UNIVERSIDADE CATÓLICA DE PELOTAS
CENTRO POLITÉCNICO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Uma Contribuição à Coordenação na Computação Pervasiva com Aplicações na Área Médica

por
Rodrigo Santos de Souza

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

Orientador: Prof. Dr. Adenauer Corrêa Yamim

DM-2009/2-003

Pelotas, novembro de 2009

A Deus.
A minha esposa, Denise.
Aos meus pais, Pedro e Ceni.

AGRADECIMENTOS

Ao meu orientador Prof. Dr. Adenauer Yamin, pelo inquestionável apoio e orientação nas atividades desenvolvidas neste mestrado.

Agradeço a COPPETEC e ao projeto PERTMED, pelo amparo financeiro.

A todos os colegas do mestrado, em especial aos que ingressaram comigo, pelo agradável companheirismo nos momentos compartilhados.

Agradeço também às pessoas que me deram apoio emocional e compreenderam os momentos de ausência devido ao tempo destinado à realização deste trabalho: meus pais, Pedro e Ceni, meus irmãos e amigos, e, principalmente, minha esposa, Denise.

Por fim, agradeço a Deus pela saúde - física, mental e emocional - que permitiu a trajetória até aqui.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Program Interface*
AVU	Ambiente Virtual do Usuário
BDA	Base de Dados Pervasiva das Aplicações
CIB	*Cell Information Base*
EXEHDA	*Execution Environment for Highly Distributed Applications*
ISAM	Infraestrutura de Suporte às Aplicações Móveis Distribuídas
ISAMpe	ISAM pervasive environment
JADE	*Java Agent Development Framework*
MANET	*Mobile Ad Hoc Networks*
MIT	*Massachusetts Institute of Technology*
PDA	*Personal Digital Assistants*
OX	Objetos EXEHDA
QoS	*Quality of Service*
RMI	*Remote Method Invocation*
RPC	*Remote Procedure Call*
RSSF	Redes de Sensores sem Fio
SQL	*Structured Query Language*
SOA	*Service-oriented Architecture*
TOTA	*Tuples on The Air*
TSox	Espaço de Tuplas do OX
TSvirt	Espaço de Tuplas Virtual
UML	*Unified Modeling Language*
XML	*Extensible Markup Language*

RESUMO

A Computação Pervasiva é um novo paradigma que afirma que o ambiente computacional deve estar disponível para o usuário, a qualquer momento e local. Nesse cenário, as aplicações são amplamente distribuídas, móveis e hospedadas em ambientes inerentemente abertos e dinâmicos. Além disso, os sistemas devem ter mecanismos que promovam a cooperação entre os usuários, bem como entre as aplicações. Essas características reforçam os desafios relacionados às infraestruturas de execução para sistemas distribuídos e acrescentam novos aspectos que precisam ser considerados. Modelos de coordenação existentes empregados em sistemas distribuídos possuem limitações considerando as demandas dos ambientes computacionais modernos, especialmente em relação à escalabilidade, à mobilidade e à desconexão. Este trabalho apresenta o EXEHDA-TS, que é um mecanismo distribuído para a coordenação de aplicações na Computação Pervasiva com controle dinâmico da escalabilidade e dos custos de comunicação. A arquitetura de software proposta permite uma coordenação e troca de informações proativa entre os componentes das aplicações distribuídas, promovendo a escalabilidade em um ambiente móvel. O EXEHDA-TS foi modelado como um serviço e um protótipo foi implementado para ser executado no ambiente gerenciado pelo middleware EXEHDA. O protótipo foi avaliado através de estudos de caso da área médica, considerando os desafios inerentes ao projeto PERTMED, e atendeu às exigências dos cenários focados.

Palavras-chave: Computação pervasiva, computação ubíqua, middleware, modelos de coordenação, espaço de tuplas.

TITLE: “A CONTRIBUTION TO COORDINATION IN PERVASIVE COMPUTING WITH APPLICATIONS IN MEDICAL AREA”

RESUMO

Pervasive Computing is a new paradigm that states that the computational environment must be available to the user at any time and location. In this scenario, the applications are widely distributed, mobile and hosted on inherently open and dynamic environments. Moreover, the systems must have mechanisms that enhance the cooperation among users as well as among applications. These features strengthen the challenges related to the execution infrastructure for distributed systems and add new aspects that need to be considered. Existing coordination models employed in distributed systems are limited considering the demands of modern computing environments, specially regarding scalability, mobility and disconnection. This work presents the EXEHDA-TS, which is a distributed mechanism for coordinating applications in the Pervasive Computing with dynamic control of scalability and communication costs. The proposed software architecture allows a proactive coordination and information exchange among components of distributed applications, promoting scalability in a mobile environment. The EXEHDA-TS was modeled as a service and implemented as a prototype running on the pervasive environment managed by the middleware EXEHDA. The prototype was evaluated through case studies of the medical field, considering the challenges inherent to the project PERTMED, and it accomplished the requirements of the focused scenarios.

Palavras-chave: pervasive computing, ubiquitous computing, middleware, coordinations models, tuple space.

1 INTRODUÇÃO

Este capítulo tem por objetivo caracterizar o esforço de pesquisa realizado, apresentando o tema do trabalho desenvolvido, sua motivação e objetivos, bem como a estrutura deste documento.

Mark Weiser, em seu visionário artigo "*The Computer for the 21st Century*" (WEISER, 1991), introduz uma nova visão de computação, chamada Computação Ubíqua ou Pervasiva. Segundo ele a computação irá evoluir em direção a um cenário no qual não mais iremos notar a sua presença. De maneira autônoma, ela irá satisfazer as nossas necessidades agindo de forma transparente e integrada ao meio.

Esse modelo idealizado por Weiser ainda está distante de aplicações práticas, tendo em vista os produtos e serviços disponíveis atualmente no mercado. Porém, muitos pesquisadores atualmente têm voltado suas atenções para trabalhos que venham a contribuir em direção à consolidação desse cenário, através de pesquisas na área de software e hardware (COSTA; YAMIN; GEYER, 2008). Também têm sido observados recentes avanços tecnológicos na área de micro-eletrônica, redes de comunicação (em especial as sem fio), micro-sensores, somando-se à ampla disponibilidade de equipamentos móveis, entre outros.

Na perspectiva da Computação Pervasiva, o foco das aplicações está no usuário e em suas atividades, provendo o acesso as suas informações e ambiente computacional a partir de qualquer dispositivo e a qualquer momento.

Em direção a tal cenário, os sistemas apresentam-se amplamente distribuídos, possivelmente com alcance global, e pressupõem uma forte integração com o mundo real. As aplicações do usuário são estimuladas a trocarem informações com o meio a fim de prover a adaptação das suas computações ao ambiente e às atividades do usuário. A transparência em relação às questões da infraestrutura também são uma característica do modelo, desobrigando tanto o usuário final quanto o desenvolvedor das aplicações a atuarem diretamente nesses aspectos.

O EXEHDA-TS consiste em um modelo de coordenação escalável com comportamento dinâmico para o middleware EXEHDA. Sua consolidação consiste em um serviço que gerencia um Espaço de Tuplas cujas tuplas são distribuídas entre os nodos do sistema. A proatividade é considerada como elemento central na concepção do EXEHDA-TS, cujo objetivo é prover um modelo dinâmico capaz de reagir às mudanças no Espaço de Tuplas decorrentes das atividades dos componentes das aplicações. Assim, estão disponibilizados mecanismos de manipulação de eventos capazes de notificarem as aplicações sobre informações relevantes à medida que estas são inseridas na estrutura de dados compartilhados. Outrossim, o EXEHDA-TS também concentra esforços no tratamento dos

aspectos decorrentes da mobilidade lógica e física, além de ter compromisso com as desconexões, características intrínsecas da Computação Pervasiva.

1.1 Tema

Este trabalho de pesquisa tem como enfoque principal a apresentação de um modelo de coordenação baseado em Espaço de Tuplas voltado às aplicações da Computação Pervasiva chamado EXEHDA-TS. A consolidação desse modelo consiste na especificação de prototipação de um serviço para o middleware EXEHDA.

Na Computação Pervasiva, os sistemas devem facilitar o compartilhamento de informações entre componentes de aplicações e dispositivos, a fim de estimular o trabalho cooperativo entre usuários e entre processos. Para promover essas atividades, faz-se necessária a utilização de modelos de coordenação adequados às características dos sistemas computacionais modernos, que atuem de forma a interfacear as comunicações entre os componentes de aplicações de modo a abstrair aspectos característicos da infraestrutura física e lógica, promovendo a troca de informações.

A Computação Pervasiva caracteriza-se por uma elevada distribuição, heterogeneidade de hardware e de software, mobilidade física e lógica; a implementação de modelos de coordenação neste cenário introduz grandes desafios que devem ser vencidos. Esta dissertação discute tais desafios e propõe estratégias para superá-los, explorando as características do middleware EXEHDA.

1.2 Contexto de Pesquisa

O objetivo desta seção é caracterizar o contexto no qual este trabalho de dissertação foi desenvolvido. Nesse sentido, dois projetos ofereceram suporte ao desenvolvimento da pesquisa realizada.

1.2.1 Projeto PERTMED

PERTMED é um projeto financiado pela FINEP (Financiadora de Estudos e Projetos) e executado em conjunto pela UFSM (Universidade Federal de Santa Maria), UFPEL (Universidade Federal de Pelotas) e UCPEL (Universidade Católica de Pelotas). O objetivo do projeto é conceber um sistema de TeleMedicina para disponibilizar informações médicas de maneira pervasiva. Está previsto o desenvolvimento de um sistema de gerenciamento baseado em serviços e um conjunto de aplicações, servindo como elo entre os dispositivos móveis e o sistema informatizado do hospital. Esses dispositivos móveis devem servir como interface de comunicação paciente-médico ou médico-médico enquanto que a rede fixa do hospital como infraestrutura para o gerenciamento das aplicações móveis distribuídas (PERTMED, 2008).

O projeto PERTMED prevê a utilização de telefones celulares e *smartphones* para disponibilizar ao médico informações eletrônicas dos seus pacientes em qualquer lugar que ele esteja, auxiliando na tomada de decisão, principalmente em situações de emergência.

Na perspectiva da Computação Pervasiva, um sistema informatizado de saúde contempla um ambiente computacional heterogêneo e dinâmico em que os dispositivos são integrados ao meio físico, captando informações e transmitindo aos mecanismos de

gerência. No PERTMED, as aplicações com o suporte de tais mecanismos de gerência tomam decisões em relação às situações detectadas, encaminhando as informações aos dispositivos das pessoas interessadas.

1.2.2 Middleware EXEHDA

O EXEHDA (*Execution Environment for Highly Distributed Applications*) é um middleware voltado para o desenvolvimento e execução de aplicações da Computação Pervasiva. Sua concepção iniciou no contexto do projeto ISAM (Infra-Estrutura de Suporte às Aplicações Móveis Distribuídas) (ISAM, 2008), com o intuito de prover suporte ao gerenciamento de um ambiente pervasivo. As aplicações da Computação Pervasiva são largamente distribuídas, móveis e com adaptação ao contexto em que estão inseridas. Para satisfazer as demandas oriundas dessas aplicações, o EXEHDA é organizado através de serviços e adaptável ao contexto. O middleware provê uma estrutura de software que permite a disponibilização de dados e aplicações a partir de qualquer lugar e dispositivo a todo o tempo (LOPES; PILLA; YAMIN, 2007).

Para atender a grande variação em relação aos recursos disponíveis, inerente à Computação Pervasiva, o EXEHDA é constituído por um núcleo mínimo e um conjunto de serviços carregados sob demanda. Na versão atual do EXEHDA, esses serviços são organizados em quatro grandes grupos: execução distribuída, adaptação e reconhecimento do contexto, comunicação e acesso pervasivo.

1.3 Motivação

Na Computação Pervasiva, as infraestruturas de hardware e software precisam dar suporte à comunicação e à coordenação, a fim de promover a execução de complexas aplicações distribuídas para, por exemplo, apoiar as atividades cooperativas entre usuários, acompanhar e controlar o ambiente, além de proporcionar a interação com o mundo físico. Apesar de vários projetos estarem sendo desenvolvidos nesse sentido, ainda existe muito trabalho de pesquisa a ser explorado nessa área (CABRI et al., 2007).

Durante os estudos relacionados a esta dissertação de mestrado, foi realizada uma revisão dos conceitos envolvidos com o middleware EXEHDA (YAMIN, 2004). Como um dos resultados desta revisão, identificou-se a necessidade de aprimorar o modelo de coordenação baseado em Espaço de Tuplas previsto em sua concepção. Tal constatação foi o principal aspecto que motivou o aprofundamento de um estudo em relação aos modelos de coordenação focados na Computação Pervasiva e à consolidação de um serviço para o EXEHDA que atenda a essas demandas, chamado EXEHDA-TS.

Esses aspectos foram potencializados com a especificação do projeto PERTMED, na qual foi identificada a necessidade de uma infraestrutura de software capaz de processar demandas da área médica em qualquer lugar e a todo o tempo.

O conjunto destes fatores motivou os esforços de pesquisa e desenvolvimento do EXEHDA-TS, que consiste em um serviço para o middleware EXEHDA, voltado à gerência de um modelo de coordenação baseado em Espaço de Tuplas distribuído, escalável e com atuação proativa, direcionado às aplicações da Computação Pervasiva.

1.4 Objetivos

O objetivo geral desta dissertação é conceber um modelo voltado à coordenação desacoplada de componentes de software com suporte à reatividade focado nas demandas da Computação Pervasiva.

O esforço de estudo e pesquisa correspondente à concepção do EXEHDA-TS contemplou os seguintes objetivos:

- caracterizar o contexto atual das pesquisas relativas aos ambientes de execução voltados à Computação Pervasiva;
- revisar os princípios que norteiam a concepção do EXEHDA enquanto middleware para Computação Pervasiva;
- avaliar os requisitos de um modelo para a coordenação de aplicações baseadas em componentes em ambientes pervasivos;
- sistematizar as funcionalidades necessárias à concretização de um modelo de coordenação para a Computação Pervasiva compromissado com aspectos como distribuição, mobilidade e desconexão;
- otimizar custos de comunicação do modelo proposto através da utilização de modos de operação passíveis de serem definidas pelo programador;
- potencializar a escalabilidade do modelo proposto, promovendo a distribuição das tarefas de gerenciamento;
- difundir o conhecimento pertinente à área de suporte à execução de aplicações na Computação Pervasiva, sobretudo no que diz respeito aos aspectos de coordenação dos componentes das aplicações;
- fornecer subsídios para a elaboração de relatórios, artigos e trabalhos futuros relacionados ao tema pesquisado, no âmbito do grupo de pesquisa.

1.5 Estrutura do Texto

Este texto é constituído por seis capítulos. No capítulo 2, é apresentado o escopo deste trabalho; nele, são explorados conceitos básicos intimamente relacionados a esta dissertação. São discutidos os aspectos fundamentais da Computação Pervasiva e seu contexto atual. Na sequência, são apresentados os modelos de coordenação utilizados em sistemas distribuídos tradicionais, com atenção especial ao modelo baseado em Espaço de Tuplas. Por fim, o capítulo 2 traz alguns trabalhos relacionados ao modelo de coordenação proposto neste trabalho de pesquisa.

O middleware EXEHDA, que serve de base fundamental para a estruturação do modelo apresentado nesta dissertação de mestrado, teve uma atenção especial devido ao estudo aprofundado realizado. No capítulo 3, é apresentado um resumo das funcionalidades básicas modeladas no middleware, bem como sua arquitetura de software.

O capítulo 4 traz a modelagem do EXEHDA-TS, sendo abordados os principais desafios de pesquisa enfrentados na concepção de um modelo de Espaço de Tuplas que satisfaça as demandas das aplicações na Computação Pervasiva.

Os aspectos envolvidos com a prototipação do EXEHDA-TS, assim como a análise do comportamento do serviço desenvolvido através de cenários da área médica, foram tratados no capítulo 5. Por fim, o capítulo 6 apresenta as considerações finais a respeito do trabalho.

2 ESCOPO DE PESQUISA DO TRABALHO

O capítulo atual aborda temas de fundo pertinentes à pesquisa realizada. Na primeira seção, é feita uma revisão conceitual em relação à Computação Pervasiva, sendo também apresentada uma breve discussão sobre alguns dos principais projetos identificados na área.

Na segunda seção, são discutidos os modelos de coordenação, principalmente os aspectos referentes a Espaços de Tuplas. O texto aponta as principais características de cada modelo selecionado, destacando os pontos positivos e negativos envolvidos considerando as demandas da Computação Pevasiva.

Por fim, são apresentados os principais trabalhos relacionados que foram identificados durante o estudo, sendo discutidas as características de suas modelagens.

2.1 Fundamentos da Computação Pervasiva

"As tecnologias mais profundas são aquelas que desaparecem. Elas se integram na vida cotidiana até se tornarem indistinguíveis da mesma". Com essa afirmação, Mark Weiser começa o seu artigo (WEISER, 1991), no qual introduz a Computação Ubíqua. Na sua visão, esta caracteriza-se pela elevada transparência em relação aos aspectos computacionais, sendo integrada totalmente ao cotidiano das pessoas e ao meio. Esse cenário é considerado futurista até mesmo nos dias atuais.

Deste modo, o termo Computação Pervasiva tem sido utilizado por alguns autores como sendo uma etapa intermediária que deve ser consolidada para que se possa atingir a Computação Ubíqua idealizada por Weiser (ROBINSON; VOGT; WAGEALLA, 2005). Em outras palavras, a principal diferença entre os dois conceitos é que a Computação Pervasiva surge como uma visão *bottom-up*, tendo como ponto de partida as tecnologias existentes, enquanto a Computação Ubíqua é visão *top-down* (vide figura 2.1). Em muitos casos ambos os termos têm sido utilizados como sinônimos devido ao fato de que o propósito final é o mesmo. Neste trabalho, optou-se por utilizar Computação Pervasiva para se referir a este novo paradigma computacional (COSTA, 2008).

O termo Computação Pervasiva ficou associado à IBM devido à edição intitulada *Pervasive Computing* do *IBM Systems Journal* (IBM, 1999), em que foram apresentados os aspectos promissores da Computação Pervasiva. De maneira sintética, o termo prevê a mobilidade do usuário com ou sem o seu equipamento, garantindo o acesso ao seu ambiente computacional (dados e aplicativos) independente de localização, tempo e dispositivo utilizado.

Na perspectiva da Computação Pervasiva, os sistemas são amplamente dis-

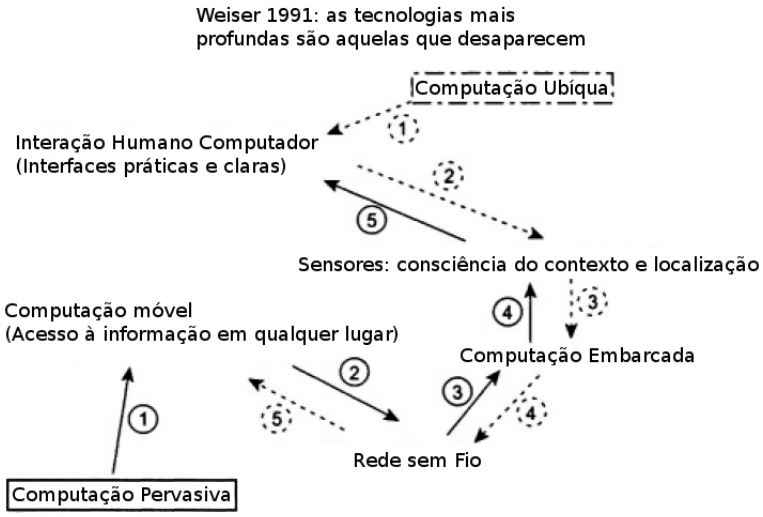

Figura 2.1: Visões da Computação Ubíqua e Pervasiva (adaptado de (ROBINSON; VOGT; WAGEALLA, 2005))

tribuídos e pressupõe uma forte integração com o mundo real, mantendo uma elevada transparência nas questões referentes à infraestrutura, tendo como foco o usuário e suas atividades.

2.1.1 Consolidação da Computação Pervasiva

A Computação Pervasiva tem sido considerada por muitos autores como a convergência de diversas áreas da Ciência da Computação, entre as quais uma das mais evidentes é a área de sistemas distribuídos. Porém, na premissa clássica da computação distribuída, busca-se liberar o programador das preocupações em relação ao meio no qual ocorre o processamento, enquanto que, na Computação Pervasiva, o programador deve considerar as informações de contexto no desenvolvimento das aplicações, criando sistemas capazes de se adaptarem ao ambiente operacional.

Para superar as limitações dos sistemas distribuídos atuais, (GRIMM et al., 2004) destacaram três requisitos necessários para um sistema de suporte às aplicações pervasivas:

- as pessoas se deslocam através do mundo físico, quer transportando os seus próprios dispositivos portáteis ou trocando de dispositivos. Assim a localização das aplicações e o contexto de execução muda constantemente. Então, sistemas para Computação Pervasiva devem administrar mudanças contextuais e disponibilizá-las às aplicações, para que estas possam desenvolver suas próprias estratégias para o tratamento das mudanças;

- os usuários utilizam seu equipamentos em diferentes contextos. Os sistemas devem suportar variações nas suas composições e não assumir o ambiente como algo estático. Nesse sentido, os dispositivos e aplicações devem funcionar de maneira integrada, adaptando-se às constantes mudanças com o mínimo de intervenção dos usuários;

- os sistemas para Computação Pervasiva devem facilitar o compartilhamento de

informações entre as aplicações e dispositivos, promovendo atividades colaborativas entre os usuários ou entre aplicativos.

Para o desenvolvimento de aplicações na Computação Pervasiva, é necessária uma infraestrutura de software que eleve o nível da programação a ponto de abstrair as características intrínsecas do sistema, de modo que a participação humana no processo seja reduzida.

Desse modo, a dificuldade consiste no desenvolvimento de aplicativos que continuamente se adaptem ao ambiente, e assim, mantenham seu funcionamento à medida que as pessoas se movimentam no mundo físico ou trocam de dispositivo (GRIMM et al., 2004). Além disso, é necessário fazer com que o ambiente de trabalho do usuário o acompanhe caso o hardware não se mova com ele. Sendo assim, os sistemas de gerência devem processar as mudanças do meio e disponibilizar informações contextuais às aplicações para que estas façam o seu próprio tratamento do contexto.

Uma característica predominante na Computação Pervasiva é a grande heterogeneidade de hardware. Por isso, a adaptação dinâmica tem sido um tema bastante discutido na Computação Pervasiva, e é utilizada quando existe uma diferença muito grande entre o fornecimento de recursos e a demanda por eles. Através da adaptação é feita a seleção de determinados aspectos do contexto de uso, como localização, tempo e atividades do usuário, permitindo o desenvolvimento de aplicações sensíveis ao contexto (*context-aware*) (HENRICKSEN; INDULSKA, 2005).

O amadurecimento dos conceitos em direção à Computação Pervasiva pode ser sintetizado através da figura 2.2. Segundo a visão de (YAMIN, 2004), a consolidação do complexo cenário da Computação Pervasiva pode ser obtida através da integração de três áreas da computação: Computação Móvel, Computação em Grade e a Computação Consciente ao Contexto.

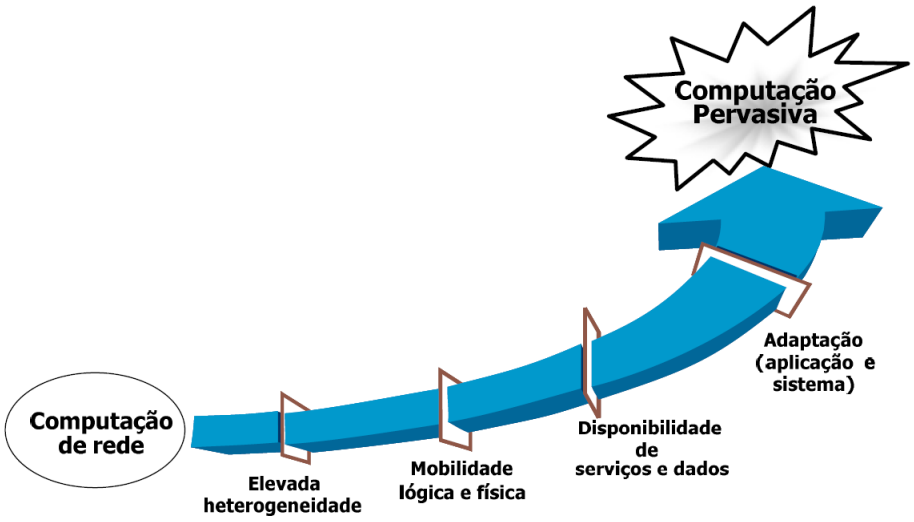

Figura 2.2: Consolidação da Computação Pervasiva (YAMIN, 2004)

Para satisfazer as demandas das aplicações pervasivas, é necessário um middleware que sirva de interface entre as aplicações do usuário final e os diversos dispositivos existentes na rede (COSTA; YAMIN; GEYER, 2008). Além dos novos desafios que surgem com o advento da Computação Pervasiva, também todas as questões envolvidas

com os sistemas distribuídos tradicionais são herdadas. Nesse sentido, a utilização de middleware já vinha sendo apontada há algum tempo como uma solução para os problemas envolvidos com os sistemas distribuídos.

Um middleware para a Computação Pervasiva também deve permitir ao usuário o acesso ao seu ambiente computacional (dados e aplicativos) em qualquer lugar, a qualquer momento e independente de dispositivo. Assim, é necessário que componentes de software tenham a capacidade de migrar de um dispositivo para outro conforme o usuário troca de equipamento. Alguns autores dão o nome de agentes aos componentes de software que têm capacidade de migração além de alguma autonomia computacional.

A adaptação é fundamental para a visão da Computação Pervasiva, e envolve a percepção do contexto (*context awareness* ou consciência de contexto) e o próprio ajuste do sistema baseado na informação percebida (gerência do contexto).

2.1.2 Projetos para Computação Pervasiva

As aplicações pervasivas necessitam de um middleware como interface entre elas e os dispositivos, cujo objetivo é esconder a complexidade do ambiente, isolando as aplicações da gerência explícita dos aspectos da infraestrutura. Com o middleware é possível gerenciar o problema da heterogeneidade de arquiteturas, de sistemas operacionais, de tecnologias de rede e até mesmo de linguagens de programação.

Dentre os ambientes para computação pervasiva, cabe destacar o projeto ISAM (Infraestrutura de Suporte às Aplicações Móveis) (ISAM, 2008). A ideia desse projeto é construir uma infraestrutura de Computação Pervasiva, integrando uma linguagem de programação e um middleware para suportar sua execução. Mais detalhes sobre o ISAM são apresentados no capítulo 3.

Projeto Aura

O Projeto Aura (GARLAN D.; SIEWIOREK, 2002; AURA, 2008) está sendo desenvolvido na Universidade de Carnegie Mellon desde o ano 2000. O conceito do projeto cria a visão de uma "Aura de Informações Pessoais" através do desenvolvimento de arquiteturas, algoritmos, interfaces e demais técnicas necessárias. Aura é composto por diversos sistemas, sendo que alguns já estavam em desenvolvimento antes do início do projeto e foram incorporados ao mesmo. Os sistemas são os seguintes: *Odyssey*, que suporta adaptação e monitoramento de recursos; *Coda*, que provê acesso a arquivos de forma distribuída e adaptável à largura de banda e conectividade; *Spectra*, que é um mecanismo adaptativo de execução remota baseado em contextos; e *Prisma*, que captura e gerencia a intenção do usuário.

Projeto Gaia

O Projeto Gaia (ROMáN et al., 2002; GAIA, 2008) consiste em um middleware distribuído que coordena entidades de software em redes de dispositivos heterogêneos. A ideia do projeto é criar uma espécie de sistema operacional (Gaia OS) que abstrai os espaços e recursos como uma única entidade programável. O Gaia OS provê os principais serviços de um sistema operacional: execução de programas, operações de E/S, sistema de arquivos, comunicação, detecção de erros e alocação de recursos. O projeto foi concebido na Universidade de Illinois em Urbana-Champaign. A principal diferença em relação ao projeto Aura é que Gaia enfatiza a programação em espaços (*Active Spaces*)

e as aplicações utilizam os recursos neles disponíveis.

Projeto Endeavour

Projeto da Universidade da Califórnia em Berkeley desde 1999, Endeavour (ENDEAVOUR, 2008) propõe desenvolver a especificação, projeto e prototipação de um "utilitário de informação" em escala planetária, auto-organizável e adaptativo. O projeto cria um ambiente pervasivo em que os componentes se movimentam na infraestrutura, se adaptando aos equipamentos utilizados e cooperando entre si. No projeto tais componentes pervasivos são denominados de *fluid* software, isto é, código com a habilidade de adaptação e distribuição de forma automática pela infraestrutura de hardware. O sistema pode ser composto de componentes pré-existentes de software e hardware para satisfazer requisições de serviços (SAHA; MUKHERJEE, 2003).

Projeto Oxygen

O projeto Oxygen (OXYGEN, 2008) está sendo desenvolvido no Massachusetts Institute of Technology (MIT), e defende que, no futuro, a computação será disponível gratuitamente em todo o lugar, como oxigênio no ar. O objetivo é prover computação pervasiva, centrada no usuário através de dispositivos móveis e estacionários conectados por uma rede autoconfigurável. O projeto usa dispositivos baseados em computação embarcada e PDAs. Toda a interação com o usuário é feita por visão e voz, ao invés de teclado e mouse. O software é adaptável e funciona com a menor intervenção possível do usuário.

A infraestrutura de software do Oxygen é constituída dos seguintes componentes: *Pebbles*, que são partes de software independentes de plataforma; *Goals*, responsável pela criação automática de componentes que atendem a determinados requisitos do sistema; *MetaGlue*, que provê comunicação e descoberta de serviços, possibilitando interação dos usuários com software e dados; *Core*,que permite estruturar as aplicações como grafos de componentes interconectados, permitindo a depuração e teste de componentes; *Click*, que é um roteador de software modular; *Suds*, que disponibiliza um mecanismo para automaticamente atualizar código em um banco de dados distribuído e orientado a objetos; e *IOA*, que é uma linguagem e um conjunto de ferramentas gerados para permitir a implementação de sistemas distribuídos confiáveis (SAHA; MUKHERJEE, 2003).

Existem diversas iniciativas de middlewares para a Computação Pervasiva; alguns mereceram destaque durante o estudo realizado. Uma das características comuns entre os middlewares apresentados nesta seção é o esforço despendido na intenção de atender o maior número possível de tipos de aplicações e não apenas a casos específicos. Outro aspecto pertinente é que todos esses projetos apresentam preocupações especiais com questões envolvendo mobilidade e desconexão, o que aponta o tipo de ambiente que deve ser suportado pelo modelo apresentado neste trabalho.

2.2 Modelos de Coordenação em Sistemas Distribuídos

Os modelos de coordenação sistematizam mecanismos e políticas segundo as quais um grupo de atores pode organizar suas atividades em conjunto. Tarefas de coordenação vão além das aplicações da Ciência da Computação, estando presentes nas mais diversas ciências. No entanto o foco deste trabalho está direcionado às atividades de coordenação de aplicações colaborativas em ambientes computacionais distribuídos.

Modelos baseados em coordenação caracterizam-se pela separação entre computação e coordenação (TANENBAUM; STEEN, 2007). Em sistemas distribuídos, a computação corresponde aos processos que são executados, cada um com suas atividades computacionais específicas, que podem ser independentes ou cooperativas. A coordenação, por sua vez, consiste em um sistema que manipula a comunicação e cooperação entre os diferentes processos envolvidos. Assim, em sistemas distribuídos baseados em coordenação, um programa é organizado a partir dos processos que realizam as computações, e um modelo de coordenação que promove a cooperação entre os componentes da aplicação (vide figura 2.3).

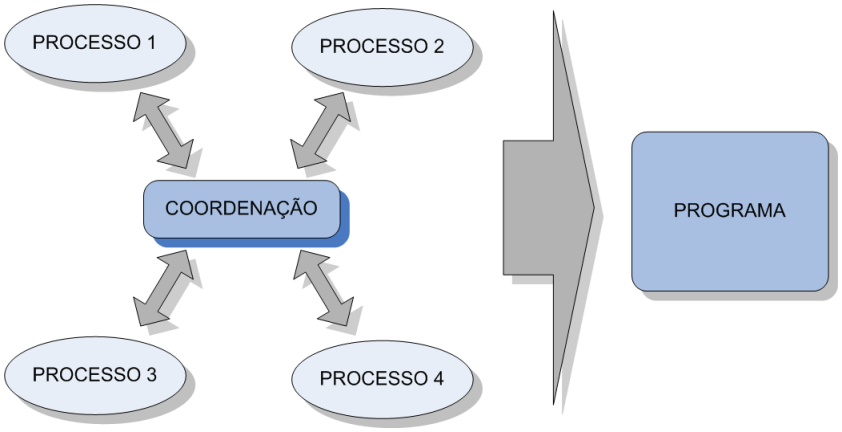

Figura 2.3: Modelo Baseado em Coordenação

Em sistemas baseados em coordenação, o foco está no modo como ocorre a interação entre os processos (TANENBAUM; STEEN, 2007). Assim, a coordenação exige a definição da forma de comunicação entre os processos, bem como da forma de sincronização de suas atividades. Exemplos nesse sentido seriam opções pelo uso de troca de mensagens, semáforos, etc. (MAMEI; ZAMBONELLI, 2006a).

Os modelos de coordenação de uso consolidado podem ser classificados em três grandes grupos (MAMEI; ZAMBONELLI, 2006a): Modelos de Coordenação Direta, Modelos de Coordenação Baseados em Eventos e Modelo de Coordenação de Espaço de Dados Compartilhados. Esses modelos são resumidos nas seções seguintes, em que suas características são destacadas.

2.2.1 Modelos de Coordenação Direta

Nesta classe de coordenação, os processos distribuídos coordenam suas atividades com os demais através da comunicação direta e explícita. Os parceiros de comunicação devem conhecer-se previamente para que a comunicação entre ambos seja possível. As implementações desse modelo de coordenação geralmente se subdividem em duas abordagens (MAMEI; ZAMBONELLI, 2006a): (*i*) passagem de mensagem e (*ii*) modo cliente-servidor.

Na primeira abordagem, a coordenação se dá através da troca de mensagens de textos formatadas. Os processos que recebem tais mensagens devem ter a capacidade de compreender o conteúdo sintático e semântico das mensagens para decidir as ações que devem ser tomadas. Um exemplo de estrutura que pode ser adotada nos textos trocados durante os processos de comunicação é o padrão XML. Como exemplo desta abordagem,

pode-se indicar o JADE (JADE, 2008).

No modo cliente-servidor, a comunicação se dá através de operações de mais alto nível que na abordagem anterior. Nesse caso, os processos geralmente adotam mecanismos de chamada remota de procedimentos (RPC). Nos middlewares que utilizam orientação a objetos, isso pode ser implementado através da chamada de métodos de objetos instanciados remotamente. Assim, um processo exporta uma interface de modo a permitir que os demais invoquem métodos dessa interface desconsiderando a implementação atual do tal método. Um exemplo de tal abordagem seria o Jini (JINI.ORG, 2008), um middleware que utiliza a tecnologia SOA (*Service-oriented Architecture*) e explora o recurso RMI da linguagem JAVA.

Nesse modelo de coordenação, os processos devem conhecer antecipadamente o endereço de rede e a porta dos parceiros de comunicação. Desse modo, existe um acoplamento entre ambos, o que implica a coexistência no tempo e um conhecimento explicito das suas localidades.

2.2.2 Modelos de Coordenação Baseados em Eventos

Um evento de software é uma porção de dados gerada para indicar a ocorrência de algo em um sistema, por exemplo, a chegada de um datagrama através da rede ou um aumento brusco de temperatura detectado por um sensor (MAMEI; ZAMBONELLI, 2006a). Essas ocorrências podem ser modeladas através de eventos, e as informações sobre o que aconteceu podem ser incluídas como atributos dos acontecimentos em si.

A programação baseada em eventos é uma prática bastante utilizada por desenvolvedores de sistemas; consiste em um processo de especificação “quando isso acontece, fazer isso”. Em programação gráfica, isso é particularmente evidente: quando o mouse se desloca, o cursor move com ele; quando o usuário clicar neste botão, executa este procedimento.

Em modelo de coordenação baseada em eventos, são utilizados mecanismos de publicação e subscrição (*publish/subscribe*) para promover a interação entre processos distribuídos (MAMEI; ZAMBONELLI, 2006a). Nesse modelo, existem duas classes de processos: os que produzem eventos (*event publishers*) através da sua publicação e os consumidores desses eventos (*event subscribers*) que utilizam mecanismos de subscrição.

Os *event subscribers* fornecem uma descrição do tipo de evento do qual querem ser notificados, e os associam a procedimentos que devem tratá-los. Então, conforme os *event publishers* publicam seus eventos, estes são repassadas aos *event subscribers* correspondentes. Todo o processo de publicação e subscrição é gerenciado através do sistema *publish/subscribe*, ou seja, pelo middleware

Esse modelo de coordenação promove um desacoplamento referencial entre os processos, pois toda a comunicação é gerenciada pelo *middleware*. Porém, existe entre os participantes da coordenação um acoplamento temporal, o que significa que os envolvidos na comunicação precisam estar executando ao mesmo tempo para que possam ser notificados imediatamente após um evento ter sido publicado.

2.2.3 Modelo de Coordenação de Espaço de Dados Compartilhados

O modelo de espaço de dados compartilhado abordado neste texto é o Espaço de Tuplas. Esse conceito foi introduzido pelo modelo Linda (CARRIERO; GELERNTER, 1989), tendo suas diferentes implementações largo emprego na atualidade. Espaço de

Tuplas consiste de uma memória associativa independente, compartilhada entre todos os nodos do sistema (YAMIN, 2004). Essa memória pode ser considerada como uma espécie de repositório para as estruturas denominadas tuplas (ver figura 2.4). A comunicação entre processos se dá através da inserção e leitura dessas tuplas no espaço compartilhado.

Figura 2.4: Abstração de um Espaço de Tuplas (ANDRADE FIGUEIREDO, 2003)

As tuplas podem ser definidas como um conjunto de dados estruturados que têm valores associados a tipos, tal como $< \text{“}foo'', 12, 53.2 >$. Os processos podem colocar qualquer tipo de registro no Espaço de Tuplas. Na versão original do Linda, as operações básicas são as seguintes: `out`, `in` e `rd`.

Uma tupla `t` é criada e inserida em um espaço tuplas através de uma primitiva `out(t)`; os processos de leitura são feitos de forma associativa. Dessa forma, o processo que deseja fazer uma leitura deve informar o modelo da tupla desejada através de um *template*. Este nada mais é do que uma tupla que possui alguns campos com valores informados explicitamente, enquanto outros apenas são associados a tipos, como $< \text{“}foo'', ?integer, ?float >$. A tupla que combinar com essa especificação é repassada para o processo solicitante.

As primitivas utilizadas para ler ou extrair as tuplas são `in(p)` e `rd(p)`, respectivamente. Essas primitivas recebem um *template* (ou *pattern*) `p` como parâmetro e retornam a primeira tupla correspondente. Uma vez que um espaço de tuplas é um conjunto desestruturado, a escolha entre várias tuplas correspondentes é arbitrária e depende da implementação. As operações `in(p)` e `rd(p)` são bloqueantes; nesse caso, se nenhuma tupla que combine com o *template* for encontrada, o processo é suspenso até que uma seja inserida. Evidentemente, um tempo limite deve ser definido para isso. Como uma alternativa a tais operações, Linda introduz duas outras primitivas: `inp(p)` e `rdp(p)`, que têm funções semelhantes às anteriores; porém, se não for localizada nenhuma tupla com as características desejadas, a operação retorna um valor `null`.

Cada processo pode inserir ou ler tuplas nesse espaço a qualquer momento, caracterizando, assim, um assincronismo na comunicação. O tempo de vida de uma tupla é independente do processo que as gerou. O espaço de tuplas possibilita a comunicação entre processos mesmo que eles não estejam operantes ao mesmo tempo. Dessa maneira, a coordenação ganha um desacoplamento temporal, além do desacoplamento referencial característico do sistema *publish/subscribe*. Esses dois desacoplamentos favorecem

a utilização do modelo de espaço de tuplas em ambientes móveis e dinâmicos, em que as desconexões são frequentes (YAMIN, 2004; AUGUSTIN, 2004), característica intrínseca à Computação Pervasiva.

Basicamente, uma das principais diferenças entre o modelo de Espaço de Tuplas e o modelo *publish/subscribe* é a existência de uma área de armazenamento temporário para as informações que são trocadas entre as entidades envolvidas na comunicação.

Espaço de Tuplas Centralizado Versus Distribuído

O modelo básico de Espaço de Tupla prevê a utilização de servidores centralizados como repositório das tuplas. Nessa situação, todas as tuplas são armazenadas em um mesmo local, o que simplifica as tarefas de gerência do middleware. Tal visão não satisfaz os requisitos da Computação Pervasiva, pois tem sérios problemas de escalabilidade e inviabiliza a comunicação em dispositivos com desconexões constantes. Como exemplos desse modelo têm-se o Linda, TSpaces (IBM, 2008) e JavaSpaces (JAVASPACES, 2008).

Uma alternativa a servidores centralizados é a utilização de espaços de tuplas distribuídos entre as máquinas participantes da comunicação. Na Computação Pervasiva, a distribuição das computações é preponderante; por isso, um Espaço de Tuplas para atender ás suas necessidades tem a distribuição como requisito básico. Segundo (TANENBAUM; STEEN, 2007) os espaços de tuplas distribuídos podem ser divididos em três abordagens:

- abordagem estática: nessa abordagem, existem basicamente duas possibilidades: (*i*) os Espaços de Tuplas são replicados em todos os dispositivos ou (*ii*) os Espaços de Tuplas existentes em cada máquina são distintos. Na primeira situação, quando forem realizadas operações de escrita ou remoção de tuplas, todos os espaços devem ser atualizados através de um envio de *broadcast*, quando este for permitido. Qualquer operação que modifique o Espaço de Tuplas deve executar um procedimento para a atualização de todos os Espaços de Tuplas do sistema. Assim, as operações de leitura são realizadas sempre localmente. No segundo caso, a condição é o inverso. As operações de escrita é que são feitas de forma local, enquanto as de leitura se dão pelo envio de uma tupla de gabarito através de *broadcast*. Desse modo, uma resposta é retornada se algum receptor tiver alguma tupla que combine com o gabarito. Cada dispositivo ou processo em execução tem seu próprio Espaço de Tuplas, que é disponibilizado aos demais nodos formando um Espaço de Tuplas compartilhado entre todos os processos em execução (vide figuras 2.5 e 2.6). Esse procedimento é mais simples que o anterior, pois as operações de sincronização são feitas localmente. Tais soluções apresentam problemas de escalabilidade à medida que crescer a quantidade de tuplas e o tamanho da rede;

- abordagem dinâmica: essa abordagem prevê a utilização de regras de replicação que podem ser predefinidas. O objetivo das regras é dar maior flexibilidade à maneira como são feitas as operações sobre o Espaço de Tuplas. Conforme são feitas chamadas a tais operações, um manipulador localiza as regras específicas que devem ser seguidas para então efetivamente realizar as operações segundo essas regas. Em relação à abordagem estática, esta possibilita melhorar o desempenho do Espaço de Tuplas ou prover uma melhor disponibilidade dos dados; para isso, basta que sejam utilizadas regras adequadas a cada situação;

- abordagem adaptativa: nesse caso, as operações sobre o Espaço de Tuplas são regidas por políticas dinâmicas, sendo que as regras podem ser manipuladas de forma automática pelo middleware, inclusive sendo adaptadas dinamicamente segundo o comportamento do sistema. Condições como largura de banda, latência e utilização da memória e demandas das aplicações podem ser frequentemente analisadas e utilizadas na definição e na adaptação das regras. Em uma abordagem desse tipo é possível ter um sistemas mais adequado às necessidades das aplicações, porém, a carga computacional que possa ser necessária merece atenção especial devido à "inteligência" que deve ser dada ao middleware.

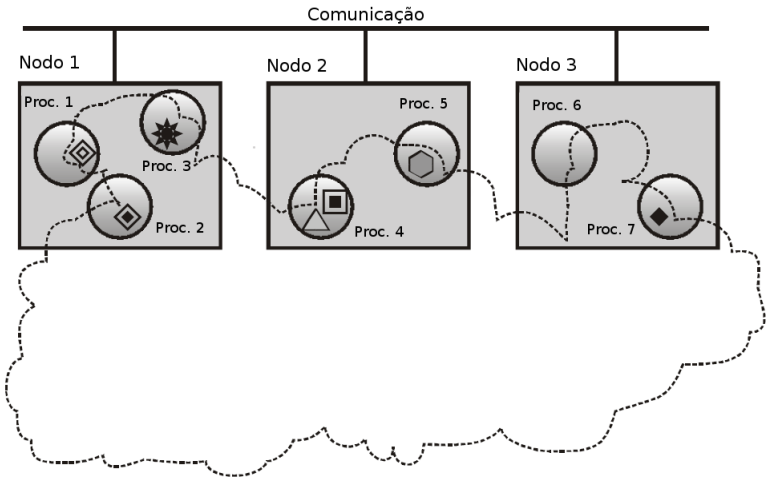

Figura 2.5: Espaço de Tuplas Distribuído Entre os Nodos (ANDRADE FIGUEIREDO, 2003)

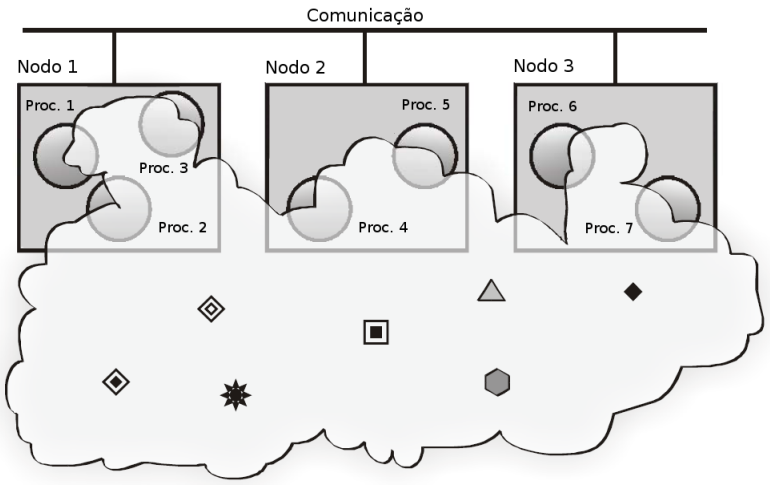

Figura 2.6: Abstração Promovida pelo Espaço de Tuplas Distribuído (ANDRADE FIGUEIREDO, 2003)

As implementações de Espaços de Tuplas podem estender o modelo tradicional adicionando funcionalidades para tratar problemas específicos de um determinado grupo de aplicações. Um exemplo disso é a transição do modelo de armazenamento centralizado

Tabela 2.1: Grau de acoplamento dos modelos de coordenação

Modelo	Referencial	Temporal
Coordenação Direta	acoplado	acoplado
Espaço de Dados Compartilhados	desacoplado	desacoplado
Orientado a Eventos	desacoplado	acoplado

para um modelo distribuído, devido principalmente às grandes limitações que o primeiro modelo oferece às aplicações em ambientes dotados de equipamentos móveis.

Uma comparação em relação ao grau de acoplamento dos três modelos de coordenação apresentados pode ser observado na tabela 2.1.

2.3 Trabalhos Relacionados

Esta seção apresenta alguns projetos que foram selecionados por possuírem características intimamente relacionadas à proposta deste trabalho. Todos implementam modelos de coordenação com base em Espaço de Tuplas distribuído, com foco em aplicações móveis e com preocupações em relação à escalabilidade dos sistemas, aspectos centrais nas aplicações da Computação Pervasiva.

2.3.1 Middleware Lime

Lime (*Linda in a Mobile Environment*) (LIME, 2008; MURPHY; PICCO; ROMAN, 2006) é um middleware desenvolvido em Java, que oferece uma camada de coordenação direcionada às demandas das aplicações voltadas aos ambientes móveis *ad hoc*. A principal motivação para o desenvolvimento do Lime foi a observação do autor referente às complexidades existentes no projeto de aplicações em ambientes com mobilidade. Em tais ambientes, as mudanças na disponibilidade de recursos são constantes (tanto dos dados como dos elementos computacionais), e ocorrem conforme os dispositivos se deslocam no ambiente físico ou quando as aplicações migram de um dispositivo para outro.

O middleware Lime refina o modelo original de Espaço de Tuplas proposto por Linda (CARRIERO; GELERNTER, 1989) com o objetivo de prover um modelo de coordenação para ser utilizado em ambientes móveis. Em Linda, a comunicação entre os processos se dá através da leitura e escrita de dados em um Espaço de Tuplas persistente e globalmente compartilhado por todas as aplicações. O Lime adapta esse conceito, adequando o modelo às características dos ambientes com mobilidade. Para isso o Espaço de Tuplas deixa de ser centralizado e passa a ser distribuído entre os componentes móveis, promovendo um modelo que estabelece um compartilhamento transitório dos Espaços de Tupas baseado em critérios de conectividade.

No Lime, o Espaço de Tuplas não é associado a um dispositivo específico, mas sim aos processos que estiverem sendo executados, tratados como agentes móveis. Sendo assim, cada um deles tem seu próprio Espaço de Tuplas que o acompanha durante os

processos de migração; esse espaço pode ser compartilhado de forma transparente com os demais processos existentes em suas proximidades, formando um Espaço de Tuplas federado. Isso significa que os processos envolvidos na comunicação devem estar em execução no mesmo dispositivo ou acessíveis através de uma rede de comunicação (vide figura 2.7), provendo, dessa forma, um tratamento para a mobilidade.

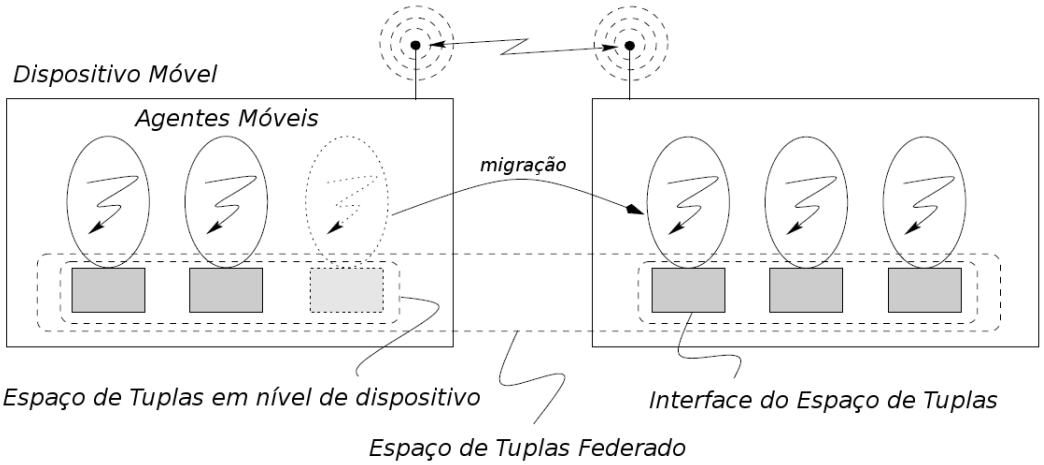

Figura 2.7: Espaço de Tuplas Federado (adaptado de (MURPHY; PICCO; ROMAN, 2006))

No modelo original de Espaço de Tuplas definido em Linda, não existem informações a respeito dos processos geradores ou consumidores das tuplas. Porém, em sistemas distribuídos móveis, eventualmente pode ser conveniente ter acesso a essas informações, por questões de otimização e adaptação das aplicações. Por isso, Lime estende as operações de Linda, adicionando parâmetros que identifiquem os processos ou *hosts* envolvidos, introduzindo a noção de localidade às tuplas.

Lime também introduz, no modelo de Espaço de Tuplas, a ideia de reação, que permite atuar sobre as constantes mudanças que acontecem no dinâmico ambiente definido pela mobilidade. A reação consiste na definição de um fragmento de código que deve ser executado quando uma tupla de determinado padrão é inserida no Espaço de Tuplas.

Para manipular o Espaço de Tuplas, a API do Lime herda as operações básicas do modelo original do Linda: `out(t)`, `in(p)` e `rd(p)` (vide seção 2.2.3). Por conveniência, o Lime introduz operações não-bloqueantes com funcionalidade similar às anteriores para consulta de tuplas em massa: `rdg(p)` e `ing(p)`.

2.3.2 Middleware TOTA

O TOTA (*Tuples On The Air*) (MAMEI; ZAMBONELLI, 2006b) é um middleware voltado às aplicações distribuídas com suporte à adaptação e sensibilidade ao contexto em cenários dinâmicos. A ideia chave do TOTA consiste na utilização de um Espaço de Tuplas distribuído, cujo objetivo é fornecer um mecanismo para a comunicação desacoplada entre os componentes das aplicações distribuídas, além de servir de meio para a disponibilização de informações de contexto.

No TOTA, cada nodo do sistema executa uma versão do middleware, que se comunica com os demais através de uma rede *peer-to-peer*. A estrutura da rede é constituida através de relações de vizinhança, em que cada um dos TOTA nodos possui referência a

um número limitado de nodos vizinhos, com os quais comunica diretamente, como em um cenário MANET (*Mobile Ad Hoc NETworks*). Cada processo pode gravar tuplas localmente e as propagar através de múltiplos saltos baseado em regras de propagação. No TOTA, as tuplas são constituídas por conteúdo ("C") e regras de propagação ("P"):

```
T=(C,P)
```

O conteúdo é constituído pelas informações que são trocadas entre aplicações distribuídas, enquanto as regras de propagação servem para definir como as tuplas devem ser transmitidas através da rede. As tuplas não têm necessariamente valores fixos, a medida que se propagam podem assumir valores diferentes em diferentes nodos. Dessa forma podem ser utilizadas para expressar informações de espaço e contexto. Além das regras de propagação, pode-se determinar a forma como o conteúdo das tuplas deve ser modificado enquanto elas são distribuídas através da rede. O middleware fica constantemente monitorando a entrada e saída de nodos na rede, possibilitando que a composição da mesma também seja um aspecto a ser transmitido através das tuplas.

No TOTA, as consultas são realizadas no Espaço de Tuplas local enquanto as inserções são feitas de forma distribuída de acordo com as regras de propagação associadas a cada tupla. A reatividade também está presente no TOTA, sendo que através da sua API, as aplicações podem realizar subscrições caracterizando o motivo do evento que deve ser gerado, que pode ser associado a modificações nos dados armazenados no Espaço de Tuplas ou a mudanças na topologia da rede local.

Conforme pode ser observado na figura 2.8, o TOTA é formado por três partes básicas: TOTA API, EVENT INTERFACE e o TOTA ENGINE.

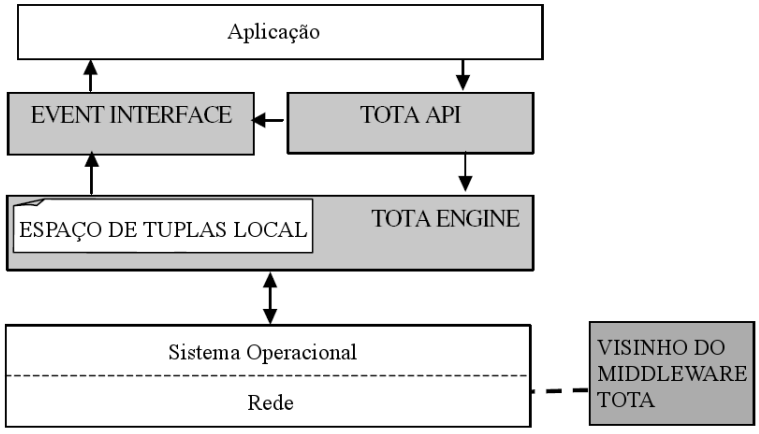

Figura 2.8: Arquitetura do TOTA (adaptada de (MAMEI; ZAMBONELLI; LEONARDI, 2003))

A TOTA API é a principal interface entre as aplicações e o middleware. Provê um conjunto de operações cujas funcionalidades são a inserção de novas tuplas no sistema, o acesso ao Espaço de Tuplas local e as subscrições no *event interface*. A principal operação serve para injetar tuplas na rede TOTA que são passadas como argumento:

```
public void inject (Tuple tuple)
```

Assim que a tupla é injetada, inicia-se o processo de propagação, segundo a regra definida na tupla.

O acesso à lista das tuplas gravadas é feito localmente, e, para isso, são disponibilizadas duas operações:

```
public ArrayList read (Tuple template)
public ArrayList delete (Tuple template)
```

A operação *read* acessa o Espaço de Tuplas local e retorna uma lista com as tuplas encontradas que combinam com o *template* passado como parâmetro. A operação *delete* funciona de forma semelhante, porém extraindo do Espaço de Tuplas local todas as tuplas que combinam com o *template*, as retornando em uma lista.

Para manipular eventos sobre o Espaço de Tuplas, são disponibilizadas duas operações:

```
public void subscribe (Tuple template, String reaction)
public void unsubscribe (Tuple template)
```

O middleware TOTA considera que qualquer evento pode ser representado por uma tupla, incluindo eventos ocorridos pela inserção de uma nova tupla ou pela conexão/desconexão de um nodo. Desse modo, a operação *subscribe* associa a execução de um método de reação (cujo nome é especificado no segundo parâmetro), em resposta à ocorrência de eventos que combinem com o modelo (*template*) passado no primeiro parâmetro. Essa inscrição pode ser retirada através da operação *unsubscribe*.

O EVENT INTERFACE é o componente que manipula a ocorrência dos eventos, notificando as aplicações quando são inseridas novas tuplas ou quando ocorre uma alteração na topologia da rede com a entrada ou saída de um nodo.

O TOTA ENGINE é o núcleo do middleware. É ele quem gerencia a rede através da referência a nodos vizinhos, além de manipular as tarefas de propagação das tuplas. Também, é responsável pela monitoração da entrada e saída de nodos no sistema.

2.3.3 Modelo PeerSpace

PeerSpace (PEREIRA et al., 2002) é um modelo de coordenação baseado em Espaço de Tuplas com foco em redes *ad hoc*. No PeerSpece, cada nodo possui o seu próprio Espaços de Tuplas, que pode ser compartilhado por aplicações que estejam em execução no próprio nodo ou em nodos remotos. O modelo prevê três utilizações típicas:

- coordenar processos que executem no mesmo nodo. Nessa situação, o PeerSpace se comporta como no modelo original – Linda;
- coordenar processos em execução em nodos distintos. Para viabilizar esse processo, o PeerSpace disponibiliza variantes das operações tradicionais de Linda;
- armazenar e disponibilizar informações a respeito de serviços disponíveis no nodo local além de informações sobre a rede.

A rede assumida pelo PeerSpace é constituída por nodos dotados de conexão sem-fio, formando uma infraestrutura *ad hoc*. Os nodos são móveis e autônomos, e sua conectividade é transitória. Por isso, a topologia da rede está em constante mudança. No

PeerSpace, também considera-se que os nodos podem atuar como roteadores capazes de propagar mensagens entre elementos da rede conectados diretamente.

Na concepção do PeerSpace, a rede é organizada através de grupos de nodos e um conjunto de subgrupos, organizando-se hierarquicamente e formando uma estrutura em árvore. Grupos são um conceito natural em redes *ad-hoc*, já que as mesmas são frequentemente utilizadas para coordenar equipes de trabalho, como, por exemplo, durante um determinado evento. No PeerSpace, os grupos servem para delimitar o escopo de consulta de serviços. Sendo assim, sempre que possível, as pesquisas são restringidas aos nodos de um grupo específico da rede (OLIVEIRA VALENTE, 2002).

A preocupação em relação à delimitação do escopo das consultas é recorrente nas discussões referentes aos sistemas de gerenciamento voltados aos ambientes da Computação Pervasiva, com o intuito de promover a escalabilidade. Na concepção do middleware EXEHDA, esse aspecto também foi considerado e motivou a definição da organização do ambiente pervasivo a ser gerenciado (vide seção 3.1).

As primitivas utilizadas pelo PeerSpace para acesso ao Espaço de Tuplas local são as mesmas definidas por Linda: `out (t)`, `in (p)` e `rd (p)`. O PeerSpace, por sua vez, introduz a primitiva `chgrp (g)`, usada para alterar o grupo de um nodo para aquele especificado pela tupla `g`.

O modelo PeerSpace não prevê transparência no acesso aos Espaços de Tuplas remotos, sendo tal decisão justificada devido aos compromissos com escalabilidade e desempenho. Em contrapartida, são disponibilizadas variações das primitivas básicas para acesso a um Espaço de Tupla localizado em um nodo `h` previamente conhecido, possivelmente localizado pela operação `find`. Assim, são oferecidas as primitivas `out(h, v)`, `in(h, v, x)` e `rd(h, v, x)` para acesso remoto, sendo `v` uma tupla ou um *template* e `x` uma variável que pode receber o valor de uma tupla como resultado.

Para dar suporte às primitivas remotas, o PeerSpace introduziu a operação `find(g,p)`, que tem por objetivo a busca de serviços disponíveis na rede, representados por tuplas (como impressoras ou sistemas de arquivos). Essa primitiva busca em todos os nodos do grupo `g` por tuplas compatíveis com o padrão `p`.

Uma importante contribuição do PeerSpace é a implementação de consultas contínuas, que correspondem à reatividade do modelo. No projeto das consultas contínuas, uma das questões centrais diz respeito à forma com que as consultas são encerradas. Nesse modo de consulta, se nenhuma tupla que combine com o padrão procurado tenha sido localizada em uma primeira varredura, a consulta se mantém ativa por um tempo previamente determinado a espera de que uma tupla com as características desejadas seja inserida no espaço de tuplas. Caso novos nodos se conectem na rede enquanto a consulta estiver ativa, ela é propagada para esses nodos transferindo o tempo de vida restante juntamente com a mesma.

2.4 Discussão

Os cenários da Computação Pervasiva são caracterizados pela presença de uma grande quantidade de dispositivos heterogêneos interligados através de redes de comunicação, nos quais são executadas aplicações amplamente distribuídas, móveis e fortemente integradas ao mundo real.

Para simplificar as tarefas de desenvolvimento dessas aplicações, é necessário um middleware para servir de interface entre as mesmas e os dispositivos existentes, a fim de

abstrair as características da infraestrutura. Na perspectiva do projeto ISAM, o middleware também deve ser capaz de disponibilizar ao usuário o seu ambiente computacional, independentemente de lugar, tempo ou dispositivo de acesso (YAMIN, 2004).

Tendo em vista as características da Computação Pervasiva pode-se afirmar que seria impraticável a utilização do modelo de coordenação direta neste cenário. Uma das razões para isso é o acoplamento referencial provido pelo modelo. A tarefa de descobrimento da localização dos demais dispositivos na rede demandaria um elevado custo computacional. Além disso, a comunicação seria inviabilizada no momento em que algum dos processos envolvidos migrasse para outro dispositivo.

Em relação ao modelo de espaço de tuplas, o desacoplamento referencial e temporal promove notável flexibilidade para operação em ambientes abertos e dinâmicos. As desconexões, características dos ambientes móveis, não são problemas com a utilização desse modelo de coordenação, pois as tuplas podem ser acessadas assim que a comunicação é restabelecida; nesse caso, existe a necessidade de prover espaços de tuplas descentralizados e associados aos processos móveis, para que tais espaços possam manter, mesmo que momentaneamente, as operações sobre as tuplas durante as desconexões.

Na abordagem de coordenação baseada no modelo *publish/subscribe*, a utilização de eventos para a notificação da ocorrência de fatos importantes reduz os esforços computacionais dos processos envolvidos, bem como do suporte as comunicações; isso coloca essa abordagem à frente das demais no requisito escalabilidade e esforço computacional. Por outro lado, o acoplamento temporal é um fator limitante na sua utilização em sistemas dinâmicos, em que dispositivos podem estar eventualmente inoperantes ou desconectados.

A flexibilidade promovida pela abordagem de Espaço de Tuplas, associada à reatividade promovida pelo modelo *publish/subscribe*, constitui uma solução de coordenação adequada aos ambientes pervasivos. Nesse caso, os eventos podem ser utilizados para notificar as aplicações quando tuplas de seu interesse são inseridas no espaço de dados. Essa estratégia tem sido adotada por alguns middlewares, entre eles o TOTA, LIME e PeerSpace, que têm como uma das vantagens o fato de liberar as aplicações do constante monitoramento. Essa abordagem também será base para a proposta apresentada na seção 4.3.

Na Computação Pervasiva, é necessário estabelecer um mecanismo que possibilite coordenar componentes de software e que disponibilize os dados de interesse do usuário independente de sua localização e tempo. A complexidade de qualquer solução proposta nesse sentido se potencializa conforme aumenta a quantidade de informações que devem ser movimentadas pelo ambiente pervasivo.

3 MIDDLEWARE EXEHDA: REVISÃO ARQUITETURAL E FUNCIONAL

Neste capítulo, será apresentado um resumo do estudo realizado em relação ao middleware EXEHDA, envolvendo as suas principais características e funcionalidades. Será dada atenção especial ao subsistema de comunicação, por estar diretamente relacionado ao tema deste trabalho.

O middleware EXEHDA (*Execution Environment for Highly Distributed Applications*) faz parte do projeto ISAM (Infraestrutura de Suporte às Aplicações Móveis) (ISAM, 2008). Esse projeto consiste em um ambiente consolidado para Computação Pervasiva que servirá de base para a materialização dos conceitos desenvolvidos neste texto. O foco do ISAM é construir uma infraestrutura para a Computação Pervasiva, integrando um *framework* de desenvolvimento e um middleware para suportar sua execução.

A arquitetura ISAM foi criada para permitir que as aplicações obtenham informações do ambiente no qual executam e para reagirem de maneira adaptativa, possibilitando alteração do comportamento conforme modificações no ambiente. No ISAM, as aplicações são inerentemente móveis e distribuídas, e, nesse caso, são consideradas tanto a mobilidade lógica quanto a física. Entende-se por mobilidade lógica a movimentação entre equipamentos de artefatos de software, e por mobilidade física o deslocamento do usuário, portando ou não seu equipamento (YAMIN, 2004; AUGUSTIN, 2004).

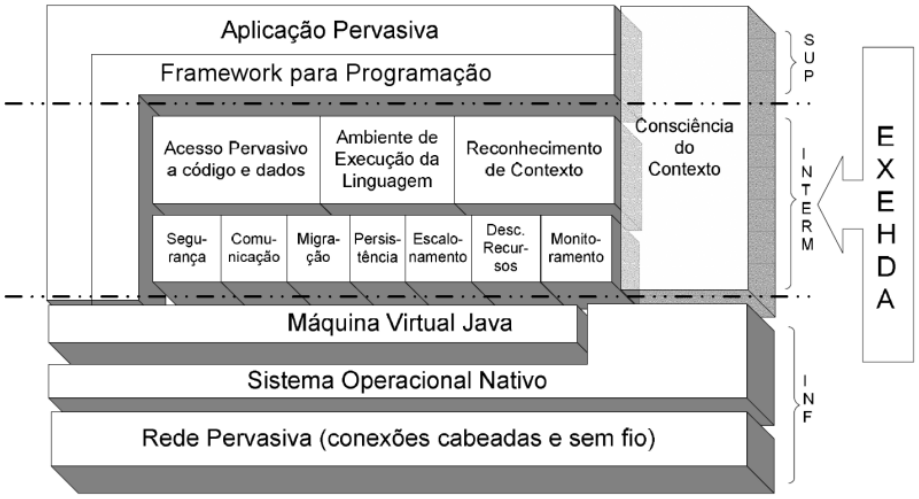

Figura 3.1: Arquitetura do ISAM (YAMIN, 2004)

Conforme pode ser visto na figura 3.1, a estrutura do projeto ISAM é dividida em três camadas: camada inferior, composta pela infraestrutura para execução das camadas superiores (hardware, sistema operacional e máquina virtual Java); camada intermediária, constituída pelo middleware EXEHDA (LOPES; PILLA; YAMIN, 2007) responsável por toda a gerência da execução das aplicações; e a camada superior, formada pelo framework para programação que possibilita o suporte ao comportamento adaptativo em tempo de desenvolvimento e pelas aplicações pervasivas.

A representação da consciência de contexto na figura 3.1 representa um módulo virtual. O objetivo é ressaltar sua importância na arquitetura e caracterizar a sua presença na criação dos demais componentes.

O middleware EXEHDA é dividido em dois níveis. O primeiro nível possui os módulos de serviço à aplicação: Acesso Pervasivo a dados e código, Reconhecimento do Contexto e Ambiente de Execução da Linguagem. O segundo nível caracteriza os serviços básicos, que são os seguintes: segurança, comunicação, migração, persistência, escalonamento, descoberta de recursos e monitoramento.

O Ambiente de Execução da Linguagem é o módulo encarregado pelo gerenciamento da aplicação durante seu tempo de uso. O serviço de Reconhecimento de Contexto, por sua vez, informa o estado dos elementos de contexto de interesse da aplicação e do próprio ambiente de execução. Por fim, o módulo de Acesso Pervasivo a dados e códigos disponibiliza todo o ambiente pervasivo denominado ISAMpe (ISAM *pervasive environment*).

3.1 Organização do Ambiente Pervasivo

Na visão do middleware EXEHDA, a Computação Pervasiva pode ser obtida através da integração de três cenários: (*i*) computação em grade, (*ii*) computação móvel e (*iii*) computação sensível ao contexto. Sendo assim, o ambiente pervasivo (ISAMpe) gerenciado pelo middleware EXEHDA é formado por um conjunto de células constituindo uma infraestrutura semelhante a uma estrutura de grade computacional (ver figura 3.2). As abstrações envolvidas nessa estrutura são mapeadas em três partes básicas, conforme (YAMIN, 2004):

- EXEHDAcel: consiste na área de atuação de uma EXEHDAbase, sendo composto pelo próprio EXEHDAbase e por EXEHDAnodos. Para definir a área de abrangência de uma célula, os principais aspectos considerados são estes: o escopo institucional, a proximidade geográfica e o custo de comunicação;

- EXEHDAbase: é o elemento de referência da célula, sendo ponto de contato para os EXEHDAnodos. É responsável por todos os serviços básicos do ISAMpe e, embora constitua uma referência lógica única, seus serviços, sobretudo por aspectos de escalabilidade, poderão estar distribuídos entre vários equipamentos;

- EXEHDAnodos: são os equipamentos de processamento disponíveis no ISAMpe, sendo responsáveis pela execução das aplicações. Um subcaso desse tipo de recurso é o EXEHDAnodo móvel. São os nodos do sistema com elevada portabilidade, tipicamente dotados de interface de rede para operação sem fio e, nesse caso, integram a célula à qual seu ponto-de-acesso está subordinado. São funcionalmente análogos

aos EXEHDAnodos, porém eventualmente têm uma capacidade mais restrita (por exemplo, PDAs).

A forma celular como o ISAMpe é organizado possibilita um mecanismo de gerência do ambiente pervasivo que resguarde a autonomia das instituições envolvidas. Nesse sentido, a abrangência de cada célula seria limitada ao alcance da rede corporativa de cada instituição. Desse modo, o acesso pervasivo aos recursos internos às células, como impressoras, scaners, etc., podem ser gerenciados pelas instituições e disponibilizados através de mecanismos de comunicação inter-célula.

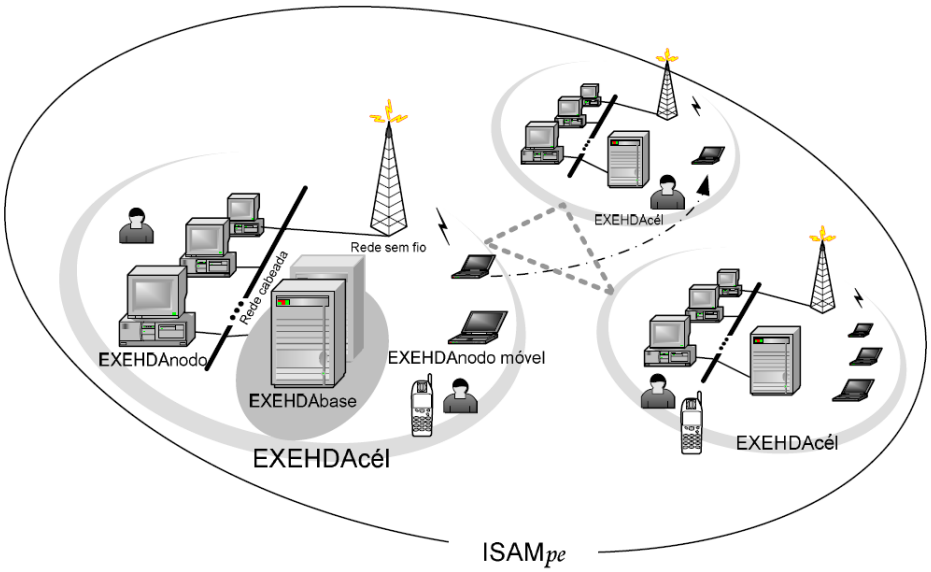

Figura 3.2: Organização do ISAMpe (YAMIN, 2004)

3.2 Estrutura do EXEHDA

Em um ambiente altamente heterogêneo, como é o caso da Computação Pervasiva, em que existe uma grande variedade de recursos de hardware (processamento e memória) e software (sistema operacional e bibliotecas), a adoção de uma abordagem baseada em serviços foi motivada. Então, o EXEHDA é estruturado a partir de um núcleo mínimo do middleware, tendo suas funcionalidades estendidas por serviços carregados sob demanda. Tal abordagem proporciona um perfil adaptativo, em que poderão ser utilizadas versões de um determinado serviço mais adequadas às características de cada dispositivo.

No EXEHDA, os relacionamentos entre instância de nodo e celular (intracelular) assemelham-se à estratégia de Proxies, enquanto que em instâncias celulares (intercelular) assumem um caráter P2P. A utilização de P2P nas operações intercelulares vai de encontro aos compromissos com a escalabilidade do sistema.

A organização do núcleo mínimo do EXEHDA e do conjunto atual de serviços é apresentada nas seções seguintes. Os serviços estão organizados em quatro subsistemas: execução distribuída, adaptação, comunicação e acesso pervasivo (vide figura 3.3).

Na concepção do EXEHDA, são empregados identificadores únicos no escopo do ISAMpe. Considerando a demanda por escalabilidade inerente à Computação Pervasiva,

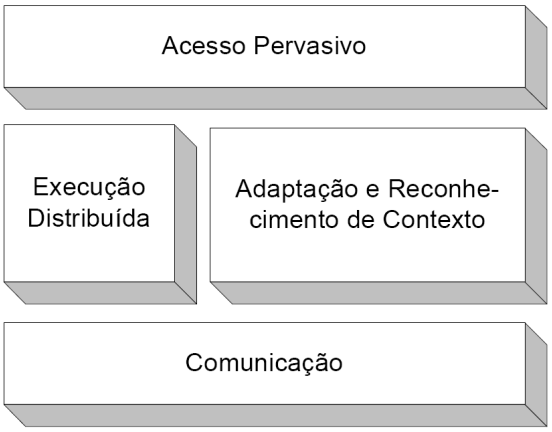

Figura 3.3: Organização dos Subsistemas do EXEHDA (YAMIN, 2004)

esses identificadores únicos são construídos por composição, tendo por base o identificador do EXEHDAnodo. Os principais identificadores empregados no EXEHDA são os seguintes:

- HostId: identifica unicamente um EXEHDAnodo no ISAMpe. É composto por um nome de célula e por um identificador numérico de 32 bits único no escopo daquela célula;

- ApplicationId: identifica unicamente uma aplicação em execução no ISAMpe. É composto por um identificador único de 512 bits, agregado ao HostId do EXEHDAnodo a partir do qual a aplicação foi disparada;

- ObjectId: identifica unicamente um OX (Objeto eXehda), no escopo do ISAMpe. É composto por um identificador inteiro de 16 bits acrescido de um hash também de 16 bits e do HostId do nodo em que o OX foi inicialmente instanciado. Observe-se que o identificador da aplicação ao qual o OX pertence não é incluído explicitamente no ObjectId; contudo, essa informação é considerada no cálculo do hash e disponibilizada na forma de um atributo do OX, acessível por intermédio do serviço *OXmanager*.

3.2.1 Núcleo do EXEHDA

As funcionalidades do EXEHDA podem ser personalizadas através de perfis de execução, que são especificados através de um documento XML (vide figura 3.4). Cada nodo pode definir o conjunto de serviços a serem ativados, podendo ter implementações específicas de acordo com as suas necessidades e adaptado aos recursos disponíveis. O perfil de execução também pode controlar a política de carga a ser utilizada para cada serviço: (*i*) quando da ativação do nodo (*bootstrap* do middleware) ou (*ii*) sob demanda.

O núcleo mínimo do EXEHDA é formado pelos seguintes componentes (vide figura 3.5):

- ProfileManager: interpreta a informação disponível no perfil de execução do nodo e a disponibiliza aos outros serviços do middleware;

- ServiceManager: realiza a ativação dos serviços no EXEHDAnodo a partir das informações disponibilizadas pelo *ProfileManager*. Para isso, carrega o código dos serviços do middleware de acordo com a política adotada, a partir do repositório de serviços, que pode ser local ou remoto.

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

Figura 3.4: Formato do Documento de Definição dd Perfil de Execução do EXEHDA (YAMIN, 2004)

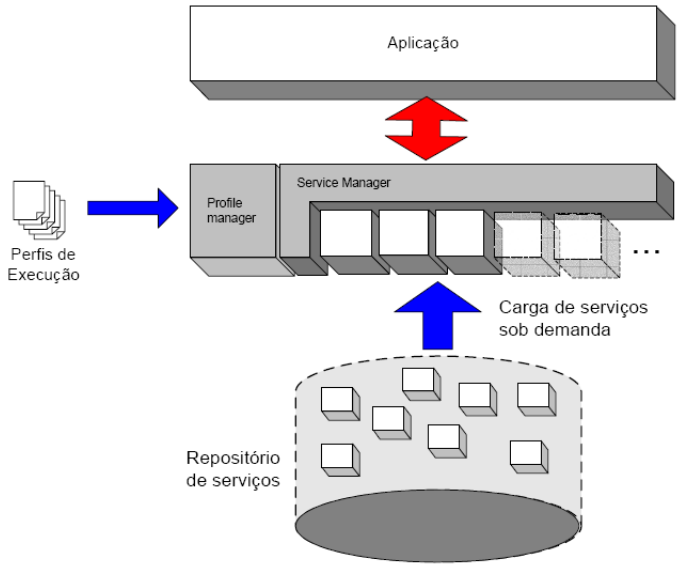

Figura 3.5: Organização do Núcleo do EXEHDA (YAMIN, 2004)

3.2.2 Subsistema de Execução Distribuída

O Subsistema de Execução Distribuída é constituído por um conjunto de serviços que dão suporte ao processamento pervasivo no EXEHDA. Ele interage com o subsistema de reconhecimento de contexto e adaptação, de forma a prover comportamento distribuído e adaptativo às aplicações.

O Subsistema de Execução Distribuída é formado pelos seguintes serviços:

Executor

No EXEHDA, a abstração OX (Objetos eXehda) consiste na unidade computacional base das aplicações, que podem migrar entre quaisquer dispositivos do ISAMpe. O

serviço *Executor* acumula as funções de disparo de aplicações e de criação e migração de seus objetos (OX). Na implementação dessas funções, é empregada a instalação de código sob demanda e para isso interage com os serviços CIB e BDA.

Cell Information Base

O serviço *Cell Information Base* (CIB) implementa a base de informações da célula, e está relacionado à manutenção da infraestrutura do ISAMpe. Esse serviço mantém os dados estruturais da EXEHDAcel, tais como informações sobre os recursos que a compõem, informações sobre os nodos vizinhos e atributos que descrevem as aplicações em execução. O serviço CIB é ainda responsável pela manutenção das informações referentes aos usuários registrados na célula.

OXManager

Esse serviço é responsável pela gerência e manutenção da meta-informação associada aos OXs. O serviço *OXManager* confere o caráter pervasivo às operações de consulta e atualização dos atributos dos OXs, permitindo que os mesmos sejam acessados a partir de qualquer nodo pertencente ao ISAMpe.

Discoverer

O serviço *Discoverer* é responsável pela localização de recursos específicos no ISAMpe a partir de especificações abstratas. As especificações caracterizam o recurso a ser descoberto por meio de atributos e seus respectivos valores, apresentados na forma de documento XML. A amplitude da pesquisa pode ser definida através de um parâmetro que pode assumir um dos seguintes valores:

0 : próprio nodo;

1 : segmento local;

2 : célula local;

3 : vizinhança estática da célula local;

4 : vizinhança completa da célula local, nível 1;

5 : vizinhança completa da célula local, nível 2.

ResourceBroker

Esse serviço tem como função controlar a alocação de recursos para que possam ser utilizados pelas aplicações. As requisições podem ser originárias da própria EXEHDAcel ou oriundas de outras células do ISAMpe.

Gateway

O serviço *Gateway* faz a intermediação das comunicações entre os nodos externos à célula e os recursos internos a ela. Dessa maneira, o controle de acesso aos recursos de uma EXEHDAcel é decorrente da ação conjunta entre *ResourceBroker* o *Gateway*.

StdStreams

O *StdStreams* dá suporte ao redirecionamento dos *streams* padrões de entrada, saída e erro. Sua funcionalidade se dá por aplicação, de modo que cada aplicação em execução em um determinado EXEHDAnodo possui um componente Console individualizado que agrupa os três *streams* padrões.

Logger

O serviço Logger faz o registro do rastro da execução das aplicações, que pode ser utilizado nas fases de desenvolvimento para a depuração de um programa, auxiliando a identificar comportamentos errôneos ou imprevistos. Também pode ser utilizado para registro de operações importantes e/ou críticas realizadas, facilitando a identificação de situações de intrusão ou de uso indevido do sistema.

Dynamic Configurator

O serviço *Dynamic Configurator* tem o objetivo de realizar a configuração do perfil de execução do middleware em um determinado EXEHDAnodo de forma automatizada.

3.2.3 Subsistema de Reconhecimento do Contexto e Adaptação

O Subsistema de reconhecimento do contexto e adaptação é formado pelos serviços que são responsáveis pelo comportamento adaptativo do middleware EXEHDA. Esse subsistema é constituído por serviços que vão desde a extração bruta das informações sobre as características dinâmicas e estáticas dos recursos que compõem o ISAMpe, passando pela identificação em alto nível dos elementos de contexto, até o disparo das ações de adaptação em reação a modificações no estado dos elementos de contexto.

Os serviços que compõem o subsistema reconhecimento do contexto e adaptação são os seguintes:

Collector

Esse serviço é responsável pela extração da informação bruta, que, após um tratamento adequado, dá origem aos elementos de contexto. Isso se dá através de um processo em que o *Collector* reúne a informação originada em vários componentes monitores, que, por sua vez, coletam-nas em um conjunto de sensores parametrizáveis. Entre os consumidores de tais informações estão os serviços *ContextManager* e *Scheduler*.

Deflector

A função do serviço *Deflector* é disponibilizar uma abstração à comunicação usada na disseminação das informações monitoradas. O serviço utiliza canais de multicast nessa tarefa e busca garantir a escalabilidade para a arquitetura de monitoração do EXEHDA.

ContextManager

Esse serviço é responsável pelo tratamento da informação bruta extraída pelo *Collector* para dar origem às abstrações referentes aos elementos de contexto. O serviço

ContextManager utiliza uma descrição em formato XML que informa a maneira como devem ser produzidos os elementos de contexto a partir da informação proveniente da monitoração. Após a definição de um dado contexto, os interessados podem registrar-se junto ao *ContextManager* para receber essas informações.

AdaptEngine

O serviço *AdaptEngine* é responsável pelo controle das funções de adaptação do EXEHDA através de facilidades para definição e gerência de comportamentos adaptativos por parte das aplicações. Desse modo, libera o programador de gerenciar os aspectos de mais baixo nível envolvidos na definição e liberação dos elementos de contexto junto ao *ContextManager*.

Scheduler

O *Scheduler* é o serviço que realiza a gerência das adaptações de cunho não-funcional no EXEHDA, isto é, que não implicam alteração de código. Esse serviço emprega a informação de monitoração, obtida junto ao serviço *Collector*, para orientar operações de mapeamento. Essas operações decorrem de instanciações remotas ou migrações realizadas pelo serviço *Executor*, ou de chamadas de reescalonamento, originadas quando o estado atual de um recurso não satisfazer mais as necessidades de um objeto anteriormente a ele alocado.

3.2.4 Subsistema de Comunicação

O subsistema de comunicação do EXEHDA disponibiliza mecanismos que atendem os aspectos intrínsecos da Computação Pervasiva. Nesse sentido, os serviços voltados à comunicação consideram a natureza da mobilidade do hardware e software, o que caracteriza uma interação intermitente entre os componentes da aplicação distribuída. Nos ambientes pervasivos, as desconexões são comuns devido à existência de alguns links sem fio e também a estratégias voltadas à economia de energia nos dispositivos móveis.

O subsistema de comunicação dispõe dos seguintes serviços:

Dispatcher

O serviço *Dispatcher* provê o modelo de comunicação mais elementar do EXEHDA, e consiste na troca de mensagens ponto-a-ponto com garantia de entrega e ordenamento. As mensagens trafegam entre instâncias do *Dispatcher* localizadas em diferentes nodos do ISAMpe. Na inicialização, o *Dispatcher* atualiza a informação do EXEHDAnodo na CIB provendo uma lista de protocolos e endereços que podem ser utilizados para alcançar aquele EXEHDAnodo.

A interface do serviço *Dispatcher* contempla métodos para alocação e liberação de portas e buffers (métodos `createPort`, `releasePort`, `allocBuffer` e `releaseBuffer`), além de operações para envio e recepção de mensagens (métodos `send` e `receive`) sobre portas previamente alocadas. Ainda são disponibilizados métodos para a suspensão, reativação e redirecionamento de portas (métodos `suspendPort`, `resumePort` e `redirectPort`, `respectivamente`) para serem utilizados nos casos de migração de componentes de software. Do ponto de vista do emissor da mensagem, a migração de uma porta destino para outro EXEHDAnodo é

transparente, sendo as mensagens destinadas à localização antiga automaticamente redirecionadas para a nova localização da porta.

O serviço *Dispatcher* utiliza a abstração de canal, que consiste numa ligação ponto-a-ponto estabelecida entre EXEHDAnodos envolvidos na comunicação; os canais são criados em nível de aplicação constituindo estruturas de seu uso privado. A alocação e a liberação de canais ocorrem de forma implícita em função das operações de envio e recepção de mensagens.

O protocolo de estabelecimento de um canal compreende, além da autenticação dos EXEHDAnodos envolvidos, a identificação da aplicação à qual aquele canal se destina. O modelo de comunicação também define chamadas específicas para a alocação dos buffers que serão empregados na composição das mensagens. Nesse sentido, além do destino da mensagem, na alocação de um buffer (`MessageBuffer`), é definido o nível de privacidade requerido por aquela mensagem. Com base nessas informações, a implementação do *Dispatcher* seleciona a tecnologia de comunicação a ser utilizado pelo canal empregado para conexão com a instância do *Dispatcher* em execução no EXEHDAnodo destino. Baseado na tecnologia selecionada, o *Dispatcher* procede, então, a alocação do buffer adequado à mesma.

No que se refere à manutenção da consistência das comunicações durante os períodos de desconexão planejada, o *Dispatcher* emprega internamente um mecanismo de *checkpointing* do estado dos canais, provendo um tratamento transparente da desconexão planejada ao utilizador da abstração canal. Durante o período de desconexão as mensagens geradas pela aplicação são armazenadas no canal, sendo transmitidas quando da reconexão.

WORB

O serviço WORB oferece um modelo de comunicação baseado em invocações remotas de método, similar ao RMI, porém sem exigir uma conexão permanente durante toda a execução da chamada remota. Esse serviço é construído a partir das funcionalidades do serviço *Dispatcher*, herdando suas características, como de escopo individualizado por aplicação, assim como um tratamento básico da desconexão planejada. Adicionalmente, o WORB emprega um protocolo de invocação remota de métodos também considerando a premissa da desconexão, de forma a atingir uma melhor eficiência nas comunicações. Por esse protocolo, cada chamada remota recebe um identificador único que se mantem válido durante uma eventual desconexão, sendo utilizado quando da reconexão para recuperação do resultado da invocação remota.

A interface do serviço WORB disponibiliza métodos para o registro e cancelamento do registro de um objeto (métodos `exportService` e `unexportService`), respectivamente habilitando e cancelando o suporte para que tal objeto receba invocações remotas de método. Também disponibiliza o método `lookupService` que possibilita a obtenção de uma referência para um dado serviço remoto, e `setExceptionHandler` que permite modificar o tratador de exceções de comunicação associado a uma dada referência remota. Por sua vez, o método `getClientHost` quando utilizado dentro de um método previamente exportado pelo WORB, permite identificar o EXEHDAnodo originador da invocação remota que está sendo tratada.

CCManager

O serviço CCManager consiste em um mecanismo baseado na abstração de espaço de tuplas, um mecanismo de comunicação com característica de desacoplamento temporal e espacial; portanto, não exige a coexistência temporal dos elementos envolvidos na comunicação. Isso é particularmente oportuno no que diz respeito a aplicações pervasivas, pois simplifica as tarefas de desenvolvimento. A abstração de espaço de tuplas também facilita a implementação de outros padrões de comunicação, além do ponto-a-ponto.

A interface do serviço *CCManager* disponibiliza métodos para criação e destruição de espaços de tuplas (`createSpace` e `destroySpace`), obtenção e liberação de referência para um espaço de tuplas existente (`joinSpace` e `leaveSpace`), assim como métodos para inserção e consumo de tuplas (`in` e `out`).

Como estratégia para conferir escalabilidade à solução, os espaços de tuplas são inicialmente criados locais ao EXEHDAnodo. Na medida em que requisições de `joinSpace` são emitidas em outros EXEHDAnodos, o espaço de tuplas é dinamicamente expandido de forma a tornar-se acessível também àquele EXEHDAnodo. Esse procedimento de expansão dinâmica envolve a instância celular do serviço *CCManager*.

3.2.5 Subsistema de Acesso Pervasivo

Os serviços que compõem o subsistema de acesso pervasivo vão de encontro à premissa da Computação Pervasiva de acesso a dados e código em qualquer lugar, todo o tempo.

Esse subsistema do EXEHDA é composto pelos seguintes serviços:

BDA

A Computação Pervasiva tem como premissa que o usuário pode disparar aplicações a partir de qualquer nodo integrante do ambiente pervasivo. Após o disparo, pode haver a mobilidade parcial ou integral de tais aplicações em resposta a 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.

Manter todo o universo de software disponível instalado e atualizado em todos os dispositivos do sistema seria impraticável. Portanto, no contexto da Computação Pervasiva, é necessário que exista a capacidade de instalação de código sob demanda. A implementação de mecanismos que contemplem essa funcionalidade requer a existência de um repositório de código que forneça a mesma visão do software disponibilizado a partir de qualquer dispositivo do ISAMpe.

O serviço BDA (Base de Dados pervasiva das Aplicações) do EXEHDA contempla métodos para a recuperação do código integral de uma aplicação ou de componentes específicos e suas dependências, além de dispor métodos para a gerência das aplicações instaladas.

O padrão de utilização concebido para o serviço BDA define que as requisições geradas nos EXEHDAnodos de uma determinada célula são sempre direcionadas à instância celular do serviço BDA da mesma EXEHDAcel em que foi gerada a requisição. Esta, se necessário, realiza o acesso a instâncias remotas (BDAs de outras células) para satisfazer uma dada requisição.

AVU

O serviço AVU (Ambiente Virtual do Usuário) do EXEHDA é responsável pela manutenção do acesso pervasivo ao ambiente virtual do usuário. A premissa siga-me definida no ISAM reflete-se não só nas aplicações que um usuário atualmente executa, mas no seu ambiente computacional como um todo. Esse ambiente engloba, além das aplicações em execução, as informações de personalização das aplicações definidas pelo usuário, o conjunto de aplicações instaladas (i.e. passíveis de serem disparadas por aquele usuário), como também seus arquivos privados.

Com esse objetivo, o serviço AVU adota uma estratégia de utilização análoga à empregada no serviço BDA. As requisições geradas pela instância local ao EXEHDAnodo do serviço AVU são direcionadas à instância celular do mesmo serviço em execução na célula local.

A instância celular, por sua vez, consulta a célula "home" do usuário de forma a descobrir a última localização física de seu ambiente virtual. De posse dessa informação, o usuário pode acessar o serviço AVU remoto para conclusão do tratamento da requisição. Quando oportuno, considerando o estado do link de rede, a instância do serviço AVU em execução na célula local pode optar por mover integralmente o ambiente virtual do usuário para a célula local, de forma a otimizar acessos futuros.

SessionManager

O serviço SessionManager do EXEHDA é responsável pela gerência da sessão de trabalho do usuário, definida pelo conjunto de aplicações correntemente em execução para aquele usuário. A informação que descreve o estado da sessão de trabalho é armazenada no AVU, estando portando disponível de forma pervasiva.

Tipicamente, o serviço SessionManager é ativado através do serviço Gatekeeper, após a conclusão com sucesso do procedimento de autenticação do usuário. Como resultado, o SessionManager recupera o estado da sessão default do usuário, restaurando a execução das aplicações correspondentes. Adicionalmente, a funcionalidade do SessionManager pode estar acessível através de aplicações de usuário, permitindo assim um controle mais preciso das aplicações contidas em uma dada sessão, não ficando restrito à manipulação da sessão default.

Gatekeeper

O serviço Gatekeeper é responsável por intermediar acessos entre as entidades externas à plataforma ISAM e os serviços do middleware de execução, conduzindo os procedimentos de autenticação necessários e posteriormente permitindo que este execute o disparo de aplicações.

3.3 Discussão

O EXEHDA é um middleware voltado para o desenvolvimento e execução de aplicações da Computação Pervasiva, provendo o suporte ao gerenciamento de um ambiente pervasivo chamado ISAMpe. O middleware é organizado através de serviços e é adaptável ao contexto. O EXEHDA provê uma estrutura de software que permite a disponibilização de dados e aplicações a partir de qualquer lugar e dispositivo a todo o

tempo.

O ambiente pervasivo gerenciado pelo EXEHDA é formado por um conjunto de células formando uma infraestrutura semelhante a uma estrutura de grade computacional. Nesse modelo, as abstrações são mapeadas em três partes básicas: EXEHDAcel, EXEHDAbase e EXEHDAnodos.

Para atender à grande variação em relação aos recursos disponíveis, inerente à Computação Pervasiva, o EXEHDA possui um núcleo mínimo, tendo suas funcionalidades extendidas através de um conjunto de serviços carregados sob demanda. Na versão atual do EXEHDA, existe um conjunto de serviços organizados em quatro grandes grupos: execução distribuída, adaptação e reconhecimento do contexto, comunicação e acesso pervasivo.

Uma questão central no EXEHDA é o suporte à mobilidade lógica e física. Entende-se por mobilidade lógica a movimentação entre equipamentos de artefatos de software, e por mobilidade física o deslocamento do usuário, portando ou não seu equipamento (YAMIN, 2004; AUGUSTIN, 2004). Nesse sentido, a mobilidade lógica é suportada pelo EXEHDA através de operações sobre a abstração OX (Objeto eXehda), unidade computacional base das aplicações, que podem migrar entre quaisquer dispositivos do ISAMpe.

4 EXEHDA-TS: CONCEPÇÃO E MODELAGEM

Esta dissertação de mestrado apresenta o EXEHDA-TS cuja modelagem é abordada neste capítulo. O EXEHDA-TS consiste em um modelo de coordenação para a Computação Pervasiva, o qual provê uma infraestrutura proativa, escalável, com gerenciamento distribuído e com controle dinâmico dos custos de comunicação.

Inicialmente será feita uma análise dos aspectos promissores da abordagem de Espaço de Tuplas como suporte ao desenvolvimento de aplicações pervasivas, sendo identificados os principais desafios de pesquisa enfrentados para se consolidar um modelo computacional com tal finalidade. Por fim, será apresentado o EXEHDA-TS, sua fundamentação, funcionalidades e arquitetura, que está sendo desenvolvido de acordo com o escopo do projeto EXEHDA.

4.1 Fundamentos para o EXEHDA-TS

A seção atual apresenta os conceitos envolvidos na coordenação de processos, utilizando a abordagem de Espaço de Tuplas no contexto da Computação Pervasiva. Serão apresentadas algumas das contribuições que este modelo trás para o desenvolvimento e coordenação de aplicações no âmbito desse novo paradigma computacional, bem como os aspectos positivos e os desafios envolvidos na consolidação de uma solução que seja adequada ao mesmo.

Os conceitos relacionados a Espaço de Tuplas foram introduzidos no final da década de 80 por um modelo denominado Linda (CARRIERO; GELERNTER, 1989), porém, recentemente, os mesmos têm recebido a atenção de diversos autores devido, em parte, ao advento da Computação Pervasiva. Nesse novo paradigma, os sistemas computacionais devem ter a capacidade de estabelecer comunicação e coordenação uns com os outros de acordo com o contexto, executando complexas aplicações distribuídas para, por exemplo, apoiar as atividades cooperativas entre usuários, acompanhar e controlar o ambiente, além de proporcionar a interação com o mundo físico (CABRI et al., 2007).

Na Computação Pervasiva, grande parte dos sistemas são embarcados e intrinsecamente móveis; como exemplo, teríamos dispositivos que acompanham um carro ou um ser humano, caracterizando a natureza de uma mobilidade física. Outro aspecto característico nesse modelo computacional é que o ambiente do usuário deve estar a sua disposição independentemente do tempo e localização (YAMIN, 2004). Nesse sentido, torna-se necessário que os componentes de software se desloquem entre os dispositivos

computacionais conforme as demandas do usuário, sendo tal característica definida como mobilidade lógica.

Os principais desafios enfrentados na especificação de um modelo de coordenação para a Computação Pervasiva origina-se devido ao ambiente de execução das aplicações ter elevada complexidade, sendo intrinsecamente aberto e dinâmico (EDWARDS; GRINTER, 2001). A seguir, alguns fundamentos considerados característicos desse tipo de sistema:

- os dispositivos, na sua maioria, são interconectados através de redes sem fio. Nessa perspectiva, os componentes são adicionados ou removidos a todo o momento, alterando a composição da rede dinamicamente. Uma aplicação móvel em execução em um ambiente dessa natureza deve adaptar-se a tais mudanças, tendo capacidade de executar em diferentes contextos;

- as infraestruturas computacionais são heterogêneas e de grande escala. No mundo moderno, a quantidade de equipamentos dotados de algum poder computacional é muito elevada e esses possuem as mais variadas características. No cenário de uma grande metrópole, por exemplo, várias estações-base sem fio serão distribuídas pelos edifícios, interligando os inúmeros dispositivos computacionais e constituindo uma grande rede de comunicação. Com o advento da Computação Pervasiva, a comunicação entre tais elementos será dada de forma global, formando uma rede de elevada escala composta por elementos heterogêneos;

- os sistemas deverão ser mais seguros e robustos. Nos cenários da Computação Pervasiva, os sistemas estão intimamente integrados à vida das pessoas, e por isso as consequências de falhas não previstas e possíveis invasões serão extremamente impactantes;

- os sistemas são inerentemente difíceis de testar e depurar. Situações inesperadas estão passíveis de surgir apenas quando o sistema está implantado e operacional em situações reais de utilização; muitos problemas dificilmente podem ser detectados em simulações durante as etapas de desenvolvimento. Além disso, em um sistema dinâmico em que existe interação entre dispositivos móveis e sem fio, são extremamente difíceis as tarefas de depuração.

Na Computação Pervasiva, as aplicações são amplamente distribuídas e dotadas de mobilidade, tanto física quanto lógica. No âmbito do EXEHDA, essa possibilidade traz consigo diversos desafios em relação à coordenação, pois a execução de um OX (vide seção 3.2) não está associada a um dispositivo físico específico e, por consequência, não conta com referências mais usuais, como, por exemplo, uma associação fixa a um determinado endereço de rede (IP e/ou hostname) (YAMIN, 2004).

Nesse sentido, o EXEHDA-TS, enquanto parte do middleware EXEHDA, deve permitir a coordenação entre aplicações, bem como o acesso a dados independentemente de localização e tempo. Como premissa, na perspectiva de atender as demandas da Computação Pervasiva, o modelo de coordenação concebido para o EXEHDA-TS prevê a adoção de estratégias que promovam a escalabilidade, a gerência das desconexões e as mobilidades física e lógica.

A dificuldade no desenvolvimento e manutenção de aplicações na Computação Pervasiva tem promovido o uso de middleware para liberar o programador do gerenciamento de tarefas de baixo nível (COSTA; YAMIN; GEYER, 2008). Na seção 4.2, a

seguir, estão resumidos os desafios identificados como necessários a um modelo para a coordenação de aplicações na Computação Pervasiva, para que sejam potencializados os esforços de desenvolvimento.

4.2 Principais Desafios de Pesquisa do EXEHDA-TS

Na concepção do EXEHDA-TS, foram considerados os aspectos conceituais da área, bem como os trabalhos relacionados ao mesmo, que são abordados no capítulo 2. Através deste estudo, identificaram-se diversos desafios de pesquisa, os quais foram empregados na concepção do modelo apresentado nesta dissertação de mestrado, e estão resumidos a seguir.

- desacoplamento: a utilização de Espaço de Tuplas na coordenação de aplicações distribuídas proporciona um desacoplamento espacial e temporal. Em cenários como da Computação Pervasiva, dispositivos entram e saem do sistema a todo instante, ocasionando momentos de desconexão. Uma situação semelhante pode ocorrer com componentes de software quando se movem através dos dispositivos. Por isso um componente de software pode estar enviando uma informação a outro componente quando este se encontra momentaneamente desconectado, e consequentemente impossibilitado para receber. O desacoplamento referencial e temporal resolve esse tipo de situação;

- endereçamento associativo: conforme já comentado anteriormente, para acessar uma tupla é necessário fornecer um *template*, a partir do qual são localizadas tuplas que combinem com ele. Em ambientes amplos e dinâmicos, é impraticável ter um conhecimento atualizado da localização de todos os dispositivos e das aplicações em execução. O endereçamento associativo proporcionado por esse modelo de coordenação reduz a complexidade do desenvolvimento de aplicações nesse cenário;

- consciência do contexto: o Espaço de Tuplas tem a possibilidade de agir como um repositório para informações contextuais e de agir de maneira reativa, informando aos processos da ocorrência de mudanças no contexto;

- segurança e robustez: o Espaço de Tuplas, sendo utilizado para intermediar as interações entre os componentes de software, introduz comportamento assíncrono entre emissor e receptor, o que contribui para aumentar o grau de robustez dos sistemas baseados nesse modo de coordenação;

- separação de preocupações: a utilização de mecanismos de coordenação simplifica as tarefas dos desenvolvedores de aplicativos porque separa a programação das tarefas de gerenciamento de baixo nível das comunicações. Dessa maneira, abstrai questões referentes à heterogeneidade dos hardwares envolvidos, infraestrutura de rede, desconexões, entre outros.

4.3 Aspectos de Modelagem do EXEHDA-TS

Nesta seção, serão discutidos os aspectos de modelagem do EXEHDA-TS, os quais consideraram as demandas da Computação Pervasiva, bem como as características

específicas do middleware EXEHDA.

No âmbito do middleware EXEHDA, a mobilidade é um elemento central desde o início do projeto. Para o EXEHDA, tanto as aplicações quanto seus componentes (OX) são intrinsecamente móveis e podem-se deslocar pelos nodos que compõem a infraestrutura. Sendo assim, é conveniente que o middleware disponibilize mecanismos que possibilitem a interação transparente entre os componentes de software independentemente de localização, simplificando assim as tarefas de desenvolvimento das aplicações para esse cenário. Tal aspecto constitui o foco central da pesquisa desenvolvida para a consolidação do EXEHDA-TS.

O ambiente pervasivo gerenciado pelo EXEHDA consiste em uma infraestrutura dotada de equipamentos fixos e de nodos móveis. Um outro aspecto a ser observado é que nodos dotados de mobilidade física que utilizam conexões sem fio podem-se desconectar e, nesse caso, tais desconexões não são consideradas como falhas na rede, mas sim situações normais, factíveis de serem previstas. Portanto, um dos aspectos com o qual o EXEHDA-TS deve contribuir é o suporte à desconexão.

Em relação à mobilidade, tanto física quanto lógica, os desafios enfrentados são decorrentes dos endereçamentos usuais, notadamente os números IP dos equipamentos, que podem, eventualmente, mudar devido às movimentações dos dispositivos ou componentes de software em relação à infraestrutura. Portanto isso deve ser tratado pelo EXEHDA-TS, de forma a ser mantida a sua operacionalidade quando da ocorrência desses eventos, disponibilizando para as aplicações um modelo de coordenação transparente em relação à mobilidade.

O EXEHDA-TS foi modelado como um serviço para o middleware EXEHDA, fazendo parte do subsistema de comunicação (vide figura 4.1). Para dar suporte às funcionalidades do EXEHDA-TS, será feita uma integração com os demais serviços existentes no middleware, entre eles o *Worb*, *Gateway*, *ResourceBroker*, *Executor* e *OXManager*. Todos esses são discutidos detalhadamente na seção 3.2.

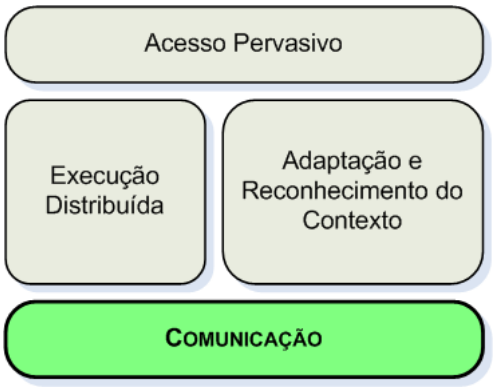

Figura 4.1: Organização dos Subsistemas do EXEHDA com Destaque ao Subsistema de Comunicação

4.3.1 Características e Funcionalidades

As funcionalidades modeladas para o EXEHDA-TS foram projetadas para satisfazer as demandas das aplicações-alvo do EXEHDA, as quais são distribuídas, dotadas de mobilidade e adaptativas ao contexto em que são processadas (YAMIN, 2004). Por-

tanto, como ponto de partida, foram consideradas as características desse middleware e o ambiente pervasivo gerenciado por ele.

4.3.1.1 Composição do Espaço de Tuplas

No modelo de coordenação que está sendo proposto para o EXEHDA-TS, o armazenamento das tuplas distribui-se fisicamente entre os nodos do ambiente pervasivo gerenciado pelo middleware EXEHDA. Para tanto, foram concebidas duas abstrações centrais para o modelo:

- **TSox**: consiste no Espaço de Tuplas elementar do EXEHDA-TS que é associado aos OX. Cada OX pode ter um ou mais TSox, cada um contemplando um diferente perfil (vide seção 4.3.1.5);
- **TSvirt**: consiste em um Espaço de Tuplas virtual constituído pelo conjunto de vários TSox. Assim como acontece com os TSox, cada aplicação pode ter um ou mais TSvirt.

O TSvirt consiste em um conjunto de referências aos TSox distribuídos pelos nodos do sistema (vide figura 4.2), servindo de base para todas as operações realizadas pelos OX. Em outras palavras, cada OX que queira realizar uma operação qualquer sobre o Espaço de Tuplas irá atuar diretamente sobre o TSvirt abstraindo qualquer informação sobre a localização física das tuplas, assim como em relação às características da infraestrutura. A composição do TSvirt é gerenciada pelo EXEHDA-TS de acordo com o comportamento e perfil dos Espaços de Tuplas criados.

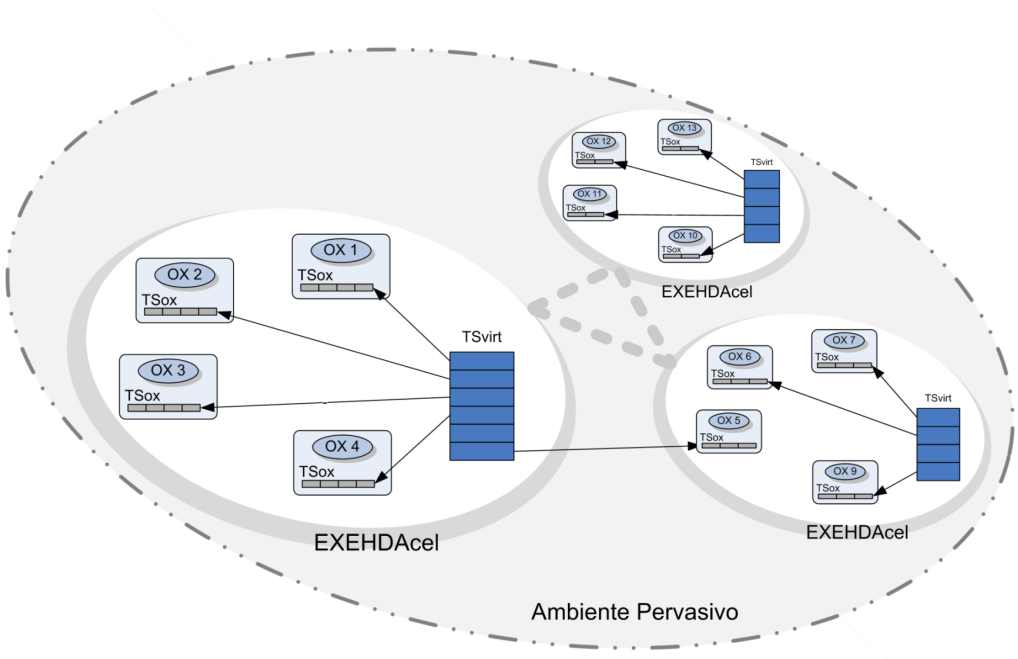

Figura 4.2: Composição do Espaço de Tuplas

Na perspectiva do modelo de Espaço de Tuplas distribuído adotado para o EXEHDA-TS, as operações de escrita são sempre realizadas no TSox do próprio OX. Já as operações de leitura podem ser locais ou remotas, porém são realizadas de maneira

transparente ao OX requisitante, pois têm sempre como intermediário o TSvirt. Assim, uma operação de leitura é sempre realizada no TSvirt, sendo que o sistema EXEHDA-TS redistribui a operação a todos TSox que o constituem, sempre que necessário.

4.3.1.2 Localização das Tuplas

A proposta do modelo do EXEHDA-TS estabelece que, via de regra, os TSox localizam-se no nodo em que o OX proprietário está em execução. Além disso, o modelo também prevê que, através de configuração específica em cada nodo, pode-se optar pelo armazenamento dos TSox juntos à EXEHDAbase. A decisão de uma ou outra situação deve ser tomada pelo administrador da EXEHDAcel, que deve avaliar o impacto da solução adotada no desempenho do sistema.

A possibilidade de manter os TSox apenas na EXEHDAbase simplifica a tarefa de gerenciamento e agiliza as consultas remotas, que não necessitariam ser redistribuídas pelos EXEHDAnodos. Essa alternativa apresenta-se mais indicada para nodos fixos dotados de rede de alta velocidade, para não comprometer a operacionalidade do serviço; porém, deve ser considerado o perfil das aplicações em execução.

Considerando a premissa de operação desconectada, no caso de nodos móveis, o conveniente é a utilização do modo de armazenamento local, permitindo a continuidade das operações do EXEHDA-TS nos momentos de desconexão. Nesse caso, as referências aos TSox localizados no nodo são removidas do TSvirt e as operações de leitura e escrita são realizadas apenas localmente, enquanto as operações remotas são bufferizadas. Assim que a conexão é reestabelecida, as operações bufferizadas são encaminhadas e as referência dos TSox são refeitas no TSvirt, tornando-o novamente acessível aos componentes de software em execução nos demais nodos do sistema.

4.3.1.3 Subscrição de Eventos

Com o intuito de reduzir o custo computacional e o uso da rede, foram modelados para o EXEHDA-TS mecanismos que promovem a reatividade. São eles a subscrição de eventos e as consultas pervasivas, sendo que o segundo mecanismo será discutido na próxima seção. Esta estratégia libera os componentes de software do ônus de ter que acompanhar as mudanças nos dados armazenados no Espaço de Tuplas por conta própria, através de sucessivas leituras das tuplas e posterior avaliação da sua relevância para o andamento das computações.

Através da subscrição de eventos, os componentes de software podem ser notificados sobre a existência de tuplas com informações relevantes imediatamente após elas terem sido inseridas no Espaço de Tuplas. Para isso, a aplicação deve utilizar uma operação de subscrição (*subscribe*) através da qual é passado como parâmetro um *template* com o padrão da tupla desejada, além de um apontador para um fragmento de código que deve ser executado quando o evento ocorrer. Dessa maneira, o EXEHDA-TS, através de um módulo gerenciador de eventos, compara cada tupla com os *templates* subscritos à medida que elas são inseridas no Espaço de Tuplas. Isso irá ocorrer indefinidamente enquanto o OX subscritor existir ou até que seja executada uma operação cancelamento (*unsubscribe*).

4.3.1.4 Consultas Pervasivas

A concepção do mecanismo de consulta pervasiva do EXEHDA-TS teve como motivação os cenários de elevada distribuição e densidade de equipamentos da Computação Pervasiva, nos quais os componentes de software estão distribuídos pelo sistema e, muitas vezes, desenvolvem atividades colaborativas. A coordenação das atividades desses componentes necessita do acesso a dados que estão inerentemente espalhados pelo sistema.

A busca por informações no TSvirt necessita que as operações sejam repassadas primeiramente aos diversos dispositivos em que estão fisicamente localizados os TSox que compõem o mesmo, em nível celular, e, posteriormente, aos demais TSox distribuídos em outros escopos celulares. Se nenhuma tupla com as características desejadas for localizada após o processo de busca, pode ser interessante mantê-la ativa nos nodos, a fim de detectar uma possível disponibilização futura, evitando, assim, a execução repetida das complexas consultas distribuídas. No EXEHDA-TS, esse tipo de busca é chamada de consulta pervasiva, consolidada pela operação *find*, constituindo mais uma contribuição para a reatividade neste trabalho. Uma abordagem semelhante a essa é utilizada no modelo PeerSpace (PEREIRA et al., 2002).

Assim como na subscrição de eventos, deve ser repassado como parâmetro na operação um *template* contendo o padrão da tupla desejada e um apontador para um fragmento de código que deve ser executado através de um evento quando a tupla for localizada. As consultas pervasivas não devem ser mantidas indefinidamente ativas à espera de informações. Por isso, um *timeout* deve ser estabelecido pelo desenvolvedor, e, quando ultrapassado, a consulta é extinta.

Basicamente, o que diferencia a subscrição de eventos da consulta pervasiva é que a última é cancelada logo após uma tupla ter sido localizada, e caso isso não ocorra, o cancelamento se dá por *timeout*. Já na subscrição de eventos a operação deve ser explicitamente cancelada através da execução de uma operação *unsubscribe*.

4.3.1.5 Abrangência por Perfil

Um Espaço de Tuplas único que contemple todos os dados compartilhados de uma aplicação distribuída pode ser composto por uma grande quantidade de informações de diferentes tipos. Essa situação poderia causar uma elevada carga computacional nos processos de localização executados pelo EXEHDA-TS, comprometendo a escalabilidade do sistema.

De modo geral, o desenvolvedor de aplicações, ao utilizar um Espaço de Tuplas para a coordenação de processos, deve ter antecipadamente uma ideia em relação aos tipos de informações que serão armazenadas nas tuplas que ele deseja manipular, devido às características do processo computacional em questão. Baseado nisso, o EXEHDA-TS contempla a abrangência por perfil com o intuito de dividir com o desenvolvedor parte da tarefa de gerência do Espaço de Tuplas.

O perfil consiste em um identificador que indica a característica das informações que devem compor cada TSvirt. Cabe ao serviço EXEHDA-TS gerenciar os diversos TSox de cada aplicação e organizá-los em seus respectivos TSvirt de acordo com o perfil de cada um. Sendo assim, uma aplicação pervasiva pode ter tantos TSvirt quantos forem os diferentes perfis definidos pelo desenvolvedor. Como o TSvirt é a unidade de referência para as operações do EXEHDA-TS, essa estratégia limita o escopo dos dados a ser gerenciado.

Em relação ao projeto PERTMED, está sendo considerada a utilização de um perfil relacionado a informações contextuais. Nesse sentido, o EXEHDA-TS pode ser utilizado como um canal para a disseminação de informações de contexto, em que após serem processadas podem ser inseridas através de um OX monitor, ficando associadas a um perfil de contexto. Essa abordagem simplifica o acesso aos dados por parte dos serviços do middleware responsáveis pelo gerenciamento de informações contextuais. Outros tipos de perfis poderão ser utilizados à medida que o projeto PERTMED for avançando.

O perfil do Espaço de Tuplas deve ser definido pelo desenvolvedor de aplicações durante a implementação do OX, mais especificamente no momento da criação do TSox pela operação que dá origem ao Espaço de Tuplas. Isso acontece quando o serviço EXEHDA-TS é instanciado, de modo que ele passe a fazer parte de um conjunto de dados com características específicas, determinadas pelo tipo de utilização.

4.3.1.6 Modos de Sincronização

Para coordenar aplicações com componentes de software localizados em células distintas, ou nos casos em que os custos de comunicação podem ser elevados, a estratégia concebida para o EXEHDA-TS é dividir com o desenvolvedor de aplicações parte da tarefa de gerenciamento. Assim, de acordo com o perfil da aplicação, durante a fase de desenvolvimento, podem ser adotadas diferentes soluções que irão influenciar no comportamento do EXEHDA-TS e na composição do TSvirt.

Os TSox sincronizam com os TSvirt para possibilitar que as operações de leitura distribuída disparadas pelos OX sejam encaminhadas aos mesmos. Situação similar acontece com a gerência de eventos, as quais são operacionalizadas em nível de TSvirt.

Sendo assim, o termo sincronização está sendo utilizado para referir-se ao modo de operação de um OX em relação ao Espaço de Tuplas cujo respectivo TSox tem sua referência estabelecida junto ao TSvirt. No modo não sincronizado, as operações originadas no OX têm o TSvirt como referência; porém, as que forem originadas pelos demais processos não são encaminhadas para o TSox do respectivo OX, pois não existe a referência ao mesmo no TSvirt. Os aspectos relacionados ao gerenciamento dos modos de sincronização são discutidos na seção 4.3.3.

Então, no EXEHDA-TS, estão disponíveis dois modos de sincronização. São eles:

- sincronização permanente: nesse modo de operação, uma vez a referência ao TSox tenha sido alocada junto ao TSvirt, o mesmo somente deixará de ter acesso de sincronização nos momentos de desconexão, independentemente dos destinos do OX decorrentes de processos de migração;

- sincronização sob demanda: nesse modo, o TSox sincroniza com o TSvirt apenas em aguns casos específicos, a fim de reduzir os custos de comunicação, pois não estando sincronizado, as operações direcionadas ao TSvirt não são repassadas ao TSox.

A utilização de um ou outro modo de sincronização é uma decisão de projeto que deve ser tomada durante o processo de desenvolvimento das aplicações. Através da API do EXEHDA-TS, na operação que cria o Espaço de Tuplas, é utilizado um parâmetro para especificar o modo de sincronização a ser utilizado. Assim, de forma transparente ao desenvolvedor, o serviço cria um TSox associado ao OX que procedeu a operação e o vincula a um determinado TSvirt, que será a referência para todas as operações realizadas

sobre o Espaço de Tuplas. No caso de sincronização permanente, uma referência a este TSox também é criada junto ao TSvirt.

O modo de sincronização permanente eleva o custo de gerenciamento e das comunicações e, portanto, tal opção deve ser adotada apenas em casos específicos. Um exemplo disso seria a situação de um OX que tenha como característica o monitoramento de um ou mais sensores, cujo número de operações de escrita seja substancialmente maior do que o de leitura de tuplas.

Na sincronização sob demanda, o TSvirt mantém a referência ao TSox em alguns casos, que estão relacionados à forma de utilização do Espaço de Tuplas por parte da aplicação associada ao TSox. Tais situações são descritas a seguir:

- devido a uma operação realizada pelo OX, cujo resultado seja retornado através de um evento;
- no caso de uma operação `insert` em que as tuplas inseridas são acompanhadas de um parâmetro adicional, indicando que a sincronização deve ser mantida até que as tuplas inseridas tenham sido lidas;
- pela operação `redefine`, nesse caso, através da troca do modo de sincronização. Essa tarefa esta que pode ser repassada ao usuário caso seja disponibilizada pelo desenvolvedor da aplicação.

Mesmo que o TSox não esteja sincronizado com o TSvirt, o mesmo continua tendo o TSvirt como referência para as operações. Portanto, ele continua podendo interagir com os demais componentes de software da aplicação de origem, porém o fluxo de comunicação é reduzido. Essa estratégia foi adotada no EXEHDA-TS com o intuito de promover a escalabilidade.

4.3.2 Arquitetura de Software

A arquitetura do EXEHDA-TS foi concebida com o propósito de construir um ambiente de alto nível, a fim de promover o trabalho colaborativo entre os componentes de software em execução no ambiente pervasivo. O gerenciamento da execução dessas aplicações dá-se através do middleware, que disponibiliza, além do EXEHDA-TS, outros serviços que podem ser utilizados durante as computações. Os demais serviços que compõem o EXEHDA estão detalhados na seção 3.2.

O bloco intermediário, em destaque na figura 4.3, representa a arquitetura do EXEHDA-TS, que é composta por uma estrutura de dados (TSvirt) e um conjunto de módulos, responsáveis pela composição dessa estrutura e pela gerência das políticas de acesso às informações.

O módulo de Gerenciamento proativo é responsável pela composição do TSvirt. Com esse propósito, o mesmo avalia as ações dos OX sobre o TSvirt, considerando os modos de sincronização utilizados e as condições de abrangência por perfil.

No EXEHDA-TS, quem possibilita que cada OX possa subscrever eventos ou realizar consultas pervasivas é o módulo denominado Reatividade. Assim, esse módulo monitora as modificações nos dados presentes no TSvirt e reage avisando os OX interessados.

As operações mais simples são aquelas realizadas sobre o próprio TSox, no EXEHDA-TS. Entre essas estão as operações de escrita, que são sempre executadas nesse

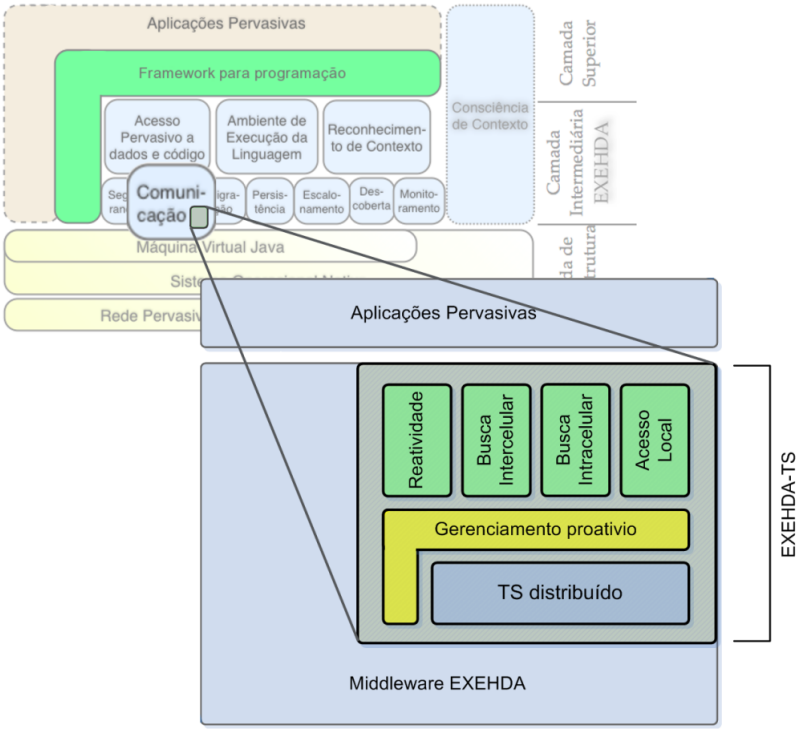

Figura 4.3: Arquitetura do EXEHDA-TS

âmbito e também constituem a primeira etapa das operações de leitura. Os processos de leitura sempre iniciam em nível local, para posteriormente serem repassados aos demais TSox distribuídos através da rede. As etapas dos processos de acesso às tuplas dos TSox existentes no mesmo nodo do OX que executou a operação são gerenciados por um módulo específico na arquitetura, identificado como Acesso Local. Essa operação é executada na instância local do serviço EXEHDA-TS.

Na medida em que uma operação de leitura não é satisfeita em âmbito local, esta é repassada aos demais TSox do TSvirt em questão. O processo consiste, primeiramente, no encaminhamento da tarefa de busca para a EXEHDAbase; em um segundo momento, a busca é realizada nos demais nodos associados ao TSvirt, que, por sua vez, procedem a busca nos seus respectivos TSox, abrangendo, assim, todo escopo celular. Esse processo é gerenciado por um módulo identificado como Busca Intracelular.

A última etapa das operações de leitura são realizadas nos TSox externos à célula; essa etapa é realizada pelo módulo de Busca Intercelular. Nesse processo, é necessária uma negociação entre as EXEHDAbases das diferentes células através dos seus respectivos serviços EXEHDA-TS, que, por sua vez, deverão distribuir as tarefas aos EXEHDAnodos internos à célula, e, posteriormente, repassar a tarefa da mesma forma às suas vizinhanças, se necessário.

4.3.3 Gerenciamento Distribuído

Um aspecto central na administração de ambientes pervasivos é o compromisso com a escalabilidade do sistema. Sendo assim, foi adotada, na especificação do EXEHDA-TS, uma estratégia de distribuição das tarefas de gerenciamento do TSvirt entre os nodos.

4.3.3.1 Instâncias do Serviço EXEHDA-TS

Com o intúito de promover a otimização das computações envolvidas com o processamento das operações distribuías bem como do uso da rede, as atividades de gerenciamento do EXEHDA-TS são distribuídas entre diferentes instâncias do serviço.

Assim como no middleware EXEHDA, o serviço EXEHDA-TS é executado em cada nodo do sistema, tendo uma instância nodal e uma base. Cada instância tem suas atribuições bem definidas, as quais são descritas a seguir:

- instância base: é executada na base da célula sendo responsável pela interação com as demais células através das suas respectivas instâncias base do serviço. Também recaem sobre ela as operações realizadas sobre os TSox armazenados na base além da distribuição das tarefas entre as instâncias nodais do serviço;

- instância nodal: é executada nos nodos e gerencia os TSox localizados no próprio dispositivo. É responsável pela computação das operações oriundas nos OX locais, repassando posteriormente as respectivas tarefas à instância base do serviço, quando necessário.

4.3.3.2 Organização das Estruturas de Dados de Gerenciamento

O gerenciamento do EXEHDA-TS é realizado de maneira distribuída entre as duas instâncias do serviço, e tem como suporte um conjunto mínimo de dados que organizam as informações estruturais do EXEHDA-TS. Esses dados são utilizados para gerenciar a composição do TSvirt assim como os respectivos *templates* associados às subscrições de eventos e consultas pervasivas submetidas pelos OX.

Para gerenciar cada TSvirt, o EXEHDA-TS utiliza um conjunto de listas de informações, as quais podem ser resumidas da seguinte forma:

- lista de composição do TSvirt: contém as referências aos TSox e são organizadas da seguinte forma:

  - nos nodos: contém os TSox localizados no nodo, que consiste no núcleo mínimo do serviço, chamado EXEHDA-TScore. Associados a cada TSox estão os ObjectID dos seus respectivos OX. Contém também as informações a respeito dos modos de sincronização utilizados. Como exemplo, tem-se a tabela 4.1;
  - na base: contém duas listas. A primeira delas é semelhante à do nodo, mencionada anteriormente, e organiza os TSox localizados na própria base. A segunda lista contém as referência aos nodos (HostID) em que estão localizados os TSox que compõem o TSvirt; esta é exemplificada pela tabela 4.2.

- lista de *templates*: contém as informações necessárias para o gerenciamento de eventos, e são organizadas da seguinte forma:

  - nos dodos: organiza os *templates* submetidos pelos OX em execução no nodo. Associado a cada *template* está o ObjectID do OX que o produziu, além da informação a respeito do tipo de operação que foi realizada. Como exemplo, tem-se a tabela 4.3;

Tabela 4.1: Gerência de um TSvirt

ObjectID	TSox	Modo Sinc.
app:122e2803965@hostid:1.ucpel	<EXEHDA-TScore>	permanente
app:518a489257d@hostid:1.ucpel	<EXEHDA-TScore>	demanda
app:971b204c7e5@hostid:1.ucpel	<EXEHDA-TScore>	demanda

Tabela 4.2: Gerência de um TSvirt na Instância Base com TSox nos Nodos

HostID
host:3,ID:{cell:ucpel}
host:5,ID:{cell:ucpel}
host:4,ID:{cell:ifsul}

– na base: contém uma lista semelhante à gerenciada pelo nodo; porém, nesta, constam todos os *templates* associados ao TSvirt. Por esse motivo, foi acrescentada a informação a respeito do nodo (HostID) em que está em execução o OX que produziu tal *template*. Tais informações são replicadas em todas as instâncias base do serviço que participam do TSvirt. Como exemplo dessa lista, tem-se a tabela 4.4.

4.3.3.3 Gerenciamento da Composição do TSvirt

A composição do TSvirt é gerenciada pelo EXEHDA-TS através das listas de dados já mencionadas, as quais contêm as informações a respeito das localizações dos TSox que constituem o TSvit. Nas listas localizadas nas instâncias nodais do serviço, cada TSox é referenciado diretamente. No entanto, em nível da instância base, o gerenciamento não é feito através dos TSox individualmente, mas sim pelos respectivos nodos em que estes estão localizados.

A referência de um TSox com sincronização sob demanda é efetivamente retirada do TSvirt apenas quando todos os demais TSox localizados no mesmo nodo e pertencentes ao mesmo TSvirt também tenham sido marcados para remoção. Nessa situação, a referência ao nodo é removida da instância base do serviço. Isto é justificado porque a redução dos custos de comunicação somente será efetiva se todos os TSox do nodo deixarem de participar do TSvirt, uma vez que a troca de informações não se dá diretamente em nível de OX, mas sim entre as instâncias do serviço.

No exemplo apresentado na tabela 4.2, pode ser observado que os nodos referenciados são de duas diferentes células: ucpel e ifsul. Isso indica que, na tabela de exemplo,

Tabela 4.3: Gerência de Eventos em um TSvirt na Instância Nodal

ObjectID	Template	Operação
app:122e2803965@hostid:1.ucpel	<“id”,10296,“oxim”,?integer>	subscribe
app:518a489257d@hostid:1.ucpel	<“id”,10296,“pres”,14..24,14..24>	subscribe
app:971b204c7e5@hostid:1.ucpel	<“id”,10296,“bat”,?integer>	find

Tabela 4.4: Gerência de Eventos em um TSvirt na Instância Base

ObjectID	HostID	Template	Operação
app:122e2803965@ hostid:1.ucpel	host:1,ID:{cell:ucpel}	<“id”,10296,“oxim”,?integer>	subscribe
app:971b204c7e5@ hostid:1.ucpel	host:1,ID:{cell:ucpel}	<“id”,10296,“bat”,?integer>	find
app:971b204c7e5@ hostid:6.ucpel	host:6,ID:{cell:ucpel}	<“id”,10916,“bat”,?integer>	find

pelo menos um dos nodos não pertence à mesma célula que a base, o que significa que existem TSox localizados em outras instâncias celulares. As informações mais específicas em relação ao TSox externo à célula são armazenadas no nodo onde o mesmo está localizado. Nesses casos, em que existe a necessidade de comunicações intercelulares no gerenciamento do TSvirt, a troca de informações entre as instâncias base do EXEHDA-TS envolvidas deve ser intermediada pelos serviços *Gateway* e *ResourceBroker* do middleware EXEHDA.

Com o suporte dos aspectos de gerenciamento discutidos, os processos de leitura realizados sobre o TSvirt iniciam pela consulta aos TSox locais do nodo em que se encontra o OX que procedeu a operação. Em seguida, a instância nodal do EXEHDA-TS repassa a tarefa à instância base do serviço, que procura pela tupla desejada nos TSox localizados na EXEHDAbase. Posteriormente, a consulta é encaminhada em paralelo às instâncias nodais dos demais nodos e às instâncias base dos outros escopos celulares que constituem o TSvirt, até que o processo seja finalizado.

Caso a consulta tenha sido processada à procura de uma única tupla, a busca é interrompida assim que ela tiver sido encontrada. No caso de busca por um bloco de tuplas, o processo é realizado até o final.

4.3.3.4 Gerenciamento de Eventos

Assim como na composição do TSvirt, as tarefas de gerenciamento de eventos no EXEHDA-TS também são distribuídas entre as duas instâncias do serviço. A instância nodal processa os eventos gerados por tuplas locais e cujas subscrições são oriundas dos OX em execução no próprio nodo. Já a instância base do EXEHDA-TS gerencia os eventos produzidos por tuplas em cujas subscrições são originadas nos demais nodos.

A subscrição de eventos pode ser realizada a partir de qualquer nodo do ambiente pervasivo, abrangendo o escopo do TSvirt. Dessa forma, torna-se necessário que esta subscrição se propague pelo ambiente, permitindo que cada instância do EXEHDA-TS envolvida tenha acesso a todos os *templates* relacionados com o TSvirt ao qual pertence.

Para gerenciar o tratamento dos eventos, os *templates* produzido são organizados em listas de acordo com os respectivos TSvirt, tanto nas instâncias nodais do serviço quanto na base. Nos nodos, constam apenas os templates cujas operações foram originadas no próprio nodo, sendo que estes são replicados na instância base. Nas demais instâncias base do EXEHDA-TS que participam do mesmo TSvirt, a lista de *templates* deve ser reproduzida.

Toda vez que uma tupla tiver sido inserida no Espaço de Tuplas, a instância nodal do EXEHDA-TS compara essa tupla com todos os templates correspondentes àquele

TSvirt e, havendo um que combine, um evento é gerado na aplicação que produziu tal *template*. Em um segundo estágio, uma cópia da tupla inserida é encaminhada para a instância base, que faz uma nova comparação com os templates lá armazenados, desconsiderando os que já foram analisados no nodo, e, assim, finalizando o processo.

A lista armazenada na instância base do EXEHDA-TS deve conter todos os *templates* produzidos pelos nodos agrupados em seus respectivos TSvirt. Isso implica que, a cada nova operação de subscrição (*subscribe* ou *unsubscribe*) ou consulta pervasiva, uma atualização em série deve ser realizada nas bases do sistema participantes de um mesmo TSvirt. Para manter a consistência das replicações, foi adotada uma estratégia de verificação do *template* na origem e uma validação no final. Assim, na medida em que uma notificação for propagada em direção ao OX solicitante, o template deve ser novamente avaliado junto ao nodo em que o OX a ser notificado se encontra em execução.

O repasse de tarefas aos nodos referenciados nas listas do serviço EXEHDA-TS que estiverem momentaneamente desconectados possuem um tratamento diferenciado. Se a comunicação não tiver sido estabelecida, uma nova tentativa é realizada um determinado tempo depois. Após sucessivas tentativas sem sucesso, o *timeout* associado à operação poderá ser ultrapassado; nesse caso, a tarefa é abortada. A definição do tempo referente ao intevalo entre as tentativas de acesso ao nodo, assim como do *timeout*, é deixada a cargo do desenvolvedor de aplicações através da operação *adjustTime* (vide seção 5.2).

4.3.4 Tratamento da Mobilidade e Desconexão

São discutidos, nesta seção, os compromissos do EXEHDA-TS com os requisitos mobilidade e desconexão, características intrínsecas das aplicações pervasivas gerenciadas pelo middleware EXEHDA.

Considerando que os aspectos de rede de baixo nível são gerenciados pelo middleware EXEHDA através do serviço CIB, na medida em que um nodo se move através da infraestrutura, enquanto não existir alteração no seu escopo celular, não são necessários ajuste nas referências dos TSox; apenas torna-se necessário uma bufferização das operações remotas. Assim, durante os períodos de desconexão, as operações locais são mantidas funcionais enquanto aquelas que envolvem a rede são armazenadas em uma fila e são processadas após a reconexão.

No entanto, se existir mobilidade física com a troca do escopo celular, são necessários ajustes na lista de hosts junto à instância base do EXEHDA-TS (vide tabela 4.2), alterando o HostID do nodo e apontando este para outra célula. Também, nesse caso, é preciso realizar alterações na lista de *templates* armazenada na instância base do serviço, ajustando também o HostID do nodo para que este seja localizado corretamente no caso da ocorrência de algum evento (vide tabela 4.4).

Uma das características do middleware EXEHDA é o tratamento à mobilidade lógica através dos OX, nos quais há capacidade de migrar entre os nodos do ambiente pervasivo. Nesse sentido, é realizada uma operação combinada com o serviço *Executor*, que deve notificar o EXEHDA-TS a respeito da movimentação do OX, repassando a sua nova localização em relação à infraestrutura. A posição física do OX deve ser gerenciada pelo EXEHDA-TS, migrando, juntamente ao OX, o TSox correspondente. Durante esse processo, pode haver novamente a necessidade de reajustes dos HostID tanto na lista de gerenciamento do TSvirt quanto na de *templates*; porém, em tal circunstância, isso deve ser feito tanto na instância base quanto na nodal do serviço.

Os aspectos relacionados à mobilidade justificam a preferência por manter o TSox

referenciado diretamente ao OX, que decorre principalmente devido ao compromisso em manter a operacionalidade do serviço, mesmo nos momentos de desconexão. Nessas situações, as operações sobre o Espaço Tuplas são mantidas, porém de forma local, enquanto as operações remotas são bufferizadas para serem processadas no novo destino do OX.

Umo outro elemento importante em relação ao tratamento da desconexão é a utilização do serviço *worb* do EXEHDA nas operações sobre os TSox remotos; tal serviço utiliza a abstração canal que é herdada através do serviço *dispatcher*. Os canais consistem em mecanismos ponto-a-ponto, estabelecidos entre EXEHDAnodos que necessitam de comunicação, sendo sinônimo das propriedades de garantia da entrega e ordenamento das mensagens no EXEHDA. Suas características foram discutidas na seção 3.2.4.

4.4 Discussão

As características de elevada distribuição e mobilidade presentes na Computação Pervasiva impulsionam a utilização dos modelos de coordenação baseados em Espaço de Tuplas nesses cenários. Entres os principais aspectos favoráveis ao uso dos mesmos no desenvolvimento e manutenção de aplicações em tais ambientes, estão o desacoplamento temporal e referencial, além do endereçamento associativo. Outro fator importante é que os modelos de coordenação promovem a separação entre o desenvolvimento das aplicações e as tarefas de gerência da infraestrutura, simplificando os processos. O Espaço de Tuplas ainda pode ser usado como canal para a disseminação de informações contextuais de forma a simplificar o acesso às mesmas por parte dos serviços de tratamento de contexto.

Esta dissertação de mestrado apresenta o EXEHDA-TS, que consiste em um modelo de coordenação proativo e escalável baseado em Espaço de Tuplas distribuído, concebido como um serviço para o middleware EXEHDA. Este trabalho vem contemplar uma demanda do projeto, prevista quando da sua concepção. Nesse sentido, o serviço EXEHDA-TS disponibiliza um ambiente de alto nível com o intuito de estimular as atividades cooperativas entre os componentes das aplicações. Considerando os desafios enfrentados no desenvolvimento de modelos de coordenação para o ambiente pervasivo do middleware EXEHDA, foram adotadas estratégias que promovam a escalabilidade, a gerência das desconexões e a mobilidade física e lógica.

No EXEHDA-TS, cada OX tem o seu próprio Espaço de Tuplas, chamado TSox, que consiste no núcleo mínimo do EXEHDA-TS. Os diversos TSox podem ser agrupados formando um ambiente virtual chamado TSvirt, este associado às aplicações; o TSvirt é formado por um conjunto de referências aos TSox distribuídos pelos nodos do sistema. Do ponto de vista das aplicações, o TSvirt constitui um Espaço de Tuplas compartilhado que aglutina tuplas provenientes dos componentes de software distribuídos pelos nodos do sistema.

Esse modelo foi adotado em detrimento ao modelo distribuído com replicação (vide seção 2.2.3), por ser considerado mais oportuno no caso de sistemas heterogêneos de larga escala e de alcance global (TANENBAUM; STEEN, 2007). Os modelos tradicionais que utilizam estratégias de replicação de tuplas tornam-se inviáveis quando o número de nodos, de processos em execução e a quantidade de dados são elevados. A restrição de recursos, tipicamente presente em diversos dispositivos presentes nos ambientes pervasivos, pode inviabilizar o número de informações que deverão ser

armazenadas.

A abrangência do Espaço de Tuplas deve ser limitada para não comprometer o desempenho envolvido no seu gerenciamento; na modelagem do EXEHDA-TS, foram adotadas algumas estratégias nessa direção. A primeira delas foi a delimitação do Espaço de Tuplas ao escopo das aplicações, sendo o TSvirt, então, compartilhado entre os componentes de software que a constituem.

A segunda estratégia foi a concepção de dois diferentes modos de sincronização entre os TSox e o TSvirt: permanente ou sob demanda. No modo de sincronização permanente, uma vez o TSox tenha sido integrado ao TSvirt, somente deixará de ser acessado nos momentos de desconexão, independentemente dos destinos do OX decorrentes de processos de migração. Já no modo de sincronização sob demanda, o TSox apenas sincroniza com o TSvirt por determinados momentos, a fim de realizar atividades específicas, o que reduz os custos de comunicação.

Por fim, a terceira estratégia adotada em prol do desempenho foi a restrição da amplitude do espaço de dados a ser gerenciado. A fim de favorecer a escalabilidade do sistema, o EXEHDA-TS agrupa os Espaços de Tuplas distribuídos de acordo com o perfil de cada um, compondo um TSvirt com características específicas determinadas pela aplicação.

O EXEHDA-TS ainda disponibiliza mecanismos que promovem a reatividade do Espaço de Tuplas, que são mapeados em duas operações básicas: *subscribe* e *find*. A primeira refere-se à subscrição de eventos, e a segunda às consultas pervasivas. Tais estratégias promovem a redução da carga computacional das aplicações e do uso da rede, por liberar os componentes de software da realização de sucessivas consultas ao Espaço de Tuplas quando na procura por tuplas do seu interesse.

5 EXEHDA-TS: PROTOTIPAÇÃO E ESTUDOS DE CASO

A prototipação do EXEHDA-TS e os estudos de caso são os assuntos tratados neste capítulo. Nos cenários propostos, foram avaliadas as funcionalidades referentes ao acesso pervasivo aos dados, a reatividade do modelo, bem como o suporte à desconexão e à mobilidade física e lógica.

Na primeira seção, são abordadas as principais tecnologias utilizadas na prototipação. Em seguida, são apresentados os detalhes em relação às operações disponibilizadas pelo EXEHDA-TS. Por fim, são discutidos os estudos de caso, em que foram considerados cenários da área médica.

5.1 Tecnologias Utilizadas na Prototipação do EXEHDA-TS

Esta seção trata dos aspectos de prototipação do EXEHDA-TS, caracterizando as principais tecnologias utilizadas na sua implementação.

5.1.1 Aspectos de Programação

A linguagem Java foi utilizada no desenvolvimento dos serviços do EXEHDA, situação esta prevista desde sua concepção, enquanto middleware do projeto ISAM (YAMIN, 2004); esse aspecto foi a principal motivação para a sua utilização também no desenvolvimento do serviço EXEHDA-TS. Dentre as características de Java entendidas como oportunas para o desenvolvimento do EXEHDA-TS, destacamos as seguintes:

- **portabilidade:** na Computação Pervasiva, um aspecto presente é a heterogeneidade de software e hardware. Sendo assim, a independência de plataforma é uma característica do Java que favorece a sua utilização nesses ambientes. Java é uma linguagem de programação orientada a objetos, logo os programas são constituídos por um conjunto de classes. Compilando tais classes, o código fonte é convertido para uma representação no formato de *bytecodes*. Essa sequência de *bytecodes* é interpretada e executada através de um elemento intermediário que apresente uma interface uniforme sobre as diferentes arquiteturas de hardware existentes; tal elemento é a Máquina Virtual Java (JVM). Essa abordagem transfere a manutenção da portabilidade da linguagem para a JVM e esta sim deve ser implementada e

compilada para cada plataforma alvo a ser suportada. Atualmente, versões da JVM são encontradas nas mais diversas plataformas, tais como Windows, Linux, Solaris, MacOS, além de versões desenvolvidas especificamente para dispositivos embarcados, como *Smartphones*;

- **carga dinâmica de código:** o processo de carga de classes é feito de forma dinâmica na JVM, sendo controlado pelo *Class Loader*. Este é um objeto plugável do sistema, podendo ser personalizado de forma a implementar a carga de classes, por exemplo, a partir do sistema de arquivos da máquina, ou de um servidor HTTP na Internet, ou mesmo através de um banco de dados (YAMIN, 2004);

- **segurança:** a máquina Java (JVM) executa as aplicações realizando uma forte verificação dinâmica sobre as classes Java, controlando os acessos à memória para evitar possíveis problemas de segurança. Na plataforma Java, também são controlados os acessos aos recursos do sistema operacional hospedeiro (sistema de arquivos, periféricos, *sockets*), sendo gerenciados pela JVM e não acessados diretamente pela aplicação. Esses aspectos favorecem a distribuição de código em um sistema com grande heterogeneidade de hardwares, sem que comprometa a segurança dos nodos;

- **concorrência e sincronização:** o *framework* Java disponibiliza de forma nativa em sua API um controle da concorrência através da classe Thread. Esse aspecto reforça o uso da linguagem nos ambientes heterogêneos da Computação Pervasiva por favorecer a portabilidade das aplicações que necessitam desse recurso. Outras linguagens, como C e C++, muitas vezes necessitam de bibliotecas para dar suporte à concorrência, e essas, em muitos casos, são específicas para determinadas plataformas;

- **produtividade no desenvolvimento estruturado de software:** em Java, vários aspectos tendem a beneficiar o processo de desenvolvimento de software de um modo geral. A independência de plataforma reduz os custos de desenvolvimento e manutenção por não necessitar de versões do código específicas para cada plataforma. A suscetibilidade de erros é reduzida devido a um gerenciamento automático de memória, à inexistência de aritmética de ponteiros e a um tratamento estruturado de exceções. Por ser uma linguagem orientada a objetos, o Java favorece o encapsulamento de dados e componentes. Somado a isso, o Java apropriou-se de características interessantes de outras linguagens de programação, como C e C++, o que resultou em uma linguagem com pouca ocorrência de erros e de fácil aprendizado;

- **suporte ao desenvolvimento de aplicações distribuídas através de RMI:** este tópico, devido a sua impotância na prototipação do EXEHDA-TS, será abordado na seção 5.1.2, a seguir.

5.1.2 Aspectos da Distribuição das Computações

A invocação de métodos remotos (*Remote Method Invocation* - RMI) é um recurso disponível em Java para dar suporte à computação distribuída baseado em RPC (DEITEL; DEITEL; SANTRY, 2002). A RMI permite que objetos Java em execução em JVMs distintas comuniquem-se entre si via chamadas de método remoto, estejam elas no mesmo nodo ou em nodos distintos distribuídos pela rede.

Em RMI, uma vez que um método de um objeto Java tenha sido registrado como sendo remotamente acessível, um cliente pode localizar esse serviço recebendo uma referência a ele, passando então a estar habilitado a executar o método de forma remota. A maneira de chamar os métodos remotos não difere daquela utilizada nas chamadas de método da mesma aplicação. A ordenação dos dados que trafegam entre os clientes e o servidor é tratada de forma transparente pela RMI. No caso destes dados serem objetos com estruturas de dados complexas, é utilizado um mecanismo de serialização de objeto.

Na prototipação do EXEHDA-TS, uma forte motivação ao uso do Java surgiu em decorrência da existência dessa tecnologia. RMI foi empregada nas comunicações entre processos em execução no mesmo nodo, os quais consistem nas trocas de dados entre os OX e a respectiva instância nodal do EXEHEDA-TS. Embora a RMI dê suporte à comunicação pela rede, essa é tratada de forma síncrona, sendo uma característica indesejada nos casos de desconexão, típicos nos ambientes da Computação Pervasiva. Esse problema foi uma preocupação na concepção do middleware EXEHDA e, em decorrência disso deu-se origem ao serviço *worb* (vide seção 3.2.4), que tem como foco a invocação de métodos remotos com suporte à desconexão. Porém, a utilização desse serviço apenas se justifica em comunicações via rede devido ao alto grau de abstração utilizado, sendo empregado apenas nas comunicações entre as diferentes instâncias do serviço EXEHDA-TS.

5.1.3 Aspectos do Espaço de Tuplas

LighTS é uma implementação em Java minimalista e não distribuída do conceito de Espaço de Tuplas, que foi originalmente desenvolvido para servir como suporte para o middleware LIME (BALZAROTTI; COSTA; PICCO, 2007); LighTS é *open source* e tem seus fontes disponíveis em (PICCO, 2009). No mesmo endereço, também existe uma versão pré compilada acompanhada do Javadoc, com a documentação de sua API.

Um dos pacotes disponíveis na API do LighTS contempla as operações tradicionais definidas em Linda. Entre elas, estão as operações `out`, para inserir tuplas, `in` e `rd`, para pesquisar de forma bloqueante e, ainda, `inp` e `rdp`, para pesquisa não-bloqueante. Complementando-nas, a API disponibiliza operações para manipulação de tuplas em massa `outg`, `ing`, `rdg` (PICCO; BALZAROTTI; COSTA, 2005).

Nos modelos da coordenação baseados em Espaço de Tuplas, tradicionalmente, as consultas são realizadas através de um *template* que deve conter os tipos ou valores exatos de alguns dos campos da tupla desejada. Como esse método mostra-se insuficiente diante das necessidades das aplicações conscientes do contexto (PICCO; BALZAROTTI; COSTA, 2005), o LighTS disponibiliza um conjunto de funcionalidades voltadas a tais demandas. Assim, (PICCO; BALZAROTTI; COSTA, 2005) aponta a necessidade de métodos de combinações adicionais aqueles oferecidos nas operações de consulta através dos *templates* tradicionais. Essas funcionalidades são mapeadas em três grupos:

- combinação por faixa de valores: possibilita a consulta por uma faixa de valores através da classe `RangeField`, que permite definir os limites inferior e superior para o valor do campo. Esse aspecto é bastante interessante em aplicações sensíveis ao contexto, por exemplo, nos casos do projeto PERTMED, possibilitando que seja gerado um alerta quando a temperatura de um determinado paciente tenha ultrapassado um valor pré estipulado, que pode ser obtido definindo uma faixa de valores do tipo 37°C – 41°C;

- combinação através de lógica fuzzy: em alguns casos, uma consulta através de uma faixa de valores pode não ser suficiente pela ausência de meios para uma definição precisa desta faixa. As imprecisões ou incertezas nas informações a serem consultadas podem ser tratadas através de lógica fuzzy, utilizando operações disponibilizadas na API do LighTS para essa funcionalidade;

- agregação de dados: permite uma consulta construindo uma forma de correspondência com base nos valores de dois ou mais campos. Para viabilizar tal funcionalidade, o LighTS utiliza um *template* associado ao conceito de tupla virtual, cujos campos podem conter a associação de um ou mais campos, que deve existir na tupla a ser localizada.

Para implementar o protótipo do EXEHDA-TS, foram definidos os requisitos que deveriam estar presentes nas tecnologias adotadas: (*i*) ter um baixo custo computacional, (*ii*) ser implementado em Java e (*iii*) ter código fonte disponível (*open source*). Pode-se observar que o LighTS, além de satisfazer tais requisitos, apresenta outras funcionalidades que foram determinantes para a sua utilização na prototipação realizada. Assim, o LighTS foi utilizado como base para o núcleo do EXEHDA-TS, sendo o componente de armazenamento relacionado ao TSox e do qual foram herdadas algumas das operações disponibilizadas na sua API.

5.2 Operações Disponibilizadas no EXEHDA-TS

Para que as operações definidas para o EXEHDA-TS atendam as funcionalidades previstas para o modelo proposto, as mesmas são mapeadas como métodos das seguintes classes básicas:

- `ETupleSpace`: classe que implementa o espaço de tuplas no EXEHDA-TS. O seu construtor cria o TSox e, através dele, são realizadas todas as operações sobre o Espaço de Tuplas;

- `ETuple`: as instâncias dessa classe constituem as tuplas que são armazenadas no Espaço de Tuplas;

- `EField`: os objetos instanciados a partir dessa classe consistem nos campos que compõem cada tupla.

Tal estrutura de classes, cujo conjunto forma a API do EXEHDA-TS, é representada na figura 5.1, em que se tem uma visão gráfica de que o Espaço de Tuplas é uma instância da classe ETuplesSpace, que é constituída por tuplas as quais são instâncias da classe ETuple. As tuplas, por sua vez, são compostas por objetos do tipo EField.

No EXEHDA-TS, a criação de Espaço de Tuplas é feita através do construtor da classe ETupleSpace. Ao criar um Espaço de Tuplas no EXEHDA-TS, pode ser utilizada uma *flag*, que indica o modo de sincronização: (*i*) permanente ou (*ii*) sob demanda. Por padrão, o modo de sincronização é sob demanda; caso seja necessária a manutenção permanente da referência junto ao TSvirt, isso deve ser informado explicitamente. O processo de criação do Espaço de Tuplas consiste basicamente na criação do TSox e em sua

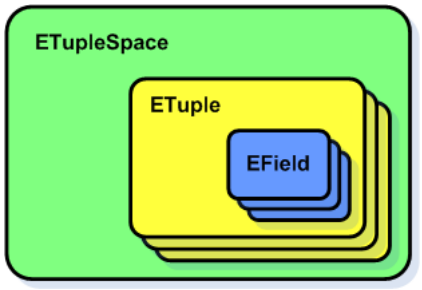

Figura 5.1: Organização das Classes da API o EXEHDA-TS

incorporação ao TSvirt da aplicação que tenha o mesmo perfil. Nesse caso, o EXEHDA-TS realiza uma interação com o serviço *Executor* do EXEHDA para capturar o ObjectID do OX ao qual o TSox pertence.

Para alternar os modos de sincronização do Espaço de Tuplas entre permanente e sob demanda, é utilizada a operação `redefine`, na qual deve ser passado, como parâmetro, o modo de sincronização desejado. Sendo assim, o desenvolvedor deve avaliar as necessidades da aplicação e realizar a operação mencionada quando for necessário.

Para a inserção de tuplas, o EXEHDA-TS disponibiliza a operação `insert`, que poderá ser usada tanto para uma tupla individual como para um *array* de tuplas, dependendo do tipo de parâmetro de entrada. Nessa operação, as tuplas inseridas podem ser acompanhadas por um parâmetro adicional, utilizado para manter a referência do TSox junto ao TSvirt enquanto as tuplas inseridas não tenham sido lidas. Esse procedimento garante que a tupla possa ser lida por OX remotos mesmo que o modo de sincronização utilizado seja sob demanda.

As operações `read` e `consume` são disponibilizadas para leitura ou consumo de tuplas, sendo que `consume` remove a tupla do Espaço de Tuplas enquanto que a operação `read` a mantém; As duas últimas operações são propagadas pelo sistema a todo o escopo do TSvirt. Adicionalmente, de maneira semelhante ao serviço *Discover* do EXEHDA, essas operações incorporam um parâmetro que define a amplitude da pesquisa. Nesse sentido, os seguintes valores são aplicados:

1. nodo local;
2. na base;
3. demais nodos da célula local;
4. vizinhança da célula local;
5. vizinhança completa da célula local, nível 1;
6. vizinhança completa da célula local, nível 2.

As operações `read` e `consume` têm comportamento não bloqueante, o que significa que, se não for localizada alguma tupla que satisfaça o padrão informado, a operação retorna um elemento vazio. As operações bloqueantes tradicionalmente usadas em sistemas baseados em Espaço de Tuplas não são aplicadas ao EXEHDA-TS. Nessa classe de operação, se nenhuma tupla que satisfaça o *template* for encontrada, o processo

Tabela 5.1: Operações do EXEHDA-TS

Operação	Descrição
`redefine`	Alterna entre os modos de sincronização permanente e sob demanda.
`insert`	Inserção de tuplas.
`read`	Lê uma ou mais tuplas sem retirar do espaço de tuplas.
`consume`	Lê uma ou mais tuplas retirando do espaço de tuplas.
`subscribe` e `unsubscribe`	Faz e retira a subscrição de um evento.
`find`	Submete uma consulta pervasiva.
`adjustTime`	Ajusta os tempos associados ao tratamento dado aos nodos desconectados.

fica aguardando até que uma seja inserida no Espaço de Tuplas ou até que um *timeout* tenha sido ultrapassado. Um resultado semelhante é obtido no EXEHDA-TS através da utilização de uma consulta pervasiva; porém, nesse caso, a resposta é obtida por intermédio de um evento ao invés de manter a aplicação bloqueada.

Além das operações de inserção, leitura e consumo de tuplas, tradicionalmente existentes em sistemas semelhantes, o EXEHDA-TS também disponibiliza operações que dão reatividade ao modelo. Aplicações podem ser notificadas quando uma nova tupla que combine com um determinado *template* for inserida no Espaço de Tuplas através de eventos. Essas operações são semelhantes às utilizadas em sistemas *publish/subscribe*, e são usadas para subscrever um evento e retirar a subscrição: `subscribe` e `unsubscribe`. Para implementar consultas pervasivas, é utilizada a operação `find`. Esta funciona de forma semelhante a uma subscrição, porém, um evento é gerado apenas uma vez. Ambas as operações são realizadas no escopo do TSvirt, mas a operação `find`, especificamente, incorpora, além do *template*, um parâmetro relativo à amplitude da busca de forma idêntica ao que foi definido nas operações `read` e `consume`, descritas anteriormente.

Os tempos relacionados às tentativas de acesso aos nodos referenciados no TSvirt que se encontram desconectados são ajustados através da operação `adjustTime`. Nesta são passados dois parâmetros: o primeiro diz respeito ao intervalo de tempo entre as sucessivas tentativas, enquanto o segundo refere-se ao *timeout* correspondente ao tempo total de espera pela possível reconexão do nodo.

A tabela 5.1, resume as operações disponibilizadas pela API do EXEHDA-TS, discutidas nesta seção.

5.3 Estudos de Caso: Cenários Direcionados à Medicina

As pesquisas realizadas para a consolidação do EXEHDA-TS foram desenvolvidas de forma integrada ao projeto PERTMED, cujo objetivo é conceber um sistema de Telemedicina para disponibilizar informações da área médica de maneira pervasiva. Com o intuito de potencializar a integração com o PERTMED, foram definidos alguns cenários baseados em ambientes da medicina, para avaliar o comportamento do EXEHDA-TS.

Os cenários propostos basieam-se em situações ocorridas em um ambiente hospitalar. A previsão é que, na continuidade do projeto PERTMED, esse ambiente seja parcialmente reproduzido em um hospital real e, então, além do retorno acadêmico, possa haver também um retorno social dos resultados à comunidade usuária em geral.

No ambiente hospitalar dos cenários de estudo, os pacientes considerados em situação de saúde crítica são monitorados através de um conjunto de sensores, e estes se comunicam com o sistema informatizado da unidade médica através de conexões semfio; os dados provenientes dos sensores alimentam dinamicamente o Espaço de Tuplas do EXEHDA-TS. Caso a aplicação médica em execução sobre o middleware detecte que algum paciente esteja prestes a ter uma crise, o EXEHDA-TS reage prevendo suporte para a geração de um alerta nos computadores da enfermaria mais próxima, fornecendo as informações a respeito do problema detectado. Dependendo da gravidade identificada, seja pela equipe de plantão ou pelo próprio sistema, pode ser enviado todo o prontuário médico do paciente, em formato digital, para o *smartphone* do médico responsável pela equipe de atendimento de urgência do hospital. O EXEHDA-TS provê suporte também a esse tipo de operação.

O sistema computacional do hospital possui um complexo Sistema de Informações, responsável por manter a organização dos dados, como registro e histórico dos pacientes, registro dos profissionais, gerenciamento financeiro e administrativo, entre outros. O EXEHDA-TS, por sua vez, constitui uma infraestrutura de apoio, responsável pelo gerenciamento *on-line* pervasivo das informações monitoradas nos pacientes, dando suporte ao pronto repasse dos dados aos profissionais responsáveis, bem como ao Sistema de Informações.

A infraestrutura computacional prevê a existência de uma EXEHDAcel abrangendo todo o hospital. Demais hospitais próximos são gerenciados de forma a constituírem outras EXEHDAcel, formando o ambiente pervasivo. Tanto o sistema computacional embarcado, que permanece junto ao paciente coletando as informações dos sensores, quanto os computadores dos hospitais, os computadores pessoais e *smartphones* dos médicos e dos demais agentes de saúde constituem nodos do sistema (EXEHDAnodos).

O software médico desenvolvido para avaliar o protótipo do EXEHDA-TS possui uma interface gráfica cujas telas são apresentadas nas figuras 5.2, 5.3 e 5.4. O Prontuário Eletrônico desenvolvido prevê a utilização do EXEHDA-TS para suporte à aquisição e acesso *on-line* das informações produzidas. Nesse caso, o Sistema de Informações já existente no hospital aglutinaria os dados de forma persistente, podendo ser utilizado para armazenar o histórico dos pacientes, além de informações administrativas, como dados de cadastro tanto de pacientes como dos profissionais de saúde. Dessa maneira, a reatividade e a pervasividade dos dados promovida pelo EXEHDA-TS é utilizada para gerenciar as informações e os eventos de interesse conforme eles acontecem, enquanto que através do Sistema de Informações, o profissional pode acessar o prontuário do paciente para auxiliar na tomada de decisão.

Para avaliar o comportamento do EXEHDA-TS em situações de uso, será considerado o caso em que um determinado paciente deu entrada no hospital com um quadro clínico crítico, e, logo após de ser atendido e medicado, o mesmo recebeu um conjunto de sensores para monitorar os sinais vitais.

Os três cenários a seguir descrevem possíveis interações e movimentações do paciente e do médico responsável em relação ao ambiente pervasivo, a fim de avaliar o com-

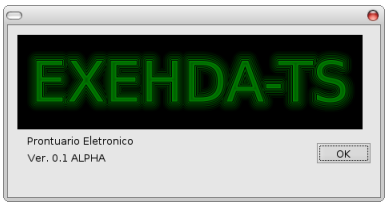

Figura 5.2: Tela de Apresentação do Prontuário Eletrônico

Figura 5.3: Tela de Monitoramento on-line do Prontuário Eletrônico

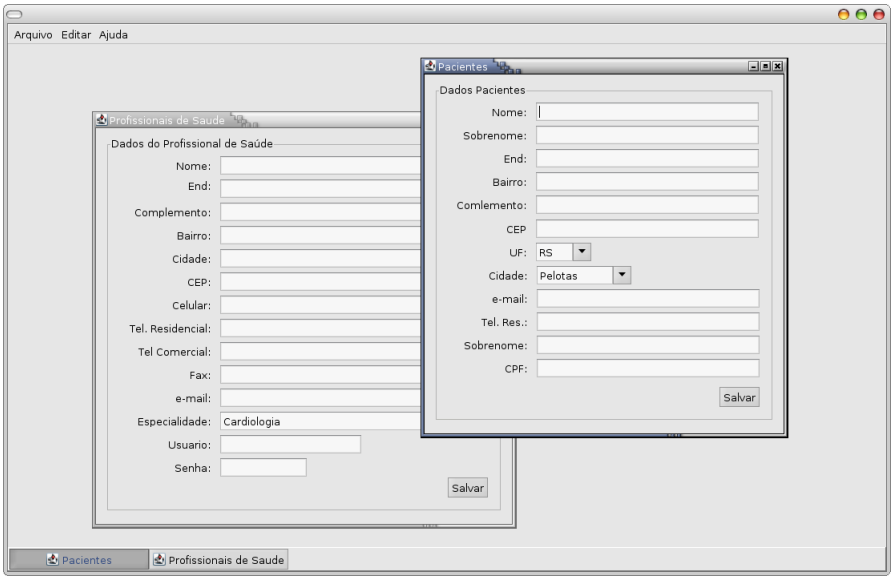

Figura 5.4: Telas para o Cadastro de Pacientes e Profissionais de Saúde

portamento do EXEHDA-TS no atendimento às demandas desses cenários da medicina pervasiva.

5.3.1 Cenário 1: Paciente Desloca-se Dentro do Escopo Celular

A primeira situação prevê a necessidade de um deslocamento do paciente dentro do hospital, para realizar um determinado exame. No deslocamento, o paciente sofre uma crise que é detectada através do sistema de sensoriamento. Em poucos instantes, o médico, que estava usando seu computador móvel pessoal nas dependências do hospital, recebe um alerta do sistema informando o problema. Então, para avaliar o caso, através da interface do Prontuário Eletrônico, o médico solicita outros sinais vitais do paciente aquisitados no momento da crise.

5.3.1.1 Objetivo do Cenário

Esse cenário tem por objetivo avaliar o comportamento do EXEHDA-TS em situações em que existe a mobilidade física de um nodo no escopo da célula. Para tanto, serão observadas as seguintes funcionalidades:

- desacoplamento temporal e referencial;
- reatividade;
- busca distribuída.

5.3.1.2 Caracterização do Cenário

O cenário apresentado destaca a possibilidade de existir mobilidade física tanto do paciente quanto do médico. Porém, para avaliar esse caso, será considerado que, enquanto o paciente se desloca pelo hospital, o médico permanece na mesma posição em relação à infraestrutura.

Conforme o EXEHDAnodo móvel associado ao paciente se desloca pela célula, pode eventualmente ocorrer perda momentânea do sinal, ou mesmo a troca do ponto de acesso à rede. Isso constitui um cenário em que acontecem desconexões momentâneas do nodo móvel seguidas de uma realocação do dispositivo em relação à rede, caracterizando uma rede com composição dinâmica.

Um outro cenário cujo gerenciamento do EXEHDA-TS teria demandas semelhantes seria o de o médico mover-se pelo hospital enquanto o paciente permanece na mesma posição.

5.3.1.3 Análise Sistematizada dos Comportamentos

Esta seção apresenta a análise do comportamento do EXEHDA-TS de acordo com a avaliação dos resultados obtidos.

Requisito Desacoplamento Temporal e Referencial

Conforme o paciente desloca-se pelo hospital, seu EXEHDAnodo móvel eventualmente pode migrar de uma subrede para outra, possivelmente alterando seu endereçamento em relação à infraestrutura. Esse aspecto é gerenciado pelo middleware EXEHDA através de seu subsistema de acesso pervasivo, provendo um identificador

de alto nível para o nodo (HostID) único no escopo celular, abstraindo os aspectos de endereçamento de rede.

Durante os eventuais momentos de desconexão, o EXEHDA-TS faz a bufferização das operações que tiverem sido realizadas pelos OXs de monitoramento que acompanham o paciente, e, posteriormente, quando a conexão é reestabelecida, as operações são processadas. Observa-se que mesmo um dos envolvidos estando momentaneamente impossibilitado de ser acessado, a comunicação é oportunizada pelo EXEHDA-TS, pois o processo mantém suas operações locais. Sendo assim, as informações produzidas durante o período de desconexão podem ser acessadas posteriormente.

O EXEHDA-TS, por sua vez, constitui uma referência para os dados sensorados independentemente da localização ou da movimentação do produtor em relação à infraestrutura, abstraindo os aspectos de localidade e simplificando as tarefas de desenvolvimento da aplicação.

Requisito Suporte à Reatividade

A reatividade provida pelo EXEHDA-TS pode ser utilizada como suporte básico ao mecanismo de sensibilidade ao contexto do middleware, viabilizando uma aplicação que permita que o médico seja notificado da ocorrência de algum problema com o paciente. A funcionalidade disponível no serviço para essa finalidade é a subscrição de eventos, disparada pelo médico por intermédio do Prontuário Eletrônico através de uma operação *subscribe*.

Um exemplo de *template* fornecido na subscrição pode ser <"João da Silva","id",10296,"pressão arterial",14..24,14..24>. Dessa forma, se uma determinada tupla que combine com esse *template* for identificada, um evento é gerado no ambiente computacional do médico. Um exemplo de tupla seria: <"João da Silva", "id",10296, "pressão arterial", 22, 17>. No *template* do exemplo, os campos que têm a indicação "14..24" correspondem a uma faixa de valores que deve estar presente no campo da tupla, para que esta seja retornada.

A subscrição feita através da interface do componente de software disponibilizado ao médico é inserida na tabela de *templates* alocada na instância nodal do EXEHDA-TS (vide seção 4.3.3), que, por sua vez, repassa uma cópia à instância base. O TSox associado à interface de software utilizada pelo médico eventualmente pode pertencer a um TSvirt que tenha componentes em mais de uma célula. Nesse caso, o *template* deve ser propagado também às demais instâncias base do serviço EXEHDA-TS em operação nas outras células. Essa estratégia promove a escalabilidade do sistema através da distribuição das tarefas de gerenciamento dos eventos entre os nodos que compõem o TSvirt.

Cada vez que uma tupla associada ao paciente é inserida no TSox, esta é comparada com os *templates* gerenciados pela instância nodal do EXEHDA-TS. Em seguida, uma cópia da tupla é repassada para a instância base, em que é comparada com os demais *templates*. Como na instância base do EXEHDA-TS são armazenados todos os *templates* associados aos TSvirts gerenciados pela célula, e o processo é finalizado. Quando o *template* submetido pelo médico é localizado, o serviço produz um evento no OX associado ao mesmo, repassando a tupla identificada. Esse mecanismo é apresentado na figura 5.5.

Nessa análise, verifica-se que, mesmo que o EXEHDAnodo móvel do paciente mude sua posição em relação à infraestrutura da célula, o médico permanece recebendo as notificações do sistema sem que seja necessário nenhum procedimento extra por parte do mesmo. A abstração promovida pelo EXEHDA-TS através do TSvirt permite que a

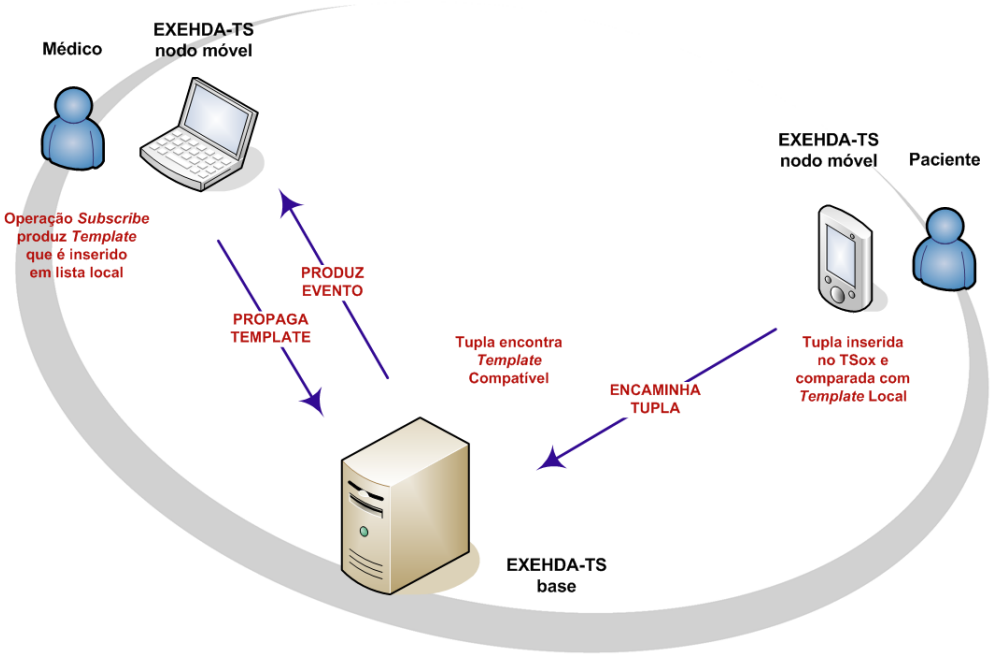

Figura 5.5: Fluxo de Processamento de Evento

aplicação continue operacional independentemente da dinâmica topológica da rede devido às movimentações dos envolvidos em relação à infraestrutura da célula.

Requisito Busca Distribuída

Após o recebimento de uma notificação do sistema devido a um problema no estado clínico do paciente, o médico solicita outras informações atuais a respeito da situação deste. Para tal, através do Prontuário Eletrônico, é realizada uma busca por tuplas associadas ao paciente, utilizando a operação `read`. Um exemplo de template a ser utilizado seria <“João da Silva”,“id”,10296,“batimentos”,?Integer>.

O processo de busca inicia de forma local, verificando os TSox gerenciados pela instância nodal do EXEHDA-TS (vide tabela 4.1). Não sendo encontrada nenhuma tupla que combine com o *template* fornecido, a tarefa é repassada à instância base do serviço. Esta faz a comparação com os TSox armazenados nela própria e, caso ainda não tenha sido localizada a tupla, a busca é encaminha às instâncias nodais do EXEHDA-TS dos demais nodos participantes do TSvirt (vide tabela 4.2). No cenário analisado, a tupla de interesse encontra-se no EXEHDAnodo do paciente, que retorna o resultado ao dispositivo do médico finalizando, então, o processo de busca. Esste mecanismo pode ser observado através da figura 5.6.

5.3.2 Cenário 2: Médico Troca de Dispositivo

Neste cenário, o médico tem a possibilidade de utilizar o Prontuário Eletrônico através de diferentes computadores disponíveis nas instalações do hospital.

Para acompanhar um determinado paciente com quadro clínico crítico, o médico configura o Prontuário Eletrônico para ser avisado caso algum problema venha a acontecer com o paciente. Momentos depois, ele se desloca para outro setor do hospital em que utiliza um computador diferente para acessar o seu AVU (Ambiente Virtual do Usuário -

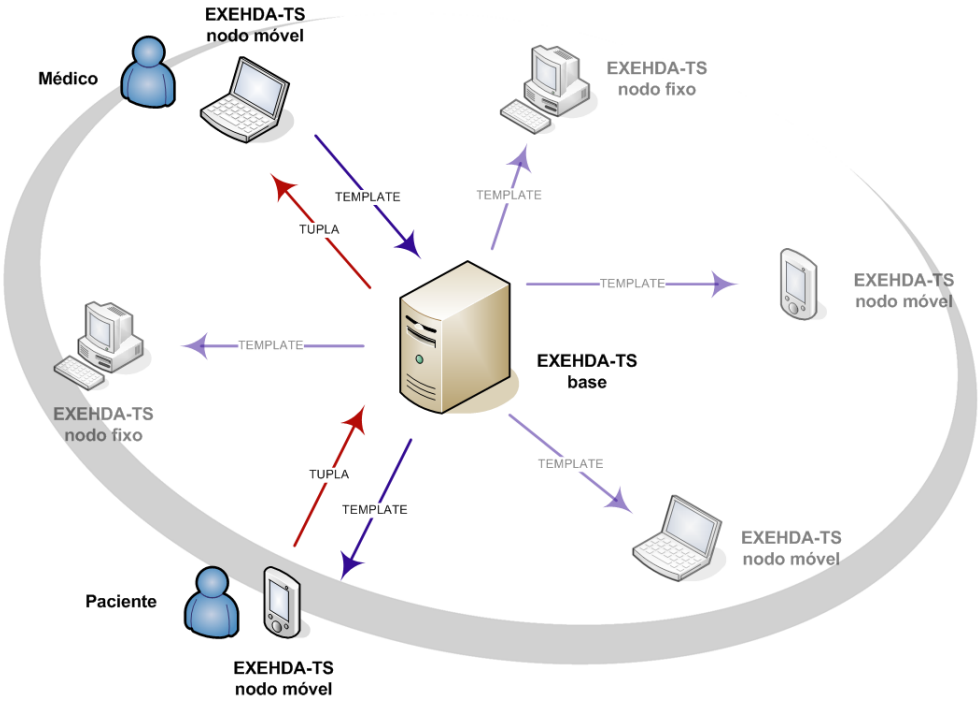

Figura 5.6: Fluxo do Processo de Busca Distribuída em Nível Celular

vide seção 3.2.5) que, por sua vez, permite operar o Prontuário Eletrônico.

Em um determinado momento, o paciente sofre uma crise, detectada através do sistema de sensoriamento que o acompanha. Em instantes, o médico é avisado do ocorrido por meio da interface do Prontuário Eletrônico, e, imediatamente, desloca-se ao encontro do paciente para tomar as providências necessárias.

5.3.2.1 Objetivo do Cenário

O objetivo deste cenário é avaliar o comportamento do EXEHDA-TS diante da ocorrência de mobilidade lógica por parte de um componente de software participante de um TSvirt. Nesse caso, as funcionalidades do EXEHDA-TS consideradas importantes na avaliação são as seguintes:

- desacoplamento referencial;
- reatividade.

A funcionalidade de busca distribuída, nessa circunstância, comporta-se de maneira semelhante àquela avaliada no cenário anterior.

5.3.2.2 Caracterização do Cenário

O cenário analisado caracteriza-se pela presença de mobilidade lógica do ambiente computacional utilizado pelo médico. Inicialmente, o componente de software associado à interface do médico encontra-se em execução em um determinado EXEHDAnodo e, posteriormente, quando o médico acessa seu AVU através de outro EXEHDAnodo, a interface do Prontuário Eletrônico utilizado por ele migra para o segundo dispositivo. O disparo da migração é gerenciado pelo AVU e processado pelo serviço *Executor* do middleware EXEHDA.

5.3.2.3 Análise Sistematizada dos Comportamentos

Esta seção discute os comportamentos do EXEHDA-TS frente à existência de mobilidade lógica dos componentes de software. Como elemento de análise, serão avaliados os procedimentos executados pelo serviço no que diz respeito ao desacoplamento referencial, bem como em relação à manutenção da reatividade frente à mobilidade do OX entre os nodos.

Requisito Desacoplamento Referencial

A migração do componente de software de um EXEHDAnodo para outro implica a mudança do meio físico de execução e a troca do endereço de rede associado ao OX (número IP). Por sesta razão, para que o EXEHDA-TS seja mantido operacional, são necessários ajustes tanto na posição física do TSox quanto nas listas de gerenciamento do TSvirt. Para isso, no processo de migração, o EXEHDA-TS articula uma ação conjunta com o serviço *Executor*.

Nenhuma alteração é necessária em relação à localização física do TSox caso a configuração da instância nodal do serviço tenha por opção o armazenamento do TSox junto à instância base. Caso contrário, é necessário que o TSox seja migrado juntamente com o OX, e, por consequência, deve haver uma realocação nas listas de gerenciamento do TSvirt, tanto na instância nodal do TSox quanto na instância base. O novo nodo deverá conter, em sua lista de gerenciamento do TSvirt, a referência ao TSox que foi migrado, enquanto que, no nodo antigo, a referência deve ser removida. O procedimento a ser realizado na instância base consiste no ajuste das referências dos nodos (HostID) envolvidos na migração.

Tais procedimentos garantem a manutenção do TSvirt quando da realocação do TSox para outro nodo físico. O mecanismo discutido é realizado de forma transparente em relação aos processos em execução, o que proporciona o desacoplamento referencial promovido pelo EXEHDA-TS.

Requisito Reatividade

O modelo de reatividade analisado nesse cenário tem as mesmas características do caso anteriormente abordado. O suporte à reatividade provido pelo EXEHDA-TS será utilizado para notificar o médico de algum problema com um paciente, independentemente da localização do mesmo.

Utilizando um determinado computador, através da interface de Prontuário Eletrônico, o médico submete uma operação *subscribe*, pela qual é repassado um *template* com o modelo da tupla desejada. O *template* é associado ao ObjectID do OX que procedeu a operação e ambos são alocados tanto na lista de *templates* da instância nodal como na que é gerenciada pela instância base do EXEHDA-TS.

Posteriormente, o médico troca de nodo, sendo realizada a migração da interface do Prontuário Eletrônico para o mesmo. Esse processo requer um ajuste nas listas que gerenciam os eventos, para que o *template* submetido pela interface do médico seja apontado para a nova localização do processo associado. Na instância nodal do serviço, é necessário que todos os *templates* relacionados ao OX envolvido na migração sejam realocados no novo nodo; na instância base do EXEHDA-TS, deve ser ajustado o HostID em todos os *templates* associados ao OX que sofreu a migração.

Após os devidos reajustes nas listas de gerenciamento de *templates*, os procedi-

mentos envolvidos com a geração dos eventos são realizados da mesma maneira que o caso discutido no cenário anterior, ilustrado pela figura 5.5.

A análise comportamental do cenário apresentado nesta seção reforça o suporte do EXEHDA-TS à mobilidade lógica no que diz respeito aos aspectos envolvidos com a reatividade do serviço.

5.3.3 Cenário 3: Médico Troca de Escopo Celular

Neste cenário, o médico divide seu tempo entre o trabalho no hospital e os atendimentos em seu consultório. Assim, durante o período no hospital, ele solicita um exame de glicemia de um determinado paciente. Porém, por estar no final de seu plantão, configura o sistema para ser notificado assim que o resultado do exame seja inserido.

Após o término de seu plantão no hospital, o médico desloca-se para o seu consultório e, chegando lá, acessa o sistema através do seu computador pessoal. Em certo momento, o médico, que se encontra em uma célula diferente que a do paciente, recebe um alerta do sistema, informando o resultado do exame. Imediatamente, o médico entra em contato com os profissionais em plantão no hospital passando as recomendações necessárias.

5.3.3.1 Objetivo do Cenário

Este cenário tem por objetivo avaliar o comportamento do EXEHDA-TS diante da mobilidade física de um nodo para fora de seu escopo celular. Nessa situação, será avaliado o comportamento reativo do modelo especificamente em relação à consulta pervasiva.

5.3.3.2 Caracterização do Cenário

A situação apresentada caracteriza a mobilidade física do EXEHDAnodo móvel do médico entre células distintas. Esse fato requer uma realocação da tabela de templates e um reajuste na composição do TSvirt, para que o mesmo possa ser encontrado na ocorrência de um evento e nas operações de busca.

Outro caso que acarretaria em um modo de gerenciamento semelhante por parte do EXEHDA-TS seria a situação em que um determinado paciente precisasse fazer um exame e, para isso, fosse necessário deslocá-lo a outro hospital, mudando para a área de abrangência de outra EXEHDAcel.

5.3.3.3 Análise dos Comportamentos

O cenário analisado destaca um modelo de reatividade não abordado nos dois casos anteriores. Nesse, o resultado da operação é retornado através de um evento, que é processado uma única vez a cada subscrição realizada.

Nesse caso, ao solicitar um exame de glicemia, o médico não tem como precisar o momento exato que o mesmo será processado. Para que ele tenha a informação desejada o mais rápido possível, através da interface do Prontuário Eletrônico é realizada uma consulta pervasiva.

Para executar a tarefa desejada, uma operação *find* é encaminhada ao EXEHDA-TS, utilizando um *template* que poderia ter o seguinte formato: <“João da Silva”,“id”,10296,“glicemia”,?Integer>. A execução da consulta pervasiva requer a especificação de um *timeout*, o qual deve ser informado pelo médico ou atribuído automaticamente pelo Prontuário Eletrônico, considerando a expectativa de tempo que tal

exame leva para ser processado.

O template encaminhado pela operação *find* é associado ao ObjectID do OX que procedeu a operação, sendo inseridos na lista de eventos gerenciado pela instância nodal do EXEHDA-TS. Uma cópia das informações deve ser enviada à instância base do serviço juntamente com o HostID do nodo em que o OX encontra-se em execução.

No cenário avaliado, após processar a consulta pervasiva, o nodo associado ao médico desloca-se para outra célula, o que implica a mudança do identificador do nodo (HostID). Para que o mesmo permaneça sendo acessado a partir do TSvirt, são necessários ajustes nas suas referências junto à instância base do serviço EXEHDA-TS na célula de origem, tanto em relação aos TSox hospedados no nodo móvel quanto aos *templates* associados aos OX em execução em tal nodo.

Devido à incorporação de uma nova célula no escopo do TSvirt em questão, uma cópia da lista dos *templates* a ele relacionados deve ser encaminhada à instância base do EXEHDA-TS localizada célula de destino.

Após o nodo associado ao médico ter-se estabelecido em seu consultório, que se localiza em célula distinta àquela ao qual o hospital pertence, o resultado do exame de glicemia solicitado é processado e inserido no sistema através de um nodo localizado no escopo do hospital. Assim que a tupla com o resultado da glicemia do paciente é inserida no TSox, é feita uma comparação com os templates produzidos no nodo local. Em seguida, uma cópia dessa tupla é encaminhada à instância base do EXEHDA-TS, em que também é comparada com a lista de templates por ela gerenciados. Como na base constam todos os *templates* associados ao TSvirt, aquele relacionado à solicitação do médico é encontrado. Por meio da identificação do HostID referente ao nodo do médico, uma notificação é encaminhada à instância base do EXEHDA-TS da célula em que este encontra-se estabelecido. A notificação é processada pelo serviço que, por sua vez, produz um evento no Prontuário Eletrônico, apresentando ao médico o resultado do exame. Esse procedimento é exemplificado através da figura 5.7.

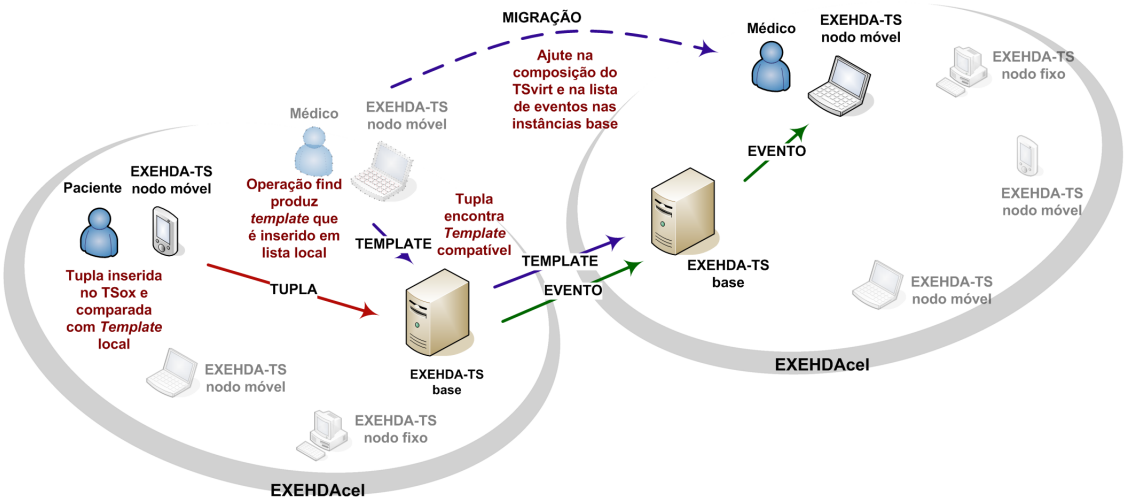

Figura 5.7: Fluxo do Processo de Consulta Pervasiva Após Mobilidade Física Intercelular

5.4 Discussão

Neste capítulo foram resumidas as tecnologias utilizadas na prototipação do EXEHDA-TS, sendo discutidos os aspectos favoráveis da linguagem Java em relação à implementação do modelo apresentado no capítulo 4. Uma das características do Java mais exploradas na prototipação foi a RMI, um recurso da linguagem para o suporte à distribuição de computações.

Para constituir o núcleo do EXEHDA-TS, foi utilizada uma implementação minimalista do modelo de Espaço de Tuplas, chamada LighTS. O LighTS foi desenvolvido em Java e possui código fonte disponível, aspectos que foram decisivos para a sua adoção na consolidação do protótipo.

Como suporte às funcionalidades do EXEHDA-TS, foi definido um conjunto mínimo de operações para a manipulação do Espaço de Tuplas, mapeadas em três classes básicas: *ETupleSpace*, *ETuple* e *EField*. As operações disponibilizadas compreendem tarefas como a escrita e leitura de tuplas, controle da reatividade e manipulação dos modos de sincronização do Espaço de Tuplas.

O EXEHDA-TS, apresentado nesta dissertação de mestrado, foi avaliado através da análise de seu comportamento no suporte a uma aplicação da área médica, em conformidade com as demandas típicas do projeto PERTMED. Assim, foram analisados três cenários de uso, que exploraram as principais funcionalidades propostas no modelo de coordenação apresentado nesta dissertação de mestrado, bem como o seu suporte em relação à mobilidade e desconexão. De acordo com os casos verificados, observou-se a viabilidade da proposta e o suporte que o modelo proporciona ao desenvolvimento de aplicações em cenários da Computação Pervasiva.

6 CONSIDERAÇÕES FINAIS

A Computação Pervasiva é um novo paradigma computacional que está sendo objeto de várias pesquisas científicas atualmente, e tem por premissa a disponibilização do ambiente do usuário independentemente de localização, tempo e dispositivo de acesso. Nessa visão, os sistemas computacionais são largamente distribuídos e caracterizam-se por manter uma operação integrada ao mundo real, provendo uma elevada transparência em relação à infraestrutura e tendo foco no usuário. Devido à complexidade desse ambiente, é necessário um middleware que dê suporte ao desenvolvimento e execução das aplicações, abstraindo características da infraestrutura.

Para promover a cooperação entre processos em cenários de elevada distribuição, uma estratégia adotada é a utilização de modelos baseados em coordenação, que têm por característica promover a separação entre as computações e as tarefas de gerenciamento das coordenações entre componentes de software. Um dos modelos de coordenação mais difundidos é o de Espaço de Tuplas, que consiste em uma memória associativa independente, compartilhada entre os nodos do sistema (YAMIN, 2004).

Os modelos de coordenação tradicionais apresentam limitações de utilização considerando as necessidades da Computação Pervasiva (MAMEI; ZAMBONELLI, 2006a). Para atender as demandas por coordenação das aplicações alvo do middleware EXEHDA, que se caracterizam pela distribuição, mobilidade e adaptação ao contexto, o EXEHDA-TS foi modelado tendo como premissa gerenciar um Espaço de Tuplas de forma distribuída e promovendo um comportamento proativo e escalável. No EXEHDA-TS, os Espaços de Tuplas são fisicamente associados aos OX, podendo ser compartilhados formando um Espaço de Tuplas virtual. Essa estratégia viabiliza a mobilidade dos componentes do Espaço de Tuplas juntamente com as migrações dos OX, para atender os requisitos das aplicações distribuídas.

Um dos desafios encontrados nas pesquisas envolvendo os modelos de Espaço de Tuplas distribuídos em ambientes móveis e de composição heterogenia dizem respeito aos aspectos de escalabilidade, que acabam limitando a utilização dos mesmos no atendimento a um número elevado de dispositivos. Para promover um melhor aproveitamento dos recursos disponíveis, são adotados no EXEHDA-TS mecanismos de limitação do escopo de atuação do Espaço de Tuplas, bem como técnicas de composição dinâmica, que se ajustam de acordo com o comportamento dos componentes de software. Assim, o EXEHDA-TS sistematiza a composição do Espaço de Tuplas através de um agrupamento por perfil, reunindo os dados distribuídos de acordo com o tipo de informação armazenada, sendo organizados de forma a atender necessidades específicas das aplicações.

Outra estratégia que é utilizada no EXEHDA-TS em relação à composição do

Espaço de Tuplas refere-se à adoção de mecanismos de sincronização que atuam no modo de compartilhamento dos dados associados aos OX. De acordo com as características dos dados injetados no mesmo, através da interface de programação, o desenvolvedor pode optar por compartilhar ou não os dados, reduzindo o escopo do Espaço de Tuplas distribuído e, por consequência, simplificando as tarefas de gerenciamento. Tal estratégia é dinâmica e pode ser ajustada no decorrera das computações.

Um aspecto central na concepção do EXEHDA-TS é a sua atuação reativa em relação às tuplas injetadas no Espaço de Tuplas. Essa funcionalidade consiste em um mecanismo que intercepta as informações oriundas das operações de escritas e notifica os componentes de software interessados em tais informações. O gerenciamento desse processo é feito de forma distribuída entre as duas instâncias do serviço (nodal e base), a fim de reduzir os esforços computacionais.

6.1 Principais Conclusões

As atividades desenvolvidas ao longo desta dissertação, especialmente os estudos realizados, a modelagem, prototipação e avaliação dos cenários de uso do EXEHDA-TS, permitiram obter as seguintes conclusões:

- nos modelos de coordenação voltados aos cenários da Computação Pervasiva, têm tido destaque o uso de abstrações de Espaço de Tuplas por proporcionar desacoplamento tanto temporal como referencial;

- para não comprometer a escalabilidade do sistema é necessário que o Espaço de Tuplas tenha um escopo de abrangência limitado e controlado;

- a incorporação de mecanismos reativos são fundamentais em cenários de larga distribuição para evitar que as aplicações tenham que monitorar o Espaço de Tuplas por conta própria, evitando um intenso uso da rede assim como uma elevação no custo das computações;

- o Espaço de Tuplas compartilhado deve ser distribuído e dinâmico, permitindo a entrada e saída de processos, de modo a acompanhar o dinamismo do ambiente.

Considerando os objetivos iniciais desta dissertação, pode-se afirmar que as pesquisas envolvidas na consolidação do EXEHDA-TS contribuíram para o estabelecimento de um modelo de coordenação proativo para o EXEHDA com base no conceito de Espaço de Tuplas distribuído. Assim, as características de desacoplamento referencial e temporal, oportunas ao desenvolvimento das aplicações alvo do middleware, foram promovidas.

Com foco na premissa da Computação Pervasiva de independência de localização, tempo e dispositivo, o EXEHDA-TS foi concebido de forma a dar suporte ao compartilhamento de informações de maneira transparente entre os componentes móveis e distribuídos que constituem as aplicações pervasivas, aprimorando os mecanismos de comunicação do EXEHDA. Além disso, através de estratégias de gerenciamento do escopo do Espaço de Tuplas, promove a escalabilidade, requisito importante nos middlewares para a Computação Pervasiva.

Tabela 6.1: Características do EXEHDA-TS Versus Trabalhos Relacionados

	LIME	TOTA	PeerSpace	EXEHDA-TS
Arquitetura	Descentralizada	Descentralizada	Descentralizada	Descentralizada
Mobilidade	Física (MANET) e lógica	Física (MANET) e lógica	Física (MANET)	Física (híbrida) e lógica
Consciência do Contexto	Contexto como dados	Contexto como dados	Não aplicado	Como suporte aos mecanismos de mais alto nível
Operação Desconectada	Sim	Não	Sim	Sim
Reatividade	Sim (dados)	Sim (dados e ambiente)	Sim (dados)	Sim (dados), através de subscrição de eventos e consulta pervasiva
Tipo de distribuição	TS associado aos processos, leitura distribuída, escrita local	TS associado aos nodos, escrita distribuída (regras) e leitura local	TS associado aos nodos, leitura local, escrita remota explícita	TS associado aos OX, com compartilhamento transparente, mas influenciado pelas atividades do usuário através dos modos de sincroniação
Escopo	TS Federado ou por dispositivo	Local ou um salto	Local e por grupo	Por aplicação e perfil

Na tabela 6.1, é feita uma caracterização do EXEHDA-TS bem como dos trabalhos relacionados (vide seção 2.3), utilizados como referência para a definição do modelo proposto no capítulo 4, e a solução empregada no EXEHDA-TS.

6.2 Contribuições da Pesquisa

Considerando o foco da pesquisa envolvida na consolidação do EXEHDA-TS, destacam-se as seguintes contribuições:

- sistematização dos modelos de coordenação voltados aos sistemas distribuídos, que estão apresentados no capítulo 2;
- identificação dos trabalhos relacionados à coordenação de processos móveis e distribuídos baseados em Espaços de Tuplas. Esses trabalhos estão organizados no capítulo 2;
- organização dos desafios de pesquisas envolvidos na especificação de um modelo de coordenação para a Computação Pervasiva. A sistematização desses desafios é

apresentada no capítulo 4;

- definição das funcionalidades que devem estar presentes no modelo proposto para o EXEHDA-TS, descritas no capítulo 4;

- especificação da arquitetura de software do EXEHDA-TS de forma a viabilizar as funcionalidades propostas. A arquitetura do EXEHDA-TS é apresentada no capítulo 4;

- introdução de mecanismos de sincronização do Espaço de Tuplas distribuído, adaptáveis sob demanda conforme o andamento das computações. Tais mecanismos são apresentados no capítulo 4;

- desenvolvimento de um protótipo e avaliação de estudos de caso focados nas demandas da área médica. Ambos são sumarizados no capítulo 5;

- contribuição aos esforços de pesquisa do projeto PERTMED, com estudo de caso voltado às demandas do mesmo;

- divulgação dos resultados dos trabalhos realizados durante o desenvolvimento desta dissertação de mestrado, com publicações apresentadas na seção 6.3;

- repasse do conhecimento e das tecnologias associadas ao EXEHDA-TS, no site http://paginas.ucpel.tche.br/∼rsouza/exehda-ts;

6.3 Publicações Realizadas

Ao longo do mestrado, o trabalho realizado foi divulgado através de algumas publicações, que são apresentadas a seguir:

- **9^a Escola Regional de Alto Desempenho - ERAD 2009.** Rodrigo Santos de Souza e Adenauer Yamin. EXEHDA-TS: um Modelo de Espaço de Tuplas para a Computação Pervasiva.

- **8^a Escola Regional de Alto Desempenho - ERAD 2008.** Rodrigo Santos de Souza e Adenauer Yamin. Redes de Sensores na Computação Pervasiva.

- **II Workshop on Pervasive and Ubiquitous Computing - WPUC 2008.** Rodrigo Santos de Souza, Adenauer Yamin e Iara Augustin. Em direção a um modelo de Coordenação para a Computação Pervasiva.

- **7^a Mostra de Pós-Graduação da Universidade Católica de Pelotas.** Rodrigo Santos de Souza e Adenauer Yamin. Um Modelo de Coordenação para a Computação Pervasiva.

- **Jornadas Chilenas de Computación - Jcc 2008.** Frederico Corrêa da Silva, Marilton Sanchotene de Aguiar, Rodrigo Santos de Souza e Adenauer Corrêa Yamin. Quadtrees and genetic algorithms applied with high performance in information segmentation and classification from medical images.

- **IX Workshop de Software Livre - WSL 2008.** Frederico da Silva, Adenauer Yamin, Andre Du Bois, André Moraes, Marilton Aguiar e Rodrigo Santos de Souza. Algoritmos Genéticos e Quadtrees Aplicados com Alto Desempenho na Segmentação e Classificação de Informações a partir de Imagens Médicas.

- **VII Simpósio de Informática da Região Sul - SIRC 2008.** Frederico da Silva, Adenauer Yamin, Andre Du Bois, André Moraes, Marilton Aguiar e Rodrigo Santos de Souza. Uma Aplicação Bag-of-Task baseada em Algoritmos Genéticos e Quadtrees para a Segmentação de Imagens.

- **8^a Escola Regional de Alto Desempenho - ERAD 2008.** Frederico da Silva, Adenauer Yamin, Andre Du Bois, André Moraes, Marilton Aguiar e Rodrigo Santos de Souza. Segmentando e Classificando Informações a partir de Imagens Médicas com Alto Desempenho utilizando Algoritmos Genéticos e Quadtrees.

6.4 Trabalhos Futuros

Consciente de que a pesquisa que consolidou o EXEHDA-TS apresentado nesta dissertação de mestrado não é um trabalho finalizado, mas sim a conclusão de uma etapa, a seguir são apresentados aspectos a serem explorados em trabalhos futuros:

- melhorar o mecanismo de tratamento das replicações relacionadas às listas de *templates*;

- definir e avaliar a possibilidade da utilização de um *timeout* para a manutenção temporária das referências dos TSox com sincronização sob demanda;

- explorar a criação dinâmica de perfis de dados no EXEHDA-TS a com a utilização de regras;

- definir heurísticas para o ajuste dos tempos associados ao gerenciamento de nodos desconectados, assim como ao número de tentativas a serem realizadas;

- estender a abrangência do Espaço de Tuplas para coordenação entre aplicações distintas;

- definir mecanismos para a identificação e eliminação de TSox sem OX associado e inativos.

REFERÊNCIAS

ANDRADE FIGUEIREDO, O. de. **Implementação de espaços de tuplas do tipo JavaSpaces**. 2003. Dissertação (Mestrado em Ciência da Computação) — Universidade de São Paulo - São Carlos.

AUGUSTIN, I. **Abstrações para uma Linguagem de Programação Visando Aplicações Móveis em um Ambiente de Pervasive Computing**. 2004. Tese (Doutorado em Ciência da Computação) — Universidade Federal do Rio Grande do Sul.

AURA. **Project Aura Distraction-free Ubiquitous Computing**. <http://www.cs.cmu.edu/ aura/>. Acesso em julho de 2008.

BALZAROTTI, D.; COSTA, P.; PICCO, G. P. The LighTS tuple space framework and its customization for context-aware applications. **Web Intelli. and Agent Sys.**, Amsterdam, The Netherlands, The Netherlands, v.5, n.2, p.215–231, 2007.

CABRI, G.; FERRARI, L.; LEONARDI, L.; MAMEI, M.; ZAMBONELLI, F. **Uncoupling Coordination**: Tuple-Based Models for Mobility. [S.l.]: Auerbach Publications, 2007.

CARRIERO; GELERNTER. How to Write Parallel Programs: A Guide to the Perplexed. **CSURV: Computing Surveys**, [S.l.], v.21, 1989.

COSTA, C. A. da. **Continuum**: A Context-aware Service-based Software Infrastructure for Ubiquitous Computing. 2008. Tese (Doutorado em Ciência da Computação) — Universidade Federal do Rio Grande do Sul.

COSTA, C. da; YAMIN, A.; GEYER, C. Toward a General Software Infrastructure for Ubiquitous Computing. **Pervasive Computing, IEEE**, [S.l.], v.7, n.1, p.64–73, Jan.-March 2008.

DEITEL, H. M.; DEITEL, P. J.; SANTRY, S. **Advanced Java 2 platform**: how to program. [S.l.]: Prentice-Hall, 2002.

EDWARDS, W. K.; GRINTER, R. E. At Home with Ubiquitous Computing: Seven Challenges. In: UBICOMP '01: PROCEEDINGS OF THE 3RD INTERNATIONAL CONFERENCE ON UBIQUITOUS COMPUTING, 2001, London, UK. **Anais...** Springer-Verlag, 2001. p.256–272.

ENDEAVOUR. **Endeavour Project**. <http://endeavour.cs.berkeley.edu/>. Acesso em julho de 2008.

GAIA. **Gaia Project**. <http://choices.cs.uiuc.edu/ActiveSpaces/>. Acesso em agosto de 2008.

GARLAN D.; SIEWIOREK, D. S. A. S. P. Project Aura: toward distraction-free pervasive computing. **Pervasive Computing, IEEE**, [S.l.], v.1, n.2, p.22–31, 2002.

GRIMM, R.; DAVIS, J.; LEMAR, E.; MACBETH, A.; SWANSON, S.; ANDERSON, T. E.; BERSHAD, B. N.; BORRIELLO, G.; GRIBBLE, S. D.; WETHERALL, D. System support for pervasive applications. **ACM Trans. Comput. Syst**, [S.l.], v.22, n.4, p.421–486, 2004.

HENRICKSEN, K.; INDULSKA, J. **Developing contextaware pervasive computing applications**: Models and approach.

IBM. PERVASIVE Computing. IBM System Journal. **IBM System Journal**, [S.l.], v.38, n.4, 1999. <http://www.research.ibm.com/journal/sj38-4.html> Acessado em novembro de 2008.

IBM. **IBM's TSpaces middleware**. <http://www.almaden.ibm.com/cs/TSpaces/>. Acesso em maio de 2008.

ISAM. **Infra-estrutura de Suporte às Aplicações Móveis**. <http://www.inf.ufrgs.br/ isam/>. Acesso em fevereiro de 2008.

JADE. **Java Agent DEvelopment Framework**. <http://jade.tilab.com/>. Acesso em julho de 2008.

JAVASPACES. **JavaSpaces Service Specification**. <http://java.sun.com/products/jini/2.0.2/doc/specs/html/jsTOC.html>. Acesso em setembro de 2008.

JINI.ORG. **JINI**. <http://www.jini.org/>. Acesso em agosto de 2008.

LIME. **Linda in a Mobile Environment**. <http://lime.sourceforge.net/>. Acesso em agosto de 2008.

LOPES, J. L.; PILLA, M. L.; YAMIN, A. C. EXEHDA: a Middleware for Complex, Heterogeneous and Distributed Applications. **Conferência Nacional em Inteligência Computacional Aplicada à Indústria de Petróleo**, [S.l.], junho 2007.

ISHIDA, T.; JENNINGS, N. R.; SYCARA, K. (Ed.). **Field-Based Coordination for Pervasive Multiagent Systems**. [S.l.]: Springer, 2006.

ISHIDA, T.; JENNINGS, N. R.; SYCARA, K. (Ed.). **Field-Based Coordination for Pervasive Multiagent Systems**. [S.l.]: Springer, 2006.

MAMEI, M.; ZAMBONELLI, F.; LEONARDI, L. Tuples On The Air: A Middleware for Context-Aware Computing in Dynamic Networks. In: ICDCS WORKSHOPS, 2003. **Anais...** IEEE Computer Society, 2003. p.342–347.

MURPHY, A. L.; PICCO, G. P.; ROMAN, G.-C. LIME: A coordination model and middleware supporting mobility of hosts and agents. **ACM Transactions on Software Engineering and Methodology**, [S.l.], v.15, n.3, p.279–328, jul 2006.

OLIVEIRA VALENTE, M. T. de. **Mobilidade e Coordenação de Aplicações em Redes sem Fio**. 2002. Tese (Doutorado em Ciência da Computação) — Universidade Federal de Minas Gerais.

OXYGEN. **MIT Project Oxygen**. http://www.oxygen.lcs.mit.edu/.

PEREIRA, F. M. Q.; OLIVEIRA VALENTE, M. T. de; BIGONHA, R. S.; BIGONHA, M. A. S. Uma Linguagem para Coordenação de Aplicações em Redes Móveis Ad Hoc. In: **XXI Simpósio Brasileiro de Linguagens de Coordenação**. [S.l.: s.n.], 2002. p.152–165.

PERTMED. **Projeto PertMed**. <http://pertmed.wkit.com.br/>. Acesso em novembro de 2008.

PICCO, G. P. **LighTS**. <http://lights.sourceforge.net/>. Acesso em março de 2009.

PICCO, G. P.; BALZAROTTI, D.; COSTA, P. LighTS: a lightweight, customizable tuple space supporting context-aware applications. In: SAC '05: PROCEEDINGS OF THE 2005 ACM SYMPOSIUM ON APPLIED COMPUTING, 2005, New York, NY, USA. **Anais...** ACM, 2005. p.413–419.

NETHERLANDS, S. (Ed.). **Some Research Challenges in Pervasive Computing**. [S.l.]: Privacy, Security and Trust within the Context of Pervasive Computing, 2005. p.1–16.

ROMáN, M.; HESS, C.; CERQUEIRA, R.; RANGANATHAN, A.; CAMPBELL, R. H.; NAHRSTEDT, K. A middleware infrastructure for active spaces. **Pervasive Computing, IEEE**, [S.l.], v.1, n.4, p.74–83, 2002.

SAHA, D.; MUKHERJEE, A. Pervasive computing: a paradigm for the 21st century. **IEEE Computer**, [S.l.], v.36, n.3, p.25–31, 2003.

TANENBAUM, A. S.; STEEN, M. V. **Distributed Systems**: Principles and Paradigms. 2.ed. [S.l.]: Prentice Hall, 2007.

WEISER, M. The Computer for the 21st Century. **Scientific American**, [S.l.], v.265, n.3, p.94–104, setembro 1991.

YAMIN, A. C. **Arquitetura para um Ambiente de Grade Computacional Direcionado às Aplicações Distribuídas, Móveis e Conscientes do Contexto da Computação Pervasiva**. 2004. Tese (Doutorado em Ciência da Computação) — Universidade Federal do Rio Grande do Sul.