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

Jeziel Torres Pereira

Modelo de Gerenciamento baseado em Ferramentas de Baixo Custo para Redes de Pequeno Porte

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

Prof.a. Elizabeth Sueli Specialski
Orientadora

Florianópolis, Dezembro de 2002.

Modelo de gerenciamento baseado em ferramentas de baixo custo voltado para redes de pequeno porte

Jeziel Torres Pereira

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

Prof.a. Elizabeth Sueli Specialski, Dr.
Orientadora
beth@inf.ufsc.br

Prof. Fernando Álvaro Ostuni Gauthier, Dr.
Coordenador do Curso
gauthier@inf.ufsc.br

Banca Examinadora

Prof. João Bosco da Mota Alves, Dr.
jbosco@inf.ufsc.br

Prof. João Bosco Mangueira Sobral, Dr.
bosco@inf.ufsc.br

“O coração do homem pode fazer planos, mas a resposta certa vem dos lábios do Senhor.”

Provérbios 16:1 (Bíblia Sagrada)

À minha amada esposa Rossana e meus filhos Guilherme e Joyce, igualmente amados, pela compreensão, apoio e liberação do tempo necessário a mais essa conquista.

Agradecimentos

Primeiramente, a Deus, que me criou, me amou e me preservou, em vitória, até este momento.

Aos meus pais, Dulcino Pereira e Jacira Torres Pereira, pelo incentivo e valorização dos meus estudos, mesmo em meio à dificuldades, sem esmorecer.

Aos colegas Fernando Lopes, Wagner G. Valente e Amauri S. Ghisleri, companheiros de discussões, trabalhos e apresentações, bem como aos demais colegas da turma de Joinville.

Aos professores do CPGCC da UFSC, por terem repassado os seus conhecimentos e experiências e pelo incentivo à busca do saber e do aprender a aprender.

À Minancora & Cia Ltda, na pessoa da Diretora Comercial, Lourdes Maria Dória Duarte, pelo incentivo à minha participação neste curso e obtenção deste título.

À minha orientadora, Elizabeth Sueli Specialski (Beth), por ter acreditado em mim, ter me recebido na intimidade do seu lar, ter me exortado e incentivado, nas horas e medidas certas. Também agradeço a seus familiares, que sempre me receberam com atenção e carinho.

Ao amigo Marco André Lopes Mendes, pelas palavras de encorajamento, pela leitura crítica do trabalho e pelas sugestões no decorrer da escrita do mesmo.

Ao colega Mateus Casanova Pereira pela valiosa contribuição no comparativo das ferramentas de gerenciamento.

Finalmente e destacadamente, à minha esposa Rossana pelo amor, paciência, compreensão e suporte nos momentos de desânimo, tensão e cansaço, bem como aos meus filhos Guilherme e Joyce, de quem foram subtraídos preciosos períodos de convivência.

Resumo

Este trabalho apresenta as necessidades de gerenciamento uma rede de pequeno porte, analisando também as limitações e dificuldades encontradas para que as mesmas possam ser supridas.

Um comparativo das ferramentas de gerenciamento disponíveis no mercado, priorizando-se as ferramentas de baixo custo, foi realizado, com o objetivo de analisar a possibilidade de que alguma delas pudessem atender aos requisitos necessários.

Buscando suprir as necessidades de gerenciamento apresentadas, um modelo de gerenciamento baseado em ferramentas de baixo custo e voltado para redes de pequeno porte é proposto neste documento.

A implementação do modelo proposto, utilizando-se de ferramentas disponíveis na rede e a realização de um estudo de caso, colocando em prática o que o modelo propõe, vieram a corroborar a viabilidade de implantação do modelo, bem como a possibilidade de que o mesmo satisfaça às necessidades levantadas.

A utilização do *software* MS-Excel para implementação do modelo deu-se em função da sua ampla utilização nos ambientes de pequenas redes, à facilidade de manipular tabelas e pela sua poderosa linguagem de macros (VBA), que possibilitou o acesso às API’s do sistema operacional.

Na implementação, foram desenvolvidas rotinas internas para a execução de comandos do tipo ping (PING-J), envio de e-mails (Envia_Email) e emulação de terminais (TELNET-J), visando o controle no processamento dos mesmos e a independência dos utilitários do sistema operacional.

Um estudo de caso, realizado a partir da implantação do modelo em uma empresa real, foi necessário para confirmar a funcionalidade do mesmo em um ambiente de rede de pequeno porte.

Palavras chaves: Modelo de gerenciamento, redes, monitoramento, *winsock.*

Abstract

This paper presents the management necessities of a small size network, also analyzing the found out limitations and difficulties so that the same ones can be supplied.

A comparative degree of the available tools of management in the market, prioritizing itself the tools of low cost, was carried through, with the objective to analyze the possibility of that some of them could take care of to the necessary requirements.

Searching to supply the presented necessities of management, a model of management based on tools of low cost and directed toward networks of small size is proposed in this document.

The implementation of the considered model, using of available tools in the net and the accomplishment of a case study, placing in practical what the model considers, had come to corroborate the viability of implantation of the model, as well as the possibility of that the same it could satisfy to the raised necessities.

The use of MS-Excel software for implementation of the model was given in function of its ample use in environments of small nets, to the easiness to manipulate tables and for its powerful language of macros (VBA), that it made possible the access to the API's of the operational system.

In the implementation, had been developed internal routines for the execution of commands of the type ping (PING-J), of e-mails (Envia_Email) and emulation of terminals (TELNET-J), aiming at the control in the processing and independence of the utilitarian ones of the operational system.

A study of case, carried through from the implantation of the model in a real company, it was necessary to confirm the functionality of exactly in an environment of network of small size.

Words keys: Model of management, networks, monitoring, winsock.

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Apresentação

A necessidade de gerenciamento de uma rede é notória desde o instante em que o segundo computador é conectado, pois a partir deste momento os eventos ocorridos no conjunto de computadores interligados, bem como o grau de complexidade dos mesmos, aumenta consideravelmente, comparando-se com o uso do computador de maneira isolada.

A preocupação com a disponibilidade dos recursos compartilhados bem como o desempenho dos mesmos, face ao acréscimo de número de solicitações encaminhadas a eles, passa a ter um lugar especial nesse contexto, pois o que se espera é que a qualidade dos serviços prestados deve ser melhorada ou, no mínimo, mantida.

Tanto em redes de pequeno porte quanto nas de grande porte, faz-se necessário o uso de ferramentas de gerenciamento capazes de auxiliar a execução das tarefas básicas de gerenciamento. Numa rede de pequeno porte as tarefas de gerenciamento podem ser executadas até mesmo sem a utilização de uma ferramenta específica. Ao contrário, numa rede de grande porte, uma ferramenta de gerenciamento torna-se indispensável, pois a quantidade de objetos e eventos a serem gerenciados inviabiliza um acompanhamento que não seja automatizado.

A diversidade de ferramentas de gerenciamento existentes, para as mais diversas plataformas de hardware e software, faz com que a escolha do melhor produto a ser utilizado, tenha a um nível de dificuldade bastante elevado. Cada um dos produtos tem as suas particularidades ou características com relação aos diversos fatores envolvidos no gerenciamento de uma rede, tais como quantidade e tipo dos objetos a serem gerenciados, quantidade e tipo dos eventos a serem monitorados em cada objeto e a carga de utilização dos recursos da rede em função do gerenciamento. Assim a ferramenta mais adequada para uma determinada configuração de rede pode não ser a mais apropriada para uma rede com características diferentes.

O fator custo também deve ser levado em consideração quando da análise de uma ferramenta de gerenciamento. As aplicações comerciais disponíveis no mercado, notadamente as de grandes fabricantes de software, impõem um custo elevado em função da complexidade da ferramenta, pois na maioria dos casos essas ferramentas foram especificadas para grandes redes, com arquiteturas diversificadas de hardware e software, com uma estrutura de comunicação complexa além de grande quantidade e diversidade de dispositivos a serem gerenciados. As ferramentas de gerenciamento com custo mais acessível, tem limitações quanto ao tamanho e complexidade da rede a ser gerenciada, além de dispor de um número menor de recursos no próprio software, como alarmes, ações automatizadas e registro de eventos.

Os programas de gerenciamento, no seu processo de gestão, necessariamente utilizam-se dos recursos computacionais e de comunicação de dados da própria rede a ser gerenciada, podendo causar uma sobrecarga na utilização destes recursos, se não houver um dimensionamento adequado do gerenciamento face aos recursos disponíveis.

No universo do software livre ou de código aberto, podem ser encontradas ferramentas de gerenciamento de rede que poderiam suprir outra dificuldade existente, que é o fato de que, na grande maioria dos casos, o software é de arquitetura fechada, não se podendo implementar nenhum novo recurso que seja desejável e não disponível. Este tipo de software, entretanto, para que seja alterado, requer um conhecimento profundo da linguagem de programação utilizada, notadamente as linguagem C e PHP, além de exigir conhecimento do sistema operacional utilizado e de dispor da ferramenta utilizada para o desenvolvimento do software. Estes fatores caracterizam uma das principais dificuldades neste tipo de software.

1.2 Estado da Arte

O gerenciamento de redes tem se desenvolvido muito e tem sido alvo de muitas pesquisas, principalmente voltadas às grandes redes, nas quais, obviamente, o nível de complexidade do problema é muito maior do que nas pequenas redes.

Existem inúmeras ferramentas disponíveis no mercado com a proposta de prover as redes de um gerenciamento eficaz, podendo ser classificadas em ferramentas

proprietárias, *shareware*[1], *freeware*[2] e de código aberto[3]. A apresentação e avaliação destas ferramentas serão feitas no capitulo 3.

1.3 Objetivo do Trabalho

Como mencionado anteriormente, no ambiente das pequenas redes a necessidade de gerenciamento se faz presente, mas encontra-se a dificuldade da escolha da ferramenta de gerenciamento adequada a esse contexto.

Além das dificuldades mencionadas, pode-se acrescentar que a grande maioria dos softwares existentes, particularmente os do tipo *freeware* e *shareware*, apresenta seus menus e mensagens em língua inglesa, a qual não é muito conhecida no meio que se utiliza das pequenas redes de computadores. Como a facilidade de utilização e manutenção da ferramenta é uma das premissas desse trabalho, a elaboração da mesma em língua portuguesa é um dos objetivos propostos.

Este trabalho tem por objetivo propor um modelo de gerenciamento baseado em ferramentas que estejam disponíveis na maioria dos computadores da rede, utilizando linguagem de programação de conhecimento da maioria dos usuários e administradores de rede, de tal maneira que possa ser implementada, com pouco ou até nenhum investimento em termos de software e hardware.

O publico alvo deste trabalho são os gerentes ou administradores de pequenas redes, que precisam gerenciá-las sem dispor de grandes verbas orçamentárias, e que não tem necessidade de todos os recursos disponíveis nos grandes softwares de gerenciamento, mas ao mesmo tempo precisam acrescentar recursos ao software, com uma curva de aprendizagem suave. Dessa maneira, propõe-se que as ferramentas utilizada para implementar o modelo sejam de fácil aprendizagem e estejam disponíveis no ambiente de administração e gerenciamento da rede.

A partir da monitoração dos eventos da rede, o modelo propõe que sejam automatizados alguns procedimentos, bem como sejam emitidos alarmes ao se atingir os

---

[1] *Software* com permissão de uso e cópia, por tempo limitado, após o qual deve ser registrado mediante pagamento do valor especficado.
[2] *Software* com permissão para que qualquer pessoa o use, copie e distribua, gratuitamente.
[3] *Software* com permissão para que qualquer pessoa o use, copie e distribua com ou sem modificações, gratuitamente ou cobrando alguma taxa com código fonte disponível.

limites definidos para os mesmos. Pretende-se também proporcionar o registro dos principais eventos ocorridos na rede, possibilitando a geração de estatísticas, gráficos e histórico dos mesmos.

1.4 Materiais e métodos

Foi feito o levantamento das principais necessidades de gerenciamento dentro de uma pequena rede, bem como das ferramentas disponíveis em termos de aplicativos existentes e utilitários do sistema operacional. Também foi analisado se as ferramentas disponíveis eram capazes de suprir a necessidade existente, bem como possibilitar a automatização de alguns dos procedimentos de gerenciamento envolvidos no processo. Estas análises e levantamentos realizados estão descritos no capítulo 5.

A partir das análises feitas, observou-se que a plataforma dominante nas pequenas empresas, tanto para sistema operacional de rede e estação quanto para aplicativos de uso geral é a fornecida pela *Microsoft Corporation*.

Com base nas conclusões acima, definiu-se para a elaboração dos estudos, o ambiente de rede formado pelo sistema operacional Microsoft Windows NT 4.0 no servidor, Microsoft Windows 95 e suite Microsoft Office 97 Standard nas estações. O aplicativo de gerenciamento será desenvolvido baseado no Microsoft Excel e na sua linguagem de macros, o VBA (Visual Basic for Applications).

1.5 Resultados esperados

Com a finalização dos experimentos, espera-se que a ferramenta desenvolvida possa atingir os seguintes objetivos:

- Utilizar-se de recursos comumente disponíveis nos computadores, sem onerar a estrutura da rede;

- Ser instalada e utilizada com facilidade pelos gerentes e administradores de pequenas redes;

- Possibilitar, com facilidade, a alteração da mesma para acrescentar recursos de gerenciamento;

- Promover o monitoramento dos elementos da rede, gerando o registro dos principais eventos e emitindo alarmes, caso necessário;

- Minimizar o custo total de propriedade (TCO[4]) dos elementos da rede;

- Abrir caminho para projetos similares, baseado em outros ambientes de rede ou aplicativos disponíveis na rede;

1.6 Organização do trabalho

Este trabalho encontra-se organizado da seguinte maneira: No capítulo 2 são apresentados os fundamentos teóricos necessários para a compreensão do modelo a ser apresentado. O capítulo 3 descreve o gerenciamento de uma rede de pequeno porte, abordando os principais problemas e os itens necessários e passíveis de gerenciamento. No capítulo 4 apresenta-se e analisa-se as principais ferramentas de gerenciamento disponíveis, de grande, médio e pequeno porte, abrangendo produtos comerciais, de código aberto e do tipo *shareware* e *freeware*, ressaltando as principais características positivas e negativas. O capítulo 5 propõe o modelo de gerenciamento simplificado para pequenas redes. No capítulo 6 é apresentado um estudo de caso, que trata da implementação do modelo proposto em uma estrutura de rede de pequeno porte. No capítulo 7 as considerações finais são apresentadas. O Anexo A contempla a implementação do modelo de gerenciamento proposto, em linguagem Visual Basic. O Anexo B traz algumas amostras de resultados obtidos ao se aplicar o programa Telnet a alguns dispositivos e serviços da rede.

[4] TCO – Total Cost Ownership (Custo Total de Propriedade)

Capítulo 2

Fundamentos Teóricos

2.1 Introdução

Este capítulo tem por objetivo trazer alguns conceitos que se fazem necessários para a compreensão do modelo proposto. A abordagem dos temas não terá a pretensão de esgotar cada um dos assuntos, mas somente de dar o fundamento teórico essencial.

2.2 Conjunto de protocolos TCP/IP

A denominação oficial é Pilha de Protocolos de interligação em redes TCP/IP, conforme [COM00], mas geralmente é citado apenas como TCP/IP. Isto se dá porque o *Transmission Control Protocol* - TCP[5] e o *Internet Protocol* - IP[6] são os principais protocolos da pilha [COM99], apesar de não serem os únicos componentes da pilha TCP/IP. Outros protocolos da pilha TCP/IP seriam: ICMP[7], UDP[8], SMTP[9], OSPF[10], RIP[11], POP[12], entre outros [IBM00].

O TCP/IP foi o primeiro conjunto de protocolos criado especificamente para a interligação de redes ou inter-redes, de onde vem o termo internet. Ao se usar o termo Internet com a primeira letra maiúscula está se fazendo menção à Internet global ou Internet pública, que é a forma mais comum de utilização desse termo [COM99]. Pode-se usar o termo com a primeira letra minúscula (internet) dando o significado de duas redes quaisquer que estejam interligadas.

Como grande parte dos protocolos de rede, o TCP/IP tem a sua modelagem baseada em camadas, tendo cada uma delas os seus protocolos específicos, explicando assim o termo “pilha de protocolos” [IBM00]. Os protocolos internet são apresentados

---

[5] TCP – Transmission Control Protocol (Protocolo de Controle de Transmissão)
[6] IP – Internet Protocol (Protocolo de Internet ou Inter-Redes)
[7] ICMP – Internet Control Message Protocol (Protocolo de Mensagem de Controle Internet)
[8] UDP – User Datagram Protocol (Protocolo de Datagrama de Usuário)
[9] SMTP – Simple Mail Transfer Protocol (Protocolo de Transferência de Correio Simples)
[10] OSPF – Open Shortest Path First (Abrir Caminho mais Curto Primeiro)
[11] RIP – Routing Information Protocol (Protocolo de Informação de Roteamento)
[12] POP – Post Office Protocol (Protocolo de Correio Eletrônico)

em quatro camadas: Camada de Aplicação, Camada de Transporte, Camada de Inter-Redes ou Internet, Camada de Interface de Rede, conforme mostra a Figura 2.1.

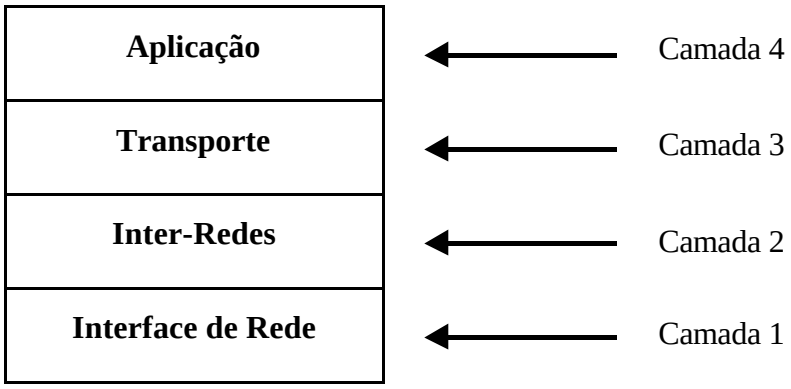

Figura 2.1: Camadas da arquitetura TCP/IP

A representação em camadas pode ser utilizada para posicionar, mas não para comparar funcionalmente, a arquitetura TCP/IP com modelos de referência, como o OSI. Murhammer em [IBM00] menciona que, em virtude de haverem diferenças básicas entre os modelos em camadas, fica bastante difícil comparar a arquitetura TCP/IP com o modelo de referência OSI. As camadas dos protocolos Internet serão descritas a seguir, conforme [COM99], [IBM00]:

- Camada 1 – Interface de Rede: A camada 1 é a interface com o *hardware* da rede, podendo ou não fornecer entrega confiável sendo permitido o uso de qualquer interface de rede (NIC[13]) disponível.

- Camada 2 – Inter-Redes : A camada 3 traz a especificação do formato dos pacotes enviados pela inter-rede bem como os mecanismos de roteamento dos pacotes até o destino final.. O principal protocolo dessa camada é o IP, existindo também o ICMP, como um protocolo auxiliar.

- Camada 3 – Transporte: Os protocolos da camada 4, fornecem a transferência de dados de uma ponta à outra. Essa camada também trata da especificação de como assegurar a confiabilidade na transferência dos pacotes através da inter-rede. Os protocolos da Camada de Transporte no modelo TCP/IP são o TCP e o UDP.

[13] NIC – Network Interface Card (Cartão de Interface de Rede ou Placa de Rede)

- Camada 4 – Aplicação: O programa que utiliza o TCP/IP para comunicação fornece a camada de aplicação. Podemos citar como exemplos de aplicações o Telnet[14], o FTP[15] e o SMTP. A definição da interface entre essa camada e a camada de transporte é feita através dos números de porta e dos soquetes.

2.3 Protocolo IP

O IP é um protocolo de camada 3 na arquitetura TCP/IP, não orientado à conexão, baseado no melhor esforço ("best effort") para entrega dos pacotes. Melhor esforço significa que o tratamento da recepção dos pacotes, ordem seqüencial dos mesmos ou duplicação não é tratado pelo IP, mas sim pelos protocolos das camadas superiores, como TCP [IBM00]. A especificação atual do protocolo IP pode ser encontrada em [RFC791], [RFC950], [RFC919] e [RFC922], com atualizações em [RFC1349].

Esse protocolo possui um endereçamento próprio para cada host[16] da rede, que é um número binário de 32 bits, único, e que é utilizado em todo o processo de comunicação com o host [COM99]. Normalmente o endereço IP é representado em formato decimal com cada byte[17] dividido por um ponto, como por exemplo, 192.168.0.25. Existe uma divisão entre cada endereço IP em prefixo e sufixo, sendo que o prefixo identifica a rede física ao qual o host está acoplado e o sufixo identifica o próprio host dentro da rede [COM99]. Os padrões para o endereçamento IP são apresentados em [RFC1166].

2.4 Protocolo ICMP[18]

Apesar do IP ser um protocolo não confiável, por usar a semântica da entrega de melhor esforço, o mesmo inclui um sistema de mensagens de controle de erros, que é o ICMP. O ICMP e o IP são interdependentes, pois o ICMP utiliza-se dos pacotes IP para transmissão das suas mensagens e o IP usa o ICMP quando envia uma mensagem de erro conforme mencionado em [COM99], [COM00] e [IBM00].

---

[14] Telnet – Protocolo para acesso terminal a outros computadores da rede
[15] FTP – File Transfer Protocol (Protocolo de Transferência de Arquivos)
[16] Host – Computador de usuário final conectado à rede
[17] Byte – Agrupamento de 8 bits
[18] ICMP – Internet Control Message Protocol (Protocolo de Controle de Mensagens da Internet)

Existem diversos tipos de mensagem utilizados pelo protocolo ICMP, com finalidades específicas, podendo ser mensagens de erro ou informativas. Entre as mensagens de erro podemos destacar a *Time Exceeded* (Tempo Excedido) e a *Destination Unreachable* (Destino inatingível) e entre as informativas a *Echo Request/Reply* (Solicitação/Resposta Eco).

Pelo menos dois aplicativos utilizam o protocolo ICMP em suas implementações: o Ping e o Traceroute, ambos de ampla utilização. O Ping, cujo nome é um acrônimo de Packet InterNet Groper [IBM00], envia um ou mais datagramas IP para um host de destino, solicitando uma resposta e a partir dela, calculando o tempo entre a ida e a volta do datagrama. Para tanto, o aplicativo Ping utiliza-se das mensagens ICMP de *Echo Request* e *Echo Reply*, pois o ICMP é obrigatório em todas as implementações TCP/IP.

Outro aplicativo a utilizar-se do ICMP é o Traceroute, que também usa o protocolo UDP, conforme [IBM00]. A função do Traceroute é determinar a rota percorrida pelo datagrama IP entre o host origem e o host destino. Uma das maneiras do aplicativo realizar sua tarefa é mencionada em [COM99], através do envio de datagramas UDP em combinação com as respostas ICMP *Time Exceeded* e *Destination Unreachable.*

O ICMP tem a sua padronização definida por Postel em [RFC0792] e atualizada por Braden em [RFC1122]

2.5 Protocolo TCP

O TCP é o principal protocolo da camada de transporte na arquitetura TCP/IP, do qual também faz parte o protocolo UDP. Apesar de utilizar-se do IP, que possui um serviço de datagrama não-confiável, o TCP fornece um serviço confiável de entrega de dados, é orientado à conexão, tem comunicação ponto a ponto e permite comunicação Full Duplex[19]. Para garantir a entrega dos pacotes e na ordem correta, o TCP utiliza-se de *buffers* para armazenar os pacotes, bem como procede a retransmissão de pacotes que tenham se perdido na rede. O programa aplicativo que se utiliza do TCP, solicita que

[19] Full Duplex – Permite que os dados trafeguem em ambos os sentidos, recebendo e enviando ao mesmo tempo.

seja aberta uma conexão com outro host e em seguida pode enviar ou receber dados na ordem certa e sem duplicação. Após o término da comunicação, o programa solicita ao TCP o término da conexão [COM00], [IBM00]. A descrição do protocolo TCP é feita pelo Information Sciences Institute em [RFC793] e aperfeiçoado por Jacobson e outros em [RFC1323].

Para determinar qual processo local está se comunicando com cada processo em outro host da rede e usando qual protocolo, o TCP utiliza-se de portas e soquetes. Portas são números de 16 bits usados para identificar para qual protocolo ou aplicativo deve entregar as mensagens. As portas TCP se classificam em portas bem conhecidas (well-know) e portas efêmeras. As portas bem conhecidas utilizam-se do intervalo 0-1023 dos números de portas TCP, são controladas e atribuídas pelo IANA[20] e destinam-se a utilizações específicas. Como exemplo, o Telnet utiliza a porta 23, o SMTP a 25, o POP usa a porta 110 e o ICMP a porta 29, segundo referência em [NTRK97], [IBM00], [RFC1060] e [RFC1340]. As portas efêmeras não são controladas pela IANA, podem ser usadas por processos e programas dos usuários e estão no intervalo entre 1024 e 65535.

A API[21] de soquetes ou sockets funciona na camada de Aplicação da arquitetura TCP/IP e serve de auxílio no envio e recepção de dados para diversos serviços TCP/IP como Telnet, SMTP e FTP. Inicialmente desenvolvida na Universidade da Califórnia em Berkeley, como componente da versão do sistema operacional BSD UNIX, acabou tornando-se padrão e adotada em vários outros sistemas operacionais, tendo inclusive uma versão específica para a plataforma Windows, a API Windows Sockets ou Winsock, conforme [NTRK97], [IBM00], [COM99] e [COM00].

2.6 Protocolo SMTP

Provavelmente uma das maiores utilizações das interligações inter-redes ou internets é o envio e recebimento de mensagens eletrônicas ou e-mail. O protocolo que executa esse serviço dentro da arquitetura TCP/IP é o SMTP[22]. O padrão para este

[20] IANA – Internet Assigned Number Authority – Grupo responsável pela atribuição de constantes utilizadas pelos protocolos TCP/IP.
[21] API – Application Programming Interfaces (Interfaces de Programação de Aplicativos)
[22] SMTP – Simple Mail Transfer Protocol (Protocolo de Transferência de Correio Eletrônico)

protocolo foi definido por Postel em [RFC0821], tendo sido aperfeiçoado por Klensin em [RFC2821].

Por definição, o protocolo SMTP é bastante simples, utilizando-se da porta TCP número 25 para efetuar a troca de informações entre o cliente e o servidor. O servidor SMTP, após efetivada a conexão, retorna uma mensagem “220 READY FOR MAIL”, indicando estar pronto para receber comandos do cliente SMTP, conforme mencionado em [COM00]. A partir dessa indicação, o servidor aguarda o recebimento de outros comandos do cliente, que serão processados e respondidos com uma mensagem correspondente, mencionado por [IBM00] e [COM00] e mostrado na Tabela 2.1, sendo que a definição precisa do formato das mensagens é explanada por Crocker em [RFC0822], aperfeiçoada por Braden em [RFC1123] e por Resnick em [RFC2822].

Tabela 2.1 – Mensagens do Protocolo SMTP

Emissor	Sentido	Receptor
Estabelece a conexão pela porta TCP número 25	→	
	←	220 <domínio do receptor> Serviço Pronto
HELO <domínio do emissor>	→	
	←	250 <domínio do receptor>
MAIL FROM: <endereço e-mail remetente>	→	
	←	250 OK
RCPT TO: <endereço e-mail destinatário>	→	
	←	250 OK
DATA	→	

Tabela 2.1 – Mensagens do Protocolo SMTP (Continuação)

Emissor	Sentido	Receptor
	←	354 Iniciar entrada de correio, terminar com <CRLF[23]>.<CRLF>
Linha 1	→	
Linha 2	→	
Linha n	→	
<CRLF>.<CRLF>	→	
	←	250 OK
QUIT	→	
	←	221 <domínio do receptor> encerrando conexão

2.7 CRC[24]

Entre as situações enfrentadas quando se trata de comunicação de dados está a questão da confiabilidade dos dados transmitidos. Existem muitas maneiras de se identificar e tratar essa questão, sendo que uma das maneiras adotadas é o uso do CRC.

Conforme [COM00], CRC é um valor inteiro de 32 bits que é calculado pelo dispositivo transmissor a partir dos dados a serem transmitidos e anexado ao fim da mensagem. Ao receber a mensagem, o dispositivo receptor calcula novamente o CRC e compara com o valor recebido juntamente com a mensagem. A igualdade nos valores do CRC calculado e lido representa que a mensagem transmitida na origem é a mesma que foi recebida no destino. [COM99] aponta que apesar de ser mais complexo calcular o CRC que outros mecanismos de controle de erro de transmissão, o mesmo é eficiente na detecção deste tipo de erro.

Os dispositivos de rede que são passíveis de gerenciamento, tais como roteadores[25], *switches*[26], hubs[27] e outros, possuem um registro da quantidade de erros

[23] CRLF – Seqüencia de caracteres CR (Carriage Return) + LF (Line Feed)

provenientes de diferenças no CRC nos pacotes recebidos. Essa quantidade de erros CRC pode ser utilizada como um indicativo de problemas na rede, sendo que o número excessivo de erros de CRC pode indicar a iminência de uma queda na comunicação da rede.

2.8 Colisões

As redes locais que adotam o padrão Ethernet, e que são, conforme [COM00], bastante populares, utilizam o esquema de acesso chamado CSMA/CD[28]. Esse esquema de acesso permite que apenas um equipamento transmita informações no cabo da rede, através da procura da onda portadora, processo esse chamado de *detecção de portadora* ou *carrier sense*. Quando um equipamento necessita transmitir um pacote, ele verifica se não existe outra mensagem sendo transmitida. Caso não identifique outra transmissão, o equipamento passa a transmitir a mensagem, caso contrário aguarda por um tempo aleatoriamente definido para novamente tentar transmitir a sua mensagem.

Pode acontecer, segundo [COM00], em algumas situações específicas, que dois equipamentos consigam transmitir ao mesmo tempo, o que caracteriza uma colisão. Esta colisão não prejudica o equipamento nem a rede, mas deteriora o sinal que está sendo transmitido. Cada equipamento faz um monitoramento do cabo de rede no sentido de identificar se algum sinal está interferindo na sua transmissão. Este monitoramento recebe o nome de *detecção de colisão* ou *collision detect*. A quantidade de colisões ocorridas é registrada por alguns equipamentos de rede, podendo servir de parâmetro para identificar possíveis problemas na rede ou em algum segmento da mesma.

2.9 Conclusão

A partir do conhecimento dos conceitos apresentados, será apresentada no próximo capítulo uma análise das necessidades de gerenciamento em uma rede de

---

[24] CRC – Cyclic Redundancy Check (Teste de Redundância Cíclico)
[25] Equipamento de rede, cuja função é estabelecer a rota dos datagramas de rede. Atua na camada 3 da arquitetura TCP/IP.
[26] Equipamento de rede, cuja função é segmentar a rede, melhorando a sua performance. Atua na camada 2 da arquitetura TCP/IP.
[27] Equipamento de rede, cuja função é servir de centralizador das ligações de rede. Atua na camada 2 da arquitetura TCP/IP.
[28] CSMA/CD – Carrier Sense Multiple Access with Collision Detect (Acesso Múltiplo com Detecção de Portadora e Detecção de colisão)

pequeno porte. Também serão analisadas as ferramentas normalmente disponíveis, bem como as dificuldades encontradas neste tipo de ambiente.

Capítulo 3

Gerência de Rede de Pequeno Porte

3.1 Introdução

Inicialmente é necessário que se apresente uma definição de rede de pequeno porte. A forma mais comum de classificação de redes é com respeito à distancia dos interprocessadores, gerando a classificação em Redes Locais (LAN[29]), Redes Metropolitanas (MAN[30]) e Redes Geograficamente Distribuídas (WAN[31]) [TAN 96]. Esta classificação, no entanto, não é suficiente para a caracterização do objeto alvo deste trabalho, pois pode-se ter uma rede local com muitos computadores interligados ou uma rede metropolitana com poucos computadores interligados.

A fim de caracterizar o conceito de rede de pequeno porte, necessário para o desenvolvimento deste trabalho, considera-se alguns pressupostos, tais como:

a) Levando em consideração o número de dispositivos de rede interligados, considera-se de pequeno porte uma rede com até 100 dispositivos interligados.

b) Apesar do crescimento do uso da plataforma Linux, principalmente como sistema operacional de servidores com o uso voltado à Internet, no ambiente de pequenas redes, considera-se a plataforma Windows como dominante, tanto como sistema operacional de servidor e estação quanto como pacote de automação de escritório (editor de texto, planilha eletrônica e software de apresentação)

Em um ambiente de pequenas redes, o gerenciamento geralmente é feito quase manualmente, sendo enfatizado o gerenciamento de falhas, trabalhando-se de forma corretiva para que o erro possa ser corrigido. Essa metodologia de gerenciamento dos recursos da rede é prejudicial à qualidade dos serviços prestados, pois dificilmente o

[29] LAN - Local Area Network (Rede Local)
[30] MAN - Metropolitan Area Network (Rede Metropolitana)
[31] WAN - Wide Area Network (Rede Geograficamente Distribuída)

dispositivo com falha pode ser substituído rapidamente de modo a suprir a falta do mesmo.

Outra característica importante das pequenas redes é a falta de automatização das rotinas de administração e gerenciamento da rede. A monitoração dos dispositivos é feita a partir de comandos digitados em *prompt* de comando e dos resultados obtidos na tela a partir da execução destes comandos.

Neste capítulo serão levantadas as necessidades de gerenciamento no ambiente de pequenas redes, quais processos de gerenciamento deveriam ser automatizados e quais as ferramentas geralmente disponíveis para propiciar a realização do que se deseja. Também serão discutidas quais as dificuldades inerentes a uma rede de pequeno porte e que possam interferir no processo de gerenciamento da mesma.

3.2 Necessidades de gerenciamento

De uma forma geral, as necessidades de gerenciamento de uma pequena rede são as mesmas de uma grande rede. O que faz a diferença é o número de objetos gerenciáveis ou passíveis de gerenciamento. A proximidade física ou geográfica entre os objetos, associada ao reduzido número de componentes a serem gerenciados, tende a facilitar o controle sobre eles. Observando a partir desta ótica, no cotidiano, tem-se um conjunto reduzido de necessidades de gerenciamento, tais como:

- Disponibilidade de um determinado componente
- Registro dos principais eventos
- Monitoração de informações relevantes
- Automatização de procedimentos
- Flexibilidade no Uso
- Custo acessível
- Idioma

Cada uma destas necessidades de gerenciamento é detalhada nos itens a seguir.

3.2.1 Disponibilidade de um determinado componente

A identificação da disponibilidade de um determinado componente da rede, ou seja, se um determinado componente da rede está ativo e operante, é a principal necessidade no gerenciamento de uma pequena rede.

Este dispositivo ou componente da rede pode ser um servidor, uma estação de trabalho, uma impressora, um roteador, um *switch*, um *hub*, uma linha de comunicação ou qualquer outro dispositivo considerado relevante.

Alguns desses dispositivos podem não ser gerenciáveis, ou seja, não serem capazes de executar um processo de gerenciamento SNMP[32] [TAN 96]. Essa característica dificulta o gerenciamento desses dispositivos, fazendo-se necessário encontrar uma forma de gerenciá-los.

3.2.2 Registro dos principais eventos

Alguns eventos devem ser armazenados em um arquivo de log, para posterior verificação. Estes eventos podem ser a entrada ou saída de um dispositivo da rede, um valor considerado anormal para uma das variáveis ou um erro ocorrido em algum dos dispositivos. Para cada um dos tipos de eventos, as informações a serem registradas podem variar, mas de uma forma geral as informações necessárias são:

- Data do evento;
- Hora do evento;
- Nome do dispositivo;
- Identificação lógica do dispositivo;
- Descrição do evento;

O arquivo de registro gerado deve estar num formato que possa ser facilmente aberto por qualquer aplicativo, visando facilitar a análise do mesmo.

3.2.3 Monitoração de informações relevantes

A monitoração consiste na observação de informações relevantes ao gerenciamento [SPEC01]. No processo de monitoração, existem dois tipos de componentes na rede: agentes e gerentes. O agente é um dispositivo que possui um conjunto de *software* destinado às tarefas de coletar informações sobre as atividades relacionadas à rede, armazenar estatísticas localmente e responder aos comandos do centro de controle da rede. O gerente é um dispositivo designado para as tarefas de controle da rede e possui uma coleção de *software* chamada Aplicação de Gerenciamento da Rede [SPEC01].

A informação de gerenciamento é coletada e armazenada por agentes e repassada para um ou mais gerentes. Duas técnicas podem ser utilizadas na comunicação entre agentes e gerentes: *polling* e *event-reporting*. A técnica de *polling* consiste em uma interação onde o gerente solicita ao agente o envio de um ou mais valores entre as informações armazenadas no agente. Na técnica de *event-reporting*, a iniciativa é do agente. O gerente fica aguardando pela informação a ser enviada pelo agente, que pode ser um relatório periódico, com intervalo de tempo previamente definido ou um relatório ocasional, relatando um erro ocorrido ou um evento significativo [SPEC01].

Considerando a pequena quantidade de dispositivos e o conseqüente pequeno tráfego de informações de gerenciamento na rede, utiliza-se a técnica de *polling* para o monitoramento das informações de uma rede de pequeno porte.

3.2.4 Automatização de procedimentos

Visando facilitar o processo de gerenciamento da rede, deve-se procurar automatizar o maior número possível de procedimentos. Notadamente, deve ser previsto o envio de alarmes referenciando situações de erro ou falha na rede. Tais alarmes devem ser dotados de critérios de análise para identificar o grau de importância do evento, bem como o grau de urgência em se tomar uma ação efetiva.

---

[32] SNPM - Simple Network Management Protocol - Protocolo de Gerenciamento de Rede Simples

Preferencialmente, deve ser identificada também qual ação deverá ser tomada, informando aos gerentes da rede através de mensagem na tela ou em arquivos do tipo log, bem como enviando uma mensagem de correio eletrônico (*e-mail*) informativo do evento.

3.2.5 Flexibilidade no Uso

Uma característica desejável no processo de gerenciamento é a flexibilização no uso da ferramenta. Isso representa a possibilidade de se ampliar ou reduzir os recursos de gerenciamento, como introdução de novos campos de dados, criação de novos relatórios de gerenciamento ou inclusão de novas rotinas de gerenciamento.

Em função das mudanças freqüentes na especificação dos equipamentos, protocolos de rede e também do surgimento de necessidades específicas, a possibilidade de se fazer uma manutenção nos processos e procedimentos da ferramenta de gerenciamento torna-se quase obrigatória. Essa manutenção deve ser facilitada para o administrador ou gerente da rede, proporcionando uma curva de aprendizagem suave do processo de manutenção, bem como simplificando o processo de alteração na estrutura da ferramenta.

3.2.6 Custo Acessível

Uma das características mais freqüentes nos ambientes de redes de pequeno porte é a escassez de recursos financeiros. Isto acontece em função de que, normalmente, essas redes estão servindo a empresas de pequeno porte ou faturamento e que não tem o uso da rede como objetivo maior do seu negócio.

Em função desse fato, o fator custo passa a ser altamente relevante no uso de uma ferramenta de gerenciamento da rede.

3.2.7 Idioma

No universo da Tecnologia da Informação, o idioma dominante é o inglês. Este fato deve-se a que a maioria dos livros, revistas, artigos e sites Internet que tratam sobre esse assunto é escrita na língua inglesa.

Ocorre que no ambiente de pequenas redes, nem sempre o administrador ou gerente de rede tem a fluência no idioma inglês suficiente para a compreensão plena dos termos e conceitos utilizados em um programa de gerenciamento.

Os arquivos de ajuda, as telas e mensagens apresentadas deveriam utilizar o idioma português, preferencialmente, visando facilitar o uso do programa.

Também para o caso de expansão ou manutenção de partes da ferramenta de gerenciamento, a linguagem de programação ou de macro deveria ser em português ou possuir documentação nessa língua, para que a atividade fosse facilitada e até incentivada para o administrador ou gerente da rede.

3.3 Ferramentas Disponíveis

Inicialmente, as principais ferramentas disponíveis ao administrador ou gerente da rede são os utilitários do sistema operacional. Cada sistema operacional dispõe de diferentes utilitários, sendo que alguns guardam uma certa semelhança, independentemente do sistema operacional utilizado. Notadamente os utilitários relacionados ao conjunto de protocolos TCP/IP[33], têm estas características. Considerando-se que o TCP/IP é a arquitetura de rede mais utilizada nos ambientes de pequeno porte, os seus comandos ou utilitários são aproveitados para efetivar o gerenciamento da rede. Apenas os utilitários ping e telnet serão mencionados, por serem os que se aplicam ao contexto estudado.

3.3.1 Ping

Ferramenta que faz parte do conjunto de protocolos TCP/IP e que ajuda a verificar a conectividade ao nível IP. Utilizado principalmente para verificar se um dispositivo de rede está conectado a uma rede TCP/IP e a seus recursos [NTRK97] [NDR95].

O comando ping envia pacotes do tipo *echo* ICMP para o dispositivo destino, escutando os pacotes de resposta e, desta maneira, verificando as conexões para o dispositivo remoto. É feita uma validação de cada pacote recebido em relação aos pacotes inicialmente transmitidos [NTRK97] [NDR95].

A sintaxe simplificada do comando é **ping** *endereço_IP* , onde *endereço_IP* é o endereço IP do dispositivo a ser verificado [NTRK97] [NDR95].

3.3.2 Telnet

Ferramenta que faz parte do conjunto de protocolos TCP/IP e que faz a emulação do terminal com um *host* remoto executando um serviço do servidor Telnet, utilizando os serviços baseados em conexão do TCP [NTRK97] [NDR95].

O Telnet utiliza os serviços baseados em conexão do TCP, proporcionando emulação do tipo VT 100, VT 52 ou TTY [NTRK97] [NDR95].

Os principais componentes ativos da rede, tais como roteadores, switches e hubs gerenciáveis, possuem serviços de servidor Telnet que fornecem informações relevantes sobre a situação dos mesmos.

Conforme mencionado em [COM99], o Telnet pode ser utilizado para acessar serviços TCP, como POP[34], SMTP[35] e DAYTIME[36], pois permite informar o endereço IP e a porta correspondente ao serviço a ser acessado.

Em cada tipo de dispositivo de rede, o servidor Telnet fornece uma resposta diferenciada, devendo-se conhecer a estrutura da mesma, a fim de interpretar os resultados apresentados. Da mesma maneira, cada serviço TCP tem a sua estrutura de comandos e respostas padrão, além de possuir porta de comunicação específica.

3.4 Dificuldades

Os utilitários e comandos mencionados no item anterior podem fornecer informações que supririam a necessidade de gerenciamento em uma rede de pequeno porte. A principal dificuldade na utilização dessas ferramentas é que as mesmas são interativas, pois requerem a digitação de seus parâmetros e também a observação na tela dos resultados obtidos.

---

[33] TCP/IP - Transfer Control Protocol / Internet Protocol
[34] POP – Post Office Protocol (Protocolo de Correio Eletrônico)
[35] SMTP – Simple Mail Transfer Protocol (Protocolo de Transferência de Correio Simples)
[36] DAYTIME – Serviço TCP/IP que imprime a data e a hora do dia

Também não existe a possibilidade de se programar a execução dessas ferramentas em um intervalo de tempo pré-definido, nem o registro automático dos resultados obtidos em um arquivo, para posterior análise.

Outra dificuldade patente é a impossibilidade, ao se usar somente as ferramentas citadas, de automatizar-se algum procedimento, baseado nas informações obtidas.

3.5 Conclusão

Conforme se pode observar a partir dos requisitos mínimos para o gerenciamento de uma rede de pequeno porte e, em particular, das dificuldades encontradas na utilização direta das ferramentas disponíveis, não se pode gerenciar uma rede, mesmo sendo de pequeno porte, somente com as ferramentas disponíveis no sistema operacional.

Desta forma, apresenta-se, no capítulo 4, um estudo comparativo das ferramentas de gerenciamento existentes, focalizando aquelas que atendam às necessidades descritas neste capítulo, enfatizando ainda as ferramentas que apresentam um baixo custo de implantação, tipicamente aquelas do tipo *freeware* e de código aberto.

Capítulo 4

Ferramentas de Gerenciamento

4.1 Introdução

O estudo de algumas das ferramentas de gerenciamento disponíveis mostrou-se necessário para que se pudesse verificar a existência de uma ferramenta que satisfizesse as necessidades básicas de gerenciamento de uma rede de pequeno porte, já descritas no capítulo 3. Neste capítulo são apresentadas algumas das ferramentas conhecidas, abrangendo produtos comerciais, de código aberto e do tipo *shareware* e *freeware*, sendo a seguir apresentadas as principais características de cada uma delas.

4.2 Ferramentas de Gerenciamento

Dentre as ferramentas disponíveis, foram selecionadas aquelas que mais se aplicam ao contexto estudado. Estas ferramentas são listadas a seguir e explanadas nos itens de 4.2.1 a 4.2.7.

4.2.1 WhatsUP Gold

Esta ferramenta é fabricada pela empresa IPSwitch, cujo site é http://www.ipswitch.com e foi analisada na versão 6.0. A sua plataforma de utilização é baseada em MS Windows, nas versões 9X, NT e 2000. Com respeito ao licenciamento, a sua utilização requer pagamento de aproximadamente U$[37] 700,00. O software é do tipo *AS IS*[38], sendo que o seu código fonte não está disponível. Toda a documentação do software está em língua inglesa.

A instalação do mesmo pode ser considerada fácil, ocupa pouco espaço em disco (9 MB) e não requer grande capacidade computacional para executar, sendo que o *hardware* mínimo recomendado é PC 486/66 MHz, 16 Mb de memória RAM (Win 9X) ou 32 MB de memória RAM (NT/2000)

[37] U$ - Valor expresso em moeda americana (dólar americano)
[38] AS IS (Como é) – Tipo de software cujo código fonte não é fornecido com o produto.

O WhatsUP possui uma boa interface, com fácil instalação e utilização. Permite mapear a rede, montar uma rede personalizada e monitorar os dispositivos da rede. Os mapas são criados através do varredura em redes TCP/IP, Novell NetWare IPX, e Microsoft NetBIOS.

Monitora dispositivos como computadores, servidores, roteadores, concentradores LAN (hubs ou switches) e impressoras, bem como as portas de serviços como SMTP, POP3, FTP, Telnet e WWW. Com relação ao *SNMP* ele permite uma boa utilização do serviço apresentando alguns serviços como: *get, getnext, getallsubitens*, monitor. Gera arquivos de log, contendo as informações obtidas, através de cada varredura do tipo *polling*.

4.2.2 AdventNet V5 Monitor

Esta ferramenta é fabricada pela Advent, cujo site é http://www.adventnet.com e foi analisada na versão 2.4. A sua plataforma de utilização é baseada em MS Windows, nas versões 9X, NT e 2000, além do Linux. Com respeito ao licenciamento, a sua utilização requer pagamento de aproximadamente U$ 7500,00. O software é do tipo *AS IS*, sendo que o seu código fonte não está disponível. Toda a documentação do software está em língua inglesa.

A instalação do mesmo pode ser considerada fácil, ocupa pouco espaço em disco e não requer grande capacidade computacional para executar, sendo que o hardware mínimo recomendado é computador com processador Pentium 166 MHz, 64 Mb de memória RAM e 10 MB de espaço disponível em disco.

O AdventNet V5 Monitor possui interface gráfica, possibilitando a geração de graficos representando a utilização dos recursos gerenciados. O software também pode ser usado para a criação de *applets java*, que podem ser visualizados em um WEB Browser e permitem a troca de informações de gerenciamento com agentes SNMP. Sua principal característica é o monitoramento das variáveis SNMP, que são obtidas através de varredura (*polling*) entre as sessões de monitoramento. Permite a geração de páginas web, dependendo de servidor WWW para tanto.

A ferramenta provê flexibilidade fixando filtros, baseados em condições sobre as variáveis SNMP obtidas e ativando ações baseadas nos dados apanhados, podendo ser a geração de arquivos do tipo log ou envio de e-mail.

4.2.3 Server Alive

Esta ferramenta é fabricada pela Woodstone Computer Consulting, cujo site é http://www.woodstone.nu e foi analisada na versão 2.0.70333. A sua plataforma de utilização é baseada em MS Windows, nas versões 9X, NT e 2000. Com respeito ao licenciamento, ele é to tipo *freeware* para gerenciamento de até 10 hosts; acima destes o seu licenciamento requer pagamento de aproximadamente U$ 70,00, além de contrato de suporte de aproximadamente U$ 60,00, caso se deseje adquirir. O software é do tipo *AS IS*, sendo que o seu código fonte não está disponível. Toda a documentação do software está em língua inglesa.

A instalação do mesmo pode ser considerada fácil, ocupa pouco espaço em disco e não requer grande capacidade computacional para executar, sendo que os teste foram realizados em um computador com processador Pentium 166 e 32 MB de memória RAM.

A ferramenta possui recursos de verificação de disponibilidade via comandos ICMP (Ping) e permite utilizar-se de comandos TCP ou UDP (Telnet) para monitorar serviços que se utilizem destes protocolos, incluindo-se alguns jogos. Também possui acesso a informações obtidas pelo protocolo SNMP, sem permitir que se realizem operações sobre os hosts gerenciados. Permite a utilização de um gerenciador de banco de dados para armazenar as informações obtidas ou pode gerar arquivos do tipo log. Também possibilita que se defina qual o intervalo de varredura (*polling*) entre as sessões de monitoramento.

4.2.4 Network View

Esta ferramenta é fabricada pela Network View, cujo site é http://www.networkview.com e foi analisada na versão 1.1. A sua plataforma de utilização é baseada em MS Windows, nas versões 9X, NT e 2000. Com respeito ao licenciamento, a sua utilização requer pagamento de aproximadamente U$ 30,00. O

software é do tipo *AS IS*, sendo que o seu código fonte não está disponível. Toda a documentação do software está em língua inglesa.

A instalação do mesmo pode ser considerada fácil, ocupa pouco espaço em disco e não requer grande capacidade computacional para executar, sendo que o hardware mínimo recomendado é computador com processador 486 DX4/100 MHz, 16 Mb de memória RAM e 5 Mb livre em disco.

O Network View possui interface gráfica, representando toda a rede de maneira esquemática com cada tipo de dispositivo com uma figura específica. Para gerar o diagrama da rede, a ferramenta pode efetuar uma varredura na rede, identificando os dispositivos através de seus endereços IP. A ferramenta armazena, para cada nó da rede, informações das portas TCP definidas, disponibilidade (através de Ping) e informações SNMP, caso estejam disponíveis. Faz a varredura do tipo polling, podendo ter o tempo configurado, gerando relatórios ou enviando e-mail, caso alguma anormalidade seja identificada. Não permite operações SNMP sobre os objetos gerenciados, nem possui visualizador (browser) das MIB's.

4.2.5 MRTG

Esta ferramenta foi desenvolvida por Tobias Oetiker além de muitos contribuintes, e está disponível no site do próprio MRTG[39] e foi analisada na versão 2.8.12. A sua plataforma de utilização é baseada em Linux, podendo ser executado em MS Windows, nas versões 9X, NT e 2000. Com respeito ao licenciamento, o software é do tipo *freeware* e *open source*, não necessitando pagamento para sua utilização. O código fonte está disponível, sendo que foi desenvolvido utilizando-se as linguagens C e Perl. Toda a documentação do software está em língua inglesa, sendo que é fornecido um script de tradução de algumas partes do menu, existindo uma versão em português.

A instalação do mesmo pode ser considerada difícil, devido à necessidade de se recompilar o fonte, no caso do Linux ou necessitar da instalação de interpretador Perl, no caso de utilização com o Windows. A ferramenta ocupa bastante espaço em disco, mas não requer grande capacidade computacional para executar, sendo que o hardware

[39] Site do MRTG - http://ee-staff.ethz.ch/~oetiker/webtools/mrtg/mrtg.html

mínimo recomendado é computador com processador 486 DX4/100 MHz, 16 Mb de memória RAM.

Sua principal função é monitorar o tráfego na rede, a utilização de interfaces, links de redes, utilização dos modems, utilização de *CPU* e qualquer outra variável numérica de equipamentos que suportem características de gerenciamento SNMP. Ele gera automaticamente páginas HTML com imagens *GIF* atualizadas em um determinado período de tempo. Estas páginas representam os dados obtidos dos dispositivos gerenciados, portanto é aconselhável ser usado juntamente com um servidor WEB para facilitar as consultas aos gráficos de monitoramento.

A ferramenta utiliza-se do SNMP para ler as informações dos dispositivos gerenciados e programas escritos em linguagem *C* para montar os gráficos, portanto para bom funcionamento da mesma, todos os hosts devem possuir suporte a este protocolo.

4.2.6 Netsaint

Esta ferramenta foi desenvolvida por Ethan Galstad além de muitos contribuintes, e está disponível no site http://www.netsaint.org e foi analisada na versão 0.0.5. A sua plataforma de utilização é baseada em Linux e em outras versões Unix, não existindo versão para o MS Windows. Com respeito ao licenciamento, o software é do tipo *freeware* e *open source*, não necessitando pagamento para sua utilização. O código fonte está disponível, sendo que foi desenvolvido utilizando-se a linguagem C e alguns módulos em Perl. Toda a documentação do software está em língua inglesa.

A instalação do mesmo pode ser considerada fácil, ocupa pouco espaço em disco e não requer grande capacidade computacional para executar, não sendo documentado o hardware mínimo recomendado, contudo os testes foram executados em computador com processador Pentium 166 e 32 MB de memória RAM.

NetSaint é um *software* de monitoramento de serviços de rede escrito em C. Apresenta alguns módulos escritos em Perl (CGI`s) que lhe permitem ver o estado atual e histórico de um determinado h*ost*, por meio de uma interface Web, necessitando para tanto de um servidor WWW.

Ele monitora diversos serviços de rede como SMTP, POP3 e HTTP, além de identificar a disponibilidade de um recurso gerenciado através do protocolo ICMP (PING), registrando os eventos ocorridos em arquivo do tipo log. Trabalha com a estrutura de rede de uma forma hierárquica, permitindo notificações caso ocorram situações anormais, podendo enviar mensagens via e-mail, pager, ou alarme sonoro. Também monitora os recursos locais do computador onde está instalado, como carga do processador e uso de disco ou memória.

4.2.7 LetUknow

Esta ferramenta é fabricada pela MH Software, cujo site é http://www.mhsoftware.co.uk e foi analisada na versão 1.03. A sua plataforma de utilização é baseada em MS Windows, nas versões 9X, NT e 2000. Com respeito ao licenciamento, o software é to tipo *shareware* e o seu registro requer pagamento de aproximadamente U$ 89,00. O software é do tipo *AS IS*, sendo que o seu código fonte não está disponível. Toda a documentação do software está em língua inglesa.

A instalação do mesmo pode ser considerada fácil, ocupa pouco espaço em disco e não requer grande capacidade computacional para executar, não sendo documentado o hardware mínimo recomendado, contudo os testes foram executados em computador com processador Pentium 166 e 32 MB de memória RAM.

O LetUknow tem como sua principal função monitorar a rede local e os seus dispositivos.Com o LetUknow tem-se um sistema simples de monitoramento da rede que pode ser agendado previamente e executado automaticamente, através da definição dos intervalos de varredura. Possui o recurso de emitir avisos para os administradores de rede através de e-mail, mensagem pela rede ou cópia impressa, sobre qualquer problema na rede, além de possuir registro em arquivo do tipo log.

A cada varredura a ferramenta testa todos os nós da rede, através de pacotes ICMP, checa espaço disponível em volumes compartilhados, disponibilidade de caminhos UNC, além de outras verificações que podem ser programadas no *software*.

4.3 Conclusão

Cabe ressaltar que nem todas as ferramentas foram objeto de análise neste trabalho, pois são inúmeras. Não foram consideradas, neste estudo, as ferramentas comerciais, voltadas a redes de grande porte, assim como as ferramentas de gerenciamento voltadas ao monitoramento de recursos muito específicos, como espaço em disco, leitura de variáveis SNMP ou tráfego de rede. Também deve ser mencionado que nem todos os *softwares* estudados foram mencionados aqui; apenas aqueles que poderiam ter a possibilidade de suprir as necessidades de gerenciamento observadas.

Em virtude de nenhuma das ferramentas, individualmente, ter atendido plenamente às premissas de gerenciamento mencionadas, o próximo capítulo tratará da proposta de um modelo de gerenciamento simplificado, voltado para redes de pequeno porte.

Capítulo 5

Um Modelo de Gerenciamento para Pequenas Redes

5.1 Introdução

Conforme observado no estudo de ferramentas de gerenciamento, descrito no capítulo 4, não se pôde identificar uma ferramenta que pudesse satisfazer os requisitos de gerenciamento de redes de pequeno porte, apresentados no capítulo 3.

Neste capítulo é apresentada uma proposta de modelo de gerenciamento que se propõe a cumprir os requisitos de gerenciamento de redes de pequeno porte, bem como resolver as dificuldades no gerenciamento, também apresentadas no capítulo 3.

O modelo proposto busca atender os seguintes requisitos:

a) Baixa complexidade

b) Baixo custo

c) Flexibilidade no uso

d) Facilidade de instalação

e) Facilidade de operação

f) Idioma Português

A apresentação do modelo proposto segue as seguintes etapas:

- **Estrutura do modelo proposto** mostra quais são as principais características do modelo, descrição do ambiente previsto no modelo, diagrama esquemático, bem como o relacionamento entre os diversos módulos do mesmo.

- **Variáveis a serem gerenciadas** descreve os parâmetros considerados importantes para o gerenciamento de uma pequena rede.

- **Processos a serem automatizados** enumera e apresenta as rotinas de gerenciamento passíveis de automatização, bem como descreve como se processa a automatização.

- **Implementação** é onde se mostra, de maneira prática, como o modelo pode ser implantado, mostrando as telas da aplicação, os detalhes de parametrização e a descrição das rotinas de gerenciamento.

5.2 Estrutura do modelo proposto

O modelo está organizado em seis componentes ou módulos, cada um com a seu objetivo definido, mas mantendo um relacionamento com os demais módulos, baseando-se nas informações armazenadas ou obtidas através dos mesmos. O inter-relacionamento entre os módulos é representado pela Figura 5.1.

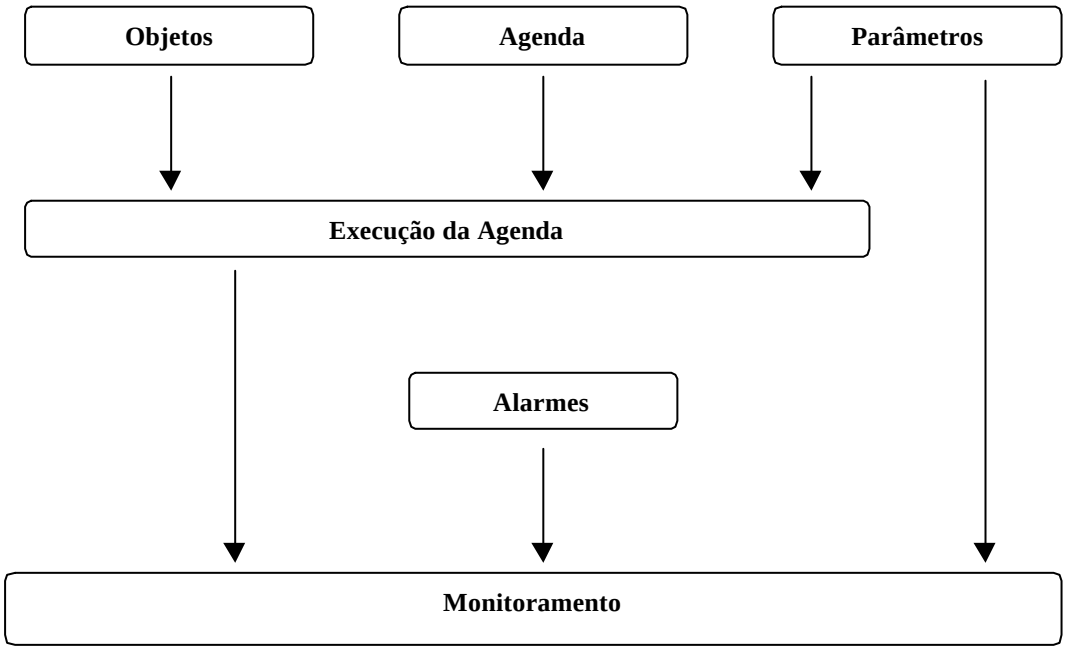

Figura 5.1 – Diagrama esquemático do modelo proposto

Cada um dos módulos será apresentado a seguir, descrevendo o seu funcionamento, seus objetivos e suas contribuições para o funcionamento do modelo.

5.2.1 Módulo de Cadastro dos Objetos Gerenciados

Este módulo deve ser o primeiro a ser planejado e preenchido, pois é baseado principalmente nos dados contidos nele que os demais módulos poderão executar o seu trabalho. As informações a serem cadastradas, bem como a descrição das mesmas estão relacionadas na Tabela 5.1

Tabela 5.1 – Campos do cadastro de Objetos Gerenciados

CAMPO	DESCRIÇÃO
CODIGO	Código sequencial que identifica o objeto. Não pode ser duplicado. Este campo é de preenchimento obrigatório.
NOME	Nome do objeto. Será utilizado nos registros de log e nas mensagens de alarme e aviso. É recomendado que seja um nome que identifique o objeto perante a rede. Este campo é de preenchimento obrigatório.
DESCRICAO	Pode ser utilizado este campo para descrever melhor o equipamento, visando um maior detalhamento na documentação.
TIPO	Campo destinado à identificação do tipo do objeto gerenciado. A finalidade deste campo é auxiliar na identificação do objeto. Ex.: HOST, ROUTER, PLACA_REDE, PORTA_WAN, PORTA_LAN, WWW_SERVER, SMTP_SERVER e outros.
MARCA	Marca ou fabricante do objeto gerenciado. A finalidade deste campo é auxiliar na identificação do objeto.
MODELO	Modelo do objeto gerenciado. A finalidade deste campo é auxiliar na identificação do objeto

Tabela 5.1 – Campos do cadastro de Objetos Gerenciados (Continuação)

CAMPO	DESCRIÇÃO
LIGACAO	Este campo identifica a qual outro objeto esse objeto está ligado. Utilizado para representar a hierarquia dos objetos, considerando a ligação ao objeto superior. Caso não haja ligação com outro objeto, repetir a informação do campo CODIGO. Este campo é de preenchimento obrigatório.
ENDERECO_IP	Endereço IP do objeto, segundo a forma decimal com pontos, como no exemplo 192.168.0.10. Este campo é de preenchimento obrigatório.
PORTA_TCP	Este campo identifica qual a porta TCP utilizada pelo objeto gerenciado. Este campo é de preenchimento obrigatório, a não ser para objetos que não se utilizem de porta TCP
PARAM01 a 09	Estes nove campos destinam-se ao envio de informações, comandos ou senhas a serem enviados para o dispositivo, na seqüencia correta de execução, sendo que cada um deles representa uma linha completa de informação, comando ou senha.
PARAM10	Este campo fica reservado para indicar a expansão da quantidade de parâmetros neste cadastro.

5.2.2 Módulo de Agendamento de Tarefas

Neste módulo serão alimentados os dados referentes ao agendamento das tarefas de gerenciamento a serem aplicadas sobre os objetos cadastrados no Módulo de Cadastro de Objetos. A partir das informações contidas nesta tabela é que o gerenciamento efetivamente vai acontecer. As informações a serem cadastradas, bem como a descrição das mesmas estão relacionadas na Tabela 5.2. O detalhamento do campo PAR1 encontra-se na Tabela 5.3 e o detalhamento do campo PAR2 encontra-se na Tabela 5.4.

Tabela 5.2 – Campos do Cadastro de Agendamento de Tarefas

CAMPO	DESCRIÇÃO
CÓDIGO	Código sequencial que identifica o item da agenda. Não pode ser duplicado. Este campo é de preenchimento obrigatório.
EVENTO	Especifica qual o tipo de evento de gerenciamento será utilizado. Este campo é de preenchimento obrigatório.
RECORRENCIA	Informa qual o tipo de recorrência a ser aplicada sobre o item da agenda. As opções previstas são UNICA, MIN, HORA, DIA, SEM, MES e ANO. Este campo é de preenchimento obrigatório.
A_CADA	Indica a peridiocidade em que o evento de gerenciamento irá acontecer. Está diretamente relacionando à opção escolhida no campo RECORRENCIA. Este campo é de preenchimento obrigatório, exceto quando o campo RECORRENCIA for definido como UNICA.
PAR1	Este campo tem o seu significado dependente da opção escolhida no campo RECORRENCIA, conforme nos mostra a Tabela 5.3.
PAR2	Este campo tem o seu significado dependente da opção escolhida no campo RECORRENCIA, conforme nos mostra a Tabela 5.4.
OBJETO	Código do objeto a ser gerenciado. Deve ser utilizado o valor contido no campo CODIGO da tabela OBJETOS. Este campo é de preenchimento obrigatório.
ALARME	Caso seja necessário definir um alarme ou aviso neste evento de gerenciamento, incluir neste campo o alarme definido na Tabela Alarmes, conforme conteúdo do campo TIPO.

Tabela 5.3 – Detalhamento do Campo PAR1 da Tabela AGENDA

OPÇÃO	DESCRIÇÃO	OBRIG.
UNICA	Informar a data de execução do evento, no formato dd/mm/aa.	SIM
MIN	Deixar em branco	NÃO
HORA	Informar em qual minuto de cada hora programada o evento deverá ser executado, no formato mm.	SIM
DIA	Informar o horário de execução do evento, no formato hh:mm	SIM
SEM	Informar o dia da semana em que o evento será executado, com um número, sendo 1-Domingo, 2-Segunda-feira, 3-Terça-feira, 4-Quarta-feira, 5-Quinta-feira, 6-Sexta-feira e 7-Sábado.	SIM
MÊS	Informar o dia do mês em que o evento será executado, no formato dd.	SIM
ANO	Informar a data de execução do evento, no formato dd/mm/aa.	SIM

Tabela 5.4 – Detalhamento do Campo PAR2 da Tabela AGENDA

OPÇÃO	DESCRIÇÃO	OBRIG.
UNICA	Informar a hora de execução do evento, no formato hh:mm.	SIM
MIN	Deixar em branco	NÃO
HORA	Deixar em branco	NÃO
DIA	Deixar em branco	NÃO
SEM	Informar a hora de execução do evento, no formato hh:mm.	SIM

Tabela 5.4 – Detalhamento do Campo PAR2 da Tabela AGENDA (Cont.)

OPÇÃO	DESCRIÇÃO	OBRIG.
MÊS	Informar a hora de execução do evento, no formato hh:mm.	SIM
ANO	Informar a hora de execução do evento, no formato hh:mm.	SIM

5.2.3 Módulo de Configuração dos Alarmes e Avisos

Neste módulo serão alimentados os dados referentes às definições dos alarmes e avisos. Os alarmes e avisos serão definidos em função dos resultados obtidos nas tarefas de gerenciamento executadas sobre os objetos gerenciáveis.

Também neste módulo serão definidas as variáveis a serem monitoradas e quais os valores ou intervalo de valores dentro dos quais serão emitidos avisos ou alarmes. Neste modelo são consideradas apenas dois níveis de gravidade nos alarmes, sendo o aviso com menor grau de gravidade e o alarme com maior grau de gravidade.

As informações a serem cadastradas, bem como a descrição das mesmas estão relacionadas na Tabela 5.5. Os operadores de comparação a serem utilizados estão descritos na Tabela 5.6 e os operadores lógicos na Tabela 5.7.

Tabela 5.5 – Campos do Cadastro de Alarmes e Avisos

CAMPO	DESCRIÇÃO
CÓDIGO	Código sequencial que identifica o item de alarme. Não pode ser duplicado. Este campo é de preenchimento obrigatório.
TIPO	Identificação do tipo de aviso ou alarme definido, podendo ser usado letras, números, traço (-) ou underscore (_). Pode haver mais de uma linha na tabela com o mesmo identificador de tipo, relacionando mais de uma variável a ser monitorada. Este campo é de preenchimento obrigatório.
VAR	Identificação da variável a ser monitorada, podendo ser usado letras, números, traço (-) ou underscore (_). Não podem existir dois registros neste cadastro com o campo TIPO e VAR idênticos, pois os dois campos compõem a chave primária do mesmo. Este campo é de preenchimento obrigatório.
DEPOIS	Conjunto de caracteres que se apresenta, de maneira fixa, antes do valor da variável monitorada. O texto selecionado não deve sofrer repetição dentro da resposta do gerenciamento . Este campo é de preenchimento obrigatório.
ANTES	Conjunto de caracteres que se apresenta, de maneira fixa, depois do valor da variável monitorada. O texto seleconado não deve sofrer repetição dentro da resposta do gerenciamento. Este campo é de preenchimento obrigatório.
AV_COMP1	Operador de comparação a ser aplicado entre o valor da variável monitorada e o campo AV_VALOR1. Os operadores de comparação a serem utilizados estão descritos na Tabela 5.6. Este campo é de preenchimento obrigatório, caso se queira definir um aviso.

Tabela 5.5 – Campos do Cadastro de Alarmes e Avisos (Continuação)

CAMPO	DESCRIÇÃO
AV_VALOR1	Valor a ser comparado com a variável para definir a execução da ação de aviso. Este campo é de preenchimento obrigatório, caso se queira definir um aviso.
AV_LOGICO	Operador lógico a ser aplicado em caso de utilização de um intervalo de valores para monitorar a variável. Os operadores de comparação a serem utilizados estão descritos na Tabela 5.7 Este campo é de preenchimento obrigatório, caso se queira definir um intervalo de valores a serem comparados.
AV_COMP2	Operador de comparação a ser aplicado entre o valor da variável monitorada e o campo AV_VALOR2. Os operadores de comparação a serem utilizados estão descritos na Tabela 5.6. Este campo é de preenchimento obrigatório, caso se queira definir um intervalo de valores para determinar a execução da ação de aviso.
AV_VALOR2	Valor a ser comparado com a variável para definir a execução da ação de aviso. Este campo é de preenchimento obrigatório, caso se queira definir um intervalo de valores para determinar a execução da ação de aviso.

Tabela 5.5 – Campos do Cadastro de Alarmes e Avisos (Continuação)

CAMPO	DESCRIÇÃO
AV_ACAO	Ação de aviso a ser executada, caso a comparação definida pelo campo AV_COMP1 entre o valor da variável e o campo AV_VALOR1 resulte em verdadeiro. Caso seja definido o uso do intervalo de valores, será aplicado o operador lógico definido no campo AV_LOGICO sobre o resultado da primeira comparação e a comparação definida pelo campo AV_COMP2 entre o valor da variável e o campo AV_VALOR2. Caso o resultado da operação lógica seja verdadeiro, será executada a ação de aviso definida neste campo. Este campo é de preenchimento obrigatório, caso se queira definir um aviso.
AV_DEST	Indicação do endereço do destinatário da mensagem de aviso. Este campo é de preenchimento obrigatório, caso a ação de aviso definida no campo AV_ACAO requeira um destinatário.
AL_COMP1	Operador de comparação a ser aplicado entre o valor da variável monitorada e o campo AL_VALOR1. Os operadores de comparação a serem utilizados estão descritos na Tabela 5.6. Este campo é de preenchimento obrigatório, caso se queira definir um alrme.
AL_VALOR1	Valor a ser comparado com a variável para definir a execução da ação de alarme. Este campo é de preenchimento obrigatório, caso se queira definir um alarme.

Tabela 5.5 – Campos do Cadastro de Alarmes e Avisos (Continuação)

CAMPO	DESCRIÇÃO
AL_LOGICO	Operador lógico a ser aplicado em caso de utilização de um intervalo de valores para monitorar a variável. Os operadores de comparação a serem utilizados estão descritos na Tabela 5.7 Este campo é de preenchimento obrigatório, caso se queira definir um intervalo de valores a serem comparados.
AL_COMP2	Operador de comparação a ser aplicado entre o valor da variável monitorada e o campo AL_VALOR2. Os operadores de comparação a serem utilizados estão descritos na Tabela 5.6. Este campo é de preenchimento obrigatório, caso se queira definir um intervalo de valores para determinar a execução da ação de alarme.
AL_VALOR2	Valor a ser comparado com a variável para definir a execução da ação de alarme. Este campo é de preenchimento obrigatório, caso se queira definir um intervalo de valores para determinar a execução da ação de alarme.
AL_ACAO	Ação de aviso a ser executada, caso a comparação definida pelo campo AL_COMP1 entre o valor da variável e o campo AL_VALOR1 resulte em verdadeiro. Caso seja definido o uso do intervalo de valores, será aplicado o operador lógico definido no campo AL_LOGICO sobre o resultado da primeira comparação e a comparação definida pelo campo AL_COMP2 entre o valor da variável e o campo AL_VALOR2. Caso o resultado da operação lógica seja verdadeiro, será executada a ação de aviso definida neste campo. Este campo é de preenchimento obrigatório, caso se queira definir um alarme.

Tabela 5.5 – Campos do Cadastro de Alarmes e Avisos (Continuação)

CAMPO	DESCRIÇÃO
AL_DEST	Indicação do endereço do destinatário da mensagem de aviso. Este campo é de preenchimento obrigatório, caso a ação de alarme definida no campo AL_ACAO requeira um destinatário.

Tabela 5.6 – Operadores de Comparação

OPERADOR	DESCRIÇÃO
=	Igual a
<	Menor que
<=	Menor que ou Igual a
>	Maior que
>=	Maior que ou Igual a
<>	Diferente de

Tabela 5.7 – Operadores Lógicos

OPERADOR	DESCRIÇÃO
AND	Operação lógica AND
OR	Operação lógica OR

5.2.4 Módulo de Cadastro dos Parâmetros Gerais

Neste módulo serão alimentados os dados referentes aos parâmetros gerais de funcionamento do modelo. O cadastro de Parametros foi definido utilizando-se o conceito de Palavra-Chave ou Variável para identificar o parâmetro. As quantidades e nomes das palavras-chave podem variar dependendo da forma de implementação do modelo proposto.

A estrutura do cadastro de Parâmetros é mostrada na Tabela 5.8.

Tabela 5.8 – Campos do Cadastro de Parâmetros

CAMPO	DESCRIÇÃO
VARIAVEL	Nome da variável que identifica o parâmetro a ser informado. As opções de variável previstas estão descritas na Tabela 5.9. Este campo é de preenchimento obrigatório.
DESCRICAO	Descreve o parâmetro, facilitando a compreensão do mesmo
VALOR	Valor a ser atribuído ao parâmetro definido no campo VARIAVEL. Pode ser de tipo numérico ou caracter, dependendo do parâmetro associado. Este campo é de preenchimento obrigatório.

5.2.5 Módulo de Execução da Agenda

A função deste módulo é efetuar a varredura no cadastro de Agendamento, verificando a data e hora atual e analisando se o evento de gerenciamento deve ser executado. Caso deva ser executado, este módulo deve verificar qual a rotina de gerenciamento vai ser aplicada e fazer a passagem dos parâmetros adequados à execução da mesma, conforme definidos no cadastro de Objetos. Também é função deste módulo receber as informações obtidas das tarefas de gerenciamento e repassar as mesmas para o módulo de Monitoramento.

5.2.6 Módulo de Monitoramento dos Resultados

Este módulo tem como função receber as informações obtidas das rotinas de gerenciamento, fazer a comparação com as variáveis definidas no cadastro de Alarmes e, caso necessário, executar a ação de aviso ou alarme, conforme tenha sido definido.

5.3 Variáveis a serem gerenciados

As variáveis a serem gerenciadas, bem como os valores limítrofes de cada uma delas são dependentes do tipo de dispositivo ou serviço objeto do gerenciamento. Através do cadastro de Alarmes, o modelo propõe que a definição das variáveis e dos valores passíveis de aviso ou alarme, seja feita específicamente para cada dispositivo ou serviço a ser gerenciado.

5.4 Processos a serem automatizados

Com a utilização do Módulo de Execução da Agenda, todo o processo de periodicidade no gerenciamento fica automatizado, bastando alterar o campo RECORRENCIA no Cadastro de Agendamento para que o evento seja reprogramado. Sempre que um objeto gerenciado estiver sob suspeita de mal funcionamento ou necessitar de um monitoramento mais freqüente, pode-se definir um intervalo de tempo menor entre cada sessão de gerenciamento. Em contrapartida, um objeto gerenciado que não necessite de observação freqüente, pode ter as suas sessões de gerenciamento mais espaçadas no tempo. O tempo mínimo de intervalo entre sessões de gerenciamento para um determinado objeto é um minuto e o tempo máximo pode chegar até vários anos.

O Módulo de Monitoramento, em conjunto com o Cadastro de Alarmes, permite observar o comportamento das variáveis, emitindo avisos ou alarmes que podem ser repassados para a execução das ações definidas nos campo AV_ACAO e AL_ACAO do Cadastro de Alarmes.

5.5 Implementação

A idéia de se implementar o modelo de gerenciamento proposto no item 5.2 deste capítulo é demostrar a viabilidade de realizar o que foi proposto, de modo a atingir os objetivos mencionados no Capítulo 3.

5.5.1 Ferramentas utilizadas

As ferramentas necessárias à implementação do modelo serão descritas a seguir. Para a implementação em si, apenas o Microsoft Excel foi utilizado. As demais ferramentas foram auxiliares na obtenção de informações necessárias à implementação, principalmente na obtenção do padrão de respostas dos dispositivos aos comandos efetuados.

5.5.1.1 Microsoft Excel

Conforme já mencionado no Capítulo 1, a principal ferramenta utilizada para implementação do modelo proposto foi o MS Excel 97, particularmente a sua linguagem de macros VBA[40]. A escolha desta ferramenta deu-se em função da ampla utilização da mesma, pela facilidade de dispor de toda a documentação on-line, inclusive do VBA, em português e pelo conhecimento geral da linguagem Basic, nas suas diversas versões (caracter, VB e VBA). Também foi fator determinante na escolha desta ferramenta o fato do VBA permitir o acesso à API Winsock, que facilitará a automatização das rotinas de PING, TELNET e do envio automático de e-mail, utilizando-se do protocolo SMTP.

Como fonte de consulta durante a implementação do modelo, foi utilizada a função de Ajuda do Microsoft Excel, tanto nas funções de planilha quanto no uso da linguagem de macros VBA. Na construção das rotinas de PING, TELNET e envio de e-mail, foram preciosas as informações obtidas em [COM97], bem como as orientações e dicas obtidas em [MCO02] e [VBIP02].

5.5.1.2 Ping

O aplicativo PING encontra-se na grande maioria das implementações dos protocolos TCP/IP, inclusive nos Sistemas Operacionais da família Windows, pois o protocolo no qual é baseado, o ICMP é obrigatório ao se implementar o TCP/IP, conforme [IBM00]. O mesmo é utilizado, tradicionalmente, para determinar se um determindo host está alcançável ou disponível. A utilização do mesmo na implementação do modelo será a anteriormente descrita, além de permitir obter o tempo de ida e volta do pacote.

[40] VBA – Visual Basic for Applications

5.5.1.3 Telnet

A utilização do aplicativo Telnet será preponderante, pois o mesmo permite acessar dispositivos de rede, serviços e aplicativos baseados no protocolo TCP, quer seja através de sua porta TCP padrão (21), ou através de qualquer outra porta TCP.

Através do Telnet, podemos enviar comandos para os objetos gerenciados e obter deles a resposta, podendo analisá-las e definir as variáveis a serem gerenciadas. A importância do uso do Telnet reside no fato de que cada objeto gerenciado, dependendo do serviço TCP utilizado e das suas características próprias, tem comandos específicos e respostas específicas a cada um dos comandos enviados. O Telnet permite que se possa salvar as informações trocadas em um arquivo, para que possam ser analisadas *a posteriore*, facilitando a execução dessa tarefa.

5.5.2 Eventos de gerenciamento

Na implementação do modelo, foram utilizadas dois eventos de gerenciamento, o PING-J e o TELNET-J. A escolha destes dois eventos de gerenciamento se deu em função de que, através deles, pode-se suprir as necessidades de gerenciamento de uma rede de pequeno porte, mencionadas no Capítulo 3.

Os eventos PING-J e TELNET-J foram implementados através da API Winsock, versão 2.2, sendo que as definições e chamadas às API's foram feitas utlizando-se o VBA, a partir do arquivo Timer.xls. O evento PING-J segue as especificações do protocolo ICMP, pois utiliza-se do evento *Echo Reply* para seu funcionamento. O evento TELNET-J utiliza-se de uma conexão TCP, aberta através da porta TCP definida no Cadastro de Objetos Gerenciados, no campo PORTA_TCP.

5.5.3 Variáveis a serem gerenciados

Através do evento de gerenciamento PING-J, pode-se trabalhar com todas as informações definidas pelo protocolo ICMP, sendo que nesta implementação trabalhamos apenas com o tempo de resposta, expresso em milisegundos (ms). Ao se obter a resposta do dispositivo ao evento PING-J também é possível certificar-se da conectividade de um determinado elemento da rede.

Na utilização do evento de gerenciamento TELNET-J, a diversidade de variáveis é bem maior, dependendo do dispositivo ou serviço a ser acessado, pois qualquer equipamento ou serviço que se utilize do protocolo TCP e que possua uma porta TCP definida, pode ser gerenciado a partir do modelo proposto. As variáveis monitoradas nesta implementação são: quantidade de erros de CRC, quantidade de colisões e disponibilidade do dispositivo de rede, porta de comunicação ou serviço TCP, identificada a partir da resposta, sem ocorrência de erros, à conexão TCP. A escolha pela utilização destas variáveis segue a análise feita no Capítulo 3, mas podem ser adotadas outras variáveis, conforme a necessidade.

O processo de determinação da variável a ser gerenciada passa por uma análise dos dados obtidos do dispositivo gerenciado. Para esta implementação, foi utilizado o programa Telnet, disponível no Windows 95, aplicado aos objetos gerenciados. Um exemplo dos dados obtidos pode ser verificado no Anexo B, onde são apresentados dados de roteadores, switches, servidores POP e SMTP.

Nesta implementação foi considerado que os valores a serem obtidos pelas variáveis de gerenciamento devem permitir a transformação em valores numéricos, podendo ser constituídos apenas dos caracteres que indicam quantidade (*0*, *1*, *2*, *3*, *4*, *5*, *6*, *7*, *8* e *9*), além dos caracteres que delimitam as milhares (.) e decimais (,).

5.5.4 Cadastro de Parâmetros

Na modelagem proposta, o Cadastro de Parâmetros tem a funcionalidade de eatabelecer definições gerais que podem ser alteradas de acordo com a forma de implementação do modelo. A relação das palavras-chave ou variáveis adotadas nesta implementação do modelo é apresentada na Tabela 5.9.

Tabela 5.9 – Relação de Palavras-Chave do Cadastro de Parametros

PALAVRA-CHAVE	DESCRIÇÃO
QTD_TIMEOUT	Qtdade erros Ping timeout. Requer valor do tipo numérico. Define a quantidade de vezes que o evento Ping-J será executado, sem que haja resposta do destino, até que seja considerado o erro de timeout.
ORIG_EMAIL	E-mail origem. Requer valor do tipo caracter na forma nome@dominio. Define o endereço de e-mail considerado como origem no envio de aviso ou alarme através de mecanismo de e-mail.
SERVER_SMTP	Servidor SMTP. Requer valor do tipo caracter. Estabelece o servidor de SMTP a ser acessado para envio das mensagens e aviso ou alarme através de mecanismo de e-mail.
PORT_SMTP	Porta SMTP. Requer valor do tipo numérico. Informa a porta TCP referente ao servidor SMTP indicado na palavra-chave SERVER_SMTP.
PASTA_PLA	Pasta da Planilha. Requer valor do tipo caracter. Indica qual o caminho completo de localização da planilha de execução da agenda e monitoramento. O último caracter válido deve ser o "\".
NOME_PLA	Nome da Planilha. Requer valor do tipo caracter. Fornece o nome do arquivo que corresponde à planilha de execução da agenda e monitoramento

Tabela 5.9 – Relação de Palavras-Chave do Cadastro de Parametros (Cont.)

PALAVRA-CHAVE	DESCRIÇÃO
PASTA_LOG	Pasta dos Arquivos de Log. Requer valor do tipo caracter. Indica qual o caminho completo de localização do arquivo de log ou registro das atividades de gerenciamento. O último caracter válido deve ser o "\".
NOME_LOG	Nome do Log.. Requer valor do tipo caracter. Fornece qual o nome do arquivo de log ou registro das atividades de gerenciamento.
NOME_LOG_ALARME	Nome do Log de Alarme. Requer valor do tipo caracter. Fornece qual o nome do arquivo de log de alarme ou aviso.
NOME_TIMER	Nome do Arquivo de Timer. Requer valor do tipo caracter. Fornece qual o nome do arquivo que indica a execução da rotina execução da agenda e monitoramento.
EXEC_AUTO	Executa o Timer Automaticamente. Requer valor do tipo caracter. Indica se a execução da agenda e o monitoramento acontecerá automáticamente ao abrir-se a planilha, quando for informado o valor "SIM".

5.5.5 Interface

A interface principal do aplicativo, bem como a interface de entrada de dados nos cadastros de Objetos, Agenda, Alarmes e Parâmetros foi elaborada baseada no recurso de formulários do Microsoft Excel.

5.5.5.1 Tela Principal

A tela principal foi implementada através de uma planilha do Microsoft Excel, denominada Cadastro.xls composto de botões, que executarão as funções de Iniciar o Timer, Parar o Timer e Gravar os Cadastros , além de direcionar para as telas dos Cadastros de Objetos, Agenda, Alarmes e Parâmetros , conforme Figura 5.3

Figura 5.3 – Tela Principal

5.5.5.2 Tela de Cadastro dos Objetos Gerenciados

A planilha Objetos, dentro do arquivo Cadastro.xls, segue a estrutura definida no item 5.2.1 deste capítulo para armazenar os dados referentes aos objetos gerenciados. Também foi utilizado o recurso de formulários do Excel para a tela de entrada de dados, conforme Figura 5.4.

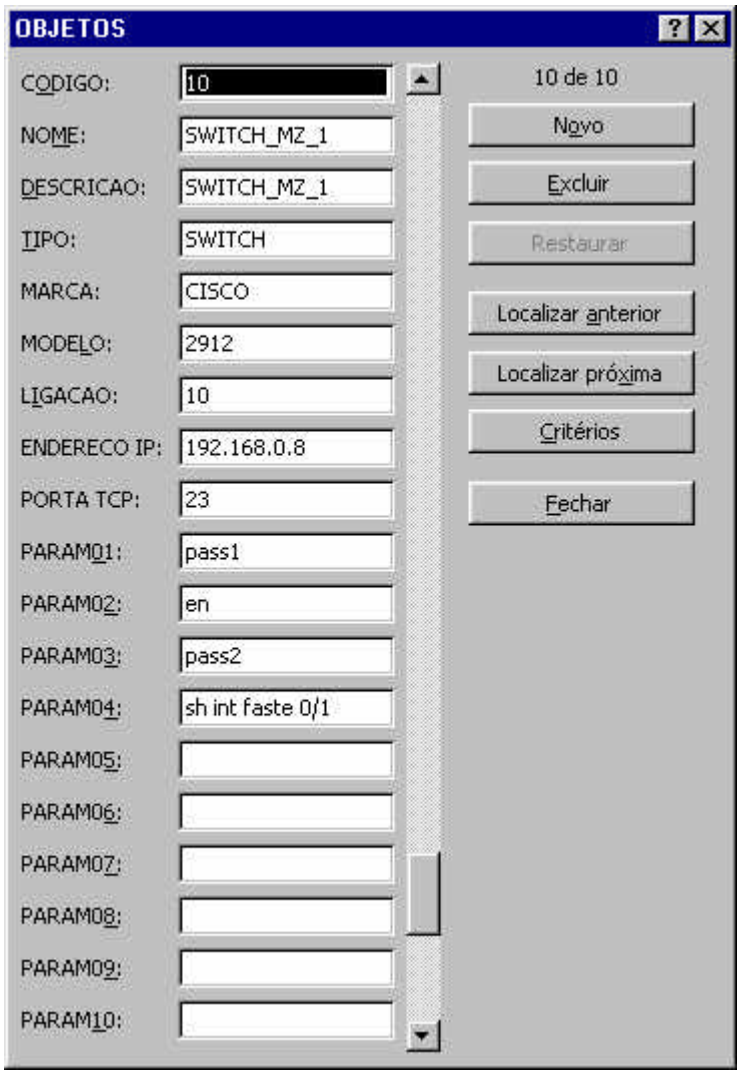

Figura 5.4 – Tela de Cadastro dos Objetos Gerenciados

5.5.5.3 Tela de Agendamento das Tarefas de Gerenciamento

A planilha Agenda, dentro do arquivo Cadastro.xls, segue a estrutura definida no item 5.2.2 deste capítulo para armazenar os dados referentes à agenda das tarefas de gerenciamento. Também foi utilizado o recurso de formulários do Excel para a tela de entrada de dados, conforme Figura 5.5.

Figura 5.5 – Tela de Agendamento das Tarefas de Gerenciamento

5.5.5.4 Tela de Configuração dos Alarmes

A planilha Alarmes, dentro do arquivo Cadastro.xls, segue a estrutura definida no item 5.2.3 deste capítulo para armazenar os dados referentes à definição dos alarmes e avisos, bem como à definição das variáveis a serem gerenciadas. Também foi utilizado o recurso de formulários do Excel para a tela de entrada de dados, conforme Figura 5.6.

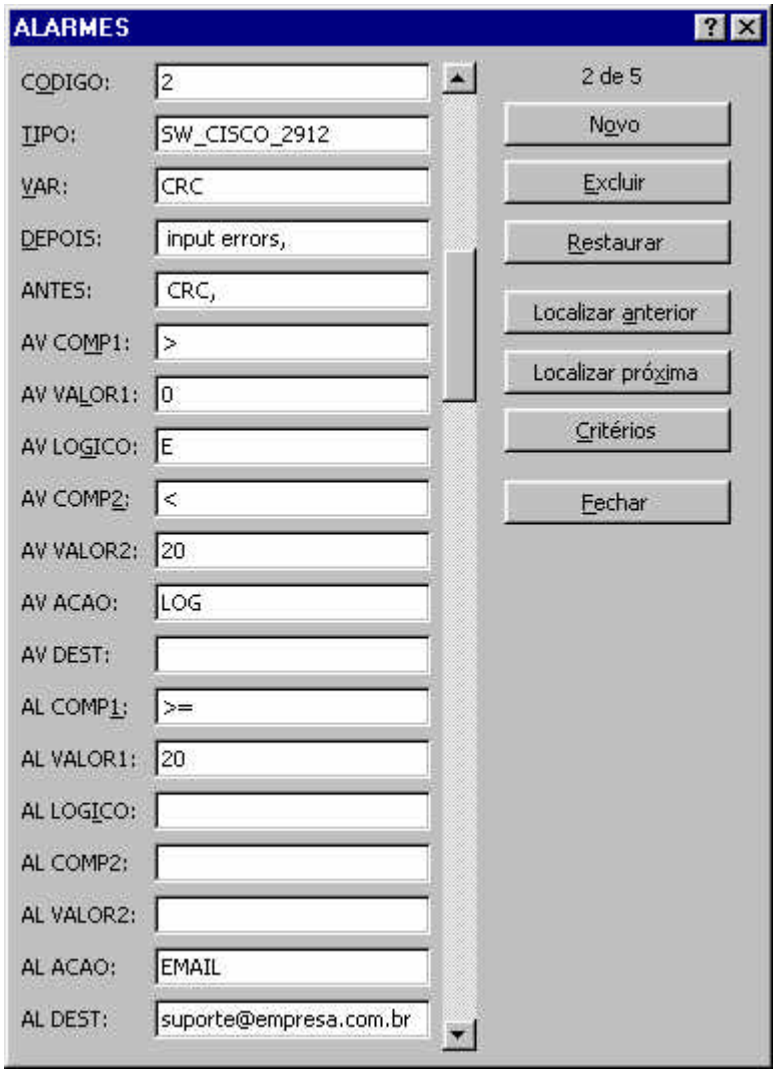

Figura 5.6 – Tela de Configuração dos Alarmes

5.5.5.5 Tela de Configuração dos Parâmetros

A planilha Parametros, dentro do arquivo Cadastro.xls, segue a estrutura definida no item 5.2.4 deste capítulo para armazenar os dados referentes aos Parâmetros Gerais. Também foi utilizado o recurso de formulários do Excel para a tela de entrada de dados, conforme Figura 5.7.

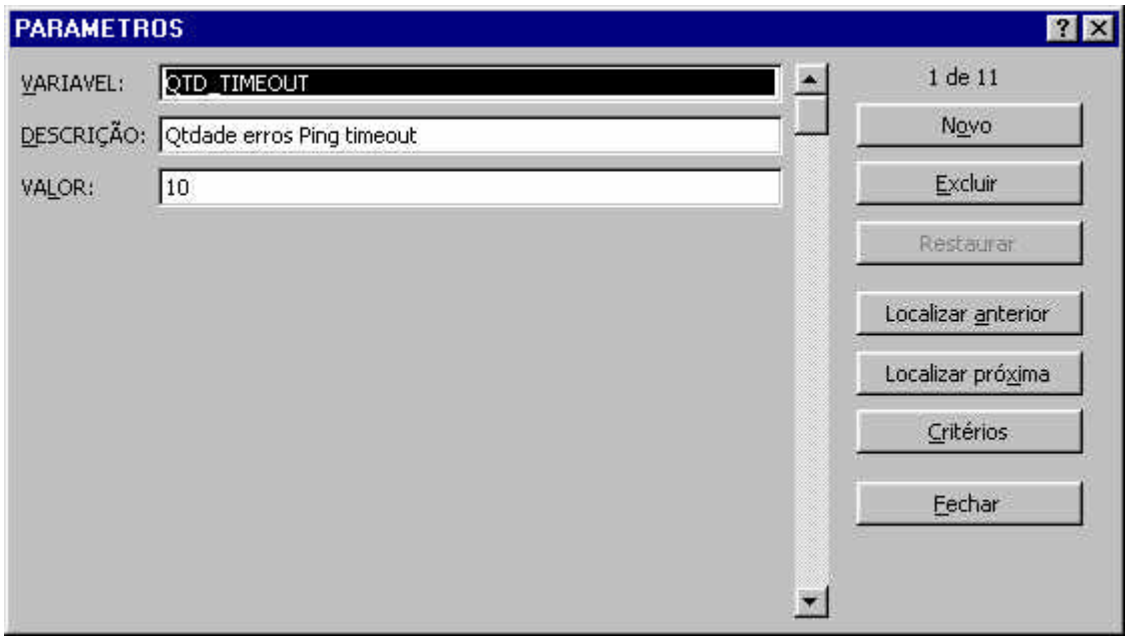

Figura 5.7 – Tela de Configuração dos Parâmetros

5.5.6 Rotinas

5.5.6.1 Inicialização do Monitoramento

A alimentação dos cadastros de Objetos, Agenda, Alarmes e Parâmetros foi feita em um arquivo do Excel chamado Cadastro.xls. Para cada cadastro será montada uma planilha, conforme especificação feita no item 5.1 deste capítulo. Após o preenchimento de todos os cadastros, deve-se clicar sobre o botão denominado “Grava Cadastro”. Este procedimento acionará uma macro que irá gravar os cadastros em arquivos no formato texto, com cada campo separado pelo caracter “;”. A finalidade da geração destes arquivos é facilitar a leitura dos cadastros durante a varredura das rotinas de execução de agenda e monitoramento.

Para inicializar as rotinas de execução da agenda e monitoramento, deve-se clicar sobre o botão intitulado “Iniciar Timer”. Este procedimento irá criar um arquivo auxiliar, conforme definições feitas através das palavras-chave PASTA_LOG e NOME_TIMER no cadastro de parâmetros, e em seguida irá carregar o arquivo do Excel descrito através das palavras-chave PASTA_PLA e NOME_PLA no cadastro de parâmetros, que inicialmente será intitulado Timer.xls. O arquivo Timer.xls, na sua inicialização, fará a verificação da existência do arquivo discriminado pelo campo NOME_TIMER. Caso o mesmo exista, dará início à varredura do monitoramento.

Na abertura do arquivo Timer.xls, será verificada a palavra-chave EXEC_AUTO do Cadastro de Parâmetros. Caso a mesma esteja contendo a palavra “SIM”, será iniciada a varredura de monitoramento da rede. Caso contrário, o arquivo Timer.xls permanecerá aberto, porém sem executar o monitoramento da rede. A finalidade desta opção é para permitir que o arquivo Timer.xls seja aberto para manutenção, acréscimo ou adequação das rotinas de gerenciamento da rede, que estão contidas nele.

5.5.6.2 Execução da Agenda

O Módulo de Execução da Agenda, foi implementado conforme definição do modelo proposto no item 5.2.5 deste capítulo. Durante a varredura de monitoramento da rede, é executada a rotina de leitura dos dados de todos os cadastros, para obter as possíveis alterações efetuadas nos mesmos durante a varredura anterior. A seguir é feita a execução da agenda. A mesma consiste em verificar cada item do Cadastro da Agenda de Gerenciamento no que diz respeito à execução do evento de gerenciamento. Caso a data e a hora atual coincida com a data e hora prevista para execução do evento, a execução da ação de gerenciamento é liberada, caso contrário, passa-se ao proximo registro da agenda.

5.5.6.3 Execução da Ação de Gerenciamento

No evento de gerenciamento, que nesta implementação pode ser do tipo PING – J ou TELNET-J, o comando especificado no campo EVENTO do cadastro de Agenda, é encaminhado ao dispositivo ou serviço a ser monitorado mencionado no campo OBJETO do cadastro de Agenda. Após o envio do comando, são recebidas as respostas do dispositivo ou serviço monitorado. Após a execução da ação de gerenciamento, será criado um arquivo de Log de Execução, caso o mesmo ainda não exista. O nome do

arquivo a ser criado segue indicação da palavra-chave NOME_LOG do Cadastro de Parâmetros. Caso o arquivo mencionado já exista, será gravado ao final do mesmo um registro de log de execução, com cada um dos campos definidos pela Tabela 5.10, separados pelo caracter ";". As informações obtidas do dispositivo ou serviço monitorado são repassadas para o módulo de monitoramento.

Tabela 5.10 – Campos do arquivo de Log de Execução

CAMPO	DESCRIÇÃO
1	Indica qual o evento de gerenciamento foi executado, sendo gravada a expressão "Ping" ou "Telnet", conforme o caso.
2	Data e hora da ocorrência do alarme, no formato dd/mm/aa hh:mm
3	Descrição obtida no campo DESCRICAO da tabela OBJETOS, do objeto de nível superior
4	Descrição obtida no campo DESCRICAO da tabela OBJETOS, do objeto de nível inferior. Apenas dois níveis são considerados nesta implementação.
5	Endereço IP do objeto, segundo a forma decimal com pontos.
6	Caso a ação de gerenciamento seja TELNET-J, será gravado o número da porta TCP, caso contrário, será gravado o tamanho da mensagem enviada pelo PING-J, acrescido do indicativo " bytes".
7	Caso a ação de gerenciamento seja PING-J, será gravada o tempo de resposta do comando, acrescido do indicativo " ms" da unidade de tempo adotada. Caso contrário, será gravado o caracter "0".
8	Caso a ação de gerenciamento seja PING-J, será gravada a informação de TTL para o pacote ICMP, acrescido do indicativo " ttl". Caso a ação seja TELNET-J, será gravado um conjunto de caracteres indicando o ponto de ocorrência do erro, se tiver havido erro na execução ou o caracter "0".
9	Será gravada a informação de Status do comando, sendo "0" a condição de sucesso e o código do erro, em caso de falha no comando.

5.5.6.4 Monitoramento dos Resultados

O resultado obtido das rotinas de gerenciamento, serão encaminhadas para o módulo de Monitoramento dos Resultados, que também está incluído no arquivo Timer.xls. Este módulo irá receber os dados obtidos pelas rotinas de gerenciamento e irá atribuir valores às variáveis de gerenciamento definidas no Cadastro de Alarmes e compará-los com os valores ou intervalos de valores, definidos no mesmo cadastro, para aviso e alarme. Caso identifique que o valor de alguma variável está contido no intervalo previsto, será acionada a rotina especificada, que pode ser LOG ou EMAIL, passando como parâmetros o nome do objeto gerenciado, o nome da variável que atingiu o limiar e o valor observado.

A rotina de LOG irá gravar os dados solicitados em um arquivo no formato texto, com cada campo separado pelo caracter ";", conforme definido na palavra-chave NOME_LOG_ALARME do Cadastro de Parâmetros. Os campos a serem gravados no arquivo de Log de Alarme estão descritos na Tabela 5.11.

A rotina de EMAIL irá enviar um e-mail, segundo o protocolo SMTP, utilizando-se de conexão TCP através da porta 25. O endereço do destinatário será aquele informado na variável AV_DEST em caso de Aviso ou AL_DEST em caso de Alarme. O servidor SMTP será o informado na palavra-chave SERVER_SMTP no Cadastro de Parâmetros. Através da palavra-chave ORIG_EMAIL do mesmo cadastro, será identificado o endereço do remetente da mensagem. O assunto do e-mail será composto do campo TIPO, DESCRIÇÃO PAI e DESCRIÇÃO FILHO, intercalados pelos caracteres " – ". Para descrição dos campos citados, ver tabela 5.11. O corpo do e-mail será composto pela data e hora da geração do alarme, o nome da variável gerenciada e o valor atribuído à ela.

5.5.6.5 Finalização do Monitoramento

Para finalizar o monitoramento da rede, deve-se clicar sobre o botão nomeado "Parar Timer", existente no arquivo cadastro.xls, que executará uma macro cuja função é apagar o arquivo auxiliar criado para iniciar o arquivo Timer.xls. No início de cada varredura de monitoramento da rede, o arquivo Timer.xls verifica a existência do arquivo definido pelo campo NOME_TIMER. Caso o mesmo exista, dá início à nova

varredura de monitoramento da rede. Caso contrário, encerra-se o monitoramento e o arquivo Timer.xls é fechado.

Tabela 5.11 – Campos do arquivo de Log de Alarme

CAMPO	DESCRIÇÃO
TIPO DO LOG	Para diferenciar se o Log é de Aviso ou Alarme, será gravada a expressão “AVISO” ou “ALARME”.
DATA	Data e hora da ocorrência do alarme, no formato dd/mm/aa hh:mm
DESCRIÇAO PAI	Descrição obtida no campo DESCRICAO da tabela OBJETOS, do objeto de nível superior
DESCRIÇÃO FILHO	Descrição obtida no campo DESCRICAO da tabela OBJETOS, do objeto de nível inferior. Apenas dois níveis são considerados nesta implementação.
ENDEREÇO IP	Endereço IP do objeto, segundo a forma decimal com pontos.
PORTA TCP	Caso o objeto possua porta TCP associada, será gravado o número da porta, caso contrário, ficará em branco.
VARIÁVEL	Nome da variável que está sendo gerenciada.
VALOR	Valor atribuido à variável

5.6 Conclusão

Com a implementação do modelo de gerenciamento proposto, foi cumprido o propósito de comprovar a possibilidade de realização prática do mesmo. Com a utilização, nesta implementaçao, da ferramenta Microsoft Excel, com as suas já mencionadas características, pôde-se também comprovar a implementação do modelo utilizando-se uma ferramenta simples, acessível e de conhecimento notório entre os administradores de pequenas redes.

Capítulo 6

Estudo de Caso

6.1 Introdução

Após o término da implementação, o modelo proposto foi implantado em uma empresa, para avaliação do mesmo na prática. A empresa em questão será denominada como EmpresaX. O período de observação foi de 15 dias, tendo sido considerado o gerenciamento de todos os equipamentos da empresa, durante as 24 horas do dia.

6.2 Estrutura da Rede

O parque computacional da EmpresaX é de 44 computadores, de diversas marcas e modelos, além de dois servidores. A EmpresaX tem sua estrutura estabelecida em duas localizações distintas, chamados de SiteA e SiteB, interligados por uma linha de dados dedicada e utilizando-se de dois Roteadores, um no SiteA e outro no SiteB. No SiteA, onde ficam os servidores, existe um Switch de 100 Mbps, destinado à ligação dos servidores e hubs, sendo que o restante da rede é atendida com Hubs de 10 Mbps. O diagrama da rede da EmpresaX é mostrado na Figura 6.1.

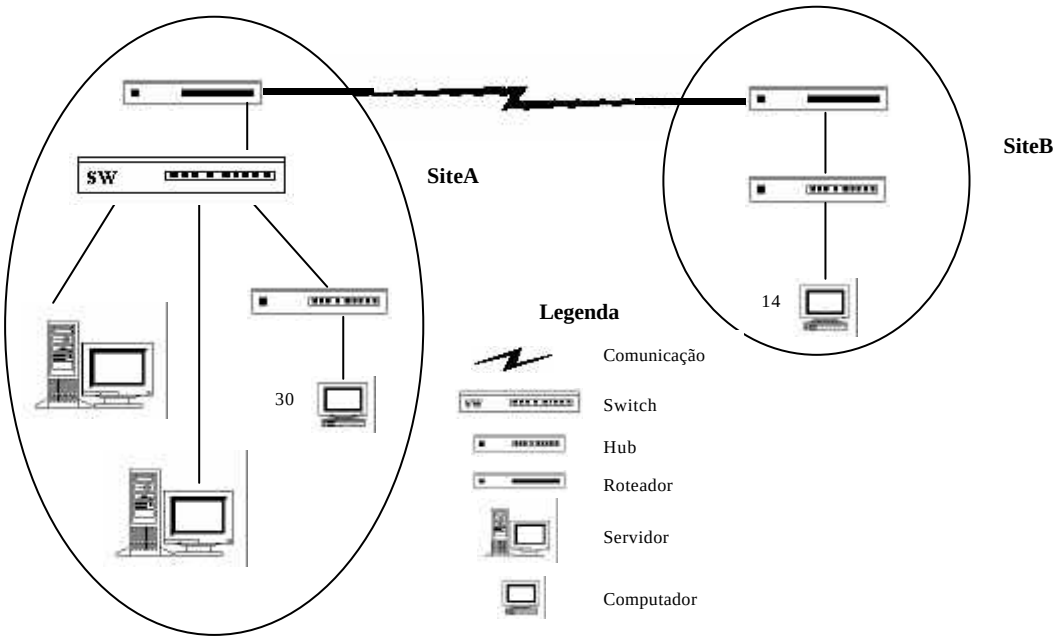

Figura 6.1 - Diagrama da rede da EmpresaX

6.3 Processo de implantação do modelo

A implantação se deu em um computador da rede, de uso não dedicado ao gerenciamento da rede, tendo sido copiados os arquivos Cadastro.xls e Timer.xls em uma pasta no seu disco rígido, denominada Adm_Rede. Apenas aos administradores da rede foi liberado o acesso à essa pasta, pois as planilhas do modelo podem conter informações de acesso exclusivo dos administradores. Desta maneira as planilhas de gerenciamento poderão ser executadas apenas por usuários com direito de administrar a rede. Inicialmente foi preenchido o Cadastro de Parâmetros, conforme estrutura mostrada na Tabela 5.9 do Capítulo 5. A Tabela 6.1 nos mostra os valores informados neste estudo de caso.

Tabela 6.1 – Dados do Cadastro de Parâmetros no estudo de caso

VARIÁVEL	DESCRIÇÃO	VALOR
QTD_TIMEOUT	Quantidade erros Ping timeout	10
ORIG_EMAIL	E-mail origem	SW_Ger@EmpresaX.com.br
SERVER_SMTP	Servidor SMTP	smtp.empresax.com.br
PORT_SMTP	Porta SMTP	25
PASTA_PLA	Pasta da Planilha	L:\Adm_Rede\Modelo_Ger
NOME_PLA	Nome da Planilha	timer.xls
PASTA_LOG	Pasta dos Arquivos de Log	L:\Adm_rede\Modelo_Ger
NOME_LOG	Nome do Log	log.txt
NOME_LOG_ALARME	Nome do Log de Alarme	logaviso.txt
NOME_TIMER	Nome do Arquivo de Timer	não_pare.txt
EXEC_AUTO	Executa o Timer Automaticamente	NAO

Foram feitos os cadastros de todos os equipamentos e o agendamento das tarefas de gerenciamento. Para identificação dos equipamentos foram utilizados os nomes já definidos na configuração de rede ou outro que pudesse identificá-los inequivocamente. Os computadores ou servidores com apenas uma placa de rede e apenas um endereço IP associado, foram cadastrados em apenas um nível de hierarquia. Equipamentos como *switches* e roteadores, que dispõe de mais de uma porta lógica de comunicação, foram cadastrados em dois níveis hierárquicos, sendo o nível superior o do próprio equipamento e o nível inferior a da porta de comunicação.

A definiçao das variáveis de gerenciamento e dos alarmes e avisos foi realizada em seguida. Para definição dos alarmes, foi necessário executar os aplicativos Ping.exe e Telnet.exe sobre todos os dispositivos de mesmo tipo, para que se pudesse determinar

os limiares de cada variável para cada um dos objetos a serem gerenciados. O Anexo B nos mostra algumas telas obtidas da utilização destes aplicativos.

Como exemplo de definição de variáveis de gerenciamento, será mostrada a definição da variável ERRO_CRC dentro dos alarmes denominados ROT_A e ROT_B, sendo ROT_A um roteador local e ROT_B um roteador remoto. Primeiramente identificou-se na resposta do objeto, o valor a ser analisado, conforme Figura 6.2.

```
0 input errors, 0 CRC 0 frame, 0 overrun, 0 ignored, 0 abort
```

Figura 6.2 Fragmento de resposta de Roteador

Neste exemplo, pode-se identificar através do fragmento “0 CRC” que o valor seria 0. A partir desta identificação, observou-se também, que o valor pesquisado encontra-se entre as cadeias de caracteres “ input errors, “ e “ CRC, “. Desta forma, pode-se utilizar a primeira cadeia de caracteres no campo DEPOIS do alarme, pois o valor desejado encontra-se após a mesma e a segunda cadeia de caracteres no campo ANTES, pois o valor desejado encontra-se anterior à mesma. A Tabela 6.1 nos apresenta um exemplo dos valores adotados para os alarmes ROT_A e ROT_B .

Tabela 6.1 – Exemplo do alarme ROT_A e ROT_B

CAMPO	**VALOR**	**VALOR**
CODIGO	7	8
TIPO	ROT_A	ROT_B
VAR	CRC	CRC
DEPOIS	input errors,	Input errors,
ANTES	CRC,	CRC,
AV_COMP1	>	>
AV_VALOR1	0	50
AV_LOGICO	AND	AND
AV_COMP2	<	<
AV_VALOR2	20	150
AV_ACAO	LOG	LOG
AV_DEST		
AL_COMP1	>=	>=
AL_VALOR1	20	150
AL_LOGICO		
AL_COMP2		
AL_VALOR2		
AL_ACAO	EMAIL	EMAIL
AL_DEST	suporte@empresa.com.br	suporte@empresa.com.br

Os avisos foram direcionados a um arquivo de Log de Alarme e os alarmes foram direcionados para o *e-mail* do administrador da rede. Um exemplo de registro de log é mostrado na Figura 6.3 e um exemplo de mensagem de alarme, através de correio eletrônico, é mostrado na Figura 6.4.

```
"Aviso",#2002-10-13 14:22:48#,"PCPM"," ","192.168.1.74","0","TEMPO_RES",110
"Aviso",#2002-10-13 14:26:42#,"ROTEADOR_B"," ","192.168.1.200","23","CRC",9
```

Figura 6.3 – Exemplo de registro de log de alarme

```
De: <sw_ger@empresa.com.br>
Para: <suporte@empresa.com.br>
Assunto: Alarme - SWITCH_A – ETH12
Data: quarta-feira, 13 de outubro de 2002 12:41
13/10/2002 12:41:43 - CRC = 220
```

Figura 6.4 – Exemplo de mensagem de alarme através de correio eletrônico

Neste estudo de caso, as variáveis monitoradas, freqüência de monitoração e valores para avisos e alarmes foram definidos conforme nos mostra a Tabela 6.2. Os valores ou intervalos de valores para aviso ou alarme foram obtidos da observação dos dados monitorados e dos eventos de falha na rede, observando-se os intervalos de monitoramento. Alguns trechos mais relevantes do log de execução, log de alarme e alguns e-mail's de alarme estão apresentados no Apêndice C.

Tabela 6.2 – Variáveis monitoradas no Estudo de Caso

TIPO DE EQUIPAMENTO	DESCRIÇÃO DA VARIÁVEL	FREQÜENCIA DE MONITORAÇÃO	VALOR DE AVISO	VALOR DE ALARME
Servidor	Tempo de Resp.(ms)	5 minutos	> 25 e < 50	>= 50
Portas do Switch	Erro de CRC	5 minutos	> 0 e < 20	>= 20
Portas do Switch	Colisões	5 minutos	> 0 e < 20	>= 20
Roteador	Erro de CRC	5 minutos	> 0 e < 20	>= 20
Roteador Local	Tempo de Resp.(ms)	5 minutos	> 25 e < 50	>= 50
Roteador Remoto	Tempo de Resp.(ms)	5 minutos	> 50 e < 150	>= 150
Estação Local	Tempo de Resp.(ms)	15 minutos	> 25 e < 50	>= 50
Estação Remota	Tempo de Resp.(ms)	15 minutos	> 50 e < 150	>= 150

6.4 Utilização do Modelo

Durante a utilização do modelo, foram observadas algumas situações que a implementação do modelo não estava preparada para tratar. Uma delas é a indisponibilidade de um equipamento. Para monitorar a disponibilidade de um determinado equipamento, foi utilizada a ação de gerenciamento PING-J. Para definir a quantidade de repetições até que o equipamento fosse considerado indisponível, foi utilizada a palavra-chave QTD_TIMEOUT do Cadastro de Parâmetros, que foi definida como 10 neste estudo de caso. Quando um equipamento está desligado ou indisponível na rede, são criados registros no arquivo de log de execução na quantidade definida pela palavra-chave QTD_TIMEOUT e no campo de erro do log é mostrado o código correspondente 11001, cujo significado é "*Host not found*" ou "Equipamento não encontrado". Um exemplo desta situação é mostrado no trecho de log da Figura 6.5, que reflete a indisponibilidade do computador de nome FATURAMENTO, endereço IP 192.168.1.45.

```
"Ping  ",#2002-10-11 18:34:17#,"FATURAMENTO"," ","192.168.1.45",
"0 bytes","2097172 ms","84 ttl","11001"
"Ping  ",#2002-10-11 18:34:27#,"FATURAMENTO"," ","192.168.1.45",
"0 bytes","2097172 ms","84 ttl","11001"
"Ping  ",#2002-10-11 18:34:37#,"FATURAMENTO"," ","192.168.1.45",
"0 bytes","2097172 ms","84 ttl","11001"
"Ping  ",#2002-10-11 18:34:47#,"FATURAMENTO"," ","192.168.1.45",
"0 bytes","2097172 ms","84 ttl","11001"
"Ping  ",#2002-10-11 18:34:57#,"FATURAMENTO"," ","192.168.1.45",
"0 bytes","2097172 ms","84 ttl","11001"
"Ping  ",#2002-10-11 18:35:07#,"FATURAMENTO"," ","192.168.1.45",
"0 bytes","2097172 ms","84 ttl","11001"
"Ping  ",#2002-10-11 18:35:17#,"FATURAMENTO"," ","192.168.1.45",
"0 bytes","2097172 ms","84 ttl","11001"
"Ping  ",#2002-10-11 18:35:27#,"FATURAMENTO"," ","192.168.1.45",
"0 bytes","2097172 ms","84 ttl","11001"
"Ping  ",#2002-10-11 18:35:37#,"FATURAMENTO"," ","192.168.1.45",
"0 bytes","2097172 ms","84 ttl","11001"
"Ping  ",#2002-10-11 18:35:47#,"FATURAMENTO"," ","192.168.1.45",
"0 bytes","2097172 ms","84 ttl","11001"
```

Figura 6.5 – Trecho de log de execução de equipamento indisponível na rede

Também não foi identificado qual o impacto ocasionado pela utilização do gerenciamento da rede pelo modelo proposto. Em função dos eventos PING-J e TELNET-J executados periódicamente, pacotes de comunicação estão trafegando pela rede e aumentando o tráfego na rede. Para que fosse possível esta comparação, teria que ser feita uma medição do tráfego na rede, em diversos momentos do dia e em todos os

dias da semana, sem a utilização da ferramenta proposta e depois disso repetir o procedimento após a execução da ferramenta. A partir destes dados coletados e após o tratamento estatístico dos mesmos, poder-se-ia chegar à uma conclusão a respeito do impacto da implementação do modelo na performance geral da rede.

Neste estudo de caso não foram monitorados serviços TCP como SMTP ou POP. Este monitoramento é possível, pois ao se aplicar a ação de gerenciamento TELNET-J sobre um servidor SMTP em sua porta TCP, que é a de número 25, o mesmo fornece uma resposta contendo o código 220, caso esteja em funcionamento. De maneira semelhante pode-se testar um servidor POP, atravéz da sua porta TCP de número 110. Alguns exemplos de respostas de equipamentos e serviços TCP estão relacionados no Apêndice B.

Também foi identificado no período do estudo de caso a importância de se poder tratar variáveis contendo dados do tipo caracter no monitoramento de alguns dispositivos, como nos mostra o fragmento de resposta do *Switch* da Figura 6.7. Na amostra mencionada, a porta FastEthernet0/1 é indicada estar em funcionamento a partir dos caracteres “is up”. Caso a indicação seja diferente, pode existir alguma anormalidade no funcionamento desta porta do *switch*, que deveria ser apontada através de aviso ou alarme, conforme o caso. Esta opção de tratamento de variáveis de monitoramento não está comtenplada nesta implementação do modelo.

```
FastEthernet0/1 is up, line protocol is up
```

Figura 6.7 – Fragmento de resposta de *switch* à comando de Telnet

Alterações no código fonte desta implementação podem ser feitas através da utilização dos recursos de macro do Microsoft Excel. Uma das formas de acesso à estes recursos é através do menu Ferramentas, sub-menu Macros e opção Editor do Visual Basic, sendo possível utilizar um atalho pressionando-se ao mesmo tempo as teclas Alt e F11. A estrutura dos comandos do Visual Basic encontra-se no menu Ajuda do Editor do Visual Basic, em língua portuguesa. A Figura 6.8 mostra um trecho de programa que faz a gravação dos dados digitados no Cadastro de Objetos em um arquivo denominado Objetos.txt, para posterior leitura a cada ciclo de gerenciamento.

```
Private Sub Grava_Objetos()
    Dim StrReg As String
    Worksheets("OBJETOS").Activate
    Open Parametros.PASTA_LOG & "OBJETOS.txt" For Output As #1
    With Worksheets("OBJETOS")
        For i = 3 To 65535
            If .Cells(i, 1) = "" Then Exit For
            StrReg = ""
            For j = 1 To 30
                StrReg = StrReg & .Cells(i, j) & " ; "
            Next j
            Print #1, StrReg
        Next i
        Close #1
    End With
End Sub
```

Figura 6.8 – Trecho de programa em Visual Basic

6.5 Resultados obtidos

De maneira geral, os resultados obtidos foram satisfatórios, pois o monitoramento da rede foi executado a partir de um computador modesto em termos de recursos computacionais, com processador Pentium 100 MHz e 32 MB de memória RAM, em modo não-dedicado, permitindo que o mesmo fosse utilizado normalmente como estação de trabalho. As rotinas de Execução da Agenda e Monitoramento funcionaram a contento, executando as tarefas conforme agendadas e extraindo os valores das variáveis de gerenciamento definidas e, a partir deles, executando os alarmes e avisos, conforme definidos.

6.6 Conclusão

O estudo de caso mostrou-se uma etapa importante deste trabalho, pois veio a corroborar a viabilidade de utilização do modelo proposto em uma rede de pequeno porte. Demonstrou também a simplicidade de implantação, sem utilizar-se de grandes recursos computacionais, pois não necessitou de um equipamento dedicado à sua execução e sem comprometer o desempenho geral da rede. Também foi possível identificar algumas deficiências na implementação, permitindo a sua correção em futuros trabalhos a serem desenvolvidos.

Apesar de ter sido considerada uma configuração estática para as variáveis monitoradas, a observação dos resultados pode induzir a uma alteração nos limiares das variáveis. Esta alteração não exige a parada do sistema, pois a cada varredura o mesmo verifica os cadastros, acatando as alterações efetivadas.

Capítulo 7

Considerações Finais

Este trabalho apresentou, como proposta de dissertação, um modelo de gerenciamento baseado em ferramentas de baixo custo voltado para redes de pequeno porte que poderá vir a ser utilizado em pequenas redes, onde um gerenciamento baseado em ferramentas do mercado venha a se apresentar inviável.

Inicialmente foram apresentadas as características de uma rede de pequeno porte, ressaltando as questões dos recursos computacionais e de pessoas disponíveis nas mesmas, as suas necessidades de gerenciamento, que normalmente diferem em quantidade e intensidade das informações das redes de médio e grande porte, ressaltando também as dificuldades encontradas pelo administrador de uma pequena rede em gerenciá-la a contento. Para facilitar a compreensão dos conceitos inerentes ao trabalho, foram apresentados alguns pontos de fundamento teórico, abordando temas considerados úteis para este fim. Também foram analisadas as ferramentas de gerenciamento disponíveis no mercado, dando ênfase às ferramentas de baixo custo, do tipo *freeware*, *shareware* e de domínio público, buscando nelas uma solução para a problemática apresentada anteriormente.

Ao identificar-se que os programas de gerenciamento de redes encontrados no mercado não eram capazes de suprir as necessidades levantadas, nem de resolver os problemas enfrentados em uma rede de pequeno porte, passou-se ao estudo de um modelo que pudesse cumprir ambos os papéis. Após a finalização dos estudos e da proposição do modelo, passou-se ao desenvolvimento da implementação do mesmo, pois identificou-se necessária a comprovação da eficácia do mesmo de uma maneira prática. Um estudo de caso, a partir da implantação do modelo em uma empresa real, veio a confirmar a viabilidade de utilização do modelo proposto, bem como a sua funcionalidade em um ambiente de rede de pequeno porte.

7.1 Discussão dos resultados

Analisando os objetivos traçados, pode-se observar que os mesmos foram atingidos, pois o modelo proposto:

1. Atende à questão do baixo custo, pois pode ser implementado utilizando-se de uma ferramenta bastante difundida, o Microsoft Excel;

2. Apresenta, na sua implementação, uma documentação em língua portuguesa, inclusive no que diz respeito aos arquivos de ajuda e literatura disponível sobre o Microsoft Excel;

3. Traz a flexibilidade da definição dos recursos (equipamentos e serviços) e eventos a serem gerenciados, bem como permite que o administrador defina quais as variáveis e valores devem ser considerados;

4. Permite que, com facilidade, possam ser acrescentados outros eventos de gerenciamento, outras ações de mostra dos resultados monitorados, além de permitir qualquer melhoria na implementação do modelo, uma vez que o código fonte do mesmo é aberto;

5. Dispõe de uma instalação facilitada, pois basta a cópia dos dois arquivos do Microsoft Excel e o preenchimento dos cadastros para que o mesmo possa entrar em funcionamento;

6. Promove o monitoramento de todos os elementos da rede, permitindo até que alguns serviços de rede possam ter o seu funcionamento observado;

7. Oferece a automatização de alguns dos processos de gerenciamento, tais como o agendamento das tarefas, o monitoramento dos recursos de rede e a emissão dos alarmes, quando necessários;

8. Tende a minimizar o tempo de parada da rede, pois permite que algumas falhas da mesma possam ser previamente apontadas, através dos avisos e alarmes, favorecendo a tomada de decisões no sentido de que o funcionamento da rede não venha a sofrer descontinuidade;

9. Reduz o custo total de propriedade (TCO), ao trazer as vantagens do gerenciamento da rede, utilizando-se dos recursos computacionais já existentes;

7.2 Trabalhos Futuros

A implementação de uma camada de apresentação visual dos dispositivos e serviços gerenciados, através de desenhos e cores, indicando a sua condição de funcionamento pode ser de grande ajuda na visualização do funcionamento da rede. Também poderiam ser desenvolvidas implementações do modelo utilizando-se de outras ferramentas, gratuitas, que pudessem diminuir ainda mais o custo total de propriedade da rede.

A inclusão de outras formas de envio das mensagens de aviso e alarme, como mensagens de aviso na tela ou sinais audíveis poderiam ser acrescentadas à implementação do modelo, com expressiva contribuição à eficácia do mesmo

Melhorias no processo de segurança, através de criptografia, seja no armazenamento das senhas dos dispositivos gerenciados ou na proteção dos datagramas que irão circular pela rede, poderiam ser implementadas. A validação dos dados alimentados, no processo de digitação ou no processo de gravar os dados poderia diminuir a possibilidade de incidência de erros na execução do modelo.

O desenvolvimento de outras rotinas para a resolução dos problemas identificados na versão atual, bem como o levantamento das parametrizações para os diversos modelos de equipamentos e serviços gerenciados e a divulgação dos mesmos podem vir a ser de grande valia para aqueles que vierem a utilizar-se do modelo em suas instalações de rede.

Referências Bibliográficas

[COM00] Comer, D., **Internetworking with TCP/IP Volume I: Principles, Protocols, and Architecture**, Prentice Hall, Englewood Cliffs, New Jersey, 2000

[COM97] Comer, D. and Stevens, D., **Internetworking with TCP/IP Volume III: Client-Server Programming and Applications-Windows Socket Version**, Prentice Hall, Englewood Cliffs, New Jersey, 1997

[COM98] Comer, D. and Stevens, D., **Internetworking with TCP/IP Volume II: Design, Implementation, and Internals**, Prentice Hall, Englewood Cliffs, New Jersey, 1998

[COM99] Comer, D., **Computer network and internets**, Prentice Hall, Englewood Cliffs, New Jersey, 1999

[HELD00] HELD, G. **Managing TCP/IP networks: techniques, tools and security considerations.** John Wiley & Sons, England, 2000

[IBM00] Murhammer, Martin W. and others, **TCP/IP Tutorial and Technical Overview**, Prentice Hall, Englewood Cliffs, New Jersey, 2000

[MCO02] Microsoft Corporation, **Microsoft Corporation Home Page**, http://www.microsoft.com

[NDR95] DYSON, P. **Novell dicionário de redes**. Campus, Rio de Janeiro, 1995

[NTRK97] CORPORATION, M. **Microsoft Windows NT Server Networking Guide.** Makron Books, São Paulo, 1997

[RFC0791] Information Sciences Institute, **RFC 0791, Internet Protocol - Protocol Specification**, Internet Engineering Task Force, 1981

[RFC0792] Postel, J., **RFC 0792, Internet Control Message Protocol**, Internet Engineering Task Force, 1981

[RFC0793] Information Sciences Institute, **RFC 0793, Transmission Control Protocol - Protocol Specification**, Internet Engineering Task Force, 1981

[RFC0821] Postel, J., **RFC 0821, Simple Mail Transfer Protocol**, Internet Engineering Task Force, 1982

[RFC0822] Crocker, D., **RFC 0822, Standard for the Format of ARPA Internet Text Messages**, Internet Engineering Task Force, 1982

[RFC0919] Mogul, Jeffrey, **RFC 0919, Broadcasting Internet Datagrams**, Internet Engineering Task Force, 1984

[RFC0922] Mogul, Jeffrey, **RFC 0922, Broadcasting Internet Datagrams in the presence of SubNets**, Internet Engineering Task Force, 1984

[RFC0950] Mogul, Jeffrey and Postel, J., **RFC 0950, Internet Standard Subnetting Procedure**, Internet Engineering Task Force, 1985

[RFC1060] Reynolds, J. and Postel, J., **RFC 1060, Assigned Numbers**, Internet Engineering Task Force, 1990

[RFC1122] Braden, R., **RFC 1122, Requirements for Internet Hosts--Communication Layers**, Internet Engineering Task Force, 1989

[RFC1123] Braden, R., **RFC 1123, Requirements for Internet Hosts—Application and Support**, Internet Engineering Task Force, 1989

[RFC1166] S. Kirkpatrick, **RFC 1166, Internet Numbers**, Internet Engineering Task Force, 1990

[RFC1323] Jacobson, V., Braden, R. and Borman, D., **RFC 1323, TCP Extensions for High Performance**, Internet Engineering Task Force, 1992

[RFC1340] Reynolds, J. and Postel, J., **RFC 1340, Assigned Numbers**, Internet Engineering Task Force, 1992

[RFC1349] Almquist, P., **RFC 1349, Type of Service in the Internet Protocol Suite**, Internet Engineering Task Force, 1992

[RFC2821] Klensin, J., **RFC 2821, Simple Mail Transfer Protocol**, Internet Engineering Task Force, 2001

[RFC2822] Resnick, P., **RFC 2822, Internet Message Format**, Internet Engineering Task Force, 2001

[SPEC01] SPECIALSKI, E. **Gerência de Redes de Computadores e de Telecomunicações.** UFSC, 2001

[TAN96] TANENBAUM, A. S. **Computer Networks.** Prentice Hall, New Jersey, 3. ed., 1996

[VBIP02] Visual Basic Internet Programming – Winsock API, **VBIP Home Page**, http://www.vbip.com/winsock-api,Novembro,2002.

Apêndice A

Implementação do Modelo Proposto em Linguagem Visual Basic

Código Fonte da Pasta de Trabalho – Arquivo Cadastro.xls

```
Dim strAgenda(1 To 200, 1 To 200) As String

Sub Le_Parametros()

    Worksheets("Parametros").Activate

    For Each c In Worksheets("Parametros").Range("A2:A65535")

        Select Case c.Value

            Case ""
                Exit For
            Case "QTD_TIMEOUT"
                Parametros.QTD_TIMEOUT = Cells(c.Row, 3)
            Case "ORIG_EMAIL"
                Parametros.ORIG_EMAIL = Cells(c.Row, 3)
            Case "SERVER_SMTP"
                Parametros.SERVER_SMTP = Cells(c.Row, 3)
            Case "PORT_SMTP"
                Parametros.PORT_SMTP = Cells(c.Row, 3)
            Case "PASTA_PLA"
                Parametros.PASTA_PLA = Cells(c.Row, 3)
            Case "NOME_PLA"
                Parametros.NOME_PLA = Cells(c.Row, 3)
            Case "PASTA_LOG"
                Parametros.PASTA_LOG = Cells(c.Row, 3)
            Case "NOME_LOG"
                Parametros.NOME_LOG = Cells(c.Row, 3)
            Case "NOME_LOG_ALARME"
                Parametros.NOME_LOG_ALARME = Cells(c.Row, 3)
            Case "NOME_TIMER"
                Parametros.NOME_TIMER = Cells(c.Row, 3)
            Case "EXEC_AUTO"
                Parametros.EXEC_AUTO = Cells(c.Row, 3)

        End Select

    Next c

    Worksheets("Menu").Activate

End Sub

Private Sub Workbook_BeforeClose(Cancel As Boolean)

    For Each pt In Application.Workbooks
        pt.Save
    Next pt
```

```
    Application.Quit

End Sub

Private Sub Workbook_Open()

    Esconde_tudo

    Le_Parametros

    Worksheets("menu").Activate

End Sub
```

Código Fonte do Módulo 1 – Arquivo Cadastro.xls

```
Sub Esconde_tudo()

    Application.DisplayFullScreen = True

    With ActiveWindow
        .DisplayGridlines = False
        .DisplayHeadings = False
        .DisplayOutline = False
        .DisplayZeros = False
        .DisplayHorizontalScrollBar = False
        .DisplayVerticalScrollBar = False
        .DisplayWorkbookTabs = False
    End With

    With Application

        .WindowState = xlNormal
        .Left = 0
        .Top = 62
        .Width = 650
        .Height = 325

        .DisplayFormulaBar = False
        .DisplayStatusBar = False

        For Each barra In Application.CommandBars
            If barra.Visible Then
                strBarra = barra.Name
                If strBarra <> "Worksheet Menu Bar" Then
                    Application.CommandBars(strBarra).Visible = False
                End If
            End If
        Next

    End With

End Sub
```

Código Fonte da Pasta de Trabalho – Arquivo Timer.xls

```
Private Sub Workbook_Open()

    Le_Parametros

    If UCase(Parametros.EXEC_AUTO) = "SIM" Then
        Tempo
    End If

End Sub


Public Sub Tempo()

    Do While Not Checa_Fim
        Laco1
    Loop

    For Each pt In Application.Workbooks
        pt.Save
    Next pt

    Application.Quit

End Sub

Sub Laco1()

    Dim i As Integer, j As Integer, K As Integer
    Dim strEndIP As String, strDescPai As String
    Dim strDescFilho As String
    Dim strSoma_Minuto As String
    Dim intLoopPing As Integer
    Dim strComando As TNET_Comando

    Le_Parametros
    Le_Objetos
    Le_Agenda
    Le_Alarmes
    Copia_Agenda

    For i = 1 To 200 ' Laço que percorre a Agenda I=Item da Agenda

        strTimeIni = Time

        If strAgenda(i, 2) = "" Or Checa_Fim Then
            Exit For
        End If

        Dim blnExecuta As Boolean, blnPrimExec As Boolean, _
            datExec As Date

        blnExecuta = False

        If strAgenda(i, 9) = "" Then
            strAgenda(i, 9) = Date
            strAgenda(i, 10) = Time
            blnPrimExec = True
        End If
```

```
strAlarme = strAgenda(i, 8)

datExec = CDate(strAgenda(i, 9) & " " & _
    Format(strAgenda(i, 10), "hh:mm"))

Select Case strAgenda(i, 3)

    Case "ÚNICA", "UNICA", "única", "unica"
        If blnPrimExec Then
            datFinal = datExec
        Else
            datSoma = CDate(Format("00:30", "hh:mm"))
            datFinal = Now + datSoma
        End If

    Case "MIN", "min"
        If blnPrimExec Then
            datFinal = datExec
        Else
            datSoma = CDate(Format("00:" & strAgenda(i, 4), _
                       "hh:mm"))
            datFinal = datExec + datSoma
        End If

    Case "HOR", "hor"
        If blnPrimExec Then
            If Minute(datExec) >= Val(strAgenda(i, 5)) Then
                datFinal = datExec
            End If
        Else
            datSoma = CDate(Format(strAgenda(i, 4) & _
                      ":00", _ "hh:mm"))
            datFinal = datExec + datSoma
            datFinal = CDate(Left(CStr(datFinal), 12) & _
                        strAgenda(i, 5))
        End If

    Case "DIA", "dia"
        If blnPrimExec Then
            If TimeValue(datExec) >= _
               TimeValue(Format(strAgenda(i, 5), "hh:mm"))
            Then
                datFinal = datExec
            End If
        Else
            datSoma = Val(strAgenda(i, 4))
            datFinal = datExec + datSoma
            datFinal = CDate(Left(CStr(datFinal), 9) & _
                       Format(strAgenda(i, 5), "hh:mm"))
        End If

    Case "SEM", "sem"
        If blnPrimExec Then
            If WeekDay(datExec) = Val(strAgenda(i, 5)) And _
              TimeValue(datExec) >= _
               TimeValue(Format(strAgenda(i, 6), "hh:mm"))
            Then
                datFinal = datExec
            End If
        Else
```

```
            datSoma = Val(strAgenda(i, 4)) * 7
            datFinal = datExec + datSoma
            datFinal = CDate(Left(CStr(datFinal), 9) & _
                       Format(strAgenda(i, 6), "hh:mm"))
        End If


    Case "MÊS", "MES", "mês", "mes"
        If blnPrimExec Then
            If Day(datExec) = Val(strAgenda(i, 5)) And _
               TimeValue(datExec) >= _
                     TimeValue(Format(strAgenda(i, 6), _
                     "hh:mm"))
               Then
                datFinal = datExec
            End If
        Else
            datSoma = Val(strAgenda(i, 4)) * 30
            datFinal = datExec + datSoma
            strDia = Format(Left(strAgenda(i, 5), 2), "00")
            strMesAno = Mid(CStr(datFinal), 3, 6)
            strHora = Format(strAgenda(i, 6), "hh:mm")
            datFinal = CDate(Format(Left(strAgenda(i, 5), 2),_
                   "00") & Mid(CStr(datFinal), 3, 6) & " " & _
                    Format(strAgenda(i, 6), "hh:mm"))
        End If
    Case "ANO", "ano"
        If blnPrimExec Then
            If Format(datExec, "dd/mm/yy") = _
                Format(strAgenda(i, 5), "dd/mm/yy") And _
               TimeValue(datExec) >= _
                TimeValue(Format(strAgenda(i, 6), "hh:mm"))
             Then
                datFinal = datExec
            End If
        Else
            datSoma = Val(strAgenda(i, 4)) * 365
            datFinal = datExec + datSoma
            strDia = Format(Left(strAgenda(i, 5), 2), "00")
            strMesAno = Mid(CStr(datFinal), 3, 6)
            strHora = Format(strAgenda(i, 6), "hh:mm")
            datFinal = CDate(Format(Left(strAgenda(i, 5), 2),_
                   "00") & Mid(CStr(datFinal), 3, 6) & " " & _
                    Format(strAgenda(i, 6), "hh:mm"))
        End If
End Select

If Now >= datFinal Then

    strHoje = CDate(Format$(Now, "Short Date"))
    strAgora = Format$(Now, "Short Time")
    Workbooks(Parametros.NOME_PLA). _
    Worksheets("AGENDA").Cells(i + 2, 9) = strHoje
    Workbooks(Parametros.NOME_PLA). _
    Worksheets("AGENDA").Cells(i + 2, 10) = strAgora

    Select Case strAgenda(i, 2)

    Case "PING"

        For K = 1 To 200
```

```
            If strObjetos(K, 1) = "" Then Exit For

            If strAgenda(i, 7) = strObjetos(K, 1) Then
               strCodigo = strObjetos(K, 1)
               strDesc = strObjetos(K, 3)
               strEndIP = strObjetos(K, 8)
               strLigacao = strObjetos(K, 7)
               Exit For
            End If

        Next K

        If strLigacao = strCodigo Then

            strDescPai = strDesc
            strDescFilho = " "

        Else

            For L = 1 To 200

                If strObjetos(L, 1) = "" Then Exit For

                    If strObjetos(L, 7) = strLigacao Then
                        strDescPai = strObjetos(L, 3)
                        strDescFilho = strDesc
                        Exit For
                    End If

            Next L

        End If

        intLoopPing = Parametros.QTD_TIMEOUT
        If intLoopPing = 0 Then
            intLoopPing = 5
        End If

        strStatus = Ping_Obj(strEndIP, strDescPai, _
                     strDescFilho, intLoopPing)

    Case "TELNET"

        For K = 1 To 200

            If strObjetos(K, 1) = "" Then Exit For

            If strAgenda(i, 7) = strObjetos(K, 1) Then
                strCodigo = strObjetos(K, 1)
                strDesc = strObjetos(K, 3)
                strLigacao = strObjetos(K, 7)
                strEndIP = strObjetos(K, 8)
                strPortaIp = strObjetos(K, 9)
                For M = 1 To 10
                    strComando.strCom(M) = strObjetos(K, _
                     M + 9)
                Next M

                Exit For
```

```
                    End If

                Next K

                If strLigacao = strCodigo Then

                    strDescPai = strDesc
                    strDescFilho = " "

                Else

                    For L = 1 To 200

                        If strObjetos(L, 1) = "" Then Exit For

                        If strObjetos(L, 7) = strLigacao Then
                            strDescPai = strObjetos(L, 3)
                            strDescFilho = strDesc
                            Exit For
                        End If

                    Next L

                End If


                strStatus = Telnet_Obj(strEndIP, strPortaIp, _
                             strComando, strDescPai, strDescFilho)
            End Select

        End If


        strTimeFim = Time


    Next i

End Sub

Sub Abre_Cadastro()
      Workbooks.Open FileName:=Parametros.PASTA_PLA & "Cadastro.XLS",_
                     ReadOnly:=True
End Sub

Sub Fecha_Cadastro()
      Workbooks("cadastro.xls").Close SaveChanges:=False
End Sub


Function Checa_Fim() As Boolean

    strExiste = Dir(Parametros.PASTA_LOG & Parametros.NOME_TIMER)
    If strExiste <> "" Then
        Checa_Fim = False
    Else
        Checa_Fim = True
    End If

End Function
```

Código fonte do Módulo 1 – Arquivo Timer.xls

```
Public lngBytesReceived As Long

Public lngCodErro As Long
Public strDescErro As String

Public strEmailOrigem As String
Public strServSMTP As String
Public strPortaSMTP As String


Public strAgenda(1 To 200, 1 To 20) As String
Public strEventos(1 To 200, 1 To 20) As String
Public strObjetos(1 To 200, 1 To 30) As String
Public strAlarmes(1 To 200, 1 To 30) As String
Public strParametros(1 To 50) As String

Public strAlarme As String

Public Const SOL_SOCKET = 65535
Public Const SO_PROTOCOL_INFO = &H2004

Public Type TNET_Comando
    'Comandos, usuários ou senhas a serem enviados para o Telnet
    strCom(1 To 10)     As String
End Type

Public Parametros As TNET_PARAM

Public Type TNET_PARAM

    QTD_TIMEOUT As String
    ORIG_EMAIL As String
    SERVER_SMTP As String
    PORT_SMTP As String
    PASTA_PLA As String
    NOME_PLA As String
    PASTA_LOG As String
    NOME_LOG As String
    NOME_LOG_ALARME As String
    NOME_TIMER As String
    EXEC_AUTO As String

End Type

Public Type SOCKET_OPTIONS
    mvarConnectionless As Variant
    mvarGuaranteedDelivery As Variant
    mvarGuaranteedOrder As Variant
    mvarMessageOriented As Variant
    mvarPseudoStream As Variant
    mvarGracefulClose As Variant
    mvarExpeditedData As Variant
    mvarConnectData As Variant
    mvarDisconnectData As Variant
    mvarSupportBroadcast As Variant
    mvarSupportMultipoint As Variant
    mvarMultipointControlPlane As Variant
    mvarMultipointDataPlane As Variant
```

```
    mvarQoSSupport As Variant
    mvarUniSend As Variant
    mvarUniRecv As Variant
    mvarIFSHandles As Variant
    mvarPartialMessage As Variant
    mvarMultipleProtoEntries As Variant
    mvarRecommendedProtoEntry As Variant
    mvarMatchesProtocolZero As Variant
    mvarProviderId As Variant
    mvarCatalogEntryId As Variant
    mvarVersion As Variant
    mvarAddressFamily As Variant
    mvarMaxSockAddr As Variant
    mvarMinSockAddr As Variant
    mvarSocketType As Variant
    mvarProtocol As Variant
    mvarProtocolMaxOffset As Variant
    mvarNetworkByteOrder As Variant
    mvarSecurityScheme As Variant
    mvarMessageSize As Variant
    mvarProtocolName As Variant
End Type

Private Const WSAPROTOCOL_LEN = 256

Private Type Guid
    Data1 As Long
    Data2 As Integer
    Data3 As Integer
    Data4(0 To 7) As Byte
End Type

Private Const MAX_PROTOCOL_CHAIN = 6

Private Type WSAPROTOCOLCHAIN
  ChainLen As Long
  ChainEntries(MAX_PROTOCOL_CHAIN) As Long
End Type

Private Type WSAPROTOCOL_INFO
    dwServiceFlags1 As Long
    dwServiceFlags2 As Long
    dwServiceFlags3 As Long
    dwServiceFlags4 As Long
    dwProviderFlags As Long
    ProviderId As Guid
    dwCatalogEntryId As Long
    ProtocolChain As WSAPROTOCOLCHAIN
    iVersion As Long
    iAddressFamily As Long
    iMaxSockAddr As Long
    iMinSockAddr As Long
    iSocketType As Long
    iProtocol As Long
    iProtocolMaxOffset As Long
    iNetworkByteOrder As Long
    iSecurityScheme As Long
    dwMessageSize As Long
    dwProviderReserved As Long
    szProtocol As String * WSAPROTOCOL_LEN
End Type
```

```
Public Type ALARME_OPCOES
    strCodigo       As String
    strTipo         As String
    strVar          As String
    strDepois       As String
    strAntes        As String
    strAV_Comp1     As String
    lngAV_Valor1    As Long
    strAV_Logico    As String
    strAV_Comp2     As String
    lngAV_Valor2    As Long
    strAV_Acao      As String
    strAV_Dest      As String
    strAL_Comp1     As String
    lngAL_Valor1    As Long
    strAL_Logico    As String
    strAL_Comp2     As String
    lngAL_Valor2    As Long
    strAL_Acao      As String
    strAL_Dest      As String
End Type

Public Const INADDR_NONE = &HFFFF
Public Const SOCKET_ERROR = -1
Public Const INVALID_SOCKET = -1

Private Const IP_SUCCESS As Long = 0
Private Const IP_STATUS_BASE As Long = 11000
Private Const IP_BUF_TOO_SMALL As Long = (11000 + 1)
Private Const IP_DEST_NET_UNREACHABLE As Long = (11000 + 2)
Private Const IP_DEST_HOST_UNREACHABLE As Long = (11000 + 3)
Private Const IP_DEST_PROT_UNREACHABLE As Long = (11000 + 4)
Private Const IP_DEST_PORT_UNREACHABLE As Long = (11000 + 5)
Private Const IP_NO_RESOURCES As Long = (11000 + 6)
Private Const IP_BAD_OPTION As Long = (11000 + 7)
Private Const IP_HW_ERROR As Long = (11000 + 8)
Private Const IP_PACKET_TOO_BIG As Long = (11000 + 9)
Private Const IP_REQ_TIMED_OUT As Long = (11000 + 10)
Private Const IP_BAD_REQ As Long = (11000 + 11)
Private Const IP_BAD_ROUTE As Long = (11000 + 12)
Private Const IP_TTL_EXPIRED_TRANSIT As Long = (11000 + 13)
Private Const IP_TTL_EXPIRED_REASSEM As Long = (11000 + 14)
Private Const IP_PARAM_PROBLEM As Long = (11000 + 15)
Private Const IP_SOURCE_QUENCH As Long = (11000 + 16)
Private Const IP_OPTION_TOO_BIG As Long = (11000 + 17)
Private Const IP_BAD_DESTINATION As Long = (11000 + 18)
Private Const IP_ADDR_DELETED As Long = (11000 + 19)
Private Const IP_SPEC_MTU_CHANGE As Long = (11000 + 20)
Private Const IP_MTU_CHANGE As Long = (11000 + 21)
Private Const IP_UNLOAD As Long = (11000 + 22)
Private Const IP_ADDR_ADDED As Long = (11000 + 23)
Private Const IP_GENERAL_FAILURE As Long = (11000 + 50)
Private Const MAX_IP_STATUS As Long = (11000 + 50)
Private Const IP_PENDING As Long = (11000 + 255)
Private Const PING_TIMEOUT As Long = 500
Private Const WS_VERSION_REQD As Long = &H101
Private Const MIN_SOCKETS_REQD As Long = 1
Private Const MAX_WSADescription As Long = 256
```

```
Private Const MAX_WSASYSStatus As Long = 128

Private Type ICMP_OPTIONS
    Ttl             As Byte
    Tos             As Byte
    flags           As Byte
    OptionsSize     As Byte
    OptionsData     As Long
End Type

Public Type ICMP_ECHO_REPLY
    Address         As Long
    Status          As Long
    RoundTripTime   As Long
    DataSize        As Long 'formerly integer
   'Reserved        As Integer
    DataPointer     As Long
    Options         As ICMP_OPTIONS
    Data            As String * 250
End Type



'/*
' * All Windows Sockets error constants are biased by WSABASEERR from
' * the "normal"
' */
Public Const WSABASEERR = 10000


Public Const AF_INET = 2        '/* internetwork: UDP, TCP, etc. */
'Socket types
'

Public Const SOCK_STREAM = 1    ' /* stream socket */


'/*
' * Protocols
' */

Public Const IPPROTO_TCP = 6            '/* tcp */

Public Const SOCKET_VERSION_11 = &H101
Public Const SOCKET_VERSION_22 = &H202

'


'/*
' * Windows Sockets definitions of regular Microsoft C error constants
' */
Public Const WSAEINTR = (WSABASEERR + 4)
Public Const WSAEBADF = (WSABASEERR + 9)
Public Const WSAEACCES = (WSABASEERR + 13)
Public Const WSAEFAULT = (WSABASEERR + 14)
Public Const WSAEINVAL = (WSABASEERR + 22)
Public Const WSAEMFILE = (WSABASEERR + 24)

'/*
' * Windows Sockets definitions of regular Berkeley error constants
' */
```

```
Public Const WSAEWOULDBLOCK = (WSABASEERR + 35)
Public Const WSAEINPROGRESS = (WSABASEERR + 36)
Public Const WSAEALREADY = (WSABASEERR + 37)
Public Const WSAENOTSOCK = (WSABASEERR + 38)
Public Const WSAEDESTADDRREQ = (WSABASEERR + 39)
Public Const WSAEMSGSIZE = (WSABASEERR + 40)
Public Const WSAEPROTOTYPE = (WSABASEERR + 41)
Public Const WSAENOPROTOOPT = (WSABASEERR + 42)
Public Const WSAEPROTONOSUPPORT = (WSABASEERR + 43)
Public Const WSAESOCKTNOSUPPORT = (WSABASEERR + 44)
Public Const WSAEOPNOTSUPP = (WSABASEERR + 45)
Public Const WSAEPFNOSUPPORT = (WSABASEERR + 46)
Public Const WSAEAFNOSUPPORT = (WSABASEERR + 47)
Public Const WSAEADDRINUSE = (WSABASEERR + 48)
Public Const WSAEADDRNOTAVAIL = (WSABASEERR + 49)
Public Const WSAENETDOWN = (WSABASEERR + 50)
Public Const WSAENETUNREACH = (WSABASEERR + 51)
Public Const WSAENETRESET = (WSABASEERR + 52)
Public Const WSAECONNABORTED = (WSABASEERR + 53)
Public Const WSAECONNRESET = (WSABASEERR + 54)
Public Const WSAENOBUFS = (WSABASEERR + 55)
Public Const WSAEISCONN = (WSABASEERR + 56)
Public Const WSAENOTCONN = (WSABASEERR + 57)
Public Const WSAESHUTDOWN = (WSABASEERR + 58)
Public Const WSAETOOMANYREFS = (WSABASEERR + 59)
Public Const WSAETIMEDOUT = (WSABASEERR + 60)
Public Const WSAECONNREFUSED = (WSABASEERR + 61)
Public Const WSAELOOP = (WSABASEERR + 62)
Public Const WSAENAMETOOLONG = (WSABASEERR + 63)
Public Const WSAEHOSTDOWN = (WSABASEERR + 64)
Public Const WSAEHOSTUNREACH = (WSABASEERR + 65)
Public Const WSAENOTEMPTY = (WSABASEERR + 66)
Public Const WSAEPROCLIM = (WSABASEERR + 67)
Public Const WSAEUSERS = (WSABASEERR + 68)
Public Const WSAEDQUOT = (WSABASEERR + 69)
Public Const WSAESTALE = (WSABASEERR + 70)
Public Const WSAEREMOTE = (WSABASEERR + 71)

'/*
' * Extended Windows Sockets error constant definitions
' */
Public Const WSASYSNOTREADY = (WSABASEERR + 91)
Public Const WSAVERNOTSUPPORTED = (WSABASEERR + 92)
Public Const WSANOTINITIALISED = (WSABASEERR + 93)
Public Const WSAEDISCON = (WSABASEERR + 101)
Public Const WSAENOMORE = (WSABASEERR + 102)
Public Const WSAECANCELLED = (WSABASEERR + 103)
Public Const WSAEINVALIDPROCTABLE = (WSABASEERR + 104)
Public Const WSAEINVALIDPROVIDER = (WSABASEERR + 105)
Public Const WSAEPROVIDERFAILEDINIT = (WSABASEERR + 106)
Public Const WSASYSCALLFAILURE = (WSABASEERR + 107)
Public Const WSASERVICE_NOT_FOUND = (WSABASEERR + 108)
Public Const WSATYPE_NOT_FOUND = (WSABASEERR + 109)
Public Const WSA_E_NO_MORE = (WSABASEERR + 110)
Public Const WSA_E_CANCELLED = (WSABASEERR + 111)
Public Const WSAEREFUSED = (WSABASEERR + 112)

Public Const WSAHOST_NOT_FOUND = 11001
Public Const WSADESCRIPTION_LEN = 257
Public Const WSASYS_STATUS_LEN = 129
Public Const WSATRY_AGAIN = 11002
```

```
Public Const WSANO_RECOVERY = 11003
Public Const WSANO_DATA = 11004

Public Const FD_SETSIZE = 64

Private Const OFFSET_2 = 65536
Private Const MAXINT_2 = 32767

Public Type WSADATA
    wversion       As Integer
    wHighVersion   As Integer
    szDescription  As String * WSADESCRIPTION_LEN
    szSystemStatus As String * WSASYS_STATUS_LEN
    iMaxSockets    As Integer
    iMaxUdpDg      As Integer
    lpVendorInfo   As Long
End Type

Public Type HOSTENT
    hName     As Long
    hAliases  As Long
    hAddrType As Integer
    hLength   As Integer
    hAddrList As Long
End Type


Public Type sockaddr_in
    sin_family       As Integer
    sin_port         As Integer
    sin_addr         As Long
    sin_zero(1 To 8) As Byte
End Type

Public Type fd_set
  fd_count                  As Long '// how many are SET?
  fd_array(1 To FD_SETSIZE) As Long '// an array of SOCKETs
End Type

Public Declare Function WSAStartup Lib "WSOCK32.DLL" (ByVal wVR As _
      Long, lpWSAD As WSADATA) As Long

Public Declare Function WSACleanup Lib "WSOCK32.DLL" () As Long

Public Declare Function gethostbyaddr Lib "WSOCK32.DLL" (addr As _
      Long,ByVal addr_len As Long, ByVal addr_type As Long) As Long

Public Declare Function gethostbyname Lib "WSOCK32.DLL" (ByVal _
      host_name As String) As Long


Public Declare Function inet_addr Lib "WSOCK32.DLL" (ByVal cp As _
      string) As Long

Public Declare Function htons Lib "WSOCK32.DLL" (ByVal hostshort As _
      Integer) As Integer

Public Declare Function socket Lib "WSOCK32.DLL" (ByVal af As Long, _
      ByVal s_type As Long, ByVal protocol As Long) As Long

Public Declare Function closesocket Lib "WSOCK32.DLL" (ByVal S As _
```

```
      Long) As Long

Public Declare Function connect Lib "WSOCK32.DLL" (ByVal S As Long, _
      ByRef name As sockaddr_in, ByVal namelen As Long) As Long

Public Declare Function vbselect Lib "WSOCK32.DLL" Alias "select" _
      (ByVal nfds As Long, ByRef readfds As Any, ByRef writefds As _
      Any, ByRef exceptfds As Any, ByRef TimeOut As Long) As Long

Public Declare Function recv Lib "WSOCK32.DLL" (ByVal S As Long, _
      ByRef buf As Any, ByVal buflen As Long, ByVal flags As Long) _
      As Long

Public Declare Function send Lib "WSOCK32.DLL" (ByVal S As Long, _
      ByRef buf As Any, ByVal buflen As Long, ByVal flags As Long) _
      As Long

Public Declare Sub RtlMoveMemory Lib "kernel32" (hpvDest As Any, _
      ByVal hpvSource As Long, ByVal cbCopy As Long)

Public Declare Function timeGetTime Lib "winmm.dll" () As Long

Public Declare Function getsockopt Lib "WSOCK32.DLL" (ByVal S As _
      Long, ByVal level As Long, ByVal optname As Long, ByRef optval _
      As Any, ByRef optlen As Long) As Long

Public Declare Function StringFromGUID2 Lib "ole32.dll" (rguid As _
      Any, ByVal lpsz As String, ByVal cchMax As Long) As Long

Private Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" _
      (xDest As Any, xSource As Any, ByVal nbytes As Long)

Private Declare Function IcmpCreateFile Lib "Icmp.dll" () As Long

Private Declare Function IcmpCloseHandle Lib "Icmp.dll" (ByVal _
      IcmpHandle As Long) As Long

Private Declare Function IcmpSendEcho Lib "Icmp.dll" (ByVal _
      IcmpHandle As Long, ByVal DestinationAddress As Long, _
      ByVal RequestData As String, ByVal RequestSize As Long, _
      ByVal RequestOptions As Long, ReplyBuffer As ICMP_ECHO_REPLY, _
      ByVal ReplySize As Long, ByVal TimeOut As Long) As Long

Public Declare Function WSAIsBlocking Lib "WSOCK32.DLL" () As Long

Public Declare Function WSACancelBlockingCall Lib "WSOCK32.DLL" () _
      As Long

Public Declare Function GetEnvironmentVariable Lib "kernel32.dll" _
      Alias "GetEnvironmentVariableA" (ByVal lpName As String, _
      ByVal lpBuffer As String, ByVal nSize As Long) As Long

Public Declare Function SetEnvironmentVariable Lib "kernel32.dll" _
      Alias "SetEnvironmentVariableA" (ByVal lpName As String, _
      ByVal lpValue As String) As Long

Public Declare Function GetShortPathName Lib "kernel32" Alias _
    "GetShortPathNameA" (ByVal lpszLongPath As String, ByVal _
    lpszShortPath As String, ByVal lBuffer As Long) As Long

Private strVarTemp As String
```

```
Private Sub LeVarTemp()

    Dim envvar As String
    Dim slength As Long
    envvar = Space(256)

    slength = GetEnvironmentVariable("TEMP", envvar, Len(envvar))

    If slength = 0 Then

        strVarTemp = "C:\Windows\Temp\"
    Else
        strVarTemp = Left(envvar, slength) & "\"
    End If

End Sub



Public Function GetShortPath(ByVal strFileName As String) As String

    Dim lngRes As Long
    Dim strPath As String

    strPath = String$(165, 0)
    lngRes = GetShortPathName(strFileName, strPath, 164)

    If lngRes <> 0 Then
        GetShortPath = Left$(strPath, lngRes)
    Else
        GetShortPath = strFileName
    End If

End Function

Public Function Replace(ByVal strTextoOri As String, _
      ByVal strTextoProc As String, ByVal strTextoTroca As String) _
      As String

    Dim intPos As Integer, intTamOri As Integer, intTamProc As Integer

    intPos = 1
    intTamProc = Len(strTextoProc)

    Do While intPos <> 0

        intTamOri = Len(strTextoOri)

        intPos = InStr(intPos, strTextoOri, strTextoProc)

        If intPos <> 0 Then

            strTextoOri = Left(strTextoOri, intPos - 1) & _
             strTextoTroca & Right(strTextoOri, intTamOri - intPos - _
             intTamProc + 1)

        End If

    Loop
```

```
    Replace = strTextoOri

End Function


Public Function Pausa(intSeg As Integer)

    Dim intSegIni As Long, intSegFim As Long

    intSegIni = Timer
    intSegFim = intSegIni + intSeg

    Do While intSegIni < intSegFim
        intSegIni = Timer
    Loop

End Function

Public Function Testa_Condicao(ByVal lngValor1 As Long, _
      ByVal strCond As String, ByVal lngValor2 As Long) As Boolean

    Select Case strCond

        Case ">"
            Testa_Condicao = (lngValor1 > lngValor2)
        Case "<"
            Testa_Condicao = (lngValor1 < lngValor2)
        Case ">="
            Testa_Condicao = (lngValor1 >= lngValor2)
        Case "<="
            Testa_Condicao = (lngValor1 <= lngValor2)
        Case "<>"
            Testa_Condicao = (lngValor1 <> lngValor2)
        Case "="
            Testa_Condicao = (lngValor1 = lngValor2)
        Case Else
            Testa_Condicao = False

    End Select

End Function

Public Function Testa_Logico(ByVal blnValor1 As Boolean, _
      ByVal strCond As String, ByVal blnValor2 As Boolean) As Boolean


    Select Case strCond

        Case "AND"
            Testa_Logico = blnValor1 And blnValor2
        Case "OR"
            Testa_Logico = blnValor1 Or blnValor2
        Case "XOR"
            Testa_Logico = blnValor1 Xor blnValor2
        Case Else
            Testa_Logico = blnValor1 Or blnValor2

    End Select


End Function
```

```
Function Cria_Winsock() As Long

    Dim lngSocket As Long
    Dim lngAddressFamily As Long
    Dim lngSocketType As Long
    Dim lngProtocol As Long
    '
    lngAddressFamily = CLng(AF_INET)
    lngSocketType = CLng(SOCK_STREAM)
    lngProtocol = CLng(IPPROTO_TCP)
    lngSocket = vbSocket(lngAddressFamily, lngSocketType, lngProtocol)
    '
    If lngSocket = INVALID_SOCKET Then
        '
        'If the function has returned the INVALID_SOCKET
        'value the socket was not created.

        lngCodErro = Err.LastDllError
        strDescErro = GetErrorDescription(lngCodErro)
        Cria_Winsock = CLng(Err.LastDllError)
        '
    Else
        '
        Cria_Winsock = lngSocket

    End If

End Function

Function Conecta_Winsock(ByVal lngSocket As Long, _
      ByVal strEndIP As String, Optional ByVal strPorta As String) _
      As Long


        If strPorta = "" Then strPorta = "23"

        strRemoteHost = strEndIP
        intRemotePort = CInt(strPorta)
        lngRetValue = vbConnect(lngSocket, strRemoteHost, _
            intRemotePort)
        '
        If lngRetValue = SOCKET_ERROR Then
            '
            lngCodErro = Err.LastDllError
            strDescErro = GetErrorDescription(lngCodErro)
            Conecta_Winsock = Err.LastDllError
            '
        Else
            '
            Conecta_Winsock = lngRetValue
            '
        End If
        '
End Function
```

```
Function Recebe_Dados(ByVal lngSocket As Long) As String
    Dim strData As String
    Dim lngContBytes As Long

    lngBytesReceived = vbRecv(lngSocket, strData)
    lngContBytes = lngBytesReceived

    Start = timeGetTime
    While lngBytesReceived > 0

        Recebe_Dados = Recebe_Dados & strData


        Tmr = timeGetTime - Start

        DoEvents
        lngBytesReceived = vbRecv(lngSocket, strData)
        If lngBytesReceived > 0 Then
            lngContBytes = lngContBytes + lngBytesReceived
        End If

    Wend

    If lngBytesReceived = SOCKET_ERROR And CStr(Err.LastDllError) = _
      "0" Then
        lngBytesReceived = lngContBytes
    End If


    If lngBytesReceived > 0 Then
    ElseIf lngBytesReceived = SOCKET_ERROR Then
            lngCodErro = Err.LastDllError
            strDescErro = GetErrorDescription(lngCodErro)
            Recebe_Dados = CStr(Err.LastDllError)
            '
    End If

End Function

Function Envia_Dados(ByVal lngSocket As Long, ByVal strDataToSend _
      As String, Optional ByVal strCrLf As String) As String

    Dim arrDataToSend() As Byte
    Dim bolCrLf As Boolean

    If strCrLf = "" Or UCase(strCrLf) = "S" Then
        bolCrLf = True
    Else
        bolCrLf = False
    End If
    '
    If bolCrLf Then
        strDataToSend = strDataToSend & vbCrLf
    End If

    If vbSend(lngSocket, strDataToSend) = SOCKET_ERROR Then

        lngCodErro = Err.LastDllError
        strDescErro = GetErrorDescription(lngCodErro)
        Envia_Dados = Err.LastDllError
    Else
```

```
        Envia_Dados = ""
    End If

End Function

Sub SendMimetxt(lngSocket As Long, txtFrom, txtTo, txtSubjekt,
txtMail)

    Dim strToSend As String
    Dim strDataToSend As String

    'Send the E-Mail without Attachment
    strDataToSend = "From: " & txtFrom & vbCrLf
    strDataToSend = strDataToSend & "To: " & txtTo & vbCrLf
    strDataToSend = strDataToSend & "Date: " & Format$(Now, "DDDD , _
      dd Mmm YYYY hh:mm:ss AM/PM") & vbCrLf
    strDataToSend = strDataToSend & "Subject: " & txtSubjekt & vbCrLf
    strDataToSend = strDataToSend & "X-Mailer: " & "Sistema de _
      Gerenciamento - Versão Excel 97" & vbCrLf & vbCrLf
    strDataToSend = strDataToSend & vbCrLf & vbCrLf & Trim$(txtMail)

    strDataToSend = Replace$(strDataToSend, vbCrLf & "." & vbCrLf, _
      vbCrLf & "." & Chr$(0) & vbCrLf)

    For i = 1 To Len(strDataToSend) Step 8192
        strToSend = Trim$(Mid$(strDataToSend, i, 8192))

        strEnvia = Envia_Dados(lngSocket, strToSend)
        DoEvents
    Next i

    'Send Data and finish it!
    strEnvia = Envia_Dados(lngSocket, vbCrLf & "." & vbCrLf, "N")

End Sub

Public Function GetBySocketHandle(lngSocket As Long) As SOCKET_OPTIONS
    '
    Dim lngRetValue     As Long
    Dim udtProtocolInfo As WSAPROTOCOL_INFO
    Dim lngBufferSize   As Long
    '
    lngBufferSize = LenB(udtProtocolInfo)
    '
    lngRetValue = getsockopt(lngSocket, SOL_SOCKET, _
      SO_PROTOCOL_INFO, udtProtocolInfo, lngBufferSize)
    If lngRetValue = 0 Then
        '
        With udtProtocolInfo
            '
            GetBySocketHandle.mvarConnectionless = .dwServiceFlags1 _
                  And XP1_CONNECTIONLESS
            GetBySocketHandle.mvarGuaranteedDelivery = _
                  .dwServiceFlags1 And XP1_GUARANTEED_DELIVERY
            GetBySocketHandle.mvarGuaranteedOrder = .dwServiceFlags1 _
                  And XP1_GUARANTEED_ORDER
            GetBySocketHandle.mvarMessageOriented = .dwServiceFlags1 _
                  And XP1_MESSAGE_ORIENTED
            GetBySocketHandle.mvarPseudoStream = .dwServiceFlags1 _
                  And XP1_PSEUDO_STREAM
            GetBySocketHandle.mvarGracefulClose = .dwServiceFlags1 _
```

```
                And XP1_GRACEFUL_CLOSE
            GetBySocketHandle.mvarExpeditedData = .dwServiceFlags1 _
                And XP1_EXPEDITED_DATA
            GetBySocketHandle.mvarConnectData = .dwServiceFlags1 _
                And XP1_CONNECT_DATA
            GetBySocketHandle.mvarDisconnectData = .dwServiceFlags1 _
                And XP1_DISCONNECT_DATA
            GetBySocketHandle.mvarSupportBroadcast = .dwServiceFlags1_
                And XP1_SUPPORT_BROADCAST
            GetBySocketHandle.mvarSupportMultipoint = dwServiceFlags1_
                And XP1_SUPPORT_MULTIPOINT
            GetBySocketHandle.mvarMultipointControlPlane = _
                .dwServiceFlags1 And XP1_MULTIPOINT_CONTROL_PLANE
            GetBySocketHandle.mvarMultipointDataPlane = _
                .dwServiceFlags1 And XP1_MULTIPOINT_DATA_PLANE
            GetBySocketHandle.mvarQoSSupport = .dwServiceFlags1 And _
                XP1_QOS_SUPPORTED
            GetBySocketHandle.mvarUniSend = .dwServiceFlags1 And _
                XP1_UNI_SEND
            GetBySocketHandle.mvarUniRecv = .dwServiceFlags1 And _
                XP1_UNI_RECV
            GetBySocketHandle.mvarIFSHandles = .dwServiceFlags1 And _
                XP1_IFS_HANDLES
            GetBySocketHandle.mvarPartialMessage = .dwServiceFlags1 _
                And XP1_PARTIAL_MESSAGE
            GetBySocketHandle.mvarMultipleProtoEntries = _
                .dwProviderFlags And PFL_MULTIPLE_PROTO_ENTRIES
            GetBySocketHandle.mvarRecommendedProtoEntry = _
                .dwProviderFlags And PFL_RECOMMENDED_PROTO_ENTRY
            GetBySocketHandle.mvarMatchesProtocolZero = _
                .dwProviderFlags And PFL_MATCHES_PROTOCOL_ZERO
            GetBySocketHandle.mvarProviderId = _
                GuidToString(.ProviderId)
            GetBySocketHandle.mvarCatalogEntryId = .dwCatalogEntryId
            GetBySocketHandle.mvarVersion = .iVersion
            GetBySocketHandle.mvarAddressFamily = .iAddressFamily
            GetBySocketHandle.mvarMaxSockAddr = .iMaxSockAddr
            GetBySocketHandle.mvarMinSockAddr = .iMinSockAddr
            GetBySocketHandle.mvarSocketType = .iSocketType
            GetBySocketHandle.mvarProtocol = .iProtocol
            GetBySocketHandle.mvarProtocolMaxOffset = _
                .iProtocolMaxOffset
            GetBySocketHandle.mvarNetworkByteOrder = _
                .iNetworkByteOrder
            GetBySocketHandle.mvarSecurityScheme = .iSecurityScheme
            GetBySocketHandle.mvarMessageSize = .dwMessageSize
            GetBySocketHandle.mvarProtocolName = .szProtocol
            '
        End With

    End If
    '
End Function


Public Function GuidToString(udtGuid As Guid) As String
    '
    Dim strGuid As String
    Dim lngRetVal As Long
    '
    strGuid = String(80, Chr(0))
```

```
    '
    lngRetVal = StringFromGUID2(udtGuid, strGuid, 80&)
    '
    If lngRetVal > 0 Then
        GuidToString = StrConv(strGuid, vbFromUnicode)
    End If
    '
End Function


Public Function UnsignedToInteger(Value As Long) As Integer
    '
    If Value < 0 Or Value >= OFFSET_2 Then Error 6
    '
    If Value <= MAXINT_2 Then
        UnsignedToInteger = Value
    Else
        UnsignedToInteger = Value - OFFSET_2
    End If
    '
End Function

Public Function GetAddressLong(ByVal strHostName As String) As Long
    Dim lngPtrToHOSTENT As Long

    Dim udtHostent      As HOSTENT

    Dim lngPtrToIP      As Long
    Dim lngAddress As Long

    lngAddress = inet_addr(strHostName)

    If lngAddress = INADDR_NONE Then
        '
        lngPtrToHOSTENT = gethostbyname(strHostName)
        '
        If lngPtrToHOSTENT <> 0 Then

            RtlMoveMemory udtHostent, lngPtrToHOSTENT, _
                  LenB(udtHostent)

            RtlMoveMemory lngPtrToIP, udtHostent.hAddrList, 4

            RtlMoveMemory lngAddress, lngPtrToIP, udtHostent.hLength
            '
        Else
            '
            lngAddress = INADDR_NONE
            lngCodErro = Err.LastDllError
            strDescErro = GetErrorDescription(lngCodErro)
            '
        End If
        '
    End If
    '
    GetAddressLong = lngAddress
    '
End Function
```

```
Public Function GetErrorDescription(ByVal lngErrorCode As Long) _
      As String
    '
    Dim strDesc As String
    '
    Select Case lngErrorCode
        '
        Case WSAEACCES
            strDesc = "Permission denied."
        Case WSAEADDRINUSE
            strDesc = "Address already in use."
        Case WSAEADDRNOTAVAIL
            strDesc = "Cannot assign requested address."
        Case WSAEAFNOSUPPORT
            strDesc = "Address family not supported by protocol _
                  family."
        Case WSAEALREADY
            strDesc = "Operation already in progress."
        Case WSAECONNABORTED
            strDesc = "Software caused connection abort."
        Case WSAECONNREFUSED
            strDesc = "Connection refused."
        Case WSAECONNRESET
            strDesc = "Connection reset by peer."
        Case WSAEDESTADDRREQ
            strDesc = "Destination address required."
        Case WSAEFAULT
            strDesc = "Bad address."
        Case WSAEHOSTDOWN
            strDesc = "Host is down."
        Case WSAEHOSTUNREACH
            strDesc = "No route to host."
        Case WSAEINPROGRESS
            strDesc = "Operation now in progress."
        Case WSAEINTR
            strDesc = "Interrupted function call."
        Case WSAEINVAL
            strDesc = "Invalid argument."
        Case WSAEISCONN
            strDesc = "Socket is already connected."
        Case WSAEMFILE
            strDesc = "Too many open files."
        Case WSAEMSGSIZE
            strDesc = "Message too long."
        Case WSAENETDOWN
            strDesc = "Network is down."
        Case WSAENETRESET
            strDesc = "Network dropped connection on reset."
        Case WSAENETUNREACH
            strDesc = "Network is unreachable."
        Case WSAENOBUFS
            strDesc = "No buffer space available."
        Case WSAENOPROTOOPT
            strDesc = "Bad protocol option."
        Case WSAENOTCONN
            strDesc = "Socket is not connected."
        Case WSAENOTSOCK
            strDesc = "Socket operation on nonsocket."
        Case WSAEOPNOTSUPP
            strDesc = "Operation not supported."
        Case WSAEPFNOSUPPORT
```

```
            strDesc = "Protocol family not supported."
        Case WSAEPROCLIM
            strDesc = "Too many processes."
        Case WSAEPROTONOSUPPORT
            strDesc = "Protocol not supported."
        Case WSAEPROTOTYPE
            strDesc = "Protocol wrong type for socket."
        Case WSAESHUTDOWN
            strDesc = "Cannot send after socket shutdown."
        Case WSAESOCKTNOSUPPORT
            strDesc = "Socket type not supported."
        Case WSAETIMEDOUT
            strDesc = "Connection timed out."
        Case WSATYPE_NOT_FOUND
            strDesc = "Class type not found."
        Case WSAEWOULDBLOCK
            strDesc = "Resource temporarily unavailable."
        Case WSAHOST_NOT_FOUND
            strDesc = "Host not found."
        Case WSANOTINITIALISED
            strDesc = "Successful WSAStartup not yet performed."
        Case WSANO_DATA
            strDesc = "Valid name, no data record of requested type."
        Case WSANO_RECOVERY
            strDesc = "This is a nonrecoverable error."
        Case WSASYSCALLFAILURE
            strDesc = "System call failure."
        Case WSASYSNOTREADY
            strDesc = "Network subsystem is unavailable."
        Case WSATRY_AGAIN
            strDesc = "Nonauthoritative host not found."
        Case WSAVERNOTSUPPORTED
            strDesc = "Winsock.dll version out of range."
        Case WSAEDISCON
            strDesc = "Graceful shutdown in progress."
        Case Else
            strDesc = "Unknown error."
    End Select
    '
    GetErrorDescription = strDesc
    '
End Function

Public Function InitializeWinsock(Version As Long) As Long
    '
    Dim udtWinsockData  As WSADATA
    Dim lngRetValue     As Long
    '
    'start up winsock service
    lngRetValue = WSAStartup(Version, udtWinsockData)
    '
    'assign returned value
    InitializeWinsock = lngRetValue
    '
End Function


Public Function vbSocket(ByVal AdrFamily As Long, ByVal SckType As
Long, ByVal SckProtocol As Long) As Long
    '
    On Error GoTo vbSocket_Err_Handler
```

```
    '
    Dim lngRetValue As Long 'value returned by the socket API function
    '
    'Call the socket Winsock API function
    'in order create a new socket
    lngRetValue = socket(AdrFamily, SckType, SckProtocol)
    '
    'Assign returned value
    vbSocket = lngRetValue
    '
EXIT_LABEL:
    Exit Function

vbSocket_Err_Handler:
    '
    lngCodErro = Err.LastDllError
    strDescErro = GetErrorDescription(lngCodErro)
    vbSocket = INVALID_SOCKET
    '
End Function

Public Function vbConnect(ByVal lngSocket As Long, _
      ByVal strRemoteHost As String, ByVal intRemotePort As Integer) _
      As Long
    '
    Dim udtSocketAddress As sockaddr_in
    Dim lngReturnValue   As Long
    Dim lngAddress       As Long
    '
    On Error GoTo ERROR_HANDLER
    '
    vbConnect = SOCKET_ERROR
    '

    If Not lngSocket > 0 Then
        Exit Function
    End If
    '

    If Len(strRemoteHost) = 0 Then
        Exit Function
    End If
    '

    If Not intRemotePort > 0 Then
        Exit Function
    End If

    Dim objProtocol As SOCKET_OPTIONS

    Dim lngAdrFamily As Long
    '
    objProtocol = GetBySocketHandle(lngSocket)
    '
    lngAdrFamily = objProtocol.mvarAddressFamily
    '

    lngAddress = GetAddressLong(strRemoteHost)
    '
    If lngAddress = INADDR_NONE Then
        '
```

```
        Exit Function
        '
    End If
    '
    With udtSocketAddress
        '
        .sin_addr = lngAddress

        .sin_port = htons(UnsignedToInteger(CLng(intRemotePort)))

        .sin_family = lngAdrFamily

    End With
    '
    vbConnect = connect(lngSocket, udtSocketAddress, _
      LenB(udtSocketAddress))
    '
EXIT_LABEL:
    Exit Function
    '
ERROR_HANDLER:
    '
    lngCodErro = Err.LastDllError
    strDescErro = GetErrorDescription(lngCodErro)
    vbConnect = SOCKET_ERROR

    '
End Function


Public Function vbSend(ByVal lngSocket As Long, strData As String) _
      As Long
    '
    Dim arrBuffer()     As Byte
    Dim lngBytesSent    As Long
    Dim lngBufferLength As Long
    '
    lngBufferLength = Len(strData)
    '
    If IsConnected(lngSocket) And lngBufferLength > 0 Then
        '
        'Convert the data string to a byte array
        arrBuffer() = StrConv(strData, vbFromUnicode)
        '
        'Call the send Winsock API function in order to send data
        lngBytesSent = send(lngSocket, arrBuffer(0), _
            lngBufferLength, 0&)
        '
        vbSend = lngBytesSent
        '
    Else
        '
        lngCodErro = Err.LastDllError
        strDescErro = GetErrorDescription(lngCodErro)
        vbSend = SOCKET_ERROR
        '
    End If
    '
End Function
```

```
Public Function vbRecv(ByVal lngSocket As Long, strBuffer As String) _
      As Long
    '
    Const MAX_BUFFER_LENGTH As Long = 8192
    '
    Dim arrBuffer(1 To MAX_BUFFER_LENGTH)   As Byte
    Dim lngBytesReceived                    As Long
    Dim strTempBuffer                       As String

    If IsDataAvailable(lngSocket) Then
        '

        lngBytesReceived = recv(lngSocket, arrBuffer(1), _
            MAX_BUFFER_LENGTH, 0&)
        '
        If lngBytesReceived > 0 Then

            strTempBuffer = StrConv(arrBuffer, vbUnicode)
            '

            strBuffer = Left$(strTempBuffer, lngBytesReceived)

        End If
        vbRecv = lngBytesReceived
        '
    Else

        lngCodErro = Err.LastDllError
        strDescErro = GetErrorDescription(lngCodErro)
        vbRecv = SOCKET_ERROR
    End If
    '
End Function

Public Function IsConnected(ByVal lngSocket As Long) As Boolean
    '
    Dim udtRead_fd      As fd_set
    Dim udtWrite_fd     As fd_set
    Dim udtError_fd     As fd_set
    Dim lngSocketCount  As Long
    '
    udtWrite_fd.fd_count = 1
    udtWrite_fd.fd_array(1) = lngSocket
    '
    lngSocketCount = vbselect(0&, udtRead_fd, udtWrite_fd, _
            udtError_fd, 0&)
    '
    IsConnected = CBool(lngSocketCount)
    '
End Function

Public Function IsDataAvailable(ByVal lngSocket As Long) As Boolean
    '
    Dim udtRead_fd As fd_set
    Dim udtWrite_fd As fd_set
    Dim udtError_fd As fd_set
    Dim lngSocketCount As Long
    '
    udtRead_fd.fd_count = 1
    udtRead_fd.fd_array(1) = lngSocket
    '
```

```
    lngSocketCount = vbselect(0&, udtRead_fd, udtWrite_fd, _
      udtError_fd, 0&)
    '
    IsDataAvailable = CBool(lngSocketCount)
    '
End Function

Public Sub SocketsCleanup()

   If WSACleanup() <> 0 Then
       MsgBox "Windows Sockets error occurred in Cleanup.", _
            vbExclamation
   End If

End Sub


Public Function SocketsInitialize() As Boolean

   Dim WSAD As WSADATA

   SocketsInitialize = WSAStartup(WS_VERSION_REQD, WSAD) = IP_SUCCESS

End Function

Public Function GetIPFromHostName(sHostName As String) As String

  'converts a host name to an IP address.

   Dim nbytes As Long
   Dim ptrHosent As Long  'address of hostent structure
   Dim ptrName As Long    'address of name pointer
   Dim ptrAddress As Long 'address of address pointer
   Dim ptrIPAddress As Long
   Dim sAddress As String

   sAddress = Space$(4)

   ptrHosent = gethostbyname(sHostName & vbNullChar)

   If ptrHosent <> 0 Then

      ptrName = ptrHosent
      ptrAddress = ptrHosent + 12
      CopyMemory ptrName, ByVal ptrName, 4
      CopyMemory ptrAddress, ByVal ptrAddress, 4
      CopyMemory ptrIPAddress, ByVal ptrAddress, 4
      CopyMemory ByVal sAddress, ByVal ptrIPAddress, 4

      GetIPFromHostName = IPToText(sAddress)

   Else

      lngCodErro = Err.LastDllError
      strDescErro = GetErrorDescription(lngCodErro)
      GetIPFromHostName = CStr(lngCodErro)

   End If

End Function
```

```
Public Function Ping(sAddress As String, _
                     sDataToSend As String, _
                     ECHO As ICMP_ECHO_REPLY) As Long

   Dim hPort As Long
   Dim dwAddress As Long

   dwAddress = inet_addr(sAddress)

   If dwAddress <> INADDR_NONE Then

      hPort = IcmpCreateFile()

      If hPort Then

         Call IcmpSendEcho(hPort, dwAddress, sDataToSend, _
            Len(sDataToSend), 0, ECHO, Len(ECHO), PING_TIMEOUT)

         Ping = ECHO.Status
         Call IcmpCloseHandle(hPort)

      End If

   Else:
      Ping = INADDR_NONE

   End If

End Function

Public Function GetStatusCode(Status As Long) As String

   Dim msg As String

   Select Case Status
      Case IP_SUCCESS:               msg = "ip success"
      Case INADDR_NONE:              msg = "inet_addr: bad IP format"
      Case IP_BUF_TOO_SMALL:         msg = "ip buf too_small"
      Case IP_DEST_NET_UNREACHABLE:  msg = "ip dest net unreachable"
      Case IP_DEST_HOST_UNREACHABLE: msg = "ip dest host unreachable"
      Case IP_DEST_PROT_UNREACHABLE: msg = "ip dest prot unreachable"
      Case IP_DEST_PORT_UNREACHABLE: msg = "ip dest port unreachable"
      Case IP_NO_RESOURCES:          msg = "ip no resources"
      Case IP_BAD_OPTION:            msg = "ip bad option"
      Case IP_HW_ERROR:              msg = "ip hw_error"
      Case IP_PACKET_TOO_BIG:        msg = "ip packet too_big"
      Case IP_REQ_TIMED_OUT:         msg = "ip req timed out"
      Case IP_BAD_REQ:               msg = "ip bad req"
      Case IP_BAD_ROUTE:             msg = "ip bad route"
      Case IP_TTL_EXPIRED_TRANSIT:   msg = "ip ttl expired transit"
      Case IP_TTL_EXPIRED_REASSEM:   msg = "ip ttl expired reassem"
      Case IP_PARAM_PROBLEM:         msg = "ip param_problem"
      Case IP_SOURCE_QUENCH:         msg = "ip source quench"
      Case IP_OPTION_TOO_BIG:        msg = "ip option too_big"
      Case IP_BAD_DESTINATION:       msg = "ip bad destination"
      Case IP_ADDR_DELETED:          msg = "ip addr deleted"
      Case IP_SPEC_MTU_CHANGE:       msg = "ip spec mtu change"
      Case IP_MTU_CHANGE:            msg = "ip mtu_change"
      Case IP_UNLOAD:                msg = "ip unload"
      Case IP_ADDR_ADDED:            msg = "ip addr added"
      Case IP_GENERAL_FAILURE:       msg = "ip general failure"
```

```
      Case IP_PENDING:              msg = "ip pending"
      Case PING_TIMEOUT:            msg = "ping timeout"
      Case Else:                    msg = "unknown  msg returned"
   End Select

   GetStatusCode = CStr(Status) & "   [ " & msg & " ]"

End Function

Private Function IPToText(ByVal IpAddress As String) As String

   IPToText = CStr(Asc(IpAddress)) & "." & _
              CStr(Asc(Mid$(IpAddress, 2, 1))) & "." & _
              CStr(Asc(Mid$(IpAddress, 3, 1))) & "." & _
              CStr(Asc(Mid$(IpAddress, 4, 1)))

End Function

Public Function Le_Alarme(ByVal intindice As Integer, _
      almAlarme As ALARME_OPCOES) As ALARME_OPCOES

        almAlarme.strCodigo = strAlarmes(intindice, 1)
        almAlarme.strTipo = strAlarmes(intindice, 2)
        almAlarme.strVar = strAlarmes(intindice, 3)
        almAlarme.strDepois = strAlarmes(intindice, 4)
        almAlarme.strAntes = strAlarmes(intindice, 5)
        almAlarme.strAV_Comp1 = strAlarmes(intindice, 6)

        If strAlarmes(intindice, 7) = "" Then
            almAlarme.lngAV_Valor1 = "0"
        Else
            almAlarme.lngAV_Valor1 = strAlarmes(intindice, 7)
        End If

        almAlarme.strAV_Logico = strAlarmes(intindice, 8)
        almAlarme.strAV_Comp2 = strAlarmes(intindice, 9)

        If strAlarmes(intindice, 10) = "" Then
            almAlarme.lngAV_Valor2 = "0"
        Else
            almAlarme.lngAV_Valor2 = strAlarmes(intindice, 10)
        End If

        almAlarme.strAV_Acao = strAlarmes(intindice, 11)
        almAlarme.strAV_Dest = strAlarmes(intindice, 12)
        almAlarme.strAL_Comp1 = strAlarmes(intindice, 13)

        If strAlarmes(intindice, 14) = "" Then
            almAlarme.lngAL_Valor1 = "0"
        Else
            almAlarme.lngAL_Valor1 = strAlarmes(intindice, 14)
        End If

        almAlarme.strAL_Logico = strAlarmes(intindice, 15)
        almAlarme.strAL_Comp2 = strAlarmes(intindice, 16)

        If strAlarmes(intindice, 17) = "" Then
            almAlarme.lngAL_Valor2 = "0"
        Else
            almAlarme.lngAL_Valor2 = strAlarmes(intindice, 17)
        End If
```

```
        almAlarme.strAL_Acao = strAlarmes(intindice, 18)
        almAlarme.strAL_Dest = strAlarmes(intindice, 19)

        Le_Alarme = almAlarme

End Function

Public Function Envia_Email(Optional ByVal strIp As String, _
      Optional ByVal strPortaTCP As String, _
      Optional ByVal strDest As String, _
      Optional ByVal strTipo As String, _
      Optional ByVal strDesc1 As String, _
      Optional ByVal strDesc2 As String, _
      Optional ByVal strNomeVar As String, _
      Optional ByVal strValor As String) As String

    Dim strDados As String
    Dim lngInicializa As Long
    Dim lngConecta As Long
    Dim lngSocket As Long
    Dim strRecebe As String
    Dim strEnvia As String
    Dim almAlarme As ALARME_OPCOES
    Dim lngValor As Long
    Dim strSubject As String
    Dim strMail As String


    lngInicializa = Inicializa_WinSock
    If lngInicializa <> 0 Then
        strPontoErro = "Inicializa"
        strStatus = CStr(lngInicializa)
        GoTo Grava_Erro
    End If

    lngSocket = Cria_Winsock

    lngConecta = Conecta_Winsock(lngSocket, strIp, strPortaTCP)

    If lngConecta >= 10000 Then
        strPontoErro = "Conecta"
        strStatus = CStr(lngConecta)
        GoTo Grava_Erro
    End If


    strDados = ""

    strEmailOrigem = Parametros.ORIG_EMAIL

    strSubject = strTipo & " - " & strDesc1 & " - " & strDesc2

    strMail = Now & " - " & strNomeVar & " = " & strValor

    ResponseCode = "220"

    For x = 1 To 7

        strRecebe = Recebe_Dados(lngSocket)
```

```
If Val(strRecebe) >= 10000 Then
    strPontoErro = "Recebe"
    strStatus = strRecebe
    GoTo Grava_Erro
End If

If ResponseCode = Left$(strRecebe, 3) Then

Else
    strPontoErro = "Cod.Resposta"
    strStatus = Left$(strRecebe, 3)
    GoTo Grava_Erro
End If

If ResponseCode = "221" Then GoTo Fim

strDados = strDados & strRecebe

Select Case x

    Case 1

        Helo = strEmailOrigem

        ResponseCode = 250
        strEnvia = Envia_Dados(lngSocket, "HELO " & Helo)

    Case 2

        ResponseCode = 250
        strEnvia = Envia_Dados(lngSocket, "MAIL FROM: <" & _
                Trim$(strEmailOrigem) & ">")

    Case 3

        ResponseCode = 250
        strEnvia = Envia_Dados(lngSocket, "RCPT TO: <" & _
                Trim$(strDest) & ">")

    Case 4

        ResponseCode = 354
        strEnvia = Envia_Dados(lngSocket, "DATA")

    Case 5

        ResponseCode = 250
        Call SendMimetxt(lngSocket, strEmailOrigem, _
                Trim$(strDest), strSubject, strMail)

    Case 6

        ResponseCode = 221
        strEnvia = Envia_Dados(lngSocket, "QUIT")

End Select

If Len(strEnvia) <> 0 Then
    strPontoErro = "Envia"
    strStatus = strEnvia
    GoTo Grava_Erro
```

```
        End If

    Next x

    strStatus = "0"
    strPontoErro = "0"

    GoTo Fim


Grava_Erro:

    strData = Now()
    strTipoLog = "E-MAIL"
    strNomeArq = Parametros.PASTA_LOG & Parametros.NOME_LOG

    Open GetShortPath(strNomeArq) For Append As #1

    Write #1, strTipoLog, strData, strDesc1, strDesc2, strIp, "0", _
      "0", "0", IIf(lngCodErro <> 0, CStr(lngCodErro), "999999")

    Close #1    ' Fecha o arquivo.

Fim:

    Envia_Email = strStatus

End Function

Public Function Telnet_Obj(strIp As String, _
      ByVal strPortaTCP As String, strCmd As TNET_Comando, _
      strDesc1 As String, strDesc2 As String) As String

    Dim strDados As String
    Dim lngInicializa As Long
    Dim lngConecta As Long
    Dim lngSocket As Long
    Dim strRecebe As String
    Dim strEnvia As String
    Dim almAlarme As ALARME_OPCOES
    Dim lngValor As Long
    Dim blnAviso As Boolean

    lngSocket = Cria_Winsock

    lngConecta = Conecta_Winsock(lngSocket, strIp, strPortaTCP)

    If lngConecta >= 10000 Then

        strPontoErro = "Conecta"
        strStatus = CStr(lngConecta)



       For y = 1 To 200
            If strAlarmes(y, 2) = strAlarme Then
                almAlarme = Le_Alarme(y, almAlarme)
            End If
       Next y

        blnAlarme = True
```

```
            almAlarme.strVar = "SEM_COMUNICAÇÃO"
            strValor = "Erro: " & CStr(lngCodErro) & " - " & strDescErro
            GoTo Grava

        End If

        strDados = ""

        For x = 1 To 10

            strRecebe = Recebe_Dados(lngSocket)

            If Val(strRecebe) >= 10000 Then
                strPontoErro = "Recebe"
                strStatus = strRecebe
                GoTo Grava
            End If

            strDados = strDados & strRecebe

            If strCmd.strCom(x) = "" Then Exit For

            strEnvia = Envia_Dados(lngSocket, strCmd.strCom(x))

            If Len(strEnvia) <> 0 Then
                strPontoErro = "Envia"
                strStatus = strEnvia
                GoTo Grava
            End If

        Next x

        strStatus = "0"
        strPontoErro = "0"

    Grava:

        strData = Now()
        strTipoLog = "Telnet"
        strNomeArq = Parametros.PASTA_LOG & Parametros.NOME_LOG

        Open GetShortPath(strNomeArq) For Append As #1
        Write #1, strTipoLog, strData, strDesc1, strDesc2, strIp, _
                strPortaTCP, "0", strPontoErro, strStatus

        Close #1

        If strStatus <> "0" Then
            lngCodErro = Err.LastDllError
            strDescErro = GetErrorDescription(lngCodErro)
        End If

        If lngConecta >= 10000 Then GoTo Alarme

        If strStatus <> "0" Then GoTo Fim

    Alarmes:

        If strAlarme <> "" Then

            For x = 1 To 200
```

```
If strAlarmes(x, 2) = strAlarme Then

    almAlarme = Le_Alarme(x, almAlarme)

    intpos1 = InStr(strDados, almAlarme.strDepois) + _
            Len(almAlarme.strDepois) - 1

    strDepois = Right(strDados, Len(strDados) - intpos1)

    intpos2 = InStr(strDepois, almAlarme.strAntes)

    If intpos2 > 0 Then

        strValor = Mid(strDados, intpos1 + 1, intpos2 - 1)

        lngValor = CLng(strValor)

        blnCond1 = Testa_Condicao(lngValor, _
            almAlarme.strAV_Comp1, _
            CLng(almAlarme.lngAV_Valor1))
        blnCond2 = Testa_Condicao(lngValor, _
            almAlarme.strAV_Comp2, _
            CLng(almAlarme.lngAV_Valor2))
        blnAviso = Testa_Logico(blnCond1, _
            almAlarme.strAV_Logico, blnCond2)
     Else
        blnAviso = False
    End If

    If blnAviso Then

        Select Case almAlarme.strAV_Acao

            Case "LOG"
                strData = Now()
                strTipoLog = "Aviso"
                strNomeArq = Parametros.PASTA_LOG & _
                  Parametros.NOME_LOG_ALARME

                Open GetShortPath(strNomeArq) _
                  For Append As #1

                Write #1, strTipoLog, strData, strDesc1, _
                        strDesc2, strIp, strPortaTCP, _
                        almAlarme.strVar, lngValor

                Close #1    ' Fecha o arquivo.

            Case "EMAIL"

                strData = Now()
                strTipoLog = "Aviso"
                strDest = almAlarme.strAV_Dest
                strOrigem = Parametros.ORIG_EMAIL
                strServSMTP = Parametros.SERVER_SMTP
                strPortaSMTP = Parametros.PORT_SMTP
```

```
                    blnEnvia = Envia_Email(strServSMTP, _
                           strPortaSMTP, strDest, strTipoLog,_
                           strDesc1, strDesc2, _
                           almAlarme.strVar, strValor)

                End Select

            End If

            If intpos2 > 0 Then

                blnCond1 = Testa_Condicao(lngValor, _
                   almAlarme.strAL_Comp1, _
                   CLng(almAlarme.lngAL_Valor1))
                blnCond2 = Testa_Condicao(lngValor, _
                   almAlarme.strAL_Comp2, _
                   CLng(almAlarme.lngAL_Valor2))
                blnAlarme = Testa_Logico(blnCond1, _
                   almAlarme.strAL_Logico, blnCond2)

            Else

                blnAlarme = False

            End If

Alarme:
            If blnAlarme Then

                Select Case almAlarme.strAL_Acao

                    Case "LOG"
                        strData = Now()
                        strTipoLog = "Alarme"
                        strNomeArq = Parametros.PASTA_LOG & _
                          Parametros.NOME_LOG_ALARME

                        Open GetShortPath(strNomeArq) For _
                          Append As #1

                        Write #1, strTipoLog, strData, strDesc1, _
                          strDesc2, strIp, strPortaTCP, _
                          almAlarme.strVar, lngValor

                        Close #1    ' Fecha o arquivo.

                    Case "EMAIL"

                        strData = Now()
                        strTipoLog = "Alarme"
                        strDest = almAlarme.strAL_Dest
                        strOrigem = Parametros.ORIG_EMAIL
                        strServSMTP = Parametros.SERVER_SMTP
                        strPortaSMTP = Parametros.PORT_SMTP

                        blnEnvia = Envia_Email(strServSMTP, _
                          strPortaSMTP, strDest, strTipoLog, _
                          strDesc1, strDesc2, almAlarme.strVar, _
                          strValor)

                End Select
```

```
                    If lngConecta >= 10000 Then GoTo Fim

                End If

            End If

        Next x

    End If

Fim:

    lngCloseSocket = closesocket(lngSocket)
    SocketsCleanup
    Telnet_Obj = strStatus

End Function

Public Function Ping_Obj(strIp As String, strDesc1 As String, _
      strDesc2 As String, Optional intQtdErro As Integer) As String

   Dim ECHO As ICMP_ECHO_REPLY
   Dim pos As Long
   Dim Success As Long
   Dim strResposta(10) As String
   Dim strSend As String * 32
   Dim sIPAddress As String
   Dim lngValor As Long
   Dim almAlarme As ALARME_OPCOES
   Dim blnAviso As Boolean
   Dim blnAlarme As Boolean

   If intQtdErro = 0 Then intQtdErro = 5


   If SocketsInitialize() Then

        For i = 1 To intQtdErro

            blnAviso = True
            blnAlarme = True
            strSend = "Comando Ping"

            sIPAddress = GetIPFromHostName(strIp)

            Success = Ping(sIPAddress, strSend, ECHO)

            strResposta(0) = GetStatusCode(Success)
            strResposta(1) = ECHO.Address
            strResposta(2) = ECHO.RoundTripTime & " ms"
            strResposta(3) = ECHO.DataSize & " bytes"

            If Left$(ECHO.Data, 1) <> Chr$(0) Then
                pos = InStr(ECHO.Data, Chr$(0))
                strResposta(4) = Left$(ECHO.Data, pos - 1)
            End If

            strResposta(5) = ECHO.DataPointer
            strResposta(6) = ECHO.Options.Ttl & " ttl"
            strResposta(7) = ECHO.Status
```

```
            SocketsCleanup

            strData = Now()
            strTipoLog = "Ping  "
            strNomeArq = Parametros.PASTA_LOG & Parametros.NOME_LOG

            Open GetShortPath(strNomeArq) For Append As #1
            Write #1, strTipoLog, strData, strDesc1, strDesc2, _
                  strIp, strResposta(3), strResposta(2), _
                  strResposta(6), strResposta(7)

            Close #1    ' Fecha o arquivo.

            If strAlarme <> "" Then

                For x = 1 To 200

                    If strAlarmes(x, 2) = strAlarme Then

                        almAlarme = Le_Alarme(x, almAlarme)

                        Select Case Trim(almAlarme.strAntes)
                            Case "ms"
                                lngValor = ECHO.RoundTripTime
                            Case Else
                                blnAviso = False
                                blnAlarme = False

                        End Select

                        If blnAviso Then
                            blnCond1 = Testa_Condicao(lngValor, _
                              almAlarme.strAV_Comp1, _
                              CLng(almAlarme.lngAV_Valor1))
                            blnCond2 = Testa_Condicao(lngValor, _
                              almAlarme.strAV_Comp2, _
                              CLng(almAlarme.lngAV_Valor2))
                            blnAviso = Testa_Logico(blnCond1, _
                              almAlarme.strAV_Logico, blnCond2)
                        End If
Aviso:
                        If blnAviso Then
                            Select Case almAlarme.strAV_Acao
                                Case "LOG"
                                    strData = Now()
                                    strTipoLog = "Aviso"
                                    strNomeArq = Parametros.PASTA_LOG_
                                          & Parametros.NOME_LOG_ALARME
                                    strVar = almAlarme.strVar
                                    strPortaTCP = "0"
                                    strValor = CStr(lngValor)

                                    Open GetShortPath(strNomeArq) _
                                          For Append As #1

                                    Write #1, strTipoLog, strData, _
                                          strDesc1, strDesc2, strIp, _
                                          strPortaTCP, strVar, lngValor

                                    Close #1    ' Fecha o arquivo.
```

```
                    Case "EMAIL"
                        strData = Now()
                        strTipoLog = "Aviso"
                        strDest = almAlarme.strAV_Dest
                        strOrigem = Parametros.ORIG_EMAIL
                        strServSMTP = Parametros._
                              SERVER_SMTP
                        strPortaSMTP = Parametros._
                              PORT_SMTP
                        strValor = CStr(lngValor)


                        blnEnvia = Envia_Email(_
                              strServSMTP, strPortaSMTP, _
                              strDest, strTipoLog, _
                              strDesc1, strDesc2, _
                              almAlarme.strVar, strValor)

                    End Select

                End If

                If blnAlarme Then
                    blnCond1 = Testa_Condicao(lngValor, _
                      almAlarme.strAL_Comp1, _
                      CLng(almAlarme.lngAL_Valor1))
                    blnCond2 = Testa_Condicao(lngValor, _
                      almAlarme.strAL_Comp2, _
                      CLng(almAlarme.lngAL_Valor2))
                    blnAlarme = Testa_Logico(blnCond1, _
                      almAlarme.strAL_Logico, blnCond2)
                End If

Alarme:
                If blnAlarme Then

                    Select Case almAlarme.strAL_Acao

                        Case "LOG"
                            strData = Now()
                            strTipoLog = "Alarme"
                            strNomeArq = Parametros._
                                  PASTA_LOG & Parametros._
                                  NOME_LOG_ALARME

                             strVar = almAlarme.strVar
                            strPortaTCP = "0"

                            Open GetShortPath(strNomeArq) _
                                  For Append As #1

                            Write #1, strTipoLog, strData, _
                                  strDesc1, strDesc2, strIp, _
                                  strPortaTCP, strVar, intValor
                            Close #1    ' Fecha o arquivo.

                        Case "EMAIL"
                            strData = Now()
                            strTipoLog = "Alarme"
                            strDest = almAlarme.strAL_Dest
                            strOrigem = Parametros.ORIG_EMAIL
```

```
                                    strServSMTP = Parametros._
                                         SERVER_SMTP
                                    strPortaSMTP = Parametros._
                                         PORT_SMTP
                                    strValor = CStr(lngValor)

                                    blnEnvia = Envia_Email(_
                                         strServSMTP, strPortaSMTP, _
                                         strDest, strTipoLog, _
                                         strDesc1, strDesc2, _
                                         almAlarme.strVar, strValor)

                                End Select

                                If Val(sIPAddress) > 10000 Then

                                    GoTo Grava_Erro

                                End If

                            End If

                        End If

                    Next x

                End If

                If ECHO.Status = 0 Then Exit For

            Next i

            Ping_Obj = ECHO.Status

    Else

            GoTo Grava_Erro

    End If

    GoTo Fim

Grava_Erro:
     strData = Now()
     strTipoLog = "Ping  "
     strNomeArq = Parametros.PASTA_LOG & Parametros.NOME_LOG

     Open GetShortPath(strNomeArq) For Append As #1

     Write #1, strTipoLog, strData, strDesc1, strDesc2, strIp, "0", _
       "0", "0", IIf(lngCodErro <> 0, CStr(lngCodErro), "999999")

     Close #1    ' Fecha o arquivo.

Fim:


End Function
```

```
Public Sub Le_Objetos()

    Open GetShortPath(Parametros.PASTA_LOG & "OBJETOS.txt") _
      For Input As #1

    For i = 1 To 65535

        Line Input #1, strTexto

        intPos = 1

        For j = 1 To 30

            intPos = InStr(intPos, strTexto, " ; ")

            If intPos = 0 Then Exit For

            strObjetos(i, j) = Left(strTexto, intPos - 1)

            strTexto = Right(strTexto, Len(strTexto) - intPos - 2)

            intPos = 1

        Next j

        If EOF(1) Then Exit For

    Next i

    Close #1

End Sub

Public Sub Le_Alarmes()


    Open GetShortPath(Parametros.PASTA_LOG & "ALARMES.txt") _
      For Input As #1

    For i = 1 To 65535

        Line Input #1, strTexto

        intPos = 1

        For j = 1 To 30

            intPos = InStr(intPos, strTexto, " ; ")

            If intPos = 0 Then Exit For

            strAlarmes(i, j) = Left(strTexto, intPos - 1)

            strTexto = Right(strTexto, Len(strTexto) - intPos - 2)

            intPos = 1

        Next j

        If EOF(1) Then Exit For
```

```
    Next i

    Close #1

End Sub

Public Sub Le_Agenda()

    Open GetShortPath(Parametros.PASTA_LOG & "AGENDA.txt") _
      For Input As #1

    For i = 1 To 65535

        Line Input #1, strTexto

        intPos = 1

        For j = 1 To 20

            intPos = InStr(intPos, strTexto, " ; ")

            If intPos = 0 Then Exit For

            strAgenda(i, j) = Left(strTexto, intPos - 1)

            strTexto = Right(strTexto, Len(strTexto) - intPos - 2)

            intPos = 1

        Next j

        If EOF(1) Then Exit For

    Next i

    Close #1

End Sub

Public Sub Le_Parametros()


    LeVarTemp

    Open GetShortPath(strVarTemp & "Parametros.txt") For Input As #1

    Line Input #1, Parametros.PASTA_LOG
    Line Input #1, Parametros.EXEC_AUTO

    Close #1


    Open GetShortPath(Parametros.PASTA_LOG & "Parametros.txt") _
      For Input As #1

    For i = 1 To 65535

        Line Input #1, strTexto

        intPos = 1
```

```
For j = 1 To 3

    intPos = InStr(intPos, strTexto, " ; ")

    If intPos = 0 Then Exit For

    If j = 1 Then
        strchave = Left(strTexto, intPos - 1)
    End If

    If j = 3 Then

        Select Case strchave

            Case ""
                Exit For
            Case "QTD_TIMEOUT"
                Parametros.QTD_TIMEOUT = Left(strTexto, _
                     intPos - 1)
            Case "ORIG_EMAIL"
                Parametros.ORIG_EMAIL = Left(strTexto, _
                     intPos - 1)
            Case "SERVER_SMTP"
                Parametros.SERVER_SMTP = Left(strTexto, _
                     intPos - 1)
            Case "PORT_SMTP"
                Parametros.PORT_SMTP = Left(strTexto, _
                     intPos - 1)
            Case "PASTA_PLA"
                Parametros.PASTA_PLA = Left(strTexto, _
                     intPos - 1)
            Case "NOME_PLA"
                Parametros.NOME_PLA = Left(strTexto, _
                     intPos - 1)
            Case "PASTA_LOG"
                Parametros.PASTA_LOG = Left(strTexto, _
                     intPos - 1)
            Case "NOME_LOG"
                Parametros.NOME_LOG = Left(strTexto, _
                     intPos - 1)
            Case "NOME_LOG_ALARME"
                Parametros.NOME_LOG_ALARME = Left(strTexto, _
                     intPos - 1)
            Case "NOME_TIMER"
                Parametros.NOME_TIMER = Left(strTexto, _
                     intPos - 1)
            Case "EXEC_AUTO"
                Parametros.EXEC_AUTO = Left(strTexto, _
                     intPos - 1)

        End Select

    End If

    strTexto = Right(strTexto, Len(strTexto) - intPos - 2)

    intPos = 1

Next j

If EOF(1) Then Exit For
```

```
    Next i

    Close #1

End Sub

Public Sub Copia_Agenda()

    With Workbooks(Parametros.NOME_PLA).Worksheets("AGENDA")

    For i = 1 To 200

        If strAgenda(i, 1) = "" Then Exit For

        For j = 1 To 8

            Select Case j
                Case 5 And IsDate(strAgenda(i, j))
                    strAgendaPla = strAgendaPla & _
                        Format(.Cells(i + 2, j), "Short Date")
                Case 6 And IsDate(strAgenda(i, j))
                    strAgendaPla = strAgendaPla & _
                        Format(.Cells(i + 2, j), "hh:mm")
                Case Else
                    strAgendaPla = strAgendaPla & .Cells(i + 2, j)
            End Select

            strAgendaMem = strAgendaMem & strAgenda(i, j)
            If j = 5 And IsDate(strAgenda(i, j)) Then
                .Cells(i + 2, j) = CDate(strAgenda(i, j))
            Else
                .Cells(i + 2, j) = strAgenda(i, j)
            End If

        Next j


        If strAgendaPla = strAgendaMem Then
            strAgenda(i, 9) = .Cells(i + 2, 9)
            strAgenda(i, 10) = .Cells(i + 2, 10)
        Else
            strAgenda(i, 9) = ""
            strAgenda(i, 10) = ""
            .Cells(i + 2, 9) = strAgenda(i, 9)
            .Cells(i + 2, 10) = strAgenda(i, 10)
        End If

        strAgendaPla = ""
        strAgendaMem = ""

    Next i

    End With

End Sub
```

Apêndice B

Exemplos de Resultados Obtidos Utilizando-se o Aplicativo Telnet.exe

Resultado obtido de Switch Cisco Catalyst 2912

```
FastEthernet0/1 is up, line protocol is up
  Hardware is Fast Ethernet, address is 0004.4dbf.a001 (bia
0004.4dbf.a001)
  MTU 1500 bytes, BW 100000 Kbit, DLY 100 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
  Keepalive not set
  Auto-duplex (Full), Auto Speed (100), 100BaseTX/FX
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input never, output 00:00:01, output hang never
  Last clearing of ""show interface"" counters never
  Queueing strategy: fifo
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  5 minute input rate 54000 bits/sec, 6 packets/sec
  5 minute output rate 33000 bits/sec, 7 packets/sec
     25704852 packets input, 2123147562 bytes
     Received 117149 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 9 ignored
     0 watchdog, 30978 multicast
     0 input packets with dribble condition detected
     27356942 packets output, 401309083 bytes, 0 underruns
     0 output errors, 0 collisions, 1 interface resets
     0 babbles, 0 late collision, 0 deferred
     0 lost carrier, 0 no carrier
     0 output buffer failures, 0 output buffers swapped out
```

Resultado obtido de Roteador Cisco CPA 2501

```
Router>en
Password:
Router#sh int s0
Serial0 is up, line protocol is up
  Hardware is HD64570
  Interface is unnumbered.  Using address of Ethernet0 (192.168.0.200)
  MTU 1500 bytes, BW 64 Kbit, DLY 20000 usec, rely 255/255, load
11/255
  Encapsulation HDLC, loopback not set, keepalive set (10 sec)
  Last input 0:00:06, output 0:00:00, output hang never
  Last clearing of ""show interface"" counters 0:01:45
  Input queue: 0/75/0 (size/max/drops); Total output drops: 0
  Output queue: 0/64/0 (size/threshold/drops)
     Conversations  0/10 (active/max active)
     Reserved Conversations 0/0 (allocated/max allocated)
  5 minute input rate 1000 bits/sec, 2 packets/sec
  5 minute output rate 3000 bits/sec, 2 packets/sec
     356 packets input, 26644 bytes, 0 no buffer
```

```
     Received 14 broadcasts, 0 runts, 0 giants
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     369 packets output, 170573 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets, 0 restarts
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
Router#
```

Resultado obtido de Servidor POP

```
+OK POP-3 mailserver-99.ig.com.br - Sun Internet Mail Server -
sims.4.0.2001.07.26.11.50.p9 at Sat, 12 Oct 2002 23:21:22 -0300 (EST)
+OK User name accepted, password please
+OK Mailbox open, 105 messages
+OK BYE
```

Resultado obtido de Servidor DayTime

```
52588 02-11-10 19:29:18 00 0 0 943.4 UTC(NIST) *
```

Resultado obtido de Servidor SMTP

220 pavuna.terra.com.br ESMTP

Apêndice C

Lista de Siglas e Abreviaturas

API Application Programming Interfaces (Interfaces de Programação de Aplicativos)

BYTE Agrupamento de 8 bits

CRC Cyclic Redundancy Check (Teste de Redundância Cíclico)

CRLF Seqüência de caracteres CR (Carriage Return) + LF (Line Feed)

CSMA/CD Carrier Sense Multiple Access with Collision Detect (Detecção de Portadora em Acessos Múltiplos com Detecção de Colisão)

DAYTIME Serviço TCP/IP que imprime a data e a hora do dia

FTP File Transfer Protocol (Protocolo de Transferência de Arquivos)

HOST Computador de usuário final conectado à rede

IANA Internet Assigned Number Authority – Grupo responsável pela atribuição de constantes utilizadas pelos protocolos TCP/IP.

ICMP Internet Control Message Protocol (Protocolo de Controle de Mensagens da Internet)

IP Internet Protocolo (Protocolo de Inter-redes ou Internet)

LAN Local Area Network (Rede Local)

MAN Metropolitan Area Network (Rede Metropolitana)

NIC Network Interface Card (Cartão de Interface de Rede ou Placa de Rede)

OSPF Open Shortest Path First (Protocolo de Abertura do Caminho mais Curto Primeiro)

POP Post Office Protocol (Protocolo de Correio Eletrônico)

RFC Request for Coments

RIP Routing Information Protocol (Protocolo de Informações de Roteamento)

SMTP Simple Mail Transfer Protocol (Protocolo de Transferência de Correio Simples)

SNPM Simple Network Management Protocol (Protocolo Simplificado de Gerenciamento de Rede)

TCO Total Cost Ownership (Custo Total de Propriedade)

TCP Transfer Control Protocol (Protocolo de Controle de Transferência)

TCP/IP Família de protocolos utilizada na internet, cujo nome é formado pelos nomes dos dois principais protocolos

TELNET Protocolo para acesso terminal a outros computadores da rede

UDP User Datagram Protocol (Protocolo de datagrama do Usuário)

VBA Visual Basic for Applications

WAN Wide Area Network (Rede Geograficamente Distribuída)

WWW World Wide Web (Rede de Alcance Mundial)