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

Middlewares e Redes de Sensores: aspectos conceituais e arquiteturais

por
Rodrigo Santos de Souza

Trabalho Individual I
TI-2007/2-08

Orientador: Prof. Dr. Adenauer Corrêa Yamim

Pelotas, dezembro de 2007

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

AGRADECIMENTOS

Agradeço a COPPETEC, pelo amparo financeiro.

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

À todos os colegas do mestrado 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 a realização deste trabalho: meus pais, Pedro e Ceni, meus irmãos e amigos, e principalmente a minha esposa, Denise.

Por fim, agradeço a Deus pela saúde - física, mental e emocional, e pelas oportunidades recebidas que permitiram esta trajetória até aqui.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	Aplication Program Interface
EXEHDA	Execution Environment for Highly Distributed Applications
ISAM	Infraestrutura de Suporte às Aplicações Móveis Distribuídas
ISAMpe	ISAM pervasive environment
QoS	Quality of Service
RSSF	Redes de Sensores sem Fio
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
UDDI	Universal Description Discovery and Integration
WSDL	Web Services Description Language
XML	Extensible Markup Language

RESUMO

A computação pervasiva pressupõe uma elevada integração entre os sistemas computacionais e o mundo real, tendo como foco o usuário e suas atividades. Em paralelo a isto, as redes de sensores, que têm sido objeto de vários estudos recentes, possibilitam que os sistemas computacionais sejam "alimentados" com informações oriundas do ambiente real. Estas informações sensoradas têm utilização em aplicações nas mais variadas áreas (ambiental, industrial, médica, entre outras), e através da computação pervasiva, podem estar disponíveis independente de localização, tempo ou dispositivo de acesso. O objetivo central deste trabalho é avaliar os aspectos envolvidos nas aplicações para redes de sensores na computação pervasiva e sua gerência. A premissa buscada é utilizar middlewares para prover infraestrura para facilitar o desenvolvimento, manutenção e execução das aplicações.

Palavras-chave: Redes de sensores, middleware, computação pervasiva, SOA.

TITLE: “SENSORS NETWORK MIDDLEWARE”

ABSTRACT

The pervasive computing presupposes a high integration of the computer systems and the real world, with the focus on user and your activities. In parallel to this, the sensor networks, which have been the subject of several recent studies, which allow the computer systems are “fed’ with information from the real environment. This sensored information have use in applications in various areas (environmental, industrial, medical, among others), and through pervasive computing, may be available regardless of location, time or access device. The main objective of this work is to evaluate the issues involved in applications to sensor networks in the pervasive computing and its management. The pursued premise is the use of middleware to provide infraestrura to facilitate the development, implementation and maintenance of applications.

Keywords: sensor network, middleware.

1 INTRODUÇÃO

Nos dias atuais é notório o crescimento do uso da computação no dia-a-dia das pessoas, seja na grande utilização dos computadores pessoais (*desktops* e *laptops*), PDAs, *smartphone*, aparelhos celulares, *players* de áudio e vídeo, ou mesmo em alguns eletrodomésticos. Com toda esta esta tecnologia disponível e com a evolução da Internet e das redes de comunicação, é natural a integração entre estes dispositivos, constituindo um sistema distribuído. Porém, para uma utilização em massa destas tecnologias por usuários leigos é fundamental uma certa transparência em relação às particularidades da infraestrutura, dos dispositivos utilizados. Neste contexto, a computação pervasiva surge com grande força, pois pressupõe uma elevada integração dos sistemas computacionais com o mundo real, tendo como foco o usuário e suas atividades. Mark Weiser (WEISER, 1991) resume que a computação pervasiva ou ubíqua deve permitir que o usuário tenha acesso ao seu ambiente computacional, de todo lugar e a todo momento, por meio de qualquer dispositivo. Grande parte dos trabalhos nesta área, desde então, vêm perseguindo estratégias para prover este novo paradigma de computação.

Para tornar realidade a computação pervasiva são necessários mecanismos que permitam que os sistemas computacionais tenham conhecimento da realidade na qual a computação está imersa. Neste contexto, uma tecnologia está em ascensão atualmente e promete agregar uma gama de possibilidades às aplicações da computação pervasiva: as redes de sensores. Em contrapartida, as contribuições da computação pervasiva em prol das redes de sensores são evidentes: prover o acesso a dados sensorados em qualquer lugar, a qualquer tempo e através de qualquer dispositivo.

As redes de sensores estão bastante presentes nos dias atuais, especialmente nas indústrias que utilizam processos automatizados, comumente encontrados nas indústrias automobilísticas, alimentícias, metal-mecânicas, marítimas, entre outras. O sensoriamento de variáveis ambientes não é novidade, porém o que está no centro de várias pesquisas nas áreas da Ciência da Computação e Engenharia são as redes de sensores sem fio. Estas utilizam sensores chamados "inteligentes", que são dotados de certo poder computacional, possuem autonomia de energia e comunicação sem fio. Eles são capazes de se auto-organizar e construindo a topologia de rede e estratégias de comunicação de forma a transmitirem as informações coletadas do meio para aplicações interessadas, ponderando entre precisão, latência e eficiência em energia.

Os middlewares são tidos como uma forma adequada para gerenciar, tanto de aplicações na computação pervasiva (SAHA; MUKHERJEE, 2003) (YAMIN, 2004) quanto redes de sensores (LOUREIRO et al., 2003) (DELICATO, 2005). Desta maneira, um estudo envolvendo estes dois mundos e as suas formas de gerenciamento pode levar a

nova visão: o gerenciamento de redes de sensores na computação pervasiva.

Em paralelo a isso, as Arquiteturas Orientadas a Serviço apresentam-se como uma forma adequada para integração de aplicações distribuídas, devido a interoperabilidade que o conceito propõe, por isso este texto traz um estudo referente a este assunto, com o intuito de considerar o uso deste paradigma no modelo de middleware para gerenciar redes de sensores na computação pervasiva.

1.1 Tema

Este trabalho tem como principal enfoque a avaliação dos mecanismos envolvidos na gerência de redes de sensores na computação pervasiva. As redes de sensores capturam informações do ambiente real e as transformam em dados computáveis, de forma que possam ser tratados por alguma aplicação. Sob a ótica da computação pervasiva, as aplicações interessadas nestes dados podem estar em qualquer lugar e em qualquer dispositivo e mesmo assim devem ter acesso a estas informações com adequada precisão e latência.

Desta maneira, este trabalho contempla um estudo avaliando as demandas envolvidas nestas situações, considerando a utilização de middlewares como uma forma adequada de prover uma infraestrutura de software que possibilite satisfazer tanto as necessidades das aplicações na computação pervasiva quanto das rede de sensores.

1.2 Motivação

A computação pervasiva tem sido alvo de várias pesquisas atualmente e é apontada como um dos grandes desafios desta década (CARVALHO et al., 2006). Os dias atuais são caracterizados por uma grande utilização de sistemas computacionais de diversas capacidades e recursos, dotados ou não de mobilidade. Neste contexto a computação pervasiva ganha grande força, pois propõe foco no usuário e em suas atividades, provendo o seu ambiente de trabalho em qualquer lugar, independente de tempo e equipamento. Contempla aspectos que permitam mobilidade física (usuário com ou sem equipamento) e de software.

Uma das premissas da computação pervasiva é uma forte integração com o mundo real. É neste sentido que vem se destacando as redes de sensores sem fio. Estes sistemas consistem de um número muito grande de sensores autônomos, capazes de se auto-organizarem em uma rede e enviarem informações do ambiente para as aplicações interessadas. O relatório do seminário sobre os Grandes Desafios da Pesquisa em Computação no Brasil - 2006-2016 (CARVALHO et al., 2006) aponta para o uso maciço de sensores nas mais diversas áreas dentro de uma década. A previsão é que neste período estes elementos vão invadir casas, escritórios, fábricas, carros, ruas, e fazendas, impulsionando a computação pervasiva.

Especialmente na indústria da construção naval, a utilização de redes de sensores e computação pervasiva para acompanhar e controlar os processos, pode prover uma maior eficiência em seus complexos processos produtivos.

1.3 Objetivos

O Objetivo central deste trabalho é avaliar os mecanismos que devem estar presentes em uma infraestrutura para gerência de redes de sensores, na computação pervasiva.

Como principais objetivos perseguidos destacaríamos:

- identificar as atuais demandas da computação pervasiva e os principais projetos na área;
- estudar as tendências de middlewares para sistemas distribuídos e as principais características perseguidas por estes sistemas;
- caracterizar as demandas no gerenciamento das redes de sensores e os middlewares utilizados;
- organizar os conceitos envolvidos no paradigma de Arquitetura Orientada a Serviços, avaliando sua utilização na modelagem e implementação de middlewares para sistemas distribuídos.

1.4 Estrutura do texto

O texto está organizado em cinco capítulo. O capítulo 2 apresenta as premissas da computação pervasiva e uma visão atual da área, incluindo alguns projetos envolvendo este tema. No capítulo 3 são apresentadas as principais características dos middlewares para sistemas distribuídos e alguns tipos de middlewares identificados. O capítulo 4 aborda os conceitos envolvidos nas redes de sensores e as características dos middlewares utilizados em sua gerência. E por fim, o capítulo 5 apresenta o paradigma SOA (*Service Oriented Architecture*) como um modelo de referência para modelagem de middlewares para sistemas distribuídos.

2 COMPUTAÇÃO PERVASIVA: PRINCIPAIS CONCEITOS E PROJETOS

Nos dias atuais a computação mostra-se cada vez mais integrada na vida das pessoas, seja na utilização dos computadores pessoais ou mesmo através dos inúmeros controles automatizados existentes nos mais diversos produtos eletro-eletrônicos utilizados no dia-a-dia, como televisores, aparelhos de som, sistemas de alarmes, aparelhos celulares, etc., e tudo isto de uma forma muito natural. Com toda esta quantidade de aparelhos com maior ou menor poder computacional e com a evolução da Internet e das redes de comunicação, começa a se tornar mais comum a integração destes diversos elementos antes isolados. A integração de toda esta diversidade de aparelhos não é tarefa simples, devido a grande heterogeneidade de hardware, software, redes de comunicação e protocolos. A transparência nesta integração é fator fundamental para que os usuários possam desfrutar de todos os benefícios que isto poderá trazer. O usuário deve ter disponível o acesso e o controle de seu ambiente (pessoal ou profissional) onde quer que ele esteja, independente de sua localização ou meio de acesso. Inúmeros são os requisitos que devem ser satisfeitos para que isso ocorra tornando bastante complexa a tarefa de concretizar esta visão de computação.

Nesta perspectiva a quantidade de elementos comunicando-se é muito elevada, muitos deles com conexões sem fio o que torna este tipo de rede de comunicação diferente das existentes na maior parte dos ambientes atuais. Novos problemas surgem neste contexto, impulsionando o surgimento de uma nova área que transcende as características da maioria dos sistemas distribuídos em uso hoje. Nesta área, denominada computação pervasiva ou ubíqua, os sistemas são largamente distribuídos e pressupõe uma forte integração com o mundo real, mantendo alta transparência as questões de infraestrutura e o foco no usuário e em suas atividades. Para o desenvolvimento de aplicativos nesse cenário, é necessária uma infra-estrutura de software apropriada para prover um elevado grau de transparência as características intrínsecas do sistema de modo que a participação humana neste processo seja mínimo.

2.1 Premissas da Computação Pervasiva

No artigo sobre computação para o século 21 de Mark Weiser (WEISER, 1991), ele resume o que é esperado da computação pervasiva ou ubíqua: acesso do usuário ao ambiente computacional, de todo lugar e a todo momento, por meio de qualquer dispositivo. A dificuldade está no desenvolvimento de aplicativos que irão continuamente se

adaptar ao ambiente e continuar funcionando, a medida que as pessoas se movem ou trocam de dispositivos. Outro problema consiste em deslocar o ambiente de trabalho do usuário sem que o hardware se mova com ele.

Aplicações pervasivas precisam de um middleware para servirem de interface entre os muitos dispositivos diferentes e as aplicações do usuário final (SAHA; MUKHERJEE, 2003). O objetivo deste é abstrair a complexidade do ambiente, isolando aplicações do gerenciamento explícito de protocolos, acesso distribuído a memória, replicação de dados, falhas de comunicação, etc. Um middleware também pode resolver problemas de heterogeneidade relacionados às arquiteturas, sistemas operacionais, tecnologias de redes e até mesmo de linguagens de programação, promovendo a interoperação entre esses componentes.

Um middleware deve permitir que o usuário acesse o ambiente computacional dele (dados e aplicativos) de qualquer lugar e a qualquer momento. Uma solução possível é aplicar a semântica siga-me (YAMIN, 2004). A idéia desse conceito é que aplicativos e dados vão juntos com os usuários, fornecendo um ambiente virtual adaptado ao contexto corrente. Essa adaptação é fundamental para a visão de 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).

2.2 Computação Pervasiva: contexto atual

"As tecnologias mais profundas são aquelas que desaparecem, elas se integram na vida cotidiana até se tornarem indistinguíveis da mesma" (WEISER, 1991). A frase de Mark Weiser sintetiza um pouco do que é esperado hoje em dia com a computação pervasiva. O termo designa o acesso ao ambiente computacional do usuário de todo lugar, todo o tempo com qualquer dispositivo. A dificuldade da computação pervasiva está no desenvolvimento de aplicações que se adaptem continuamente ao ambiente e permaneçam funcionando mesmo quando o indivíduo se movimentar ou trocar de dispositivo (GRIMM et al., 2004).

O objetivo da mobilidade de prover computação a qualquer momento e em qualquer lugar é considerado uma abordagem reativa ao acesso da informação, entretanto prepara para a pró-atividade da computação pervasiva: todo tempo, em todo lugar (SAHA; MUKHERJEE, 2003). Surge o termo *everywhere* (em todo lugar) para denominar a nova classe de software necessária para a computação pervasiva. Todavia, ainda existem muitas limitações para o desenvolvimento de tais softwares, pois poucas linguagens e ferramentas estão disponíveis para a programação.

Adaptação dinâmica é um tema 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 aspectos determinados do contexto de uso, como localização, tempo e atividades do usuário, permitindo o desenvolvimento de aplicações sensíveis ao contexto (*context-aware*) (HENRICKSEN; INDULSKA, 2005).

As aplicações sensíveis ao contexto devem prever a mobilidade de equipamentos e usuários, denominada mobilidade física, e também dos componentes da aplicação e serviços, chamada de mobilidade lógica. Para isso, as aplicações devem permitir que o usuário acesse seu ambiente computacional independente da localização e do tempo (YAMIN, 2004).

2.3 Projetos para Computação Pervasiva

As aplicações pervasivas necessitam de um middleware como interface entre os diversos dispositivos (desktops, notebooks, PDAs, equipamentos de rede, etc.) e as aplicações do usuário (SAHA e MUKHERJEE, 2003). O objetivo é esconder a complexidade do ambiente isolando as aplicações da gerência explícita dos aspectos da infreaestrutura. Com o middleware é possível resolver o problema da heterogeneidade de arquiteturas, de sistemas operacionais, de tecnologias de rede e até mesmo de linguagens de programação.

Existem diversas iniciativas de middlewares para a computação pervasiva. Dentre estes, a seguir são apresentados alguns projetos que foram identificados durante este estudo.

2.3.1 Projeto Aura

O Projeto Aura (GARLAN D.; SIEWIOREK, Apr-Jun 2002) (AURA, 2007) está sendo concebido na Universidade de Carnegie Mellon desde 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 ambiente. Os sistemas são: 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.

2.3.2 projeto Gaia

O Projeto Gaia (ROMAN M.; HESS, Oct-Dec 2002) (GAIA, 2007) consiste de um middleware distribuído que coordena entidades de software e redes de dispositivos heterogêneos. A idéia do projeto é criar um metasistema 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 está sendo criado 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 disponíveis nestes.

2.3.3 Projeto One.World

O projeto One.World (GRIMM, July-Sept. 2004) (PROJECT, 2007) projeto da Universidade de Nova York e foca a construção e distribuição de aplicações adaptativas. Foi concebido por Robert Grimm em sua tese de doutorado na Universidade de Washington. Dentre os principais serviços é possível destacar Discovery que auxilia na localização e conexão de serviços em outros dispositivos e Migration que permite a mobilidade de aplicativos para implementar a semântica siga-me. Aplicativos armazenam dados, comunicam-se através de tuplas e são organizados na forma de componentes. Tuplas e componentes são organizados em ambientes que provêem estrutura e controle.

2.3.4 Projeto Endeavour

Projeto da Universidade da Califórnia em Berkeley desde 1999, Endeavour (ENDEAVOUR, 2007) 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 infra-estrutura, 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 infra-estrutura de hardware. O sistema pode ser composto de componentes pré-existentes de software e hardware para satisfazer requisições de serviços (SAHA; MUKHERJEE, 2003).

2.3.5 Projeto Oxygen

O projeto Oxygen (OXYGEN, 2007) 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 auto-configurável. 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 infra-estrutura de software do Oxygen é constituída dos seguintes componentes: *Pebbles* que são partes de software independentes de plataforma; *Goals* responsível 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* 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* 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).

2.3.6 Projetos Comerciais

Algumas iniciativas da indústria na computação pervasiva também são relatadas em (SAHA; MUKHERJEE, 2003):

- Cooltown: projeto da Hewlett-Packard (HP Computadores) que objetiva estender as tecnologias da web, de redes sem fio e de dispositivos portáteis para a criação de um ambiente virtual pervasivo;

- Sentient Computing: projeto em desenvolvimento nos laboratórios AT&T em parceria com a Universidade de Cambridge que propõe um modelo de mundo compartilhado entre usuários e aplicações;

- EasyLiving: projeto da Microsoft Research Vision Group que visa conceber arquitetura e tecnologias relacionadas com ambientes inteligentes;

- WebSphere Everyplace: em desenvolvimento pela IBM o projeto propõe padrões abertos para suportar aplicações e uma nova geração de dispositivos na computação

pervasiva. O projeto é mantido em conjunto com a Palm, Symbol Technologies e Handspring.

2.4 O projeto ISAM

Dentre os ambientes para computação pervasiva cabe destacar o projeto ISAM (Infra-estrutura de Suporte às Aplicações Móveis) (ISAM, 2007). A idéia deste projeto é construir uma infra-estrutura de computação pervasiva, integrando uma linguagem de programação e um middleware para suportar sua execução. Diferentemente de outros projetos, ISAM foca no desenvolvimento de aplicações ao invés do ambiente e de serviços. Por causa disso, o projeto contém um modelo, uma linguagem, e um suporte para execução que permite construir e executar aplicativos.

ISAMadapt (ISAMADAPT, 2007) é uma linguagem de programação que facilita o desenvolvimento de aplicações pervasivas. A linguagem fornece meios de expressar adaptação dinâmica e consciência de contexto em tempo de projeto. ISAMadapt é baseada em um modelo multiparadigma denominado Holoparadigma (HOLO, 2007). Em Holoparadigma, um blackboard lógico, denominado história, implementa os mecanismos de coordenação e uma nova entidade de programação, chamada de ente, organiza vários níveis de encapsulamento de entes e histórias (multi-domínios). Os entes são a principal abstração do Holoparadigma, representando os componentes lógicos e físicos do sistema que está sendo modelado.

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. A Figura 2.1 apresenta a estrutura da arquitetura ISAM.

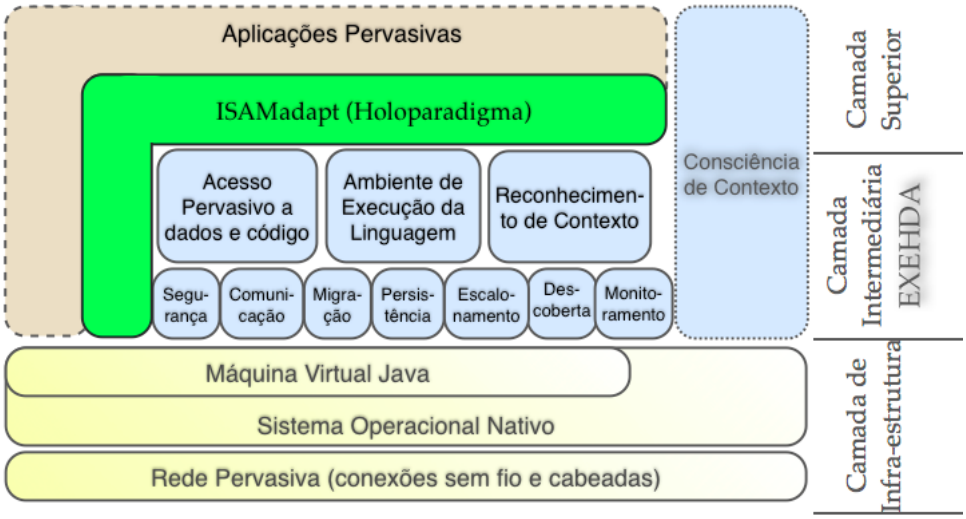

Figura 2.1: Arquitetura do ISAM (YAMIN, 2004)

A estrutura do projeto ISAM é divida em três camadas: camada inferior, composta pela infra-estrutura para execução das camadas acima (hardware, sistema operacional e máquina virtual Java); camada intermediária, constituída do middleware EXEHDA (YAMIN, 2004) responsável por toda a gerência da execução das aplicações; e, a camada

superior, formada pelo ISAMadapt, que possibilita o suporte a comportamento adaptativo em tempo de desenvolvimento, e pelas aplicações pervasivas.

A representação da consciência de contexto na Figura 2.1 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 e o segundo nível caracteriza os serviços básicos, quais sejam: 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 MIDDLEWARES PARA SISTEMAS DISTRIBUÍDOS

Sistemas distribuídos são constituídos por inúmeros dispositivos, com as mais variadas características físicas ou lógicas. A heterogeneidade é uma característica predominante na maioria dos sistemas distribuídos modernos, onde um mesmo sistema pode ter ao mesmo tempo *desktops*, *laptops*, PDA's, aparelhos celulares, cada um deles com recursos de *hardware*, de *software* e de comunicação bem distintos. Integrar estes sistemas de forma adequada, tornando a interação entre dispositivos menos penosa é um desafio, que segundo grande parte das referências, pode ser superado de forma eficiente através da utilização de middlewares.

O Middleware é uma camada de software que fica entre o sistema operacional e as aplicações e tem como objetivo prover uma abstração às principais características das aplicações distribuídas, facilitando e agilizando o desenvolvimento e reduzindo as ocorrências de erros. Esta abstração vai no sentido a tornar transparente para o desenvolvedor elementos como tecnologia de rede, tipo de arquitetura de hardware, sistema operacional e linguagem de programação utilizada. Sua utilização em aplicações distribuídas promove uma abstração da heterogeneidade física e lógica intrínseca deste tipo de sistema, principalmente em sistemas distribuídos *ad-hoc* e nômades. O gerenciamento destes sistemas é bastante complexo devido a intermitência da rede, a alta taxa de conexões e desconexões e a grande variedade de dispositivos de características diferentes.

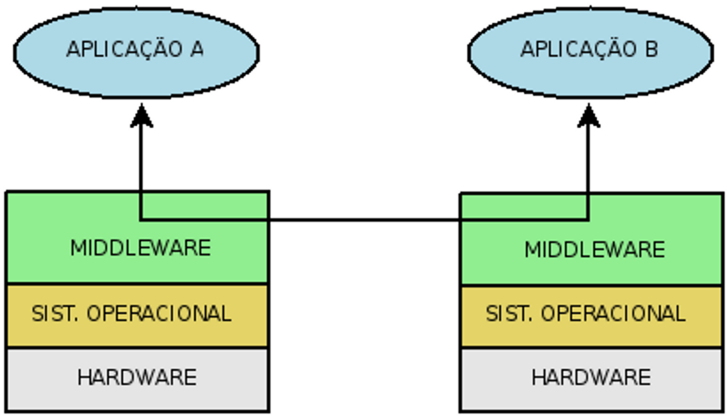

Figura 3.1: Modelo de middleware para sistemas distribuídos

Sem o suporte de um middleware no desenvolvimento de aplicações distribuídas, a programação tem que se dar no nível da camada de rede, o que é bastante trabalhoso e propenso a erros. Segundo (DELICATO, 2005), nestas condições, os desenvolvedores têm que lidar com vários requisitos não funcionais dos sistemas distribuídos, como replicação e localização de dados, tratamento de falhas da rede, gerência de transações e concorrência, entre outros, tornando o desenvolvimento e a manutenção de aplicações extremamente lentos.

Para dar suporte às aplicações, os middlewares disponibilizam serviços que devem ser o mais genérico possíveis, permitindo o seu uso por diferentes aplicações e não apenas por uma em particular. Seus serviços devem ser independentes de plataforma, o que faz da portabilidade uma característica importante dos middlewares, permitindo a transferência entre diferentes plataformas com o mínimo de esforço (MACIEL; ASSIS, 2004). O acesso a estes serviços disponibilizados pelos middlewares se dá através de API's (*Application Programmer Interface*), que constituem uma forma transparente para desenvolvedor utilizar os recursos do middleware.

Para satisfazer requisitos de extensibilidade, que serão discutidos na próxima seção, os middlewares são construídos de forma a incorporarem serviços de forma modular. Desta forma pode-se adicionar e remover serviços de acordo com a necessidade das aplicações - sob demanda. Os serviços disponibilizados variam bastante de um middleware para outro e de acordo com os requisitos das aplicações que executam sobre ele. Segundo (MACIEL; ASSIS, 2004) os serviços mais comuns disponíveis pelos middlewares são:

- gerenciamento de apresentação;
- computação;
- gerenciamento de informação;
- comunicação;
- controle;
- gerenciamento do sistema;
- sistema de entrega;
- comunicação entre processos;
- interface com o usuário.

3.1 Principais características perseguidas

Alguns requisitos são comuns a uma grande parte dos middlewares desenvolvidos para sistemas distribuídos. Esta seção tem por objetivo apresentar estes requisitos caracterizando cada um deles segundo a ótica de (PIETZUCH, 2004).

3.1.1 Escalabilidade e Expressividade

Escalabilidade é a capacidade do middleware poder vir a suportar um grande número de nodos sem que haja comprometimento da performance do sistema. Já a expressividade refere-se ao nível de granulosidade das informações que são transmitidas entre os nodos do sistema. Quanto mais fina a granulosidade, maior é a expressividade. A relação entre estes dois requisitos é bastante forte. Uma alta expressividade, acarreta em um elevado número de pacotes, de tamanhos pequenos, sendo transmitido entre as aplicações distribuídas. A consequência disso é um custo maior na comunicação e na gerência, e portanto uma menor Escalabilidade.

3.1.2 Segurança

Em middlewares para sistemas distribuídos a segurança nas comunicações é um requisito que tem uma importância elevada. As informações transmitidas devem ser entregues nos nodos de destino com a garantia de que estão completas e sem distorções. Em sistemas distribuídos as falhas de comunicação não são incomuns e devem ser eficientemente tratadas para não comprometer a transmissão correta dos dados. Diversas estratégias são usadas neste sentido de acordo com os requisitos aos quais cada middleware se propõe.

3.1.3 Administrabilidade

Um sistema distribuído por si só é bastante complexo, portanto é desejável que ele possua uma certa facilidade no que tange a administrabilidade do sistema como um todo. Quando componentes novos são adicionado ao sistema, seja para agregar novas funcionalidades, ou para melhorar funcionalidades existentes, o middleware deve ter capacidade de auto-adaptação de forma que a necessidade de intervenção humana seja a menor possível. A capacidade de auto-adaptação do middleware vem no sentido de reduzir a necessidade de tomadas de decisão por parte do administrador do sistema, facilitando a gerência do middleware.

3.1.4 Usabilidade

Um middleware deve ser fácil de utilizar, isto é, o desenvolvedor não deve encontrar dificuldades em utilizar as API's disponibilizadas pelo middleware para construir as aplicações distribuídas, as quais devem ser integradas à linguagem de programação de forma simples. O sucesso de um middleware depende desta facilidade, caso contrário dificilmente terá uma utilização muito difundida.

3.1.5 Extensibilidade

A extensibilidade em um middleware diz respeito a possibilidade de se adicionar novas funcionalidades conforme a necessidade das aplicações. Para isto é preciso que ele seja construído de forma modular e assim novos módulos possam ser adicionados sem ter que alterar o sistema todo. Esta propriedade garante que o middleware possa ter um longo tempo de vida, pois pode ser atualizado e melhorado de acordo com as demandas das aplicações e assim se mantenha atualizado com o passar do tempo.

3.1.6 Interoperabilidade

A heterogeneidade característica dos sistemas distribuídos faz com que a interoperabilidade seja um requisito importante dos middlewares. A troca de informações entre diferentes middlewares em algum momento se tornará necessária, portanto deve existir uma forma padronizada de troca de mensagens entre aplicações de diferentes origens. Uma alternativa comumente utilizada é o uso do padrão XML como alternativa para transladar entre diferentes formatos utilizados por plataformas distintas.

Neste sentido, a independência de plataforma e linguagem de programação também vai em direção à interoperabilidade. A API disponibilizada não deve ser associada uma linguagem de programação única, pois dá liberdade ao desenvolvedor para que ele possa escolher a linguagem mais adequada a cada situação.

3.2 Principais tipos de Middlewares

Atualmente os middlewares se subdividem em algumas sub-áreas, cada uma seguindo um diferente modelo de programação. Neste sentido, grande parte da bibliografia segue, com pequenas variações, a seguinte classificação: Middlewares Reflexivos, Middlewares Baseado em Eventos, Middleware Orientado a Objetos, Middleware Orientado a Mensagens e Middleware baseado em Espaço de Tuplas. Cada um destes modelos são apresentados a seguir, identificando suas características principais.

3.2.1 Middleware Reflexivo

O conceito de Reflexão Computacional tem por objetivo prover uma maior flexibilidade ao sistema, permitindo que ele interfira em si mesmo a fim de ajustar seu comportamento para que tenha um controle mais efetivo sobre resultado de sua execução (FERNANDES, 2007). Um sistema reflexivo tem a capacidade de inspeção e adaptação em tempo de execução, onde a inspeção permite que o sistema possa observar seu estado atual, enquanto a adaptação permite que ele altere o seu comportamento em tempo de execução a fim de obter uma melhor compatibilidade com o ambiente. Nos sistemas computacionais tradicionais, uma vez o que o desenvolvedor tenha modelado e implementado o software, fica muito difícil adequar o comportamento da aplicação para casos não convencionais em que precisaria algum ajuste especial (FERNANDES, 2007). Esta necessidade é muito comum em sistemas distribuídos devido a heterogeneidade intrínseca deste tipo de sistema, onde existe uma grande variedade de tipos de hardwares, de sistemas de comunicação e no qual as desconexões são freqüentes (caso dos sistemas distribuídos móveis ou nômades). Com a introdução do conceito de reflexão em um middleware, alguns aspectos do seu comportamento interno são disponibilizados para as aplicações, de modo que ela possam inspecionar e alterar estes aspectos conforme a conveniência.

Uma das principais funções dos middlewares é servir como uma camada de abstração para as aplicações, ocultando assim as complexidades decorrentes do gerenciamento dos requisitos ao qual ele foi desenvolvido. Quanto maior a transparência que o middleware proporcionar as aplicações, maior serão as condições que terão que ser controladas por ele, acarretando em um sistema pesado no que se refere a computação. Isto impossibilitaria que um middleware com estas características fosse executado em um dispositivo móvel com poucos recursos (DELICATO, 2005). A reflexão permite que se implemente middlewares com mecanismos de reconhecimento de contexto (*context-*

aware) possibilitando, entre outras coisas, que o mesmo middleware se auto-adapte de forma que possa ser utilizado em um hardware com alta capacidade de processamento, com todos os seus serviços disponíveis, ou em um hardware com poucos recursos (como computadores de mão ou celulares), e neste caso utilizem apenas os serviços estritamente necessários (DELICATO, 2005).

3.2.2 Middleware baseado em Eventos

O middleware baseado em eventos é um middleware para sistemas distribuídos de larga escala que utiliza o modelo de comunicação publish/subcribe entre componentes. Sendo que um sistema publish/subcribe tradicional vai em direção a uma solução apenas referente à comunicação para sistemas distribuídos, sem se preocupar com outros requisitos que um sistema deste tipo exige. Já o middleware baseado em eventos tem o compromisso com os demais requisitos que um middleware tradicional deve suprir, como usabilidade, administrabilidade, interoperabilidade e extensibilidade. O texto desta seção foi baseada em (PIETZUCH, 2004).

O modelo publish/subcribe é um modelo de comunicação para sistemas distribuídos em que a esta se dá de forma assíncrona e no modelo muitos-para-muitos. Para disseminar as informações provenientes dos eventos existe um gerenciador, o sistema publish/subcribe, e dois tipos de clientes, os produtores de informações (*event publishers*) e os consumidores de informações (*event subcribers*). Os consumidores de informações fornecem uma descrição com o tipo de evento do qual querem ser notificados, e então conforme os produtores de informações publicam os eventos, estes são repassadas pelo sistema publish/subcribe aos consumidores adequados. Neste modelo de comunicação, os clientes são desacoplados o que define o assincronismo da comunicação, isto é, os clientes não precisam necessariamente terem conhecimento a respeito dos demais, pois toda a comunicação entre eles é gerenciada pelo sistema publish/subcribe, no caso o middleware. Este modelo de comunicação possibilita a sua utilização em sistemas de larga escala, isto é, provendo um meio eficiente de comunicação entre um grande número de clientes.

3.2.3 Middleware Orientado a Objetos

Esta classe de middleware se utiliza do paradigma de orientação a objetos, com a diferença que neste caso os objetos podem estar distribuídos por todo o sistema e não localmente como na concepção tradicional. Baseia-se no Princípio RPC (*Remote Procedure Call*) como forma de comunicação entre os dispositivos que compõem o sistema. Em uma chamada de programa uma aplicação pode fazer uma consulta ou realizar uma execução de um objeto que está localizado em um outro dispositivo da rede como se fosse um objeto local. Os objetos distribuídos podem ser considerados como procedimentos que são executados remotamente. A forma de comunicação durante estas chamadas se dá de forma síncrona, portanto a aplicação que realizou a chamada fica bloqueada até que obtenha uma resposta a sua solicitação.

Middlewares que implementam o modelo de orientação a objetos caracterizam-se por ter uma elevada carga computacional e elevada transparência na comunicação, o que compromete a sua aplicação em sistemas distribuídos que tenham participação de dispositivos com poucos recursos computacionais ou que necessitem algum tipo de consciência do contexto. A transparência da comunicação fica parcialmente comprometida devido

ao sincronismo da comunicação, pois devido aos tempos envolvidos entre as requisições e suas respectivas respostas, a diferença entre chamadas locais ou remotas fica relativamente explícito (YAMIN, 2004).

3.2.4 Middleware Orientado a Mensagens

Esta classificação caracteriza os middlewares que realizam a comunicação entre os elementos distribuídos através de troca de mensagens. A mensagem é uma informação que é enviada de um processo para outro, estando ele na mesma máquina ou em uma máquina remota. As mensagens podem ser enviadas para apenas um destinatário, caracterizando uma comunicação *point-to-point*, ou pode ser enviada para vários destinatário de uma só vez, que é o caso de comunicação *publich/subscribe*.

O modelo de comunicação *point-to-point* é a forma geralmente implementada em grande parte dos middlewares deste modelo, e isto permite uma comunicação tanto síncrona quanto assíncrona (SOCORRO VâNIA L. ALVES ENOQUE CALVINO ALVES, 2006). A forma assíncrona permite um desacoplamento entre o cliente e o servidor. A aplicação pode continuar o seu processamento assim que a mensagem é despachada, ficando a cargo do middleware o gerenciamento do envio e o recebimento da resposta, na qual o poderá ser lida pela aplicação assim que ela considerar conveniente (YAMIN, 2004).

A padronização das mensagens geradas pelas aplicações devem ser conheciadas pelos seus destinatários, porque a tarefa do middleware é de apenas rotear as mensagens sem interferir nos seus conteúdos (SOCORRO VâNIA L. ALVES ENOQUE CALVINO ALVES, 2006).

3.2.5 Middleware baseado em Espaço de Tuplas

Sistemas distribuídos móveis necessitam que a comunicação entre nodos remotos se dê de forma assíncrona para que exista um desacoplamento entre os processos que são executados nestes nodos. Isto se faz necessário porque as freqüentes desconexões que são características deste tipo de sistema não devem impossibilitar a comunicação entre os dispositivos. Métodos síncronos necessitam que a conexão não seja interrompida por isso da necessidade de assincronismo nestas situações.

O espaço de tuplas consiste de uma memória associativa compartilhada entre todos os nodos do sistema (YAMIN, 2004). Esta memória pode ser considerada como uma espécie de repositório para as estruturas denominadas tuplas. As tuplas podem ser definidas como um conjunto de dados estruturados que têm valores associados a tipos. Cada processo pode inserir ou ler tuplas neste espaço a qualquer momento, caracterizando o assincronismo da comunicação. O tempo de vida de uma tupla é independente do processo que as gerou, por isso não é necessário que os processo transmissor e receptor estejam simultaneamente disponíveis. Isto possibilita que uma informação possa ser enviada para um nodo, mesmo que ele esteja desconectado, que por sua vez pode ler esta informação assim que a conexão se restabeleça. Esta característica favorece sua utilização em ambientes móveis e dinâmicos em que as desconexões são freqüentes (YAMIN, 2004) (LOUREIRO et al., 2003).

4 MIDDLEWARE PARA REDES DE SENSORES

As redes de sensores são um tipo de sistema largamente distribuído com várias demandas particulares. Neste sentido este capítulo irá abordar as características intrísecas das redes de sensores e os aspectos envolvidos na sua gerência, considerando para isso a utilização de middlewares.

4.1 Redes de Sensores

As redes de sensores possibilitam prover uma ligação entre o mundo digital e o real, permitindo que os sistemas comutacionais tenham consciência do que acontece no mundo real e assim poderem fornecer serviços mais adequados. Outra perspectiva é permitir que os usuários tenham acesso às informações referentes ao seu ambiente de interesse sem que estejam fisicamente inseridos nele.

No contexto da computação pervasiva ou ubíqua, em que o ambiente computacional está em todo lugar, a todo momento e por meio de qualquer dispositivo (WEISER, 1991), é necessário que este ambiente, de alguma forma, tome conhecimento da realidade na qual está imerso (conciência do contexto). Se por um lado as redes de sensores "alimentam" a computação pervasiva fornecendo a ela informações do contexo, por outro lado a computação pervasiva permite que o usuário tenha acesso a as informações da rede de sensores de qualquer lugar. Sob esta ótica, as redes de sensores e a computação pervasiva têm uma relação próxima.

As redes de sensores estão bastante presentes nos dias atuais, em especial dentro das indústrias. Em um mundo globalizado, um dos principais fatores que levam a permanência de um produto no mercado é a sua relação qualidade/custo. Esta garantia de qualidade necessariamente passa por um controle muito rigoroso nos processos de produção. Por isso se faz necessário processos automatizados para monitorar e controlar os sistemas e assim conseguir qualidade a um custo de produção que possibilita a competitividade no mercado. Em processos industriaias automatizados, o uso de sensoriamento de todos ou partes dos processos é realidade ja há bastante tempo. Grande parte das plantas industriais atuais contemplam um elevado número de sensores monitorando as mais diversas grandezas físicas: temperatura, pressão, umidade, presença, etc. Estas grandezas, quando estão relacionada direta ou indiretamente ao processo de produção (variáveis de processo) devem ser monitoradas periodicamente por humanos ou por máquina (caso de processos automatizados) para que sejam controladas de acordo com a necessidade.

O acesso a estas informações geralmente é feito remotamente, principalmente quando se trata de um número muito grande de dados. Com a computação pervasiva estas informações podem ser acessadas em qualquer lugar, a qualquer momento em qualquer dispositivo, evidenciando a grande contribuição que este paradigma de computação pode proporcionar as redes de sensores e aos processo industriais de uma maneira geral.

Esta seção tem alguns conceitos que são centrais: (*i*) sensor, (*ii*) fenômeno e (*iii*) componente de software, que em algumas situações pode ser considerado como o observador. Segundo (PEREIRA; AMORIM; CASTRO, 2003), sensor é um dispositivo que implementa a monitoração física de um fenômeno ambiental e gera relatórios de medidas. Ele é o responsável por capturar um evento do ambiente (temperatura, pressão, umidade, etc.) e transformar em dados computáveis. Cabe salientar que esta definição não é unânime, porém é a que mais se adequada ao tipo de elemento considerado neste estudo. O segundo termo é o componente de software, que é uma adequação à definição apresentada por (PEREIRA; AMORIM; CASTRO, 2003) em que ela define um ente chamado observador. Componente de software é a entidade interessada no dados gerados pelos sensores em relação a um fenômeno físico. Em uma redes de sensores pode existir, simultaneamente, múltiplos componentes de software interessados nas informações geradas pelos sensores da rede. Considerando ao conceito dado anteriormente ao termo sensor, em que a informação produzida por ele é um dado computável, é mais adequado considerar o componente de software do que observador, quando nos referimos a este elemento, pois a informação deve necessariamente ter que ser tratada por algum software antes de se dar qualquer destino a ela. Por fim, definimos fenômeno como o aspecto de interesse do componente de software, isto é, a própria realidade que está sendo observada.

As redes de sensores podem ser fisicamente estruturadas, onde cada sensor têm posição fixa, tanto lógica quanto física. Neste caso, geralmente os sensores são interligados através de condutores elétricos utilizados para comunicação de dados e alimentação. Este tipo de infraestrutura não está sendo muito exploradas em pesquisas atualmente, mas sim uma outra concepção de redes de sensores: as redes de sensores sem fio. Neste caso, a complexidade envolvida é bem maior maior do que as existentes nas redes estruturadas, logo um sistema capaz de gerenciar esta complexidade poderá também vir a ser utilizado em redes estruturadas.

4.2 Redes de Sensores Sem Fio

O avanço da tecnologia na área da eletrônica e a demanda por mobilidade na computação móvel impulsionaram a evolução nas redes sem fio. Juntando-se a isso a constante redução no tamanho dos dispositivos eletrônicos e aos micro sistemas eletromecânicos impulsionaram a viabilidade das Redes de Sensores Sem Fio (RSSF) (RUIZ, 2003) (LOUREIRO et al., 2003). Este tema está sendo bastante explorado em vários trabalhos científicos atuais e por este motivo esta seção tem o intuito de explorar os principais aspectos em relação a este assunto.

As características principais deste tipo de rede é o elevado número de dispositivos (podendo chegar a centenas de milhares), o reduzido tamanho de cada elemento e possível mobilidade dos sensores fazendo com que a rede seja dinâmica, mudando freqüentemente a sua organização física e lógica. Cada sensor é um elemento autônomo capaz de captar a informação do meio, tratar e enviar estas informações através de uma comunicação sem-fio, para isso é necessário ter no mínimo um transceptor para comunicação, uma

unidade de sensoriamento, fonte de energia, memória e uma unidade de processamento (RUIZ et al., 2004). Sendo que tudo isso é projetado para ter dimensões bastante reduzidas. Todas essas características são necessárias para que possam ser utilizados, por exemplo, na monitoração de áreas remotas, sendo lançados sobre reservas ambientais, florestas, vulcões, rios, etc (RUIZ, 2003). As atuais condições climáticas sugerem aplicações das Redes de Sensores no acompanhamento da evolução das conseqüências do efeito estufa, como monitorar a temperatura e o derretimento das geleiras, o aumento do nível e da temperatura do mar, monitoração dos desertos e das condições atmosféricas. Dentro da indústria também são observadas algumas aplicações potenciais, como monitoração de temperatura ambiente, presença de gases tóxicos, temperatura em caldeiras e fornalhas, qualidade de grãos armazenados em silos, entre outras. Na medicina, suas características são favoráveis a utilização no monitoramento das condições de saúde de paciêntes críticos.

4.2.1 Caracterização das RSSF e de sua gerência

Devido as características intrínsecas das redes de sensores, um sistema de gerência deve ter diversos quesitos importantes, como tolerância a falhas, escalabilidade, custo dos elementos, ambiente operacional, topologia da rede, restrições de hardware, meio de transmissão e consumo de energia (RUIZ et al., 2004). Com isso vários trabalhos científicos vêm sendo apresentados fornecendo algumas soluções a essas novas questões, como (RUIZ, 2003) (DELICATO, 2005).

Os sensores são geralmente alimentados por baterias e em geral não recarregáveis. Muitas vezes estes sensores estão distribuídos em locais de difícil acesso ou até mesmo em ambientes inóspitos e por isso a interferência humana nem sempre é possível. Desta maneira, questões referentes ao gerenciamento eficiente da energia são amplamente estudadas para prolongar ao máximo o tempo de vida destes sensores. A transmissão dos dados é responsável por uma grande parte do consumo de energia (LOUREIRO et al., 2003) e para otimizar isso algumas propostas prevêem a redução do número e tamanho das mensagens a fim de transmitir os dados com uma maior eficiência. Outra idéia é aproveitar alta densidade de sensores para fazer ajustes na taxa de dados, no ciclo de operação, fazer o desligamento de sensores redundantes por determinados períodos de tempo e agregar dados para reduzir o número de pacotes transmitidos. O tratamento eficiente da transmissão das informações também contribui para atenuar diversos problemas decorrentes da grande quantidade de dados na rede, como colisões.

Estes sensores atuam de forma colaborativa aquisitando informações do ambiente e transmitindo para um ou mais pontos da rede para serem analisados e processados. Devem possuir a capacidade de se realocarem dentro da rede a cada vez que um novo sensor entra ou sai por um motivo qualquer, seja por término da energia ou por alguma interferência exterior.

As redes sem fio têm aspecto de segurança das informações bastante crítico. Como a limitação restrição física praticamente não existe (exceto pelo controle da potência do sinal), o controle da integridade e do vazamento dos dados deve se feito principalmente a nível de software. Existem métodos de criptografia bastante seguros, mas demandam um certo poder de processamento. Por isso as restrições computacionais dos sensores torna a segurança um problema crítico neste tipo de rede.

Os aspectos monitorados pelas redes de sensores referem-se a fenômenos reais que muitas vezes precisam ser observados em um determinado espaço de tempo que depende

do tipo de fenômeno. Existe um compromisso entre precisão, latência e eficiência em energia (PEREIRA; AMORIM; CASTRO, 2003). Quanto maior a freqüência com que as informações são aquisitadas menor será a vida útil do sensor, devido ao maior consumo de energia. Soma-se a isso também uma provável redução na escalabilidade do sistema. O sistema de gerênciamento deve ter capacidade de se adaptar para permitir o ajuste do tempo de retardo das informações e assim proporcionar uma maior eficiência do sistema.

O término da bateria dos sensores ou problemas na comunicação, assim como avarias devido as condições do ambiente podem levar a falhas na rede fazendo com que dados se percam. Em grande parte dos casos não é possível a substituição dos sensores defeituosos, e em outros casos o tempo para se realizar esta substituição pode não ser aceitável, por isso é desejável que o sistema de gerência tenha algum método de tolerâcia a falhas. Desta maneira as aplicações não são afetadas por estas falhas na rede. Algumas formas de resolver este problema são a replicação dos dados e utilização de roteamentos múltiplos (PEREIRA; AMORIM; CASTRO, 2003), porém novamente existe um compromisso entre esses métodos e a eficiência em energia.

As características das Redes de Sensores sugerem inúmeras utilizações que não eram possíveis com as estruturas convencionais. O baixo custo de cada sensor assim como da infra-estrutura, a capacidade de se auto-organizar e a mobilidade, tornam a quantidade de possibilidades de aplicações muito grande. Na indústria, o alto custo das instalações, a alta taxa de falha em conectores e a dificuldade em detectar a localização da falha tem impulsionado a busca por soluções sem fio (SHEN; WANG; SUN, 2004). Os sensores convencionais, comumente usados nas indústrias, são peças robustas que possuem invólucros grandes e utilizam fios tanto para a energia quanto para a transmissção das informações coletadas. Cada tipo de sensor tem suas próprias características que devem ser adequadas seja eletricamente ou por meio de software, e isto geralmente é feito de forma manual. Por isso estes sensores são instalados de modo a permanecer o maior tempo possível no mesmo lugar, devido aos elevados custos de adequação de software e de instalação. Em grande parte dos casos o número de elementos sensores usados é o mínimo possível por uma questão de economia. Com as Redes de Sensores estes custos caem bruscamente, permitindo que sejam usados mais sensores e em situações diferenciadas, possibilitando a monitoração de um número muito maior de variáveis ambiente.

Figura 4.1: Tipos de sensores (HW-GROUP, 2007) (SENSOR-WEBS, 2007) (SMART-DUST, 2001)

Já existem algumas aplicações de Redes de Sensores na Produção Industrial, no

monitoramento de parâmetros como fluxo, pressão, temperatura, e nível, identificando problemas como vazamento e aquecimento, em indústrias petroquímicas, fábricas, refinarias e siderúrgicas (LOUREIRO et al., 2003). Alguns exemplos mais específicos que representam um grande potencial para o uso de Redes de Sensores sem Fio são o gerenciamento de plantas químicas devido a necessidade de um controle bastante preciso dos processos, na indústria do papel no controle dos complexos sistemas de rolamento e nas refinaria de óleo onde as condições de ambiente são extremas e o número de pontos de sensoriamento muito elevado (SHEN; WANG; SUN, 2004). Em aplicações industriais se faz necessário o mínimo de operação e intervenção no sistema (SHEN; WANG; SUN, 2004). Por este motivo as Redes de Sensores aparecem com um papel fundamental neste processo, onde o baixo custo dos sensores, a não necessidade de cabos e a possibilidade de auto-organização da rede, torna este tipo de sistema extremamente interessante para o meio industrial.

4.2.2 Clssificação das RSSF

As RSSF são um tipo de sistema que tem grande dependência com relação ao modelo de aplicação para o qual foi projetada. As demandas das aplicações influenciam em aspectos físicos e lógicos como: arquietura do sensor utilizado, quantidade de dispositivos (nodos sensores ou nós), na disposição destes sensores (e conseqüentemente da organização da rede), protocolos de comunicação, etc.

Para classificar as RSSFs, conforme (RUIZ, 2003), deve-se levar em conta vários aspectos como: configuração (tabela 4.1), sensoriamento (tabela 4.2), comunicação (tabela 4.4 e 4.5) e processamento (tabela 4.3).

4.2.3 Middleware para Redes de Sensores

Um middleware para RSSF deve gerenciar os aspectos pertinentes aos sistemas distribídos móveis e dinâmicos além dos aspectos característicos das redes de sensores, como o gereciamento eficiente da energia dos nodos sensores, robustez e escalabilidade (LOUREIRO et al., 2003). Portanto os principais propósisto dos middlewares para RSSF é dar suporte ao desenvolvimento e execução de aplicações sensoradas, provendo uma interface de programação de alto-nível e gerenciar a utilização dos recursos da rede ao atender as demandas das aplicações (DELICATO, 2005). Desta maneira, as solicitações das aplicações devem ser gerenciadas repassando tarefas para a RSSF, coordenando os nós da rede na execução das tarefas e posteriormente coletando e agregando os dados de modo a reportar para a aplicação uma informação de alto nível.

Os aspectos de baixo nível da rede de sensores, como as características da infraestrutura e prococolos inferiores, devem ser ocultados das aplicações. O middleware deve gerenciar a heterogeneidade física e lógica do sistema, abstraindo estes aspectos das aplicações. Deve tomar decisões a respeito da topologia e dos protocolos utilizados entre os nós da rede, sempre levando em conta requisitos de QoS (*Quality of Service*), escalabilidade, robustez e de gerenciamento eficiênte da energia (DELICATO, 2005) (BARBOSA et al., 2005).

Como as redes de sensores monitoram aspectos físicos do ambiente real, é necessário que se tenha um gerenciamento de tempo e espaço em relação aos fenômenos observados. O dado sensorado deve estar associado a uma grandeza de tempo e outra de localização, caracterizando onde e quendo aconteceu tal fenômeno. Portanto é desejável

Tabela 4.1: RSSF segundo a configuração (RUIZ, 2003).

Composição	Homogênea	Rede composta de nós que apresentam a mesma capacidade de hardware. Eventualmente os nós podem executar software diferente.
	Heterogênea	Rede composta por nós com diferentes capacidades de hardware.
Organização	Hierárquica	RSSF em que os nós estão organizados em grupos (*clusters*). Cada grupo terá um líder (cluster-head) que poderá ser eleito pelos nós comuns. Os grupos podem organizar hierarquias entre si.
	Plana	Rede em que os nós não estão organizados em grupos
Mobilidade	Estacionária	Todos os nós sensores permanecem no local onde foram depositados durante todo o tempo de vida da rede.
	Móvel	Rede em que os nós sensores podem ser deslocados do local onde inicialmente foram depositados.
Densidade	Balanceada	Rede que apresenta uma concentracção e distribuição de nós por unidade de área considerada ideal segundo a função objetivo da rede.
	Densa	Rede que apresenta uma uma alta concentração de nós por unidade de área.
	Esparça	Rede que apresenta uma baixa concentração de nós por unidade de área.
Distribuição	Irregular	Rede que apresenta uma distribuição não uniforme dos nós na área monitorada.
	Regular	Rede que apresenta uma distribuição uniforme de nós sobre a área monitorada

que o middleware tenha algum tipo de protocolo de sincronização entre os nodos sensores assim com método de localização física dos mesmos.

Segundo (DELICATO, 2005), o projeto de um middleware para RSSF deve seguir alguns princípios. São eles:

- o middleware deve fornecer mecanismos centrados em dados para o processamento e a consulta de dados dentro da rede;
- conhecimento da aplicação deve ser usado para otimizar o funcionamento da rede. É, então, importante integrar conhecimento da aplicação aos serviços fornecidos pelo middleware;
- todas as soluções devem, de preferência, ser baseadas na utilização de algoritmos localizados. Tais algoritmos fornecem robustez e escalabilidade ao sistema;
- o middleware deve ser leve em termos de requisitos de comunicação e computação. Tal requisito indica a necessidade de usar heurísticas simples e eficientes que gerem soluções sub-ótimas para as decisões tomadas pelo middleware;

Tabela 4.2: RSSF segundo o sensoriamento (RUIZ, 2003).

Coleta	Periódica	Os nós sensores coletam dados sobre o(s) fenômeno(s) em intervalos regulares. Um exemplo são as aplicações que monitoram o canto dos pássaros. Os sensores farão a coleta durante o dia e permaneceram desligados durante a noite.
	Contínua	Os nós sensores coletam os dados continuamente. Um exemplo são as aplicações de exploração interplanetária que coletam dados continuamente para a formação de base de dados para pesquisas.
	Reativa	Os nós sensores coletam dados quando ocorrem eventos de interesse ou quando solicitado pelo observador. Um exemplo são as aplicações que detectam a presença de objetos na área monitorada.
	Tempo Real	Os nós sensores coletam a maior quantidade de dados possível no menor intervalo de tempo. Um exemplo são aplicações que envolvem risco para vidas humanas tais como aplicações em escombros ou áreas de desastres. Um outro exemplo são as aplicações militares onde o dado coletado é importante na tomada de decisão e definição de estratégias.

- devido aos recursos limitados, é muito provável que os requisitos de desempenho de todas as aplicações em execução não possam ser simultaneamente satisfeitos. Portanto, é necessário que o middleware negocie inteligentemente a QoS de várias aplicações umas contra as outras.

As RSSF são um tipo sistema que depende a aplicação para o qual se destina (MACEDO et al., 2004), por isso o middleware deve ter conhecimento desta aplicação e utilizar isto nas suas tomadas de decisões. Para manter a generalidade do middleware, deve haver uma forma de caracterizar a aplicação de modo a evidenciar a sua especificidade. Isto pode ser feito criando categorias de aplicações semelhantes, onde cada aplicação encaixa-se em um perfil específico.

O ambiente de uma RSSF é altamente dinâmico, em grande parte dos casos, por isso o middleware deve ter a capacidade de se autoadaptar durante a execução das aplicações. Por este motivo é desejável que ele tenha característica reflexíva. Tanto o middleware quanto as aplicações devem ter conciência do contexto (*context-aware*), para então terem condições de decidir sobre as ações a serem tomadas em relação ao comportamento do sistema.

O contexo consiste em aspectos internos e externos (CAPRA; EMMERICH; MASCOLO, 2001). Os aspectos internos referem-se ao as características físicas do dispositivo, isto é, seus recursos: energia da bateria, capacidade de processamento, quantidade de memória, etc. Já os aspectos externos são referentes ao ambiente onde o dispositivo está inserido, como: características da rede e localização. Em RSSF aida existe uma outra forma de conciência do contexto, representada pelos fenômenos medidos através dos sensores que refletem o estado da aplicação (DELICATO, 2005).

Tabela 4.3: RSSF segundo o processamento (RUIZ, 2003).

Cooperação	Infraestrutura	Os nós sensores executam procedimentos relacionados à infra-estrutura da rede como por exemplo, algoritmos de controle de acesso ao meio, roteamento, eleição de líderes, descoberta de localização e criptografia.
	Localizada	Os nós sensores executam além dos procedimentos de infra-estrutura, algum tipo de processamento local básico como por exemplo, tradução dos dados coletados pelos sensores baseado na calibração.
	Sob Correlação	Os nós estão envolvidos em procedimentos de correlação de dados como fusão, supressão seletiva, contagem, compressão, multi-resolução e agregação.

4.2.4 Classificação dos middlewares para RSSF

O middleware provê uma elevada abstração da RSSF de modo que as aplicações possam realizar consultas de alto nível aos dados sensorados. Sob esta perpectiva, algumas soluções propõem uma abstração da rede de sensores como um banco de dados distribuído, possibilitando consultas aos dados em linguagem SQL (*Structured Query Language*). Outras propostas sugerem que as aplicações acessem aos dados utilizando um espaço de tuplas.

Sob esta ótica, para (HENRICKSEN; ROBINSON, 2006) a maioria das soluções atuais de middlewares para redes de sensores se encaixam em uma das seguintes categorias:

- *Abordagem baseada em banco de dados*: esta categoria faz uma abstração da rede de sensores como se fosse um banco de dados distribuído. Desta maneira utiliza a linguagem SQL como forma padão de consulta aos dados sensorados. Como exemplos deste tipo de abordagem podemos citar o COUGAR (BONNET; GEHRKE; SESHADRI, 2001), SINA (SHEN; SRISATHAPORNPHAT; JAIKAEO, 2001) e TinyDB (MADDEN et al., 2005);

- *Abordagem baseada em espaço de tuplas*: utiliza um espaço de tuplas como uma espécie de repositório central onde os dados dos sensores são armazenados. Nesta abordagem, os dados podem ser inseridos, lidos e retirados do espaço de tuplas que é comum para as aplicações. O middleware TinyLIME (CURINO et al., 2005) é um exemplo desta abordagem;

- *Abordagem baseada em eventos*: esta abordagem utiliza o conceito de middleware baseado em evendos (ver seção 3.2.2). Este tipo de middleware proporciona uma forte dissociação entre produtor e consumidor de informações o que é bastante apropriado em aplicações em RSSF. Como exemplo desta abordagem tem-se o middleware Mires (SOUTO et al., 2006);

- *Abordagem baseada na descoberta de serviços*: esta abordagem é representada pelo middleware MiLAN (HEINZELMAN et al., 2004), que utiliza um protocolo de

Tabela 4.4: RSSF segundo a comunicação - Tabela A (RUIZ, 2003).

Disseminação	Programada	Os nós disseminam em intervalos regulares.
	Contínua	Os nós disseminam os dados continuamente.
	Sob Demanda	Os nós disseminam os dados em resposta à consulta do observador e à ocorrência de eventos.
Tipo de conexão	Simétrica	Todas as conexões existentes entre os nós sensores, com exceção do nó sorvedouro têm o mesmo alcance.
	Assimétrica	As conexões entre os nós comuns têm alcance diferente.
Transmissão	Simpléx	Os nós sensores possuem transceptor que permite apenas transmissão da informação.
	Half-Duplex	Os nós sensores possuem transceptor que permite transmitir ou receber em um determinado instante.
	Full-Duplex	Os nós sensores possuem transceptor que permite transmitir ou receber dados ao mesmo tempo.

descoberta de serviços para identificar os sensores que satisfazem as necessidades de sensoriamento informadas pelas aplicações.

Tabela 4.5: RSSF segundo a comunicação - Tabela B (RUIZ, 2003).

Alocação de Canal	Estática	Neste tipo de rede se existirem "n" nós, a largura de banda é dividida em "n" partes iguais na freqüência (FDMA - *Frequency Division Multiple Access*), no tempo (TDMA - *Time Division Multiple Access*), no código (CDMA - *Code Division Multiple Access*), no espaço (SDMA - *Space Division Multiple Access*) ou ortogonal (OFDM - *Orthogonal Frequency Division Multiplexing*). A cada nó é atribuída uma parte privada da comunicaçao, minimizando interferência.
	Dinâmica	Neste tipo de rede não existe atribuição fixa de largura de banda. Os nós disputam o canal para comunicação dos dados.
Tipo de Fluxo de Informação	*Flooding*	Neste tipo de rede, os nós sensores fazem *broadcast* de suas informações para seus vizinhos que fazem *broadcast* desses dados para outros até alcançar o ponto de acesso. Esta abordagem promove um alto overhead mas está imune às mudanças dinâmicas de topologia e a alguns ataques de impedimento de serviço (DoS - Denial of Service).
	Multicast	Neste tipo de rede os nós formam grupos e usam o *multicast* para comunicação entre os membros do grupo.
	Unicast	Neste tipo de rede, os nós sensores podem se comunicar diretamente com o ponto de acesso usando protocolos de roteamento multi-saltos.
	Gossiping	Neste tipo de rede, os nós sensores selecionam os nós para os quais enviam os dados.
	Bargaining	Neste tipo de rede, os nós enviam os dados somente se o nó destino manifestar interesse, isto é, existe um processo de negociação.

5 SOA: UM MODELO DE REFERÊNCIA PARA A IMPLEMENTAÇÃO DE SERVIÇOS E MIDDLEWARES

Web services é uma tecnologia baseada no paradigma de Arquitetura Orientada a Serviços (SOA) que é capaz de prover aos sistemas uma elevada transparência, interoperabilidade, independência de linguagem e plataforma (MARQUEZAN; SILVA CARISSIMI; NAVAUX, 2006). Estas características são desejáveis em middlewares para redes de sensores por isso este capítulo aborda os conceitos envolvidos na Arquitetura Orientada a Serviços e as tecnologias que constituem os web services.

5.1 Descrição do paradigma

Uma definição para Arquitetura Orientada a Serviços é apresentada por (MACKENZIE et al., 2006) como "um paradigma para organização e utilização de competências distribuídas que estão sob controle de diferentes domínios proprietários". As competências são criadas por pessoas para resolver algum tipo de problema encontrado em uma atividade qualquer, seja ela pessoal ou profissional. Podem ser definidas como a capacidade que esta entidade adquiriu para poder satisfazer uma necessidade sua em algum momento. É provável que outras pessoas ou organizações venham ter necessidades semelhantes em alguma situação e assim possam utilizar as competências de terceiros sem que estas tenham que adquirir suas próprias competências para satisfazer essas necessidades. Tipicamente, enquanto umas entidades possuem competências que podem ser disponibilizadas, outras possuem necessidades que podem ser satisfeitas com a utilização das competências que as primeiras disponibilizaram. Uma única competência pode ser suficiente para atender uma ou mais necessidades. O inverso também é verdadeiro, uma necessidade pode precisar de várias competências para encontrar a solução adequada e assim uma nova competência pode estar sendo criada.

Do ponto de vista computacional, este paradigma pode ser considerado como uma evolução da computação distribuída, onde o indivíduo que tem as competências pode ser considerado como um fornecedor de serviços enquanto o que tem necessidades como um consumidor destes serviço, e assim serão chamados no restante do texto. Tipicamente o fornecedor e consumidor de serviços ficam em máquinas distintas, embora isso não seja uma obrigatoriedade. Desta forma uma aplicação que tiver uma determinada necessidade pode se tornar um um consumidor de serviços, e através de uma rede de comunicação pode utilizar algum serviço disponibilizado por aplicações fornecedoras que estiverem

em uma outra máquina a fim de solucionarem seus problemas.

O serviço é um elemento central na descrição do paradigma SOA, por isso será apresentada uma definição para este termo. Em dicionários a definição mais comum diz que serviço é um trabalho que alguém realiza para outro. Para (MACKENZIE et al., 2006), este termo tem um sentido mais específico e está associado a três idéias: *(i)* deve existir uma competência envolvida no trabalho a ser realizado, *(ii)* uma especificação deste trabalho e *(iii)* a sua oferta. Desta maneira, não basta que exista uma competência para que uma entidade se torne apta a ser fornecedora de serviços, ela precisa fornecer uma especificação detalhada deste serviço e então disponibilizá-lo para que então possa ser utilizado. Completando este conceito, (BOOTH et al., 2004) diz que serviço é um recurso abstrato que pode realizar tarefas que representam uma funcionalidade do ponto de vista de entidades fornecedoras e entidades consumidoras.

Para que um serviço se torne usável, o fornecedor deve disponibilizar uma descrição do serviço oferecido. Esta descrição permite aos consumidores conhecerem as competências disponibilizadas pelo fornecedor através do serviço e as suas condições de utilização, de modo que o consumidor em potencial possa avaliar se o serviço satisfaz adequadamente as suas necessidade. A descrição deve conter também todas as informações necessárias ao estabelecimento da interação entre as entidades quando da utilização do serviço, como o formato, conteúdo e a seqüencia das informações trocadas, assim como as semânticas relacionadas.

O serviço se comporta como uma espécie de "caixa preta" para o consumidor de serviços, por que são ocultados aspectos da sua implementação. O fornecedor é o proprietário do serviço e tem o controle total da solução desenvolvida, não necessitando assim informar como isso foi feito. Ele apenas deve fornecer os aspectos realmente necessários à utilização do serviço como descrição do serviço e a interface. Esta última especifica os aspectos sintáticos necessários para a interação com o serviço.

5.2 A interação entre consumidores e fornecedores de serviço

Antes de iniciar uma interação entre fornecedor e consumidor de serviços, é necessário que alguns aspectos sejam satisfeitos. A primeira situação que deve ocorrer é que ambos tenham consciência da existência do outro. O requisito inicial para que isso ocorra é que o fornecedor de serviços disponibilize a descrição do serviço e as políticas que regem a sua utilização. Tipicamente o fornecedor de serviços deve anunciar seu serviços em um diretório de serviços de modo que os consumidores possam localizá-los. Porém no caso de haver uma maior demanda do que do que a efetiva disponibilidade de uma determinada capacidade, é conveniente a situação inversa, isto é, os consumidores anunciarem suas necessidades e aguardarem fornecedores aptos a satisfaze-las. Além disso, para que exista a interação deve haver a intenção mútua de que isto aconteça e que ambos tenham condições favoráveis, isto é, estejam aptos para se comunicarem e realizarem suas atividades. Este conjunto de condições é chamado de visibilidade.

Estando satisfeitas as condições de visibilidade, fornecedores e consumidores estão aptos a começarem uma interação, que consiste na comunicação entre ambos e na execução das ações envolvidas no serviço em questão de modo a satisfazer as necessidades do consumidor de serviços. Tipicamente a comunicação entre fornecedor e

consumidor de serviços se dá através da troca de mensagens, embora outras formas possam ser utilizadas. Esta troca de informações deve seguir padrões que sejam de pleno conhecimento de ambos e chamaremos este padrão de modelo de informação. Este modelo independe da arquitetura utilizada para implementar o fornecedor e o consumidor de serviços, o que define uma importante característica do paradigma SOA: a interoperabilidade. O modelo de informação deve ter um forte comprometimento sintático e semântico para que as informações trocadas entre as partes sejam corretamente interpretadas e assim não acarretem problemas na execução do serviço.

Para uma interação bem sucedida é imprescindível um conhecimento básico a respeito do comportamento das ações envolvidas no serviço e as suas conseqüências em um determinado processo na qual este serviço possa estar envolvido. Por exemplo, um serviço que faça um acesso a uma informação que necessite uma autenticação prévia, exige que seja enviado os dados de autenticação e antes de efetivamente acessar a informação, portanto a seqüência de ações, bem como a correta autenticação são pré-requisitos para que se possa receber a informação desejada. Também são necessárias noções de aspectos temporais envolvidos nas ações realizadas durante a utilização do serviço. Estes aspectos representam o modelo comportamental do serviço.

Quando o consumidor utiliza um serviço com o propósito de satisfazer uma necessidade própria, o resultado desta interação pode ser chamado efeito do serviço. Este efeito pode ser uma resposta a uma requisição ou mesmo uma mudança em algum estado compartilhado por ambos. Se por exemplo, consideramos um serviço de gerenciamento de contas bancárias, a primeira situação poderia ser a consulta do saldo de uma conta e a segunda uma transferência monetária entre contas bancárias.

5.3 Considerações sobre o serviço

O serviço é o elemento central do paradigma SOA. Por este motivo, é pertinente tratar os conceitos que se referem diretamente ao serviço a fim de detalhar melhor o paradigma.

5.3.1 Descrição do serviço

Para (MACKENZIE et al., 2006), a descrição do serviço tem a finalidade de facilitar a interação e a visibilidade, principalmente quando fornecedor e consumidor estão em domínios proprietários diferentes. Nem todas as informações a respeito do serviço precisam ser informadas, pelo contrário, detalhes referentes a sua implementação são ocultadas em grande parte dos casos. Tipicamente são disponibilizadas somente informações críticas, isto é, as que são essenciais para o serviço possa ser utilizado, como as suas funcionalidades e seu modelo de informação. Os itens que devem ser informados são (MACKENZIE et al., 2006):

- função ou conjunto de funções que executa;
- disponibilidade e acessibilidade;
- restrições e políticas de uso;
- modelo de informação e de comportamento.

A descrição deve apresentar claramente as funcionalidades do serviço os seus efeitos produzidos. As informações devem ser suficientes para que o consumidor possa analisar as competências oferecidas pelo serviço e avaliar se elas são úteis na solução de seus problemas. Esta descrição deve apresentar uma forma textual para leitura por pessoas e palavras-chave para facilitarem a busca por máquina.

Algumas questões referentes a acessibilidade podem ser independentes do fornecedor e consumidor de serviço, como problemas na infraestrutura de rede. Porém algumas informações devem ser disponibilizadas para facilitar a interação entre as partes, por exemplo, tipos de protocolos requeridos, auxílios na sua localização e informações dinâmicas que indiquem se ele está atualmente acessível.

As políticas de utilização devem ser apresentadas de forma que o consumidor possa avaliar se estas condizem com suas próprias restrições.

Para que um consumidor utilize um serviço, é necessário que ele utilize uma interface de serviço. Esta interface possui os protocolos, comandos e tipos de dados usados nas informações trocadas, de modo a apresentar a sintaxe que deve ser utilizada na interação entre as partes. Constitui o modelo de informações do serviço e precisa der informado na sua descrição em um formato que permita um entendimento claro pelo consumidor.

5.3.2 Políticas e contratos

As políticas são as condições que definem as restrições envolvidas na utilização do serviço podendo ser definidas por qualquer participante, seja ele um fornecedor ou consumidor de serviços. Já os contratos, são acordos feitos entre as partes envolvidas, porém também têm por finalidade definir condições de utilização do serviço.

Desta maneira, a política sempre representa o ponto de vista de um dos participantes (MACKENZIE et al., 2006). Ela se apresenta geralmente com uma condição de imposição da parte geradora da política. Por exemplo, o fornecedor de serviços pode impor que todas as mensagens devem ser transmitidas criptografadas e se isso não acontecer a interação não será estabelecida. Porém pode ocorrer que esta política represente apenas uma condição de desejo, caracterizando assim uma política não impositiva. Por exemplo se uma das partes indicasse a criptografia dos dados apenas como preferencial e não imperativa.

As possibilidades de aplicação das políticas no SOA são bastante diversificadas, podendo abranger desde questões referentes a infraestrutura, como segurança, privacidade, qualidade de serviço, quanto a aspectos específicos do serviço que está sendo prestado, como no caso de aplicações financeiras em que são impostas restrições de alguns aspectos de acordo com o tipo de usuário que está utilizando o serviço.

Ao contrário das políticas que estão associadas às condições apresentadas por apenas um dos participantes do serviço, os contratos são definidos a partir de acordos estabelecidos entre ambos. No entanto podem abranger aspectos semelhantes aos das políticas como restrições e condições de utilização dos serviços. Deste modo questões como segurança e qualidade de serviços também podem ser regidas através de acordos, assim como os aspectos envolvidos as funcionalidades do serviço.

As política impositivas envolvidas no serviço devem ser apresentadas na descrição do serviço (MACKENZIE et al., 2006). Já os acordos, podem ou não serem expressos na descrição, visto existir a possibilidade que estes sejam produzidos dinamicamente em uma negociação entre as partes.

5.4 Web services: um exemplo de aplicação

O paradigma SOA tem uma grande gama de possíveis aplicações no projeto e desenvolvimento de software de uma forma geral. As definições apresentadas até aqui abordam aspectos gerais do paradigma sem abordar nenhuma aplicação específica. Como exemplo de aplicação do paradigma será apresentado o modelo de arquitetura de serviços web ou web services.

Um web service é uma aplicação que implementa o SOA, cujas trocas de mensagens entre os participantes se dá utilizando o padrão XML, transmitidos pela Internet ou rede corporativa utilizando protocolos padronizados amplamente utilizados. O principal benefício dos web services é facilitar a integração de sistemas distribuídos acessíveis através da web (DELICATO, 2005). A interoperabilidade que a arquitetura de serviços web possibilita, permite que aplicações desenvolvidas por diferentes domínios proprietários sejam integrados independente do modelo de programação e plataforma utilizada. Para que seja possível esta integração são necessários apenas um conjunto de protocolos abertos de comunicação e formato de dados, como HTTP, SOAP e XML, o que garante a interoperabilidade desejada.

A estrutura básica dos web services é constituída por fornecedores e consumidores de serviços além de um registro, onde os serviços são publicados para facilitar a sua localização. O fornecedor deve publicar seus serviços no registro de serviços e fornecer a sua descrição em um documento que deve conter todas as informações necessárias para que o serviço possa ser acessado. O registro de serviços concentra uma lista de serviços disponíveis e suas respectivas descrições, para que possam ser localizados pelos consumidores. Deste modo, para que um consumidor possa acessar um serviço, ele deve primeiramente procurar em um ou mais registros pelo serviço desejado. Ao encontra-lo, ele deve utilizar as informações contidas na sua descrição e então efetivamente acessar o serviço junto ao fornecedor.

As operações básicas envolvidas na utilização de um web service são: *(i)* a publicação do serviço por parte do fornecedor, *(ii)* a descoberta dos serviços disponíveis por parte do consumidor junto aos registros e *(iii)* a ligação ao serviço (DELICATO, 2005). Esta seqüência de operações pode ser visualizado na figura 5.1.

As padronizações envolvidos na arquitetura de web services abrangem a descrição dos serviços, os protocolos utilizados na comunicação e a publicação e descoberta dos serviços. O que é comum entre ambos é o padrão XML o que possibilita a interoperabilidade entre os participantes da comunicação.

A descrição dos serviços é representada em um documento que utiliza a linguagem WSDL (*Web Services Description Language*) que é baseada no padrão XML - outras informações na seção 5.4.1.2. Na descrição estão todas as informações necessárias para que o web service possa ser utilizado, incluindo o endereço para que ele seja encontrado na rede além de informações específicas do serviço, como tipos de dados e ações fornecidas.

As operações de descoberta e publicação de serviços na arquitetura de web services utiliza o protocolo UDDI (*Universal Description Discovery and Integration*), que é um protocolo baseado em XML utilizado para comunicação com registros de serviços. Ele prove uma organização dos serviços em uma estrutura de diretórios, onde eles são organizados de forma a facilitar a publicação dos serviços e sua descoberta. Tipicamente, os fornecedores de serviço publicam os seus serviços e interfaces em um registro através de um protocolo UDDI e então os consumidores de serviços podem acessar este regis-

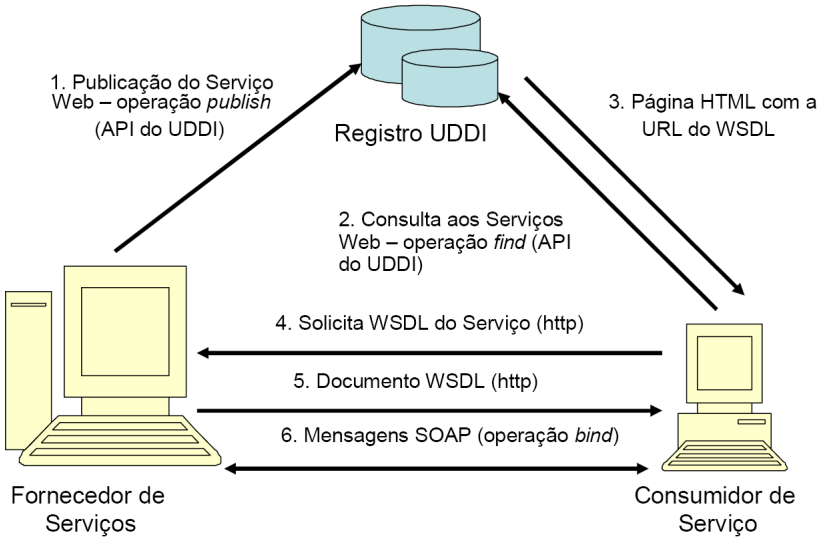

Figura 5.1: Modelo de web services (DELICATO, 2005)

tro na busca por serviços que satisfaçam suas necessidades num processo chamado de descoberta. Informações adicionais na seção 5.4.1.3

5.4.1 Tecnologias utilizadas

Esta seção descreve as tecnologias utilizadas nos web services. Entre elas o SOAP - um protocolo utilizado no transporte, o WSDL - um modelo de documento para descrever web services e o UDDI que é um protocolo utilizado para a comunicação com registros de serviços.

5.4.1.1 SOAP (*Simple Object Access Protocol*)

O SOAP (*Simple Object Access Protocol*) é um protocolo de "transporte" baseado no padrão XML, que tem por objetivo a troca de informações entre aplicações distribuídas (MARQUEZAN; SILVA CARISSIMI; NAVAUX, 2006) (DELICATO, 2005). Este protocolo, que atualmete está na versão SOAP 1.2 (MITRA; LAFON, 2007), contribuiu para se obter interoperabilidade entre processos comunicantes, uma das características que faz com que os web services tenham tido tanta atenção por diversas áreas da computação atualmente. As mensagens transmitidas pelo SOAP não necessitam obrigatoriamente de uma resposta por parte do receptor, pois simplesmente definem um conjunto de dados a ser enviado. Porém se for necessário, ele permite a construção de formas complexas de interação como requisição/resposta ou requisição/múltiplas-respostas (MARQUEZAN; SILVA CARISSIMI; NAVAUX, 2006).

A especificação do SOAP permite criar regras para estruturar os dados a serem transmitidos, como as informações que devem ser trocadas entre as aplicações: nome de operações, parâmetros e valores de retorno e mensagens de erro. Para que exista compatibilidade entre os tipos de dados, o SOAP possui sua própria convenção de tipos, ficando a cargo das aplicações fazerem as conversões necessárias ao seu padrão particular. As definições do SOAP permitem que se crie mecanismos de chamada de procedimentos remotos através de um mecanismo chamado SOAP-RPC (SOAP *Remote Procedure Call*). Ele estabelece regras para definir o nome do procedimento remoto, parâmetros a serem

enviados e mensagens de resposta.

As mensagens SOAP possuem uma estrutura que contêm basicamente três elementos: o envelope, que é o elemento principal, e o cabeçalho e corpo que são partes do envelope (Figura 5.2). Estes três elementos são descritos a seguir:

envelope : é um elemento obrigatório que trata-se da própria mensagem que é transmitida entre as aplicações distribuídas;

cabeçalho (*header*) : é um elemento opcional definido no esquema do envelope SOAP que agrega características adicionais e acordos entre as partes envolvidas na comunicação. Contempla informações que qualificam a forma como a mensagem deve ser processada. Por exemplo, informações de autenticação no serviço, isto é, o restante da mensagem só será processada se os dados de autenticação estiverem corretos. Uma mensagem SOAP pode conter mais de um cabeçalho.

corpo (*body*) : é um elemento obrigatório que contém as informações que efetivamente deverão ser processadas pelo receptor. Pode servir para vários propósitos, como para transmissão de documentos ou para chamadas de procedimentos remotos (SOAP-RPC). Em uma mensagem SOAP pode existir mais de um corpo.

Figura 5.2: Mensagem SOAP (MARQUEZAN; SILVA CARISSIMI; NAVAUX, 2006)

5.4.1.2 WSDL (*Web Services Description Language*)

O WSDL (*Web Services Description Language*) é um modelo de documento baseado no padrão XML usado para descrever web services. Sua estrutura é dividida entre parte abstrata (definição de serviços e mensagens) e parte concreta (referente a sua

implementação). Atualmente está na versão WSDL 2.0 (CHINNICI et al., 2007), mas a versão WSDL 1.1 ainda é utilizada (CHRISTENSEN et al., 2001).

A estrutura de um documento WSDL deve apresentar os tipos de dados, operações e os protocolos utilizados pelo serviço. Para o padrão WSDL, cada operação é um comportamento ou ação oferecida pelo web service, permitindo especificar para cada um, os parâmetros de entrada e saída, assim como os tipos de dados utilizados. Os protocolos de ligação também devem ser informados nesta especificação.

Os documentos WSDL utilizam um conjunto de definições para descrever um web service, que são pertencentes a dois grandes grupos: as definições concretas e as abstratas. Cada uma delas apresenta um conjunto de elementos com funções bem específicas. Estes elementos são apresentados abaixo segundo as definições extraídas de (MARQUEZAN; SILVA CARISSIMI; NAVAUX, 2006) e (DELICATO, 2005).

As definições abstratas são formadas por elementos que correspondem às típicas documentações de API. Esses elementos são:

- *types*: fornece a definição dos tipos de dados (independente de plataforma ou linguagem) usados para descrever as mensagens trocadas entre aplicações;
- *message*: descreve os dados trocados, utilizando as definições dos tipos de dados;
- *porttype*: é um conjunto de operações oferecidas por um ou mais endpoints, cada operação se refere a mensagens de entrada, saída ou erro;
- *operation*: descreve uma ação fornecida pelo serviço.

As definições concretas são formadas por elementos que definem as especificidades de como acessar, através da rede, as funcionalidades que estão sendo descritas. Ou seja, definem a URL, o tipo de protocolo (por exemplo SOAP ou XML-RPC) e protocolos de camadas inferiores (por exemplo, HTTP ou SMTP). Esses elementos são:

- *binding*: define uma especificação de protocolo e formato de dados para as operações e mensagens descritas em um *porttype*; aqui é que se define o protocolo de comunicação de alto nível que será usado entre um cliente e um servidor específicos;
- *port*: é um único endpoint, definido como uma combinação de um binding e um endereço de rede;
- *service*: é uma coleção de elementos ports relacionados. Cada elemento port se relaciona com um elemento binding particular, indicando qual interface e qual protocolo de comunicação estão sendo utilizados nessa implementação.

A Figura 5.3 mostra um exemplo de estrutura WSDL com os campos apresentados.

5.4.1.3 UDDI (*Universal Description Discovery and Integration*)

O UDDI (*Universal Description Discovery and Integration*) é um um protocolo para comunicação com registros de serviços. Ele permite a publicação, descoberta e integração de serviços, utilizando estruturas de dados em formato XML para a descrição

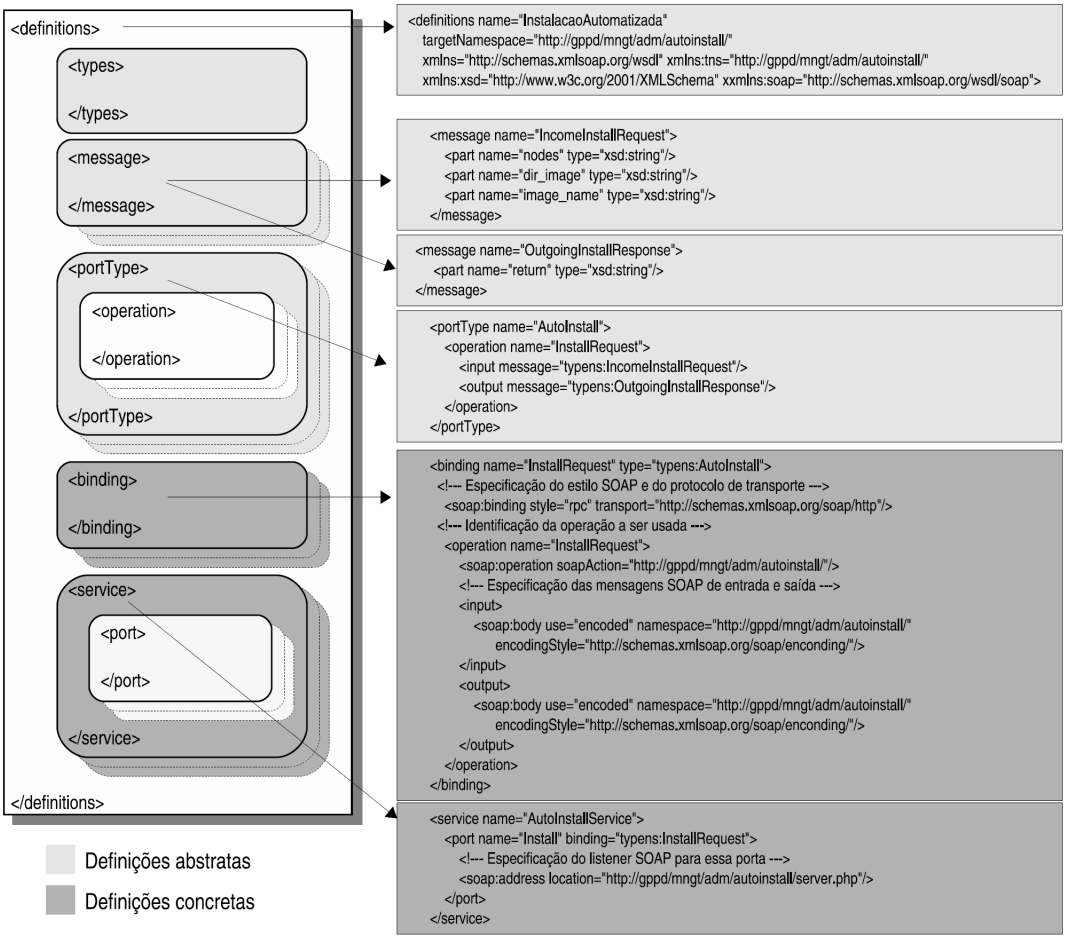

Figura 5.3: Mensagem WSDL (MARQUEZAN; SILVA CARISSIMI; NAVAUX, 2006)

e classificação de serviços, e uma API baseada no protocolo SOAP que permite acessar essas informações. A versão atual é o UDDI 3.0.2 (CLEMENT et al., 2004).

A organização da estrutura de informação são geralmente divididas na seguintes partes (MARQUEZAN; SILVA CARISSIMI; NAVAUX, 2006) (DELICATO, 2005):

páginas brancas (*businessEntity*) : contém informações a respeito da organização que publicou o serviço (fornecedor do serviço), como nome, endereço, web site, categoria, etc.;

páginas amarelas (*businessService*) : contém informações gerais sobre o serviço oferecido, como nome, descrição, categoria, etc.;

páginas verdes (*bindingTemplate*) : contém as informações técnicas do serviço, incluindo o endereço e forma de acessar o serviço.

A estrutura UDDI não é restrita a aplicações de web services, podendo ser usado para descrever qualquer tipo de serviço, como CORBA e Java RMI (MARQUEZAN; SILVA CARISSIMI; NAVAUX, 2006).

6 CONSIDERAÇÕES FINAIS

A computação pervasiva é um novo paradigma computacional que está sendo objeto de várias pesquisas científicas atualmente, tanto no meio acadêmico como na iniciativa privada. Suas premissas pressupõem forte integração com o mundo real, provendo uma elevada transparência às características da infraestrutura e tendo como foco o usuário e suas atividades. Na computação pervasiva, os sistemas são largamente distribuídos, mas suas características transcendem as existentes na maioria dos sistemas atuais. O desenvolvimento de aplicativos nesse cenário, necessita uma infra-estrutura de software que abstraia as características intrínsecas do sistema, de modo que a participação humana neste processo seja mínimo. Uma forma adequada de atingir este nível de transparência é através do uso de middlewares (SAHA; MUKHERJEE, 2003).

Os sistemas distribuídos modernos são constituídos por inúmeros dispositivos de vários tipos diferentes, caracterizando uma elevada heterogeneidade de hardware e software básico. Integrar estes sistemas de forma adequada, tornando a interação entre dispositivos menos penosa é atingida através da utilização de middlewares. O middleware promove uma abstração da infraestrutura, facilitando o desenvolvimento, manutenção e execução de aplicações neste cenário.

O avanço da tecnologia na área da eletrônica e dos microsistemas eletromecânicos, juntamente com a demanda por mobilidade na computação móvel, impulsionaram a evolução das Redes de Sensores Sem Fio (RUIZ, 2003) (LOUREIRO et al., 2003). As RSSF são caracterizadas por possuirem um número muito elevado de dispositivos sensores, geralmente de pequenas dimensões, dotados de comunicação sem fio e autônomos em energia. Cada sensor é capaz de captar informações do ambiente, tratar e enviar estas informações para as aplicações interessadas.

As RSSF são um tipo particular de sistema distribuído e assim como estes, necessita de uma infraestrutura de software para facilitar o desenvolvimento, manutenção e execução das aplicações que utilizam as informações sensoradas. Os middlewares em RSSF devem gerenciar os aspectos pertinentes aos sistemas distribuídos móveis e dinâmicos além daques específicos das redes de sensores, como o gerenciamento eficiente da energia dos nodos sensores, robustez e escalabilidade (LOUREIRO et al., 2003). O middleware deve prover uma interface de alto nível para a consulta as informações da rede e gerenciar aspectos referentes ao tempo e localização dos fenômenos monitorados.

As infraestruturas de software para sistemas distribuídos modernos têm apontado para o uso de Arquitetura Orientada a Serviços como forma de prover interoperabilidade ao sistema. O paradigma SOA possibilita a utilização de recursos de software disponibilizados em máquinas remotas através da chamada de serviços. A troca de informações

entre os fornecedores e consumidores de serviços utilizam protocolos abertos geralmente utilizando o padrão XML. Esta padronização permite a integração de aplicações independentemente do modelo de programação e plataforma utilizada.

REFERÊNCIAS

AURA. **Project Aura Distraction-free Ubiquitous Computing**. http://www.cs.cmu.edu/ aura/.

BARBOSA, T. M. G. A.; JR., I. G. S.; CARVALHO, H. S.; ROCHA, A. F. da; O. NASCIMENTO, F. A. de. Arquitetura de Software para Redes de Sensores sem Fios: a Proeminência do Middleware. **Anais do XXV Congresso da Sociedade Brasileira de Computação**, São Leopoldo/RS, julho 2005.

BONNET, P.; GEHRKE, J.; SESHADRI, P. Towards Sensor Database Systems. **Lecture Notes in Computer Science**, [S.l.], v.1987, 2001.

BOOTH, D.; HAAS, H.; MCCABE, F.; NEWCOMER, E.; CHAMPION, M.; FERRIS, C.; ORCHARD, D. **Web Services Architecture**. Acessado em 11/2007, World Wide Web Consortium, Note NOTE-ws-arch-20040211.

CAPRA, L.; EMMERICH, W.; MASCOLO, C. Reflective Middleware Solutions for Context-Aware Applications. **Lecture Notes in Computer Science**, [S.l.], 2001.

CARVALHO, A. C. P. de Leon F. de; BRAYNER, A.; LOUREIRO, A.; FURTADO, A. L.; STAA, A. von; LUCENA, C. J. P. de; SOUZA, C. S. de; MEDEIROS, C. M. B.; LUCCHESI, C. L.; SILVA, E. S. e; WAGNER, F. R.; SIMON, I.; WAINER, J.; MALDONADO, J. C.; OLIVEIRA, J. P. M. de; RIBEIRO, L.; VELHO, L.; GONçALVES, M. A.; BARANAUSKAS, M. C. C.; MATTOSO, M.; ZIVIANI, N.; NAVAUX, P. O. A.; SILVA TORRES, R. da; ALMEIDA, V. A. F.; JR., W. M.; KOHAYAKAWA, Y. **Grandes Desafios da Pesquisa em Computação no Brasil - 2006 - 2016**. Relatório sobre o Seminário realizado em 8 e 9 de maio de 2006.

CHINNICI, R.; MOREAU, J.-J.; RYMAN, A.; WEERAWARANA, S. **Web Services Description Language (WSDL) Version 2.0 Part 1**: Core Language. http://www.w3.org/TR/2007/REC-wsdl20-20070626, World Wide Web Consortium, Recommendation REC-wsdl20-20070626.

CHRISTENSEN, E.; CURBERA, F.; MEREDITH, G.; WEERAWARANA, S. **Web Services Description Language (WSDL) 1.1**. http://www.w3.org/TR/wsdl, World Wide Web Consortium (W3C).

CLEMENT, L.; HATELY, A.; RIEGEN, C. von; ROGERS, T. **UDDI Version 3.0.2**. http://uddi.org/pubs/uddi-v3.0.2-20041019.htm, Organization for the Advancement of Structured Information Standards, UDDI Spec Technical Committee Draft.

CURINO, C.; GIANI, M.; GIORGETTA, M.; GIUSTI, A.; MURPHY, A. L.; PICCO, G. P. TinyLIME: bridging mobile and sensor networks through middleware. In: PERVASIVE COMPUTING AND COMMUNICATIONS, 2005. **Anais...** IEEE Computer Society, 2005. p.61–72.

DELICATO, F. C. **Middleware Orientado a Serviços para Redes de Sensores sem Fio**. 2005. Tese (Doutorado em Ciência da Computação) — Universidade Federal do Rio de Janeiro.

ENDEAVOUR. **Endeavour Project**. http://endeavour.cs.berkeley.edu/.

FERNANDES, A. P. **Reflexão computacional**. http://attila.urcamp.tche.br / acauan/art_ccei_rc.html.

GAIA. **Gaia Project**. http://choices.cs.uiuc.edu/ActiveSpaces/.

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, Apr-Jun 2002.

GRIMM, R. One.world: experiences with a pervasive computing architecture. **Pervasive Computing, IEEE**, [S.l.], v.3, n.3, p.22–30, July-Sept. 2004.

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.

HEINZELMAN, W. B.; MURPHY, A. L.; CARVALHO, H. S.; PERILLO, M. A. Middleware to Support Sensor Network Applications. **IEEE Network**, [S.l.], v.18, n.1, p.6–14, 2004.

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

HENRICKSEN, K.; ROBINSON, R. A survey of middleware for sensor networks: state-of-the-art and future directions. In: MIDSENS '06: PROCEEDINGS OF THE INTERNATIONAL WORKSHOP ON MIDDLEWARE FOR SENSOR NETWORKS, 2006, New York, NY, USA. **Anais...** ACM Press, 2006. p.60–65.

HOLO. **Holoparadigma**. http://www.inf.unisinos.br/ holo/.

HW-GROUP. **HW Group**. http://www.hw-group.com/.

ISAM. **Infra-estrutura de Suporte às Aplicações Móveis**. http://www.inf.ufrgs.br/ isam/.

ISAMADAPT. **Explorando Comportamento Sensível ao Contexto em Ambientes da Pervasive Computing**. http://www.inf.ufrgs.br/ isam/IsamAdapt/.

LOUREIRO, A. A. F.; NOGUEIRA, J. M. S.; RUIZ, L. B.; FREITAS MINI, R. A. de; NAKAMURA, E. F.; FIGUEIREDO, C. M. S. Redes de Sensores Sem Fio. In: **XXI Simpósio Brasileiro de Redes de Computadores**. [S.l.: s.n.], 2003. p.179–226.

LOUREIRO, A. A. F.; RUIZ, L. B.; FRANSISCANY, F. P.; COUTO, R. R. P.; NOGUEIRA, J. M. S. **Middlewares para redes de sensores sem fio. (tutorial)**.

MACEDO, D.; CORREIA, L.; FIGUEIREDO, C. M.; NAKAMURA, E.; RUIZ, L.; MINI, R.; CâMARA, D.; MAIA, E.; LOUREIRO, A. A. F.; NOGUEIRA, J. M.; VIEIRA, L. F.; VIEIRA, M. A.; SILVA, D. C. da. **Arquitetura para Redes de Sensores Sem Fio**. 22° Simpósio Brasileiro de Redes de Computadores.

MACIEL, R. S. P.; ASSIS, S. R. de. Middleware: uma solução para o desenvolvimento de aplicações distribuídas. **CienteFico, Ano IV, v. I**, Salvador,, janeiro-junho 2004.

MACKENZIE, C. M.; LASKEY, K.; MCCABE, F.; BROWN, P. F.; METZ, R. **Reference Model for Service Oriented Architecture 1.0**. [S.l.]: Organization for the Advancement of Structured Information Standards, 2006.

MADDEN, S.; FRANKLIN, M. J.; HELLERSTEIN, J. M.; HONG, W. TinyDB: an acquisitional query processing system for sensor networks. **ACM Trans. Database Syst**, [S.l.], v.30, n.1, p.122–173, 2005.

MARQUEZAN, C. C.; SILVA CARISSIMI, A. da; NAVAUX, P. O. A. **Web Services para Computação de Alto Desempenho**. VII Workshop em Sistemas Computacionais de Alto Desempenho.

MITRA, N.; LAFON, Y. **SOAP Version 1.2 Part 0**: Primer. http://www.w3.org/TR/2007/REC-soap12-part0-20070427/, World Wide Web Consortium, Recommendation REC-soap12-part0-20030624.

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

PEREIRA, M. R.; AMORIM, C. L. de; CASTRO, M. C. S. de. Tutorial sobre Redes de Sensores. **Cadernos do IME - Série Informática**, [S.l.], v.14, Junho 2003.

PIETZUCH, P. R. **Hermes**: A scalable event-based middleware. [S.l.]: University of Cambridge, 2004.

PROJECT one.world. **one.world**. http://cs.nyu.edu/rgrimm/one.world/.

ROMAN M.; HESS, C. C. R. R. A. C. R. N. K. A middleware infrastructure for active spaces. **Pervasive Computing, IEEE**, [S.l.], v.1, n.4, p.74–83, Oct-Dec 2002.

RUIZ, L. B. **MANÁ**: Uma Arquitetura para Gerenciamento de Redes de Sensores Sem Fio. 2003. Tese (Doutorado em Ciência da Computação) — Universidade Federal de Minas Gerais.

RUIZ, L. B.; CORREIA, L. H. A.; VIEIRA1, L. F. M.; MACEDO, D. F.; NAKAMURA, E. F.; FIGUEIREDO, C. M. S.; VIEIRA, M. A. M.; BECHELANE, E. H.; CAMARA, D.; LOUREIRO, A. A.; NOGUEIRA, J. M. S.; SILVA JR., D. C. da; FERNANDES, A. O. Arquiteturas para Redes de Sensores Sem Fio. In: **XXII Simpósio Brasileiro de Redes de Computadores**. [S.l.: s.n.], 2004. p.167–218.

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

SENSOR-WEBS. **JPL Sensor Webs**. http://sensorweb.jpl.nasa.gov/, NASA.

SHEN, C.-C.; SRISATHAPORNPHAT, C.; JAIKAEO, C. Sensor information networking architecture and applications. **Personal Communications, IEEE**, [S.l.], v.8, p.52–59, 2001.

SHEN, X.; WANG, Z.; SUN, Y. Wireless sensor networks for industrial applications. In: FIFTH WORLD CONGRESS ON INTELLIGENT CONTROL AND AUTOMATION, 2004. **Anais...** [S.l.: s.n.], 2004. v.4, p.3636–3640.

SMART-DUST. **SMART DUST**: Autonomous sensing and communication in a cubic millimeter. http://robotics.eecs.berkeley.edu/ pister/SmartDust/, Berkeley.

SOCORRO VâNIA L. ALVES ENOQUE CALVINO ALVES, R. S. M. B. XMOM - Um Middleware Orientado a Mensagens. **I Jornada Científica da UNIBRATEC**, [S.l.], 2006.

SOUTO, E.; GUIMARãES, G.; VASCONCELOS, G.; VIEIRA, M.; ROSA, N. S.; FERRAZ, C. A. G.; KELNER, J. Mires: a publish/subscribe middleware for sensor networks. **Personal and Ubiquitous Computing**, [S.l.], v.10, n.1, p.37–44, 2006.

WEISER, M. The Computer for the Twenty-First 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.