UNIVERSIDADE FEDERAL DE SANTA CATARINA

DEPARTAMENTO DE INFORMÁTICA E DE ESTATÍSTICA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Gilberto Medeiros de Souza

Modelo de Alta Disponibilidade Aplicada a Monitoramento de Pacientes em UTI Baseado em Interfaces de Alto-Desempenho

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

Orientador: Prof. Dr Mário Antonio Ribeiro Dantas

Florianópolis, Agosto de 2005

Modelo de Alta Disponibilidade Aplicada a Monitoramento de Pacientes em UTI Baseado em Interfaces de Alto-Desempenho

Gilberto Medeiros de Souza

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação Área de Concentração (Computação Aplicada) e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

_____

Profº. Dr. Raul Sidnei Wazlawick

Coordenador do PPGCC

Banca Examinadora:

Mário Antonio Ribeiro Dantas, Dr.	João Bosco da Mota Alves, Dr.
João Cândido Lima Dovicchi, Dr.	Mauro Roisenberg, Dr.
Vitório Bruno Mazzola, Dr.	Li Shih Min, Dr.

O ato de monitorar e observar vem acompanhando o ser humano há tempos sendo muitas vezes, em épocas remotas assim como hoje, a fronteira entre permanecer ou perecer.

DEDICATÓRIA

Dedico este trabalho à memória de
minha inesquecível Mãe.
Maria Amélia Medeiros Costa.

AGRADECIMENTOS

Primeiramente a Deus, por me conceder amor, força, saúde, família e amigos que sempre acreditaram em mim e que juntos, fizeram deste sonho uma realidade.

A minha querida Mãe, que agora está no céu, mas pôde me ensinar que na vida devemos ser honestos, batalhadores e não desistir jamais. Hoje vive em minhas lembranças e levo comigo seu exemplo de vida.

A minha irmã Creusa e meu mano querido Marcel, que amo tanto e sempre demonstraram alegria e paz necessária para que eu não fraquejasse.

A Dirlei minha noiva e futura esposa, presente Divino que o destino me concedeu. Sua presença carinhosa e compreensiva nos momentos de angústia, tornou essa jornada menos pesada.

Ao meu orientador e querido amigo Professor João Bosco da Mota Alves, que foi a pessoa que acreditou e me fez entrar nessa loucura, defendendo e lutando comigo nos momentos mais díficeis desta história.

Ao meu novo orientador Professor Mário Dantas por ter me aceitado como orientando num momento difícil deste mestrado.

Ao meu querido e inesquecível “Guru” e “Lorde Inglês” Professor Luiz Fernando Jacinto Maia. Jamais vou esquecer dos diálogos regados a cafés e cervejas. Deveria ter te ouvido mais.

Aos amigos professores Li e Beth que me acolheram num momento de desespero, mas que com calma e tranquilidade adquiridas com o tempo, me ajudaram muito.

Aos professores do departamento pelo convívio e pelos ensinamentos que me proporcionaram.

A querida amiga Heloise Mânica, que sempre me incentivou para entrar nesta jornada.

Aos meus amigos do Coral da UFSC, que permitiram eu crescer musicalmente, resgatando-me ao convívio famíliar.

Ao meu amigo Tony pelas longas conversas em nosso cortiço, mas que me trazem boas recordações.

Ao meu amigo Jô que considero como um irmão na minha vida. Me ajudou piscológicamente muitas vezes para que eu não fraquejasse. Que saudade dos almoços improvisados.

Ao meu amigo Andrézão e família, que assim como o Jô tenho igual consideração, pois foi uma pessoa que abriu meus olhos para o mundo da pesquisa.

A minha querida amiga Verinha, que sempre me ajudou nos momentos de dificuldade. Muitíssimo obrigado!!

Ao meu amigo Ederson, vulgo “Boi”, que foi a pessoa que me acolheu dentro de seu laboratório quando eu não tinha onde fazer minhas pesquisas. Agradeço também por ter sido a pessoa que me arrumou a primeira bolsa dentro da UFSC.

Aos meus amigos do RexLab, Andréa por ter segurado a barra nos momentos difíceis, ao Gibinha pelas discussões, ao Gordinho Alex que me fez rir e chorar algumas vezes, a Tânia que no momento mais difícil acreditou que eu poderia continuar.

Ao meu amigo Maurício que me ajudou a enxergar aquilo que eu não conseguia ver e contribuiu significativamente para conclusão deste trabalho.

ÍNDICE

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE ACRÔNIMOS

A/D	Analógico/Digital
APD	Assitente Pessoal Digital
ARP	Address Resolution Protocol
ARPANET	Advanced Research Project Agency
BD	Banco de Dados
CGI	Commom Gateway Interface
CIM	Common Information Management
CIMOM	Common Information Management Object Managed
CMIP	Common Management Information Protocol
ECG	Electrocardiograma
EEG	Electroencefalograma
EEPROM	Electric Erasable Programable Read Only Memory
EMG	Electromiograma
FAPERGS	Fundação de Amparo a Pesquisa do Rio Grande do Sul
GUI	Graphical Universal Interface
HTML	Hipertext Markup Language
HTTP	Hipertext Transfer Protocol
ICMP	Internet Control Message Protocol
INCOR	Instituto do Coração
IP	Internet Protocol
ISO	International Organization for Standardization
MIB	Management Information Base
MSW	Micro Servidor *Web*
NASA	National Science América
NMS	Network Management Station
OD	Oxigênio
OpenNMS	Open Network Management System
OSI	Open Systems Interconnection
PC	Personal Computer
PCODE	Pseudocódigo

PDU	Protocol Data Unit
pH	Potencial Hidrogeniônico
PUCRS	Pontifícia Universidade Católica do Rio Grande do Sul
RAM	Random Access Memory
RexLab	Laboratório de Experimentação Remota
RFC	Request For Comments
RISC	Reduced Instructions Set Component
RNP	Rede Nacional de Ensino e Pesquisa
SEEPROM	Serial Electric Erasable Programable Read Only Memory
SMS	Short Message Service
SNMP	Simple Network Management Protocol
SRAM	Serial Random Access Memory
SUS	Sistema Único de Saúde
TCP	Transport Control Protocol
TCP/IP	Transport Control Protocol/ Internet Protocol
UART	Universal Asynchronous Receiver/Transmiter
UDP	User Datagram Protocol
UFPel	Universidade Federal de Pelotas
UFSC	Universidade Federal de Santa Catarina
URL	Uniform Resource Locator
UTI	Unidade de Terapia Intensiva
WBEM	*Web*-based Enterprise Management

RESUMO

A manutenção da vida tem se tornado um grande desafio. Novas tecnologias surgem diariamente para apoiar os profissionais da área da saúde. Tecnologias essas que podem falhar, seja por desgaste natural do hardware, interferências humanas ou do meio. Buscando amenizar essa problemática, o presente trabalho propõe um modelo de alta disponibilidade no monitoramento de pacientes em Unidades de Terapia Intensiva. Para tanto, abordou-se conceitos de alta disponibilidade, a teconoligia Micro Servidor *Web* (MSW), gerência de redes e o *software* Nagios. O MSW não possui gerência implementada, assim, um processo agente embarcado fica responsável por ler tanto as interrupções geradas pela aplicação de El Shheibia (2003) como as interrupções do microcontrolador AT90S8515. O processo agente envia a comunicação ao processo gerente, localizado no servidor, que tem a função de analisar a informação, identificando a origem do alarme e a causa (problemas com paciente ou hardware), acionando o Nagios que notifica por telefone os responsáveis por cada área (Técnica/Médica) e dispara um alarme sonoro para o corpo clínico da UTI. Estas tecnologias em conjunto fornecem uma alta disponibilidade no monitoramento dos sinais vitais do paciente, sem aumento substancial do custo financeiro.

Palavras Chave: Agente, Monitoração de Pacientes, Tolerância a Falha, Alta Disponibilidade.

ABSTRACT

The maintenance of the life has if turned a great challenge. New technologies appear daily to give support to the professionalsof the area of the health. Technologies that can fail, for natural aging of the hardware, human interferences or environment. Looking for to liven up that problematic one, this work proposes a model of high availability in the patients monitoring in Intensive Care Unit (ICU). For so much, it was approached concepts of high availability, the technology Micro *Web* Server (MSW), management networks and the software Nagios. The studied MSW not possess implemented management, thus, a process embedded agent is responsible for reading the interruptions generated by El Shheibia(2003) application as the interruptions of the microcontroller AT90S8515. The process agent sends the communication to the process manager, located in the server, that has the function of analyzing the information, identifying the origin of the alarm and the cause (problems with patient or hardware), alerting Nagios that notifies for telephone the responsible persons for each area (Technique/Medical) that execute a sound alarm for the clinical body of ICU. These technologies together supply a high availability in the monitoring of the patients vital signs, without substantial increase of the financial cost.

Key words: Agent, Patients Monitoring, Fault Tolerance, high availability.

1 INTRODUÇÃO

Com o crescimento das necessidades humanas, a tecnologia, por meio de pesquisas, vem procurando fornecer soluções para atender, do mais simples ao mais complexo e crítico dos serviços. A mescla de necessidades com soluções tecnológicas cerca a vida do ser humano de tal sorte que o mesmo vê-se, extremamente, dependente dessas. Soluções que devem fornecer mobilidade, confiabilidade, disponibilidade e segurança.

A necessidade do homem realizar monitoramentos mais precisos cresce diariamente nas mais diversas áreas, podendo ir desde monitoramento simples até situações complexas como o caso da astronomia, onde há lançamentos de satélites com tecnologia de ponta para realização de monitoramento do sistema hídrico do planeta (Ciência Hoje – acessado em 13/07/2005).

Na área da saúde, a função de monitorar torna-se fundamental, o que tem levado a um crescimento na procura por sistemas de monitoramento de pacientes de 6,7% ao ano e a uma perspectiva de faturamento de $8.2 bilhões até 2008 somente nos Estados Unidos, segundo publicação pelo *Freedonia Group* no site mindbranch.

Nos hospitais, o monitoramento de pacientes, principalmente, os em estado crítico, tornou-se imprescindível para um acompanhamento seguro e preventivo. Segundo Edward[1] (1990) *apud* El Shheibia (2003), já nos anos 50 havia uma preocupação por parte do corpo clínico em realizar observações constantes nos pacientes, que eram anotadas manualmente de tempo em tempo.

[1] EDWARD, H.; LESLIE, E. **Medical informatics:** computer applications in health care. USA: Addison-Wesley, 1990.

Segundo Charnovsck (2004), os pacientes internados em uma Unidade de Terapia Intensiva (UTI) apresentam condições críticas de saúde e por isso precisam de cuidados intensivos e acompanhamento constante.

Embora exista a real necessidade de monitoramento de pacientes críticos em hospitais, o monitoramento contínuo de sinais vitais de pacientes em leitos ainda é uma realidade distante para a maioria dos hospitais públicos, o que se deve ao alto custo de aquisição destes equipamentos (El Shheibia, 2003).

Buscando tornar o monitoramento acessível às mais diversas áreas, Silva (2002), pesquisador do RexLab (Laboratório de Experimentação Remota), apresentou um dispositivo de baixo custo capaz de realizar aquisições de sinais, processamento, análise e visualização destes via navegador *web*, possibilitando monitorar e controlar outros dispositivos elétricos remotamente. Com isso, El Shheibia (2003), também pesquisador do RexLab, pôde apresentar mais tarde, um modelo de monitoramento de sinais vitais em leitos de UTI com custo mais acessível.

O modelo apresentado por El Shheibia (2003) cumpre bem o papel de monitoramento dos sinais vitais, entretanto, por oferecer acompanhamento via *web*, torna-se dependente da internet, da estabilidade da rede e do estado funcional do próprio MSW que poderia falhar e não realizar o trabalho de aquisição dos sinais, sem a possibilidade de emitir alarmes para o corpo clínico da UTI, que devido a isso, somente tomaria ciência da falta do monitoramento algum tempo depois , tempo crucial para pacientes de risco, podendo custar a própria vida.

Diante deste cenário, abre-se algumas questões a serem respondidas:

- Como garantir maior disponibilidade no monitoramento dos sinais vitais dos pacientes?
- Em caso de falhas na monitoração tanto do paciente quanto da rede, como encaminhar as mensagens de alarmes?

- De que forma monitorar o estado interno do MSW se o próprio não possui gerência implementada?

A característica do MSW o classifica como um equipamento essencial e ao mesmo tempo crítico em um sistema de computação, exigindo, assim, ser acompanhado e gerenciado com finalidade de evitar falhas que poderão acarretar danos irrversíveis aos pacientes.

Tendo em vista a importância de aumentar a disponibilidade e prover mais estabilidade, ao modelo de El Shheibia (2003), para assegurar uma monitoração contínua dos pacientes, mesmo na presença de falhas no MSW, o presente trabalho vem com uma proposta de desenvolver um modelo que fosse capaz de atender às necessidades e questões levantadas.

1.1 Objetivos

1.1.1 Objetivo Geral

Propor um modelo que incorpore conceitos de tolerante a falha, capaz de proporcionar alta-disponibilidade na monitoração remota contínua dos sinais vitais dos pacientes, alarmando, as áreas médica e técnica, frente a presença iminente de falhas no hardware ou no paciente.

1.1.2 Objetivos Específicos

- Criar um modelo de integração entre o monitoramento de hardware e a do paciente;
- Modelar a função agente embarcado no MSW;
- Modelar a função gerente no Servidor;
- Adotar uma aplicação de gerência que possa ser usada juntamente com o modelo e que não implique em maiores investimentos.
- Implementar uma simulação do modelo;

1.2 Organização do Trabalho

Além do exposto neste capítulo, a seqüência do presente trabalho obedece a seguinte estrutura:

- No capítulo 2 é abordado as áreas funcionais de Gerência de Redes, o Protocolo SNMP, a plataforma WBEM e o Nágios. Introdução a informática e Saúde e alguns trabalhos de monitoramento remoto de pacientes. Uma apresentação do MSW e algumas aplicações que fazem uso da tecnologia.

- No capítulo 3 é feito uma discussão dos trabalhos e dos assuntos levantados nos capítulos anteriores, justificando e demonstrando a real contribuição desta pesquisa.

- No capítulo 4 é apresentado o modelo e os pré-requisitos necessários para realização da experimentação deste trabalho, comenta-se uma abordagem de Dependabilidade, onde enfatiza-se os meios de Tolerância, Prevenção e Previsão de Falhas.

- No capítulo 5 é exposto a conclusão do trabalho juntamente com as recomendações de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Gerência de Redes

As redes de computadores surgiram com o intuito de compartilhar recursos e informações. Com o passar tempo, seu crescimento, complexidade, tamanho e variedade de dispositivos, tornou sua administração uma tarefa um tanto quanto árdua aos administradores de rede. Diante disso, a gerência de redes tornou-se indispensável. Não bastasse a complexidade da rede com o compartilhamento de dispositivos e dados, a confiabilidade e disponibilidade destes tornaram-se de suma importância para as organizações (Specialski, 2000).

A importância de manter a estabilidade da rede cresce diariamente e, segundo Specialski (2000), afirma que por menor e mais simples que uma rede possa ser, com certeza necessitará de ferramentas de gerência, a fim de garantir aos seus usuários disponibilidade e desempenho aceitáveis.

De acordo com Specialski (2000), para efeito de tratamento destas e de outras questões, o gerente da rede deve focalizar um conjunto inicial de recursos a serem monitorados, a fim de estabelecer níveis de desempenho aceitáveis que inclui associar métricas e valores apropriados aos recursos. Afirma que por meio da monitoração o gerente da rede obterá informações suficientes sobre o nível de operação da rede.

Specialski (2000) relata que, colecionando e analisando estas informações, o gerente da rede pode ficar mais capacitado no reconhecimento de situações indicativas de degradação de desempenho, podendo atuar rapidamente na deficiência, estabelecendo assim, uma pro-atividade em relação a rede.

Franceschi (2003), relata que o gerenciamento de falhas se torna um ponto importante, afirmando que é muito comum a ocorrência de falhas durante as operações em uma rede. O autor coloca que as redes são equipamentos que

operam de forma distribuída e estão sujeitos a diversos tipos de falhas como falhas nos equipamentos devido à ação do tempo, como umidade e calor excessivo, falhas operacionais ou de uso indevido dos equipamentos e falhas de sobrecarga.

Franceschi (2003), também descreveu que, além de falhas, existem outras preocupações para os que utilizam ou administram redes. O autor comenta que, a configuração, o desempenho, a contabilização e a segurança são outros aspectos fundamentais na atividade de gerência de redes.

Para formalizar essas atividades consideradas fundamentais colocadas por Franchesci (2003), a ISO (*International Standards Organization*) procurou definir as seguintes áreas funcionais de gerência de redes:

- Gerência de Falhas;
- Gerência de Contabilização;
- Gerência de Configuração;
- Gerência de Desempenho; e
- Gerência de Segurança.

Stallings (1999) escreveu que, embora esta classificação funcional tenha sido desenvolvida para o ambiente OSI[2] (*Open System Interconnection*), teve uma grande aceitação por vendedores e proprietários de sistemas de gerenciamento de rede.

2.1.1 Gerência de Falhas

O objetivo da gerência de falhas, segundo Stallings (1999), é identificar falhas tão depressa quanto possível e identificar a causa de forma a agir rapidamente para restabelecer a situação.

Segundo Soares *et al.* (1995), a gerência de falhas se refere à detecção e ao diagnóstico de falhas ocorridas na rede, podendo ser feita de forma automática

[2] Padrão de arquitetura aberta baseado em camadas. Conhecido como Modelo de Referência OSI.

ou por meio de comandos de um operador. A gêrência de falhas, segundo a visão dos pesquisadores, é composta das seguintes funções:

- Detecção automática de erros de software e hardware;
- Monitoramento de parâmetros críticos como estatísticas de erros em portas de comutadores ou em enlaces físicos;
- Testes sobre módulos de hardware, enlaces físicos, portas etc., de acordo com a solicitação de um operador.

Na detecção de falhas e defeitos, qualquer mal funcionamento é detectado por verificação contínua ou periódica, tendo como resultado, informações de eventos de manutenção ou alarmes que podem ser disparados (Soares *et al*, 1995).

Lopes *et al* (2003), expõem que, gerência de falhas diz respeito à detecção de eventos anormais, o diagnóstico de problemas que levaram a esses eventos, acompanhamento e solução dos problemas. Alguns destes problemas dificultam a monitoração, o controle e o diagnóstico de falhas. Dupuy *et al*. (1989[3]) *apud* Stallings (1999), listaram alguns problemas específicos associados a observação de falhas:

- **Falhas não observáveis**: Certas falhas são localmente não observáveis, por exemplo, a existência de uma paralisação entre cooperação de processos distribuídos. Outras falhas podem não ser observáveis porque o equipamento do fornecedor não está instrumentado para registrar a ocorrência de uma falha.
- **Falhas parcialmente observáveis**: Um nó com falha pode ser observável, mas a observação pode ser insuficiente para definir e reconhecer o problema. Por exemplo, um nó pode não estar respondendo devido a falha de algum protocolo de baixo nível em algum dispositivo anexado ao nó.
- **Incerteza em observação**: Mesmo quando observações detalhadas de falhas são possíveis, pode haver incerteza e até inconsistências

[3] **DUPUY**, A., et al. **Network Fault Management: A User´s View**. Proceedings, First International Symposium on Integrated Network Management, May; published by North-Holland, 1989.

associadas com as observações. Por exemplo, falha de resposta de um dispositivo distante pode significar que o dispositivo está travado, a rede ficou dividida, o congestionamento causou a demora na resposta ou o relógio local está defeituoso.

Para Boutaba e Polyrakis (2002), o gerenciamento de falhas tem a responsabilidade de monitorar o estado de cada um dos objetos gerenciados, por sua manutenção e ações necessárias ao restabelecimento ou isolamento das unidades com problemas/falhas. Confrontando as informações coletadas com o mapa da rede, pode-se identificar quais elementos da rede permanecem funcionando, quais operam de forma precária e aqueles que, definitivamente, estão fora de operação.

Barotto (2002), ressalta a importância do gerenciamento de falhas criar um registro de ocorrências, um diagnóstico de falhas e uma correlação entre os resultados do diagnóstico, provendo com isso algumas ações de reparo. A autora ainda afirma que no gerenciamento de falhas, também pode ser utilizado hardware e software de tolerância a falhas ou redundantes, que podem continuar a fornecer serviços de rede mesmo quando ocorrer falha.

De acordo com Specialski (2000), o impacto e a duração do estado de falha pode ser minimizado pelo uso de componentes redundantes e rotas de comunicação alternativas, para dar à rede um grau de “tolerância à falhas”.

Em sua pesquisa, Specialski (2000) aborda que a falha pode ou não ser causada pela ausência de recurso, afirmando que a gerência de falhas pode ser reativa ou pró-ativa, onde, segundo sua concepção, o gerenciamento pró-ativo de falhas (muitas vezes tratado como gerenciamento de faltas) busca evitar a ocorrência de falhas no sistema gerenciado. Considera que uma falha é uma condição anormal cuja recuperação exige uma intervenção imediata e, preferencialmente, a ação de gerenciamento deve ser executada antes da ocorrência da falha, quando o sistema de gerenciamento detecta um comportamento anormal do recurso gerenciado.

Segundo Brisa[4] (1992) *apud* Specialski (2000), quando a falta induz à ocorrência de uma falha, é importante que seja possível rapidamente:

- determinar exatamente onde a falha ocorreu;
- isolar o resto da rede da falha, tal que ela continue a funcionar sem interferências;
- reconfigurar ou modificar a rede para minimizar o impacto da operação sem o componente que falhou;
- reparar ou trocar o componente com problemas para restaurar a rede ao seu estado anterior.

Pode-se então dizer que uma falha em um determinado serviço pode ser causada tanto por problemas existentes nos recursos dos quais o serviço depende, quanto por problemas na própria configuração do serviço ou, ainda, por ações erradas de seus usuários (Specialski, 2000).

2.1.2 Gerência de Configuração

O objetivo principal do gerenciamento de configuração é manter um registro detalhado das configurações de rede antigas, atuais e propostas. Dependendo do ambiente, esse conhecimento detalhado pode compreender uma lista muito grande de informações sobre configuração (Barotto, 2002).

Barotto (2002) afirma, ainda, que documentar a configuração de rede auxilia o administrador a entender os efeitos das alterações e falhas da estrutura. Acredita que, como todas as redes necessitam de manutenção, é de fundamental importância ter uma visão completa das mesmas.

Para Specialski (2000), o gerenciamento de configuração tem uma relação com a inicialização de recursos/serviços e com uma eventual desativação do todo ou em parte dos mesmos. A autora ainda comenta sobre a importância do

[4] **BRISA** – Sociedade Brasileira para Interconexão de Sistemas Abertos, Gerenciamento de Redes: **Uma abordagem de Sistemas Abertos**. MAKRON Books, São Paulo, 1992.

gerenciamento de configuração estar relacionado com as tarefas de manutenção, adição e atualização de relacionamentos entre os componentes e do *status* (estado atual) dos componentes durante a operação do sistema.

2.1.3 Gerência de Desempenho

A gerência de desempenho consiste no monitoramento das atividades e do controle dos recursos através de ajustes e trocas (Specialski, 2000). A autora aborda várias questões relativas ao gerenciamento do desempenho, dentre elas:

- Qual é o nível de utilização da capacidade?
- O tráfego é excessivo?
- O *throughput*[5] foi reduzido para níveis aceitáveis?
- Existem gargalos?
- O tempo de resposta está aumentando?
- Os usuários deste serviço estão satisfeitos?
- Qual a freqüência de manutenções realizadas sobre este equipamento?

Barotto (2002) traça um paralelo expondo que, enquanto o gerenciamento de falhas é principalmente reativo, o gerenciamento de desempenho é ativo, envolvendo a coleta e interpretação das medições periódicas dos indicadores de desempenho, identificando gargalos, avaliando tendências e fazendo previsões do desempenho futuro da rede.

Segundo Boutaba e Polyrakis (2002), a gerência de desempenho se preocupa com o desempenho corrente da rede, incluindo parâmetros estatísticos tais como atrasos, vazão, disponibilidade e número de retransmissões, consistindo num conjunto de funções que são responsáveis por manter e examinar registros com histórico dos estados do sistema para fins de planejamento e análise.

[5] Expressa o número de pacotes que foram gerados e recebidos sem perda de pacotes, por este parâmetro podemos medir o desempenho do chaveador quanto a recepção e transmissão de pacotes, procurando-se ter a menor perda possível, pois cada pacote perdido irá gerar uma retransmissão.

Specialski (2000), descreve que para o controle potencial de um sistema, cada componente essencial deve ser monitorado individualmente para garantir o seu perfeito funcionamento e, com isso, manter a estabilidade da rede como um todo. Afirma a autora, que a melhor maneira na qual uma rede pode ser gerenciada é atingir cada uma das cinco áreas da gerência de rede eficazmente. Para ajudar nesta tarefa, aplicações de monitoramento e protocolos de rede são usados de forma que o processo de gerência de rede torne-se automatizado.

2.2 Monitoramento e Controle de Redes

De acordo com Stallings (1999), a monitoração de uma administração de rede diz respeito a observação e análise do estado e comportamento de um *host* (sistema final), intermediário e sub-redes que compõem a configuração a ser administrada.

Os pesquisadores Chiu e Sudama (1992[6]) *apud* Stallings (1999), sugeriram que a monitoração de redes consiste em três áreas principais:

- **Acesso a Informação Monitorada**: Definição do monitoramento da informação e obtenção da informação de um recurso para um gerente;
- **Plano de Monitoração dos Mecanismos**: Obtenção da informação a partir dos recursos;
- **Aplicação da Informação Monitorada**: Utilização da informação monitorada nas várias áreas funcionais da gerência de redes.

2.2.1 Informações de Monitoração de Rede

Stallings (1999), afirma que antes de considerar um planejamento de monitoração de rede, faz-se necessário considerar quais informações seriam interessantes monitorar. Dentro deste contexto, o autor classifica as informações de monitoramento em estática, dinâmica e estatística.

[6] **CHIU**, d., and **SUDANA**, R. **Network Monitoring Explained: Design and Application**. New York: Ellis Horwood, 1992.

A informação estática é gerada tipicamente pelo elemento envolvido. Assim, um roteador poderá manter sua própria informação de configuração, podendo, esta informação, ficar disponível diretamente a um gerente se o elemento monitorado possuir o software de agente apropriado. Caso o elemento não seja gerenciável, alternativamente, a informação pode ficar disponível por meio de um agente *proxy* (procurador) que disponibilizará para um gerente (Stallings, 1999).

Segundo Stallings (1999), a informação dinâmica geralmente é colecionada e armazenada pelo elemento da rede responsável pelos eventos subordinados a ele. Porém, se um sistema é anexado a uma rede local, então muito de sua atividade pode ser observado por outro sistema na própria rede.

Informação estatística pode ser gerada por qualquer sistema que tem acesso à informação dinâmica, podendo ser gerada pelo próprio monitor da rede. Isto requeriria que toda as "linhas de informação" fossem transmitidas ao monitor onde seriam analisadas e resumidas (Stallings, 1999). Segundo o autor, o processamento no monitor poderia ser poupado caso o elemento da rede, que detém a informação dinâmica, pudesse processar e enviar o resumo ao monitor.

2.2.2 SNMP (Simple Netware Management Protocol)

Segundo Tanenbaum (1999), quando ainda se operava a ARPANET (*Advanced Research Project Agency*), caso houvesse uma demora inesperada no acesso a um *host*, a pessoa que detectasse o problema simplesmente executava o programa *Ping.* De acordo com autor, com um pouco de trabalho, normalmente era localizado o problema e alguma providência era tomada.

Com o passar do tempo, quando a ARPANET deu lugar a rede mundial, surgiram também *backbones* e operadores diversos, mostrando que a solução do *ping* já não era mais viável. Com isso surge a necessidade de se criarem

melhores ferramentas para o gerenciamento de rede. Houveram algumas tentativas de definição de ferramentas de gerenciamento com as RFC´s 1028 e 1067, as quais tiveram vida curta (Tanenbaum, 1999).

De acordo com Tanenbaum (1999), as pesquisas continuaram e em maio de 1990, a RFC 1157 foi publicada, surgindo assim a primeira versão do SNMP. Juntamente com o documento complementar à RFC 1155, que procurava tratar de informações de gerenciamento, segundo o pesquisador, o SNMP oferecia uma forma sistemática de monitorar e gerenciar uma rede de computadores.

O protocolo SNMP teve seu desenvolvimento nos anos 80 como resposta para os existentes problemas de gerenciamento em ambiente TCP/IP Internet, envolvendo redes heterogêneas. Inicialmente foi concebido para ser apenas uma solução provisória até o desenvolvimento de um protocolo de gerenciamento mais completo, o CMIP (*Common Management Information Protocol*), que é o protocolo de gerenciamento baseado na arquitetura OSI. Entretanto, a sua simplicidade, somada ao fato do protocolo já suprir as principais necessidades no gerenciamento de redes, fez com que o SNMP passasse a ser amplamente utilizado, sendo hoje considerado um padrão no gerenciamento de redes de computadores (Malowidzki, 2001[7], *apud* Barotto, 2002; Lopes *et al*.,2003).

Conforme pode ser observado na Figura 2.1, o SNMP é um protocolo da camada de aplicação e foi desenvolvido para facilitar a troca de informações de gerenciamento entre dispositivos de rede (Barotto, 2002).

[7] **MALOWIDZKI**, Marek. **The management of the mobile network with COM+ and SNMP**. Military Communications Conference,2001, Washington. Proceedings…[s.l.]: IEEE Press, 2001. 5 p., p. 1456 – 1460.

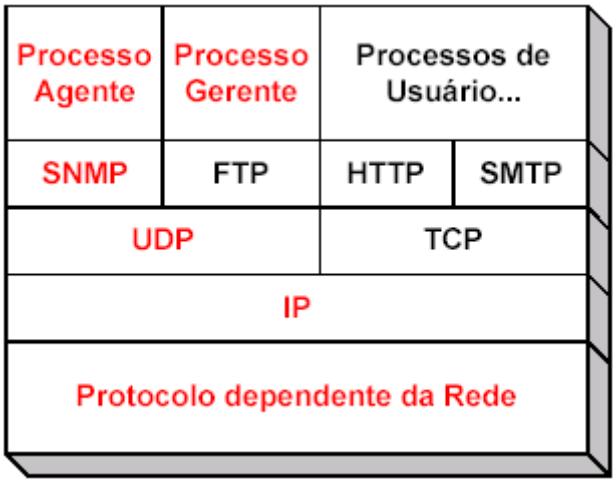

Figura 2.1 – Protocolo SNMP sobre as camadas TCP/IP (Barotto, 2002).

O modelo SNMP descrito por Tanenbaum (1999), consiste em quatro componentes básicos:

- Nós gerenciados;
- Estações de gerenciamento;
- Informações de gerenciamento; e
- Um protocolo de gerenciamento.

Segundo Tanenbaum (1999), os nós gerenciados podem ser *hosts*, roteadores, pontes, impressoras ou qualquer outro dispositivo com capacidade de comunicar informações de *status* (estado atual) para o mundo externo. O autor comenta que para um dispositivo ser gerenciável via SNMP, deve ser capaz de executar um processo de gerenciamento denominado agente SNMP e afirma que todos os computadores atendem a esse requisito, da mesma forma que os roteadores e dispositivos periféricos projetados para serem usados em rede.

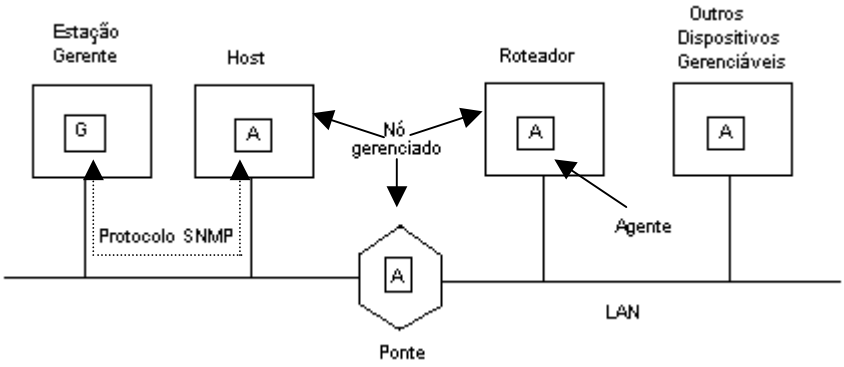

Figura 2.2 – Arquitetura do modelo SNMP baseado em Tanenbaum (1999).

De acordo com a arquitetura proposta por Tanenbaum (1999), o gerenciamento de rede é feito a partir de estações de gerenciamento, que na verdade são computadores genéricos que executam um software especial para gerenciar a rede. As estações de gerenciamento contém processos, como ilustrado na Figura 2.2, que se comunicam com os agentes que ficam espalhados pela rede. Essa comunicação é feita via o protocolo SNMP. Rech Filho (2004) traz uma demonstração mais ilustrativa do modelo (Gerente-Agente), conforme pode-se observar pela Figura 2.3.

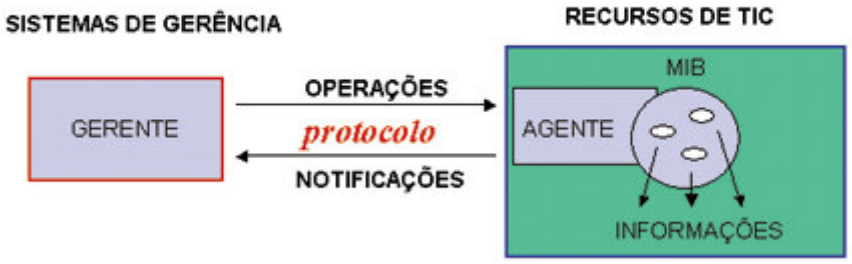

Figura 2.3 – Modelo de Gerente-Agente (Fonte: Rech Filho, 2004).

Tanenbaum (1999) afirma que nesse modelo (Gerente-Agente), toda a inteligência fica nas estações de gerenciamento. Com isso, os agentes podem ser mais simples, reduzindo o impacto nos dispositivos nos quais estão sendo executados.

Cada dispositivo mantém uma ou mais variáveis que descrevem seu estado. Logo, para que um dispositivo seja gerenciável, seria necessário ele ter implementado dentro de si o agente e a sua MIB (Management Information Base), pois, segundo Tanenbaum (1999), de nada adianta fazer perguntas de um determinado estado para o dispositivo se o mesmo não possui controle sobre este para dar respostas.

2.2.3 Paradigma Gerente-Agente

Segundo Soares *et al*. (1995), a monitoração do tráfego, estado e desempenho de um nó da rede, assim como a monitoração dos meios de transmissão e de outros sinais, é imprescindível e necessária para o gerenciamento da rede. A monitoração tem a condição de possibilitar a detecção de erros e diagnoses e resoluções de problemas/falhas.

Os processos que implementam as funções de gerenciamento atuam ou como agentes ou como gerentes. Os agentes coletam junto aos dispositivos gerenciados, as informações relevantes ao gerenciamento da rede. O gerente, por sua vez, processa essas informações com o objetivo de detectar falhas no funcionamento dos elementos da rede para que, rapidamente, possam ser tomadas providências no sentido de contornar os problemas (Soares *et al*., 1997).

Segundo publicação da RNP – Rede Nacional de Ensino e Pesquisas, a estação de gerenciamento é onde fica a aplicação gerente, servindo como interface para os administradores e gerentes de rede. O agente de gerenciamento responde às solicitações de informações e de ações da estação de gerenciamento e deve também prover assincronamente informações importantes que não foram solicitadas por esta estação (Carville, 2000).

O protocolo SNMP define as mensagens, unidades de dados chamadas PDU (*Protocol Data Unit*), para serem trocadas durante uma comunicação entre o gerente e o agente. A tabela 2.1 apresenta os cinco tipos de PDU´s SNMP.

Tabela 2.1 – PDU´s trocadas entre Gerente e Agente no SNMP

PDU	**Descrição**
get-request	mensagem enviada pelo gerente ao agente solicitando o valor de uma variável
get-next-request	mensagem utilizada pelo gerente para solicitar o valor da próxima variável depois de uma ou mais variáveis que foram especificadas
set-request	mensagem enviada pelo gerente ao agente para solicitar que seja alterado o valor de uma variável
get-response	mensagem enviada pelo agente ao gerente, informando o valor de uma variável que lhe foi solicitado
trap	mensagem enviada pelo agente ao gerente, informando um evento ocorrido

Fonte: RNP – Rede Nacional de Ensino e Pesquisas.

Uma vez que quatro das cinco mensagens SNMP são do tipo pergunta reposta, o protocolo SNMP usa o protocolo de transporte UDP (*User Datagram Protocol*). Além de ter sido projetado para operar sob UDP, um protocolo não orientado a conexão, o próprio SNMP também é um protocolo não orientado a conexão, sendo cada troca de mensagens uma transação diferente entre o agente e a estação de gerenciamento. Cada estação de gerenciamento, como também o agente, devem implementar os protocolos SNMP e, por conseqüência, UDP e IP para poderem se comunicar. Tal imposição exclui do processo de gerenciamento dispositivos que não suportam parte dos protocolos TCP/IP ou que, apesar de implementarem o TCP/IP para suportar suas aplicações, não desejam adicionar mais carga ao seu sistema com o suporte ao protocolo SNMP (RNP – Rede Nacional de Ensino e Pesquisa).

2.3 Sistemas de Gerenciamento de Redes

Os sistemas de gerenciamento de redes apresentam a vantagem de ter um conjunto de ferramentas para análise e depuração da rede (Oda, 1994). Estes sistemas podem apresentar uma série de mecanismos que facilitam a identificação, notificação e registro de problemas, como:

- Alarmes que indicam, através de mensagens ou bipes (sinal sonoro) de alerta, anormalidades na rede;
- Geração automática de relatórios contendo as informações coletadas;
- Facilidades para integrar novas funções ao próprio sistema de gerenciamento;
- Geração de gráficos estatísticos em tempo real;
- Apresentação gráfica da topologia das redes.

Em redes IP, o sistema de gerenciamento segue o modelo gerente-agente, onde o gerente é o próprio sistema de gerenciamento e o agente é um *software* que deve ser instalado nos equipamentos gerenciados. A tarefa do agente é responder as requisições feitas pelo gerente em relação ao equipamento no qual está instalado. Esta interação é viabilizada pelo protocolo de gerenciamento SNMP. Dessa forma, o gerente consegue conversar com qualquer máquina que fale SNMP, independentemente do tipo de *hardware* e sistema operacional. O conjunto de informações ao qual o gerente pode fazer requisições ou alterações é denominado de MIB - *Management Information Base* (Cisco, 1996).

Em alguns casos é necessário que a troca de informações seja em sentido inverso, isto é, o agente tem de passar informações para o gerente. O SNMP define a operação *Trap* para que um agente informe ao gerente a ocorrência de um evento específico (Cisco, 1996).

2.4 Ferramentas de Gerência

Atualmente existem no mercado diversos tipos de ferramentas que auxiliam o administrador nas atividades de gerenciamento. Segundo Oda (1994), estas ferramentas podem ser divididas em quatro categorias:

- Ferramentas de nível físico, que detectam problemas em termos de cabos e conexões de hardware;
- Monitores de rede, que se conectam as redes monitorando o tráfego;

- Analisadores de rede, que auxiliam no rastreamento e correção de problemas encontrados nas redes;
- Sistemas de gerenciamento de redes, os quais permitem a monitoração e o controle de uma rede inteira a partir de um ponto central.

Em seu livro – Melhores Práticas para Gerência de Rede de Computadores – Lopes *et al*. (2003), afirmam que as ferramentas de gerência são imprescindíveis no cotidiano das atividades de gerência, ajudando a detectar problemas quando estes ocorrem, ou antes mesmo de ocorrerem, classificando esse tipo de abordagem como gerência de rede pró-ativa.

2.4.1 Ferramentas Simples de Gerência

Segundo Lopes *et al*. (2003), as ferramentas simples são aquelas que não permitem um visão geral da rede, mas que muitas vezes podem ajudar a descobrir características mais internas de determinados elementos da rede. São oferecidas juntamente com o sistema operacional de rede dos próprios hospedeiros. Citam como exemplos o *traceroute* (*tracert* para máquinas *windows*), *ping*, *route*, *netstat*, *ifconfig* (sendo essas duas últimas ferramentas *Unix Like*) e o *ipconfig* (*windows*).

Os autores concluem que, embora em alguns momentos fosse necessário o uso de tais ferramentas, estas sozinhas não seriam suficientes para realizar o trabalho de gerência. Afirmam, ainda que geralmente essas ferramentas são utilizadas quando descobre-se um problema na rede, caracterizando um tipo de gerência totalmente reativa. Quando se faz uso somente de ferramentas simples, é utilizado a técnica da porta aberta, ou seja, reação a problemas que não podem ser previstos.

Lopes *et al*. (2003), consideram que esse tipo de gerência não possibilita a gerência de uma rede de grande porte. Em contrapartida, acreditam que seria muito mais rápido detectar os problemas com auxílio de uma estação de gerência onde o mapa da rede é apresentado e alarmes são gerados

automaticamente quando limiares ou mudanças de estado operacional são detectados.

2.5 Plataformas de Gerência de Rede

Normalmente, uma estação de gerência é formada por uma plataforma composta por diversas aplicações de gerência, comparado-a a um sistema operacional de gerência. A plataforma permite que aplicações individuais de gerência possam se "plugar" para formar uma solução de gerência completa, compondo a arquitetura apresentada na Figura 2.4. Pode-se dizer, portanto, que uma plataforma de gerência é uma solução incompleta de gerência (ou um *framework* de gerência) que as aplicações de gerência completam (Lopes *et al*., 2003).

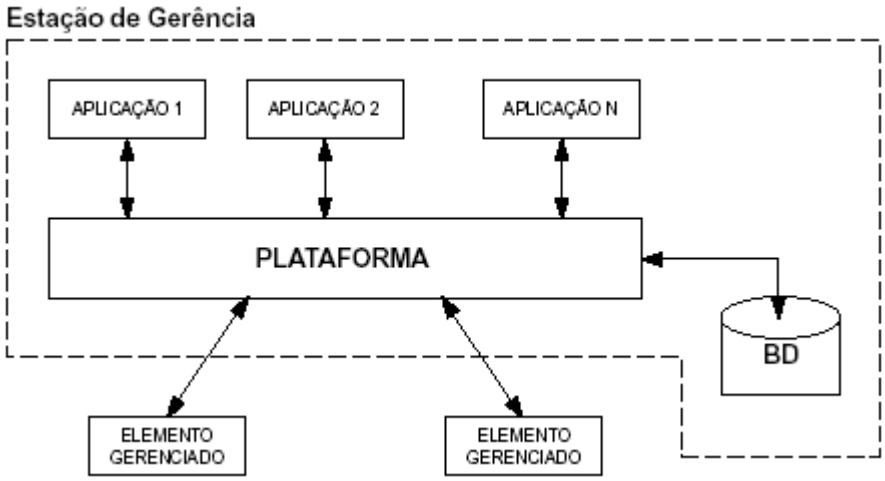

Figura 2.4 – Arquitetura geral de um Sistema de Gerência de Redes (Fonte: Lopes *et al*., 2003).

Lopes et al. (2003), comentam que as plataformas são projetadas para fornecer condições para construção de diversas aplicações de gerência usadas pelos operadores da rede. Nas Tabelas 2.2 e 2.3 são apresentadas algumas plataformas e aplicações de gerência que, segundo a visão dos autores, são consideradas populares.

Tabela 2.2 – Plataformas populares.

Plataforma de Gerência	Fabricante
OpenView	Hewlett Packard
Tivoli	IBM
Spectrum	Aprisma e CA (Unicenter da Computer Associates)

Fonte: Lopes *et al*. (2003).

Tabela 2.3 – Aplicações populares

Aplicações de Gerência	Proprietários
Netclarity	Lanquest
Alarm Manager	Aprisma e CA (Unicenter da Computer Associates)
AssetView	Hewlett Packard
CiscoWorks	Cisco

Fonte: Lopes *et al*. (2003).

Para Lopes *et al.* (2003), na maioria das vezes as soluções de gerência proprietárias ficam muito caras, podendo até não se adaptar às mudanças de tecnologia e requisitos de gerência dos usuários, surgindo assim, as plataformas de código aberto como o OpenNMS que é um projeto de código aberto dedicado à criação de uma plataforma de gerência de rede.

O OpenNMS trabalha com o protocolo de gerência SNMP, o que torna mais fácil a monitoração dos serviços oferecidos pela rede, podendo gerar relatórios de nível de serviço e notificação de problemas (Lopes *et al*., 2003).

2.5.1 Plataforma de Gerência de redes baseada em *Web*

A gerência de redes via *Web*, foi definida por Carvilhe (2000) como a utilização das tecnologias *Web* para implementação da gerência corporativa. Segundo o autor, a tecnologia baseada nesta arquitetura apresenta como principal característica o fácil acesso a qualquer ponto de rede corporativa ou da internet

a custos mais acessíveis, devido a utilização de interfaces baseadas em navegadores *Web*. Na Tabela 2.4 são descritos os componentes da arquitetura *Web*.

Tabela 2.4 – Arquitetura de gerenciamento *Web*

Componentes	Descrição
Cliente *Web*	É o computador de gerência corporativa que possue o navegador *web*
Servidor *Web*	Programa servidor responsável por atender as requisições realizadas por intermédio do navegador e retornar as respostas
Função de Mediação de Dados	Função responsável pela conversão dos dados de gerência do repositório de dados para um formato que seja reconhecido pelo servidor *Web* e pelas aplicações de gerência.
Repositório de Dados	Banco de dados onde são armazendas as informações de gerência
Função de Agrupamento e Mapeamento	Responsável pela conversão dos dados coletados pelos agentes em um formato padrão reconhecido pelo repositório de gerência
Gerentes de Protocolos	Lado gerente responsável pela gerência convencional e a *Web*
Agente de gerência	Corresponde à entidade de processo localizado no dispositvo gerenciado

Fonte: Job e Simões (2002).

Conforme Carvilhe (2000), o gerenciamente *web* apresenta uma série de benefícios, dentre eles o baixo custo de implementação em relação às soluções convencionais, a independência tanto de plataforma, quanto de localização do cliente de gerência, facilidade no uso, por se tratar de manuseio via *browsers* da internet, e o alto grau de interoperabilidade que pode ser alcançado usando linguagens como Java, *script* e demais tecnologias disponíveis adaptadas a quase todas plataformas de computadores.

O elemento central da arquitetura WBEM é o CIMOM que, segundo Job e Simões (2002), pode aceitar todos os *frameworks* de gerência de sistemas e de rede que existem. Ele faz a mediação dos dados, transformando as informações enviadas pelos objetos gerenciados em um formato padrão e armazena-os no CIM. Também tem a responsabilidade de fornecer a aplicação de gerenciamento WBEM com as informações solicitadas, conforme arquitetura apresentada na Figura 2.5.

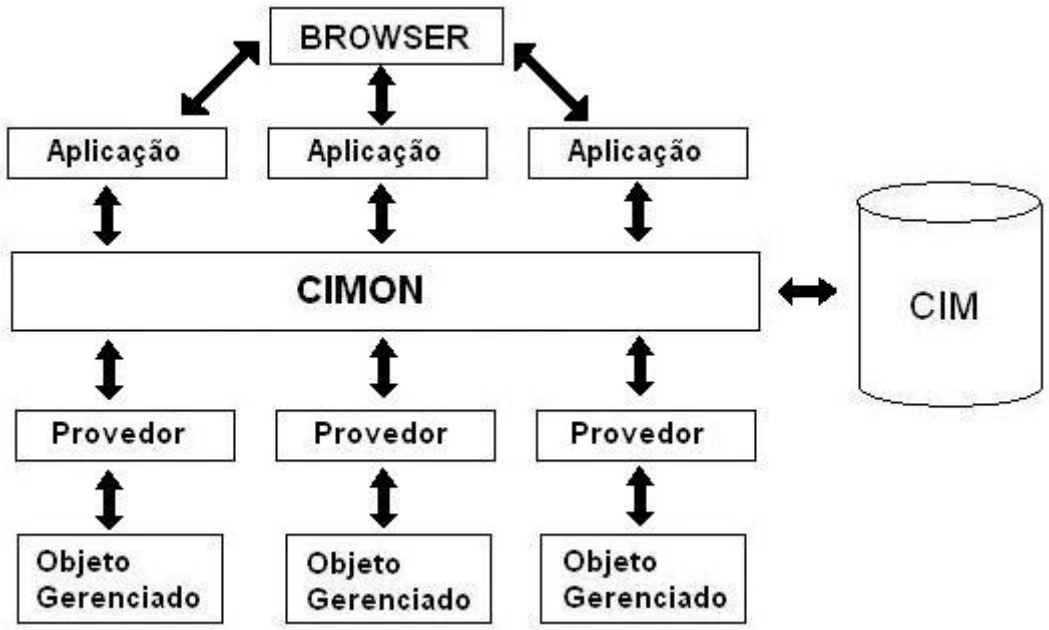

Figura 2.5 – Modelo da Arquitetura WBEM (Job e Simões, 2002).

Segundo Job e Simões (2002), somente os dados estáticos são armazenados no CIM enquanto os dinâmicos são consultados diretamente por meio do CIMON aos provedores.

2.5.2 Nagios

O Nagios (antigo *Netsaint*) é um aplicativo de código aberto cuja finalidade é a monitoração de *hosts*, de serviços e de rede. Tem a funcionalidade de informar os problemas de rede antes que seus clientes, usuários finais ou gerentes o façam. Embora tenha sido projetado para rodar em GNU/Linux, trabalha muito

bem com outras variantes *NIX. O *daemon* de monitoração roda checagens intermitentes nos *hosts* e serviços que são especificados usando *plugins* [8]externos que retornam informações de estado para o aplicativo. Uma vez detectado problemas, o *daemon* pode enviar notificações para contatos administrativos de várias formas diferentes (email, *Instant Messages*, SMS, etc.). Informações sobre o estado atual, histórico de *logs* e *reports* podem ser todos acessados através de um navegador *web* (Soares, 2004).

Segundo Martins e Lopes (2003), para utilizar o Nagios com interface *Web* será necessário configurar o arquivo *cgi.cfg* e também o arquivo do *Servidor Web Apache* (*httpd.conf*).

A figura 2.6 demonstra uma interface amigável por onde o administrador da rede, pode acompanhar o estado de uma determinada estação (Júnior, 2003).

State	Time	% Total Time	% Known Time
UP	7d 23h 26m 29s	53.18%	100%
DOWN	0d 0h 0m 0s	0.00%	0.00%
UNREACHABLE	0d 0h 0m 0s	0.00%	0.00%
Undetermined	7d 0h 33m 31s	46.62%	
All	15d 0h 0m 0s	100%	100%

Figura 2.6 – Status de disponibilidade em uma estação (Júnior, 2003).

Segundo Martins e Lopes (2003), o Nagios consiste em um sistema central que trabalha com *plugins*. O sistema central, escrito na linguagem C, invoca os *plugins*, escritos na linguagem C ou Perl, para fazerem a monitoração. Um exemplo disto pode ser observado na figura 2.7.

[8] Plugins são *script*s ou binários que fazem a verificação dos serviços e *hosts* da rede.

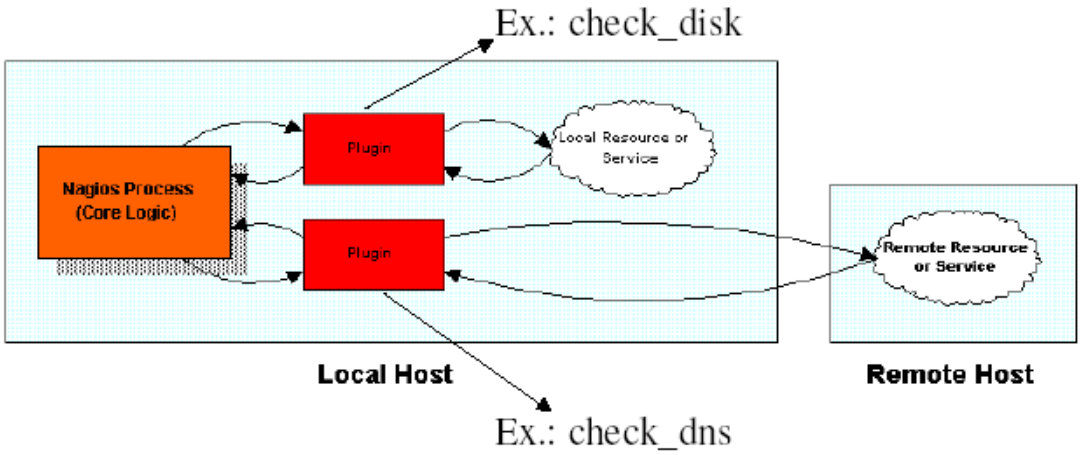

Figura 2.7 – Interação entre o Nagios e os plugins (Machado, 2004).

De acordo com Júnior (2003), há um arquivo de registro dos dados apresentados na figura 2.6, juntamente com outras informações apresentadas ao administrador da rede, como o gráfico de alerta, que demonstra o número de eventos ocorridos no período de um dia, confrontando-o com os limeares de advertência e crítico.

Segundo Martins e Lopes (2003), o Nagios disponibiliza alguns *plugins* básicos, entretanto, pode-se criar novos *plugins* de acordo com a necessidade de monitorar outros serviços e *hosts*. Os plugins que acompanham o Nagios tem ajuda disponível quando executados com a opção –h.

Em SOARES (2004), são citadas as características do Nagios:

- Monitoração de serviços de rede como: POP3, SMTP, HTTP, Ping, NNTP, e outros;
- Monitoração de recursos de *hosts* como: uso de disco, memória, processamento e outros;
- Definição de hierarquia de *hosts* de rede usando "*parent*" *hosts*;
- Notificação de contatos quando ocorrem problemas em *hosts* ou serviços;
- Uso de *event handlers* para serem rodados em eventos de *hosts* ou serviços
- Automação de rotação de *logs*

De acordo com SOARES (2004), o ambiente necessário para instalação do Nagios é:

- Sistema operacional do tipo Linux ou BSD e um compilador C;
- Software básico :
    - Nagios Core (Programa Central);
    - Apache *Web* Server;
    - Biblioteca gd 1.6.3 ou superior

Na tabela 2.5 pode-se observar a estrutura montada pelo Nagios:

Tabela 2.5 – Estrutura de diretórios do Nagios (SOARES, 2004).

Sub Diretório	Conteúdo
bin/	Programa Central do Nagios
etc/	Arquivos de configuração principal, resources, objects e CGIs
libexec/	Plugins
sbin/	CGIs
share/	Arquivos HTML e documentação (Interface *web*)
var/	Diretório inicialmente vazio para arquivos de log

Em Júnior (2003) é enfatizado uma das vantangens do Nagios com relação ao uso da rede. Segundo o autor, o Nagios somente captura dados, por esse motivo, não sobrecarrega a rede.

O Nagios também oferece suporte a banco de dados como MySQL e PostGreSQL, permitindo com isso, o armazenando de informações (Martins e Lopes, 2004).

Segundo Martins e Lopes (2003), a configuração do Nagios é uma tarefa que requer muita atenção, especialmente quando se trata de gerenciamento de redes complexas. De acordo com os autores, os arquivos de configuração podem ser divididos em cinco tipos:

1. Configuração do arquivo principal (nagios.cfg);
2. Arquivos de resources (resource.cfg);

3. Arquivos de configuração de objetos;
4. Arquivo de configuração de CGIs (cgi.cfg);
5. Arquivos de configuração para informação extendida.

De acordo com Martins e Lopes (2003), o Nagios é altamente customizável, o que o torna extramamente flexível a várias situações de monitoramento. O arquivo de configuração principal (nagios.cfg) é o primeiro arquivo a ser editado e configurado. Ele contém diretivas que afetam o modo de funcionamento do nagios. Também é lido pelo processo do Nagios e pelos CGIs.

Os arquivos de *resources* (resource.cfg) podem ser usado para definição de *macros*[9], além de contar com outras informações como (configuração de ligação de banco de dados). O propósito de se ter arquivos de *resources* é de guardar informações importes e não disponibilizar para os CGIs. Pode-se especificar um ou mais arquivos de *resources* usando a diretiva *resource_file* no arquivo de configuração principal (nagios.cfg) (Martins e Lopes, 2003).

Segundo Martins e Lopes (2003), os arquivos de configuração de objetos, por motivos históricos também são chamados de arquivos de configuração de *hosts*, são usados para definir *hosts*, serviços, grupos de *hosts*, contatos, grupos de contato, comandos e etc. Conforme os autores, são nestes arquivos que fica definido (o quê?) será monitorado e (como?) isso será feito. Os arquivos que contém definições de objetos são os seguintes:

- checkcommands.cfg: lista de comandos de verificação dos serviços previstos no arquivo services.cfg;
- contactgroups.cfg: contém grupos de contatos separados por área de serviço (HTTP, SMTP e outros);
- contacts.cfg: contém as informações que diz respeito aos contatos que poderão ser notificados na eventualidade de uma emergência.;
- dependencies.cfg: lista de dependência entre os *hosts* monitorados;

[9] Macros são variáveis usadas nas definições de comandos, as quais são substituídas pelos seus valores na altura que estes são executados, permitindo definir comandos genéricos que preencham a todo tipo de necessidades (Martins e Lopes (2003)).

- escalations.cfg;
- *host*groups.cfg: lista de grupos de *hosts* a serem monitorados;
- *hosts*.cfg: lista dos *hosts* que serão monitorados;
- misccommands.cfg;
- services.cfg: lista dos serviços a serem monitorados nos *hosts* listados no arquivo *hosts.cfg*;
- timeperiods.cfg: definição de período de notificações com relação aos problemas da rede.

2.5.2.1 Nagios usando fax/modem

Segundo Cassaro (2005), uma das vantagens grandes do Nagios é sua flexibilidade em trabalhar em conjunto com outras aplicações, como é o caso da aplicação *mgetty* que permite enviar mensagens de áudio via linha telefônica. Cassaro afirma que o tipo de mensagem enviada não é como torpedos SMS e sim como secretária eletrônica. Para isso, são necessários alguns pré-requisitos:

- Nagios devidamente instalado e configurado;
- Software Mgetty (somente a parte de *voice* do pacote);
- Um *Voice Modem*, onde o autor aconselha usar um *modem* antigo, configurável via *jumpers* ou, de preferência, um *modem* externo.

De acordo com Cassaro (2005), existe a dependência dos serviços de envio de mensagens SMS e email com a internet. Para contextualizar melhor este cenário, o autor levantou os seguintes questionamentos:

- **Primeiro Problema**: Se o servidor de e-mail falhar?
    - Fica-se sabendo quando tentar acessá-lo e não conseguir?;
    - Fica-se sabendo quando acessar a página do Nagios?;
    - Configura-se o Nagios para emitir SMS?
- **Segundo Problema**: Se o roteador (com a Internet) falhar?
    - Fica-se sabendo quando tentar acessar a Internet e não conseguir?;

- Fica-se sabendo quando acessar a página do Nagios?;
- Configura-se o Nagios para emitir SMS? COMO?

A motivação em usar um serviço que fosse independente da internet, levou Cassaro (2005) a configurar o Nagios em conjunto com o aplicativo *mgetty*, com a finalidade de ser informado dos problemas da rede, mesmo que o roteador de acesso a internet falhasse, ou seja, recebendo um telefonema do seu servidor.

Segundo Cassaro (2005), o Nagios realiza a verificação dos serviços e caso haja algum problema, emite os avisos que estão configurados nos arquivos (contacts.cfg, misccommands.cfg e services.cfg).

Diante dos problemas levantados, Cassaro (2005) afirma que a dependência de notificações via internet pode ser um problema para os administradores de rede, caso haja uma falha física no dispositivo responsável pelo serviço. O autor ainda comenta o problema de envio do SMS, onde existe a dependência tanto do provedor de internet, quanto das provedoras do serviço, podendo as mensagens sofrer um certo atraso nas entregas, prejudicando, assim, uma tomada de decisão mais rápida por parte dos administradores.

2.6 A Informática na Saúde

Somente nos Estados Unidos, 8 milhões de pessoas precisam de cuidados médicos em casa, estima-se que a telemedicina já responda por mais de 100.000 consultas a distância por ano. Os pacientes usam equipamentos que medem a pressão arterial, os batimentos cardíacos e a taxa de açúcar no sangue e captam outras informações que podem ser monitoradas a distância pelos médicos. Os aparelhos transmitem dados por sinais de rádio ou pela internet e custam cerca de 100 dólares a unidade (Reportagem da revista Veja, 30/5/2001, pg 77 a 84).

2.6.1 Home Care

Como alternativa ao atendimento da saúde, surgiu o conceito de *Home Care*, que significa atendimento ambulatorial ou internação domiciliar por pessoal de enfermagem especializada, entenda-se por internação domiciliar todo e qualquer tratamento multidisciplinar especializado que requeira atendimento 24 horas de equipe médica-enfermagem na casa do paciente (Falcão, 1999).

Segundo (Falcão, 1999), o movimento de *Home Care* surgiu nos Estados Unidos em 1947 na era do pós-guerra. Foi quando várias enfermeiras se reuniram e passaram a atender e cuidar dos pacientes em casa. Somente na década de 1960 é que este movimento tomou mais vulto e a idéia de "desospitallização precoce" começou a ser levada a sério devido a lotação de hospitais e a leitos insuficientes, ocasionados pelo aumento da população e da longevidade de idosos precisando cada vez mais de cuidados médicos.

Ao contrário do que os médicos imaginavam, em vez de queda, houve um salto de eficiência com este tipo de tratamento, promovendo-se uma recuperação precoce do paciente aliada a uma drástica redução nos custo que variavam entre 20 e 70% mais barato do que os cobrados pelos hospitais, de acordo com a enfermidade. Além da baixa de custos, na Europa, estatísticas demonstraram um aumento no número de leitos hospitalares em torno de 30 a 40%, permitindo uma maior rotatividade nos leitos e abrindo espaço para pacientes instáveis que precisam de UTI´s, cirurgias ou tratamento de enfermidades agudas (Falcão, 1999).

No Brasil não é novidade que o sistema de saúde é arcaico e funciona precariamente e, por outro lado, pouco se investe em medicina preventiva. Em 2004 houveram 17,3 milhões de pacientes internados, dos quais 65% pelo SUS. Em função disso, o *Home Care* surge como uma importante alternativa no tratamento de pacientes pela significativa otimização do binômio custo/benefício (Falcão, 1999).

De acordo com Falcão (1999), qualquer tipo de enfermidade pode ser tratado pela internação domiciliar, desde que o quadro do paciente seja estável. No entanto, a maioria dos pacientes em *Home Care* são idosos com mais de 65 anos, portadores de doença de Alzenheimer ou esclerose múltipla, cujo acompanhamento está baseado em prontuários médicos, com os respectivos relatórios da enfermagem (ou de outros profissionais envolvidos no caso), que ficam na casa do paciente a disposição do médico.

Na medida em que a sofisticação dos aparelhos de telemedicina se desenvolvem, será possível que um maior número de pacientes possam ser monitorados à distância, abrindo possibilidade de outras doenças serem tratadas com o *Home Care*. Segundo previsão de Falcão (1999), a telemedicina será futuramente uma importante ferramenta no monitoramento de idosos que moram sozinhos e uma excelente alternativa para cidades do interior do país onde existe grave carência de especialistas.

2.6.2 Telemedicina

Atualmente, a área médica vem sendo profundamente afetada pelas mudanças do mundo. Particularmente, os recentes avanços nas tecnologias de comunicações que, estimularam o desenvolvimento e a apresentação de projetos de telemedicina (Hu *et al.*,1997[10], *apud* Christ *et al*., 2004).

A telemedicina começou com a NASA há mais de duas décadas. Com a generalização de redes de comunicação, levou-se a integração de sistemas medicos em redes de dados sob a supervisão de um sistema central, seguindo uma arquitetura cliente/servidor (Pizarro *et al*. ,2001).

Segundo Lin (1999), a definição do termo telemedicina remete à utilização das telecomunicações nos diagnósticos médicos e no atendimento ao paciente.

[10] **HU**, B.*et al*. **A Communication Server for Telemedicine Applications. IEEE Transactions on Information Technology**. v.1, n.3, p. 295-209.

Pode também ser descrita como o transporte de informações médicas digitais entre duas locações, complementa Hayes (1996).

O uso dos serviços de internet/intranet pelos profissionais da área de saúde, bem como seus pacientes, vem aumentando significativamente nos últimos anos (Siau, 2003). Os protocolos TCP/IP e UDP, são comumente usados no desenvolvimento de aplicações para internet/intranet, de forma que, representam ferramentas conhecidas, altamente testadas e qualificadas para a transmissão de informações. Sinais vitais como eletrocardiograma (ECG), a freqüência cardíaca e a temperatura corpórea, entre outros, também podem ser divididos em pacotes e transmitidos (Chris *et al*., 2004).

Dentro deste contexto, Pizarro *et al*. (2001), desenvolveram um projeto com o objetivo de criar uma solução distribuída para o monitoramento de sinais bioelétricos de pacientes, o controle remoto dos equipamentos de aquisição e visualização dos dados pela internet.

Para tanto, cada paciente localizado no hospital carrega consigo um módulo portátil, composto por um computador pessoal com uma interface de rede *wireless*[11]*,* um hardware para aquisição de sinais e uma bateria. Cada módulo recebe um endereço IP privado e interage com a central remota, cuja comunicação em rede torna-se possível pelo uso do protocolo TCP/IP.

No trabalho desenvolvido por Pizarro *et al.* (2001), um protocolo de rede foi especificado e implementado para transmissão em tempo real de sinais bioelétricos do paciente (os dados são lidos do periférico na taxa de transferência de 200 Kbps) e para configuração remota de parâmetros de aquisição, através de uma rede de computadores, tornando possível implantar uma arquitetura cliente/servidor. O projeto de telemedicina é dividido em três subsistemas:

- Subsistema de aquisição (Cliente)

[11] Tecnologia que dispensa o uso cabos para transmissão de dados.

- Subsistema central (Servidor)
- Subsistema de visualização (Cliente)

O subsistema central é um servidor de transferência de mensagens de múltiplos usuários, utilizando um protocolo de aplicação específico. O subsistema de aquisição envia os dados (sinais bioelétricos digitalizados) para o subsistema central que se encarrega de fornecê-los para o(s) subsistema(s) de visualizar correto(s). O subsistema de visualização por sua vez, permite que vários médicos possam monitorar um mesmo paciente, visualizando em tempo real mais de um sinal bioelétrico ao mesmo tempo (via navegadores da *web*) e, possibilita a configuração dos parâmetros de aquisição e do software para armazenar os dados em arquivos para análise posterior (Pizarro et al, 2001).

2.6.3 O Projeto MonitorSV

A universidade Federal de Pelotas (UFPel) em parceria com, a Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS), a empresa PARKS e a Fundação de Amparo a Pesquisa do Rio Grande do Sul (FAPERGS), desenvolveram um protótipo de monitor capaz de realizar monitoramento remoto, não invasivo, dos sinais vitais em pacientes, intitulado como “Projeto MonitorSV”.

Segundo os idealizadores do projeto, mesmo com inúmeros dispositivos para monitoramento de sinais vitais, acreditavam que, alguns pacientes, dentro e fora dos hospitais, necessitariam de dispositivos portáteis que pudessem fornecer um monitoramento de seus sinais vitais, sugerindo com isso, dispositivos com parâmetros programáveis, de fácil usabilidade e leitura, versátil para agregar novas funções, com tamanho e peso reduzidos e, principalmente, que fossem altamente confiáveis. O projeto contempla:

- Aquisição e conversão analógico-digital do sinais monitorados;
- Processamento dos sinais;
- Verificação da posição dos sinais dentro dos limites preestabelecidos pelo médico;

- Comparação com as zonas de alvo;
- Sinalização visual e auditiva.

O projeto MonitorSV é baseado em microcontroladores da empresa Microchip e conta com um canal de comunicação com um microcomputador pessoal, onde o médico passa os parâmetros de cada sinal monitorado, de acordo com a situação individual de cada paciente. O sinais capturados são interpretados por um módulo de controle que confere se o sinal está dentro dos limites delimitados pelo médico.

Este projeto teve como coordenadores os professores: Fabian Vargas e Marcello Macarthy, com a participação de Elton Spode, Djones Lettnin, Leonardo Ribeiro, Sérgio Muller e Letícia Bolzani.

2.6.4 Usando iPAQ no gerenciamento de pacientes e UTI´s

Segundo publicação no site do winCEBr@sil, os parceiros, Compaq, Microsoft, Incor – Instituto do Coração de São Paulo e AMR, promoveram a apresentação de uma solução pioneira, utilizando a tecnologia iPAQ Pocket PC no monitoramento de pacientes em unidades de terapia intensiva (UTI).

De acordo com o publicado, o novo sistema é capaz de receber os sinais vitais dos pacientes que estão tanto na UTI, quanto no centro cirúrgico, via handhelds iPAQ [12]Pocket PC, informando em tempo real, as informações, aos médicos que podem atender as necessidades dos pacientes em questão de minutos, graças à essa tecnologia, que permite conexões sem fio.

2.6.5 Sistema de Monitoração Remota de Pacientes em Tempo-Real Através da intranet do Hospital

Com chegada de novos equipamentos eletromédicos (o monitor de pacientes), implicou no aumento da aquisição de sinais vitais, que são informações médicas. Entretanto, esses equipamentos não foram, numa visão geral, projetados procurando visar sua utilização em sistemas de monitoração

[12] Computadores de mão conhecidos também como handhelds e PDA´s (Assistente Pessoal Digital).

remota, ficando sua capacidade remota resumida a interface RS232. Observa-se também que, não existe, na prática, um padrão de comunicação, ficando a implementação do protocolo por conta de cada fabricante. Mesmo com todos obstáculos, a utilização de equipamentos eletromédicos comerciais representa um grande auxílio no projeto de sistemas de telemedicina (Christ *el al*. , 2004).

Dessa forma, Christ *et al.* (2004), propuseram um sistema de telemedicina que utiliza um monitor de pacientes comercial e protocolo UDP para permitir o monitoramento remoto de pacientes, em tempo real, através da intranet de um hospital. A aplicação implementa uma arquitetura cliente/servidor, conforme pode-se acompanhar na figura 2.8.

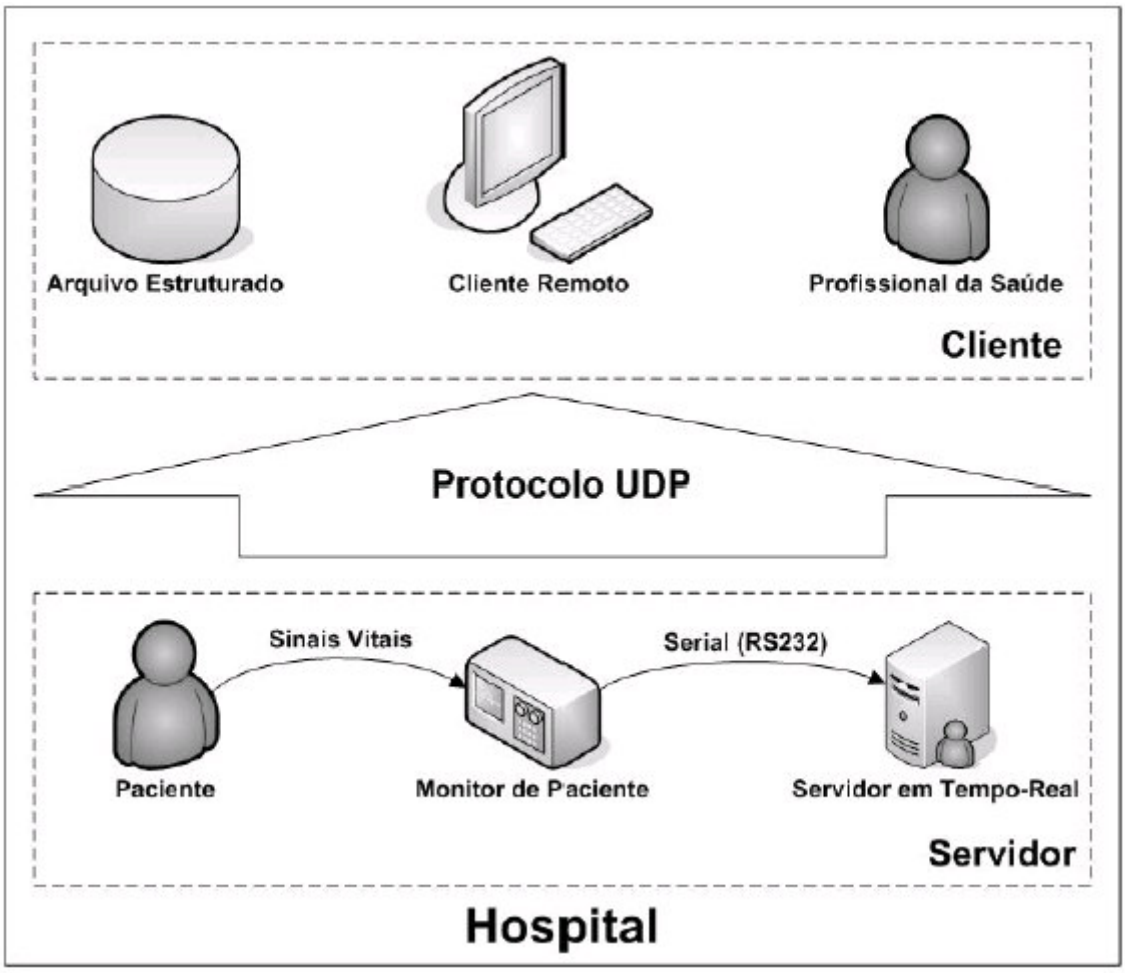

Figura 2.8 – Diagrama esquemático do sistema (Christ *el al.*, 2004).

Segundo Christ et al. (2004), o sistema funciona com o equipamento médico se comunicando com um servidor de sua interface serial e este trata a informação, disponibilizando-a para um cliente remoto. O lado cliente tem a aplicação que permite a visualização dos sinais vitais transmitidos, permitindo também que estes sejam salvos em arquivos estruturados.

De acordo com a metodologia usado por Christ *et al.* (2004), o funcionamento do servidor fica assim descrita:

- Aquisição dos sinais: primeiramente é realizada pelos monitores que adquirem e analisam e, por intermédio da porta de comunicação serial, são transmitidos para o servidor.
- Servidor: responsável em receber os sinais vitais enviados pelo monitor de pacientes e transmiti-los ao cliente via protocolo UDP. Seu funcionamento é apresentado na figura 2.8. Uma vez que, tanto a aquisição quanto a transmissão, são processos concorrentes, resolveu-se fazer uso de *thread*s para cada tarefa.
    - A *Thread* I fica responsável em gerenciar a comunicação serial entre o servidor e o monitor de pacientes. Ela configura o monitor de pacientes sobre quais sinais vitais deverão ser disponbilizados e a velocidade de transmissão, gerando uma fila circular para cada nova informação lida na porta serial RS232. Lembrando que, a fila circular gerada implementa o algoritmo de exclusão mútua para evitar a sobrescrita de dados.
    - A *Thread* II, por sua vez, fica responsável em controlar a comunicação com o cliente. Antes mesmo de iniciar a transmissão, ela abre um *socket* no cliente que abre uma conexão para efetuar a leitura da fila circular gerada pela *thread* I e, gera um pacote UDP para que este possa ser transmitido ao cliente.
- Cliente Remoto: Fica responsável pela recepção e tradução dos pacotes UDP.

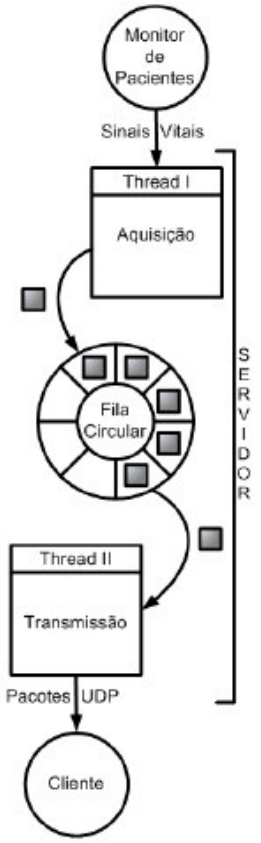

Figura 2.9 – Funcionamento do Servidor (Christ et al., 2004).

2.7 O Micro Servidor *Web* (MSW)

O Micro Servidor *Web* (MSW), segundo Silva (2002), é um projeto nascido de pesquisas feitas dentro do RexLab Laboratório de Experimentação Remota da UFSC. Silva (2002), resume a função do MSW em uma tecnologia capaz de conectar dispositivos elétricos a uma rede padrão Ethernet, baseado em um microcontrolador de baixo custo, que utiliza para a comunicação de dados o protocolo TCP/IP, permitindo adquirir, controlar e monitorar remotamente estes dispositivos de maneira segura, eficiente e econômica, mediante o uso de um navegador padrão para Internet. Na figura 2.10 é demonstrado o projeto finalizado do MSW.

Tamanho(85mm x 95mm)

Figura 2.10 – Micro Servidor *Web* desenvolvido no RexLab (Silva, 2002).

Segundo Silva (2002), a possibilidade de conexão a *web*, embarcadas em equipamentos convencionais, traz vantagens no monitoramento e controle com custos relativamente baixos, utilizando uma interface familiar para os usuários e com um alto grau de padronização.

Para Silva (2002), um servidor *Web* embarcado poderá conter páginas *web* dinâmicas que podem controlar as comunicações com outros dispositivos utilizando os protocolos TCP/IP. De acordo com o autor, esta metodologia provê uma padronização na comunicação entre os dispositivos por meio do uso de páginas *web* simples, de fácil instalação e manutenção, dispondo ainda, de uma conhecida Graphical Universal Interface (GUI). O browser poderá descarregar processamento para um controlador embarcado, ou habilitar o servidor *web* para controle de dados via Java*Script* e Java.

Segundo Benhardt (2003), embora o MSW, aparentemente, demonstre um dispositivo de arquitetura simplificada, com finalidade de conectar-se a outros dispositivos e prover o acesso via *web*, sua linha aplicabilidade torna-se bastante flexível, possibilitando seu uso em ambientes domésticos, industriais, médicos e outros, conforme pode-se observar pela Figura 2.11.

Figura 2.11 – Aplicações para o Micro Servidores *Web* (Silva, 2002).

2.7.1 Composição Física do MSW

Segundo Silva (2002), o MSW é composto por quatro circuitos integrados: o microcontrolador AVR AT90S8515, o RTL8019AS, o MAX232 e a mémoria EEPROM 24LC515. Sua composição também conta com alguns componentes discretos, conforme pode-se observar na Figura 2.12.

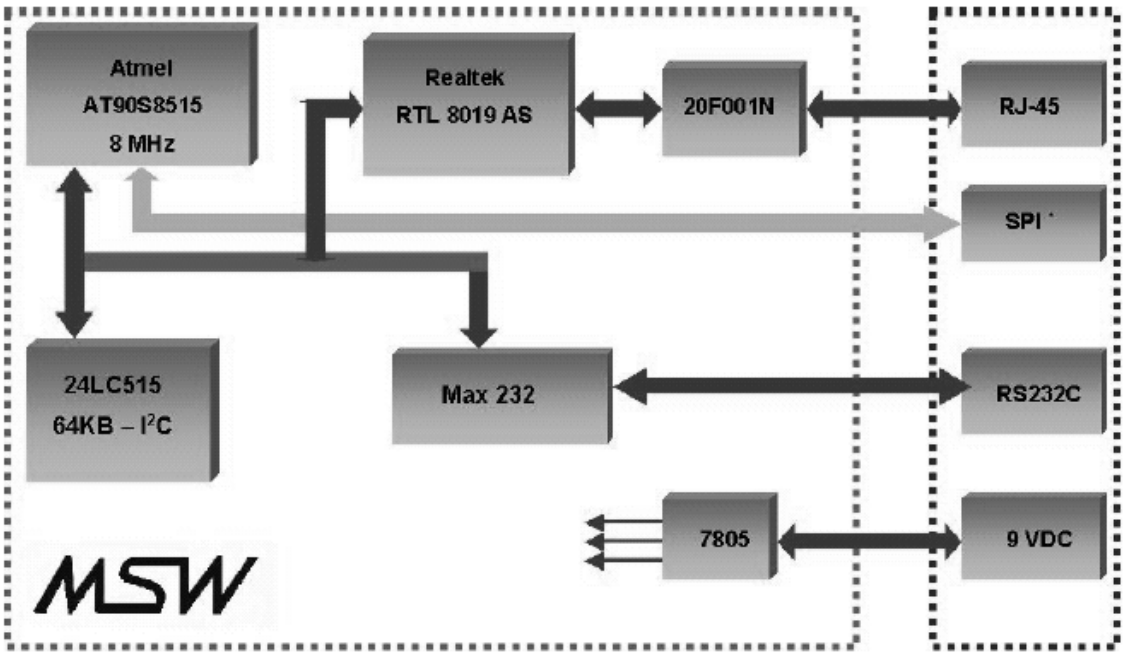

Figura 2.12 – Diagrama de blocos do MSW (Silva , 2002).

O microcontrolador AVR AT90S8515, segundo Silva (2002), dispõe internamente de 3 tipos de memória: RAM estática (512 bytes), memória flash programável (8 Kbytes), e EEPROM "*on-chip*" (512 bytes), em sua arquitetura AVR RISC (*Reduced Instructions Set Component*). Para o armazenamento do código das aplicações e imagens é utilizada a memória I^2C EEPROM Serial, modelo 24LC515 fabricada pela Microchip. Para o interfaceamento Ethernet é utilizado o circuito integrado RTL8019AS, com 16 Kbytes de SRAM integrados, modulador e demulador para a interface física, controlador de protocolo Ethernet, dispondo de todos os requisitos necessários para transmissão e recepção de pacotes Ethernet. O último circuito a ser comentado é o MAX 232 que é um *driver/tranceiver* para comunicação RS232S fabricado por diversas empresas do mercado (Silva, 2002).

2.7.2 Software Básico do MSW

O *firmware* do MSW é composto, segundo Silva (2002), por um *kernel* (núcleo) legeiramente simples e sua organização pode ser vista na Figura 2.13. O autor comenta que o controlador ethernet proporciona suporte para as duas camadas mais baixas de controle, por conseguinte o microcontrolador Atmel cuida de todas as que ficam acima. Em seqüência, as duas próximas camadas contêm o driver do adaptador de rede e a pilha TCP/IP. A camada de aplicação é o servidor *web* http.

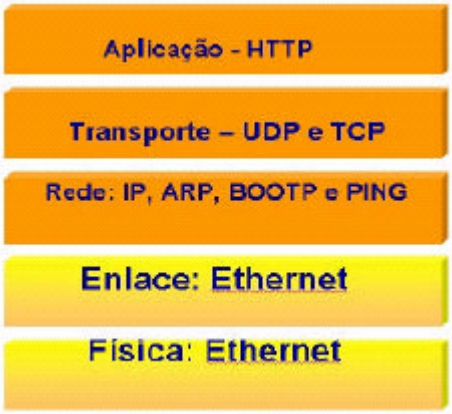

Figura 2.13 – Camadas do *Firmware* do MSW (Silva, 2002).

O *firmware* conta com um *debugger* simples favorecendo, *dumps* de memória, alterações da EEPROM e pcode. Existem rotinas de *firmware* que permitem que a EEPROM I^2C de 64 KB, seja programada remotamente via interface Ethernet. Uma utilidade do programa é prover que dados, imagens gráficas, HTML, Java *applets* e rotinas pcode sejam carregados na memória EEPROM I^2C. Os segmentos desta, são então transferidos através da rede utilizando um programa TCPbased (baseado no protocolo TCP). Esta característica também permite a imagens e rotinas pcode serem atualizadas mesmo quando o MSW está ativo (Silva, 2002).

Assim sendo, Silva (2002) coloca que o *firmware* padrão do MSW apresenta um kernel simples, um *debug* minúsculo, um interpretador para pseudocódigo (pcode), driver adaptador de rede, pilha TCP/IP e servidor HTTP. Uma vez conectado à rede, o *firmware* do MSW suporta uma série de protocolos de rede como o ARP, o BOOTP, o ICMP, o UDP e o TCP/IP que permite o MSW responder a solicitações HTTP GET, endereçadas a porta TCP80, conforme pode ser visualizado na Figura 2.13.

O *firmware* do MSW, segundo Silva (2002), ainda, restringido a uma resposta HTTP GET, a um tamanho aproximado de 1400 bytes para um único pacote Ethernet, podendo assim, conservar recursos de memória. De acordo com o autor, diversas técnicas de codificação HTML podem ser utilizadas para que se possa trabalhar dentro desses limites, incluindo frames HTML e múltiplas imagens GIF e JPEG. A resposta básica do *firmware* para uma solicitação HTTP GET é o retorno de uma página *web*, afirma o autor.

De acordo com Silva, 2002, o MSW responde, em um nível mais baixo, a solicitações de rede ARP (Address Resolution Protocol), que permite que outros computadores possam realizar uma associação entre o endereço IP determinado para o MSW e seu endereço *Ethernet*. Para todo MicroServidor é fixado um único endereço Ethernet que é enviado como parte do pacote ARP replicado.

Segundo Silva (2002), o *firmware* do MSW é escrito fazendo um misto de linguagem assembly AVR padrão, que é um tipo de linguagem assembly interpretada por uma máquina virtual 16-bit, com a utilização de um número predefinido de macros da linguagem assembly AVR por parte do *firmware* do MSW. Essas macros são apresentados na Tabela 2.6, sendo as mesmas utilizadas para permitir a manipulação conveniente dos dados.

Tabela 2.6 – Macros pré-definidas pelo AVR Assembly.

Macros	Operando	Descrição	Operação	Flags
Addw	Wd, Wr	Adiciona palavras sem Carry	Wd ← Wd + Wr	Z, C, N, V, H
Addwi	Wd, K	Adiciona imediato à palavra	Wd ← Wd + K	Z, C, N, V
andwi	Wd, K	Logical AND Word with Immediate	Wd ← Wd - K	Z, N, V
clrw	wd	Limpa Palavra	Wd ← 0	Nenhum
cmpw	Wd, Wr	Compara Palavras sem Carry	Wd - Wr	Z, C, N, V, H
cmpwi	Wd, K	Compara Palavra com Imediato	Wd - K	Z, C, N, V, H
decw	wd	Decrementa Palavra	Wd ← Wd - 1	Z, C, N, V, H
incw	wd	Incrementa Palavra	Wd ← Wd + 1	Z, C, N, V, H
ldsbw	wr	Carrega Byte a partir da SRAM na Palavra	Wd ← (k)	Nenhum
ldsw	Wd, k	Carrega Palavra Direto da SRAM	Wd ← (k+1,k)	Nenhum
movw	Wd, Wr	Move Palavras	Wd ← Wr	Nenhum
Movwi	Wd, K	Move Immediate to Word	Wd ← K	Nenhum
Popw	Wd	Desempilha Palavra	Wd ← STACK	Nenhum
Pushw	Wr	Empilha Palavra	STACK ← Wr	Nenhum
Shlw	Wd	Troca lógica da palavra à esquerda	Wd ← Wd << 1	Z, C, N, V, H
Shrw	Wd	Troca lógica da palavra à direita	Wd ← Wd >> 1	Z, C, N, V, H
Stsw	Wd, k	Armazena Palavra Direto na SRAM	(k+1,k) ← Wd	Nenhum
Subw	Wd, Wr	Subtrai Palavras sem Carry	Wd ← Wd - Wr	Z, C, N, V, H
subwi	Wd, K	Subtrai Imediato da Palavra	Wd ← Wd - K	Z, C, N, V

Fonte:Silva (2002).

O arquivo de projeto do MSW fornece as informações necessárias para a implementação, como o *firmware* e as páginas *web* que contêm a apresentação das aplicações. As definições do pré-processador, existentes no arquivo de projeto, são utilizadas para assinalar alguns parâmetros existentes, na construção de processos do *firmware* do MSW. As definições consideradas padrão do pré-processador são demonstradas na Tabela 2.7 (Silva,.2002).

Tabela 2.7 – Definições do Pré-Processador

Definição do Pré-processador	Valor Default	Descrição
BAUD_RATE	19200	Define a taxa de transmissão da porta serial. É importante observar que nem todas as taxas de transmissão serão possíveis, elas dependem da freqüência de clock do microcontrolador. A diretiva #define CLOCK deverá ser corretamente setada
CLOCK	7372000	Define a freqüência de clock do microcontrolador Atmel (Hz). É também utilizada para definir a taxa de transmissão da UART e para outros cálculos
DEBUGGER	Não definido	Se definido, inclui o firmware debugger da porta serial como parte da construção de processos do projeto
EEPROM_IP	Não definido	Se definido, especifica que o endereço IP do MSW está armazenado na EEPROM on-chip (o endereço será proveniente do arquivo texto ip)
ENABLE_WATCHDOG	Não definido	
NET_CONFIG_IP	Não definido	Definido para permitir que o endereço IP do MSW seja modificado através da Ethernet utilizando o programa set ip
NO_ETHER_ADDR_ON_BOOT	Não definido	Definido para inibir impressão do endereço Ethernet do MSW para a porta serial
PKT_WATCHDOG_MAX	250	Pacote de rede do Watchdog Timer limitado pelo slow idle loop (alcance recomendado: 100-254)
SEEPROM_IP	Não definido	Definição do endereço EEPROM serial que possui o endereço IP do MSW. Define, se desejado, o endereço IP armazenado em uma memória EEPROM serial
STATIC_IP_LSB STATIC_IP_MSB	Não definido	Definição de duas constantes de 16-bit, no caso de se desejar que o endereço IP do MSW seja codificado dentro da memória de programa (define MSBs e LSBs do endereço IP 32-bit)
USE_BOOTP	Não definido	Utiliza o protocolo BOOTP para obter o endereço IP do MSW

Fonte: Silva (2002).

De acordo com Silva (2002), as declarações apresentadas na Tabela 2.7 são indicadas por “#define” e ficam localizadas em um arquivo que leva o mesmo nome do arquivo de projeto, seguido da extensão “.h”. A Figura 2.14 demonstra um exemplo.

```
#define EEPROM_IP              /*  utiliza  arquivo  "ip"  para
                              configuração do endereço IP */
#define NET_CONFIG_IP          /*  permite  reconfiguração  do
                              endereço IP via net */
#define ENABLE_WATCHDOG /* utiliza o WatchDog Timer Atmel */
#define DEBUGGER               /* inclui o firmware debugger*/
#define CLOCK 7372000          /*  pulso  de  clock  do
                              microcontrolador (em HZ) */
#define BAUD_RATE 19200        /*  taxa  de  transmissão  da  porta
                              serial */
```

Figura 2.14 – Exemplo de declarações no arquivo de projeto (Silva, 2002).

Segundo Silva (2002), o arquivo de projeto apresenta uma lista das páginas *web* que estarão inclusas no projeto, significando que existem arquivos que contêm código HTML e/ou imagens, que serão armazenados na memória SEEPROM do MSW como parte integrante de seu sistema de arquivos. Conforme o autor, as rotinas CGI são apresentadas em uma lista, fornecida pelo usuário e que serão utilizadas pelas páginas *web* do MSW. Ainda são listados os labels dos ponteiros de entrada dedicados às rotinas pcode que, normalmente, são fornecidas nas seções de código seguintes do arquivo de projeto. O autor ainda lembra, que a extensão ".cgi" tem que ser anexada a cada label listado.

```
//
 // rotinas CGI de aplicações específicas
 //
 temperatura.cgi   //(retorna a temperatura ASCII (graus C))
```

Figura 2.15 – Exemplo de uma rotina pcode CGI (Silva, 2002).

De acordo com a Figura 2.15, o nome desta rotina CGI é "temperatura.cgi", ela reside no *firmware* do MSW. Uma vez listada dentro do arquivo de projeto, esta rotina pode ser acessada externamente em um *browser* de internet com uma solicitação HTTP GET, como mostra o exemplo:

http://x.y.z.w/temperatura.cgi

onde as letras x, y, z e w, representam o número do endereço ip do MSW.

2.7.3 Programando Aplicações no MSW

De acordo com Silva (2002), a grande questão que envolve a implementação do MSW é basicamente compreender como todo *firmware* irá se ajustar ao tamanho reduzido de 8 Kb de memória *flash* do microcontrolador. Diante disso, o autor propõe que seja feito o uso de uma técnica de pseudocódigo [13](pcode) para conservar o espaço de código em troca de uma velocidade de execução um pouco reduzida.

Dessa forma, Silva (2002) relata que o interpretador de instruções pcode, desenvolvido pela Lightner , favorece a simplificação do código e a opção de executá-lo fora da EEPROM (incluindo EEPROM serial externa, e em alguns casos, uma redução no tamanho do código do programa quando comparado ao código nativo).

Conforme Silva (2002), o código fica simplificado devido ao pcode não fazer referências aos registros nativos, e porque a "máquina virtual" pcode utiliza tipos de dados "16-bitwide" para a maioria dos operandos. O autor afirma, ainda, que a execução do pcode a partir da EEPROM também é possível, devido ao ponteiro da instrução pcode ser de 16 bits, com a indicação do bit mais alto (quando ativado), a próxima instrução pcode buscaria a partir da EEPROM e não a partir dos 8 KB da memória flash programável do microcontrolador Atmel. A redução do tamanho do programa resulta da combinação de fatores, incluindo aplicações específicas e eficientes de rotinas pcode, e modos de endereçamento do operando pcode flexíveis, afirma o autor.

Silva (2002) relata que para a construção de uma aplicação, o ambiente de desenvolvimento pode ser baseado tanto em Linux, quanto *Windows*. Entretanto, segundo o autor, o projeto MSW procurou sempre trabalhar com arquitetura aberta, baseando-se no uso de aplicativos de código aberto.

[13] Pseudocode –São palavras pré-definidas na sua maioria adaptadas de uma linguagem de programação estruturada. (Bernhardt & Paladini, 2001).

Embora se faça uso do Assembler Atmel padrão para a geração do código do *firmware*, afirma Silva (2002) que, primeiramente, o código fonte, deve passar por um processador C e um *script* Perl antes de ser enviado para o assembler, ficando o procedimento de construção controlado por um arquivo *batch*[14] simples.

Assim, uma vez definido o arquivo de projeto e compilados todos os arquivos necessários ao funcionamento do microservidor, este pode ser acessado por meio da rede como um servidor *web* tradicional, uma vez que as URLs, direcionadas à porta TCP80, são respondidas da maneira convencional, retornando a página HTML e as imagens através das solicitações HTTP GET.

2.7.4 Registradores do AVR AT90S8515

No AT90S8515 da ATMEL, assim como nos demais sistemas microprocessados, uma característica básica é a presença de um grupo de registradores internos. A arquitetura AVR presente neste modelo é composta por 32 registradores de 8 bits, podendo ser manipulados tanto para leitura, quanto para escrita. Além disso, seus três registradores mais altos (X, Y, e Z), podem ser utilizados no endereçamento direto da memória. Há também os registradores de I/O (*Input/Output*) que são em número de 64 e podem ser endereçados diretamente em instruções de apenas um ciclo de *clock* (Silva, 2002). No Anexo 1 é demonstrado, por meio da Tabela 2.8, os registradores de I/O do modelo AT90S8515.

Segundo Silva (2002), todas as entradas do AT90S8515 e periféricos estão localizados no espaço de I/O. As alocações são acessadas por meio das instruções de IN e OUT, que transfere dados entre os 32 registradores de uso geral e o espaço de I/O. De acordo com o autor, os registradores de I/O dentro do espaço de endereço $00 - $1F, são diretamente acessados utilizando as

[14] Arquivo texto com extensão ".bat", contendo um ou mais comandos a serem executados por meio de uma chamada externa a ele.

instruções SBI e CBI, sendo que nesses registradores, os valores de bits únicos podem ser verificados pela utilização das instruções SBIS e SBIC. Para os comandos IN e OUT , de I/O, os endereços de $00 - $3F devem ser utilizados. Quando do endereçamento dos registradores de I/O como SRAM, o valor $20 deve ser adicionado a este endereço (Silva, 2002).

O microntrolador AT90S8515 conta a arquitetura AVR RISC, suportanto com isso diferentes modos de endereçamentos, que segundo Silva (2002), são muito eficientes para que o microcontrolador possa acessar sua memória de programas. (*flash*) e a memória de dados (SRAM, Register File e memória de I/O).

2.7.5 As Interrupções do AVR AT90S8515

Segundo Silva (2002), as interrupções são mudanças ocorridas no fluxo de controle, cuja causa não se relaciona à execução de um programa, mas geralmente, nas operações de I/O. O autor expõe que existem três fontes de interrupção: por software (instrução), a pedida por periférico externo e a pedida por periférico interno (*Timer*/Contador, porta serial, dentre outros).

Silva (2002), argumenta que o microcontrolador AT90S8515, conta com 13 interrupções vetoradas[15] no início da memória de programa (*flash*), possibilitando com isso, a implementação de um código de inicialização, que simplifica o trabalho do compilador C. De acordo com o autor, existem registradores de I/O específicos para realizar o controle do nível, da prioridade e da habilitação de cada interrupção, afirmando que a maior prioridade é do vetor RESET, na seqüência o vetor INT0 e assim sucessivamente, conforme pode ser observado na Tabela 2.8.

Na arquitetura AVR RISC, segundo Silva (2002), uma interrupção tem um tempo de resposta de 4 ciclos de *clock*, tanto para atender, quanto para

[15] As interrupções vetoradas são aquelas que possuem o vetor de interrupção (endereço de início da interrupção) fixo, e não pode ser modificado pelo usuário. Para o AT90S8515 todas as interrupções têm um vetor de reset próprio no espaço de memória de programa (Silva, 2002).

retornar. Ao atender, coloca o autor, o valor do PC (*program counter*) é empilhado (*stack*) e o *jump* para o vetor é realizado, por outro lado, no retorno, o valor da pilha é restaurado no PC e o fluxo do programa continua. Conforme Silva (2002), uma interrupção jamais é atendida durante a execução de uma instrução, quando a mesma durar mais que um ciclo, e também, jamais uma interrupção é atendida quando se está retornando de outra. De acordo com o autor, é executado, no mínimo, mais uma instrução do programa.

Tabela 2.8 – Tabela completa de interrupções do AT90S8515

Nº de vect.	Program Address	Source	Interrupt Definition
1	$000	RESET	Hardware Pin and Watchdog Reset
2	$001	INT0	External Interrupt Request 0
3	$002	INT1	External Interrupt Request 1
4	$003	TIMER1 CAPT	Timer/Counter1 Capture Event
5	$004	TIMER1 COMPA	Timer/Counter1 Compare Match A
6	$005	TIMER1 COMPB	Timer/Counter1 Compare Match B
7	$006	TIMER1 OVF	Timer/Counter1 Overflow
8	$007	TIMER0 OVF	Timer/Counter0 Overflow
9	$008	SPI, STC	Serial Transfer Complete
10	$009	UART, RX	UART, Rx Complete
11	$00A	UART, UDRE	UART Data Register Empty
12	$00B	UART, TX	UART, Tx Complete
13	$00C	ANA_COMP	Analog Comparator

Fonte: Silva (2002).

O microcontrolador AT90S8515, de acordo com Silva (2002), possui dois registradores de 8 bits para o controle das interrupções, o GIMSK – registrador geral de interrupções – e o TIMSK – registrador para interrupções do *Timer/Counter*. Quando uma interrupção é produzida, o bit 1 (um) do registrador de estado que permite produzir interrupções é posto a 0 (zero), desabilitando assim, todas as interrupções. Tendo atendido a interrupção e executado o código de programa necessário, o usuário precisa voltar a por este bit em 1 para que de novo as interrupções possam ser produzidas.

2.7.6 A Serial UART

A arquitetura AVR conta com uma UART (*Universal Asynchronous Receiver/Transmiter - Transmissor/Receptor*) que permite a comunição com outros dispositivos microcontrolados, microprocessados, sendo muito

semelhante às encontradas em equipamentos convencionais, como em microcomputadores pessoais. Está programada para trabalhar em modo de transmissão/recepção simultânea (full-duplex[16]), para tal, conta com dois registradores independentes: um somente de leitura e o segundo somente de escrita (Silva, 2002).

Segundo Silva (2002), possui a capacidade de receber e enviar bytes completos, com o stop bit programável permite a otimização da transmissão serial, o que possibilita taxas de comunicação que vão de 2400 a 115200 bauds[17] (faixa da norma RS232C).

Silva (2002), ainda comenta a respeito de detalhes que vem tornar o dispositivo mais robusto, no que diz respeito à agilidade e confiabilidade:

- Possui interrupções independentes para RX Complete, TX Complete e Data Registre Empty;
- Possui um mecanismo de confirmação de dados automáticos, tais como o *Framing Error*, *Overrun* e *False Startbit*;
- Filtro para eliminação de ruídos por amostragem múltipla;
- 8 ou 9 bits de dados;
- Alta taxa de transmissão em baixas freqüências XTAL.

Em Silva (2002), nos quatro possíveis modos de operação existentes, denominados modos 0,1,2 e 3; sendo que, somente o primeiro (modo 0) possui transmissão e recepção síncronas e os demais são assíncronas (Silva Junior[18], 1999). Segundo Silva (2002), o dispositivo serial UART no microcontrolador AT90S8515, especificamente, trabalha em modo 2 de operação, assim, em cada pacote são recebidos 11 (onze) bits, sendo: um start bit (nível 0), 8 (oito) bits de dados, um nono bit (relacionado ao registrador de controle do

[16] Full-Duplex ou simplesmente duplex, consiste num modo pelo qual os sistemas podem transmitir e receber dados simultaneamente (Silva, 2002).
[17] Unidade de Medida; o número máximo de variações de sinal por segundo (Silva, 2002).
[18] **SILVA JUNIOR**, Vidal Pereira da. **Aplicações Práticas do Microcontrolador 8051**. 8ª Edição. São Paulo; Érica, 1999. 270p.

dispositivo responsável pela comunicação serial) e o stop bit (nível 1) (Silva, 2002).

2.7.7 Algumas Aplicações do MSW

Silva (2002), comenta que teve o firme propósito de utilizar a Internet como uma rede global para monitoramento remoto de dispositivos. A expansão dos recursos disponíveis para a Internet e a ampla documentação sobre suas características técnicas e particularmente dos protocolos usados facilita o desenvolvimento das aplicações, custos e o tempo no desenvolvimento de sistemas que venham a utiliza-la. O autor procurou mostrar a viabilidade de desenvolvimento e implementação de um sistema embarcado, autônomo, de pequeno porte, que fosse flexível, escalável e baseado em microcontrolador que pudesse ser usado para controlar outros dispositivos elétricos da área doméstica como, lâmpadas, termômetros, câmeras, cafeteiras, equipamentos industriais como fornos, câmaras frias, equipamentos médicos, entre outros.

Silva (2002), realizou este experimento com a pretensão de mostrar uma aplicação para o MSW, onde houvesse uma aquisição e controle de temperatura via *Web*. Ele comenta que a implementação desta aplicação pode incluir controles termostáticos, industriais, sistemas, produtos domésticos, termômetros, ou qualquer sistema sensível térmico.

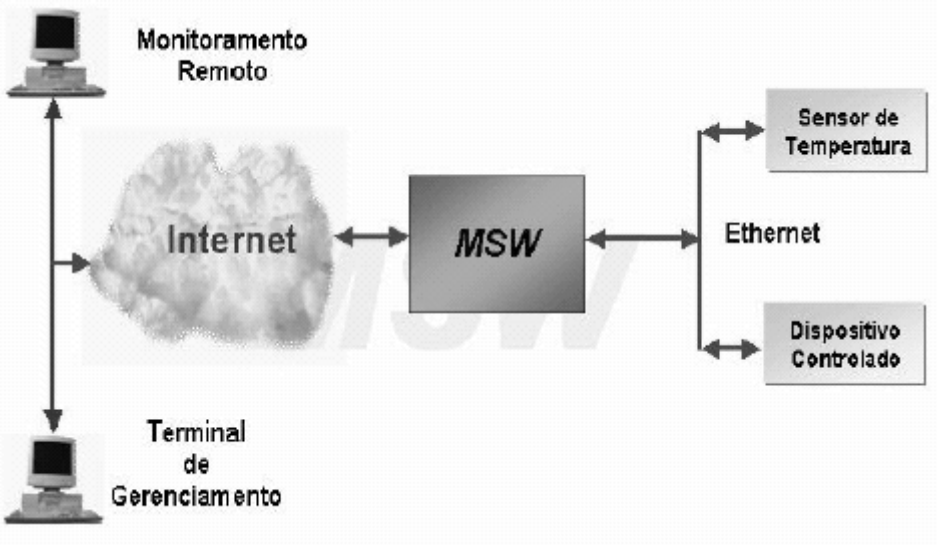

Figura 2.16 – Modelo de Aplicação de Controle de Temperatura (Silva, 2002).

Segundo Silva (2002), “a função principal do MSW é retornar páginas *web* e imagens em resposta às solicitações HTTP GET endereçadas pelas URLs, tendo como objetivo seu servidor HTTP. Assim, uma vez compilados todos os arquivos necessários ao funcionamento do microservidor, este pode ser acessado através da rede como um servidor *web* tradicional, uma vez que as URLs, direcionadas à porta TCP80, são respondidas da maneira convencional.”

Diante disso, o resultado da implementação de Silva (2002), pode ser acompanhado pela figura 2.17 que demonstra a página html armazenada dentro do MSW.

Figura 2.17 – Página *Web* Disparada pelo MSW (Silva, 2002).

De acordo com Silva (2002), a implementação do micro servidor *web* em aplicações na área médica é uma porta que se abre ao monitoramento remoto via Internet. O MSW pode ser conectado por meio da interface serial RS-232 com qualquer equipamento que disponha também deste tipo de interface. Os dispositivos remotos podem ser controlados e monitorados por um microcomputador convencional, desde que esteja conectado na mesma rede ethernet onde estão também os MSW.

Neste seguimento, El Shheibia (2003) propôs um modelo voltado para o monitoramento remoto de sinais vitais de leitos em UTI (Unidade de Terapia Intensiva), a Figura 2.18 demonstra o modelo proposto. Segundo o autor, a arquitetura básica do modelo é composta por equipamentos médicos para o monitoramento de sinais vitais dos pacientes, dispositivos autônomos de armazenamento temporário e visualização dos dados denominados Micro Servidores *Web* (MSW) e servidores de aplicações e dados para armazenamento definitivo dos dados.

O MSW faz aquisição de dados por meio da saída RS232 dos monitores de sinais vitais do paciente, realizando uma etapa de tratamento dos dados, disparando um alarme, caso os dados lidos, estiverem fora dos limites definidos pelo corpo clínico. Ressalta a importância de usar a memória EEPROM de 64*Kbytes* disponível no MSW para armazenamento dos dados lidos no período de 24 horas (El Shheibia, 2003).

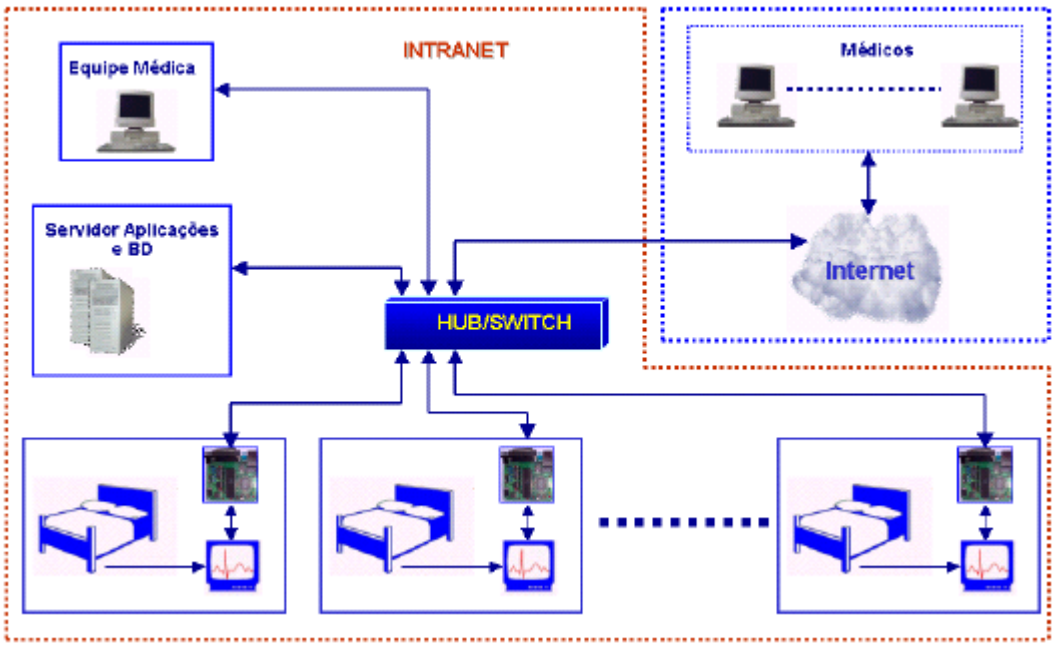

Figura 2.18 – Modelo de Monitoração de Pacientes El Shheibia (2003).

Segundo El Shheibia (2003), este mesmo modelo poderia ser ampliado para que os pacientes pudessem receber atendimento médico em suas residências. Acredita que este tipo de monitoramento remoto permitiria ao médico efetuar

acompanhamento do paciente através do monitoramento de sinais vitais e a partir destes, realizar tomadas de decisão. O autor ainda ressalta que nos dias atuais dispor de conexões *web*, seja comerciais ou residenciais, não representa dificuldades nem custos elevados.

Para este experimento El Shheibia (2003) monitorou algumas variáveis do conjunto de sinais vitais: freqüência cardíaca, nível de saturação de oxigênio no sangue (SpO2) e temperatura. Observe a Figura 2.19 que demonstra uma das páginas HTML embutida no MSW e que disponibiliza o acesso ao corpo clínico.

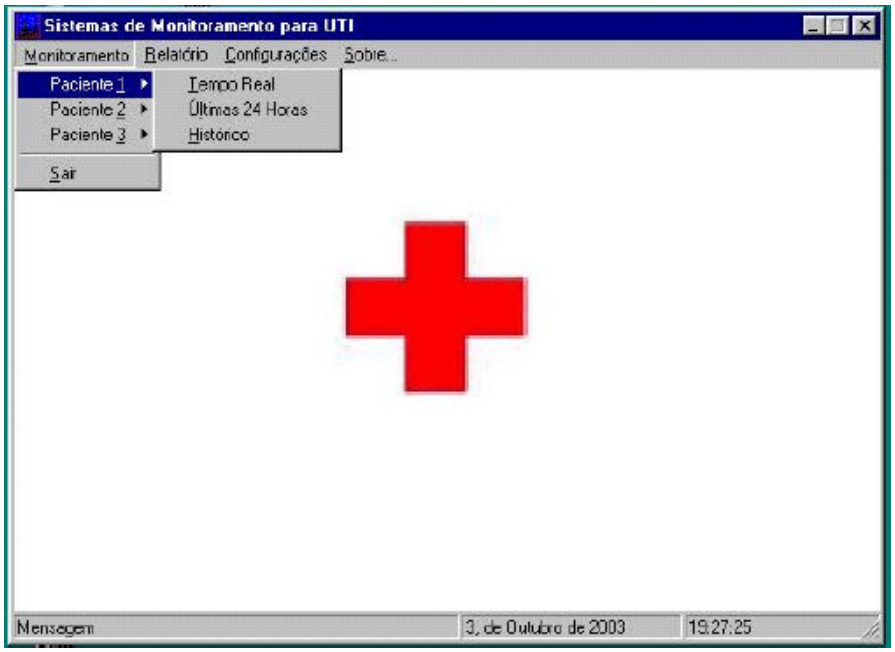

Figura 2.19 –Tela Princiapal do Sistema em El Shheibia (2003).

2.7.8 Aquisição de Dados e Monitoramento Remoto da Qualidade da Água

No trabalho de pesquisa de Benhardt, 2003, a aplicação do MSW foi voltada para o monitoramento da qualidade da água para cultivo de camarões ou como os cultivadores chamam “carcinicultura”.

Segundo a pesquisadora, “a qualidade da água de qualquer natureza está relacionada ao monitoramento de parâmetros físicos, químicos e biológicos,

cujos índices definem a finalidade para a qual a água será utilizada". Observe modelo proposto demonstrado pela figura 2.20.

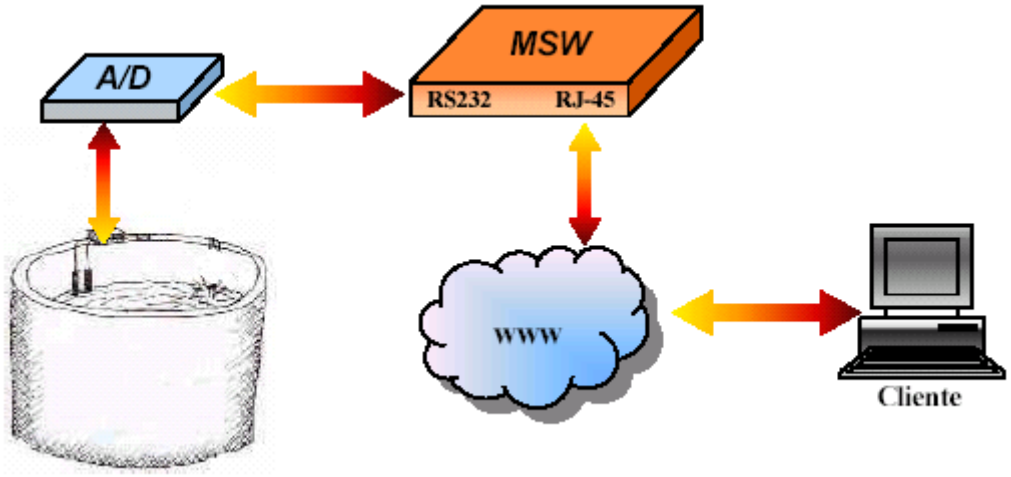

Figura 2.20 – Modelo de Monitoramento de Qualidade da Água (Benhardt, 2003).

Em seu experimento foram monitoradas as seguintes variáveis: temperatura, oxigênio dissolvido (OD), potencial hidrogeniônico (pH), salinidade e condutividade elétrica, ferro, sulfeto, amônia, nitrito, nitrato, ortofosfato e sílica. Veja na figura 2.21 como ficou a página disparada pelo MSW neste experimento.

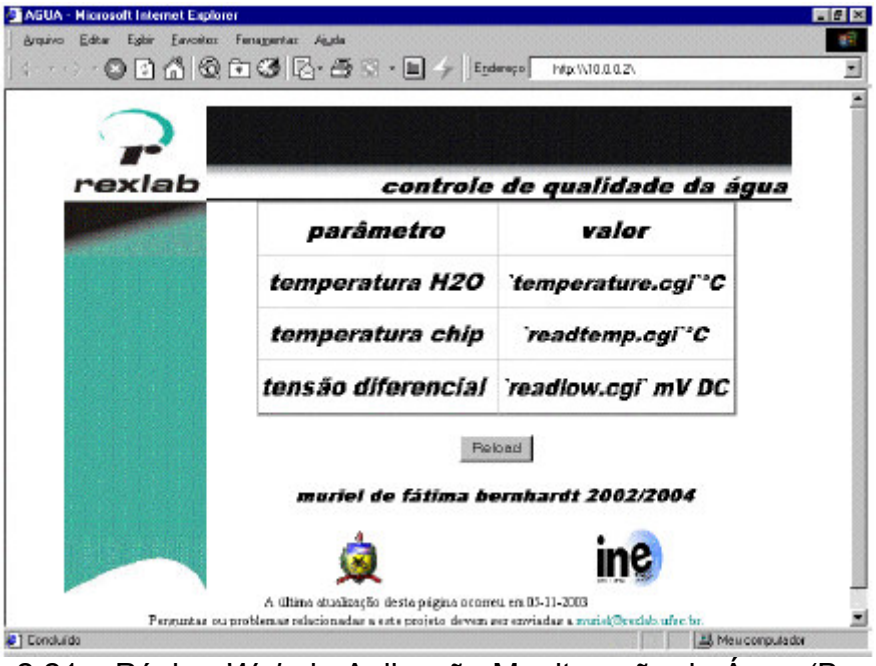

Figura 2.21 – Página *Web* da Aplicação Monitoração da Água (Benhardt, 2003).

3 MODELO PROPOSTO

Com o crescimento das necessidades humanas, a tecnologia, por meio de pesquisas, vem procurando fornecer soluções para atender, do mais simples ao mais complexo e crítico dos serviços. As soluções tecnológicas envolvem o ser humano de tal forma que a vida deste, muitas vezes, se torna extremamente dependente dessas, podendo ir desde situações simples até monitorações complexas que se mal sucedidas podem colocar em risco a própria vida. Soluções que por sua vez devem fornecer mobilidade, confiabilidade, disponibilidade e segurança, dando origem ao termo *dependabilidade*[19].

Com este intuito, a proposta deste trabalho procurou inspiração no modelo de EL SHHEIBIA (2003), que pode ser observado na Figura 2.18, uma vez que a área de abrangência de sua pesquisa demonstrou ser instigante, pois trata de monitoração de sinais vitais, onde,necessariamente, esta teria que contar com uma alta disponibilidade e controle de falhas nos dispositivos de aquisição de sinais, ou seja, o microservidor *web* (MSW).

Analisou-se o modelo de monitoração proposto por EL SHHEIBIA (2003) e foram observados os seguintes pontos:

- o modelo não contempla as fases detecção, reconhecimento e ação das possíveis falhas nos dispositivos MSW usados na monitoração do paciente;
- não apresenta uma abordagem eficiente de envio dos alarmes gerados pelo MSW;
- não conta com redundância nos MSW para dar maior disponibilidade no monitoramento dos sinais vitais do paciente.

[19] D*ependabilidade* é a propriedade dos sistemas computacionais que define a capacidade dos mesmos prestar um serviço no qual se pode justificadamente confiar. Dantas (2004).

Segundo Charnovsck (2004), os pacientes internados em uma Unidade de Terapia Intensiva (UTI) apresentam condições críticas de saúde e por isso precisam de cuidados intensivos e acompanhamento constante.

Florence e Calil (2003), relatam que há uma grande preocupação, por parte dos profissionais da saúde, com a segurança dos pacientes em ambiente hospitalar, sendo uma delas, o perigo associado aos problemas funcionais dos equipamentos médicos, que podem ocorrer durante sua via útil.

Visando contribuir, tanto para a amenização de falha desses equipamentos, quanto para o monitoramento remoto dos sinais vitais do paciente, o presente trabalho vem com uma proposta de aumentar a disponibilidade deste monitoramento, aplicando técnicas de tolerância a falha como a redundância dos MSW´s, aliada ao paradigma gerente-agente da gerência de redes e a versatilidade e funcionalidades do sistema de monitoramento de redes, o Nagios. Além disso, achou-se de grande importância, gerar um arquivo de *logs* com base nos alarmes produzidos pelo monitoramento, provendo assim, dados para futura análise por parte da equipe de gerenciamento de risco de equipamentos médicos do hospital.

A aplicação da redundância se faz necessário em monitoramentos que envolvem alto risco, evitando assim, o único ponto de falha ou *SPOF* (*Single Point Of Failure*). Caso um dos MSW´s vem apresentar falha, o outro MSW assume, de forma transparente para os usuários, o monitoramento do paciente. Espera-se, com isso, garantir que o monitoramento do paciente fique disponível 24 horas por dia e 7 dias por semana. O modelo proposto neste trabalho pode ser observado na figura 3.1 como segue.

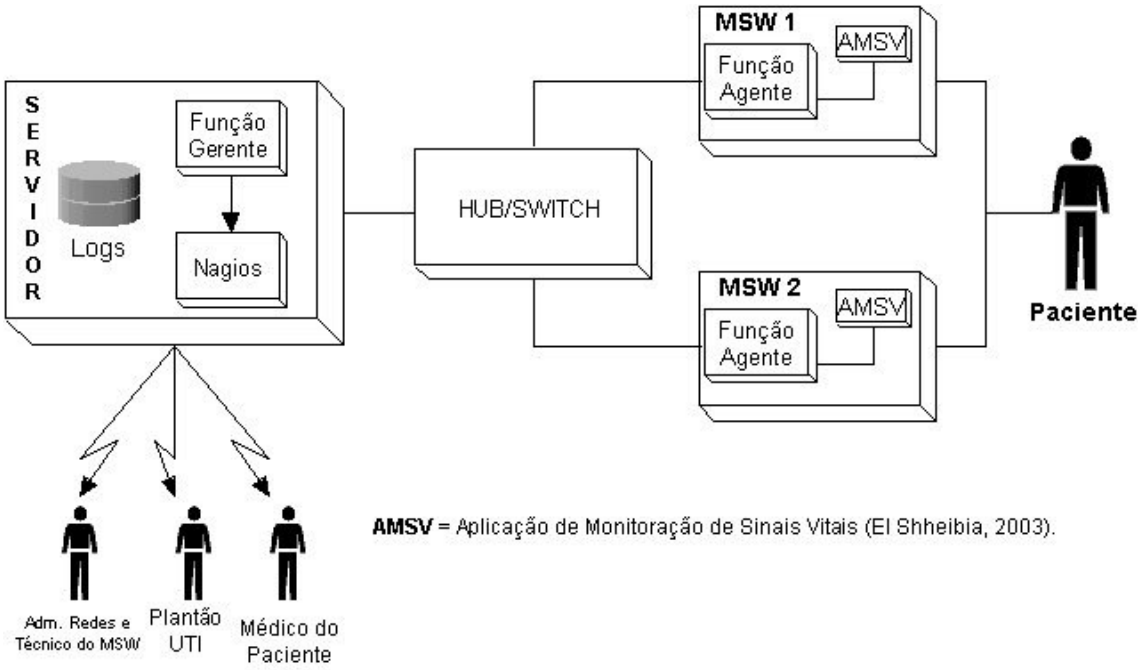

Figura 3.1 – Arquitetura geral do modelo proposto.

Conforme pode-se observar, a figura 3.1 demonstra o uso de dois MSW para realização do monitoramento dos sinais vitais de cada paciente. Cada MSW conta com a implementação da função agente embarcada, responsável em monitorar o estado do MSW e as interrupções de alarme geradas pela aplicação de El Shheibia (2003), enviando uma notificação para a função gerente, que fica no servidor, em caso de falha no dispositivo ou no paciente. A função gerente recebe a notificação e a classifica para envio imediato de mensagens aos responsáveis por cada área: Técnica e Administração da Rede, UTI e Médicos. Após o procedimento de envio das mensagens de alerta, a notificação é gravada em um arquivo localizado no servidor da aplicação de gerente que proverá dados para futuras análises tanto da parte de hardware, quanto do paciente.

Observou-se também que na proposta de El Shheibia (2003) não fica claro como se dá o alerta no corpo clíncio da UTI. Neste sentido, o presente trabalho procura demonstrar uma alternativa para solucionar esta questão.

O agente, embarcado no MSW, envia notificações de alarme ao gerente no servidor e este por sua vez, trabalho com alguns scripts que são responsáveis em montar as mensagens e encaminhá-las aos contatos publicados no arquivo *contacts.cfg* do Nagios. O gerente também fica responsável em enviar mensagens de alarme a estação do plantão da UTI, que conta com um *daemon* (programa que roda infinitamente no computador), checando o arquivo de alarmes se chegou alguma notificação, alarmando de forma sonora a equipe de plantão.

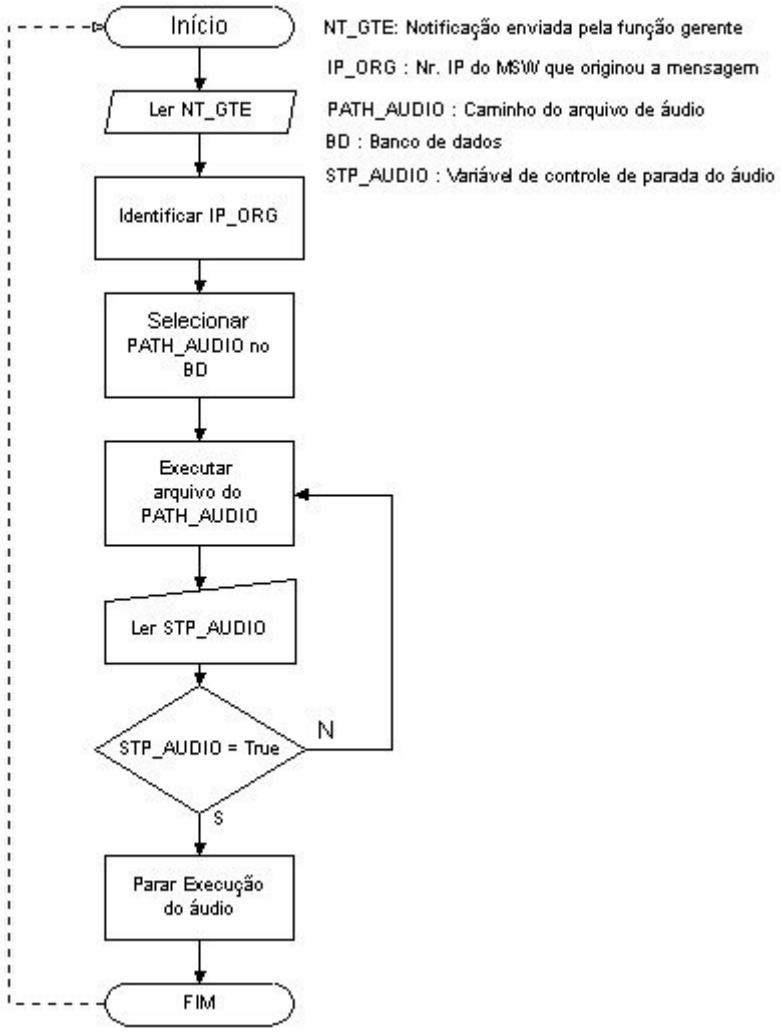

Figura 3.2 – Fluxograma do daemon na estação cliente do corpo clínico.

O *daemon* ao receber uma notificação de alarme, faz a leitura do arquivo de alarmes e identifica o endereço ip de origem (IP_ORG) que identifica o leito do paciente. Em seguida toca o arquivo de áudio da mensagem de alerta,

executando-o logo em seguida de forma ininterrupta, até que sua execução seja interrompida pelos integrantes do plantão da UTI. Para tanto, se faz necessário ter mensagens de alarme gravadas que possam ser executadas pelo *daemon*, conforme descrito em Cassaro (2004). Observe a estrutura de funcionamento do *daemon* na Figura 3.2.

3.1 Metodologia

Para o desenvolvimento da experimentação do modelo proposto são necessários alguns passos e adoção dos requisitos básico listados a seguir:

- Um sensor de temperatura para simular os sinais vitais do paciente;
- Dois MSW para a simular a redundância do serviço de monitoramento do sensor de temperatura;
- Dois micros-computadores:
    - Um como servidor:
        - Instalação do Sistema operacional FreeBSD versão 5.1;
        - Serviço de email;
        - Servidor Apache;
        - Compilador C gcc 3;
        - Nagios versão 1.1 instalado e configurado;
        - Placa *modem* modelo USR *Voice* ou *modem* externo;
        - Linha telefônica;
        - Aplicativo Mgetty da Alpha (somente instalar o pacote de voice) - ftp://alpha.greenie.net/pub/mgetty/source/1.1;
    - Um como estação, representando o plantão da UTI.
        - Sistema operacional Linux ou FreeBSD;
        - Arquivos de som previamente gravados (mensagens de alerta);
        - Kit multimídia (Placa de som, caixas de som, tocador de áudio e etc.).

3.1.1 Escolha da plataforma de Monitoramento da Rede

Como todo desenvolvimento do projeto MSW, desde Silva (2002), contemplava uma estrutura de código aberto, optou-se pelo Nagios, não apenas por ser de código aberto, mas por sua flexibilidade, versatilidade e ser muito bem aceito na comunidade de gerência de redes. Com sua capacidade de verificar constantemente a disponibilidade dos serviços de forma local ou remota, alertando por intermédio de diversas formas de mensagens sobre o problema ocorrido e , ainda, contar com relatórios de disponibilidade e configurar ações corretivas para os problemas na rede, o Nagios mostrou-se extremamente qualificado para este modelo.

Uma vez que delongaria por demais, este trabalho, explanar passo-a-passo todas as instalações e configurações tanto do Nágios, quanto das demais ferramentas, parte-se do pressuposto que já se tenha uma máquina servidora com sistema operacional FreeBSD versão 5.1, serviço de email, servidor *Web* apache, Nagios versão 1.1, MGetty e placa *modem* modelo USR *voice* ligado a uma linha telefônica devidamente instalados. Para instalação e configuração do Nagios usou-se os passos descritos em Lopes e Martins (2004), no que se refere ao aplicativo MGetty foi usado os passos descritos em Cassaro (2005).

3.1.2 Configurando o Nagios para notificações

Para que o Nagios possa realizar o trabalho de notificação aos profissionais da área médica e técnica, se fazem necessários alguns scripts e algumas alterações nos arquivos de configuração:

- **nagios.cfg:** arquivo principal de configuração do Nagios. Muitas configurações já trazem propriedades como default para facilitar o uso da ferramenta, entretanto, se faz necessário observar as configurações de algumas variáveis:
    - **log_file**: variável que indica o caminho e o arquivo aonde o Nagios escreverá os erros de configuração.
        - Exemplo: log_file = /var/spool/nagios/archives /nagios.log

- **cfg_file**: variável que indica o caminho dos arquivos de configuração dos serviços, hosts, contatos, comandos e etc., que o Nagios usará para realizar o monitoramento.
  - Exemplo: cfg_file = usr/local/etc/nagios/hosts.cfg
- **cfg_dir:** variável que indica o caminho dos diretórios dos arquivos de configuração indicados na variável cfg_file.
  - Exemplo: cfg_dir = /usr/local/etc/nagios/hosts
- **status_file:** variável que indica o caminho do arquivo de log que o Nagios usa para gravar os estados dos serviços configurados na variável cfg_file.
  - Exemplo: /var/spool/nagios/archives/status.log
- **date_format**: variável que indica o formato de data que será armazenado e visualizado pela interface web.
  - Exemplo: date_format = euro
- **admin_email:** variável que indica o endereço de email do administrador da máquina onde o Nagios está instalado.
  - Exemplo: admin_email = root@localhost.domíniolocal

- **contacts.cfg:** arquivo de publicação dos contatos a serem notificados, veja exemplo:

```
define contact {
    contact_name                    Adm-cel
    alias                           Gilberto Medeiros
    service_notification_period     24x7
    host_notification_period        24x7
    service_notification_options    c,r
    host_notification_options       d,r
    service_notification_commands   notify-by-modem
    host_notification_commands      none
    email                           giba@inf.ufsc.br
    pager                           99990000}
```

Na opção *email* e *pager* são fornecidos, respectivamente, o endereço de *email* e o telefone que receberão as notificações vindas do servidor da rede pelo Nágios.

Obs.: Para receber a notificação de voz do Nagios é necessário colocar o número do telefone celular ou outro telefone que os contatos estejam sempre disponíveis.

- **misccommands.cfg:** arquivo responsável em avisar o Nagios quais os tipos de notificação devem ser enviadas aos contatos. Os contatos podem receber notificações por vários meios de comunicação. No exemplo tem-se uma definição de envio de notificação via modem.

```
# 'notify-by-modem' command definition
define command{
    command_name            notify-by-modem
    command_line            /usr/local/bin/send_modem.sh
    "$CONTACTPAGER$" "$HOSTNAME$" "$SERVICEDESC$"
    "$NOTIFICATIONTYPE$"}
```

A opção *command_line* recebe o caminho do *send_modem.sh* que é o script responsável por colocar as mensagens no diretório *spool* do *mailmodem*, que posteriormente serão enviadas via *modem*.

Mediante uma falha crítica o Nagios envia, de forma paralela, notificações para todos os contatos publicados no arquivo *contacts.cfg* e *contacts_groups.cfg*. O que gera um problema para o enviar a notificação via *modem*, uma vez que o mesmo trabalha de forma serial.

Para amenizar este problema usou-se a metodologia descrita em Cassaro (2005), onde alguns scripts que simulam um “servidor de email” para mensagem de voz. Com isso, consegue-se enviar as notificações de forma serial e não paralela. Os scripts podem ser vistos no anexo 2 deste trabalho.

De acordo com Cassaro (2005), para que se possa enviar as notificações via *modem*, os arquivos de som, contendo informações referente ao alarme, devem ser gravados em *WAV* no formato **22050/16bits/Mono**. Depois disso, é necessário converter o arquivo para o formato que o *voice modem* entenda, no caso em questão (.rmd).

Para cada evento de alarme são necessários a gravação de 3 arquivos de som no formato descrito no parágrafo anterior, um identificando o local da ocorrência (hospital), outro identificando o objeto da falha (paciente ou MSW) e o outro identificando o local da falha. Veja exemplo na tabela 3.1.

Tabela 3.1 – Arquivos de sons referente as mensagens de notificação.

Nr.	Responsável	Msg 1	Msg 2	Msg 3
1	Médico	Hospital Universitário	Paciente	Leito X
2	Médico	Hospital Universitário	Monitoramento	Indisponível
3	Adm. Rede	Hospital Universitário	MSW	Leito X
n	...	...	...	...

Uma vez gravado os arquivos de som com as mensagens, o *daemon* do *mailmodem.sh*, script encarregado em vasculhar, a cada 60 segundos, o diretório "/var/spool/mailmodem" configurado no arquivo *mailmodem.conf*, onde as mensagens são armazenadas, tem a função de concatenar os arquivos de som. Pode-se observar na figura 3.3 as configurações do arquivo *mailmodem.conf* que por sua vez deve ficar no diretório /*etc.*

```
# Mail Modem Conf file

# Where the playmodem.sh stay
playmodem="/usr/local/bin/playmodem.sh"

# The place of spool messages
spool="/var/spool/mailmodem"

# Temporary directory
tmp="/tmp"

# The translation file
translation="/etc/nagios/translation.txt"

# Directory of sounds
sounddir="/usr/local/share/sounds"
```

Figura 3.3 – Arquivo de configuração *mailmodem.conf* (Cassaro,2005).

Os arquivos de som concatenados formarão uma mensagem que será enviada via modem para o telefone publicado no arquivo *contacts.cfg* do Nagios. O contato receberá, algo como: "**Hospital Universitário, Paciente, Leito X**".

Os *scripts* que realizam o tratamento e envio das mensagens podem ser encontrados nos anexos deste trabalho.

3.1.3 Funções Gerente e Agente

Para modelagem das funções agente e gerente este trabalho usou o conceito de gerente-agente, por se tratar de um modelo bem difundido na área de monitoração de redes, de fácil entendimento e aplicável a este modelo de experimento.

O gerente é uma aplicação que fica armazenda no servidor e o agente é a aplicação que fica armazenada no dispositivo monitorado. Segundo Stallings (1999), existe a possibilidade do gerente e o agente estarem no mesmo ambiente.

Neste trabalho, a função gerente fica localizada no servidor da rede, onde a aplicação de gerência está instalada, no caso o Nagios. Por outro lado, o agente é a função embarcada na flash do MSW e é responsável em comunicar com o gerente e notificando-o do risco de falha do dispositivo ou do paciente. Observe a figura 3.4 que demonstra o modelo gerente-agente.

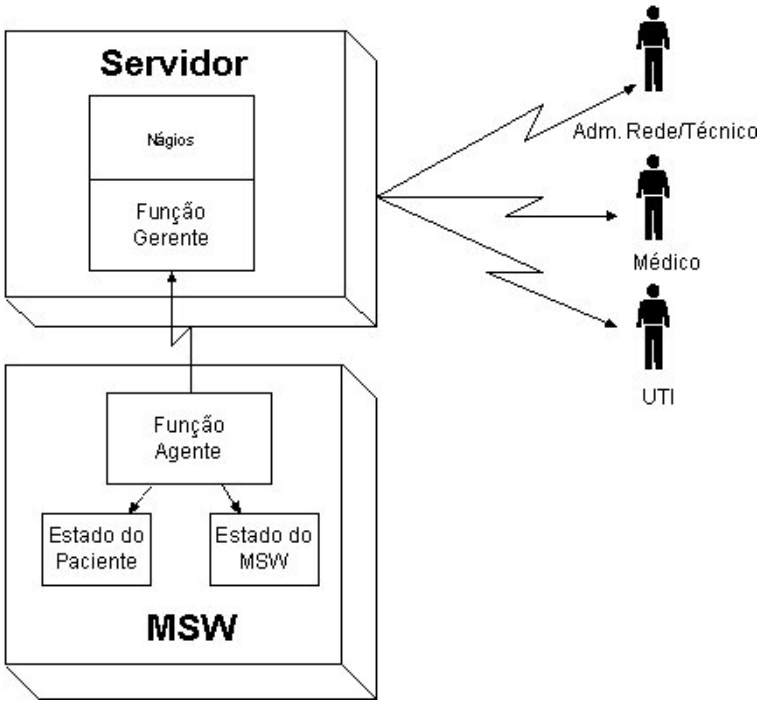

Figura 3.4 – Modelo de Monitoração Gerente-Agente.

Analisando a figura 3.4, pode-se observar a responsabilidade de cada módulo:

- **Nagios**: responsável em monitorar o estado da estrutura da rede, entretanto, contará com o *script* da função gerente que receberá notificações do agente do MSW e enviará mensagens aos profissionais responsáveis (Administrador da rede, técnico do MSW, Médico do paciente e Plantão da UTI).
- **Função Gerente**: é o *script* responsável em receber as notificações da função agente, sobre o estado do MSW e do paciente, aciona o nagios a enviar mensagens de alerta aos contatos, alimenta o arquivo de alarmes na estação cliente do corpo clínico e grava o evento ocorrido no arquivo de log para futuras estatísticas de controle.
- **Função do Agente**: é o *script* responsável em ler o estado do MSW, apurando se a leitura está dentro dos limites pré-estabelecidos e ler o sinal de alarme da aplicação de El Shheibia (2003), notificando a função gerente em caso de falha.
- **Estado MSW**: sinais dentro do MSW, que poderiam indicar um indício de falha que indisponibilizaria a monitoração do paciente. Uma vez

identificado esses sinais, o agente seria programado para ler, analisar e notificar a função gerente.

A Figura 3.5 objetiva esclarecer o funcionamento do agente embarcado no MSW.

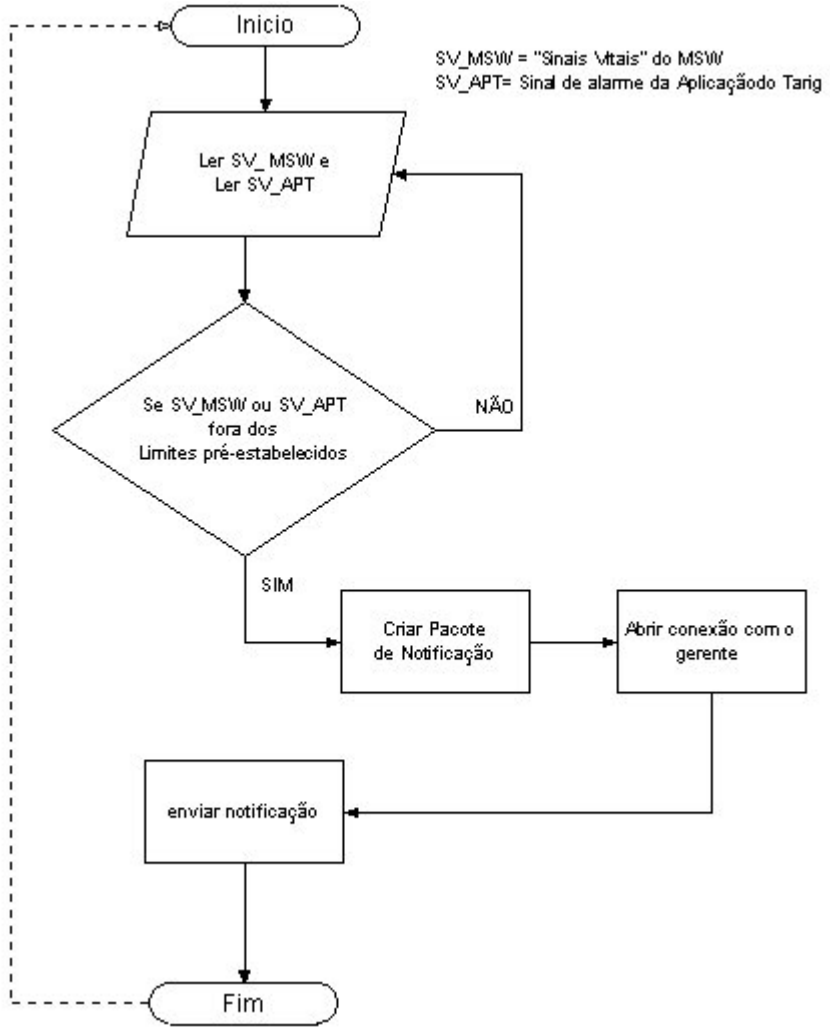

Figura 3.5 – Fluxograma do Agente embarcado no MSW.

Observando a Figura 3.5, percebe-se que por meio da leitura desses dois sinais, pode-se ter uma posição do estado tanto do MSW, quanto do paciente. Com essas informações recebidas pelo agente e enviadas ao gerente que aciona os *scripts* (*send_modem.sh e mailmodem.sh*), que farão a montagem da mensagem e enviarão as notificações para os contatos pubicados no arquivo *contacts.cfg* do Nagios, observando o tipo do alerta, ou seja, se o

gerente identificou um alarme de problemas no MSW, será enviado uma mensagem para o pessoal técnico (administrador da rede e o técnico do MSW) do contrário, será alertado a UTI e enviado uma mensagem para o médico do paciente.

O fluxo deste trabalho pode ser observado na Figura 3.6 que demonstra o funcionamento do gerente localizado no servidor.

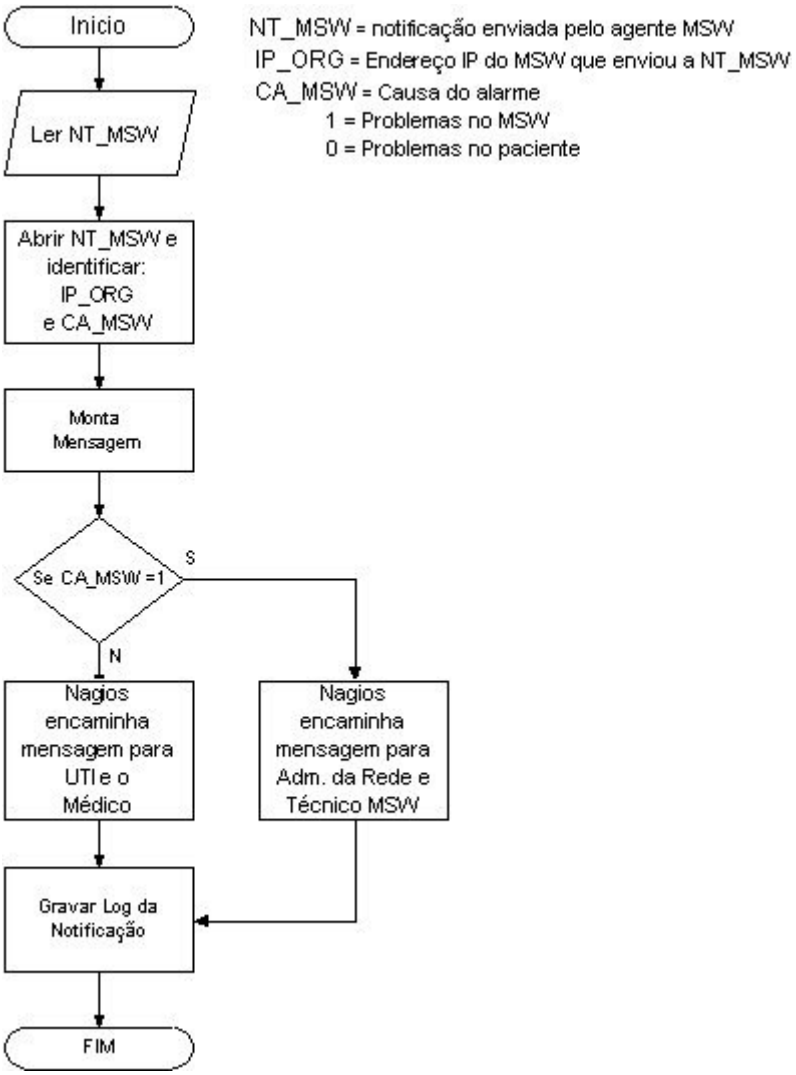

Figura 3.6 – Fluxograma da função gerente no servidor.

3.2 Simulação do Modelo Proposto

Partindo da arquitetura de rede demonstrada no modelo de El Shheibia (2003), figura 2.18, a tabela 3.2 procura observar os possíveis casos onde pode ocorrer falha:

Tabela 3.2 – Casos de falha na estrutura de monitoramento de pacientes.

	Pontos de Falhas					Alarmes			
Casos	PAC	MSW1	MSW2	ROT	SERV	UTI	MED	ADM	TÉC
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	1	1
2	0	0	0	1	0	1	0	1	1
3	0	0	0	1	1	1	0	1	1
4	0	0	1	0	0	0	0	1	1
5	0	0	1	0	1	0	0	1	1
6	0	0	1	1	0	1	0	1	1
7	0	0	1	1	1	1	0	1	1
8	0	1	0	0	0	0	0	1	1
9	0	1	0	0	1	0	0	1	1
10	0	1	0	1	0	1	0	1	1
11	0	1	0	1	1	1	0	1	1
12	0	1	1	0	0	1	0	1	1
13	0	1	1	0	1	1	0	1	1
14	0	1	1	1	0	1	0	1	1
15	0	1	1	1	1	1	0	1	1
16	1	0	0	0	0	1	1	0	0
17	1	0	0	0	1	1	1	1	1
18	1	0	0	1	0	1	1	1	1
19	1	0	0	1	1	1	1	1	1
20	1	0	1	0	0	1	1	1	1
21	1	0	1	0	1	1	1	1	1
22	1	0	1	1	0	1	1	1	1
23	1	0	1	1	1	1	1	1	1
24	1	1	0	0	0	1	1	1	1
25	1	1	0	0	1	1	1	1	1
26	1	1	0	1	0	1	1	1	1
27	1	1	0	1	1	1	1	1	1
28	1	1	1	0	0	1	1	1	1
29	1	1	1	0	1	1	1	1	1
30	1	1	1	1	0	1	1	1	1
31	1	1	1	1	1	1	1	1	1

De acordo com os casos levantados na tabela 3.2, foram encontradas suas respectivas soluções e tomada de decisão.

Para ilustrar os casos apontados na tabela 3.2, foi necessário montar um ambiente, desenvolvido na linguagem Object Pascal no ambiente Delphi, que pudesse demonstrar a estrutura de rede com seus dispositivos, os MSW, o paciente, os responsáveis por cada área, o agrupamento de falhas, o Nagios que encaminha as notificações aos contatos publicados no arquivo *contacts.cfg* e o banco de dados que representa o arquivo de log dos eventos ocorridos. A figura 3.7 demonstra o ambiente construído para a simulação.

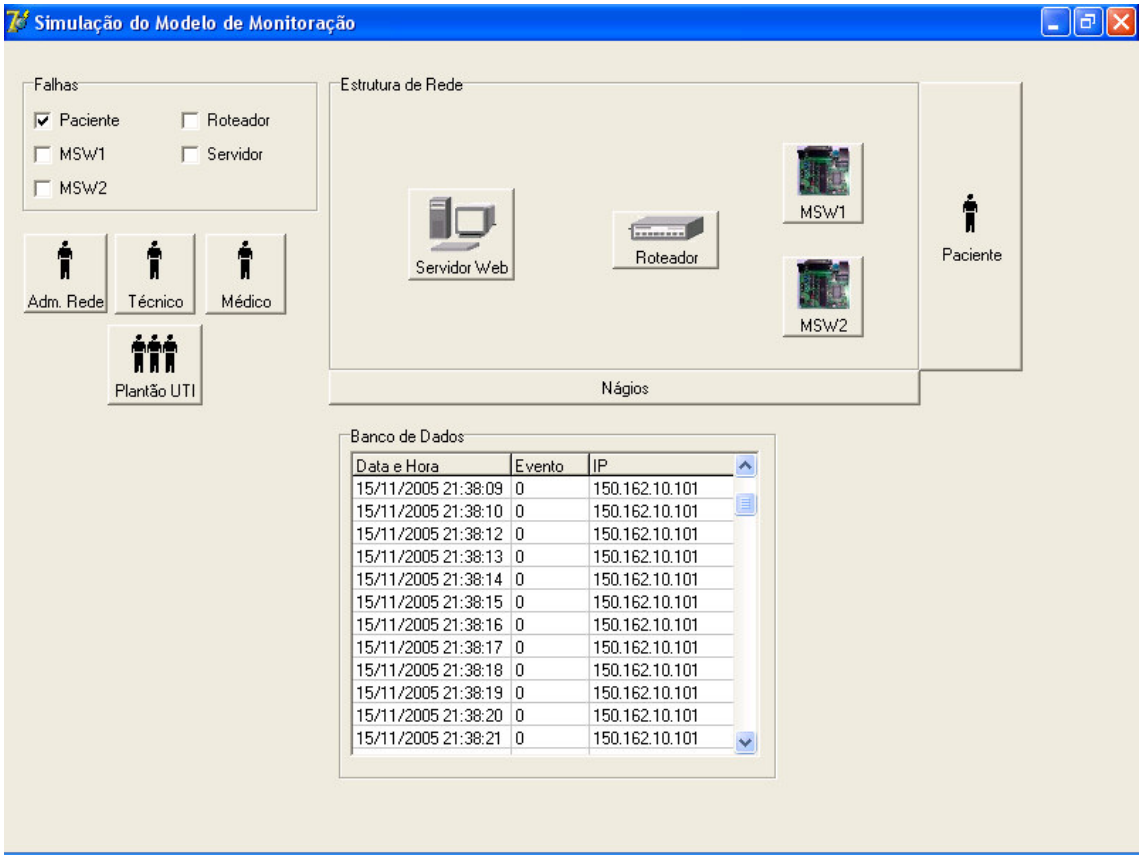

Figura 3.7 – Ambiente da simulação do modelo proposto.

Por meio deste ambiente foi possível demonstrar o comportamento de cada falha. Conforme observa-se na figura 3.7, basta escolher no agrupamento de falhas, o dispositivo que deseja-se colocar em situação de falha para o ambiente mostrar, visualmente, através de pulsações os componentes da estrutura que foram afetados pela falha e a ação do Nagios no envio de notificações.

Ainda observando a figura 3.7, pode-se observar que a simulação demonstra uma falha no paciente, logo, os componentes que representam a falha no paciente começam a pulsar. Observa-se também as informações armazenadas no banco de dados, a hora que ocorreu o evento, o evento indicando que a falha é no paciente e não de *hardware* e o endereço ip do MSW que informa a localização do leito na UTI. Essas informações são tratadas pela função gerente e os scripts mencionados na seção 3.1.2 e enviadas via modem para os contatos publicados no arquivo de contatos do Nagios (*contacts.cfg*).

4 DISCUSSÃO

Lopes et al. (2003) fizeram uma analogia entre ferramentas de gerência de redes e ferramentas médicas, onde estas se tornam presença indispensável no cotidiano dos profissionais dessas áreas. Assim como os médicos fazem uso de ferramentas que possam auxiliá-los na geração de diagnósticos mais precisos, com a gerência de rede temos uma situação bastante semelhante. Quando não estamos bem instrumentados, não somos capazes de descobrir problemas e por conseqüência, não seremos capazes de solucioná-los. Isso nos afastará substancialmente de nosso objetivo, que é manter o bom funcionamento da rede.

O sucesso da Internet e Intranets, juntamente com a utilização de sistemas embutidos nos equipamentos modernos, fez com que fabricantes fornecessem serviços para conexão TCP/IP em seus produtos. A expansão popular da *Web* atraiu esforços no sentido de se gerenciar equipamentos de rede via *Web* (Mendes e Fernandes, 1998).

Neste sentido, para elaboração do modelo proposto buscou-se uma ferramenta que pudesse não somente ajudar no diagnóstico da estrutura de rede, que apóia os MSW´s, mas também prover condições de alertar os interessados em caso de problemas tanto com o paciente, quanto do MSW. Tendo em vista que o Nagios é um aplicativo de código aberto, com abertura de implementação de novas formas de gerência, possuindo a funcionalidade de informar os problemas de rede, sagrou-se como a ferramenta escolhida.

Uma vez detectado problemas, o *daemon* do nágios pode enviar notificações para contatos administrativos de várias formas diferentes (email, mensagens instantâneas, SMS, etc.), bem como informações sobre o estado atual, histórico de *logs* e *reports,* podem ser acessados através de um navegador *web* (Soares, 2004).

É importante ressaltar que este trabalho não tem a pretensão de explorar todas as possibilidades de configuração associadas ao nágios, mas somente aquelas que permeiam a pesquisa.

A internet, mesmo com toda polêmica que a ronda, vem provando ser uma excelente mediadora entre o problema e a solução, sendo palco para apresentação de muitos trabalhos e experimentos científicos no que tange a área de saúde. O surgimento da telemedicina, com diagnósticos e atendimentos médicos a pacientes a distância, propriciou o uso crescente e significativo destes serviços tanto por profissionais da área de saúde, quanto seus pacientes.

O modelo de monitoração apresentado tem como enfoque a monitoração de pacientes em UTI. Entretanto, como mencionado por Falcão (1999), não é novidade que no Brasil o sistema de saúde é precário. Anualmente cerca de 10% da população brasileira sofre algum tipo de internção, saturando as instalações hospitalares, causando um verdadeiro caos devido a falta de leitos. Uma possível alternativa para contribuir com a solução da problemática levantada, seria o emprego do modelo proposto aliado aos conceitos de *wireless* (sem fio) e de *HomeCare* (cuidados em casa), podendo-se observar o paciente remotamente dentro do seu ambiente familiar, contribuindo com o desafogamento dos leitos hospitalares.

A grande necessidade do desenvolvimento de tecnologias capazes de encurtar distâncias e de fornecer mobilidade, conforto e segurança na área de saúde, fomentou sobremaneira o crescimento de pesquisas neste campo.

Neste contexto, Pizarro *et al.* (2001) propuseram o monitoramento remoto de sinais vitais de pacientes, fazendo uso de tecnologia *wireless* (sem fio) numa arquitetura cliente/servidor, permitindo, com isso, uma maior mobilidade aos pacientes que poderiam estar em ambiente hospitalar ou domiciliar. Nesta arquitetura são necessários um sistema de aquisição (cliente) e um subsistema

central (servidor) que intermedia as informações, direcionando-as aos subsistemas de visualização (cliente), o que ocasiona um certo atraso no sinal.

O mesmo problema é observado na pesquisa desenvolvida por Ortis *et. al*. (2004), onde se objetivou criar um software para visualização dos sinais capturados na tela de um APD (Assitente Pessoal Digital) e a transmissão destes sinais para monitoração remota. Neste caso, o software para a plotagem dos sinais (Medplot) é um sistema em tempo crítico, pois o atraso na plotagem dos sinais de EEG, ECG e EMG poderá prejudicar a sua análise.

A arquitetura proposta neste trabalho, quando comparada às arquiteturas propostas por Pizarro *et al*. (2001) e por Ortis et al. (2004), tem como vantagem o fato de o próprio dispositivo de aquisição de dados capturar, processar e fornecer as informações diretamente para visualização por meio de um navegador *web*, em tempo real.

Isto vem ao encontro da idéia apresentada por Christ et al. (2004) que utiliza um monitor de pacientes comercial e o protocolo UDP para permitir a monitoração remota de pacientes, em tempo real, através da intranet de um hospital. Entretanto, apesar de não haver atraso no envio das informações, estas ficam restritas a rede interna do hospital.

Torna-se necessário esclarecer que a monitoração via internet requer a aplicação do protocolo TCP (em lugar do UDP), que é o caso da proposta do presente trabalho. A escolha da utilização de UDP como protocolo de transporte ocorre, por vezes, devido ao fato deste tipo de protocolo não orientado a conexão ter habilidade de continuar funcionando quando a rede falha. Entretanto, segundo Lima (2005), isto não passa de mito, pois 100% dos gerenciamentos de redes vem sendo feitos quando a rede não está falhando, e até porque, hoje quando há uma pane de rede é devido a problemas físicos e, assim, nem o TCP nem tão pouco o UDP iriam funcionar. Assim sendo, este trabalho procurou trabalhar com o TCP que, contrário ao UDP, fornece uma forma de entrega confiável de pacotes.

A arquitetura do modelo deste trabalho ainda prevê a existência de um agente embarcado em cada dispositivo de aquisição (MSW), com a função de coletar e verificar o estado do MSW e as interrupções geradas por parte da aplicação de El Shheibia (2003), enviando notificações ao servidor da aplicação gerente que as classifica, alertando por intermédio do nagios os contatos da área médica e técnica, gerando ao final uma base de *logs* desses alertas.

Segundo Specialski (2000), para se obter uma administração de redes eficiente, seria necessário abodar todas áreas funcionais da gerência de redes. Entretanto, este trabalho abordou somente as áreas: Gerência de Falhas, Gerência de Configuração e Gerência de Desempenho que, segundo Stallings (1999), são as três áreas funcionais que geralmente são mais importantes para o monitoramento da rede.

Sendo assim, o modelo proposto neste trabalho, demonstrou suas vantagens frente aos modelos dos trabalhos expostos uma vez que, além de apresentar características de aquisição de sinais, processamento, análise e visualização das informações via *web*, dispõe ,ainda, de redundância dos dispositivos de aquisição dos sinais, aumentando com isso, a disponibilidade, garantindo a continuidade do monitoramento do paciente, mesmo na presença de falhas no dispositivo de aquisição dos sinais (MSW).

Segundo os idealizadores do MSW do RexLab (Silva e Dpaula), veja comunicação no anexo 6, ainda, não são conhecidas as variáveis que poderiam servir como base para a monitoração de falha no MSW, uma vez que seu o *kernel* (núcleo) é fechado. Para contornar este problema, ainda sem solução, o presente trabalho propõe a redundância no monitoramento, como alternativa a ausência de definição das variáveis de controle do MSW. Vê-se, portanto, a importância de não somente dar condições de alta disponibilidade no monitoramento de pacientes, mas também há necessidade de realizar um estudo visando a identificação dos possíveis pontos de falha dentro do MSW.

5 CONSIDERAÇÕES FINAIS

5.1 Conclusões

A característica do MSW no monitoramento de sinais vitais de pacientes em UTI, o classifica como um equipamento essencial e ao mesmo tempo crítico, exigindo, assim, ser acompanhado e gerenciado com finalidade de evitar falhas que possam acarretar danos irreversíveis ao paciente.

A aplicação de redundância dos MSW aliados ao seu monitoramento, por meio de agente e gerente, garante maior segurança e disponibilidade à monitoração dos sinais vitais do paciente.

Embora o Nagios não seja uma ferramenta muito fácil de instalar e configurar, demonstra ser um software muito flexível e versátil, com a vantagem de poder enviar notificações e não haver a necessidade de se ter um equipamento dedicado ao monitoramento, dando mobilidade ao administrador da rede para o acompanhamento do estado da mesma.

A utilização do Nagios contribui para o controle do monitoramento da rede e do paciente, uma vez que, pode-se incorporar novas funcionalidades como foi o caso de trabalhar em conjunto com o aplicativo Mgetty que, juntamente com outros scripts, realiza o envio das notificações via modem.

Em redes que exigem alto grau de confiabilidade e disponibilidade, torna-se indispensável fazer uso de sistemas de monitoração que possuam a habilidade de enviar mensagens de alerta em caso de problemas nos serviços e na estrutura da rede.

O modelo proposto atende aos requisitos de baixo custo, garante confiabilidade e é capaz de auxiliar e alertar os profissionais das respectivas áreas: saúde e hardware.

Considerando que não são conhecidas as variáveis que poderiam servir como base para a monitoração dos possíveis pontos de falha do MSW, o presente trabalho propôs a redundância no monitoramento como alternativa para contornar este problema, ainda sem solução. Vê-se, portanto, a importância de não somente dar condições de alta disponibilidade no monitoramento de pacientes, mas também há necessidade de realizar um estudo visando a identificação dos possíveis pontos de falha dentro do MSW.

A informática na saúde encurta distâncias e ganha tempo, em conjunto com bons profissionais provê suporte sem igual à manutenção da vida, surge como salva-vidas em mar revolto, garantindo, muitas vezes, o suspiro da continuidade.

5.2 Recomendações Futuras

- Estudar mais a fundo a estrutura do MSW, buscando identificar os possíveis pontos de falha, bem como seus limeares, a fim de melhorar a monitoração preventiva de falhas, dando maior confiabilidade no uso da tecnologia não somente em leitos hospitalares, mas em todas áreas de monitoramento que exigem alto grau de disponibilidade e confiabilidade.

- Implementar o modelo proposto testando-o dentro da realidade de um ambiente hospitalar, para monitoramento de pacientes em UTI.

- Realização de testes com o MSW em tratamentos tipo *HomeCare* (cuidado domiciliar), como alternativa de realização de consultas e desafogamento de leitos hospitalares.

- Realização de análise e desenvolvimento de um sistema especialista que integre as informações obtidas, por meio da monitoração do paciente, com o sistema de prontuário eletrônico, resultando em informações que possam auxiliar os médicos num diagnóstico mais efetivo.

6 REFERÊNCIAS BIBLIOGRÁFICAS

ARMBRUST,Tom**. Network-Enabling Technology for Medical Equipment:** Build or Buy?. Disponível em: http://www.devicelink.com/mem/archive/01/09/005.html. Acessado em 14/07/2005

BAROTTO, Ângela M. **Gerência de Alarmes em Ambientes *Web*, Wap e SMS usando as tecnologias SNMP, WBEM e CORBA**. 2002. 114 f. Dissertação(Mestrado em Ciência da Computação) – Universidade Federal de Santa Catarina, Florianópolis.

BEBERT, Wanderson. **Nagios - Um poderoso programa de monitoramento de rede (parte 2).** Publicado em 18/01/2004. Disponível em http://www.vivaolinux.com.br/artigos/verArtigo.php?codigo=596. Acessado em 08/07/2005.

BERNHARDT, Muriel F. **Micro Servidor *Web* no Monitoramento e Controle Remoto da Qualidade da Água na Carcinicultura**. 2003. 112 f. Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Santa Catarina, Florianópolis.

BERNHARDT, Muriel F.; **PALADINI**, Suenoni. **MicroServidor Chiron**. 304 f. 2001. Monografia (Bacharelado em Ciência da Computação) - Universidade do Sul de Santa Catarina, Araranguá..

BOUTABA, Raouf.; **POLYRAKIS**, A Andreas. Projecting advanced enterprise network and service management to active networks. IEEE Network. [s.l.]. v. 16, n. 1, p. 28 –33, Jan.-Fev. 2002 .

CARVILHE, J. L. V. **A utilização de tecnologias *web* em sistemas de gerência corporativa**. Curitiba:, 2000. Monografia (Curso de Especialização em Sistemas Distribuídos) Pontifícia Universidade Católica do Paraná.

CASSARO, Allan. **Nágios Falante.** Viva o Linux . Publicado 24/01/2005. Disponível em: www.vivaolinux.com.br. Acessado em 08/07/2005.

CHARNOVSCKI, Rafael. **Registro Eletrônico para Acompanhamento Médico de Pacientes em uma UTI.** Artigo 2004.

CHRIST, Rafael E. *et al*. **Sistema de Monitoração Remota de Pacientes em Tempo-Real Através da Intranet do Hospital**. Curitiba, 2004.

CISCO Systems Inc. **Simple Network Management Protocol (SNMP)**. In Cisco Systems Inc., Cisco Connection Documentation - Enterprises Series, San Jose, 1996.

DIEGUEZ, Carla. **Incor utiliza iPAQ e Information Management 2001 em gerenciamento de pacientes e UTIs.** WinCEBrasil. Publicado em 23/08/2001. Disponível em http://www.wince.com.br/cgi-in/falandodece/03.idc?IDSobreCE=112Acessado em 29/1/2005

EL SHHEIBIA, Tarigi Ali A. **Um Modelo de Monitoração de pacientes na UTI usando Micro Servidor *Web***. 2003. 115 f. Tese (Doutorado em Ciência da Computação) – Universidade Federal de Santa Catarina, Florianópolis.

FALCÃO, Horácio A. ***Home Care* – Uma Alternativa ao Atendimento da Saúde.** Med Online**.** Ed.7, 1999. Disponível em http://www.medonline.com.br/med_ed/med7/homecar.htm. Acessado em 17/01/2005.

FLORENCE, Gerson; **CALIL**, Saide Jorge. **Gerenciamento de Risco Aplicado ao Desempenho de Equipamentos Médicos**. Sociedade Brasileira de Metrologia (SBM). Pernambuco, 2003.

FRANCESCHI, Analúcia S. M. **Aplicação de Técnicas de Inteligência Artificial no Desenvolvimento de Agentes para Gerência de Redes.** 2003. 145 f. Tese (Doutorado em Engenharia Elétrica) – Universidade Federal de Santa Catarina, Florianópolis.

JOB, João C. C.; **SIMÕES**, Marco A. C. **Gerenciamento de Sistemas Baseado no Padrão WBEM**. Revista Eletrônica de Iniciação Científica, SBC - Sociedade Brasileira de Computação, Ano II, N^{o} 4, Dezembro de 2002.

JÚNIOR, Egídio L. **Uma Proposta de Metodologia para Análise de Desempenho de Redes IEEE 802.11 Combinando a Gerência SNMP e Ferramentas de Simulação.** Santa Rita do Sapucaí, 2003. Dissertação (Mestrado em Engenharia Elétrica) – Instituto Nacional de Telecomunicações.

LIMA, **Michele M. A. E..** Gerenciamento de Redes TCP/IP - Boletim bimestral sobre tecnologia de redes produzido pela Rede Nacional de Ensino e Pesquisa (RNP), volume 1, número 7, 12 de dezembro de 1997.

LOPES, Raquel V., **SAUVÉ**, Jacques P., **NICOLLETTI**, Pedro S. **As Melhores Práticas de Gerência de Redes de Computadores**. Ed. Campus, Rio de Janeiro, 2003.

MACHADO, Leonardo H. **Monitorando Redes com o Nagios**.

MARTINS, Ricardo D. e **LOPES**, RUI D. **Estudo de uma Ferramenta de Gestão de Redes**. Portugal, 2003. Estudo (Cadeira de Inteligência e Gestão de Redes e Serviços) – ISCTE.

MENDES, Carlos L., **FERNANDES**, Antonio O. **Um Sistema de Gerenciamento para No-Break Trifásico Engetron**. (Artigo publicado em Maio de 1998).

Oda, Cybelle Suemi. **Introdução à Gerência de Redes**.Disponível em: http://www.gt-er.cg.org.br/operacoes/gerencia-redes 1994.

ORTIZ, R. S. *et al*. **Monitorização de Sinais Biomédicos em Assistentes Pessoais Digitais**. Brasilia, 2004.

PIZARRO. P. J. C. *et al.* **Monitoramento Remoto de Sinais Bioelétricos.** II Congresso Latino Americano de Engenharia Biomédica. Habana, 2001.

RAMALHO, Marina. **Monitoramento de Sistema Hídrico Global ajudará a medir impacto humano sobre clima.** CIÊNCIA HOJE Online. Publicado em 04/05/2002. Disponível em : http://cienciahoje.uol.com.br/controlPanel/materia/view/1758. acessado em 13/07/2005.

RNP – Rede Nacional de Ensino e Pesquisa. **Boletim bimestral sobre tecnologia de redes**. 01 de agosto de 1997. vol. 1, nr. 3. Disponível emhttp://www.rnp.br/newsgen/9708/n3-2.html. Acessado em 23-12-2004.

ROCHA*,* L. F*.,* **BARROS***,* A. P*.* **CSO e a importância da monitoração remota** Disponível em: http://www.nbso.nic.br/docs/reportagens/2003/2003-08-11.html. Acessado em 28/01/2005.

SILVA, Juarez Bento. **Monitoramento, aquisição e controle de sinais elétricos via *web***. 2002. 110 f. Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Santa Catarina, Florianópolis.

SOARES, Luiz F., **SOUZA**, Guido L. e **COLCHER**, Sérgio. **Redes de computadores: das LANs, MANs e WANs às redes ATM**. Ed. Campus, 2º edição, 1995.

SOARES, Luiz F. G., **SOUZA FILHO,** Guido L., e **COLCHER,** Sérgio. **Redes de computadores: das LANs, MANs e WANs às redes ATM**. Campus, 2º edição, 1997.

SOARES, Rivanor P. **Gerenciamento e Monitoração de Redes com o Nagios**. 2004. Disponível em http://br.linuxchix.org/evento/2004/palestras/nagios_linuxchix.pdf. Acessado em 25/06/2005.

SPECIALSKI, Elizabeth Sueli. **Modelo de Informação Baseado em Relacionamentos entre Objetos Gerenciados para a Gerência Integrada de Ambientes de Telecomunicações.** 2000. 140 f. Tese(Engenharia de Produção). Universidade Federal de Santa Catarina, Florianópolis.

STALLINGS, William. **SNMP, SNMPv2, SNMPv3, and RMON 1 and 2** – The Practical Guide to Network-Management Standards. Ed. 3. Addison Wesley, 1999.

TANEMBAUM, Andrew S. **Insight Serviços de Informática – Redes de Computadores**. Ed. 4. Campus. Rio de Janeiro, 1999.

RECH FILHO, Armando. **Estudos para a implantação de uma gerência de rede corporativa utilizando arquitetura de protocolos abertos**. Curitiba, 2004. 147 f. Dissertação (Mestrado em Engenharia Elétrica e Informática Industrial) - Centro Federal de Educação Tecnológica do Paraná.

7 ANEXOS

ANEXO 1

Registradores do AVR AT90S8515.

Endereço	Nome	Função
0x3f	SREG	Registrador de Status
0x3e	SPH	Ponteiro de Pilha alto
0x3d	SPL	Ponteiro de Pilha baixo
0x3b	GIMSK	Máscara de Interrupção geral
0x3a	GIFR	Flags de Interrupção
0x39	TIMSK	Máscara de Interrupções do Timer
0x38	GIFR	Flags do Timer
Endereço	**Nome**	**Função**
0x35	MCUCR	Controle geral do MCU
0x33	TCCR0	Controles do Timer 0
0x32	TCNT0	Timer 0
0x2f	TCCR1A	Controles do Timer 1
0x2e	TCCR1B	Controles do Timer 1
0x2d	TCNT1H	Timer 1 alto
0x2c	TCNT1L	Timer 1 baixo
0x2b	OCR1AH	Timer 1 Output Compare A alto
0x2a	OCR1AL	Timer 1 Output Compare A baixo
0x29	OCR1BH	Timer 1 Output Compare B alto
0x28	OCR1BL	Timer 1 Output Compare B baixo
0x25	ICR1H	Timer 1 Input Capture alto
0x24	ICR1L	Timer 1 Input Capture baixo
0x21	WDTCR	WatchDog Controls
0x1f	EEARH	Endereço EEPROM alto
0x1e	EEARL	Endereço EEPROM baixo
0x1d	EEDR	Dado EEPROM
0x1c	EECR	Controles EEPROM
0x1b	PORTA	Latch de saída
0x1a	DDRA	Direção do Dado
0x19	PINA	Leitura
0x18	PORTB	Latch de saída
0x17	DDRB	Direção do Dado
0x16	PINB	Leitura
0x15	PORTC	Latch de saída
0x14	DDRC	Direção do Dado
0x13	PINC	Leitura
0x12	PORTD	Latch de saída
0x11	DDRD	Direção do Dado
0x10	PIND	Leitura
0x0f	SPDR	Dado SPI
0x0e	SPSR	Status SPI
0x0d	SPCR	Controles SPI
0x0c	UDR	Dado UART
0x0b	USR	Status UART
0x0a	UCR	Controles da UART
0x09	UBRR	Baud Rate UART
0x08	ACSR	Status Comparador Analógico

Fonte: Silva (2002).

ANEXO 2

Este é o arquivo *send_modem.sh* (Cassaro, 2005).

```
#!/bin/bash

. /etc/mailmodem.conf

file="`date +%s%N`"

if [ "$#" != "4" ]; then
  echo "Wrong number of paramets"
  echo "Usage: $0 tel_number sound1 sound2 sound3"
  exit 255
fi

# Telefone
TEL="$1"
# Nome do Local (Hospital)
P1="$2"
MSG1="`echo $P1 | awk '{print tolower($1)}'`"

# Objeto da falha (Paciente ou MSW)
P2="$3"
MSG2="`echo $P2 | awk '{print tolower($0)}'`"

# Leito da UTI
P3="$4"
MSG3="`echo $P3 | tr A-Z a-z`"

test -e $spool/$file || echo -n "$TEL:$MSG1.rmd:$MSG2.rmd:$MSG3.rmd" >
$spool/$file
chmod 666 $spool/$file

exit 0
```

ANEXO 3

Este é o *deamon* responsável em vasculhar o diretório *spool/mailmodem*, configurado no arquivo *mailmoden.conf*, em busca dos arquivos de som que representam a notificação a ser encaminhada (Cassaro, 2005)..

```
#!/bin/bash

. /etc/mailmodem.conf

DELAY="1m"

while :; do
for message in `find $spool -type f | sort`; do
   CONT="`cat $message`"
   TEL="`echo $CONT | cut -f1 -d:`"
   MSG1="`echo $CONT | cut -f2 -d:`"
   MSG2="`echo $CONT | cut -f3 -d:`"
   MSG3="`echo $CONT | cut -f4 -d:`"
   if [ -z "`ps ax | grep playmodem | grep -v grep`" ]; then
     $playmodem $TEL $MSG1 $MSG2 $MSG3
     ERROR="$?"
     rm $message
   fi
   if [ "$ERROR" == "0" ]; then
     /usr/bin/logger -t MailModem "Send |Tel: $TEL - MSG1: $MSG1 - MSG2:
$MSG2 - MSG3: $MSG3| Success"
   else
     /usr/bin/logger -t MailModem "Send |Tel: $TEL - MSG1: $MSG1 - MSG2:
$MSG2 - MSG3: $MSG3| Failed, errnum: $ERROR"
   fi
   sleep 1s
done
  sleep $DELAY
done

exit 0
```

ANEXO 4

Este é o arquivo responsável em acionar o script do arquivo *messages.sh* para tocar os três arquivos de som que representam a mensagem de notificação (Cassaro, 2005).

```
#!/bin/bash

TEL="$1"
SOM1="$2"
SOM2="$3"
SOM3="$4"
PREFIX="/usr/local"
DIRSOM="$PREFIX/share/sounds"

err=`sudo -u root $PREFIX/bin/vm shell -l cua0 $PREFIX/bin/message.sh $TEL
$DIRSOM/$SOM1 $DIRSOM/$SOM2 $DIRSOM/$SOM3`

if [ "$err" == "OK: message sent" ];then
  out="0"
  mess="OK"
else
  out="255"
  mess="ERROR"
fi

echo "$mess"
exit $out
```

ANEXO 5

Este arquivo recebe um número de telefone e toca uma mensagem (Cassaro, 2005).

```
#
# This script calls the given phone number and plays a message.
#
# $1 - phone number to call
# $2 - filename of the message to play (must be a .rmd file, that
#      can be played on the modem used for dialout)
#
# $Id: message.sh,v 1.5 1999/12/04 15:07:34 marcs Exp $
#
#
# Define the function to receive an answer from the voice library
#

function receive
    {
    read -r INPUT <&$VOICE_INPUT;
    echo "$INPUT";
    }

#
# Define the function to send a command to the voice library
#

function send
    {
    echo $1 >&$VOICE_OUTPUT;
    kill -PIPE $VOICE_PID
    }

#
# Check command line options
#

if [ $# -ne 4 ]; then
    echo "usage: $0 <phone_number> <filename1> <filename2> <filename3>"
>&2
    exit 99
fi

#
# Let's see if the voice library is talking to us
#

ANSWER=`receive`
```

```
if [ "$ANSWER" != "HELLO SHELL" ]; then
    kill -KILL $$
fi

send "HELLO VOICE PROGRAM"

ANSWER=`receive`

if [ "$ANSWER" != "READY" ]; then
    kill -KILL $$
fi

#
# Enable events
#

send "ENABLE EVENTS"

ANSWER=`receive`

if [ "$ANSWER" != "READY" ]; then
    kill -KILL $$
fi
#
# Start dialout
#

send "DIAL $1"

ANSWER=`receive`

if [ "$ANSWER" != "DIALING" ]; then
    kill -KILL $$
fi

ANSWER=`receive`

if [ "$ANSWER" != "READY" ]; then
    echo "ERROR: $ANSWER, aborting"
    exit 99
fi

#
# Disable events
#

send "DISABLE EVENTS"

ANSWER=`receive`
```

```
if [ "$ANSWER" != "READY" ]; then
    kill -KILL $$
fi

#
# Now play the message file
#

send "PLAY $2"

ANSWER=`receive`

if [ "$ANSWER" != "PLAYING" ]; then
    kill -KILL $$
fi

ANSWER=`receive`

if [ "$ANSWER" != "READY" ]; then
    kill -KILL $$
fi

send "PLAY $3"

ANSWER=`receive`

if [ "$ANSWER" != "PLAYING" ]; then
    kill -KILL $$
fi

ANSWER=`receive`

if [ "$ANSWER" != "READY" ]; then
    kill -KILL $$
fi

send "PLAY $4"

ANSWER=`receive`


if [ "$ANSWER" != "PLAYING" ]; then
    kill -KILL $$
fi

ANSWER=`receive`

if [ "$ANSWER" != "READY" ]; then
```

```
    kill -KILL $$
fi



#
# Let's say goodbye
#

send "GOODBYE"

ANSWER=`receive`

if [ "$ANSWER" != "GOODBYE SHELL" ]; then
    kill -KILL $$
fi

echo "OK: message sent"
exit 0
```

ANEXO 6

Data: Thu, 28 Jul 2005 13:34:49 -0300 [28/07/2005 13:34:49 BRST]
De: dpaula@unisul.br
Para: gmsouza@inf.ufsc.br
Assunto: Re: URGENTE!!
Prioridade: normal
Cabeçalhos: Exibir Todos os Cabeçalhos

Oi Giba, te telefonei na segunda a tarde, mas você não estava.

On 27 Jul 2005 at 16:06, gmsouza@inf.ufsc.br wrote:
> Olá Maurício!
> Gostaria que respondesse a algumas perguntas para que eu pudesse referenciar no
> trabalho, ok?
> -> Quando o MSW trava, existe alguma forma de saber o que realmente aconteceu
> para ele travar?

Atualmente fizemos suposições. Procuramos identificar o problema da camada física (1)
até a de aplicação (7).
> -> Existe alguma variável que poderíamos monitorar que pudesse auxiliar na
> dectecção do problema?

Giba, normalmente monitoramos as portas de I/O e o registrador de status.
> ->Seria interessante monitorar o processamento, a memória e I/O? Por quê?

Não sei se é possível na arquitetura da picoWeb monitorar o processamento do microcontrolador, mas a memória e as portas de I/O sim.
> -> Vc acha interessante realizar um estudo mais profundo sobre estas questões?

Depende do que definisse nos objetivos do trabalho.
> Se puder meu amigo, responda ainda hj, no mais tardar amanhã, ok?
>
> []s
> Giba
>

Não respondi antes porque estava a dois dias em reunião.

abraços,
Maurício.
(48) 521-3058

Data:	Fri, 8 Jul 2005 15:09:32 -0300 [08/07/2005 15:09:32 BRST]
De:	Allan Cassaro <allan.cassaro@gmail.com>
Para:	gmsouza@inf.ufsc.br
Endereço para Resposta (Reply-To):	Allan Cassaro <allan.cassaro@gmail.com>
Assunto:	Re: [Viva o Linux] Nágios com FreeBSD
Cabeçalhos:	Exibir Todos os Cabeçalhos

```
Corretissimo!
O nagios funciona em qualquer *nix.
No caso do nagios falante, basta instalar os mesmos pacotes.

Abraços.

On 7/8/05, Suporte Viva o Linux <suporte@vivaolinux.com.br> wrote:
```
[Ocultar Texto Citado]
```
>
>
> A seguinte mensagem foi enviada para voce atraves do site
> Viva o Linux.com.br:
>
> Login: gibams
> Nome: Gilberto Medeiros de Souza
> E-mail: gmsouza@inf.ufsc.br
>
> Assunto: Nágios com FreeBSD
> ------------------------------------------------------------
>
> Olá! Parabéns pelo artigo!
> Acredito que posso tentar realizar este trabalho no BSD tb,
correto?
> []s
>
```