UNIVERSIDADE CATÓLICA DE PELOTAS
ESCOLA DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Gerenciamento de Dispositivos de Borda Reconfiguráveis na Computação Pervasiva

por
Eduardo da Silva Möller

Dissertação apresentada como
requisito parcial para a obtenção do grau de
Mestre em Ciência da Computação

Orientador: Prof. Dr. Maurício Lima Pilla
Co-orientador: Prof. Dr. Adenauer Corrêa Yamin

DM-2008/1-004

Pelotas, abril de 2008

AGRADECIMENTOS

Agradeço a Deus por me dar força dia após dia ao longo desta caminhada.

A minha família, em especial ao meu pai pela força e colaboração nesta jornada.

A minha esposa Fabiana pela compreensão e carinho nos momentos de stress.

Aos meus amigos e colegas da UFPel pelo apoio na realização deste trabalho, em especial ao meu chefe Lúcio e ao Diretor do Centro de Informática da UFPel e colega de mestrado João Ladislau.

A todos os meus colegas de mestrado pelo apoio nos momentos difíceis em especial: Rosaura, Rogério e Vanessa.

A Secretária Kátia pela colaboração e carinho.

A todo o corpo docente, em especial ao Prof. Dr. Adenauer C. Yamin pelo incentivo e apoio .

Ao Prof. Dr. Maurício L. Pilla pela competência e empenho na orientação desta dissertação e pela confiança depositada em mim.

E por fim, a todas as pessoas de forma direta ou indireta, que contribuíram para a realização deste trabalho.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ASIC	*Application-Specific Integrated Circuit*
AVA	Ambiente Virtual de Aplicação
AVU	Ambiente Virtual de Usuário
BDA	Base de Dados Pervasiva das Aplicações
CIB	*Cell Information Base*
CLB	*Configurable Logic Blocks*
DCM	*Digital Clock Manager*
EPROM	*Erasable Programmable Read-Only Memory*
E/S	Entrada e Saída
EXEHDA	*Execution Environment for Highly Distributed Applications*
FIPSOC	*Field Programmable System On a Chip*
FPGA	*Field Programmable Gate Array*
IBM	*International Business Machines*
ICC	*Integrated Circuit Cards*
IOB	*Input/Output Blocks*
I/O	*Input/Output*
J2SE	*Java Standard Edition*
J2ME	*Java Micro Edition*
LUT	*Look-up Table*
MAC	*Media Acess Control*
MCS	*Master Control Station*
OX	*Objeto* EXEHDA
P2P	*Peer-to-Peer*
PCT	Pacote
PDA	*Personal Digital Assistent*

PLD	*Programmable Logic Device*
RCS	Repositório Celular do Sistema
RL	Repositório Local
RMI	*Remote Method Invocation*
RPUs	*Reconfigurable Processing Units*
RTR	*Run-Time Reconfiguration*
SACH	Sistema de Atualização de Configurações de *Hardware*
SDR	Sistema de Distribuição Reconfigurável
SoC	*System-on-a-chip*
TCP/IP	Transmission Control Protocol/Internet Protocol
UI	*User Interface*
UML	*Unified Modeling Language*
XML	*eXtensible Markup Language*

RESUMO

Com o crescente desenvolvimento de novas tecnologias para redes de computadores com e sem fio, a miniaturização de dispositivos e o aumento da capacidade de processamento e armazenamento de informações observadas nos últimos anos, a computação e os diversos tipos de dispositivos computacionais estão cada vez mais presentes no dia-a-dia. Uma nova área que surge desse cenário é a Computação Pervasiva. Ao contrário dos sistemas *desktop* tradicionais, nesse novo paradigma os sistemas computacionais estão em contato com o usuário em qualquer lugar, a qualquer momento. A Computação Pervasiva pode ser entendida como um ambiente computacional, no qual o usuário fica liberado do gerenciamento da execução das suas aplicações, podendo dedicar-se às funcionalidades da computação pretendida. A Computação Reconfigurável, por sua vez, fornece dispositivos lógicos com a possibilidade de agregar o desempenho de um hardware fixo com a flexibilidade do hardware reprogramado através de software. No entanto, não há ferramentas que possam executar reconfigurações nestes dispositivos em ambientes de alta heterogeneidade, elevada dispersão geográfica e grande número de dispositivos. Neste sentido, este trabalho implementa um Serviço de Atualização de Configurações de *Hardware* - SACH, que possibilita a propagação de pacotes de *bitstream* de forma distribuída e concorrente para vários dispositivos de borda do tipo FPGA (*Field Programmable Gate Array*) para o *middleware* EXEHDA. O trabalho tem como principal objetivo: Desenvolver um Serviço de Atualização de Configurações de *Hardware* - SACH, incorporado ao *middleware* EXEHDA - *Execution Environment for Highly Distributed Applications*, utilizando dispositivos lógicos programáveis - FPGAs (*Field Programmable Gate Array)* como dispositivos de borda reconfiguráveis, em um ambiente pervasivo. Os resultados obtidos nos testes foram satisfatórios, o SACH gerenciou pacotes de reconfiguração, validando e distribuindo os mesmos aos dispositivos.

Palavras-chave: Computação Pervasiva, Computação Reconfigurável.

TITLE: “MANAGING BORDER RECONFIGURABLE DEVICES IN PERVASIVE COMPUTING”

RESUMO

With the increasing development of new technologies for networks of computers with wireless and the miniaturization of devices and increase the capacity of processing and storage of information seen in recent years, the computing and the different types of computing devices are increasingly present in the day-to-day. A new area that comes of this scenario is the Pervasive Computing. Unlike traditional desktop systems, computing systems are in contact with the user anywhere at any time in this new paradigm. The Pervasive Computing can be understood as a computational environment, in which the user is freed of the management of the implementation of its applications, and can devote itself to the desired features. Reconfigurable Computing, in turn, provides logic devices with the possibility of adding the performance of a fixed hardware with the flexibility of the hardware reprogrammed through software. However, there are tools that can perform reconfigurations these devices in environments of high heterogeneity, high geographic dispersion and large number of devices. In that sense, this work implements a service to update reconfigurable hardware configurations - SACH, which enables the spread of package bitstreams to multiple FPGA (Field Programmable Gate Array) devices for the middleware EXEHDA. The work has as main goal: develop a service-Update Settings Hardware - SACH, embedded in the middleware EXEHDA - Execution Environment for Highly Distributed Applications, using programmable logic devices - FPGAs - Field Programmable Gate Array, as devices to edge reconfigurable, an environment pervasivo. The results obtained in the tests were satisfactory, the SACH managed packages of reconfiguration, validating and distributing them to the devices.

Palavras-chave: Pervasive Computing, Reconfigurable Computing.

1 INTRODUÇÃO

Este capítulo apresenta o contexto da pesquisa realizada, as motivações deste trabalho, bem como discrimina os objetivos tanto geral como os especificos desta dissertação.

1.1 Contexto e Histórico

Vários benefícios são decorrentes da difusão dos sistemas computacionais, tais como mobilidade, velocidade e comodidade. Porém, esses benefícios têm seu custo. Na maioria das vezes, as pessoas são obrigadas a mudar de hábitos e métodos a fim de se adaptarem à onipresença do computador, ou seja, à ubiqüidade (WEISER, 1991). Segundo Weiser, as facilidades computacionais devem ser incorporadas aos ambientes, a fim de auxiliar as atividades humanas, mudando minimamente a forma como tais atividades são realizadas.

Em sua visão em 1991, Weiser previa o crescimento no mercado dos dispositivos computacionais ubíquos de vários tamanhos: pequenos e pessoais (*inch-scale*), de médio porte (*foot-scale*) e grandes e de uso coletivo (*yard-scale*). De fato, isso acabou acontecendo: dispositivos pessoais de pequeno e médio porte tais como PDAs, *tablets* digitais e *laptops* tornaram-se comuns no final da década de 90. Da mesma forma dispositivos maiores, tais como lousas eletrônicas, passaram a fazer parte de ambientes de uso comum, tais como salas de reuniões, salas de aula e laboratórios.

Um outro aspecto importante colocado por Weiser, que também se confirmou é dado pelas novas aplicações computacionais que surgiram para explorar o uso desses novos dispositivos. O desenvolvimento de tais aplicações está diretamente associado a três temas, que atualmente são os principais focos de pesquisa na área da Computação Pervasiva, (*i*) interfaces naturais, (*ii*) computação ciente de contexto e (*iii*) captura e acesso de atividades humanas (ABOWD; MYNATT, 2000).

À medida que ambientes pervasivos tornam-se uma realidade e a computação não intrusiva passa a fazer parte da vida das pessoas, uma nova questão torna-se relevante que é a dimensão temporal das interações humanas. Essa questão fez surgir um novo tema que se associa aos três anteriormente citados: a computação no cotidiano *(everyday computing)*, segundo a qual se faz necessário prover disponibilidade de serviço vinte quatro horas por dia, sete dias por semana, de forma a auxiliar computacionalmente atividades informais e não estruturadas comuns no dia-a-dia das pessoas.

A *Internet* e a difusão da Computação Distribuída estão conduzindo para o paradigma da Computação Pervasiva. A era do *mainframe* é caracterizada por muitas pes-

soas compartilhando um único e raro recurso computacional. Na era da computação pessoal, essa relação muda para um computador para cada pessoa. A *Internet* e a difusão da computação distribuída representam uma fase de transição, agregando componentes das eras do *mainframe* e da computação pessoal através do uso massivo do modelo Cliente-Servidor, em que os computadores pessoais são os clientes *webservices* e os *mainframes*, os servidores *webservices*. Essa transição conduz a uma terceira era: a da Computação Pervasiva, com vários computadores disponíveis para cada pessoa. Esses computadores estariam embutidos em praticamente tudo (nas paredes, na mobília e até nas roupas, etc.) e, principalmente, estariam interligados à *Internet*, formando, assim, uma gigantesca rede e possibilitando as mais diversas formas de utilização, (WEISER; BROWN, 1997).

1.2 Motivação

Três grandes motivações foram centrais no desenvolvimento deste trabalho de dissertação: a perspectiva de poder explorar a capacidade das arquiteturas reconfiguráveis no provimento de soluções para o atendimento das demandas introduzidas pela Computação Pervasiva; prover uma proposta que permita a reprogramação de dispositivos de forma distribuída e concorrente, sem intervenção humana direta, de arquiteturas reconfiguráveis de borda - FPGAs e dotar o *middleware* EXEHDA de um novo serviço. Este *middleware* é direcionado à Computação Pervasiva e o *hardware* previsto é do tipo *FPGA*.

Dentre os exemplos de aplicações deste novo serviço para o EXEHDA, destaca-se: os sistemas básicos de medição geral de energia elétrica e água, sistemas embarcados como as máquinas de auto-atendimento, postos de atendimento de sistemas bancários e máquinas de refrigerantes e outros.

1.3 Objetivos

O presente trabalho tem por objetivo geral:

- Desenvolver um Serviço de Atualização de Configurações de *Hardware* - SACH, incorporado ao *middleware* EXEHDA - *Execution Environment for Highly Distributed Applications*, utilizando dispositivos lógicos programáveis - FPGAs (*Field Programmable Gate Array)* como dispositivos de borda reconfiguráveis, em um ambiente pervasivo.

Por sua vez, destacaríamos como objetivos específicos:

- Modelar um Serviço de Atualização de Configurações de pacotes de *bitstream* para os dispositivos de borda - FPGAs na perspectiva da Computação Pervasiva;

- Implementar e testar o serviço modelado, utilizando como estudo de caso o *middleware* EXEHDA.

- Validar a operação do SACH no gerenciamento dos pacotes de reconfiguração de *hardware* a serem utilizados.

1.4 Estrutura do Texto

No Capítulo 2 são abordadas as definições e evoluções, relacionadas ao paradigma da Computação Pervasiva e as as definições e serviços do *middleware* EXEHDA. Já no Capítulo 3 são descritas as definições de Computação Reconfigurável. No Capítulo 4 é abordada a modelagem do Sistema de Atualização de Configurações de *Hardware*, bem como, a implementação de um protótipo, incorporando este serviço ao conjunto de serviços que o *middleware* EXEHDA fornece. Já no Capítulo 5 são descritos o ambiente de avaliação, os resultados obtidos em relação ao número de dispositivos simulados e tamanho de pacote distribuído e no Capítulo 6 são apresentadas as considerações finais, principais conclusões, publicações realizadas e os trabalhos futuros.

2 COMPUTAÇÃO PERVASIVA

Este capítulo apresenta diversas definições que contextualizam o escopo do trabalho, bem como revisa algumas tecnologias utilizadas na área.

2.1 Evolução e Definições

Segundo Weiser (WEISER, 1991), a Computação Ubíqua ou Pervasiva promove facilidades computacionais que devem ser incorporadas ao ambiente a fim de auxiliar atividades humanas de forma transparente. Dentro deste contexto, o computador tem a capacidade de acessar informações do ambiente no qual ele está inserido para a construção dinâmica de modelos computacionais.

Na perspectiva da Computação Pervasiva, os dispositivos que hoje nos cercam serão enriquecidos de novas facilidades de uso, necessitarão pouca manutenção, serão leves e portáteis, contemplarão bom poder de processamento e terão suporte à conectividade intermitente. As áreas de aplicação da computação pervasiva incluem: ensino, trabalho colaborativo, residências e automóveis inteligentes, entre outras (AUGUSTIN et al., 2006).

É possível verificar as transformações da Computação Móvel e da Computação em Grade, em relação a demanda real e qualificada de produtos, serviços e pesquisas. A Computação Pervasiva pode ser construída através da integração de três áreas da Computação: Computação Móvel, Computação em Grade e a Computação Consciente ao Contexto (YAMIN et al., 2005).

A Computação Pervasiva começou a ser potencializada com a evolução das Redes de Computadores e dos Dispositivos Móveis e Embarcados, quando confrontada com aspectos de heterogeneidade elevada, mobilidade, disponibilidade de dados e serviços, e adaptação ao contexto (YAMIN et al., 2005). A Figura 2.1 caracteriza, a consolidação do cenário em direção à Computação Pervasiva.

A área é de grande importância e vem sendo tratada sob diversas visões: Computações em Grade, com Consciência ao Contexto, Móvel, Pervasiva ou Ubíqua e outros tantos que têm sido usados muitas vezes como sinônimos, embora sejam diferentes conceitualmente, e empreguem diferentes idéias de organização e gerenciamento dos serviços computacionais. Com a evolução da área, os conceitos associados as visões com o passar do tempo vão se consolidando. Resumos dos conceitos associados são apresentados na seqüência.

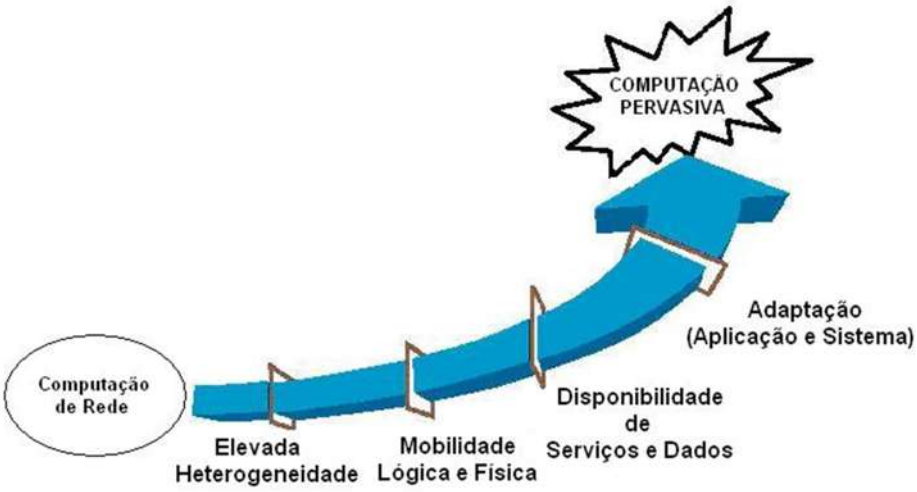

Figura 2.1: Consolidação do Cenário da Computação Pervasiva
(YAMIN, 2004)

2.1.1 Computação em Grade

A Computação em Grade introduz conceitos novos para infra-estruturas de tecnologia da informação, pois suporta uma Computação Distribuída sobre uma rede de recursos dispersos e heterogêneos, e trabalha para otimizar o uso de recursos já disponíveis com vistas à redução de investimentos de capital. Registra-se que a Computação em Grade tem um histórico de uso bem sucedido na computação de alguns tipos de aplicação de alto desempenho. A infra-estrutura pode consistir-se de todos os recursos em rede, desde computadores e dispositivos de armazenamento de variados tipos de capacidade até base de dados. A próxima geração associa novas tecnologias à Computação em Grade, como *webservices*, P2P e Computação Móvel (YAMIN, 2004).

2.1.2 Computação com Consciência de Contexto

Uma das primeiras pesquisas de investigação em computação com consciência de contexto foi apresentada no trabalho *Olivetti Active Badge* (WANT et al., 1992). Outro trabalho que teve forte significado na consolidação da área foi introduzido por Schilit que considera a Computação com Consciência de Contexto ser uma estrutura de *software* que se adapta de acordo com sua localização, coleção de pessoas e objetos próximos, bem como mudanças que ocorrem com os objetos durante determinado tempo (SCHILIT; THEIMER, 1994). Desde então, existem muitas tentativas para definir a Computação com Consciência de Contexto.

Segundo Dey, consciência de contexto foi se tornando de certa forma sinônimo de outros termos: adaptável, reativo, responsivo, sensível ao contexto. As diferentes definições de consciência de contexto estão inseridas em duas grandes categorias: uso de contexto e adaptação de contexto (DEY; ABOWD, 2000).

As definições prévias para estabelecer aplicações com consciência de contexto não são adequadas. Por isso Dey, define que um sistema é consciente de contexto quando o sistema usa o contexto para fornecer informações relevantes e/ou serviços para o usuário (DEY, 2001).

2.1.3 Computação Móvel

Dependendo da área, a Computação Móvel pode ser definida de diferentes formas:

- No caso do *hardware*, associamos ao termo Computação Móvel, a possibilidade de mobilidade física dos equipamentos. Um exemplo é o uso de *notebooks*, *laptops* e PDAs;

- No caso do *software*, chamamos de Computação Móvel quando um programa se desloca entre equipamentos interconectados por rede (CARDELLI, 2000). Neste caso, a Computação Móvel pode começar sua execução em uma célula de uma rede e em certo ponto ser movida para uma outra célula da estrutura distribuída dando continuidade a sua execução. Assim a computação pode ser executada localmente em várias células, utilizando o máximo de processamento e recursos disponíveis na rede. A mobilidade de *software* traz também grandes vantagens para os usuários de dispositivos móveis. Um usuário de um dispositivo de pouco poder computacional pode enviar um programa para ser executado em um outro equipamento disponível na rede e logo depois desconectar-se, reconectando novamente mais tarde para receber os resultados dessa computação.

2.1.4 Computação Pervasiva ou Ubíqua

Mark Weiser, considerado o pai da Computação Pervasiva ou Ubíqua, vislumbrou há uma década atrás, que no futuro, os computadores habitariam os mais triviais objetos.

Exemplos desses objetos seriam: etiquetas de roupas, xícaras de café, interruptores de luz, canetas e outros objetos de forma invisível para o usuário (WEISER, 1991).

Nesta perspectiva, a Computação Pervasiva, promove facilidades computacionais que devem ser incorporadas ao ambiente a fim de auxiliar atividades humanas mudando minimamente a forma como tais atividades são realizadas. Essa forma transparente de integrar tecnologia às atividades diárias das pessoas foi denominada Computação Pervasiva, (WEISER, 1991).

De maneira geral, o **grau de embarcamento** fornece o nível de inteligência dos computadores que estão inseridos dentro de um ambiente pervasivo, detectando, explorando e construindo dinamicamente vários modelos computacionais de seus ambientes.

Neste sentido, entende-se como Computação Pervasiva aquela onde os recursos de computação estariam onipresentes na vida diária de cada usuário e que estariam conectados com a finalidade de fornecer a informação ou serviços que os usuários solicitassem de qualquer lugar e em qualquer momento.

De modo mais específico, Mark Weiser definiu como característica da Computação Pervasiva, uma intercomunicação entre os computadores presentes em vários objetos PDAs, PCs, *laptops*, celulares, geladeiras, etc.).

Por sua vez, a disseminação dos computadores tanto em número quanto na dispersão geográfica, sugere além da disponibilidade de infra-estrutura computacional, um novo paradigma inspirado pelo acesso constante à informação e às capacidades computacionais (ABOWD; MYNATT, 2000).

O termo Computação Pervasiva ficou agregado à IBM quando da edição intitulada *Pervasive Computing* do *IBM System Journal* (PERVASIVE *Computing IBM System Journal*, 1999) (M., 1999), onde foi organizada uma digressão sobre os aspectos promissores da Computação Pervasiva, nesta mesma edição foi resgatada por Weiser, no artigo

intitulado *The origins of ubiquitous computing research at PARC in the late 1980s*, a sua visionária proposta quanto ao futuro da computação, na qual recursos de computação onipresentes se ajustariam, de forma autônoma, para atender os usuários.

Do ponto de vista da usabilidade, a Computação Pervasiva contempla um ambiente carregado de dispositivos de computação e comunicação que vai interagir com o homem de forma tranqüila - *(Calm Technology)*. O usuário não perceberá que estará envolvido com máquinas, o ambiente interage com ele de forma autônoma e interativa (WEISER; BROWN, 1997). Embora esta proposta esteja ainda longe da realidade, vem sendo concretizada aos poucos, através da disponibilização de novas tecnologias como PDAs, *SmartPhones, Palms* e outros e também a consolidação de padrões para redes sem fio como o *bluetooth* e o IEEE 802.11, (YAMIN, 2004).

O objetivo da Computação Pervasiva é criar Ambientes Inteligentes (salas de aula, residências, escritório, hospitais, automóveis, etc.), onde os dispositivos em redes embebidos no ambiente forneça conexão discreta todo o tempo, melhorando assim a experiência do homem e a qualidade de vida sem conhecimento explícito sobre as comunicações e as tecnologias de computação. A informação caracteriza a situação de uma entidade, no caso, pessoa, lugar e objeto que é importante para o usuário e a aplicação.

2.2 EXEHDA: Conceitos e Tecnologias

O EXEHDA é um *middleware* direcionado a promover o suporte às aplicações distribuídas, móveis e conscientes do contexto da Computação Pervasiva (YAMIN et al., 2005). O EXEHDA fornece serviços com diferentes funcionalidades, dentre estes destacaríamos: migração de componentes de *software* de uma localização física para outra; persistência, aumentando a disponibilidade e o desempenho no acesso aos dados; descoberta de recursos, dando suporte ao movimento dos dispositivos móveis e dos componentes entre diferentes células; comunicação, que possibilita ser anônima e assíncrona; escalonamento, decidindo onde executar as aplicações, (YAMIN, 2004).

Com o objetivo de baixar o custo de especificar os aspectos necessários para o tratamento da semântica **siga-me** relacionados à mobilidade lógica e física, de distribuição e de adaptação, os mecanismos referentes aos aspectos oferecidos na linguagem programação, estão integrados ao ambiente de execução. A arquitetura de *software* do EXEHDA está resumida na Figura 2.2.

Desta forma, a arquitetura, modelada com esta perspectiva, apresenta uma organização lógica em três camadas (YAMIN, 2004):

- Camada de aplicação;
- Camada de suporte e ambiente de execução;
- Camada de sistemas básicos.

A **camada de aplicação** situa-se no topo, ilustrada na Figura 2.2, está a linguagem de programação, no qual disponibiliza abstrações para programação de aplicações distribuídas, móveis e conscientes do contexto direcionadas à Computação Pervasiva.

Na **camada intermediária** encontram-se os mecanismos de suporte à execução da aplicação pervasiva e às estratégias de adaptação ao *middleware* EXEHDA (YAMIN, 2004). Esta camada é formada por dois níveis.

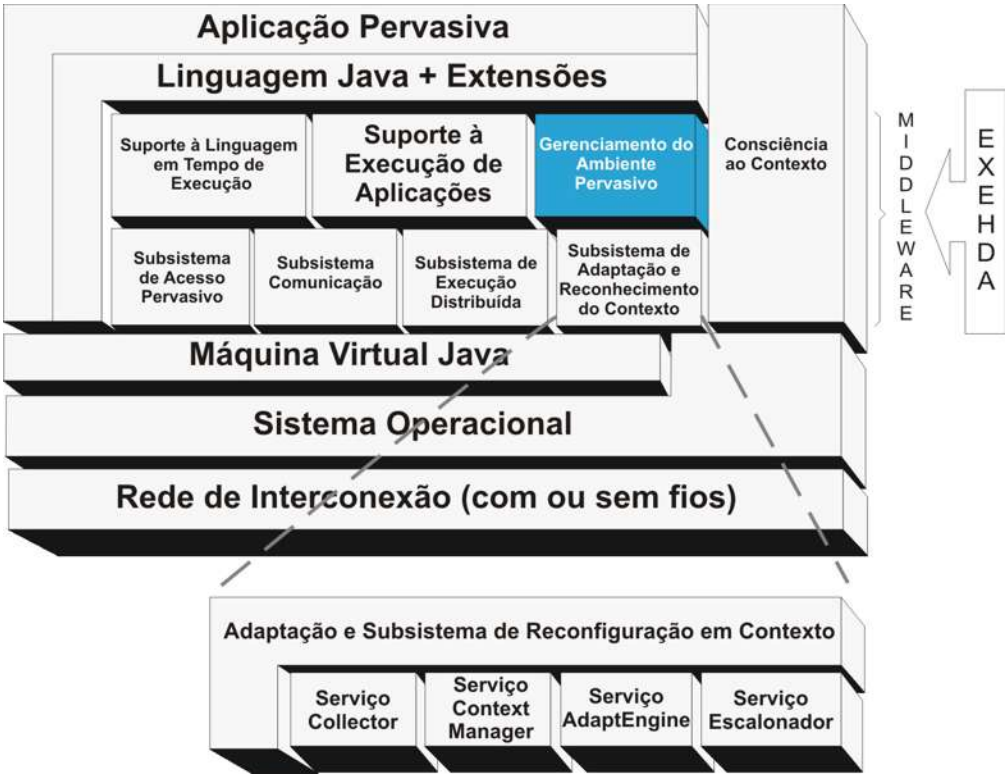

Figura 2.2: Visão Geral da Arquitetura EXEHDA
(YAMIN, 2004)

O **primeiro nível** é composto por três módulos de serviço à aplicação:

- Acesso pervasivo a código e dados;
- Reconhecimento de contexto;
- Ambiente de execução da aplicação.

O Acesso Pervasivo a Dados e Código é formado pelos componentes que disponibilizam o Ambiente Pervasivo, incluindo Ambiente Virtual do Usuário (*AVU*), Ambiente Virtual da Aplicação (*AVA*) e Base de Dados pervasiva das Aplicações (*BDA*).

O Ambiente Virtual do Usuário é composto pelos elementos que integram a interface de interação do usuário com o sistema. Este módulo é responsável pela implementação do suporte para que a aplicação que o usuário está executando em uma localização possa ser instanciada e continuada em outra localização sem descontinuidade, permitindo o estilo de aplicações (*follow me*) em um ambiente pervasivo.

O desafio da adaptabilidade para implementar o AVU é suportar os usuários em diferentes localizações, com diferentes sistemas de interação, que demandam diferentes sistemas de apresentação, dentro dos limites da mobilidade.

O *AVU* é o conjunto de atributos que identifica uma execução específica de uma aplicação, enquanto a *BDA* constitui o repositório de códigos das aplicações em geral.

O Ambiente de Execução da Linguagem tem a função de se encarregar pelo gerenciamento da aplicação durante seu tempo de vida. O SACH está aninhado na camada intermediária do primeiro nível no módulo de Gerenciamento do Ambiente Pervasivo, conforme é apresentado na Figura 2.2.

O Serviço de Reconhecimento de Contexto tem a função de informar o estado dos elementos de contexto de interesse da aplicação e do próprio ambiente de execução.

No **segundo nível**, localizada na camada intermediária, estão os serviços básicos do EXEHDA, que são compostas pelas funcionalidades necessárias para o primeiro nível e que cobrem vários aspectos, tais como:

- Migração: são os mecanismos para deslocar um componente de *software* de uma localização física para outra;
- Persistência: é o mecanismo para aumentar a disponibilidade e o desempenho do acesso aos dados;
- Descoberta de recursos - dá suporte ao movimento dos dispositivos móveis e dos componentes entre diferentes células;
- Comunicação: possibilidade de ser anônima e assíncrona;
- Escalonamento: permite a decisão do melhor nodo para criar os componentes da aplicação;
- Monitoramento: os sensores que fornecem informações sobre o ambiente de execução e aplicação.

A camada inferior da arquitetura é constituída por sistemas e linguagens nativas que integram o meio físico de execução. Por questões de portabilidade, nesta camada a plataforma base de implementação é a Máquina Virtual Java em suas diferentes abordagens. Duas plataformas são utilizadas:

- J2SE (Java *Standard Edition*);
- J2ME (Java *Micro Edition*).

A existência de uma rede global, como a *Internet*, é suposta por esta arquitetura, e essa rede por sua vez é composta de um suporte à operação sem fio, interligada e outra cabeada que disponibiliza uma infra-estrutura de equipamentos e serviços em escala global (Rede Pervasiva).

A organização da arquitetura é em camadas lógicas, com níveis diferenciados de abstração e está direcionada para a busca de manutenibilidade de qualidade de serviços que é oferecida ao usuário através do conceito de adaptação.

O sistema se adapta para fornecer qualidade dos serviços prestados, enquanto que a aplicação se adapta para atender a expectativa do usuário, mantendo a funcionalidade da aplicação (YAMIN, 2004).

2.2.1 Aspectos de Modelagem e de Serviços

Nesta seção, além dos aspectos de modelagem, será feita uma (*i*) descrição dos principais serviços que compõem o EXEHDA, (*ii*) da sua interdependência, (*iii*) e da sua contribuição na construção do suporte à Computação Pervasiva.

A caracterização dos serviços tem o foco na descrição das funcionalidades providas através da sua interface, não sendo priorizados aspectos de implementação. Particularmente, a interface de cada serviço é documentada usando diagramas de classe da UML *(Unified Modeling Language)* de diagramas de classe.

É entendido como ambiente pervasivo, o ambiente computacional onde recursos e serviços são gerenciados pelo EXEHDA, no objetivo de atender os requisitos impostos pelo perfil da aplicação-alvo do projeto. A sua composição acontece tanto pelos dispositivos dos usuários, como pelos equipamentos da infra-estrutura de suporte, todos instanciados pelo seu respectivo perfil de execução do middleware.

O EXEHDA tem duas grandes perspectivas: (ilustradas na Figura 2.3)

- A primeira é com relação às aplicações da Computação Pervasiva. O EXEHDA é o provedor dos serviços que dão suporte às abstrações definidas quando do desenvolvimento. A interação das aplicações com o meio físico distribuído, através dos serviços disponibilizados pelo EXEHDA proporciona a estas aplicações a visão do ambiente pervasivo;
- E em uma segunda perspectiva, os recursos que compõem o meio físico distribuído: o EXEHDA executa a definição das políticas que normatizam a organização dos recursos e os mecanismos para sua gerência.

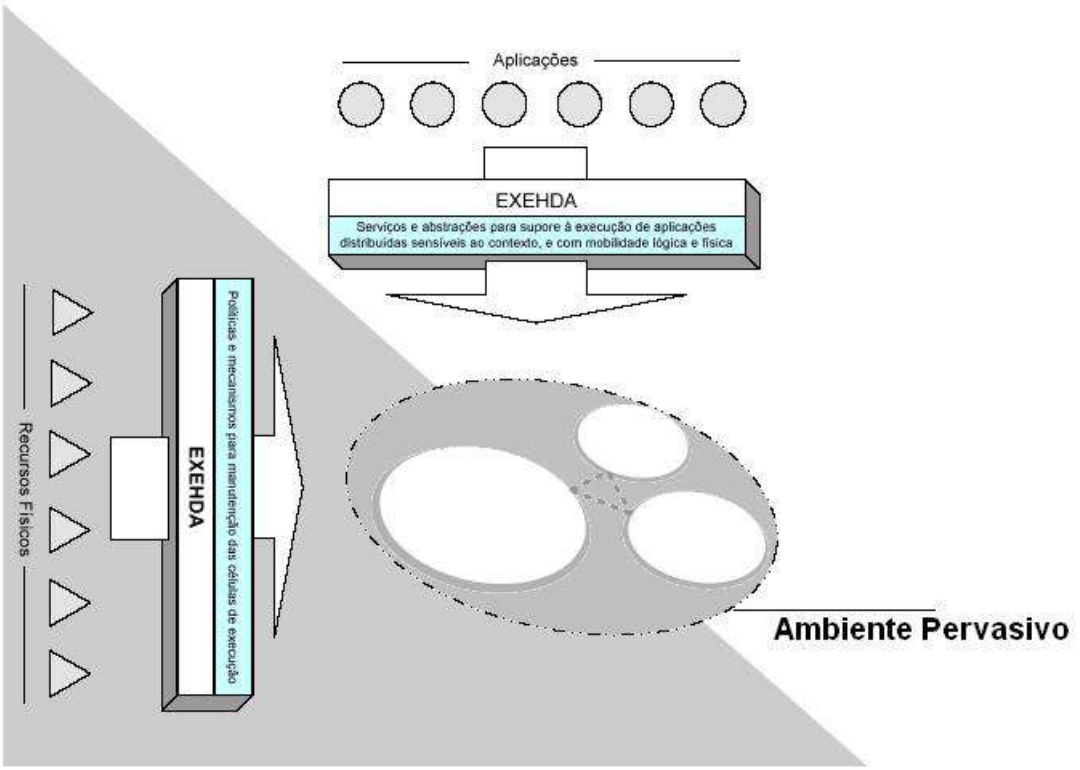

Figura 2.3: Visões de Atuação do EXEHDA
(YAMIN, 2004)

2.2.2 A Organização do Ambiente Pervasivo

Com o objetivo de integralizar os cenários da Computação em Grade, da Computação Móvel e da Computação Sensível ao contexto, é mapeada em uma organização composta pela agregação de células de execução - em EXEHDAcels,

ilustrada na Figura 2.4. O meio físico o qual o ambiente pervasivo é definido é constituído por uma rede de infra-estrutura, no qual sua composição final pode ser modificada pela agregação dinâmica de nodos móveis, (YAMIN, 2004).

Os recursos da infra-estrutura física são mapeados para três abstrações básicas, utilizadas na construção do ambiente pervasivo:

- EXEHDAcel: é composta por duas abstrações principais: EXEHDAbase e EXEHDAnodos. Os principais aspectos que devem ser considerados na definição da abrangência de uma célula são: o escopo institucional, a proximidade geográfica e o custo de comunicação;

- EXEHDAbase: pode-se definir como o ponto de contato para os EXEHDAnodos. Esta abstração tem a responsabilidade por todos os serviços básicos do ambiente pervasivo e, embora constitua uma referência lógica única, seus serviços, sobretudo por aspectos de escalabilidade, poderão estar distribuídos entre vários equipamentos. Isto é ilustrado na Figura 2.4 pelo sombreado associado ao símbolo da EXEHDAbase;

- EXEHDAnodo: representam os equipamentos de processamento disponíveis no ambiente pervasivo, sendo responsáveis pela execução das aplicações. Um subcaso deste tipo de recurso é o EXEHDAnodo móvel.

Esses nodos do sistema têm uma elevada portabilidade, tipicamente dotados de interface de rede para operação sem fio e, neste caso, integram a célula a qual seu ponto-de-acesso está subordinado. São funcionalmente análogos aos EXEHDAnodos, porém eventualmente com uma capacidade mais restrita (por exemplo, PDAs) (YAMIN, 2004).

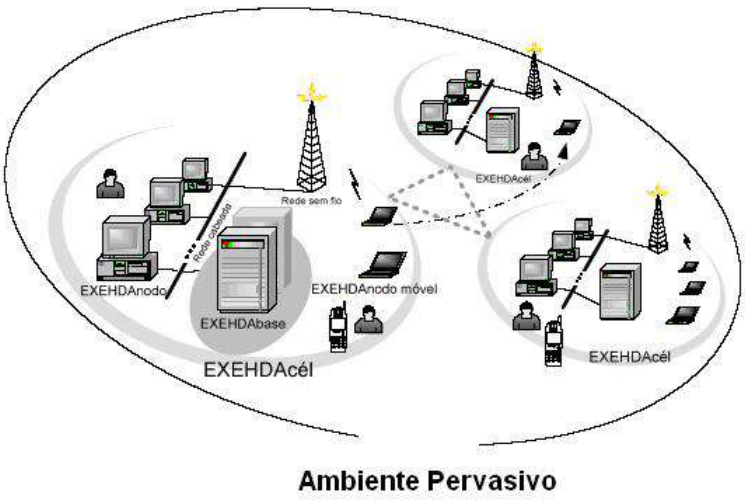

Figura 2.4: Composição do Ambiente Pervasivo
(YAMIN, 2004)

O ambiente pervasivo é composto por equipamentos multi-institucionais, o que direciona a um procedimento de gerência análogo ao utilizado nos ambientes de Grade Computacional - *Grid Computing* (YAMIN, 2004). A forma em que a organização celular do ambiente pervasivo é gerenciada e tem o objetivo central de resguardar a autonomia das instituições envolvidas. A procura de soluções para o gerenciamento multi-institucional tem feito parte da agenda de pesquisa de projetos de Computação em Grade.

2.2.3 O Núcleo

A funcionalidade promovida pelo EXEHDA é personalizável individualmente em cada nodo, sendo determinada pelo conjunto de serviços ativos e controlada por meio de perfis de execução. Um perfil de execução define um conjunto de serviços a ser ativado em um EXEHDAnodo, associando a cada serviço uma implementação específica dentre as disponíveis, bem como definindo parâmetros para sua execução.

O perfil de execução também controla a política de carga a ser utilizada para um determinado serviço, a qual se traduz em duas opções:

- Quando da ativação do nodo (*bootstrap* do *middleware*);
- Sob demanda.

Entretanto, a informação que é definida nos perfis de execução é também consultada quando da carga de serviços sob demanda, assim, a estratégia adaptativa para carga dos serviços acontece tanto na inicialização do nodo, quanto após este já estar em operação e precisar instalar um novo serviço (YAMIN, 2004).

Esta política para carga dos serviços é disponibilizada pelo núcleo mínimo do EXEHDA, o qual é instalado em todo EXEHDAnodo que for integrado ao ambiente pervasivo. Este núcleo é formado por dois componentes, ilustrado na Figura 2.5:

ProfileManager: este componente executa a interpretação da informação disponível nos perfis de execução e a disponibiliza aos outros serviços do middleware. Cada EXEHDAnodo apresenta um perfil de execução individualizado;

ServiceManager: este componente tem a função de ativar os serviços no EXEHDAnodo a partir das informações disponibilizadas pelo *ProfileManager*. Para que aconteça isto, ele carrega sob demanda o código dos serviços do middleware, a partir do repositório de serviços que pode ser local ou remoto, dependendo da capacidade de armazenamento do EXEHDAnodo e da natureza do serviço (YAMIN, 2004).

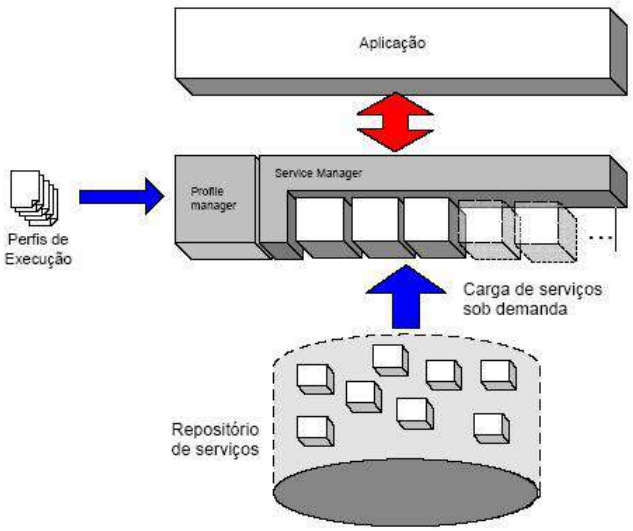

Figura 2.5: Organização do Núcleo do EXEHDA (YAMIN, 2004)

2.2.4 Definindo Perfis de Execução

A definição do perfil de execução do EXEHDA, implica que cada EXEHDAnodo tenha o seu perfil de execução do middleware registrado através de um documento XML *(Extensible Markup Language)* . Este documento traz a associação de nomes simbólicos de serviços a componentes que implementam a interface definida para os mesmos. O nome canônico de um serviço é padronizado, sendo determinado pela interface exportada. É possível, também, definir no perfil de execução para um determinado nodo, propriedades em que os valores sejam recuperados em tempo de processamento, com o objetivo de personalizar o funcionamento de um serviço. A forma genérica do documento XML que descreve o perfil de execução do middleware é ilustrado na Figura 2.6. Entretanto, um exemplo de documento instanciado para definição de perfil de execução do EXEHDA pode representado na Figura 2.7.

```
<EXEHDA>
 <PROFILE name="profileName">
  <SERVICE name="sName" impl="className" loadPolicy="boot"|"demand">
   <PROP name="paramName" value="paramValue" />
  </SERVICE>
 </PROFILE>
</EXEHDA>
```

Figura 2.6: Formato do Documento de Definição de Perfil de Execução do EXEHDA (YAMIN, 2004)

Na Figura 2.7, um bloco *PROFILE* define um perfil de execução, sendo o valor do atributo *name* quem define o nome associado àquele perfil de execução. Internamente ao bloco *PROFILE*, blocos *SERVICE* são utilizados para descrição dos serviços que integram aquele perfil de execução específico.

No tocante ao bloco *SERVICE* pode-se descrever os seguintes atributos: *name* que caracteriza o nome do serviço, o *impl* especifica o componente (uma classe Java na versão atual do protótipo do EXEHDA) e o atributo *loadPolicy* define a política de carga que o núcleo do EXEHDA deve utilizar para o serviço.

Os elementos PROP são usados dentro de um bloco *SERVICE* para definir as propriedades que poderão ser recuperadas em tempo de execução pelo serviço, para personalização de sua execução e a parametrização dos serviços, via perfil de execução, também tem o papel de suportar a visão unificada da EXEHDAbase, repassando para os serviços em execução nos EXEHDAnodos à estratégia específica que deve ser utilizada para localização da instância celular daquele serviço (YAMIN, 2004).

Duas estratégias estão previstas:

- A primeira estratégia é uma referência para o dado armazenado na CIB *(Cell Information Base)*;
- A segunda estratégia é uma referência direta para o nodo que hospeda a instância celular daquele serviço.

Esta segunda estratégia, via de regra, é reservada para indicação da instância celular da CIB.

```
<EXEHDA>
        <PROFILE name="desktop-node">

         <SERVICE name="logger"loadPolicy="boot">
           <PROP name="impl" value="org.isam.exehda.services.logging.BasicLogger"/>
           <PROP name="logLevel" value="5000"/>
         </SERVICE>

         <SERVICE name="worb"loadPolicy="boot">
           <PROP name="impl" value="org.isam.exehda.services.worb.WorbImpl"/>
         </SERVICE>

         <SERVICE name="cib"loadPolicy="boot">
           <PROP name="impl" value="org.isam.exehda.services.cib.CibImplClient"/>
           <PROP name="contactAddress" value="//143.54.7.137/cib"/>
         </SERVICE>

         <SERVICE name="bda"loadPolicy="boot">
           <PROP name="impl" value="org.isam.exehda.services.bda.BdaImplClient"/>
           <PROP name="bdaHost" value="lunaris.inf.ufrgs.br"/>
         </SERVICE>

         <SERVICE name="avu"loadPolicy="demand">
           <PROP name="impl" value="org.isam.exehda.services.avu.AvuImplClient"/>
           <PROP name="bdaHost" value="lunaris.inf.ufrgs.br"/>
         </SERVICE>

         <SERVICE name="executor"loadPolicy="boot">
           <PROP name="impl" value="org.isam.exehda.services.primos.ExecutorImpl"/>
         </SERVICE>

         <SERVICE name="gatekeeper"loadPolicy="boot">
           <PROP name="impl" value="org.isam.exehda.services.avu.GatekeeperServices"/>
           <PROP name="port" value="29901"/>
         </SERVICE>

         <SERVICE name="objectseed"loadPolicy="demand">
           <PROP name="impl" value="org.isam.exehda.services.avu.ObjectSeedImpl"/>
         </SERVICE>

          ----
  <\PROFILE>
<\EXEHDA>
```

Figura 2.7: Exemplo de Documento de Definição de Perfil de Execução do EXEHDA (YAMIN, 2004)

2.2.5 Organização Baseada em Serviços

Em um ambiente altamente heterogêneo, o requisito de operação, onde não só o *hardware* que exibe capacidades variadas de processamento e memória, mas também as bibliotecas de *software* disponíveis em cada dispositivo, trouxeram uma motivação com relação a adoção de uma abordagem na qual um núcleo mínimo do middleware tem suas funcionalidades estendidas por serviços carregados sob demanda.

Essa organização se reflete em um padrão de projeto do tipo micro-kernel. Adiciona-se a isto, que a carga sob demanda tem perfil adaptativo. Entretanto, poderá ser utilizada versão de um determinado serviço, melhor sintonizada às características do dispositivo em questão. Isto é possível, porque na modelagem do EXEHDA, os serviços estão definidos por sua interface, e não pela sua implementação (YAMIN, 2004).

Por isso que a contra-proposta à estratégia micro-kernel de um único binário monolítico, cujas funcionalidades cobrissem todas as combinações de necessidades das aplicações e dispositivos, se mostra impraticável na Computação Pervasiva, cujo ambiente computacional apresenta elevada heterogeneidade de recursos de processamento.

Contudo, o requisito do middleware de manter-se operacional durante os períodos de desconexão planejada motivou, além da concepção de primitivas de comunicação adequadas a esta situação, a separação dos serviços que implementam operações de natureza distribuída em instâncias locais ao EXEHDAnodo (instância nodal), e instâncias locais à

EXEHDAbase (instância celular) (YAMIN, 2004).

Neste sentido, o relacionamento entre instância de nodo e celular assemelha-se à estratégia de *proxies*, enquanto que o relacionamento entre instâncias celulares assume um caráter P2P. A abordagem P2P nas operações inter-celulares está alinhada com requisito de escalabilidade. Com isso, os componentes da aplicação em execução em determinado dispositivo podem permanecer operacionais, desde que, para satisfação de uma dada requisição pelo middleware, o acesso a um recurso externo ao dispositivo seja prescindível.

Por outro lado, a instância celular, em execução na base da célula, provê uma referência para os outros recursos, no caso da realização de operações que requeiram coordenação distribuída. Neste sentido, observe-se que a EXEHDAbase é uma entidade estável dentro da EXEHDAcel, permitindo que os demais integrantes, no caso os recursos, da célula tenham um caráter mais dinâmico no que se refere a sua disponibilidade na célula.

A organização do núcleo mínimo do EXEHDA, e o conjunto atual de serviços projetados para atender as necessidades da arquitetura, são apresentados nas seções seguintes. Os serviços do EXEHDA estão organizados em quatro grandes subsistemas, ilustrado na Figura 2.8.

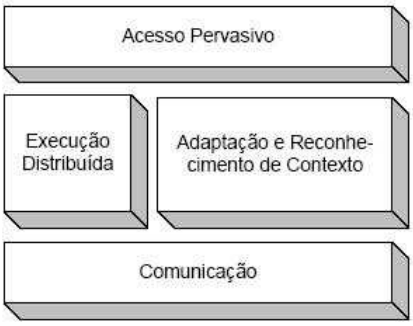

Figura 2.8: Organização dos Subsistemas do EXEHDA
(YAMIN, 2004)

1. Execução distribuída;
2. Adaptação;
3. Comunicação;
4. Acesso pervasivo.

A seguir estão descritos os vários subsistemas que compõem o EXEHDA. Seu estudo teve por objetivo reunir subsídios para modelagem e implementação do protótipo SACH, resguardando a integração do mesmo a infra-estrutura de *software* já existente.

Com o intuito de preservar a objetividade do texto serão comentados com um pouco mais de detalhes, os serviços cuja a relação com o SACH é mais direta. É importante observar de modo indireto, quando do processamento de aplicações ocorrem uma interação que via de regra irá envolver os serviços do EXEHDA como um todo.

2.2.5.1 Subsistema de Execução Distribuída

Este subsistema tem a responsabilidade de dar suporte ao processamento distribuído no EXEHDA. O objetivo é produzir uma execução pervasiva. O subsistema interage com outros subsistemas do EXEHDA como o subsistema de reconhecimento de contexto e adaptação, para produzir o comportamento distribuído e adaptativo às aplicações da arquitetura. O subsistema é composto pelos seguintes serviços: (YAMIN, 2004)

Serviço *Executor*

Este serviço tem o objetivo de acumular as funções de disparo de aplicações, e de criação e migração de seus objetos. Com relação à implementação destas funções é empregada a instalação de código sob demanda. Ele interage com os serviços *CIB* e *BDA*. Já a interface do serviço Executor define métodos que controlam os ciclos de vida das aplicações e dos objetos.

Cell Information Base - CIB

O objetivo deste serviço é implementar a base de informações da célula. A funcionalidade deste serviço se relaciona com a manutenção da infra-estrutura distribuída que constrói o ambiente pervasivo. Os dados estruturais do EXEHDAcel são mantidos por este serviço, tais como:

- Informações sobre os recursos que a compõem;
- Informação de vizinhança;
- Atributos que descrevem as aplicações em execução.

O serviço se responsabiliza também pela manutenção das informações referentes aos usuários registrados na célula. Estas mesmas informações incluem um certificado assinado que mantém a chave pública do usuário.

OXManager

Toda a gerência e manutenção da meta-informação associada a um *OX* - Objeto eXehda é uma função do serviço *OXManager*. Este serviço consiste de uma instância de objeto, que é desenvolvida por intermédio do serviço Executor, podendo ser associada a meta-informação em tempo de execução.

O serviço faz todas as operações de consulta e atualização dos atributos do OX no ambiente pervasivo, possibilitando o acesso a partir de qualquer nodo.

Este subsistema também é formado pelos serviços *Discoverer, ResourceBroker, Gateway, StdStreams, Logger* e *Dynamic Configurator*. Estes serviços embora componham a infra-estrutura de suporte à execução, e deste modo participem das computações em geral, apresentam uma relação indireta com o SACH.

2.2.5.2 Subsistema de Reconhecimento de Contexto e Adaptação

O *middleware* EXEHDA apresenta um suporte à adaptação que está associado à operação do Subsistema de Reconhecimento de Contexto e Adaptação. O mesmo subsistema proporciona serviços desde a extração da informação bruta sobre as características dinâmicas e estáticas dos recursos que compõem o ambiente pervasivo, passando pela identificação em alto nível dos elementos de contexto, até o disparo das ações

de adaptação em reação a modificações no estado de tais elementos de contexto. O subsistema é composto pelos seguintes serviços (YAMIN, 2004). Este subsistema também é formado pelos serviços *Collector, Deflector, ContextManager, AdaptEngine* e *Scheduler*. Os quais não tem interface direta com o SACH.

2.2.5.3 Subsistema de Comunicação

Tanto a mobilidade de *hardware* como a de *software*, na maior parte das vezes, não possibilita a interação contínua entre os componentes da aplicação distribuída. As desconexões ocorrem constantemente, por motivo de links sem fio que podem existir ou por uma estratégia de economizar energia nos dispositivos móveis. Este subsistema oferece mecanismos que apresentam estes aspectos da Computação Pervasiva (YAMIN, 2004). O subsistema é composto pelos seguintes serviços:

WORB

O objetivo do serviço *WORB* é de simplificar a construção de serviços distribuídos. Este serviço possibilita aos programadores focalizarem os esforços no refinamento da semântica distribuída integrada ao serviço em desenvolvimento, abstraindo aspectos de baixo nível relativos ao tratamento de comunicações em rede. Este serviço oferece também um modelo de comunicação que se baseia nas invocações remotas de método, similar ao *Remote Method Invocaction* - RMI, porém sem exigir a manutenção da conexão durante toda a execução da chamada remota.

Este subsistema também é formado pelos serviços *Dispatcher* e *CCManager*, os quais se relacionam de forma indireta com o SACH.

2.2.5.4 Subsistema de Acesso Pervasivo

O acesso a dados e códigos em qualquer lugar, a qualquer tempo, na Computação Pervasiva necessita de um suporte do middleware. Os serviços que fazem parte deste subsistema no EXEHDA são: *BDA*, *AVU*, *SessionManager* e *Gatekeeper* (YAMIN, 2004). O subsistema é composto pelos seguintes serviços:

BDA

Um ambiente de Computação Pervasiva tem por características a possibilidade do usuário, disparar aplicações a partir de qualquer nodo integrante do sistema e, após esse disparo, ter uma mobilidade parcial ou integral de tais aplicações em resposta as modificações em seu contexto de execução, como, por exemplo, a movimentação do usuário ou alteração na condição de carga dos dispositivos atualmente em uso pela aplicação.

Com isso a capacidade de instalação de código sob demanda é uma necessidade prescindível à execução de aplicações para a Computação Pervasiva. Seria impossível manter todo o universo de *software* disponível instalado e atualizado em todos os dispositivos do sistema. Entretanto, a implementação do mecanismo de instalação sob demanda, necessita da existência de um repositório de código que forneça a mesma visão do *software* disponibilizado a partir de qualquer dispositivo do ambiente pervasivo, mesmo após migrações.

Com isso, outras funcionalidades para o repositório pervasivo são necessárias. Considerando as diversas aplicações disponibilizadas segue linhas de evolução indepen-

dentes, o suporte a controle de versões é oportuno na direção da manutenção da operacionalidade das diferentes aplicações.

Este subsistema também é formado pelos serviços *AVU, SessionManager* e *Gatekeeper*, os quais não são usados diretamente pelo SACH.

2.2.6 Conclusão

Este capítulo, apresenta as evoluções e definições com relação ao paradigma da Computação Pervasiva ou Obíqua, bem como, outras definições com relação a Computação em Grade, Computação com Consciência de Contexto e também apresenta o *middleware* EXEHDA desenvolvido para promover o suporte à execução de aplicações para a Computação Pervasiva, principalmente em relação aos problemas de heterogenidade, mobilidade física e lógica, disponibilidade de dados e serviços e adaptação do contexto com relação a aplicação e serviços. Este capítulo descreve a definição do *middleware*, a sua arquitetura, os aspectos de modelagem, a organização do ambiente pervasivo e dos serviços, como é formado o núcleo e no final descreve resumidamente alguns dos serviços que são utilizados de forma direta no SACH e apenas relaciona os outros serviços que não são diretamente usados.

3 COMPUTAÇÃO RECONFIGURÁVEL

Neste capítulo são discutidos vários tópicos com relação as arquiteturas reconfiguráveis. Primeiramente é abordada uma breve introdução sobre a classificação das soluções de *hardware* em função da implementação, relatando suas vantagens e desvantagens. Na seção 3.2 apresenta as definições da Computação Reconfigurável. Já na seção 3.3 é descrita uma abordagem geral em relação ao FPGA (*Field Programmable Gate Array*), apresentando a arquitetura do mesmo, as vantagens e desvantagens na utilização, a classificação conforme o modo de configuração e granularidade dos dispositivos, tipos de FPGAs com reconfiguração dinâmica total e parcial como os da família *Virtex II Pro*, as aplicações do FPGA e, finalizando, a conclusão do capítulo.

3.1 Introdução

Atualmente percebe-se a ocorrência de um crescimento considerável na utilização de computadores. As soluções de *hardware* podem ser classificadas em função da sua implementação em dois grandes grupos (SILVA MARTINS et al., 2005):

- *Hardware* fixo;
- *Hardware* programável através de *software*.

A escolha entre *hardware* fixo e programável através de *software* envolve uma escolha entre desempenho e flexibilidade. *Hardware* fixo é mais eficiente e atinge melhor desempenho e menor consumo. No entanto, não permite a flexibilidade de atualização que o *hardware* programável através de *software*, disponibiliza.

Geralmente estes tipos de implementações são denominadas como: soluções em *hardware* e soluções em *software*.

As desvantagens relacionadas com o desempenho, flexibilidade e custo de cada um desses tipos de implementações de soluções abordadas anteriormente são os principais problemas que deram origem ou motivaram o surgimento da computação reconfigurável.

A falta de uma maior flexibilização e a sub-utilização com relação as aplicações diversas são os principais problemas ou deficiências do paradigma de *hardware* fixo. O motivo dessas deficiências é que o dispositivo fixo não pode ser alterado após a sua fabricação e também por não se conseguir adaptá-lo em certas situações que ocorrem durante as execuções de diversas tarefas computacionais.

Já no paradigma de *software* (*hardware* programável mais *software*), o desempenho é a principal deficiência. Mesmo com a grande flexibilidade para a execução de

qualquer tipo de trabalho que envolva processamento computacional, registra-se menor desempenho para executar certos tipos de tarefas e operações. Caso essas operações sejam executadas em um dispositivo de *hardware* fixo, o tempo de execução seria menor, pois não haveria necessidade de buscar instruções na memória, decodificá-las, buscar dados, e executar as operações requeridas. Além disso, a execução mais eficiente em *hardware* também tende a consumir menos energia.

Quando ocorre um aumento do número de operações mais críticas, isto é, operações que não podem parar de funcionar ou que não podem perder os dados, verifica-se um acréscimo do tempo de execução e por conseqüência o desempenho através dos dispositivos de *hardware* programável mais *software* pode ser insatisfatório. Devido a este motivo, são utilizados os dispositivos com *hardware* fixo para conseguir um aumento satisfatório da velocidade de algumas operações ou aplicações.

Entretanto, apesar das deficiências, os dois paradigmas possuem suas aplicações mais indicadas e recomendadas. Para escolher uma delas irá depender das características de cada projeto.

Existe ainda a necessidade de que haja um desenvolvimento de novos modelos e tipos de implementações de soluções computacionais. Devido a este fato, necessita-se de um novo tipo de implementação intermediária entre as tradicionais soluções em *hardware* fixo e *software* (*Hardware* Programável mais *Software*). Com o uso desse novo tipo de implementação computacional, será possível alcançar ou aproximar o desempenho das soluções implementadas em *hardware* fixo e da flexibilidade das soluções implementadas em *hardware* programável e *software* (SILVA MARTINS et al., 2005).

Então, surge a terceira forma intermediária para implementar soluções de reconfiguração de *hardware* chamadas de Arquiteturas Reconfiguráveis. Com o uso desse novo tipo de implementação computacional, é possível aproximar ou até mesmo alcançar o desempenho das soluções implementadas em *hardware* fixo com uma flexibilidade similar àquela encontrada nas soluções implementadas em *hardware* programável por *software* (SILVA MARTINS et al., 2005).

As soluções baseadas em arquiteturas reconfiguráveis agregam o desempenho de um *hardware* fixo com a flexibilidade de um *hardware* programável, solucionando problemas mais complexos.

3.2 Definições da Computação Reconfigurável

A Computação Reconfigurável é uma solução intermediária entre as soluções de *hardware* e *software*, associada a uma melhor flexibilidade, generalidade, eficiência, custo e um desempenho que não chega a ser igual a de um processador de propósito geral, mas se aproxima. Outro fato importante que deve ser mencionado, é que este tipo de solução requer menos potência e consequentemente o consumo de energia é menor do que nas soluções de *hardware* fixo onde a potência é maior.

Dentre os motivos necessários para estudar a computação reconfigurável, destacam-se os seguintes:

- Demanda computacional das aplicações;
- Inadequação de alguns modelos e estilos de computação atuais em termos de desempenho e flexibilidade;

- Posição e interesse das principais empresas e universidades do mundo;
- Evolução dos dispositivos computacionais em termos de arquitetura e tecnologia.

Na Computação Reconfigurável, os agentes de *hardware* de propósito geral são configurados para executar tarefas específicas, podendo também serem reconfigurados sob demanda para realizar outras tarefas específicas em alguns casos (RECCONFIGURABLE, 2007).

A Computação Reconfigurável também pode ser considerado como um paradigma da computação, no qual circuitos integrados programáveis são utilizados juntamente com *software* para reconfigurar dinamicamente dispositivos lógicos programáveis chamados de FPGA - (*Field Programmable Gate Array*) (STAR BRIDGE SYSTEMS, 2007). Onde os FPGAs são dispositivos lógicos programáveis compostos por uma matriz de blocos lógicos, programáveis cercada de blocos de entrada e saída, e conectada por fios de interconexão também programáveis (MESQUITA, 2002).

A Computação Reconfigurável teve início na década de 60 (ESTRIN, 1960)(ESTRIN, 2002), porém apenas há pouco tempo começavam a surgir novas tecnologias que proporcionaram a sua implementação e aplicação na prática. O interesse deste novo paradigma começou no início dos anos 90 quando o número de portas lógicas dos FPGAs ultrapassou dez mil (GUCCIONE, 2001). A partir daí a Computação Reconfigurável, com a possibilidade de aumentar a velocidade de muitas aplicações, tornou-se um objeto de investigação intensiva. A característica mais importante é a capacidade de realizar computações em *hardware* com o objetivo de incrementar o desempenho, ficando ao mesmo tempo com a flexibilidade do *software* (COMPTON; HAUCK, 2002). Com a finalidade de atingir um desempenho elevado e suportar um conjunto enorme de aplicações, os sistemas reconfiguráveis são formados normalmente pela lógica reconfigurável e um processador de uso geral.

Para que se consiga executar a aplicação de uma maneira mais eficiente, aquelas partes que não são facilmente mapeadas na lógica reconfigurável são executadas em um processador hospedeiro, enquanto que as outras partes que necessita de uma computação muito intensiva e que podem se beneficiar da sua implementação de *hardware*, são executadas nos FPGAs como é mostrado na Figura 3.1 a estrutura de um sistema reconfigurável.

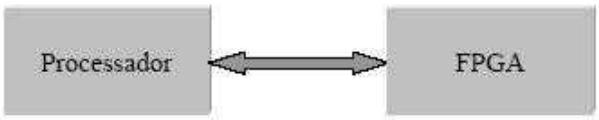

Figura 3.1: Estrutura de um Sistema Reconfigurável com Processador Hospedeiro (SKLIAROVA, 2004)

3.3 Dispositivos Lógicos Programáveis - FPGAs

Com o surgimento de novos tipos de dispositivos lógicos programáveis, os projetos de sistemas digitais, tiveram muitas alterações nas últimas décadas. Atualmente, uma quantidade considerável de sistemas digitais são implementados com o auxílio destes dispositivos com elevada densidade.

Com o crescimento atual do mercado, uma variedade enorme de dispositivos, está surgindo. Entretanto, esses dispositivos se diferem uns dos outros pelas características da tecnologia de programação e também pela granularidade dos principais elementos lógicos.

O FPGA é um dispositivo semicondutor que contém componentes básicos programáveis e interconexões programáveis. Estes componentes lógicos podem ser programados para expandir a funcionalidade de portas lógicas básicas (tais como portas AND, OR, XOR, etc.) ou funções combinatórias mais complexas, como decodificadores ou funções matemáticas.

Em muitos FPGAs, estes componentes lógicos programáveis (ou blocos lógicos, ou células) também incluem elementos de memória, que podem ser simples flip-flops ou módulos de memória completos.

Uma hierarquia de interconexões programáveis permite aos blocos lógicos de um FPGA serem interconectados pelo projetista do sistema como necessário. Estes blocos lógicos e interconexões podem ser programados de maneira tal que o FPGA pode executar qualquer função lógica (DUARTE, 2006).

3.3.1 Arquitetura de um FPGA

Um FPGA é formada por um arranjo de células configuráveis (ou blocos lógicos) contidos em um único *chip*. Cada uma dessas células, contém certa capacidade computacional para implementar funções lógicas e/ou realizar o roteamento para permitir a comunicação entre células. Essas operações podem acontecer simultaneamente no arranjo das células (BROWN; ROSE, 1996).

A arquitetura básica de um FPGA, ilustrada na Figura 3.2 (BROWN; ROSE, 1996), é composta por matrizes de blocos lógicos programáveis, cercadas de blocos de entrada e saída, sendo conectadas por fios programáveis de interconexão também programáveis (MESQUITA, 2002). Alguns FPGAs podem ter suas funções lógicas reprogramadas dinamicamente, através do envio de pacotes de *bitstream*.

A comunicação entre blocos é feita através dos recursos de interconexão. A borda externa do arranjo consiste de blocos especiais, capaz de realizar operações de entrada e saída (E/S). Atualmente, no mercado há um elevado número de tipos de arquiteturas de FPGAs. Por esse motivo, são definidos três principais aspectos para a arquitetura de uma FPGA, como:

- Tecnologia de programação;
- Arquitetura das células;
- Estrutura de roteamento.

Estes aspectos influenciam diretamente no desempenho e densidade das diferentes arquiteturas de FPGA. Entretanto, não se pode afirmar que há uma melhor arquitetura, e sim a mais adequada para uma determinada aplicação. Genericamente, uma célula de FPGA típica é composta de uma LUT - (*Look-up Table*) de 4-entradas e um *flip-flop* como é ilustrado na Figura 3.3.

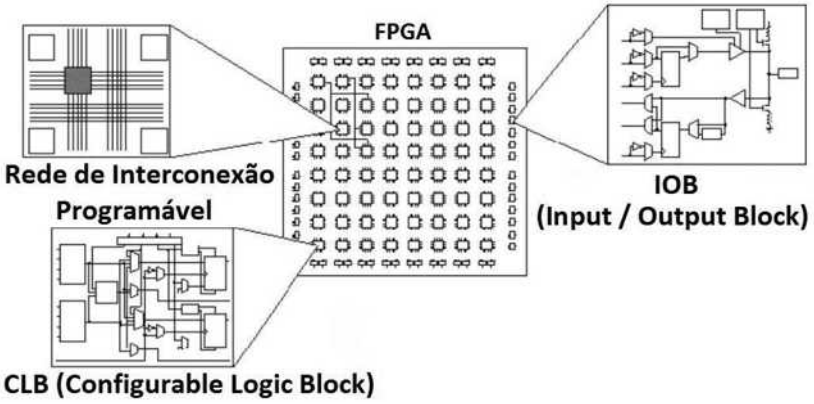

Figura 3.2: Estrutura Básica de um FPGA
(BROWN; ROSE, 1996)

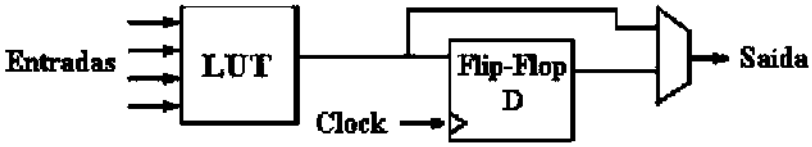

Figura 3.3: Célula Genérica da Maioria dos FPGAs
(BROWN; ROSE, 1996)

No caso da célula da Figura 3.3, existe apenas uma saída, que pode ser a saída direta da *LUT* ou a saída vinda do registrador. A célula possui 4 entradas para a LUT e uma entrada de *clock*. A disposição das entradas e da saída da célula é ilustrada na Figura 3.4.

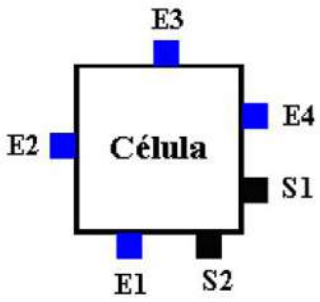

Figura 3.4: Localização das Entradas e Saídas de uma Célula
(BROWN; ROSE, 1996)

Cada entrada é acessível de um lado da célula, enquanto que o pino de saída pode se conectar aos fios de roteamento em ambos os canais, à direita e abaixo da célula. Cada pino de saída da célula por sua vez pode conectar a qualquer um dos segmentos dos canais adjacentes a ele. A Figura 3.5 ilustra esta situação.

A arquitetura genérica de um FPGA consiste em uma cadeia de células e canais de roteamento, conforme mostra a Figura 3.6. Geralmente, todos os canais de roteamento

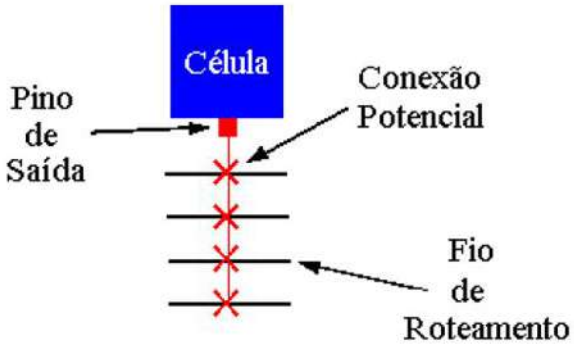

Figura 3.5: Ligação do Pino de Saída aos Segmentos do Canal de Roteamento
(BROWN; ROSE, 1996)

possuem o mesmo comprimento, isto é, números de fios. Algumas arquiteturas possuem além da rede de roteamento, linhas globais que provêm conexões de alta velocidade, com poucos desvios, para todas as células da matriz da FPGA, suprindo sinais como clock, resets e outros tipos de sinais globais.

A implementação da célula ilustrada na Figura 3.6 não é única. Existem várias outras propostas de arquitetura, onde as células implementam lógicas de maior complexidade, tais como unidades lógicas aritméticas (ULAs), ou até mesmo núcleos de processamento completos como microprocessadores com módulos de memórias. Esta diferença no tamanho da célula é usualmente referida como granularidade.

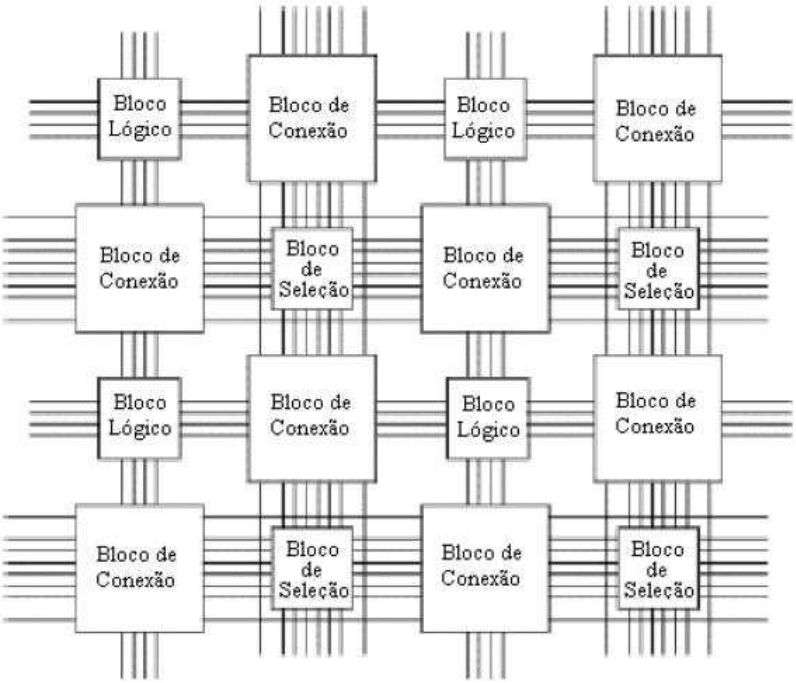

Figura 3.6: Estrutura Genérica de um FPGA
(COMPTON; HAUCK, 2002)

3.3.2 Vantagens e Desvantagens dos FPGAs

Vantagens:

- Ter uma arquitetura flexível que possibilita a utilização em muitas aplicações;
- Conseguem unificar várias estruturas heterogêneas como: blocos de memória, que permite a implementação de sistemas completos em um único encapsulamento;
- O consumo de energia é uma vantagem muito importante em alguns sistemas. Os FPGAs são dispositivos baseados em PROM, EPROM e SRAM onde requerem menos potência que os dispositivos baseados em EEPROM e FLASH, que necessitam de maior potência e com isso ocorre um consumo maior.

Desvantagens:

- Tempo de configuração: é o tempo para carregar uma configuração no FPGA, sendo um fator que pode ser um impecílho para sua utilização em sistemas dinamicamente reconfiguráveis;
- Compatibilidade em relação a novos dispositivos: para obter benefícios de uma nova família de FPGAs, os sistemas desenvolvidos devem ser resintetizados e muitas vezes até mesmo reescritos;
- Restrições de tamanho: atualmente os FPGAs disponíveis no mercado somente permitem que sejam implementados núcleos de tamanho limitado e consequentemente as funcionalidades disponíveis nos dispositivos se tornam pequenas;
- Tempo de compilação: no caso as tarefas de síntese, roteamento e posicionamento consomem muito mais tempo que a compilação de *software* dos processadores de propósito geral (MESQUITA, 2002).

3.3.3 Classificação dos FPGAs

Atualmente, os FPGAs apresentam um número considerável de arquiteturas, tendo cada fabricante implementado um modelo próprio. Existe, no entanto, vários pontos comuns entre as diversas arquiteturas, sendo por isso possível agrupá-las tendo em consideração certas características.

Uma das possibilidades de classificação dos FPGAs, é apresentado na Figura 3.7, com relação aos modos de configuração que pode ser realizado neste tipo de arquitetura. A classificação pode ser configurável ou reconfigurável, reconfigurável estático ou dinâmico (SANCHEZ et al., 1999) e o dinâmico pode ser parcial ou total (GERICOTA et al., 2003).

A Reconfiguração Estática: o funcionamento do FPGA ocorre permanentemente, após ser configurada conforme Figura 3.8. Certamente, o modo estático não proporciona grande flexibilidade, mas permite um bom desempenho através da utilização do *hardware* otimizado para qualquer aplicação (SKLIAROVA; FERRARI, 2001).

A Reconfiguração Dinâmica: algumas vezes é necessário ativar configurações diferentes de acordo com a necessidade de uma aplicação. Certamente isto acarretará mais seções de aplicação a serem mapeadas em *hardware* do que poderá caber em uma FPGA. Contudo, grande parte da aplicação pode ser acelerada em um sistema reconfigurável, conseguindo assim um desempenho mais alto.

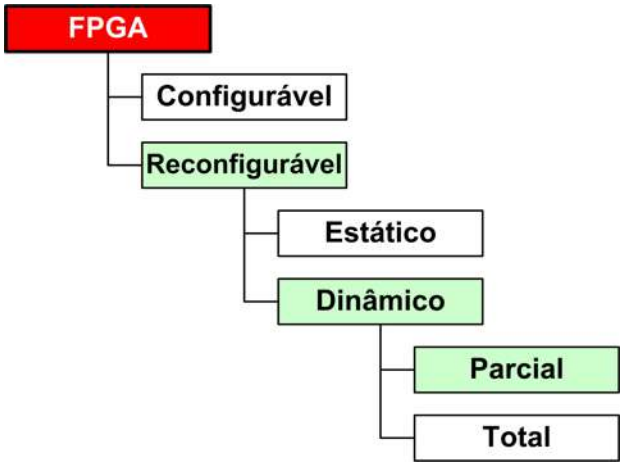

Figura 3.7: Classificação das FPGAs em Função dos Modos de Configuração
(GERICOTA et al., 2003)

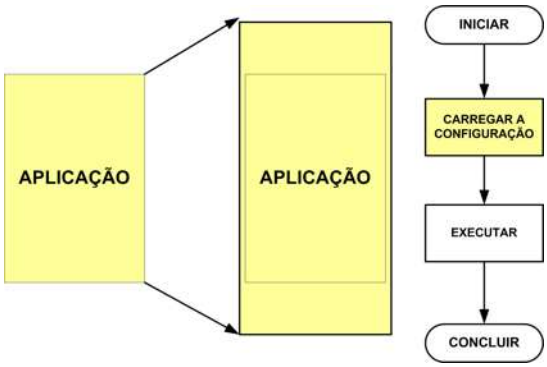

Figura 3.8: Reconfiguração Estática
(SKLIAROVA; FERRARI, 2001)

Este termo é chamado de reconfiguração dinâmica a que está agregada a noção de *hardware* virtual. Neste caso, os recursos disponíveis no FPGA são bastante menores que o conjunto de recursos necessários para todas as configurações. Mas em vez de reduzir o número de configurações mapeadas, estas são transferidas para o FPGA de acordo com a necessidade atual (COMPTON; HAUCK, 2002).

Uma das vantagens dos sistemas reconfiguráveis dinamicamente comparando-os com os sistemas programáveis só no início da aplicação, é a potencialidade de efetuar otimizações do *hardware* com base na informação determinada durante a execução.

Outros termos atualmente encontrados na literatura que se referem à Reconfiguração Dinâmica são *Run-Time Reconfiguration - RTR*, *Real-Time Reconfiguration*, *On-The-Fly Reconfiguration* e *In-Circuit Reconfiguration*. Todas essas expressões podem ser traduzidas também como reconfiguração em tempo de execução (LYSAGHT; DUNLOP, 1993).

A **Reconfiguração Dinâmica** pode ser dividida em: Reconfiguração Global e Reconfiguração Parcial.

A **Reconfiguração Global** é quando uma reserva de todos os recursos de *hardware* é feita para cada fase de execução. Depois que uma fase é concluída, todos os recursos da FPGA são reconfigurados para uma próxima fase ilustrado na Figura 3.9.

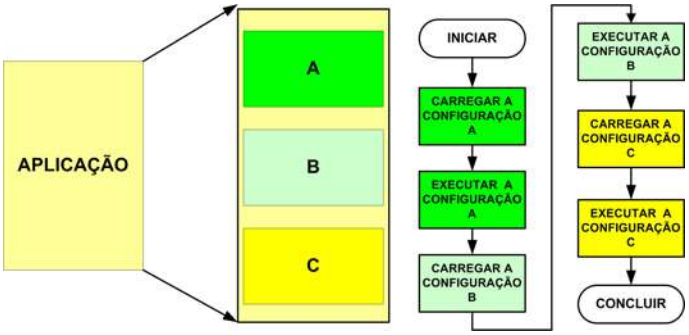

Figura 3.9: Reconfiguração Dinâmica Global
(SKLIAROVA; FERRARI, 2001)

Os FPGAs são classificados como dinamicamente reconfiguráveis, se seus circuitos internos de armazenamento da configuração podem ser atualizados seletivamente, sem prejudicar o funcionamento da lógica restante que pode estar em operação. Estes dispositivos podem assim ser reconfigurados seletivamente enquanto estiverem ativos.

A **Reconfiguração Parcial** é quando acontece a modificação seletiva dos recursos de *hardware* ao longo da execução de uma aplicação, ilustrada na Figura 3.10. Esta flexibilidade permite que o *hardware* seja mais personalizado com relação às necessidades correntes da aplicação. Esta reconfiguração exige que só os recursos selecionados sejam reprogramados, o que resulta em um *overhead* de configuração menor do que no caso anterior.

Um dispositivo é definido como reconfigurável parcialmente se é possível reconfigurá-lo seletivamente, enquanto o resto do dispositivo permanece inativo, mas retém sua configuração. Caso o dispositivo só possa ser reconfigurado por completo, mesmo que na própria placa do sistema, ou fora dela como com dispositivos EPROMs, ele é considerado apenas reconfigurável, pois, não permanece ativo em nenhum dos casos, por não permitir configuração seletiva (LYSAGHT; DUNLOP, 1993).

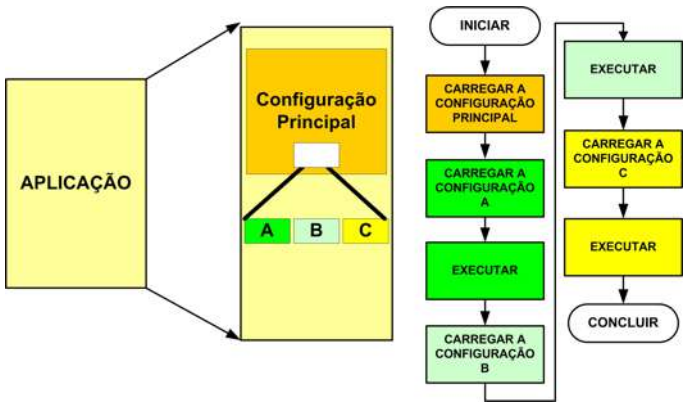

Figura 3.10: Reconfiguração Dinâmica Parcial
(SKLIAROVA; FERRARI, 2001)

3.3.3.1 Tipos de FPGA com Reconfiguração Dinâmica

Existem três tipos de dispositivos com reconfiguração dinâmica (COMPTON; HAUCK, 2002):

- Dispositivos de contexto único: necessitam de uma reconfiguração completa para alterar qualquer um dos bits de programação, como ilustrado na Figura 3.11A. A maioria das FPGAs disponíveis no mercado pertencem a esta classe. Nota-se que existe uma técnica especial que permite a reprogramação parcial deste grupo de FPGAs. Estes baseia-se em modelos (*hardware* templates);

- Dispositivos de contexto múltiplo: possuem várias camadas de *bits* de programação, estando em cada instante ativa apenas uma camada, ilustrado na Figura 3.11B. A vantagem principal destes dispositivos é a possibilidade de mudar de contexto rapidamente. Além disso, é permitida a configuração nos bastidores *background* possibilitando reprogramar um contexto enquanto um outro contexto está ativo;

- Dispositivos parcialmente reconfiguráveis: permitem que as áreas pequenas da FPGA sejam modificadas sem necessidade de reprogramar todo o dispositivo, apresentado na Figura 3.11C. As reconfigurações parciais requerem um tempo menor do que a reconfiguração total. Frequentemente, as zonas da FPGA que não são alteradas poderão continuar a execução permitindo que a reconfiguração seja concretizada simultaneamente com a execução (COMPTON; HAUCK, 2002).

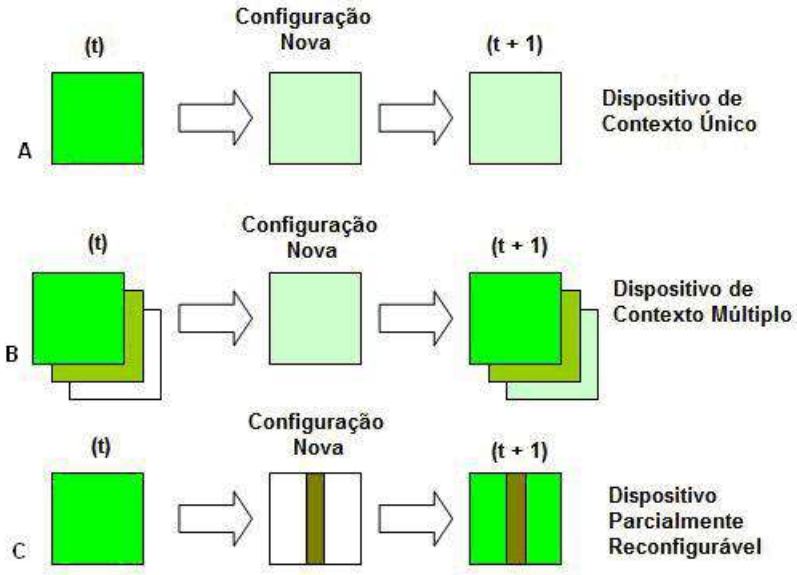

Figura 3.11: Vários Tipos de Dispositivos Reconfiguráveis
(SKLIAROVA; FERRARI, 2003)

Uma característica importante que se deve ter em relação aos sistemas reconfiguráveis dinamicamente, é que o tempo usado ou gasto para realizar uma reconfiguração não deve superar a aceleração ganha. Devido a isto, são frequentemente aplicadas várias técnicas destinadas a diminuir o *overhead* da reconfiguração (COMPTON; HAUCK, 2002) (KENNEDY, 2003). As técnicas podem ser: utilização de métodos de compressão de dados de configuração e carregamento dos arquivos de configuração em *background*, enquanto o processador principal está executando uma outra parte da aplicação.

Exemplos destes dispositivos dinamicamente reconfiguráveis são os da família Virtex da *Xilinx*, AT6000 e AT40K da Atmel.

Estes FPGAs possuem características comuns: eles são formados por arranjos de blocos lógicos, com granularidade fina e média, cuja configuração é controlada por escrita de dados em uma memória estática. Para um FPGA ser reconfigurável dinamicamente, a definição implica que ele deve ser capaz de ser reconfigurado parcialmente enquanto ativo, isto é, energizado e em operação. Em nível de sistema, um módulo que contenha múltiplos FPGAs reconfiguráveis, total ou parcialmente, pode ser classificado como reconfigurável dinamicamente se os componentes FPGAs são reconfigurados individualmente.

Como os tempos de configuração não são desprezíveis, a habilidade de intercalar execução e reconfiguração, sem prejuízo do desempenho é uma questão que ainda merece atenção e esforços de pesquisa. Um sistema de reconfiguração dinâmica inclui pelo menos uma área de reconfiguração onde blocos lógicos podem ser carregados em tempo de execução.

A Figura 3.12 ilustra a reconfiguração dinâmica de um sistema composto de cinco circuitos ou tarefas. A tarefa A é uma função de entrada que alimenta as tarefas B, C ou D, que alimentam a tarefa E que é a função de saída do sistema. As tarefas de entrada e saída são permanentemente residentes no FPGA enquanto as três dinâmicas alternam-se sob o controle de um sinal de *Swap* (LYSAGHT; DUNLOP, 1993).

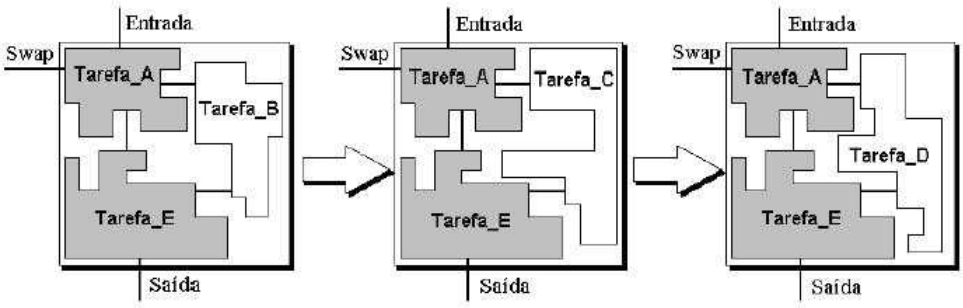

Figura 3.12: Exemplo de Reconfiguração Dinâmica
(LYSAGHT; DUNLOP, 1993)

A capacidade de reconfiguração dinâmica permite o compartilhamento ao longo do tempo de tarefas diferentes, o que pode reduzir significativamente a área de silício exigida. Esta tecnologia torna possível o conceito de *hardware* ilimitado ou *Hardware* Virtual (CARDOSO; VESTIAS, 2007). Este conceito é similar ao de memória virtual e está sendo adotado para permitir que múltiplos conjuntos de configuração sejam armazenados para serem usados pelas *Reconfigurable Processing Units* - RPUs, quando necessário. O *hardware* virtual é implementado por compartilhamento de tempo de uma dada RPU, necessitando que um escalonador seja responsável pelas configurações, execuções e comunicações entre as partições temporais.

Com o surgimento de dispositivos que suportam reconfiguração dinâmica total ou parcial como a família VIRTEX (XILINX, 2007), surge assim uma nova geração de dispositivos SoCs, baseadas em FPGAs, já anunciada pela própria família Virtex II. Diferentemente dos dispositivos SoCs, FIPSoC da SIDSA e o FPSLIC da Atmel, que possuíam FPGAs como blocos de seus sistemas. A família Virtex II Pro fornece vários elementos de granularidade grossa distribuídos pelo dispositivo, indo além dos elementos básicos

comumente encontrados em arquiteturas FPGAs, mostrado na Figura 3.13. Porém, nem por isso estes dispositivos deixam de ser FPGAs, com os tradicionais blocos lógicos, chamadas de CLBs por este fabricante, caracterizando uma arquitetura híbrida de granularidade média com elementos de granularidade grossa. Entretanto, a Xilinx não se limita a chamar este dispositivo de FPGA, mas também de Plataforma FPGA.

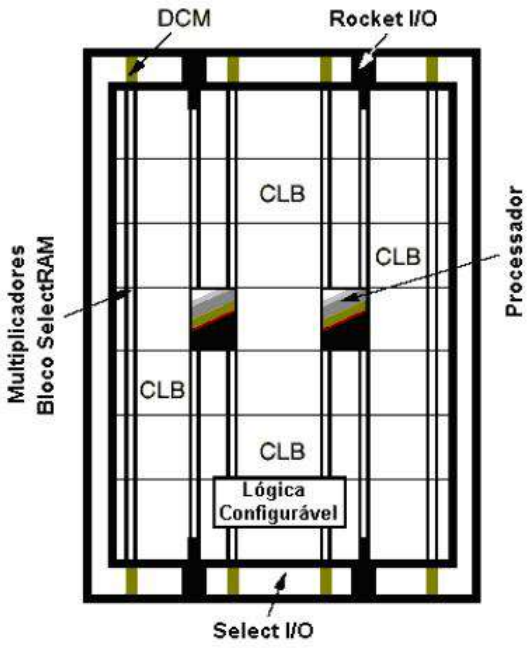

Figura 3.13: Arquitetura do FPGA Virtex-II Pro
(XILINX, 2007)

A Figura 3.13 ilustra alguns destes elementos da arquitetura Virtex II Pro (XILINX, 2007). Nesta arquitetura, além dos elementos comuns à família Virtex II como os CLBs, IOBs, blocos SelectRAM, Multiplicadores e DCMs, existem outros elementos como: Rocket Input/Output - I/O e Processador PowerPC.

3.3.4 Aplicações dos FPGAs

Os dispositivos lógicos programáveis os FPGAs são dispositivos que podem ser utilizados em:

- Controladores digitais;
- Codificadores de comunicação;
- Filtros;
- Aplicadas em grandes sistemas através de vários FPGAs interconectadas;
- Em máquinas customizadas - FPGAs executando *software* ao invés de compilar o *software* para executar na CPU;
- Em dispositivos de auto-atendimento como máquinas de refrigerantes;

- Em dispositivos de transações financeiras, como por exemplo, os terminais bancários, das instituições financeiras;
- Na Robótica, no desenvolvimento de robôs.

3.4 Conclusão

A Computação Reconfigurável é um paradigma que surge fortemente no mercado mundial. A utilização de arquiteturas reconfiguráveis como os FPGAs, são estruturas que possibilitam grande integração de circuitos lógicos com pequeno consumo de energia. As implementações que anteriormente se utilizavam de diversos tipos de circuitos integrados, em grandes placas com elevado consumo de energia podem ser substituídas por um único circuito integrado que responde por um sistema digital completo reprogramável. Inúmeras vantagens podem ser observadas como: flexibilidade, custo e generalidade.

4 SERVIÇO DE ATUALIZAÇÃO DE CONFIGURAÇÕES DE *HARDWARE* - SACH

Atualmente, o grande problema que se verifica em relação as arquiteturas reconfiguráveis do tipo FPGA, é a forma como é executada a distribuição de pacotes de recofiguração aos dispositivos lógicos programáveis. A forma de distribuição de pacotes ocorre individualmente, tornando a tarefa muito onerosa. Esta situação se potencializa em um ambiente pervasivo onde poderá existir milhares de dispositivos instalados na rede. A partir deste momento, surgiram três motivações centrais para este trabalho de dissertação que foram: explorar a capacidade das arquiteturas reconfiguráveis para solucionar as demandas introduzidas pela Computação Pervasiva; desenvolver uma proposta que permitisse a reprogramação de dispositivos reconfiguráveis - FPGAs de forma distribuída e concorrente, sem que houvesse a intervenção direta do ser humano e dotar o *middleware* EXEHDA de um novo Serviço.

O **Serviço de Atualização de Configurações de *Hardware*** tem por objetivo propagar pacotes de *bitstream* para reconfiguração dinâmica parcial de dispositivos reconfiguráveis, dentro de um ambiente pervasivo. Esta reconfiguração de pacotes para os FPGAs acontece de maneira dinâmica e parcial, distribuindo pacotes para vários tipos de dispositivos lógicos concorrentemente, e sem a intervenção direta do operador. Entende-se por reconfiguração dinâmica parcial aquela que ocorre quando algumas funções lógicas dos FPGAs são alteradas dinamicamente, sem ter a necessidade de desativar o dispositivo. Estas reconfigurações são possíveis apenas sobre arquiteturas reconfiguráveis que suportem este tipo de reconfiguração, como por exemplo, o VIRTEX II-PRO (XILINX, 2007).

Este capítulo apresenta a concepção do SACH, sendo discutidas a sua modelagem, apresentando as visões de arquitetura do serviço, caso de uso, implementação e estratégias adotadas para o desenvolvimento do protótipo SACH. Os pressupostos definidos nos capítulos de fundamentação integraram as decisões nas diferentes etapas do trabalho.

4.1 Modelagem do Serviço de Atualização de Configurações de *Hardware* - SACH

As descrições em UML, utilizada ao longo deste capítulo, tem por finalidade apresentar a modelagem do SACH, caracterizando as funcionalidades que o serviço dispõe e a relação entre as mesmas.

Esta seção apresenta o SACH modelado em UML. São utilizadas quatro visões

para modelar o serviço, (GUEDES, 2005):

- Visão da arquitetura do SACH;
- Visão de caso de uso: atores e atividades do SACH;
- Visão da implementação do protótipo SACH;
- Visão das estratégias da instalação do SACH.

4.1.1 Visão da Arquitetura do SACH

Na arquitetura do SACH, ilustrada na Figura 4.1, é possível identificar os seus serviços básicos. A arquitetura está dividida em duas grandes partes: Cliente e Servidor.

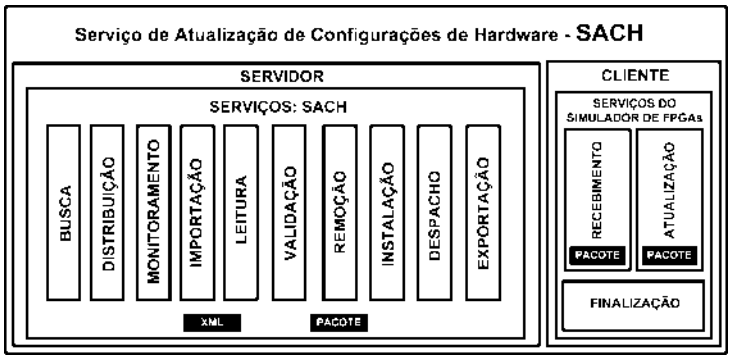

Figura 4.1: Visão da Arquitetura do Sistema SACH

O Servidor é formado por dez serviços:

1. Busca - este serviço tem a função de localizar e buscar pacotes entre as células;
2. Distribuição - é responsável pela propagação de pacotes entre as células;
3. Monitoramento - o serviço verifica a entrada de pacote na pasta TMP da célula;
4. Importação - tem a função de importar os arquivos pacote e descritor da TMP da célula, renomeando os mesmos;
5. Leitura - tem o objetivo de executar a leitura dos dados dos arquivos descritores tanto dos pacotes como dos dispositivos da célula;
6. Validação - valida as informações armazenadas do arquivo descritor do pacote, para que ele possa ser posteriormente distribuído aos dispositivos que necessitam ser reconfigurados na célula;
7. Remoção - é o serviço que tem a função de excluir tanto o pacote como seu descritor, mais antigo no RCS da célula;
8. Instalação - tem a função de instalar o pacote e seu arquivo descritor no RCS da célula;

9. Despacho - o objetivo é disparar o serviço de exportação de pacotes para os dispositivos da célula;

10. Exportação - executa a propagação de pacote de reconfiguração para os FPGAs identificados através do serviço de validação, enviando o pacote para os dispositivos que necessitam ser reconfigurados.

O Cliente é composto de três serviços:

1. Recebimento - tem a função de receber o pacote de reconfiguração;

2. Atualização - é responsável pela atualização do dispositivo, ou seja, executa a reconfiguração do mesmo. Este serviço é dependente do modelo de FPGA utilizado, sua implementação na versão atual do SACH é minimalista, destinada a dar suporte somente aos testes que dizem respeito aos objetivos centrais desta dissertação;

3. Finalização - este serviço tem como finalidade executar a finalização do processo de reconfiguração, após a execução da atualização do dispositivo pelo serviço de atualização. O término da operação é verificada pelo gerente, ao receber por meio da interface gráfica do SACH, uma mensagem que o processo de reconfiguração dos dispositivos foram realizados com sucesso.

Descritores Utilizados na Gerência de Dependências para Validação de Pacotes

Um dos desafios centrais no gerenciamento de pacotes de atualização são os problemas de dependências e conflitos que podem ocorrer entre os mesmos. Em muitos sistemas de atualização, como por exemplo o apt-get no *Linux* (HARDWARE.NET, 2007), onde as dependências entre pacotes são complexas, pelo motivo que muitas das vezes se relacionam não somente com os pacotes propriamente ditos, como também com suas versões.

Este problema de dependências e conflitos entre pacotes é gerenciado pelo SACH através do processamento de arquivos descritores dos pacotes de reconfiguração de *hardware* (sach.xml). O gerenciamento acontece através de validações executadas sobre os arquivos descritores do pacote que está sendo inserido no serviço, considerando os arquivos descritores dos pacotes que já se encontram instalados no Repositório Celular do Serviço - (RCS).

Estas comparações são feitas através dos metadados que estão armazenados nos arquivos descritores. Por sua vez, os pacotes de reconfiguração, propriamente ditos, são formados por um conjunto de *strings* que constituem atualizações das funcionalidade lógicas dos dispositivos. Outro arquivo descritor utilizado é o fpga.xml, referente aos dispositivos reconfiguráveis. Cada célula tem um arquivo descritor deste tipo que apresenta as informações dos dispositivos instalados nela.

A seguir, são apresentadas as informações que são armazenadas nos arquivos descritores XML correspondentes aos pacotes e aos dispositivos de borda - FPGAs, ilustrado na Figura 4.2.

Figura 4.2: Arquivos Descritores - XML

O serviço executa a reconfiguração dos FPGAs, a partir da ocorrência de dois eventos, como ilustrado na Figura 4.3:

1. Evento 1 - no momento em que são feitas atualizações de pacotes de reconfiguração de dispositivos do tipo FPGAs que estão armazenados no Repositório Celular do Serviço - (RCS) de cada célula, isto é, pacotes com versões mais atualizadas em relação aqueles que estão armazenados nos repositórios de cada célula, para posteriormente serem distribuídos para cada dispositivo da célula compatível com este pacote que está sendo inserido;

2. Evento 2 - e no momento em que ocorre a substituição de um ou mais dispositivos, por motivo de defeito ou pelo acréscimo de novos dispositivos lógicos na célula. O processo de configuração ocorre no momento em que o novo dispositivo é instalado na célula. Caso o operador não forneça o pacote, o SACH busca inicialmente um pacote compatível para este novo dispositivo no RCS da própria célula origem. Ao ser verificado pelo serviço que não existe o pacote, é executada uma próxima busca entre as células mais próximas ou vizinhas até encontrar o mesmo. Não sendo encontrado o pacote para este dispositivo, o gerente é informado pelo serviço que não existe o mesmo entre as células. Logo após, o gerente solicita ao operador o referido pacote.

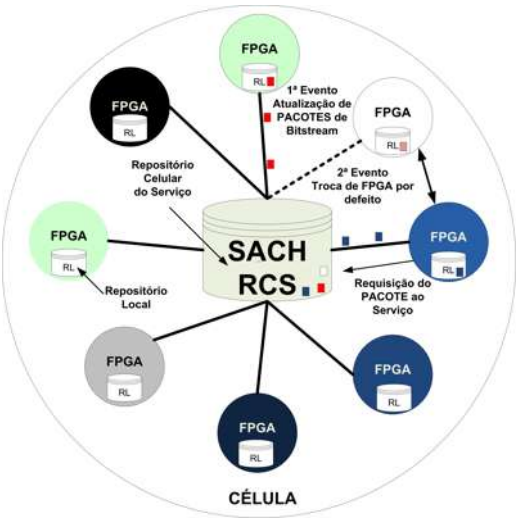

Figura 4.3: Operações do SACH

.

4.1.2 Visão de Caso de Uso: Atores e Atividades do Serviço

Esta sub-seção apresentam as seguintes visões: do caso de uso do SACH, da atualização de pacotes de *bitstream* e dos diagramas de atividades do SACH: do operador, do gerente, da interface, da busca e distribuição de pacotes, das atividades e validações do SACH e do simulador.

4.1.2.1 Caso do Uso do SACH

A seguir, está descrita em linhas gerais o Caso de Uso do SACH, sendo feito destaque para os seguintes aspectos:

- Quais atores interagem com o serviço;
- Quais etapas devem ser executadas pelo ator e pelo SACH;
- Para que o caso de uso executa sua função;
- Quais os parâmetros devem ser fornecidos;
- Quais restrições e validações o caso de uso deve possuir.

A seguir é feita a descrição das funções que o SACH contempla, bem como os respectivos diagramas de atividades:

Caso de Uso: Atualizar Pacotes de *Bitstream*

- Propósito: efetuar a atualizações de pacotes de configuração de *hardware*;
- Atores: operador e gerente;
- Descrição: atualizações de pacotes de *bitstream* para reconfiguração de FPGAs;
- Referências: serviço de atualização de pacotes de *bitstream*.

A Figura 4.4 apresenta as ações dos atores que utilizam o Serviço de Atualização de Configurações de *Hardware* - SACH, com suas atividades dentro do serviço.

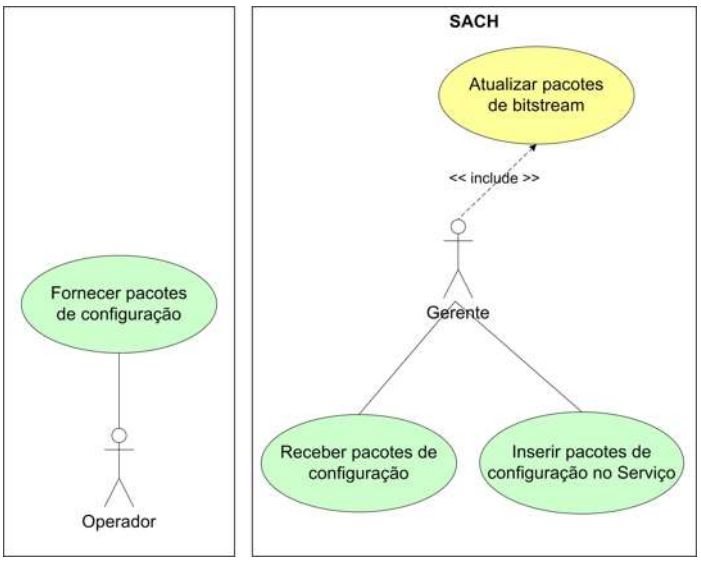

Figura 4.4: Atores do Sistema SACH

Operador: é o responsável pelo fornecimento dos pacotes de *bitstream*. Estes pacotes são constituídos por uma seqüência binária de bits, que são utilizados para reconfigurar os dispositivos de borda - FPGAs. Este procedimento é externo em relação à operação de atualização, e não corresponde a uma ação direta no serviço como é o caso das operações envolvendo o gerente.

Gerente: tem a função de gerenciar o SACH, recebendo o pacote de reconfiguração de *hardware* do operador, inserindo o mesmo no serviço através de uma interface de fácil operação.

4.1.2.2 Diagramas de Atividades do SACH

A seguir são ilustrados os diagramas de atividades referentes ao SACH, seus atores e o simulador de dispositivos, representados pelas Figuras 4.5 a 4.11. Estes diagramas caracterizam as relações do SACH com cada Ator, bem como os relacionamentos das diversas funcionalidades do serviço. A interligação entre os diagramas está indicada por conectores numerados, os quais apontam os pontos de entrada e saída (E/S).

Operador

O diagrama da Figura 4.5 mostra a função do ator Operador em relação ao SACH. Ele não atua diretamente no serviço e sim indiretamente, fornecendo apenas os pacotes de reconfiguração e seus arquivos descritores para o Gerente.

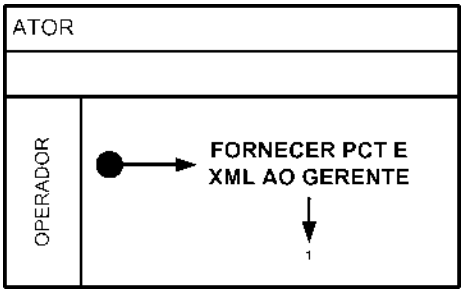

Figura 4.5: Diagrama de Atividades do Operador

Gerente

O gerente tem como função central receber pacotes de recofiguração e coordenar as funcionalidades do SACH nas questões de: inserção de pacotes e seus arquivos descritores, como também, de inserção, edição e exclusão de dispositivos na célula, ilustrado no Diagrama da Figura 4.6. Toda esta operacionalidade é proporcionada pelo SACH, onde o gerente poderá reconfigurar dispositivos.

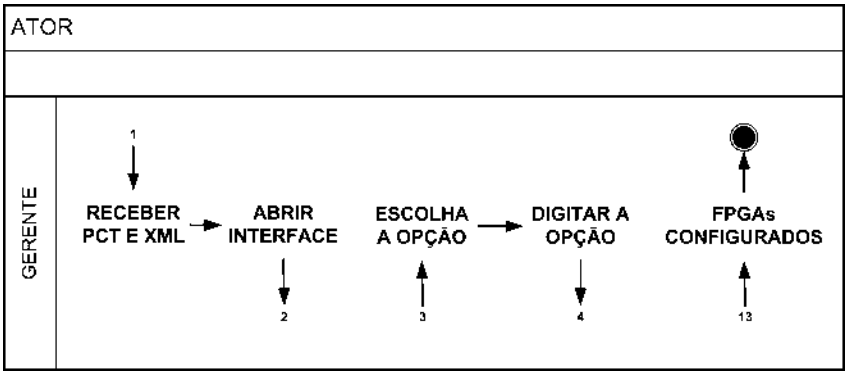

Figura 4.6: Diagrama de Atividades do Gerente

Interface

No diagrama de atividades da Interface vide Figura 4.7, foi desenvolvido para facilitar ao usuário, a questão da utilização do SACH. É constituída por um menu dividido em 3 partes, cujos os ítens estão descritos a seguir:

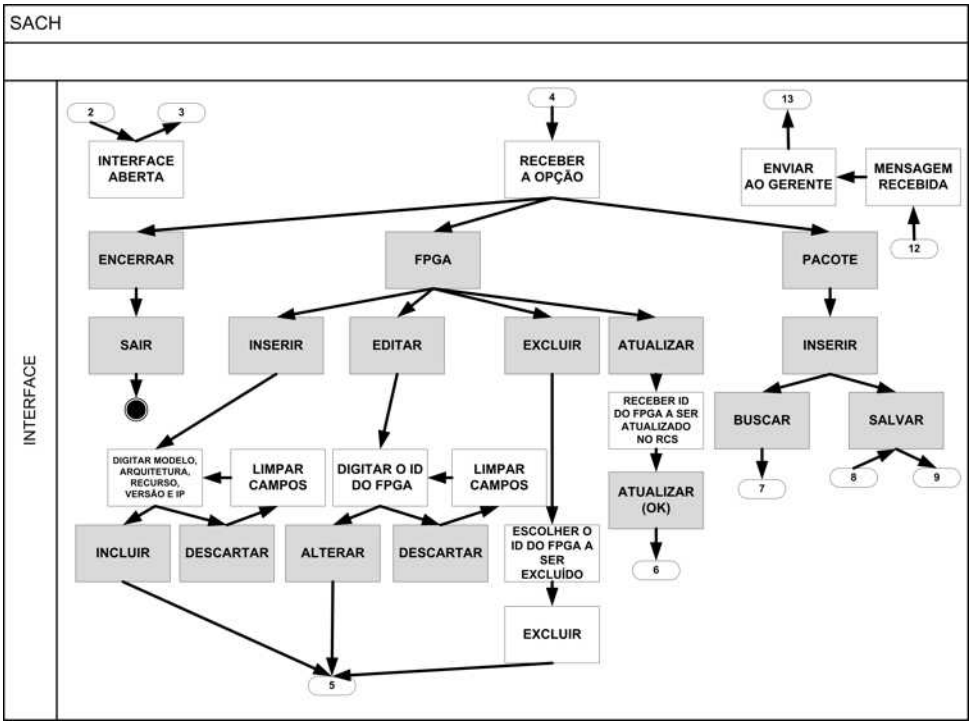

Figura 4.7: Diagrama de Atividades da Interface do Serviço

1. ENCERRAR - o gerente ao escolher esta opção, a interface apresenta o sub-item sair que, ao ser executado, tem a função de finalizar o SACH;

2. FPGA - neste item acionado pelo gerente, há quatro sub-ítens que manipulam as informações dos arquivos descritores dos dispositivos instalados na célula e também com relação aos novos dispositivos que futuramente poderão ser instalados. Os sub-ítens com relação ao FPGA, são:

(a) Inserir: este sub-item tem a função de inserir um novo dispositivo na célula. As informações que são apresentadas e que devem ser preenchidas, são as seguintes: IP da célula - identificando a localização do FPGA, em que célula ele se encontra instalado; modelo do FPGA; arquitetura do FPGA; recursos do FPGA; e versão do pacote. Após o preenchimento dos campos, o gerente aciona o botão inserir que tem a função de gravar as informações do novo dispositivo no arquivo descritor fpga.xml da célula. Caso contrário, se o gerente ainda não pressionou ele poderá pressionar o botão descartar que tem a função de limpar os dados digitados por ele na tela;

(b) Editar: este sub-item tem a função de proporcionar modificações nas informações do arquivo descritor do dispositivo lógico. Apresenta uma tela informando uma lista de IDs referentes aos FPGAs instalados na célula. O gerente ao proceder a escolha do dispositivo através do seu ID, o SACH apresenta na tela as informações com relação a esse dispositivo instalado na célula como: IP da célula onde o FPGA se encontra instalado, modelo do FPGA, arquitetura do FPGA, recursos do FPGA e versão do pacote. A partir deste momento, o gerente poderá alterar os dados do seu descritor e a seguir para confirmar as alterações realizadas clica no botão aplicar. Caso ele resolva manter as informações anteriores do dispositivo, o gerente deverá pressionar o botão descartar antes de aplicar, mantendo as informações anteriores do arquivo descritor;

(c) Excluir: este sub-item tem a função de apagar as informações do dispositivo instalado na célula. A tela utilizada é a mesma usada no sub-item editar, com os mesmos campos. Ao ser escolhido o ID do dispositivo que será excluído da célula, o serviço coloca na tela as informações com relação a este dispositivo. Ao ser apresentado o gerente pressiona o botão aplicar que excluirá os dados do arquivo descritor do dispositivo no arquivo descritor (fpga.xml) e o dispositivo poderá ser retirado de operação. Esta retirada pode acontecer por defeito dispositivo ou pela atualização do próprio dispositivo pelo fabricante, caso for necessário;

(d) Atualizar: este sub-item é utilizado quando, anteriormente, é inserido um novo FPGA ou pela alteração das informações do arquivo descritor de um dispositivo que já se encontra instalado na célula. Quando as informações são alteradas, o dispositivo devará sofrer uma nova reconfiguração. É apresentada uma tela com uma lista de IDs dos dispositivos instalados na célula origem, onde o gerente, escolhe o dispositivo que deverá ser atualizado. Após escolha, o gerente pressiona o botão atualizar, onde o serviço de busca do SACH é disparado, procurando o pacote inicialmente no RCS da célula origem caso não encontre, a busca de pacote é realizada entre as células mais próximas ou vizinhas. Caso encontre o pacote, é feito um *download* do mesmo da célula origem para as devidas validações necessárias até realizar a reconfiguração. Caso não encontre o pacote desejado entre as células, o gerente é informado.

3. PACOTE - quando esta opção do menu é acionada, apresenta um sub-item chamado de Inserir pacotes. Isto significa que o gerente irá inserir um novo pacote de uma célula através da interface do SACH. Uma tela com dois botões é apresentada pelo serviço: busca - que procura o arquivo descritor do novo pacote e salvar que grava

o arquivo descritor com o objeto pacote da célula origem para posteriormente ser distribuído entre as células vizinhas através do serviço de distribuição de pacotes.

Serviço de Busca de Pacotes

Já o diagrama de atividades do Serviço de Busca de Pacotes, representado na Figura 4.8, é composto por um mecanismo que tem a função de buscar pacotes entre as células. Cada célula, tem uma lista de células vizinhas cadastradas e gerenciadas pelo gerente no SACH que utiliza a busca de pacotes entre células que compõem esta lista. Isto ocorre quando um novo dispositivo é inserido na célula ou quando são modificadas as informações do seu arquivo descritor. Logo após, o arquivo descritor dos dispositivos da célula tendo suas informações alteradas, o gerente escolhe o FPGA que deve ser reconfigurado, através de uma lista de IDs apresentado pelo SACH. Ao escolher o FPGA, o gerente tem a opção de atualizar o dispositivo acionando o sub-item Atualizar, o qual funciona da seguinte forma:

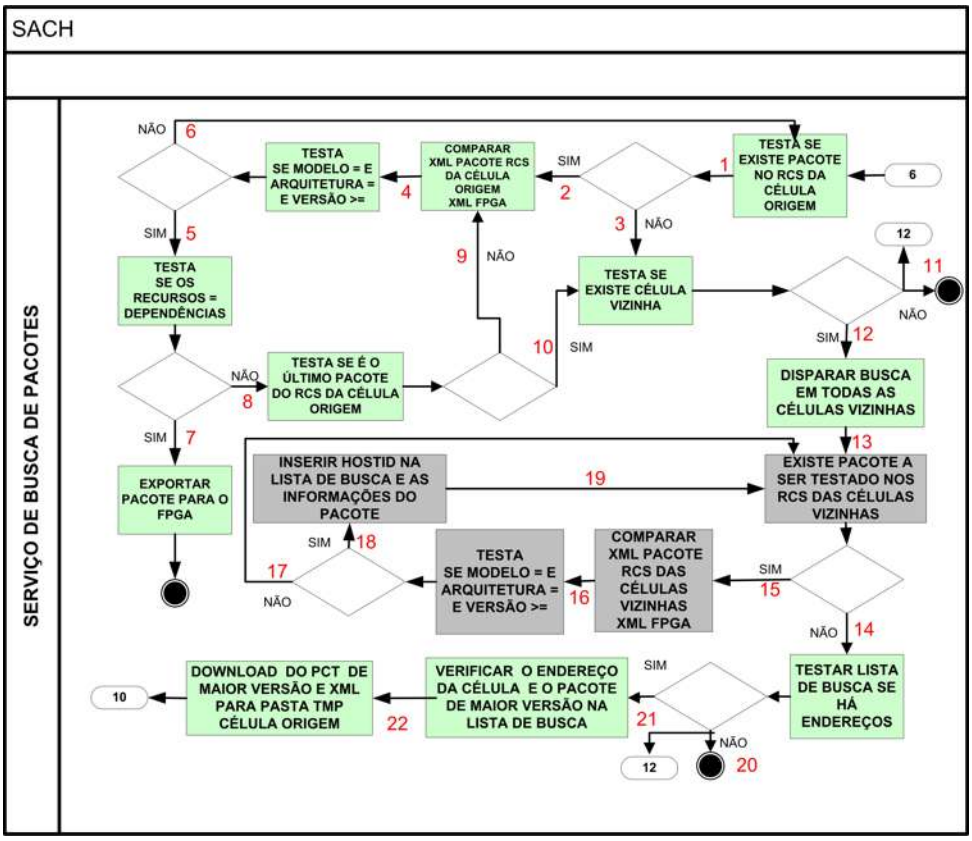

Figura 4.8: Diagrama de Atividades do Serviço de Busca de Pacotes

1. O serviço de busca testa inicialmente se existe pacotes no RCS da célula origem;

2. Se existem pacotes, são executadas as validações necessárias entre as informações contidas nos arquivos descritores dos pacotes do RCS da célula origem em relação ao novo FPGA ou aquele que foi modificado na célula;

3. Se o resultado for negativo, não existem pacotes no RCS da célula origem para este dispositivo. Verifica-se a existência de células vizinhas que possam fornecer o pacote desejado;

4. O teste de validação do item 2 é feito em relação as informações dos arquivos descritores definidos como: modelos e arquiteturas do FPGA que devem ser iguais e a versão do pacote igual ou superior em relação ao dispositivo que se quer reconfigurar;

5. Caso o teste do item 4 for positivo, os modelos e arquiteturas são iguais e se a versão do pacote do RCS for mais recente, é executado outro teste de validação das dependências dos pacotes em relação aos recursos do FPGA;

6. Caso o teste do item 4 resultar negativo, é executado novamente a validação do próximo pacote do RCS da célula origem, indo para o item 2;

7. Se o teste do item 5 for positivo, ou seja, as dependências do pacote são iguais aos recursos do FPGA é executado a exportação do pacote para o FPGA, através do serviço de exportação, finalizando a operação;

8. Caso o item 5 for negativo, as dependências são diferentes dos recursos e é feito um teste se é o último pacote do RCS da célula origem a ser testado;

9. Em caso negativo, volta a executar a validação do item 2, comparando novamente o próximo pacote do repositório da célula origem com o FPGA, através dos arquivos descritores;

10. Em caso positivo, é o último pacote do RCS da célula origem, é executado outro teste para verificar a existência ou não de células vizinhas;

11. Em caso negativo, se não existe células vizinhas o serviço é encerrado e enviado uma mensagem ao gerente de que não existe um pacote para este dispositivo;

12. Em caso positivo, ou seja, existe células vizinhas, é disparado uma busca a todas as células vizinhas relacionadas em uma lista;

13. Após é executado outro teste se existe pacotes a serem testados nos RCS das células vizinhas;

14. Caso não tenha mais pacotes a serem testados nos RCS das células vizinhas, é feito outro teste em relação à lista de busca, verificando a existência de endereços de células vizinhas que possam fornecer o pacote desejado;

15. Em caso positivo, se há pacotes no RCS das células a serem ainda testados, é feita novamente uma outra validação comparando os arquivos descritores dos pacotes do RCS das células vizinhas em relação ao FPGA;

16. Testa-se se os modelos e arquiteturas são iguais e a versão do pacote dos repositórios testados é superior ou igual em relação ao FPGA;

17. Caso o teste do item 16, obtiver um resultado negativo, o serviço de busca volta ao item 13 para verificar se algum pacote não foi testado ainda;

18. Caso o teste do item 16 for positivo, o resultado é inserido em uma lista de busca de endereços de células vizinhas que podem fornecer o pacote para este dispositivo;

19. A partir deste momento, ao inserir o resultado testado na lista de busca, volta para o item 13, para testar o próximo pacote do RCS, até o último pacote a ser testado em cada repositório das células vizinhas;

20. Caso a lista de busca não tenha tido endereços inseridos conforme teste do item 14, o serviço é encerrado e ao mesmo tempo é enviado uma mensagem ao gerente da não existência de pacote para este dispositivo;

21. Caso a lista de busca contenha endereços inseridos pelo serviço, após testes, é executado uma verificação daquele endereço da célula que contém o pacote de maior versão;

22. Com o resultado obtido do teste do item 21, executa-se uma comparação dos endereços existentes na lista de busca, verificando assim o endereço da célula que fornecerá o pacote de maior versão, é executado o *download* deste pacote e seu arquivo descritor, da célula vizinha para a célula origem.

Serviço de Distribuição de Pacotes

Outro diagrama de atividades é o do Serviço de Distribuição, na Figura 4.9, onde ocorre a transferência de pacote, onde o mecanismo funciona da seguinte forma:

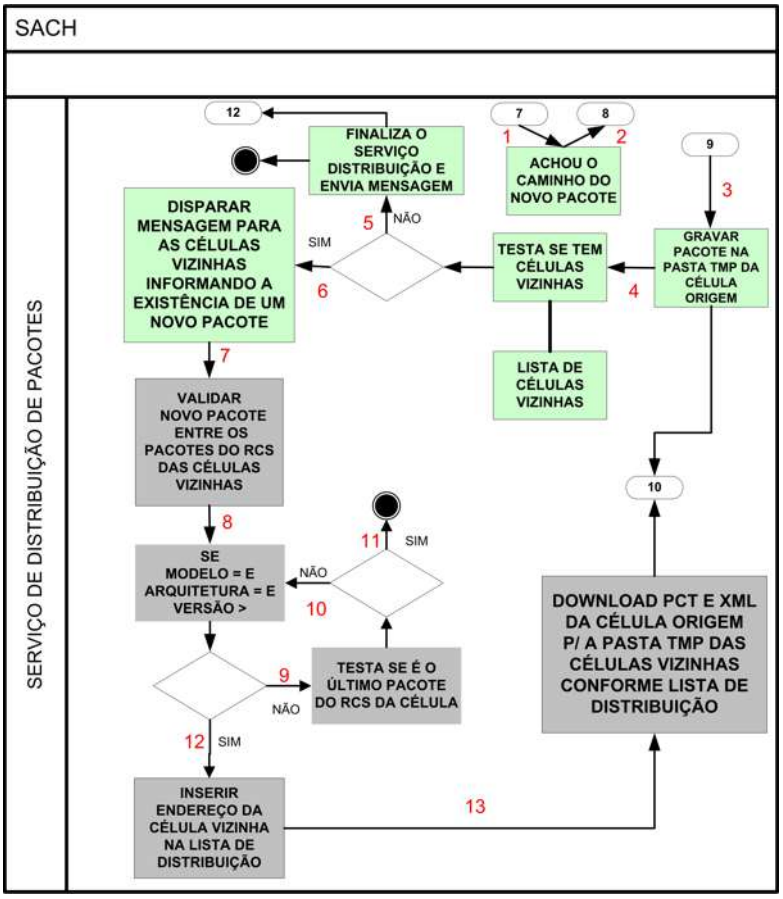

Figura 4.9: Diagrama de Atividades do Serviço de Distribuição de Pacotes

1. 0 gerente inicialmente busca o pacote através da interface gráfica, pressionando o botão buscar localizando o pacote;

2. Ao encontrar o pacote o gerente pressiona o botão salvar;

3. O pacote e seu arquivo descritor são gravados na pasta TMP da célula origem;

4. A seguir o serviço de distribuição executa um teste se existem ou não células vizinhas;

5. Caso o teste do item 4 for negativo, o serviço de distribuição é finalizado e uma mensagem é enviada ao gerente, informando a não existência de células vizinhas;

6. Caso o teste do item 4 for positivo, é disparado uma mensagem a todas as células vizinhas, informando a existência de um novo pacote inserido na célula origem;

7. A partir deste momento, o serviço de distribuição executa testes validações entre o pacote inserido e os pacotes armazenados nos RCS de cada célula vizinha;

8. O teste executado pelo serviço de distribuição no item 7, executa a comparação das informações contidas nos arquivos descritores do pacote inserido em relação aos pacotes dos RCS das células vizinhas, testando se os modelos e arquiteturas são iguais e a versão do pacote inserido é superior aos que se encontram armazenados nos repositórios das células vizinhas;

9. Tendo como resultado negativo o teste do item 8, o serviço de distribuição executa um teste para saber se é o último pacote a ser testado do repositório da célula;

10. Ao ser realizado o teste do item 9 pelo serviço de distribuição, para saber se é o último e o resultado for negativo, é feito novamente o teste em relação ao próximo pacote do repositório, voltando para o item 8;

11. Caso o resultado sendo positivo no teste do item 9, ou seja, foi o último pacote comparado do repositório da célula e o serviço de distribuição é encerrado, significando que esta célula não precisa deste pacote;

12. Caso o teste do item 8 tenha uma resposta positiva, isto significa que o pacote tem o modelo e arquitetura de algum dispositivo instalado na célula vizinha e também que a versão deste pacote é superior a aquela que está armazenada no RCS da célula. Com isso é enviado uma mensagem de retorno das células vizinhas para uma lista de distribuição, informando da necessidade deste pacote, onde esta mensagem é composta dos endereços das células;

13. No momento em que a lista de distribuição começa a receber os endereços de células que necessitam do pacote, o serviço de distribuição executa um *download* da célula origem para todas as células vizinhas validadas.

Atividades e Validações do SACH

Já o diagrama de atividades do SACH ilustrado na Figura 4.10, há vários procedimentos até que seja realizado o seu principal objetivo, que é o envio de pacotes de *bitstream* para os dispositivos reconfiguráveis. O mecanismo funciona da seguinte forma:

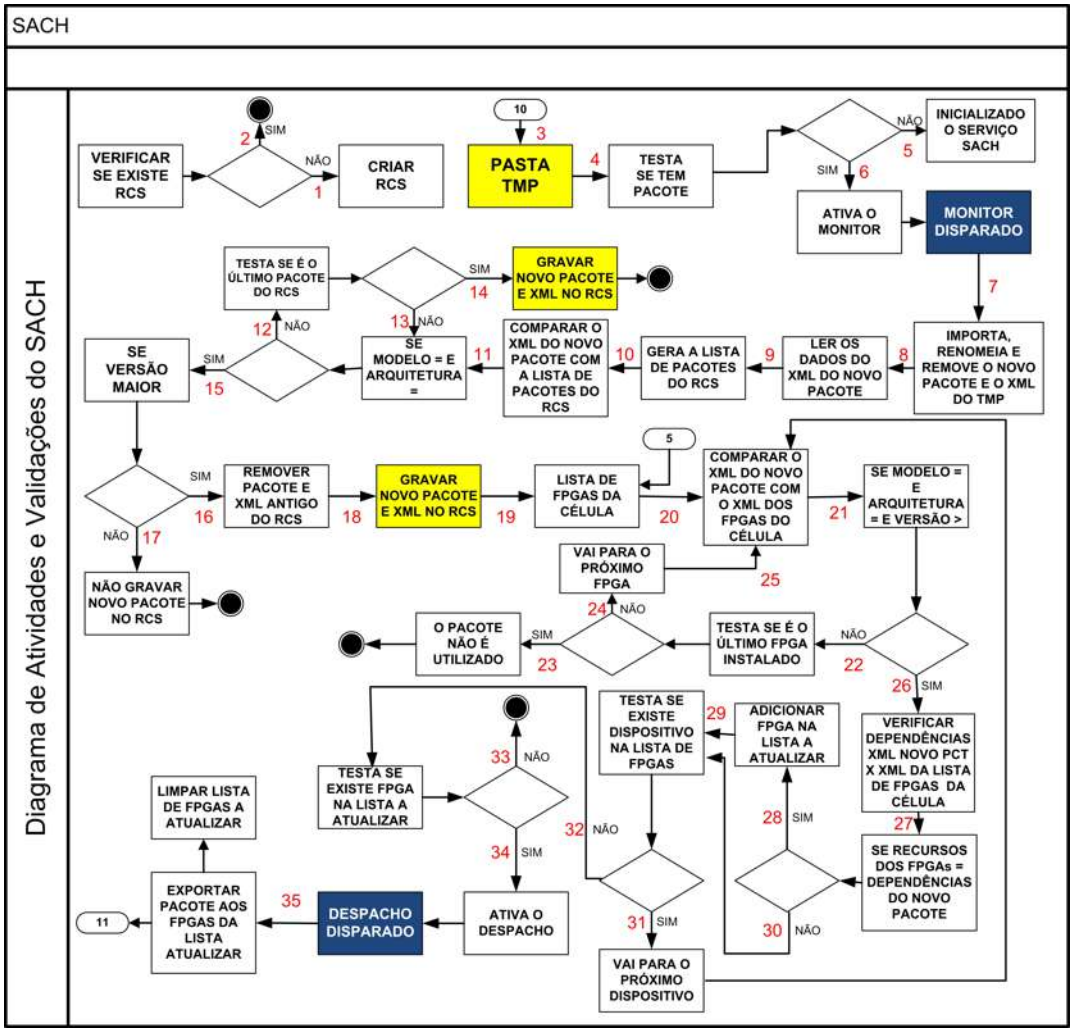

Figura 4.10: Diagrama de Atividades com as Validações de Pacotes do SACH

1. O SACH executa inicialmente um teste se existe ou não o RCS na célula. Caso o resultado for negativo, o repositório é criado;

2. Em caso o resultado seja contrário no item 1, o serviço finaliza o teste. É apenas um teste de inicialização do SACH;

3. O pacote e seu arquivo descritor são gravados pelo serviço de instalação na pasta TMP da célula pelo gerente, através da interface gráfica do SACH;

4. O SACH testa se existe pacote na TMP da célula;

5. No teste do item 4, se o resultado for negativo, o SACH fica apenas inicializado;

6. Caso o teste do item 4, tiver um resultado positivo, ou seja, existe um pacote na TMP da célula. O serviço de monitoramento é ativado pelo SACH;

7. Logo após sua ativação, o serviço de monitoramento dispara o serviço de importação que importa e renomeia o pacote e o seu arquivo descritor da TMP da célula. Os arquivos localizados na TMP são removidos pelo o serviço de remoção do SACH;

8. A seguir, as informações do arquivo descritor do pacote são lidas pelo serviço de leitura;

9. É gerado pelo SACH uma lista de pacotes do RCS da célula;

10. O SACH executa um teste de validação, comparando os arquivos descritores do novo pacote em relação a lista de pacotes do RCS;

11. O teste de comparação do item 10 é realizada da seguinte forma: se modelos forem iguais e arquiteturas iguais em relação aos arquivos descritores comparados do RCS da célula;

12. Em caso negativo desta comparação do item 11, é feito outro teste se é o último pacote do RCS da célula;

13. Deste teste, do item 12, tiver um resultado negativo, volta para o item 11 e testa o próximo pacote do RCS da célula;

14. Caso o teste do item 12, tiver um resultado positivo, mesmo assim, é gravado o pacote e seu arquivo descritor no RCS da célula, para que este pacote seja aproveitado futuramente em alguma reconfiguração de algum dispositivo que vier a ser instalado;

15. Se o resultado do teste no item 11 for positivo, é executado uma outra validação agora em relação a versão do pacote, que deve ser superior aos armazenados no repositório da célula;

16. Caso a versão pacote inserido não seja superior aos armazenados no RCS, o SACH não efetua a gravação do pacote no RCS da célula e encerra o serviço descartando o mesmo;

17. Caso a versão for superior, é feita a exclusão do antigo pacote e seu arquivo descritor do RCS da célula, pelo serviço de remoção;

18. E a seguir, é executada pelo serviço de instalação a gravação do novo pacote e seu descritor no RCS da célula;

19. Logo após, o SACH gera uma lista de FPGAs instalados na célula;

20. É disparada pelo SACH uma outra validação, comparando os arquivos descritores do pacote gravado no RCS da célula em relação ao arquivo descritor da lista de FPGAs;

21. Desta comparação do item 20 é executado um teste da seguinte forma: Se modelo for igual e arquitetura igual e a versão do pacote maior;

22. Caso o teste do item 21 obtiver um resultado negativo, é feito outro teste verificando se é o último FPGA da lista a ser testado;

23. Caso o resultado do teste do item 22 for positivo, sendo o último FPGA a ser testado, o pacote não é utilizado e o SACH é encerrado;

24. Caso o teste for negativo em relação ao item 22, o SACH pega o próximo FPGA da lista;

25. E logo a seguir o SACH volta a executar novamente todos os procedimentos a partir do item 20, comparando o pacote com o próximo FPGA da lista;

26. O SACH, se obtiver no teste do item 21, uma resposta positiva, onde o modelo for igual e arquitetura igual e o pacote inserido tiver uma versão superior a algum FPGA que se encontra instalado na célula, é executado uma nova validação em relação as dependências do pacote e os recursos do FPGA;

27. O próximo teste é a comparação do item 27, onde é realizada da seguinte forma: Se as dependências do pacote forem iguais aos recursos do FPGA;

28. Caso o resultado do teste do item 27 for positivo é gerado pelo SACH uma lista de FPGAs a atualizar onde é feita a inserção do identificador ID deste FPGA nesta lista;

29. A seguir é executado outro teste pelo SACH, se existe ou não dispositivos na lista de FPGAs que ainda não foram testados;

30. Se caso a comparação do item 27, tiver um resultado negativo, se os recursos do FPGA são diferentes das dependências do pacote, o FPGA não é inserido na lista de FPGAs a serem atualizados indo para o item 29 para testar se existe FPGA na lista;

31. Caso o teste do item 29 tiver um resultado positivo, ou seja, existe dispositivos ainda na lista de FPGAs, o SACH pega o próximo dispositivo e retorna para o item 20, para executar todas as validações necessárias;

32. Caso o teste do item 29 for negativo, é executado uma nova validação em relação à lista de FPGAs a atualizar, verificando a existência ou não de FPGAs a serem atualizados na lista;

33. Caso o resultado deste teste do item 32 for negativo, o SACH é encerrado;

34. Caso contrário, se o resultado do teste do item 32 for positivo é ativado o serviço de despacho;

35. O serviço de despacho, dispara o serviço de exportação que distribui o pacote de reconfiguração a todos os FPGAs que estão inseridos na lista de FPGAs a serem atualizados. No final, o serviço de exportação é finalizado, e à lista de FPGAs a atualizar é limpa. Finalizando a operação de distribuição de pacotes de forma distribuída e concorrente aos FPGAs validados da célula.

Simulador

No diagrama de atividades do Simulador de FPGAs, o mecanismo funciona da seguinte maneira, conforme o diagrama ilustrado na Figura 4.11:

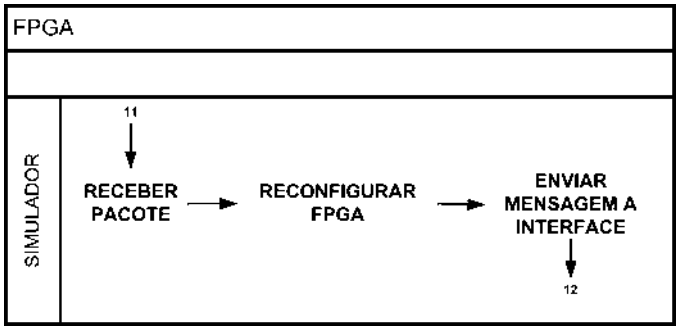

Figura 4.11: Diagrama de Atividades do Simulador de FPGA

1. O SACH através do serviço de despacho, dispara o serviço de exportação que exporta o pacote para o simulador de FPGAs, que recebe o pacote de *bitstream*;

2. Quando o simulador recebe o pacote através do serviço de recebimento, a reconfiguração do(s) dispositivo(s) lógico(s) - FPGA(s) é executada pelo serviço de atualização;

3. No final, o *feedback* é reforçado ao gerente que recebe uma mensagem pela Interface do SACH de que a operação de atualização foi concretizada com sucesso.

4.1.2.3 Diagrama das Funcionalidades dos Módulos do SACH

A Figura 4.12 apresenta um resumo do que foi visto anteriormente no que se refere aos diagramas de atividades do funcionamento do SACH instalado no EXEHDA. A figura ilustra os módulos de funcionamento de uma maneira mais resumida desde a inserção do pacote de configuração no Repositório Celular do Serviço - RCS, passando por todas as validações propostas pelo serviço até conseguir o objetivo que é a propagação de pacotes aos FPGAs que necessitem ser atualizados.

O Serviço apresenta três módulos de operação, onde o **primeiro módulo** é o de inserção de pacotes. O mecanismo funciona da seguinte forma:

1. O gerente utiliza a interface para inserir o pacote e seu descritor;

2. O serviço de distribuição insere na TMP da célula origem o pacote e seu arquivo descritor;

3. No mesmo momento que é distribuído para a TMP da célula origem, é feito um disparo de uma mensagem a todas as células vizinhas informando da existência de um novo pacote;

4. Cada célula vizinha faz a validaçao deste pacote, ou seja, verifica da necessidade ou não do pacote para reconfigurar algum dispositivo que necessite ser atualizado;

5. Caso a validação seja positiva é enviado uma mensagem de retorna com seu endereço para uma lista de distribuição;

6. A cada validação feita pelas células vizinhas, e que o resultado for positivo, o endereço é inserido na lista de distribuição e logo a seguir é feito o *download* do pacote e do arquivo descritor a todas as células vizinhas validadas, relacionadas na lista de distribuição;

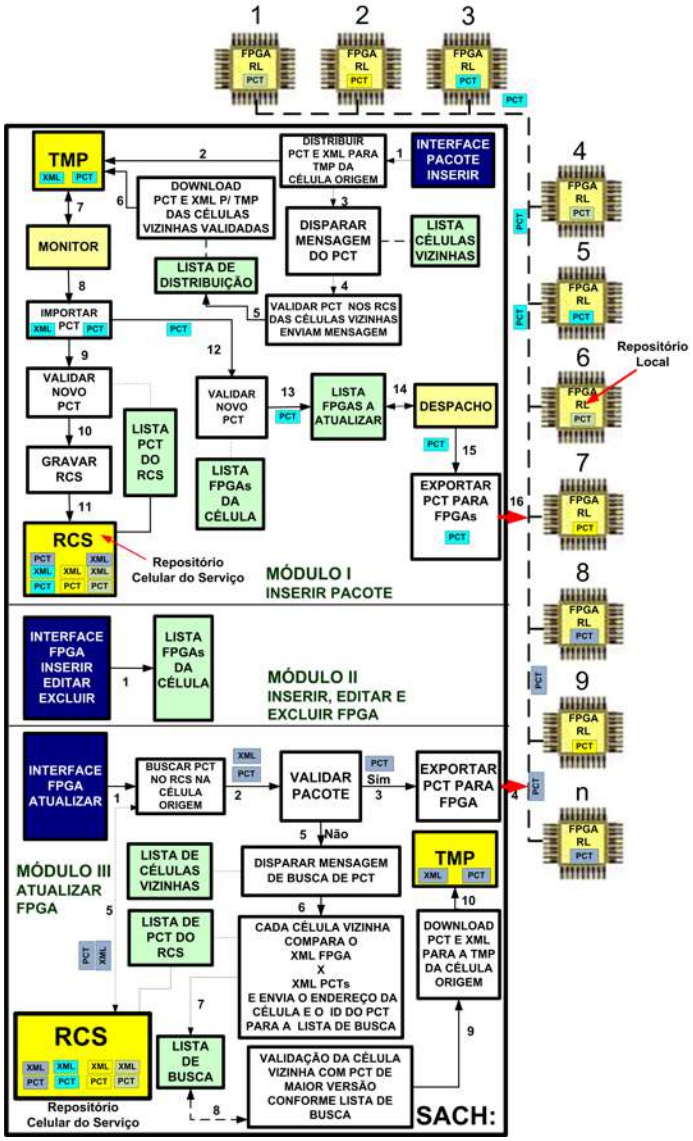

Figura 4.12: Funcionamento do SACH
na EXEHDAbase

7. A seguir o serviço de monitoramento é ativado, verificando a existência na TMP do pacote e descritor;

8. Com isso o serviço de monitoramento aciona o serviço de importação de pacotes, que importa o mesmo e seu descritor para que o serviço de leitura possa ler as informações contidas no arquivo descritor, para que as devidas validações possam ser realizadas;

9. Logo após é realizada a primeira validação em relação ao modelo e arquitetura do FPGA e a versão do pacote. A validação é realizada entre os arquivos descritores do pacote x pacotes do RCS da célula, onde o modelo e arquitetura tem que ser igual e a versão do pacote inserido maior que aqueles que se encontram no RCS da célula;

10. Em caso positivo do teste anterior do item 9, é feita a gravação do novo pacote pelo

serviço de instalação no RCS da célula;

11. E a remoção do pacote e seu descritor mais antigo do RCS da célula;

12. Após a gravação no repositório da célula é feita uma nova validação entre os arquivos descritores do novo pacote x FPGAs instalados na célula;

13. A cada validação positiva feita no item 12, é inserido o ID do FPGA em uma lista de FPGAs a atualizar;

14. A seguir o serviço de despacho, monitora à lista de FPGAs a atualizar;

15. No momento em que o serviço de despacho, verifica a existência de algum ID de FPGA na lista, é disparado o serviço de exportação;

16. Em seguida o serviço de exportação distribui o pacote a todos os FPGAs relacionados na lista de FPGAs a atualizar e a operação é encerrada.

O **segundo módulo** do serviço disponibiliza 3 operações: a inserção de um novo FPGA, a edição e a exclusão de um FPGA já instalado na célula. A inserção é feita incluindo no arquivo descritor dos dispositivos um novo FPGA. A mesmo procedimento ocorre em relação a edição, isto é, alterar as informações do arquivo descritor do referido dispositivo por exemplo e por fim a exclusão que retira do arquivo descritor as informações relativas a esse dispositivo que foi retirado da área de operação fisicamente.

O **terceiro módulo** é representado pela operação de atualização de pacote de um novo FPGA que é instalado na célula e que necessite ser atualizado. Isso acontece em dois momentos: quando um novo FPGA é instalado ou quando o arquivo descritor de um FPGA foi modificado e que necessita de um pacote de *bitstream*. Este módulo funciona da seguinte forma:

1. O gerente ao clicar no sub-item atualizar no menu FPGA da interface gráfica do SACH, tem a opção de escolher um dispositivo que deva sofrer reconfiguração, ou pelo fato de ele ter tido o seus dados alterados no arquivo descritor dos dispositivos da célula ou também pelo motivo de uma nova instalação de um FPGA, onde o arquivo descritor recebe os dados deste novo dispositivo;

2. O gerente ao escolher o dispositivo a ser atualizado, confirma no botão ok e com isso é disparado o serviço de busca de pacote inicialmente no RCS da própria célula origem para o dispositivo selecionado;

3. O serviço de busca dispara o serviço de validação em relação aos pacotes existentes no RCS da célula, executando uma comparações entre modelos, arquiteturas, versões, dependências e recursos;

4. Caso o resultado da validação do item 3, tenha sido positivo em alguma comparação executada, é verificado com isso a existência de um pacote para este dispositivo e com isso o serviço de exportação acionado pelo serviço de despacho, distribui o pacote ao dispositivo;

5. O resultado do teste no item 3, sendo negativo, não existindo pacote na célula origem para este dispositivo, é disparado, pelo serviço de busca, uma mensagem a todas as células vizinhas da célula origem, conforme lista;

6. Cada célula vizinha recebe a mensagem da necessidade de pacote para reconfigurar o dispositivo na célula origem. Logo após, cada célula vizinha tenta validar os pacotes dos seus repositórios para este dispositivo;

7. A cada validação positiva que ocorra, o resultado obtido por cada célula vizinha é enviado por uma mensagem de retorno composta do endereço da célula e o ID do pacote que servirá para reconfigurar o dispositivo em questão, onde essa mensagem de retorno são inseridas em uma lista de busca;

8. Após o término e à lista de busca estiver com mensagens enviadas das células vizinhas é feita uma nova validação para descobrir qual o pacote relacionado na lista é o de maior versão. Com isso garante a reconfiguração do dispositivo com um pacote de versão mais atualizada possível;

9. Após verificado qual o pacote de maior versão relacionado na lista de busca, é feito o *download* da célula vizinha para a pasta TMP da célula origem que solicitou o pacote;

10. A seguir o pacote e o seu arquivo descritor são gravados na pasta TMP da célula origem pelo serviço de instalação. Após a gravação, o SACH realiza as validações necessárias para que o pacote possa ser distribuído ao dispositivo e a reconfiguração possa ser realizada.

4.1.3 Visão da Implementação do Protótipo SACH

Nesta sub-seção são apresentados os diagramas de classe da implementação do protótipo SACH. É formado pelos seguintes diagramas: da interface gráfica, do SACH e do simulador dos dispositivos, totalizando 27 classes.

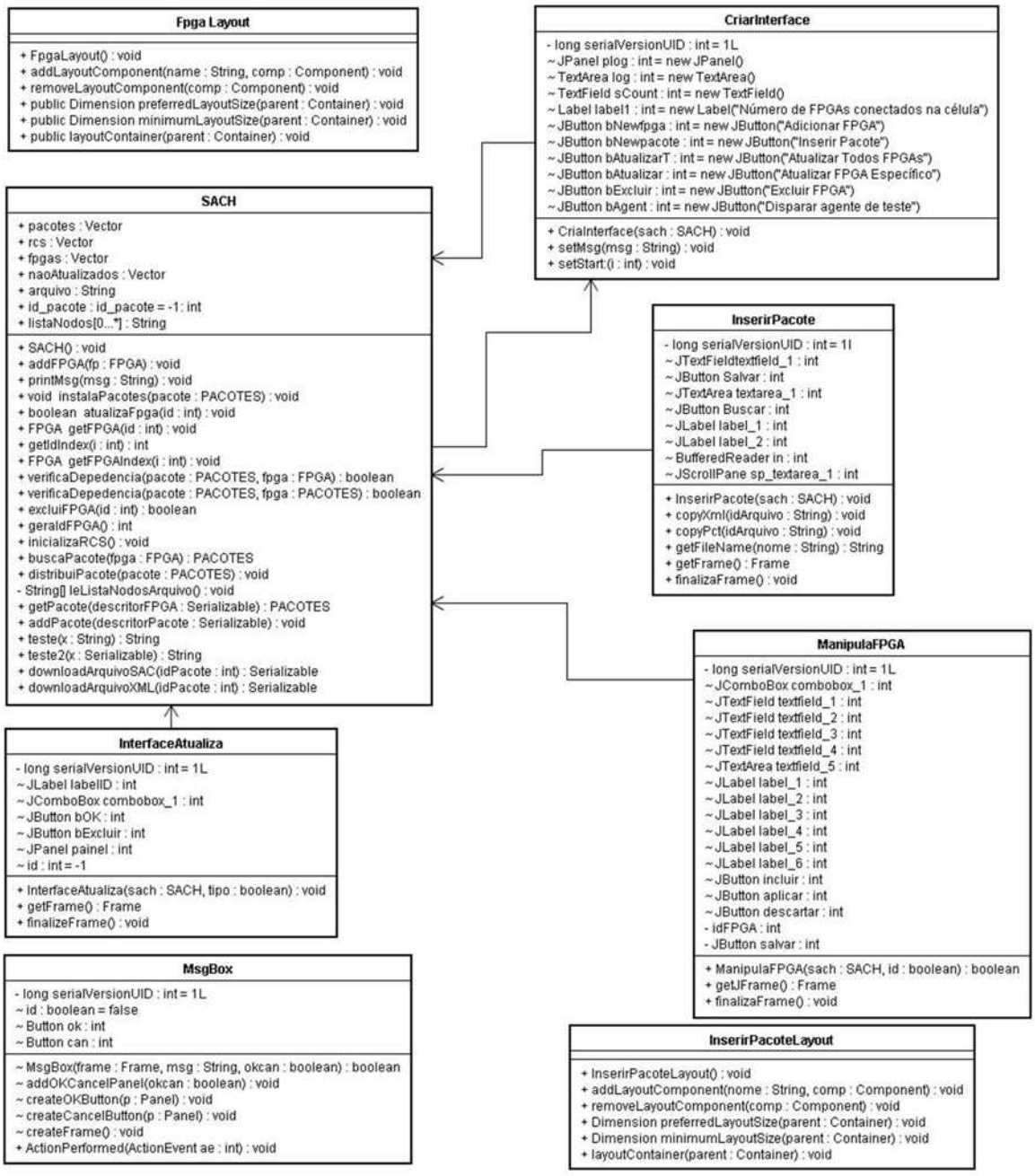

Figura 4.13: Diagrama de Classe da Interface do SACH

A Figura 4.13, ilustra o diagrama de classes da Interface do SACH, formado por oito classes sendo que a classe SACH é a principal classe do serviço, relacionando-se de maneira direta ou indireta com quase todas as classes que compõem o serviço. Este diagrama, apresenta classes responsáveis por todo o layout da implementação da interface gráfica do serviço e também das manipulações de pacotes e FPGAs.

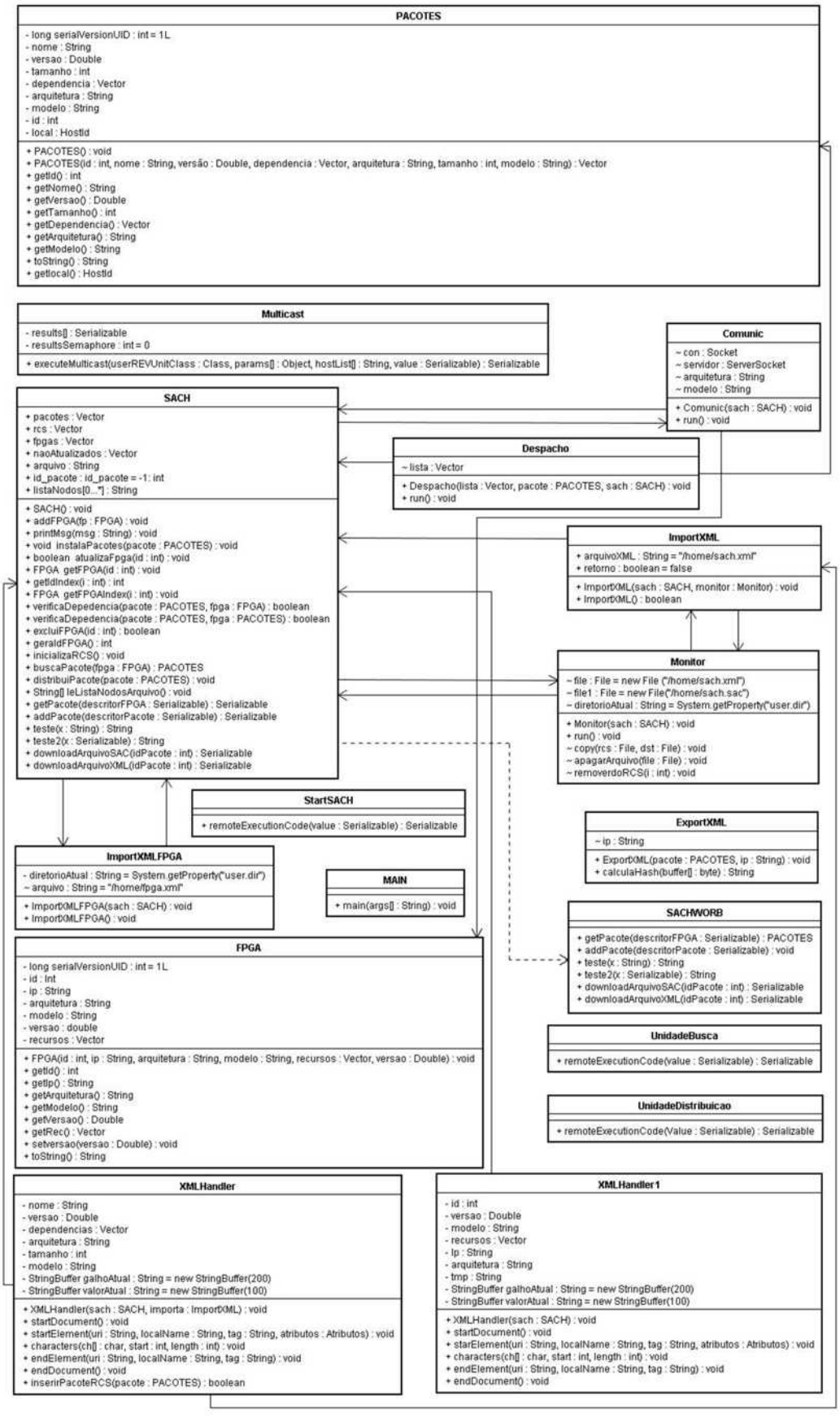

Figura 4.14: Diagrama de Classe do SACH

A Figura 4.14, apresenta o diagrama de classes do SACH, formado por dezessete classes, apresentando além das classes, os seus atributos e métodos utilizados, onde a classe central de todo serviço é o SACH. Este diagrama apresenta todas as funcionalidades do SACH, proporcionando assim todas as validações necessárias para que o pacote possa ser distribuído aos dispositivos que necessitam ser atualizados.

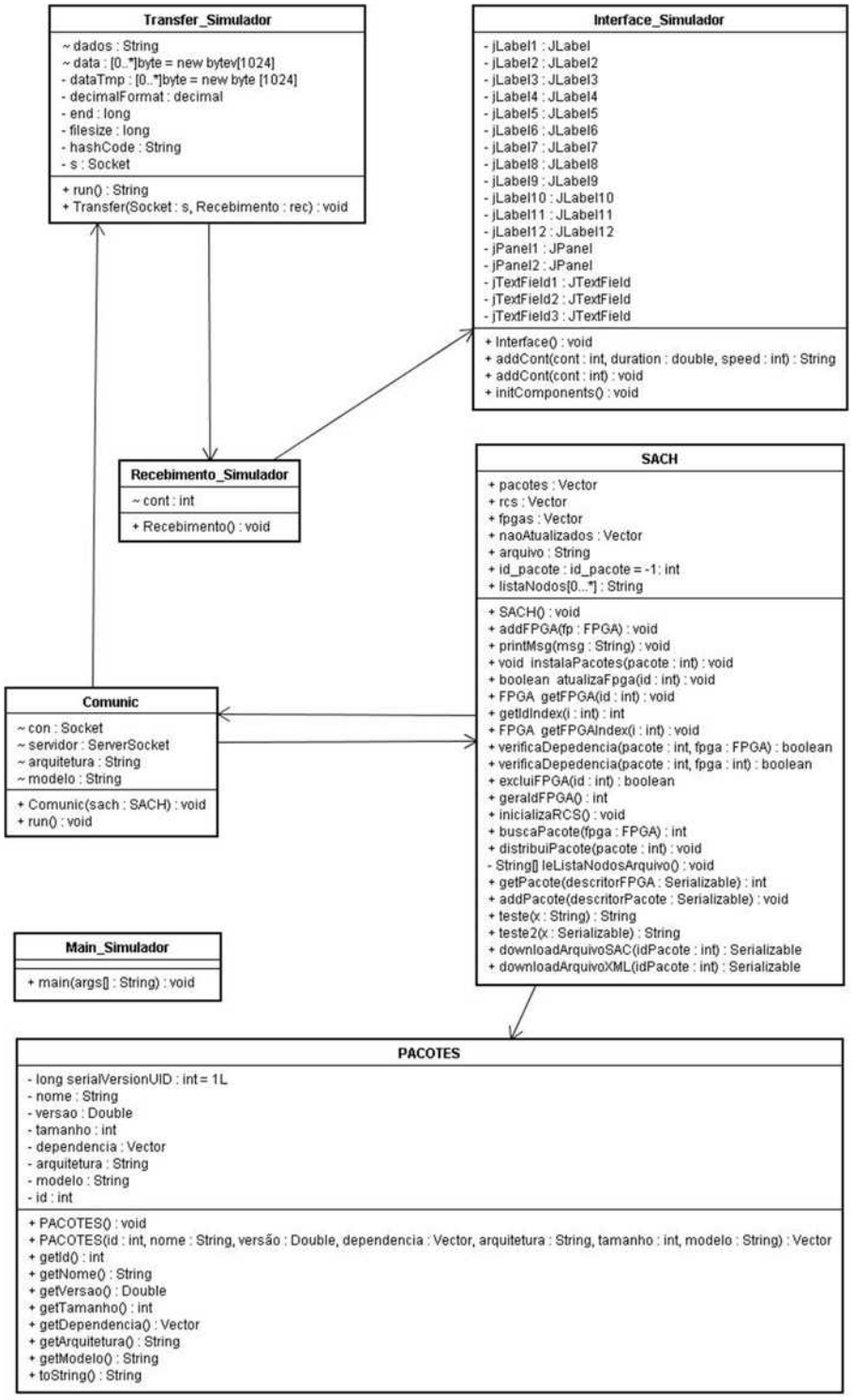

Figura 4.15: Diagrama de Classe do Simulador de FPGA

A Figura 4.15, ilustra o diagrama de classes relativo ao Simulador de FPGAs, formado por sete classes, sendo que três fazem parte do diagrama do SACH que são as classes: SACH, Comunic e Pacotes. O Simulador de FPGA, executa as simulações dos dispositivos, recebendo pacotes de reconfiguração validados pelo SACH para que os dispositivos possam realizar as atualizações necessárias, podendo alterar as funcionalidades lógicas dos dispositivos. O Simulador possui uma porta que recebe o pacote, por socket.

As tecnologias utilizadas para implementar o protótipo SACH foi a Linguagem Java onde foi gerado todo o código correspondente e o sistema operacional usado é o Linux.

4.1.4 Visão das Estratégias de Instalação do SACH

Esta sub-seção apresenta as estratégias de instalação e organização da arquitetura física no qual o SACH é instalado e executado no que se refere ao *hardware*, ou seja, computadores, servidores ou qualquer tipo de dispositivos que suportam o serviço, além de definir como estes equipamentos estão conectados e qual o protocolo de comunicação é utilizado para o envio dos pacotes.

A Figura 4.16 mostra o diagrama de instalação do serviço no que se refere a esta arquitetura para suportar o Serviço de Atualização de Configurações de *Hardware*. A figura apresenta as células em um ambiente pervasivo e cada célula é composta por um Servidor, onde se encontra instalado o SACH e também os Clientes, representados pelos dispositivos - FPGAs.

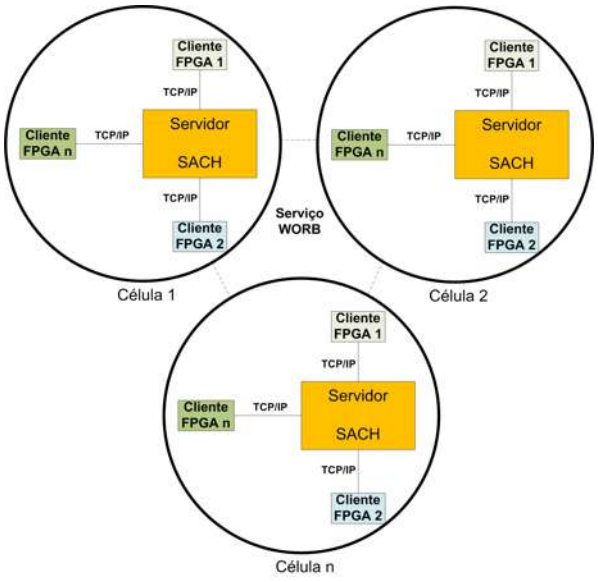

Figura 4.16: Diagrama de Instalação do SACH

A instalação é baseada na modelagem dos casos uso que foram abordadas em seções anteriores deste capítulo. O serviço é desenvolvido para proceder propagações de pacotes de reconfiguração de maneira dinâmica e parcial dos FPGAs, podendo reconfigurar um dipositivo ou um grupo de dispositivos totalmente de forma distribuída e concorrente, sem a intervenção direta do ser humano.

Este serviço é integrado ao *middleware* EXEHDA que proporciona o suporte às aplicações distribuídas dentro de um ambiente pervasivo, utilizando parcialmente alguns

serviços como por exemplo o *Executor* (Serviço de Disparo de Execução), *BDA* (Serviço da Base de Dados pervasiva das Aplicações) e o *CIB* (Serviço que implementa a Base de Informações da Célula), buscando assim atender as demandas da Computação Pervasiva.

A estratégia empregada foi integrar o SACH instalando o mesmo na EXEHDAbase de cada célula. Sobre a comunicação interna do serviço é feita na forma Cliente-Servidor e entre as EXEHDAbases é empregado o protocolo P2P, e inerente ao EXEHDA. O protocolo P2P é organizado na forma denominada super-peer, na qual existe o conceito de células vizinhas. As células vizinhas são priorizadas no momento das comunicações ou buscas (YAMIN, 2004).

O protocolo de comunicação é utilizado TCP/IP de modo a simplificar o suporte necessário de *middleware* nos dispositivos reconfiguráveis. Além disso, as EXEHDAbases usam um esquema de comunicação ponto a ponto, minimizando a centralização e aumentando o potencial de escalabilidade e esta comunicação utiliza um serviço do EXEHDA chamado de WORB que é um protocolo de Invocação Remota de Métodos - RMI, que administra a desconexão, suportando assim o perfil das comunicações no meio pervasivo.

A ilustração na Figura 4.17 mostra como o SACH situa-se no *middleware* EXEHDA. O serviço é instalado em cada EXEHDAbase dentro deste ambiente. Vários dispositivos de borda, em especial os FPGAs, são instalados em cada célula.

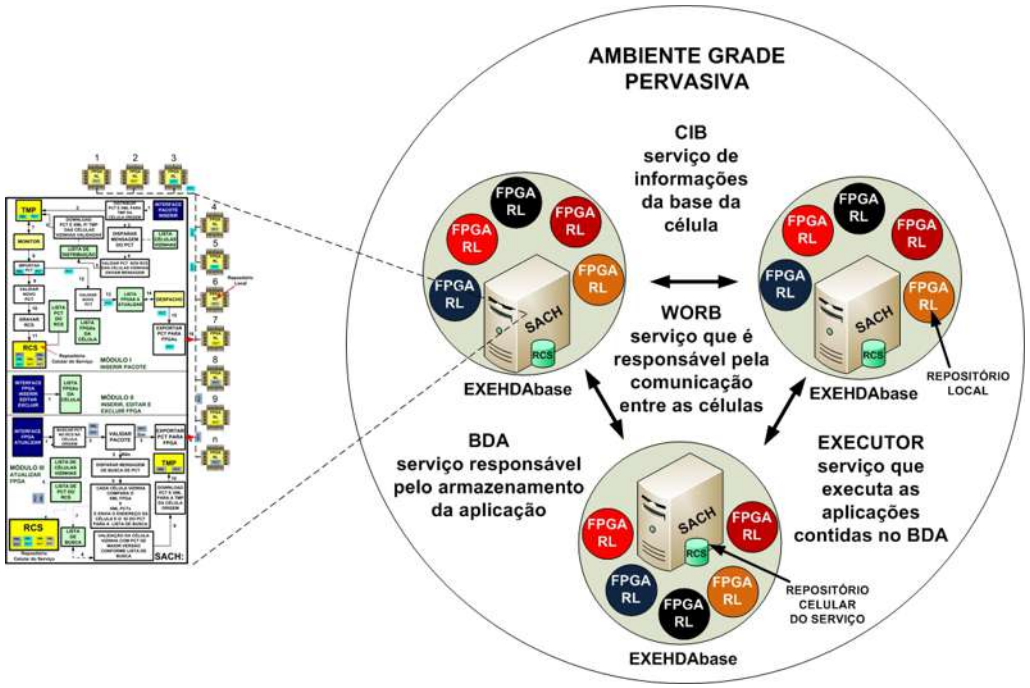

Figura 4.17: SACH em um Ambiente de Grade Pervasiva

Cada célula tem um Repositório Celular do Serviço - RCS onde se encontram armazenados todos os pacotes referentes a cada FPGA, juntamente com seu arquivo descritor. O tipo de envio de pacotes do SACH aos dispositivos é executado de maneira que permita diminuir o uso da banda, enviando pacotes de carga maior apenas para um determinado grupo de dispositivos ou células concorrentemente e de maneira eficiente, diminuindo assim o tráfego na rede.

O modelo utiliza este tipo de tecnologia de transmissão para que não haja um

tráfego na rede excessivo, diminuindo assim o problema de *overhead*. Já a comunicação entre as células é executada pelo Serviço *WORB* do EXEHDA, que executa o acesso remoto as células, onde a comunicação entre as mesmas é executada ponto a ponto, minimizando a centralização e aumentando o potencial de escalabilidade.

O SACH utiliza, principalmente os seguintes serviços do EXEHDA:

- EXECUTOR: que controla e dispara os componentes de *software* da aplicação;
- CIB (Cell Information Base): onde se encontram as informações da célula de execução, bem como as dos usuários;
- WORB: responsável pela comunicação dentro do ambiente pervasivo provido pelo EXEHDA, onde o seu mecanismo é semelhante a RMI, sem que necessite que seja mantida uma conexão ativa durante toda a execução remota da aplicação;
- BDA (Code Repository): é o serviço responsável pelo armazenamento do código pervasivo da aplicação ou serviço que é executada, no caso o SACH é armazenado dentro deste repositório.

4.1.5 Ativações Típicas na Operacionalização do SACH

Nesta seção, são ilustradas várias telas que compõem o SACH, onde a mesma, tenta mostrar uma visão do ambiente de execução do serviço. Inicialmente, antes de entrar no serviço é executado vários comandos até que o mesmo seja inicializado. Os comandos utilizados estão a seguir relacionados:

- exehda-start - dispara a execução do EXEHDA;
- export - para que outra célula possa executar o EXEHDA, o perfil do EXEHDA necessário para execução do SACH é exportado;
- isam-run/home/eduardo/exehda/bin/sach.isam - que é o comando que dispara a execução do Serviço SACH em qualquer célula gerenciada pelo EXEHDA.

A Figura 4.18, ilustra a tela de abertura do serviço a qual é composta do logotipo do SACH. O logotipo do SACH apresenta 3 células conectadas, onde as linhas de interligações são ilimitadas, caracterizando a possibilidade da participação de um grande número de células.

Figura 4.18: Tela de Abertura do SACH

A Figura 4.19, ilustra a tela inicial do SACH. Um menu de opções e um log onde as mensagens são apresentadas pelo serviço, como mostra a figura: inicializando o serviço e o arquivo de configuração de FPGAs sendo importado com sucesso. O menu é composto de três ítens que são:

- Encerrar - apresenta a opção *finalizar* que finaliza a execução do SACH;
- FPGA - composto por 4 sub-ítens: incluir um novo FPGA, editar um dispositivo já instalado, excluir um dispositivo e o de atualizar que busca um pacote na célula origem em que o dispositivo se encontra instalado;
- Pacote - este item do menu dispara o serviço de distribuição do novo pacote, inicialmente para a célula origem e depois para todas as células vizinhas;

Figura 4.19: Tela Inicial do SACH

A Figura 4.20, mostra a tela de inserção de um novo dispositivo, composta pelos seguintes ítens: IP do FPGA, modelo do FPGA, arquitetura do FPGA, recursos do FPGA e versão do software.

Figura 4.20: Tela de Inserção de um Novo FPGA

A Figura 4.21, apresenta a tela de edição ou alteração dos dados de um dispositivo. Esta tela é composta pelos seguintes campos: ID do FPGA onde o gerente escolhe o dispositivo a ser alterado.

Ao escolher o ID, o SACH informa na tela o IP da célula que o FPGA está instalado, o modelo do FPGA, a arquitetura do FPGA, os recursos do FPGA e a versão do pacote que ele utiliza.

Figura 4.21: Tela de Alteração de um FPGA

A Figura 4.22, mostra a tela de exclusão de um dispositivo, composta pelo: ID do FPGA onde o gerente escolhe o dispositivo a ser excluído e logo a seguir pressiona o botão excluir.

Figura 4.22: Tela de Exclusão de um FPGA

A Figura 4.23, apresenta a tela de atualização de pacote, no qual o gerente escolhe o ID do dispositivo a ser reconfigurado na célula.

Logo a seguir pressiona o botão ok para que o serviço de busca possa ser disparado para encontrar entre as células, o pacote necessário para a reconfiguração deste dispositivo.

Esta atualização ocorre em dois momentos: após a inserção de um novo FPGA ou no caso de alteração dos metadados do arquivo descritor de algum dispositivo instalado na célula.

Figura 4.23: Tela de Atualização de um FPGA

Ao clicar na tela principal ilustrada na Figura 4.19, no item pacote, surge o subitem inserir pacote que ao ser clicado surge a Figura 4.24, que mostra a tela de inserção de um novo pacote de reconfiguração nas células.

Esta tela apresenta dois botões:

- Busca - que localiza o descritor do novo pacote em algum diretório, e a partir da localização o conteúdo do arquivo é apresentado na tela;
- Salvar - responsável pela gravação do novo pacote e seu XML na TMP da célula origem.

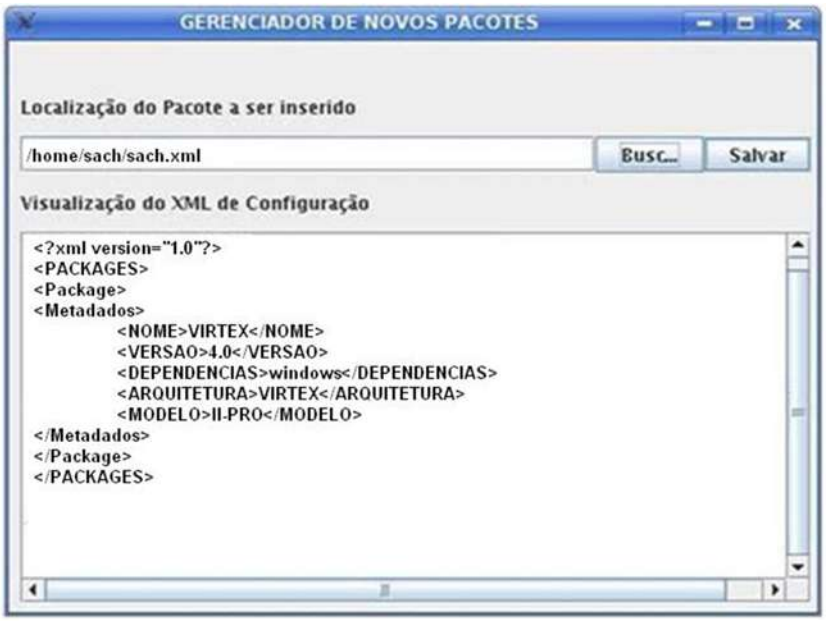

Figura 4.24: Tela de Inserção de um Novo Pacote

A Figura 4.25, apresenta a tela de controle do SACH, que mostra as mensagens das operações executadas pelo serviço, desde a entrada do pacote e seu descritor, passando

pelas validações realizadas até que o serviço de exportação de pacote distribua ao Simulador de FPGAs. No final de todas as mensagens é apresentado o resultado do Tempo de Processamento e Distribuição de pacotes.

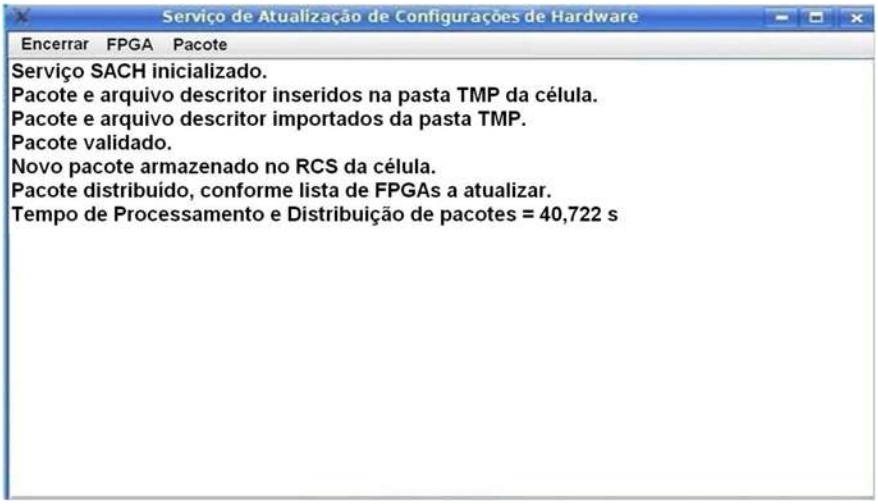

Figura 4.25: Tela de Controle e Tempo de Processamento e Distribuição de pacote do SACH

- Serviço SACH inicializado - significa que o serviço está pronto ser utilizado;
- Pacote e arquivo descritor inseridos na pasta TMP da célula - o gerente insere o novo pacote e seu descritor na TMP da célula;
- Pacote e arquivo descritor importados da pasta TMP - acontece quando o pacote e descritor foram extraídos da TMP;
- Pacote validado - significa que o pacote é válido para ser utilizado para reconfigurar algum dispositivo que se encontre instalado na célula;
- Novo pacote armazenado no RCS da célula - significa que o novo pacote foi armazenado no RCS, substituíndo o antigo pacote pelo novo ou até mesmo inserido o novo pacote sem substituir nenhum, caso não haja algum deste tipo, para que no futuro, o pacote possa ser utilizado por esta célula para reconfigurar algum novo dispositivo que se queira instalar ou até mesmo fornecer este pacote para outras células. O pacote é descartado somente quando o modelo e arquitetura são iguais e a versão é inferior a que se encontra no RCS;
- Pacote distribuído, conforme lista de FPGAs a atualizar - esta mensagem surge quando é feita uma nova validação entre os dados do XML do novo pacote que foi instalado no RCS e o XML dos FPGAs que se encontram instalados na célula. A medida em que é feita a validação é gerada à lista de FPGAs a Atualizar, ou seja, à lista de dispositivos a serem atualizados, no qual o serviço de exportação é acionado e dispara o novo pacote somente aos dispositivos que se encontram identificados nesta lista. Após surge a mensagem na tela;
- Tempo de Processamento e Distribuição de pacotes: 40,722 segundos - é o tempo, desde o processamento ou validações até a distribuição de pacotes executado pelo

serviço de exportação do SACH, conforme à lista de FPGAs a serem atualizados, reconfigurando 2.000 FPGAs, com um pacote de 1 MiB.

A Figura 4.26, mostra a tela do Módulo Simulador de FPGA, onde o simulador fornece três dados:

Figura 4.26: Tela do Módulo Simulador de FPGA

- O número de FPGAs que receberam o pacote de reconfiguração, no momento do teste realizado;
- O tempo de atualização unitária, é o tempo unitário que cada pacote é distribuído a cada dispositivo;
- A velocidade da rede no momento da distribuição do pacote aos dispositivos.

4.2 Conclusão

Neste capítulo é apresentado a modelagem do SACH dividida pelas seguintes visões: visão da arquitetura do SACH - com seus serviços básicos; visão de caso de uso dos atores e as atividades do SACH; visão da implementação do protótipo e visão das estratégias de instalação do serviço e finalizando com as ativações típicas na operacionalização do SACH, apresentando os comandos utilizados e a interface gráfica implementada para o serviço.

5 RESULTADOS

Para validar o SACH, foi desenvolvido um protótipo onde vários testes foram realizados. O protótipo SACH funcionou bem, validando e distribuíndo pacotes de reconfiguração a vários FPGAs simulados em cada célula em um ambiente pervasivo.

Os resultados obtidos através dos testes realizados com o protótipo SACH, mesmo na sua versão inicial, foram considerados satisfatórios, alcançando o objetivo central do trabalho.

A escalabilidade do protótipo é uma ótima característica do serviço propiciando reconfigurações distribuídas e concorrentes para um número relativamente razoável de dispositivos, chegando a 50.000 FPGAs simulados, de mesmo modelo e arquitetura, em cada célula.

Foram realizados testes para medir o tempo que o protótipo SACH leva desde a inserção do pacote no serviço, passando por todas as validações necessárias, até a exportação do pacote aos dispositivos. Este tempo é denominado de Tempo de Processamento e Distribuição de pacotes para os dispositivos a serem reconfigurados, que é a grandeza medida nos testes realizados com o protótipo e seus valores estão discutidos nas seções 5.2 e 5.3.

Neste capítulo são apresentadas as seguintes seções: 5.1 Ambiente de avaliação do SACH; 5.2 Resultados por número de dispositivos simulados; 5.3 Resultados por tamanho de pacote distribuído e na 5.4 Conclusões.

5.1 Ambiente de Avaliação

Na Tabela 5.1, estão relacionados os equipamentos que foram utilizados para os testes do SACH, juntamente com as informações técnicas de cada máquina. Foram usadas cinco máquinas, sendo que uma para o Simulador de FPGAs, onde são realizadas as simulações do recebimento de pacote de reconfiguração pelos dispositivos FPGAs. Nos testes é usado a célula número 1 como sendo a célula origem e as células de números 0, 2, 3 e 4 são as chamadas células vizinhas.

Os testes foram realizados com a utilização de cinco máquinas, sendo uma utilizada para o Simulador de FPGA, responsável pelo recebimento e a reconfiguração do dispositivo e quatro máquinas utilizadas cada uma caracterizando uma célula de execução, particularmente uma EXEHDAbase, ilustrada na Tabela 5.1.

Tabela 5.1: Informações Técnicas dos Equipamentos Utilizados

Descrição	Célula 0	Célula 1	Célula 2	Célula 3	Célula 4	Simulador
Modelo	$Intel$	$Intel$	$Intel$	$Intel$	$Intel$	$Intel$
Processador	$Celeron$	$Core\ Duo$	$Celeron$	$Celeron$	$Celeron$	$Celeron$
Velocidade (Hz)	$300M$	$1.66G$	$300M$	$300M$	$300M$	$2.66G$
HD (GB)	6.4	60	6.4	6.4	6.4	60
RAM (MiB)	128	512	128	128	128	512

Foi medido também o tempo de atualização unitária de cada dispositivo que o Simulador calculou, onde não houve uma variação considerável ficando em torno de 0,015 a 0,016 segundos e a velocidade de rede se manteve entre 62,50 e 66,67 Kb/s para todos os testes realizados, onde a velocidade máxima da rede utilizada foi de 100 Mb/s.

Nota-se que a velocidade máxima da rede utilizada ficou em torno de 60 a 70% da nonimal. A seguir são apresentados os resultados obtidos em relação ao Tempo de Processamento e Distribuição de pacotes do SACH, nas seguintes perspectivas: do número de dispositivos simulados e do tamanho de pacote distribuído.

5.2 Resultados por Número de Dispositivos Simulados

Nesta seção, serão discutidos os resultados obtidos de tempo de processamento e distribuição de pacotes, em relação ao número de dispositivos simulados na célula.

O primeiros testes realizados foram em relação a escalabilidade do protótipo SACH, ilustrados nos gráficos das Figuras 5.1 e 5.2. Estes testes apresentam todas as quantidades simuladas de FPGAs com o referido Tempo de Processamento e Distribuição de pacotes de cada grupo de dispositivos, de pacotes de tamanhos fixos de 30 KiB e 1000 KiB.

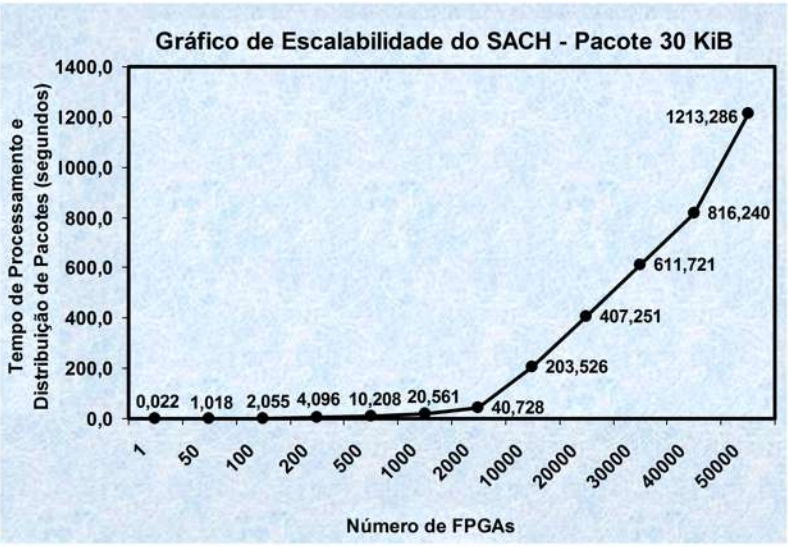

Figura 5.1: Gráfico de Escalabilidade x Tempo de Processamento e Distribuição de pacotes do SACH - Pacote de 30 KiB

Nos testes referente ao gráfico apresentado na Figura 5.1 verifica-se que até 2.000 FPGAs, o tempo gasto para realizar todo o procedimento e a distribuição de pacote, utilizando um pacote de 30 KiB, ficou entre 0,022 a 41 segundos. A partir de 10.000 a 40.000 FPGAs simulados, o tempo variou entre 203 a 818 segundos, representando uma variação de 3,4 a 13 minutos. O aumento em relação ao número de dispositivos de dez em dez mil, proporciona um acréscimo de tempo na faixa de 200 segundos até 40.000 FPGAs. Com relação a um número maior, no caso 50.000 FPGAs, se obteve nos testes um valor de tempo gasto que ficou em 1213,286 segundos, ou seja, em torno de 20 minutos para reconfigurar 50.000 dispositivos.

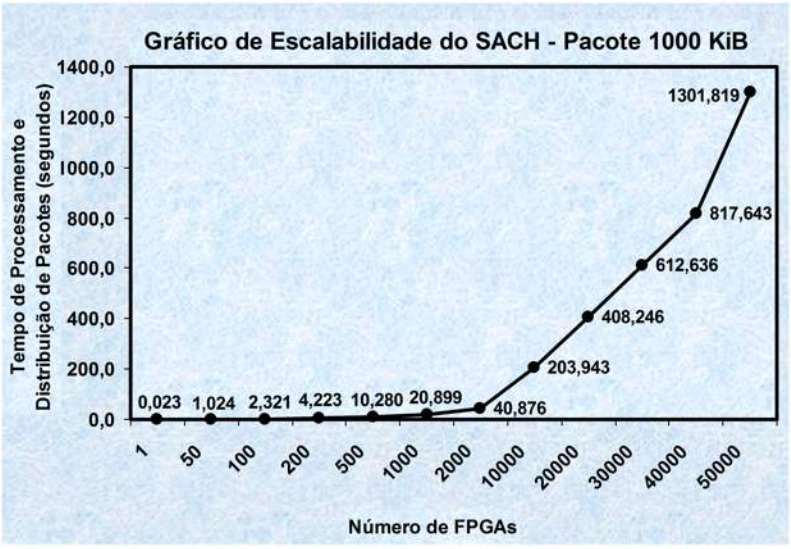

Figura 5.2: Gráfico de Escalabilidade x Tempo de Processamento e Distribuição de pacotes do SACH - Pacote de 1000 KiB

Nos testes referente ao gráfico apresentado na Figura 5.2 verifica-se que a reconfiguração de 1 até 2.000 FPGAs, o tempo gasto para realizar todo o procedimento e a distribuição de pacote, utilizando um pacote de 1000 KiB, ficou entre 0,023 a 41 segundos. A partir de 10.000 a 40.000 FPGAs simulados, o tempo variou entre 204 a 819 segundos, representando uma variação de 3,5 a 13,65 minutos. Verifica-se também que o aumento em relação ao número de dispositivos de dez em dez mil, proporciona um acréscimo de tempo que se mantém na faixa de 200 segundos até 40.000 FPGAs. Com relação a 50.000 FPGAs, obteve-se um valor de tempo que se manteve em torno de 1301,819 segundos, ou seja, entre 21 e 22 minutos para reconfigurar 50.000 dispositivos.

5.3 Resultados por Tamanho de Pacote Distribuído

Nesta seção, serão discutidos os resultados obtidos de tempo de processamento e distribuição de pacotes, em relação ao tamanho de pacote distribuído para os dispositivos na célula.

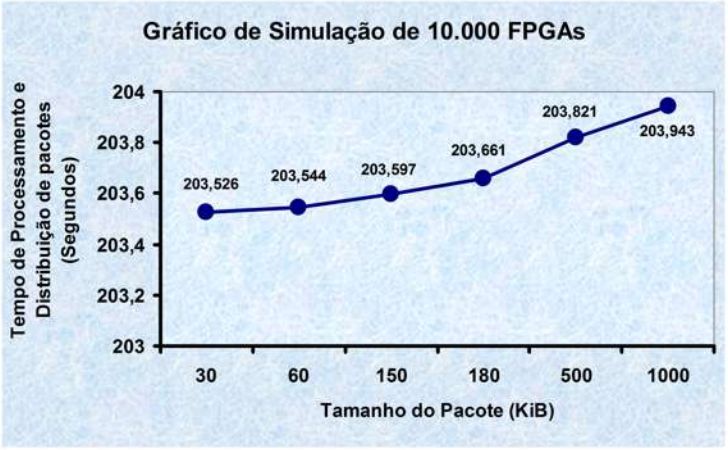

Figura 5.3: Gráfico de Tempo de Processamento e Distribuição de pacotes do SACH - 10.000 FPGAs

O gráfico ilustrado na Figura 5.3 apresenta a simulação de 10.000 FPGAs que recebem pacotes de reconfiguração de tamanhos entre 30 KiB a 1000 KiB, onde o Tempo de Processamento e Distribuição permaneceu entre 203,526 a 203,943 segundos, representando em torno de 3,5 minutos gastos para executar a distribuição. A diferença do tempo e percentual de aumento em relação ao tamanho dos pacotes testados, em uma faixa de 30 KiB até 1000 KiB foi de 0,417 segundos, ou seja, equivalendo a um aumento de tempo de 0,205%.

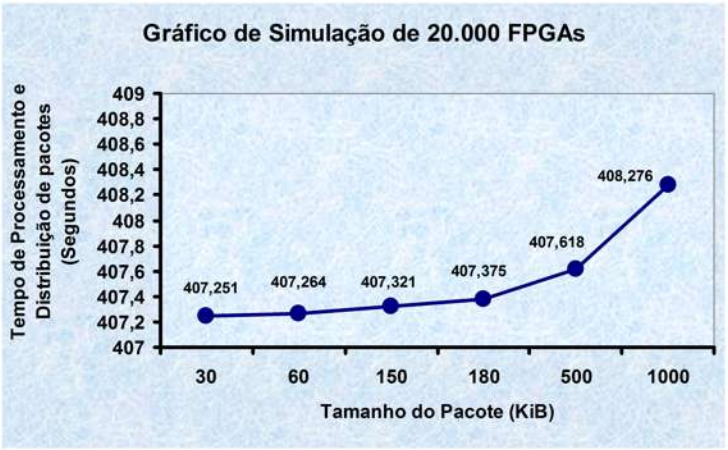

Figura 5.4: Gráfico de Tempo de Processamento e Distribuição de pacotes do SACH - 20.000 FPGAs

O gráfico ilustrado na Figura 5.4 apresenta a simulação de 20.000 FPGAs que recebem pacotes de reconfiguração de tamanhos entre 30 KiB a 1000 KiB, onde o Tempo de Processamento e Distribuição permaneceu entre 407,251 a 408,276 segundos, representando em torno de 6,5 minutos gastos para executar a distribuição. A diferença do tempo e percentual de aumento em relação ao tamanho dos pacotes testados, em uma faixa de 30 KiB até 1000 KiB foi de 0,995 segundos, ou seja, equivalendo a um aumento de tempo de 0,244%.

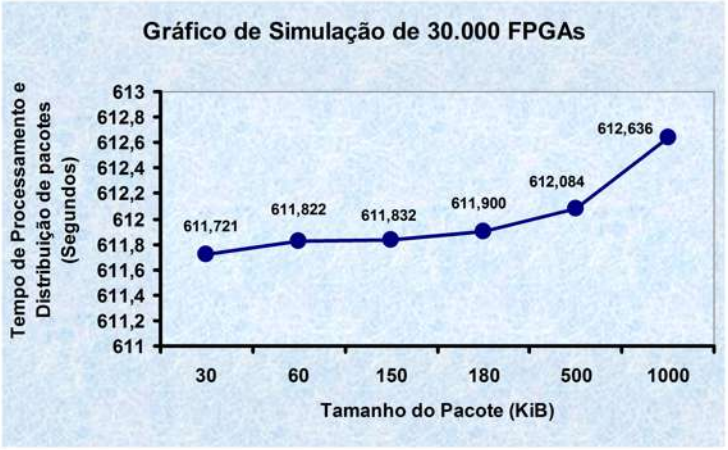

Figura 5.5: Gráfico de Tempo de Processamento e Distribuição de pacotes do SACH - 30.000 FPGAs

O gráfico ilustrado na Figura 5.5 apresenta a simulação de 30.000 FPGAs que recebem pacotes de reconfiguração de tamanhos entre 30 KiB a 1000 KiB, onde o Tempo de Processamento e Distribuição permaneceu entre 611,721 a 612,636 segundos, representando em torno de 10 minutos gastos para executar a distribuição. A diferença do tempo e percentual de aumento em relação ao tamanho dos pacotes testados, em uma faixa de 30 KiB até 1000 KiB foi de 0,915 segundos, ou seja, equivalendo a um aumento de tempo de 0,149%.

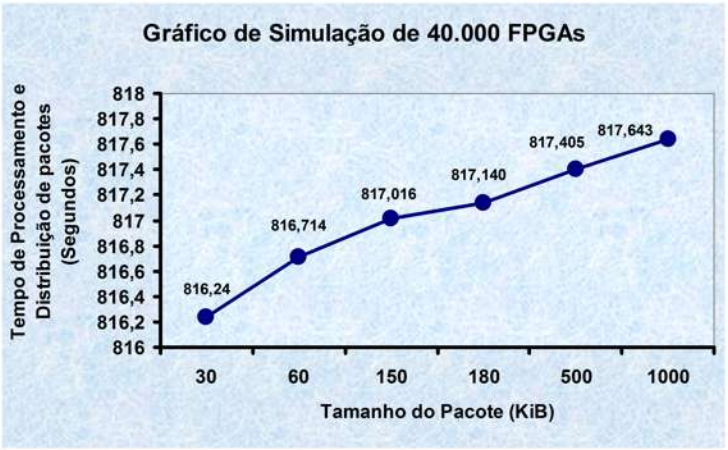

Figura 5.6: Gráfico de Tempo de Processamento e Distribuição de pacotes do SACH - 40.000 FPGAs

O gráfico ilustrado na Figura 5.6 apresenta a simulação de 40.000 FPGAs que recebem pacotes de reconfiguração de tamanhos entre 30 KiB a 1000 KiB, onde o Tempo de Processamento e Distribuição permaneceu entre 816,240 a 817,643 segundos, representando em torno de 13 e 14 minutos gastos para executar a distribuição. A diferença do tempo e percentual de aumento em relação ao tamanho dos pacotes testados, em uma faixa de 30 KiB até 1000 KiB foi de 1,403 segundos, ou seja, equivalendo a um aumento de tempo de 0,171%.

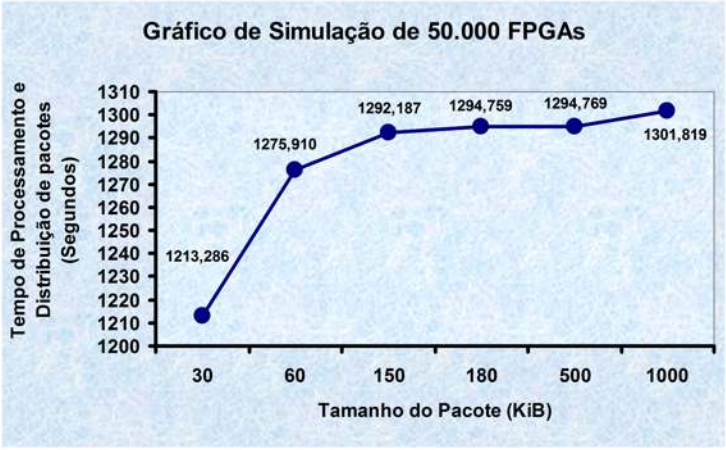

Figura 5.7: Gráfico de Tempo de Processamento e Distribuição de pacotes do SACH - 50.000 FPGAs

O gráfico ilustrado na Figura 5.7 apresenta a simulação de 50.000 FPGAs que recebem pacotes de reconfiguração de tamanhos entre 30 KiB a 1000 KiB, onde o Tempo de Processamento e Distribuição permaneceu entre 1213,286 a 1301,819 segundos, representando em torno de 20 e 22 minutos gastos para executar a distribuição. A diferença do tempo e percentual de aumento em relação ao tamanho dos pacotes testados, em uma faixa de 30 KiB até 1000 KiB foi de 88,533 segundos, ou seja, equivalendo a um aumento de tempo de 7,296%.

Neste gráfico observa-se que ocorreu uma variação diferente em relação a outros testes realizados. Houve um acréscimo maior na variação de tempo, devido o aumento do número de dispositivos ter atingido um patamar no qual o Protótipo SACH começou a ter uma demora bem mais acentuada do que em números inferiores, como é o caso nos testes que utilizaram 40.000 FPGAs.

Outro fato importante observado nos testes é que aumentando-se o número de dispositivos simulados acima de 50.000 o SACH é bloqueado. O motivo deste bloqueio, é o tamanho do arquivo descritor (fpga.xml) dos dispositivos, que quando atinge um valor acima de 10 MiB, o serviço termina com uma exceção. Mesmo assim, o SACH contempla os objetivos do trabalho, possibilitando a distribuição de pacotes para até 50.000 FPGAs, de mesmo modelo, para cada célula. Neste sentido, este número alcançado pelo SACH na distribuição de pacotes é satisfatório, sendo ele um protótipo.

O referido problema acima descrito poderá ser solucionado na continuidade de trabalhos futuros, inserindo as informações contidas no arquivo descritor de cada dispositivo da célula, em uma base de dados. Em conseqüência, o serviço terá um aumento na sua escalabilidade, podendo distribuir pacotes para um número de dispositivos acima de 50.000 em cada célula.

5.4 Conclusão

O protótipo SACH atingiu suas metas. Os testes realizados tiveram resultados satisfatórios em relação as validações e distribuições de atualizações de pacotes para os dispositivos simulados, propagando pacotes aos dispositivos na célula em um ambiente pervasivo.

Os valores de tempo de processamento e distribuição de pacotes alcançados nos testes do protótipo, indicaram um bom desempenho em relação ao número máximo de dispositivos atualizados.

Duas características foram observadas nos testes: o número de dispositivos simulados, que teve uma influência importante no tempo de todo o procedimento realizado pelo serviço até a propagação de pacotes válidos aos dispositivos, onde o tempo aumentou, à medida que o número de dispositivos simulados foi acrescido na célula; outra característica seria o tamanho de pacote, onde é observado nos testes um aumento mínimo de tempo, que não causou influência no desempenho do SACH.

6 CONSIDERAÇÕES FINAIS

Neste trabalho foram feitos estudos que possibilitaram fundamentar o modelo do serviço desenvolvido para reconfigurar dispositivos de borda, em especial os FPGAs, dentro de um ambiente pervasivo. Uma das contribuições do SACH é a possibilidade que o mesmo confere ao serviço de poder de distribuir pacotes de reconfiguração a dispositivos lógicos de forma concorrente, diminuindo o custo de tempo e também pela diminuição dos problemas de *overhead* na rede pelo uso de reconfigurações dinâmicas e parciais.

Este capítulo descreve uma pequena introdução e se divide em 3 seções: 6.1 Principais conclusões obtidas em relação aos testes executados com o protótipo SACH; 6.2 As publicações realizadas por este trabalho, e por fim a seção 6.3. Os trabalhos futuros que podem ser explorados para aperfeiçoar o SACH.

6.1 Principais Conclusões

A Computação Reconfigurável é uma solução intermediária desenvolvida para unir o desempenho de um hardware fixo (ASIC *Application-Specific Integrated Circuit*) com a flexibilidade de um hardware programável (processadores de propósito geral).

Com isso, a unificação do desempenho e a flexibilidade, tornam as arquiteturas reconfiguráveis - FPGAs, dependendo da aplicação que se queira executar, bem atraente. A flexbilidade é uma vantagem primordial tendo assim a possibilidade de alterar as funções lógicas do dispositivo, através de reconfigurações dinâmicas parciais ou totais se for caso, pelo serviço.

Foram abordados também os desafios de pesquisa para a atualização dinâmica de dispositivos reconfiguráveis, onde o objetivo do trabalho é o SACH que foi concebido para facilitar o processo de propagação de pacotes de reconfiguração de forma dinâmica e parcial, há vários FPGAs concorrentemente em um ambiente pervasivo, sem a necessidade de intervenção direta de operadores para reconfigurar os dispositivos. Neste trabalho, o *middleware* EXEHDA é quem proporciona o suporte à execução distribuída na perspectiva da Computação Pervasiva.

O SACH procura diminuir o uso da banda enviando os pacotes apenas para as células que necessitem dos mesmos para atualizar seus dispositivos e também se valer de concorrência no procedimento de atualização, no momento que diversas células de execução podem ter o SACH de forma concorrente.

Desta forma, o procedimento de configuração dos dispositivos se torna mais rápido, principalmente dentro de um ambiente pervasivo onde poderão ter um número elevado de dispositivos. Atualmente, o SACH já possui um protótipo para atualização de

FPGAs e um simulador de FPGAs para testes.

A implementação do protótipo foi realizada com êxito, conforme Seção 4.1.3 desta dissertação, onde o SACH foi desenvolvido e devidamente testado, integrando-se ao grupo de serviços já existentes do *middleware* EXEHDA. O SACH atingiu os objetivos desejados pelo trabalho, gerenciando todas as questões de validação e distribuição de pacotes aos FPGAs.

Os resultados obtidos, descritos nas seções 5.2 e 5.3 desta dissertação, foram satisfatórios, pois, a utilização do SACH para reconfigurar dispositivos do tipo FPGA, diminui consideravelmente o tempo gasto para reconfigurar um número elevado de dispositivos. O SACH funcionou bem, atingindo os objetivos do trabalho.

6.2 Publicações Realizadas

Ao longo do desenvolvimento do trabalho de dissertação, foram realizadas publicações, as quais estão relacionadas abaixo:

- Pervasividade no Contexto de Dispositivos de Borda Reconfiguráveis - ERAD 2007: 7 Escola Regional de Alto Desempenho - Fórum de Pós-Graduação - 16 a 19 de Janeiro de 2007, PUC - Porto Alegre/RS - Brasil;
- Gerenciamento de Dispositivos de Borda Reconfiguráveis na Computação - 6 Mostra de Pós-Graduação da UCPel 2007, 22 a 24 de Outubro de 2007, Pelotas/RS - Brasil - Prêmio Destaque Pesquisador;
- Gerenciamento de Dispositivos de Borda Reconfiguráveis na Computação Pervasiva - IX Encontro de Extensão da UFPel 2007, 27 a 29 de Novembro de 2007, Pelotas/RS;
- Gerenciamento de Dispositivos de Borda Reconfiguráveis na Computação Pervasiva - WPUC 2007: I Workshop on Pervasive and Ubiquitous Computing, 24 a 27 de outubro de 2007, Gramado/RS - Brasil;
- Serviço de Gerenciamento de Dispositivos de Borda Reconfiguráveis em um Ambiente Pervasivo - SPL 2008 - IEEE IV Southern Programmable Logic Conference, 26 a 28 de Março de 2008, Bariloche, Argentina (aceito para publicação);
- Gerenciamento de Dispositivos de Borda Reconfiguráveis na Computação Pervasiva - ERAD 2008: 8 Escola Regional de Alto Desempenho - Fórum de Pós-Graduação - 11 a 14 de Março de 2008, na Universidade de Santa Cruz do Sul - UNISC, Santa Cruz do Sul/RS - Brasil (aceito para publicação);

6.3 Trabalhos Futuros

No desenvolvimento dos esforços de estudo e pesquisa do SACH, foram identificadas algumas oportunidades de trabalho. Abaixo estão relacionadas as entendidas como mais importantes:

- Desenvolver uma estratégia para que os dispositivos reconfiguráveis possam propagar pacotes entre si de forma *peer-to-peer*;

- Criar uma base de dados para que as informações dos descritores XML dos FPGAs possam ser manipuladas com maior conforto, inclusive com perspectiva de registro histórico;
- Dotar o SACH de uma interface de gerenciamento acionada pela *Internet*, a ser integrada aos serviços do SACH através de *webservice*;
- Usar primitivas de *Multicast* para aproveitar características de rede física empregada diminuíndo o tráfego na rede.

REFERÊNCIAS

ABOWD, G. D.; MYNATT, E. D. Charting past, present, and future research in ubiquitous computing. **ACM Trans. Comput.-Hum. Interact**, USA, v.7, n.1, p.29–58, 2000. Disponível em: <http://doi.acm.org/10.1145/344949.344988>. Acesso em: Agosto, 2006.

AUGUSTIN, I.; YAMIN, A. C.; SILVA, L. C. da; REAL, R. A.; FRAINER, G.; GEYER, C. F. R. ISAMadapt: abstractions and tools for designing general-purpose pervasive applications: Experiences with Auto-adaptive and Reconfigurable Systems. **Softw. Pract. Exper.**, New York, NY, USA, v.36, n.11‐12, p.1231–1256, 2006.

BROWN, S. D.; ROSE, J. FPGA and CPLD Architectures: A Tutorial. **IEEE Design & Test of Computers**, [S.l.], v.13, n.2, p.42–57, 1996. Disponível em: <http://doi.ieeecomputersociety.org/10.1109/54.500200>. Acesso em: Maio. 2007.

CARDELLI, L. Mobility and Security. In: FOUNDATIONS OF SECURE COMPUTATION, 2000. **Anais...** IOS Press, 2000. p.3–37. (NATO Science Series). Disponível em: <http://www.luca.demon.co.uk/Bibliography.htm>. Acesso em: Setembro, 2006.

CARDOSO, J. M. P.; VESTIAS, M. P. **Architectures and Compilers to Suport Reconfigurable Computing**: ACM Crossroads Student Magazine. Disponível em: <http://acm.org/crossroads/xrds5-3/rcconcept.html>. Acesso em: Abril 2007.

COMPTON, K.; HAUCK, S. Reconfigurable computing: a survey of systems and software. **ACM Computing Surveys**, [S.l.], v.34, n.2, p.171–210, jun 2002.

DEY, A. K. Understanding and Using Context. **Personal Ubiquitous Computing**, London, UK, v.5, n.1, p.4–7, 2001.

DEY, A. K.; ABOWD, G. D. Towards a Better Understanding of Context and Context-Awareness. In: WORKSHOP ON THE WHAT, WHO, WHERE, WHEN, AND HOW OF CONTEXT-AWARENESS, AS PART OF THE 2000 CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, 2000, Netherlands. **Anais...** [S.l.: s.n.], 2000.

DUARTE, F. L. **PHOENIX - Um Framework para Trabalhos em Síntese de Alto Níveis de Circuitos Integrados**. 2006. Mestrado em Ciência da Computação — Universidade Federal de Uberlândia - Minas Gerais. Disponível em:<http://dominiopublico.mec.gov.br/pesquisa/DetalheObraForm.do?select_action=co_obra=42813>.Acesso em: Maio 2007.

ESTRIN, F. Organization of Computer Systems – The Fixed Plus the Variable Structure Computer. In: WESTERN JOINT COMPUTER CONFERENCE, 1960, Hollywood, CA. **Proceedings...** Western Periodicals, 1960. p.33–37.

ESTRIN, G. Reconfigurable Computer Origins: The UCLA Fixed-Plus-Variable (F+V) Structure Computer. **IEEE Annals of the History of Computing**, [S.l.], v.24, n.4, p.3–9, 2002. Disponível em: <http://computer.org/annals/an2002/a4003abs.htm>. Acesso em: Abril. 2007.

GERICOTA, M. G.; ALVES, G. R.; SILVA, M. L.; FERREIRA, J. M. Programmable Logic Devices: A Test Approach for the Imput/Output Blocks and Pad-to-Pin Interconnections. **4th IEEE Latin-American Test Workshop Digest of Papers**, [S.l.], p.72–77, 2003. Disponível em: <http://purl.pt/281/1/artigos/latw2003/latw03.pdf>. Acesso em: Maio 2007.

GUCCIONE, S. A. Reconfigurable Computing at Xilinx. In: DSD, 2001. **Anais...** IEEE Computer Society, 2001. p.102. Disponível em: <http://csdl.computer.org/comp/proceedings/dsd/2001/1239/00/12390102.pdf>. Acesso em: Março. 2007.

GUEDES, G. T. A. **Guia de Consulta Rápida - UML 2**. São Paulo/SP: Novatec Editora Ltda, 2 Edição, 2005.

HARDWARE.NET, G. do. **Tutorial Completo do Apt-Get**. Disponível em: <http://www.guiadohardware.net/tutoriais/tutorial-completo-apt-get/>. Acesso em: Setembro 2007.

KENNEDY, I. Exploiting Redundancy to Speedup Reconfiguration of an FPGA. In: FPL, 2003. **Anais...** Springer, 2003. p.262–271. (Lecture Notes in Computer Science, v.2778). Disponível em: <http://springerlink.metapress.com/openurl.asp?genre=article&issn=0302-9743&volume=2778&spage=262>. Acesso em: Abril 2007.

LYSAGHT, P.; DUNLOP, J. Dynamic Reconfiguration of FPGAs. In: MORE FPGAs: PROCEEDINGS OF THE 1993 INTERNATIONAL WORKSHOP ON FIELD-PROGRAMMABLE LOGIC AND APPLICATIONS, 1993, Oxford, England. **Anais...** [S.l.: s.n.], 1993. p.82–94.

M., I. B. PERVASIVE Computing. IBM System Journal. **IBM System Journal**, [S.l.], v.38, n.4, p.339–388, 1999. Disponível em: <¡http://www.research.ibm.com/>. Acesso em: Maio 2006.

MESQUITA, D. G. **Contribuções para reconfiguração parcial, remota e dinâmica em FPGAs**. 2002. Mestrado em Ciência da Computação — Pontifícia Universidade Católica do Rio Grande do Sul, PUC/RS, Brasil. Disponível em: <http://www.inf.pucrs.br/ moraes/papers/dissertacao_mesquita.pdf>. Acesso em: Junho. 2007.

RECCONFIGURABLE. **Computing Definition**. Disponível em: <http://www.acm.uiuc.edu/sigarch/projects/reconf/report_1.html>. Acesso em: Maio 2007.

SANCHEZ, E.; SIPPER, M.; HAENNI, J.-O.; BEUCHAT, J.-L.; STAUFFER, A.; URIBE, A. P. Static and Dynamic Configurable Systems. **IEEE Trans. Computers**, [S.l.], v.48, n.6, p.556–564, 1999. Disponível em: <http://dblp.uni-trier.de/db/journals/tc/tc48.html#SanchezSHBSP99>. Acesso em: Maio. 2007.

SCHILIT, B.; THEIMER, M. Disseminating Active Map Information to Mobile Hosts. **IEEE Network**, USA, p.22–32, 1994.

SILVA MARTINS, C. A. P. da; MORENO, E. D.; ORDONEZ, J. B. T. M.; CARVALHO, M. B. Computação Reconfigurável: Conceitos, Tendências e Aplicações. **Ciência, Tecnologia e Inovação: Atalhos para o Futoro - Anais**, [S.l.], v.2, p.339–388, 2005. Disponível em: <http://www.ppgee.pucminas.br/gsdc/papers/martins_jai03.pdf>. Acesso em: Maio 2007.

SKLIAROVA, I. **Arquiteturas Reconfiguráveis para Problemas de Otimização Combinatória**. 2004. Monografia de Pós-Graduação Doutorado — Departamento de Eletrônica e Telecomunicações da Universidade de Aveiro, Portugal.

SKLIAROVA, I.; FERRARI, A. B. Design and Implementation of Reconfigurable Processor for Problems of Combinatorial Computations. In: DSD, 2001. **Anais...** IEEE Computer Society, 2001. p.112–119. Disponível em: <http://doi.ieeecomputersociety.org/10.1109/DSD.2001.952250>. Acesso em: Maio. 2007.

SKLIAROVA, I.; FERRARI, A. B. **Introdução à Computação Reconfigurável**. v.2, n.6. Disponível em: <http://www.ieeta.pt/ iouliia/Papers/2003/1_SF_ETSet2003.pdf>. Acesso em: Junho 2007.

STAR Bridge Systems. Disponível em: <http://www.starbridgesystems.com/>. Acesso em: Março 2007.

WANT, R.; HOPPER, A.; FALCAO, V.; GIBBONS, J. The Active Badge Location System. **ACM Transactions on Information Systems**, NY, USA, v.10, n.1, p.91–102, jan 1992.

WEISER; BROWN. The Coming Age of Calm Technology. In: DENNING, P. J.; METCALFE, R. M. (Ed.). **Beyond Calculation**: The Next Fifty Years of Computing, Copernicus, 1997. USA: [s.n.], 1997.

WEISER, M. The computer for the 21st century. **Scientific American**, USA, v.265, n.3, p.94–104, 1991.

XILINX. **Virtex II Pro Platform FPGAs**: Introduction and Overview. Disponível em: <http://www.xilinx.com/partinfo/ds083.pdf/>. Acesso em: Abril 2007.

YAMIN, A. C. **Arquitetura para um Ambiente de Grade Computacional Direcionado, às Aplicações Distribuídas, Móveis e Conscientes do Contexto da Computação Pervasiva**. 2004. Tese de Doutorado — UFRGS, Porto Alegre/RS.

YAMIN, A. C.; AUGUSTIN, I.; BARBOSA, J.; SILVA, L. C. da; REAL, R. A.; FILHO, A. S.; GEYER, C. F. R. EXEHDA: Adaptive Middleware for Building a Pervasive Grid

Environment. **Frontiers in Artificial Intelligence and Applications - Self-Organization and Autonomic Informatics**, [S.l.], v.135, p.203–219, 2005.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo