviço reencaminha os acontecimentos SNMP para a nova localização do AM) e sem ter que incluir um *stack* SNMP completo no seu código. Para funcionar de forma satisfatória, este modelo necessita de um suporte para comunicação entre AM suficientemente flexível (independente da localização, com passagem de mensagens assíncronas e com bom desempenho) para não limitar o funcionamento do sistema.

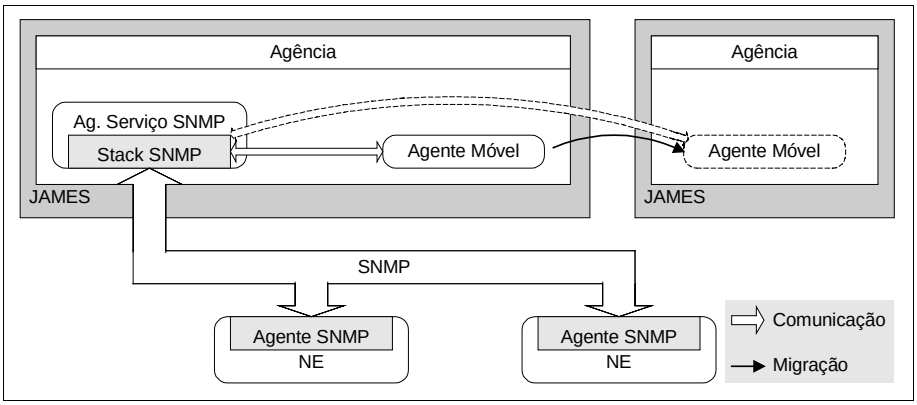

Figura 5: Agente-Serviço SNMP: "Agente Especial"

Comparação entre Alternativas. A Tabela 1 mostra uma comparação sumária entre as alternativas de integração descritas, segundo vários critérios relevantes para o problema. Para implementação no JAMES optou-se pelo modelo "Agente Serviço SNMP". Esta escolha deveu-se a várias razões:

- um Agente-Serviço SNMP agrega muitas das vantagens de cada uma das outras alternativas. Como foi dito, esta alternativa é híbrida na sua essência;
- grande parte dos problemas existentes nas outras alternativas é resolvida;
- os problemas que ficam por resolver associados às restantes alternativas têm uma resolução simples.

O ponto crítico deste modelo é, como já foi mencionado, a comunicação inter-agentes. Esta comunicação é importante não só para a alternativa de integração a implementar, mas também para toda a plataforma JAMES. Para já o sistema funciona sobre mecanismos rudimentares de comunicação, estando a ser ultimado em paralelo um serviço de comunicação mais completo e funcional construído sobre o JavaSpaces [JavaSpaces], da Sun.

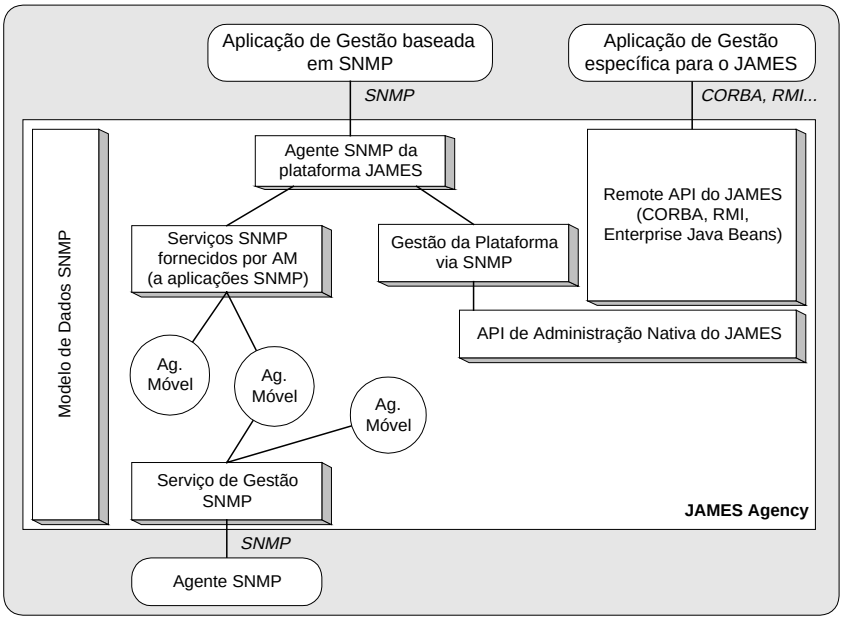

Figura 6: Arquitectura das Extensões SNMP do JAMES

	Stack no agente	Stack na Agência	Agente Serviço SNMP
Impacto na Agência	Nenhum.	Forte: agência com maior *footprint* (o que não deverá ser significativo) e mais computação.	Baixa: bastará existir um mecanismo de "agent directory" que permita contactar/invocar agentes específicos.
Impacto no Agente	Forte, nos agentes com código SNMP: maiores dimensões do código, maior carga computacional.	Baixo: bastará utilizar serviços disponibilizados pela agência.	Baixo: bastará utilizar serviços disponibilizados pela agência/agente.
Modelo de Programação	Necessidade de "rearmar comunicação" à chegada a cada agência. Modelo de programação ajustado ao *stack* SNMP e *migration aware*.	Necessidade de definir um interface de serviço SNMP agente/agência Agente deverá conhecer este interface e formas de manipular dados.	Necessidade de definir um interface de serviço SNMP agente/agente. Agente deverá conhecer este interface e formas de manipular dados.
Mobilidade	Impossibilita recepção de Traps. Impossibilita conclusão de "transacções" em curso.	Agência pode concluir transacções pelo agente, enviando-lhe posteriormente as mensagens adequadas.	Elevada: "agente serviço" pode concluir transacções e enviar, posteriormente, mensagens para novo paradeiro.
Competição por recursos (portos reservados)?	Competição por portos reservados. Cada agente tem o seu motor protocolar.	Agência toma conta dos recursos críticos.	Apenas poderá existir um agente serviço para o processamento de *traps*. Poderão existir vários para as outras transacções SNMP, de modo a partilhar a carga.
Soluções para Competição?	Trap-Multiplexer: num agente "especial" ou na agência (neste caso com filtragem limitada à origem da *Trap*).	Agência não tem competidores.	Agente processador de *Traps* deve ser único (agência deve fornecer mecanismos de controlo e localização de agentes suficientes para garantir isto).
Plug-in dinâmico e "on-demand" ?	Sim, com reservas no mecanismo de recepção de *traps*.	Integração pode ser decidida no momento da instalação da agência (ou, eventualmente, de forma dinâmica), mas retira elegância à solução.	Sim: instalar serviço SNMP corresponde a invocar agentes móveis.
Comunicação entre AM?	Nula (a não ser num eventual "Trap-Mux" localizado num agente particular)	Nula: serviço prestado pela agência	Forte: serviço prestado por agentes
Escalabilidade	Elevada: cada agente dispõe do seu próprio motor protocolar.	Baixa: único motor protocolar partilhado pelos agentes (tal como no modelo clássico). Comunicação com agentes pode também ser factor limitativo.	Agentes serviço podem ser múltiplos (excepto para *Traps*), aliviando a carga de cada motor protocolar. Comunicação entre AM é o factor limitativo.
Integração	Pouco integrada: no limite qualquer *stack* SNMP em Java poderia ser utilizado.	Fortemente integrada: SNMP é um serviço oferecido pela plataforma. Modelo de integração pode considerar-se pouco elegante e pouco flexível.	Fortemente integrada: SNMP é um serviço fornecido por um agente (ainda que com algum controlo por parte da plataforma). Modelo de integração integra-se no paradigma.
Extensibilidade a outros serviços?	Sim, mas pouco interessante.	Sim, mas com eventual sobrecarga da agência (se plug-in dinâmico não for suportada correctamente).	Sim: dependerá da capacidade do serviço de directoria.

Tabela 1: Comparação entre as alternativas de integração

3.4 Arquitectura Geral das Extensões SNMP do JAMES

Nesta secção é apresentada de forma bastante sucinta a arquitectura geral das Extensões SNMP presentes na plataforma JAMES, estabelecida com base nos requisitos e opções discutidos nas Subsecções 3.2 e 3.3, e que engloba o API SNMP discutido neste artigo. Uma apresentação mais alargada desta arquitectura pode ser encontrada em [Simoes99].

As extensões SNMP do JAMES são constituídas por diversos serviços modulares (Figura 6) colocados fora do núcleo da plataforma e dinamicamente instaláveis "on-demand", não impondo assim uma sobrecarga permanente na infra-estrutura. A maior parte dos serviços consistem eles próprios em AM ("Agentes Serviço", cfr. Subsecção 3.3), com permissões para aceder aos diversos recursos necessários, que fornecem serviços aos outros AM. A agência fornece um serviço de directoria onde os AM podem localizar (ou requerer a instalação de) os "Agentes Serviço". No futuro, novos tipos de serviços podem ser integrados no JAMES da mesma forma, sem necessidade de modificar o núcleo da plataforma.

São incluídos três tipos distintos de serviços (Tabela 2):

- serviços de manipulação de dados SNMP;

- serviços de comunicação com recursos SNMP;
- e serviços de comunicação com aplicações SNMP.

Grupo Serviços	Módulo, Descrição e Observações
Manipulação de Dados	**Manipulação de Dados** Descrição. Os serviços de manipulação de dados incluem diversas ferramentas para manipulação das mensagens e tipos de dados definidos no protocolo SNMP. Estas ferramentas asseguram todo o tratamento necessário, desde a codificação de baixo nível até à representação de alto nível. Suporte/Requisitos de Mobilidade. Ainda que estes serviços tenham sido desenvolvidos de raiz, não haveria qualquer impedimento técnico no uso de ferramentas comerciais, uma vez que o paradigma de AM não lhes impõe quaisquer restrições específicas. Formato de implementação. Estas ferramentas estão disponíveis na forma de classes Java disponíveis para o programador de AM e usadas na implementação dos restantes serviços.
Serviço de Gestão SNMP (comunicação com recursos SNMP)	**Stack SNMP (Manager API)** Descrição. Estes serviços permitem aos AM efectuar transacções (operações *request*) sobre agentes SNMP, por meio de um interface que usa os conceitos clássicos de sessão, contexto, operações de request e notificações assíncronas. Suporte/Requisitos de Mobilidade. A mobilidade dos agentes móveis não é afectada, uma vez que eles recebem as notificações assíncronas independemente da sua localização. Formato de implementação. Estes serviços são prestados por um ou mais "Agentes Serviço", acessíveis a partir dos AM "correntes" por comunicação inter-agente. Os "Agentes Serviço" são instalados/desinstalados de forma dinâmica, de acordo com as necessidades.
	Trap Listener Descrição. Estes serviços recebem Traps SNMP e reenviam-nas para os AM que tenham, previamente, registado o seu interesse em tomar conhecimento da sua chegada. O serviço usa conceitos de registo, filtragem e notificação semelhantes aos de outros *Trap Multiplexers*. Suporte/Requisitos de Mobilidade. A mobilidade dos agentes móveis não é afectada, uma vez que eles recebem as notificações assíncronas independemente da sua localização. Formato de implementação. Estes serviços também são prestados por um "Agente Serviço".
Comunicação com aplicações SNMP	**Agente SNMP da plataforma JAMES** Descrição. Constitui o motor protocolar de um agente SNMP embutido na plataforma como forma de comunicação com aplicações baseadas em SNMP. É alimentado de informação (MIBs) pelos dois serviços seguintes, por meio de um interface proprietário mas semelhante ao AgentX [RFC2257]. Suporte/Requisitos de Mobilidade. Mobilidade não é relevante. Formato de implementação. Estes serviços também são prestados por um "Agente Serviço".
	Mediador da gestão da plataforma Descrição. Usa o serviço anterior para permitir a administração da própria infraestrutura JAMES via SNMP. Implementa uma MIB específica com diversos dados sobre o funcionamento da agência JAMES. Funciona como mediador entre o formato SNMP e o interface, mais rico e funcional, fornecido pelo API de gestão nativo do JAMES. Suporte/Requisitos de Mobilidade. A mobilidade não é relevante. Formato de implementação. Estes serviços também são prestados por um "Agente Serviço".
	Mediador dos serviços fornecidos por AM Descrição. Mediador entre o Agente SNMP da plataforma e os AM que pretendem fornecer serviços usando o interface SNMP. Suporte/Requisitos de Mobilidade. Suporta a migração dos AM interlocutores. Formato de implementação. Estes serviços também são prestados por um "Agente Serviço".

Tabela 2: Componentes das Extensões SNMP do JAMES

4. Interface de Programação SNMP para a Plataforma JAMES

Nesta secção irão ser abordados diversos aspectos directamente relacionados com o trabalho de implementação do API SNMP. Relativamente á Tabela 2, este API engloba o Serviço de Manipulação de Dados, o *stack* SNMP (Manager API) e o *Trap Listener*. A implementação foi faseada por vários módulos. A Figura 7 mostra o processo de implementação da infra-estrutura de integração SNMP.

O Modelo de Dados contém as primitivas essenciais para a implementação do protocolo SNMP; o módulo de APIs apoia-se no Modelo de Dados: permite estabelecer sessões e gerir a comunicação com agentes SNMP; o API de Alto Nível fornece ao programador um interface que lhe permite abstrair-se dos detalhes do protocolo SNMP, bem como utilizar as funcionalidades da plataforma JAMES para desenvolver aplicações de GSR.

O API construído suporta actualmente a versão 1 do protocolo SNMP, estando prevista para breve o suporte da versão 2c.

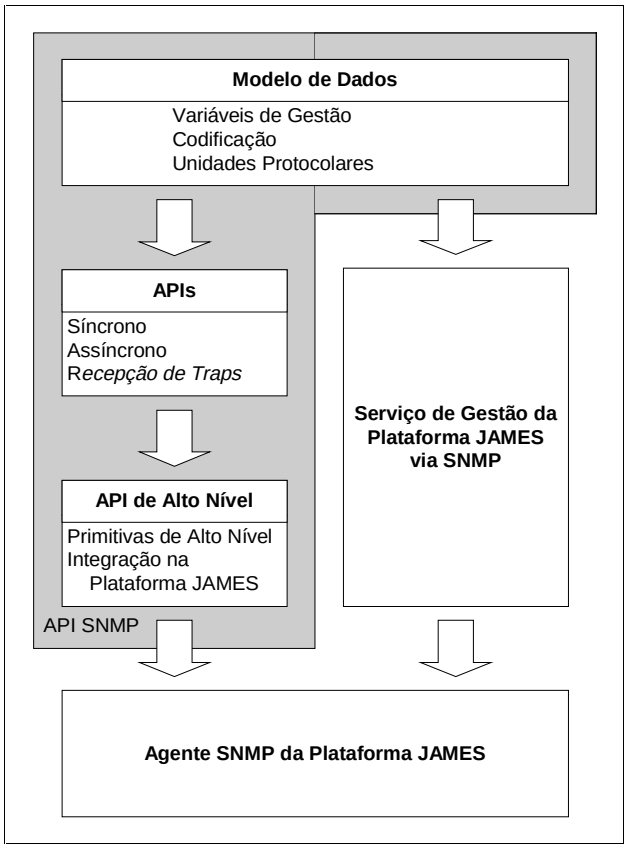

Figura 7: Processo de Implementação da Infra-Estrutura SNMP

4.1 Estrutura do Interface de Programação

A Figura 8 mostra a estrutura do API SNMP. Este é dividido em seis módulos. Os módulos de Codificação/Descodificação, Sintaxe e Unidades Protocolares (PDUs) contém as classes que servem de base à interacção com o protocolo SNMP. Os APIs Síncrono e Assíncrono, bem como o *Trap Listener* são suportados por esta base, contendo classes orientadas às tarefas necessárias no estabelecimento de uma sessão e interacção com um Agente SNMP.

Como se pode ver, os módulos não funcionam isoladamente, existindo ligações resultantes da partilha de informação bem como da utilização de primitivas de um módulo por outro. Cada módulo do API desempenha uma tarefa específica. As secções seguintes descrevem a função que cada módulo desempenha no contexto do API SNMP.

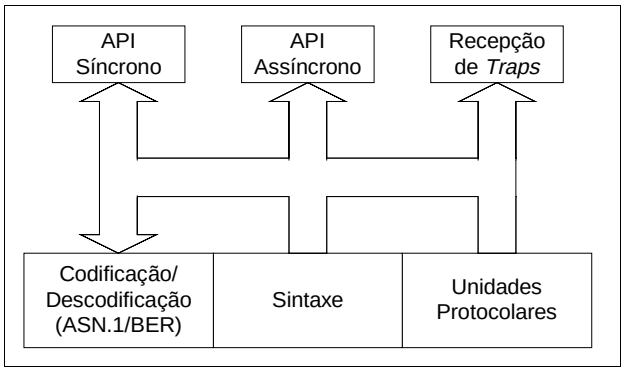

Figura 8: Estrutura do API SNMP

4.2 Módulo de Sintaxe

O módulo de Sintaxe contém classes que representam os tipos de dados definidos pela Estrutura de Informação de Gestão SNMP [RFC1155]. O encapsulamento destes em classes permite um maior rigor

na utilização destes tipos. Para além da representação, estas classes contém vários métodos úteis para uma manipulação mais fácil da informação de gestão. A figura 9 mostra a estrutura das classes incluídas no módulo de Sintaxe. Dado que são muitas classes, foi preocupação neste módulo tornar o Interface de Programação o mais homogéneo possível, de forma a normalizar a estrutura das classes.

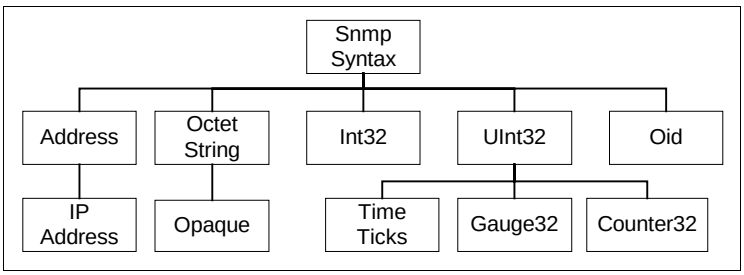

Figura 9: Estrutura do Módulo de Sintaxe

4.3 Módulo de Unidades Protocolares

A comunicação entre gestor e agente no protocolo SNMP é feita recorrendo a 5 PDUs: *get-request, get-response get-next-request*, *set-request* e *trap*. O módulo de Unidades Protocolares contém classes que encapsulam estas PDUs. A Figura 10 mostra a estrutura deste módulo. A classe *Pdu* encapsula as PDUs *get-request, get-response get-next-request* e *set-request*. A única diferença na estrutura destas é o tipo, de resto têm estrutura iguais. Por esta razão é que a classe *Pdu* encapsula estas quatro PDUs. A classe *TrapPdu* encapsula a PDU *trap*. Esta PDU tem uma estrutura diferente das restantes PDUs.

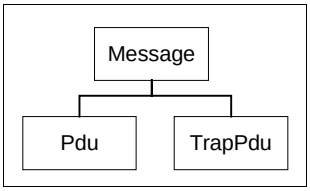

Figura 10: Estrutura do Módulo de PDUs

4.4 Módulo de Codificação/Descodificação ASN.1/BER

Antes de enviar ou receber uma PDU SNMP, é necessário proceder à sua codificação/descodificação. A codificação das PDUs é feita usando as *Basic Encoding Rules* para a *Abstract Syntax Notation One* [ISO87]. O Módulo de Codificação/Descodificação ASN.1/BER é o responsável por proceder a esta codificação/descodificação. Não é necessário ao programador conhecer ou utilizar este módulo, pois a sua utilização é encapsulada de forma transparente.

Este módulo é composto por várias classes estáticas, sem uma estrutura hierárquica. A mecânica deste módulo consiste numa classe principal que recebe as PDUs e que invoca outras classes do módulo para realizar tarefas mais específicas, distribuindo o processo de codificação e descodificação.

4.5 API Síncrono

Este módulo permite estabelecer uma sessão com um agente SNMP. A classe Session contém a informação relativa ao Agente SNMP, nomeadamente a sua localização (endereço IP) e a comunidade. Esta classe é comum ao API Síncrono, Assíncrono e *Trap Listener*.

A classe SynchronousSession, derivada da classe Session, é responsável pela comunicação com o Agente SNMP. Esta comunicação resume-se ao envio de PDUs e posterior recepção de respostas. Os métodos responsáveis pelo envio são bloqueantes; porém pode ser especificado um *timeout* para controlar o tempo de espera de uma resposta. Para além deste mecanismo existe outro permitindo definir o número de retransmissões das PDUs, caso não se receba resposta do Agente SNMP.

4.6 API Assíncrono

Usando este módulo é possível comunicar com um Agente SNMP de uma forma assíncrona. Os métodos de envio de PDUs da classe *AsynchronousSession* não bloqueiam à espera da resposta. Tal como no API Síncrono, pode-se definir o *timeout* e o número de retransmissões de um pedido. Este módulo encarrega-se da recepção das respostas do Agente SNMP. Aquando da chegada de uma resposta, esta é armazenada numa lista para posterior tratamento pelo AM.

4.7 Recepção de *Traps*

Uma *Trap* é uma PDU enviada por um Agente SNMP destinada a notificar o gestor acerca de informação importante. Estas *Traps* são recebidas de forma assíncrona. Este módulo é constituido apenas por uma classe, de nome *TrapListener*. Uma instância desta classe encarrega-se de receber *Traps* e de armazená-las numa lista para posterior tratamento pelo AM.

5. Conclusões

Este artigo descreveu o trabalho de implementação de um subconjunto dos serviços SNMP incluídos na plataforma de AM do projecto JAMES. A arquitectura proposta para integração de SNMP nesta plataforma aumenta a interoperabilidade entre SNMP e AM, alargando significativamente o campo de aplicação desta nova tecnologia. Os AM deixam de ser soluções isoladas e passam a interactuar com os principais componentes da arquitectura SNMP: recursos (agentes) SNMP e aplicações de gestão SNMP.

Relativamente a outros trabalhos similares, a arquitectura de integração do JAMES permite maior grau de interoperabilidade, por também abranger as aplicações SNMP, e distingue-se pela forma dinâmica como integra os serviços SNMP, que podem ser instalados ou não de acordo com as necessidades.

Agradecimentos

O projecto JAMES é patrocinado pelo programa Eureka (E!1921) e parcialmente financiado pela Agência de Inovação (ADI). Agradece-se o apoio de todos os colaboradores do projecto, sem os quais o trabalho que deu origem a este artigo não teria sido possível.

Referências

[AdventNet] AdventNet SNMP, http://www.adventnet.com/products/snmpbeans

[AMETAS] Projecto AMETAS, Universidade de Frankfurt, http://www.witrans.uni-frankfurt.de/WiTrans/messe/aktuell/cebit98_ex6e.html

[Aglets] IBM Aglets Workbench, http://www.trl.ibm.co.jp/aglets/

[Bieszczad97] Bieszczad, A., "Advanced Network Management in the Network Management Perpetuum Mobile Procura Project", Technical Report SCE-97-07, Systems and Computer Engineering, Carleton University, 1997

[Chess95] D. Chess, B. Grosof, C. Harrison, D. Levine, C. Parris, G. Tsudik. "Itinerant Agents for Mobile Computing", IEEE Personal Communicatons Magazine, 2(5), pp. 34-97, October 1995

[Concordia] Concordia, http://www.meitca.com/HSL/Projects/Concordia/

[FIPA] Foundation for Intelligent Physical Agents, http://www.fipa.org/

[Grasshopper] Grasshopper, http://www.ikv.de/products/grasshopper/

[Hermans] Björn Hermans, "Intelligent Software Agents on the Internet", http://www.hermans.org/agents/index.html

[ISO87] "Information Processing - Open Systems Interconnection – Specification of Basic Encoding Rules for Abstract Syntax Notation One", International Organization for Standardization, 1987

[ISO90] "ISO/IEC 9595: Information technology - Open Systems Interconnection - Common management information Service definition", International Organization for Standardization, International Electrotechnical Commission, 1990

[JavaSpaces] JavaSpaces, http://java.sun.com/products/javaspaces

[JDMK] Java Dynamic Management Kit, http://www.sun.com/java-dynamic

[JMX] Java Management Extensions, http://www.javasoft.com/products/JavaManagement

[JumpingBeans] Jumping Beans, http://www.JumpingBeans.com/

[Lazar97] S. Lazar, D. Sidhu, "Discovery, A Mobile Agent Framework for Distributed Application Development",

Technical Report, Maryland Center for Telecommunications Research, University of Maryland Baltimore County, 1997

[Magedanz96] T.Magedanz, K.Rothermel, S.Krause. "Intelligent Agents: An Emerging Technology for Next Generation Telecommunications", Proc. INFOCOM'96, March 1996, San Francisco, CA

[Marcus99] K. Marcus, P. Conti, M. Cheikhrouhou, J. Labetoulle."Intelligent Agents for Network Management: Fault Detection Experiment". In Proceedings of IM'99 (Sixth IFIP/IEEE International Symposium on Integrated Network Management), Boston, EUA, 1999

[Masif98] "Mobile Agent System Interoperability Facilities Specification", OMG TC Document orbos/97-10-05, 1998

[Nicklish98] J. Nicklish, J. Quittek, A. Kind, S. Arao, "INCA: an Agent-based Network Control Architecture", in Proceedings of IATA'98, Paris, 1998

[Odyssey] General Magic Odyssey, http://www.genmagic.com/agents/

[OMG] OMG, "The Common Object Request Broker Architecture and Specification", 1995

[Pham98] V. Pham, A.Karmouch. "Mobile Software Agents: An Overview", IEEE Communications Magazine, pp. 26-37, Julho 1998

[RFC1155] M. Rose, K. McCloghrie, "Structure and Identification of Management Information for TCP/IP based Internets", RFC 1155, 1990

[RFC2257] M. Daniele, B. Wijnen, D. Francisco, "Agent Extensibility(AgentX) Protocol Version 1", RFC 2257, 1998

[Rose94] M. Rose, "The Simple Book - An Introduction to Management of TCP/IP-based Internets, 2nd Edition", Prentice-Hall International Inc., 1994

[Sahai98] A. Sahai, C. Morin, "Enabling a Mobile Network manager (MNM) Through Mobile Agents", in Proceedings of Mobile Agents, Second International Workshop MA'98, Stuttgart, Alemanha, 1998

[Silva99a] L. Silva, P. Simões, J. Gabriel e Silva, J. Boavida, P. Monteiro, J. Rebhan, C. Renato, L. Almeida, N. Sthor, "Using Mobile Agents for the Management of Telecommunication Networks", Actas da ConfTele99 (2a. Conferência de Telecomunicações), Sesimbra, 1999

[Silva99b] L. Silva, P. Simões, G. Soares, P. Martins, V. Batista, C. Renato, L. Almeida, N. Sthor, "JAMES: A Platform of Mobile Agents for the Management of Telecommunication Networks", Proceedings of IATA'99 (3rd International Workshop on Intelligent Agents for Telecommunication Applications), Estocolmo, Suécia, Agosto de 1999

[Simões99] P. Simões, L. Silva, F. Boavida Fernandes, "Integrating SNMP into a Mobile Agent Infrastructure", Proceedings of DSOM'99 (10th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management), Zurique, Suiça, Outubro de 1999

[Voyager] voyager, http://www.objectspace.com/voyager/

[Znaty97] S. Znaty, J. Martin-Flatin, "Annotated Typology of Distributed Network Management Paradigms", Technical Report SSC/1997/008, École Polytechnique Fédérale de Lausanne, Suiça, 1997

[Zapf99] M. Zapf, K. Herrmann und K. Geihs, "Decentralized SNMP Management with Mobile Agents", in Proceedings of IM'99 (Sixth IFIP/IEEE International Symposium on Integrated Network Management), Boston, EUA, 1999