Descoberta e Monitoramento de Recursos em Redes de Computadores usando Agentes Móveis

Inácio Dutra de Melo Júnior[1], Joaquim Celestino Júnior[1]

[1]Departamento de Estatística e Computação – Universidade Estadual do Ceará (UECE)
Av. Paranjana, 1700 – Campus do Itaperi – 60.740-020 – Fortaleza – CE – Brazil

inaciodutra@secrel.com.br, celestino@uece.br

Resumo. *Com a expansão e proliferação de novas tecnologias de rede e a demanda de novos serviços vem tornando cada vez mais difícil administrar redes em larga escala sem um sistema de gerenciamento inteligente, automatizado. Descobrir e manter um mapeamento da topologia de uma rede são essenciais para se ter uma administração efetiva. Neste artigo apresentamos uma abordagem baseada na utilização do paradigma de agentes móveis que tem como objetivos a identificação dos elementos ativos, o desenho da topologia e o monitoramento dos recursos de redes. Será mostrado também uma análise de desempenho entre o modelo tradicional de gerenciamento de redes com a utilização do modelo centralizado, e com a proposta da introdução de agentes móveis.*

Abstract. *Once new network technologies have been expanding and proliferating as well as new services, the task of administrating huge networks has become harder and harder, and some times almost impossible without using automated and intelligent management systems. To find out as well as keep a network topology mapped are the keys to reach an effective network administration. Throughout this article we will be introducing the utilization of mobile agents paradigm whose goals are to identify active network elements, network topology chart and the resources monitoring. It will be also introduced a performance analysis between the traditional network management model using the centralized model, and applying the proposal of mobile agents.*

1. Introdução

A utilização de redes de computadores tem crescido de forma significativa nos últimos anos. Ao mesmo tempo, a utilização em conjunto das tecnologias de telecomunicações e informática, de natureza e porte diferentes, vem atingindo atualmente um estágio de grande amadurecimento. O surgimento de inúmeras tecnologias e novas aplicações foi crescendo na mesma proporção do aumento de sua utilização. O rápido crescimento, o aumento da competição econômica e a proliferação de novas aplicações têm mudado a característica das redes de computadores e da Internet nos últimos anos, fazendo de tarefas como monitoração e análise, verdadeiros desafios. Este aumento de complexidade interfere no custo de gerenciamento. Segundo [1], este custo pode chegar a 15% do custo total de uma organização com sistemas de informação.

Um dos maiores problemas enfrentados pelas empresas e profissionais de Tecnologia da Informação (TI), é simplesmente não saber onde se encontram os dispositivos e recursos de rede e como eles estão conectados na infra-estrutura

corporativa. Este artigo propõe uma aplicação distribuída onde o paradigma orientado a objetos (OO) será utilizado na integração de agentes móveis, sustentado pela plataforma Aglets, com o protocolo de gerenciamento SNMP, para a descoberta de topologia e monitoramento de recursos baseados em redes TCP/IP. A resolução destes problemas ajuda o administrador a atuar de forma eficiente no que diz respeito a defeitos (*troubleshoot*) de rede reduzindo o tempo de reparo, contribui também para identificar precisamente a configuração dos recursos, além de gerenciar nós ativos/inativos.

Nosso artigo está organizado em seis seções. Na próxima seção apresentaremos a tecnologia de agentes móveis (AM). Na seção 3 introduzimos uma proposição para gerenciamento de redes utilizando agentes móveis. A quarta seção mostrará a metodologia por nós aplicada para descoberta de recursos usando AMs. A seção 5 relata os resultados do desempenho de uma aplicação de gerenciamento centralizada e com o uso de AMs. E, finalmente, a seção 6 apresenta a conclusão deste trabalho.

2. Agentes Móveis

Um agente móvel (AM) é uma entidade de *software* independente que age em nome de seu usuário para realizar uma determinada tarefa, interagindo e cooperando com o seu ambiente e com outros agentes [2]. O AM tem a habilidade e autonomia de selecionar e mudar de ambiente em tempo real, viajando e se executando em redes remotas se necessário. Os AMs são vistos como um subconjunto especial da população geral de agentes de *software*. Semelhante aos seus antecessores (agentes estacionários), os AMs têm um diferencial de possuir a habilidade de aprender, se adaptar, argumentar e se comunicar com usuários e outros componentes de *software* apoiados em um ambiente de rede.

AMs oferecem várias vantagens significantes em cima do modelo cliente/servidor. Muitos protocolos de comunicação requerem várias trocas de mensagens entre o servidor e as aplicações cliente, especialmente quando a segurança é habilitada. AMs usando a aplicação cliente podem empacotar a conversação inteira em uma única transmissão de agente que pode reduzir o tráfego de rede sensivelmente. Também ao lidar com grandes volumes de dados, os AMs podem aumentar a eficiência executando a maioria dos cálculos e seleção de informações localmente onde os dados são armazenados. Finalmente, outra vantagem dos AMs fundamenta-se em sistemas de natureza altamente distribuídas que oferece um aumento significante nos quesitos robustez e tolerância à falhas [3].

Atualmente, grande número de plataformas de AMs têem sido desenvolvidas baseadas em diferentes tecnologias e linguagens de programação, podendo ser executado em diferentes sistemas operacionais. Novas linguagens estão sendo criadas com o fim específico de suportar AMs. Entretanto, este novo desenvolvimento apresenta fortes tendências para linguagens OO como o Java, servindo de fundamento para a construção da maioria das plataformas de AMs. Padrões e interoperabilidade entre plataformas de AMs são propostos pela *Foundation for Intelligent Physical Agents* [4]. Dentre as plataformas existentes, selecionamos a tecnologia *Aglets* [5, 6] como foco de nosso estudo.

Nosso intuito neste artigo é mostrar como um AM baseado na plataforma Aglets, pode ser desenvolvido dentro de uma aplicação que deverá gerenciar os dispositivos de *hardware* (com suporte ao SNMP) inseridos em uma rede de computadores.

3. Gerenciamento de Redes usando Agentes Móveis

Uma das suposições que uma ferramenta de administração de redes deve fazer é que o administrador configure a informação sobre recursos. Embora esta suposição simplifique o desígnio de uma ferramenta de gerenciamento, também impõe responsabilidades em qualquer administrador humano. A motivação primária para um *software* de administração de redes é simplificar e tornar mais eficiente as tarefas de gerenciamento, além de introduzir informações precisas sobre os recursos e a forma como eles estão interligados. A própria ferramenta deve descobrir esta informação, bem como poder descrever a topologia de uma rede baseado no que foi identificado. Além da topologia, a aplicação de gerenciamento deve determinar quais nós encontram-se ativos na rede e quais suas capacidades.

3.1. Descoberta de Topologia e Recursos

A topologia descreve como estão interconectados os diferentes elementos de rede, tais como roteadores, servidores, estações e *switches*. Existem dois tipos diferentes de topologia dentro de uma rede IP:

1. A topologia física descreve como os elementos estão fisicamente conectados.

2. A topologia lógica descreve o caminho que um pacote percorre entre qualquer dois pontos na rede.

A topologia física da rede é relativamente estática e pouco muda. A menos que o administrador realize mudanças na rede, a topologia física não muda. As ações que podem mudar a topologia física incluem a adição ou exclusão de ligações de rede, ou melhorando uma ligação para capacidades mais altas. A maioria destas operações acontece em um período relativamente longo de tempo. Dependendo do domínio de aplicação para o qual se deseja gerenciar, a quantia de informação de topologia requerida é diferente. Para algumas aplicações, bastaria simplesmente saber os nomes das várias máquinas que constituem uma rede. Para outras, precisamos obter a lista de máquinas, como também as características das redes e ligações que as conectam. Existem várias formas de se descobrir a topologia e os recursos ativos de uma rede [7], dentre elas podemos destacar:

1. DNS: é um mecanismo que traduz nomes simbólicos de máquinas para o endereço IP correspondente. A maioria dos servidores DNS suporta uma função que reconhece a zona de transferência entre cliente e servidor. A informação devolvida pela zona de transferência DNS para cada máquina consiste no nome de domínio e o endereço IP de cada interface dos dispositivos registrados no sistema DNS. A informação provida pela zona de transferência não inclui dados sobre as máquinas de outras sub-redes, nem devolve informação de como estas redes estão conectadas. Caso uma máquina esteja em manutenção no momento do monitoramento, isto não afeta o mecanismo de descoberta, pois a zona de transferência proverá a informação para a ferramenta de descoberta de topologia. Em contra-partida não é possível identificar elementos ativos.

2. SNMP/MIB: descrito anteriormente, este protocolo pode ser utilizado para descoberta da topologia física e lógica. Utiliza-se da informação disponível na MIB de cada *host* que possui suporte ao SNMP. As informações definidas nas MIBs contêm muitos dados para se determinar uma avaliação bastante precisa da conectividade física dos vários dispositivos, como também o seu estado atual dentro da rede.

3. ICMP: o método final apresentado por [7] para descoberta de topologia é o uso de programas que identificam os elementos ativos como o *traceroute* ou *ping*. A desvantagem deste mecanismo é que ele gera uma significativa carga na rede. Porém, este método trabalha sem exigir que os agentes SNMP intervenham operacionalmente em quaisquer roteadores.

Na prática, a melhor opção é usar uma combinação das várias técnicas. Combinamos os métodos 2 e 3 descritos por [7]. Com o ICMP identificamos o estado atual do dispositivo e com o SNMP descobrimos a topologia da rede.

3.2. Integração SNMP/Aglets

Nossa proposição de integração seguirá a abordagem proposta em [8], onde criamos uma interface SNMP nos AMs a partir de diferentes objetos Java, habilitando-os a execução de comandos do SNMP. Nossa interface SNMP-Java consiste de bibliotecas de classes modeladas a partir da estrutura de dados ASN.1 usada nos padrões originais do SNMP. Genérica, ela é importada de outro pacote SNMP-Java, o AdventNet [9]. Desta forma, nossa proposta se torna simples, não havendo necessidade de implementação de novas estruturas de dados, deixando os mecanismos de comunicação para o sistema de agentes.

Para interação com agentes SNMP, é necessário incluir capacidades SNMP aos AMs. Desta forma, eles se tornam habilitados a realizarem comandos GET e SET. Isso é possível através da importação, pelo AM Aglets, de classes Java provenientes do AdventNet [8]. Adventnet SNMP é o conjunto de bibliotecas de classe Java para desenvolver aplicações e *applets* de gerenciamento SNMP de redes. Adotamos o AdventNet SNMP API 4 [9], pois suporta o JDK 1.1 e superiores. O pacote pode ser usado para desenvolver aplicações para gerenciamento de agentes SNMPv1, SNMPv2 e SNMPv3 e interagir com AMs. Para a camada de comunicação utilizaremos o *Agent Transfer Protocol* (ATP), que é um protocolo no nível de aplicação modelado em HTTP. ATP é projetado para transmitir um agente independentemente do sistema de agente. O ATP também suporta uma técnica chamada tunelamento HTTP, onde um pedido ATP é encapsulado e enviado como uma requisição HTTP.

O protótipo da arquitetura de integração SMNP-Java-Aglets, onde aplicamos AMs no gerenciamento de redes, é constituída pelos componentes descritos na Figura 1. Os AMs dentro da Figura 1 são implementados utilizando Aglets.

4. Descoberta de Recursos usando AMs

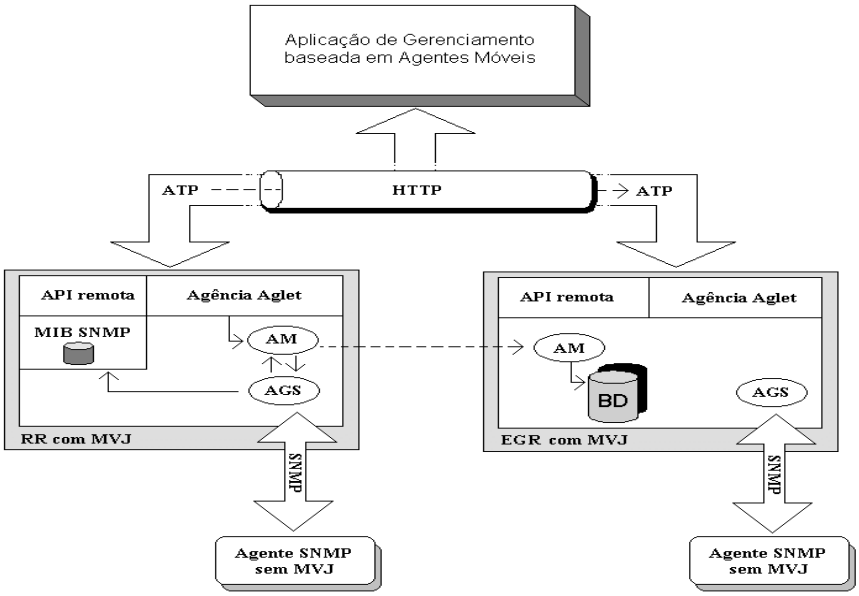
O objetivo básico de uma ferramenta de gerenciamento de redes é traduzir em um alto nível a configuração dos vários dispositivos no sistema. Um dos componentes fundamentais de uma aplicação de administração de redes é o módulo de descoberta de recursos. Um dos itens que o processo de descoberta de recursos deve ter é informação sobre a topologia da rede. Além da topologia, a aplicação de gerenciamento tem que determinar também quais nós encontram-se ativos na rede e quais suas capacidades.

4.1 Metodologia proposta

Antes dos recursos poderem ser administrados, o *software* de gerência de redes tem que prover um mecanismo que automaticamente descubra o conjunto de recursos disponíveis no segmento de rede em questão [10]. Descoberta de recursos é então uma

parte vital para qualquer função de gerenciamento de redes [11]. Nossa metodologia é descrita nas seguintes fases:

1. Descoberta dos recursos que se encontram ativos em uma rede de computadores, dentro de uma faixa de IPs pré-definidas pelo gerente de rede;
2. Envio de agentes móveis para os IPs ativos;
3. Instalação dos agentes móveis nos dispositivos com a finalidade de monitoramento;
4. Geração de um arquivo contendo todos nós e conexões ativas;
5. Mapeamento da topologia da rede.



AM: Agente Móvel **MVJ**: Máquina Virtual Java **AGS**: Agente SNMP
RR: Recurso de Rede **EGR**: Estação de Gerenciamento de Rede

Figura 1. Arquitetura de implementação.

4.2 Descoberta de Elementos Ativos

Desenvolvemos uma classe Java, denominada “Ping”, que será responsável por efetuar um comando *ping* no endereço IP passado pelo construtor. Caso a resposta seja positiva, a aplicação cria um agente e o despacha para o referido IP. Esta classe implementa o método “fazPing”, onde além de efetuar o *ping*, é também realizado um teste para saber se o dispositivo possui suporte ao SNMP. O pseudocódigo desta classe é mostrado logo a seguir através da Figura 2.

4.3 Informação da topologia

Vimos que a descoberta de topologia pode ser obtida de várias formas. O escopo por nós abordado trata das informações dispostas na MIB-II [RFC-1213]. Os grupos de interesse deste artigo são: *system*, *interfaces* e o *ip*. Os objetos destes grupos são de extrema importância para a descoberta de topologia de redes, conforme veremos a seguir:

1. Grupo *system*. Este grupo é composto por sete objetos chamados, *sysDescr*, *sysObjectID*, *sysUpTime*, *sysContact*, *sysName*, *sysLocation* e *sysServices*. O conteúdo destes objetos é auto-explicativo. O valor do *sysServices* pode ser usado para determinar os tipos de serviços que um nó suporta. Este valor pode ser interpretado como um código de 7 bits, onde cada bit representa uma camada na arquitetura OSI. O último bit significativo corresponde à camada física e o mais significativo representa a camada de aplicação.

2. Grupo *interfaces*. O grupo *interfaces* contém o objeto *ifNumber*, que indica o total de interfaces no nó gerenciado. O outro objeto deste grupo é o *ifTable*. O conteúdo de *ifTable* é composto por uma linha de cada interface, onde temos as informações de tipo, velocidade e status operacional da interface.

3. Grupo *ip*. O grupo *ip* contém três tabelas que são utilizadas para geração de mapas de topologia, chamadas nomeadamente: *ipAddrTable*, *ipRouteTable* e *ipNetToMediaTable*. A tabela *ipAddrTable* contém o endereço IP da interface gerenciada. As colunas *ipRouteDest* e *ipRouteMask* da tabela *ipAddrTable* e a coluna *ipRouteType* da tabela *ipRouteTable*, determinam as rotas para subredes no nó gerenciado. A coluna *ipRouteNextHop* pode ser usada para armazenar a informação do próximo roteador da subrede do nó gerenciado. O endereço IP de alguns dispositivos em subredes que estão diretamente conectados ao nó gerenciado pode ser encontrado na tabela *ipNetToMediaTable*.

A partir destes grupos, várias técnicas e algoritmos para descoberta de topologia em redes IP foram propostas. Nosso artigo segue os padrões adotados por [11].

```
class Ping implements Runnable {     /** IP do elemento ao qual será feita à  */
     private String ipAtual = null;     /**  tentativa de conexão */
     /** @param ipAtual IP do elemento que deve ser verificado se está ativo.  */
     public Ping(String ipAtual) {
              this.ipAtual = ipAtual;    }
     public void run() {                    /** Método para executar o ping */
   // obter um objeto responsável pela conexão
        Conexao con = new Conexao(ipAtual, Conexao.PORTA);
   // realiza três tentativas de ping, atualizando as coleções PING, SNMP e AGLETS
        con.fazPing(ipAtual, Conexao.DEFAULT_TIMEOUT,
                  3, listOnlyPing, listSNMP, listAGLETS);
        con.close();     // fecha a conexão
       }
   }
}
```

Figura 2. Pseudocódigo para descoberta de recursos em redes.

4.4. Cenário da proposição

Os passos de nossa proposição são descritos resumidamente da seguinte maneira (um fluxograma dos passos da aplicação é mostrado na Figura 3):

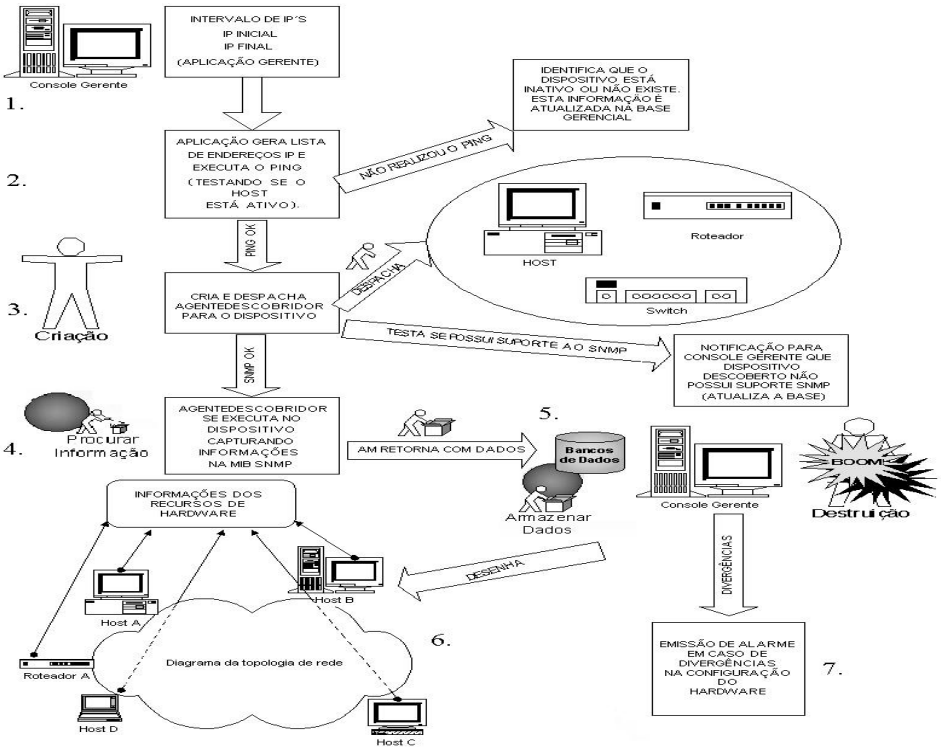


Figura 3. Fluxograma da proposição.

1. A aplicação (gerente) solicita uma faixa de endereços IP;

2. O programa tenta realizar um *ping* com o endereço IP montado, caso a resposta seja positiva, é criado e despachado um agente móvel (*AgenteDescobridor*) ao destino;

3. O *AgenteDescobridor* se instala no dispositivo e captura informações dispostas na MIB SNMP. Esta informação é processada localmente;

4. O *AgenteDescobridor* finaliza o processo de captura de informações na entidade gerenciada e retorna para a console de gerenciamento trazendo as informações de interfaces físicas e configurações do dispositivo (tamanhos de memória física, memória volátil e processador), atualizando assim a base gerente e encerrando seu ciclo de vida;

5. A aplicação gerente recebe todas as informações dos AMs, cataloga-as e realiza o processamento;

6. Um mapa de conectividade da rede é exibido na console de gerenciamento;

7. Eventualmente a aplicação poderá emitir alarmes ao gerente em casos de mudanças na configuração dos dispositivos (mencionados no item 4).

A implementação da aplicação descrita neste artigo faz parte de uma dissertação de mestrado do autor.

4.5 Classes da Aplicação

Na Figura 4, apresentamos o diagrama UML de classes da aplicação na fase de envio dos AMs, bem como as classes de exibição do mapa da topologia. Foram implementadas funcionalidades para conseguir um controle efetivo sobre as propriedades dos grafos a serem desenhados.

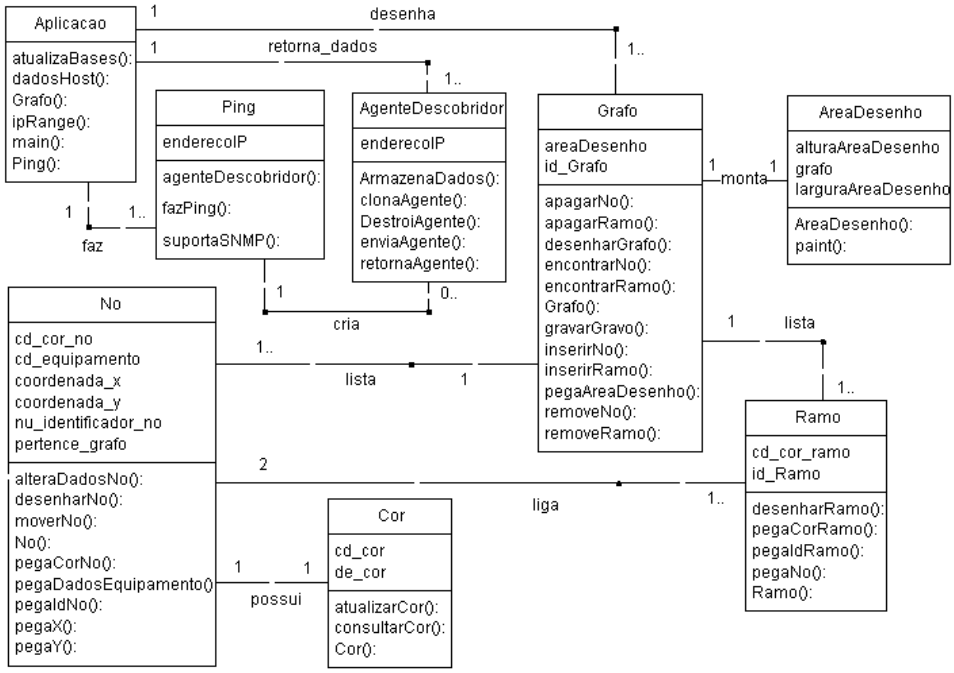


Figura 4. Diagrama UML do envio dos AMs e desenho da topologia de rede.

No momento em que tratamos o controle de recursos de *hardware*, passamos à implementação do inventariado propriamente dita. Esta etapa adiciona a funcionalidade de gerenciamento de patrimônio, onde se torna possível automatizar a utilização de inventário manual, reduzindo significativamente o custo operacional. A aplicação mantém uma base de dados de inventário de *hardware* que fornece informações, tais como identidade do equipamento, quantidade de memória, etc. Na Figura 5, apresentamos o diagrama UML de classes da aplicação na fase de armazenamento de informações dos recursos de *hardware*.

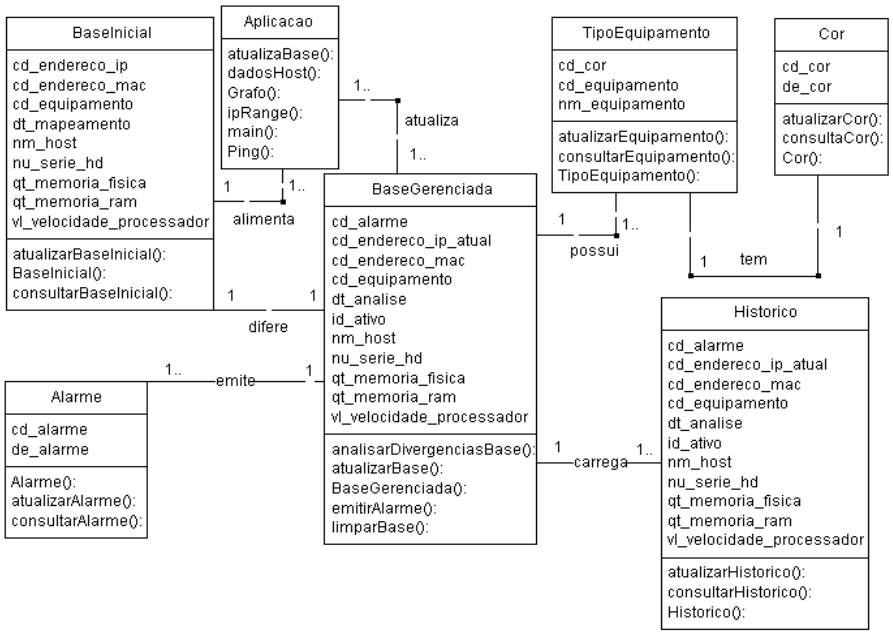


Figura 5. Diagrama UML do controle de inventariado de *hardware*.

4.6 Interface da aplicação

Nesta subseção mostraremos o resultado da interface final da implementação de nossa aplicação. Os parâmetros de números IP inicial e final devem ser informados na opção *Processamento* do módulo *Principal.* As Figuras 6 e 7 mostram como isto se aplica.

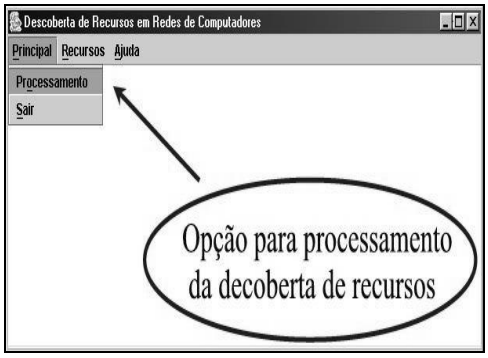


Figura 6. Opção processamento.

Figura 7. Intervalos de IPs.

Neste momento, resumidamente, o programa executa os seguintes procedimentos:

1. Executa Pings, no caso da descoberta de todos os elementos de rede ativos (com e/ou sem suporte a SNMP/Aglets).

2. Envia os Aglets, no caso da descoberta usando agentes (apenas para máquinas com suporte a SNMP/Aglets).

3. O agente realiza Gets no dispositivo.

4. Retorno do agente e atualização da base gerencial.

No módulo de *Recursos* da aplicação, as opções contendo os resultados de processamento podem ser consultadas conforme mostrado na Figura 8. O resultado das informações processadas e exibidos pelo módulo de *Recursos* (Figura 9), são assim resumidos:

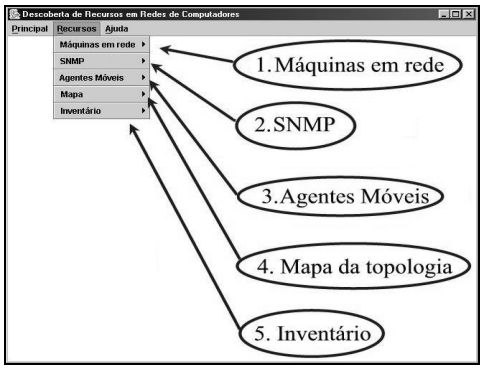
1. Máquinas em rede – Mostra o tempo que foi gasto para descobrir todas as máquinas que estão ativas na rede, independente do suporte a Aglets/SNMP.

2. SNMP – Informa o tempo que a aplicação levou para descobrir dispositivos que possuem suporte ao SNMP, testando a porta 161, além de exibir o tempo que foi gasto para reunir as informações da topologia a partir de comandos na MIB SNMP.

3. Agentes Móveis – Informa o tempo que a aplicação levou para descobrir dispositivos que possuem suporte aos Aglets, testando a porta 4434, além de exibir o tempo que foi gasto para reunir as informações da topologia a partir da interação do AM com a MIB SNMP.

4. Mapa da Topologia – Exibe um desenho da topologia a partir das informações coletadas pelos AMs.

5. Inventário – Informa possíveis divergências nos dados de *hardware* dispostos na base em detrimento com as novas informações de processamento.



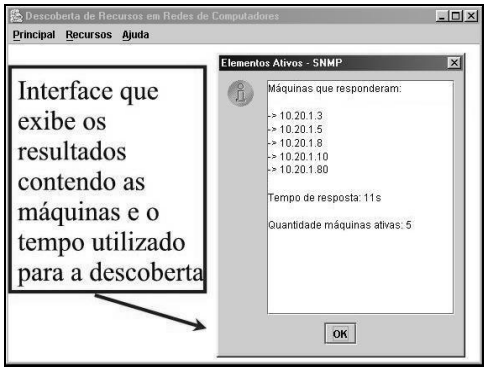


Figura 8. Módulo de Recursos.

Figura 9. Tempo de descoberta.

O resultado final de nossa aplicação é o desenho da topologia física da rede, através de um algoritmo desenvolvido em JAVA, onde podemos observar o exemplo de uma interface exibida através da Figura 10.

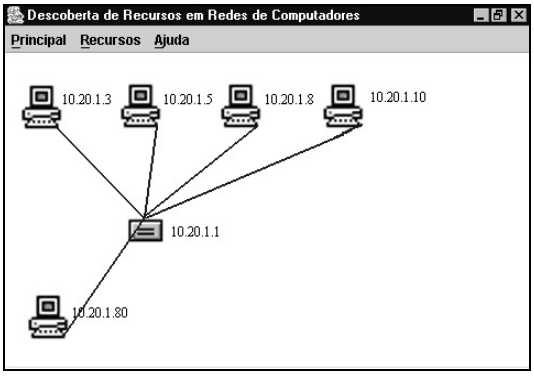


Figura 10. Desenho da topologia.

5. Resultados experimentais

Os testes experimentais foram realizados em uma rede contendo quarenta máquinas WinNT. Nós configuramos cinco dispositivos com o sistema de agentes aglets, o *Tahiti*, bem como habilitamos o serviço SNMP. Estes dispositivos estão distribuídos no mesmo segmento de rede 10.20.1.0, onde a console gerente (10.20.1.80) encontra-se em uma *switche* sem suporte ao SNMP e as outras quatro máquinas (10.20.1.3, 10.20.1.5, 10.20.1.8, 10.20.1.10) em outra *switche* com suporte ao SNMP, conforme podemos observar na Figura 11.

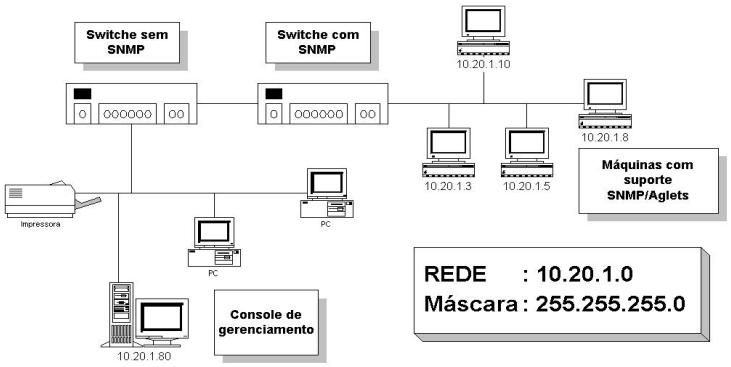


Figura 11. Ambiente de rede utilizado nos testes experimentais.

Os resultados de desempenho foram divididos em duas etapas que são mostrados nas Figuras 12 e 13, respectivamente. Estes resultados foram executados várias vezes e utilizamos a média aritmética de três observações. A primeira etapa (Figura 12) exibe um gráfico mostrando o tempo utilizado na descoberta dos elementos ativos na rede. Observamos uma diferença mínima de tempo para descobrir as cinco máquinas com suporte ao SNMP e com suporte a Aglets. Como o algoritmo de implementação é o mesmo, diferenciando apenas a porta que deve ser checada (161 ou 4434), concluímos que os possíveis fatores de influência foram: latência da rede, *jitter* e uma eventual ocupação do processador. Sendo assim, não consideramos como relevante a diferença de apenas 1 segundo, pois ainda não implementamos o uso dos agentes neste processamento. Com relação a descoberta de todas as máquinas disponíveis em rede, verificamos que o programa utilizou um tempo bem menor e descobriu 36 (trinta e seis) dispositivos. Neste caso, nosso programa demonstrou um ótimo desempenho quando se quer apenas descobrir elementos ativos em redes. Em termos matemáticos (aplicando um simples cálculo em regra de três), verificamos uma diferença de 35,90 %.

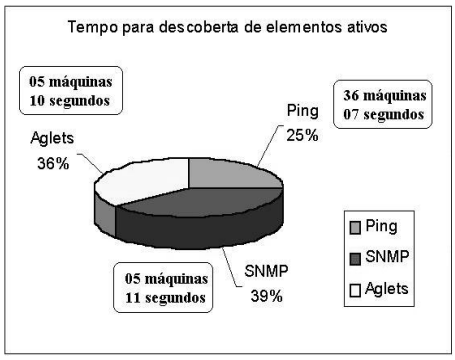


Figura 12. Descoberta elementos ativos.

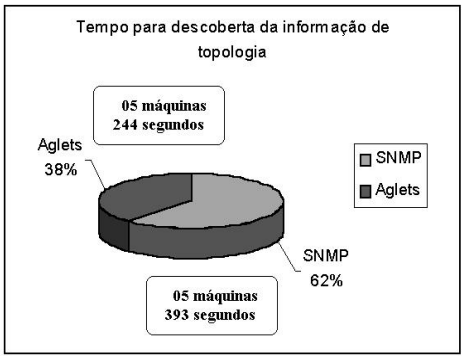


Figura 13. Informação de topologia.

Analisemos agora o gráfico da segunda etapa (Figura 13), onde observamos o tempo empregado para reunir as informações necessárias na montagem do desenho da topologia e armazenamento de dados das cinco máquinas. Nesta fase empregamos o uso dos agentes móveis e do SNMP, e ao calcularmos (aplicando regra de três), identificamos um ganho significativo no desempenho, algo em torno de 27,58 % em favor dos AMs. Com estes testes conseguimos demonstrar que o emprego de AMs pode ser muito interessante no gerenciamento de redes. Apesar de simularmos em uma pequena rede local, isto pode se tornar bastante expressivo em redes de larga escala. Supondo que uma rede leve 10 horas para ser mapeada usando o SNMP, teremos então uma economia de tempo na ordem de quase 3 horas com o uso de AMs.

6. Conclusão

Com o crescimento das redes de computadores, a descoberta de recursos torna-se essencial para introduzir automação onde o gerenciamento manual ainda é largamente utilizado. A solução aqui apresentada pretende introduzir mecanismos facilitadores na investigação das características existentes dos recursos descobertos, tornando o gerenciamento de redes numa tarefa menos complexa.

Este artigo explora o paradigma de agentes móveis no campo do gerenciamento de redes. Nós estamos desenvolvendo uma aplicação baseada nas plataformas Java-Aglets-SNMP, onde temos algumas metas importantes em andamento: integração SNMP-Aglets, desenho da topologia e o controle efetivo do inventariado de *hardware*.

Os resultados preliminares no desenvolvimento desta ferramenta se mostram bastante promissores. As imagens geradas pela ferramenta apresentam uma maior facilidade de visualização da rede, porém não permitem a edição por parte do usuário. Tal funcionalidade deve ser acrescentada em futuras versões. O problema de desenhar a topologia de uma rede com o mínimo de cruzamentos entre as linhas que a compõem é NP-Difícil. Na medida em que o número de redes envolvidas cresce a visualização torna-se mais difícil. Seria interessante adotar apresentações tridimensionais, topologias hierárquicas e desenhos particionados.

Referências

[1] STALLINGS, WILLIAM. (1999) “SNMP, SNMPv2, SNMPv3 and, RMON1 and 2”, Addison-Wesley, Third Edition, September.

[2] KARMOUCH, AHMED and PHAM, VU ANH. (1998) “Mobile Software Agents: An Overview”, *IEEE Communications Magazine*, July.

[3] TOLMAN CAM, (2003) “A Fault-Tolerant Home-Based Naming Service for Mobile Agents”, Rensselaer Polytechnic Institute, Troy, New York, April.

[4] Foundation for Intelligent Physical Agents – FIPA. URL = http://www.fipa.org.

[5] AGLETS WORKBENCH – IBM/Japan. http://www.trl.ibm.co.jp/aglets/.

[6] OSHIMA, MITSURU; LANGE, DANNY B. (1998) “Programming and Deploying Java Mobile Agents with Aglets”, Addison-Wesley, First Edition: November.

[7] VERMA, DINESH C. (2000) “Policy-Based Networking: Architecture and Algorithms”, New Riders, First Edition: November.

[8] CARDOSO, A. (2002) “Integrando Agentes Móveis com Sistemas Legados para Gerenciamento de Redes ATM Heterogêneas”, Dissertação Mestrado, Universidade Federal de Campina Grande.Paraíba, Agosto.

[9] AdventNet v2c Release 3.3 URL= http://www.adventnet.com.

[10] DUNNE, CAMERON R. (2001) “Using Mobile Agents for Network Resource Discovery in Peer-to-Peer Networks”, Paper of Dublin City of University.

[11] HWA-CHUN LIN, SHOU-CHUAN LAIN, et al. (1998) “An Algorithm for Automatic Topology Discovery of IP Networks”, Paper of National Tsing Hua University, Taiwan, IEEE, Department of Computer Science.