UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Cleverson Bússolo Klettenberg

Um Modelo de Sistema Multiagente de Auxílio à Administração de Redes

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação.

Dr. João Bosco da Mota Alves

Florianópolis, 07/2001

Um Modelo de Sistema Multiagente de Auxílio à Administração de Redes

Cleverson Bússolo Klettenberg

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação, àrea de concentração, Sistemas de Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

Prof. Fernando Á. O. Gauthier, Dr.
Coordenador do PGCC

Banca Examinadora

Prof. João Bosco da Mota Alves, Dr.
Orientador

Prof. João Cândido Lima Dovicchi, Dr.

Prof. Elizabeth Specialski, Drª.

Epígrafe

"Fazer melhor neste momento, o coloca na melhor situação para o próximo momento"

(Ranulfo Alves Pereira)

Agradecimentos

Atingir qualquer meta exige, além de dedicação e perseverança, apoio para ultrapassar barreiras. Nesta conduta, encontrei muitos colaboradores e incentivadores que acabaram permitindo de uma forma ou de outra o complemento deste trabalho. Para tanto, deixo registrados meus sinceros agradecimentos:

Ao Dr. João Bosco da Mota Alves que dignamente soube desempenhar com dinamismo e sempre bem humorado seu papel de orientador contribuindo para a realização deste trabalho.

A UNIVALE – União das Escolas Superiores do Vale do Ivaí que além de me encaminhar a dar os primeiros passos rumo a minha profissionalização, disponibilizou ajuda de custos e permitiu a realização deste trabalho em suas instalações.

A minha mulher e filhos que souberam compreender a minha ausência durante a realização de meu trabalho.

A Deus, pela permissão e saúde de concluir com satisfação mais esta etapa de minha vida.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE SIGLAS

API	Application Programming Interface
BNF	Backus-Naur-Form
DARPA	Defence Advanced Research Projects Agency
FAT	File Allocation Table
ID	Identificação
IHL	Internet Header Length
KB	Kbytes
MB	Megabits
MHZ	Megahertz
IP	Internet Protocol
IPng	Internet Protocol Next Generation
IPv6	Internet Protocol versão 6
PC	Personal Computer
SQL	Structured Query Language
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
DOS	Denied Off Service

RESUMO

Fundamentado na grande quantidade de recursos computacionais existentes no mercado e devido ao avanço da informática e dos meios de comunicação, funcionários, alunos e outros usuários utilizam-se de programas e acessam páginas da Internet que contêm assuntos não apropriados para um ambiente profissional ou educacional. Para solucionar o problema citado, este trabalho propõe o desenvolvimento de uma aplicação multiagente que auxilie na administração destas redes através da busca heurística de programas e palavras que se relacionam com as necessidades de restrições da organização.

PALAVRAS-CHAVE – administração de redes, restrição de aplicações, multiagentes, heurística, cliente/servidor.

ABSTRACT

Based on the great amount of computational features that exist on the market and are the advance of computer science and the media, employees, students and other users use programs and have access to page of the Internet that contain inappropriate subjects for a professional or educational environment. To solve the problem, this work will consider the development of a multiagent application which assists the management of networks through the heuristic match of programs and words related to the restrictions imposed by the organization.

KEYWORDS - management of networks, restriction of applications, multiagents, heuristic, client/server.

1. INTRODUÇÃO

1.1. Considerações Iniciais

Milhares de instituições, negócios e indivíduos estão diariamente conectados a Internet. Esse fenômeno – que recebeu vários nomes diferentes – é comumente denominado *Explosão da Internet*. Esta explosão alterou drasticamente a composição da Internet. Além de educação e assuntos militares, como havia nas décadas de 1960 e 1970, agora temos também comércio e entretenimento. Com isso, grande parte das empresas que possuem um departamento de informática, possuem também, uma elevada quantidade de recursos computacionais dividida em seus setores, principalmente em empresas que atuam em áreas educacionais, onde os alunos dispõem de laboratórios de informática para fazer seus trabalhos educacionais. Estes avanços que trazem grandes facilidades à educação, estratégias organizacionais, comunicação de informações etc., trazem também o problema da má utilização destes valiosos recursos dispostos pelas organizações para fins, que não são os previstos por elas quando estes foram adquiridos. Outro problema é a falta de experiência ou o vandalismo dos usuários, que alteram a configuração, instalam programas que não pertencem a seu âmbito profissional e prejudicam a imagem da empresa ou setor onde trabalha, ocasionando gastos com pessoal qualificado para a reconfiguração, limpeza do disco e manutenção dos equipamentos em geral.

1.2. Motivações

As motivações para definição do tema deste trabalho foram elaboradas mediante a falta de recursos de software para auxílio à administração de redes e devido à quantidade de páginas e programas disponíveis na Internet que cresce constantemente, tornando-se assim, cada vez mais difícil o cadastro e controle destas páginas e programas que deverão ser restringidos. Inspirado nesta dificuldade tem-se a idéia de não mais

cadastrar todas as restrições, mas sim, uma amostra delas e através destas, detectar as demais irregularidades.

1.3. Objetivos Gerais

Este trabalho tem por objetivo expor uma forma de amenizar e até extinguir os transtornos causados por uso impróprios de recursos computacionais através do desenvolvimento de uma aplicação que fique observando tudo o que está sendo executado nos microcomputadores espalhados nas redes existentes em organizações e aplique restrições que vão desde uma simples mensagem de aviso até o desligamento do computador quando uma ocorrência for encontrada, auxiliando assim na administração destas redes.

Outro objetivo é estabelecer a comunicação e o controle remoto entre os setores das empresas.

1.4. Objetivos Específicos

O desenvolvimento de um sistema multiagente para auxílio à administração de redes é relevante por contribuir para as tecnologias existentes através dos seguintes fatores:

- Facilitar o cadastro de programas e páginas de Internet a serem restringidas através do uso da busca heurística destas aplicações;
- Possibilitar ao administrador ficar sabendo instantaneamente de uma tentativa de acesso a algo indesejável;
- Inibir o usuário de fazer uma tentativa de acesso a uma aplicação imprópria pelo fato de saber que pode ser abordado pelo administrador;

- Permitir comunicação entre os agentes e desta forma entre os diversos setores da empresa;
- Controlar qualquer tipo de aplicação seja uma página da Internet, um jogo ou um programa de configuração;
- Possibilitar controle remoto de alguns dispositivos e aplicações;
- Transmitir arquivos entre os agentes.

1.5. Tecnologias e Métodos Abrangentes

Os métodos empregados para definição do modelo proposto seriam, o uso de algoritmos de busca heurística para detecção de irregularidades e o uso de *soquetes*[1] para estabelecer a interface da aplicação com o protocolo *IP*[2] usado para conexão entre os agentes cliente e servidor.

Este sistema será tratado sobre o ponto de vista da utilização por uma instituição educacional, embora possa ser utilizado, como dito anteriormente, por qualquer empresa que tenha uma rede e que necessite de delimitar o uso de programas em suas estações de trabalho.

Dentre os trabalhos já pesquisados nesta mesma área temos o *Firewall*[3], que permite bloquear o acesso a Internet de alguns tipos de páginas, embora não permita controlar aplicações que são executadas localmente nas máquinas da rede como jogos, configurações dos computadores etc. Outro trabalho pesquisado foi sobre os *gateways*[4] de aplicativo, sua vantagem é que eles impedem o tunelamento de pacotes de IP em sua rede.

---

[1] Interface de comunicação bidirecional destinado à entrada e saída de dados, que permite que um aplicativo acesse os protocolos TCP/IP.
[2] Responsável pelo endereçamento e pelo envio de datagramas em uma inter-rede.
[3] Faz o rastreamento de protocolos entre a comunicação entre redes.
[4] Computador que conecta várias redes TCP/IP para roteamento ou entrega de pacotes IP entre elas. O gateway é utilizado de forma intercambiável com o IP.

Para o desenvolvimento destes mecanismos é importante identificar tecnologias abrangentes, que permitam uma ampla utilização da solução. A seguir, uma menção das fronteiras de pesquisas utilizadas para este trabalho:

- **Mecanismos de comunicação**: o sistema será implementado em dois agentes que agirão em máquinas distintas, o Agente Espião, localizado nos terminais de trabalho e o Agente Gerente da Rede, localizado no computador do administrador da rede, comunicando-se entre si através de uma rede de endereçamento IP. A interface de conexão utiliza o paradigma cliente/servidor;

- **Técnicas de busca heurística**: não existe como definir com certeza quando uma página de internet ou um programa é impróprio ou não, desta forma, quando lidamos com problemas reais, geralmente é difícil medir precisamente o valor de uma determinada solução. Portanto, faz-se necessário construir uma estrutura de controle que não mais garanta encontrar a melhor resposta, mas que quase sempre encontre uma resposta muito boa. A heurística é uma técnica que melhora a eficiência de um processo de busca, possivelmente sacrificando pretensões de completeza;

- **APIs, Applications Programming Interface (Interface de programação de aplicativos)**: Para obter os aplicativos que estão executando no sistema operacional Windows, precisamos fazer chamadas as APIs. Através destes recursos conseguimos também obter o *handle*[1] que permite enviar ordens ao sistema operacional sobre a aplicação;

- **Agentes de raciocínios distribuídos:** o modelo apresentado neste trabalho é multiagente por ser constituído de dois agentes de raciocínios distribuídos. Definimos um sistema de raciocínio distribuído como aquele que é composto por um conjunto de módulos separados já que cada módulo assume o papel de

Comentário: Espécie de có
do processo da aplicação

[1] Código do processo do aplicativo junto ao sistema operacional.

uma entidade de solução de problemas e por um conjunto de caminhos de comunicação entre eles;

- **A Metalinguagem BNF – Backus-Naur-Form:** na comunicação entre os agentes do sistema foram utilizadas técnicas de Metalinguagem para definição da sintaxe de troca de mensagens para reconhecimento de solicitação de comando do agente remoto.

Os trabalhos pesquisados estão entre 1981 e 2001.

1.6. Estrutura do Trabalho

Neste Capítulo 1 estão mencionadas as considerações iniciais do trabalho, o que motivou a definição do tema do trabalho, seus objetivos gerais e específicos e as tecnologias e métodos utilizados.

Este trabalho foi estruturado de forma a conter no Capítulo 2 a fundamentação teórica necessária para o desenvolvimento do modelo de sistema proposto, detalhando as áreas de tecnologias utilizadas para atingir o objetivo do trabalho e a eficácia com relação ao sistema desenvolvido.

O Capítulo 3 descreve com detalhes, todas as partes integrantes do modelo proposto e a metodologia utilizada para o desenvolvimento do sistema com especificação para cada agente de seus objetivos, interfaces e os métodos utilizados na construção.

Todos os métodos de construção foram avaliados no Capítulo 4, onde podemos comparar os resultados obtidos com o modelo proposto. Os resultados descritos neste capítulo são referentes à perda de desempenho dos equipamentos e programas da rede, à redução de ocorrências de irregularidades, e também, às falhas detectadas. Os testes do modelo referenciados neste capítulo proporcionaram o surgimento de novas idéias que se pretende implementar no futuro com objetivo de enriquecer a precisão do sistema.

Por último, o Capítulo 5 apresenta as conclusões em torno do modelo proposto que vão refletir se o rumo tomado deste trabalho foi de encontro ao alcance dos objetivos e o que será necessário para aperfeiçoar o trabalho desenvolvido através de implementações futuras.

2. FUNDAMENTAÇÃO TEÓRICA

2.1. Mecanismos de Comunicação entre os Agentes

Para desenvolvimento deste sistema, serão utilizados duas tecnologias que têm despontado no mercado hoje por sua padronização, disponibilidades e baixo custo: os protocolos TCP/IP e *Ethernet*[1]. *"Pela capacidade de juntas proverem o meio de interligação entre sistemas de computação, estas tecnologias têm sido aplicadas em muitas áreas científicas como centros de pesquisa, universidades e indústrias, e seu vasto uso hoje é uma realidade nestes meios"* [ROSA, 1998]. Uma das características que tem ajudado na vasta aplicação destas tecnologias é a possibilidade de diferentes fornecedores e implementadores proverem soluções compatíveis, com conseqüente redução de custos.

2.1.1. Paradigma Cliente/Servidor

O paradigma cliente/servidor é muito utilizado para aplicação de banco de dados SQL, mas também pode ser utilizada para transmissão de mensagens, como é o caso deste trabalho. Os processos clientes enviam pedidos a um servidor, que responde com os resultados para esses pedidos. Como o nome indica, os processos servidores oferecem serviços aos clientes, normalmente por meio de processamento específico que só eles podem fazer. Segundo RENAUD, 1994, *"A interação entre os processos cliente e servidor é uma troca cooperativa, transacional, em que o cliente, é ativo e o servidor é reativo"*, mas essa situação poderá ter seu papel trocado a medida que o servidor toma uma atitude ou recebe uma solicitação do usuário de enviar mensagens aos clientes e desta forma, tornando-os reativos.

---

[1] Protótipo para redes baseadas no controle de acesso ao meio físico desenvolvida pela Xerox Corporation.

Em um verdadeiro ambiente cliente/servidor, os processos cliente e servidor são indiferentes a se executam na mesma máquina ou em máquinas diferentes. Alguns protocolos ou redes podem não aceitar o processo do cliente e do servidor no mesmo sistema. Entretanto, essa é uma restrição de um protocolo em particular – e não uma característica do paradigma cliente/servidor. Por uma perspectiva teórica, o tamanho da máquina não importa também. Não há motivo para o processo servidor não poder executar em um desktop PC enquanto o processo cliente executa em uma máquina maior. A simplicidade da teoria de cliente/servidor a torna muito poderosa e é este o principal motivo de sua utilização neste trabalho.

O usuário do sistema interage com um cliente, que por sua vez emite pedidos e recebe resultados do servidor, conforme ilustrado na **figura 1** a seguir.

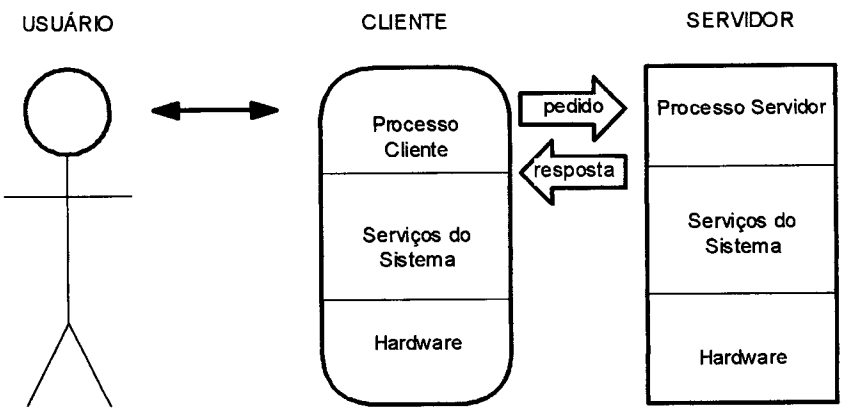

Figura 1 – Paradigma cliente/servidor

2.1.1.1. Atributos do Cliente

O processo de cliente é ativo, emitindo pedidos ao servidor. Ele normalmente é dedicado à sessão do usuário, começando e terminando com a sessão. Um cliente pode

interagir com um único servidor ou com vários servidores para realizar seu trabalho, entretanto, pelo menos um processo servidor é sempre necessário.

Em nível de aplicação, o cliente é responsável por manter e processar o diálogo inteiro com o usuário. Isso normalmente inclui a realização do seguinte:

- Manipulação de tela
- Interpretação de menus ou comandos
- Entrada e validação de dados
- Processamento de ajuda
- Recuperação de erro

2.1.1.2. Atributos do Servidor

O processo servidor é reativo, disparado pela chagada de pedidos dos seus clientes. Um processo servidor geralmente está ativo o tempo todo, oferecendo serviços a muitos clientes.

Um servidor realiza toda lógica exigida para processar uma transação e não interage com outros servidores. Se um cliente usa vários servidores, é responsabilidade do cliente ativá-los.

2.1.2. Soquetes

"Um soquete é um endereço formado pela concatenação do endereço IP com o número de porta" [CASAD, 1999]. Exemplo: o endereço de soquete 192.168.0.10:21 refere-se à porta 21 no computador com o endereço IP 192.168.0.10.

Veja agora na **figura 2** um exemplo de como um computador acessa uma aplicação na máquina de destino através de um soquete:

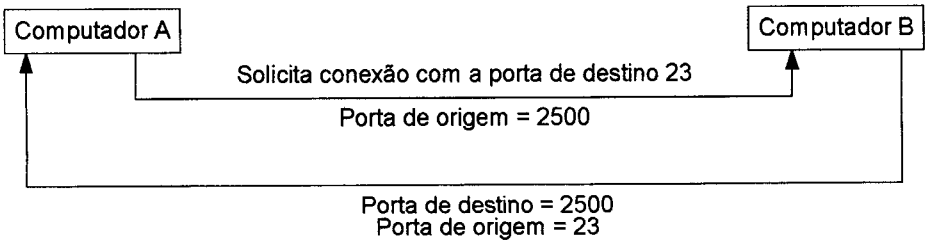

Figura 2 – Conexão através de soquetes

a) O Computador A inicia uma conexão com uma aplicação no Computador B por meio de uma porta conhecida. Combinada com o endereço IP, a porta torna-se o endereço de soquete de destino para o Computador A. O pedido inclui um campo de dados dizendo ao Computador B qual número de soquete deve usar ao enviar informações do volta ao Computador A. Esse é o endereço de soquete de origem do Computador A;

b) O Computador B recebe o pedido do Computador A por meio de uma porta conhecida e direciona uma resposta para o soquete listado como endereço de origem do Computador A. Esse soquete torna-se o endereço de destino para as mensagens enviadas da aplicação no Computador A.

2.1.3. Arquitetura TCP/IP

O desenvolvimento da arquitetura TCP/IP foi patrocinado pela Agência de Projetos e Pesquisas Avançadas de Defesa (DARPA – Defence Advanced Research Projects Agency) baseando-se principalmente em: um serviço de transporte orientado à conexão, fornecido pelo Transmission Control Protocol (TCP), e em um serviço de rede não-orientado à conexão (datagrama não confiável), fornecido pelo Internet Protocol (IP) [POSTEL, 1981].

"O software TCP/IP normalmente reside no sistema operacional" [COMER, 1999], onde pode ser compartilhado por todos os programas aplicativos executados no computador.

Para Interligar duas redes distintas é necessário conectar uma máquina a ambas as redes. Tal máquina fica responsável pela tarefa de transferir mensagens de uma rede para outra. Uma máquina que conecta duas ou mais redes é denominada *internet gateway* ou *internet router. "Para ser capaz de rotear corretamente as mensagens, os gateways precisam conhecer a topologia da inter-rede, ou seja, precisam saber como as diversas redes estão interconectadas. Já os usuários Vêem a inter-rede como uma rede virtual única à qual todas as máquinas estão conectadas, não importando a forma física de interconexão"* [SOARES,1995].

"A arquitetura preconiza a Interoperabilidade de redes de forma transparente ao usuário, que não precisa se preocupar com as especificações do nível físico visto a arquitetura mascarar este pormenor".[GASPARINI, 1993].

2.1.3.1. Camadas da Arquitetura TCP/IP

A arquitetura Internet TCP/IP é organizada em quatro camadas conceituais construídas sobre uma quinta camada que não faz parte do modelo, a camada intra-rede. *"O*

modelo TCP/IP não tem as camadas de sessão e de apresentação. Como não foi percebida qualquer necessidade, elas não foram incluídas." [TANENBAUM, 1997]. A **figura 3** mostra as camadas e o tipo de dados que é passado entre essas camadas:

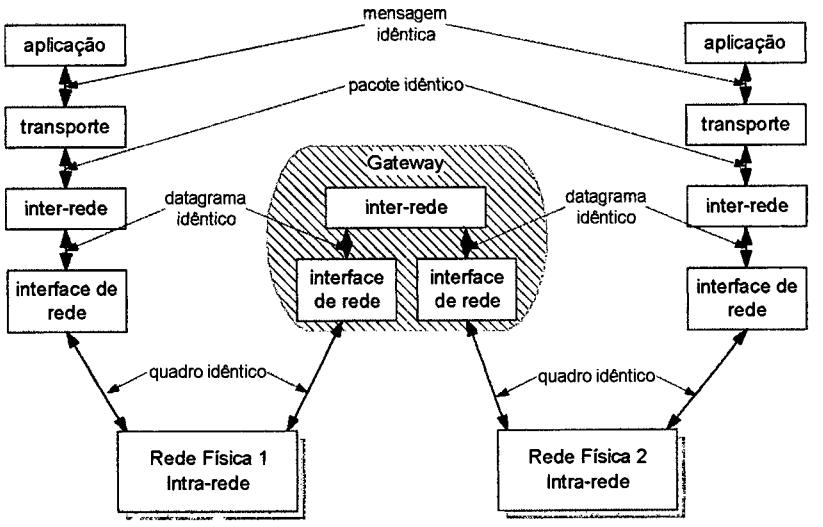

Figura 3 – Camadas conceituais da arquitetura TCP/IP

Nível de aplicação: os usuários usam programas de aplicação para acessar os serviços disponíveis na inter-rede. As aplicações interagem com o nível de transporte para enviar e receber dados.

Nível de transporte: tem a função de permitir a comunicação fim-a-fim entre aplicações. Os seguintes serviços são fornecidos: controle de erro, controle de fluxo, seqüenciação e multiplexação do acesso ao nível inter-rede.

Nível inter-rede: é o responsável pela transferência de dados através da inter-rede, desde a máquina de origem até a máquina de destino. Esse nível recebe pedidos do nível de transporte para transmitir pacotes que, ao solicitar a transmissão, informa o

endereço da máquina onde o pacote deverá ser entregue. O pacote é encapsulado em um datagrama IP, e o algoritmo de roteamento é executado para determinar se o datagrama pode ser entregue diretamente, ou se deve ser repassado para um gateway. Com base no resultado da avaliação do algoritmo de roteamento o datagrama é passado para a interface de rede apropriada para então ser transmitido. O nível inter-rede também processa pacotes recebidos das interfaces de rede. Nesse caso, o algoritmo de roteamento é utilizado para decidir se o datagrama deve ser passado para o nível de transporte local, ou se deve ser passado adiante através de uma das interfaces de rede.

Nível de interface de rede: tem a função de compatibilizar uma interface de rede qualquer com o protocolo IP, pois esta arquitetura não faz restrição a tipos de redes. Para realizar esta tarefa, nesse nível, os endereços IP, que são endereços lógicos, são traduzidos para os endereços físicos dos hosts ou gateways conectados à rede.

2.1.3.2. Datagrama

A camada de transporte recebe os dados para transmitir e divide estes dados em pacotes de até 64 Kbytes. *"Cada datagrama é transmitido pela Internet, sendo possivelmente fragmentado em unidades menores durante o percurso até o destino"* [TANEMBAUM, 1997]. O formato de um datagrama IP é apresentado na **figura 4**. Os números na parte superior da figura representam a posição dos *bits* em cada palavra de 32 *bits*. A seguir são detalhadas as funções dos vários campos, assim como os valores que eles podem assumir.

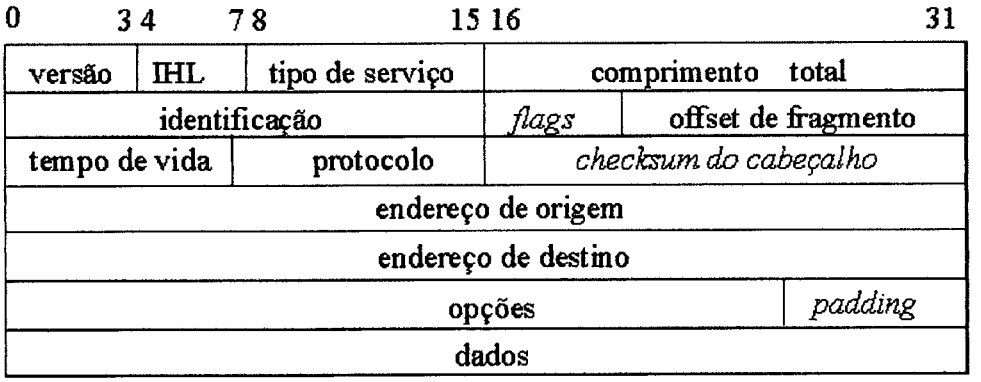

Figura 4 – formato do datagrama IP

O datagrama tem inicio com o campo **versão** (4 *bits).* Este campo indica a versão do protocolo IP sendo usada, o que determina o formato do cabeçalho *internet. O* campo seguinte, o **comprimento do cabeçalho** (IHL -*Internet Header Length* - 4 *bits),* fornece o referido comprimento em número de palavras de 32 *bits,* indicando o início do campo de dados. O valor mínimo válido para o comprimento do cabeçalho é de cinco palavras. O campo **tipo de serviço** (8 *bits)* fornece uma indicação dos parâmetros da qualidade de serviço desejada. Estes parâmetros são usados como orientação na seleção dos serviços de transmissão de dados das sub-redes. O campo **comprimento total** (16 *bits)* fornece o comprimento do datagrama, medido em octetos, incluindo o cabeçalho e a parte de dados. O comprimento máximo de um datagrama é de 65.535 octetos. Datagramas desse tamanho são impraticáveis para a grande maioria dos computadores. Na verdade, todos os computadores de uma rede *internet* devem estar preparados para aceitar datagramas de 576 octetos. Datagramas superiores a 576 octetos podem ser enviados quando o emissor tem certeza de que o destinatário está apto a aceitá-los.

O campo **identificação** (ID - 16 bits) é usado na montagem dos fragmentos de um datagrama. O campo **flags** (3 bits) serve ao controle de fragmentação, indicando se um datagrama pode ou não ser fragmentado e se houve fragmentação. O campo **offset de fragmento** (13 bits) indica o posicionamento do fragmento dentro do datagrama original.

Este posicionamento é medido em unidades de 8 octetos (64 bits). Este campo vale zero em datagramas não-fragmentados e no primeiro fragmento de um datagrama.

O campo **tempo de vida** (8 *bits)* indica o tempo máximo que o datagrama pode trafegar em uma rede *internet,* sendo este campo decrementado em cada *gateway.* Quando o seu conteúdo chega a zero, o datagrama é descartado. O objetivo é eliminar datagramas que não atingem o seu de destinatário. O campo **protocolo** (8 bits) indica o protocolo usuário do IP, cujos dados serão transportados na parte de dados do datagrama. O campo **checksum do cabeçalho** (16 bits) serve para identificar erros ocorridos durante a transmissão ou na atualização do cabeçalho; desta forma, o *checksum* é recalculado e verificado a cada ponto onde o cabeçalho é processado.

Em seguida estão os campos de endereços: **endereço de origem** (32 bits) e **endereço de destino** (32 bits), respectivamente, endereços IP do emissor e do destinatário do datagrama. O campo **opções** possui tamanho variável, podendo conter nenhuma ou várias opções. O campo é dividido em duas partes, uma indicando a classe da opção e a outra, o número da opção. As classes podem ser de controle, de indicação de erros e de medição ou testes. Dentro de cada classe, há os números de opção que identificam as funções auxiliares disponíveis. Finalmente, o campo **padding**, de tamanho variável é usado para garantir que o comprimento do cabeçalho do datagrama sempre seja um múltiplo inteiro de 32 bits.

2.1.3.3. Endereçamento

Um endereço IP é composto de um identificador de sub-rede mais um identificador da estação nessa sub-rede. Esta identificação independe da sub-rede física subjacente. Assim, para efeito de encaminhamento local (dentro da mesma sub-rede), o endereço IP é utilizado na estação emissora para a obtenção do endereço físico da estação de destino. Esse procedimento denominado **mapeamento** e também ocorre no sentido inverso: a partir do endereço físico é obtido o endereço IP (muito usado no caso de estações sem disco). Para ambos os processos existem protocolos específicos.

Por exemplo, no envio de um datagrama através de uma sub-rede local *Ethernet* para uma estação na mesma sub-rede, a camada IP da estação origem obtém o endereço físico *(Ethernet 48 bits)* a partir do endereço IP destino, fornecido junto com o datagrama pela camada superior (TCP ou UDP). Assim, o datagrama é diretamente enviado a estação de destino sem interferência de nenhum *gateway.*

2.1.3.4. Classes e Formatos de Endereços

Como foi visto no item anterior, um endereço IP possui dois campos: o identificador de estação (id. estação) e o identificador de sub-rede (id. rede). A capacidade de representação de endereços de sub-redes e estações é limitada pelo número de *bits* alocados em cada campo. Por outro lado, a necessidade de representação de sub-redes e estações varia para cada rede *internet;* em alguns casos, uma rede *internet* pode interconectar mais sub-redes que estações; em outros casos, pode ocorrer exatamente o contrário.

Por esse motivo foram criadas cinco classes de endereços IP (A, B, C, D, E). O número de *bits* de um endereço IP é fixo: *32 bits,* mas a forma como esses *bits* são alocados para a representação das sub-redes e estações varia de acordo com a **classe** da rede *internet.* A identificação da classe utilizada é feita através dos *bits* iniciais do campo endereço.

As classes são representadas na **Figura 5**:

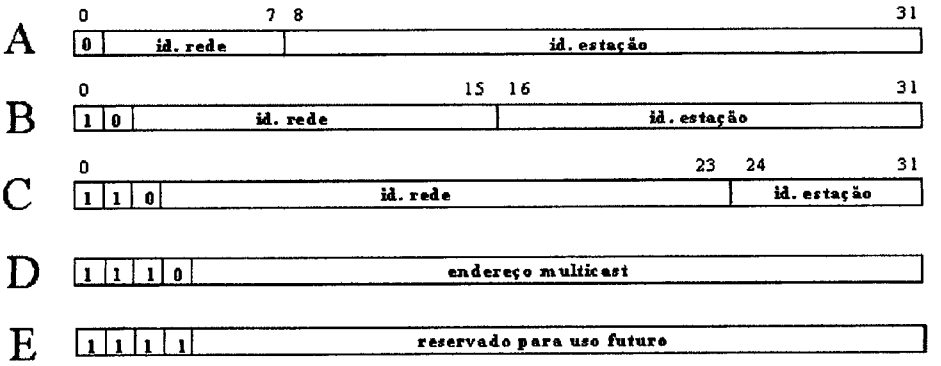

Figura 5 – classes de endereçamento

A **classe A** atende as necessidades de redes de grande abrangência constituídas de poucas sub-redes e com elevado número de estações, estando disponíveis 8 *bits* (o *bit* mais significativo vale 0) para identificação das sub-redes e 24 *bits* para a identificação das estações. A **classe B** representa redes intermediárias, com 16 *bits* (os dois *bits* mais significativos valem 1 e 0) para a identificação das sub-redes e 16 para as estações. A **classe C** atende tipicamente à faixa das redes locais. Como estas são bastante numerosas, são reservados 24 *bits (os 3 bits* mais significativos valem 1, 1 e 0) para a identificação das sub-redes e apenas 8 *bits* para a identificação das estações. A **classe D** identifica um endereço *multicast* (de difusão) e a **classe E** está reservada para uso futuro.

2.1.3.5. IPv6

Também conhecido como **IPng** (IP Next Generation) o **IPv6** é a nova versão do **Internet Protocol**. Ela deverá substituir progressivamente o IPv4 (o protocolo atual da Internet), estendendo o espaço de endereçamento corrente, o qual se tornou muito pequeno para acomodar a atual taxa de crescimento da rede. Projetado para atender redes de alta performance (Gigabit Ethernet, ATM, etc.), também é eficiente em redes de banda limitada

como no caso das redes sem fio, atendendo assim plataformas para as novas funcionalidades da Internet.

Os endereços são formados por 128 bits de comprimento (contra os atuais 32 do IPv4), implicando num aumento em potencial no número de hosts para 3.4×10^{38} o que equivale ao quádruplo do IPv4 que está nitidamente se esgotando.

Os objetivos principais do IPng eram [TANENBAUM, 1997]:

- suportar bilhões de *hosts*, mesmo com uma alocação ineficiente do espaço de endereçamento;
- reduzir o tamanho das tabelas de roteamento;
- simplificar o protocolos, para permitir que os roteadores processassem os pacotes mais rapidamente;
- prover uma melhor segurança (através de autenticação e privacidade) do que o IP atual;
- prestar mais atenção ao tipo de serviço, especialmente para dados de tempo-real;
- auxiliar o *multicasting* através da especificação de escopos;
- permitir que *hosts* móveis circulem sem precisar mudar de endereço;
- possibilitar a evolução futura do protocolo;
- viabilizar a coexistência do novo e do antigo protocolo por vários anos.

Embora pilhas IPv6 nativas não possam inter-operar diretamente, com as pilhas IPv4, mecanismos de compatibilidade IPv4 foram introduzidos no IPv6, para assegurar uma transição menos dolorosa.

2.2. Técnicas de Busca Heurísticas

2.2.1. O uso de Heurísticas

Para resolver eficientemente muitos problemas difíceis, geralmente é necessário comprometer as exigências de mobilidade e sistematicidade e construir uma estrutura de controle que não mais garanta encontrar a melhor resposta, mas que quase sempre encontre uma resposta muito boa. Deste modo apresentamos a idéia de heurística. A heurística é uma técnica que melhora a eficiência de um processo de busca, possivelmente sacrificando pretensões de completeza. A heurística é como um guia de turismo. Ela é válida no sentido de que aponta para direções geralmente interessantes; é imprópria no sentido de que pode deixar de fora pontos de interesse para determinados indivíduos. Certas heurísticas podem ajudar a guiar um processo de busca sem sacrificar a completeza que o processo possa ter tido anteriormente. Outras podem também, ocasionalmente, fazer com que um excelente caminho seja negligenciado (na verdade é o que ocorre com muitas das melhores delas). Porém na média, elas melhoram a qualidade dos caminhos que são explorados. Usando uma boa heurística, podemos esperar obter boas soluções (embora possivelmente não ótimas) para problemas difíceis. *"Há boas heurísticas de propósito geral úteis em uma ampla variedade de domínios. Além disso, é possível construir heurísticas de propósito específico que exploram conhecimento específico do domínio da aplicação para resolver determinados problemas."* [RICH, 1993].

Sem a heurística, estaríamos irremediavelmente presos em uma explosão combinatória. Só isto já é argumentado suficiente em favor do seu uso. Mas há outros argumentos também:

- *"Na verdade, há evidências de que as pessoas, quando solucionam problemas, não são otimizadoras, mas satisfazedoras"* [SIMON, 1981]. Em outras palavras, elas buscam qualquer solução de que satisfaça a um grupo de exigências e, quando encontram uma, param de procurar. Um bom exemplo é a procura por um

lugar para estacionar o carro. Muitas pessoas param assim que encontram um lugar relativamente bom, mesmo que possa haver um lugar melhor adiante.

- *"Embora as aproximações produzidas pela heurística possam não ser, na pior das hipóteses, muito boas, a pior hipótese raramente surge no mundo real. Por exemplo, embora muitos grafos não sejam separáveis (ou quase isto) e, portanto não possam ser considerados um conjunto de pequenos problemas em lugar de um problema grande, muitos grafos que descrevem o mundo real o são"* [SIMON, 1981].

- Tentar entender por que uma heurística funciona ou por que não funciona geralmente resulta em uma compreensão mais profunda do problema.

2.2.2. A Aplicação de Heurísticas

A técnica de busca heurística consiste em determinar qual passo tomado se aproxima mais da meta final na resolução de um problema. Para aplicar esta técnica em um ambiente de programação, devemos primeiramente desenvolver uma medida quantitativa, com a qual um programa possa determinar qual dentre os vários estados (nós) é considerado mais próximo da meta. *"Um ser humano, diante de uma decisão, tende a selecionar a opção que lhe parece mais próxima da meta"* [BROOKSHEAR, 2000].

Um algoritmo de busca heurística geralmente corresponde aos seguintes passos:

a) Selecione arbitrariamente um estado inicial.

b) Para selecionar o próximo passo em busca da resolução do problema, procure o que estiver mais perto da meta final.

c) Enquanto o estado atual não atingir a meta final, retorne ao passo b.

2.3. API – Applications Programming Interface

Uma forma de detectar as aplicações que estão sendo executadas e ter uma ação sobre ela é utilizar-se da **Interface de Programação de Aplicativos** (API) do Windows.

As aplicações em nível de usuário não se comunicam diretamente com o hardware, despreocupando-se das partes primitivas do sistema operacional. O sistema operacional tem como princípio apresentar os dispositivos do computador de forma padronizada, onde independendo do dispositivo a interface para eles é uniforme, ou seja, gravar um arquivo para um disquete, disco rígido ou interface de rede, utiliza-se do mesmo comando. A parte do sistema operacional que se preocupa com a comunicação entre as aplicações do usuário com o hardware é denominada Interface de Programação de Aplicativos (API).

Por causa das APIs, os aplicativos não precisam cuidar sozinhos do trabalho escravo de manutenção de sistema. A existência de uma API torna a criação de programas Windows muito mais fácil, porque o programador não precisa escrever código para realizar tarefas comuns em grande quantidade de que todo programa necessita. Outra razão pela qual as primitivas do sistema operacional são tratadas pelas chamadas a API – elas ajudam a proteger o sistema operacional de falhas. Se todos os programas em execução no computador tivessem permissão para acesso direto a recursos como monitor, teclado, disco rígido e portas, seu computador poderia estar em grandes problemas. Sua tabela de alocação de arquivos (FAT) poderia se danificar rapidamente e, em suma, todos seus arquivos seriam perdidos. Ou se dois arquivos de dois programas diferentes fossem impressos ao mesmo tempo, os dois iriam para a impressora ao mesmo tempo.

Quando os programas utilizam as chamadas a API pré-escrita do Windows tudo fica bem. Os aplicativos que obedecem às regras devem se preocupar apenas com os problemas de que devem tratar, recebendo ajuda adequada do Windows. Quando todos os programas que estiverem sendo executados forem programas do Windows, normalmente não

há problemas. *"Se um programa de comunicação necessita de acesso a uma porta COM, por exemplo, o acesso é fornecido pelo Windows. Se essa porta COM especifica já estiver sendo utilizada, o Windows informa algo como 'Sinto muito, esta porta não está disponível'"* [COWART, 1999].

No modelo apresentado neste trabalho faz-se necessário à solicitação de uma relação das aplicações que estão sendo executadas pelo sistema operacional e a partir destes dados enviar ordens de bloqueio a estas aplicações. Isto só é possível, somente através da API.

A **figura 6** apresenta a forma de atuação da API, onde uma aplicação de usuário se comunica com os dispositivos do computador e também com outras aplicações de usuário através desta interface.

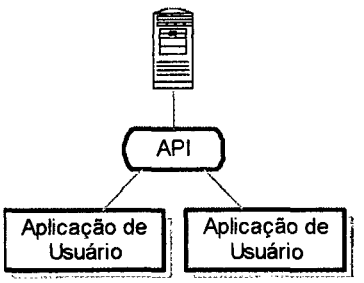

Figura 6 – API – Applications Programming Interface

2.4. Agentes de Raciocínios Distribuídos

Muitas vezes, os sistemas de resolução de problemas estão envolvidos com sistemas de raciocínios distribuídos. *"Definimos um sistema de raciocínio distribuído como aquele que é composto por um conjunto de módulos separados (em geral chamados agentes, já que cada módulo assume o papel de uma entidade de solução de problemas) e por um conjunto de caminhos de comunicação entre eles"* [RICK, 1993].

Para muitos tipos de aplicação, os sistemas de raciocínio distribuído apresentam vantagens significativas sobre os grandes sistemas monolíticos. Estas vantagens podem ser:

a) Modularidade do Sistema – é mais fácil criar e manter uma coleção de módulos quase independentes do que um módulo imenso;

b) Eficiência – nem todos os conhecimentos são necessários para todas as tarefas. Quando o modularizamos ganhamos a possibilidade de focalizar os esforços do sistema de solução de problemas do modo mais compensador possível.

Uma arquitetura para raciocínio distribuído precisa oferecer:

a) Um mecanismo para assegurar que as atividades dos vários agentes do sistema sejam coordenadas, para que o sistema global de solução de problemas alcance seu objetivo;

b) Uma estrutura de comunicação que permita a troca de informações;

c) Versões distribuídas das técnicas de raciocínio necessárias.

2.5. A Metalinguagem BNF (Backus-Naur-Form)

Na comunicação inter-agentes do modelo apresentado neste trabalho, foram utilizadas regras gramaticais através de metalinguagem. *"As regras gramaticais devem ser precisas de tal forma que um comando não seja confundido com outro, durante o processo da tradução"* [SHIMIZU, 1987].

A metalinguagem BNF oferece uma descrição precisa e que pode ser utilizada como algoritmos para programar a verificação de forma sintática correta de cada comando em programas tradutores ou compiladores. Nessa verificação sintática os tradutores utilizam algoritmos denominados "árvore de Parsing".

"Essas linguagens são bastante claras na descrição formas das regras gramaticais" [SHIMIZU, 1987], ou seja, formas corretas dos comandos, mas pobres ou incapazes de indicar a função semântica ou operação a ser executada por cada comando. Uma tentativa para suprir essa deficiência foi feita através da formalização de uma linguagem denominada *Vienna Definition Language*, que ao lado da descrição sintática define também a operação a ser executada pelo comando.

Uma linguagem de programação pode ser definida pelos seguintes elementos:

- **Conjunto de instruções:** é um conjunto S de seqüências (ou strings) finitas de caracteres e que são denominadas instruções ou comandos de linguagem;

- **Alfabeto da linguagem:** são os caracteres usados nas instruções que formam um conjunto finito;

- **Gramática ou sintaxe:** cada comando ou instrução possui uma regra sintática que identifica a sua forma correta de representação. As regras sintáticas de todos os comandos de uma linguagem formam a gramática dessa linguagem;

- **Semântica:** cada instrução ou comando possui uma finalidade ou significado semântico bem definido e que será executado por um trecho finito de programa de determinado computador.

2.5.1. Símbolos Utilizados

Esta metalinguagem utiliza os seguintes símbolos básicos para definir os comandos de uma linguagem:

<elemento>	: onde são colocados os elementos a serem definidos
: :=	: significa "é definido por"
/	: significa "ou"
{...}n m	: indica a ocorrência repetida do elemento interno aos sinais {...}, no mínimo **m** vezes e no máximo **n** vezes.
Símbolos terminais	: são sinais, letras ou algarismos diretamente utilizados nos comandos, sem sofrerem alteração

Cada regra definida pela linguagem BNF é formada pela sentença do tipo:

<elemento definido> : := <definição do elemento>

Exemplos:

a) Letra é um dos caracteres de A a Z:

<letra> : := A|B|C|D|...|Z

b) Dígito é um dos números de 0 a 9:

<dígito> : := 0|1|2|3|4|5|6|7|8|9

c) Uma letra ou uma letra seguida de lista de até 5 dígitos:

<nome de variável> : := <letra> | <letra>{<dígito>}5
1

3. METODOLOGIA DE DESENVOLVIMENTO DOS AGENTES

Uma forma de atingir o objetivo deste trabalho é utilizar-se de multiagentes para operar em máquinas distintas da rede. Estes agentes de auxílio à administração de redes são divididos em dois módulos executáveis: o Agente Espião (cliente) e o Agente Gerente da Rede (servidor).

A **figura** 7 apresenta o fluxo de mensagens trocadas entre os agentes que são:

Aplicações Locais: Após a conexão com o Agente Gerente da Rede, o Agente Espião passa a enviar todas as aplicações que o usuário está utilizando para serem analisadas.

Ordens de Bloqueio: Conforme a análise feita pelo servidor resultar em uma aplicação imprópria, o Agente Gerente da Rede toma a atitude de ordenar o Agente Espião para que encerre a aplicação e avise o usuário da irregularidade da utilização de tal aplicação.

Mensagens de Chat: Tem o objetivo de estabelecer comunicação entre os setores e usuários, podendo haver comunicação em qualquer sentido entre os agentes. Para enviar mensagens entre os Agentes Espiões, estas mensagens devem primeiro passar pelo Agente Gerente da Rede.

Controle Remoto: O Gerente da Rede pode sinalizar um ou um grupo de Agentes Espiões para que execute ações. Estas ações podem ser: desligar o computador, executar uma aplicação, reiniciar o computador, etc.

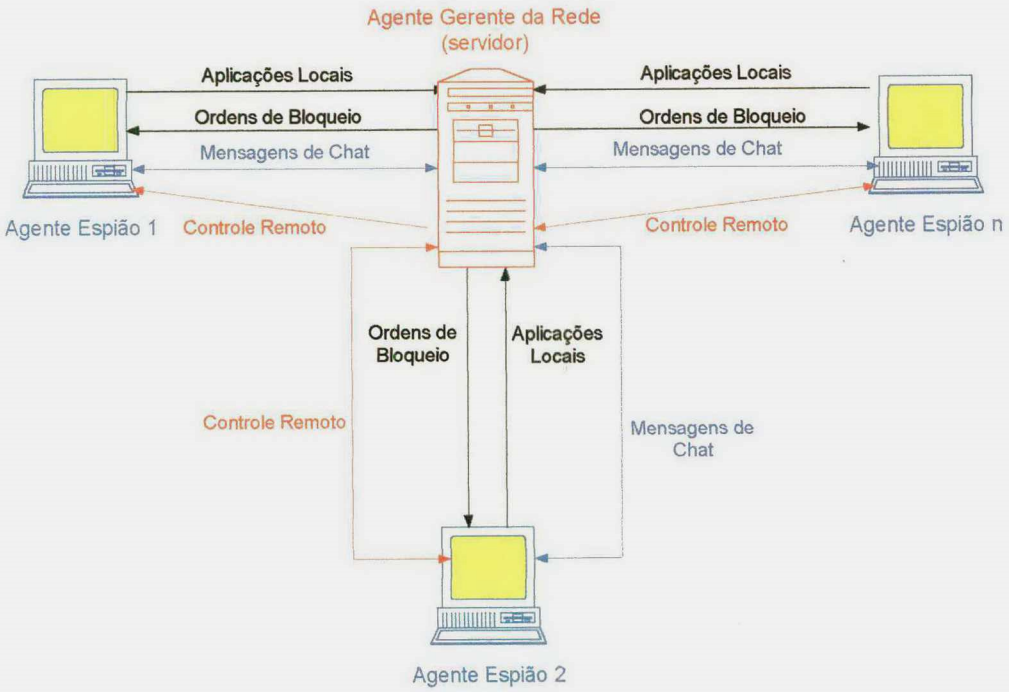

Figura 7 – Fluxo de mensagens entre os agentes

3.1. Especificações Técnicas dos Programas

A linguagem de programação escolhida para o desenvolvimento de ambos os agentes foi o Delphi 5 da Borland, pela facilidade de se trabalhar com tecnologias cliente/servidor, Interface de Programação de Aplicativos (API) do Windows e pelo código puro *assembly*[1] gerado. Para a execução do sistema são necessárias as seguintes configurações:

[1] Linguagem de programação de baixo nível.

- Devido à comunicação entre os agentes é necessário que o computador disponibilizado para o Agente Gerente da Rede possua um endereço IP fixo para ser conectado pelos outros agentes;

- Atualmente, ambos os agentes estão disponíveis apenas para o sistema operacional Windows;

- A configuração mínima para o computador do Agente Gerente da Rede deve possuir um processador acima de 233 MHz, com 32 MB de memória e o espaço em disco utilizado é de aproximadamente 4 MB;

- A configuração mínima para o computador dos Agentes Espiões deve possuir um processador acima de 133 MHz, com 16 MB de memória e o espaço em disco utilizado é de 520 KB.

3.2. Forma de Comunicação Entre Agentes

Cada um dos agentes, quando receber uma mensagem, deverá identificá-la e executar uma operação conforme o tipo de mensagem recebida. Isso exige uma sintaxe de comunicação entre os agentes. No modelo disposto neste trabalho, foram utilizadas seqüências de caracteres especiais que identificam o início, o fim e o tipo de uma mensagem trocada entre os agentes. Desta forma, quando se recebe uma mensagem do computador remoto, um evento é disparado dando início a uma operação de tradução e execução da mensagem.

O conjunto de instruções ou comandos utilizados na comunicação entre agentes foi definido obedecendo-se as regras da metalinguagem BNF – Backus-Naur-Form, por ser necessário utilizar um meio que descreva de maneira precisa e não ambígua a forma correta de cada comando.

Backus-Naur-Form é uma meta-linguagem que é usada para descrever a sintaxe de uma linguagem. Foi desenvolvida para descrever uma sintaxe de linguagem de

uma maneira mais natural que uma linguagem é capaz. A especificação consiste em um terminal do lado esquerdo, e uma ou mais produções do lado direito separadas pelo símbolo "::=".

Os meta-símbolos de BNF são:

::=	é o símbolo da metalinguagem que associa a um não-terminal um conjunto de cadeias de Terminais e/ou não-terminais, incluindo o símbolo da cadeia vazia. O não-terminal em questão é escrito à esquerda deste símbolo, e as diversas cadeias, à sua direita. Lê-se "define-se como".
|	Significado " ou ". É o símbolo da metalinguagem que separa as diversas cadeias que constam à direita do símbolo ::=
<*x*>	Representa um não-terminal, cujo nome é dado por um cadeia *x* de caracteres quaisquer. Os caracteres < e > são usados para delimitar o nome do não-terminal.
x	Representa um *terminal* da linguagem que está sendo definida. Deve ser denotado tal como figura nas sentenças da linguagem, e não entre os caracteres < e >, como ocorre no caso da denotação escolhida para os não-Terminais.
ε	Representa a cadeia vazia na notação BNF.
Yz	Representa uma cadeia construída pela concatenação dos elementos *y* e *z* nesta ordem. Estes dois elementos podem, por sua vez, ser símbolos de terminais, de não-terminais, de cadeia vazia, ou mesmo outras cadeias.

Quanto se quer delimitar um espaço válido para não terminais, utiliza-se a seguinte expressão:

$$\{\text{<terminal>}\}_{m}^{n}$$

Onde *m* determina o menor elemento permitido e *n* o maior elemento, exemplo:

$$\text{<bit>} ::= \{\text{<numero>}\}_{0}^{1}$$

Exemplo de um código em pascal:

<programa> ::= **begin** <lista_sentenças> **end**

<lista_sentenças> ::= <sentença> | <sentença> **;** <lista_sentenças>

<sentença> ::= <variável> ' **:=** ' <expressão>

<variável> ::= **A** | **B** | **C** | **D** | **E**

<expressão> ::= <variável> ' + ' <variável> | <variável> ' – ' <variável> | <variável>

Como podemos ver no exemplo anterior, os não-terminais da gramática BNF acima são 5 (programa, lista_sentenças, sentença, variável e expressão). Os terminais são 11 (**begin**, **end**, **;** **:=**, **A**, **B**, **C**, **D**, **E**, + e -). Um programa válido nesta linguagem é o que se encontra abaixo:

```
begin
    A := B + C ;
    B := B - D;
    A := B
end
```

É interessante notar que um programa só contém símbolos terminais, daí o porque do nome.

A seguir a sintaxe de cada instrução dos comandos utilizados para comunicação entre os agentes do modelo deste trabalho:

ELEMENTO		DEFINIÇÃO DO ELEMENTO
<letra>	::=	a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z|ã|á|à|â|é|ê|í|ó|õ|ú|ü
<digito>	::=	0|1|2|3|4|5|6|7|8|9
<string>	::=	ε | <letra> | <letra><digito> | <letra><string> | <digito><string>
<octeto>	::=	<digito> | <digito><digito> | {<digito>}$_{1}^{2}$ {<digito>}$_{0}^{5}$ {<digito>}$_{0}^{5}$
<endereço IP>:	::=	<octeto> '.' <octeto> '.' <octeto> '.' <octeto>
<numero>	::=	<digito> | <digito><numero>
<commando MEN>	::=	'<MEN:' <string> ':MEN>'
<comando CHA>	::=	'<CHA:' <endereço IP> '>' <string> ':CHA>'
<comando FEC>	::=	'<FEC:' <numero> ':FEC>'
<comando EXE>	::=	'<EXE:' <string> ':EXE>'
<comando TEM>	::=	'<TEM:' <numero> ':TEM>'
<comando ENC>	::=	'<ENC>'
<comando DES>	::=	'<DES>'
<comando REI>	::=	'<REI>'
<comando ATU>	::=	'<ATU>'
<comando KEY>	::=	'<KEY>'
<comando IMG>	::=	'<IMG>'
<comando ARQ>	::=	'<ARQ:' <string> ':ARQ>'
<comando APL>	::=	'<APL:' <string> ':APL>'
<comando ATC de Ag. Gerente para Ag. Espião>	::=	'<ATC:' <string> ':ATC>'
<comando ATC de Ag. Espião para Ag. Gerente>	::=	'<ATC>'

Objetivo de cada comando:

COMANDO	OBJETIVO	EMITENTE	
		Ag. Espião	Ag. Gerente
<comando MEN>	Ordenar o Agente Espião apresentar uma mensagem de aviso em vídeo.		X
<comando CHA>	Apresentar uma mensagem em uma tela especial de chat possibilitando retorno da mensagem. Esta mensagem pode ser de trocada entre qualquer agente, embora sempre a mensagem passe pelo Agente Gerente da Rede	X	X
<comando FEC>	Fechar o aplicativo impróprio detectado pelo Agente Gerente da Rede cujo código do processo foi passado como parâmetro.		X
<comando EXE>	Executar uma aplicação local cujo caminho do executável foi passado como parâmetro		X
<comando TEM>	Atualizar o tempo de envio de mensagens com o número em milissegundos passado como parâmetro		X
<comando ENC>	Encerrar a aplicação do Agente Espião		X
<comando DES>	Desligar o computador do Agente Espião		X
<comando REI>	Reiniciar o computador do Agente Espião		X
<comando ATU>	Solicitar ao Agente Espião o envio das aplicações imediatamente, sem esperar vencer o tempo estipulado.		X
<comando KEY>	Desabilitar as teclas CTRL+ALT+DEL		X
<comando IMG>	Solicita ao Agente Espião o envio de uma imagem mostrando a tela do computador		X
<comando ARQ>	Solicita ao Agente Espião o envio de um arquivo cujo caminho e nome do arquivo passado como parâmetro		X
<comando APL>	Envia aplicações locais do usuário para o Agente Gerente da Rede analisar	X	
<comando ATC de Ag. Gerente para Ag. Espião>	Envia lista de todos os Agentes Espiões conectados para atualizar a lista da tela de chat		X
<comando ATC de Ag. Espião para Ag. Gerente>	Solicita ao Agente Gerente que envie a lista de Agentes Espiões conectados	X	

O envio de imagem e arquivo do Agente Espião para o Agente Gerente da Rede é feito através de blocos de mensagem sem identificadores de sintaxe devido à variedade de tamanho dos arquivos não comportados em apenas um bloco. Este método é efetuado através do método *SendStream*[1] para envio das mensagens. O código para recepção está exemplificado na **figura 8** a seguir:

```
MemoryStream := TMemoryStream.Create;
Try
  while True do begin                                    {Enquanto for verdadeiro}
    BytesReceived := Socket.ReceiveBuf(Buffer,SizeOf(Buffer)); {recebe dados do buffer}
    if (BytesReceived <= 0) then                         {Se não recebeu nenhum dado do buffer}
      Break                                              {para de tentar}
    else begin                                           {Senão}
      MemoryStream.Write(Buffer,BytesReceived);          {escreve dados do buffer na memória}
      Sleep(100);                                        {aguarda recepção de novos dados}
    end;
  end;
finally
  MemoryStream.Free;
end;
```

Figura 8 – Código fonte exemplo para recepção de imagem e arquivo.

3.3. Agente Espião

3.3.1. Objetivo

O Agente Espião dá início a sua execução juntamente com o sistema operacional do computador que deve ser vistoriado de forma oculta, sendo encarregado de

enviar a cada intervalo de tempo, uma relação de todas as aplicações locais ao Agente Gerente da Rede, informando-o das aplicações que estão sendo executadas nas máquinas que deverão ser analisadas da rede. Outro objetivo é emitir ordens de bloqueio ao sistema operacional sobre as aplicações através da API do Windows e avisar o usuário das irregularidades que está cometendo como forma de inibição, caso algo impróprio for encontrado.

3.3.2. Interface

A interface do Agente Espião só pode ser acessada através do ícone apresentando próximo ao relógio do sistema operacional Windows, conhecido como *Icon Tray*, desde que, o usuário tenha a senha de acesso. A apresentação do Agente Espião é dividida nos seguintes módulos:

3.3.2.1. Módulo principal

Apenas os administradores da rede que possuem a senha do sistema podem acessar este módulo. Veja **figura 9** onde estão numerados os objetos e a seguir, uma menção de cada item:

---

[1] Método de envio de dados que consiste em enviar as informações em blocos.

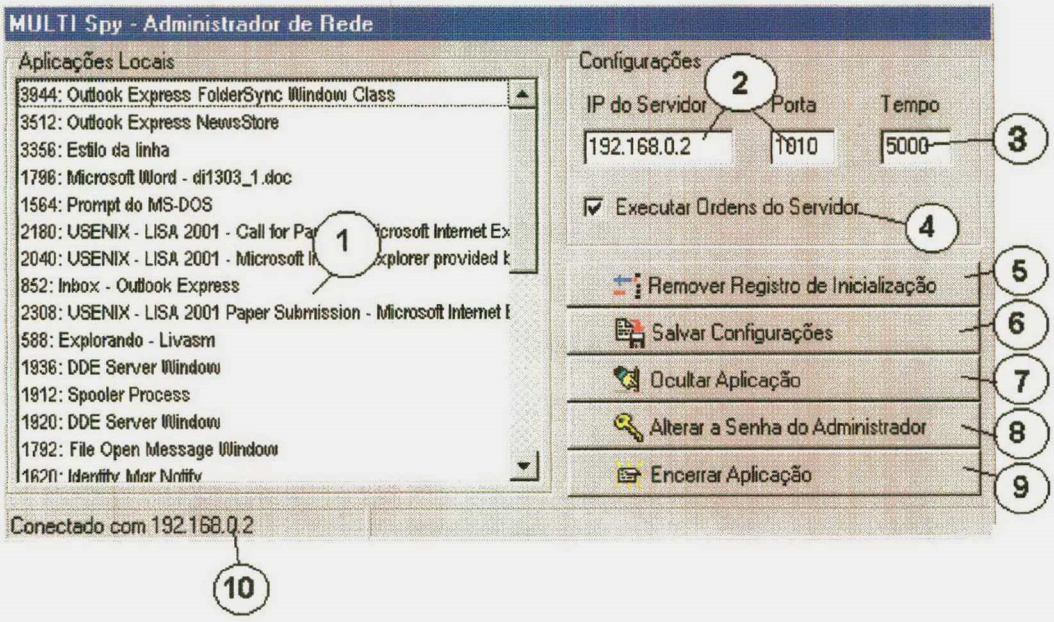

Figura 9 – Interface do módulo principal do Agente Espião

1 – Aplicações locais e informações do usuário que devem ser enviadas ao Agente Gerente da Rede. Estas informações são obtidas através da API do sistema operacional. As informações de usuário só são possíveis se o usuário estiver acessado a rede com sua senha;

2 – Configurações de conexão com o Agente Gerente da Rede contendo o endereço IP e a porta de comunicação utilizada;

3 – Tempo de reenvio de mensagens em milissegundos, ou seja, todas as informações contidas no quadro de aplicações locais e informações de usuário (Item 1 acima descrito) são atualizadas e enviadas ao Agente Gerente da Rede a cada evento *OnTimer*[1] disparado neste intervalo de tempo.

[1] Evento que acontece a cada período de tempo em uma Programação Orientada a Objetos.

4 – Permite imunizar o computador do Agente Espião de todas as ordens enviadas pelo Agente Gerente da Rede. Isto é útil em caso de manutenção do microcomputador que necessite utilizar um programa proibido pela organização.

5 – Remove a chamada automática na pasta de inicialização do Registro do Windows. Maiores informações poderão ser obtidas neste trabalho no subtítulo: *Forma de inicialização*;

6 – Salva todas as configurações do Agente Espião no registro do sistema operacional.

```
begin
  registro:=Tregistry.create;
  registro.RootKey:=HKEY_LOCAL_MACHINE;
  registro.openkey('SOFTWARE',false);
  registro.CreateKey('cnf');
  registro.CloseKey;
  registro.openKey('SOFTWARE\cnf',false);
  registro.WriteString('IPServer',form1.edit2.text);
  registro.WriteString('Porta',form1.edit3.text);
  registro.WriteString('Tempo',form1.edit4.text);
  if not registro.KeyExists('Codigo') then registro.WriteString('Codigo','');
  if Form1.CheckBox1.Checked then registro.WriteString('ExeOrdServer','T')
       else registro.WriteString('ExeOrdServer','F');
  registro.CloseKey;
  registro.free;
end;
```

Figura 10 – Código fonte do procedimento para salvar as configurações do Agente Espião

7 – Oculta a interface do Agente Espião. Após a utilização de qualquer item da interface o Agente Espião deve tornar-se oculto novamente para impedir que estranhos tenham acesso as configurações locais;

8 – Altera a senha do administrador da rede. Para maior segurança do sistema é necessário a alteração periódica da senha;

9 – Finaliza a aplicação do Agente Espião;

10 – Barra indicando o status da conexão com o servidor. O status poderá ser:

- Conectado com <endereço IP do Agente Gerente da Rede>
- Problemas na conexão... <número de tentativas falhas>

3.3.2.2. Módulo Chat

Com o objetivo de facilitar a comunicação entre os setores de uma organização, o Módulo Chat permite o envio de mensagens entre o Agente Espião e o Agente Gerente da Rede e também de Agente Espião para Agente Espião, podendo ser bloqueada pelo servidor caso o administrador da rede não tenha disponibilizado este tipo de serviço.

A comunicação de Agente Espião para Agente Espião é feita através do Agente Gerente da Rede, ou seja, a mensagem é encaminhada ao Agente Gerente da Rede que por sua vez, verifica o destinatário e redireciona a mensagem. Desta forma, faz-se possível bloquear a comunicação entre usuários e setores.

Para envio de uma mensagem, basta escrever a frase, selecionar o endereço do destinatário e pressionar o botão Enviar, como mostra a **figura 11** a seguir:

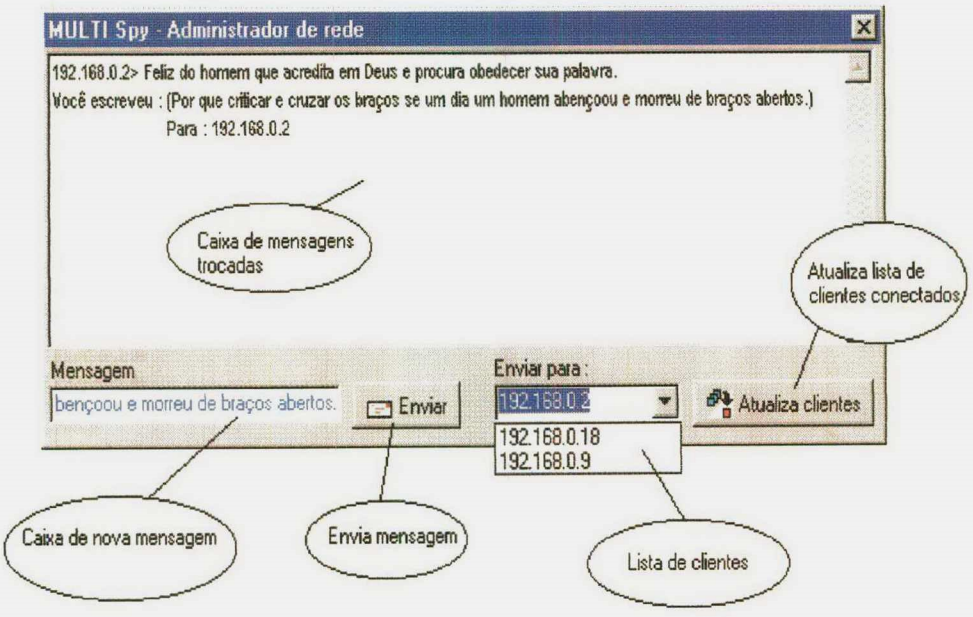

Figura 11 – Interface de comunicação entre os agentes

3.3.3. Forma de inicialização

O registro do Windows funciona como um coração para o sistema operacional. Este registro possui uma pasta com o caminho dos programas que deverão ser iniciados junto com o Windows, esta pasta localiza-se em: *HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run*, como mostra a **figura 12** a seguir:

administrador do sistema na organização durante a instalação do Agente Espião, podendo ser alterado posteriormente através da solicitação pelo Agente Gerente da Rede. Esta delimitação de tempo é necessária devido à topologia cliente/servidor utilizada, onde o cliente é ativado pelo usuário. Como o usuário não terá interação com o Agente Espião, define-se este tempo que dá início então ao evento OnTimer cujo algoritmo do procedimento é descrito na **figura 13** a seguir:

```
Procedure TForm1.Timer1Timer(Sender: TObject);
var EWProc: EnumWindowsProc;
begin
  if ClientSocket1.Active then begin                          {se a conexão do cliente estiver ativa}
    if StatusBar1.Panels.Items[0].text<>'Conectado com '+ClientSocket1.Address
       then Statusbar1.Panels.items[0].text:='Conectado com '+ClientSocket1.Address;  {coloca estatus da
conexão na barra de tarefa}
    Listbox1.Clear;                                           {limpa caixa de texto}
    EWProc := GetTitle;                                       {obtém títulos das aplicações}
    EnumWindows (@EWProc, 0);                                 {Carrega as aplicações na caixa de texto}
    Form1.ListBox1.Items.Add('====== USUÁRIO LOGADO : '+usuario+'  TEMPO : '+edit4.text+' ======
v1.2');          {adiciona a caixa de texto o nome do usuário e o tempo de disparo deste evento OnTimer}
    ClientSocket1.Socket.SendText('<APL>'+Listbox1.Items.Text);  {Envia a caixa de texto ao Agente
Espião}
    Tentativa:=0;                                             {zera o número de tentativas falhas}
  end else begin
    ClientSocket1.Active:=true;                               {tenta ativar a conexão}
    Inc(Tentativa);                                           {incrementa número de tentativas}
    if Tentativa>4 then Timer1.Interval:=120000;  {se o número de tentativas for maior que 4 passa a aguardar
2 minutos}
    if Tentativa>60 then application.terminate;  {se o número de tentativas for maior que 60 finaliza a
aplicação}
  end;
end;
```

Figura 13 – Código fonte do evento OnTimer

A sintaxe das mensagens é definida de uma forma que se possa identificar o código da aplicação (*handle*), para quando uma eventual ocorrência for detectada, o Agente Gerente da Rede retorne uma ordem de fechamento da aplicação através deste código.

3.3.5. Dispositivos de segurança

Para que o administrador de rede possa acessar as configurações do Agente Espião, o ícone da aplicação será disponibilizado no *Icon Tray*[1]. Clicando-se com o botão direito do mouse sobre este ícone o usuário tem as seguintes opções:

- Abrir a tela do módulo principal;
- Abrir a tela de chat;
- Desativar e ativar o servidor.

Apenas a tela de chat não solicita uma senha para acesso, ficando definido acesso às outras rotinas apenas pelo administrador da rede através desta senha.

Outra medida de segurança utilizada foi a desvinculação do processo com o *Kernel*[2] do sistema operacional, para impossibilitar o usuário de pressionar as teclas CTRL+ATL+DEL e finalizar a aplicação do Agente Espião. Para isso, utilizou-se a chamada a seguinte função disponível na biblioteca KERNEL32.DLL do núcleo do sistema operacional:

---

[1] Ícones que ficam próximo ao relógio do Windows
[2] Núcleo do sistema operacional

Function RegisterServiceProcess(dwProcessID, dwType: Integer): Integer;
stdcall; external 'KERNEL32.DLL';

3.4. Agente Gerente da Rede

3.4.1. Objetivo

O Agente Gerente da Rede é uma aplicação que deverá ser disponibilizada em um microcomputador constantemente ligado na rede aguardando a conexão dos Agentes Espiões. Sua principal função é receber as mensagens de aplicações locais do usuário, analisá-las e se encontrar ocorrências, conforme suas configurações predeterminadas pelo administrador da rede, retornar uma ou várias ordens de bloqueio a estes usuários.

3.4.2. Interface

Baseado em aplicativos Windows a interface do Agente Gerente da Rede foi definida com critérios para facilitar o compreendimento, utilização e configuração do aplicativo. O principal critério foi a divisão da interface em módulos, como mostra a **figura 14** a seguir:

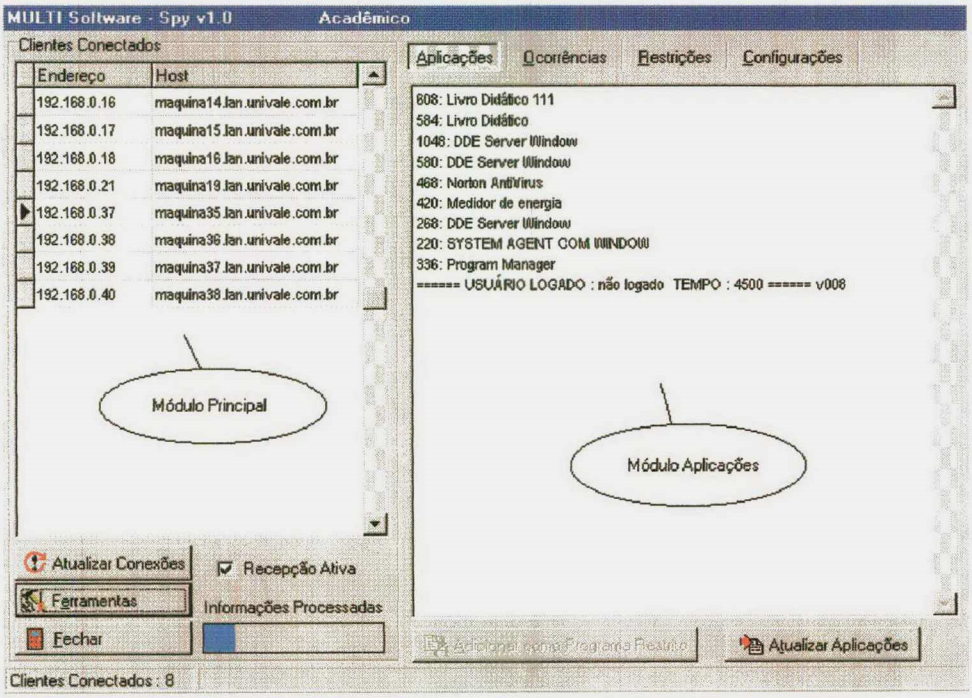

Figura 14 – Módulo principal e de aplicações

3.4.2.1. Módulo Principal

Clientes Conectados: Este item da tela principal apresenta a lista de Agentes Espiões conectados contendo a coluna de endereço IP e a coluna de Nome de *Host*[1]. Esta lista possui um evento que ocorre quando o usuário seleciona um novo item enviando uma mensagem ao Agente Espião para que retorne as aplicações imediatamente para atualizar as aplicações no Módulo de Aplicações.

[1] Qualquer sistema ou dispositivo de computador que esteja associado à inter-redes.

Atualizar Conexões: Quando um micro da rede é desligado incorretamente, o Agente Espião não solicita desconexão, permanecendo nesta forma como cliente ativo. Este botão serve para verificar se realmente os Agentes da lista estão conectados atualizando assim os dados apresentados.

Fechar: Fecha aplicação do Agente Gerente da Rede desconectando todos os Agentes Espiões.

Recepção Ativa: Quando este item está desmarcado, nenhuma verificação de irregularidade é feita. Esta opção é desmarcada automaticamente em caso de processamento complexo como o recebimento de uma imagem ou arquivo e também na manutenção de uma palavra ou programa restrito no Módulo Restrições.

Informações Processadas: Indica o percentual de informações processadas conforme número máximo estabelecido para verificação pelo administrador da rede. A partir de 100% de informações processadas, as aplicações recebidas são ignoradas para evitar sobrecarga de processamento do computador ou redução considerável de desempenho de modo a atrapalhar a execução de outras aplicações no servidor.

3.4.2.2. Módulo Aplicações

Aplicações: Apresenta as aplicações, versão da aplicação, usuário que acessou a rede e tempo de disparo do Agente Espião selecionado no Módulo Principal.

Adicionar como Programa Restrito: Adiciona o programa selecionado na lista de aplicações como programa restrito. Esta opção só é habilitada se a Recepção Ativa estiver desligada.

3.4.2.3. Módulo Ocorrências

Este módulo armazena em uma lista de todas ocorrências de irregularidades encontradas, com hora, medida tomada pelo Agente Gerente da Rede e identificação do usuário, assim como também, todos os comandos enviados e recebidos dos Agentes Espiões.

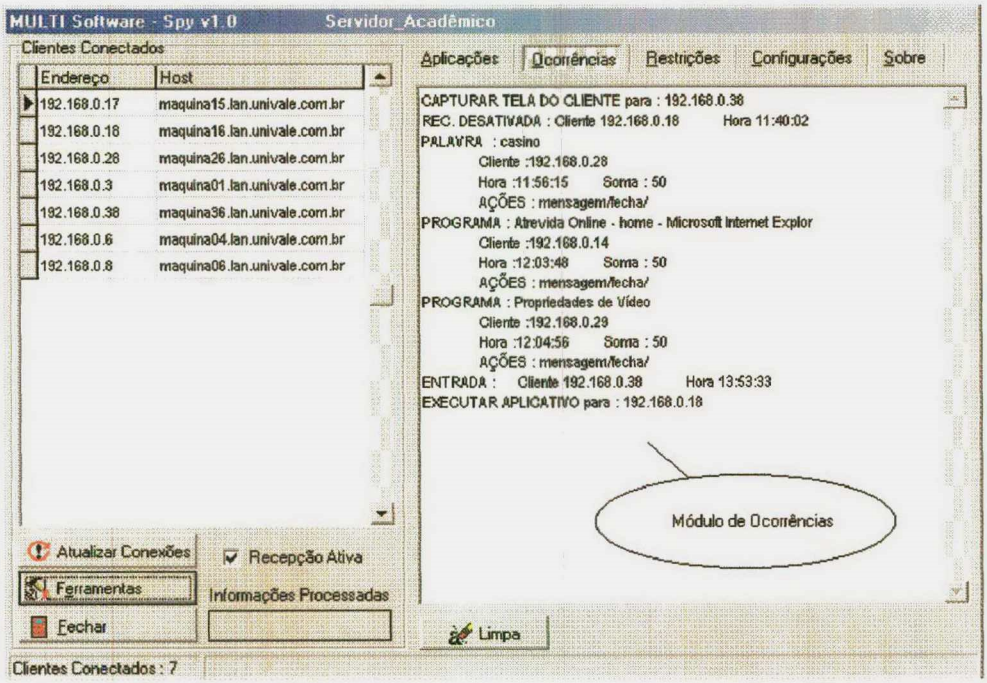

Figura 15 – Módulo ocorrências

Limpa: Serve para limpar a lista de ocorrências.

3.4.2.4. Módulo Restrições

Este módulo permite manutenção do cadastro de Programas e Palavras restritas.

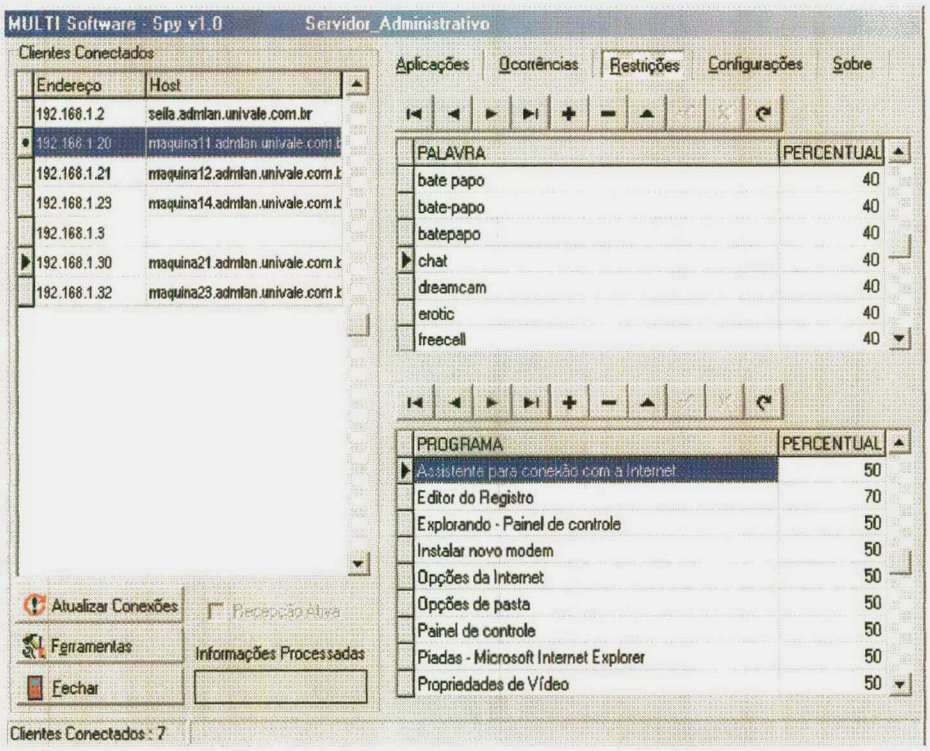

Figura 16 – Módulo Restrições

Tabela de palavras restritas: Onde são cadastradas palavras restritas com um percentual de confiabilidade definido para cada tipo de necessidade de restrição.

Tabela de programas restritos: Onde são cadastrados os programas restritos que não possuem palavras tituladas impróprias, mas devem ser restringidos por provocarem insatisfação da sua utilização pela organização.

3.4.2.5. Módulo Configurações

O Módulo de configurações disponibiliza ao administrador da rede um total controle sobre a aplicação tanto do Agente Espião quanto o Agente Gerente da Rede.

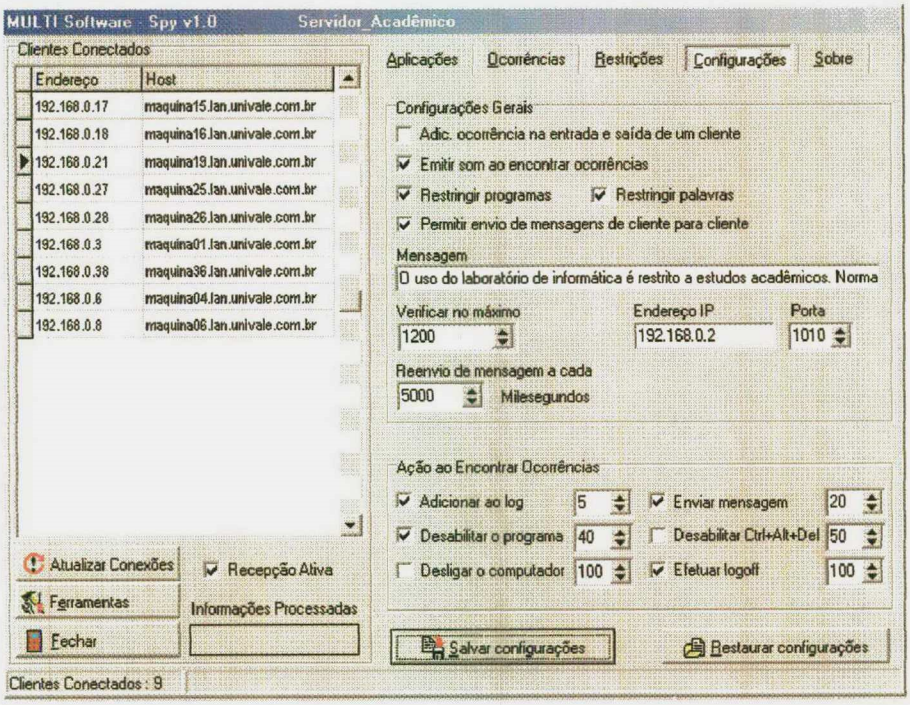

Figura 17 – Módulo Configurações

Configurações Gerais: Neste item de módulo pode se configurar os seguintes quesitos:

Se deverá ser adicionadas ocorrências de entrada e saída de clientes;

Se deverá emitir som ao encontrar ocorrências;

Se deverá restringir programas;

Se deverá restringir palavras;

Se será permitido enviar mensagens pelos usuários de clientes para clientes;

O número máximo de verificações por segundo;

O endereço IP do servidor;

A porta de comunicação do servidor;

O tempo de aguardo do cliente para reenvio de aplicações.

Ação ao Encontrar Ocorrências: Configura a ação a ser tomada e a que percentual de irregularidade encontrada deverá ser atribuída esta ação correspondente. As ações poderão ser:

Adicionar como ocorrência;

Enviar mensagem de aviso ao Agente Espião;

Desabilitar o programa;

Efetuar *logoff*[1] do computador;

Desligar o computador.

Salvar Configurações: Salva todas as configurações;

Restaurar Configurações: Retorna configurações originais;

3.4.2.6. Módulo Ferramentas

As ferramentas estão dispostas em um formulário separado contendo uma lista de Agentes Espiões conectados para possibilitar a seleção de um ou vários clientes para ser tratado, desta forma, pode-se enviar uma mensagem, executar uma aplicação ou emitir um comando para vários Agentes Espiões de uma só vez selecionando-os na lista de clientes. A **figura 18** a seguir apresenta o Módulo de Ferramentas:

[1] Estabelecer outro usuário para conexão da rede

Figura 18 – Módulo Ferramentas

Lista de clientes: Lista que permite a seleção de um ou vários Agentes Espiões conectados para executar uma ação disponível do módulo.

Adicionar todos: Adiciona todos os Agentes Espiões conectados a lista.

Excluir selecionados: Exclui Agentes Espiões selecionados da lista.

Excluir todos: Exclui todos os Agentes Espiões da lista.

Lista de mensagem de Chat: Apresenta um histórico de mensagens trocadas.

Limpa: limpa histórico de mensagens.

Enviar mensagem: Envia mensagem de chat.

Executar aplicação: Executa aplicação ou página de Internet na estação de trabalho do Agente Espião.

Download de arquivo: Transmite arquivo do Agente Espião para o Agente Gerente da Rede.

Capturar tela: Captura e apresenta a tela do Agente Espião.

Encerra clientes: Finaliza os Agentes Espiões, imunizando assim os microcomputadores da rede.

Desligar o computador: Desliga os computadores selecionados.

Efetuar logoff: Encerra todas as aplicações do cliente e solicita para que um novo usuário acesse a rede.

3.4.3. Análise da lista de aplicativos dos Agentes Espiões

3.4.3.1. Análise de restrições por programas restritos

Os programas são encontrados na barra de título dos aplicativos do Windows e deverão ser cadastrados pelo administrador da rede conforme surgem as necessidades de bloqueio de um programa que não é atraente para a empresa. Estes programas cadastrados, além de suas descrições, deverão também ter um percentual identificando o seu grau de problemas que poderá este programa causar. Exemplos:

PROGRAMA	PERCENTUAL
Painel de controle	80
Propriedades de vídeo	30
Propriedades de rede	100
Playboy – Microsoft Internet Explorer	100
Editor do Registro	100
Campo minado	30

É considerada operação imprópria do usuário quando se encontra um programa previamente cadastrado e que este programa atinja o percentual máximo permitido determinado pelo administrador da rede. Não existe nenhuma heurística de busca para este método, portanto, os programas cadastrados, geralmente são programas locais da máquina, ou aplicações que não existem palavras impróprias mas é considerado impróprio, como por exemplo: Painel de controle; Campo minado; Propriedades de vídeo etc. e não têm como cadastrar palavras para detecção da irregularidade por não possuírem palavras restringíveis.

3.4.3.2. Análise de restrições por palavras restritas

O cadastro de palavras se difere do cadastro de programas, pelo fato de que, as palavras poderão ser unidas para formar uma expressão que indique um grau de certeza para que haja o bloqueio. Este grau de certeza também é determinado pelo percentual máximo permitido. Exemplo:

PALAVRA	PERCENTUAL
Hacker	60
Netbus	90
Bate-papo	80
Chat	50
Invasão	30

Neste caso, se a configuração do servidor estiver para percentual máximo permitido em oitenta porcento (80%), o Agente Espião que for encontrado, por exemplo, com o seguinte conjunto de palavras {"hacker","Invasão"}, todas as aplicações que foram detectadas uma das duas palavras receberão ordens de bloqueio. Se apenas uma das duas palavras for encontrada, nada acontecerá, pois cada uma, individualmente pode não significar perigo.

É considerada operação imprópria do usuário quando for encontrado uma ou um conjunto de palavras que atinja um grau de confiabilidade, possibilitando assim determinar ao administrador da rede o nível de restrição aplicável a cada rede. Por exemplo, imaginemos um laboratório de pesquisa escolar, que utiliza microcomputadores conectados a Internet, se for encontrado a palavra "hacker", pode o aluno estar pesquisando sobre a cultura hacker e não estar em uma página hacker perigosa. Mas se for encontrado a palavra "hacker" em conjunto com outra palavra que também não dá certeza, como "invasão", aumenta o grau de certeza considerado suficiente para o bloqueio da aplicação. Caso apenas a palavra "hacker" for encontrada e o aluno estiver em uma página perigosa o programa não será bloqueado, fazendo-se necessário então, cadastra-lo como um programa restrito.

Este método de busca que sacrifica a certeza e permite nos desvincular de explosões combinatórias é conhecido como heurística, e é extremamente necessário para este caso, pois não existe maneira de prevenir com exatidão o que poderá ser encontrado, ficando assim, impossível manter um cadastro de todas os programas a bloquear pela sua infinidade e também pelo constante crescimento da internet.

A técnica de busca heurística consiste em determinar qual passo tomado se aproxima mais da meta final na resolução de um problema. Desta forma a busca pode não atingir a meta, mas, se aproximar muito até um ponto considerado satisfatório.

Para aplicar esta técnica em um ambiente de programação, devemos primeiramente desenvolver uma medida quantitativa, com a qual um programa possa determinar qual dentre os vários estados (nós) é considerado mais próximo da meta. Na aplicação em questão, para determinar o grau de certeza de cada palavra cadastrada, deverá ser cadastrado juntamente com a palavra um percentual que identifique ao processo de busca o nível de satisfação da mesma. Além disso, pode-se também determinar um percentual máximo permitido, possibilitando ao administrador de rede, aumentar ou diminuir o nível confiabilidade dos bloqueios.

O bom desempenho deste sistema, quanto à detecção das irregularidades, depende fortemente do cadastro de programas e palavras que a gerência da Intranet pretende bloquear.

3.4.4. Gerenciando o tempo de envio de mensagens do agente espião

Basicamente em uma aplicação cliente/servidor, o servidor é ativado quando recebe uma solicitação de prestação de serviço pelo cliente que por sua vez atende diretamente o usuário. Nesta aplicação modelo, o Agente Gerente da Rede, que é a aplicação servidor, também é ativada pelo cliente, mas o cliente não atende a solicitações do usuário. O que acontece na verdade é que o Agente Espião (cliente) deve enviar mensagens das aplicações que estão sendo executadas pelo usuário no computador local para o Agente Gerente da Rede analisar. Para isso, foi estabelecido um determinado intervalo de tempo entre as mensagens que deverá ser configurável pelo administrador da rede conforme suas necessidades e também conforme o tráfego da rede.

Para entender melhor a necessidade de alteração pelo servidor do tempo de envio de mensagens do Agente Espião, é necessário comentar sobre o crescimento do processamento do Agente Gerente da Rede conforme as palavras e programas forem cadastrados e também conforme o número de microcomputadores conectados.

Exemplo: Imagine um cadastro de restrições que tenha quarenta palavras cadastradas e vinte e cinco programas com vinte Agentes Espiões conectados simultaneamente. Cada Agente envia cerca de dez programas para serem avaliados em cada dois segundos. Com tudo isso, teria os seguintes dados:

40	palavras cadastradas
25	programas cadastrados
20	agentes
10	programas por agente
2	segundos de tempo

Com estes dados, qual seria o processamento do servidor?

O método utilizado para encontrar o número de verificações por segundo seria:

Se a cada mensagem recebida do Agente Espião contém uma lista de programas e a cada programa é verificado se está entre os programas restritos cadastrados, temos:

ProgramasParaAvaliar = (NumeroDeAgentes*ProgramaPorAgente*ProgramasCadastrados)

Se a cada mensagem recebida pelo Agente Espião é verificada uma vez para cada palavra, temos:

PalavrasParaAvaliar = (NumeroDeAgentes*PalavrasCadastradas)

PROCESSOS = (ProgramasParaAvaliar)+(PalavrasParaAvaliar)/segundos

Aplicando:

ProgramasParaAvaliar = (20*10*25)
PalavrasParaAvaliar = (20*40)
PROCESSOS = (5000+800)/2
PROCESSOS = 2900

Desta forma, conclui-se que, aproximadamente dois mil e novecentas verificações por segundo, em uma rede de apenas vinte Agentes Espiões ativos, pode haver grande perda de desempenho do computador disponibilizado como servidor se este for disposto a muitos clientes. A forma de amenizar o problema e reduzir o tráfego para um Agente Gerente da Rede pode ser feita de três maneiras:

a) descentralizar um Agente Gerente da Rede para vários servidores, onde cada Agente Gerente da Rede fica responsável por um grupo de Agentes Espiões;

b) priorizar palavras e programas de forma a descartar os menos importantes em caso de alto fluxo de tráfego ou alto nível de processamento no servidor;

c) gerenciar o tempo de reenvio de mensagens dos Agentes Espiões.

O gerenciamento do tempo, então, fica a cargo do administrador da rede, conforme achar necessário pode aumentar ou reduzir o tempo de reenvio de mensagens dos Agentes Espiões, podendo ser estes alterados através do módulo de configurações do Agente Gerente da Rede. No exemplo acima citado, se aumentássemos o tempo de dois segundos para quatro segundos, o tráfego da rede e o processamento do servidor cairiam pela metade.

Para evitar ataques por DOS (Denied Off Service) e sobrecarga de recebimento de mensagens, foi estabelecido um número máximo a ser verificado, se este número máximo for ultrapassado, o Agente Gerente da Rede passa a ignorar as mensagens recebidas. Esse valor máximo pode ser alterado através do módulo de configurações.

4. RESULTADOS

4.1. Delimitação e Área de Atuação dos Testes

Os dados foram obtidos através de testes realizados em uma rede acadêmica, com quarenta e oito microcomputadores divididos em dois laboratórios e uma central e na rede administrativa, com vinte e dois microcomputadores espalhados em sete setores da *UNIVALE – União das Escolas Superiores do Vale do Ivaí*. Ambas as redes possuem cabeamento par-trançado e placas de rede e roteadores de dez Mega Bits por segundo.

O pessoal envolvido nos testes pode ser classificado em três categorias:

a) **Funcionários:** pessoas de média de idade entre dezoito a cinqüenta anos que utilizam os recursos computacionais com o sistema de gerenciamento escolar e também utilizam a Internet para envio e recepção de mensagens eletrônicas e para acessar páginas referentes ao âmbito profissional;

b) **Alunos da faculdade:** pessoas de média de idade de dezoito a cinqüenta anos que utilizam os recursos computacionais para fins de aprendizado como pesquisa e prática de trabalhos acadêmicos, sendo estes, durante aula com acompanhamento do professor ou não;

c) **Alunos do colégio:** pessoas de média de idade entre seis e vinte anos que utilizam os recursos computacionais para fins de aprendizado como pesquisa e prática de trabalhos acadêmicos, sendo estes, durante a aula com acompanhamento do professor ou com autorização da direção do colégio.

4.2. Desempenho dos Microcomputadores e da Rede

Microcomputadores de usuários onde estão instaladas as aplicações do Agente Espião: Não foram registradas perdas significativas no desempenho dos computadores dos usuários, tanto na inicialização quanto na velocidade dos outros aplicativos, mesmo quando estes estiverem relacionados à comunicação através da Internet ou Intranet.

Microcomputador Servidor onde está instalada a aplicação do Agente Gerente da Rede: O computador disponibilizado para utilização do servidor, houve perda de desempenho quanto ao uso do processador. A média de uso do processador registrado para 20 (vinte) Agentes Espiões conectados aproximou-se a quarenta porcento (40%) como mostra a **figura 19** a seguir:

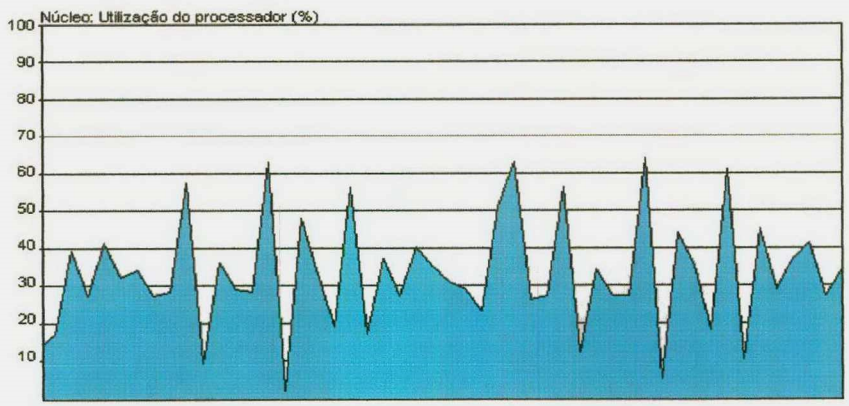

Figura 19 – Utilização do Processador

Rede: O tempo registrado para a transferência de um arquivo de cinqüenta e sete megabytes (57 MB) entre duas estações de trabalho da rede sem a presença de qualquer Agente Espião e Agente Gerente da Rede foi de cinqüenta e três segundos (53") e com todas as estações de trabalho sendo vistoriadas pelo Agente Gerente da Rede o tempo registrado

atingiu cinqüenta e quatro segundos (54"), uma perda considerada insignificante ocorrida pelo maior número de informações roteadas.

4.3. Redução das Ocorrências de Irregularidades

O principal objetivo deste trabalho é reduzir o uso de programas não apropriados para o âmbito profissional e organizacional de instituições. Os resultados obtidos nos testes foram divididos em fases que estão descritas nos gráficos a seguir indicando o número de ocorrências encontradas para cada fase.

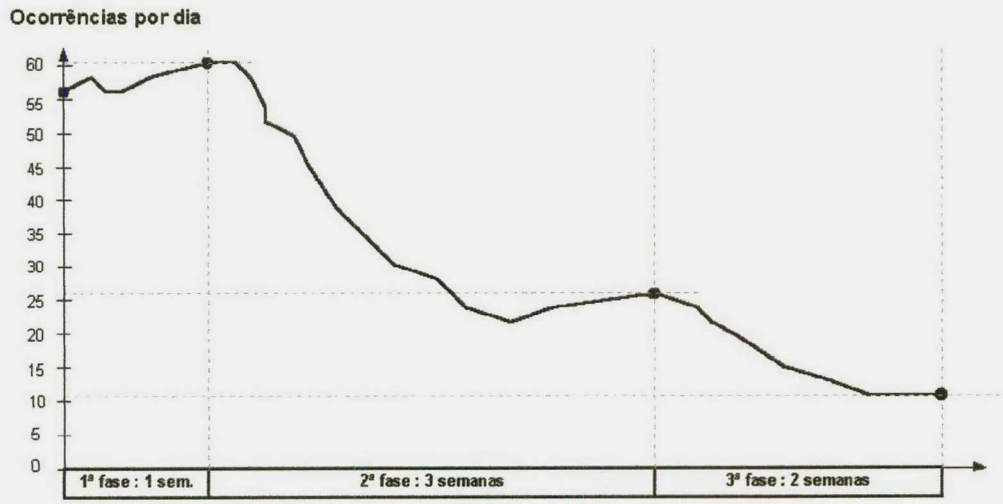

Figura 20 – Redução de Ocorrências Encontradas

1ª fase: Foram instalados os agentes e sem avisar ou restringir o usuário e apenas adicionando a uma lista de ocorrências, foi o método utilizado para analisar e cadastrar palavras e programas restritos. Nesta fase houve um pequeno crescimento nas ocorrências atingindo um total de 61 (sessenta e uma) aplicações impróprias por dia, sendo atribuído também ao crescimento do uso do laboratório de informática nos finais de semana.

2ª fase: Deu-se início as restrições. Toda aplicação cuja confiabilidade não fosse aceita, passou a ser restringida com o encerramento forçado das aplicações. O Gerente da Rede passou a avisar o responsável da rede, e emitir solicitação de uma mensagem de aviso ao usuário como forma de inibição. O resultado foi uma grande redução das ocorrências diárias, de 61 (sessenta e uma) para 26 (vinte e seis) com um pequeno acréscimo na última semana, ficando como piso um total de 22 (vinte e duas) ocorrências.

3ª fase: A redução do desempenho do sistema na última semana se deu devido aos usuários que, conhecendo o funcionamento do sistema, tentaram abrir novas páginas de Internet, novos jogos e formas de burlar o sistema. Assim, surge a necessidade de uma revisão no cadastro de palavras e programas restritos, sendo alteradas e incluídas novas possibilidades de ocorrências. Nesta fase, houve uma nova melhoria, com redução de 26 (vinte e seis) para 11 (onze) ocorrências diárias.

Desta forma, podemos observar que, quanto melhor está o cadastro das restrições, melhor é o desempenho do sistema e um bom cadastro se consegue com observação constante sobre os novos programas e páginas de internet que vão surgindo.

4.4. Falhas Detectadas

Os testes realizados para obtenção dos resultados também apresentaram falhas que deverão ser corrigidas para que haja um melhor funcionamento deste sistema. As falhas são:

a) Com o Agente Gerente da Rede desligado ou com falha na comunicação dos dados, os Agentes Espiões ficam ativos, mas, sem enviar informações das aplicações locais, permitindo a execução de qualquer programa, inclusive o Regedit.exe e também MsConfig.exe, que são aplicativos do Windows que permitem a remoção da chamada do executável do Agente Espião no registro de inicialização, com essa possibilidade, quando o computador reiniciar, estará totalmente livre de qualquer restrição;

b) Quando o usuário desliga o computador de forma incorreta ou quando ocorre uma queda de energia o Agente Espião não solicita desconexão com o Agente Gerente da Rede, e este fica com a conexão aberta acarretando erro na utilização desta conexão;

c) A única forma de o usuário descobrir o nome do arquivo executável do Agente Espião é utilizar-se de aplicações que mostram todos os processos ativos não vinculados ou registrados no sistema operacional ou através do registro de inicialização, mas, se isso acontecer, ele pode reiniciar o computador em modo de segurança ou em modo MS-DOS e excluir a aplicação do Agente Espião.

d) Uma impressora instalada e compartilhada na rede pelo protocolo TCP/IP pode abrir uma porta de comunicação entre uma estação de trabalho e um dos agentes, se isso acontecer e se o intruso conhecer a sintaxe de envio de mensagens entre os agentes pode ter total controle sobre as estações.

Com exceção da falha 'd' acima citada, as falhas realmente ocorreram na fase de testes e todas deverão ser implementadas soluções para um melhor funcionamento do sistema.

5. CONCLUSÃO

5.1. Objetivos Atingidos

As tecnologias utilizadas como os agentes conectados através da rede TCP/IP e a detecção de irregularidades através da busca heurística no desenvolvimento do sistema possibilitaram ao trabalho ir de encontro com os objetivos propostos.

Sem a busca heurística de aplicações impróprias seria impossível controlar programas e páginas de Internet devido seu crescimento descontrolado. Com o cadastro de apenas algumas palavras e alguns programas, pode-se chegar a um resultado satisfatório sem necessidade do cadastro específico para a aplicação através da heurística.

A utilização de agentes no desenvolvimento do sistema contribuiu para atingir o objetivo do trabalho pelos seguintes aspectos:

- Possibilitou ao administrador da rede ficar sabendo instantaneamente de uma tentativa de acesso a algo indesejável;

- Auxiliou no processo de inibição do usuário por saber que pode ser abordado pelo administrador e pelo próprio sistema que fecha a aplicação, disponibiliza uma mensagem do que está irregular em vídeo e emite aviso sonoro;

- Permitiu comunicação entre os setores e usuários;

- Permitiu controlar alguns dispositivos do computador remoto;

5.2. Implementações Futuras

Os testes realizados após a fase de implementação do sistema, além de falhas detectadas, proporcionaram o surgimento de novas idéias para implementação que estão relacionadas a seguir:

a) Devido o crescimento e a diversificação que vêm surgindo na área de sistemas operacionais, pretende-se apresentar uma versão do sistema para a plataforma Linux.

b) Como o bom funcionamento do sistema depende do cadastro de suas restrições, será possível no futuro, a atualização através da Internet dos dados através de cadastro pré-estabelecidos por assuntos. Exemplo: imaginemos que uma determinada instituição queira restringir em sua empresa páginas pornográficas, ela poderá fazer a atualização pela Internet de seu cadastro de programas e palavras restritas selecionando este assunto para restringir, podendo ainda incluir e excluir estes itens. Seria como muitos softwares antivírus existentes no mercado hoje que atualizam sua base de dados para detecção de novos vírus;

c) Para melhorar a forma de detecção de irregularidades, pretende-se usar novas técnicas de inteligência artificial como as *redes neurais* para melhorar o desempenho das buscas;

d) Atualmente no sistema, toda irregularidade é encontrada pelo título da aplicação junto ao sistema operacional. Pretende-se também, implementar uma forma de detectar as irregularidades por todo o fluxo de palavras transmitidas e recebidas pela Internet, desde que haja uma suspeita;

e) O foco do trabalho foi voltado a detecção de irregularidades através de agentes distintos exigindo uma comunicação entre os terminais, isso facilitou também a criação de uma tela de comunicação de mensagens e arquivos entre os usuários. O resultado

dessa comunicação foi muito aceita pela parte administrativa da empresa e por isso, pretende-se no futuro melhorar esta interface criando-se contas e grupos de usuários.

f) Correção das falhas detectadas nas fases de testes.

Estas implementações futuras não foram feitas para este trabalho devido a curta disponibilidade de tempo e a necessidade de encerramento deste estudo, mas o programa, não necessariamente se encerra aqui, pretende-se cada vez mais expandir a abrangência e também o desempenho do sistema buscando novas formas para desempenhar seus objetivos.

5.3. Considerações Finais

Os resultados obtidos com o desenvolvimento deste sistema multiagente comprovaram que podem auxiliar e evoluir a administração de redes, reduzindo o acesso indevido de aplicações indesejáveis para empresa e organizações que possuem redes com recursos modernos. Isso, melhora a produtividade dos funcionários e, do ponto de vista educacional, auxilia no aprendizado e na boa educação dos alunos. Além disso, reduz o trabalho dos administradores de redes que têm como função, manter uma boa imagem da empresa através de seus equipamentos de informática.

6. REFERÊNCIAS BIBLIOGRÁFICAS

[BROOKSHEAR, 2000] BROOKSHEAR, J. Glenn, **Ciência da computação: uma visão abrangente,** 5ª Ed.- Porto Alegre: Bookman, 2000.

[CASAD, 1999] CASAD, Joe, **Aprenda em 24 horas TCP/IP** / Joe Casad, Bob Willsey, Rio de Janeiro: Campus, 1999.

[COMER, 1999] COMER, Douglas E., **Interligação em rede com TCP/IP,** Volume 2 / Douglas E. Comer, David L. Stevens, Rio de Janeiro: Campus, 1999.

[COWART, 1999] COWART, Robert, **Dominando o Windows 98 "A Bíblia"**. São Paulo: Makron Books, 1999.

[GASPARINI, 1993] GASPARINI, Anteu Fabiano Lúcio, **TCP/IP: solução para conectividade** / Anteu Fabiano Lúcio Basparini, Francisco Eugênio Barrella, São Paulo: Erica, 1993.

[POSTEL, 1981] POSTEL, J.B., **Transmission Control Protocol** – Darpa Internet Program Protocol Specification. Request for comments 793, University of Southern California / Information Sciences Institute, Setembro 1981.

[RENAUD, 1994] RENAUD, Paul E., **Introdução aos sistemas cliente/servidor: um guia prático para profissionais de sistemas**. Rio de Janeiro: Infobook, 1994.

[RICH, 1993] RICH, Elaine, **Inteligência artificial** – São Paulo: Makron

Books, 1993.

[ROSA, 1998] ROSA, Rubens Freire, **Estudo e Implementação dos Protocolos TCP/IP para um Sistema Operacional Tempo Real.** São José dos Campos, 1998. Tese em Engenharia Eletrônica e Computação na Área de Dispositivos e Sistemas Eletrônicos – Divisão de Pós-Graduação do Instituto Tecnológico da Aeronáutica.

[SHIMIZU, 1987] SHIMIZU, Tamio, **Introdução à Ciência da Computação,** 2° Ed. – São Paulo: Atlas, 1987.

[SIMON, 1981] SIMON, H A. **The Sciences of the Artificial**, 2nd ed. Cambridge, MA, MIT Press, 1981.

[SOARES, 1995] SOARES, Luiz Femando G., **Release de computadores: das LANs, MANs e WANs às redes ATM /** Luiz Fernando Gomes Soares, Guido Lemos, Sergio Colcher. – Rio de Janeiro: Campus 1995.

[TANENBAUM, 1997] TANENBAUM, Andrew S., **Redes de computadores,** 3ª Edição, Rio de Janeiro: Campus, 1997.

7. BIBLIOGRAFIA

STARLIN, Gorki, **Segurança na Internet – Microsoft Proxy Cachê, Novell Border Manager – Um guia de tecnologia e inteligência contra hackers;** Rio de Janeiro: Book Express, 1998.

GERSTING, **Fundamentos matemáticos para a ciência da computação**; tradução Lúcio Leão Filho. – Rio de Janeiro: LTC, 1995.

Guia Oficial Microsoft: Soluções para Intranet; tradução: Altair Dias Caldas; revisão técnica : Nabuo Sato. – São Paulo: Makron Books, 1999.

HARRISON, **Internet: kit de conexão**, tradução: Claudio de Souza Soares. – São Paulo: Berkeley, 1995.

TANENBAUM, Andrew S., **Sistemas operacionais,** 2ª Edição, Porto Alegre: Bookman, 2000.

TANENBAUM, Andrew S., **Redes de computadores,** 3ª Edição, Rio de Janeiro: Campus, 1997.

FERNANDES, André, KANE, Roberto, ARCOVERDE, Rodrigo, **Delphi 4 completo básico – avançado**, Rio de Janeiro : Book Express Ltda., 1998.

SONNINO, Bruno, **Desenvolvendo aplicações com Delphi 5,** São Paulo: Makron Books, 2000.

RUMBANGH, James, **Modelagem e projetos baseados em objetos,** 8ª edição, Rio de Janeiro: Campus, 1994.

MINASI, Mark, **Dominando o Windows 2000 server "A Bíblia";** São Paulo: Makron Books, 2001.

VASKEVITCH, David, **Estratégias cliente/servidor;** São Paulo: Berkeley, 1995.