Suporte para Monitoramento e Controle de Carga em Sistemas Distribuídos

Rodrigo Cardoso Uchôa *Noemi de La Rocque Rodriguez*

Departamento de Informática
Pontificia Universidade Católica
Rio de Janeiro, RJ
E-mail: rodrigo,noemi@inf.puc-rio.br

Resumo

Este trabalho discute a utilização do modelo de gerência como base para provisão de informações sobre o estado de utilização de recursos em um sistema distribuído. Propõe-se uma extensão à base de informações gerenciais (MIB) específica para a manutenção de informações de carga do sistema. Além disto, descreve-se uma biblioteca orientada a objetos, que oferece uma interface de alto nível independente de protocolos, para a programação de aplicações gerentes. Finalmente, apresenta-se um exemplo de utilização dessa biblioteca que, em conjunto com a ferramenta de programação distribuída PVM, permite o melhor aproveitamento dos recursos ociosos de processamento.

Abstract

This paper discusses the use of the network management framework as a basis for the provision of information about the state of resources in a distributed system. An extension to the management information base (MIB) is proposed in order to deal specifically with system load information. Furthermore, the paper describes an object oriented library which provides a high level, protocol independent interface for programming management applications. Finally, an example is presented, where the programming tool PVM is coupled with the management library to allow a better use of idle resources in the system.

1. Introdução

Atualmente, estamos convivendo com um crescente processo de transformação de ambientes computacioniais centralizados em ambientes distribuídos. Esta transformação é motivada por um conjunto de vantagens que incluem menor custo, maior poder computacional global e maior confiabilidade. No entanto, as tecnologias utilizadas hoje ainda não permitem que estas vantagens sejam efetivamente obtidas. O estado atual de integração dos sistemas distribuídos impede que os usuários alcancem os resultados esperados. Na maior parte dos casos, são oferecidos mecanismos como servidores de arquivos e impressão, que permitem apenas o compartilhamento de discos e impressoras, mas basicamente o sistema é formado por unidades independentes de processamento.

A área de gerência de redes foi inicialmente impulsionada pela necessidade de monitorar e controlar os dispositivos componentes das redes de comunicação. O modelo utilizado para gerência baseia-se em um paradigma de comunicação cliente-servidor, onde cada

dispositivo gerenciado é controlado localmente por um *agente* (servidor de informações gerenciais), e uma aplicação *gerente* (cliente) que se comunica com os agentes adequados. Atualmente, investiga-se a utilização deste modelo para o controle de qualquer componente de hardware ou software. A flexibilidade no uso que a aplicação gerente pode fazer das informações obtidas pelos agentes abre possibilidades de soluções para o problema de falta de integração citado anteriormente.

Entre os diversos protocolos de gerência propostos, o SNMP [RFC1157, RFC1442] tem até hoje a maior divulgação e uso. Como em muitos outros protocolos de gerência, os serviços oferecidos pelo SNMP para criação de aplicações gerentes são bastante primitivos, consistindo apenas de um conjunto de operações básicas do tipo *GET* e *SET*.

Neste trabalho, investigamos a utilização do modelo de gerência para o monitoramento e controle dos recursos das diversas estações em um sistema distribuído, em particular, uma rede de estações Sun executando SunOS 4.1.1. Para isto, foi definida uma extensão da base de informações gerenciais (MIB), chamada HLMMIB (*Host Load Metrics MIB*), que engloba medidas do uso de CPU, memória, etc. Isto fornece suporte, inexistente na maior parte dos sistemas operacionais de amplo uso, para a manutenção de uma visão global da utilização dos recursos do sistema. Tal visão pode ser usada para decisões de configuração, como base para avaliação de expansões necessárias, ou ainda como suporte para um mecanismo de balanceamento de carga.

Baseado na HLMMIB, também foi definida uma biblioteca de alto nível para criação de aplicações de gerência, denominada HLMLIB. Esta biblioteca, escrita em C++, é composta por classes que modelam os dispositivos remotos gerenciados. A provisão desta interface de alto nível orientada a objetos torna a programação de aplicações gerentes totalmente independente do protocolo de gerência utilizado. A mesma biblioteca poderia ser construída utilizando como base outro protocolo, por exemplo, o CMIP [IS9596, RFC1095] definido e adotado como padrão pela OSI. Discute-se ainda a implementação de uma aplicação gerente específica, que utiliza as informações de utilização de recursos para obter melhor aproveitamento dos recursos ociosos na rede.

O trabalho é organizado da seguinte forma. A próxima seção apresenta o modelo de gerência, discutindo sua arquitetura e, resumidamente, o protocolo SNMP. A seção 3 descreve a implementação de um agente SNMP para monitoramento e controle de carga. A biblioteca HLMLIB é discutida na seção 4. A seção 5 discute a aplicação gerente exemplo. Finalmente, a seção 6 apresenta as conclusões do trabalho.

2. Modelos de Gerência

Atualmente, as redes de computadores estão se tornando parte fundamental dos sistemas de informação [STA93]. É cada vez mais importante mantê-las em operação, pois cada hora parada representa grande prejuízo. O crescimento exponencial das redes e dos dispositivos de comunicação, formando ambientes altamente heterogêneos, exige mecanismos de gerência padronizados, cada vez mais sofisticados e automatizados [ROS93].

Todo processo de gerenciamento é dividido em três etapas básicas: monitoramento, análise e controle.

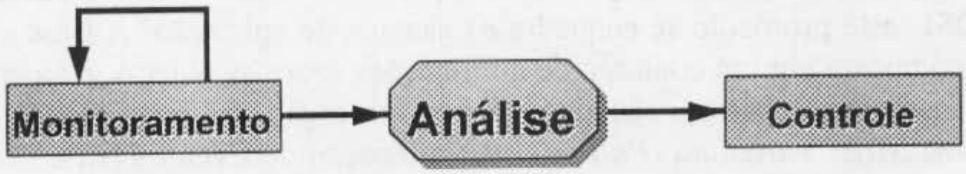

Figura 1 - Etapas de um processo gerencial

Durante o monitoramento (contínuo) é obtido um conjunto de informações sobre o estado do objeto gerenciado; a partir daí é realizada uma análise dessas informações e, caso seja necessário, inicia-se uma etapa de controle, onde são realizadas ações que permitam a transformação do estado atual do objeto.

A OSI definiu cinco grandes áreas funcionais de gerência, sendo elas: Gerência de Falhas, Gerência de Contabilização, Gerência de Configuração, Gerência de Performance e Gerência de Segurança.

Devido à grande necessidade de gerência em ambientes heterogêneos, tornou-se fundamental a definição de padrões, que pudessem ser adotados por todos os fabricantes de equipamentos de comunicação e desenvolvedores de *softwares* de gerência.

Alguns modelos de gerência distribuída foram definidos e padronizados, inicialmente, com a finalidade de permitir o monitoramento e controle apenas dos dispositivos e protocolos de comunicação. Atualmente, estes padrões estão sendo estendidos, possibilitando o gerenciamento de qualquer tipo de equipamento de hardware ou sistemas de software, que compõem todo o ambiente distribuído, por exemplo: estações de trabalho, impressoras, sistemas de correio eletrônico, etc. O ideal é que todos os componentes sejam gerenciados utilizando-se os mesmos mecanismos e protocolos de gerência.

2.1 Arquitetura Gerente/Agente

Os modelos de gerência definidos atualmente seguem a arquitetura Gerente/Agente, ilustrada na Figura 2. Esta arquitetura é composta por quatro elementos básicos: Gerente, Agente, Protocolo de Gerenciamento e Base de Informações Gerenciais (MIB).

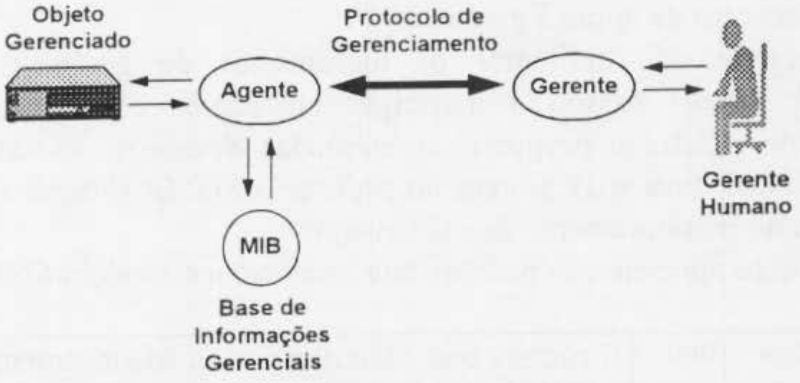

Figura 2 - Elementos básicos do Modelo de Gerência

O gerente é responsável pela coleta de informações, oferecendo ao gerente humano ferramentas de monitoramento e controle do objeto gerenciado. A comunicação entre Gerente e Agente é realizada através de um protocolo de gerenciamento, que define a sintaxe e

semântica das unidades de informação, PDUs (*Protocol Data Unit*), trocadas. Segundo a arquitetura OSI este protocolo se enquadra na camada de aplicação. A Base de Informações Gerenciais é composta por um conjunto de informações sobre os objetos gerenciados.

Dois mecanismos básicos são utilizados na obtenção das informações mantidas, pelo agente, em uma MIB: Varredura (*Polling*) ou Notificação de Eventos [RFC1215] (Figura 3). No mecanismo de *polling* o gerente realiza uma comunicação com o agente utilizando um esquema síncrono *Request/Response*. A principal desvantagem do mecanismo de *polling* é o alto tráfego gerado, pois deve haver um monitoramento periódico dos diversos agentes. Uma maneira de se resolver este problema é utilizar o mecanismo de notificação de eventos. Sempre que um agente detecta algum evento interessante é iniciada uma comunicação com o gerente, notificando-o.

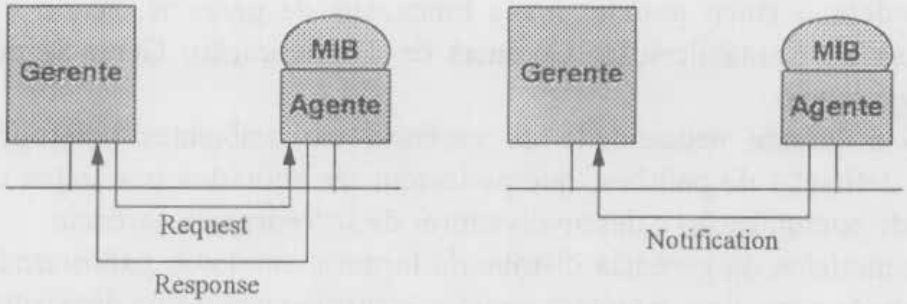

Figura 3 - Modelos de comunicação entre gerente e agente

2.2 SNMP - Simple Network Management Protocol

Inicialmente, o único mecanismo que permitia monitorar o comportamento de inter-redes TCP/IP era o protocolo ICMP (*Internet Control Message Protocol*). Utilizando este protocolo e algumas opções do IP, foram implementados alguns utilitários de auxílio ao administrador da rede, tais como: PING e TRACEROUTE. O utilitário PING permite verificar se existe uma rota operacional até uma determinada estação, se a própria estação está operacional e determinar qual o retardo na comunicação. TRACEROUTE permite descobrir qual rota os datagramas IP estão percorrendo na comunicação com uma determinada estação na inter-rede. Ainda hoje, muitos administradores de rede utilizam apenas estes utilitários básicos como ferramentas de apoio à gerência.

Com o objetivo de aprimorar os mecanismos de gerência, o IETF (*Internet Engineering Task Force*) passou a investigar protocolos especializados para este fim [RFC1157, RFC1095]. Entre as propostas apresentadas, decidiu-se adotar o protocolo SNMP como padrão. O SNMP teve suas origens no protocolo SGMP (*Simple Gateway Monitoring Protocol*) utilizado no monitoramento dos *Gateways*.

A tabela abaixo apresenta os padrões que descrevem o modelo SNMP.

RFC 1155	Maio 1990	Structure and Identification of Management Information for TCP/IP based Internets
RFC 1157	Maio 1990	A Simple Network Management Protocol (SNMP)
RFC 1213	Março 1991	Management Information Base for Network Management of TCP/IP based internets MIB-II

O SNMP segue o modelo Gerente/Agente descrito anteriormente. Dos elementos básicos, apenas o protocolo de comunicação e a MIB são padronizados. O RFC1155 padroniza o mecanismo de descrição das informações gerenciais. O protocolo de comunicação entre gerente e agente é definido pelo RFC 1157. A MIB-II, definida no RFC 1213, é o conjunto básico de variáveis utilizadas no gerenciamento de redes TCP/IP.

2.2.1 Protocolo de Gerenciamento (SNMP)

O protocolo SNMP é o elo de ligação entre os processos gerente e agente, formando uma aplicação distribuída. Este protocolo da camada de aplicação utiliza os serviços de comunicação oferecidos por protocolos de camadas inferiores. O padrão define a utilização dos serviços UDP (*User Datagram Protocol*) como mecanismos de transporte, mas nada impede a utilização de outros protocolos. É importante notarmos que, tanto o protocolo UDP, como o protocolo SNMP, definem apenas serviços não confiáveis sem conexão. A introdução de mecanismos de controle de perda ou duplicação de mensagens SNMP é uma decisão de implementação das aplicações que utilizam este protocolo.

O protocolo SNMP ofereçe apenas quatro operações básicas: GET, GET-NEXT, SET e TRAP. O monitoramento é realizado utilizando as operações GET e GET-NEXT. A operação SET permite o controle remoto do objeto gerenciado (alteração de estado). O objetivo da operação TRAP é permitir que o agente informe eventos ao gerente de maneira assíncrona.

O padrão define cinco mensagens SNMP (PDUs) diferentes: GetRequest, GetNextRequest, SetRequest, GetResponse e Trap. A relação entre os serviços definidos e a troca de mensagens necessária é ilustrada na Figura 4.

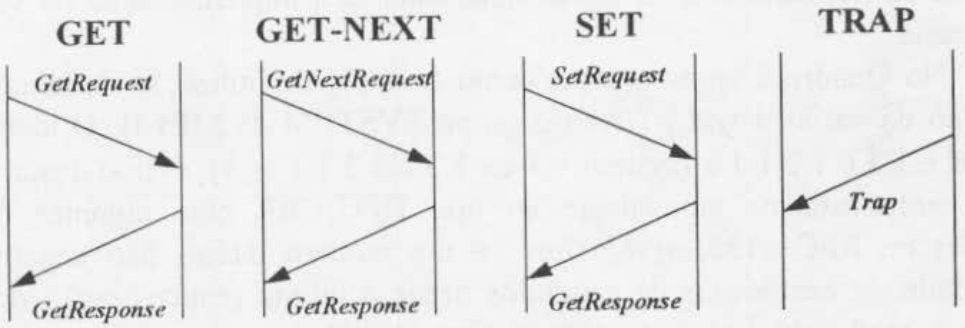

Figura 4 - Serviços SNMP

2.2.2 Estrutura de Informações Gerenciais (SMI)

Os objetos gerenciados são modelados como um conjunto de variáveis definidas formalmente através de uma subparte da linguagem ASN.1 (*Abstract Syntax Notation One*) [NEU92], denominada SMI (*Structure of Management Information*). O conjunto de todas as variáveis padronizadas é denominado MIB (*Management Information Base*). A macro ASN.1 OBJECT-TYPE (Quadro 1), definida no RFC 1155, é utilizada na definição das variáveis SNMP.

A SMI permite a definição apenas de variáveis atômicas ou tabelas compostas de variáveis atômicas. Apesar da possibilidade de definição de tabelas, o protocolo SNMP só oferece mecanismos de recuperação e modificação de variáveis atômicas. O acesso às tabelas é realizado através de vários acessos aos seus componentes. Vários mecanismos de acesso às tabelas como uma unidade de dados estão em processo de padronização, como por exemplo, a

operação GET-BULK oferecida pela pelo protocolo SNMPv2, sucessor do SNMP, ainda em processo de padronização e aceitação.

```
OBJECT-TYPE MACRO ::=
        BEGIN
          TYPE NOTATION ::= "SYNTAX" type (TYPE ObjectSyntax)
                            "ACCESS" Access
                            "STATUS" Status
          VALUE NOTATION ::= value (VALUE ObjectName)

          Access ::= "read-only"
                   | "read-write"
                   | "write-only"
                   | "not-accessible"
          Status ::= "mandatory"
                   | "optional"
                   | "obsolete"
        END
```

Quadro 1 - Macro ASN.1 OBJECT-TYPE

Cada variável definida possui um identificador único, que é utilizado em toda operação SNMP. Esse identificador é do tipo ASN.1 OBJECT IDENTIFIER. A macro OBJECT-TYPE também permite definir o tipo da variável, restrições de acesso e status. Os únicos tipos ASN.1 permitidos são: INTEGER, OCTET STRING, OBJECT IDENTIFIER, SEQUENCE, SEQUENCE OF e NULL. Este conjunto limitado de tipos simplifica bastante a implementação dos agentes. Uma variável pode ser apenas de leitura, apenas de escrita, leitura e escrita ou não acessível. O status determina se a implementação da variável é opcional ou obrigatória.

No Quadro 2 apresentamos como exemplo de utilização da macro OBJECT-TYPE, a definição da variável *sysUpTime* do grupo SYSTEM da MIB-II. O identificador único desta variável é 1.3.6.1.2.1.1.3 (system + 3 ⇔ 1.3.6.1.2.1.1 + .3). O tipo desta variável é *TimeTicks*, que é sintaticamente equivalente ao tipo INTEGER com algumas restrições semânticas definidas no RFC 1155. *sysUpTime* é um número inteiro não negativo que representa a quantidade de centésimos de segundos desde a última reinicialização do sistema. O tipo de acesso é read-only, ou seja, apenas para leitura e o status é mandatory, ou seja, a sua implementação é obrigatória.

```
sysUpTime OBJECT-TYPE
        SYNTAX  TimeTicks
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
            "The time (in hundredths of a second) since the
            network management portion of the system was last
            re-initialized."
        ::= { system 3 }
```

Quadro 2 - Definição de uma variável SNMP.

2.2.3 Base de Informações Gerenciais (MIB)

A MIB é um banco de dados ativo cujo esquema é definido utilizando-se SMI, como descrito na seção anterior. Cada agente mantém sua própria instância da MIB relacionada com

os dispositivos gerenciados em seu domínio. O RFC 1213 define a MIB-II, que é um conjunto padrão de variáveis, utilizadas no monitoramento e controle de redes baseadas nos protocolos TCP/IP. Estas variáveis são organizadas em diversos grupos. Cada grupo é formado por um conjunto de variáveis relacionadas, que juntas representam as informações necessárias para o gerenciamento de algum dispositivo ou protocolo. A Tabela 1 mostra os grupos definidos na MIB-II.

Grupos	Tipo de Informações oferecidas
SYSTEM	Genéricas sobre o sistema.
INTERFACES	Interfaces de rede e suas características.
AT (*Address Translation*)	Mapeamentos entre endereços físicos e endereços lógicos de rede, conhecidos pela estação (Ex.: Ethernet ⇔ IP).
TRANSMISSION	Características e estatísticas específicas de cada tipo de interface de rede (Ex.: Ethernet, Token Ring, FDDI, X.25, ...).
IP, ICMP, TCP, UDP, EGP e SNMP	Opções de implementação e estatísticas sobre o comportamento dos protocolos.

Tabela 1 - Grupos definidos na MIB-II

Para cada novo dispositivo a ser gerenciado, deve ser definido um conjunto de novas variáveis necessárias ao monitoramento e controle desse dispositivo. Atualmente existe uma série de propostas de extensão da MIB-II que permitem gerenciar dispositivos de comunicação (por exemplo: *Hubs*, roteadores, *bridges*, modems) ou outras pilhas de protocolos (por exemplo: X.25 [RFC1382], IPX/SPX, ATM).

3. Agente SNMP

A implementação de um agente SNMP requer a codificação de métodos para recuperação e alteração das variáveis pertencentes à base de informações gerenciais conhecida pelo agente.

Alguns protocolos foram definidos com a finalidade de permitir que agentes sejam facilmente estendidos, permitindo que partes da MIB sejam implementadas por outros processos, chamados sub-agentes ou processos peer. Utilizando algum destes protocolos o processo sub-agente se registra junto ao agente, local ou remoto, informando que é responsável por uma parte da MIB, ou seja, um conjunto de variáveis. Quando requisições provenientes de gerentes envolvendo alguma destas variáveis é recebido pelo agente, este repassa para o sub-agente a requisição recebida. O sub-agente executa a operação desejada, devolvendo ao agente o resultado da operação. Finalizando, o agente transmite ao gerente o resultado da requisição recebida inicialmente.

SMUX (*SNMP Multiplexer*) [RFC1227] e SNMP-DPI (*SNMP Distributed Program Interface*) [RFC1228] são dois exemplos destes protocolos. Além de permitir a extensão do agente SNMP, a utilização de agentes que implementem SMUX ou SNMP-DPI permite o desenvolvimento de aplicações que possam ser monitoradas e controladas via protocolo de gerência.

Parte do trabalho desenvolvido foi estender um agente SNMP, implementando uma extensão da MIB definida especialmente para o monitoramento de carga das estações, dando suporte para a implementação de uma aplicação gerente de balanceamento de carga.

Resolvemos adotar o pacote SNMP da Universidade de Carnegie Mellon (CMU-SNMP) devido à sua disponibilidade e ampla utilização. Este pacote inclui um agente SNMP MIB-II que não implementa nenhum mecanismo de extensão, como SMUX ou SNMP-DPI; a única possibilidade, portanto, foi estender o código fonte do agente, incluindo a extensão da MIB definida.

3.1 HLMMIB: HOST LOAD METRICS MIB

Antes de partirmos para a definição de uma MIB proprietária, investigamos a possibilidade de utilização de alguma já existente, que pudesse ser utilizada para o monitoramento de carga. Infelizmente não encontramos nenhuma MIB definida para este propósito.

Uma possibilidade seria a utilização da *HOST RESOURCES MIB* [RFC1514], definida para permitir o monitoramento e controle de estações de trabalho. O grupo que definiu a *HOST RESOURCES MIB* não teve a preocupação de definir e padronizar variáveis específicas para o monitoramento de carga (gerência de performance), pois o objetivo principal era apenas a gerência de configuração e falhas de uma estação e seus periféricos.

O apêndice A apresenta uma parte da definição de uma MIB proprietária, denominada HLMMIB, que reune variáveis definidas especialmente para o processo de monitoramento de carga. Esta MIB define grupos de variáveis relacionados com a CPU, memória fisica, memória virtual, interfaces de rede e unidades de armazenamento. Cada grupo deve conter características de arquitetura, parâmetros de configuração e estatísticas de utilização.

3.2 Obtenção das informações de carga

O sistema operacional UNIX mantém estatísticas sobre o comportamento e utilização dos recursos de hardware como CPU, discos, memória, rede, etc. Estas estatísticas podem ser usadas como medida de ociosidade da estação e servir como base para um mecanismo de balanceamento de carga.

Exemplos de estatísticas relevantes são:

- número de tarefas na fila de processos prontos (a espera da CPU);
- quantidade de memória livre;
- taxa de *page faults*;
- taxa de chamadas ao sistema;
- taxa de trocas de contexto;
- tempo de CPU livre durante um intervalo *X*;
- taxa de acesso ao disco.

O sistema BSD UNIX [LEF89] permite o acesso às estatísticas mantidas através do dispositivo virtual /dev/kmem, que oferece uma imagem da memória do núcleo do sistema operacional. O Quadro 3 apresenta um programa exemplo que acessa o *kernel* do sistema através da API oferecida pela biblioteca libkvm.a, que facilita a utilização do mecanismo especial */dev/kmem*.

No programa exemplo do Quadro 3, o valor da variável *freemem* é recuperado utilizando-se a função *kvm_read()*. Esta variável é uma indicação da quantidade total de memórial real disponível.

As informações necessárias para a imlementação do agente SNMP HLMMIB são obtidas, acessando diretamente o núcleo do sistema operacional através do mecanismo discutido nesta seção.

```
/*  Programa exemplo de acesso ao núcleo do Sistema Operacional.  */
/* Plataforma: SunOS 4.1.1                                        */

#include <stdio.h>
#include <kvm.h>
#include <nlist.h>
#include <fcntl.h>
#include <sys/vmsystm.h>

static kvm_t        *kvm_des;
static struct nlist nl[] = {
  { "_freemem" },   /* variavel desejada */
  { "" },
};

main(argc, argv)
  int argc; char **argv;
{
  char *symbol;
  unsigned long value;
  int freemem, ret;

   /* Abre dispositivos virtuais de acesso ao núcleo */
  kvm_des = kvm_open("/vmunix", "/dev/kmem", (char *)NULL, O_RDONLY, argv[0]);

  /* Recupera posição dos nomes especificados na estrutura nl */
  ret = kvm_nlist(kvm_des, nl);

  /*  Le o núcleo do Sistema Operacional */
  symbol = nl[0].n_name;
  value  = nl[0].n_value;
  ret = kvm_read(kvm_des, value, &freemem, sizeof(freemem));
  printf("Quantidade de memória livre = %d\n", freemem);
  exit (0);
}
```

Quadro 3 - Programa exemplo de acesso ao núcleo do sistema operacional.

4. HLMLIB - *Host Load Metrics LIBrary*

Como consequência da simplicidade do protocolo SNMP, as operações oferecidas por bibliotecas SNMP [TOG93] são muito básicas. Desenvolver aplicações gerentes utilizando apenas uma API SNMP é um trabalho complexo, onde uma série de detalhes devem ser conhecidos, tais como: OBJECT IDENTIFIERS das variáveis, mecanismos de autenticação (COMMUNITY NAMES, MIB VIEW), etc.

Uma maneira de isolar as aplicações gerentes dos detalhes dos protocolos de gerência é definir uma camada intermediária, que ofereça uma interface de mais alto nível. Neste trabalho, estamos propondo uma biblioteca de auxílio à construção de aplicações gerentes (HLMLIB) que necessitem de informações relacionadas com medidas de carga das estações, em um ambiente distribuído.

A biblioteca HLMLIB implementa um conjunto de classes que modelam os diversos componentes de uma estação (Memória, CPU, Disco, ...). A Figura 5 apresenta a inclusão desta biblioteca na arquitetura de gerência SNMP. Uma aplicação gerente utiliza esta biblioteca através de sua interface para programação de aplicações, denominada HLMAPI.

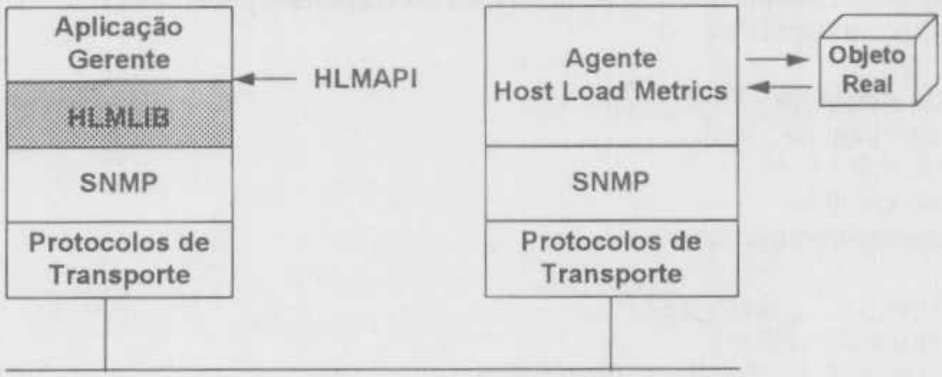

Figura 5 - Biblioteca HLMLIB

Quando uma aplicação cria uma instância de uma classe (chamada de *dispositivo virtual*) e interage com ela através de invocações de métodos, estas interações são refletidas no dispositivo real remoto utilizando-se o protocolo de gerência SNMP.

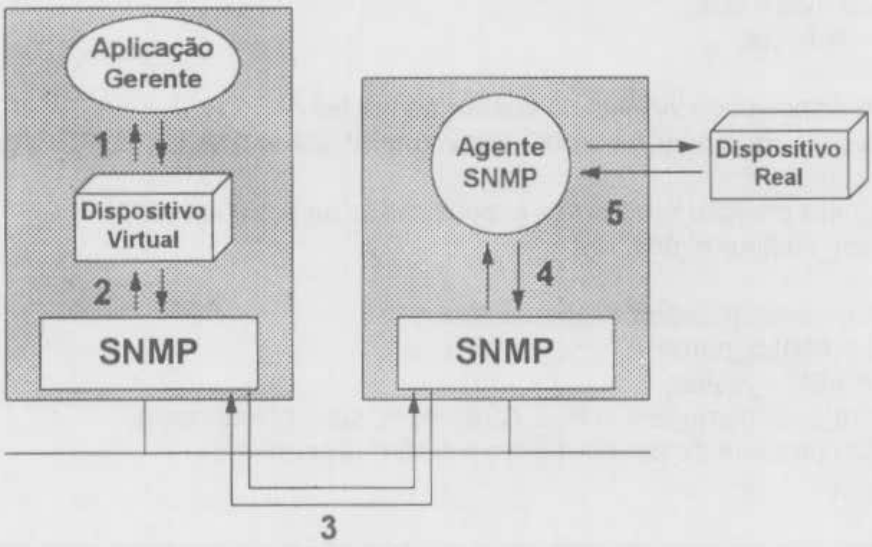

Figura 6 - Gerenciamento remoto através de dispositivos virtuais

1. **Invocação de métodos da classe que modela o dispositivo real**
2. **Chamada a rotinas da biblioteca CMU-SNMP libsnmp.a**
3. **Troca de mensagens SNMP**
4. **Chamada a rotinas da biblioteca CMU-SNMP libsnmp.a**
5. **Chamada a rotinas que implementam a MIB**

Os dispositivos virtuais funcionam de maneira análoga aos *stubs* do mecanismo de chamada remota de procedimentos (*RPC*). A Figura 6 ilustra as etapas do processo de gerência utilizando o modelo proposto.

4.1 Interface HLMAPI

No quadro abaixo apresentamos uma parte da interface para programação de aplicações gerentes implementada. Os diversos dispositivos são modelados utilizando-se classes C++. A classe Host modela uma estação completa com todos os seus dispositivos.

```
Class CPU {
   public:
      CPU();
      ~CPU();
      GetCPUFree();
      GetNumProcRunning();
      GetCPUType();
};

Class Memory {
   public:
      Memory();
      ~Memory();
      GetPhyMemory();
      GetFreeMemory();
};

Class Disk {
   public:
      Disk();
      ~Disk();
      IsDiskLess();
      GetNumDiskIO();
      GetDiskFree();
};

Class Host {
   public:
      Host(char *);
      ~Host();
      CPU cpu;
      Memory memory;
      Disk disk;
};
```

Quadro 4 - Interface para programação de aplicações gerentes.

O programa seguinte é um exemplo de uma aplicação gerente extremamente simples que utiliza a biblioteca HLMLIB.

```
void main(int argc, char **argv) {

  Host *host;
  char *estacao = argv[1];
  int nump;

  host = new Host(estacao);
  nump = host->cpu.GetNumProcRunning();
  cout << nump << " processos rodando na estação " << estacao;
}
```

Quadro 5 - Programa exemplo de utilização da interface definida.

O exemplo ilustra o modelo discutido nesta seção. A variável host representa o dispositivo virtual do tipo Host, que corresponde a um dispositivo real controlado remotamente. É importante notar que a aplicação fica totalmente independente do modelo de gerência SNMP, encapsulado nos métodos das classes definidas.

5. Uma aplicação: PVM + Balanceamento

Nesta seção apresentamos uma aplicação específica da biblioteca de gerência de recursos HLMLIB. Esta aplicação visa o aproveitamento de recursos ociosos em uma rede de estações. Na implementação corrente, utilizamos estações Sun executando o sistema operacional SunOS 4.1.1.

O problema da melhor utilização de recursos em um sistema distribuído recai na questão de balanceamento de carga, discutida na seção 5.1. A seção seguinte discute a ferramenta de programação PVM, escolhida como mecanismo de suporte para programação de aplicações, que fazem uso da capacidade de processamento ociosa distribuída pela rede. Finalmente, a seção 5.3 descreve a arquitetura utilizada para a aplicação de gerência.

5.1 Balanceamento de Carga

O objetivo do Balanceamento de Carga [KAR94a, KAR94b, KUN91, OSS92] é melhor utilizar o ambiente distribuído, permitindo que estações ociosas possam ser utilizadas em benefício de estações sobrecarregadas. Estudos realizados provaram que um grande número de estações ficam a maior parte do tempo ociosas. Como exemplo mais concreto, podemos citar algumas medições (Tabela 2) realizadas na rede local do Laboratório do Departamento de Informática da PUC-RIO, composta por estações de trabalho Sun.

Estação	Tempo total desde o último boot	Tempo ocioso	Tempo útil
oia	48 horas	46 horas (95 %)	2 horas (5 %)
exu	107 horas	97 horas (90 %)	10 horas (10 %)
oxum	264 horas	253 horas (95 %)	11 horas (5 %)
oxala	102 horas	93 horas (91 %)	9 horas (8 %)

Tabela 2 - Utilização de CPU

Analisando a Tabela 2, podemos concluir que grande parte do poder computacional oferecido pelo ambiente distribuído não é utilizado, normalmente pela falta de suporte oferecida pelo próprio sistema operacional.

O escalonamento de tarefas visando balanceamento de carga é um problema que vem sendo estudado desde a introdução dos sistemas multiprogramados. Em sistemas distribuídos, no entanto, novas questões são adicionadas, como a heterogeneidade dos componentes do ambiente e a dificuldade de obtenção das informação sobre utilização dos recursos, necessárias ao escalonador para a tomada de decisões. O modelo de gerência parece oferecer uma boa solução para estas questões. A utilização de um protocolo aberto de gerência permite a troca de informações entre arquiteturas e sistemas operacionais distintos.

Soluções para a melhor utilização do poder computacional em ambientes distribuídos são alvos de vários trabalhos atuais [BEG93, OZA92a]. Alguns deste trabalhos têm como objetivo o desenvolvimento de sistemas operacionais que se enquadrem melhor no paradigma

computacional distribuído, enquanto outros enfocam a construção de ambientes de suporte ao desenvolvimento de aplicações que utilizem melhor o poder computacional oferecido.

Os usuários de um sistema de computação distribuído não devem se preocupar com a localização física dos recursos computacionais compartilhados, como unidades de processamento, discos ou impressoras. Transparência é um requisito fundamental para que tenhamos ambientes computacionais realmente distribuídos, eficientes e, principalmente, eficazes.

A maioria dos sistemas operacionais atuais, como Unix e MS-DOS/Windows, foram projetados inicialmente para sistemas centralizados e estão sendo estendidos para que possam oferecer acesso a recursos compartilhados distribuídos pela rede. Esta solução não é a ideal, principalmente pela dificuldade de enquadrar características antigas destes sistemas operacionais com as novas necessidades. Sistemas operacionais baseados em novos modelos devem ser desenvolvidos, principalmente com o objetivo de oferecer ao usuário a transparência mencionada acima. AMOEBA [MUL87], MACH [ACC86] e CHORUS [ROZ88] são alguns exemplos de sistemas operacionais modernos realmente distribuídos.

A desvantagem da adoção de novos sistemas operacionais é o esforço de migração necessário para substituição dos sistemas operacionais em uso. É razoavelmente claro que esta migração terá que ocorrer a médio prazo, mas antes que ela possa se dar em larga escala estes novos sistemas terão que alcançar um patamar de maturidade, estabilidade e facilidade de manutenção, ainda não disponível.

Utilizando os sistemas operacionais disponíveis atualmente, pode-se fazer uso do poder computacional de um ambiente distribuído através da utilização de ferramentas de suporte à programação distribuída. Ainda que mais complexa do que o desejável, a programação baseada nestas ferramentas permite a utilização do potencial de ambientes distribuídos a curto prazo. Alguns exemplos de ambientes de suporte deste tipo são o PVM [BEG93, GEI92, SUN90] e PARALEX [OZA92a, OZA92b], ambos baseados na idéia de uma máquina paralela virtual, formada por um conjunto de máquinas em rede.

Uma aplicação que executa nesta máquina paralela virtual é composta por um conjunto de processos, distribuídos pelas estações que formam a máquina paralela, comunicando-se por troca de mensagens. A grande maioria destes ambientes para programação distribuída, no entanto, não oferece suporte algum para decisões de escalonamento dinâmico de processos. A máquina onde é disparado cada processo é tipicamente escolhida através de um escalonamento estático, baseado na arquitetura das máquinas, ou segundo um critério de seleção rotativo (*round-robin*) entre as máquinas envolvidas no *pool* de processamento.

5.2 *PVM - Parallel Virtual Machine*

O PVM (*Parallel Virtual Machine*) como mencionado na seção anterior, provê a abstração de uma máquina paralela sobre uma rede de estações, fornecendo uma biblioteca de funções para comunicação e disparo de processos. Escolheu-se trabalhar com este pacote devido ao seu amplo uso e disponibilidade em diversas plataformas.

A biblioteca PVM é distribuída em duas versões, uma para C, utilizada em nosso trabalho, e outra para FORTRAN. O modelo de programação oferecido é bastante simples, com suporte para criação de novos processos e para comunicação entre eles. A função *pvm_spawn()* dispara novos processos, retornando identificadores que são usados para envio de mensagens. Os dados enviados devem ser explicitamente empacotados pelo processo transmissor (através de chamadas a funções da biblioteca) e desempacotados pelo receptor. O empacotamento transforma os dados da representação nativa para uma representação interna

única, o processo inverso é realizado pelo mecanismo de desempacotamento. Com isto, é possível a comunicação em um ambiente heterogêneo. A biblioteca oferece ainda suporte para criação de grupos de processos, identificados por um nome, e comunicação entre grupos. Processos podem integrar-se a grupos ou sair deles dinamicamente. A abstração de grupo facilita o envio de mensagens em *multicast* e o uso de barreiras de sincronização.

Além da biblioteca, o pacote PVM inclui uma console, que pode ser usada interativamente para configuração do ambiente de execução e para disparo de aplicações. O conjunto de máquinas usado por uma aplicação PVM pode ser definido através de um arquivo de configuração, ou interativamente, através desta console. Ao chamar a função *pvm_spawn()*, a aplicação PVM pode especificar ou não a máquina onde este deve ser executado. Caso isto não seja feito, o mecanismo de definição da estação segue um algoritmo básico *round-robin*.

Em sua forma atual, o PVM não dispõe de mecanismos que permitam à aplicação investigar a carga de utilização das estações que compõem a máquina paralela. O que leva uma aplicação a especificar uma determinada estação, onde será disparado um novo processo, são apenas requisitos de arquitetura.

5.3 Mecanismo de Suporte a Balanceamento

Baseado no mecanismo de gerência de estações descrito anteriormente, definimos um mecanismo de suporte a balanceamento que pode ser utilizado juntamente com o PVM, incluindo funções que retornam informações sobre carga das estações e, mais especificamente, uma função que identifica a "melhor máquina".

Esta função é implementada através de uma chamada (RPC) a um processo, chamado *coletor*, que mantém uma visão global do sistema (*snapshot*) a partir das informações de carga obtidas das estações envolvidas. O coletor mantém uma lista ordenada dessas estações, onde a cada instante a primeira estação na lista é a "melhor", segundo a visão global corrente. Vários algoritmos e heurísticas podem ser utilizados na determinação da "melhor" máquina.

É importante notar que o algoritmo ideal, de escolha da melhor estação para executar uma determinada tarefa, deve levar em consideração características específicas da tarefa, tais como quantidade de memória necessária, volume de acesso a disco e tempo de duração da tarefa. Como o problema de prever o comportamento de uma tarefa é bastante complexo, neste trabalho, assim como na maioria das propostas para balanceamento distribuído, levamos em consideração apenas as medidas de carga das estações.

A escolha da melhor máquina para disparar um novo processo pode, em princípio, levar em conta todas as medidas de utilização de recursos mencionadas na seção 3.2. Por uma questão de simplicidade, levamos em conta em nossa implementação apenas duas medidas, o número de processos na fila de prontos e a quantidade de memória disponível. Além disto, a velocidade de processamento de cada estação é também levada em consideração.

Em cada uma das estações, que compõem a máquina paralela virtual, é executado o agente SNMP discutido anteriormente. O agente implementado fornece as informações do estado dos recursos em resposta a solicitações do gerente. Estas solicitações são feitas pelo coletor em intervalos de tempo que podem ser configurados.

A determinação do intervalo ideal entre solicitações é delicada, uma vez que dela depende o balanço entre a qualidade do *snapshot* do sistema, mantido pelo coletor, e o tráfego de comunicação introduzido por ele. Quanto menor o intervalo de *polling*, melhor a qualidade e maior o tráfego. Deve-se notar aqui que o próprio mecanismo de balanceamento de carga interfere de forma indesejável na carga do sistema. Trabalhamos inicialmente com intervalos da

ordem de 1 segundo. Pretende-se experimentar a utilização de diferentes intervalos, comparando os resultados obtidos para uma mesma aplicação PVM.

O coletor pode ser organizado segundo um modelo mestre-escravos, onde o processo mestre dispara um escravo para cada uma das estações que formam a máquina paralela. Este escravo é responsável pelo monitoramento contínuo de uma estação, passando para o mestre as informações obtidas. Isto permite que o processo mestre não fique bloqueado na coleta de informações, podendo responder imediatamente a pedidos das aplicações PVM.

A diferença de tempo que pode existir (da ordem do intervalo de *polling*) entre o momento da consulta e o momento em que partes do *snapshot* foram obtidas causa, obviamente, uma imprecisão nas informações utilizadas. Esta imprecisão faz com que este mecanismo não nos pareça adequado para o escalonamento de tarefas com tempo de execução pequeno. A utilização das informações de carga para escalonamento de processos PVM se torna interessante somente para aplicações, fortemente *CPU-bound,* compostas por processos com tempo de execução da ordem de dezenas de minutos.

5.4 Outras aplicações Gerentes SNMP

Diversas outras aplicações gerentes podem ser definidas tendo como base as informações disponíveis através da biblioteca HLMLIB.

Dando continuidade a este trabalho, pretendemos implementar uma aplicação de monitoramento que produza estatísticas de uso do sistema distribuído, oferecendo ao administrador da rede indicações de gargalos e de recursos ociosos. Estas indicações podem servir como base para reconfigurações, tanto no que diz respeito à redefinição de parâmetros do sistema, como área de swap, número de processos a serem mantidos em memória principal, etc, como também no que diz respeito a necessidades de expansão e redistribuição de recursos de hardware.

Um projeto bem mais complexo seria o desenvolvimento de um *shell* que realizasse compartilhamento e migração de carga [LIT92], disparando cada comando digitado pelo usuário na máquina mais apropriada, de forma semelhante aos sistemas Condor [LIT88] e Utopia [ZHO92]. Este mecanismo deve oferecer ao usuário a transparência de utilização de recursos mencionada anteriormente. Este projeto apresenta, no entanto, uma série de dificuldades. Em primeiro lugar, para dar suporte a decisões de escalonamento deste porte, as informações sobre utilização de recursos devem necessariamente ser atualizadas a intervalos de tempo muito pequenos, o que, como já discutido, pode ter um resultado bastante negativo sobre a carga do sistema. Um outro problema é a necessidade de levar em consideração uma série de fatores, como por exemplo, a quantidade de acessos remotos gerados pela execução de uma aplicação fora do seu ambiente original.

6. Observações Finais

Apresentamos neste trabalho uma proposta de utilização do modelo de gerência como suporte ao monitoramento e controle da carga em um sistema distribuído. A utilização deste modelo fornece uma solução simples de integração em sistemas heterogêneos. Qualquer sistema que execute um agente SNMP pode ser integrado em uma aplicação de controle de utilização de recursos, bastando para isto que se estenda o agente com a MIB proposta.

O modelo utilizado pela biblioteca HLMLIB, além de fornecer ao programador de aplicações gerentes as facilidades de uma interface de alto nível, vem ao encontro de uma necessidade bastante importante atualmente, o desacoplamento entre aplicações gerentes e

protocolos de gerência. A indefinição em relação à adoção definitiva de um padrão dificulta o investimento em aplicações gerentes de porte significativo. Com o modelo proposto, uma aplicação fica independente do protocolo utilizado.

Pretende-se continuar o estudo de aplicações deste modelo, investigando principalmente sua utilização em mecanismos de balanceamento de carga. Em primeiro lugar, deve ser conduzido um experimento comparativo com o mecanismo proposto na seção 5, baseado em uma mesma aplicação PVM executada com e sem a utilização deste mecanismo, sob diferentes condições de carga. Caso sejam obtidos os resultados esperados, o mecanismo de balanceamento deve ser incorporado a uma ferramenta de programação distribuída de forma que sua utilização seja automática, e independa de chamadas explícitas inseridas no programa.

7. Bibliografia

[ACC86] M. Accetta, R. Baron, W. Bolosky, D. Golub, R.Rashid, A. Tevanian and M. Young. "Mach: a New Kernel Foundation for Unix Development". *Proceedings of USENIX Summer Conference1986*. Junho 1986.

[BEG93] A. L. Beguelin, J. J. Dongarra, A. Geist, R. J. Mancheck, and V. Sunderam. "Heterogeneous Network Computing". *Proc. Sixth SIAM Conference on Parallel Processing*. 1993.

[GEI92] A. Geist and V. Sunderam. "Network-based concurrent computing on the PVM system". *Concurrency: Practice and Experience*. Junho 1992.

[IS9596] Information Technology - Open Systems Interconection. "Common Management Information Protocol Specification". International Organization for Standardization. IS 9596. 1991.

[KAR94a] M. Kara. *" Using Dynamic Load Balancing in Distributed Information Systems"*. University of Leeds. Technical report 94.18. Maio 1994.

[KAR94b] M. Kara. *" A Global Plan Policy for Coherent Cooperation in Distributed Dynamic Load Balancing Algorithms"*. University of Leeds. Technical report 94.21. Julho 1994.

[KUN91] T. Kunz. "The Influence of Different Workload Descriptions on a Heuristic Load Balancing Scheme". *IEEE Transactions on Software Engineering*. pp 725-730. Julho 1991.

[LEF89] S. J. Leffler, M. K. McKusick, M. J. Karels and J. S. Quarterman. *"The Design and Implementation of the 4.3 BSD Unix Operating System"*. Addison-Wesley Publishing Company. 1989.

[LIT88] M. Litzkow, M. Livny and M. Mutka. "Condor - A Hunter of Idle Workstations". *Proceedings of the 8th Internarional Conference on Distributed Systems*. Junho 1988.

[LIT92] M. Litzkow and M. Solomon. "Supporting Checkpointing and Process Migration Outside the Unix Kernel". *Usenix Winter 1992 Technical Conference*. pp. 283-290. Janeiro 1992.

[MUL87] S. J. Mullender. "The Amoeba Distributed Operating System". *Selected Papers 1984-1987*. CWI Tract nº 41. Amsterdam, Netherlands. 1987.

[NEU92] G. Neufeld and S. Vuong. "An Overview of ASN.1". *Computer Networks and ISDN Systems*. 23. pp. 393-415. 1992.

[OSS92] W. Osser. *"Automatic Process Selection for Load Balancing"*. University of California Santa Cruz. Master Thesis. Junho 1992.

[OZA92a] B. Ozalp et al. *"Run-time Support for Dynamic Load Balancing and Debugging in Paralex"*. University of Bologna. Technical Report UBLCS-92-3. Setembro 1992.

[OZA92b] B. Ozalp et al. *"Paralex: An Environmeny for Parallel Programming in Distributed Systems"*. University of Bologna. Technical Report UBLCS-92-4.

Outubro 1992.

[RFC1095] U. Warrier and L. Besaw. *"Common Management Information Services and Protocol over TCP/IP (CMOT)"*. RFC 1095. Internet Engineering Task Force. Abril 1989.

[RFC1155] M. Rose and K. McCloghrie. *"Structure and Identification of Management Information for TCP/IP-based Internets"*. RFC 1155. Internet Engineering Task Force. Maio 1990.

[RFC1157] M. Schoffstall, M. Fedor, J. Davin and J. Case. *"A Simple Network Management Protocol (SNMP)"*. RFC 1157. Internet Engineering Task Force. Maio 1990.

[RFC1213] M. Rose. *"Management Information Base for Network Management of TCP/IP-based Internets: MIB-II"*. RFC 1213. Internet Engineering Task Force. Março 1991.

[RFC1215] M. T. Rose. *"A Convention for Defining Traps for use with the SNMP"*. RFC 1215. Internet Engineering Task Force. Março 1991.

[RFC1227] M. T. Rose. *"SNMP MUX Protocol and MIB"*. RFC 1227. Internet Engineering Task Force. Maio 1991.

[RFC1228] G. Carpenter and B. Wijnen. *"SNMP-DPI - Simple Network Management Protocol Distributed Program Interface"*. RFC 1228. Internet Engineering Task Force. Maio 1991.

[RFC1382] D. Throop. "*SNMP MIB Extension for the X.25 Packet Layer*". RFC 1382. Internet Engineering Task Force. Novembro 1992.

[RFC1441] J. Case, K. McCloghrie, M. Rose and S. Waldbusser. *"Introduction to version 2 of the Internet-standard Network Management Framework"*. RFC 1441. Internet Engineering Task Force. Abril 1993.

[RFC1514] P. Grillo and S. Waldbusser. *"Host Resources MIB"*. RFC 1514. Internet Engineering Task Force. Setembro 1993.

[ROS93] M. T. Rose. "*The Simple Book - An Introduction to Internet Management*". 2° Edição. Prentice-Hall. ISBN 0-13-812611-9. 1993.

[ROZ88] M. Rozier, V. Abrossimov, F. Armand, et al. "Chorus Distributed Operating System". *Computing Systems Journal*. 1(4). Dezembro1988.

[STA93] W. Stallings. *"SNMP, SNMPv2 and CMIP"*. Addison Wesley Publishing Company. Abril 1993.

[SUN90] V. S. Sunderam. "PVM: A framework for parallel distributed computing". *Concurrency: Practice and Experience*. Dezembro 1990.

[TOG93] J. W. Togtema. *"Specification and Implementation of a Network Management Service based on SNMPv2"*. M.Sc. Thesis. University of Twente. 1993.

[ZHO92] S. Zhou, X. Zheng, J. Wang and P. Delisle. "*Utopia: A Load Sharing Facility for Large, Heterogeneous Distributed Computer Systems*". Technical Report CSRI n° 257. University of Toronto. Abril 1992.

8. Apêndice A

A seguir é apresentado uma parte da HLM-MIB proposta, utilizada na implementação do agente Host Load Metrics descrito na seção 3. Como dito anteriormente, o principal objetivo desta extensão é manter apenas medidas de carga dos subsistemas de um estação.

```
HOST-LOAD-METRICS-MIB DEFINITIONS ::= BEGIN
IMPORTS
  OBJECT-TYPE        FROM RFC-1212
  experimental       FROM RFC1155-SMI;

hostLoads            OBJECT IDENTIFIER ::= { experimental 555 }
hlmCPU               OBJECT IDENTIFIER ::= { hostLoads 1 }
hlmMemory            OBJECT IDENTIFIER ::= { hostLoads 2 }
```

```
hlmVirtualMemory        OBJECT IDENTIFIER ::= { hostLoads 3 }
hlmNetwork              OBJECT IDENTIFIER ::= { hostLoads 4 }
hlmDisk                 OBJECT IDENTIFIER ::= { hostLoads 5 }

-- Convenções textuais

-- Valor booleano
Boolean ::= INTEGER { true(1), false(0) }

-- Tamanho da memória expressa em unidades de 1024 bytes
KBytes ::= INTEGER (0..2147483647)

-- Tamanho da memória expressa em unidades de bytes
Bytes ::= INTEGER (0..2147483647)

-- Percentual
Percentual ::= INTEGER (0..100)

---------------------------------------------------------------
-- Grupo CPU

hlmCPUType OBJECT-TYPE
    SYNTAX OBJECT IDENTIFIER
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        " Identifica o tipo da CPU da estação"
    ::= { hlmCPU 1 }

hlmCPUIdle OBJECT-TYPE
    SYNTAX Percentual
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        " Percentual de CPU ociosa durante
          o último segundo "
    ::= { hlmCPU 2 }

hlmCPUUser OBJECT-TYPE
    SYNTAX Percentual
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        " Percentual de CPU ocupada com processos usuário
          durante o último segundo "
    ::= { hlmCPU 3 }

hlmCPUNice OBJECT-TYPE
    SYNTAX Percentual
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        " Percentual de CPU ocupada com processos usuário
          nice durante o último segundo "
    ::= { hlmCPU 4 }

hlmCPUSystem OBJECT-TYPE
```

```
    SYNTAX Percentual
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
          " Percentual de CPU ocupada pelo sistema operacional
           durante o último segundo "
    ::= { hlmCPU 5 }

-------------------------------------------------------------
-- Grupo Memory

hlmPhysMemory OBJECT-TYPE
    SYNTAX Bytes
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
          " Quantidade total de memória física
           expressa em bytes "
    ::= { hlmMemory 1 }

hlmFreeMemory OBJECT-TYPE
    SYNTAX Bytes
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
          " Quantidade total de memória física livre
           expressa em bytes "
    ::= { hlmMemory 2 }

-------------------------------------------------------------
-- Grupo Virtual Memory

hlmPagesIn OBJECT-TYPE
    SYNTAX INTEGER
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
          " Quantidade de páginas paged in durante o
           último segundo "
    ::= { hlmVirtualMemory 1 }

hlmPagesOut OBJECT-TYPE
    SYNTAX INTEGER
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
          " Quantidade de páginas paged out durante o
           último segundo "
    ::= { hlmVirtualMemory 2 }

-------------------------------------------------------------
-- Grupo Network

hlmPacketsReceived OBJECT-TYPE
    SYNTAX INTEGER
    ACCESS read-only
    STATUS mandatory
```

```
        DESCRIPTION
              " Quantidade total de pacotes recebidos por todas
                as interfaces durante o último segundo "
        ::= { hlmNetwork 1 }

    hlmPacketsSend OBJECT-TYPE
        SYNTAX INTEGER
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
              " Quantidade total de pacotes enviados por todas
                as interfaces durante o último segundo "
        ::= { hlmNetwork 2 }

    ---------------------------------------------------------
    -- Grupo Disk

    hlmWorkStationDiskless OBJECT-TYPE
        SYNTAX Boolean
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
              " Indica se uma estação possui discos locais ou não:
                hlmWorkStationDiskless = 1 : Estação sem discos locais
                hlmWorkStationDiskless = 0 : Estação com discos locais "
        ::= { hlmDisk 1 }

    hlmDiskIO OBJECT-TYPE
        SYNTAX INTEGER
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
              " Quantidade total de acesso aos discos durante
                o último segundo"
        ::= { hlmDisk 2 }

    END
```