UNIVERSIDADE FEDERAL DE SÃO CARLOS
DEPARTAMENTO DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ORDENAÇÃO DE EVENTOS E INTERPRETAÇÃO DE CONTEXTOS EM REDES DE ATUADORES E SENSORES SEM FIO PARA A CLASSE DE APLICAÇÕES DE PREPARAÇÃO PARA EMERGÊNCIA

Fernando Henrique dos Santos e Silva

São Carlos – SP
Junho/2006

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

S586oe

Silva, Fernando Henrique dos Santos e.
Ordenação de eventos e interpretação de contextos em redes de atuadores e sensores sem fio para a classe de aplicações de preparação para emergência / Fernando Henrique dos Santos e Silva. -- São Carlos : UFSCar, 2007.
76 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2006.

1. Redes de computação. 2. Redes de sensores sem fio. 3. Ordenação de eventos. I. Título.

CDD: 004.6 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

"Ordenação de Eventos e Interpretação de Contextos em Redes de Atuadores e Sensores Sem Fio para a Classe de Aplicações de Preparação para Emergência"

FERNANDO HENRIQUE DOS SANTOS E SILVA

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Membros da Banca:

Profa. Dra. Regina Borges de Araujo
(Orientadora – DC/UFSCar)

Prof. Dr. Hélio Crestana Guardia
(DC/UFSCar)

Prof. Dr. Edmundo Roberto Mauro Madeira
(UNICAMP)

São Carlos
Junho/2006

Dedico este trabalho:

Aos meus pais, Victor Armando dos Santos e Silva e
Leni Caetano dos Santos e Silva que sempre me
incentivaram e se esforçaram para me oferecer a
oportunidade de chegar até aqui.
À Patrícia, minha namorada, pelo carinho e paciência nos
momentos difíceis.

Agradecimentos

Primeiro, gostaria de agradecer à minha orientadora, profª Drª Regina Borges de Araújo, pela motivação e direção dos meus estudos para escrever esta dissertação e os artigos que publicamos juntos.

À minha família que sempre me apoiou.

À todos meus colegas do LRV*Net*, Altieres, Diego, Néstor, Ricardo e Leandro, pelas discussões que contribuíram para a realização desta dissertação. E também, ao Rodrigo que morou comigo aqui em São Carlos.

Às funcionárias do DC, Cristina e Mirian, pelos favores concedidos.

Resumo

Redes de Atuadores e Sensores sem fios (RASSFs) estão cada vez mais sendo utilizadas para o monitoramento mais preciso de ambientes sujeito a situações de emergência, como fogo, vazamento de gases tóxicos e explosões. Nestes ambientes, é muito importante uma interpretação confiável dos eventos capturados. No entanto, a ordem em que os eventos são recebidos pode interferir diretamente na correta interpretação do que está acontecendo no ambiente físico monitorado. Portanto, a ordenação de eventos em RASSF pode ser um requisito importante para aplicações de monitoramento de ambientes sujeitos a situações de emergência porque, através da ordenação de eventos, ambigüidades podem ser tratadas, aumentando a precisão do que esta sendo monitorado. Este trabalho descreve um novo algoritmo para ordenação de eventos chamado OBC (*Ordering by Confirmation*). Este algoritmo ciente de energia e de baixa latência foi discutido e avaliado em relação a soluções existentes, mostrando que pode economizar mais energia e atingir latências mais baixas em situações onde a densidade da rede, a taxa de eventos e a dimensão da rede aumentam. Este trabalho também apresenta uma solução de interpretação de contextos para redes de atuadores e sensores sem fios que pode interpretar desde contextos simples até complexos. Nessa solução, atuadores são usados para agregar eventos dos sensores, eliminar ambigüidade e redundância e realizar a interpretação de contexto. Cada atuador da RASSF é configurado com regras determinadas pela aplicação na fase de configuração da rede. Após serem ordenados, os eventos correlacionados são interpretados resultando em contextos que vão desde simples consultas, como: “Qual é a temperatura no setor A ?”, até mais complexas como :“Por que a asa da aeronave rachou ?”. Resultados de simulações mostraram que o uso dos atuadores para a interpretação de contexto descentralizada, e também para a ordenação de eventos de forma rápida, ciente de energia podem ser uma boa alternativa para aplicações de preparação para emergências onde vidas e patrimônios estão em risco.

Abstract

Wireless Sensor and Actor Networks (WSANs) are increasingly being deployed for fine-grain monitoring of physical environments subjected to emergency situations such as fire, leaking of toxic gases and explosions. In these environments a reliable interpretation of the collected events is very important. However, the order in which events are received can interfere directly with the correct interpretation of what is going on in the physical environment being monitored. Therefore, the ordering of events in WSAN can be a major requirement for critical security monitoring applications because by handling event ordering, ambiguities can be dealt with, increasing the accuracy of what is being monitored. This work describes a novel algorithm for event ordering in WSAN called OBC (Ordering by Confirmation). This low latency and energy aware protocol was discussed and evaluated against existing solutions, showing that it can save more energy and achieve lower latencies in situations where the network density, event rate and network dimension increase. This work also presents a context interpretation solution that can interpret from simple to complex contexts. In this solution, actors are used to aggregate sensor events, eliminate ambiguities and redundancy and realize context interpretations. Each actor of the WSAN is configured with rules that are determined by the application in the network configuration phase. After ordered, correlated events are interpreted resulting in contexts that go from simple questions, such as "What is the temperature of sector A?" to more complex ones "Why did the aircraft wing crack?". Context information is exchanged among actors and sink(s) of the WSAN through the publish/subscribe technique based on topics. Our solution was compared to a centralized solution, in which events are collected by the sink. Results from simulations have shown that by having actors for decentralized context interpretation and a fast energy-aware event ordering algorithm, our solution can be a good alternative for emergency preparedness applications where lives and patrimony are at risk.

Sumário

Índice de Figuras

Índice de Tabelas

Lista de Abreviaturas e Siglas

ADC:	**Analog to Digital Converter**
CBR:	**Constant Bit Rate**
CSMA:	**Carrier Sense Multiple Access**
DAC:	**Digital to Analog Converter**
FIFO:	**First In First Out**
MAC:	**Medium Access Control**
NS-2	**Network Simulator**
OBC:	**Order By Confirmation**
RASSF:	**Rede de Atuadores e Sensores sem Fios**
RNF:	**Requisito Não Funcional**
RSSF:	**Rede de Sensores sem Fios**
TCP:	**Transmission Control Protocol**
TMOS:	**Temporal Message Ordering in Sensor Networks**
VBR:	**Variable Bit Rate**
XML:	**Extensible Markup Language**

1. Introdução

Recentes avanços tecnológicos permitiram o desenvolvimento de pequenos dispositivos economicamente viáveis, de baixo consumo de energia e com capacidades de sensoriamento e de comunicação sem fio. Da organização destes pequenos dispositivos em uma arquitetura computacional e de comunicação, surgiu um novo tipo rede sem fio, denominada de rede de sensores sem fios (RSSF), que tem atraído muitos pesquisadores em virtude do seu vasto campo de aplicação [EST 99] [CER 01] [AKY 02]. A RSSF permite monitorar remotamente um ambiente e obter informações mais precisas do que as técnicas de instrumentação tradicionais, pois possui a vantagem de ser empregada bem próxima ao fenômeno de interesse [PIN 04].

A inserção de novos elementos chamados de atuadores fornece vantagens adicionais à RSSF, possibilitando a reação automática, e em tempo real, aos eventos coletados pelos nós sensores. Com isso, baixamos a latência na tomada de decisões, o que é muito útil em aplicações de preparação para emergências e aplicações que, além desta, demandem baixa latência. Estas redes são denominadas redes de atuadores e sensores sem fios (RASSF).

O monitoramento de ambientes sujeitos a situações de emergência é umas das classes de aplicações cientes de contexto de maior apelo em RASSF. Nesses ambientes, contextos podem ser capturados e interpretados na camada de aplicação das RASSF, auxiliando na prevenção, combate e avaliação de situações de fogo, explosão, vazamento de gás ou substancias tóxicas, etc. Em uma condição de incêndio, por exemplo, é muito importante manter um monitoramento confiável e preciso do ambiente físico para que seja possível às equipes de resgate entender o que está acontecendo e tomar as melhores medidas emergenciais sobre o ambiente. Entretanto, para que se possa compreender o que esta ocorrendo no ambiente e tomar as medidas certas, é necessário interpretar os eventos capturados corretamente.

Em rede de atuadores e sensores sem fios, os sensores coletam informações do ambiente físico, enquanto os atuadores processam estas informações para tomar decisões e realizar ações sobre o ambiente [AKY 04]. Os atuadores possuem um custo mais alto, melhores capacidades de processamento, bateria e de comunicação sem fio. Portanto, os

atuadores são candidatos apropriados para a interpretação de contextos complexos em aplicações de preparação para emergências.

Contexto é definido neste trabalho como qualquer informação sobre o ambiente físico que é importante para a aplicação. Contextos podem ser simples quando são compostos por eventos simples, como a temperatura atual, pressão do ar atual, presença de fumaça, etc., ou mais complexos quando compostos por dois ou mais eventos correlacionados, como detecção de incêndio (uma combinação de eventos correlacionados: "alta temperatura" e "presença de monóxido de carbono"). Quando a relação entre os eventos é temporal, isto é, a ordem no tempo em que cada evento ocorre é importante, contextos tão complexos como "Porque certa situação aconteceu no ambiente?" poderiam ser respondidos através de uma interpretação mais complexa que demanda uma ordenação dos eventos. Como um exemplo, uma RASSF poderia interpretar os seguintes contextos: porque a asa da aeronave quebrou? A pressão estava acima do normal (pressão acima do normal), o que provocou uma rachadura na superfície da asa (presença de rachadura) levando asa se partir – isto pode ser deduzido à partir da ordenação dos eventos: A alta pressão ocorreu antes da rachadura surgir.

Este trabalho apresenta uma solução de interpretação de contextos para redes de atuadores e sensores sem fios para interpretação de contextos simples até contextos complexos. Nessa solução, atuadores são usados para agregar eventos dos sensores, eliminar ambigüidade e redundância e realizar a interpretação de contexto de forma descentralizada. A interpretação de contexto pode ser utilizada pelos atuadores para engatinhar atuações no ambiente físico, como o acionamento de *sprinklers* (dispersores de água para extinção de incêndios), elevação automática da temperatura ambiental, etc.

Em [LAM 78] é dito que o tempo é fundamental para a nossa forma de pensar e entender determinadas situações, implicando, dessa forma, diretamente na correta interpretação do estado do ambiente. Logo, a ordenação dos eventos torna-se um requisito fundamental quando desejamos garantir a corretude das informações interpretadas. Este trabalho descreve um algoritmo para ordenação de eventos coletados de uma rede de atuadores e sensores sem fios. O algoritmo descrito aqui é ciente de energia e mais rápido que outros algoritmos encontrados na literatura.

1.1. Motivação e Objetivos

A motivação para este trabalho surgiu a partir da avaliação de alguns sistemas existentes usados por equipes de combate ao fogo, para monitoramento de situações de emergência.

Estes sistemas apresentam limitações, principalmente em termos de funcionalidade e interface de acesso, que poderiam ser superadas com as novas tecnologias de redes de atuadores e sensores sem fio e de software ciente de contexto.

Este trabalho objetiva apresentar, desenvolver e avaliar uma solução de interpretação de contexto e um novo algoritmo de ordenação de eventos que atendam as necessidades de aplicações de monitoramento de condições críticas. A solução de interpretação de contexto já foi abordada em trabalhos anteriores a este, pertencentes ao projeto geral do laboratório LRVNet, onde alguns esquemas já foram definidos, como: a comunicação de troca de informação de contexto através do mecanismo *Publish/Subscribe*; a interface *XML (Extensible Markup Language)* para comunicação entre a aplicação e o serviço responsável pela interpretação de contexto; a interface de configuração na qual o responsável de segurança define os critérios de monitoramento; além de outras. Portanto, cabe a este trabalho, quando referente à interpretação de contexto, apresentar novas considerações para tratar a interpretação de contexto em um tipo de rede de sensores (RASSF) mais apropriada para o monitoramento de situações de emergência, definindo a entidade responsável pela interpretação de contexto, bem como a forma como os dados serão coletados para a interpretação.

O desenvolvimento de um algoritmo de ordenação de eventos é um objetivo muito importante deste trabalho, visto que implica na correta interpretação de contexto. Ele está sendo abordado pela primeira vez no projeto geral de preparação para emergências, tendo como metas ser ciente de energia e de baixa latência.

1.2. Organização da Dissertação

Este trabalho está organizado da seguinte forma: O capítulo 1 faz a introdução desta dissertação, apresentando o tema de pesquisa. O capítulo 2 apresenta o conceito das aplicações de monitoramento de ambientes sujeitos a situações de emergência. Para tanto, são discutidos os requisitos e as necessidades a serem atendidos por estas aplicações. O capítulo 3 trata das tecnologias de RSSF e RASSF, destacando suas características, requisitos e seus modos de funcionamento. O capítulo 4 apresenta questões relacionadas diretamente com os objetivos deste trabalho, isto é, a interpretação de contexto e a ordenação de eventos. No capítulo 5 é apresentada a solução de interpretação de contexto, bem como o algoritmo de ordenação de eventos (OBC), incluindo suas descrições, simulações e avaliações de

desempenho. Finalmente, o capítulo 6 apresenta as conclusões deste trabalho, as limitações encontradas, e as propostas futuras, seguido de Referências Bibliográficas.

2. Monitoramento de Ambientes Sujeitos a Situações de Emergência.

Em ambientes de segurança crítica ou sujeitos a situações de emergência existe a constante preocupação com incidentes de natureza emergencial (por exemplo, incêndios e explosões) que possam ocorrer e ocasionar a perda de vidas e/ou de patrimônio. Aplicações voltadas para estes ambientes têm como objetivo prevenir e/ou minimizar essas ocorrências [KNI 02]. Destacam-se como exemplo de ambientes de segurança crítica: espaço aéreo de um aeroporto, espaço interno de aeronaves em ensaio de solo ou vôo, usina nuclear, presídios, plantas industriais que contenham elementos altamente tóxicos e perigosos, etc.

Neste trabalho, o sistema de monitoramento de ambientes sujeitos a situações de emergência explora as facilidades da computação ciente de contexto, no qual os contextos são obtidos via sensores dispersos nos ambientes.

O monitoramento de situações críticas em ambientes físicos apresenta algumas características próprias:

(a) necessidade de captura precisa de dados dos nós sensores;
(b) necessidade de monitoramento constante e consistente do ambiente;
(c) necessidade de notificação imediata das mudanças que ocorrem no ambiente, mesmo na presença de falhas de nós sensores;
(d) necessidade de atuação rápida e precisa sobre os eventos.

O ambiente físico monitorado (por exemplo, prédio, fábrica) pode possuir sensores especializados, tais como aqueles para medição de temperatura, pressão, umidade, presença de substâncias tóxicas, presença de fumaça, e outros sensores que possam capturar alterações nas condições físicas do ambiente de modo a monitorá-lo e auxiliar no combate em uma situação de emergência.

Durante a fase de monitoramento, informações como temperatura, pressão e presença de fumaça podem ser capturadas e interpretadas para verificar se indicam qualquer grau de probabilidade de alguma situação de emergência ser iniciada.

Se um incêndio vier a acontecer, no momento do combate ao incêndio, por exemplo, dados do ambiente e a localização de pessoas poderão ser capturados para verificar se alguém corre perigo de morte. Todas essas informações combinadas poderão servir para planejar estratégias de combate e salvamento. Neste caso, as informações do ambiente podem ser visualizadas de diferentes tipos de dispositivos e por múltiplos usuários, incluindo: (a) os bombeiros para traçar estratégias de combate e salvamento, (b) o engenheiro de segurança da empresa para verificar se existe alguma prioridade de salvamento, (c) a seguradora da empresa para verificar a ocorrência dos fatos, bem como as suas causas, etc., em situações de perícia.

Além de auxiliar no combate ao incêndio em tempo real, todas as informações capturadas do mundo real, podem ser armazenadas em um repositório e serem acessadas posteriormente, para por exemplo, possibilitar assistir a um combate a incêndio que ocorreu em um determinado dia e local para efeito de treinamento, ou para avaliação de desempenho individual e/ou coletivo da equipe.

2.1. Aplicações de Preparação para Emergências

Existem algumas aplicações de RSSF que objetivam monitorar as condições do ambiente e auxiliar nas decisões que precisam ser tomadas de maneira rápida e eficaz quando situações anormais são detectadas. Exemplos de alguns projetos incluem o *Siren* [CHE 04]**,** Posto de Comando do Futuro (*The Command Post of the Future*) [DAR 04], e o FIRE (*Fire Information and Rescue Equipment*) [WRI 04].

O *Siren* é um sistema ponto-a-ponto, com arquitetura de computação ciente de contexto capaz de coletar, agregar e distribuir contextos em cenários de incêndio. Ele auxilia bombeiros em operações de combate a incêndio. O *Siren* é capaz de tratar apenas contextos de localização relativa, de monitoramento da temperatura do ambiente e do nível de oxigênio. São gerados apenas cinco tipos de mensagens de alerta: “lugar perigoso”, “perigo para si mesmo”, “próximo a um lugar perigoso”, “outros em perigo” e “instruções”. Essas informações são apresentadas no *PDA* de cada bombeiro com o objetivo de manter uma comunicação entre eles.

O Posto de Comando do Futuro consiste em um conjunto de projetos de investigação para situações de campo de batalha. Esses projetos focalizam o desenvolvimento de tecnologias avançadas que possibilitem a criação de *displays* interativos do ambiente, integrados com múltiplos sensores para o comandante aperfeiçoar suas decisões de forma

rápida e com qualidade. Esses *displays* podem auxiliar em tomadas de decisões facilitando a interação de várias formas, a visualização de informações do ambiente e a formação de estratégias baseadas em conhecimento.

O FIRE é um projeto desenvolvido pelo Instituto de Manufatura de Berkeley, situado na Universidade da Califórnia, juntamente com Departamento de Corpo de Bombeiros de Chicago. Destina-se ao auxílio no combate de situações de incêndio, fornecendo informações importantes para operações de resgate. Este projeto possui as seguintes características:

- uso de uma RSSF;
- gerenciamento da situação pelo comandante, através da visualização bidimensional do ambiente. Figura 2.1(a);
- sensores para monitoramento dos sinais vitais dos bombeiros, Figura 2.1(b);
- visor acoplado a mascara do bombeiro que possibilita fácil visualização de informações sensoriadas. Figura 2.1(c).

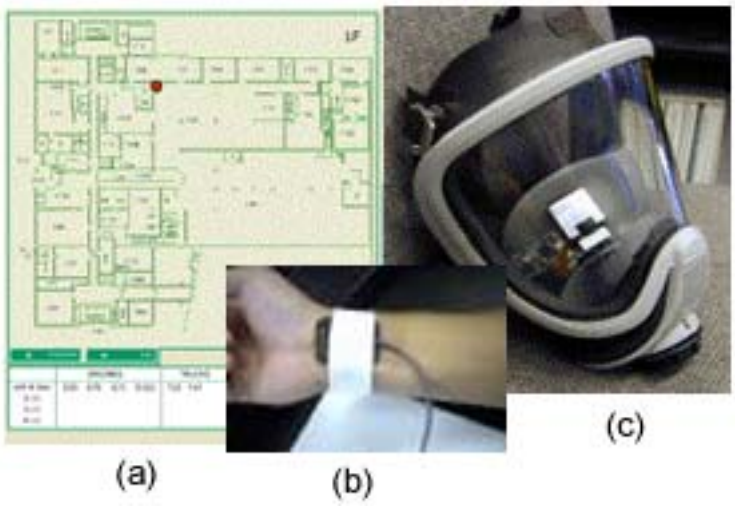

Figura 2.1- Imagens do projeto FIRE: (a) visualização bidimensional, (b) captura de sinais vitais e (c) visor acoplado a máscara.

Tanto o projeto Siren quanto o FIRE utilizam dados de tempo real para auxiliar bombeiros no combate a incêndios. Logo, não se preocupam com a persistência dos dados para acesso posterior, que podem ser muito úteis para a realização de perícia e treinamento. Além disso, a utilização das redes de sensores e de um ambiente virtual 3D oferecem um monitoramento mais preciso, quando comparado com uma visualização bidimensional.

Outro ponto a ser observado é a necessidade, nestes projetos, de meios que permitam não só o resgate e a supervisão, mas também a utilização de atuadores operando com eficiência para controle dos ambientes emergenciais. Para a operação eficiente desses atuadores seria necessário que os sistemas possuíssem mecanismos de interpretação que pudessem interpretar com confiabilidade eventos capturados do ambiente físico.

Neste trabalho de mestrado buscou-se superar alguns limites dos ambientes acima descritos. Para isso, um algoritmo de ordenação de eventos e uma solução de interpretação de contextos na rede foram implementados. A próxima seção descreve os requisitos não funcionais para classe de aplicações de preparação para emergência e mostra como esses requisitos são atendidos com as soluções propostas neste trabalho, bem como no projeto geral.

2.2. Requisitos não funcionais para classe de aplicações de preparação para emergência

Esta seção destaca os requisitos não funcionais (RNFs) para aplicações de monitoramento de condições críticas. Trata-se de uma visão geral dos requisitos relevantes a este trabalho. Em [MIC 04] os RNFs são descritos mais detalhadamente quanto ao conceito, modelos e características.

Os requisitos não funcionais são limitações globais em um sistema que especificam atributos de qualidade (desempenho, portabilidade, custo, confiabilidade, segurança), [CHU 00]. Os RNFs são também conhecidos como atributos de qualidade [BOE 76] [KEL 90] [TUR 96], restrições [ROM 85] e objetivos [MOS 85]. Os RNFs não expressam nenhuma função a ser realizada pelo sistema, mas sim comportamentos e restrições que ele deve satisfazer, isto é, descrevem "como o sistema deve fazer?".

Em sistemas de monitoramento de condições críticas é necessário considerar requisitos não funcionais, tais como confiabilidade dos dados coletados pelos sensores, uma vez que o sistema durante condições críticas pode ser colocado em risco devido a informações incorretas dos sensores. Nós sensores podem originar informações incorretas devido a interferências, tais como: água, presença de fumaça densa, ou pelo mau funcionamento. Desta forma, sistemas de monitoramento de condições críticas devem atender aos seguintes RNFs:

- **Confiabilidade** – Em sistemas de monitoramento de condições críticas é indispensável a garantia de confiabilidade no sistema, em outras palavras, a certeza de que o sistema irá operar de acordo com o esperado. Mesmo na ocorrência de falha de algum nó da rede de sensores, ou falha durante a comunicação, os dados deverão estar corretos A integridade e a consistência dos dados também devem ser garantidas neste tipo de sistema. Pode-se especificar, por exemplo, que um serviço de tratamento de eventos não pode falhar quando o sistema está monitorando uma condição crítica de alta prioridade. Uma das

funcionalidades para garantir esta exigência poderia ser a aplicação de redundância no serviço [CHU 00].

- **Segurança** – Os sistemas de monitoramento de condições de emergência podem ser definidos sob o ponto de vista de Segurança ("*Safety*") quando a conseqüência de pelo menos uma falha temporal exceda em muito os benefícios normais do sistema podendo levar a uma catástrofe [FAR 00]. Estes são subdivididos em:

  - Sistema de tempo real crítico seguros em caso de falha ("*fail safe*"), onde um ou vários estados seguros podem ser atingidos em caso de falha (por exemplo, parada obrigatória de trens na ocorrência de falha do sistema de sinalização ferroviário);

  - Sistemas de tempo real crítico operacionais em caso de falha ("*fail operational*"), onde na presença de falhas parciais, podem se degradar fornecendo alguma forma de serviço mínimo (por exemplo, sistema de controle de vôo com comportamento degradado, mas ainda seguro).

Neste trabalho, a segurança é considerada crítica, pois qualquer falha no sistema seja de componentes de hardware, de software ou até mesmo na transmissão de informações inconsistentes para quem requisitou o serviço pode implicar em conseqüências desastrosas.

- **Eficiência e Desempenho** – Eficiência e desempenho são requisitos importantes em sistemas que precisam ter resposta em tempo-real. Estes envolvem também requisitos de **tempo de resposta** (tempo aceitável do ponto de vista do usuário para que alguma operação seja concluída); de **vazão** (*throughput* **-** quantidade de dados que precisam ser processados em um intervalo pré-definido de tempo) e de temporização. Logo, em RSSFs, medidas devem ser adotadas com o intuito de atender estes requisitos, como: algoritmos que forneçam um tempo de resposta aceitável e serviços que filtram o montante de eventos, possivelmente redundantes, diminuindo o tráfego na rede e aumentando o desempenho do sistema [TUR 96] [CHU 00].
- **Causalidade -** É conceituada como o conjunto de todas as relações que determinam "causa e efeito". Por exemplo, ao apertarmos o gatilho de um revólver carregado, o projétil sendo expulso pelo cano é o efeito e a causa é a força expansiva dos gases produzidos com a queima da pólvora. Logo, entender a relação das causas que ocasionaram o efeito é objetivo que almejamos Este requisito pode ser determinante para uma correta interpretação dos eventos capturados [MIC 04].

O presente trabalho objetivou a implementação de algoritmos que atendessem a estes requisitos não funcionais e aos requisitos das RSSFs (que são descrito na seção 3.3). A seção 5.4 detalha como cada um dos requisitos funcionais foram atendidos.

2.3. Projeto de Preparação para Emergências do Laboratório de Realidade Virtual em Rede (LRVNet)

O presente trabalho está inserido em um projeto de monitoramento preciso de situações de emergência que integra realidade virtual, rede de sensores e computação ubíqua. O projeto está sendo desenvolvido de uma colaboração entre o Laboratório de Realidade Virtual (LRVNet) do Departamento de Ciência da Computação na UFSCar e o Paradise Lab da Universidade de Ottawa.

O projeto, ilustrado na Figura 2.2, está organizado como segue: (1) Algoritmos para redes de sensores que atendam às necessidades da classe de aplicações de monitoramento, isto é, protocolos tolerantes a falhas, cientes de consumo de energia e etc.; (2) protocolos que atendam requisitos não funcionais para aplicações de monitoramento de condições de emergência e, (3) uma estrutura para acesso tanto em tempo real quanto posterior que permite uma visualização mais precisa das informações através de ambientes virtuais 3D que imitam o ambiente físico [BOU 05].

Este trabalho está inserido no item (2) , com o objetivo de desenvolver um algoritmo de ordenação e interpretação de eventos que atenda a requisitos não funcionais, em uma rede atuadores e sensores sem fios para aplicações de monitoramento de condições de críticas em ambientes físicos e lógicos cientes de contexto.

2.4. Considerações Finais

Este capítulo introduziu o conceito de monitoramento de ambientes sujeitos a situações de emergência, destacando os principais requisitos não funcionais de aplicações destinadas a este fim, e as medidas adotadas para atendê-los. Alguns projetos de monitoramento destes ambientes foram apresentados, destacando-se algumas limitações como: a persistência dos dados para perícia/treinamento; o monitoramento não muito preciso; a ausência de atuadores e de mecanismos de ordenação e interpretação de eventos. O projeto geral também é apresentado, explicando cada parte em que ele é dividido e onde este trabalho se encaixa.

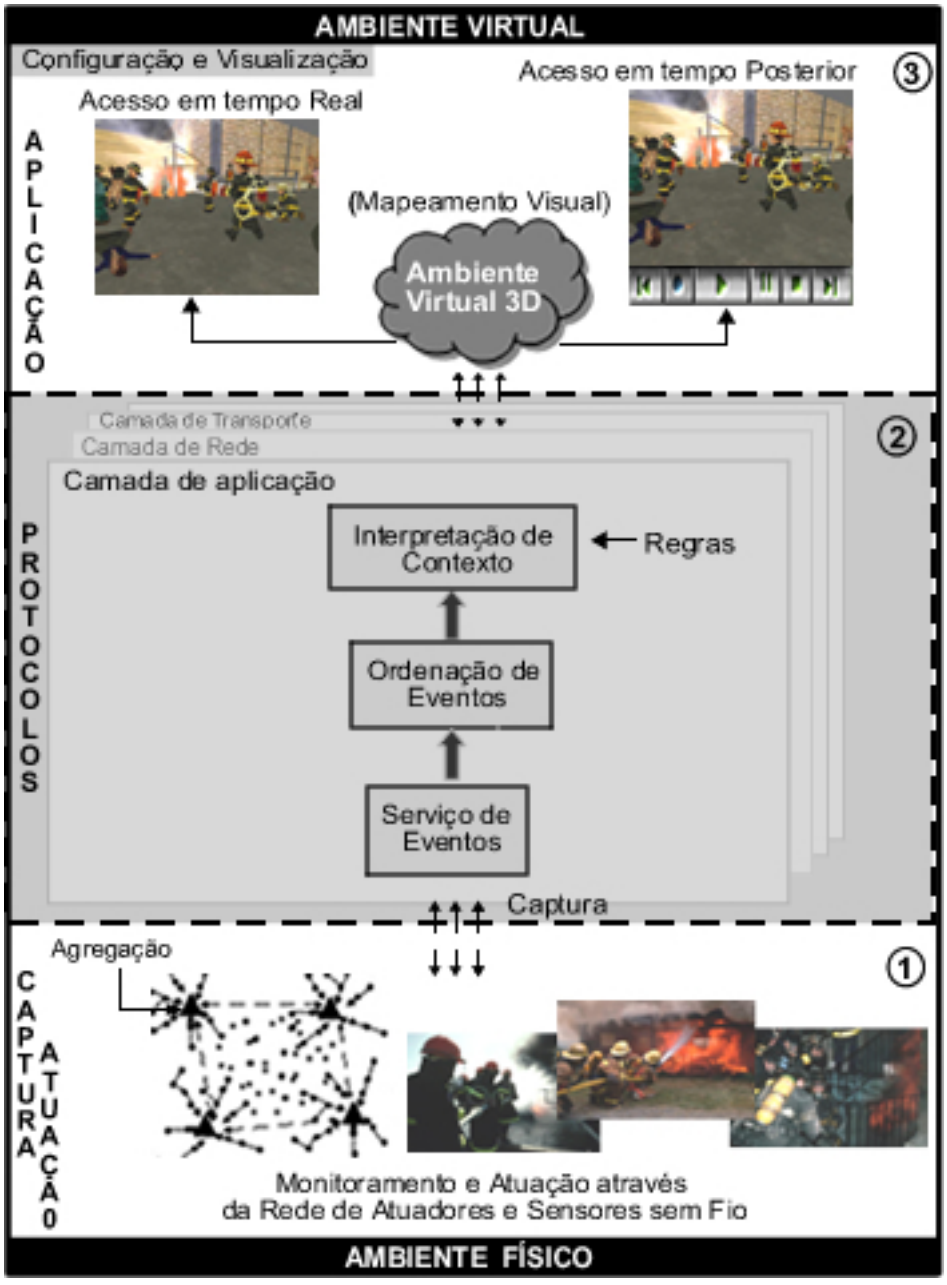

Figura 2.2. Visão geral do projeto de Preparação para Emergências do LRVNet.

3. Rede de Sensores sem Fio e Rede de Atuadores e Sensores sem Fio

Uma rede de sensores (RSSFs) é composta por um grande número de pequenos nós sensores que são colocados dentro ou muito próximo do fenômeno a ser analisado [AKY 02]. Geralmente, cada nó sensor é equipado com vários tipos de sensores (por exemplo, temperatura, pressão, sísmico, acústico, radiação, infravermelho, além de outros). A Figura 3.1 ilustra alguns tipos de sensores. Estes nós são construídos de hardware extremamente barato e com limitações computacionais, de memória, de comunicação e de energia.

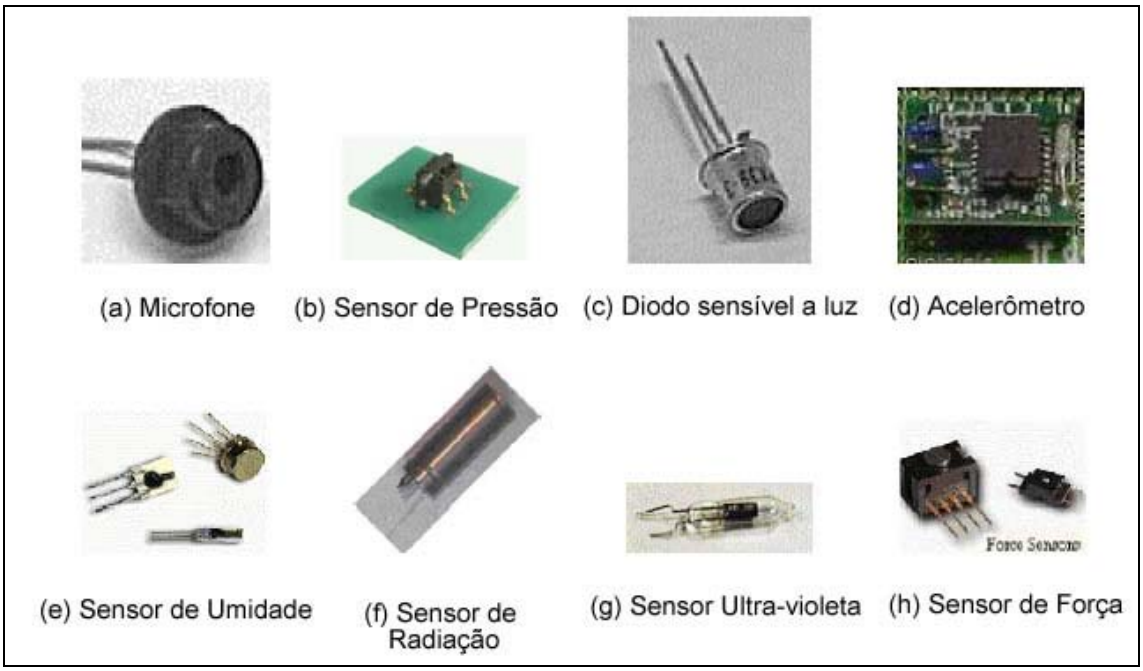

Figura 3.1. Tipos de Sensores

As RSSFs são a chave da coleta de informações necessárias em ambientes onde o uso de fios ou cabeamento não seja possível ou viável. Elas podem estar inseridas na estrutura de um prédio, ponte, no interior de máquinas, tubulações, dentro de casa, florestas, áreas de desastre, vulcões, plantações, campo de batalha e até mesmo dentro da retina do olho humano [END 06] [BUL 01][SCH 01][WAR 01]. O baixo custo dos nós sensores e o potencial desta tecnologia justificam a sua utilização em diversas áreas, tais como:

- Meio-ambiente: Rastreamento do movimento dos pássaros, pequenos animais; monitoração de condições ambientais que afetam colheitas e plantio (por exemplo: combate a geada, detecção de componentes químicos ou biológicos, irrigação); mapeamento da bio-complexidade ambiental, estudo da poluição e muitas outras [CER 01][AGR 00][MAI 02] [KAH 99] [BON 00].

- Saúde: controle de doenças contagiosas; interface para deficientes; monitorar pacientes; diagnosticar distúrbios; administrar drogas em hospitais, monitoração e localização de pacientes e médicos em hospitais [END 06] [BUL 01][SCH 01][WAR 01].

- Automação residencial: Através da instalação de nós sensores e atuadores em equipamentos utilizados no dia-a-dia, é possível automatizar tarefas comuns, permitindo além de uma interação dos aparelhos entre si e com outras redes, um gerenciamento local e remoto dos eletrodomésticos pelos seus usuários [PET 00].

- Aplicações militares: Monitoração de tropas; reconhecimento de terreno; detecção de alvos e de ataques biológicos, químicos ou nucleares [MOK 06].

- Monitoração de Estrutura/Equipamentos: Monitoração e identificação de falhas em estruturas (pontes, prédios, etc); monitoramento da fadiga de máquinas e equipamentos (motores, dutos de gás, etc); diagnósticos de máquinas [KIN 02][KIM 05].

- Aplicações comerciais: Automação de vendas e processos industriais; manutenção de inventário; monitoração de qualidade de produtos; detecção e vigilância de veículos e estabelecimentos [AGR 00] [CHO 00][EST 99][WAR 01].

Como apontado em [RUI 04b], a tendência nas rede de sensores sem fios, é a produção dos nós sensores em larga escala, barateando seu custo, e o investimento no desenvolvimento tecnológico levando a novas melhorias, capacidades e redução do tamanho dos nós sensores. Portanto, novas aplicações podem surgir aumentando a abrangência de uso das RSSFs.

Cada nó sensor não necessita ser instalado ou possuir posição predeterminada, possibilitando uma disposição aleatória em locais inacessíveis, como áreas de desastre. Logo, algoritmos e protocolos para RSSFs devem possuir a capacidade de auto-organização.

Além dos nós sensores, uma RSSF possui um ou mais nós de escoamento de dados, chamados de sorvedouros (sink, denominação do inglês, que é utilizada neste trabalho devido a "familiaridade" com esta palavra) ou estação base. O sink é o elemento através do qual a rede comunica-se com outras redes ou com um ou mais observadores [RUI 04a]. Ou seja, ele faz a interface entre a aplicação e a rede, servindo de ponto de entrada para a submissão dos interesses da aplicação e de concentrador das informações enviadas pelos nós sensores. O sink possui grande poder computacional e não tem restrições energéticas.

3.1. Componentes do nó sensor

Um nó sensor é constituído basicamente de unidades de sensoriamento, de processamento, de comunicação e de energia. Podendo também, dependendo da aplicação destinada, possuir unidades adicionais tais como: sistema de localização, gerador de energia e locomoção. A Figura 3.2 ilustra tais unidades presentes nos nós sensores.

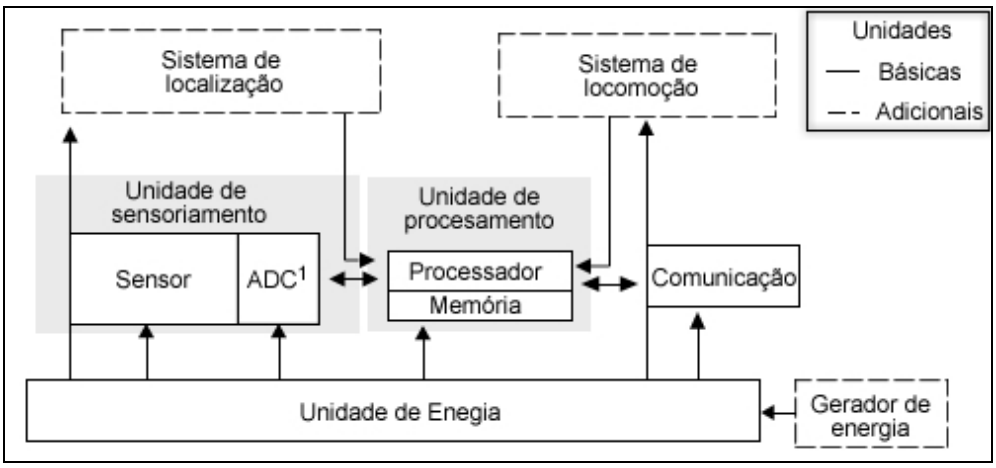

Figura 3.2. Componentes típicos de um nó sensor.

3.2. Funcionamento Básico da RSSF

Uma vez que os nós estejam espalhados no ambiente de interesse, eles, através da unidade de sensoriamento, coletam dados do ambiente (por exemplo, temperatura). Cada nó sensor possui capacidades de comunicação sem fio, no entanto transmitem os dados coletados a um conjunto de nós vizinhos, que estão na sua área de cobertura. Freqüentemente, uma

[1]ADC – Conversor analógico digital. Foi adota esta abreviação na língua inglesa em função da "familiaridade" com esta abreviatura.

transmissão direta entre os nós é possível, mas a transmissão em diversos saltos (i.e., *multihops)* usando nós intermediários é mais usual pois preserva energia [MUH 04]. Os dados, então, são roteados até o sink, e de lá disponibilizados a outras redes e aplicações. A Figura 3.3 ilustra este funcionamento básico.

A explicação do parágrafo anterior é uma visão simplificada, pois existem outras considerações adicionais para funcionamento da RSSFs. Logo após os nós serem espalhados no ambiente, uma fase de descoberta de rotas é necessária para que os nós estabeleçam as rotas que usarão para o tráfego de dados. Tarefas de sensoriamento são difundidas pelo nó sink para os nós sensores para que estes saibam quais eventos devem ser relatados. Quando um evento de interesse é detectado, os nós coletam este dado, realizam algum processamento sobre ele, e o enviam para o sink através da rota definida pelo protocolo de roteamento. Uma manutenção periódica da rota deve ser realizada devido à dinâmica da rede, provocada pelo movimento ou pela falha dos nós.

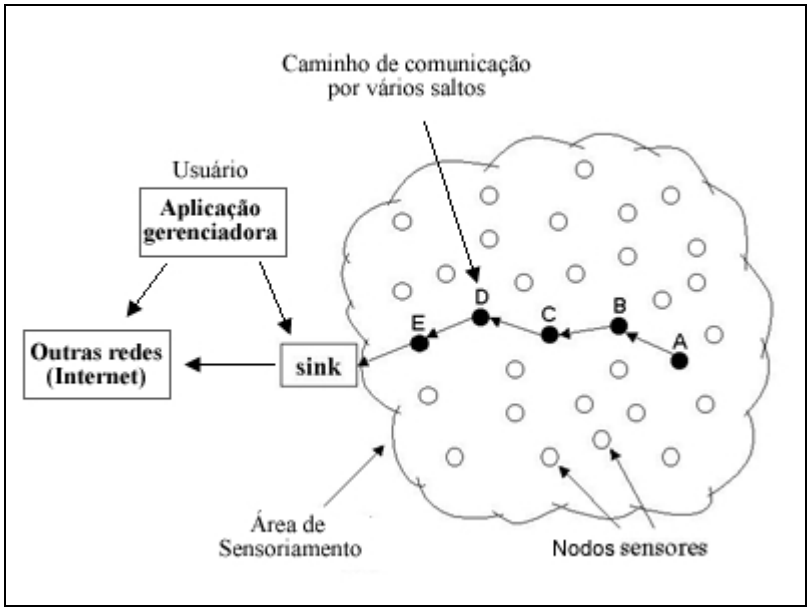

Figura 3.3. Esquema da arquitetura física da RSSFs (adaptado de [AKY 02]) .

3.3. Características da RSSF

Diversos fatores influenciam o projeto de uma RSSF, embora alguns destes fatores sejam compartilhados por várias RSSFs, independentemente do emprego. A seguir serão apresentados estes fatores (características e requisitos) que têm influência direta na arquitetura e nas decisões de projeto de uma RSSF [AKY 02][RUI 04b].

3.3.1. Consumo de Energia

Dentre todos os fatores relevantes em RSSF, podemos destacar o consumo energético como um dos mais importantes, visto que o nó sensor é um dispositivo microeletrônico com fonte de energia restrita. Além disso, na maioria das aplicações, pode ser impossível ou inviável fazer a substituição da fonte de energia. Dessa forma, as capacidades de sensoriamento, processamento e comunicação do nó sensor são limitadas pela disponibilidade de energia [AKY 02].

Otimizar o consumo de energia exige um controle de dissipação energética incorporado a todos os estágios do projeto e da operação de uma RSSF. No entanto, esta tarefa é bastante complexa, pois não se limita a diminuir o consumo de um único nó sensor, mas também prolongar a vida da rede [RAG 02].

A tarefa de um nó sensor pode ser dividida em três categorias: sensoriamento, processamento de dados, e comunicação. O consumo de energia de um módulo de sensoriamento varia de acordo com a natureza da aplicação. Como mostrado em [BOU 03][EST 02] o módulo de comunicação consome muito mais energia quando comparado aos módulos de processamento e sensoriamento. Portanto, muitas pesquisas concentram-se na conservação de energia através do desligamento do módulo de comunicação.

3.3.2. Tolerância a falhas

Em alguns sistemas é muito importante uma alta confiabilidade dos dados gerados. Características de emprego de uma RSSF em ambiente potencialmente hostil, como uma aplicação crítica, elevam em muito as chances de falhas. Os nós sensores podem falhar ou tornarem-se inoperantes por falta de energia, dano físico, ou ainda, interferência. Tolerância a falhas é a habilidade de se sustentar as funcionalidades da rede de sensores sem qualquer interrupção devido a falhas dos nós [SHE 01]. As falhas dos nós sensores não devem afetar a tarefa global da rede de sensores, que é a de fornecer informações sobre o ambiente monitorado. Protocolos e algoritmos devem ser implementados para alcançar os níveis de tolerância a falhas requisitados pelas aplicações das redes de sensores.

3.3.3. Expansibilidade

A suscetibilidade a falhas e a necessidade de cobertura de extensas áreas podem exigir um elevado número de nós sensores. O número de nós sensores empregados pode ser da ordem de

centenas ou milhares e, dependendo da aplicação, chegar a milhões. Novos esquemas devem ser capazes de funcionar com este elevado número de nós.

3.3.4. Meio de Transmissão

Uma RSSF pode explorar três possibilidades de comunicação sem fio: ótica, infravermelho e rádio freqüência. A comunicação ótica consome menos energia por bit transmitido e é sensível às condições atmosféricas. A comunicação por infravermelho é robusta contra interferências de dispositivos elétricos e possui um baixo custo de transceptores. Nenhum desses meios de comunicação necessita de uma área física para instalação de antena, mas necessitam de uma linha de visada (LOS – *Line of Sight*) para comunicação, isto é, transmissor e receptor devem estar alinhados. Logo, estes meios de comunicação não são viáveis para aplicações de monitoramento, pois a comunicação direcional nestes ambientes nem sempre é possível. A maioria dos hardwares usados em RSSFs utiliza rádio freqüência. A comunicação por rádio freqüência é baseada em ondas eletromagnéticas e suas vantagens são a facilidade de uso e a aceitação comercial, que tornam este tipo de comunicação viável para a plataforma dos sensores. Exemplos de comunicação por rádio freqüência são o μAMPS [SHI 01] que utiliza Bluetooth, e a arquitetura de rede de sensores sem fio integrado WINS (*Wireless Integrated Networks Sensors*) [POT 00].

3.3.5. Ambiente operacional

Nós sensores são empregados próximos ou dentro do ambiente de interesse. Sendo assim, eles têm que funcionar sob as mais diversas condições e ambientes, tais como: interior de grandes maquinários, fundo do oceano, campo contaminado, campo de batalha, acoplados a veículos ou a animais, etc.

3.3.6. Topologia da rede de sensor

Centenas a milhares de nós são utilizados em uma rede, podendo ser lançados ou colocados um a um no local que se quer monitorar. Devido ao fato do grande número de nós sensores e das freqüentes falhas o gerenciamento da topologia das RSSFs é uma tarefa desafiadora. Mudanças na topologia da rede são uma característica comum em RSSFs. No entanto, as mudanças de topologia das RSSFs não são atribuídas ao movimento dos nós, como acontece em redes *Ad Hoc* tradicionais ou nos sistemas celulares. Os nós sensores permanecem, em sua maioria, estacionários após o emprego. As mudanças na topologia ocorrem quando os nós

deixam de operar por falta de energia, são destruídos, ou os rádios são desligados para economizar energia [MIN 02].

3.3.7. Custo de Produção

Ao considerar o emprego de uma RSSF com um grande número de nós sensores, o custo de cada nó é importante para o custo total da RSSF. É ressaltado em [AKY 02] que o custo de um nó sensor deveria ser inferior a um dólar americano, para se viabilizar a RSSF, contudo este valor é dez vezes maior do que um rádio Bluetooth, o qual é conhecido como um dispositivo de baixo custo.

3.3.8. Classificação das RSSFs

Todos os fatores citados anteriormente são influenciados pelos requisitos da aplicação, pois uma RSSF é um tipo de sistema dependente da aplicação. Os parâmetros de configuração e manutenção variam de acordo com os objetivos da aplicação. Da mesma forma, a classificação das RSSF também depende de seu objetivo e de sua área de aplicação [RUI 04b]. A classificação das RSSFs pode ser, assim, analisada quanto à sua configuração em relação aos seguintes aspectos:

1. **Composição**
    a. Homogênea - Rede composta de nós de mesma capacidade de hardware.
    b. Heterogênea - Rede composta de nós com diferentes capacidades de hardware.
2. **Densidade**
    a. Balanceada - Quando a rede apresenta uma concentração considerada ideal de nós sensores por unidade de área.
    b. Densa - Quando a rede apresenta uma alta concentração de nós sensores por unidade de área.
    c. Esparsa - Quando a rede apresenta uma baixa concentração de nós sensores por unidade de área.
3. **Mobilidade**
    a. Estacionária - Rede em que os nós sensores são fixos durante toda a vida da rede.

b. Móvel - Rede em que os nós sensores podem ter sua posição espacial variável durante o tempo de vida de rede. Isto é, estes nós sensores podem ter a posição inicial onde foram depositados modificada por sua capacidade de locomoção, por forças da natureza e também pela entidade na qual estão acoplados.

4. **Distribuição**

   a. Irregular - Rede que apresenta uma distribuição não uniforme na área monitorada.

   b. Regular - Rede que apresenta uma distribuição uniforme na área monitorada.

5. **Organização**

   a. Hierárquica – Rede em que os nós são agrupados em grupos (*clusters*), onde cada grupo contém um líder (*cluster-head*). Em redes que possuem esta organização, os grupos formam hierarquias entre si, de forma que quando um dado necessita ser remetido ao sink, ele é remetido primeiro ao líder que irá enviar diretamente, ou através de outros líderes, ao sink. Fases de seleção dos nós líderes são necessárias. Esta organização tem primordialmente a intenção de reduzir o tráfego da rede com o objetivo de economizar energia.

   b. Plana - Rede em que os nós não são agrupados em grupos tendo seu funcionamento correspondente ao destacado na seção 3.2.

As RSSFs ainda podem ser classificadas quanto ao modo de sensoriamento e coleta dos dados da seguinte maneira:

1. **Periódica -** Os nós sensores coletam dados sobre os fenômenos em intervalos regulares. Por exemplo, relatar a temperatura a cada dez minutos.
2. **Dirigida a Eventos -** Os nós sensores coletam dados quando ocorrem eventos de interesse. Por exemplo, relatar quando temperatura for maior que 60 graus.
3. **Baseada em Consultas -** Os nós sensores coletam dados quando solicitado pelo observador. Por exemplo, relatar a temperatura agora.

3.4. Pilha de Protocolos das RSSFs

Esta seção descreve a pilha de protocolos comumente admitida para RSSFs, ilustrada na Figura 3.4, discutindo as características e funcionalidades para que se tenha conhecimento dos

objetivos de cada camada da arquitetura de comunicação. A camada de aplicação é discutida em mais detalhes por se relacionar com os objetivos deste trabalho.

Uma aplicação usando a rede não interage diretamente com o hardware da rede. Ao invés disso, a aplicação interage com o software do protocolo. A noção de protocolos em camadas fornece um conceito básico para o entendimento de como um conjunto complexo de protocolos podem trabalhar juntos com o hardware para prover um sistema de comunicação eficiente. A pilha de protocolos da RSSF caracteriza-se pela exploração do esforço colaborativo e da disponibilidade de energia dos elementos da rede para o estabelecimento de rotas, agregação de dados e a comunicação empregando o meio sem fio. A pilha da RSSF é dividida nas camadas física, de enlace de dados, de rede, de transporte e de aplicação.

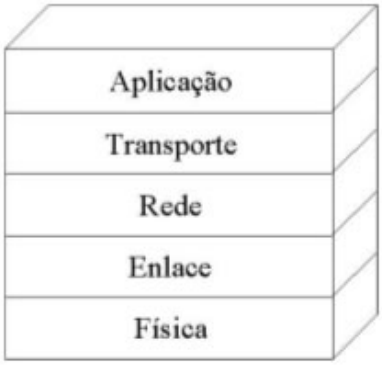

Figura 3.4. Pilha de Protocolo da RSSF.

3.4.1. Camada Física

Destina-se às necessidades de modulação simples, transmissão e recepção, uma vez que o ambiente é ruidoso e os nós podem ser móveis. Ela é responsável pela seleção de freqüência, geração da freqüência portadora, detecção do sinal, modulação e pela criptografia de dados.

Como mencionado anteriormente, a comunicação sem fio de longa distância pode ser muito custosa em termos de energia e complexidade de implementação. Durante o planejamento de uma camada física para redes de sensores, o gasto do consumo de energia é essencial, assim como os efeitos de reflexão, difração e degradação que o sinal pode sofrer. Em geral, a energia mínima requerida para transmitir um sinal sobre uma distância d é proporcional a d^n, onde $2 \leq n < 4$ [RAP 96]. A escolha de um bom esquema de modulação também é um ponto crítico para a comunicação confiável em uma rede de sensores.

3.4.2. Camada de Enlace de Dados

O objetivo da camada de enlace é assegurar conexões confiáveis em uma rede de comunicação, através das tarefas de detecção dos quadros de dados, multiplexação dos fluxos de dados, acesso ao meio e controle de erro. O ponto mais crítico e de fundamental importância para diversas aplicações de RSSF é o protocolo de acesso ao meio (MAC – Medium Access Control).

Um protocolo de controle de acesso ao meio para redes de sensores sem fio deve atingir dois objetivos [AKY 02]. O primeiro é a criação de uma infra-estrutura de rede. Como milhares de nós sensores estão densamente espalhados por um campo de sensores, o esquema MAC deve estabelecer enlaces de comunicação para transferência de dados. Isso forma a infra-estrutura básica necessária para a comunicação sem fio nó a nó e oferece a capacidade de auto-organização para a rede de sensores. O segundo objetivo é compartilhar eficientemente os recursos de comunicação entre os nós sensores [WU 00].

Em [RUI 04b] é dito que a determinação do limite inferior de energia requerida para a auto-organização da rede, os esquemas de controle de erro, os modos de operação para economizar energia e os cuidados com a mobilidade são alguns dos desafios da camada de enlace e dos protocolos de acesso ao meio.

3.4.3. Camada de Rede

Os protocolos da camada de rede são responsáveis pelo roteamento dos dados entre os nós sensores e o sink, uma vez que os nós de uma rede de sensores estão espalhados densamente pela área que se quer monitorar e precisam atuar em conjunto para entregar os dados coletados.

Ao contrário dos protocolos tradicionais que buscam diminuir o retardo fim-a-fim ou aumentar a vazão, em RSSFs eles visam estabelecer rotas que permitam estender o tempo de vida da rede por meio da racionalização do consumo de energia, muitas vezes, com sacrifício de outras métricas de desempenho [YOU 01].

Tratar das mudanças de topologia, do endereçamento, da expansibilidade e da interface com outras redes são requisitos esperados para a camada de rede

3.4.4. Camada de Transporte

Essa camada é necessária especialmente quando o sistema precisa ser acessado pela Internet ou por outras redes externas [AKY 02]. Diferentemente de protocolos como TCP (Transmission Control Protocol), os esquemas de comunicação fim-a-fim nas redes de sensores não são baseados em endereçamento global. Esses esquemas devem considerar que uma nomeação de dados baseada em atributos é utilizada para indicar os destinatários dos pacotes. Além disso, fatores como consumo de energia e expansibilidade, e características como roteamento centrado em dados (data-centric) exigem um tratamento diferente na camada de transporte de uma rede de sensores [POT 00].

3.4.5. Camada de Aplicação

Existem diversas aplicações definidas e propostas para RSSFs, como visto no Capítulo 1, cada qual com diferentes requisitos. Na literatura, vários trabalhos [TIL 02] [HEI 03] destacam a importância da participação da aplicação no processo de comunicação em RSSFs. Otimizações específicas da aplicação podem reduzir o número de transmissões e, por conseguinte, o consumo total de energia na rede. No entanto, protocolos para a camada de aplicação de RSSFs ainda constituem uma região pouco explorada.

A camada de aplicação contém uma série de protocolos de alto nível que são comumente necessários [TAN 96]. Ela faz a interface entre o protocolo de comunicação e o aplicativo que pediu ou receberá a informação através da rede. A camada de aplicação se refere ao fornecimento de serviços na rede.

Em [AKY 02] são sugeridos três possíveis protocolos para a camada de aplicação: protocolo de gerenciamento de sensores (SMP); protocolo de designação de tarefas e anúncio de dados (TADAP); e o protocolo para consulta e disseminação de dados (SQDDP). A seguir, descrevemos as funcionalidades de cada um destes protocolos.

3.4.5.1. Protocolo de gerenciamento de sensores

Este protocolo tem por finalidade tornar o hardware e o software das camadas mais baixas transparentes para as aplicações que gerenciam a rede. As redes de sensores possuem muitas áreas de aplicação diferentes, o protocolo de gerenciamento da camada de aplicação permite que elas utilizem a rede de sensores.

O protocolo deve acessar os nós (que não possuem identificadores globais) para realizar tarefas administrativas, tais como:

- Manutenção de regras de agregação e nomeação baseada em atributos;

- Sincronização

- Alteração de posição.

- Ligar e desligar os nós;

- Reconfiguração da rede;

- Autenticação e segurança na comunicação de dados.

Em [TIA 02] é proposto um esquema de escalonamento de nós, usando conhecimento da aplicação, que pode reduzir o consumo de energia geral e, portanto incrementar o tempo de vida da RSSFs. Analisando a cobertura da rede, este esquema permite garantir a cobertura da área de sensoriamento original após desligar nós redundantes.

3.4.5.2. Protocolo de designação de tarefas e anúncio de dados

Outra operação importante nas redes de sensores é a disseminação de consultas (interesses). Os usuários enviam seus interesses para a rede toda ou para um subconjunto de nós. O interesse pode ser sobre um certo atributo de um fenômeno ou um evento [AKY 02]. Outra abordagem é o anúncio de dados disponíveis onde os nós sensores divulgam os dados para os usuários, que podem então consultar aqueles de interesse. Com isso, o protocolo visa gerenciar a sinalização do sensor que possui disponibilidade de dados e a disseminação de interesses dos usuários. Uma interface eficiente para estas operações auxilia as camadas mais baixas como, por exemplo, no roteamento.

3.4.5.3. Protocolo para consulta e disseminação de dados

Estabelece uma interface para as aplicações dos usuários consultarem, responderem e coletarem respostas da RSSF. As consultas normalmente não são direcionadas a nós em particular. Ao invés disso, esquemas de nomeação de dados baseados em atributo ou localização são utilizados. Como exemplo, temos a SQTL [CHI 01], uma linguagem para consultas em sensores proposta como uma aplicação que fornece um grande conjunto de serviços.

Protocolos da camada de aplicação para RSSF devem isolar as aplicações das características de infra-estrutura de rede inferiores. Seu principal objetivo é oferecer suporte ao desenvolvimento, manutenção e execução de aplicações. Isto inclui mecanismos para formular tarefas de sensoriamento de alto nível, comunicar as tarefas para a rede, realizar a agregação de dados e coordenar os nós na realização dessas tarefas. Além disso, têm que ser robustos, tolerantes a falhas e ter requisitos mínimos de armazenamento e processamento.

3.5. Rede de Atuadores e Sensores sem Fio

A inserção de novos elementos chamados de atuadores fornece vantagens adicionais a RSSF. Esta seção enuncia as características das redes de atuadores e sensores sem fio (RASSF), cujo conhecimento é necessário para o entendimento do trabalho proposto. Muitos dos conceitos vistos para RSSF permanecem válidos.

As RASSFs oferecem as vantagens das RSSFs e adicionalmente têm capacidade de reação automática e em tempo real aos eventos capturados. Algoritmos e protocolos desenvolvidos paras as RASSFs devem atender aos mesmos requisitos das RSSFs, tais como: escalabilidade e eficiência energética; além disso, devem considerar a heterogeneidade dos nós e os requisitos de aplicações de tempo real [AKY 04].

Estas redes podem ser parte integrante das mesmas áreas de atuação das RSSFs. Quanto a um exemplo em caso de incêndio, os sensores transmitem a origem exata e intensidade do fogo para os atuadores, enquanto os atuadores são os "sprinklers" de água, que podem facilmente extinguir o fogo antes que ele se torne incontrolável. Apesar de normalmente os *sprinklers* que estamos acostumados não serem providos de capacidades tais quais às dos atuadores, atualmente, já é possível encontrar *sprinklers* que tenham capacidades iguais à dos atuadores [AKY 04].

3.5.1. Componentes do nó atuador

Nós sensores e atuadores são diferentes em vários aspectos. Primeiramente são destinados a atividades diferentes, nós sensores monitoram e nós atuadores atuam no ambiente. Além disso, nós sensores são de baixo custo, baixas capacidades de computação, comunicação sem fio e dispositivos de baixa energia com sensoriamento limitado. Já os atuadores possuem um custo mais alto, melhores capacidades de processamento, comunicação sem fio e uma bateria de maior duração.

Além disso, a quantidade de nós sensores implantados no ambiente, geralmente, é da ordem de centenas ou milhares, no entanto não é necessária uma densa distribuição de nós atuadores, pois os atuadores possuem recursos melhores e podem atuar em grandes áreas [AKY 04].

Nós atuadores possuem as seguintes unidades: energia, comunicação, processamento, decisão (controlador), atuação e um conversor digital analógico (DAC), conforme ilustra a Figura 3.5. Os eventos são recebidos pela unidade de comunicação e, então, cada evento é analisado na unidade de decisão gerando comandos que irão executar ações como saída. Estes comandos são então convertidos para sinais analógicos via DAC e são transformados em ações via unidade de atuação.

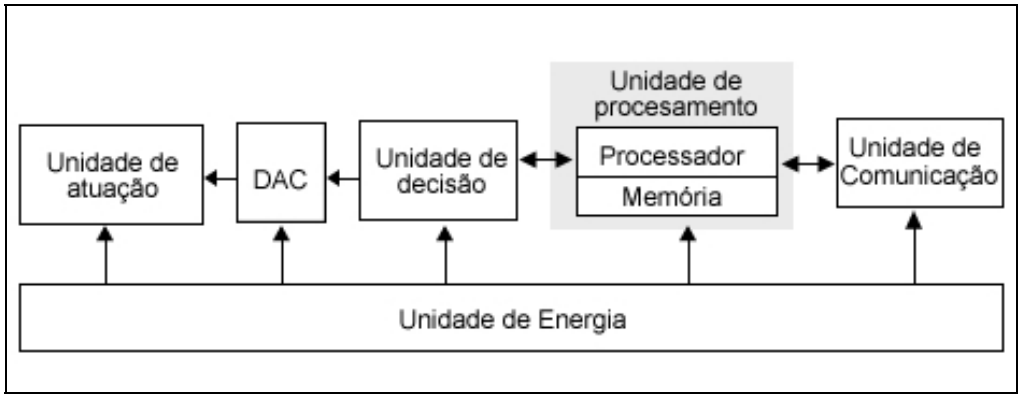

Figura 3.5. Componentes do nó atuador

3.5.2. Funcionamento Básico da RASSF

Em RASSF, os sensores coletam informações do mundo físico, enquanto os atuadores processam as informações para tomar decisões e realizar ações apropriadas sobre o ambiente. Como mostrado na Figura 3.6, nós sensores e os nós atores são espalhados sobre uma área de sensoriamento e atuação. O sink monitora toda a rede e realiza a comunicação da RASSF com a aplicação, ou ainda, disponibiliza as informações coletadas a outras redes. Isto permite que os usuários façam o sensoriamento e atuem no ambiente a distância.

Sensores, ao detectarem um fenômeno, podem operar de duas maneiras distintas: (a) rotear os dados de volta ao sink; (b) transmitir os dados ao atuador que, ao processar estes dados, inicia uma ação apropriada. Em [AKY 04] foram definidas duas arquiteturas de RASSF para os dois modos de operação que, dependendo da aplicação, uma delas se torna mais vantajosa.

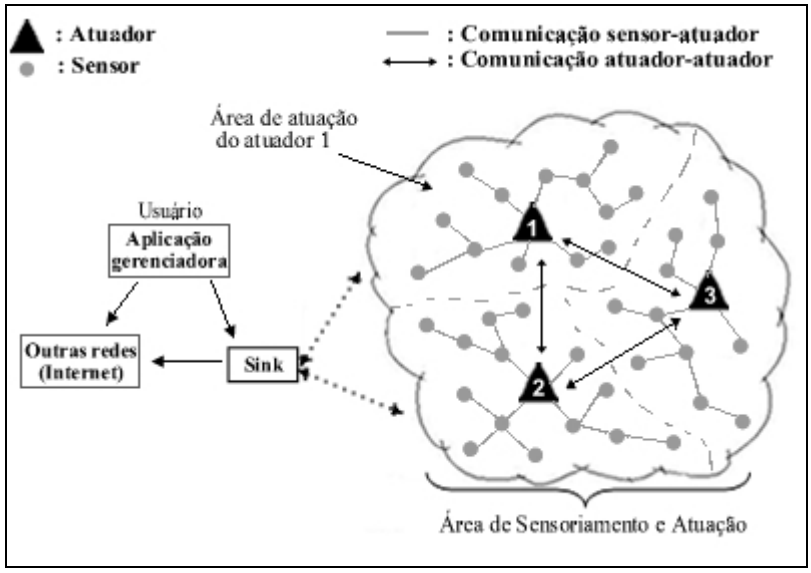

Figura 3.6. Esquema da arquitetura física das RASSFs.

A primeira arquitetura, chamada de *Semi Automática*, é bastante similar à arquitetura tradicional das RSSFs, onde os nós sensores ao detectarem um evento transmitem suas leituras ao sink, que é responsável por interpretá-las, e caso seja necessário, enviar comandos de atuação aos atuadores, conforme ilustrado na Figura 3.7(a). A segunda é a arquitetura *Automática*, esquematizada na Figura 3.7, onde as leituras detectadas pelos nós sensores são enviadas aos atuador, que é responsável por interpretá-las, e caso seja necessário, atuar no ambiente. A primeira arquitetura demonstra um modo de interpretação centralizada, onde o sink seria a entidade central responsável por coordenar todos os atuadores, enquanto a segunda demonstra um modo de interpretação distribuída, onde cada atuador seria responsável por interpretar os dados de sua região de atuação.

A interpretação de modo distribuída traz vantagens importantes no uso de ambientes de segurança crítica, como: (a) *baixa latência*, pois a informação sentida é encaminhada a atuadores que estão mais próximos do evento do que o sink; (b) *longevidade maior da rede*, pois dados não são roteados até o sink, evitando desgastar rapidamente os nós próximos ao sink. Embora a agregação de dados reduza os gastos dos nós próximos ao sink, ainda assim a falha destes nós é maior do que de outros da rede. Igualmente, com a interpretação descentralizada, podemos imaginar um maior gasto dós nós próximos ao atuador. Entretanto, os atuadores são em maior número do que o sink, e estão inseridos dentro do ambiente, logo possuem mais nós próximos a ele. E ainda é muito mais provável que diferentes atores possam ser acionados para cada evento, implicando no revezamento dos nós sensores. Como resultado, a interpretação descentralizada tem um tempo de vida maior que a centralizada [AKY 04].

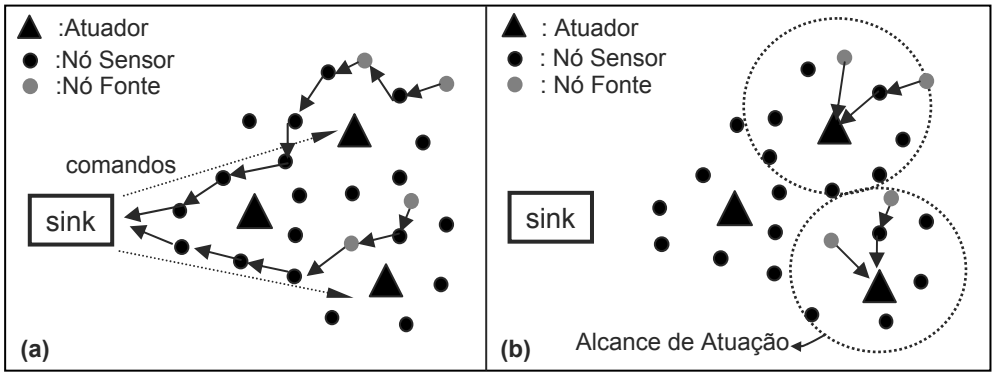

Figura 3.7. (a) Interpretação Centralizada - eventos são enviados ao sink; (b) Interpretação Descentralizada - eventos são enviados aos Atuadores.

3.6. Considerações Finais

O presente capítulo apresentou os principais aspectos envolvendo as RSSFs e RASSFs, incluindo as características, os requisitos comuns e o funcionamento de cada uma. Também foram apresentadas questões relacionadas a pilha de protocolos usada pelos nós sensores. Do que foi apresentado é necessário ressaltar que o consumo de energia constitui o fator mais importante das RSSFs e RASSFs. O tempo de vida da rede depende que sejam adotadas medidas para economizar energia em todas as camadas da pilha de protocolos. A RASSF fornece vantagens adicionais à RSSF, possibilitando a reação automática, e em tempo real, aos eventos coletados pelos nós sensores.

Uma vez apresentado os aspectos de RSSF e RASSF, resta agora dedicar ao estudo dos aspectos importantes da interpretação de contexto e ordenação de eventos (próximo capítulo) para que todo o embasamento teórico seja alcançado.

4. Interpretação de Contexto e Ordenação de Eventos

Eventos capturados pelos nós sensores podem, ou não, possuir relação entre si, ou seja, serem correlacionados. Quando dois ou mais eventos não estão correlacionados, cada evento que chega pode ser tratado independentemente. Entretanto, quando os eventos estão correlacionados, um mecanismo para capturar e interpretar estes eventos, em conjunto, é necessário. Por exemplo, em situações de incêndio, a fumaça e a temperatura alta determinam a existência de fogo, se assim for descrito nas regras de monitoramento. Logo, a existência do fogo é determinada pela ocorrência de dois eventos correlacionados, a fumaça e a temperatura para a interpretação da existência de fogo.

Eventos podem estar correlacionados com outros não só em relação ao sentido que expressam, mas também em termos do momento em que cada um deles ocorreu. Isto significa que a ordem em que os eventos foram "sentidos", quando eles foram "sentidos" e o intervalo de tempo entre cada evento podem ser importantes para a detecção das situações emergenciais. A Figura 4.1 ilustra a interpretação dos eventos através da correlação de eventos. As seções subseqüentes analisam a interpretação de contexto e ordenação de eventos, separadamente.

4.1. Interpretação de Contexto em RASSF

Antes de começar a discutir sobre interpretação de contexto é necessário compreender o que vem a ser contexto. Contexto é definido neste trabalho como qualquer informação sobre o ambiente físico que é importante para a aplicação e possa caracterizar a situação de uma entidade [DEY 01]. Sensores capturam eventos. Contextos são compostos de eventos que são correlacionados.

Contextos podem ser simples quando são compostos por eventos simples, como a temperatura atual, pressão do ar atual, presença de fumaça, etc., ou mais complexos quando compostos por dois ou mais eventos correlacionados, como detecção de incêndio (uma combinação de eventos correlacionados: "alta temperatura" e "presença de monóxido de

carbono". Quando a relação entre os eventos é temporal, isto é, a ordem no tempo em que cada evento é importante, contextos tão complexos como "Porque uma certa situação aconteceu no ambiente?" poderiam ser respondidos através de uma interpretação mais complexa que demanda uma ordenação dos eventos. Como um exemplo a RASSF poderia interpretar os seguintes contextos: porque a asa da aeronave quebrou ? A pressão estava acima do normal (pressão acima do normal) o que provocou uma rachadura na superfície da asa (presença de rachadura) levando asa a se partir – isto pode ser deduzido à partir da ordenação dos eventos: A alta pressão ocorreu antes da rachadura surgir.

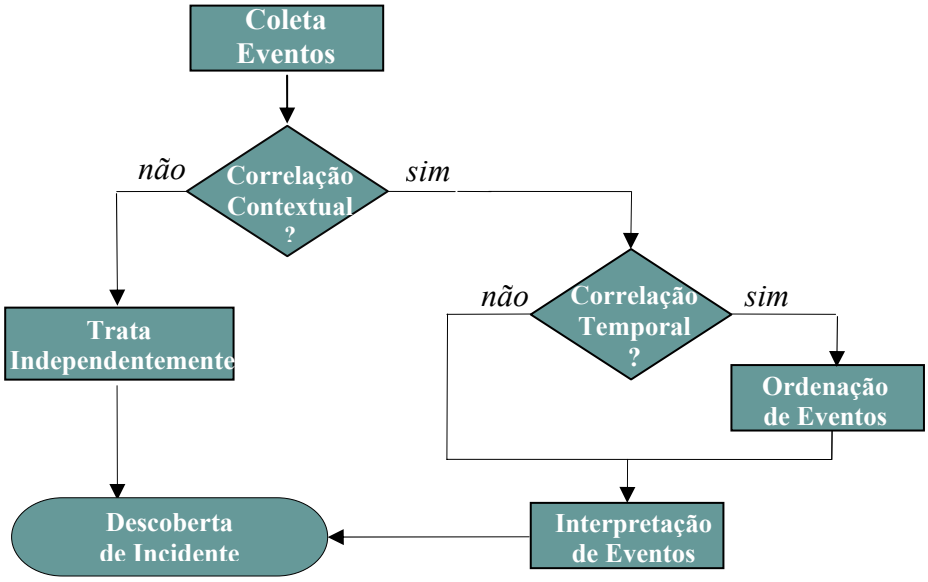

Figura 4.1. Esquema de Correlação de Eventos

Geralmente, cada nó sensor fornece um único dado. Para resolvemos tarefas de sensoriamento de alto nível, os nós sensores devem coordenar e dividir as tarefas entre eles, levando em conta as características individuais de cada nó (por exemplo, tipos dos sensores anexados, localização, energia residual). As leituras individuais dos sensores devem ser agrupadas para resultar em um alto nível de sensoriamento.

A interpretação dos eventos pode ser realizada através de técnicas do campo da inteligência artificial (desde modelos baseados em regras, sistemas de decisões baseada em casos, árvores de decisão até redes neurais). Outras técnicas de diferentes áreas de

conhecimentos ainda podem ser usadas, como: teoria dos grafos, autômatos, etc. No capítulo 5 é descrita nossa solução de interpretação de contexto; por ora, são apresentados alguns trabalhos relacionados à interpretação de contexto.

Um mecanismo de detecção de contexto chamado PROCON é descrito em [AHN 06]. Nesse mecanismo as decisões de contexto são feitas de forma distribuída, através de nós chamados "*event headers*", que são conectados via uma rede overlay de contexto. Os "*event headers*" são similares a qualquer nó sensor da rede – eles têm o mesmo poder de processamento, limitações energéticas e capacidade de transmissão. Por causa da baixa capacidade de transmissão dos nós sensores, a comunicação entre os "*event headers*" (para transmitir e interpretar contextos) é *multi-hop* (i.e, através de vários saltos) usando outros sensores da rede. É sabido que a comunicação *multi-hop* quando comparada com a comunicação direta entre atuadores é mais suscetível a falhas e implica em maior latência e consumo de energia dos nós da rede, pois as mensagens percorrem vários sensores. Diferentemente do PROCON, a solução apresentada neste trabalho usa nós atuadores com maiores capacidades como nó agregador e interpretador. Quando os atuadores ou sink precisam de um contexto de um particular atuador, eles podem subscrever para aquele contexto. Portanto, somente ele que subscreveu para o contexto irá receber.

Soluções como as apresentadas em [MAD 05],[SHE 01] e [BON 01] utilizam protocolos baseados em consultas (*Query-Based*) para a seleção e captura das informações desejadas na RSSF. Estas soluções abstraem a RSSF como uma coleção de dados distribuídos, onde consultas semelhantes à de bancos de dados relacionais são enviadas aos nós sensores. Os nós sensores que possuam as informações requeridas respondem à consulta roteando a uma entidade central responsável por interpretá-las. Como discutido na seção 3.7, o modo centralizado implica em os dados percorrerem caminhos *multi-hops* (caminhos de múltiplos saltos) maiores, gastando mais de energia dos sensores, maior latência, e diminuído o tempo de vida da rede pela sobrecarga dos nós próximos a entidade central

Em [SHE 01] é apresentada uma arquitetura para RSSF, chamada SINA, que fornece um mecanismo para efetuar consultas, monitoramento e tarefas da rede de sensores. SINA adota um esquema de nomeação centrada em dados (*data centric*) e consiste de três componentes funcionais: agrupamento hierárquico (*hierarchical clustering*), nomeação baseada em atributos (*attribute-based naming*) e ciência de localização (*location awareness*). Como parte da arquitetura, é utilizada uma linguagem chamada *Sensor Query and Tasking Language (SQTL)*. SQTL é uma linguagem de script procedimental, desenvolvida para ser

flexível e compacta, com a capacidade de interpretar consultas declarativas simples. Além de acessar ao hardware do sensor (exemplo, getTemperature, turnOn), ser ciente de localização (exemplo, isNeighbor, getPosition) e possuir primitivas de comunicação (tell, execute), ela também fornece a construção de tratamento de eventos que é apropriado para muitas aplicações de rede de sensores. Três tipos de coleta de eventos são suportados pela SQTL: baseada em consulta, periódica e de eventos causados pela expiração de um timer. Tais eventos são definidos, respectivamente, pelas palavras-chaves receive, every e expire.

COUGAR[BON 01] e TinyDB [MAD 05] permitem aos usuários inserirem consultas no servidor através de uma linguagem simples chamada *SQL-like*, esta linguagem descreve os dados que eles desejam coletar e como eles desejam combinar, transformar e resumir estes dados. Estas soluções utilizam um conceito de processamento na rede (*in-network*), onde os dados são agregados à medida que se deslocam em direção ao sink. Além disso, permitem que os resultados de consultas anteriores sejam armazenados para o uso em consultas futuras.

Não existem muitos trabalhos presentes na literatura sobre interpretação de contexto em RASSF. Como estas redes tornam-se mais poderosas, interpretações mais complexas poderão ser processadas e mais autonomia poderá ser dada às redes para atuar no ambiente físico.

4.2. Ordenação de eventos em RASSF

Em [LAM 78] é dito que o tempo é fundamental para a nossa forma de pensar e entender determinadas situações e fenômenos, sendo derivado do conceito básico da ordem em que os eventos ocorreram.

Quando tratamos de rede de sensores, que são sistemas distribuídos, é necessária a coleta de informação de eventos em diferentes partes do sistema. Dada a natureza da comunicação destes sistemas, não é possível assumir que a ordem dos eventos coletados das diferentes partes do sistema reflita a ordem em que aconteceram ou foram enviados, sem que certas precauções sejam tomadas. Dessa forma, para apresentarmos ao observador uma "imagem" consistente e coerente dos eventos monitorados é necessária uma ordenação das mensagens de monitoramento [HOF 93].

Embora existam diferentes aplicações do uso de rede de sensores, em muitas delas a ordem em que os eventos ocorrem no mundo físico é um requisito fundamental, pois implica diretamente na interpretação correta do estado atual do ambiente. A ordenação das mensagens

de monitoramento evita falsas conclusões que poderiam implicar em prejuízos de patrimônio e vidas, em aplicações de monitoramento e aplicações de monitoramento de situações emergenciais.

A ordenação das mensagens recebidas é necessária porque elas podem sofrer atrasos em sua rota desde o nó remetente até nó destinatário, devido à capacidade de comunicação tipicamente limitada que é compartilhada por nós dentro da mesma área de comunicação. Estes atrasos podem ser originados por diversos fatores:

(a) Atraso embutido na transmissão de uma mensagem de um nó a outro;

(b) Atrasos pela competição pelo uso do meio de comunicação pelos nós;

(c) Atrasos adicionados por protocolos da camada MAC;

(e) Multiplicação dos atrasos do item *a* quando utilizando esquemas de comunicação em vários saltos;

(f) Atrasos decorrentes da dinâmica da rede.

A seguir, são descritos em detalhes cada um destes atrasos com base nos trabalhos [YE 02] [KAS 89] [ROM 02].

Em cada nó sensor de uma rede, o atraso embutido na transmissão de uma mensagem de 50 bytes de um nó para outro, ou seja em único salto (*single-hop*), é aproximadamente 20ms [ROM 02]. Entretanto, este atraso aparentemente pequeno se torna bastante relevante quando em conjunto com os demais atrasos descritos a seguir.

Em uma densa rede de sensores, os alcances de sensoriamento e de comunicação se sobrepõem. Isto é, muitos nós irão "sentir" o mesmo evento a partir de diferentes pontos de vista. Estas informações redundantes são usadas para obter maior precisão e confiabilidade nos eventos sensoriados. Entretanto, quando os nós querem reportar eventos, eles devem competir pelo meio de comunicação. Assumindo que dez nós detectem o evento, o primeiro dos dez sensores a transmitir eventos terá um atraso em um único salto de *20ms*, enquanto o último terá de *200ms* no caso de uma alocação ideal [ROM 02]. Além do mais, protocolos da camada MAC (Medium Access Control) baseados em CSMA (Carrier Sense Multiple Acces) introduzem atrasos adicionais para evitar/tratar colisões [YE 02].

Quando eventos capturados pelos sensores geram mensagens que devem percorrer vários nós, através de múltiplos saltos na rede de sensores, o atraso é incrementado a cada salto. Em uma rede em que muitos eventos são monitorados, o atraso pode ser estimado

multiplicando o atraso de um único salto pela quantidade de saltos necessários. Usando os números anteriores, o atraso *d* pode ser estimado como *200ms <= d <=2s* para um caminho com dez saltos [ROM 02]. E ainda, protocolos energeticamente eficientes da camada MAC desligam periodicamente o rádio, o que acarreta num atraso variando de zero para centenas de milisegundos ou até segundos [YE 02].

Além de todas estas condições que geram atrasos, outros tipos de dinâmica de rede (por exemplo, nós que falham devido à falta de energia, nós destruídos devido a influências ambientais, obstruções ambientais temporárias, nós sendo acrescentado à rede) conduzem a freqüentes reconfigurações. Muitas reconfigurações podem ser necessárias para se entregar uma mensagem através de múltiplos saltos na rede. Cada reconfiguração freqüentemente requer procedimentos longos como descoberta de rotas e vizinhos, que aumentam ainda mais o atraso das mensagens [ROM 02]. Em [KAS 89] foi mostrado que a descoberta de novos vizinhos, por exemplo, pode levar cinco segundos em sistemas de rádio como o *Bluetooth.*

Portanto, o atraso de uma mensagem em sua trajetória pela rede pode variar muito, enquanto uma mensagem pode atrasar apenas centenas de milisegundos, outra pode atrasar até dezenas de segundos. Esta variação muito grande dos atrasos pode conduzir a uma entrega desordenada de mensagens. O atraso se torna ainda mais evidente em redes de sensores grandes e densas, principalmente porque os caminhos são mais longos, a concorrência pelo meio é maior e um número maior de reconfigurações pode ser necessário.

Para ilustrar a importância da ordenação de eventos considere o seguinte exemplo: em uma situação de incêndio, uma pessoa dentro do ambiente procura uma saída. Bombeiros de fora do ambiente monitoram a posição dela, através de uma interface, com o objetivo de realizar o resgate. À medida que a pessoa procura a saída os sensores (*1*, *2* e *3*) capturam sua presença e enviam aos bombeiros. A Figura 4.2(a) demonstra a trajetória correta desta pessoa, que se desloca na seqüência dos sensores *1, 2* e *3*. No entanto, um atraso no envio da mensagem de monitoramento do sensor *2*, faz com que a mensagem do sensor *3* chegue antes. Implicando numa interpretação incorreta da trajetória desta pessoa, como mostra a Figura 4.2(b), logo a equipe de resgate se deslocaria no sentido contrário ao movimento real desta pessoa. Portanto, a entidade ao receber as mensagens de monitoramento não pode imediatamente disparar uma interpretação, visto que, mensagens anteriores à que acabou de chegar, podem ter sofrido atrasos, e ainda estarem em trânsito pela rede.

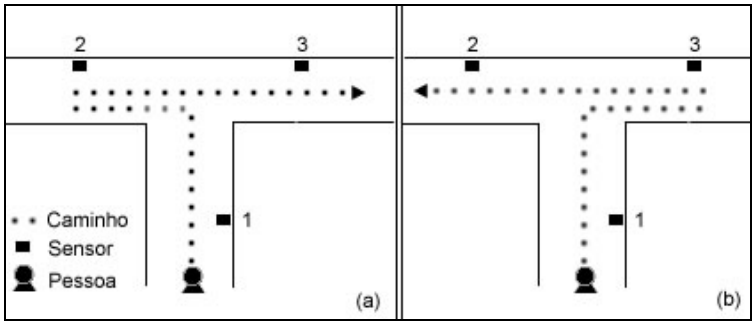

Figura 4.2. Ordenação dos Eventos: (a) Ordenação correta; (b) Ordem incorreta devido atraso da mensagem 2.

Tudo isso indica que existe uma necessidade real de ordenação das mensagens monitoradas, uma vez que o atraso das mensagens está presente na rede de sensores e pode influenciar na correta interpretação dos eventos.

4.2.1. Mecanismos de Ordenação de Eventos

Ordenação de mensagens é um problema bem estudado e com uma história construída desde o advento das redes de computadores. Entretanto, a arquitetura única das redes de sensores traz novos requisitos e uma nova dimensão para os problemas de ordenação tratados no passado.

Devido a atrasos, informações sobre uma seqüência de eventos (*e1*, *e2*, *e3*) podem chegar numa ordem incorreta (*e1*, *e3*, *e2*). A entidade ao recebê-las não deve imediatamente disparar uma interpretação, visto que mensagens anteriores à que acabou de chegar, podem ter sofrido atrasos, e ainda estarem em trânsito pela rede. O exemplo ilustrado na Figura 4.2 ilustrou exatamente este caso. Portanto, quando tratamos de uma ordenação de eventos que deve ser feita em tempo real deve-se questionar: Quanto tempo devemos esperar, ou o que devemos adotar para garantir que a mensagem, recebida neste instante, não é cronologicamente posterior a uma mensagem em trânsito.

Técnicas que visam solucionar este questionamento foram e têm sido propostos em redes de computadores tradicionais e nas redes de sensores sem fios. Logo, algumas técnicas foram desenvolvidas utilizando tanto o tempo lógico quanto o tempo físico, conforme descritas à seguir.

4.2.1.1. Tempo Físico

Uma das primeiras soluções que vem à mente para a ordenação cronológica entre pares de eventos é o uso de relógios físicos sincronizados. Timestamps[1] locais são associados aos eventos coletados pelos sensores. Com isso, a ordem cronológica e o tempo decorrido entre um par de eventos pode ser determinada simplesmente pela comparação dos timestamps associados a eles.

Podemos destacar algumas técnicas de ordenação cronológica de mensagens/eventos através do uso do tempo físico, tais como: técnica de retardamento (Delaying technique), protocolo Heartbeat e o mecanismo TMOS. Todas elas são descritas, detalhadamente, a seguir.

A. Técnica de Retardamento

Técnicas de retardamento (Delaying Techniques) como descritas em [MAN 97] [SHI 90], de idéia bastante simples, determinam que se espere por um certo tempo antes de realizar a interpretação dos eventos. Com isso, mensagens que atrasaram podem chegar e serem avaliadas junto com as outras mensagens que não atrasaram.

Havendo um sincronismo de relógio físico, assume-se que o envio de uma mensagem de nó a outro na rede leve um tempo D, ou seja D é o atraso na rede. O receptor que quer receber as mensagens *m* na ordem cronológica, mantém uma lista das mensagens ordenadas pelos seus timestamps *m:t*. Quando uma nova mensagem chega ela é inserida na ordem correta. O primeiro elemento m0 dessa lista é removido e passado para interpretação quando o relógio do receptor for maior que $m0:t + D$.

Apesar desta técnica parecer atrativa, pois não requer troca de mensagens adicionais para realizar a ordenação, o problema dela está na atribuição correta deste atraso *D*, pois, como visto, existe uma grande variação no atraso da rede. Ao definirmos um atraso *D* menor que o máximo atraso da rede, não estamos garantido uma ordenação correta dos eventos, já que uma mensagem pode chegar atrasada o suficiente para não ser avaliada antes das mensagens cronologicamente posteriores a ela. Por outro lado, ao definirmos o máximo valor de *D* sendo o máximo atraso da rede, estaremos embutindo um atraso artificial e alto a todas as mensagens. Artificial porque em muitas situações não haverá mensagens transitando pela

[1] Timestamp é um termo originado do inglês com conotação em português de "etiqueta de tempo". Corresponde à gravação do tempo corrente do sistema em uma mensagem/evento.

rede, mas o sistema ficará aguardando por um tempo *D*. Se o sistema tiver que esperar por um atraso *D* muito grande ele ficará muito tempo ocioso.

B. Protocolo Heartbeat

Protocolo Heartbeat (batida de coração) [Hay 96], uma outra idéia bastante simples, utiliza canais FIFO[1] entre todos os nós da rede. Ela assume que, ao receber uma mensagem, será confirmado que não existe nenhuma mensagem anterior a esta em trânsito, através de mensagens enviadas por todos os outros nós em intervalos Δ.

Explicando melhor, da mesma forma que a técnica do retardamento, o receptor mantém uma lista das mensagens ordenadas. A primeira mensagem *m0* é removida da lista e entregue à aplicação quando o receptor tiver recebido, de cada nó *i* da rede, mensagens *mi:t* > *m0:t*. A propriedade FIFO assegura que todas as mensagens anteriores a *m0:t* (aquelas que se atrasaram na rede) já tenha sido recebidas pelo receptor antes que o receptor tenha recebido todas as mensagens *mi*. Para evitar que o receptor espere indefinidamente, todos os nós enviam mensagens de "controle" em intervalos regulares de tempo Δ.

Como na técnica do retardamento, não é fácil atribuir o intervalo regular correto de tempo Δ. Assumir um valor pequeno a Δ, implicará em um custo adicional indesejável de mensagens de controle. E ainda, mesmo que eventos não sejam detectados e, por conseguinte, mensagens não sejam geradas, mensagens de controle estariam sendo enviadas gastando energia da rede. Assumir um valor alto a Δ, implicará no mesmo problema da técnica de retardamento quando assumindo um *D* alto. O sistema ficará muito tempo ocioso.

C. Mecanismo TMOS

Em [ROM 02] é descrito um mecanismo muito interessante para ordenação de mensagens em rede de sensores chamado TMOS (Temporal Message Ordering in Sensor Networks). Para isto ele adota um esquema que, ao contrário das técnicas anteriores, não depende da especificação de um determinado tempo. Ele assume que o canal de comunicação entre quaisquer pares de nós tenha propriedade FIFO.

[1] Canais FIFO são canais que se comportam de maneira FIFO (*first-in,first-out*), isto é, a mensagem que chega primeiro ao sensor é encaminhada/tratada primeiro, e as mensagens que chegam após esperam até que a primeira seja encaminhada/tratada.

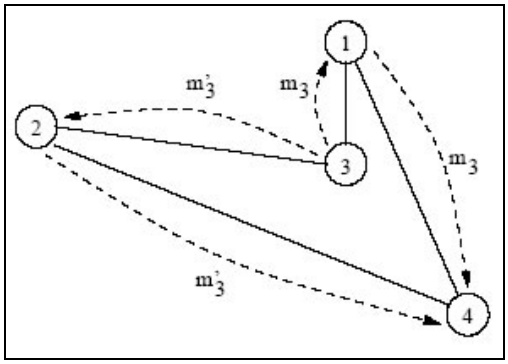

Figura 4.3. Funcionamento básico do TMOS (adotado de [ROM 02]).

Através de um exemplo explicaremos a idéia básica deste mecanismo. Neste exemplo, existem quatros nós, onde o nó quatro é o responsável pela recepção dos eventos gerados pelos outros nós. A Figura 4.3 servirá de auxílio à compreensão do algoritmo que é descrito nos passos:

1. Primeiramente, os nós são arranjados em um anel lógico, como mostrado na Figura 4.3 pela linha sólida.
2. Todo nó que deseja entregar evento sensoriado deve gerar duas cópias e enviar em sentidos contrários no anel. Ou seja, para entregar um evento detectado *m3*, o nó três envia duas copias (*m3* e *m3'*) através do anel. A mensagem *m3* é enviada em sentido horário, enquanto *m3'* é enviada em sentido anti-horário.
3. Após receber *m3*, o nó um encaminha *m3* somente depois de enviar, se existir, todo evento capturado por ele localmente. O mesmo acontecerá para o nó dois ao receber *m3*'.
4. O nó quatro insere todos os eventos recebidos dentro de uma lista que é ordenada de acordo com os timestamps das mensagens. Devido à propriedade FIFO, o nó quatro irá receber pelo menos uma cópia de todos os eventos com o *timestamp* menor que *m3.t* antes de receber a segunda cópia de *m3*. Por exemplo, se uma mensagem *m1*, gerada pelo nó um, é um evento detectado anterior a *m3,* ou seja $m1.t < m3.t$. O nó 1 irá enviar uma cópia de *m1* para o nó 4 antes de encaminhar *m3* recebida do nó três. A propriedade FIFO garantirá que *m1* chegara ao nó quatro antes de *m3.*
5. A primeira mensagem da lista (i.e., a que tem menor timestamp) será removida e entregue a aplicação quando chegar a segunda cópia.

Quando a segunda cópia de evento chega ao nó quatro, é certeza que pelo menos uma cópia de todos os eventos anteriores foi recebida e inserida na lista. Isto é, uma mensagem

somente é entregue à aplicação se todas as mensagens anteriores a ela já tiverem sido entregues.

Um problema do algoritmo TMOS é que nós do anel próximos ao nó receptor dos eventos demorarão a confirmar temporalmente suas mensagens. Estes nós ao enviarem suas mensagens terão que uma chegará rapidamente, entretanto, a outra (de sentido contrário) deverá percorrer todo o anel, que em muitos casos poderá ser grande. Algumas aplicações de RSSF e RASSF, como monitoramento de condições críticas, são extremamente sensíveis à latência.

Outro problema seria a própria utilização do anel, pois o anel fornece um caminho único para todas as mensagens da rede se deslocarem. Logo, em situações onde muitas mensagens são geradas, problemas como o engarrafamento das mensagens seria ocasionado pela grande competição pelo meio físico de transmissão. Nesse algoritmo, para cada mensagem a ser confirmada é gasto um número de mensagens correspondente ao número de nós da rede, logo, quando muitas mensagens em um tempo curto devem ser confirmadas, esta técnica desgastaria muito a rede.

4.2.1.2. Tempo lógico

Diferentemente do tempo físico, o tempo lógico define somente uma ordem parcial, isto é, um timestamp lógico aconteceu antes ($\rightarrow$), aconteceu depois ($\leftarrow$) ou é correlacionado ($\|$) a um segundo *timestamp*. Além do mais, tempo lógico não fornece o tempo decorrido entre dois eventos.

O conceito de tempo/relógio lógico foi descrito por Lamport em [LAM 78]. Baseia-se em números de seqüência que permitem verificar a ordem dos eventos. O número de seqüência pode ser um relógio local ou um contador de mensagens. O número de seqüência corrente em um nó é sempre o maior número de seqüência recebido de outro nó mais um.

Quando um nó emite uma mensagem com *timestamp 60*, os outros nós podem garantir que este evento ocorreu após quaisquer eventos com *timestamp 59* ou menor, antes de quaisquer eventos com *timestamp 61* ou maior, e não pode afirmar entre dois eventos com mesmo *timestamp* (concorrentes).

Esta solução de ordenação se mostra eficiente em sistemas distribuídos [LAM 78]. Entretanto, em rede de sensores, onde fenômenos do mundo real são capturados, pode ser muito interessante determinar em que momento cada evento ocorreu e qual o tempo decorrido entre um evento e outro. Uma vez que tratamos de RASSF, e sobretudo da interpretação dos

eventos capturados do mundo físico, não podemos simplesmente ignorar estas informações importantes que poderiam ser necessárias a diversas aplicações.

4.3. Considerações Finais

Esse capítulo apresentou os conceitos de interpretação de contexto e ordenação de eventos. Nele percebemos que quando os eventos são correlacionados semanticamente é necessária uma interpretação, e quando os eventos também possuem uma correlação temporal é necessária, ainda, uma ordenação dos eventos para se compreender a situação do ambiente. Foram apresentadas, nesse capítulo, soluções de interpretação de contexto e ordenação de eventos presentes na literatura, que de fato, não aproveitam as características das RASSFs.

Abordados os aspectos da interpretação e ordenação e os conceitos dos capítulos anteriores, conclui-se aqui todo o embasamento teórico necessário para um bom entendimento das soluções propostas, descritas no próximo capítulo.

5. Solução de Interpretação de Contexto e Algoritmo de Ordenação de Eventos Propostos

O intuito de nosso projeto, como visto nos capítulos anteriores, é oferecer suporte à situação de emergência tanto em tempo real quanto a posteriori. Logo, a arquitetura adotada deverá ser iferente das arquiteturas vistas no Capítulo 3. Como conseqüência, algoritmos e protocolos que realizam a comunicação e coordenação também serão. A arquitetura dever ser diferente, pois os eventos devem ser roteados ao atuador bem como para o sink. Deve-se rotear ao atuador para que este verifique se é uma situação que implique em uma atuação e deve-se ser roteado ao sink para que aplicação de monitoramento armazene e exiba os dados coletados.

Nas seções que se seguem são descritas as soluções de interpretação de contexto e ordenação de eventos propostas neste trabalho; logo após, são apresentados e discutidos os resultados obtidos por simulação de ambas as soluções.

5.1. Uma Solução de Interpretação de Contexto para RASSF

Quando pensamos em interpretação de contexto, surge a primeira questão relacionada sobre em que entidade da RASSF deveria ser realizada a interpretação. As entidades candidatas a esta tarefa seriam o sink, o atuador ou um nó agregador. Pelo o que foi discutido na seção 3.7, pode-se perceber que a interpretação realizada de forma distribuída nos atuadores traz vantagens sobre a interpretação centralizada feita no sink.

A utilização de nós sensores agregadores para interpretação de mensagens não é uma boa solução quando temos atuadores disponíveis, pois os atuadores possuem melhores capacidades de processamento e memória que os nós sensores, permitindo que a interpretação seja mais complexa. Com melhores capacidades de transmissão e energia, os atuadores evitam o desgaste dos nós sensores. Além disso, o uso de nós agregadores implicaria em procedimentos que seriam desnecessários quando utilizamos o atuador, como: processo de

seleção dos próximos agregadores e envio de comandos de atuação ao atuador. Portanto, o atuador é a entidade mais apropriada para a interpretação em RSSF.

A solução apresentada neste trabalho para interpretação de contexto em RASSF é ilustrada na Figura 5.1. A interpretação de contexto é descentralizada e baseada em regras (desde de regras simples até complexas, dependendo da aplicação). Uma vez que os contextos sejam interpretados, eles podem engatilhar comandos de atuação nos atuadores.

Exemplos de interpretação de contexto: Fernando está no setor A, sala 3; Temperatura do setor B é 60ºC; Existe fogo no setor C; A asa da aeronave está quebrada por causa de uma rachadura causada pelo aumento de pressão na área, etc. A forma com que as regras são implementadas nos atuadores (esquemas XML) é detalhada em [MIC 04]

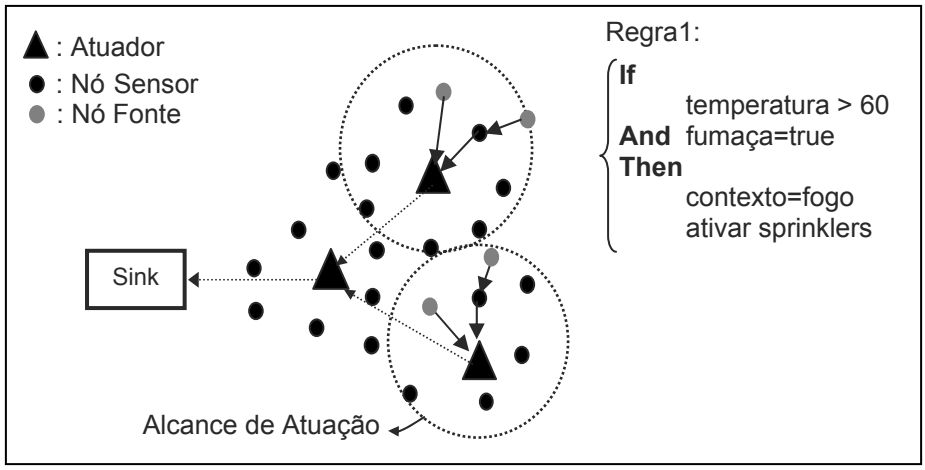

Figura 5.1. Solução de interpretação de contexto proposta neste trabalho para RASSF.

Na solução utilizada no projeto, o método *Publish/Subscribe* baseado em tópicos é usado pelos atuadores para publicar seus contextos e pelo sink e outros atuadores para subscreverem contextos nos quais estão interessados. Este trabalho não abrange este tema sendo necessário consultar [MIC 04] para compreender como notificações são enviadas aos subscritores e tópicos são implementados.

5.2. Ordering by Confirmation (OBC) - Um Algoritmo de Ordenação de Eventos Ciente de Energia e de Baixa Latência

Em geral, somente eventos gerados por um subgrupo de sensores da rede devem ser entregues em ordem temporal. Quando utilizamos RASSF, esta já possui tais subgrupos compostos pelo atuador e os nós sensores associados a ele. Logo, a ordenação dos eventos fica restrita a estes subgrupos, onde todos os subgrupos ordenados compõem uma rede inteira ordenada. Algoritmos desenvolvidos para RSSF e sistemas distribuídos não aproveitam as capacidades adicionais fornecidas pelos nós atuadores. O algoritmo OBC proposto aproveita os nós atuadores para realizar a ordenação de eventos de forma eficiente, rápida e ciente de energia.

Antes de descrevermos o algoritmo devemos destacar que, assim como os algoritmos descritos na seção, este também necessita de uma pré-configuração existente na rede de sensores. A pré-configuração trata do sincronismo de relógios físicos, da descoberta de rotas/caminhos e da adoção de canais FIFO entre todos os nós da rede. Cada uma destas pré-configurações são explicadas a seguir para uma melhor compreensão do funcionamento do algoritmo proposto.

5.2.1. Sincronismo

A primeira necessidade do algoritmo é a utilização de sincronismo de relógio físico entre os nós sensores e a entidade responsável pela coleta e avaliação cronológica das mensagens remetidas. Esta entidade pode ser um atuador, um nó agregador ou ainda o próprio sink. Uma vez que RASSFs são o alvo considerado neste trabalho para o monitoramento de condições críticas, porque possibilitam reação automática e em tempo real aos eventos coletados, a entidade que possui sincronização de relógios com os nós sensores é o atuador. Isto se dá, novamente, porque o atuador possui melhores capacidades de processamento, comunicação sem fio e uma bateria de maior duração. Nessa arquitetura de rede, os nós atuadores possuem sincronismo entre eles, e cada atuador possui sincronismo com os nós sensores dentro de sua área de atuação.

5.2.2. Algoritmo de Roteamento

A segunda necessidade é a descoberta de rotas ou caminhos entre os nós sensores da rede e o atuador. Para isto, utiliza-se o algoritmo de roteamento PEQ [BOU 04], este algoritmo foi

desenvolvido por um membro do laboratório LRVNet , tendo como características principais ser tolerante a falhas e de baixa latência. O PEQ é capaz de construir uma árvore de saltos *(hops)* que fornece os caminhos de cada nó sensor da rede até o atuador e também possibilita a reconfiguração desse caminho caso algum nó falhe. A Figura 5.2 ilustra dois estados de uma rede de sensores sem fio: (a) estado inicial da rede de sensores sem fios sem a configuração de caminhos; (b) estado final após a utilização do algoritmo PEQ, resultando na configuração dos caminhos.

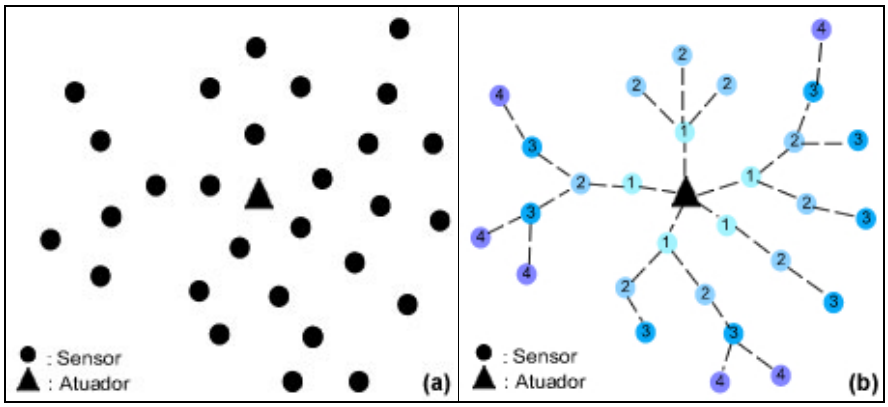

Figura 5.2. (a) Nós dispersos no ambiente sem configuração de caminhos. (b) Configuração dos caminhos obtidos através do algoritmo PEQ.

5.2.3. Canais FIFO

A adoção de canais FIFO entre todos os pares de nós da rede assegura que duas mensagens geradas em tempo diferentes e que percorram o mesmo caminho cheguem ordenadas temporalmente.

5.2.4. Descrição do algoritmo OBC

O algoritmo proposto assume que ao receber uma mensagem, será confirmado que não existe nenhuma mensagem anterior a esta em trânsito, através de mensagens de confirmação enviadas pelos nós dos pontos mais extremos dos caminhos (chamados de nós folhas). Uma vez que exista uma mensagem atrasada em trânsito na rede, esta chegará primeira do que a mensagem de confirmação enviada ao extremo do seu caminho, pois o caminho é único e a propriedade dos canais FIFO assegura que a mensagem de confirmação jamais ultrapassará a mensagem atrasada.

O algoritmo ainda utiliza *buffers* que possibilitam confirmar mais do que uma mensagem por vez, ou seja, quando uma mensagem chega e são requisitadas mensagens de confirmação a esta mensagem, pode ocorrer de chegar mais uma nova mensagem que sendo armazenada no

buffer é confirmada junto àquela que havia chegado anteriormente. A breve introdução feita neste parágrafo fornece apenas uma visão geral, e pode ser melhor compreendida a partir da explicação dos parágrafos subseqüentes.

O algoritmo pode ser melhor entendido através de três exemplos simples, conforme Figuras 5.3, 5.4 e 5.5, que representam o comportamento do algoritmo em cada caso. O primeiro representa o comportamento mais básico, onde somente uma mensagem é gerada. Este primeiro caso servirá de base também para o entendimento dos casos seguintes. Seguimos, então, para o primeiro caso. Uma vez configurada a rede de sensores sem fios, uma mensagem *m0* gerada por um nó sensor (indicado na Figura 5.3 (a) com o número *14*) é enviada pelo caminho até o nó atuador, onde o atuador a insere no buffer *1*.

Após isso, o atuador envia uma mensagem de requisição de confirmação temporal para os nós folhas (nós das extremidades dos caminhos). O sinal enviado pelo atuador é suficientemente forte para atingir os nós folhas em um único salto, conforme Figura 15 (b). Os nós que não são folhas receberão também a mensagem enviada pelo atuador, no entanto eles a descartarão.

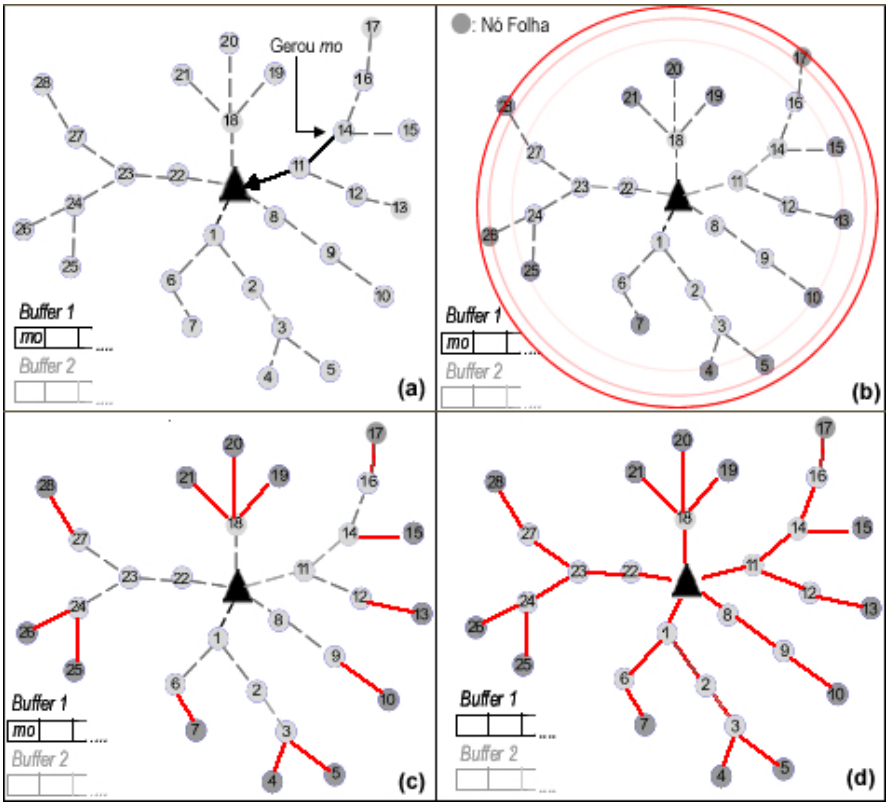

Figura 5.3. Exemplo do funcionamento do algoritmo proposto para o caso 1.

Logo após, cada nó folha remeterá uma mensagem de confirmação temporal em direção ao atuador, conforme ilustrado na Figura 5.3 (c). Nessa mesma figura, os caminhos já

percorridos pela mensagem de confirmação são representados por linhas sólidas em vermelho. Com o intuito de economizar na quantidade de mensagens geradas, cada nó do caminho só remeterá uma mensagem de confirmação caso já tenha recebido de todos os nós anteriores a ele no caminho (nós filhos), por exemplo, o nó *18* somente enviará a mensagem de confirmação ao atuador caso tenha recebido dos nós *21*, *20* e *19*. O atuador ao receber mensagens de confirmação de todos os seus vizinhos próximos (conforme Figura 5.3 (d), *1*, *8*, *11*, *18* e *22*) remove a mensagem do buffer *1* e a entrega a aplicação. Existem ainda algumas considerações adicionais a respeito do algoritmo destacados nos próximos casos.

O segundo caso demonstra a situação onde mais de uma mensagem poderia ser confirmada num único passo de confirmação. No momento em que se aguardam as mensagens de confirmações referentes a uma mensagem *m0,* uma mensagem *m1* pode chegar ao atuador oriunda de algum outro nó sensor por um caminho ainda não percorrido pelas mensagens de confirmação temporal de *m0*, conforme Figura 5.4 (b).Sendo assim, esta mensagem é inserida no buffer *1* e ordenada de acordo com o *timestamp*. A mensagem é inserida ordenada pelo *timestamp* no buffer *1,* porque a mensagem *m0* poderia ter sido gerada posteriormente a *m1*, no entanto o atraso da mensagem *m1* na rede implicou na chegada de *m0* primeiro. Logo após chegarem às mensagens de confirmação, as mensagens serão entregues ordenadamente a aplicação. Neste caso, nenhum tráfego é gerado para a confirmação temporal da mensagem *m1*.

O terceiro caso demonstra uma situação onde não foi possível que uma determinada mensagem fosse confirmada no mesmo passo de confirmação. No momento em que se aguarda as mensagens de confirmações referentes a *m0*, uma mensagem *m1* pode chegar ao atuador oriunda de algum nó sensor. Diferente do segundo caso, alguns vizinhos do atuador podem já ter repassado suas mensagens de confirmação e a espera ser conseqüência de outros vizinhos do atuador, como pode ser visto na Figura 5.5 (c) . Caso a mensagem *m1* seja oriunda deste vizinho, que ainda não repassou sua mensagem de confirmação, o algoritmo processa como no segundo caso. Caso contrário, a mensagem *m1* chegou por um caminho que já havia sido percorrido pela mensagem de confirmação, e portando ela deve ser inserida no buffer *2*, conforme ilustrado na Figura 5.5 (d). O buffer *2* armazena mensagens que não podem ser confirmadas no mesmo passo de confirmação. Após esvaziado o buffer *1* e suas mensagens entregues a aplicação, mensagens de confirmação temporal são enviadas para

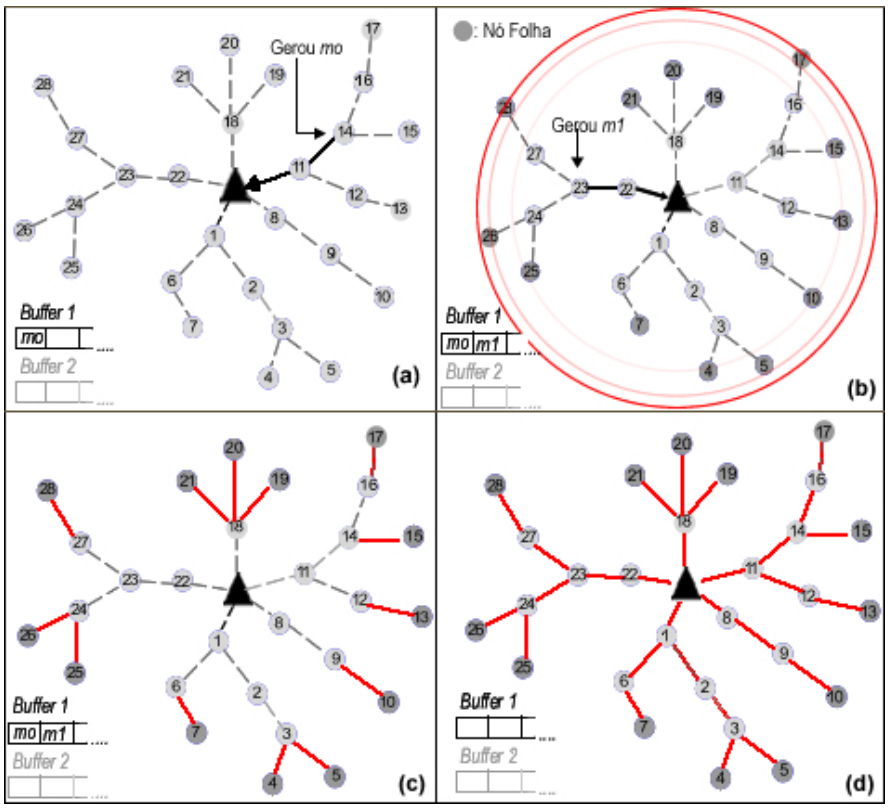

Figura 5.4. Exemplo do funcionamento do algoritmo proposto para o caso 2.

confirmação daquelas presentes no buffer *2*. Logo após chegarem as mensagens de confirmação, o buffer *2* é esvaziado e as suas mensagens são entregues a aplicação. Note que, neste caso, não é possível um aproveitamento do passo de confirmação temporal, como ocorreu no segundo caso, sendo necessário um novo passo de confirmação temporal, implicando em um tráfego adicional.

Perceba que diferentemente do *heartbeat*, onde todos os nós têm que enviar mensagens, o algoritmo proposto gasta menos energia da rede. No *heartbeat* um nó do caminho além de enviar sua mensagem de confirmação deve repassar mensagens de confirmação dos demais nós em seu caminho. Ao utilizarmos vários caminhos reduzimos a competição pelo meio físico, e ainda, sendo estes caminhos mínimos diminuímos a latência na entrega das mensagens quando comparado ao algoritmo TMOS. A utilização dos buffers permite confirmar várias mensagens de uma vez.

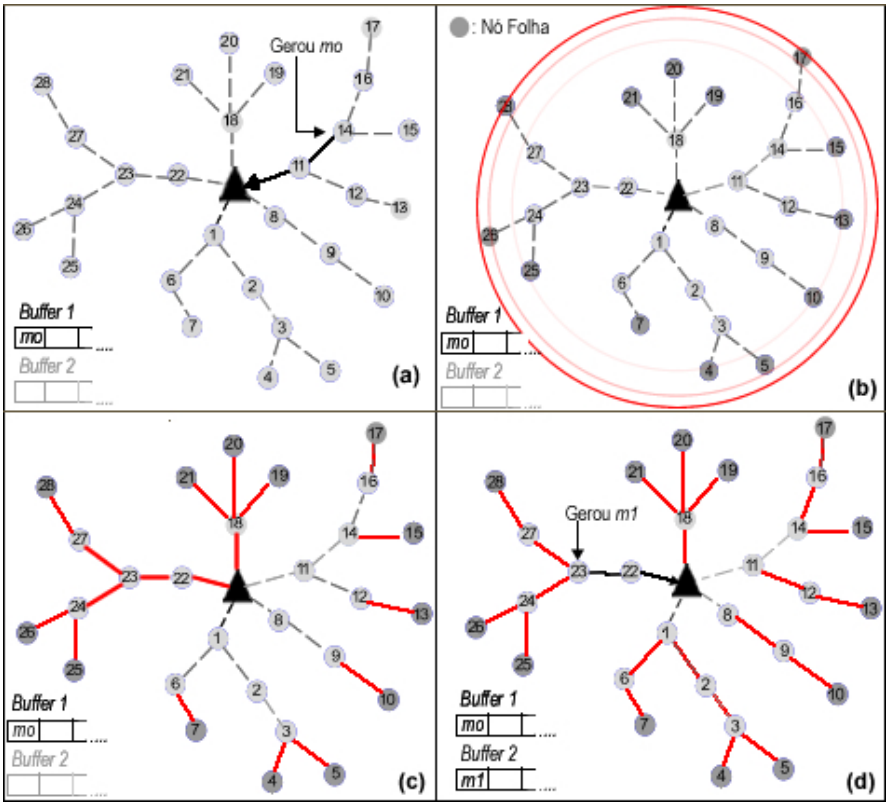

Figura 5.5. Exemplo do funcionamento do algoritmo proposto para o caso 3.

5.3. Experimentos de Simulações

Esta seção apresenta a análise de desempenho do algoritmo OBC proposto na seção 5.2 e da arquitetura de interpretação descentralizada proposta na seção 5.1 As simulações foram realizadas com o objetivo considerar diversas situações em que a rede de sensores possa estar envolvida. Para isto, vários parâmetros foram determinados aleatoriamente.

Os algoritmos aqui analisados e as simulações foram implementados utilizando a linguagem Java, conforme o Anexo deste trabalho onde se encontram as partes principais destes códigos. A simulação e os algoritmos foram implementados desta forma visando obter indicativos de desempenho e da viabilidade do algoritmo OBC e da interpretação descentralizada. Uma vez que os resultados dos algoritmos através destas simulações se mostraram satisfatórios, surgiu a necessidade de implementá-los num conceituado simulador de redes chamando NS-2 (Network Simulator) [NS2 06a]. Entretanto, por limitações descritas no capítulo 6, não foi possível concluir a implementação prevista ainda para este trabalho. Portanto, serão aqui apresentados os resultados de simulações obtidos dos algoritmos de ordenação de eventos e das soluções de interpretação implementados em Java.

5.3.1. Análise de Desempenho do Algoritmo OBC

Esta seção objetivando uma análise do algoritmo OBC apresenta o cenário de simulação, as métricas utilizadas e os resultados obtidos. Por razão de comparação, o algoritmo TMOS e o OBC foram implementados, como descritos nas seções 4.2.1 e 5.2, usando os mesmos cenários e parâmetros. Devido ao fato do algoritmo ser aplicado dentro de cada subgrupo, o anel lógico sempre é composto de todos os nós sensores - isto implica na diferença dos resultados obtidos neste trabalho e em [ROM 02].

5.3.1.1. Cenário de Simulação e Métricas

Conforme acabou de ser mencionado, a ordenação de eventos acontece dentro de cada subgrupo da RASSF, logo o cenário de simulação é um campo referente a um subgrupo de sensores da rede inteira. Estes sensores são dispersos aleatoriamente dentro deste campo, ou seja, cada nó sensor é disposto aleatoriamente a uma determinada distância do atuador, localizado no centro. Uma porcentagem dos nós sensores, chamados de nós produtores, são escolhidos para produzirem eventos, onde a escolha desses nós produtores é aleatória.

A taxa de dados de cada nó sensor escolhido para produzir evento também é determinada aleatoriamente. Embora os modelos de tráfego de dados de redes tradicionais como CBR (Constant Bit Rate), VBR (Variable Bit Rate, por exemplo, Poisson) têm sido utilizados para modelar o tráfego em RSSF, a utilização destes modelos é contestada na literatura devido ao fato de eles não considerarem as características e limitações das RSSFs [DEM 05]. Cada valor medido foi extraído de uma média de 100 simulações. Para a simulação de ambos os algoritmos, é assumido que os caminhos são perfeitos (i.e, os caminhos não quebram devido a falha de algum nó sensor).

Os parâmetros como: o tamanho da área do campo de simulação, o número de nós sensores e a porcentagem de nós produtores apresentam uma variação que medirá impacto de cada um deles nos algoritmos considerados de acordo com as métricas utilizadas. Para tanto, cada parâmetro é avaliado separadamente, isto é, ao variarmos um dos parâmetros os outros dois permanecem fixos. A tabela 5.1 apresenta os valores fixos destes três parâmetros, bem como de outros parâmetros utilizados.

Os algoritmos foram analisados baseando-se nas seguintes métricas:

1. *Latência na entrega de evento* – A latência é a quantidade de tempo decorrida desde a detecção do evento até a notificação à aplicação. Latência baixa é um requisito

importante para aplicações de monitoramento de condições críticas, onde ações devem ser tomadas no menor tempo possível, uma vez que patrimônio e vidas estão em jogo.

Tabela 5.1. Parâmetros de Simulação para a ordenação de eventos.

Parâmetros	Valores
Área de Simulação(m^2)	80x80
Número de nós sensores	100
Porcentagem de nós produtores	20
Taxa de envio de dados (eventos/s)	Aleatório
Tempo de simulação(s)	20
Alcance de Rádio (m)	10

2. *Dissipação de energia* – A dissipação de energia é importante porque os nós são dispositivos com fonte de energia restrita, e racionar seu uso aumenta o tempo de vida da rede. Além disso, a dissipação de energia pode ter impacto na latência.

3. *Uniformidade de gasto da rede* – Assim como a energia dissipada é importante, a uniformidade do gasto de energia por cada sensor da rede também é, pois o uso disforme de energia por cada sensor resultaria na degradação do serviço oferecido pela rede, uma vez que alguns sensores seriam sobrecarregados "morrendo" muito cedo por conseqüência disto.

Utilizamos a quantidade de mensagens transferidas em um único salto (*single-hop*) como uma estimativa para a energia gasta, para a latência na confirmação das mensagens e para a uniformidade de gasto da rede. Além disso, consideramos o tamanho das mensagens o mesmo para dois algoritmos.

5.3.1.2. Resultados obtidos

Para avaliar o desempenho energético dos algoritmos em redes com diferentes densidades de sensores foi medido, conforme Figura 5.6, a dissipação de energia da rede de sensores variando o número de nós sensores. Pode ser visto que a média de energia dissipada para a confirmação de uma mensagem é menor para o algoritmo OBC do que para o TMOS.

Para o algoritmo TMOS, quando o número de nós sensores aumenta, o tamanho do anel lógico também aumenta, e, conseqüentemente, a quantidade de mensagens necessárias para a confirmação temporal de cada mensagem torna-se ainda maior.

Já no algoritmo OBC, o aumento do número de nós sensores influi no gasto de energia. No entanto, o aumento de nós sensores aumentaria o número de nós produtores, geradores de mensagens, uma vez que o número de nós produtores é uma porcentagem do número de sensores. Dessa forma, embora ao aumentar o número de nós sensores, aumente o gasto de energia, este gasto é ainda compensado por mensagens que são aproveitadas no mesmo passo de confirmação. Para que compreenda melhor o que acaba se dito, é importante observar outras duas Figuras 5.7 e 5.8.

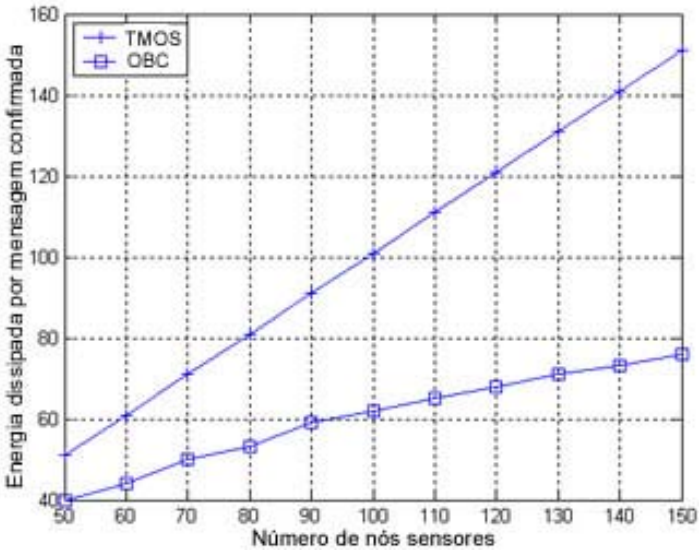

Figura 5.6.Gráfico de energia dissipada quando a densidade de nós sensores da rede varia.

A Figura 5.7 ilustra o número de passos de confirmação necessário em cada quantidade de nós sensores. Pode-se notar um aumento no número de passos de confirmação à medida que o número de nós sensores aumenta. O aumento do número de passos de confirmação implica num gasto de energia maior.

A Figura 5.8 ilustra o número de mensagens que conseguiram ser confirmadas junto a outras mensagens no mesmo passo de confirmação, à medida que a densidade da rede aumenta. Nessa figura, podemos notar que quando o número de nós sensores aumenta, o número de mensagens confirmadas sem qualquer custo de energia adicional também aumenta. Portanto, ao se aumentar o número de nós sensores, aumenta-se o número de passos de confirmação, e consequentemente o gasto de energia, entretanto esse gasto é compensado em parte pelo aumento do número de mensagens confirmadas sem qualquer custo de energia adicional.

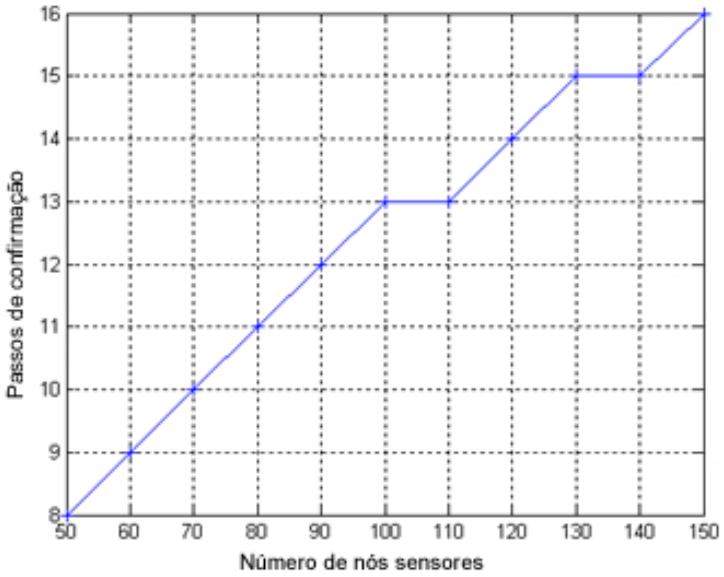

Figura 5.7. Número de passos de confirmação realizados para cada quantidade de nós sensores, utilizando o algoritmo OBC.

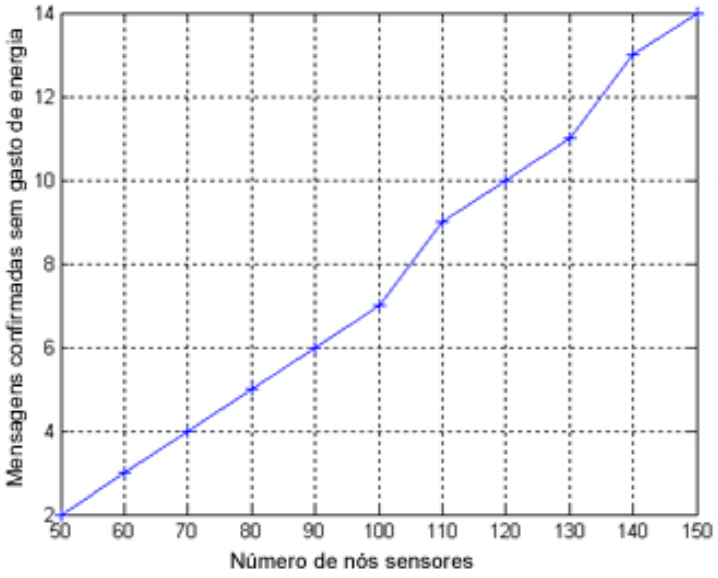

Figura 5.8. Número de mensagens confirmadas sem gasto adicional de energia para cada quantidade de nós sensores, utilizando o algoritmo OBC.

A Figura 5.9 mostra a dissipação energética dos algoritmos quando o número de nós produtores varia, e consequentemente, a taxa de eventos a serem gerados varia. Pode-se notar que o algoritmo TMOS mantém sempre um valor fixo para a média de energia dissipada para cada mensagem confirmada quando o número de nós produtores aumenta. Isto se explica porque para cada mensagem a ser confirmada, no algoritmo TMOS, é sempre gasto um número de mensagens igual ao número de nós sensores da rede mais um.

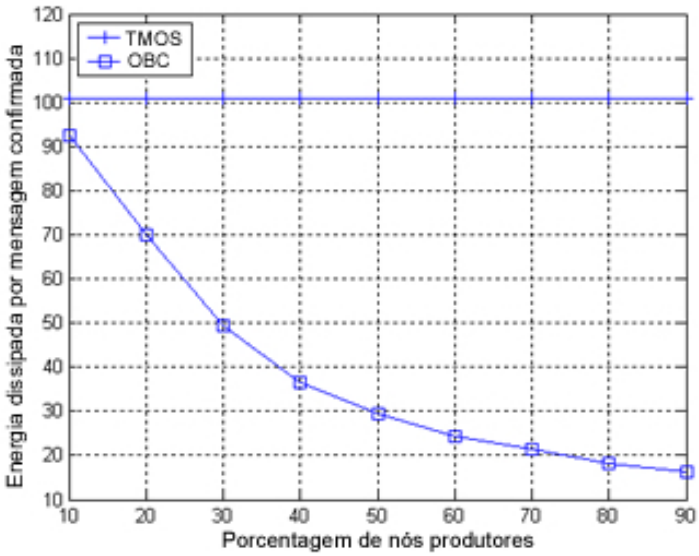

Figura 5.9. Gráfico de energia dissipada quando a porcentagem de nós produtores da rede varia.

Por outro lado, para o algoritmo OBC, quando mais mensagens são geradas, a possibilidade de confirmação temporal destas mensagens de uma vez aumenta, com isso reduz-se a energia dissipada para cada mensagem a ser confirmada.

As Figuras 5.10 e 5.11 ilustram a energia dissipada e a latência quando a área de simulação varia. Nota-se que o algoritmo TMOS mantém um valor fixo para ambos: energia dissipada e a latência. Para o algoritmo OBC, quando o tamanho da área de simulação aumenta, a energia dissipada e a latência são também aumentadas como conseqüência do aumento do tamanho dos caminhos. Entretanto, os resultados do algoritmo OBC estarão sempre abaixo dos obtidos pelo TMOS.

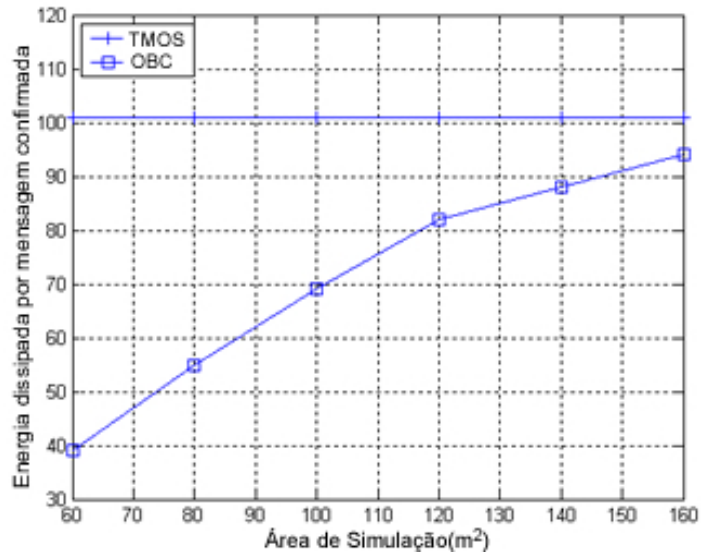

Figura 5.10. Gráfico de energia dissipada quando a área de simulação varia.

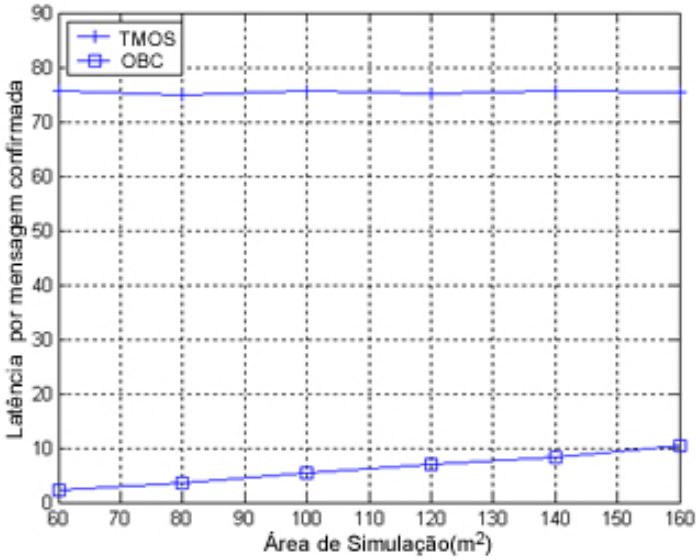

Figura 5.11. Gráfico da latência na confirmação de mensagem quando a área de simulação varia.

As Figuras 5.12 e 5.13 ilustram a latência ao variarmos, respectivamente, a densidade da rede e a taxa de eventos. A Figura 5.12 mostra que ao aumentarmos o número de nós da rede, a latência média para confirmação de uma mensagem no algoritmo TMOS aumenta. Notamos ainda na Figura 5.12, que a latência do OBC comparada ao TMOS é muito menor.

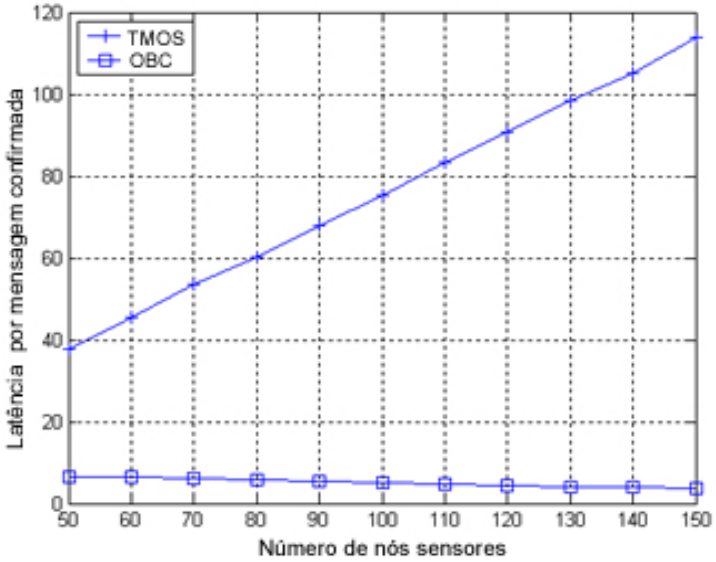

Figura 5.12. Gráfico da latência na confirmação de mensagem quando a densidade da rede varia.

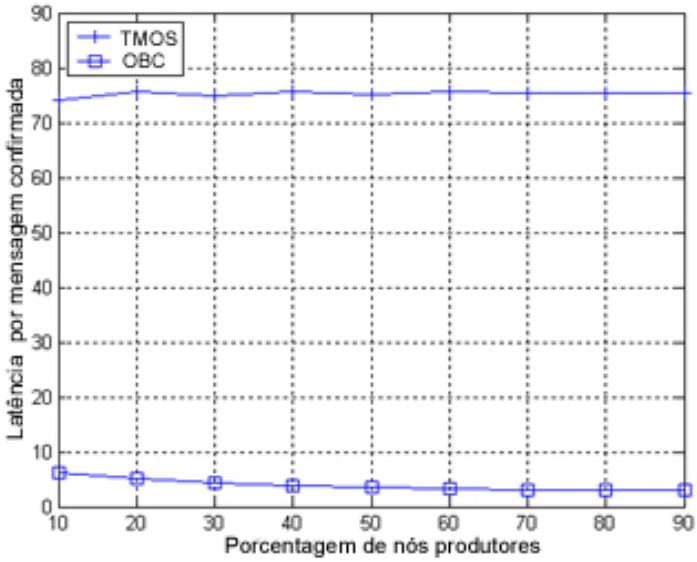

Figura 5.13. Gráfico da latência na confirmação de mensagem quando a porcentagem de produtores varia.

No algoritmo TMOS duas mensagens para um evento são transmitidas em caminhos opostos, a latência da confirmação deste evento é dominada pelo maior dos caminhos (exceto quando esta exatamente no meio do anel). A inserção de nós aumenta o tamanho do anel e conseqüentemente a latência da rede. No algoritmo OBC a latência é dominada pelo maior caminho de confirmação da mensagem, adicionando a este valor o próprio envio do evento ao atuador. A inserção de nós não aumenta o tamanho dos caminhos, isto só acontece se aumentarmos o tamanho do campo monitorado, ela aumenta a quantidade de caminhos que ao serem percorridos em paralelo não afetam a latência da rede.

Na simulação ilustrada pela Figura 5.13, ao aumentarmos a porcentagem de produtores o algoritmo TMOS não aumentamos a latência, já no algoritmo OBC a latência diminui devido ao fato de mais mensagens serem confirmadas por vez.

As Figuras 5.14 e 5.15 mostram a uniformidade de dissipação de energia para os algoritmos OBC e TMOS, respectivamente. Pode-se notar pelos gráficos que o TMOS tem uma dissipação de energia mais uniforme que o OBC. Entretanto, ainda que o OBC tenha uma maior variância de dissipação de energia por cada nó, ele é capaz de fornecer um tempo de vida maior a rede de sensores do que o TMOS, porque o nó sensor do OBC com dissipação máxima de energia é menor que o que tem dissipação mínima no TMOS.

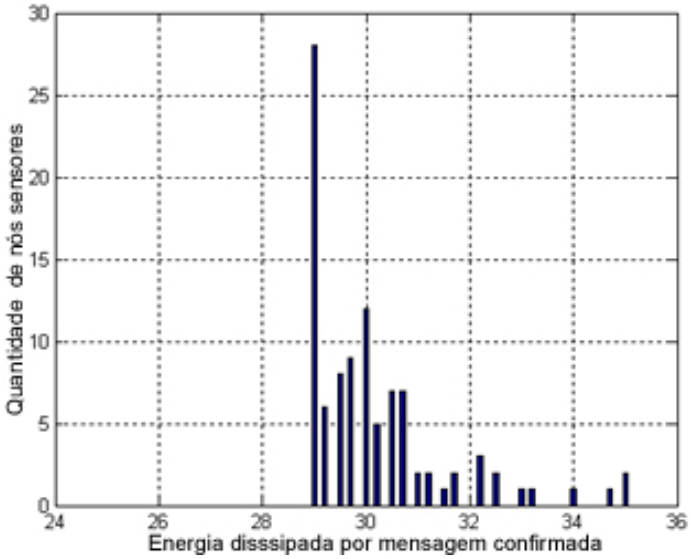

Figura 5.14. Histograma da uniformidade de energia dissipada pelos nós sensores ao utilizar o algoritmo OBC.

Dos experimentos realizados pode-se concluir que o algoritmo OBC, em RASSFs, consegue economizar mais energia, e entregar as mensagens ordenadas de forma mais rápido em situações onde a densidade da rede, a taxa de eventos e a dimensão da rede são aumentados.

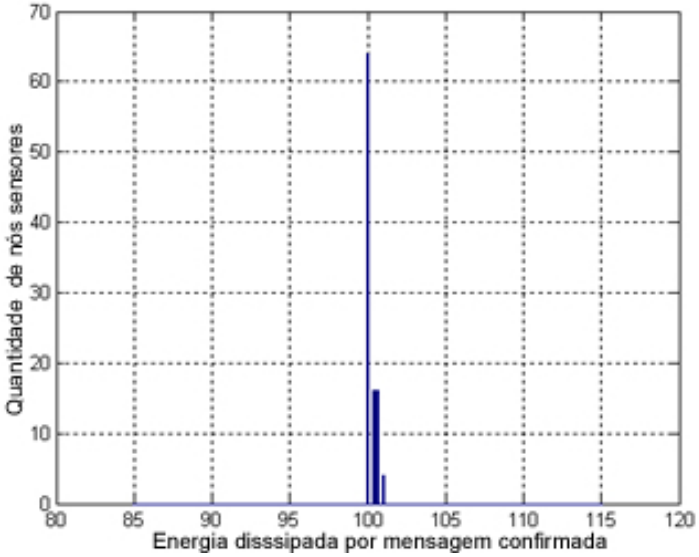

Figura 5.15. Histograma da uniformidade de energia dissipada pelos nós sensores ao utilizar o algoritmo TMOS.

5.3.2. Análise de Desempenho da Arquitetura de Interpretação Descentralizada

Esta seção objetivando uma análise da arquitetura de interpretação proposta, apresenta o cenário de simulação, as métricas utilizadas e os resultados obtidos. Por razão de comparação, implementamos o algoritmo OBC utilizando a interpretação centralizada e descentralizada, descritas na seção 3.7, usando quase os mesmos cenários e parâmetros de simulação anteriormente definidos.

5.3.2.1. Cenário de Simulação e Métricas

O cenário de simulação a ser descrito aqui é muito similar ao já utilizado; entretanto, existem algumas diferenças. Em função disso, enunciaremos somente o que for distinto.

Desta vez, o cenário de simulação não é restrito a somente um subconjunto de nós sensores, onde um único atuador gerencia. O cenário é a própria RASSF subdividida em subgrupos de nós sensores no qual, em cada subgrupo existe um atuador responsável pela atuação na área. E ainda, o tamanho da área de simulação, além de maior, não varia e contém, agora, 4 atuadores. A tabela 5.2 apresenta os parâmetros desta simulação. As métricas avaliadas aqui são, somente, a *latência na entrega de evento* e a *dissipação de energia*.

Tabela 5.2. Parâmetros de Simulação para a interpretação de contexto.

Parâmetros	Valores
Área de Simulação(m^2)	160x160
Número de nós sensores	100
Porcentagem de nós produtores	20
Taxa de envio de dados (eventos/s)	Aleatório
Alcance de Rádio (m)	10
Tempo de Simulação	20
Número de nós Atuadores	4

5.3.2.2. Resultados obtidos

Nas Figuras 5.16 e 5.17, percebemos um melhor desempenho energético da arquitetura descentralizada proposta, mesmo quando o número de nós sensores da rede e a taxa de eventos aumentam. Da mesma forma, percebemos nas Figuras 5.18 e 5.19 uma latência mais baixa. Para ambas as métricas analisadas, o motivo para o melhor desempenho da interpretação descentralizada comparada à centralizada é o mesmo, deve-se ao fato que

quando a interpretação é realizada de forma descentralizada os caminhos para transmissão das mensagens são menores implicando num consumo energético menor e numa latência mais baixa.

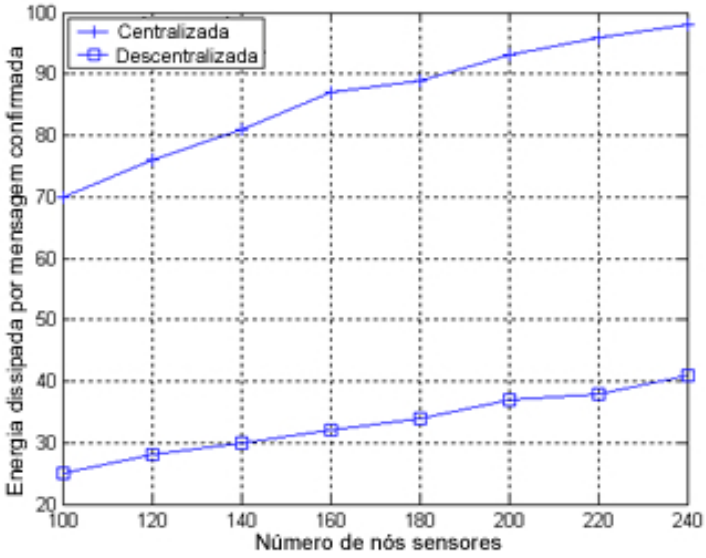

Figura 5.16. Gráfico de energia dissipada quando a densidade de nós sensores varia.

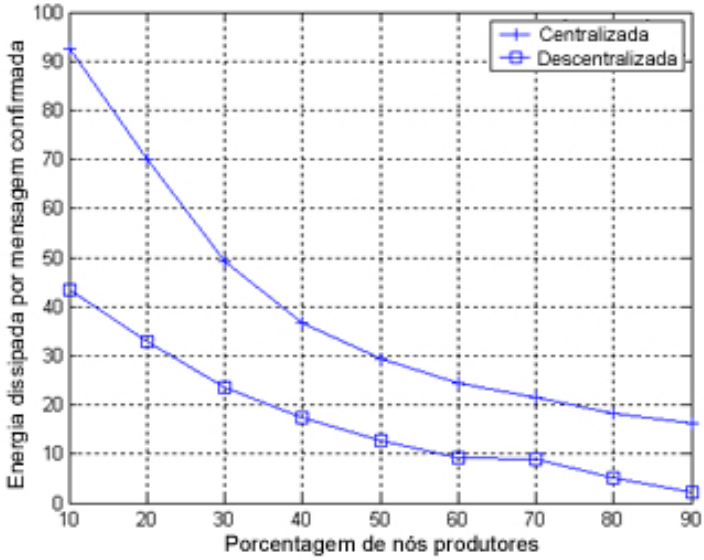

Figura 5.17. Gráfico de energia dissipada quando a porcentagem de nós produtores varia.

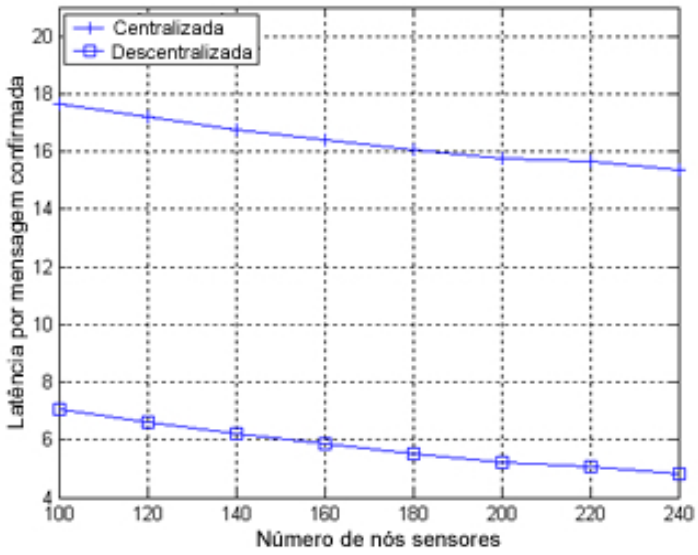

Figura 5.18. Gráfico da latência na confirmação de mensagem quando a densidade da rede varia.

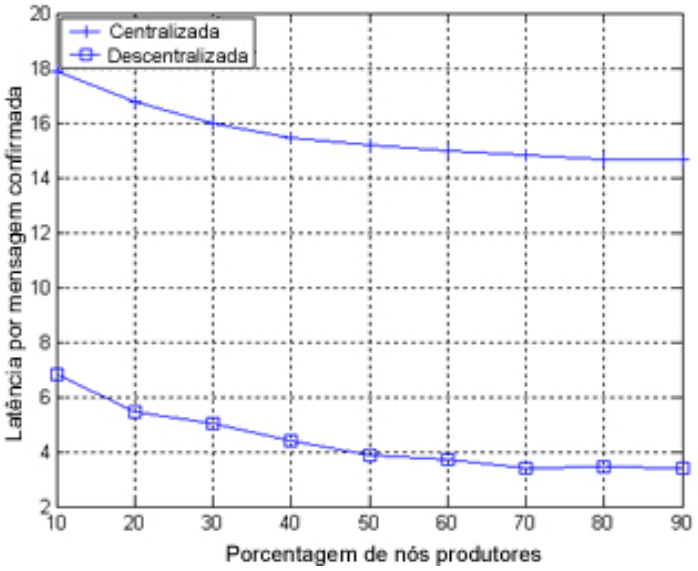

Figura 5.19. Gráfico da latência na confirmação de mensagem quando porcentagem de produtores varia.

5.4. Atendimento aos Requisitos Não Funcionais

O desenvolvimento de todo o projeto, não só deste trabalho, buscou atender a todos os requisitos principais destacados. A Tabela 2.1, apresentada a seguir, descreve as medidas adotadas para atender cada requisito não funcional enunciado na seção 2.2.

Tabela 5.3. Medidas adotadas para atender os RNFs destacados.

Requisitos Não Funcionais (RNFs)	**Medidas adotas para atender os RNFs**
Confiabilidade	Requisito atendido através do algoritmo PEQ[1], que realiza entrega de evento via multi-caminhos, ou seja, na entrega de um evento, mais de um caminho é adotado em virtude da possível quebra do caminho. E através da Ordenação de eventos.
Segurança	Requisito atendido através do uso das RASSFs. Quando numa situação de falha no processo de interpretação, devido a uma especificação incorreta por parte do analista de segurança, o sistema deve atuar via atuadores mesmo em detrimento de prejuízos financeiros, pois deve priorizar a segurança destes ambientes críticos. Exemplo, a atuação via dispersores de água automáticos em uma sala de computadores no caso de incorreta condição de incêndio.
Eficiência e Desempenho	Requisito atendido através do uso do algoritmo PEQ e da RASSF. O algoritmo PEQ, assim como a RASSF, fornece latências baixas em resposta a eventos capturados, e ainda, a RASSF aglomera eventos de forma descentralizada, reduzindo a carga circulante na rede.
Causalidade	Requisito atendido através da ordenação de eventos e da interpretação das mensagens geradas pelos eventos. Trata-se do requisito principal para Ordenação dos eventos, principal objetivo deste trabalho.

[1] PEQ– Algoritmo de roteamento tolerante a falhas e de baixa latência para RSSFs [BOU 04].

6. Conclusões

Este capítulo descreve as limitações encontradas, contribuições geradas com a realização deste trabalho, bem como trabalhos futuros e conclusões finais.

6.1. Limitações Encontradas

Conforme já foi discutido na seção 5.1, os resultados de simulações dos algoritmos OBC e TMOS, assim como das soluções de interpretação centralizada e descentralizada, apresentados neste trabalho, foram implementados em Java. Estes resultados impulsionaram a escrita e a publicação de vários artigos, relacionados na seção 6.2.1, cujas aceitações, apesar de motivarem cada vez mais os autores a progredirem na solução, acabaram impactando no tempo exíguo de implementação no NS-2.

A implementação no NS-2 objetiva um estudo mais complexo dos algoritmos avaliados no capítulo 5. Para isso, o NS-2 foi muito estudado pelo autor e os algoritmos foram parcialmente implementados. Entretanto, algumas dificuldades no uso do simulador não puderam ser superadas a tempo. No início de Abril de 2006, o trabalho de implementação dos algoritmos no NS-2 foi retomado por outro de aluno de mestrado do LRVNet que deverá concluí-lo como parte de seu trabalho.

Para obter uma melhor noção do desempenho do algoritmo OBC, seria interessante compará-lo a outros algoritmos de ordenação de eventos. Entretanto, os algoritmos de ordenação de eventos, assim como o TMOS, não possuem trabalhos publicados de desempenho em ambientes de simulação, tal qual o NS-2, implicando na necessidade de implementá-los em sua totalidade para um correto comparativo de resultados.

6.2. Contribuições

Este trabalho gerou contribuições ao projeto geral de muitas formas:

- Ao introduzir ao projeto o conceito e o uso das RASSF, tornou-se possível ir além da simples monitoração do ambiente, sendo possível atuar de forma ativa sobre a situação emergencial.

- Definiu uma solução descentralizada de interpretação de contexto que retirou o mecanismo de interpretação de contexto, antes realizado de forma centralizada, do sink, colocando-o dentro da rede, especificamente dentro de cada atuador.
- Definiu um algoritmo para a ordenação das mensagens de monitoramento, capaz de descobrir a correta ordem em que os eventos aconteceram no ambiente. Isto possibilitou a correlação de eventos e uma interpretação de contexto mais precisa e complexa.
- Aumentou a experiência do grupo com simulações no NS-2.

6.2.1. Escrita de Artigos e Publicações

1. BOUKERCHE, Azzedine; SILVA, Fernando H. S.; ARAUJO, R. B. "*A Low Latency and Energy Aware Event Ordering Algorithm for Wireless Actor Sensor Network*s". In: 8-th ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems, 2005, Montreal. ACM Proceedings of the 8-th ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems, 2005. v. 8. p. 1-8.
2. BOUKERCHE, Azzedine; SILVA, Fernando H. S.; ARAUJO, R. B. "*Ordenação em Redes de sensores sem Fio para Aplicação em Ambientes Físicos Cientes de Contexto*". In: 4th International Information and Telecommunication Technologies Symposium, 2005, Florianopolis. CD of the 4th International Information and Telecommunication Technologies Symposium, 2005.
3. BOUKERCHE, Azzedine; SILVA, Fernando H. S.; ARAUJO, R. B. "An Event Ordering Algorithm that Extends the Lifetime of Wireless Actor and Sensor Networks". [Artigo submetido para aprovação/publicação em revista em 30 de Janeiro de 2006].
4. BOUKERCHE, Azzedine; SILVA, Fernando H. S.; ARAUJO, R. B. "Wireless Actor and Sensor Networks Context Interpretation for the Emergency Preparedness Class of Applications". [Artigo submetido para aprovação/publicação em revista em 15 de fevereiro de 2006]

6.3. Trabalhos Futuros

Como continuidade ao trabalho aqui iniciado, as seguintes atividades ainda deverão ser realizadas:

1. A utilização do algoritmo OBC e da interpretação de contexto para aplicações em aeronaves em situações de ensaios de vôo e solo. Projeto do LRVNet em parceria com a Embraer.
2. A implementação e simulação completa do algoritmo OBC no NS-2 .
3. Interpretação de contextos mais complexos, tais como "porque" (porque a asa do avião rachou?) e "como" (como se alastrou o fogo?).
4. Integração de redes de sensores e algoritmos aqui descritos com simulações em conformidade com o HLA e o Kit RTI [RTI 06] para aplicações de treinamento.

6.4. Conclusões Finais

Rede de atuadores e sensores sem fio é um campo de pesquisa emergente com muitas aplicações nas mais diversas áreas. Aplicações como o monitoramento de ambientes sujeitos a situações de emergência podem ser bastante beneficiadas do uso da tecnologia de rede de atuadores e sensores sem fios, uma vez que esta possibilita uma reação automática e em tempo real aos eventos capturados pelos nós sensores.

Em ambientes sujeitos a situações de emergência, contextos podem ser capturados e interpretados para ajudar no resgate, na prevenção, e no combate contra fogo, explosões e vazamentos de gases tóxicos, etc. Este trabalho apresentou uma solução para interpretação de contexto utilizando a RASSF, em que atuadores, providos de melhores capacidades, são utilizados para agregar as mensagens de monitoramento, eliminar ambigüidades e redundâncias e realizar a interpretação contextual. Cada atuador da RASSF foi configurado com regras determinadas pela aplicação na fase de configuração da rede. Nossa solução foi comparada à solução de interpretação de contexto centralizada, em que todas as mensagens de monitoramento são remetidas ao sink. Resultados de simulação mostraram que quando a interpretação de contexto é realizada de forma descentralizada, os caminhos para transmissão das mensagens são menores implicando num consumo energético menor e numa latência mais baixa.

Redes de atuadores e sensores sem fios são sistemas onde não se pode assumir que a ordem dos eventos coletados dos diferentes nós sensores reflita a ordem em que eles ocorreram. Isto se deve ao fato de que as RASSF são sujeitas a atrasos de rede provenientes da natureza da capacidade de comunicação tipicamente limitada, compartilhada por nós de dentro da mesma área de comunicação. Portanto, a ordenação de eventos é um requisito importante para garantir à correta interpretação de eventos correlacionados no tempo.

Este trabalho descreveu o OBC, um algoritmo de ordenação de eventos para RASSF que, ao unir a melhor capacidade do nó atuador com um esquema de roteamento das mensagens monitoradas, torna-se um mecanismo de ordenação ciente de energia e de baixa latência. O algoritmo proposto foi simulado, avaliado e comparado ao algoritmo TMOS, visando mostrar que o OBC consegue ter baixa latência e economizar mais energia tanto em situações em que a densidade da rede aumenta, como quando o tráfego da rede aumenta.

A utilização conjunta de atuadores realizando a interpretação de contexto com a ordenação de eventos além de possibilitar que contextos mais complexos sejam avaliados, podem ser uma boa alternativa para aplicações de monitoramento de ambientes sujeito a situações de emergência.

Referências Bibliográficas

[AKY 02] Akyildiz, I.F.; Su, W.; Sankarasubramaniam, Y. "A Survey on Sensor Networks". IEEE Communications Magazine, p. 102-114, Agosto de 2002.

[AGR 00] Agre, J; Clare, L. "An Integrated Architecture for Cooperative Sensing Networks". Computer - IEEE Computer Society Press, 33(5): p.106–108, Califórnia, EUA, 2000.

[AHN 06] Ahn, S. and Kim, D. "System Proactive Context-Aware Sensor Networks". European Workshop on Wireless Sensor Networks (EWSN), Zurich, Switzerland, 13-15 de Fevereiro, 2006.

[AKY 04] Akyildiz, I.F; Kasimoglu, I.H. "Wireless sensor and actor networks: research challenges". Ad Hoc Networks, Elsevier, Volume 2, Issue 4, p. 351-367. Outubro de 2004.

[BOE 76] BOEHM, B., BROWN, J.R. and LIPOW, M. "Quantitative Evaluation of Software Quality". Proc. of 2nd International Conference on Soft. Eng. San Francisco, p.592-605. Outubro de1976.

[BON 00] Bonnet, P.; Gehrke, J.; Seshadri, P. "Querying the physical world". IEEE Pers. Communication, p. 10-15, NY, EUA, Outubro de 2000.

[BON 01] Bonnet, P., et al. "*Towards Sensor Database System".* Proceedings of the Second International Conference on Mobile Data Management, 2001.

[BOU 03] Boukerche, A. Cheng, X. And Linus, J. Energy-Aware Data-Centric Routing in Microsensor Networks. In MSWiM'03, September 19, 2003, San Diego, California, EUA. (2003).

[BOU 04] Boukerche, A., Pazzi, R. W N., Araujo, R. B. **"**A Fast and Reliable Protocol for Wireless Sensor Networks in Critical Conditions Monitoring Applications**"**. The 7th ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM'04). Venice, Italy, Outubro 4-6, 2004.

[BOU 05] Boukerch, A., Lopes, Araujo, R.B. "Recording and Playing 3D Collaborative Virtual Environment Simulations as Continuous Media for Critical Security Applications". 4th Internacional Information and Telecommunication Technologies Symposium (I2TS), Florianópolis, Brasil, Outubro de 2005.

[BUL 01] Bulusu, N.; Estrin, D.; Girod, L.; Heidemann, J. "Scalable coordination for wireless sensor networks: self-configuring localization systems". International Symposium on Communication Theory and Applications (ISCTA 2001), Ambleside, UK, July 2001.

[CER 01] Cerpa, A.; Elson, J.; Estrin, D.; Girod, L.; Hamilton, M.; Zhao, J. "Habitat Monitoring:Application Driver for Wireless Communications Technology". Proceeding of ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean, 3-5, Costa Rica, 2001.

[CHE 04] Chen, N.; Hong, J.; Jiang, X. E Wang K., Takayama, L., Landay, J. "Siren: Context-aware Computing for Firefighting". In: Pervasive' 2004, 2004.

[CHI 01] Chien-Chung Shen, Chavalit Srisathapornphat, And Chaiporn Jaikaeo. "Sensor Information Networking Architecture and Applications", IEEE Personal Communications", pag. 52-59. Agosto de 2001.

[CHO 00] Cho, S. H.; Chandrakasan, A. "Energy efficient protocols for low duty cycle wireless microsensor networks". "Procedings of the 33rd Annual Hawaii International Conference on System Sciences", Hawaii, Março de 2000.

[CHU 00] Chung, L.; nixon, B.; yu, E.; mylopoulos, J. "Non-Functional Requirements in Software Engineering". Kluwer Academic Publishers**,** 2000.

[DAR 04] DARPA Information Exploitation Office. "Command Post of the Future".Disponível: http://dtsn.darpa.mil/ixo/programdetail.asp?progid=18 [consulta:

Fevereiro de 2004].

[DEM 05] Demirkol, I., Alagoz, F., Delic, A. and Ersoy, C. "Wireless Sensor Networks for Intrusion Detection: Packet Traffic Modeling*"*. IEEE COMMUNICATIONS LETTERS, 2005.

[DEY 01] Dey, A.K. "Understanding and Using Context". Personal and Ubiquitous Computing**,** pags.4-7, Londres, Reino Unido, 2001.

[END 06] Endler, M. "Large scale body sensing for Infectious Disease Control". Sentient Future Competition on European Workshop on Wireless Sensor Networks. Zurich, Suíça, Fevereiro de 2006

[EST 02] Estrin, D. Sayeed, A. Srivastavaet, M. Disponível: http://nesl.ee.ucla.edu/tutorials/mobicom02 [consulta: Março de 2005].

[EST 99] Estrin, D; Govindan, R. Heidemann, J.; Kumar, S. "Next Century Challenges:Scalable Coordination in Sensor Networks". Proceeding of ACM International Conference on Mobile Computing and Networking (MOBICOM'99), 263-270, Washington, 1999.

[FAR 00] Farines, J. M., Fraga, J. S., Oliveira, R. S. "Sistemas de Tempo Real". 12ª Escola de Computação, IME-USP, São Paulo-SP, 24 a 28 de julho de 2000

[Hay 96] Hayton, R. "OASIS: An Open Architecture for Secure Interworking Services". Tese de pós-doutorado, University of Cambridge, 1996.

[HEI 03] Heideman, J., Silva, F., Estrin, D. "Matching Data Dissemination Algorithms to Application Requirements". Proceedings of the ACM SenSys Conference. Abril de 2003.

[HOF 93] Hoffner, Y. "A Survey of ordering algorithms for monitoring". Relatório técnico do projeto ANSA. Cambridge, Reino Unido. Maio de 1993.

[KAH 99] Kahn, J.; Katz, R.; Pister, K. "Next Century Challenges: Mobile Networking for Smart Dust". In Proceeding of ACM International Conference on Mobile Computing and Networking *(MOBICOM'99),* Washington 1999; 271-278

[KAS 89] Kastem O.; Langheinrich, M.; "First Experiences with Bluetooth in the Smart-Its Distributed System". Workshop on Ubiquitous Computing and Communications, PACT 01, Barcelona, Espanha , Outubro de 2001.

[KEL 90] Keller, S. E; et al. "Specifying Software Quality Requirements with Metrics". Tutorial System and Software Requirements Engineering IEEE Computer Society Press, p.145-163, 1990.

[KIM 05] Kim, S. "Structural health monitoring of the golden gate bridge", Disponível: http://www.cs.berkeley.edu/~binetude/ [consulta: Janeiro de 2006].

[KIN 02] Kinawi, H.; Taha, M.M.; El-Sheimy, N. "Gpsr: greedy perimeter stateless routing for wireless networks". In 27th Annual IEEE Conference on Local Computer Networks (LCN'02), Florida, EUA, 2002.

[KNI 02] Knight, J. C. "Safety Critical Systems: Challenges and Directions". ACM, Florida, 2002, p. 547 - 550, 2002.

[KUD 04] Kudo, T. N; Araujo, R. B. "Computação Ciente de Contexto Aplicada ao Monitoramento de Condições Críticas em Ambientes Físicos". Tese de Mestrado, São Carlos, Maio de 2004.

[LAM 78] Lamport, L. "Time, Clocks, and the Ordering of Events in a Distributed System". Communications of the ACM, p. 558-565, Julho de 1978.

[MAD 05] Madden, S. et al. "TinyDB: An acquisitional query processing system for sensor networks". Transactions on Database Systems (TODS), Nova Iorque, EUA, 2005.

[MAI 02] Mainwaring, A.; Polastre, J.; Szewczyk, R.; Culler, D. Anderson. J. "Wireless sensor networks for habitat monitoring". First ACM Workshop on Wireless Sensor Networks and Applications. Setembro de 2002.

[MAN 97] Mansouri-Samani, M.; Sloman, M. "GEM – A Generalised Event Monitoring Language for Distributed Systems". IEE/IOP/BCS Distributed Systems Engineering Journal, 4(25), Fevereiro de 1997.

[MIC 04] Micheloti, G. C.; Araújo, R. B. "Proposta de um Middleware para Monitoramento de Situações de Emergência em Ambientes Físico ou Lógicos Cientes de Contexto". Tese de mestrado da UFSCar. São Carlos, São Paulo, Julho de 2004.

[MIN 02] Mini, R. A. F., Nath, B., and Loureiro, A. A. F. A probabilist approach to predict the energy consumption in wireless sensor networks. IV Workshop de Comunicação sem Fio e Computação Móvel, São Paulo, SP. Outubro de 2002.

[MOK 06] Mokhoff, N. "Army tests wireless sensors as vehicle inspectors", Disponível: http://www.automotivedesignline.com/news/16940066 [consulta: Janeiro de 2006].

[MOS 85] MOSTOW, J.; "Towards Better Models of Design Process". AI Magazine, Vol. 6 No. janeiro 1989, p. 44-57, 1985.

[MUH 04] Mühl, G., Ulbrich, A., Ritter, H. "Content Evolution Driven Data Propagation in Wireless Sensor Networks". Em proceedings MuUlRi:2004:Funnel, p. 55-57, Berlim, Alemannha, Fevereiro de 2004.

[NS2 06a] The Network Simulator ns-2. Disponível: www.isi.edu/nsman/ns [consulta: Março de 2006].

[PET 00] Petriu, E.M.; Georganas, N.D; Petriu, D.C.; Makrakis, D.; Groza, V.Z. "Sensor-based information appliances". IEEE Instrumentation and Measurement Magazine, December 2000.

[PIN 04] Pinto, A. J. G. "Mecanismo de Agragação de Dados Empregando Técnicas Paramétricas em Rede de Sensores". Tese de Mestrado da UFRJ. Junho de 2004.

[POT 00] Pottie, G. J., and Kaiser, W. J. "Wireless integrated network sensors (WINS). Communications of the ACM 43, p. 51-58. Maio de 2000.

[RAG 02] Raghunathan, V., Schurgers, C., Park, S., and Srivastava, M. "Energy Aware Wireless Microsensor Networks".IEEE Signal Processing

Magazine 19, p. 40-50. Março de 2002.

[RAP 96] T. Rappaport. "Wireless Communications: Principles and Practice". Prentice-Hall, Englewood Clis, NJ, 1996.

[ROM 01] Römer, K. "Time synchronization in Ad Hoc Networks". ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 01), Long Beach, California, Outubro de 2001.

[ROM 02] Römer, K. "Temporal Message Ordering in Wireless Sensor Networks". IFIP Mediterranean Workshop on Ad-Hoc Networks 2003, p. 131-142, Mahdia, Tunisia, Junho de 2003.

[ROM 85] ROMAN, G. A.; "Taxonomy of Current Issues in RequirementsEngineering", IEEE Computer, Vol. 18, p. 14-23, 1985. Abril de 1985.

[RTI 06] Georgia Tech, Parallel and Distributed Simulation, RTI Implementation. Disponível: http://www-static.cc.gatech.edu/computing/pads/tech-highperf-rti.html [consulta: Janeiro de 2006]

[RUI 04a] Ruiz, L. B., Nogueira, J. M. S., and Loureiro, A. A. (2004). "Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems", volume 1, capítulo III:Sensor Network Management. CRCPress.

[RUI 04b] Ruiz, L. B. and et al. "Arquitetura para Rede de Sensores Sem Fio". Simpósio Brasileiro de Rede de Computadores. Gramado, Rio Grande do Sul.Maio de 2004.

[SCH 01] Schwiebert, L. Gupta,S. K. S.; Weinmann, J. "Research challenge in wireless networks of biomedical sensors". In Mobile Computing and Networking, p.151–165, 2001.

[SHE 01] C. Shen, C. Srisathapornphat, C. Jaikaeo, "Sensor information networking architecture and applications", IEEE Personal Communications, p. 52-59. Universidade Delaware Newark, Alemanha. Agosto de 2001.

[SHI 01] Shih, E., Cho, S.-H., Ickes, N., Min, R., Sinha, A., Wang, A., and Chandrakasan, A. "Physical layer driven protocol and algorithm design for energy-eficient wireless sensor networks". The Seventh Annual International Conference on Mobile Computing and Networking (MobiCom 2001), Roma, Italia. Julho de 2001.

[SHI 90] Shim, Y. C.; Ramamoorthy, C. V. "Monitoring and Control of Distributed Systems". First Intl. Conference of Systems Integration, p. 672–681, Morristown, EUA, 1990.

[TAN 96] Tanenbaum, A. S. "Computer Networks (3rd ed.)". Prentice- Hall, Inc. Upper Saddle River, NJ, EUA, 1996.

[TIA 02] Tian, D.; Georganas, N. D. "A coverage-preserving node scheduling scheme for large wireless sensor networks". Proc., 1st ACM International Workshop on Wireless Sensor Networks and Applications, Atlanta, Georgia, 2002.

[TIL 02] Tilak, S., Abu-Ghazaleh, N., Heinzelman, W. "A taxonomy of wireless micro-sensor network models". SIGMOBILE Mob. Comput. Commun. Rev. (revista da ACM), p. 28-36, Nova Iorque, EUA em 2002.

[TUR 96] Turine, M. A. S.; Masiero, P. C. "Especificação de Requisitos: Uma introdução". Relatório Técnico. ICMC/USP, São Paulo,1996.

[WAR 01] Warneke, B.; Last, M.; Liebowitz, B.; Pister, K. S. J. "Smart dust: Communicating with a cubic-millimeter computer". IEEE Computer, 34(1):44–51, 2001.

[WRI 04] Wright P. and et al. "Fire Information and Rescue Equipment". Disponível: http://bmi.berkeley.edu/fire/indexmain.php [consulta: Março de 2004].

[WU 00] Wu, S.;Tseng, Y.; Sheu, J. "Intelligent Medium Access For Mobile Ad Hoc Networks With Busy Tones And Power Control", IEEE Journal on Selected Areas in Communications, p.1647–1657. Setembro de 2000.

[YE 02] Ye, W.; Heidemann, J.; Estrin, D. "An Energy-Efficient MAC Protocol for Wireless Sensor Networks". IEEE Infocom 2002, New York, EUA,

Junho de 2002.

[YOU 01] Youssef, M., Younis, M., and Arisha, K. “A Constrained Shortest-Path Energy-Aware Routing Algorithm for Wireless Sensor Networks”. Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC 2002), Orlando, Florida, EUA. Março de 2001.

Anexo

Este anexo apresenta o simulador Java implementado para avaliar os algoritmos do capítulo 5. Devido à sua grande extensão, são apresentados somente alguns dos principais trechos de código do simulador.

```
/*************************************************************************
Classe que representa o escalonador de eventos, semelhante à utilizada no NS2 no
modo LIST. O escalonador mantém os eventos numa lista em ordem de tempo de
maneira FIFO
*************************************************************************//

public class scheduler
{
   List listEvents= new ArrayList();

   ...
   // adiciona a lista os eventos e o tempo em que ele ocorrerá (step)
   void addEvent(double step, Node no, message msgs)
   {

      listEvents.add(new OBJ(step,no,msgs)); //adiciona os eventos
      // ordena os eventos de acordo com seus tempos
      Collections.sort(listEvents, new A());
      ...
   }
   ...
   // método que executa os eventos escalonados no escalanador
   void run()
   {
      OBJ obj;
      Node no;
      message msg;

      // Enquanto existem eventos no escalonador eles serão:
      while(listEvents.size() != 0) {

         obj = (OBJ) listEvents.get(0);
         listEvents.remove(obj); // retirados do escalonador

         no= obj.no;
         msg=obj.msg;

         enviroment.setStep(obj.num);

         no.recv(msg,this); // executados

      }
   }

};
/*********************************************************************
Classe que representa mensagens ou pacotes trocados entre os nós da rede. Estas
mensagens possuem tipo (resposta, dados, folhas), nó fonte da mensagem e nó que
está atualmente remetendo.
*********************************************************************/
```

```
public class message
{

   public static final int ACK = 0x00;   // confirmação
   public static final int MSG = 0x01;   // dados
   public static final int LEAF = 0x02;  // definição de folha

   int msgType;
   int source;
   int sender;


   public int getMsgType(){
      return msgType;
   }

   public void setMsgType(int Type){
      msgType = Type;
   }

   public int getMsgSource(){
      return source;

   }
   public void setMsgSource(int src){
      source = src;

   }

   public int getMsgSender(){
      return sender;

   }
   public void setMsgSender(int sd){
      sender = sd;

   }
...
}

/*****************************************************************************
Classe abstrata que representa qualquer nó da rede, sendo especializada em nós
sensores e atuadores
*****************************************************************************//

public abstract class Node {

      // acrescenta quem são os nós filhos (vizinhos mais distantes no caminho)
      public abstract void addNext(sensor no);
   public abstract void recv(message msg, scheduler sch);
   public int getNum(){
      return num;
   }
   public void setNum(int num) {
      this.num=num;
   }
   private int num;
}
```

```
/*****************************************************************************
Classe que representa um nó sensor da rede
*****************************************************************************//

public class sensor extends Node
{
   Node priorNode; // Vizinho anterior no caminhos
   List nextNodes; // Vizinhos posteriores nos caminhos
   int energy;     // Energia de cada nó sensor
      scheduler sch;
   int leaf;

   public sensor (int num)
   {
      nextNodes=new ArrayList();
      this.setNum(num);
      energy=0;
      leaf = 0;

   }
   public void addNext(sensor no) {
      nextNodes.add(no);
      no.priorNode=this;
   }

   public void consumeEnergy(){...}

   public int getEnergy(){...}

   public void recv(message msg, scheduler sch) {

      //se é uma mensagem pras folhas, mas nao é uma folha, entao descarta
      if(msg.getMsgType()==message.LEAF && leaf!=1)
         return;

      processSign(msg,sch);
   }

   // método que processa a mensagem enviada ao nó
   public void processSign(message msg, scheduler sch) {

      switch (msg.getMsgType())
      {
         case  message.MSG:
            sendpkt(msg, sch);
         break;

         case message.ACK:
            numAck++;
            if(numAck==nextNodes.size()){
               numAck=0;
               sendpkt(msg,sch);
            }
         break;

         case message.LEAF:

         sendpkt(new message(message.ACK,this.getNum(),this.getNum()),sch);
         break;
      }
```

```
    }
    public void sendpkt(message msg, scheduler sch)
    {

        consumeEnergy();
        msg.setMsgSender(this.getNum());
        /* antes de enviar o pacote é adicionado ao escalonador para que dele seja
         executado/*
        sch.addEvent((enviroment.getStep())+enviroment.hopTime,this.priorNode,msg)
    }
};

/*************************************************************************
Classe que representa um nó atuador da rede, esta classe implementa parte do
OBC.
*************************************************************************//

public class actor extends Node
{
    List nextNodes;
    List arrivedAck;//Armazena os Ack que chegaram
    List buffer1;    //Armazena as mensagens do mesmo passo de confirmação
    List buffer2;    //Armazena as mensagens do próximo passo de confirmação
    boolean wait;
    simulation sim;
    sensor vetNode[];

    int num;
    public actor (int num, sensor vetNode[])
    {
        nextNodes=new ArrayList();
        buffer1=new ArrayList();
        buffer2=new ArrayList();
        arrivedAck  = new ArrayList();
        this.setNum(num);
        sim=new simulation();
        this.vetNode=vetNode;
        wait=false;

    }

    public void recv(message msg, scheduler sch) {

        ...

        processSign(msg,sch);
    }
    public void processSign(message msg, scheduler sch) {

        switch (msg.getMsgType())
        {
            case  message.MSG:


            if(arrivedAck.contains(new Integer (msg.getMsgSender())))
            {
                buffer2.add(msg);
            }
            else
            {
                buffer1.add(msg);
```

```
            if(wait == false)
            {

               wait = true;
               sendpkt(sch);
            }

         }
         break;

      case message.ACK:

         arrivedAck.add(new Integer (msg.getMsgSender()));

         if (arrivedAck.size()==nextNodes.size())
         {


            for(Iterator i = buffer1.iterator();i.hasNext();){
               System.out.print(i.next());
            }
            wait=false;
            arrivedAck = new ArrayList();
            buffer1 = new ArrayList();
            if(buffer2.size()!=0){
               sendpkt(sch);
               wait = true;
               buffer2 = new ArrayList();
            }
         }
      break;
      }

   }

   public void sendpkt(scheduler sch){
      int i;

      for(i=0;i < enviroment.numNodes;i++){
         sch.addEvent((enviroment.getStep())+enviroment.hopTime,vetNode[i],new
         message(message.LEAF,-2,this.getNum()));

      }
   }
}
```