UNIVERSIDADE DO EXTREMO SUL CATARINENSE – UNESC

CURSO DE CIÊNCIA DA COMPUTAÇÃO

ARLEI CORRÊA ZOMER

MONITORAMENTO DE *SOFTWARES* ATIVADOS POR UM USUÁRIO NO PERÍODO DE UTILIZAÇÃO DE COMPUTADORES UTILIZANDO A API WIN32

CRICIÚMA, JULHO DE 2007.

ARLEI CORRÊA ZOMER

MONITORAMENTO DE *SOFTWARE*S ATIVADOS POR UM USUÁRIO NO PERÍODO DE UTILIZAÇÃO DE COMPUTADORES UTILIZANDO A API WIN32

Trabalho de Conclusão de Curso apresentado para obtenção do Grau de Bacharel em Ciência da Computação da Universidade do Extremo Sul Catarinense.

Orientador: M.Sc. Paulo João Martins

CRICIÚMA, JULHO DE 2007.

ARLEI CORRÊA ZOMER

MONITORAMENTO DE *SOFTWARE*S ATIVADOS POR UM USUÁRIO NO PERÍODO DE UTILIZAÇÃO DE COMPUTADORES UTILIZANDO A API WIN32

Submetido ao corpo docente do Departamento de Ciência da Computação da Universidade do Extremo Sul Catarinense como um dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

_____

Profª. M.Sc. Ana Cláudia Garcia Barbosa
Coordenadora do Curso de Ciência da Computação

Banca Examinadora:

_____

Prof. M.Sc. Paulo João Martins (UNESC)
Orientador

_____

Profª. M.Sc. Silvana Campos de Azevedo (UNESC)

_____

Esp. Carina Búrigo (Departamento de Tecnologia da Informação – UNESC)

Dedico aos meus queridos pais, Agenor e Anadir, a toda minha família, minha namorada Patrícia e meus amigos.

AGRADECIMENTOS

As palavras descritas aqui são de maior gratidão, respeito e sinceridade possível. Sem essas pessoas a realização deste sonho não ocorreria. Agradeço:

A Deus, por me dar força e vontade para não desistir nos momentos mais difíceis.

Aos meus pais, Agenor e Anadir, que são os maiores responsáveis pela realização deste sonho. Pai, mãe amo vocês!

A toda minha família que sempre me apoiou e incentivou.

A minha namorada Patrícia que soube respeitar os vários dias que deixei de vê-la para desenvolver este trabalho e outros. Pati, o esforço valeu a pena!

A Filipe Niero Felisbino que me ajudou a ser o profissional que hoje sou.

Aos meus colegas de classe que por anos me fizeram companhia e especialmente a Alisson Santos Silva que foi meu melhor amigo na Universidade.

A todos os meus amigos que me ajudaram e deram força em todos os momentos dessa caminhada.

Ao meu orientador, Paulo João Martins que abraçou a minha causa e me orientou para que hoje este trabalho esteja concluído.

E por fim, a todos os professores da Unesc, especialmente a Merisandra Cortês de Mattos que procuraram sempre dar o máximo de seus conhecimentos para nós alunos. O pensamento na próxima página demonstra toda minha gratidão a vocês.

Obrigado a todos!

"Não se pode ensinar tudo a alguém.
Pode-se, apenas, ajudá-lo a
encontrar por si mesmo."

(Galileu Galilei)

RESUMO

Com o surgimento de novas tecnologias as organizações estão cada vez mais preocupadas com o que seus empregados e usuários estão utilizando em seus computadores. O evidente aumento do número de programas maliciosos que se propagam principalmente via Internet e a grande quantidade de programas sem licenciamento colocam as empresas em situações arriscadas. Este trabalho compreendeu o desenvolvimento de um *software* que monitorou as máquinas de uma empresa da região de Criciúma demonstrando quais programas os usuários estavam utilizando durante o dia-a-dia. O estudo foi realizado em um sistema cliente/servidor onde a base servidora foi desenvolvida utilizando a tecnologia Java que é multi-plataforma e pode ser implantado em qualquer sistema operacional e a base monitora foi desenvolvida em C++ que é uma linguagem que pode acessar a API do Windows.

Palavras-chave: Redes Cliente/Servidor, TCP/IP, API do Windows, Processos do Windows, Monitoramento.

ABSTRACT

With the appearance of new technologies, companies are getting more and more concerned about what employees are using in their computers. The obvious raising of pornographic and non-licenced programs, which put companies in risky situations, spread specially over the internet. This work concerned about the development of a software which followed the machines of a company in the region of Criciuma showing what programs had been used every day. The study has been done in a client/provider system where the provider basis was developed using Java technology and can be implanted in any operational system and the monitoring basis was developed in C++ which is a language that can access the Windows API.

Key words: Client/provider net, TCP/IP, Windows API, Windows process, monitoring.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

API	*Application Program Interface*
ARP	*Address Resolution Protocol*
DFD	Diagrama de Fluxo de Dados
DNS	*Domain Name System*
DOD	*Department Of Defense*
GPL	*General Public License*
ICMP	*Internet Control Message Protocol*
IP	*Internet Protocol*
ISO	*International Organization for Standardization*
LAN	*Local Area Network*
MAC	*Media Access Control*
MBPS	Megabytes Por Segundo
PID	*Process Identifier*
SGBD	Sistema Gerenciador de Banco de Dados
SMTP	*Simple Mail Transfer Protocol*
TCP	*Transmission Control Protocol*
TI	Tecnologia da Informação
UDP	*User Datagram Protocol*
UML	*Unified Modeling Language*
WAN	*Wide Area Network*

SUMÁRIO

1 INTRODUÇÃO

Nos dias atuais, as organizações estão cada vez mais preocupadas com o seu maior bem, as informações. Os gerentes sejam eles de qualquer entidade (Empresas, Universidades, entre outros), procuram profissionais capacitados que tenham comprometimento, ética e responsabilidade.

Com a expansão da Internet e a interatividade de aplicativos de *software*, os usuários podem perder horas ininterruptas de trabalho, executando e visualizando diversos serviços não condizentes com o cotidiano organizacional.

A segurança computacional é uma das principais preocupações das organizações, pois simples páginas de Internet podem conter vírus, *trojans* e *spywares* de diferentes funcionalidades, colocando em risco as informações que circulam pela rede.

Definir uma política de *software* para tratar diversos problemas que envolvem uma rede de computadores de uma organização pode ser muito difícil pelo fato de não se ter um estudo detalhado de quais problemas a empresa possui.

Esta pesquisa compreendeu o desenvolvimento de um protótipo onde o sistema recolheu as informações de quais programas um usuário utilizou durante o dia de trabalho nos computadores de uma rede. O protótipo auxiliou a empresa analisada a montar uma política de utilização de *software* aumentando assim a produtividade de seus funcionários, reduzindo o número de programas maliciosos e mantendo uma listagem de aplicativos instalados nas máquinas a fim de controlar o uso de sistemas sem licenças.

1.1 OBJETIVO GERAL

Monitoramento de *softwares* ativados por um usuário no período de utilização de computadores utilizando a API WIN32.

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos desta pesquisa são:

a) monitorar o uso de *software* nos computadores de uma rede;

b) estudar e aplicar características em uma rede cliente/servidor;

c) reduzir a utilização de sistemas que não fazem parte da política de *software* das organizações;

d) coletar informações das máquinas na rede por meio de um aplicativo desenvolvido;

e) desenvolver um estudo estatístico sobre os softwares que o usuário utilizou;

f) pesquisar e aplicar funções da API do Windows;

g) estudar os conceitos de processos do Windows;

h) desenvolver uma política de *software* na empresa analisada.

1.3 JUSTIFICATIVA

Hoje em dia a quantidade de *software* no mercado é muito extensa e a variedade de recursos cresce a cada dia. Há nas organizações (empresas, universidade, entre outros) uma grande preocupação com os programas que seus usuários estão utilizando. Com vários programas de relacionamentos, é possível que um usuário perca várias horas do dia utilizando estes serviços, o que pode infringir a política de utilização de *software*.

Outro grande fator é a crescente utilização de programas maliciosos como vírus, *trojans*, *spywares* e *worms*. Um dos maiores problemas das empresas está no acesso à Internet onde são infectados a maioria dos computadores. A pesquisa TIC Empresas 2006 demonstrou que das 2.437 empresas entrevistadas, 51,65% dos gerentes de informática reclamam de ter problemas com vírus e programas maliciosos (DALMAZO, 2007).

Dentre os problemas que as organizações enfrentam está também à utilização de *softwares* ilegais. Aproximadamente 60% das empresas brasileiras utilizam *softwares* piratas. Mas nem sempre é de conhecimento dos gerentes de TI, já que a facilidade de fazer *download* de *softwares* sem licença torna difícil o controle pela empresa. As multas por *softwares* ilegais podem ter repercussões graves em uma organização. (COMPUTERWORLD, 2007). Em outra pesquisa mostrou-se que 43% do *sites* de *download* de software trazem vírus e programas do gênero junto aos sistemas baixados (IDG NOW, 2007).

O desenvolvimento de um programa de monitoramento de sistemas tem por objetivo auxiliar as organizações a desenvolver uma política de *software* onde os gerentes poderão definir quais programas seus funcionários poderão utilizar, tentando

diminuir a ocorrência de pragas virtuais e mantendo uma listagem de ativos na empresa a fim de controlar os sistemas ilegais.

O sistema foi desenvolvido utilizando a tecnologia cliente/servidor onde a base servidora coletará as informações armazenadas nas máquinas. E a base monitora que está instalada em cada máquina da empresa ficará responsável por gravar as informações dos *softwares* que estão sendo utilizados.

1.4 ESTRUTURA DO TRABALHO

A divisão desta pesquisa é realizada em 8 (oito) capítulos relacionados nas áreas de Redes de Computadores, Desenvolvimento de *Software*, Sistemas Operacionais e Modelo Cliente/Servidor.

No primeiro capítulo é mostrada toda parte introdutória bem como os objetivos e justificativa do trabalho. No próximo capítulo são descritas as características de uma Rede de Computadores abrangendo aspectos de funcionamento e organização de uma rede. O Modelo de Referência OSI e o Modelo TCP/IP são expostos em detalhe nos próximos dois capítulos deste projeto. O desenvolvimento do sistema foi realizado utilizando uma rede Cliente/Servidor sendo relatado no quinto capítulo. A interface de programação do Windows e a utilização de processos são descritos nos dois capítulos subseqüentes. Ao final deste trabalho mostra-se como foi realizada a modelagem do protótipo Monitor no oitavo capítulo.

2 REDES DE COMPUTADORES

É um conjunto de *hosts*[1] autônomos inter-conectados, utilizando um meio de comunicação (TABEMBAUM, 1997).

Em outra breve definição sobre redes de computadores, Scrimger, Lasalle e Parihar (2002) descrevem redes como sendo dois ou mais computadores que compartilham ou trocam informações uns com os outros.

Formada por um conjunto de sistemas conectados, compartilhando recursos e interagindo com outras máquinas utilizando um *link* de comunicação. A definição de redes não se aplica somente à troca de informações, mas também ao compartilhamento de hardware (impressoras, memória, fax modem, correio eletrônico entre outros) e *software*, por meio de um sistema de comunicação (BERG,1998) .

Uma rede de comunicação é uma topologia ligada por meios de transmissão e protocolos que definem a maneira que os dados serão transmitidos (SOARES; LEMOS; COLCHER 1995).

2.1 REDES LOCAIS (LAN)

São grupos de computadores e dispositivos interconectados em uma área geográfica pequena (BERG, 1998).

Geralmente são redes privadas, conectando computadores pessoais e estações de trabalho. Não contém linha dedicada ou privada, toda conectividade é local, normalmente por roteadores (SCRIMGER; LASALLE; PARIHAR, 2002).

[1] É um conjunto de máquinas cuja finalidade é executar os programas. (TANEMBAUM 1997).

São redes rápidas, com alta taxa de transmissão que normalmente trabalham a uma velocidade de 10 Mbps a 100 Mbps. Outra forte característica é a baixa taxa de erros (TANEMBAUM, 1997).

2.2 REDES GEOGRAFICAMENTE DISTRIBUÍDAS

É uma rede de longa distância, que pode alcançar milhares de km. Podem interligar cidades, países e até continentes (BERG, 1998).

Conhecida como WAN sua estrutura é composta de duas características, onde existem as máquinas conectadas, também chamadas de *host* e a sub-rede que conecta essas máquinas entre si (TANEMBAUM, 1997).

Este tipo de conexão é realizado geralmente por companhias de comunicação que utilizam cabos de fibra ótica, comunicações via satélite, dispositivos de alta velocidade e capacidade (LOSHIN, 2003).

3 MODELO DE REFERÊNCIA OSI

Criada pela *International Organization for Standardization* (ISO), o modelo *Open Systems Interconnection* (OSI) foi criado com o objetivo de padronizar a interconexão de sistemas abertos. O modelo proposto pela ISO não impõe nenhuma regra sobre tecnologia ou modelo de conexão, mais sim o reconhecimento e suporte dos padrões implementados pela organização (SOARES; LEMOS; COLCHER, 1995).

Este modelo não define os serviços e os protocolos de cada camada, mas define funções dizendo o que cada camada deve fazer (TANEMBAUM, 1995).

3.1 CAMADA FÍSICA

Também conhecida como camada de hardware, trabalha diretamente com os dados brutos (TORRES, 2001).

É responsável por colocar esses dados na rede sem fazer qualquer tratamento. Outras funções desta camada são: definir a duração de cada *bit*[2], como será feito à conexão, como terminará esta conexão (TANEMBAUM, 1997).

Conforme Soares, Lemos e Colcher (1995), esta camada provê características mecânicas, elétricas, funcionais e de procedimentos, ativando, mantendo e desativando conexões para a transmissão de *bits*.

[2] Unidade de comunicação entre computadores, representado por apenas dois números 0 e 1.

3.2 CAMADA DE ENLACE

Esta camada é responsável por interpretar os dados recebidos da camada física, detectando possíveis erros e corrigindo algoritmos. Caso ocorra problema envia uma mensagem para a anterior requisitando o reenvio das informações (DULANEY, et al. 1998).

É de sua responsabilidade a detecção de erros, fornecendo assim um tráfego mais seguro entre as camadas adjacentes (TANEMBAUM, 1997).

Conforme Soares, Lemos e Colcher (1995) dentre outras características do está o controle de fluxo, onde o remetente conhecerá o máximo de dados que o destinatário poderá receber evitando assim que o *buffer* receba mais informações do que possa armazenar.

3.3 CAMADA DE REDE

Conforme Soares, Lemos e Colcher (1995) esta camada provê meios funcionais e procedurais para a transmissão de dados orientados ou não a conexão entre entidades do nível de transporte. Tendo-se citados dois tipos de serviços da camada de rede:

a) serviço orientado a conexão: é um serviço confiável, pois oferece um controle de fluxo de erros. Entre a origem e o destino dos dados é criado um caminho por onde é feito à comunicação.

b) serviço sem conexão: utiliza algoritmo de roteamento. Como cada mensagem é entregue de forma individual não tem garantias de que os pacotes irão chegar ao seu destino.

É de responsabilidade desta camada determinar o endereçamento e as rotas, fazendo controle de congestionamento, ordenando o caminho de cada pacote, caso haja várias rotas de tráfego (TORRES, 2001).

3.4 CAMADA DE TRANSPORTE

Uma das funções desta camada é dividir os dados em pacotes, para serem transmitidos para a camada de rede (TORRES, 2001).

É de sua responsabilidade tratar a transferência dos dados e verificar possíveis erros, como perda de pacotes e a ordenação dos dados enviados. Dois protocolos são utilizados nesta camada o *Transmission Control Protocol* (TCP) e o *User Datagram Protocol* (UDP) (SCRIMGER; LASALLE; PARIHAR, 2002).

Outra característica é o controle de fluxo, onde os dados são analisados para que o transmissor não envie mais pacotes do que o receptor consiga receber ou tratar (TANEMBAUM, 1997).

3.5 CAMADA DE SESSÃO

É de responsabilidade desta camada gerenciar as conexões entre as máquinas que estão se comunicando (DULANEY, et al. 1998).

Esta camada utiliza uma conexão virtual, onde são enviadas ao aplicativo emitente quais informações foram recebidas. Caso haja uma falha os dados serão retransmitidos (SCRIMGER; LASALLE; PARIHAR 2002).

3.6 CAMADA DE APRESENTAÇÃO

A principal função da camada de apresentação é fazer a conversão entre os protocolos, para que os mesmos se comuniquem. Dentre outros serviços oferecidos por esta camada estão: a conversão e a compressão dos dados, a manutenção de conexões de apresentação, criptografia e redirecionamento de rede (SCRIMGER; LASALLE; PARIHAR, 2002).

3.7 CAMADA DE APLICAÇÃO

Sua principal função é prover serviços de redes para aplicativos, trabalhando diretamente com o usuário final.

Alguns serviços executados pela camada de aplicação estão: transferência de arquivos, serviço de correio eletrônico, conversão de nomes, entre outros.

4 PROTOCOLO TCP/IP

Criado pelo *Department of Defense* (DOD) dos EUA, sua finalidade era que todos os computadores interconectados por esse sistema pudessem se comunicar. Tudo começou quando o DOD criou um projeto chamado *Advanced Research Projects Agency* (ARPANET), que com o passar do tempo diferentes instituições foram se conectando a essa rede, o que tornou o sistema inviável, pelos diferentes problemas que estavam ocorrendo devido a diferentes tipos de protocolos (NAUGLE, 1998).

O modelo TCP/IP é o padrão mundial de interconexão de sistemas abertos (COMER; STEVENS, 1999).

Conforme Casad e Willsey (1999) definem que o TCP/IP é um conjunto de protocolos que dão suporte a comunicação em rede.

Dentre algumas características de modelo está o protocolo padrão aberto desenvolvido independentemente de uma especificação de hardware ou sistema operacional (HUNT, 1997).

A Figura 1 demonstra a diferença entre o modelo de referência OSI e o modelo TCP/IP, demonstrando a diferença entre elas:

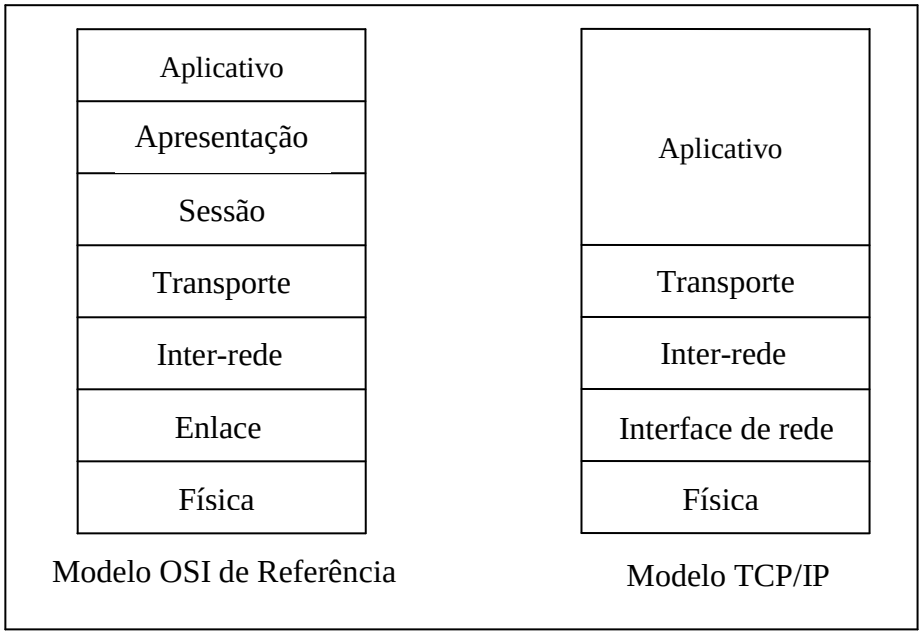

Figura 1**.** Diferença entre camada OSI e TCP/IP
Fonte: SCRIMGER, R.; LASALLE, P.; PARIHAR, M. (2002)

4.1 INTERFACE DE REDE

Dentre as camadas do TCP/IP a interface de rede é a mais baixa. Ela trabalha basicamente enviando e recebendo dados (COMER, 1999).

Conforme Scrimger, Lasalle e Parihar (2002), são listadas algumas características dessa camada:

a) exerce controle sobre o hardware da rede. Exemplo: placa de rede, *hub*, pontes entre outros;
b) mapear e converter endereços IPs em endereços de rede local;
c) controle de fluxo de dados;
d) encapsular e transmitir pacotes que saem e aceitar pacotes que chegam.

4.1.1 Controle de Fluxo

Conforme Scrimger, Lasalle e Parihar (2002), em uma transmissão de dados, existem dispositivos que são mais rápidos que outros. Se a taxa de transmissão de dados do remetente é maior do que a do receptor pode ocorrer um sobrecarga de memória causando erros nesta transmissão. Para tratar deste problema foi criado o controle de fluxo, onde o dispositivo que recebe os dados informa ao remetente se é capaz de suportar a taxa de transferência. Existem dois modelos de controle de fluxo:

a) taxa de controle de fluxo garantida: acontece quando dois dispositivos traçam uma taxa de transmissão onde tanto o transmissor quanto o receptor conseguirão entender.

b) controle de fluxo baseado em janela: sua principal característica é a utilização de um *buffer*, onde o remetente e receptor combinam a quantidade de quadros de dados que serão enviados.

4.2 CAMADA DE INTER-REDE

Camada responsável por transmitir os pacotes na rede, garantindo a chegada no seu destino, não importando a ordem (SOARES; LEMOS; COLCHER, 1995).

Conforme Scrimger, Lasalle e Parihar (2002), esta camada utiliza os endereços IP para a transmissão de dados. Utiliza técnicas de comutação, realizando uma entrega mais rápida.

a) comutação por circuito: utiliza um canal de comunicação dedicado;

b) comutação por mensagem: é realizada uma divisão das mensagens e cada parte transporta consigo as informações de seu endereço de destino;

c) comutação por pacotes: utilizada a divisão em pacotes, onde cada segmento contém as informações do remetente e do receptor.

4.3 CAMADA DE TRANSPORTE

Esta camada tem a função de transportar os dados do remetente ao destinatário (COMER, 1999).

Conforme Tanembaum (1997) nesta camada existem dois protocolos que têm por objetivo prover comunicação, o TCP e o UDP.

a) tcp: é um protocolo de conexão confiável, pois garante a entrega dos dados.
b) udp: tem por sua característica ser mais rápido, mais não garante a entrega íntegra dos dados.

Conforme Scrimger, Lasalle e Parihar (2002), descrevem dois serviços da camada de transporte:

a) conexão orientada: é um serviço onde o remetente envia os dados para o destino, ao recebê-los de forma exata ele envia de volta uma confirmação informando que os dados chegaram corretamente, mais

caso sejam perdidos ou os dados cheguem de forma errada esta confirmação não é enviada e é feito o reenvio dos mesmos pacotes.

b) sem conexão: o receptor não tem responsabilidade de emitir uma resposta para os dados enviados. Caso algum dado seja perdido ou aconteça algum erro o remetente não receberá nenhuma informação e os pacotes não serão retransmitidos.

4.4 CAMADA DE APLICATIVO

Diferentemente do modelo OSI, o TCP/IP não possui as camadas de sessão e de apresentação. A camada de aplicativo interage diretamente com o usuário e com a camada de transporte enviando e recebendo dados (COMER, 1999).

Ela trabalha com os protocolos de alto nível, oferecendo serviços de transferência de arquivos, serviços de rede e de mensagens, bancos de dados, terminais virtuais, entre outros.

5 REDE CLIENTE X SERVIDOR

Scrimger, Lasalle e Parihar (2002) definem uma rede cliente/servidor, como sendo o cliente a entidade que solicita um serviço e o servidor a entidade que provê serviços.

A comunicação se dá por meio de uma mensagem onde o cliente faz uma requisição ao servidor, pedindo que alguma tarefa seja executada. Ao processar as informações é retornada a mensagem ao cliente que a solicitou (TANEMBAUM, 1997).

Para o funcionamento de uma rede cliente/servidor deve ocorrer uma comunicação constante e assíncrona entre os dois programas. Caso o servidor possua algum problema toda à comunicação ficará comprometida. É ele quem provê as informações aos clientes da rede (SOARES; LEMOS; COLCHER, 1995).

Alguns tipos de servidores incluem serviços de arquivos, *e-mail*, impressão, fax entre outros. Geralmente são computadores de alto desempenho que possuem mais memória, maior espaço em disco, e maior capacidade de processamento (BERG, 1998).

A Figura 2 demonstra o funcionamento de uma rede do tipo cliente/servidor, onde os três menores computadores à direita da imagem representam os clientes que requisitam uma informação por meio das linhas pontilhadas e o servidor a esquerda retorna as informações ou uma resposta das solicitações.

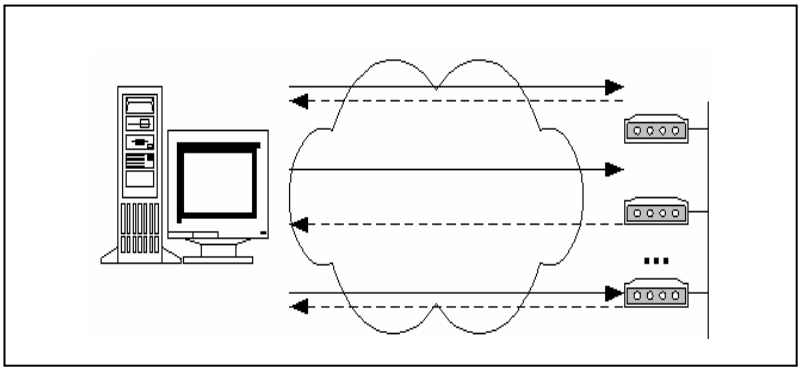

Figura 2. Representação de funcionamento de uma rede Cliente/Servidor

5.1 CARACTERÍSTICAS

Torres (2001) define uma rede cliente/servidor com algumas características:

a) podem operar em diferentes plataformas;
b) possibilita compartilhar recursos tanto de hardware quanto de *software*;
c) um servidor pode atender a vários clientes ao mesmo tempo;
d) alta segurança, pois os dados ficam centralizados;
e) maior desempenho que uma rede ponto-a-ponto;
f) custo maior do que uma rede ponto-a-ponto.

5.2 ESCALABILIDADE

Ao crescer o número de usuários e o volume de dados transferidos pela rede, podem-se adicionar mais processadores ou qualquer outro hardware, para que se tenha um aumento de desempenho nos sistemas.

5.3 CENTRALIZAÇÃO DOS DADOS

Conforme Torres (2001) os dados ficam localizados em um servidor, o que os torna mais organizados, melhorando também a segurança dos dados. Facilita a realização de *back-ups* dos dados.

5.4 TIPOS DE SERVIDORES

Existem vários tipos de servidores no mercado, cada um contendo diferentes características e funcionalidades, sendo listados dois tipos deles.

5.4.1 Servidor de Arquivos

Segundo Soares, Lemos e Colcher (2000), a função do servidor de arquivo é oferecer aos clientes serviços de armazenamento e acesso às informações e ainda compartilhamento de discos. São responsáveis por controlar as unidades de discos, podem aceitar ou não pedidos de transações com os seus clientes. Possuem um sistema de segurança que por meio de uma palavra chave, pode-se decidir se os arquivos e diretórios existentes têm acesso público, privado ou por grupo de usuários. Com esses acessos podem-se ter a atualização dos arquivos existentes, garantindo a integridade dos dados. Neste tipo de ambiente, arquivos de diferentes sistemas operacionais devem ser aceitos nas transações já que em um sistema cliente/servidor a independência de plataforma deve acontecer.

5.4.2 Servidor de Banco de Dados

Conforme Soares, Lemos e Colcher (2000) geralmente esse tipo de servidor contém um sistema gerenciador de banco de dados (SGBD), onde parte de sua função é de armazenar os dados, assim como processar consultas. Esses tipos de servidores centralizam as funções de controle de concorrência, manutenção de consistência, aumentando desempenho da rede e das aplicações.

Isso ocorre porque o processamento é realizado no servidor onde o hardware tem maior poder de processamento do que seus clientes.

Neste modelo o cliente é responsável por obter do servidor os objetos necessários, executar os comandos recebidos da aplicação e o servidor é quem processa as consultas e atualizações no banco de dados além de gerenciar as transações enviadas.

6 API WIN 32

É o nome que se dá à interface de programação de aplicativo do Windows. Sempre que uma função da API for utilizada diz-se que este programa é um sistema WIN32. Ela é composta por inúmeras funções e serviços que podem requisitar operações pré-definidas pelo sistema operacional (RICHTER, 1996).

A idéia inicial da Microsoft era desenvolver funções que fossem padrão para todas as distribuições dos seus sistemas operacionais. Mas, a cada nova distribuição do Windows a interface de programação foi sofrendo algumas alterações (COWART, 1993).

Richter (1996) afirma que a interface de programação do Windows trabalha com vários componentes dentro do sistema operacional dentre eles:

a) arquivos;
b) comunicações;
c) consoles;
d) dispositivos de entrada e saída;
e) dll’s;
f) impressão;
g) informações de sistemas;
h) processos;
i) serviços.

6.1 CARACTERÍSTICAS DA API WIN32

Hollingworth, et al. (2001) descrevem que a API do Windows é agrupada em algumas áreas:

a) gerência do windows;
b) sistema de serviços;
c) interface gráfica;
d) serviços de multimídia;
e) controles e diálogos;
f) características do *Shell*;
g) características internacionais;

6.1.1 Gerência do Windows

Conforme Petroutsos (1999) o Windows dispõe de funções para criação e gerenciamento de suas aplicações, trazidas pela biblioteca USER32.DLL. Todas as entradas e saídas do sistema passam por esta gerência.

Dentre todos os serviços disponíveis estão janelas, menus, caixas de diálogos, mouse, teclado entre outros. Estas funções de gerência são utilizadas para controlar o modo como serão disponibilizadas e criadas as janelas e usadas pelas aplicações. Existem aproximadamente 648 funções na biblioteca que controla as funções de gerência (HOLLINGWORTH, et al. 2001).

6.1.2 Sistema de Serviços

A API do Windows dispõe de vários serviços onde os aplicativos podem gerenciar e monitorar recursos do sistema, como acesso a arquivos, dispositivos de entrada e saída, memória, multithread, recursos multitarefa e tratamento de erros. A biblioteca responsável por disponibilizar estes recursos é a KERNEL32.DLL. São aproximadamente 475 serviços trazidos pelo sistema de serviços (HOLLINGWORTH, et al. 2001).

Pode-se listar alguns serviços disponibilizados pelo Windows por meio desta biblioteca:

a) comunicação: trabalha com a comunicação entre portas seriais, paralelas e modems;
b) entrada e saída: comunicação entre dispositivos de aplicações;
c) dll: suporte a criação de DLLs em tempo de execução;
d) processos e *threads*: suporte a multitarefas, gerência e criação de múltiplas *threads* e processos.

6.1.3 Interface Gráfica

O Windows dispõe de funções para gerenciamento de interfaces gráficas, por meio da biblioteca GDI32.DLL, onde incluem desenhos de linhas, textos, fontes e cores. Nesta DLL está incluído um serviço chamado de dispositivo de contexto (DC). Onde o DC é uma estrutura de dados usada para criar objetos gráficos e modelos de saída (HOLLINGWORTH et al. 2001).

6.1.4 Serviços de multimídia

Existem três bibliotecas que compõem estes serviços. Uma chamada de *Multimedia System Library*, a MMSYSTEM.DLL que provê serviços de multimídia, a segunda é a *Microsoft for Video Windows Library* que trabalha com vídeos por meio da MSVFW32.DLL e por fim o *Microsoft Áudio Compression Manager Library* que implementa serviços de áudio por meio da biblioteca MSACM32.DLL (HOLLINGWORTH, et al. 2001).

6.1.5 Diálogos e Controles

As coleções de janelas existentes no sistema operacional Windows são controladas pela biblioteca COMCTL32.DLL. As caixas de diálogos são providas pelo arquivo COMDLG32.DLL. A idéia da Microsoft era disponibilizar controles, janelas e diálogos para permitir aos programadores e desenvolvedores utilizarem interfaces gráficas. (SWART, et al. 2003).

6.1.6 Características do Shell

Conforme Swart, et al. (2003) o nome *Shell* no Windows significa que uma aplicação permite que um usuário inicie e controle de outras aplicações e arquivos. Nas versões mais atuais do Windows existem aproximadamente 120 rotinas *Shell*.

Estas características são providas pela biblioteca SHELL32.DLL. Existem duas descrições das características da *Shell*:

a) *drag-and-drop*: Permite o acesso a múltiplos arquivos do Windows Explorer;
b) associação de arquivos: Permite iniciar e buscar diferentes tipos de arquivos.

6.1.7 Características Internacionais

A interface de programação do Windows suporta múltiplas linguagens. O sistema operacional da Microsoft possui suporte a UNICODE e a símbolos. Contêm ainda a *National Language Support* (NLS), onde esta função ajuda uma aplicação a especificar vários tipos de idiomas (SWART, et al. 2003).

6.2 API WIN32S

É um conjunto de DLLs e um *driver* de dispositivo virtual que incluí a WIN 32 no Windows 3.x possibilitando que fossem criados e executados programas 32 *bits* em sistema de 16 *bits* (RICHTER, 1996).

Mas existiam muitas inconveniências já que sistema 16 *bits* não possuem suporte a *multithreading*, o que acarretava que algumas funções da WIN32 não tivessem nenhuma funcionalidade nestes sistemas operacionais (SIMON, 1998).

Richter (1996) cita o exemplo da função CreateThread() que nesta distribuição não realiza nada mais do que retornar NULL, já que sistemas 16 *bits* não

tem suporte a *threads* de processos. Este processo de conversão de 16 *bits* para 32 *bits* chama-se *thunking*.

6.2.1 Thunking

Permite que aplicações 32 *bits* executem em plataforma 16 *bits* e vice e versa. Isto é feito por meio de um processo chamado *bitness.* O *bitness* irá informar quais instruções serão codificadas pelo processador. (DABAK; PHADKE; BORATE, 1999).

Existem duas características para este modelo:

a) *universal*: permitem a chamada de uma função em 32 *bits* em um sistema 16 *bits*.

b) *generic*: permitem a chamada de uma função 16 *bits* para um sistema 32 *bits*.

Os sistemas operacionais Windows 95 e 98 permitem a chamada de *thunking* universal e genérico enquanto o Windows NT permite apenas a solicitação do tipo universal.

6.3 MICROSOFT WINDOWS 95

A API WIN32 possui três principais DLLs que fazem parte da interface de programação que são a KERNEL32, USER32 E GDI32 (DABAK, PHADKE E BOKATE, 2002).

Esta distribuição traz a interface de programação WIN32 deixando assim a versão WIN32S fora de uso, mas não oferece todas as funções da API WIN32 como no

Windows NT. As funções são restritas, pois a quantidade de memória que este sistema suporta, retirou muitas funcionalidades como de entrada e saída assíncronas, funções de depuração, registro e questões de segurança (RICHTER, 1996).

6.4 MICROSOFT WINDOWS NT

Foi à segunda plataforma do Windows a trazer a WIN32. Este sistema operacional diferencia-se dos outros sistemas da Microsoft por não herdar nada do MS-DOS (RICHTER 1996).

A API WIN32 deste sistema operacional provê seus serviços por meio de três DLLs iguais as do Windows 95. O NT trabalha com três categorias que são serviços básicos, desenhos, janelas e mensagens. São executadas via KERNEL32.DLL e incluem serviços de I/O, *threads* e gerência de memória. Os desenhos são executados pela GDI32.DLL. As janelas e mensagens são apresentadas pela USER32.DLL (DABAK; PHADKE; BORATE, 1999).

6.5 IMPLEMENTAÇÕES DA WIN32 API

O Windows NT, 95/98 trabalham com quatro *gigabytes* (Gb) de endereçamento virtual. Onde dois *gigabytes* referem-se ao espaço de endereço compartilhado que é reservado para usar operações de sistema e outros dois *gigabytes* são atribuídos ao espaço de endereçamento privado que é utilizado para executar os processos criados. No Windows 95 e 98 os sistemas operam com bibliotecas dinâmicas residentes no espaço de endereçamento compartilhado. Por operar neste espaço compartilhado podem ocorrer de uma aplicação sobrescrever outra que está na memória

afetando o bom funcionamento dos processos. No Windows NT estas DLLs carregam os processos no espaço de endereçamento privado. Sendo assim se algum processo for sobrescrito será afetado somente o processo que chamou, não apresentado problema aos outros (DABAK; PHADKE; BORATE, 1999).

7 PROCESSOS

Um processo é uma abstração usada pelo *Kernel* para representar um programa em memória. Contém informações sobre *threads* associadas aos processos (VILLANI, 2001).

Um processo é uma instância de um programa que está execução. Por exemplo, quando iniciamos o *notepad.exe* é criado uma instância deste programa na memória. No WIN32 cada processo tem seu espaço de endereçamento, além deste os processos possuem certos recursos de arquivos, *threads*, e alocações dinâmicas de memória. Sempre que um processo é criado, inicializa-se também uma *thread*, que são caminhos de execução dentro de um processo. Quando elas são iniciadas, novas *threads* de execução podem ser criadas, propiciando que várias delas sejam executadas simultaneamente no espaço de endereçamento. No momento que o processo não possuir mais nenhuma *thread* o mesmo será destruído, mas enquanto estiverem executando o processo estará ativo. O que faz o controle dos processos do Windows são os objetos *Kernel* que também possuem as funcionalidades de controlar *threads*, arquivos entre outros (RICHTER, 1999).

Conforme Oliveira, Carissimi e Toscani, (2004), as *threads* normalmente possuem os seguintes estados:

a) apto: quando uma *thread* se encontra nesse estado, ela está hábil a executar suas tarefas. Quem determina o processo que receberá o estado ativo é o escalonador;
b) ativo: é o estado que o processo aguarda para entrar em execução. No momento que todos os recursos estiverem prontos ele passará para o próximo estado;
c) em execução: estado em que o processo já está realizando suas instruções. Ele executará até que suas tarefas terminem ou outra requisição de parada seja efetuada;
d) espera: estado que poderá ser acionado por um evento ou quando um subsistema ordenar que o processo aguarde para reiniciar. O processo entra na fila de espera até que o estado Apto seja aplicado a ele;
e) transição: estágio anterior ao Apto, neste estado o processo já está pronto para executar mais os recursos de sistema ainda não estão disponíveis;
f) término: estado que um processo assume quando ele é finalizado. Ele pode ser encerrado por outro processo, por uma *thread* ou quando encerra sua tarefa.

A Figura 3 demonstra os estados de um processo desde seu início até o seu fim:

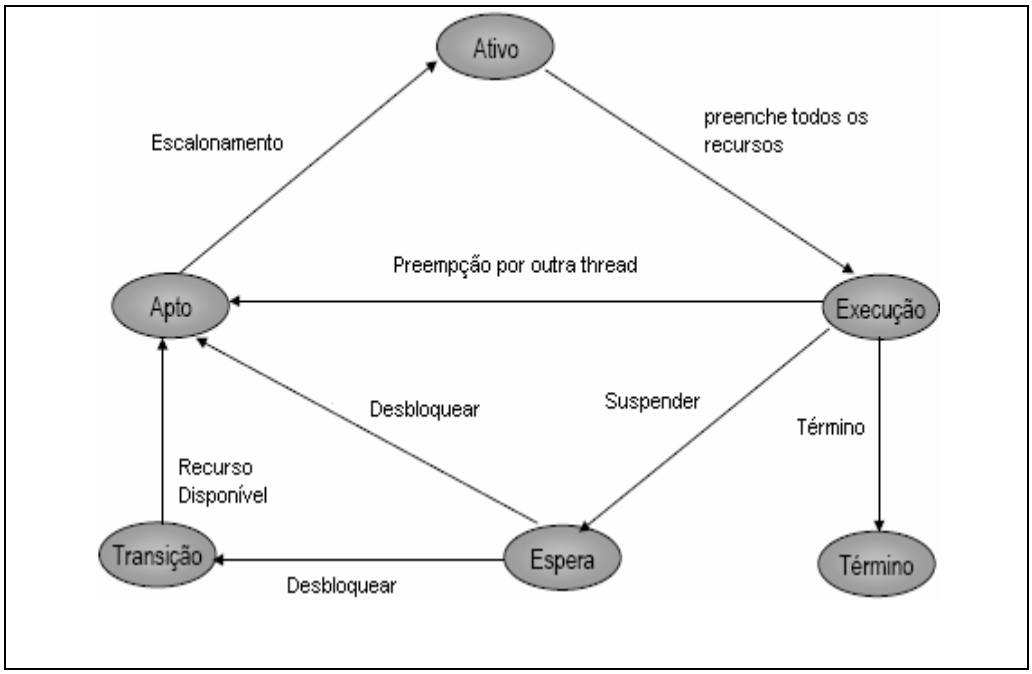

Figura 3. Demonstração do ciclo de vida de um processo
Fonte: OLIVEIRA, R.; CARISSIMI, A.; E TOSCANI, S.; (2004).

Cada versão do Windows tem sua maneira de executar processos, seja ele em fluxo dos dados ou forma de relacionamento uns com os outros, mas geralmente utiliza o conceito de “Processo Objeto” que são os recursos de sistema como memória, arquivos, e os objetos *thread* que são unidades de trabalho que pode parar de executar o sistema a qualquer momento (OLIVEIRA; CARISSIMI; TOSCANI, 2004).

O *Process Identifier* (PID) é um identificador único que cada processo recebe ao ser iniciado. Nunca poderão existir dois processos com o mesmo PID executando ao mesmo tempo, mas quando um processo é finalizado o mesmo identificador poderá ser utilizado por outro (SCHREIBER, 2001).

Os processos são objetos que possuem todos os recursos de uma aplicação. O fato de o Windows NT poder trabalhar com múltiplos processadores torna-o mais poderoso que o Windows 95, em relação à capacidade de trabalhar com processos e

threads. Quanto mais processadores mais fatias de tempo terão as *threads* para executarem. (SIMON, 1998).

7.1 TRABALHANDO COM PROCESSOS

Segundo Richter (1996) aplicações WIN32 suportam dois tipos de aplicações, baseadas em consoles e em interface gráfica. As aplicações baseadas em console se parecem com aplicativos do MS-DOS, já as que fundamentam-se em interface gráfica criam janelas com mensagens, caixas de diálogo entre outros. Sempre que um sistema for criado para executar elementos gráficos ele deverá começar pela função WinMain, já nos aplicativos em console começam apenas por *main*.

Função que inicia o aplicativo WIN32 baseado em GUI:

```
int WINAPI WinMain(HISNTANCE hinstExe, HISNTANCE hinstPrev, LPSTR lpszCmdLine, int nCmdShow);
```

Quando um arquivo .EXE é carregado no espaço de endereçamento de um processo o valor da instância criada é passado no primeiro parâmetro da função WinMain ou seja hinstExe. Este parâmetro é chamado de manipulador de instância e o valor passado por ele indica onde o sistema carregou este arquivo executável.

Na WIN32 a função CreateProcess(), cria um novo processo, onde uma *thread* primária também é criada, a partir desta pode ser criada outras novas. O valor retornado em caso de executada com sucesso é um número diferente de 0 (zero), caso haja falha o valor será 0 (zero).

Chamada da função CreateProcess:

```
BOOL WINAPI CreateProcess(
  LPCTSTR lpApplicationName,
  LPTSTR lpCommandLine,
  LPSECURITY_ATTRIBUTES lpProcessAttributes,
  LPSECURITY_ATTRIBUTES lpThreadAttributes,
  BOOL bInheritHandles,
  DWORD dwCreationFlags,
  LPVOID lpEnvironment,
  LPCTSTR lpCurrentDirectory,
  LPSTARTUPINFO lpStartupInfo,
  LPPROCESS_INFORMATION lpProcessInformation
);
```

Conforme Russinovich e Solomon (2004) sempre que a função CreateProcess() é chamada, são seguidos alguns passos para a criação de um processo. Esses passos são:

a) criar uma imagem do arquivo com extensão .exe que está sendo executado;
b) iniciar a janela do objeto executado;
c) criar uma *thread* inicial;
d) notificar o subsistema do Windows sobre o novo processo;
e) começar a execução da *thread* inicial;
f) completar a inicialização do espaço de endereçamento e iniciar a execução do programa.

Existem duas maneiras para terminar um processo, a função ExitProcess() e TerminateProcess. A função ExitProcess tem apenas um parâmetro que indica o código de saída do processo. Esta função não retorna valor algum, pois a única função dela é finalizar o processo.

Exemplo de chamada de função ExitProcess:

```
VOID WINAPI ExitProcess(
  UINT uExitCode
);
```

A função TerminateProcess() também encerra um processo mas a função ExitProcess() avisa a DLL responsável pelo processo que o mesmo foi finalizado já a função TerminateProcess() não informa as DLLs responsáveis pelo processo que ele foi terminado fazendo então as bibliotecas entenderem que o processo continua ativo o que pode ocasionar problemas na aplicação.

Exemplo de chamada de função TerminateProcess:

```
BOOL WINAPI TerminateProcess(
  HANDLE hProcess,
  UINT uExitCode
);
```

Caso haja a intenção de criar *threads*, a função chamada deve ser a CreateThread(), quando um processo é criado automaticamente uma *thread* primária é iniciada, mas novas *threads* poderão ser criadas, para que várias funções possam ser executadas simultaneamente.

Exemplo de chamada de função CreateThread:

```
HANDLE WINAPI CreateThread(
  LPSECURITY_ATTRIBUTES lpThreadAttributes,
  SIZE_T dwStackSize,
  LPTHREAD_START_ROUTINE lpStartAddress,
  LPVOID lpParameter,
  DWORD dwCreationFlags,
  LPDWORD lpThreadId
);
```

Para terminar com uma *thread*, a função ExitThread() deve ser chamada. Assim como a função ExitProcess() ela tem apenas um parâmetro onde é passado apenas o código de saída da *thread*. Esta função não retorna nenhum valor.

Exemplo de chamada de função ExitThread:

```
VOID WINAPI ExitThread(
  DWORD dwExitCode
);
```

8 MODELAGEM DO PROTÓTIPO

A modelagem do programa começou a ser definida analisando-se qual o problema que se pretendia resolver. Construir um sistema que monitorasse o uso de *software* nos computadores de uma rede era até então um assunto desconhecido por não se encontrar referências de como construir a ferramenta.

Em um primeiro estágio de estudos, pesquisou-se como seria o ponto de partida para o desenvolvimento do sistema. Logo após a esses estudos iniciou-se a modelagem de como seria construído o programa.

8.1 DIAGRAMA DE ATIVIDADES

Conforme Alhir (2003) um diagrama comunica-se com outro por meio de uma associação de objetos.

Este tipo de diagrama permite especificar como o sistema irá realizar seus objetivos. Ele demonstra as ações representando processos em um sistema (HAMILTON; MILES, 2006).

O diagrama de atividades do protótipo Monitor é mostrado na Figura 4:

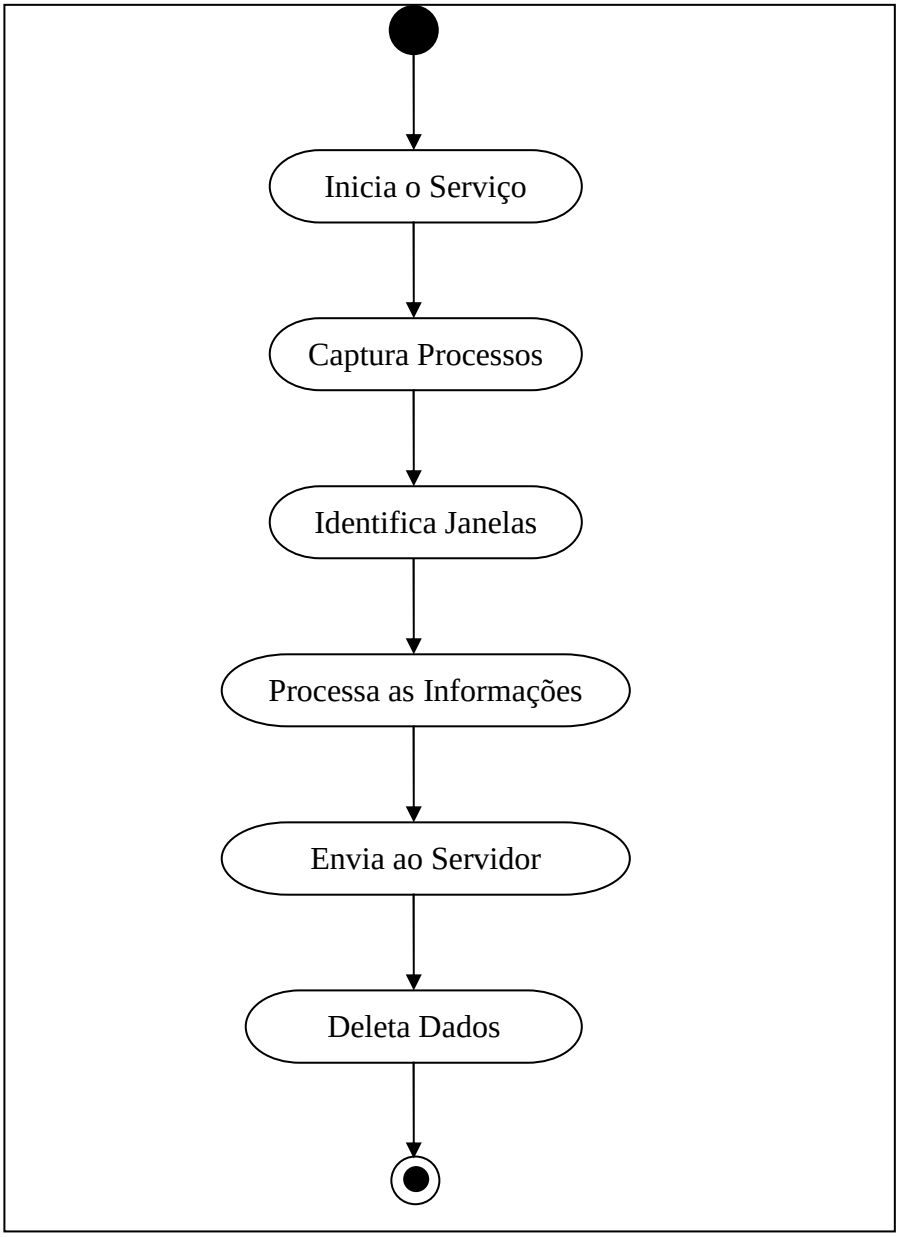

Figura 4. Diagrama de atividades do protótipo Monitor.

A primeira atividade é o início do serviço, configurando o sistema junto ao sistema operacional. Ao iniciar o Windows, o sistema automaticamente será executado e já estará apto a coletar as informações.

A busca dos dados é realizada na segunda etapa, onde são mapeados os processos e são passados para a próxima atividade o PID de cada processo onde será

analisado se ele é ou não um sistema. Logo após são processadas as informações que depois são enviadas ao servidor. Ao enviar as informações ao servidor, os dados são apagados do banco de dados da base monitora a fim de não haver redundância em uma nova captura de dados.

8.2 DIAGRAMA DE CASO DE USO

Conforme Hamilton e Miles (2006) este diagrama captura algumas funcionalidades que o sistema provê. Ele utiliza atores, que são desenhos de bonecos esquemáticos (*stick man*) ou uma caixa com o nome apropriado. A Figura 5 demonstra como é o desenho dos atores no diagrama de Caso de Uso.

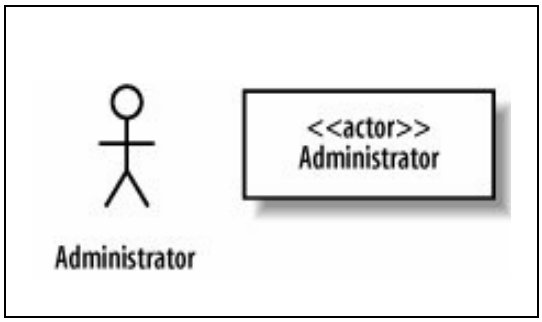

Figura 5. Atores no Diagrama de Caso de Uso.
Fonte: HAMILTON; K. e MILES; R. (2006).

Já os Casos de Uso são desenhos em forma oval com a descrição da funcionalidade em seu interior. A ligação entre o Ator e o desenho Oval é feita por meio de linhas. A Figura 6 demonstra como é representação uma linha e o Caso de Uso neste tipo de diagrama.

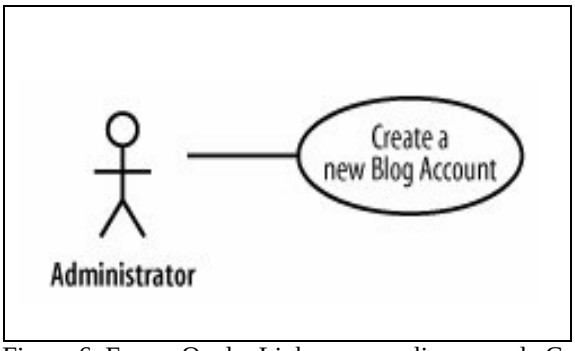

Figura 6. Forma Oval e Linha em um diagrama de Caso de Uso.
Fonte: HAMILTON; K. e MILES; R. (2006).

A representação do Diagrama de Caso de Uso no sistema Monitor é mostrada na Figura 7. Onde o administrador do sistema terá a tarefa de instalar e iniciar o serviço do sistema.

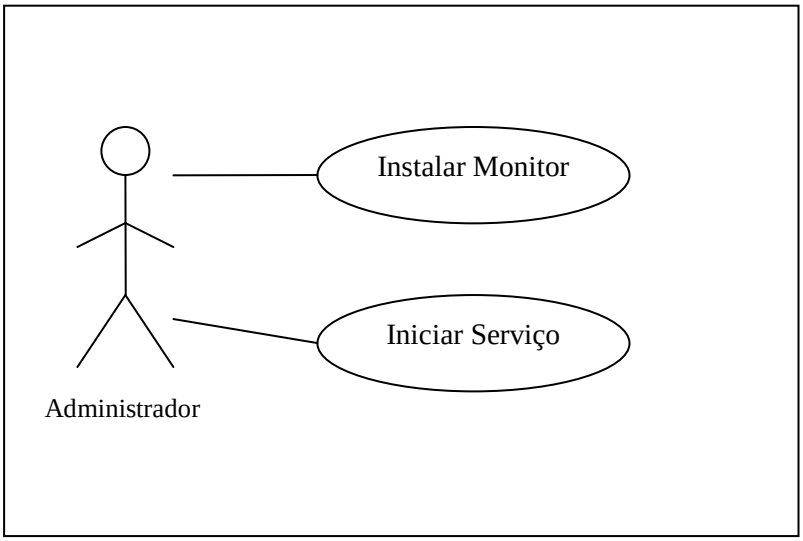

Figura 7. Diagrama de Use Case do Protótipo Monitor.

O diagrama de Caso de Uso do Servidor é representado na Figura 8. Onde o administrador iniciará o sistema para que uma vez ao dia o servidor busque os dados em cada máquina da rede. Logo após o administrador poderá analisar os dados coletados nas máquinas e fazer um levantamento estatístico de quais são os *softwares* mais utilizados e quais podem causar algum problema para a organização.

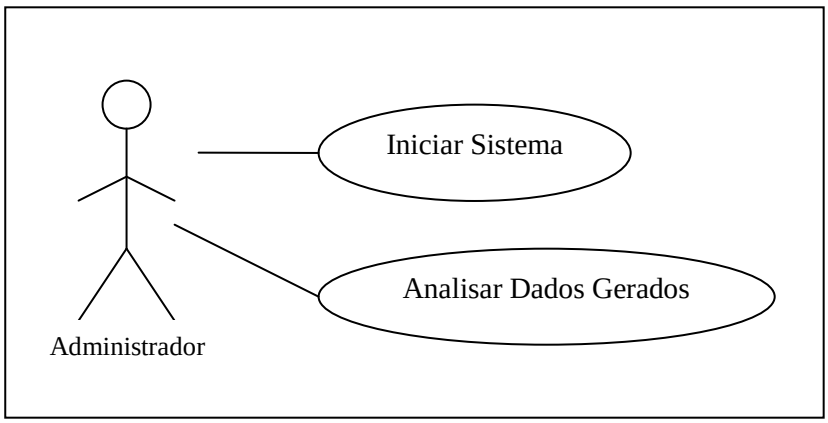

Figura 8. Diagrama de Use Case do Servidor.

8.3 DIAGRAMA DE FLUXO DE DADOS (DFD)

É um modelo que mostra como os dados serão processados dentro do sistema, demonstrando o comportamento do *software* em relação a eventuais comandos externos e internos.

Os desenhos dos processos são representados da seguinte maneira:

a) entidade externa: é representado por um retângulo, com a inscrição do meio externo em seu interior;
b) depósito dos dados: é representado por dois traços paralelos entre si e na forma vertical. Entre os dois traços é incluída a descrição do repositório;

c) processo: é representado por um símbolo oval e sua descrição deve ser apresentada em seu interior;

d) fluxo: representado o fluxo dos dados por meio de uma seta. Mostra a direção em que os dados seguirão.

A Figura 9 representa o diagrama DFD do protótipo Monitor, mostrando a direção dos dados no lado cliente e no servidor.

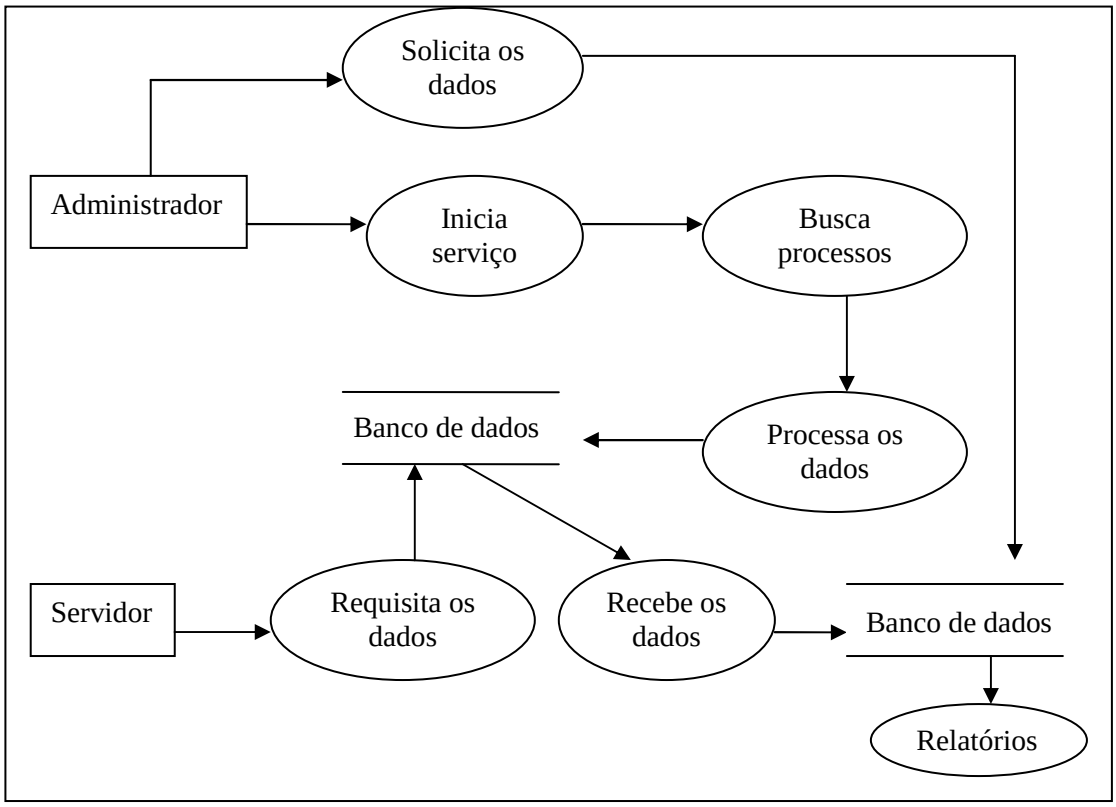

Figura 9. Representação do diagrama DFD do protótipo Monitor.

9 IMPLEMENTAÇÃO DO PROTÓTIPO

O sistema Monitor foi desenvolvido para trabalhar em uma rede cliente/servidor. O cliente foi instalado em cada máquina da rede e permaneceu executando durante todo o tempo que os computadores estiveram ligados. Foram analisadas quatro máquinas onde três delas possuem o sistema operacional Windows XP Professional e uma que continha o sistema Windows 2003 Server que além de ser monitorada também exercia a função de servidor para o protótipo desenvolvido.

9.1 MONITOR

Por meio de chamadas a API do Windows o protótipo busca as informações em processos abertos.

O Monitor foi desenvolvido utilizando a linguagem de programação C++, uma linguagem que obtém informações diretamente na interface de programação do Windows. As informações armazenadas no banco de dados do Monitor são:

a) nome do aplicativo;

b) ip da computador;

c) nome do usuário da máquina;

d) tempo em que o *software* ficou aberto;

e) tempo que o *software* ficou como janela principal;

f) título da janela.

Foi utilizada para comunicação em rede uma biblioteca *General Public License* (GPL) que tem seu código aberto podendo ser alterado à medida da necessidade

do desenvolvedor. Esta biblioteca esta disponível no site: http://www.alhem.net/Sockets/.

O banco de dados utilizado foi o SQLite, criado em tempo de execução, ou seja sempre que o sistema for iniciado pela primeira vez o arquivo de banco de dados será gerado.

Este banco é uma biblioteca em C que possui em banco de dados embutido. A opção de utilizá-lo foi que além de ser livre e multi-plataforma é armazenado em apenas um arquivo, não necessitando ser instalado e nem configurado.

9.2 SERVIDOR

A parte servidora do sistema foi desenvolvida na linguagem Java, que foi escolhida por ser multi-plataforma. Em caso de sistemas operacionais diferentes o servidor trabalhará sem nenhum problema. Por meio de Sockets, o servidor requisita as informações às máquinas onde o Monitor está instalado e recebe uma resposta com as informações armazenados no banco de dados. Após coletar as informações os dados na base monitora são apagados para que na próxima busca, dados redundantes não sejam retornados e para que o arquivo de banco de dados não fique muito grande.

O servidor após armazenar as informações no banco por meio de uma interface gráfica consegue montar um relatório diário de todos os *softwares* utilizados.

O banco de dados utilizado para desenvolver o *software* Monitor foi o MySQL, criado em 1995 por Michael Widenius e que por ser *open-source*[3] (RUSSELL, 2005) foi escolhido para que eventuais problemas com licença de *software* não ocorressem futuramente.

[3] Software livre que tem seu código aberto, para que se possam fazer correções, alterações entre outros.

9.3 DOCUMENTAÇÃO DO PROTÓTIPO MONITOR

O sistema Monitor como já citado foi criado utilizando a linguagem C++ que por meio de chamadas a funções da API do Windows busca informações dos processos que estão sendo executados.

O modo como foi desenvolvido o *software* para que se buscassem apenas informações sobre os programas que estivessem em execução precisou de várias chamadas a diferentes funções da interface de programação.

Por meio da função CreateToolhelp32Snapshot foi possível buscar uma listagem de todos os processos que estão executando no sistema operacional.

```
HANDLE WINAPI CreateToolhelp32Snapshot(
  DWORD dwFlags,
  DWORD th32ProcessID
);
```

No parâmetro dwFlags foi utilizado TH32CS_SNAPPROCESS que retorna todos os processos que estão sendo executados no sistema. No segundo parâmetro foi passado o valor 0 (zero) que indica o processo corrente. Esta função retorna um HANDLE que é um identificador único que é relacionado a uma janela.

Com o valor retornado chama-se outra função Process32First, que acessará informações do primeiro processo listado.

```
BOOL WINAPI Process32First(
  HANDLE hSnapshot,
  LPPROCESSENTRY32 lppe
);
```

No primeiro parâmetro será passado o retorno da função CreateToolhelp32Snapshot. O parâmetro seguinte será passado uma variável do tipo PROCESSENTRY32 que é uma estrutura onde se podem buscar várias informações sobre os processos como identificador, nome, tamanho e número de *threads* executadas pelos processos.

Logo após a função Process32Next é chamada para passar ao próximo processo até que toda a listagem seja encerrada.

```
BOOL WINAPI Process32Next(
  HANDLE hSnapshot,
  LPPROCESSENTRY32 lppe
);
```

Os parâmetros passados por ela são os mesmos passados na função Process32First e o retorno também será o mesmo. Retornando TRUE caso seja finalizada com sucesso.

A próxima função chamada é a IsWindow() que recebe como parâmetro a um HANDLE. Para o desenvolvimento do protótipo Monitor foram passados os HANDLEs que retornaram da função CreateToolhelp32Snapshot. Sendo assim pode-se ter uma resposta se o processo que está sendo analisado é realmente uma janela ou não. Se o processo analisado é janela ele retornara um valor diferente de 0 e se não for janela retornará 0.

```
BOOL IsWindow(HWND hWnd);
```

O sistema coleta as informações dos programas em execução a cada cinco segundos e armazena as informações no banco de dados SQLite.

9.4 DESENVOLVIMENTO DO SERVIDOR

Na criação do servidor foi criado um protocolo de reconhecimento de mensagens enviadas pela base cliente. O monitor uma vez ao dia solicita às informações que estão em cada máquina e a base monitora apenas envia os dados.

O protocolo criado para comunicação entre monitor e servidor é muito simples. Por meio de sockets o servidor faz a solicitação solicita e o monitor envia uma linha de caracteres com o seguinte modelo:

a) nome do programa;

b) tempo;

c) data;

d) *foreground*;

e) nome da máquina;

f) ip da máquina;

g) título da janela.

Exemplo do protocolo do *software* Monitor:

PRG NOTEPAD.EXE 5000 1180567676 5000 NomeMáquina Ip TituloPágina

Ao receber a informação o servidor reparte cada cadeira de caracteres por meio de um método chamado StringTokenizer e logo após as informações são enviadas para o banco de dados MySQL.

9.5 ESTUDO DE CASO

Os testes com o protótipo Monitor foram realizados em uma empresa da região. A empresa trabalha hoje com seis funcionários, mas foram monitoradas apenas quatro máquinas da rede.

A divisão de funcionários por cargo nesta empresa é feita da seguinte maneira:

a) dois programadores;

b) dois analistas de suporte;

c) dois responsáveis por vendas e financeiro.

O gerente da organização chegou à conclusão que definir uma política de *software* ainda que a empresa seja nova no mercado poderia prevenir futuros problemas com pirataria de *software*, vírus, trojans e programas do gênero e principalmente rendimento dos seus empregados no dia-a-dia do trabalho.

Foram definidos quatro problemas a serem tratados na política de *software*:

a) auxiliar a criação de uma listagem de ativos de *software*;

b) acabar com o uso de sistemas não licenciados;

c) diminuir o número de programas maliciosos;

d) aumentar o rendimento dos funcionários com a não utilização de *softwares* fora dos parâmetros do trabalho.

No primeiro tópico de controle de ativos de *software*, estariam apenas os sistemas que cada funcionário poderia utilizar. Citando um exemplo, o desenvolvedor Java somente poderia utilizar a ferramenta Netbeans e o banco de dados MySQL. Geralmente as empresas fazem um controle de ativos para que se tenha uma listagem organizada de quais programas estão instalados em cada máquina.

Com a facilidade de fazer *download* de aplicativos na Internet, a empresa se preocupou em checar se os *softwares* baixados por seus funcionários são ilegais, o que poderia ter repercussões graves caso uma fiscalização faça uma vistoria nas máquinas da empresa.

Os programas maliciosos além de afetar o bom funcionamento da rede e dos computadores poderiam roubar informações sigilosas da organização.

Foi desenvolvido um estudo estatístico de quais *softwares* os usuários de computadores da empresa estudada estão utilizando, a fim de tentar aumentar o rendimento dos mesmos na organização descrito no quarto tópico.

9.5.1 Licenciamento de *Software*

Como já citado neste trabalho hoje mais de 60% das empresas brasileiras utilizam *software* sem licenciamento. O gerente da empresa analisada com a preocupação de uma possível fiscalização colocou grande ênfase no estudo do controle de aplicativos piratas. As multas por utilizar este tipo de sistema podem chegar a quantias enormes o que poderia levar à empresa a falência. O *software* Monitor apenas auxiliará a empresa neste caso, pois este programa simplesmente monitora sistemas em execução e se acaso algum programa sem licença não for aberto logo não será detectado.

Seguem relacionadas quatro tabelas que mostram os softwares sem licença que o protótipo Monitor detectou:

Tabela 1. Máquina 192.168.0.11 e *softwares* sem licença.

192.168.0.11 - Financeiro	
Dreamweaver MX	*Software* para edição de páginas HTML
PC Cilin	Anti Vírus – Fabricado pela Trend Micro
Corel Draw	**Sistema de criação e edição de imagens**
Fireworks	Sistema de criação e edição de imagens
WinRar	Programa de compactação de arquivos

Tabela 2 Máquina 192.168.0.13 e *softwares* sem licença.

192.168.0.13 - Suporte	
Babylon	*Software* de tradução de textos
EditPad Pro	Editor de texto
PC Cilin	Anti Vírus – Fabricado pela Trend Micro
WinRar	**Programa de compactação de arquivos**

Tabela 3. Máquina 192.168.0.14 e *softwares* sem licença.

192.168.0.14 - Suporte	
Office Scan	Anti Vírus
Babylon	Programa de tradução de textos
Dreamweaver MX	Sistema para edição de páginas HTML
WinRar	Programa de compactação de arquivos

Tabela 4. Máquina 192.168.0.15 e *softwares* sem licença.

192.168.0.15 - Desenvolvimento	
Office Scan	Anti Vírus
AVG	Anti Vírus
WinRar	Programa de compactação de arquivos

Ao final dos relatórios gerados dos sistemas que não possuem licença o gerente tomou a decisão de desinstalar todos esses aplicativos para que não ocorra

nenhum problema com licenciamento dentro da empresa. Alguns programas como antivírus estão sendo analisados algumas versões de diferentes fabricantes para que seja adquirido *softwares* originais.

9.5.2 Controle de Ativos de *Software*

A listagem de ativos de *software* ajuda a ter um controle de licenças e quais sistemas a empresa possui. A empresa estudada neste trabalho criou um modelo de controle de ativos onde foram relacionados os sistemas da seguinte maneira:

a) nome do *software*: nesta coluna será inserido o nome do sistema cadastrado;
b) versão do sistema: demonstra qual a versão do sistema;
c) funcionalidade: determina qual o funcionamento do sistema; Ex: (MS Word: Editor de texto);
d) tipo de licença: descrição de qual tipo de licença o sistema possui. Ex: (MySQL: *Software* Livre);
e) cadastrado em: data em que o sistema foi cadastro nos ativos de *software*.

No relatório geral de cada máquina foram analisados todos os sistemas pelo qual o funcionário utiliza em seu trabalho e feito assim uma listagem de ativos de *software*

9.5.3 Monitoramento do Uso de *Software*

O *software* Monitor foi instalado em quatro máquinas na organização. Ele monitorou os programas que cada funcionário utilizou durante o dia-a-dia de trabalho.

Em cada máquina foram criados novos usuários para que o Administrador pudesse ter total controle sobre o Monitor. Os funcionários receberam usuários sem permissão para terminar os processos do Windows e não parar o serviço correspondente ao *software* Monitor. Nas tabelas geradas neste trabalho foram somados os tempos que os sistemas foram *Foreground*, ou seja, foram janelas principais. E também foram colocados os tempos que os sistemas ficaram executando no sistema operacional.

O período de coleta dos dados correspondeu de 15/05/2007 à 01/06/2007 e foram obtidos os seguintes resultados.

Em um total de 76 horas 08 minutos e 32 segundos o *software* Monitor mostrou que o funcionário do setor Financeiro utiliza pouco mais de 60% do seu tempo na empresa com sistemas equivalentes ao seu trabalho. Ou seja, ele perdeu 27 horas 6 minutos e 56 segundos em duas semanas de trabalho.

As Tabelas 5 e 6 demonstram o total de horas e quais programas foram executados durante o trabalho na máquina de IP 192.168.0.11.

Tabela 5. Máquina 192.168.0.11 e *softwares* que fazem parte do trabalho.

192.168.0.11 - Financeiro		
Nome	**Tempo executando**	**Tempo foreground**
Dreamweaver MX	00:02:12	00:01:36
Calculadora	00:35:00	00:18:30
WordPad	03:54:00	00:11:50
Nero	00:33:06	00:10:30
Corel Draw	05:06:00	01:30:32
Cob Caixa	03:54:16	02:42:34
GeraSerial	03:20:22	01:09:50
Fireworks	00:23:38	00:13:14
Hábil (Sist. Contábil)	05:54:00	02:24:02
Thunderbird	71:00:28	29:05:04
Windows Live Messenger	00:02:50	00:01:40
Windows Messenger 7.5	65:58:24	11:11:58
	Total Foreground:	**49:01:36**

Tabela 6. Máquina 192.168.0.11 e *softwares* que não fazem parte do trabalho

192.168.0.11 - Financeiro		
Nome	**Tempo executando**	**Tempo foreground**
Mozilla Firefox	23:51:44	10:25:12
Windows Media Player	10:16:40	00:24:52
Google Talk	06:46:24	03:42:06
Adobe Acrobat Reader	00:53:02	00:30:48
Messenger (Particular)	65:58:24	05:04:52
Paint	00:13:22	00:04:50
PC Cilin	00:34:12	00:05:56
Internet Explorer	00:06:32	00:02:50
DVD Shrink	01:51:50	00:21:22
NotePad	00:01:40	00:01:40
Explorer	43:16:02	05:56:00
Outros	65:58:24	00:26:28
	Total Foreground:	**27:06:56**

A Figura 10 demonstra em forma de gráfico o resultado final da análise da utilização de *softwares* na máquina do setor Financeiro:

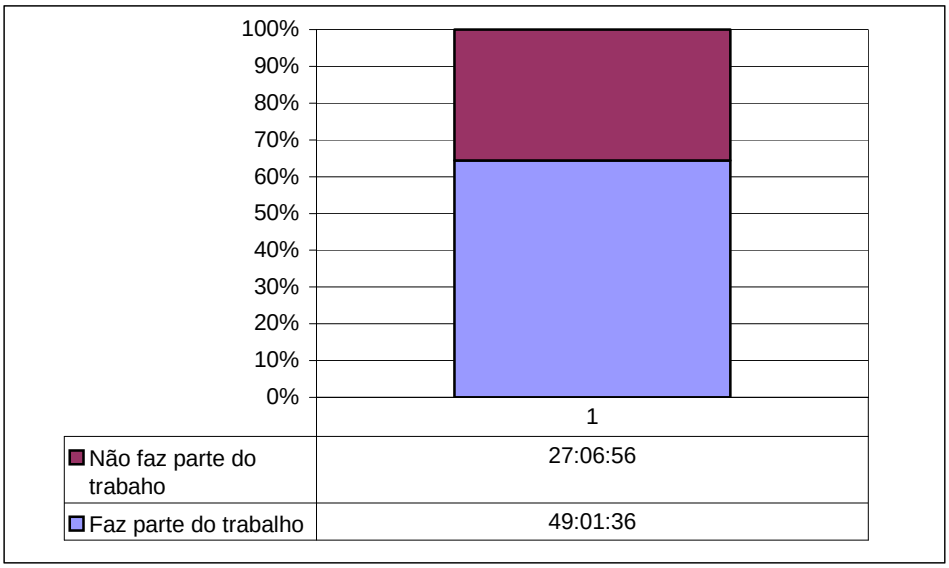

Figura 10. Estudo estatístico da máquina 192.168.0.11.

A máquina monitorada no setor de suporte foi constatada que a utilização de *softwares* não condizentes ao trabalho foi quase que equivalente ao uso dos sistemas que fazem parte do cotidiano de trabalho. A máquina correspondente a este monitoramento utiliza dois sistemas operacionais (Windows e Linux) o que explica o fato do tempo total de checagem desta máquina ser somente de 54 horas 44 minutos e 09 segundos. A Tabela 8 mostra que o tempo gasto com acesso a Internet foi superior a 15 horas e 30 minutos utilizando dois navegadores o *Firefox* e o *Internet Explorer*.

Nas tabelas 7 e 8 são demonstrados os resultados da máquina 192.168.0.13 que pertence ao setor de suporte.

Tabela 7. Máquina 192.168.0.13 e *softwares* que fazem parte do trabalho.

192.168.0.13 - Suporte		
Nome	**Tempo executando**	**Tempo foreground**
Windows Live Messenger	00:40:42	00:12:44
Thunderbird	53:06:01	06:05:43
Putty	00:27:50	00:05:34
Super EDI (Editor texto)	00:04:06	00:01:54
Babylon	55:54:01	00:18:06
Gaim	00:08:36	00:02:10
Console de Admin	21:06:04	03:46:16
Aol Instant Messenger	21:32:12	00:14:16
Yahoo Messenger	11:54:16	01:31:06
Windows Messenger 7.5	43:05:41	14:05:42
WinSCP	03:16:16	02:02:00
VNC Viwer	00:27:32	00:07:40
	Total Foreground:	**28:33:11**

Tabela 8. Máquina 192.168.0.13 e *softwares* que não fazem parte do trabalho.

192.168.0.13 - Suporte		
Nome	**Tempo executando**	**Tempo foreground**
Mozilla Firefox	41:02:51	15:24:02
Windows Media Player	00:01:22	00:01:06
Messenger (Particular)	43:05:41	03:45:10
EditPad Pro	00:17 :47	00:09:26
WinRar	00:54:26	00:10:06
Google Talk	22:44:51	00:25:52
Adobre Acrobat Reader	00:07:06	00:05:02
Internet Explorer	01:33:22	00:13:28
Skype	04:44:32	00:36:48
NotePad	02:30:50	00:49:14
Calculadora	00:01:22	00:01:22
Paint	00:45:32	00:17:04
Messenger Live Plus	00:00:32	00:00:32
SQLite Maestro	00:04:50	00:03:22
Explorer	67:16:33	02:15:10
Outros	75:02:47	01:53:14
	Total Foreground:	**26:10:58**

A Figura 11 demonstra o resultado gerado da análise do computador 192.168.0.13 que pertence ao setor de Suporte:

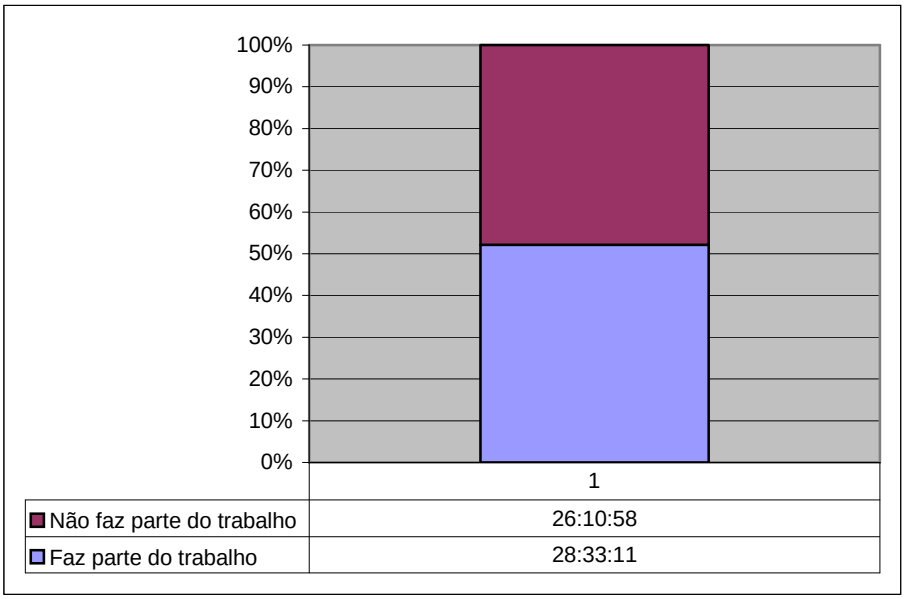

Figura 11. Estudo estatístico da máquina 192.168.0.13.

Ao fazer uma análise em mais uma máquina do setor de Suporte pode-se constatar que a utilização de *softwares* fora do parâmetro de trabalho é maior dentre todas as máquinas já analisadas o que demonstra que o usuário passou mais tempo com programas que não condizem com a rotina de trabalho. O total de horas monitoradas nesta máquina foi de 23 horas 27 minutos e 08 segundos. Esta máquina também possui três sistemas operacionais (Windows, Linux e FreeBSD) que são executados dependendo das necessidades do setor.

As tabelas 9 e 10 mostram os resultados do *software* Monitor na máquina 192.168.0.14.

Tabela 9. Máquina 192.168.0.14 e *softwares* que fazem parte do trabalho.

192.168.0.14 - Suporte		
Nome	**Tempo executando**	**Tempo foreground**
Mozilla Firefox	15:59:06	01:32:30
ICQ Lite	12:07:43	00:06:48
Dreamweaver	00:22:46	00:06:36
Habil	02:59:02	02:39:02
Babylon	15:00:50	00:00:32
Thunderbird	11:38:02	00:26:12
Console de Admin	11:20:12	00:24:48
Windows Live Messenger	29:03:02	01:39:58
Yahoo Messenger	15:40:50	01:43:10
Microsoft Word	00:03:46	00:01:47
AOL Messenger	14:18:06	00:03:28
Paint	00:45:32	00:17:04
Yahoo Messenger Insider	01:24:50	00:12:04
Messenger 7.5	12:12:16	01:03:30
Nero	00:00:32	00:00:32
	Total Foreground:	**10:18:01**

Tabela 10. Máquina 192.168.0.14 e *softwares* que não fazem parte do trabalho.

192.168.0.14 - Suporte		
Nome	**Tempo executando**	**Tempo foreground**
Mozilla Firefox	15:59:06	08:24:50
Win. Live Messenger (Part.)	29:03:02	00:33:12
Messenger 7.5 (Particular)	12:12:16	00:23:20
Windows Media Player	00:46:06	00:08:16
Free Download Manager	01:16:22	00:16:06
Google Talk	20:00:16	00:03:10
DVD Shrink	01:31:00	00:09:22
Internet Explorer	00:01:47	00:00:36
WinRar	00:00:06	00:00:06
Microsoft Word	00:03:46	00:01:47
Explorer	15:04:12	03:01:40
Notepad	00:33:56	00:05:40
Camera Digital (*Software*)	00:01:22	00:01:02
	Total Foreground:	**13:09:07**

A Figura 12 demonstra o resultado gerado da análise do computador 192.168.0.14 que pertence ao setor de Suporte:

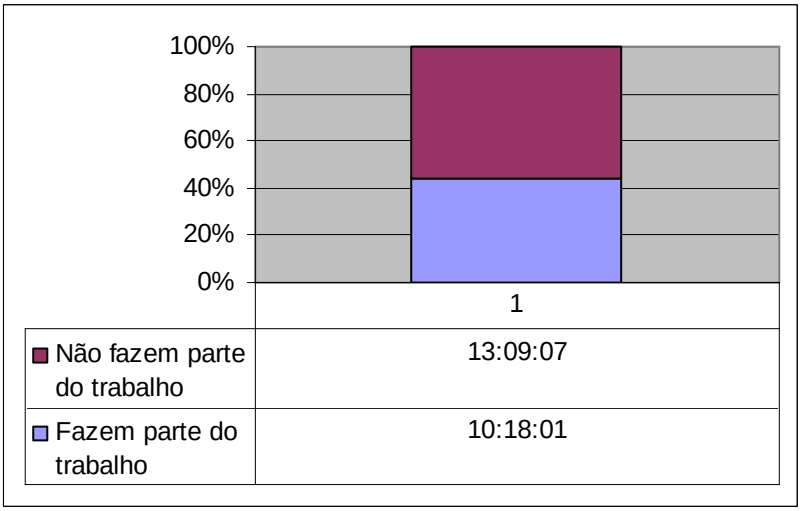

Figura 12. Estudo estatístico da máquina 192.168.0.14.

O melhor resultado entre as máquinas analisadas foi obtida no setor de Desenvolvimento aonde o resultado chegou a aproximadamente 80% de utilização de programas pertencentes ao desenvolvimento de *software*. Esta máquina utiliza também dois sistemas operacionais (Windows e Linux) o que fez com que o tempo total fosse muito menor do que a máquina do setor Financeiro que utiliza apenas sistema Windows.

O tempo total de análise nesta máquina foi de 28 horas 12 minutos e 32 segundos e a maior parte do tempo foram utilizadas com o Netbeans que é um *software* de desenvolvimento de sistemas para programação na linguagem Java.

As tabelas 11 e 12 descrevem os resultados do tempo total de utilização de *softwares* no computador com o IP 192.168.0.15.

Tabela 11. Máquina 192.168.0.15 e *softwares* que fazem parte do trabalho.

192.168.0.15 - Desenvolvimento		
Nome	**Tempo executando**	**Tempo foreground**
Netbeans	28:21:10	15:42:06
Windows Live Messenger	02:49:56	00:57:34
Messenger 7.5	03:30:32	00:17:22
Mozilla Firefox	28:02:46	03:42:36
Console de Admin	18:39:05	01:30:12
Thunderbird	00:58:12	00:11:38
Adobe Acrobat Reader	02:34:44	00:04:16
Calculadora	00:03:02	00:03:02
	Total Foreground:	**22:28:46**

Tabela 12. Máquina 192.168.0.15 e *softwares* que não fazem parte do trabalho.

192.168.0.15 - Desenvolvimento		
Nome	**Tempo executando**	**Tempo foreground**
Windows Media Player	03:38:50	00:07:02
Windows Live Messenger	02:49:56	00:22:24
Messenger 7.5	03:30:32	00:04:54
Mozilla Firefox	28:02:46	01:06:20
Wordpad	00:05:52	00:03:52
Internet Explorer	03:32:16	01:08:08
Notepad	04:17:40	00:20:32
Microsoft Word	00:08:56	00:04:50
WinRar	00:01:38	00:01:16
AVG	12:49:14	00:01:56
Microsoft Excel	00:03:22	00:02:50
OfficeScan	00:02:12	00:00:16
Explorer	19:51:22	02:18:43
	Total Foreground:	**05:43:03**

A Figura 13 demonstra o resultado gerado da análise do computador 192.168.0.15 que pertence ao setor de Desenvolvimento:

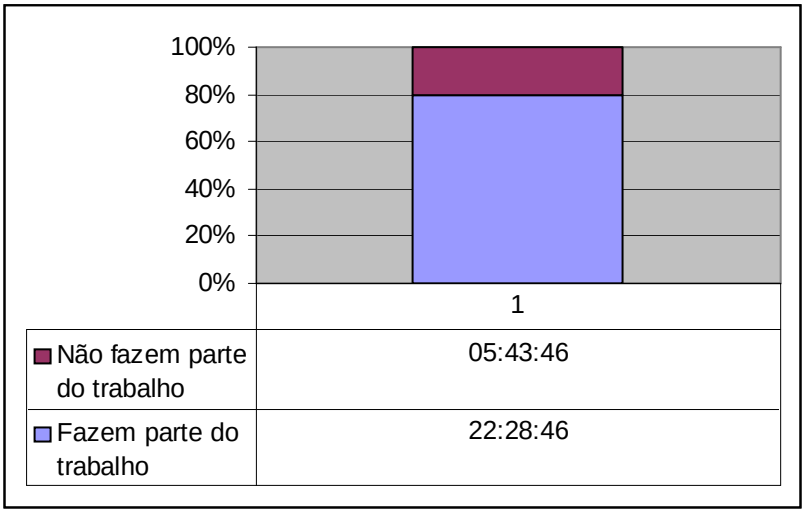

Figura 13. Estudo estatístico da máquina 192.168.0.15.

9.5.4 Definição de uma Política de *Software*

Após alguns resultados obtidos com o *software* Monitor o gerente da empresa analisada decidiu criar algumas regras para uma melhor utilização dos sistemas dentro da organização.

As normas foram descritas da seguinte maneira:

a) por meio de regras de *firewall* arquivos com extensão .exe, .zip, .rar , .bin, .bat estão proibidos de serem feitos *donwload*;

b) *softwares* que não forem livres de licença não poderão ser instalados em hipótese nenhuma;

c) a utilização de *sites* de relacionamento (Orkut, Fotologs) estão bloqueados;

d) sempre que um funcionário precisar baixar arquivos com uma extensão proibida ele deverá informar ao gerente descrevendo qual será a sua utilização do mesmo na empresa;
e) após a liberação do gerente, o *software* será incluído na listagem de ativos da empresa;
f) os *sites* de notícias estarão liberados por 30 (trinta) minutos depois no início do expediente, ou seja até às 08:30;
g) páginas web de vídeos (YouTube e Google Video) também estão proibidos no período de trabalho por utilizar muita banda;
h) ao final de cada mês o gerente da empresa passará em cada máquina examinando se algum sistema foi instalado sem sua permissão.

Com a implantação destas regras pretende-se dar maior segurança e rendimento aos funcionários. Com a utilização de regras de *firewall* e esta nova política de utilização de *software* pretende-se que o número de programas maliciosos diminua e o número de programas sem licença seja zerada. Até a conclusão deste trabalho estavam sendo estudadas novas formas de se tentar reduzir a entrada de vírus e obter um melhor controle de programas piratas.

Existem hoje alguns softwares que monitoram máquinas de uma rede citando como exemplo o CACIC que é um projeto do governo federal brasileiro onde esse sistema faz um inventário de software e hardware. Este programa atua no formato GPL e tem seu código-fonte aberto para que se possa ser feita modificações ou melhorias. Este sistema trabalha com o conceito de rede cliente/servidor igualmente o software desenvolvido neste projeto, mas traçam paralelos totalmente diferentes pois o foco deste

trabalho é monitor programas abertos e seu tempo de execução e apenas auxiliando no encontro de sistemas piratas.

O trabalho de conclusão de curso do acadêmico da Unesc, Roberto de Souza Boava também foca seu trabalho em um inventário de software e periféricos de hardware utilizando arquitetura multi-camadas.

10 TRABALHOS FUTUROS

A pesquisa do ex-acadêmico da Universidade do Extremo Sul Catarinense (UNESC) Gary Vaca D'Este busca em todas as máquinas de uma rede por meio de um agente móvel todos os *softwares* instalados. No desenvolvimento do protótipo Monitor o foco não é o controle de ativos até porque se no tempo em que o sistema ficou instalado um usuário não executasse um programa sem licença ele não iria ser identificado. Mas sim propiciar ao gerente da empresa monitorada aproximadamente quantos *softwares* piratas na empresa possuía. Mas para um trabalho futuro pode-se integrar os dois sistemas a fim de construir uma ferramenta poderosa para monitoramento de máquinas em uma organização.

11 CONCLUSÃO

As políticas de *software* nos dias de hoje ainda estão muito mal definidas em algumas organizações, principalmente em empresas de pequeno porte. A inobservância dos gerentes para com as novas tecnologias disponíveis no mercado provoca desorganizações dentro do setor de TI. A falta de inquisição dos gestores proporcionam as empresas quedas de rendimento dos funcionários, aumento de *softwares* ilegais dentre outros problemas.

Esta pesquisa tentou ajudar de várias formas uma empresa da região auxiliando a desenvolver uma política de *software* ainda que para se chegar a um nível de excelência possam-se demorar alguns meses ou anos.

O *software* desenvolvido demonstrou que os funcionários da empresa monitorada estão em um nível baixo de rendimento e que agora com o sistema instalado em cada máquina e com uma nova política de utilização de sistemas a organização possa crescer não somente sozinha mais junto a seus funcionários.

Ao final deste projeto notaram-se algumas dificuldades que poderiam ser acrescentadas em trabalhos futuros como:

a) inclusão de um banco de dados que contenha nomes de *softwares* relacionados aos seus respectivos nomes dos processos;
b) criação de uma interface gráfica para melhor administração do sistema;
c) integração com o projeto desenvolvido pelo ex-acadêmico da Unesc Gary Vaca D'Este para busca de programas instalados nos computadores de uma rede auxiliando o controle de ativos de *software*.

REFERÊNCIAS

ADAM, Keli; SCRIMGER, Rob. **MCSE, TCP/IP – Trainning Guide**. 2 ed. EUA: New Riders, 1999.

ALHIR, Sinan Si; **Learning UML**. O' Reilly: 2003.

BERG, Glenn. **MCSE, Network Essentials – Tranning Guide**. 2 ed. Indianapolis:New Riders, 1998.

CASAD, Joe; WILLSEY, Bob. **Aprenda em 24 horas TCP/IP**. Rio de Janeiro: Campus, 1999.

COMER, Douglas; STEVENS, David L. **Interligação em rede com TCP/IP**. Rio de Janeiro: Campus, 1999.

COMER, Douglas; **Internetworking with TCP/IP – Principles, Protocols and Architetures**. 4 ed. New Jersey: Prentice Hall, 1999.

COMPUTERWORLD**. Pirataria de software cai no Brasil, mas prejuízo supera US$ 1 bi. Disponível em:< http://computerworld.uol.com.br/mercado/2007/05/15/idgnoticia.2007-05-15.5103056083**>. Acessado em: 10 jun. 2007.

COWART, Robert. **Martering Windows 3.1**. Sybex: 1993.

DABAK, Prasad; PHADKE, Sandeep; BORATE, Milind. **Undocumented Windows NT**. M&T Books: 1999.

DALMAZO, Luiza. Vírus é o maior problema enfrentado por empresas que utilizam a internet. **Pesquisa**. **Ano 2007 Publicado em: 30 mai. 2007. Disponível em: <http://computerworld.uol.com.br/seguranca/2007/05/30/idgnoticia.2007-05-29.9638193019>.** Acessado em: 03 jun. 2007.

DULANEY, Emmett; LAWURENCE, Sherwood; SCRIMGER, Robert; TILKE, Anthony; WHITE, Jonh; WILLIAMS, Raymond; WOLFORD, Kevin. **MCSE, TCP/IP – Trainning Guide.** EUA: New Riders, 1998.

HAMILTON, Kim; MILES, Russel. **Learning UML 2.0**. O' Reilly, 2006.

HOLLINGWORTH Jarrod, BUTTERFIELD Dan, SWART Bob, ALLSOP Jamie. **Borland C++ Builder 5 – Developer's Guide**. New York – SAMS, 2001.

IDG Now. Estudo: 43% dos sites de downloads instalam programas maliciosos. **Disponível em: <http://idgnow.uol.com.br/seguranca/2006/12/12/idgnoticia.2006-12-12.6201545613>. Acessado em: 03 jun. 2007.**

IDG NOW. Pirataria de software cai no Brasil, mas prejuízo supera US$ 1 bi. **Disponível em: <http://idgnow.uol.com.br/mercado/2007/05/15/idgnoticia.2007-05-15.9813946103>. Acessado em: 03 jun. 2007.**

LOSHIN, Peter. **TCP/IP: Clearly Explained.** 4. ed. Boston: Morgan Kaufmann Publishers, 2003.

NAUGLE, Matthew. **Illustrated TCP/IP**. Wiley Computer Publishing, 1998.

OLIVEIRA, Rômulo Silva de; CARISSIMI, Alexandre da Silva; TOSCANI, Simão Cirineu. **Sistemas Operacionais – Instituto de Informática da UFRGS**. 3 ed. Porto Alegre: Sagra Luzzato, 2004.

PETROUTSOS, Evangelos. **Dominando o Visual Basic 6 – A Bíblia**. São Paulo : Makron Books, 1999.

RICHTER, Jeffrey. **Windows Avançado. Guia de Desenvolvimento para API WIN32, para Windows NT e Windows 95**. São Paulo: Makron Books, 1996.

RICHTER, Jeffrey**. Programing Applications for Microsoft Windows**. 4 ed. Washington: Microsoft Press: 1999.

RUSSELL, J. T. Dyer; **MySQL – In a Netshell.** California: O'Reilly, 2005.

RUSSINOVICTH, Mark E.; SOLOMON, David A.. **Microsoft Windows Internals – Microsoft Windows Server 2003, Windows XP, Windows 2000**. 4 ed. Microsoft Press: 2004.

SCHREIBER, Sven B. **Undocumented Windows 2000 Secret – A Programmers Cookbook**. Indianapolis: Pearson Education, 2001.

SCRIMGER, Rob; LASALLE, Paul; PARIHAR, Mridula; GUPTA, Meeta. **TPC/IP a Bíblia.** 6. ed. Rio de Janeiro: Campus, 2002.

SIMON, RICHARD. **Windows NT WIN32 API Superbible**. The Waite Group, 1998.

SOARES, Luiz Fernando Gomes; LEMOS, Guido; COLCHER, Sérgio**. Redes de Computadores: Das LANs, MANs e WANs às Redes ATM**. 2.ed. Rio de Janeiro: Campus, 1995.

SWART Bob, CASHMAN Mark, GUSTAVSON Paul, HOLLINGWORTH Jarrod. **Borland C++ Builder 6 – Developer's Guide**. 4 ed. New York : SAMS, 2003.

TANEMBAUM, Andrew S. **Redes de Computadores**. 4. ed. Rio de Janeiro: Campus, 1997.

TEIXEIRA JUNIOR, José Helvécio. Redes **de computadores: serviços, administração e segurança.** São Paulo: Makron Books, 1999.

TORRES, Gabriel. **Redes de Computadores: Curso Completo**. Rio de Janeiro: Books do Brasil, 2001.

VILLANI, Pat. **Programming Win32 Under The API**. CMP Books, 2001.

BIBLIOGRAFIA RECOMENDADA

DEITEL, H.M; DEITEL, P. J. **Java**: Como Programar. 4.ed. Porto Alegre: Bookman, 2003.

HSBC. **Software e hardware que ajudam na segurança de empresas**. Disponível em: <http://www.hsbc.com.br/common/seguranca/artigo-software-hardware-ajudam-empresas.shtml>. Acessado em: 01/04/2007.

UNISINOS - Universidade do Vale do Rio dos Sinos. **Redes de Computadores - modelos OSI e TCP/IP**. Disponível em: <http://jpl.com.br/redes/redes_osi.pdf>. Acessado em: 11/11/2006.