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

Luís Cassiano Goularte Rista

UMA ABORDAGEM DE MONITORAÇÃO WIRELESS EM UM AMBIENTE DE CLUSTER COM ALTA DISPONIBILIDADE

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

Orientador
Mario Antonio Ribeiro Dantas, Dr.

Florianópolis, fevereiro de 2005

UMA ABORDAGEM DE MONITORAÇÃO WIRELESS EM UM AMBIENTE DE CLUSTER COM ALTA DISPONIBILIDADE

Luís Cassiano Goularte Rista

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

_____

Prof. Raul Sidnei Wazlawick, Dr. (coordenador)

Banca Examinadora

_____

Prof. Mario Antonio Ribeiro Dantas, Dr. (orientador)

_____

Prof. Carlos Barros Montez, Dr.

_____

Prof. João Bosco Mangueira Sobral, Dr.

_____

Prof. Roberto Willrich, Dr.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

AP	Access Point
API	Application Programming Interface
ATM	Asynchronous Transfer Mode
BSC	Base Station Controller
BSS	Basic Service Set
BSS	Base Station System
BTS	Base Transceiver Station
CDMA/1xRTT	Code Division Multiple Access/1xRadio Transmission Technology
COTS	Components Off The Shelf
COW	Cluster of Workstations
CSMA	Carrier Sense Multiple Access
CSMA/CA	CSMA Collision Avoidance
CSMA/CD	CSMA Collision Detection
DS	Distribution System
DSM	Distributed Shared Memory
DSSS	Direct Sequence Spread Spectrum
ESS	Extended Service Set
FHSS	Frequency Hopping Spread Spectrum
GCC	GNU C Compiler
GFR	Guaranteed Frame Rate
GGSN	Gateway GPRS Support Node
GPL	GNU General Public License
GSM/GPRS	Global System Mobile/General Packet Radio Service
HA-OSCAR	High Availability – Open Source Cluster Application Resource
HR/DS	High-Rate Direct Sequence
IBSS	Independent Basic Service Set
ICMP	Internet Control Message Protocol
IPMI	Intelligent Power Management Interface
IR	InfraRed

LAN	Local Area Network
LLC	Logical Link Control
MAC	Medium Access Control
MIMD	Multiple Instruction Multiple Data
MISD	Multiple Instruction Single Data
MPI	Message Passing Interface
MPP	Massively Parallel Processors
MS	Mobile Station
MSC	Mobile-Services Switching Centre
NOW	Network of Workstations
OFDM	Orthogonal Frequency-Division Multiplexing
OFDMA	Orthogonal Frequency-Division Multiple Access
OSCAR	Open Source Cluster Application Resource
OSI	Open Systems Interconnection
Palm OS SDK	Palm OS Software Development Kit
PBS	Portable Batch System
PDA	Personal Digital Assistant
PHY	Physical
POSE	Palm OS Emulator
PVM	Parallel Virtual Machine
PVP	Parallel Vector Processors
QoS	Quality of Service
RISC	Reduced Instruction Set Computer
ROM	Ready Only Memory
RPM	Red Hat Package Manager
SGSN	Serving GPRS Support Node
SIMD	Single Instruction Multiple Data
SISD	Single Instruction Single Data
SMP	Symmetric Multiprocessors
SNMP	Simple Network Management Protocol
SSI	Single System Image
TCP	Transmission Control Protocol

TCP/IP	Transmission Control Protocol/Internet Protocol
TDM	Time-Dvision Multiplex
TDMA	Time Division Multiple Access
UDP	User Datagram Protocol
VoIP	Voice Over IP
WLAN	Wireless Local Area Network
WMAN	Wireless Metropolitan Area Network
WPAN	Wireless Personal Area Network
WWAN	Wireless Wide Area Network

RESUMO

O cenário que surge com a maior utilização dos sistemas computacionais de alto desempenho e mais recentemente com o advento das redes locais *wireless* tem requerido uma demanda cada vez maior por ferramentas e mecanismos para o controle e monitoração destes ambientes. Esse fato decorre, uma vez que a utilização destes sistemas computacionais em ambientes reais tem a cada dia nos indicado a necessidade do seu contínuo funcionamento. Desta forma, muitos serviços têm que estar disponíveis de forma ininterrupta. Possíveis falhas podem causar erros, fazendo com que o sistema apresente um defeito. Por esta razão, uma forma de redundância se faz necessária para evitar que tais defeitos acarretem prejuízos ao ambiente de alto desempenho.

Nesse sentido, o trabalho proposto nesta dissertação tem como objetivo descrever uma abordagem de monitoração *wireless* e efetivar uma proposta de projeto e implementação em um ambiente de *cluster* computacional com alta disponibilidade. O módulo de monitoração deverá enviar uma notificação a um dispositivo móvel, assim que o servidor primário for substituído pelo secundário. Este módulo tem por objetivo alertar o responsável pelo ambiente do *cluster* computacional sobre possíveis ocorrências de defeitos na configuração. Em prática, o que está sendo proposto e definido neste trabalho é um monitor, sendo essencialmente definido como uma ferramenta utilizada para observar as atividades de um sistema. O diferencial desta monitoração é justamente a capacidade de mobilidade que as redes *wireless* oferecem quando associadas a um dispositivo móvel.

A validação da ferramenta proposta foi verificada através de comparações com outros trabalhos relacionados como o IPMI (*Intelligent Power Management Interface*) por exemplo. Além desta medida, foram realizados inúmeros testes no ambiente experimental com o intuito de verificar o seu perfeito funcionamento. Para testar o ambiente, foi simulado um defeito através do desligamento do servidor primário. A partir deste momento o servidor secundário passou a agir como primário. Neste momento foi enviada uma notificação a um dispositivo móvel alertando sobre a possibilidade de defeito na configuração. Os resultados obtidos se mostraram bastante satisfatórios, uma vez que contemplam com exatidão a proposta de projeto e implementação da abordagem de monitoração *wireless* descrita neste trabalho.

ABSTRACT

The scenery that appears with the largest use of the high performance computing systems and more recently with the coming of the wireless local networks it has been requesting a demand every time larger for tools and mechanisms for the control and monitoring of these environments. That fact elapses, once the use of these computing systems in real environments has every day us indicated the need of yours continues operation. This way, many services have to be available in an uninterrupted way. Possible faults can cause errors, doing with that the system presents a failure. For this reason, a redundancy form is made necessary to avoid that such failure cart damages to the environmnent of high performance.

In that sense, the work proposed in this dissertation has as objective describes an approach of monitoring wireless and to effect a project proposal and implementation in an environment of cluster computing with high availability. The monitoring module should send a notification to a mobile device, as soon as the primary server is substituted by the secondary. This module has for objective to alert the responsible for the environment of the cluster computing on possible occurrences of failures in the configuration. In practice, what is being proposed and defined in this work it is a monitor, being defined essentially as a tool used to observe the activities of a system. The diferencial of this monitoring is exactly the mobility capacity that the wireless networks offers when associated to a mobile device.

The validation of the proposed tool was verified through comparisons with other related works like IPMI (Intelligent Power Management Interface) for example. Besides this measured, countless tests were accomplished in the experimental environment with the intention of verifying your perfect operation. To test the environment, a failure was simulated through the powering off of the primary server. Starting from this moment the secondary server started to act as primary. At this time was envoy a notification to a mobile device alerting about the failure possibility in the configuration. The obtained results if they showed quite satisfactory, once they contemplate with accuracy the project proposal and implementation of the approach of monitoring wireless described in this work.

1 – INTRODUÇÃO

Há muito tempo, fala-se que a utilidade dos computadores paralelos depende do desenvolvimento de algoritmos paralelos, que operem eficientemente em tais computadores e do desenvolvimento de linguagens de programação paralela, nas quais esses algoritmos possam ser expressos [KRO85]. É interessante observar que os computadores evoluíram, mas essas duas questões básicas (algoritmos e linguagens) ainda requerem soluções mais eficientes.

Com os recentes avanços nas tecnologias de microprocessadores e redes locais de alto desempenho têm se criado a possibilidade de utilização do paradigma de *cluster* computacional como um eficiente ambiente paralelo para execução de um grande número de aplicações. Essa abordagem baseia-se na idéia de utilização de diversos computadores comerciais facilmente encontrados no mercado (COTS – *Components Off The Shelf*) interligados por uma tecnologia de rede local, visando um aumento no desempenho das aplicações, através da exploração da distribuição/paralelismo.

Por outro lado, a maior utilização do paradigma de *cluster* computacional tem indicado a necessidade do continuo funcionamento desses sistemas, a fim de garantir uma perfeita simetria de execução do fluxo de instruções. Por esta razão, uma forma de redundância se faz necessária a fim de prevenir a ocorrência de interrupções indesejáveis. A capacidade de alta disponibilidade é uma forma usualmente utilizada em um ambiente de *cluster* computacional para permitir esta redundância, visando seu perfeito funcionamento. Impõe-se atualmente também conhecer a potencialidade, acompanhar o desenvolvimento, estar a par do incessante progresso, para manter condições de explorar judiciosamente os recursos disponíveis a cerca das redes *wireless* de modo a permitir que mecanismos de monitoração diferenciados possam ter soluções mais eficientes com relação à mobilidade.

Neste trabalho apresentamos nossa investigação no sentido de definir uma abordagem de monitoração em um ambiente de *cluster* computacional com alta disponibilidade através de um dispositivo móvel caracterizado por acesso *wireless*. Os mecanismos deverão ser capazes de informar ao dispositivo móvel a ocorrência de defeito, caso o mesmo venha a ocorrer no ambiente de *cluster* computacional. Sendo a capacidade de alta disponibilidade a principal responsável por assegurar ao ambiente de

cluster computacional a característica necessária para manter o seu funcionamento, caso apresente defeito em sua configuração.

Nesse sentido, o módulo de monitoração deverá enviar uma notificação ao dispositivo móvel, assim que o servidor primário seja substituído pelo secundário. O objetivo é alertar o responsável pelo ambiente de *cluster* computacional sobre possíveis ocorrências de defeitos na configuração do ambiente de alto desempenho. Na prática, o que está sendo proposto e definido neste trabalho é um monitor, descrito na literatura por [JAI91] como sendo uma ferramenta utilizada para observar as atividades de um sistema. Neste caso, com o diferencial da monitoração possuir uma capacidade de mobilidade graças às redes *wireless* associadas a um dispositivo móvel, o que permite esta facilidade à ferramenta desenvolvida.

O restante do trabalho foi disposto da seguinte forma: o próximo capítulo é dedicado somente à descrição dos ambientes de alto desempenho. Em seguida, no capítulo III, são apresentadas as redes *wireless* e suas várias classificações. Já no capítulo IV, se encontram as definições referentes ao ambiente experimental. O capítulo V foi destinado à apresentação da ferramenta de monitoração *wireless* desenvolvida, bem como os resultados obtidos para sua validação. Finalmente, o último capítulo ficou reservado para as nossas conclusões e trabalhos futuros.

2 – AMBIENTES DE ALTO DESEMPENHO

Os ambientes de computação de alto desempenho [ROS04] vêm tornando-se mais populares em função da demanda sempre crescente por poder computacional. Nesse sentido, os ambientes de alto desempenho objetivam uma melhor utilização dos recursos computacionais e conseqüentemente a melhoria de desempenho das aplicações. Por outro lado, a complexidade inerente destes ambientes, muitas vezes imposta pela arquitetura paralela do ambiente acaba tornando a tarefa de programação mais difícil, isto quando comparado com a programação seqüencial. Essa complexidade pode ser mais bem explicada através da dificuldade de se obter o conceito de imagem única (SSI – *Single System Image*), sendo que o usuário percebe a máquina paralela como um sistema único. A seguir apresentamos os conceitos básicos de arquiteturas paralelas, incluindo as suas diferentes classificações e as principais tendências para a sua construção. Um destaque especial é dado ao ambiente de *cluster* computacional, uma vez que será utilizado no ambiente experimental.

2.1 – Classificação de Flynn

A classificação de Flynn [FLY72] teve sua origem em meados dos anos 70, sendo ainda aceita e muito difundida nos dias atuais. Para a classificação, Flynn baseia-se no fato de um computador executar um fluxo de instruções (*instruction stream*) sobre um fluxo de dados (*data stream*). Logo abaixo apresentamos uma tabela (veja tabela 2.1) com as quatro classes propostas por Flynn:

	SD (*Single Data*)	**MD** (*Multiple Data*)
SI (*Single Instruction*)	SISD Máquinas Von Neumann	SIMD Máquinas vetoriais
MI (*Multiple Instruction*)	MISD Sem representante (até agora)	MIMD Multiprocessadores e multicomputadores

Tabela 2.1. Classificação de Flynn

Como podemos observar na tabela 2.1, a classe SISD (*Single Instruction Single Data*) executa apenas um único fluxo de instruções sobre um único fluxo de dados. Essa classe é caracterizada pelas máquinas Von Neumann tradicionais com apenas um processador, como apresentado na figura 2.1. O diagrama abaixo, apresenta o fluxo de instruções, definido pela linha continua, enviando instruções a uma unidade de controle (C) que aciona a unidade central de processamento (P). Neste momento, a unidade P atua sobre um único fluxo de dados, identificado pela linha tracejada, sendo então processado e reescrito na memória (M).

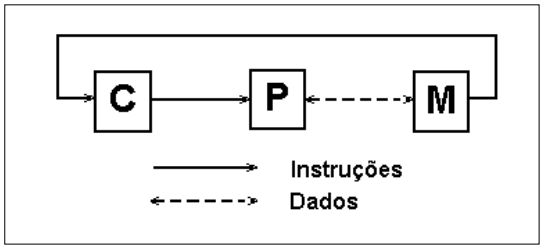

Figura 2.1. Diagrama da classe SISD

A classe MISD (*Multiple Instruction Single Data*), por sua vez, executa múltiplos fluxos de instruções sobre um único fluxo de dados. Essa classe é considerada vazia, uma vez que não faz qualquer sentido, além de ser tecnicamente inviável a sua implementação nos dias atuais [ALM89][HOC88]. Na figura 2.2, apresentamos um diagrama da classe MISD com múltiplas unidades de processamento P, cada uma com sua unidade de controle própria C, recebendo um fluxo diferente de instruções. Essas unidades de processamento executam um conjunto diferente de instruções sobre o mesmo fluxo de dados, ou seja, diferentes instruções atuam na mesma posição de memória ao mesmo tempo, executando diferentes instruções.

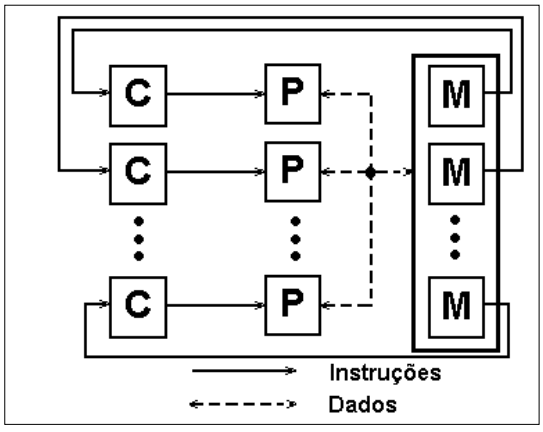

Figura 2.2. Diagrama da classe MISD

As máquinas paralelas [ROS04] concentram-se nas duas classes restantes, na classe SIMD (*Single Instruction Multiple Data*) e na classe MIMD (*Multiple Instruction Multiple Data*). A classe SIMD é caracterizada pela execução de uma única instrução ao mesmo tempo sobre múltiplos dados. Alguns exemplos de máquinas pertencentes à classe SIMD são: CM-2 [HWA93] e MasPar [HWA93]. Na figura 2.3, apresentamos o diagrama da classe SIMD, sendo que uma única unidade de controle C fica responsável pelo controle do processamento, enviado por um único fluxo de instruções, sendo a mesma instrução remetida para os demais processadores. As instruções são executadas em paralelo de forma síncrona pelos processadores sobre os diferentes fluxos de dados.

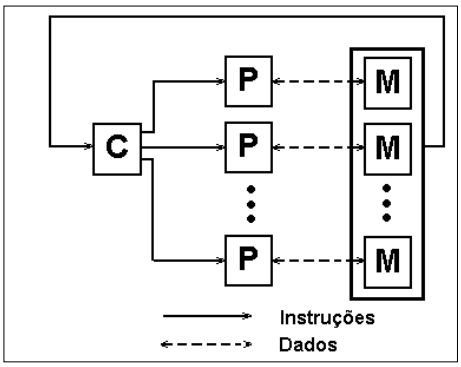

Figura 2.3. Diagrama da classe SIMD

A classe MIMD é caracterizada pelo fato de cada unidade de controle receber um fluxo de controle próprio e remeter esse fluxo para a sua unidade processadora P, para que seja executado sobre um fluxo de execução próprio, como demonstramos na figura 2.4, o diagrama da classe MIMD. Na prática, o que acontece é que cada processador executa seu próprio programa sobre seus próprios dados de forma assíncrona. Alguns exemplos de máquinas pertencentes à classe MIMD são: CM-5 [HWA93], nCUBE [CUL99], Intel Paragon [CUL99] e Cray T3D [CUL99].

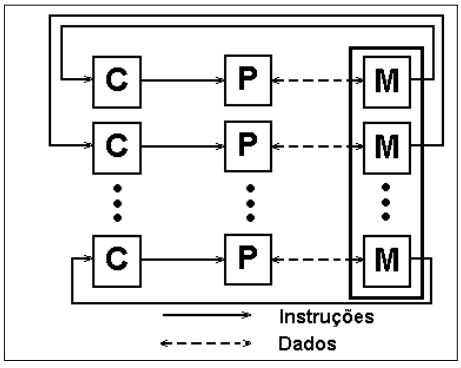

Figura 2.4. Diagrama da classe MIMD

2.2 – Classificação Segundo Compartilhamento de Memória

Um outro critério para a classificação de máquinas paralelas é segundo o compartilhamento de memória [DAN02][ROS04]. Nesse caso, a arquitetura paralela pode ser classificada como uma arquitetura de troca de mensagem ou memória compartilhada [DAN02]. No primeiro caso, ilustrado na figura 2.5, cada processador tem sua memória local e a comunicação entre processadores é efetuada através da troca de mensagens, sendo usualmente denominado como um ambiente de multicomputador. Na prática, o ambiente de multicomputador é construído a partir da replicação de toda uma arquitetura convencional, e não apenas da replicação do processador, como veremos a seguir no ambiente de multiprocessador.

Figura 2.5. Arquitetura de um multicomputador

Por outro lado, o ambiente de multiprocessador é caracterizado pela existência de uma memória global única (*shared memory*) compartilhada pelos processadores, sendo a comunicação entre processos realizada através desta memória compartilhada. Essas características resultam no fato desta arquitetura paralela ser construída a partir da replicação apenas do componente processador de uma arquitetura convencional como apresentamos na figura 2.6.

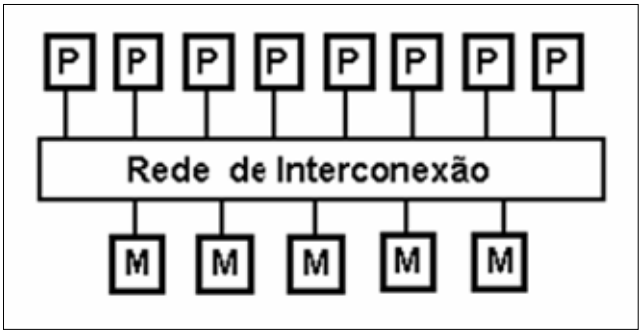

Figura 2.6. Arquitetura de um multiprocessador

2.3 – Máquinas Paralelas

A seguir apresentamos os principais modelos físicos de máquinas paralelas que constituem atualmente as principais tendências para a construção destes sistemas [ROS04]. Cabe destacar, que apenas os modelos NOW (*Network of Workstations*) e COW (*Cluster of Workstations*) caracterizam-se como ambientes de *cluster* computacionais. Outros modelos paralelos como PVP (*Parallel Vector Processors*) [ROS04], SMP (*Symmetric Multiprocessors*) [HWA98], MPP (*Massively Parallel Processors*) [HWA93][CUL99] e DSM (*Distributed Shared Memory*) [AMZ96] [CUL99] não foram explorados por essa razão.

2.3.1 – NOW

As NOW [ROS04] são sistemas constituídos por várias estações de trabalho usualmente interligadas por tecnologia tradicional de rede como *Ethernet* [SOA95] e ATM (*Asynchronous Transfer Mode*) [SOA95]. Na prática, uma rede local de estações já existente é utilizada na execução de aplicações paralelas, sendo a rede local vista como uma máquina paralela, onde vários processadores com suas memórias locais (estações de trabalho) são interligados por esta rede. A seguir apresentamos o exemplo de uma configuração típica de uma NOW, ilustrado na figura 2.7.

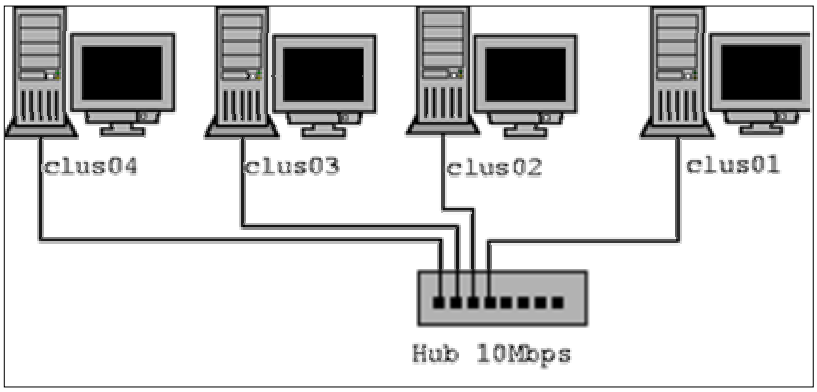

Figura 2.7. Configuração de uma NOW

Uma dificuldade encontrada nessas máquinas é a rede de interconexão, uma vez que redes tradicionais como *Ethernet* e ATM não são otimizadas para o ambiente

paralelo. Como resultado, a alta latência nas operações compromete o desempenho da máquina como um todo. Além disso, o baixo desempenho da rede de interconexão presente nas NOW, restringe a sua utilização a ambientes de ensino de processamento paralelo e distribuído e na execução de aplicações que tenham pouca comunicação entre os processos.

2.3.2 – COW

Segundo [DAN02] o paradigma de COW também denominado de *cluster* computacional, ambos termos são válidos, visa um aumento do desempenho das aplicações, através de uma maior taxa de execução dos aplicativos e um aumento no número de dados a serem considerados na execução. Em um ambiente de *cluster* computacional, pode-se imaginar que para atingir o objetivo de melhoria de desempenho, primeiramente devemos considerar alguns esforços, tais como: aumento na velocidade do processador, uso de algoritmos mais otimizados e adoção de um ambiente de computação concorrente (ou paralela).

No contexto tecnológico atual o paradigma de *cluster* computacional representa uma forma popular de configuração para execução de aplicações que requeiram um computador paralelo. O termo *cluster* computacional pode ser considerado como um conjunto de computadores interligados de tal forma, que os seus usuários tenham a convicção de estar usando um recurso computacional único. Uma boa parte dessa abstração é obtida através das camadas de *software*, dentro ou fora do sistema operacional, que permitem essa transparência.

Assim como nas NOW, os *clusters* também são constituídos por várias estações de trabalho interligadas, mas com a diferença de terem sido projetados com o objetivo de executar aplicações paralelas. Dessa forma, a máquina pode ser otimizada especificamente para esse fim. Usualmente, estas estações se caracterizam pela ausência de monitor, teclado e mouse, sendo denominadas *headless workstation*, como apresentamos na figura 2.8.

Nas configurações de *cluster*, as estações são utilizadas especificamente para executar aplicações paralelas, sendo assim, o sistema operacional pode ser melhor customizado, para garantir uma melhoria de desempenho para as aplicações, quando

comparado com as NOW. Várias camadas de rede podem ser simplificadas, ou até mesmo totalmente eliminadas, pois as necessidades de comunicação em máquinas paralelas são diferentes das necessidades em redes locais [VAN94]. Na prática, são aproveitadas as principais vantagens dos sistemas NOW, que são o ótimo custo/benefício e a grande flexibilidade de construção, e tenta-se amenizar a principal desvantagem, que é a alta latência das comunicações, construindo uma NOW dedicada ao processamento paralelo e distribuído [ROS04].

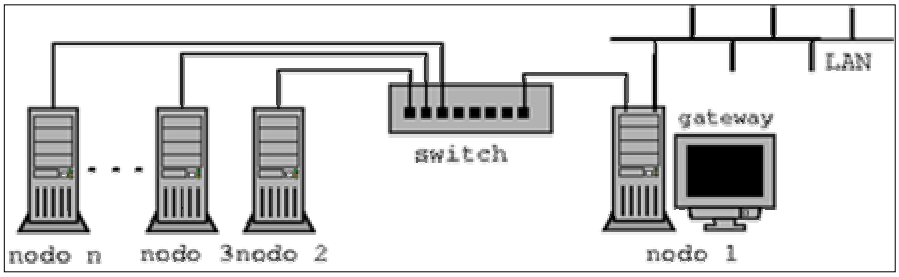

Figura 2.8. Configuração de uma COW

3 – REDES WIRELESS

Segundo [DAN02], os ambientes de redes *wireless* são configurações interessantes para agregar valor às redes das corporações. O diferencial destes ambientes pode ser ilustrado pelo custo reduzido da sua infra-estrutura e o suporte a aplicações móveis. As redes *wireless* oferecem ganhos para os processos móveis envolvidos na utilização desta tecnologia, tais como a mobilidade e o baixo custo da solução quando comparado com uma rede guiada.

Em outra definição [SWA02] considera que os componentes básicos de uma arquitetura *wireless* permanecem praticamente os mesmos ao longo de diferentes sistemas. Na prática, cada implementação e tecnologia possuem diferentes variações e opções, por outro lado mantêm muitos dos seus componentes básicos. A arquitetura apresentada a seguir, visa demonstrar a representação de uma estrutura genérica. Note que os dispositivos *wireless* recebem informação de um servidor conectado à Internet através de comunicação *wireless* (veja figura 3.1).

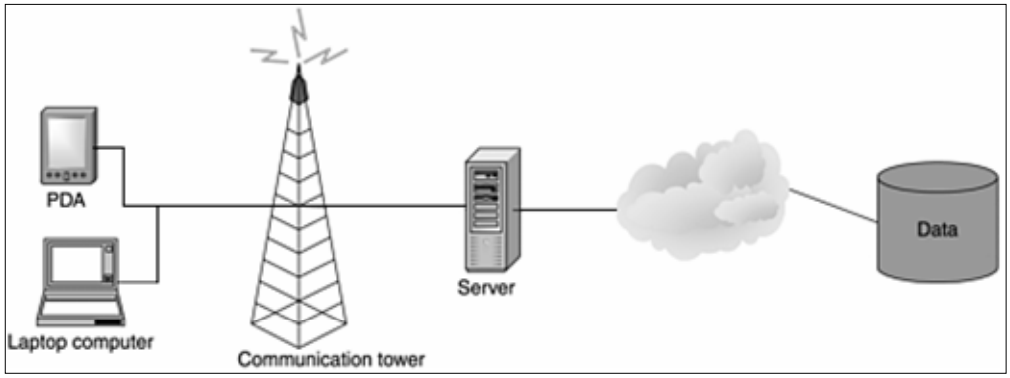

Figura 3.1. Arquitetura de um sistema genérico [SWA02]

O primeiro componente em nosso diagrama é o dispositivo. Dispositivos em um sistema *wireless* são telefones celulares, PDAs (*Personal Digital Assistants*), *laptops* com cartões de rede *wireless*, ou qualquer dispositivo que possua facilidades *wireless*. Estes dispositivos operam através de redes *wireless* e se comunicam com as torres denominadas portadoras. As portadoras enviam a informação através de sinais de radio freqüência a uma rede guiada. Estas portadoras recebem e transmitem dados a um componente que remete esta informação através de rede guiada para a Internet. Este

componente muitas vezes é um *gateway wireless* e em outras situações um servidor especializado, projetado para ser o ponto principal entre a comunicação *wireless* e a rede guiada. Um *gateway* executa traduções entre protocolos, sessões, criptografia, e todo o necessário para a transmissão *wireless* através da Internet até seu destino.

Em nosso diagrama de arquitetura (veja figura 3.1), podemos verificar um cenário típico; o dispositivo *wireless* requisitando informação de uma página *web* em um servidor *web*. O *gateway*, nesta situação, atende a requisição provendo a ligação com a Internet e posteriormente enviando a informação para o servidor apropriado. O servidor processa a requisição e retorna a informação para o *gateway* pela mesma ligação Internet guiada. Neste ponto, o *gateway wireless* realiza as alterações necessárias novamente e transmite os dados para a portadora, que em contrapartida, envia para o dispositivo. O dispositivo exibe a informação recebida na tela, e a iteração deste ciclo de comunicação está completa. As tecnologias *wireless* são geralmente a ultima ligação entre uma rede guiada, seus recursos, e a nova geração de dispositivos *wireless*.

As redes *wireless* são usualmente classificadas como: WPAN (*Wireless Personal Area Network*), WLAN (*Wireless Local Area Network*), WMAN (*Wireless Metropolitan Area Network*) e WWAN (*Wireless Wide Area Network*). A seguir descrevemos com mais detalhes essas tecnologias (veja figura 3.2), além de alguns aspectos sobre a arquitetura, princípios de operação, e protocolos de comunicação usualmente utilizados por estes ambientes.

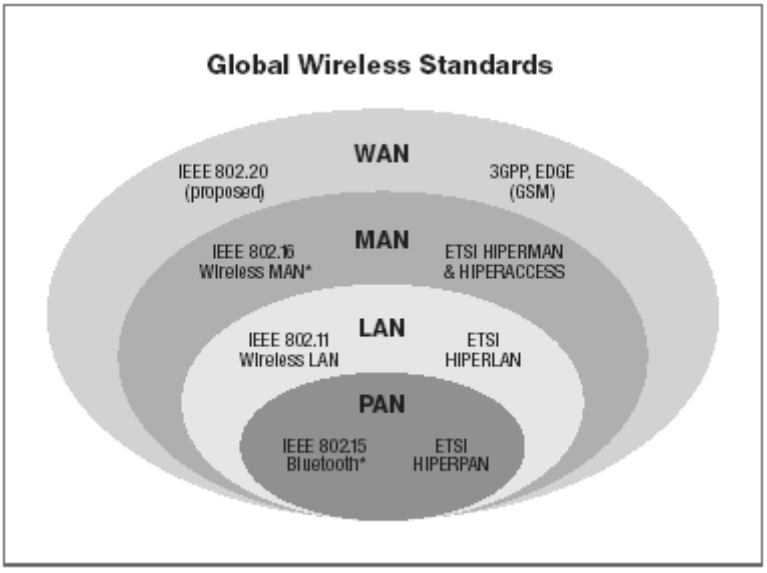

Figura 3.2. Padrões *wireless* e suas tecnologias [INT03]

3.1 – WPAN

Em março de 1999 o comitê de padronização IEEE 802 formou um novo grupo de trabalho para desenvolver padrões de comunicação nas WPANs para dispositivos móveis e portáteis. O novo grupo foi denominado IEEE 802.15 e se baseou na especificação *bluetooth* para desenvolver o padrão IEEE 802.15 para as WPANs [BRA00].

3.1.2 – IEEE 802.15

A tecnologia *wireless bluetooth* ou IEEE 802.15 é um padrão de fato e uma especificação para enlaces entre dispositivos móveis, tais como computadores pessoais, assistentes digitais pessoais, telefones celulares e outros dispositivos portáteis de baixo custo, usando ondas de rádio de curto alcance [BRA00][DAN02]. A proposta da tecnologia *bluetooth* é habilitar os usuários para a conexão de uma grande variedade de dispositivos de computação e telecomunicações de maneira simples e fácil, sem a necessidade de conectar cabos.

A operação do *bluetooth* é efetuada em uma freqüência de 2,4 GHz que está globalmente disponível e tem compatibilidade mundial sendo portanto um padrão global para conectividade *wireless* [BRA00]. As configurações das redes com a tecnologia *bluetooth* oferecem soluções interessantes, permitindo que até sete dispositivos escravos se conectem a um dispositivo mestre [BIS01][VAX03], ilustrado na figura 3.3. As microredes (*piconets*) são interoperáveis e podem formar uma rede maior e flexível, na qual vários dispositivos podem entrar e sair, sem maiores prejuízos para o conjunto.

A rede *bluetooth* tem uma capacidade máxima teórica de 10 Mbps, sendo considerado uma resposta para a necessidade de conexão *wireless* de equipamentos em distâncias curtas (até 10m) nas seguintes áreas [VAX03]:

- Pontos de acesso de dados e voz
- Substituição de cabos
- Redes *ad-hoc* (temporárias)

Com relação à camada física (PHY – *Physical*), o *bluetooth* utiliza um esquema de modulação conhecido como FHSS (*Frequency Hopping Spread Spectrum*) [SWA02][VAX03][WHE01].

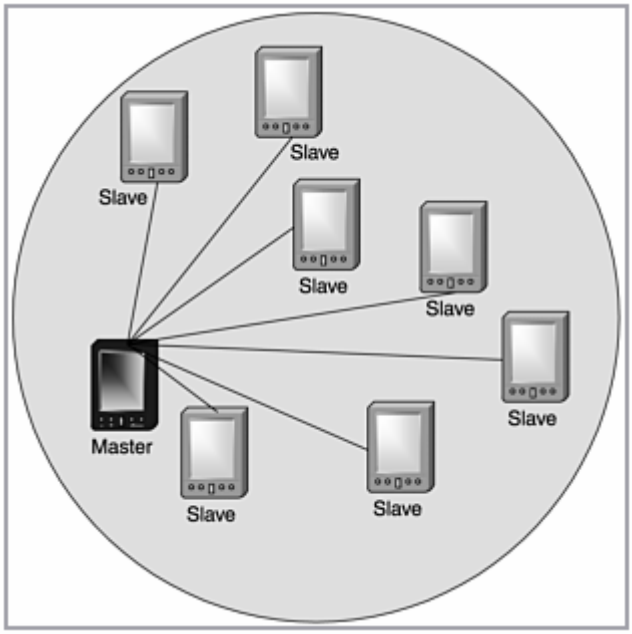

Figura 3.3. Uma *piconet* [SWA02]

Uma das facilidades mais importantes do ambiente *bluetooth* é o acesso às redes locais. Uma vez que um determinado dispositivo móvel esteja conectado a uma rede local, este poderá ter acesso a todas as facilidades disponíveis no ambiente de LAN (*Local Area Network*). A perspectiva atual da tecnologia *bluetooth* é que esta seja uma porta de comunicação *wireless* entre diversos dispositivos, como computadores pessoais, assistentes digitais pessoais, telefones celulares e outros dispositivos portáteis de baixo custo que possam se beneficiar de alguma forma com o uso desta tecnologia.

3.2 – WLAN

Na literatura [SWA02] descreve uma WLAN como sendo um sistema de comunicação de dados implementado como uma extensão, ou alternativa, a uma LAN tradicional. As WLANs utilizam uma variedade de mecanismos de comunicação com o intuito de substituir os cabos e conexões de uma LAN. Em uma LAN tradicional, dados

são transmitidos como pulsos eletrônicos ou sinais ao longo de um meio físico ou através de uma onda contínua de alta freqüência que pode ser modulada por um sinal (portadora). Alguns sistemas possuem um sinal continuo ou portadora executando sobre meio guiado, como um tom na linha telefônica, e os dados estão sobrepostos ou modulados para a portadora do sinal. Em um exemplo simplificado de telefone, o transmissor causa pequenas variações para a freqüência de tom, e o receptor é então configurado para detectar estas variações e recuperar os dados transmitidos. Similarmente, em uma WLAN, há transmissores, receptores, e uma portadora na qual os dados são modulados.

Em uma definição mais usual, podemos considerar que uma WLAN é essencialmente uma rede local *wireless*, implementada como extensão ou alternativa as redes convencionais, ou seja, é um sistema de comunicação que não requer fios para transportar os sinais. As WLANs utilizam sinais de rádio freqüência para a transmissão de dados, minimizando a necessidade de cabos de conexão dos usuários à rede. Desta forma, uma WLAN combina comunicação de dados com mobilidade dos usuários dentro da área de cobertura da rede, que pode atingir algumas centenas de metros.

A tecnologia WLAN vem se firmando cada vez mais, como um elemento fundamental para agregar valor às redes de computadores nas organizações. Por esse motivo, tem se verificado um aumento bastante significativo em sua utilização. Desta maneira, uma solução de rede móvel, provendo facilidades de resposta a determinadas solicitações de informações, pode representar um diferencial de serviços para determinado ambiente de rede.

3.2.1 – IEEE 802.11

O grupo de trabalho responsável pelo IEEE 802.11, desenvolve os padrões *wireless* para comunicação de dados no espectro de radio freqüência de 2,4 GHz e 5 GHz [VAX03][WHE01]. O IEEE 802.11 [ISO99][VAX03] é o protocolo *wireless* mais consolidado para comunicação WLAN, desenvolvido e testado por vários anos em ambientes públicos e privados, sendo uma tecnologia bastante utilizada para interconexão *wireless*.

O IEEE 802.11 define dois modos quanto a sua operação: modo de infra-estrutura e modo *ad hoc* [3CO00]. O modo de infra-estrutura é baseado em uma arquitetura celular sendo o sistema subdividido em células, onde cada célula (chamado BSS – *Basic Service Set*, na nomenclatura IEEE 802.11) é controlada por uma estação base (denominado AP – *Access Point*). Embora uma WLAN possa ser formada por uma única célula, com um único AP (podem trabalhar sem o AP também), a maioria das instalações será formada por várias células, onde os APs estão conectados diretamente de algum modo ao *backbone* (chamado DS – *Distribution System*), tipicamente *Ethernet*, e em alguns casos caracterizado também por tecnologia *wireless*. Todas as interconexões WLAN incluindo as diferentes células, seus respectivos APs e o DS, é visto como uma única rede, chamado ESS (*Extended Service Set*).

A figura a seguir demonstra uma configuração típica de uma WLAN 802.11, com os componentes descritos anteriormente:

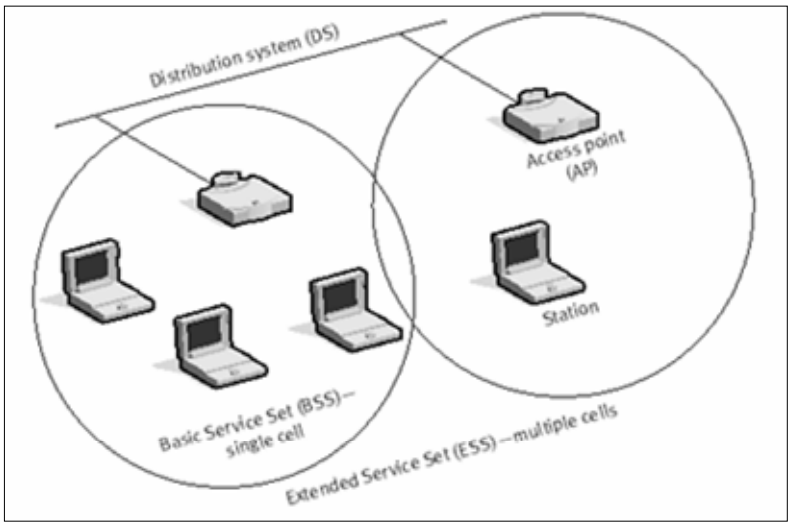

Figura 3.4. Modo de infra-estrutura [3CO00]

Em certas circunstâncias, uma WLAN pode ser configurada para operar de maneira distinta do modo de infra-estrutura, mais especificamente sem um AP. Neste caso, é usualmente denominada como modo de operação *ad hoc* (rede temporária) ou IBSS (*Independent Basic Service Set*), ambos termos são válidos, pois pode ser montada rapidamente sem um planejamento. Como exemplo de utilização, poderíamos imaginar a transferência de arquivos entre computadores (veja figura 3.5). O padrão IEEE 802.11 define esta facilidade como modo de operação *ad hoc*, como comentado anteriormente,

sendo que nestes casos não existe a necessidade de um AP e parte desta funcionalidade é executada pelas estações do usuário final (como sincronização), além de outras funções não suportadas (como economia de energia).

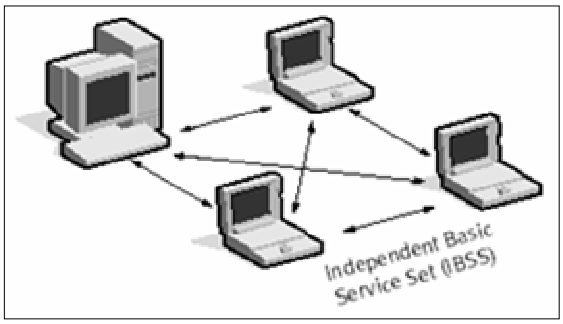

Figura 3.5. Modo *ad-hoc* ou IBSS [3CO00]

Como todos os protocolos da família IEEE 802 [3CO00], o protocolo IEEE 802.11 também está limitado aos níveis 1 e 2 do modelo de referencia OSI (*Open Systems Interconnection*), apresentado na figura 3.6. A especificação define no nível de enlace duas subcamadas: LLC (*Logical Link Control*) e MAC (*Medium Access Control*). O IEEE 802.11 utiliza o mesmo LLC do IEEE 802.2, comum a todas a redes do padrão IEEE 802, e um único MAC interage com as três formas de modulação da camada PHY. Cabe ressaltar, que a tecnologia de infravermelho suportada pelo padrão IEEE 802.11, não foi adotada por nenhum fabricante. Os esquemas de modulação da camada PHY do IEEE 802.11 [3CO00][DAN02][GAS02] são descritos como:

- FH (*Frequency Hopping Spread Spectrum*) na faixa de 2,4 GHz
- DS (*Direct Sequence Spread Spectrum*) na faixa de 2,4 GHz, e
- IR (*InfraRed*)

Em [GAS02], o MAC é definido como sendo um conjunto de regras que especificam o método para acessar o meio e enviar dados, no entanto, os detalhes de transmissão e recepção são de responsabilidade da camada PHY. Além destas funcionalidades normalmente realizadas pela camada MAC, o IEEE 802.11 MAC realiza outras funções que são tipicamente relacionados a protocolos de camadas superiores, como fragmentação, retransmissão de pacotes, e confirmação de recebimento de pacotes (ACK).

Assim como a tecnologia *Ethernet*, o padrão IEEE 802.11 utiliza um esquema CSMA (*Carrier Sense Multiple Access*) para controlar o acesso ao meio (MAC). No entanto, colisões desperdiçam valiosa capacidade de transmissão, sendo assim, ao invés do CSMA/CD (CSMA *Collision Detection*) utilizado pelo padrão *Ethernet*, o IEEE 802.11 utiliza o CSMA/CA (CSMA *Collision Avoidance*). Assim como o padrão *Ethernet*, o IEEE 802.11 utiliza um esquema de acesso distribuído com controle descentralizado. Cada estação IEEE 802.11 utiliza o mesmo método para ter acesso ao meio de transmissão. A maior diferença entre o IEEE 802.11 e o *Ethernet* é fundamentalmente o meio de transmissão.

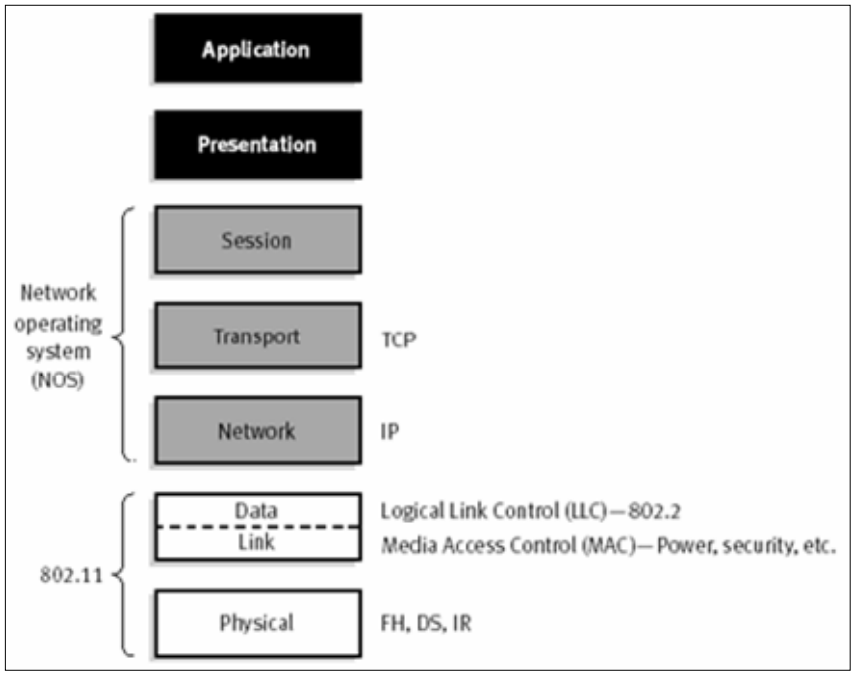

Figura 3.6. IEEE 802.11 e o modelo OSI [3CO00]

Inicialmente os sistemas IEEE 802.11 operavam em uma faixa de 2,4 GHz, atingindo taxas de transferência da ordem de 1 e 2 Mbps, sem qualquer mecanismo para prover QoS (*Quality of Service*) [CHH96][VAX03]. Devido à grande aceitação, novas versões e melhoramentos da especificação foram propostos. A mais usualmente conhecida, a especificação IEEE 802.11b, opera na faixa de 2,4 GHz, com taxas de transferência padronizadas da ordem de 5,5 e 11 Mbps. Para alcançar estas taxas de transferência, a camada PHY do IEEE 802.11b utiliza um esquema de modulação denominado HR/DS (*High-Rate Direct Sequence*) [GAS02][KAR01][VAX03]. Outra

especificação, a IEEE 802.11a, utiliza um espectro de radio freqüência de 5 GHz. A camada PHY do IEEE 802.11a utiliza o esquema de modulação OFDM (*Orthogonal Frequency Division Multiplexing*) [OHR03][VAX03], que garante uma taxa de transferência de 54 Mbps. O padrão mais recente, o IEEE 802.11g, utiliza uma freqüência de 2,4 GHz e possui uma capacidade de transferência da ordem de 54 Mbps. Isto ocorre, graças ao IEEE 802.11g também utilizar o esquema de modulação OFDM [VAX03] na camada PHY. Outra característica interessante, ainda referente ao IEEE 802.11g, diz respeito à compatibilidade que este possui como o padrão IEEE 802.11b.

3.3 – WMAN

Segundo [OHR03], uma WMAN compreende as tecnologias baseadas em rádio freqüência para interconexão *wireless* de algumas centenas de metros até muitas milhas. Duas topologias básicas de rede são suportadas por esses sistemas. O primeiro sistema denominado ponto-a-ponto (veja figura 3.7), mais simples, garante uma alta velocidade de comunicação entre duas localizações fixas. A largura de banda não é compartilhada, entretanto, necessitam de uma linha de visada entre as duas antenas.

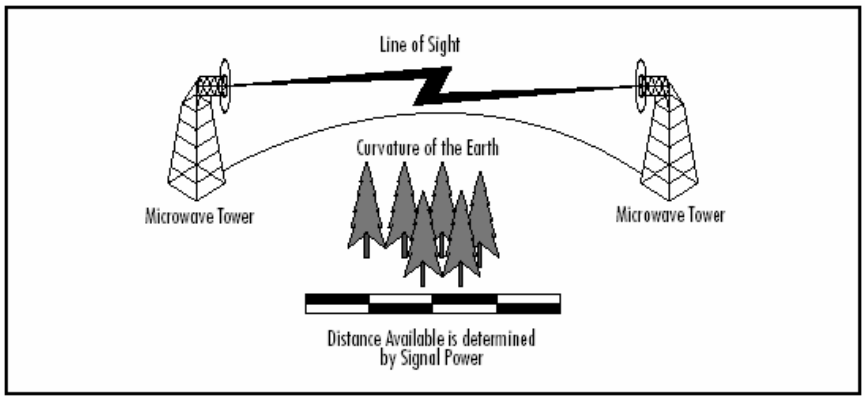

Figura 3.7. Rede ponto-a-ponto [WHE01]

A segunda topologia de rede, denominada ponto-multiponto (veja figura 3.8), envia um sinal a todos os elementos de comunicação (*broadcast*) presentes na área onde atua (célula), comunicando com as antenas fixas presentes na célula. Por esse motivo, a largura de banda na célula é limitada e compartilhada entre todos os seus usuários, uma

vez que o desempenho pode ser uma preocupação em células que possuem alta densidade. Sistemas com diferentes freqüências podem ser combinados para cobrir uma área, onde o terreno e outros obstáculos permitam uma ampla cobertura.

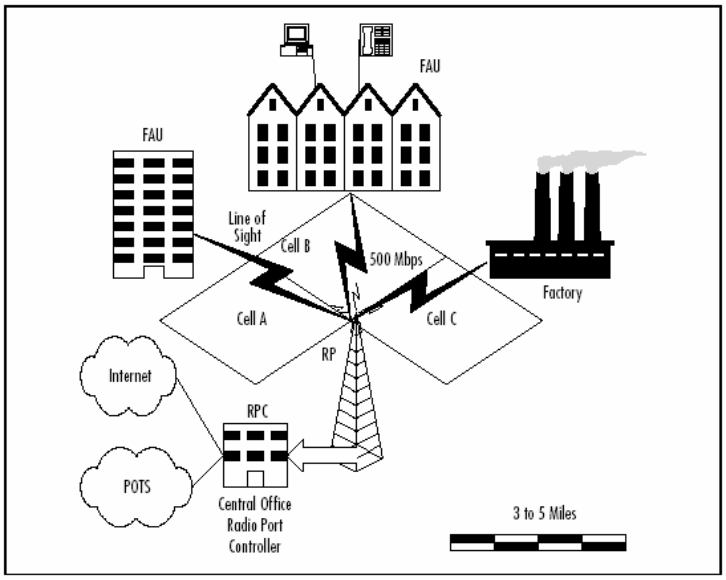

Figura 3.8. Rede ponto-multiponto [WHE01]

3.3.1 – IEEE 802.16

O padrão IEEE 802.16 [EKL02][IEE02], define a especificação de uma interface aérea para WMANs. A tecnologia abre uma nova opção para o mercado de acesso das redes residenciais e corporativas aos grandes *backbones* das provedoras de serviços de telecomunicações. Nesse sentido, o padrão IEEE 802.16 [OHR03], apóia a especificação de uma interface aérea para comunicação nas redes WMANs entre as estações transceptoras e a estação base transceptora, visando atingir uma taxa de transferência da ordem de 2 a 155 Mbps. Na prática, o padrão IEEE 802.16 atualmente atinge taxas de dados compartilhadas de até 75 Mbps [INT03]. Na figura 3.9 apresentamos a configuração típica de uma WMAN IEEE 802.16.

Originalmente, o padrão IEEE 802.16 [EKL02][IEE02], aprovado em outubro de 2001 e publicado em 8 de abril de 2002, prevê basicamente o uso de freqüências situadas entre 10 e 66 GHz, com uma possível extensão para a faixa de 2 a 11 GHz. As freqüências mais baixas permitem sistemas de menor custo e maior alcance, porém utilizam uma faixa do espectro mais concorrida e oferecem taxas de transmissão de

dados menores. Em adição, o grupo de trabalho responsável pelo padrão IEEE 802.16, finalizou a especificação necessária para a utilização de sistemas de acesso WMAN fixo [EKL02][IEE01][WHE01] coexistir com o padrão IEEE 802.16.

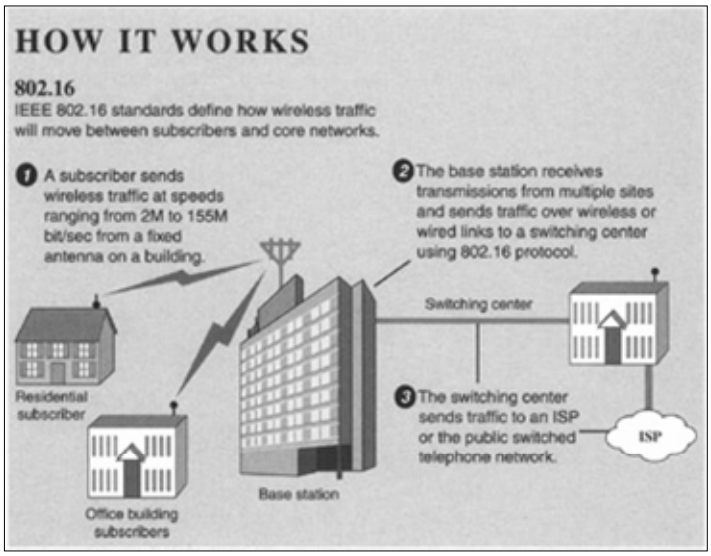

Figura 3.9. WMAN IEEE 802.16 [OHR03]

Com relação ao protocolo MAC, o padrão IEEE 802.16 foi projetado prevendo a necessidade de atender a uma ampla gama de serviços para diferentes usuários que compartilham um canal bidirecional de alta velocidade. O protocolo permite serviços de voz e dados com multiplexação por divisão de tempo (TDM – *Time Dvision Multiplex*) [EKL02][OHR03], voz sobre IP (VoIP – *Voice Over* IP) e transmissão assíncrona de pacotes de dados. Os serviços se assemelham aos oferecidos por uma rede ATM [DAN02][SOA95], acrescidos de serviços como taxa de quadros garantida (GFR – *Guaranteed Frame Rate*) [EKL02][OHR03]. A estrutura de transmissão que atende esses múltiplos serviços é formada por uma camada MAC flexível, somada a um conjunto de funções de convergência que adaptam o tráfego de diferentes serviços, tornando mais eficiente à transmissão na camada de enlace.

O protocolo MAC, por sua vez, pode interagir com os seguintes esquemas de modulação da camada PHY do IEEE 802.16 [EKL02]:

- *Single-Carrier*
- OFDM (*Orthogonal Frequency-Division Multiplexing*)
- OFDMA (*Orthogonal Frequency-Division Multiple Access*)

Em função das condições de operação de cada esquema de modulação, pode se adotar, em condições ideais uma modulação mais eficiente, entretanto, quando as condições são desfavoráveis, é a confiabilidade da comunicação que deve receber a atenção prioritária. O padrão IEEE 802.16 teve ainda o cuidado de definir mecanismos de alocação de banda e de controle de QoS dentro dos limites necessários para a interoperabilidade de equipamentos. No entanto, os detalhes de gerenciamento eficiente de reserva de banda ficaram indefinidos, de modo que cada fabricante possa definir a forma de implementação que julgar mais adequada.

Como pode se constatar, o protocolo de enlace (MAC) foi projetado para permitir a comunicação ponto-multiponto entre uma estação central e equipamentos terminais periféricos com múltiplas formas de codificação simultâneas. Assim, na direção *downlink* o compartilhamento do canal utiliza um esquema TDM, por outro lado, no sentido *uplink* o acesso ao canal utiliza um esquema do tipo acesso múltiplo por divisão de tempo (TDMA – *Time Division Multiple Access*) [EKL02][OHR03].

3.4 – WWAN

Recentemente, um grande número de tecnologias para as redes WWANs tem surgido, incluindo as tecnologias: GSM/GPRS (*Global System Mobile*/*General Packet Radio Service*) [ZHA03], e CDMA/1xRTT (*Code Division Multiple Access*/*1xRadio Transmission Technology*) [CHH02][PAR01] que serão apresentadas em detalhes nas seções seguintes, por se tratarem de tecnologias que estão em grande evidencia no momento. O padrão IEEE 802.20 [INT03] para WWANs não será explorado, uma vez que o mesmo ainda está sendo proposto.

3.4.1 – GSM/GPRS

O GSM [ARA01][SAL02] original é um padrão digital de segunda geração (2G), desenvolvido inicialmente como um sistema otimizado para voz, tendo uma baixa taxa para a transferência de dados, cerca de 9,6 Kbps. O crescimento das aplicações de dados como acesso a Internet, e-mail, entretenimento, levou à necessidade de desenvolver

soluções que permitissem o transporte de dados a taxas maiores. Nesse sentido, o GSM evoluiu para a tecnologia comumente chamada como GSM/GPRS.

A tecnologia GSM/GPRS [ARA01][SAL02] é um padrão digital de segunda geração e meio (2.5G) que oferece serviços de dados por pacotes, ou seja, é uma tecnologia voltada para transmissão de dados caracterizando uma configuração típica de uma WWAN, uma vez que pode englobar uma vasta região como um estado, um país ou até mesmo um continente. A tecnologia GSM/GPRS pode ser considerada um passo intermediário na evolução para a terceira geração (3G). As principais características da tecnologia GSM/GPRS são descritas a seguir [GSM04]:

- Taxa de transporte de dados teórica de 171,2 Kbps;
- Padronizado para o transporte de dados definidos pelos protocolos IP e X.25 [SOA95].

A figura a seguir (veja figura 3.10) apresenta a configuração típica de uma rede GSM/GPRS:

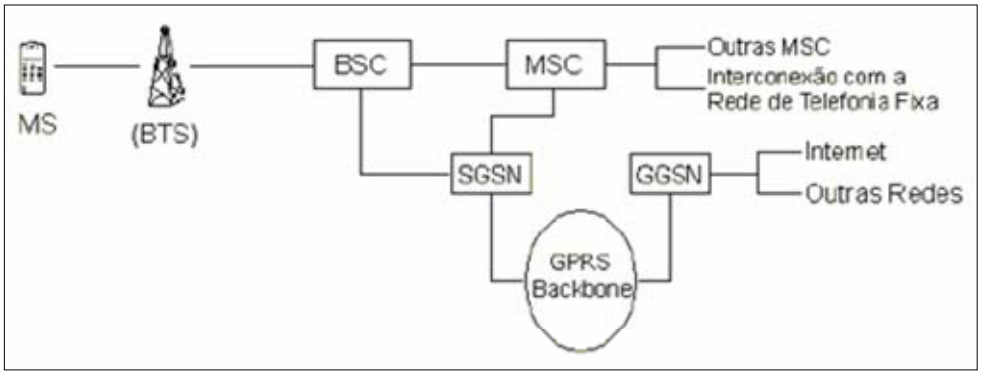

Figura 3.10. Configuração de uma rede GSM/GPRS

A partir da figura 3.10, podemos entender melhor o funcionamento de uma rede GSM/GPRS [SAL02]. O *Base Station System* (BSS) é o sistema encarregado da comunicação com as *Mobile Stations* (MSs), sendo formado por várias *Base Transceiver Stations* (BTSs) que constituem uma célula, e um *Base Station Controller* (BSC), responsável por controlar as BTSs. A seguir temos o *Mobile-Services Switching Centre* (MSC) tendo a função de comutação e sinalização para as MSs, sendo que o MSC encarregado do roteamento de chamadas para outros MSCs é usualmente denominado de *Gateway* MSC. Os dois elementos que completam o diagrama, agora em

nível de *software*, de modo a formar a rede GSM/GPRS são o *Serving GPRS Support Node* (SGSN), cuja principal finalidade é manter a conexão lógica dos usuários quando eles se movem da área de cobertura de uma célula para outra (*handover*), e o GGSN (*Gateway GPRS Support Node*), que permite a conexão com a Internet e outras redes de dados. É interessante comentar, que estes nodos estão conectados a um *backbone* GPRS do qual fazem parte outros SGSNs e GGSNs completando a configuração da rede GSM/GPRS.

Para ter acesso a rede GSM/GPRS é necessário ter uma MS que tenha suporte a este serviço obviamente. Uma vez satisfeito este requisito, a conexão da MS a rede GSM/GPRS ocorre de maneira semelhante ao que ocorre com uma MS dedicada a serviços de voz, sendo criado então um enlace lógico entre o MS e o SGSN. Neste momento a MS é registrada e autenticada na rede. O próximo passo é conseguir um endereço IP estabelecendo uma conexão na rede GSM/GPRS. Este endereço IP é usualmente obtido dinamicamente, sendo fornecido pela operadora móvel, ou outro operador dependendo de como está configurada a rede. Finalmente, a MS está então pronta para enviar e receber pacotes de dados na rede GSM/GPRS.

3.4.2 – CDMA/1xRTT

O primeiro padrão CDMA [CDG04][PAR01], conhecido como CDMA IS-95A, é considerado um padrão digital de 2G, oferecendo serviços de voz e dados a uma taxa máxima de 14,4 Kbps, como apresentado no diagrama de rede (veja figura 3.11). Assim como aconteceu com o GSM, a tecnologia CDMA também se deparou com a necessidade de desenvolver soluções que permitissem o transporte de dados a taxas mais elevadas, devido ao crescimento das aplicações de dados. Por esta razão, o CDMA desenvolveu um novo padrão denominado CDMA/1xRTT para suprir estas necessidades. Cabe ressaltar que o padrão digital CDMA IS-95B de 2.5G, pode ser considerado como um passo intermediário para o aprimoramento da tecnologia CDMA, uma vez que evolui do padrão CDMA IS-95A e serviu de base para o padrão CDMA 1xRTT. Como principal característica, o CDMA IS-95B [CDG04] possui capacidade para a transferência de dados a uma taxa teórica de 64 Kbps, em adição aos serviços de voz já suportados pelo padrão.

Por outro lado, o CDMA/1xRTT [CDG04] conhecido também como CDMA 2000 1X, ambos termos são válidos, oferece taxa de dados maiores, cerca de 153,6 Kbps, quando comparado com o padrão CDMA IS-95A original e com o padrão CDMA IS-95B. A tecnologia CDMA/1xRTT é um padrão digital de segunda geração e meio (2.5G), e desta forma concorre diretamente com o padrão GSM/GPRS, por se tratarem de tecnologias consideradas um passo intermediário na evolução para a 3G.

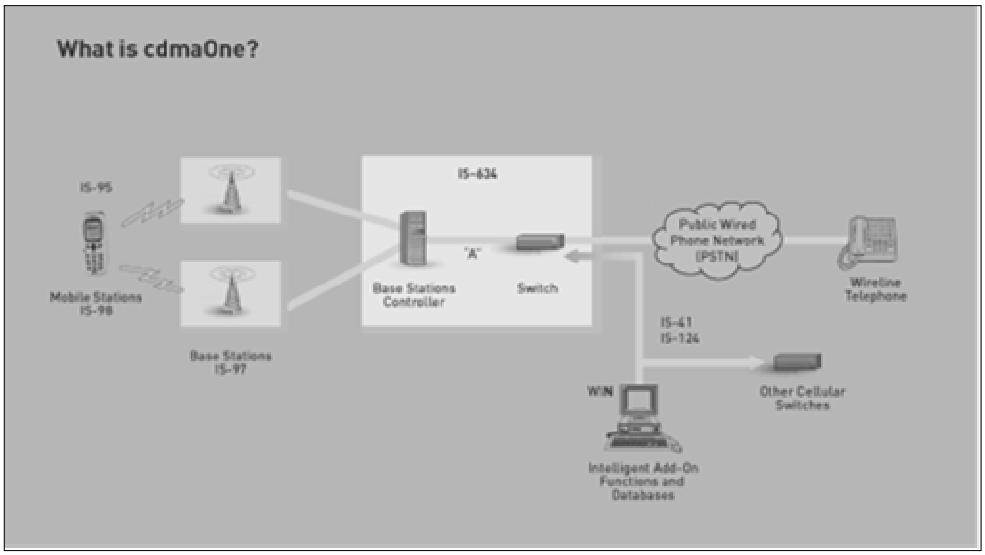

Figura 3.11. Diagrama de uma rede CDMA [CDG04]

O modelo de transmissão em uma rede CDMA/1xRTT [CHH02] é demonstrado na figura abaixo (veja figura 3.12). Por outro lado, o processo de recepção é a operação inversa, com o receptor e decodificador devidamente configurado para o recebimento de dados. Em uma linguagem mais simples, podemos dizer que o dispositivo móvel interage com as antenas por meio de sinais de rádio codificados. Quando utilizamos o CDMA/1xRTT, os milhões de bits que representam nossa voz (ou dados) são codificados numericamente e transmitidos, em um conjunto mais amplo de freqüências (*Spread Spectrum*).

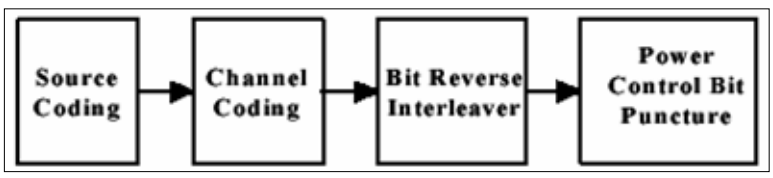

Figura 3.12. Descrição de um sistema CDMA/1xRTT [CHH02]

4 – AMBIENTE EXPERIMENTAL

Este capítulo aborda os principais tópicos sobre a configuração do ambiente experimental indispensável para a monitoração *wireless*, sendo organizado em duas partes. Inicialmente, discutem-se as principais características da ferramenta escolhida para a configuração do *cluster* computacional e seu módulo que permite a capacidade de alta disponibilidade. Em seguida, são apresentados os conceitos relacionados ao ambiente de programação, procurando focar a sua utilização em dispositivos móveis.

Cabe destacar que todas as ferramentas e pacotes de *software* apresentados neste capítulo possuem distribuição gratuita e código aberto, assim como o sistema operacional Linux utilizado na configuração do ambiente experimental. No nosso caso, essa facilidade era um ponto fundamental para a implementação e configuração de vários elementos essenciais.

4.1 – Configuração do Ambiente de Alto Desempenho

Para a instalação do ambiente de alto desempenho utilizamos o *cluster* computacional OSCAR (*Open Source Cluster Application Resource*). A capacidade de alta disponibilidade foi obtida mediante a adição do módulo HA-OSCAR (*High Availability – Open Source Cluster Application Resource*). O principal fator que motivou a escolha desta configuração para o ambiente de alto desempenho incide no fato do OSCAR ser uma ferramenta de código aberto e cuja utilização já atinge cerca de vinte e três porcento [LIG03] das configurações de *clusters* computacionais da comunidade técnico-científica. A capacidade de alta disponibilidade também foi decisiva para a escolha, sendo obtida graças à adição do módulo HA-OSCAR, como havíamos comentado anteriormente.

4.1.1 – OSCAR

Segundo [OSC03], o OSCAR é um pacote de *software* que permite simplificar a complexa tarefa de utilização e gerenciamento de um *cluster*. O OSCAR é essencialmente utilizado para a computação de alto desempenho, podendo perfeitamente

ser utilizado por qualquer aplicação, que necessite das funcionalidades de um *cluster*, para obter um aumento em seu desempenho através da exploração do paralelismo. Cabe ressaltar alguns pacotes padrões instalados nativamente, como implementações de MPI (*Message Passing Interface*) [MPI94], de PVM (*Parallel Virtual Machine*) [GEI94] e de PBS (*Portable Batch System*) [PBS03].

No OSCAR [OSC03], cada computador individual do *cluster* é usualmente denominado de nodo. Existem dois tipos de nodos: nodo servidor e nodo cliente. O nodo servidor é responsável pelo controle, atribuição e requisição de tarefas aos nodos clientes. Por sua vez, o nodo cliente é dedicado especificamente ao processamento de tarefas. Em outras palavras, podemos dizer que a configuração do *cluster* OSCAR [OSC03] consiste essencialmente de um nodo servidor, e de um ou mais nodos clientes, sendo que os nodos clientes devem possuir *hardware* homogêneo.

Uma prática comum em *clusters* é o uso da biblioteca PBS, responsável por controlar a submissão e execução das tarefas no nodo servidor, combinada com alguma implementação MPI ou PVM para o ambiente paralelo. As bibliotecas de troca de mensagens como MPI e PVM fornecem a capacidade necessária ao ambiente OSCAR para a criação de programas paralelos em um ambiente de computação distribuída. Na figura 4.1 apresentamos justamente a hierarquia destas bibliotecas no *cluster* OSCAR.

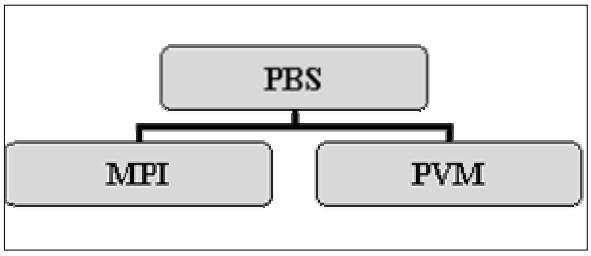

Figura 4.1. Hierarquia das bibliotecas no OSCAR

Para a instalação do gerenciador OSCAR versão 3.0, veja figura 4.2, foi utilizado o sistema operacional Red Hat Linux 9.0. A escolha pela distribuição ocorreu após uma criteriosa seleção entre as demais distribuições suportadas por esta versão do gerenciador OSCAR [COL04][OSC03][RIS04]. Os critérios que adotamos estão relacionados essencialmente com desempenho, robustez e confiabilidade. O ambiente de execução caracterizou-se por cinco máquinas Intel Pentium 4 de 1,8 GHz com 512 MB de memória principal, disco rígido com 40 GB e como rede de interconexão

utilizamos um *switch Fast Ethernet*. Cabe destacar que este ambiente encontra-se atualmente no Laboratório de *Web Software* (LABWEB) localizado no Departamento de Informática e Estatística (INE) da Universidade Federal de Santa Catarina (UFSC).

Figura 4.2. Tela principal de instalação do OSCAR

4.1.2 – HA-OSCAR

Com o crescente avanço das atividades que fazem uso de sistemas computacionais de alto desempenho, impulsionados principalmente pela maior utilização do paradigma de c*luster* computacional, surge uma preocupação muito grande no que diz respeito a problemas causados por eventuais defeitos destes sistemas. Por outro lado, a alta disponibilidade é uma das soluções que visa evitar estes defeitos.

O pacote de *software* HA-OSCAR [LEA04][MOS04][NAU03], em adição ao sistema gerenciador OSCAR, contempla a característica de alta disponibilidade que possibilita prevenir possíveis defeitos que ocasionalmente possam ocorrer ao ambiente de *cluster* computacional. Na prática, alta disponibilidade torna-se fundamental para aquelas aplicações consideradas críticas, com grandes volumes de dados normalmente complexos, que neste caso, se beneficiam também da computação de alto desempenho. Em nosso caso, esta facilidade era um ponto fundamental para a preparação e execução do ambiente, uma vez que para executar eficientemente uma aplicação de âmbito crítico

em um *cluster* computacional, técnicas de computação com alta disponibilidade se fazem necessárias para prevenir defeitos, caso os mesmos venham a ocorrer.

A seguir apresentamos na figura 4.3, a arquitetura do HA-OSCAR que consiste essencialmente dos seguintes componentes [LEA04]:

- Servidor primário: responsável pela submissão e distribuição de tarefas aos nodos clientes. Cada servidor possui três interfaces de rede; uma conectada a Internet por um endereço público; e as outras duas são conectadas a uma LAN privada, que consiste em uma ligação *Ethernet* primária e uma secundaria (*standby*). Entretanto, esta primeira versão beta (HA-OSCAR 1.0) suporta somente uma interface de rede privada.
- Servidor secundário (*Standby server*): monitora o servidor primário e assume seus serviços caso seja detectado algum defeito na configuração.
- Múltiplos clientes: são dedicados ao processamento de tarefas.
- *Switches*: Provem ligação local entre os nodos clientes, além do servidor primário e servidor secundário.

A computação com alta disponibilidade difere em alguns aspectos da computação tolerante à falha. Uma vez que a computação com alta disponibilidade é pró-ativa [MOS04][NAU03], ou seja, detectando e prevenindo potenciais ocorrências de defeito. Por outro lado, a computação tolerante à falha [MOS04][NAU03] é usualmente mais dispendiosa e reativa. Em computação tolerante à falha, usualmente os componentes replicados executam as mesmas instruções ao mesmo tempo, sendo que se um componente falhar a aplicação continua a sua execução, sem nenhuma outra diferença a não ser no nível de desempenho reduzido com base na porcentagem dos recursos perdidos. Um custo significante da tolerância à falha é a execução redundante de tarefas e a verificação de pontos de checagem freqüentemente, mesmo quando não há ocorrência de uma falha.

Para que o sistema de computação de alto desempenho contemple de maneira efetiva a característica de alta disponibilidade, se faz necessário à adoção de algumas estratégias. Estas estratégias envolvem a duplicação de *hardware* e redundância da rede, técnicas bastante comuns para permitir a confiabilidade e disponibilidade necessária para estes sistemas. Cabe destacar que a monitoração entre os servidores (primário e secundário) é realizada através do conceito de *heartbeat*, que é o principal responsável

pela monitoração do *cluster* HA-OSCAR [NAU03]. O conceito de IP virtual também é importante em um *cluster* HA-OSCAR, pois é definido como o endereço IP do serviço, e não o endereço IP real da máquina. Desta forma, o endereço IP virtual é assumido pela máquina momentaneamente responsável pelo serviço e pode ser assumido por outra máquina *standby* quando a primeira apresentar defeito [LEA04][MOS04].

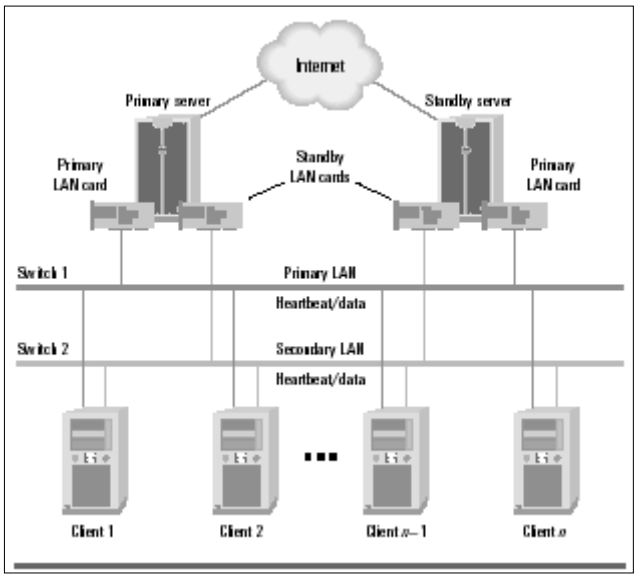

Figura 4.3. Arquitetura do *cluster* HA-OSCAR [NAU03]

Para adicionarmos o pacote de *software* HA-OSCAR, havia a necessidade de que o OSCAR estivesse previamente instalado e configurado para assegurar ao *cluster* computacional a capacidade de alta disponibilidade. Para a instalação do HA-OSCAR 1.0 versão beta, veja figura 4.4, utilizamos o ambiente OSCAR já descrito e devidamente configurado como apresentamos na seção anterior (4.1.1 - OSCAR), onde temos o OSCAR versão 3.0 e Red Hat Linux 9.0, ambos suportados pelo HA-OSCAR. O ambiente de execução caracterizou-se pela mesma configuração do OSCAR, composta de cinco máquinas Intel Pentium 4 de 1,8 GHz com 512 MB de memória principal, disco rígido com 40 GB e como rede de interconexão utilizamos um *switch Fast Ethernet*. A principal diferença, especificamente com relação à arquitetura, incide no fato de que o HA-OSCAR tem um servidor primário e um secundário, além de três máquinas clientes. Diferentemente do OSCAR que possuía apenas o servidor primário e as demais máquinas eram nodos clientes.

Figura 4.4. Tela principal de instalação do HA-OSCAR

4.2 – Ambiente de Programação

Os programas desenvolvidos para dispositivos móveis devem possuir simplicidade e facilidade para a sua utilização. As aplicações desenvolvidas para esses dispositivos devem considerar algumas restrições quanto à quantidade de memória, o armazenamento de dados, o consumo de energia e a sua conectividade, além de restrições com relação ao tamanho da tela destes dispositivos [FOS00].

Com base nesta constatação, e sabendo das restrições de recursos impostas pelos dispositivos móveis, utilizamos para o desenvolvimento da ferramenta de monitoração *wireless* a linguagem de programação C. O compilador utilizado foi o GCC [GRI02] (*GNU C Compiler*) versão 3.2.2, presente na distribuição Red Hat Linux 9.0 utilizada em nosso ambiente de alto desempenho. Para conseguirmos a portabilidade de código necessária ao dispositivo móvel, um PDA da Palm em nosso caso, foi necessário à utilização de mais quatro ferramentas básicas: PRC-Tools, *Palm OS Software Development Kit* (Palm OS SDK), PilRC e *Palm OS Emulator* (POSE).

4.2.1 – PRC-Tools, Palm OS SDK, PilRC e POSE

O Palm OS [FOS00][PAL04][POG99] é o sistema operacional desenvolvido pela Palm para ser utilizado em dispositivos móveis. Até a versão 4.0 desse sistema, o

processador utilizado nesses dispositivos foi da família Motorola 68000 [FOS00]. A partir da versão 5.0, o processador utilizado nesses dispositivos possui conformidade com a família de processadores ARM. Os processadores da série ARM [PAL04] referem-se a uma família de processadores com avançada tecnologia RISC (*Reduced Instruction Set Computer*). Para que o código C desenvolvido inicialmente no Red Hat Linux 9.0 pudesse ser compilado e portado para o sistema operacional Palm OS, houve a necessidade de utilização de um conjunto de ferramentas que fornecessem essa capacidade. A capacidade foi obtida mediante a utilização do conjunto de ferramentas: PRC-Tools, Palm OS SDK, PilRC e POSE.

Segundo [FOS00][PAL04] o PRC-Tools é um conjunto completo de ferramentas de compilação baseado no ambiente de programação GCC, suportado pelas plataformas Unix/Linux, para construção de aplicativos C/C++ para Palm OS. Inicialmente realizamos a instalação do PRC-Tools versão 2.3 através de um pacote RPM (*Red Hat Package Manager*) com o comando "rpm -Uhv prc-tools-2.3-1.i386.rpm". Logo após, houve a necessidade de instalarmos também uma versão do Palm OS SDK fornecido pela Palm, como veremos no próximo parágrafo. Após a instalação do SDK, é preciso executar o comando "palmdev-prep" como *root* para preparar o GCC e o PRC-Tools para reconhecer o SDK instalado. Cada vez que um SDK for instalado ou desinstalado, é preciso executar o comando "palmdev-prep" novamente. É importante destacar também que o programa é compilado através do comando "m68k-palmos-gcc".

O Palm OS SDK [FOS00][POG99] ou simplesmente SDK, caracteriza-se como o conjunto de cabeçalhos, bibliotecas e documentação da API (*Application Programming Interface*) do Palm OS necessários para o desenvolvimento e construção de aplicações em C/C++. O SDK contém a API que descreve o método específico para fazer solicitações ao sistema operacional ou a outros aplicativos Palm OS. O pacote RPM contendo a versão 4.0 do Palm OS SDK, foi instalado através do comando "rpm -Uhv palmos-sdk-4.0-1.noarch.rpm". Uma vez que o PRC-Tools e o SDK estejam devidamente instalados, devemos executar o comando "palmdev-prep", como explicamos anteriormente, para que o GCC e o PRC-Tools reconheçam o SDK instalado. Segundo a Palm [PAL04], as versões mais recentes dos SDKs geralmente suportam os dispositivos e versões do Palm OS mais antigas. Não existe nenhuma razão então para utilizarmos SDKs mais antigos, exceto pelo interesse histórico talvez.

A próxima ferramenta denominada PilRC [FOS00][POG99], é um compilador de recursos que transforma arquivos de texto com a extensão ".rcp" em recursos gráficos para a plataforma Palm OS. Em outras palavras, o PilRC é um compilador de recursos que transforma descrições textuais da interface do usuário e outros recursos em formato binário em um formato suportado pelo sistema operacional Palm OS. Para a instalação do pacote RPM contendo a versão 2.9 do PilRC, executamos o comando "rpm -Uhv pilrc-2.9p2-3.i386.rpm". Cabe ressaltar também que a execução do PilRC acontece através do comando "pilrc".

E finalmente a ferramenta POSE, veja figura 4.5, completa a especificação do ambiente de programação. Segundo [FOS00][POG99] o POSE é definido como um *software* que emula o *hardware* de vários modelos de PDAs da Palm. Para que o POSE funcione corretamente, é necessário um arquivo chamado ROM (*Ready Only Memory*). A ROM é a imagem da memória do PDA que permite emular diferentes versões do Palm OS. Para se obter uma imagem ROM, podemos fazer o seu *download* diretamente no *site* da Palm [PAL04], ou utilizarmos a opção "*Rom Transfer*" do POSE para copiar a imagem real de um Palm através de um programa de sincronização convencional como o *KPilot* do Linux. Cabe destacar ainda, que existem dois tipos de ROM de acordo com o tipo de serviço, com e sem detecção de erros (*debug ou non-debug*). O pacote RPM com a versão 3.5 do POSE foi instalado através do comando "rpm -Uhv pose-3.5-2.i386.rpm". Para a execução do POSE utilizamos o comando "pose -&".

Figura 4.5. Tela principal do *Palm OS Emulator* (POSE)

5 – MONITORAÇÃO WIRELESS

Este capítulo apresenta a descrição da ferramenta de monitoração *wireless* desenvolvida. Inicialmente, têm-se algumas considerações iniciais a cerca de monitores tradicionais. Em seguida, a fase de desenvolvimento da abordagem de monitoração proposta por este trabalho é detalhada. Finalizando o capítulo, são apresentados os resultados obtidos e as considerações finais.

5.1 – Considerações Iniciais

Um monitor [JAI91] é uma ferramenta utilizada para observar as atividades de um sistema. Em geral, os monitores são usados para observar o desempenho de sistemas computacionais coletando dados estatísticos, analisando os dados e mostrando os seus resultados. Alguns monitores também identificam problemas e sugerem possíveis soluções. Todavia, os monitores podem ser utilizados também em diferentes situações, entre elas podemos citar [JAI91]:

- Aperfeiçoar um sistema, através do ajuste de alguns de seus parâmetros, de maneira a aumentar o desempenho;
- Medir a utilização dos recursos, permitindo descobrir gargalos de desempenho no sistema;
- Fornecer dados para modelar um sistema, possibilitando também fazer a sua parametrização e validação.

Os monitores podem ser classificados de acordo com as suas características. Por exemplo, podemos classificar os monitores quanto à forma como são acionados. Neste caso, têm-se então dois tipos de monitores: monitor baseado em eventos e monitor baseado no tempo. O monitor baseado em eventos, é disparado por algum evento que aciona o monitor. Por outro lado, o monitor baseado no tempo é disparado em intervalos de tempo pré-determinados, ou seja, o monitor está associado a um temporizador que é responsável pelo seu acionamento. Os monitores diferem também quanto à forma como eles apresentam os dados coletados. Se isso é feito de forma contínua ou temporizada, chamamos de monitor *on-line*. Por outro lado, se a apresentação dos dados coletados é feita apenas ao final do processo de coleta, chamamos de monitor em *batch.*

Existe ainda uma outra classificação para os vários tipos de monitores quanto à sua forma de implementação. Dessa forma, divide-se os monitores em quatro categorias distintas [JAI91][NUT72]: *software, hardware, firmware* e monitores híbridos. Em geral, o monitor implementado por *software* disputa os recursos (como processador e memória por exemplo) com o sistema a ser monitorado, o que implica em um taxa de entrada baixa e um *overhead* alto. Este tipo de monitor é capaz de monitorar informações de alto nível facilmente, porém não consegue observar eventos de baixo nível com facilidade. Já o monitor implementado por *hardware*, caracteriza-se pela existência de um *hardware* acoplado ao sistema a ser monitorado através de interfaces que capturam sinais elétricos. O monitor implementado por *hardware* é capaz de monitorar sinais elétricos de um barramento e gravá-los precisamente até mesmo em alta velocidade. Entretanto, possui dificuldade para analisar ou monitorar informações de alto nível, tais como eventos do sistema operacional, a não ser que a informação seja gravada em um registrador do *hardware*. Por outro lado, o monitor híbrido procura unir as vantagens de ambas as abordagens de implementação, contemplando uma combinação entre *software* e *hardware*. Finalmente, o monitor *firmware* caracteriza-se por acrescentar instruções de monitoração no próprio micro-código de um processador.

5.2 – Ferramenta Desenvolvida

A ferramenta desenvolvida foi denominada "HA-OSCAR *Monitoring*", e contempla os objetivos idealizados no início do projeto para a monitoração de um ambiente de *cluster* computacional com alta disponibilidade, através de dispositivos móveis com tecnologia WLAN (IEEE 802.11b em nosso caso). O monitor proposto deverá avaliar continuamente o sistema computacional de alto desempenho, através da coleta e análise de dados estatísticos, e informar em intervalos de tempo pré-determinados, a situação do sistema para o dispositivo móvel através do envio de alarmes. Por essa razão, o monitor proposto deverá trabalhar em conjunto com o ambiente de *cluster* computacional para notificar periodicamente o responsável pelo sistema sobre possíveis ocorrências de defeitos na configuração. O resultado é uma concepção de monitoração diferenciada, pelas próprias características de mobilidade que as redes *wireless* oferecem aos processos móveis, bem como a interação do

ambiente de alto desempenho com os dispositivos móveis. Na figura 5.1, podemos observar a visão conceitual da abordagem de monitoração *wireless* para o ambiente de *cluster* computacional com alta disponibilidade HA-OSCAR.

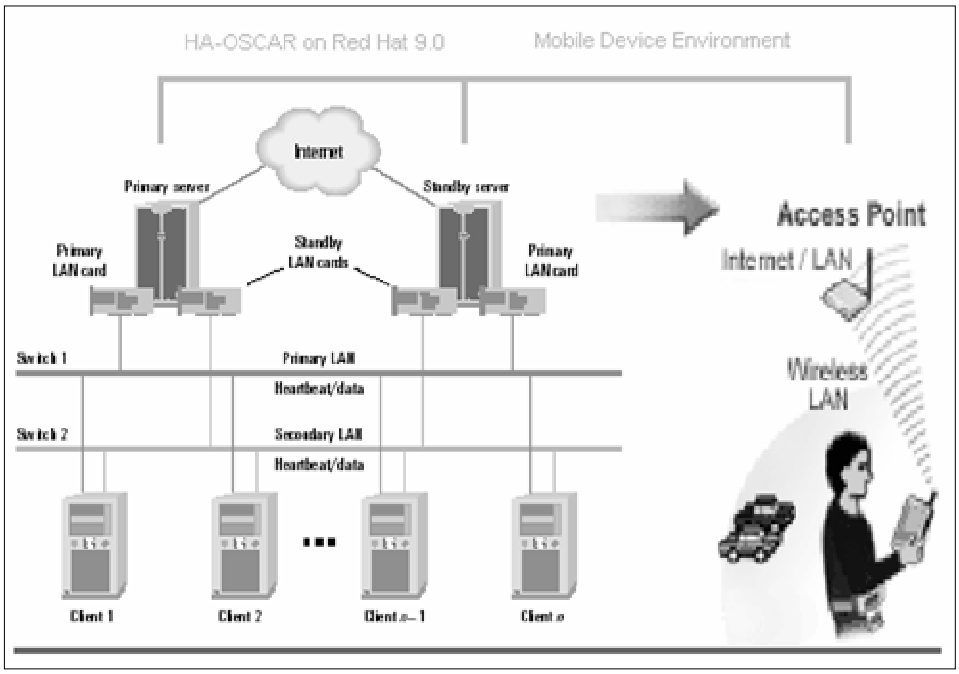

Figura 5.1. Visão conceitual da monitoração *wireless*

Para viabilizar a implementação da ferramenta de monitoração, o código escrito foi testado inicialmente no sistema operacional Linux e posteriormente portado para o sistema operacional Palm OS. Foram criados então dois módulos: “haoscar_standby.c” e “haoscar_mon.c”. O primeiro módulo é responsável pela monitoração do servidor primário e encontra-se em execução no servidor secundário. É importante lembrar que este módulo também deverá enviar as notificações ao módulo “haoscar-mon.c”. O módulo “haoscar-mon.c” por sua vez, tem como objetivo receber as notificações e exibir na tela os resultados de monitoração recebidos. Nesse primeiro momento, o módulo “haoscar-mon.c” funcionou como se fosse o dispositivo móvel para efetuarmos todos os testes necessários no próprio servidor secundário.

Neste momento temos condições de explorar mais detalhadamente os métodos envolvidos para a monitoração proposta. Como mencionamos a pouco, o módulo “haoscar_standby.c” possui todos os procedimentos necessários para a monitoração do servidor primário. Enfatiza-se também, que este módulo será o mesmo quando

portarmos o código para a plataforma Palm OS. A justificativa é simples, pois este módulo será executado apenas pelo servidor secundário, não havendo necessidade de portar o seu código para o dispositivo móvel. Apenas o módulo "haoscar-mon.c" deverá ser portado para a plataforma Palm OS, pois será executado nativamente pelo dispositivo móvel apresentando os resultados da monitoração.

O módulo "haoscar_standby.c" utiliza o conceito de *heartbeat* para realizar a monitoração do servidor primário. No entanto, não foi possível a utilização do *heartbeat* do HA-OSCAR em nossa ferramenta. A razão é simples, pois o HA-OSCAR permite que seja monitorada apenas uma interface de rede privada nesta primeira versão a qual obtivemos acesso. Essa deficiência deve ser corrigida em breve, além de outras melhorias previstas para a versão oficial do pacote HA-OSCAR [LEA04]. Outro fator que contribuiu para a criação de um novo *heartbeat*, incide especificamente na complexidade e dificuldade encontradas em adequar o *heartbeat* do HA-OSCAR para a nossa solução. Sendo assim, foi desenvolvido um novo *heartbeat* para contemplar essa característica de maneira que não comprometesse a capacidade de alta disponibilidade presente no HA-OSCAR.

O *heartbeat* foi desenvolvido a partir de uma chamada de sistema feita ao sistema operacional Linux, através do uso da função *system*, que provê este tipo de serviço ao compilador GCC. Por meio desta função, efetuamos então uma chamada de sistema com o comando *ping*. A função principal do comando *ping* foi de enviar e receber solicitações de retorno do sinal para a fonte de onde foi transmitido através do protocolo ICMP (*Internet Control Message Protocol*). O ICMP [DAN02] por sua vez, prove os mecanismos de mensagens necessários para o controle na comunicação entres nodos em um ambiente de rede TCP/IP (*Transmission Control Protocol/Internet Protocol*). Na figura 5.2, apresentamos o trecho de código contendo a função completa do *heartbeat* no módulo "haoscar_standby.c".

Ao observamos o código fonte novamente, verificamos a existência de alguns parâmetros na função *system*. Estes parâmetros foram acrescentados de modo a customizar o comando *ping* para nosso *heartbeat*. A seqüência "ping -qc 3 127.0.0.1" por exemplo, especifica que este comando deverá "escutar" o meio três vezes e exibir os resultados de forma resumida. Cabe salientar, que o comando foi executado procurando atingir o endereço IP especificado. A seqüência restante "> /root/heartbeat.db" não faz

parte da sintaxe do comando *ping*, mas é utilizada para redirecionar a saída de um comando, *ping* em nosso caso, para um arquivo de texto. Nesse caso, a saída gerada resultou no arquivo “heartbeat.db” criado no diretório “/root” do servidor secundário.

Em seguida, o arquivo de texto “heartbeat.db”, aberto através das funções *open* e *read*, é atribuído a um ponteiro do tipo *void* que será utilizado pela função *strstr*. A função *strstr* servirá então para verificar a ocorrência dos pacotes perdidos. Apenas se houver uma perda de 100% dos pacotes, o *heartbeat* retorna uma possível situação de defeito na configuração. Caso contrario, retorna uma situação de normalidade. Não poderíamos deixar de mencionar também, que o *heartbeat* é acionado em ciclos de cinco segundos, graças à função *sleep* que suspende a execução do programa. Este valor pode ser alterado de acordo com as necessidades de monitoração desejadas através da constante *t_sec*. O valor utilizado em nossa monitoração foi obtido levando-se em consideração a perturbação que a tarefa de monitorar causa normalmente ao sistema a ser observado (*overhead*).

```
static char *heartbeat (void) {

  void *buf;
  int bytes, handle;
  char *ptr = NULL, *str = "100% packet loss";
  sleep (t_sec);
  system ("ping -qc 3 127.0.0.1 > /root/heartbeat.db");
  buf = malloc (256);
  if ((handle = open("/root/heartbeat.db", O_RDONLY, S_IREAD)) == -1) {
    printf("Error Opening File\n");
    exit(1);
  }
  if ((bytes = read(handle, buf, 256)) == -1) {
    printf("Read Failed.\n");
    exit(1);
  }
  else {
    printf("Read: %d bytes read.\n\n", bytes);
    ptr = strstr (buf, str);
    free (buf);
    printf("The substring is: %s\n\n", ptr);
    if (ptr == NULL)
      return ("Primary Server Running");
    else
      return ("Primary Server FAILURE");
  }

}
```

Figura 5.2. Função contendo o *heartbeat*

Com a monitoração do servidor primário definida, através do *heartbeat* desenvolvido, havia necessidade de definirmos também os aspectos relativos à forma como procederíamos para efetivar a comunicação entre o ambiente de alto desempenho e o dispositivo móvel. O mecanismo de comunicação adotado foi os *sockets* de *Berkeley*. Os *sockets* de *Berkeley* são definidos por [GAY00] como um mecanismo de comunicação que provêem serviços de rede TCP/IP para as aplicações. Além disso, devemos destacar que os *sockets* de *Berkeley* são o mecanismo de comunicação padrão do Unix/Linux. Restava saber então, se a API do Palm OS possuía suporte para este mecanismo de comunicação.

O Palm OS SDK (versão 2.0 ou superior) [FOS00][PAL04][POG99] contêm uma biblioteca de rede que provêem serviços de rede TCP/IP. Com esta biblioteca, uma aplicação Palm OS pode se conectar a qualquer outra máquina em uma rede usando o padrão de protocolos TCP/IP. A API desta biblioteca é uma interface de *socket*, modelada de maneira muito similar a API de *sockets* de *Berkeley*. A principal diferença entre a duas APIs, é que a chamada da biblioteca de rede do Palm OS aceita três parâmetros adicionais. São eles: *AppNetRefnum*, *errno* e *AppNetTimeout*. Esses parâmetros são necessários pois a API do Palm OS não realiza a passagem destes parâmetros de modo global, como acontece com a API de *sockets* de *Berkeley*.

Ainda com relação aos *sockets* de *Berkeley*, devemos lembrar que existem dois tipos de *sockets* de acordo com o protocolo que utilizam no nível de transporte para a comunicação. São eles [GAY00]: o TCP (*Transmission Control Protocol*) e o UDP (*User Datagram Protocol*). O TCP, é um protocolo que se caracteriza por oferecer um serviço confiável orientado a conexão, por essa razão utilizamos este serviço. Por outro lado, o UDP se caracteriza por prover um serviço não orientado a conexão, desta forma não implementa nenhum mecanismo para a recuperação de erros.

Uma vez definida as regras para a comunicação, bastava então implementarmos uma função que permitisse a comunicação entre o módulo “haoscar_standby.c” e o módulo “haoscar_mon.c”, que como explicamos anteriormente funcionaria como se fosse o dispositivo móvel neste primeiro momento. O resultado foi à criação de um mecanismo utilizando os *sockets* de *Berkeley* que permitiu a comunicação entre os módulos, a partir da criação das funções *server* e *client* nos módulos “haoscar_standby.c” e “haoscar_mon.c”, respectivamente. Na figura 5.3, apresentamos

um fragmento de código extraído justamente da função *server* que demonstra a utilização dos *sockets* de *Berkeley*.

```
/* start the server loop */
for (;;) {
  /* wait for a connect */
  len_inet = sizeof adr_clnt;
  c = accept(s, (struct sockaddr *)&adr_clnt, &len_inet);
  if ( c == -1 )
    bail("accept(2)");

  /* send system info */
  strcpy (buf, heartbeat());
  z = write (c, buf, sizeof buf);
  if ( z == -1 )
    bail ("write(2)");

  close (c); /* close this client connection */
}
```

Figura 5.3. Comunicação através dos *sockets* de *Berkeley*

O passo seguinte, foi marcado pela realização de alguns testes preliminares com o intuito de comprovarmos a eficiência do método de monitoração desenvolvido. Esses testes foram realizados mediante a utilização dos módulos descritos anteriormente, apenas no ambiente de *cluster* com alta disponibilidade. Como os resultados preliminares se mostraram bastante satisfatórios, então tínhamos agora condições de portar o código fonte do módulo “haoscar_mon.c”, escrito inicialmente na plataforma Linux, para a plataforma Palm OS do dispositivo móvel. Este passo completaria então, os objetivos idealizados no inicio do projeto para a monitoração *wireless*.

Geralmente, uma aplicação Palm OS desenvolvida em linguagem C possui uma estrutura semelhante à apresentada na figura 5.4. A representação esquemática do modelo de aplicação foi concebida justamente com o intuito de facilitar a sua visualização. O modelo foi idealizado com base nos conceitos descritos no capítulo anterior, onde temos a descrição das ferramentas de programação utilizadas. Note, que a representação foi disposta de modo a conter todos os detalhes realmente importantes de uma aplicação Palm OS, mas sem conter a totalidade de suas características. Essa abstração, contudo, foi bastante meticulosa, para que não acarretasse a inclusão de erros no modelo ou excluísse do mesmo aspectos importantes.

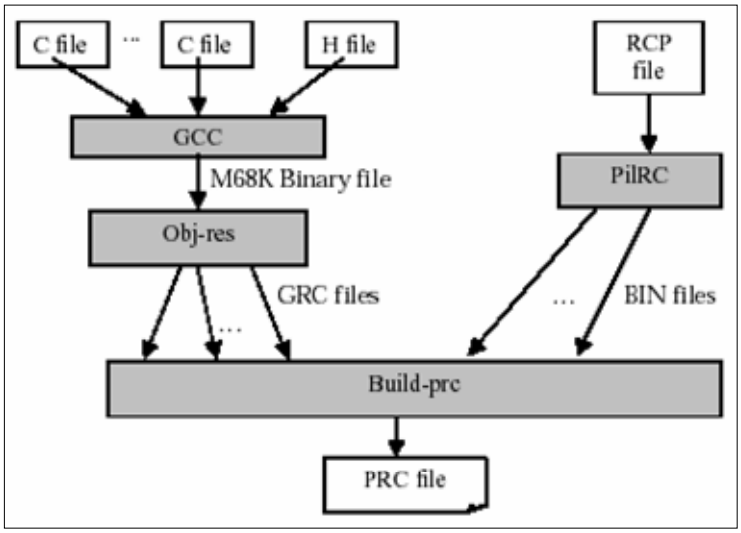

Figura 5.4. Estrutura típica de uma aplicação Palm OS

O módulo "haoscar_mon.c" foi decomposto seguindo a estrutura típica de uma aplicação Palm OS, como apresentamos na figura acima. O resultado final foi à criação de um arquivo com a extensão ".prc" que é executado pela plataforma Palm OS. Foram criados com este propósito então, os seguintes módulos de programas:

- Main.c: Programa principal que permanece em execução enquanto a condição de parada não é satisfeita. Este programa também é responsável pelo controle e execução dos demais módulos descritos a seguir;
- HaOscarRsc.h: Arquivo contendo os cabeçalhos necessários para a criação da interface gráfica;
- HaOscar.rcp: Este arquivo possui todas as interfaces e recursos gráficos utilizados para a construção da interface gráfica no dispositivo móvel;
- haoscar.h: Arquivo contendo os cabeçalhos necessários para a chamada dos mecanismos de comunicação;
- haoscar.c: Programa que contêm as funções necessárias para a utilização dos mecanismos de comunicação (*sockets* de *Berkeley*) na plataforma Palm OS. Na figura 5.5, apresentamos justamente um trecho contendo parte do código da função *client*.

Não poderíamos deixar de mencionar, que uma aplicação Palm OS é sempre orientada a eventos segundo [FOS00][POG99]. Desta forma, os eventos ocorrem e o aplicativo deve responder a eles. Alguns eventos são tratados pelos aplicativos, outros

pelo sistema. Quando um aplicativo é iniciado, ele entra em um *loop* e fica esperando um evento após o outro, e então os gerencia individualmente. O *loop* continua até que o usuário inicie um outro aplicativo, o que produz o término do anterior. Em nossa aplicação não haveria como ser diferente. Por outro lado, utilizamos determinados botões de atalho, que possuem uma função especial para o acionamento via *hardware*, ou seja, os eventos de monitoração e demais funcionalidades são iniciados através de botões de *hardware* específicos do dispositivo móvel.

```
  // connect to the server
  len_inet = sizeof adr_srvr;
  z = connect (s, &adr_srvr, len_inet);
  if ( z == -1 ) {
    (*gErrorFunc)("Error: Connect Error");
    return 0;
  }

  // receive system info
  z = read (s, &buf, sizeof buf-1);
  if ( z == -1 ) {
    (*gErrorFunc)("Error...: Read Error");
    return 0;
  }
  else {
    buf[z] = 0;
    (*gStatusFunc)(buf); // report the system info
  }

  close (s); // close the socket
  return 0;

}
```

Figura 5.5. *Berkeley sockets* na plataforma Palm OS

Para finalizarmos a descrição da ferramenta desenvolvida, devemos mencionar que para a correta monitoração do ambiente de *cluster* HA-OSCAR, o módulo "haoscar_standby.c" deve estar presente no servidor secundário do *cluster*. No momento em que este módulo estiver em execução, basta então copiarmos o arquivo gerado "HaOscar.prc" para o dispositivo móvel e iniciar a monitoração *wireless*. Para copiarmos o arquivo para o dispositivo móvel, podemos utilizar um programa de sincronização como o *KPilot* do Linux por exemplo. Na figura 5.6, podemos observar o módulo de monitoração HA-OSCAR após a sincronização com o dispositivo móvel.

Figura 5.6. Módulo de monitoração HA-OSCAR após sincronização

5.2.1 – Resultados Experimentais

Antes de iniciarmos a descrição dos resultados experimentais, vamos comentar inicialmente algumas funcionalidades a cerca da ferramenta desenvolvida. O objetivo é prover um melhor entendimento da ferramenta, antes de iniciarmos a descrição dos resultados experimentais. Nesse sentido, vamos descrever brevemente todos os passos necessários para a utilização da ferramenta de monitoração. Note, que os passos a seguir consideram que o ambiente encontra-se devidamente configurado, como explicamos nos capítulos anteriores.

Ao iniciarmos a aplicação, com um simples *click* em seu ícone, surge então a tela principal com a ferramenta de monitoração *wireless* para o *cluster* HA-OSCAR. A seguir, temos então duas possibilidades: acessar o menu de ajuda ou iniciarmos de fato a monitoração do ambiente de *cluster* HA-OSCAR. Para acessarmos o menu de ajuda, devemos efetuar um *click* no botão *Menu*. A partir desta opção, temos acesso então a informações sobre a versão da ferramenta desenvolvida, autor e direitos autorais. Com relação aos direitos autorais, devemos salientar que a ferramenta foi desenvolvida em

conformidade com a GPL (*GNU General Public License*), ou seja, possui código aberto. Por outro lado, no momento em que clicamos no botão *Applications* se inicia então a monitoração do *cluster* HA-OSCAR. Para encerrarmos a aplicação, basta clicarmos novamente no mesmo botão, finalizando desta forma a ferramenta de monitoração. Conforme explicamos anteriormente, estes eventos são acionados através de botões de *hardware* específicos do dispositivo móvel, como podemos observar na figura 5.7, onde temos a tela inicial da ferramenta de monitoração HA-OSCAR e os botões de *hardware* utilizados.

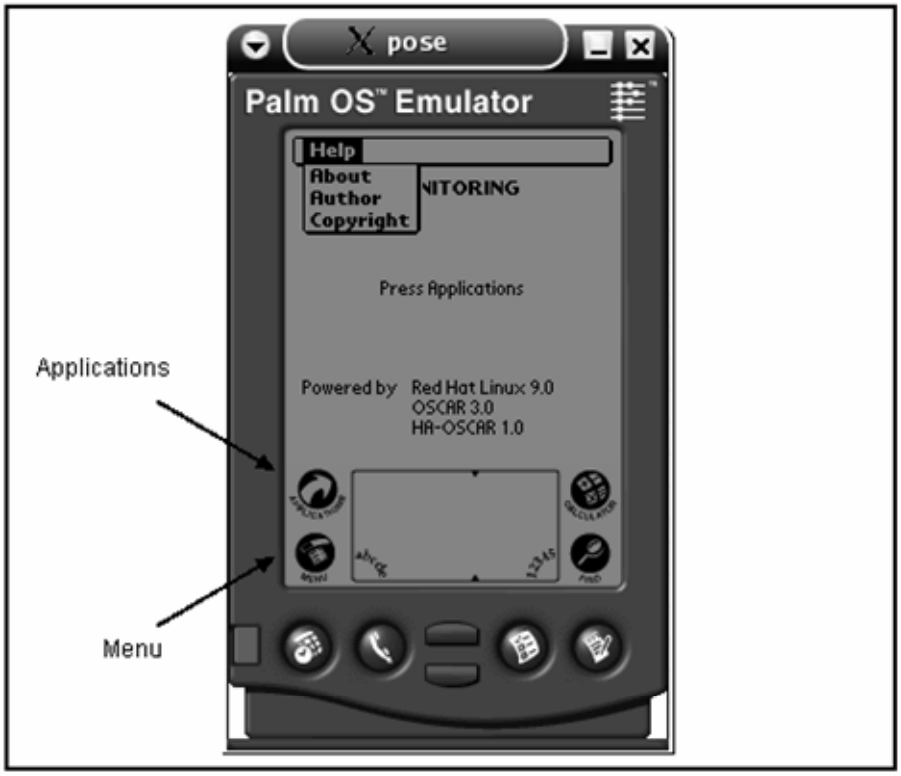

Figura 5.7. Tela inicial da ferramenta de monitoração

Agora que temos conhecimento sobre as funcionalidades da ferramenta de monitoração, podemos explorar então os aspectos relacionados aos resultados experimentais. A validação da ferramenta foi alcançada através de repetidas séries de testes visando simular a ocorrência de defeitos na configuração. Estes defeitos foram introduzidos de maneira intencional com o intuito de simular eventuais situações de defeito na configuração, para a realização de nossos testes no ambiente. Os testes

realizados possuem uma dinâmica bastante simples, contudo, houve uma preocupação muito grande no que diz respeito a sua confiabilidade e eficiência.

Deste modo, simulamos então um defeito através do desligamento de um dos servidores. Para simularmos este defeito, desligamos o servidor primário fazendo com que o servidor secundário assumisse sua função. Este procedimento foi realizado repetidamente sendo que a ferramenta de monitoração em todos os experimentos se mostrou muito confiável e com um tempo de resposta bastante satisfatório. O tempo de resposta máximo é de cinco segundos após a ocorrência de um defeito na configuração, graças ao *heartbeat* que realiza a monitoração em ciclos de cinco segundos. Na figuras 5.8 e 5.9, podemos observar as duas situações possíveis. Uma situação de normalidade e outra situação de defeito na configuração do ambiente, respectivamente.

Figura 5.8. Servidor primário executando

Além deste experimento, simulamos também um outro defeito através do desligamento do servidor secundário. Como sabemos, o *cluster* HA-OSCAR [LEA04] não implementa nenhum mecanismo para a recuperação do servidor secundário, caso este apresente defeito em sua configuração. Desta forma, utilizamos as próprias

mensagens de erros retornadas pelos *sockets* de *Berkeley*, implementadas na função *client*, para validarmos novamente a ferramenta desenvolvida. As mensagens têm como objetivo alertar o responsável pelo ambiente de alto desempenho sobre problemas na comunicação entre o servidor secundário e o dispositivo móvel. Os alertas referem-se a problemas específicos com a comunicação, através de mensagens como: *Socket Error*, *Bad Address*, *Connect Error* e *Read Error*. Os resultados obtidos, oferecem valiosos indícios caso ocorreram problemas na comunicação. A ferramenta desenvolvida mais uma vez se mostrou bastante eficiente e muito confiável no que se refere aos objetivos idealizados inicialmente para a monitoração do *cluster* HA-OSCAR, pois em todos os experimentos realizados obtivemos êxito nos resultados.

Figura 5.9. Defeito detectado no servidor primário

Para legitimarmos ainda mais o trabalho descrito nesta dissertação, vamos finalizar a descrição dos resultados experimentais mencionando o aceite de duas publicações relacionadas ao trabalho neste ano de 2004. As publicações (Anexo 1 e 2) demonstram a preparação e até mesmo evolução do trabalho. Estas publicações vêm de

certo modo a premiar o esforço e dedicação para a realização do trabalho, ressaltando assim a sua qualidade através dos resultados obtidos.

5.2.2 – Comparação dos Resultados

Esta seção descreve brevemente os resultados obtidos através da comparação dos resultados, além de discussões críticas sobre esses resultados e sobre as características de outra ferramenta de monitoração chamada IPMI (*Intelligent Power Management Interface*). A ferramenta IPMI servirá para completarmos a validação da ferramenta de monitoração *wireless* desenvolvida neste trabalho. Devemos mencionar, que a ferramenta IPMI é essencialmente utilizada para a geração de alarmes, com o intuito de monitorar um sistema antes que defeitos de *hardware* causem problemas na configuração do *cluster*. Apesar de ter uma proposta um pouco diferente, da ferramenta desenvolvida, foi à ferramenta que mais se aproximou de nossa realidade, uma vez que devemos levar em consideração as características de um ambiente de *cluster* com alta disponibilidade.

De acordo com [LIB03], o IPMI é definido como uma abordagem para utilização de interfaces comuns de *hardware* de maneira mais "inteligente", a fim de monitorar características físicas do servidor como: temperatura, voltagem, ventiladores (*fans*), fonte de alimentação e chassi (que acomoda as placas de circuitos juntamente com a fiação e os soquetes). Estas funcionalidades provem informações que permitem o gerenciamento do sistema, recuperação, recursos de rastreamento, além de contribuir para o aumento da confiabilidade do sistema computacional de alto desempenho. As novas interfaces do IPMI (versão 1.5) prometem facilidades para o gerenciamento de servidores em sistemas computacionais de alto desempenho montados em gabinetes, e também para sistemas remotos através de conexão serial, modem e LAN. As novas capacidades combinadas com as facilidades para o gerenciamento remoto permitem aos gerentes de sistemas uma série de serviços e avanços para o gerenciamento dos servidores e sistemas. Os servidores em conformidade com IPMI eliminam a necessidade de um *hardware* externo para realizar a sua função, reduzindo assim os custos com a monitoração.

Como podemos observar, a ferramenta de monitoração IPMI foi concebida para monitorar dispositivos de *hardware*. Por outro lado, a nossa ferramenta se destina à monitoração do sistema em um nível mais alto, ou seja, em nível de *software*. Desta forma, os resultados comparativos na verdade se complementam, pois ao utilizarmos as duas abordagens temos uma solução que contempla tanto à parte de monitoração do *software* quanto à monitoração de *hardware*. O ponto de intersecção entre estas duas abordagens são os sistemas computacionais de alto desempenho, sendo justamente o alvo de monitoração destas duas abordagens, porém com propostas e objetivos distintos para a monitoração.

Com relação à configuração, o IPMI garante ser bastante flexível permitindo que muitos parâmetros possam ser configurados para se ajustar às necessidades dos usuários. Estes parâmetros incluem a configuração da LAN, níveis de privilégios e funções de alerta, entre outros. Em nossa ferramenta, a flexibilidade para a configuração do ambiente permite também que a configuração dos parâmetros possa se ajustar às necessidades do usuário. Neste caso, o gerente do sistema pode escolher qual segmento de rede deseja monitorar, além do intervalo de tempo para a monitoração. Obviamente, quanto menor este valor, conseqüentemente maior será o *overhead*. O valor *default* é de cinco segundos, mas pode ser alterado para satisfazer um grau de severidade maior ou menor para a monitoração.

Segundo [DAN02][SOA95] os aspectos de segurança e desempenho são elementos vitais para o perfeito funcionamento de um ambiente de rede. Note, que esta constatação pode ser estendida também para os sistemas computacionais de alto desempenho. Sendo assim, vamos discutir alguns aspectos referentes à segurança. Primeiramente vamos comentar sobre a ferramenta IPMI que utiliza um mecanismo de autenticação. O processo de autenticação na ferramenta IPMI é confirmado através da identificação do usuário. Foram definidos então quatro modos de acordo com o nível de privilégios: *callback*, *user*, *operator* e *administrative*. Deve-se destacar que o processo de autenticação faz uso também de métodos de criptografia. Por outro lado, a nossa ferramenta não implementa nenhum mecanismo de autenticação para garantir a segurança do ambiente. Sendo assim, utilizamos os próprios recursos do dispositivo móvel, que provém uma chave de criptografia de 128 bits, para a autenticação do usuário. Desta forma asseguramos assim a integridade do ambiente. Devemos lembrar,

que o escopo inicial do projeto não prevê nenhum mecanismo com relação à segurança nesta primeira fase do projeto.

De acordo com [DAN02], em ambientes de rede TCP/IP é muito comum à utilização do protocolo SNMP (*Simple Network Management Protocol*) para o gerenciamento do ambiente. Para reforçarmos o que foi dito acima, basta nos determos nas características da ferramenta IPMI que utiliza o SNMP para enviar as mensagens de alerta. A solução que adotamos para nossa ferramenta não utiliza o SNMP, mas possui uma estrutura similar a empregada por este protocolo. Uma vez que temos dois módulos que interagem ente si, conforme descrito na seção 5.2, e que funcionam como se fossem uma estação de gerenciamento SNMP e um agente SNMP, respectivamente. A comunicação ocorre mediante a utilização dos *sockets* de *Berkeley,* segundo as regras definidas para a comunicação. O resultado é o envio de mensagens de alerta através desta estrutura de comunicação concebida especificamente para este propósito. Maiores detalhes sobre o protocolo SNMP podem ser obtidos em [DAN02].

Antes de finalizarmos a descrição desta seção, devemos fazer uma pequena citação sobre a ferramenta de monitoração IPMI, que forneceu importantes elementos para a obtenção de uma visão mais crítica com relação à ferramenta de monitoração desenvolvida. Com relação à ferramenta desenvolvida, devemos salientar, que se trata ainda de uma primeira versão (protótipo) com alguns elementos a serem melhorados, além da adição de novas funcionalidades na próxima versão. Com a comparação dos resultados nesta seção, finalizamos então a fase de validação da ferramenta proposta, demonstrando a sua legitimidade com relação aos seus objetivos delineados. Como veredicto final, podemos dizer que as duas ferramentas apresentadas complementam-se, pois uma monitora o *software* e outra o *hardware*.

5.3 – Considerações Finais

Para a realização deste trabalho, foram utilizados os recursos computacionais do Laboratório de *Web Software* (LABWEB) localizado no Departamento de Informática e Estatística (INE) da Universidade Federal de Santa Catarina (UFSC). Nesse sentido, devemos mencionar, além dos objetivos já delineados anteriormente na seção 5.2, os esforços realizados também no sentido de que o grupo de trabalho responsável pelo

pacote de *software* HA-OSCAR incorpore em sua próxima versão o módulo de monitoração *wireless* desenvolvido. Desta forma, o módulo de monitoração poderá oferecer ao ambiente de *cluster* HA-OSCAR facilidades para o envio de alarmes a dispositivos móveis via rede *wireless*, informando periodicamente sobre a configuração e o funcionamento do ambiente de alto desempenho. A intenção é desenvolvermos novas funcionalidades para a ferramenta, em adição as já existentes, além do seu continuo aperfeiçoamento de modo que possa interagir e integrar-se ao ambiente de *cluster* computacional HA-OSCAR da melhor maneira possível.

6 – CONCLUSÕES E TRABALHOS FUTUROS

O estudo apresentado neste trabalho de dissertação tem como objetivo prover a especificação e validação de uma abordagem de monitoração *wireless* para o ambiente de *cluster* com alta disponibilidade HA-OSCAR. Foram apresentados conceitos e tecnologias utilizadas na especificação da ferramenta de monitoração, além das principais tendências para sua construção. Na prática, o módulo de monitoração responsável pelo envio da notificação a um dispositivo móvel, a partir do momento que é ativado inicia a monitoração do ambiente de c*luster* HA-OSCAR informando periodicamente sobre a configuração do ambiente de alto desempenho. Este módulo tem por objetivo alertar o responsável pelo ambiente do *cluster* computacional sobre possíveis ocorrências de defeitos na configuração.

É interessante comentar algumas dificuldades encontradas, ocasionadas inicialmente devido à complexidade imposta pela configuração do ambiente experimental. Isto ocorre, uma vez que devemos levar em consideração a complexidade da comunicação, a forma com o qual os dados são acessados, as estruturas de controle e as abordagens utilizadas para o desenvolvimento da ferramenta de monitoração. Em um ambiente de computação paralela e distribuída (*cluster* computacional) estes elementos se tornam ainda mais críticos, devendo-se tomar muito cuidado para que esta complexidade não interfira no resultado final. Complexidade que foi aumentada com a utilização dos dispositivos móveis e das redes *wireless*. Isso ocorre, uma vez que foi necessária a definição de um conjunto de elementos que fossem capazes de realizar a comunicação via WLAN entre duas abordagens distintas (*cluster* computacional e dispositivo móvel).

Como sabemos, as aplicações desenvolvidas para dispositivos móveis devem possuir simplicidade e facilidade de utilização devido às restrições de recursos que possuem estes dispositivos. Sabendo disso, desenvolvemos então a ferramenta de monitoração inteiramente em linguagem C. A escolha foi motivada também, pelo fato do ambiente de *cluster* HA-OSCAR ser desenvolvido quase que inteiramente em linguagem C. A intenção é que o módulo de monitoração *wireless* seja futuramente incorporado ao pacote de *software* HA-OSCAR. Desta forma, conseguimos nos adequar

mais facilmente a realidade de implementação do *cluster* HA-OSCAR, além de contribuir também para o aperfeiçoamento do ambiente de *cluster* HA-OSCAR.

Uma preocupação constante durante a implementação da ferramenta de monitoração foi com relação ao *overhead*. Pois como sabemos, a perturbação que a tarefa de monitorar causa, por consumir recursos do sistema, poderia ocasionar uma degradação indesejável no nível de desempenho do sistema. Como estamos tratando de um sistema computacional de alto desempenho, está característica teve uma atenção especial, a fim de garantir um nível de perturbação mínimo para o sistema. Graças a este esforço, o *overhead* gerado pela ferramenta de monitoração manteve níveis mínimos de perturbação em todos os testes realizados. Esses níveis se mantiveram, mesmo quando diminuímos o intervalo de monitoração para valores inferiores há cinco segundos, que utilizamos como *default*. Este valor é inversamente proporcional, mas mesmo em casos extremos o *overhead* permaneceu em níveis plenamente aceitáveis.

A capacidade de mobilidade que os dispositivos móveis oferecem quando associados às redes *wireless*, em adição ao ambiente de *cluster* HA-OSCAR é justamente o diferencial desta monitoração. Devemos lembrar, que a ferramenta foi desenvolvida com o objetivo de fornecer elementos capazes de auxiliar o gerenciamento do ambiente de alto desempenho. A intenção, é que o gerente do sistema tenha condições para verificar sobre o funcionamento e configuração do sistema em diferentes locais, não necessitando assim permanecer no local onde se encontra o *cluster* computacional montado, graças à capacidade de mobilidade que permite esse deslocamento, obviamente limitando-se à área de cobertura que a tecnologia de rede *wireless* utilizada oferece.

A partir dos mecanismos de comunicação podemos chegar a uma última, mas não menos importante constatação. Como foi explicado anteriormente, utilizamos os *sockets* de *Berkeley* para a realização da comunicação entre o ambiente de *cluster* HA-OSCAR e o dispositivo móvel através de uma tecnologia de WLAN. Nesse sentido, definimos então todas as regras e mecanismos necessários para a comunicação entre estas duas abordagens, elementos suficientes para caracterizarmos então um protocolo de comunicação. Trata-se, de um protocolo de comunicação desenvolvido ao nível de aplicação, referindo-se ao modelo OSI. O resultado final pode ser comprovado cada vez

que o módulo de monitoração presente no *cluster* HA-OSCAR utiliza o protocolo de comunicação para enviar uma notificação ao dispositivo móvel através da WLAN.

Para um próximo passo de pesquisa, estamos investigando a possibilidade da melhoria de algumas funcionalidades já existentes, além da adição de novas funcionalidades. Para a melhoria das funcionalidades existentes, estamos aguardando a versão oficial do pacote de *software* HA-OSCAR, uma vez que ainda se encontra em fase beta. Como exemplo, poderíamos citar a intenção de utilização do *heartbeat* do próprio ambiente de *cluster* HA-OSCAR, pois no momento o *heartbeat* do HA-OSCAR monitorara apenas uma interface de rede privada. Entre as novas funcionalidades, incluem-se elementos para a autenticação do usuário na própria ferramenta de monitoração e a criação de mecanismos para monitorarmos também o servidor secundário. Além destas funcionalidades descritas, estamos considerando também a possibilidade do desenvolvimento de uma ferramenta de monitoração, nos mesmos moldes, para que estações de trabalho possam realizar também a monitoração do ambiente de *cluster* computacional HA-OSCAR.

REFERÊNCIAS BIBLIOGRÁFICAS

[3CO00] 3Com. *IEEE 802.11b Wireless LANs*. Disponível em http://www.3com.com, 2000.

[ALM89] Almasi, G.S.; Gottlieb, A. *Highly Parallel Computing*. Benjamin/Cummings Publ. Comp., Redwood City, Cal., 1989.

[BIS01] Bisdikian, C.; Bhagwat, P.; Golmie, N. Wireless Personal Area Networks. *IEEE Network Magazine*, v. 15, n. 5, September/October 2001.

[BRA00] Braley, R.C.; Gifford, I.C.; Heile, R.F. Wireless Personal Area Networks: na overview of the IEEE P802.15 Working Group. *ACM Mobile Computing and Communications Review*, v. 4, n. 1, pp. 26-33, January 2000.

[AMZ96] Amza, C.; Cox, A.L.; Dwarkads, S.; Keleher, P.; Rajamony, L.; Yu, W.; Zwaenepoel, W. Trademarks: shared memory computing on networks of workstations. *IEEE Computer*, 29(2): 18-28, February, 1996.

[ARA01] Araújo, H.; Costa, J.; Correia, L.M. Analysis of a Traffic Model for GSM/GPRS. *IEEE Computing*, pp. 124-128, 2001.

[CDG04] CDG. CDMA Development Group. Disponível em http://www.cdg.org, 2004.

[CHH96] Chhaya, H.; Gupta, S. Performance of Asynchronous Data Transfer Methods of IEEE 802.11 MAC Protocol, *IEEE Pers. Communications*, vol. 3, no. 5, pp. 8-15, Oct. 1996.

[CHH02] Chheda, A. A Spectral Efficiency Comparison of UMTS-FDD and 1xRTT cdma2000 for Voice Services. *IEEE Computing*, pp. 554-558, 2002.

[COL04] Colvero, T.A.; Pernas, A.M.; Dantas, M.A.R. Facilidades para Gerenciamento de uma Configuração de Agregado. *ERAD 2004: Anais da 4ª Escola Regional de Alto Desempenho*, Pelotas, 2004, pp.189-192.

[CUL99] Culler, David E.; Pal, Singh J.; Gupta, A. *Parallel Computer Architecture: a hardware/sofware approach*. Morgan Kaufmann Publishers, 1999.

[DAN02] Dantas, M.A.R. *Tecnologias de Redes de Comunicação e Computadores*.

Editora Axcel Books do Brasil, 2002.

[EKL02] Eklund, C.; Marks, R.B.; Stanwood, K.L.; Wang, S. IEEE Standard 802.16: A Technical Overview of the WirelessMAN Air Interface for Broadband Wireless Access. *IEEE Communications Magazine*, pp.98-107, June 2002.

[FLY72] Flynn, M.J. Some Computer Organizations and their Effectiveness. *IEEE Transaction on Computers* 21, v. 21, n. 9, pp. 948-960, 1972.

[FOS00] Foster, L.R. *Palm OS Programming Bible*. IDG Books Worldwide, Inc., 2000.

[GAS02] Gast, M. *802.11 Wireless Networks: the definitive guide*. O´Reilly, 2002.

[GAY00] Gay, W.W. *Linux Socket Programming by Example*. Macmillan Computer Publishing, Inc., 2000.

[GEI94] Geist, A.; Beguelin, A.; Dongarra, J.; Jiang, W.; Manchek, R.; Sunderam, V. *PVM: Parallel Virtual Machine – A Users Guide and Tutorial for Networked Parallel Computing*, MIT, 1994.

[GRI02] Griffith, A. *GCC: The Complete Reference*. McGraw-Hill/Osborne, 2002.

[GSM04] GSM World. Disponível em http://www.gsmworld.com, 2004.

[HOC88] Hockney, R.W.; Jesshope, C.R. *Parallel Computers 2*. Adam Hilger, Bristol and Philadelfia, 1988.

[HWA93] Hwang, K. *Advanced Computer Architecture: parallelism, scalability, programmability*. McGraw-Hill, 1993.

[HWA98] Hwang, K.; Xu, Z. *Scalable Parallel Computing: technology, architecture, programming*. McGraw-Hill, 1998.

[IEE01] IEEE 802.16.2-2001, "IEEE Recommended Practice for Local and Metropolitan Area Networks — Coexistence of Fixed Broadband Wireless Access Systems," Sept. 10, 2001.

[IEE02] IEEE 802.16-2001. "IEEE Standard for Local and Metropolitan Area Networks — Part 16: Air Interface for Fixed Broadband Wireless Access Systems". Apr. 8, 2002.

[INT03] Intel. *IEEE 802.16 and WiMAX*. Disponível em http://www.intel.com, 2003.

[ISO99] ISO/IEC. "IEEE 802.11 Local and Metropolitan Area Networks: Wireless

LAN Medium Access Control (MAC) and Physical (PHY) Specifications, ISO/IEC 8802-11:1999(E)".

[JAI91] Jain, R. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation and Modeling*. John Wiley & Sons, 1991.

[KAR01] Karaoguz, J. High-Rate Wireless Personal Area Networks, *IEEE Communications Magazine*, pp.96-102, Dec. 2001.

[KRO85] Kronsjo, L. *Computational Complexity of Sequential and Parallel Algorithms*. John Wiley & Sons, 1985.

[LEA04] Leangsuksun, C.; Liu, T.; Scott, S.L.; Libby, R.; Haddad, I. et al. HA-OSCAR Release 1.0: Unleashing HA-Beowulf. In: *International Symposium on High Performance Computing Systems (HPCS)*, Canada, May 2004.

[LIB03] Libby, R. et al. Effective HPC Hardware Management and Failure Prediction Strategy Using IPMI. In: *International Symposium on High Performance Computing Systems (HPCS)*, Canada, May 2003.

[LIG03] Ligneris, B. et al. Open Source Cluster Application Resources (OSCAR): Design, Implementation and Interest for The Computer Scientific Community. In: *International Symposium on High Performance Computing Systems (HPCS)*, Canada, May 2003.

[MOS04] Moser, S.L.; Rista, C.; Dantas, M.A.R Alta Disponibilidade – Um estudo de Caso em um Ambiente de Imagem Única de Produção. *Artigo submetido para o WSCAD 2004*, Foz do Iguaçu, 2004.

[MPI94] MPI-FORUM. *The MPI message passing interface standard*. Knoxville: University of Tennesse, 1994.

[NAU03] Naughton, T.; Scott, S.L.; Fang, Y.; Pfeiffer, P.; Ligneris, B. D.; Leangsuksun, C. *The OSCAR Toolkit: current and future developments*. Disponível em http://www.dell.com/powersolutions, 2003.

[NUT72] Nutt, G.J. Tutorial: Computer System Monitors. *IEEE Computing*, v. 8, n. 11, pp. 51-61, November 1972.

[OHR03] Ohrtman, F.; Roeder, K. *Wi-Fi Handbook: building 802.11b wireless networks*. McGraw-Hill, 2003.

[OSC03] OSCAR. *Open Source Cluster Application Resource*. Disponível em http://oscar.sourceforge.net, November 25, 2003.

[PAL04] PalmSource. Disponível em http://www.palmsource.com, 2004.

[PAR01] Paranchych, D.W. 1xEV-DO Network Design and Performance. *IEEE Computing*, pp. 153-156, 2001.

[PBS03] PBS – *Portable Batch System (OpenPBS)*. Disponível em http://www.openpbs.org, 2003.

[POG99] Pogue, D. *Palm Programming: The Developer's Guide*. O'Rielly and Associates, Inc., 1999.

[RIS04] Rista, C.; Pinto, A.R.; Dantas, M.A.R. OSCAR: Um Gerenciador de Agregado para Ambiente Operacional Linux. *ERAD 2004: Anais da 4ª Escola Regional de Alto Desempenho*, Pelotas, 2004, pp.193-196.

[ROS04] Rose, C.A.F.; Navaux, P.O.A. et al. Fundamentos de Processamento de Alto Desempenho. In: *Anais da 4ª Escola Regional de Alto Desempenho (ERAD),* Pelotas, 2004.

[SAL02] Salkintzis, A.K.; Fors, C.; Pazhyannur, R. WLAN-GPRS Integration For Next-Generation Mobile Data Networks. *IEEE Wireless Communications*, pp. 112-124, October 2002.

[SOA95] Soares, L.F.; Lemos, G.; Colcher, S. *Redes de Computadores – das LANs, MANs e WANs às Redes ATM*. Editora Campus, 1995.

[SWA02] Swaminatha, T.M.; Elden, C.R. *Wireless Security and Privacy: best practices and design techniques*. Publisher Addison Wesley, September 13, 2002.

[VAN94] Van, V.B.; Seidel, S.; Barszcz, E. Profiling the Communication Workload of an iPSC/860. *Proc. Scalable High Performance Computing Conf.*, 1994.

[VAX03] Vaxevanakis, K.; Zahariadis, T.; Vogiatzis, N. A Rewiew on Wireless Home Network Technologies. *ACM Mobile Computing and Communications Review*, v. 7, n. 2, pp. 59-68, April 2003.

[WHE01] Wheat, J.; Hiser, R.; Tucker, J.; Neely, A.; McCullough, A. *Designing a Wireless Network*. Syngress, 2001.

[ZHA03] Zhang, Q.; Guo, C.; Guo, Z.; Zhu, W. Efficient Mobility Management for

Vertical Handoff between WWAN and WLAN. *IEEE Communications Magazine*, pp. 102-108, November 2003.

ANEXOS

A seguir apresentamos as publicações relacionadas ao trabalho de dissertação em eventos realizados neste ano de 2004:

Descrição: Publicação aceita na Escola Regional de Alto Desempenho (ERAD/RS 2004), idem Anexo 1
Título: OSCAR: Um Gerenciador de Agregado para Ambiente Operacional Linux
Evento: IV Escola Regional de Alto Desempenho (ERAD/RS)
Data e local: de 13 a 17 de Janeiro de 2004, Pelotas – RS
Autores: Luís Cassiano Goularte Rista, Alex R. Pinto e Mario Antonio Ribeiro Dantas

Descrição: Publicação aceita no Simpósio de Informática da Região Centro do RS (SIRC/RS 2004), idem Anexo 2
Título: Um Protocolo para Comunicação em um Ambiente de Alto Desempenho com Alta Disponibilidade
Evento: III Simpósio de Informática da Região Centro do RS (SIRC/RS)
Data e local : de 18 a 20 de Agosto de 2004, Santa Maria – RS
Autores: Luís Cassiano Goularte Rista e Mario Antonio Ribeiro Dantas

Anexo 1 – Publicação na ERAD/RS 2004

OSCAR: Um Gerenciador de Agregado para Ambiente Operacional Linux

C. Rista, A.R. Pinto, M.A.R. Dantas

Universidade Federal de Santa Catarina – UFSC
Centro Tecnológico – CTC
Departamento de Informática e Estatística – INE
{rista, arpinto, mario}@inf.ufsc.br

Resumo

O cenário que surge com a maior utilização de agregado de computadores, como uma opção para a computação de alto desempenho, tem requerido uma demanda de ferramentas capazes de proporcionar uma melhor configuração e um gerenciamento mais eficiente desses ambientes. Neste artigo apresentamos as principais características do OSCAR (*Open Source Cluster Application Resource*). Este ambiente é uma ferramenta de *software* aberto e cuja utilização já atinge cerca de vinte e três porcento [LIG 03] das configurações de agregados da comunidade técnico-científica.

Introdução

Os recentes avanços nas tecnologias de microprocessadores e redes locais de alto desempenho têm criado a possibilidade de utilização de agregado de computadores como eficientes ambientes paralelos para execução de um grande número de aplicações. Essa abordagem baseia-se na idéia de utilização de diversos computadores comerciais facilmente encontrados no mercado (COTS – *Components Off The Shelf*) interligados por uma tecnologia de rede local, visando um aumento no desempenho das aplicações, através da exploração da distribuição ou paralelismo. Neste artigo apresentamos um estudo da ferramenta OSCAR (*Open Source Cluster Application Resource*) [OSC 03] para o gerenciamento de agregado de computadores visando um maior desempenho da distribuição/paralelismo.

Nas próximas seções apresentamos uma breve descrição sobre agregado de computadores e o gerenciador OSCAR.

Agregado de Computadores

No contexto tecnológico atual o paradigma de agregado de computadores representa uma forma popular de configuração para execução de aplicações que requeiram um computador paralelo. O termo agregado de computadores pode ser considerado como um conjunto de computadores interligados de tal forma, que os seus

usuários tenham a convicção de estar usando um recurso computacional único. Uma boa parte dessa abstração é obtida através das camadas de *software*, dentro ou fora do sistema operacional, que permitem essa transparência. Segundo [DAN 02], o paradigma de agregado visa um aumento do desempenho das aplicações, através de uma maior taxa de execução dos aplicativos e um aumento no número de dados a serem considerados na execução.

Em um ambiente de agregado de computadores, pode-se imaginar que para atingir o objetivo de melhoria de desempenho, primeiramente devemos considerar alguns esforços, tais como: aumento na velocidade do processador, uso de algoritmos mais otimizados e adoção de um ambiente de computação concorrente (ou paralela) [DAN 02].

Ambiente Experimental

Para a instalação do gerenciador OSCAR foi utilizado o sistema operacional Linux (Red Hat Linux 9). A escolha pela distribuição ocorreu após uma criteriosa seleção entre as demais distribuições suportadas pelo gerenciador OSCAR [OSC 03], [PER 04]. Os critérios que adotamos estão relacionados essencialmente com desempenho, robustez e confiabilidade.

O ambiente de execução foi o agregado de computadores do LabWeb (Laboratório de *Web Software*) localizado no INE (Departamento de Informática e Estatística) da UFSC. Este ambiente é caracterizado por cinco máquinas Intel Pentium 4 de 1,8 GHz, com 256 MB de memória principal, disco rígido com 40 GB e como rede de interconexão utilizamos um *switch Fast Ethernet*.

O Gerenciador OSCAR

Segundo [OSC 03], o OSCAR é um pacote de *software* que permite simplificar a complexa tarefa de utilização e gerenciamento de um agregado. O OSCAR é essencialmente utilizado para a computação de alto desempenho, podendo perfeitamente ser utilizado por qualquer aplicação, que necessite das funcionalidades de um agregado, para obter um aumento em seu desempenho através da exploração do paralelismo. Cabe ressaltar alguns pacotes padrões instalados nativamente, como implementações de MPI (Message Passing Interface) [MPI 94], de PVM (Parallel Virtual Machine) [PVM 02] e de PBS (Portable Batch System) [PBS 03].

Uma prática comum em agregados é o uso da biblioteca PBS, responsável por controlar a submissão e execução das tarefas no nodo servidor, combinada com alguma implementação MPI ou PVM para o ambiente paralelo. As bibliotecas de troca de mensagens como MPI e PVM fornecem a capacidade necessária ao ambiente OSCAR para a criação de programas paralelos em um ambiente de computação distribuída, como apresentado na figura 1.

Inicialmente, foram realizadas a instalação e configuração do sistema operacional no nodo servidor. Logo após foi iniciada a instalação do gerenciador de agregado OSCAR versão 2.3, como é apresentado na figura 2. Como não havia conhecimento da necessidade de instalação de pacotes extras, e por se tratarem de etapas opcionais, os três primeiros passos não foram explorados, mantendo desta forma as opções padrões do OSCAR.

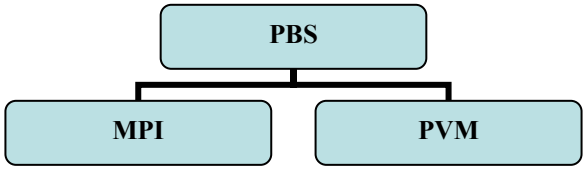

Figura 1 – Hierarquia das bibliotecas no OSCAR

Em seguida, é iniciada a instalação de vários pacotes RPMs adicionais requeridos, além de uma configuração auxiliar no nodo servidor. A fase seguinte consiste na criação da imagem para os nodos escravos. Neste momento, um determinado tempo é necessário para a construção da imagem, sendo que uma barra de progresso indica o status de instalação do processo.

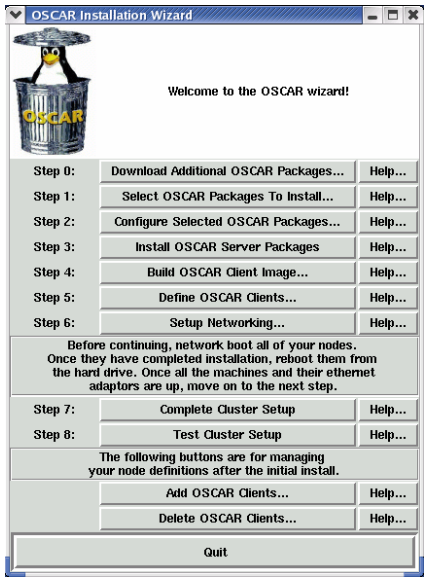

Figura 2 – Instalação do OSCAR

O próximo passo tem a finalidade de definir informações apropriadas para os nodos escravos do gerenciador OSCAR. Nesta fase, são configuradas informações como: número de hosts, atribuição de IP aos nodos escravos, mascara da subrede e gateway, por exemplo.

A seguir é realizada a detecção dos nodos escravos através do endereço MAC, identificado de forma única em uma rede, para posterior atribuição do endereço IP ao nodo escravo. Dois métodos possíveis podem ser utilizados para a inicialização pela rede. O primeiro utiliza uma entrada PXE (*Preboot eXecution Environment*), opção de inicialização da BIOS do nodo escravo, se disponível. Se esta opção não for possível, pode-se criar um disco de inicialização. Cabe lembrar ainda, que no final deste passo, antes de ser iniciado o passo seguinte, é necessário esperar algum tempo para que seja completada a instalação dos nodos escravos. O tempo necessário para a instalação, depende da capacidade do nodo servidor, dos nodos escravos, da rede local, e o número de instalações simultâneas.

As duas etapas que seguem completam a instalação do gerenciador OSCAR. Na fase seguinte, são executados apenas scripts de configurações complementares. E

finalmente o ultimo passo completa de fato a instalação do OSCAR, através de testes e demais verificações no ambiente instalado, certificando-se desta maneira do seu perfeito funcionamento.

Conclusões

O estudo do gerenciador OSCAR apresentado neste artigo tem como objetivo prover facilidades para uma melhor configuração e um gerenciamento mais eficiente em ambientes de agregado de computadores. Foram apresentados conceitos básicos utilizados no OSCAR, além das principais funções para troca de mensagens e controle de execução de tarefas, nativas do OSCAR. É interessante comentar algumas dificuldades encontradas durante as fases de instalação e configuração, ocasionadas inicialmente devido a uma determinada complexidade imposta pela configuração do ambiente apresentado, pois não havia conhecimento prévio das reais exigências do gerenciador de agregado OSCAR. A partir do momento em que estes pré-requisitos foram definidos e superados, a instalação do pacote de *software* OSCAR ocorreu de forma satisfatória e transparente.

Referências

[DAN 02] DANTAS, M.A.R. **Tecnologias de Redes de Comunicação e Computadores.** Rio de Janeiro: Editora Axcel Books do Brasil, ISBN 85-7323-169-6, 2002.

[LIG 03] LIGNERIS, B. et al. **Open Source Cluster Application Resouces (OSCAR): Design, Implementation and Interest for The Computer Scientific Community.** In. International Symposium on High Performance Computing Systems - HPCS, Canada, May 17, 2003.

[MPI 94] MPI-FORUM, MPI **: A Message Passing Interface Standard,** Int. J. Supercomputing Applicatons, MPI FORUM 8(1994), 3-4.

[OSC 03] OSCAR – **Open Source Cluster Application Resources.** Disponível em http://oscar.sourceforge.net, August 19, 2003.

[PBS 03] PBS – **Portable Batch System (OpenPBS).** Disponível em http://www.openpbs.org, 2003.

[PER 04] PERNAS, A. M.; COLVERO, Taís Appel; DANTAS, M.A.R. **Facilidades para Gerenciamento de uma Configuração de Agregado,** Artigo submetido para o ERAD 2004, Pelotas (RS), 2004.

[PVM 02] PVM – **Parallel Virtual Machine (PVM).** Disponível em http://www.csm.ornl.gov/pvm, November 24, 2002.

Anexo 2 – Publicação no SIRC/RS 2004

Um Protocolo para Comunicação em um Ambiente de Alto Desempenho com Alta Disponibilidade

C. Rista e M.A.R Dantas

Universidade Federal de Santa Catarina (**UFSC**)
Centro Tecnológico (**CTC**)
Departamento de Informática e Estatística (INE)

{rista,mario}@inf.ufsc.br

Abstract. *The scenery that appears with the largest use of the environments of high performance and more recently with the coming of the wireless local networks it has been requesting a demand every time larger for protocols and tools. In that sense, we presented in this article an approach for the implementation of a communication protocol in an environment of high performance with characteristics of high availability, providing access through devices wireless.*

Resumo. *O cenário que surge com a maior utilização dos ambientes de alto desempenho e mais recentemente com o advento das redes locais sem fio (wireless) tem requerido uma demanda cada vez maior por protocolos e ferramentas. Nesse sentido, apresentamos neste artigo uma abordagem para a implementação de um protocolo de comunicação em um ambiente de alto desempenho com características de alta disponibilidade, provendo acesso através de dispositivos wireless.*

1. Introdução

Há **muito tempo**, fala-se **que a utilidade dos computadores paralelos depende do desenvolvimento de algoritmos paralelos**, que **operem eficientemente em tais computadores e do desenvolvimento de linguagens de programação paralela**, **nas** quais **esses algoritmos possam ser expressos** [Kronsjo 1985]**.** É **interessante observar** que **os computadores evoluíram**, **mas essas duas** questões básicas (**algoritmos e linguagens**) **ainda requerem soluções mais eficientes. Neste artigo apresentamos nossa investigação no sentido de definir uma abordagem para a implementação de um protocolo de comunicação em um ambiente de alto desempenho com alta disponibilidade**, **provendo facilidades para acesso a redes locais sem fio** (*wireless*)**.** O **artigo** é **organizado da seguinte forma**: **na próxima seção apresentamos uma breve descrição sobre ambientes** *wireless***.** Na **seção** 3, **descrevemos o paradigma de** ***cluster*** **computacional. Em seguida**, **na seção** 4, é **apresentado nosso ambiente experimental**, **bem como as definições referentes ao ambiente proposto. A seção** 5 **ficou reservada a descrição da abordagem para a implementação do protocolo de comunicação. Finalmente**, **na seção** 6 **apresentamos nossas considerações e conclusões finais.**

2. Ambientes Wireless

Segundo [Dantas 2002], os ambientes de redes locais sem fio (*wireless*) são configurações interessantes para agregar valor às redes locais das corporações. O diferencial destes ambientes pode ser ilustrado pelo custo reduzido da sua infra-estrutura e o suporte a aplicações móveis. As redes locais *wireless* oferecem ganhos para os processos móveis envolvidos na utilização desta tecnologia, tais como a eficiência, a precisão e o baixo custo da solução quando comparado com uma rede local guiada.

A tecnologia de rede local *wireless* (WLAN) vem se firmando cada vez mais, como um elemento fundamental para agregar valor às redes de computadores nas organizações. Por esse motivo, tem se verificado um aumento bastante significativo em sua utilização. Desta maneira, uma solução de rede móvel, provendo facilidades de resposta a determinadas solicitações de informações, pode representar um diferencial de serviços para determinado ambiente de rede.

Nesta seção descrevemos algumas tecnologias, mais usualmente utilizadas nas conexões *wireless*. A primeira, a conhecida tecnologia ***b****luetooth*, para a conexão *wireless* de equipamentos a curtas distâncias. Por outro lado, a padronização IEEE 802.11 para redes locais *wireless* de até algumas centenas de metros.

2.1. Bluetooth

A tecnologia *wireless* ***b****luetooth* é um padrão de fato e uma especificação para enlaces entre dispositivos móveis, tais como assistentes digitais pessoais, telefones celulares e outros dispositivos portáteis de baixo custo, usando ondas de rádio de curto alcance [Dantas 2002].

A proposta da tecnologia ***b****luetooth* é habilitar os usuários para a conexão de uma grande variedade de dispositivos de computação e telecomunicações de maneira simples e fácil, sem a necessidade de carregar, comprar ou conectar cabos. Como a tecnologia é baseada em enlace via rádio, é fácil a transmissão rápida e segura de voz e dados na rede. A operação do ***b****luetooth* é efetuada em uma banda de freqüência entre 2,402 e 2,480 GHz que está globalmente disponível e tem compatibilidade mundial sendo portanto um padrão global para conectividade *wireless*.

Os dispositivos equipados com a tecnologia ***b****luetooth* carregam um pequeno chip capaz de se conectar automaticamente a outros dispositivos semelhantes por intermédio das ondas de rádio. Os fabricantes, de forma geral, garantem que as interferências são pequenas nos dispositivos ***b****luetooth*. A argumentação é que a tecnologia é muito resistente a intempéries e dificuldades eletromagnéticas do mundo moderno.

As configurações das redes com a tecnologia ***b****luetooth* oferecem soluções interessantes, permitindo que até sete dispositivos escravos se conectem a um dispositivo mestre. As microredes (*piconets*) são interoperáveis e podem formar uma rede maior e flexível, na qual vários dispositivos podem entrar e sair, sem maiores prejuízos para o conjunto. A rede ***b****luetooth* tem uma largura de banda de aproximadamente 700 Kbps, podendo atingir os 2 Mbps mediante algumas adaptações.

O ***b****luetooth* é uma resposta para a necessidade de conexão *wireless* de equipamentos em distâncias curtas (até 10m) nas seguintes áreas:

- Pontos de acesso de dados e voz.

- **Substituição de cabos.**
- **Redes *ad-hoc* (temporárias).**
- **A transmissão ocorre via rádio, na freqüência de 2,4 GHz.**
- **A especificação engloba tanto o *hardware* quanto o *software*.**

Uma das facilidades mais importantes do ambiente *bluetooth* é o acesso às redes locais. Uma vez que um determinado dispositivo móvel esteja conectado a uma rede local, este poderá ter acesso a todas as facilidades disponíveis no ambiente de rede local. A perspectiva atual da tecnologia *bluetooth* é que esta seja uma porta de comunicação sem fio entre diversos dispositivos, como assistentes digitais pessoais, telefones celulares e outros dispositivos portáteis de baixo custo que possam se beneficiar de alguma forma com o uso desta tecnologia.

2.2. IEEE 802.11

O **padrão IEEE 802.11 também conhecido como WLAN, ambos termos são válidos, é essencialmente utilizado em redes de dados. Por outro lado, este padrão aberto possibilita a utilização de uma grande variedade de aplicações, pois utiliza ondas de rádio de baixa freqüência, provendo um raio de cobertura com pouco mais de 100 m.**

Segundo [**Dantas 2002**], **a topologia 802.11 tem por objetivo a interação transparente dos componentes móveis da rede local com relação a níveis superiores, um exemplo é o controle de enlace lógico. Desta forma, as funções do padrão são caracterizadas pela implementação na placa de radiofreqüência de rede do equipamento móvel, na interface de *software* que orienta a comunicação e no ponto de acesso a rede.**

O **padrão WLAN mais conhecido é o IEEE 802.11b, o qual opera na faixa de 2,4 GHz, tendo uma taxa de transferência máxima de dados da ordem de 11 Mbps. Atualmente encontra-se em vários equipamentos e dispositivos móveis como computadores pessoais e assistentes digitais pessoais, por exemplo. Outro padrão WLAN para interconexão de dispositivos *wireless* é o padrão IEEE 802.11a, operando em uma faixa de 5 GHz, e tendo uma taxa de transferência de dados de 54 Mbps. O padrão WLAN mais recente é o IEEE 802.11g, com faixa de operação de 2,4 GHz e uma capacidade para a transferência de dados de 54 Mbps. Outra característica interessante, ainda referente ao IEEE 802.11g, diz respeito à compatibilidade que este possui como o padrão IEEE 802.11b.**

Com relação à segurança do padrão WLAN, inicialmente utilizava-se um recurso de criptografia simples conhecido como WEP (*Wired Equivalent Privacy*). Este mecanismo de segurança possuía um mecanismo de criptografia simples, o que tornava o método bastante vulnerável a possíveis falhas na segurança. Sabendo dessa vulnerabilidade e limitação que possuía o método WEP, foi desenvolvido um novo padrão denominado *Wi-Fi Protected Access*, ou simplesmente WPA, que promete um mecanismo de criptografia mais seguro e confiável ao ambiente *wireless* do que seu antecessor WEP.

3. Cluster Computacional

No contexto tecnológico atual o paradigma de *cluster* computacional representa uma forma popular de configuração para execução de aplicações que requeiram um computador paralelo. O termo *cluster* computacional pode ser considerado como um conjunto de computadores interligados de tal forma, que os seus usuários tenham a

convicção de estar usando um recurso computacional único. Uma boa parte dessa abstração é obtida através de camadas de *software*, dentro ou fora do sistema operacional, que permitem essa transparência. Segundo [Dantas 2002], o paradigma de *cluster* visa um aumento do desempenho das aplicações, através de uma maior taxa de execução dos aplicativos e um aumento no número de dados a serem considerados na execução.

Em um ambiente de *cluster* computacional, pode-se imaginar que para atingir o objetivo de melhoria de desempenho, primeiramente devemos considerar alguns esforços, tais como: aumento na velocidade do processador, uso de algoritmos mais otimizados e adoção de um ambiente de computação concorrente (ou paralela) [Dantas 2002].

3.1. OSCAR (Open Source Cluster Application Resource)

Segundo [OSCAR 2003], o OSCAR é um pacote de *software* que permite simplificar a complexa tarefa de utilização e gerenciamento de um *cluster*. O OSCAR é essencialmente utilizado para a computação de alto desempenho, podendo perfeitamente ser utilizado por qualquer aplicação, que necessite das funcionalidades de um *cluster*, para obter um aumento em seu desempenho através da exploração do paralelismo. Cabe ressaltar alguns pacotes padrões instalados nativamente, como implementações de MPI (*Message Passing Interface*) [MPI 1994], de PVM (*Parallel Virtual Machine*) [PVM 2004] e de PBS (*Portable Batch System*) [PBS 2003].

Uma prática comum em *clusters* é o uso da biblioteca PBS, responsável por controlar a submissão e execução das tarefas no nodo servidor, combinada com alguma implementação MPI ou PVM para o ambiente paralelo. As bibliotecas de troca de mensagens como MPI e PVM fornecem a capacidade necessária ao ambiente OSCAR para a criação de programas paralelos em um ambiente de computação distribuída, como apresentado na figura 1.

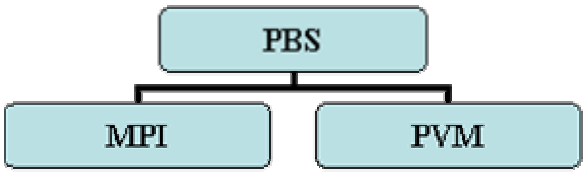

Figura 1. Hierarquia das Bibliotecas no OSCAR

3.2. HA-OSCAR (High Availability - OSCAR)

A alta disponibilidade, quando utilizada no *cluster* HA-OSCAR, possibilita prevenir possíveis defeitos que ocasionalmente possam ocorrer. No nosso caso, alta disponibilidade torna-se fundamental para aquelas aplicações consideradas críticas que se beneficiam da computação de alto desempenho. Para executar eficientemente uma aplicação de âmbito crítico em um *cluster*, técnicas de computação com alta disponibilidade se fazem necessárias para prevenir defeitos, caso os mesmos venham a ocorrer.

Para ter um sistema de computação de alto desempenho e com alta disponibilidade como o *cluster* HA-OSCAR é necessário à utilização de algumas estratégias. Estas estratégias envolvem duplicação de *hardware* e redundância da rede, técnicas bastante comuns para permitir a confiabilidade e disponibilidade necessária

para estes sistemas de computação. A figura 2 apresenta justamente essa arquitetura de sistema para o *cluster* HA-OSCAR assegurando a alta disponibilidade necessária para um ambiente de computação de alto desempenho.

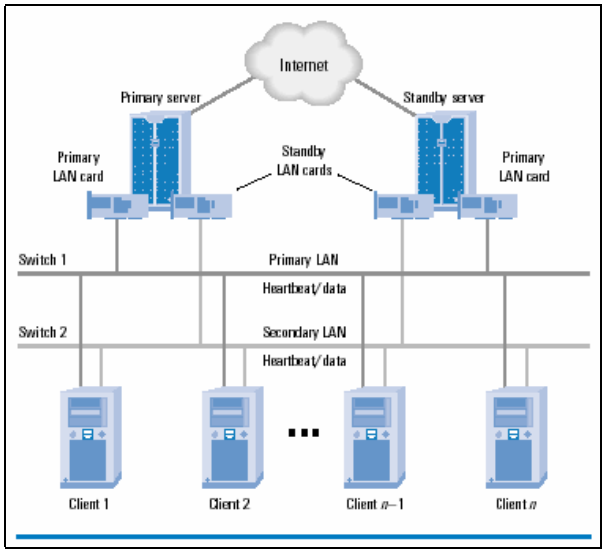

Figura 2. HA-OSCAR

A computação com alta disponibilidade, ilustrada na figura 2, é diferenciada em relação à computação tolerante a falha. A computação com alta disponibilidade é pró-ativa detectando e prevenindo potenciais defeitos, em contraste com computação tolerante à falha que é normalmente mais dispendiosa e reativa. Em computação tolerante a falha, componentes replicados executam as mesmas instruções ao mesmo tempo, ainda sim se um falhar a aplicação continua a execução, sem nenhuma outra diferença a não ser no nível de desempenho reduzido com base na porcentagem dos recursos perdidos. Um custo significante da tolerância à falha é a execução redundante de tarefas e a verificação de pontos de checagem freqüentemente, mesmo quando não há ocorrência de falha.

3.3. RAC

A tecnologia denominada Oracle 9i *Real Application Cluster* (RAC), em adição ao sistema operacional Linux, propõe uma interessante solução baseada em *hardware* de baixo custo e com uma tecnologia de *software* capaz de distribuir a base de dados através de um *cluster* computacional provendo alta disponibilidade [Matsunaga 2003]. A figura 3 ilustra uma configuração típica do RAC baseado em uma arquitetura Intel x86 de 32-bit (IA32) executando sobre o sistema operacional Linux. Essa é uma configuração escalavel e que permite essa facilidade tanto em termos de processador, memória, como também nos dispositivos de armazenamento de alto desempenho (*storage*).

No diagrama de rede (figura 3) podemos observar claramente a presença de uma configuração com uma conexão de rede pública e uma privada. A conexão de rede pública é utilizada pelos clientes para o acesso ao ambiente. Por outro lado, a conexão de rede privada é dedicada entre os nodos (*master*) e o dispositivo de armazenamento de alto desempenho. Essa conexão dedicada é beneficial e deve coexistir pelas seguintes

razões: em um ambiente 9i RAC é importante eliminar possíveis colisões e latências; a existência de uma rede separada assegura maior segurança.

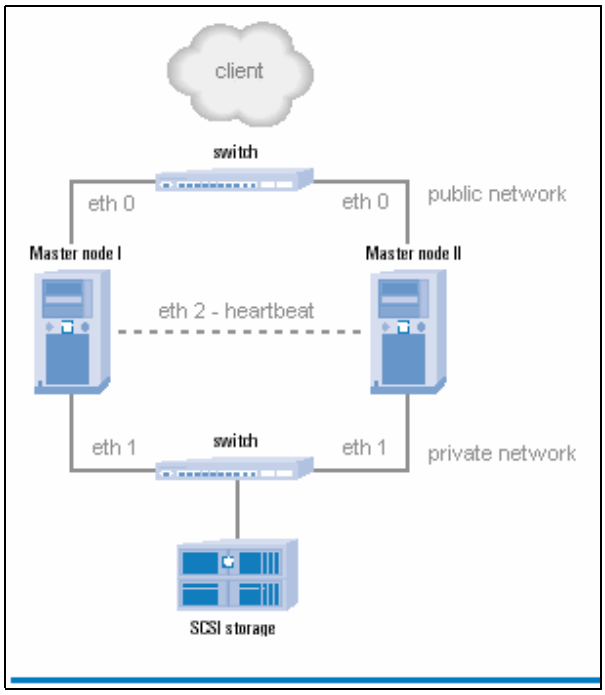

Figura 3. Oracle 9i RAC

O *cluster* possui ainda uma interconexão entre os nodos (*master*) usado para monitorar o ambiente (*heartbeat*) garantindo assim a alta disponibilidade quando necessária. Sendo que, os seus nodos (*master*) compartilham o acesso ao subsistema de armazenamento e os recursos que controlam dados, mas não compartilham fisicamente a memória principal em seus respectivos nodos.

4. Ambiente Experimental

Para a instalação do gerenciador OSCAR foi utilizado o sistema operacional Red Hat Linux 9. A escolha pela distribuição ocorreu após uma criteriosa seleção entre as demais distribuições suportadas pelo gerenciador OSCAR [Colvero 2004], [OSCAR 2003], [Rista 2004]. Os critérios adotados estão relacionados essencialmente com o desempenho, a robustez e com a confiabilidade apresentada pelo sistema operacional. Com relação à instalação do Oracle9i RAC, utilizou-se o sistema operacional Red Hat Advanced Server Linux 2.1, pois havia também a necessidade de uma distribuição certificada e totalmente suportada pelo ambiente.

Agora que se tem uma definição mais formal das reais exigências de *software* que estes ambientes necessitam, é chegado o momento em que devemos avaliar como deve ser feita esta integração, agora em um nível de arquitetura destas duas abordagens. Na figura 4 é apresentada uma solução que acreditamos atender as características de um ambiente de *cluster* computacional de alto desempenho com alta disponibilidade. Observa-se no diagrama de rede abaixo a arquitetura dos dois ambientes, Oracle 9i RAC e HA-OSCAR, de maneira a permitir essa integração.

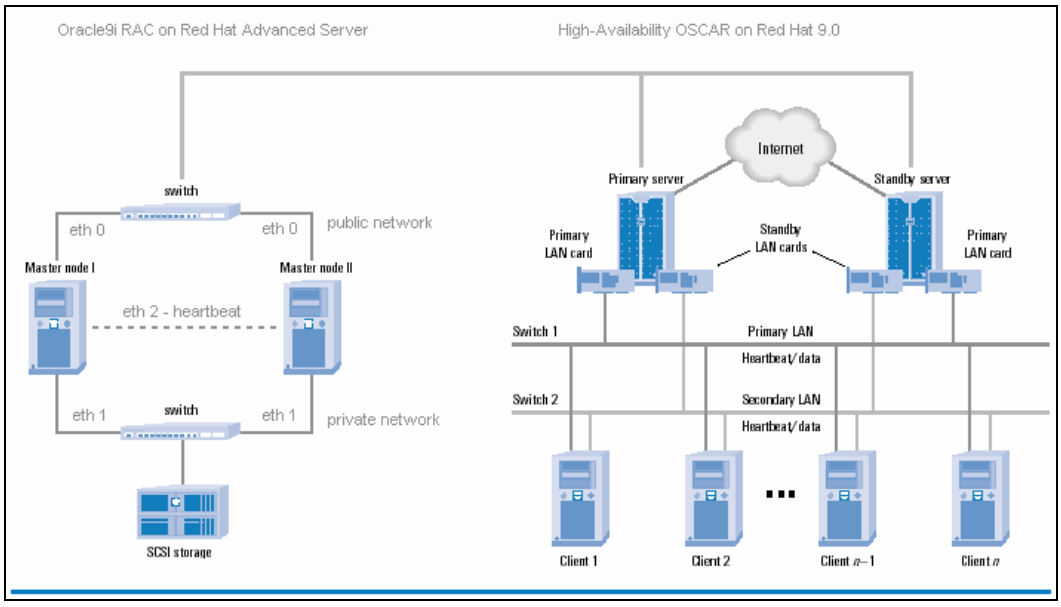

Figura 4. Arquitetura do Oracle 9i RAC e HA-OSCAR

5. Especificação do Protocolo de Comunicação

Esta seção descreve as abordagens necessárias para a implementação do protocolo de comunicação em um ambiente de alto desempenho com alta disponibilidade provendo facilidades para o acesso de dispositivos *wireless*. Na figura 5, podemos observar um protótipo do modelo conceitual o qual acreditamos atender as exigências requeridas pelo ambiente proposto. Note que o ambiente de alto desempenho, caracterizado pelo HA-OSCAR e Oracle 9i RAC encontra-se na mesma nuvem, assegurando desta forma a característica de alta disponibilidade de ambos, conforme demonstrado em detalhes nas seções anteriores (seções III e IV).

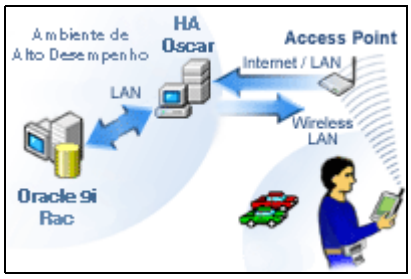

Figura 5. Modelo Conceitual do Ambiente

Após este breve esclarecimento sobre o ambiente, agora temos condições de explorar os aspectos referentes à comunicação, ou seja, as regras definidas pelo protocolo para a comunicação. Ao observarmos novamente a figura 5, percebemos que a comunicação pode ocorrer nos dois sentidos possíveis e ao mesmo tempo, caracterizando assim o modo de transmissão de dados *Full-Duplex*. Dessa forma, um dispositivo *wireless* pode estar realizando uma operação de requisição e estar recebendo os resultados de uma requisição anterior já processada pelo ambiente de alto desempenho ao mesmo tempo.

Para ilustrar melhor o funcionamento do protocolo no ambiente, vamos tentar demonstrar através de um exemplo comentado o seu funcionamento. No momento em

que um dispositivo *wireless* executa uma requisição, o protocolo de comunicação se comunica com o ambiente de alto desempenho HA-OSCAR, que por sua vez processa a informação. Uma vez processada essa informação, é imediatamente repassada ao ambiente Oracle 9i RAC que contém a base de dados do sistema. De posse dessa requisição o Oracle9i RAC remete novamente ao HA-OSCAR que por sua vez retorna os resultados ao dispositivo móvel. Na prática o maior volume de processamento irá ocorrer no HA-OSCAR, sendo o Oracle 9i RAC a ferramenta necessária ao ambiente para prover acesso a um grande volume de dados, com características de alto desempenho.

Com relação ao nível de transporte, estamos investigando duas abordagens. A primeira seria a utilização do protocolo TCP (*Transmission Control Protocol*). O TCP é um protocolo que se caracteriza por oferecer um serviço confiável orientado a conexão. Por outro lado, essa característica gera um *overhead* muitas vezes desnecessário na comunicação, ocasionando desta forma uma degradação no desempenho para a transmissão de dados. Isso ocorre devido as suas características que implementam mecanismos para a recuperação de erros. O protocolo UDP (*User Datagram Protocol*) por sua vez, é um protocolo que se caracteriza por prover um serviço não orientado a conexão. Devido a sua simplicidade, não implementa nenhum mecanismo de recuperação de erros, o UDP permite uma comunicação rápida entre os computadores envolvidos na transmissão. O protocolo UDP é utilizado essencialmente por protocolos de aplicação que se encarregam da confiabilidade fim-a-fim. O NFS (*Network File System*) é um bom exemplo, pois a garantia de consistência das informações é de sua responsabilidade, sendo responsabilidade do UDP apenas realizar a comunicação, sem preocupação com mecanismos para a recuperação de erros.

Para a implementação do protocolo de comunicação apresentado neste artigo pretendemos utilizar a linguagem C padrão (ANSI C). O sistema operacional utilizado pelo HA-OSCAR será o Red Hat 9 e para o Oracle 9i RAC utilizaremos o Red Hat Advanced Server 2.1. As razões pela escolha destas distribuições foram apresentadas na seção anterior. Com relação ao dispositivo móvel, a intenção é utilizarmos o Palm Tungsten C. Esse dispositivo se mostrou bastante satisfatório em alguns de nossos testes preliminares realizados.

6. Conclusões

O estudo apresentado neste artigo tem como objetivo prover a especificação e validação de uma abordagem para implementação de um protocolo de comunicação em um ambiente de alto desempenho com alta disponibilidade, caracterizado por acesso *wireless*. Foram apresentados conceitos e tecnologias utilizadas na especificação do ambiente, além das principais tendências para sua construção. É interessante comentar algumas dificuldades encontradas, ocasionadas inicialmente devido à complexidade imposta pela configuração do ambiente. Isto ocorre, uma vez que devemos levar em consideração a complexidade da comunicação, a forma com o qual os dados são acessados, as estruturas de controle e os dados utilizados para o desenvolvimento do protocolo de comunicação.

Como trabalho futuro, estamos implementando um módulo de monitoração que envie uma notificação a um dispositivo móvel, assim que o nodo primário seja substituído pelo secundário. Este modulo tem por objetivo alertar o responsável pelo ambiente do *cluster* sobre possíveis ocorrências de defeitos na configuração.

7. Referencias

Colvero, T.**A.**; **Pernas**, **A.M.**; **Dantas**, **M.A.R.** "**Facilidades para Gerenciamento de uma Configuração de Agregado**". *ERAD 2004: 4ª Escola Regional de Alto Desempenho,* **Pelotas**, **pp. 1**89-**192**, **Janeiro de 2**004.

Dantas, **M.A.R.** *Tecnologias de Redes de Comunicação e Computadores.* **Rio de Janeiro**: **Axcel Books do Brasil**, **2002.**

Kronsjo, L. *Computational Complexity of Sequential and Parallel Algorithms.* **London**: **John Wiley** & **Sons**, **1**985.

Matsunaga, **M.E.** *Database Scalability with RAC on Linux.* **Disponível em http**://www.**oracle.com**, **2**003.

MPI FO**RUM.** *MPI: A Message Passing Interface Standard, Int. J. Supercomputing Applicatons,* **MPI** FO**RUM** 8(**1**994), 3-4.

OS**CAR.** *Open Source Cluster Application Resource.* **Disponível em http://oscar.sourceforge.net**, **Novembro de 2**003.

PBS. *Portable Batch System (OpenPBS).* **Disponível em http**://www.**openpbs.org**, **2**003.

PVM. *Parallel Virtual Machine (PVM).* **Disponível em http**://www.**csm.ornl.**gov/**pvm**, **Março de 2**004.

Rista, **C.**; **Pinto**, **A.R.**; **Dantas**, **M.A.R.** "OS**CAR**: **Um Gerenciador de Agregado para Ambiente Operacional Linux**". *ERAD 2004: 4ª Escola Regional de Alto Desempenho,* **Pelotas**, **pp. 1**93-**1**96 , **Janeiro de 2**004.