

**UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO  
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA**

**Roteamento Sensível ao Contexto em Redes de Sensores sem  
Fio: Uma abordagem baseada em Regras de Aplicação para o  
Protocolo RPL**

**VITÓRIA  
2014**

**VINICIUS BARCELLOS ANTUNES**

**Roteamento Sensível ao Contexto em Redes de Sensores sem Fio: Uma abordagem baseada em Regras de Aplicação para o Protocolo RPL**

Dissertação submetida ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo como requisito parcial para a obtenção do grau em Mestre em Informática.

Orientador: Prof. Dr. José Gonçalves Pereira Filho

**VITÓRIA  
2014**

Dados Internacionais de Catalogação-na-publicação (CIP)  
(Biblioteca Setorial Tecnológica,  
Universidade Federal do Espírito Santo, ES, Brasil)

---

A636r Antunes, Vinicius Barcellos, 1983-  
Roteamento sensível ao contexto em redes de sensores  
sem fio : uma abordagem baseada em regras de aplicação para  
o Protocolo RPL / Vinicius Barcellos Antunes. – 2014.  
80 f. : il.

Orientador: José Gonçalves Pereira Filho.  
Dissertação (Mestrado em Informática) – Universidade  
Federal do Espírito Santo, Centro Tecnológico.

1. Redes de sensores sem fio. 2. Roteamento  
(Administração de redes de computadores). 3. RPL (Protocolo  
de rede de computação). 4. Contexto (Informática). 5.  
Reconfiguração Dinâmica. I. Pereira Filho, José Gonçalves. II.  
Universidade Federal do Espírito Santo. Centro Tecnológico. III.  
Título.

CDU: 004

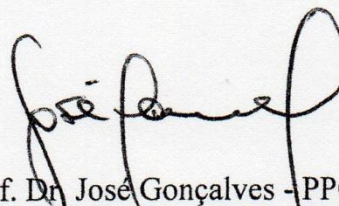
---

# ***Roteamento Sensível ao Contexto em Redes de Sensores sem Fio: Uma Abordagem Baseada em Regras de Aplicação para o Protocolo RPL***

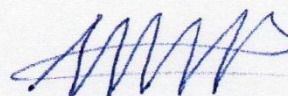
***Vinicius Barcellos Antunes***

Dissertação submetida ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo como requisito parcial para a obtenção do grau de Mestre em Informática.

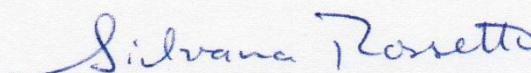
Aprovada em 29 de agosto de 2014 por:



Prof. Dr. José Gonçalves - PPGI/UFES



Prof. Dr. Magnos Martinello - PPGI/UFES



Prof. Dr. Silvana Rossetto - UFRJ/RJ

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO  
Vitória-ES, 29 de agosto de 2014

Dedico este trabalho a todos os professores  
que tive, pois de alguma maneira  
contribuíram para a minha formação.

## **AGRADECIMENTOS**

Agradeço primeiramente a Deus por esta conquista.

Aos meus pais, pelo incentivo que sempre me deram, aos os familiares que me apoiaram nesta jornada.

Aos meus colegas de mestrado, pela amizade, apoio e momentos de descontração.

Aos professores do Departamento de Informática da UFES, pelos ensinamentos.

Ao meu orientador, José Gonçalves, pela grande experiência e enorme dedicação que me serviram de exemplo profissional e pessoal.

A todos aqueles que, de uma forma ou de outra, me ajudaram a chegar até aqui, muito obrigado!

## RESUMO

Este trabalho apresenta um serviço de reconfiguração dinâmica para Redes de Sensores sem Fio. O trabalho inclui o projeto e a definição de uma arquitetura conceitual que suporta a coleta de uma variedade de informações contextuais e provê uma abstração alto nível para especificação de roteamento sensível ao contexto através de reconfiguração de métricas de roteamento e parâmetros de comunicação. O objetivo da infraestrutura proposta é possibilitar a criação de regras que adaptem o comportamento da rede em tempo de execução, em função dessas informações contextuais. Uma implementação da arquitetura para o protocolo RPL e o sistema operacional Contiki foi realizada, mostrando a viabilidade da abordagem proposta.

**Palavras-chave:** Roteamento, RPL, Contexto, Reconfiguração Dinâmica

## **ABSTRACT**

This work presents a dynamic reconfiguration service for wireless sensor networks. The work includes the design and definition of a conceptual architecture that supports collecting a variety of contextual information and provides a high level abstraction for context-sensitive routing specification through reconfiguration of routing metrics and communication parameters. The objective of the proposed infrastructure is enabling the creation of rules that change the network's behavior at run time, in the light of these contextual information. An implementation of the architecture for the RPL Protocol and the Contiki operating system was performed, showing the feasibility of the proposed approach.

**Keywords:** Routing, RPL, Context, Dynamic Reconfiguration



## Lista de Figuras

Figura 2.1 - Padrões de Redes Sem Fio .....	21
Figura 2.2 - Modelo de referência 6LoWPAN. Fonte: (SHELBY; BORMANN, 2011) .....	24
Figura 2.3 - Informações para Métricas. Fonte: (HU et al., 2008).....	27
Figura 2.4 - Aspectos de desempenho para Métricas. Fonte: (ZAHARIADIS, 2012) .....	27
Figura 2.5 - Rede RPL com três DODAGs e duas instâncias. Fonte: (GADDOUR et al., 2012).....	29
Figura 3.1 - Múltiplas Instâncias na mesma rede física. Fonte: ( ZAHARIADIS) .....	38
Figura 3.2 - Criação e Gerenciamento de Instâncias RPL.....	39
Figura 3.3 - Regras Fuzzy .....	40
Figura 3.4 - Função de Soma .....	40
Figura 3.5 - Arquitetura SA-A-WSN .....	44
Figura 4.1 - Projeto de uma RSSF .....	48
Figura 4.2 - Arquitetura Conceitual.....	49
Figura 4.3 - Base de Fatos .....	52
Figura 4.4 - Representação de Fatos .....	52
Figura 4.5 - Regra Condição-Ação .....	53
Figura 4.6 - Arquitetura Conceitual QoC.....	55
Figura 4.7 - Parâmetros de Reconfiguração .....	57
Figura 4.8 - Reconfiguração do RPL.....	58
Figura 5.1 - Comunicação RSSF-Middleware.....	59
Figura 5.2 - Reconfiguração de Métrica de Roteamento .....	63
Figura 5.3- Regras Drools .....	65
Figura 5.4 - Topologia da Simulação .....	67
Figura 5.5 - Gráfico de Perda de Pacotes .....	68
Figura 5.6 - Gráfico de RTT (Round Trip Time) .....	70
Figura 5.7 - Gráfico de Mudanças na rota Default.....	71

## **Lista de Tabelas**

Tabela 1 - Parâmetros de Reconfiguração .....	64
---	----

## Lista de Siglas/Acrônimos

CSMA-CA	–	Carrier sense multiple access with collision avoidance
DAG	–	Directed Acyclic Graph
DAO	–	Destination Advertisement Object
DIO	–	DODAG Information Object
DIS	–	DODAG Informational Solicitation
DODAGs	–	Destination Oriented DAGs
FFD	–	Full Function Device
IEEE	–	Institute of Electrical and Electronics Engineers
IETF	–	Internet Engineering Task Force
IoT	–	Internet of Things
IP	–	Internet Protocol
LLN	–	Low Power and Lossy Networks
MAC	–	Media Access Control
MRHOF	–	Minimum Rank with Hysteresis Objective Function
MTU	–	Maximum Transmission Unit
NDP	–	Neighbor Discovery Protocol
OF	–	Objective Function
QoC	–	Quality of Context
RFC	–	Request for Comments
RFD	–	Reduced Function Device)
RFID	–	Radio-Frequency Identification
RPL	–	Routing Protocol for Low-Power and Lossy Networks
RSSF	–	Redes de Sensores Sem Fio
UDP	–	Unreliable Datagram Protocol
WPAN	–	Wireless Personal Area Network

# Sumário

1.	Introdução.....	13
1.1.	Motivação.....	13
1.2.	Objetivos Gerais .....	18
1.3.	Objetivos Específicos.....	18
1.4.	Organização do Trabalho .....	19
2.	Referencial Teórico .....	20
2.1.	O Padrão IEEE802.15.4.....	20
2.2.	O Padrão 6LoWPAN.....	22
2.3.	Roteamento.....	24
2.4.	O Protocolo RPL.....	28
2.5.	Sensibilidade ao Contexto.....	31
2.6.	Reconfiguração Dinâmica.....	34
3.	Trabalhos Relacionados .....	37
3.1.	Adaptação do Protocolo RPL.....	37
3.2.	Outros Trabalhos Relacionados .....	41
4.	Arquitetura Proposta .....	45
4.1.	Introdução.....	45
4.2.	Princípios e Requisitos.....	46
4.3.	Arquitetura Conceitual.....	47
4.3.1.	Representação do Conhecimento .....	51
4.3.2.	Dados Contextuais.....	51
4.3.3.	Reconfiguração Condicional.....	53
4.3.4.	Processo de Inferência .....	53
4.3.5.	Qualidade de Contexto.....	54
4.3.6.	Atuação .....	56
5.	Implementação e Testes .....	59
5.1.	Introdução.....	59
5.2.	Cenário de Exemplo .....	64
5.3.	Testes .....	66
5.3.1.	Perda de Pacotes.....	68
5.3.2.	Round-Trip Time.....	69
5.3.3.	Estabilidade da Rede .....	70

5.4. Discussão.....	71
6. Conclusões.....	73
7. Referências.....	75

## 1. Introdução

### 1.1. Motivação

A crescente demanda pela construção de “espaços inteligentes” tem fomentado a investigação e o uso combinado de tecnologias de comunicação sem fio e de sensoriamento acopladas a dispositivos de baixo custo e de baixo consumo, com capacidades limitada de processamento e armazenamento, e fonte de energia independente. As *Redes de Sensores Sem Fio – RSSF (Wireless Sensor Networks – WSN)* (DARGIE; POELLABAUER, 2010)(AKYILDIZ; VURAN, 2010)(KARL; WILLIG, 2005) e os sistemas *RFID (Radio-Frequency Identification)* (FINKENZELLER, 2008) são exemplos de componentes considerados fundamentais para a construção de novos sistemas, ambientes e *frameworks* que permitirão conectar, em larga escala, o mundo real ao mundo digital.

O desenvolvimento de serviços e aplicações baseadas em tecnologias desses “objetos inteligentes” apresenta um potencial extraordinário de crescimento nos próximos anos mas, ao mesmo tempo, impõe uma forte demanda pela investigação e construção de novos modelos e padrões de comunicação que facilite e universalize a interação entre esses elementos e destes com a infraestrutura de rede.

A iniciativa de embarcar inteligência nos objetos do dia-a-dia (e.g., relógios, óculos, geladeiras, sapatos, ferramentas, etc.) e, posteriormente, integrá-los à Internet, inaugura uma nova fase do desenvolvimento da Web: a realização da idéia de “Internet das Coisas” (*Internet of Things – IoT*) (ATZORI et al., 2010). A IoT é um novo conceito de Internet, que caracteriza a visão de uma rede global formada por objetos inteligentes interconectados e endereçáveis, a partir do uso e adaptação de protocolos e padrões de comunicação bem definidos e conhecidos da Internet atual. Espera-se que os objetos no escopo de IoT não sejam apenas dispositivos com comunicação sem fio, memória e capacidade de processamento, mas que tenham também autonomia, comportamento pró-ativo, conhecimento sobre o contexto e sejam capazes de cooperar entre si para alcançar metas comuns.

As vantagens provenientes da integração de dispositivos inteligentes e de baixo consumo com a Internet podem ser vistas a partir de diferentes ângulos. Sob o ponto de vista das aplicações que usam esses dispositivos, o uso de padrões e protocolos bem conhecidos e testados facilita a difusão dos dados coletados e o uso combinado desse dados com informações provenientes de outras aplicações. Sob o ponto de vista da Internet, a implementação da pilha de protocolos IP em dispositivos com capacidades limitadas possibilita estender o seu alcance para além dos computadores, deixando de ser uma rede que conecta computadores para se tornar uma rede que conecta potencialmente quaisquer dispositivos (ou “*coisas*”).

Entretanto, para alcançar esses benefícios, vários desafios precisam ainda ser tratados. Entre eles estão a necessidade de ter mecanismos de endereçamento que permitam identificar unicamente cada dispositivo conectado à rede, e a redução da complexidade de implementação e operação da pilha de protocolos IP, de forma a viabilizar a sua implantação em dispositivos com capacidade de memória, processamento e fonte de energia limitados. O amplo espaço de endereçamento do protocolo IPv6 (LOSHIN, 2004) sugere a sua adoção para resolver a demanda por endereços. Por outro lado, requer mecanismos de compactação para reduzir o tamanho do seu cabeçalho. Outro desafio é a característica de operação descontinuada (alternância entre modo ativo e modo inativo) das RSSFs com vistas à economia de energia (em modo inativo – *sleep* – os nós ficam impedidos de se comunicarem, uma situação anômala para redes IP).

As Redes de Sensores sem Fio são exemplos clássicos das ditas Redes de Baixo Consumo e Baixa Potência (*Low-Power and Lossy Networks – LLNs*), as quais são reconhecidas como uma área importante dentro da IoT. As redes LLN são definidas como redes nas quais os roteadores tipicamente operam com severas restrições em termos de capacidade de processamento, armazenamento e fonte de energia. As interconexões de comunicação, por sua vez, são caracterizadas por elevadas taxas de perdas de pacotes, baixas taxas de transmissão e alta instabilidade. Assim, as Redes de Sensores Sem Fio são classificadas como um tipo de rede LLN com finalidade específica de monitoramento remoto de grandezas físicas (e.g., temperatura, nível de poluição, pressão, interferências, acústica) e atuação sobre determinados elementos de controle.

## O Roteamento em RSSF

Como salientado em (KO et al., 2011), a pesquisa na área de RSSF se voltou inicialmente para problemas fundamentais ou para o desenvolvimento de aplicações específicas, deixando as questões de padronização e interoperabilidade como desafios a posteriori. Em função das suas características e restrições particulares, diferentes protocolos de comunicação foram projetados para essas redes, em sua maioria utilizando configurações estáticas e inseridas diretamente no código (*hard-coded*).

Um dos problemas mais fundamentais de uma RSSF – tema central desta dissertação de mestrado – é o mecanismo de *roteamento* dos dados através da rede (AKKAYA; YOUNIS, 2005). O roteamento desempenha um papel importante na operação da RSSF, pois é ele o responsável pelo gerenciamento da árvore topológica de encaminhamento de pacotes, envolvendo aspectos de formação, configuração e manutenção. As métricas de roteamento, por sua vez, são definidas pelos protocolos e são responsáveis por determinar a criação dos caminhos na rede. Desta forma, o roteamento é o processo de selecionar caminhos entre origens e destinos através de um número arbitrário de nós intermediários, com o objetivo de estabelecer um caminho para a comunicação das aplicações dos nós sensores.

O protocolo de roteamento, o qual é responsável pela descoberta desses caminhos, pode associar custos a cada caminho em potencial da origem ao destino, em que esses custos são computados através de métricas de roteamento, sendo o caminho escolhido para o tráfego de dados aquele que possuir o menor custo. No entanto, a heterogeneidade de nós sensores e a topologia dinâmica da rede criam desafios no processo de desenvolvimento de soluções de roteamento. O grande volume de informações que as soluções de roteamento podem considerar para lidar com as situações de mudanças na rede e/ou requisitos do usuário é um desafio importante.

No processo de descoberta dos melhores caminhos de uma determinada origem até um destino, uma variedade de informações pode ser explorada, avaliada e considerada no processo de tomada de decisão. Informações relativas aos nós sensores (e.g., o tipo de nó, quantidade de saltos), interfaces de rádio (área de cobertura, canal) e meio de transmissão (interferência) são exemplos de dados comumente utilizados pelos protocolos de roteamento. Alguns desses parâmetros são selecionados e utilizados no



protocolo de roteamento com o objetivo de obter uma configuração ótima sob determinadas circunstâncias e objetivos da rede.

### **A Sensibilidade ao Contexto**

Há também outra questão importante relativa à natureza dos dados utilizados no processo de tomada de decisão de roteamento. Muitos deles são dinâmicos, isto é, mudam seu valor com o tempo. Desta forma, o protocolo de roteamento pode, até certo ponto, alterar seu próprio comportamento para se adaptar à mudanças na rede, potencialmente aprimorando sua confiabilidade e robustez.

Como forma de endereçar estas questões, o paradigma de Computação Sensível ao Contexto (*Context-aware Computing*) ou Computação Pervasiva (*Pervasive Computing*) (CHEN, 2000)(SATYANARAYANAN, 2001), no qual as aplicações utilizam informações contextuais para customizar seus serviços, se mostra potencialmente adequada. Neste paradigma, as aplicações utilizam informações obtidas do ambiente ou dos próprios usuários (informações contextuais) para adaptar seu comportamento de acordo com as situações e as necessidades dos seus usuários. Por contexto pode-se entender qualquer informação que possa ser usada para caracterizar a situação das entidades (uma pessoa, um lugar, ou um objeto) que são relevantes para uma aplicação, incluindo o próprio usuário e a própria aplicação (DEY, 2001). Portanto, segundo essa definição, se uma informação é utilizada para caracterizar a própria aplicação, esta informação é contexto.

Isto posto, o roteamento em RSSF também pode ser tratado como uma aplicação sensível ao contexto, possuindo habilidade de se adaptar às mudanças na topologia de rede, estado dos nós sensores ou mesmo interferência externa, de forma automatizada. Em relação à configuração dos parâmetros de roteamento, a abordagem geral (e talvez a mais comum) no desenvolvimento de aplicações para RSSF é utilizar configurações estáticas, de acordo com as características do ambiente (tamanho de fila, número de tentativas de retransmissão, *duty cycle*, etc). Por configurações estáticas entende-se que os vários parâmetros que controlam o comportamento da aplicação e da rede não sofrem alteração de valor com o tempo. Estas aplicações geralmente também são desenvolvidas

com requisitos específicos, por exemplo, maximizar o tempo de vida útil da rede (consumo de energia), latência e vazão, dentre outros. No entanto, isto não quer dizer que esses requisitos sejam fixos, já que eles podem mudar com o decorrer do tempo. Nesse sentido, a reconfiguração da rede (seja através da mudança do contexto ou intervenção manual do usuário) desempenha um papel importante, otimizando o desempenho da RSSF e atendendo aos requisitos das aplicações.

Como exemplo, pode-se citar uma aplicação de monitoramento em que os nós sensores devem aferir alguma propriedade física e enviar os dados coletados à estação base de tempos em tempos para análise. Devido à possível grande área de cobertura do ambiente monitorado, a comunicação multi-saltos se faz necessária. Além disso, o tempo de vida da rede é um requisito, já que é impraticável trocar a fonte de energia dos nós sensores (que geralmente são operados por baterias). Neste exemplo de aplicação, o modo normal de operação da rede (ou seja, com todos os parâmetros de configuração - camada 2, camada 3 - do Sistema Operacional em seus valores padrão) é o de economia de energia. Entende-se desta forma que (i) os rádios devem ficar desligados o máximo de tempo possível, (ii) haverá pouca comunicação de rede. Neste modo padrão (*low-power*), latência não é uma preocupação, e os dados coletados podem demorar um tempo razoável (em comparação com a rede em um modo de configuração que otimize latência) desde o momento de serem coletados até chegarem a estação base.

### **A Reconfiguração Dinâmica**

Em determinadas situações – por exemplo, a ocorrência de algum evento físico no ambiente monitorado – pode ser desejável que a aplicação aumente sua taxa de amostragem para confirmar o ocorrido. A quantidade de tráfego na rede também depende da quantidade de nós enviando dados simultaneamente, e pode aumentar consideravelmente. Com a rede no modo padrão (*low-power*) haverá congestionamento e perda de pacotes, o que acarretará em (i) perda de pacotes e (ii) desperdício de recursos, pois como a rede está congestionada, haverá muitas retransmissões.

Em uma situação como a descrita, a reconfiguração pode ser um recurso útil, pois a rede pode alterar o seu modo de funcionamento por um período de tempo (no

exemplo em questão, ocorrência do algum evento físico) para um modo de alta vazão de dados, e retornar ao seu estado de economia de energia posteriormente.

Outro aspecto às vezes negligenciado, é a dificuldade de se testar a aplicação projetada em condições realísticas. Durante o ciclo de desenvolvimento da aplicação, são feitos testes (em sua maioria utilizando-se simuladores/emuladores e *testbeds*). Como encontrar os parâmetros de configuração adequados dos nós sensores antes da sua distribuição em um ambiente de produção? Em (BARRENETXEA et al., 2008) é demonstrado que durante a fase de distribuição, vários problemas podem ocorrer. Além disso, a RSSF não se comporta exatamente de acordo com o planejado e esperado, tornando a reconfiguração dinâmica um requisito desejável.

Após análise da literatura da área de RSSF, nota-se uma escassez de trabalhos que tratem ou explorem, conjuntamente, os aspectos de *context-awareness* e reconfiguração dinâmica em protocolos de roteamento de RSSF. É importante destacar também que o padrão de protocolo de roteamento proposto pelo IETF para as LLNs (o RPL) não contempla essas características de maneira ampla e explícita (ao menos de forma conjunta).

## 1.2. Objetivos Gerais

Com a finalidade de simplificar e otimizar o processo de comunicação entre os nós, este trabalho propõe o desenvolvimento de um serviço de middleware, para reconfiguração dinâmica de métricas de roteamento para o protocolo RPL. Este serviço tem como foco (i) coletar uma variedade de informações contextuais e gerenciar as fontes dessas informações; e (ii) prover uma abstração alto nível para especificação de roteamento sensível ao contexto através de reconfiguração de métricas de roteamento.

## 1.3. Objetivos Específicos

Os seguintes objetivos específicos para este trabalho podem ser listados:

- Especificar um serviço que suporta a coleta e o tratamento de dados contextuais da Rede de Sensores; e a realização de inferência sobre os dados contextuais coletados.

- Especificar um serviço que suporta a reconfiguração dinâmica de parâmetros da rede de sensores sem fio (parâmetros visando a otimização do processo de roteamento);
- Especificar, no contexto do protocolo RPL, um mecanismo que possibilite a sua reconfiguração;
- Implementar e realizar testes de desempenho do protocolo RPL, segundo a abordagem definida no passo anterior.

#### **1.4. Organização do Trabalho**

Além desta Introdução, o trabalho está organizado em mais cinco capítulos, a saber:

- Capítulo 2: apresenta um breve referencial teórico, abordando conceitos básicos de roteamento em Redes de Sensores sem Fio e o protocolo de roteamento IETF RPL. O capítulo trata também de alguns conceitos básicos de sensibilidade ao contexto e reconfiguração dinâmica para Redes de Sensores.
- Capítulo 3: discute os trabalhos relacionados.
- Capítulo 4: descreve o projeto da arquitetura conceitual.
- Capítulo 5: descreve detalhes da implementação da arquitetura proposta e uma avaliação de desempenho utilizando um ambiente simulado.
- Capítulo 6: conclui a dissertação, apresentando as considerações finais e as perspectivas de continuidade do trabalho.

## 2. Referencial Teórico

Para um melhor entendimento das contribuições do trabalho, este capítulo realiza uma breve revisão dos conceitos básicos das tecnologias e dos principais tópicos de pesquisa aqui tratados: os padrões IEEE802.15.4 e IETF 6LoWPAN, roteamento e reconfiguração dinâmica em RSSF, e sensibilidade ao contexto. O capítulo está organizado da seguinte forma: as Seções 2.1 e 2.2 introduzem, respectivamente, os padrões IEEE802.15.4 e IETF 6LoWPAN, adotados nas LLNs de interesse deste trabalho; a Seção 2.3 discute o roteamento em redes de sensores sem fio; a Seção 2.4 descreve o protocolo RPL, padrão de roteamento definido pelo IETF para redes LLN; a Seção 2.5 apresenta os princípios da computação sensível ao contexto; a Seção 2.6 introduz conceitos de reconfiguração dinâmica em redes de sensores sem fio; e, finalmente, a Seção 2.7 traz as considerações finais do capítulo.

### 2.1. O Padrão IEEE802.15.4

Um dos aspectos fundamentais da pilha de protocolos TCP/IP é a sua arquitetura em camadas, que permite que cada camada evolua de forma independente das demais sem comprometer o modelo arquitetural. Por exemplo, o protocolo IP, da camada de rede (camada 3), pode utilizar serviços de vários tipos de tecnologias de camada de enlace (camada 2) sendo, por isso, também chamado de "agnóstico", ou seja, novas tecnologias de enlace podem ser suportadas na rede sem a necessidade de mudanças na arquitetura IP.

No contexto das redes sem fio, vários tipos de tecnologias de camada de enlace foram propostas, por exemplo, o padrão 802.11/*WiFi*. No cenário das redes sem fio pessoais de baixa potência (*Wireless Personal Area Networks – WPAN*) – que são as redes de interesse deste trabalho – destaca-se o padrão IEEE802.15.4 (IEEE Computer Society 2003). Este padrão foi proposto pelo IEEE com o propósito de disponibilizar uma tecnologia WPAN simples, de baixo custo e de baixa potência, que pudesse atender a uma classe de aplicações *wireless* com demandas restritas de taxa de dados e QoS, inviáveis de se atingir com as outras tecnologias WPAN, tais como IEEE802.15.1/Bluetooth ou WLAN, tais como IEEE802.11/*WiFi* (Figura 2.1). Os nós

das redes de sensores sem fio de interesse deste trabalho são exemplos de dispositivos que seguem o padrão IEEE802.15.4.

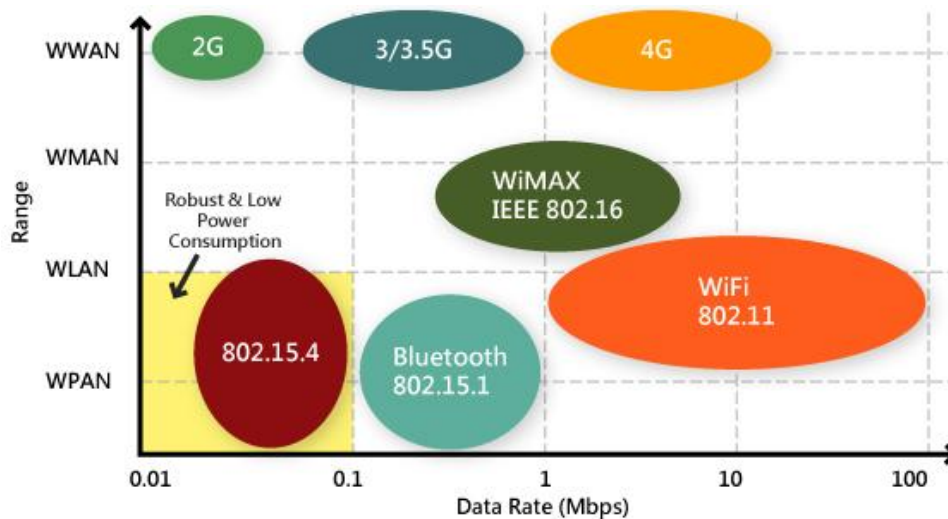


Figura 2.1 - Padrões de Redes Sem Fio

O padrão 802.15.4 especifica um enlace de comunicação sem fio para LoWPANs (*Low-power Wireless Personal Area Networks*), sendo particularmente interessante para aplicações de monitoramento que requeiram vários pontos de sensoriamento e controle com baixo custo, tais como monitoramento ambiental, monitoramento de estruturas civis, monitoramento de espécies de animais, rastreamento de objetos, entre outras. Essas aplicações tipicamente requerem um grande número de nós que se comunicam através de um ou vários saltos com a finalidade de cobrir uma determinada área geográfica de interesse. Além disso, os nós da rede devem ser capazes de operar em regiões desprotegidas e por um longo período de tempo, sustentados por baterias ou pilhas (fontes de energia esgotáveis).

Entre as características do padrão IEEE802.15.4, destacam-se:

- Baixa taxa de transmissão: a camada física pode ser configurada para operar em duas faixas de frequência — 868/915MHz (taxas de 20kbps, 40kbps e, opcionalmente, 100kbps e 250Kbps), ou 2.4GHz (taxas de 250Kbps);
- Baixa potência: os dispositivos são sustentados por baterias com perspectiva de operação durante meses ou anos;

- Baixo custo: normalmente o rádio é acoplado a dispositivos embutidos com recursos de hardware limitados;
- Distâncias curtas: o alcance do sinal varia de 10m a 100m;
- Diferentes tipos de dispositivos: os dispositivos podem ser do tipo FFD (*Full Function Device*) – podem ser coordenadores de grupos e roteadores de pacotes – ou RFD (*Reduced Function Device*) – dispositivos com funções limitadas (permanecem em modo inativo a maior parte do tempo);
- Diferentes modos de operação: dois modos de transmissão são definidos para a camada MAC, *beacon-enabled* e *nonbeacon-enabled*. Neste último, usa-se o mecanismo CSMA-CA para evitar colisões.

É importante ressaltar que, em muitos casos, tecnologias de enlace de baixa potência não são especificadas com a intenção de se utilizar para tráfego IP. Como resultado, as vezes é necessário uma camada de adaptação entre a camada de enlace e a camada de rede. Por exemplo, uma camada de enlace desenhada para carregar apenas pequenos pacotes, como no caso do IEEE802.15.4, pode necessitar de um mecanismo de fragmentação/remontagem a fim de acomodar um pacote IPv6. Um exemplo de camada de adaptação é o padrão 6LoWPAN, descrito brevemente a seguir.

## 2.2. O Padrão 6LoWPAN

6LoWPAN é um acrônimo para “*IPv6 over Low power Wireless Personal Area Networks*”, um grupo de trabalho do IETF responsável pela definição de normas que permitam a implementação do protocolo IPv6 sobre redes IEEE802.15.4.

Atualmente, a implementação de redes LoWPAN é efetuada por um conjunto de tecnologias proprietárias, o que dificulta a interoperabilidade entre diferentes redes e a sua inserção em serviços baseados na Internet. Como visto em seções anteriores, a integração do protocolo IP em redes LoWPAN permite superar este problema e introduz um conjunto de vantagens (SHELBY; BORMANN, 2010):

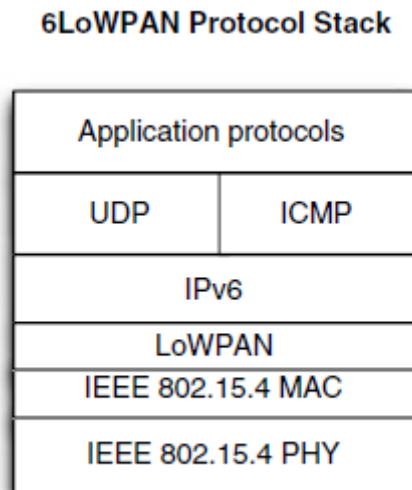
- a natureza ubíqua das redes IP permite a utilização de infraestruturas de rede já existentes;

- a tecnologia IP é bem conhecida e já se encontra devidamente testada;
- o protocolo IP é definido em especificações do IETF, as quais são disponibilizadas publicamente;
- já existem um conjunto de ferramentas disponíveis para diagnóstico e gestão de redes IP;
- dispositivos com conectividade IP podem ser ligados a uma rede IP sem necessidade de um *gateway* ou *proxy*.

No entanto, os dispositivos que tipicamente formam as redes IEEE802.15.4 possuem várias limitações que dificultam a implementação do protocolo IP, entre elas: o alcance de transmissão dos nós é de apenas algumas dezenas de metros, a taxa de transmissão máxima é de 250 kbps, os recursos de memória e fonte de energia são limitados, e, em especial, o padrão IEEE802.15.4 limita o tamanho dos quadros de enlace em 127 bytes. Por isso, a adoção do protocolo IP em redes IEEE802.15.4 traz uma série de desafios, como discutido amplamente em (SHELBY; BORMANN, 2010) e em (MONTENEGRO et al., 2007).

Entre os objetivos do grupo 6LoWPAN está o de definir mecanismos para acomodar pacotes IPv6, cujo MTU mínimo é de 1280 bytes, em quadros IEEE802.15.4, cujo MTU de apenas 127 bytes. Para isso, uma camada de adaptação é acrescentada à pilha IP, entre a camada de rede e a camada de enlace, para permitir a fragmentação e desfragmentação de pacotes IPv6 em quadros IEEE802.15.4 e efetuar a compressão do cabeçalho IPv6. A Figura 2.2 ilustra a pilha de protocolos do modelo de referência 6LoWPAN.





**Figura 2.2 - Modelo de referência 6LoWPAN. Fonte: (SHELBY; BORMANN, 2011)**

As recomendações 6LoWPAN, definidas originalmente na RFC 4919 (MONTENEGRO et al., 2007) e na RFC 4944 (RFC 4944), indicam ainda os requisitos para se efetuar o encaminhamento de pacotes em redes de malha, sugere algumas recomendações de segurança e descreve alterações no protocolo de descoberta de vizinhos (*Neighbor Discovery Protocol - NDP*) de forma a otimizá-lo para redes LoWPAN. O protocolo NDP dispõe de funções essenciais à implementação do processo de convergência IPv6 – 802.15.4: auto-configuração de endereços, resolução de endereços, detecção de duplicação de endereços e descoberta de roteadores (NARTEN, et al., 2007).

### **2.3. Roteamento**

O desenvolvimento de protocolos de roteamento robustos e eficientes para as RSSF é uma questão de pesquisa ainda bastante ativa, apresentando vários desafios devido à heterogeneidade dos nós e à natureza dinâmica das topologias. Essencialmente, o roteamento pode ser definido como o processo de selecionar caminhos entre origens e destinos, com possivelmente um número arbitrário de nós intermediários, com o objetivo de permitir a comunicação entre os mesmos.

O protocolo de roteamento pode determinar os caminhos associando custos a cada potencial caminho de uma origem a um destino, onde esses custos são calculados pelas métricas de roteamento.

É possível classificar as técnicas de roteamento em três categorias, dependendo da estrutura da rede (AL-KARAKI; KAMAL, 2004): *flat-based*, em que todos os nós tem o mesmo papel na rede; *hierarquical-based*, em que os nós possuem papéis diferentes; e *location-based*, onde os dados são roteados de acordo com a posição geográfica dos nós. Outra classificação diz respeito a como os emissores encontram os destinatários das mensagens, e resulta em três classificações: protocolos *proativos*, *reativos* e *híbridos*. Nos protocolos proativos, as rotas são calculadas em um primeiro momento, antes do envio da mensagem. Assim, no momento de enviar a mensagem, o protocolo consulta uma tabela (roteamento) para decidir por qual caminho a mensagem deve seguir. Nos protocolos reativos, as rotas são calculadas dinamicamente, isto é, quando há a necessidade de enviar uma mensagem é feito o cálculo de roteamento; assim, o protocolo não mantém tabela de roteamento. Os híbridos usam uma combinação das duas técnicas. Portanto, os protocolos proativos são melhores para ambientes com um número fixo de nós e os reativos para ambientes onde os nós são móveis.

Ainda em (AL-KARAKI; KAMAL, 2004), há outra classificação para protocolos de roteamento, denominada cooperativos. Nestes, o dado é enviado a um nó central que agrega os dados de vários emissores a fim de reduzir o custo em termos energéticos.

Em (KARL; WILLIG, 2007) é proposta uma nova classificação para os protocolos de roteamento, de acordo com o destino da mensagem (um único nó, um conjunto de nós ou todos os nós da rede). Os seguintes tipos podem ser encontrados:

- Encaminhamento *unicast Gossiping* e baseado em agente;
- Energia;
- *Broadcast e multicast*;
- Roteamento geográfico;
- Mobilidade.

No primeiro tipo (*Gossiping* e *agent-based*) são feitas tentativas de se rotear mensagens sem a necessidade de uma tabela de roteamento. O exemplo mais simples é o protocolo *flooding* (inundação), mas este não é muito eficiente.

O segundo tipo (Energia) consiste em técnicas que analisam a distribuição dos nós na rede, efetuam o cálculo do custo para transmitir a mensagem entre dois nós em determinado enlace e escolhem um algoritmo para calcular o custo mínimo. Os objetivos principais deste tipo na consideração do cálculo do custo podem ser minimizar a energia por pacote, maximizar o tempo de vida da rede, dentre outros. Esses dois primeiros tipos tentam encontrar caminhos eficientes para envio da mensagem entre dois nós, possivelmente utilizando comunicação multi-saltos.

Já o tipo *broadcast* engloba o envio de mensagens para todos os nós da rede, ao passo que o tipo *multicast* engloba o envio para um subconjunto dos nós.

Quanto ao roteamento geográfico, a sua motivação principal é que muitas aplicações necessitam conhecer a localização do nó para atividades que requerem resposta a questões como: "todos os nós de uma determinada região" ou "todos os nós perto do ponto A". Por último, a mobilidade trata protocolos em que este tipo de fenômeno está presente.

Em relação às métricas de roteamento, elas são o componente de roteamento que calculam o peso das rotas, utilizando parâmetros relevantes. Assim, possuem uma grande parcela de responsabilidade na eficiência de um protocolo de roteamento.

Vários tipos de métricas foram propostos e estudados. Alguns mais comuns para RSSF são: Hop Count, ETX (DE COUTO et al., 2005), ETT (DRAVES; PADHYE, 2004), WCETT (DRAVES; PADHYE, 2004), MIC (YANG; WANG; KRAVETS, 2005) e iAware (SUBRAMANIAN et al., 2006). Em geral, a evolução das métricas é conseguida através da adição de parâmetros extras às métricas já existentes, onde cada novo parâmetro captura um aspecto adicional de informação, tais como largura de banda, interferência, etc (HU et al., 2008). A Figura 2.3 exemplifica as principais informações que podem ser exploradas no desenho de uma métrica de roteamento. Em geral, quanto mais informação uma métrica engloba, maior é o seu entendimento das características da rede, e, por conseguinte, maior é a habilidade do protocolo de roteamento em suportar roteamento adaptativo.

	Global	Link	Local	
Static			Interface	Node
			InterfaceType	NodeType PowerSource
Dynamic	HopCount*	LinkQuality*	LinkCongestion*	PowerLevel*
	Topology	LinkCapacity* (Bandwidth)	RetransmissionCnt*	NodeStatus
		LinkLoad	TransmissionCost	Mobility
		Delivery/Loss Ratio	Channel	TrustLevel
		Channel		GeoLocation
		NoiseLevel		

\* commonly used metric information

Figura 2.3 - Informações para Métricas. Fonte: (HU et al., 2008)

Como cada métrica de roteamento leva em consideração diferentes tipos de informação, e cada uma possui suas particularidades na forma de avaliação dos parâmetros, resulta que cada métrica possui características distintas e cada uma pode estar mais adequada a determinados cenários. Na Tabela 2.1, é possível notar as diferenças no modo de operação (cálculo) do roteamento, e a principal característica de desempenho atingida por algumas métricas.

Primary routing metric	Routing operation	Main Performance characteristic
HC	Shortest path	Low latency
ETX	More reliable link with respect to layer 2 performance	Low number of retransmission- lower end-to-end energy consumption
RSSI	More reliable link with respect to signal strength (does not take into account congestion for example)	Increased reliability with respect to transmissions successfully received by neighbours, high latency
PFI	More reliable neighbor	Efficient detection of misbehaving nodes, low packet loss
RE	Avoids systematically using the same neighbors for forwarding exhausting their energy	Network lifetime elongation

Figura 2.4 - Aspectos de desempenho para Métricas. Fonte: (ZAHARIADIS, 2012)

## 2.4. O Protocolo RPL

Diferentemente das redes sem fio tradicionais, as redes RSSF requerem sistemas operacionais otimizados para uso em um ambiente cujos nós roteadores são caracterizados por severas restrições em termos de capacidade de processamento, armazenamento e energia, e cujos enlaces de comunicação se destacam pela instabilidade, baixas taxas de transmissão e altas taxas de perdas de pacotes.

O projeto de protocolos de roteamento também é influenciado por esses e outros desafios das RSSF. Por exemplo, as funções de sensoriamento e a disseminação de dados em redes 6LoWPAN são geralmente dependentes das aplicações e podem apresentar restrições temporais críticas para envio dos dados para os nós sorvedouros (*sink*). Para lidar com todos esses desafios, o IETF propôs um protocolo de roteamento para redes 6LoWPAN, chamado RPL (RFC6550). O protocolo RPL foi projetado especificamente para lidar com essas características particulares das redes LLNs e minimizar o tráfego de controle.

O RPL é um protocolo do tipo vetor de distância (*distance vector*) adaptado para uma variedade de tipos de redes LLN. O protocolo suporta três categorias de padrões de tráfego. Na primeira, *multipoint-to-point*, os nós periodicamente enviam mensagens para um ponto de coleta específico, por exemplo um nó *sink*. Na segunda, *point-to-multipoint*, o tráfego originado em nó *sink* tem como destino dispositivos específicos dentro da LLN. Por último, a comunicação *point-to-point* também é suportada.

Uma série de conceitos foram introduzidos no RPL que o fazem um protocolo bastante flexível, embora relativamente complexo. A formação da topologia da rede é baseada no conceito topológico de Grafos Acíclicos Dirigidos (*Directed Acyclic Graph* - DAG), uma estrutura em forma de árvore que define rotas *default* entre nós da LLN. Para aumentar a confiabilidade, o RPL adota o conceito de diversidade espacial. Assim, diferentemente de uma árvore tradicional, onde um nó está associado a um único *parent*, em um grafo RPL um nó pode estar associado a múltiplos potenciais *parents*, criando caminhos alternativos em direção a um nó destino.

A organização dos nós se dá em um conjunto de *Destination Oriented DAGs* (DODAGs). Nos DODAGs, os nós de maior visibilidade, como os nós *sink* e os nós *gateway*, são as raízes dos DAGs. Uma instância do protocolo RPL (*RPL Instance*),

identificada univocamente por um RPLInstanceID, pode conter múltiplos DODAGs, cada um deles possuindo um único DODAGID. Assim, uma combinação de um RPLInstanceID e de um DODAGID caracteriza univocamente um nó DODAG na rede. Cabe observar que um nó pode se unir a múltiplas instâncias RPL, mas deve pertencer a apenas um único DODAG a cada instância. Além disso, uma rede LLN pode ter múltiplas instâncias RPL executando concorrentemente. A Figura 2.5 ilustra esses conceitos.

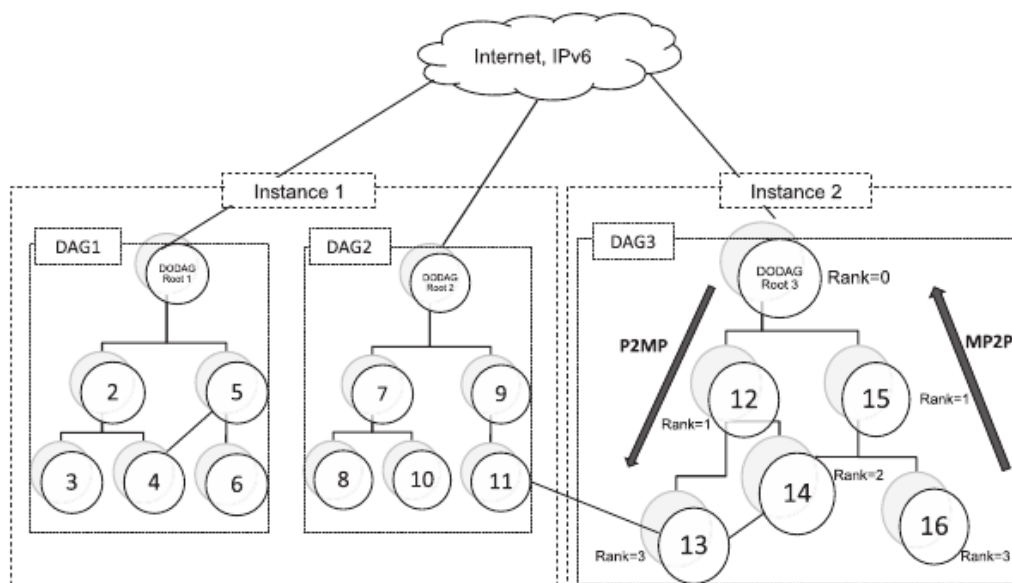


Figura 2.5 - Rede RPL com três DODAGs e duas instâncias. Fonte: (GADDOUR et al., 2012)

O RPL suporta aplicações com diferentes requisitos por meio da definição de Funções Objetivo (*Object Functions - OFs*). Basicamente, uma OF especifica como o RPL deve selecionar *parents* e possíveis sucessores, ou seja, como selecionar caminhos no DODAG. As *OFs* definem de que maneira métricas de roteamento e funções relacionadas são empregadas pelos nós para computarem o seu *Rank* dentro de uma versão do DODAG (*DODAG Version*). Em última instância, as *OFs* restringem ou otimizam as rotas selecionadas. Todos os DODAGs dentro de uma mesma instância RPL devem usar uma mesma *OF*).

Existe também o conceito de *Rank*, que é um número calculado nos nós roteadores e que identifica a sua posição em relação aos outros nós e à raiz do DODAG. O uso de *Ranks* previne *loops* de roteamento e ajuda o nó a distinguir entre nós pais e

nós irmãos. Observa-se que é possível definir OFs por aplicação (ex: prover rotas sensíveis à latência para aplicações com requisitos de tempo real). Para garantir interoperabilidade de comunicação entre diferentes aplicações, o RPL provê uma OF default chamada OF0, a qual seleciona rotas com base no número de saltos até o nó raiz do DODAG. Além do RPLInstanceID, DODAGVersionGID e do Rank, o RPL usa um outro valor que, juntamente com esses três, serve para identificar e manter a topologia da rede: o em DODAGVersionNumber.

Para construir e manter o DODAG, o protocolo RPL define mensagens ICMPv6 denominadas *DODAG Information Object* (DIO) para a descoberta de vizinhos e o estabelecimento de rotas. Na formação da topologia, cada nó raiz constrói um pacote DIO e o envia para todos os filhos. Qualquer filho que decide se juntar ao DAG repassa o DIO adiante, para os seus próprios filhos. O DIO contém um valor do Rank do nó, que é incrementado quando o filho se junta ao DAG. Nós podem armazenar um conjunto de pais e irmãos candidatos, que podem ser usados se o pai preferencial está incapacitado de rotear tráfego. Para indicação de qual OF utilizar, é utilizado um campo de cabeçalho indicada dentro de uma mensagem DIO - *Objective Code Point* (OCP).

A propagação de rotas é implementada pelo RPL através do algoritmo Trickle (RFC 6206), que manipula os *timers* visando acomodar rapidamente as mudanças de estado do roteamento. O algoritmo *Trickle* escalona o envio de mensagens DIO visando sempre minimizar a quantidade de DIOs transmitidos e garantir um tempo de convergência baixo para a rede.

Como redes LLN são bastante dinâmicas, o RPL também fornece facilidades para incorporar métricas de natureza dinâmica, tais como a ETX (*Estimated number of Transmissions*) (DE COUTO, 2005), que facilita a descoberta de caminhos de maior vazão e minimiza o número total de transmissões requeridas para a transmissão de um pacote até o seu destino.

Além disso, se um nó detecta a inexistência de uma rota em direção à raiz, um mecanismo de reparo local (*Local Repair*) é acionado para encontrar uma rota alternativa. Um mecanismo de reparo global (*Global Repair*), que atinge a topologia de toda a rede, também é fornecido pelo RPL e pode ser executado pelo nó raiz quando reparos locais sucessivos levarem eventualmente a uma topologia da árvore que não seja mais eficiente (KORTE; SEHGAL; SCHÖNWÄLDER, 2012).

A principal característica deste protocolo é que, além de ser um padrão e recomendação IETF para utilização em redes LLNs – o que facilita e promove a interoperabilidade entre sistemas – o protocolo é robusto no sentido que a lógica do protocolo está disassociada das métricas. As RFCs que especificam o RPL descrevem todo o processo de construção da árvore topológica, e deixam para o implementador a tarefa de especificar quais métricas utilizar. Além disso, os principais sistemas operacionais para RSSFs (TinyOS e ContikiOS) suportam e recomendam o seu uso.

## 2.5. Sensibilidade ao Contexto

*Contexto* tem sido objeto de investigação científica há vários anos em algumas comunidades científicas, como Linguística e Psicologia Cognitiva. Na comunidade de Ciência da Computação, os estudos sobre o tema são mais recentes; porém, pode-se observar importantes contribuições para o seu entendimento e formalização já há alguns anos, particularmente em trabalhos da área de inteligência artificial (MCCARTHY, 1997).

No dicionário Houaiss a palavra “contexto” significa a “inter-relação de circunstâncias que acompanham um fato ou uma situação”. A abrangência desse conceito leva a entender que, intuitivamente, para os sistemas computacionais, contexto pode ser entendido como tudo que está ao redor de um sistema em questão, tudo que ocorre em um determinado ambiente. O termo possui uma grande variedade de significados na literatura, dependendo do propósito da aplicação particular e/ou do ponto de vista da comunidade científica em questão. (SCHILIT, 1995), em um dos trabalhos pioneiros da área, enumera exemplos de contextos em três categorias: (i) contexto computacional (conectividade de rede, custos de comunicação, largura de banda e recursos disponíveis); (ii) contexto do usuário (perfil, localização); e (iii) contexto físico (luminosidade, níveis de ruídos, condições do trânsito e temperatura).

Várias outras definições podem ser encontradas na literatura de Computação Ubíqua, sendo a mais referenciada aquela apresentada por (ABOWD, 2000):



*Contexto é qualquer informação que pode ser usada para caracterizar uma situação de uma entidade. Uma entidade é uma pessoa, um lugar, ou um objeto que é considerado relevante para a interação entre um usuário e uma aplicação, incluindo o próprio usuário e a própria aplicação.*

Freqüentemente usadas como sinônimo de contexto, as informações contextuais são as informações que caracterizam um determinado contexto. São elas as informações relevantes para se determinar o estado atual de um contexto. Por exemplo, para o contexto localização, as informações contextuais que o caracterizam podem ser a latitude e a longitude de uma entidade.

Segundo (HENRICKSEN; INDULSKA, 2002), existe uma série de características acerca da natureza da informação contextual que provocam um grande impacto no projeto de uma plataforma sensível ao contexto, e que determinam alguns dos seus requisitos:

- (i) a informação contextual é intrinsecamente imperfeita;
- (ii) a informação contextual é altamente inter-relacionada;
- (iii) o contexto possui várias representações alternativas (e.g. existe um *gap* significativo entre a saída de um sensor e o nível de informação que é útil para as aplicações);
- (iv) a informação contextual disponibilizada pela plataforma tem validade temporal, isto é, depende de quando os dados foram obtidos.

No domínio das redes de sensores sem fio, variadas informações como características do próprio nó sensor, do ambiente que o cerca e de suas capacidades; podem ser utilizadas para fonte de informações contextuais. Alguns grupos em que essas fontes podem ser classificadas são:

- Energia (SHAH; RABAEY, 2002);
- Mobilidade (MASCOLO; MUSOLESI, 2006) (MUSOLESI; HAILES, 2005);
- Informação (LIU; ZAO, 2003);
- Privacidade (WALTERS, 2007);

- Qualidade de serviço (QoS - Quality of Service) (CHEN; VARSHNEY, 2004).

O grupo Energia explora informações de contexto relacionadas à energia do nó sensor, como nível de bateria, custo de transmissão para determinado pacote, conectividade, etc. Exemplificando, em um nó sensor com um nível de bateria baixo, sua energia possivelmente se esgotará rapidamente, ocasionando perda de pacotes na rede. O grupo Mobilidade inclui informações de localização do nó sensor, que podem ser utilizadas por algoritmos de roteamento adaptativo. O grupo Informação contém atributos relacionados às capacidades dos nós sensores de aquisição do dado sensorado. Como exemplo de atributos deste grupo, pode-se citar agregação e compressão de dados. O grupo Segurança possui atributos que tratam de aspetos como criptografia, autenticação e controle de acesso. O último grupo indica atributos de qualidade de serviço. Reserva de recursos, como energia, largura de banda, dentre outros, buscam garantir a qualidade de serviço requerida. Como exemplo, têm-se alguns nós que monitoram informações críticas ou multimídia e possuem a necessidade de QoS.

Atualmente existe um conjunto de trabalhos preocupados com a questão da sensibilidade ao contexto. Nas RSSF, esse suporte acontece em vários campos de pesquisa, inclusive no roteamento. No entanto, as abordagens de roteamento sensível ao contexto são distribuídas. Cada nó localmente processa as informações contextuais e os testes são realizados em ambientes assumidamente densos, com um pré-requisito forte em questões de restrições de recursos dos nós, o que pode representar o melhor cenário para alguns desses trabalhos, mas não é aplicável a todas as aplicações de RSSF.

Outra opção seria adotar uma abordagem centralizada para avaliação das informações contextuais. Desta forma, pode-se obter uma visão de cada nó individual, e também uma visão abrangente da rede (enxergando-a como um todo). Neste trabalho, esta abordagem se mostra mais adequada, já que devido às limitações de recursos computacionais, em alguns casos é um grande desafio fazer com todos os nós sensores cooperem e troquem informações contextuais (especificamente no caso do protocolo RPL, em que todos os nós utilizam uma métrica de roteamento em comum, a qual é provida pelo nó *sink*), e é este o foco do presente trabalho.

## 2.6. Reconfiguração Dinâmica

Algumas categorias de redes de sensores requerem a instalação de centenas de nós em locais remotos e de difícil acesso. Além disso, em algumas situações, o desenvolvimento da aplicação ao longo do tempo requer a sua reconfiguração, seja de um nó sensor individual ou da rede como um todo. A incapacidade de endereçar estas questões põe em risco a continuidade da operação da rede. Como exemplo, pode-se citar a correção de bugs (KIM, 2009), atualização de código (KULKARNI, 2009), parâmetros de segurança (PORTILLA, 2010), dentre outros.

Há muitas abordagens para se atingir a reconfiguração em redes de sensores. A alteração de funcionalidades na rede, seja ela em tempo real (*real-time*) ou tempo de projeto (*design-time*) abrange realizar mudanças em componentes de hardware e/ou mudanças em componentes de software. Em alguns trabalhos esse processo é citado como reprogramação (LEVIS, 2002)(SUN, 2010) enquanto em outros é citado como reconfiguração (KULKARNI et al., 2009)(MURALIDHAR; RAO, 2008). A diferença reside no fato de que enquanto mudanças somente no software do nó sensor são efetuadas, é denominado reprogramação, mudanças em componentes de hardware são denominadas reconfiguração. Outros termos comuns na literatura são atualização (CHONG, 2003) e adaptação (HAN, 2005). Nesta dissertação, não será feita distinção entre os termos reconfiguração, reprogramação e adaptação; e a interpretação será que os dois referem-se ao processo de efetuar uma mudança no nó sensor, que pode alterar a sua funcionalidade inicial.

Mudanças relacionadas ao software do nó sensor podem acontecer de diferentes formas. Uma primeira maneira de se efetuar a reconfiguração é através da camada de aplicação. Mudanças podem ocorrer na camada de aplicação para adequação das necessidades da mesma, e envolvem a adição, remoção e edição de constantes, variáveis e funções. Em alguns casos a utilização de diretivas de compilação como "#define" e "if" são usadas como seletores para ignorar trechos de código, bibliotecas de funções e módulos de acordo com as especificações da aplicação. Este tipo de configuração é aplicável somente em tempo de projeto. Para qualquer mudança de parâmetros, é necessário a recompilação e reimplantação do código nos nós sensores, de modo que a flexibilidade é limitada.

No entanto, há meios de superar essa limitação caso o projetista tenha (ou tente reproduzir) a experiência de campo e possa, durante o projeto do código fonte, reproduzir os aspectos relevantes e o comportamento esperado da aplicação em situações adversas. Desse modo, o próprio software deve monitorar o ambiente e se adaptar de acordo com as características do mesmo, sendo por definição proativo.

Para o nó sensor se adaptar a alguma condição, é necessário que o mesmo esteja ciente do ambiente em que está inserido. Isto quer dizer que o mesmo deve usar monitoramento para acompanhar as mudanças no ambiente, e o que é monitorado pode incluir desde uma coleção de parâmetros estatísticos sobre a rede até inspeccionamento do próprio hardware. O que monitorar e quando monitorar depende do tipo de adaptação disponível e dos objetivos da aplicação.

Tradicionalmente, a pilha de comunicação é descrita como uma arquitetura em camadas, em que cada camada implementa uma funcionalidade específica. É possível também que serviços de uma determinada camada realizem a reconfiguração de si mesmos. Um exemplo é a adaptação do "*radio duty cycle*" que, em tempos de baixa atividade da rede, se auto-reconfigura para diminuir o consumo de energia. É importante observar neste exemplo que para realizar a reconfiguração o serviço utiliza informações da própria camada, sendo, por isso, também chamado de camada adaptativa.

É possível também que camadas tenham conhecimento sobre detalhes internos das outras camadas, e usem essas informações para a sua própria adaptação. Esse tipo de abordagem, chamado de "reconfiguração inter-camadas" (*cross-layer reconfiguration*) permite que as mesmas atuem em sincronia, mas ao custo de deixar o código difícil de manter e atualizar, já que a independência entre camadas é desfeita.

Outro tipo de reconfiguração possível é através de camadas de middleware. O middleware é uma camada de abstração de software que existe entre os Sistemas Operacionais e as aplicações, e tem por objetivo simplificar operações, habilitar a heterogeneidade e mascarar o hardware e camadas de software básico do nó sensor (GRAZIOSI, 2008). Também proveem mecanismos de abstração para comunicação de rede, linguagens de programação e gerenciamento de aplicações distribuídas, através do uso de API (CHONG, 2003). Como exemplo, os nós sensores (aplicações) podem definir um conjunto de operações básicas (atualizar, incrementar, apagar, etc.) e receber comandos para execução dessas operações através de uma interface (o middleware utilizando comunicação de rede acessa o nó e realiza a mudança pretendida).

Também é possível realizar reconfiguração por meio do sistema operacional, com a adição, remoção ou edição de código. Após aferir o desempenho da rede de sensores e descobrir quais parâmetros devem ser alterados, um código de aplicação atualizado é enviado para a rede. Esta solução parece flexível, mas é muito onerosa em termos do consumo de energia necessário e pode levar um longo tempo de acordo com a quantidade de nós sensores e do mecanismo de comunicação disponível. A tarefa de como reprogramar os nós sensores é um próprio tema de pesquisa em si, e essas alterações podem ser efetuadas de algumas formas, como sobrescrever a imagem (código) inteira do nó sensor com uma nova, sobrescrever somente as diferenças identificadas entre a versão antiga e a nova da imagem; ou ainda através de módulos carregáveis, o que significa que o sistema operacional necessita dar suporte a esta abordagem.

Quanto ao hardware do nó sensor, a reconfiguração depende da natureza dos elementos de processamento (componentes que realizam o controle e processamento no nó sensor) que eles possuem (PORTILLA, 2010) (LELIGOU, 2011), como FPGA (*Field Programmable Gate Array*), SoPC (*System Programmable Chip*), dentre outros.

Apesar de um processo visando à melhoria dos serviços e operação da rede, a reconfiguração, dependendo do cenário, possui impeditivos e dificultadores, sendo os principais deles o espaço de memória e o consumo de energia. Dependendo do método empregado na reconfiguração, é possível que seja consumida uma porção grande de memória do nó sensor enquanto a imagem e/ou atualizações são armazenadas, e memória é um recurso escasso. Outra questão se refere ao consumo de energia, já que é necessária a transmissão da atualização/imagem para a rede, neste caso para cada nó individualmente.

Existem algumas iniciativas de reconfiguração em RSSF, abrangendo principalmente aspectos de otimização da rede (como diminuição do consumo de energia e maximização do tempo de vida útil da rede) e qualidade de serviço. No próximo capítulo serão exploradas algumas dessas iniciativas, à luz dos conceitos discutidos neste capítulo.

### 3. Trabalhos Relacionados

Este capítulo apresenta alguns exemplos de trabalhos correlatos, que tratam de duas questões de pesquisa relacionadas aos temas centrais desta dissertação: reconfiguração dinâmica e roteamento. O capítulo é dividido em duas seções: a Seção 3.1 descreve trabalhos que utilizam o protocolo de roteamento RPL como referência para roteamento e que introduzem alguma forma de adaptação; a Seção 3.2 apresenta alguns trabalhos em infraestruturas de middleware adaptativos bem como abordagens que propõem o uso de adaptação na própria lógica dos protocolos.

#### 3.1. Adaptação do Protocolo RPL

Em (ZAHARIADIS et al., 2012) é proposta uma abordagem que sugere o desacoplamento do nó sensor físico com a aplicação que executa no mesmo. Este conceito é chamado de *Virtual Sensor Networks (VSNs)*. Através da idéia de virtualizar um substrato de rede (recursos de hardware e enlace de comunicação), múltiplas topologias virtuais com diferentes características podem ser criadas e coexistir no mesmo hardware físico. Assim, é explorado como o protocolo RPL se adequa a este cenário e é sugerida a criação de diferentes instâncias de roteamento RPL por aplicação e o uso de diferentes métricas de roteamento por instância.

Na Figura 3.1 é possível observar que diferentes caminhos (nó sensor – *sink*) podem ser construídos para realizar diferenciação de serviços, otimizando aspectos de desempenho diferentes.

Qualquer requisição de usuário para criação de um recurso virtualizado é examinada pelo VGM (*VITRO Gateway Manager* - Figura 3.2) para verificação de disponibilidade de recursos, e a requisição somente é aceita caso a RSSF possua recursos suficientes para atender à demanda. Após aceitar a requisição, os requerimentos de roteamento para o novo recurso são analisados para determinar se uma nova instância de roteamento é necessária ou alguma já existente satisfaz. Se uma nova instância de roteamento é requerida, a função objetivo que atende aos requisitos do usuário é derivada e uma nova instância do RPL é criada, e armazenada no repositório.

Na situação em que é emitida uma requisição, há recursos disponíveis, mas os requerimentos de roteamento não podem ser garantidos (nenhuma função objetivo atende), a requisição é rejeitada.

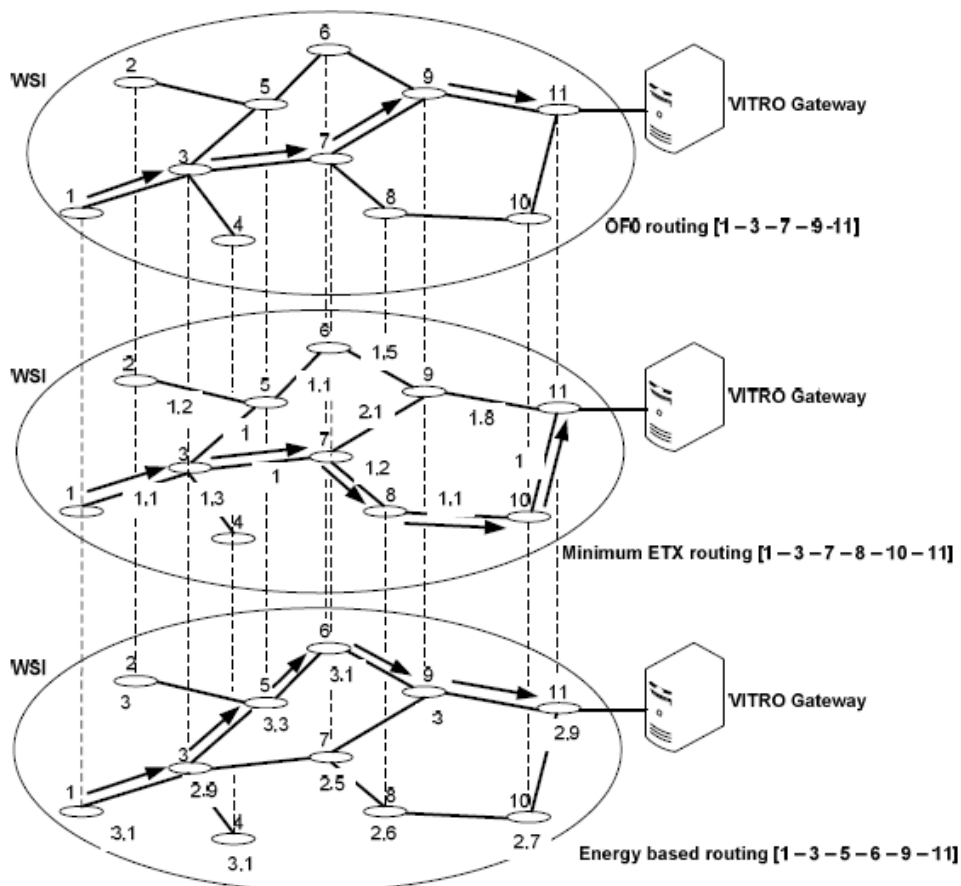
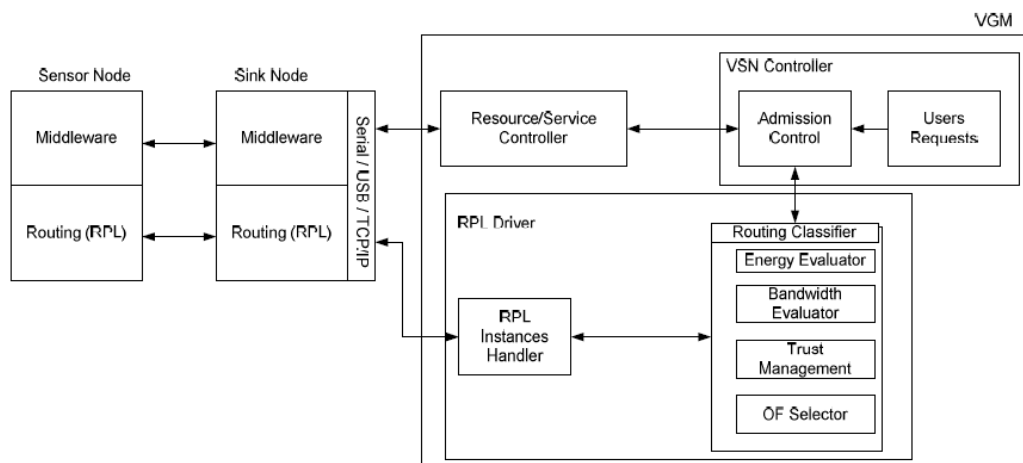


Figura 3.1 - Múltiplas Instâncias na mesma rede física. Fonte: ( ZAHARIADIS)

O módulo de classificação de rotas (*Routing Classifier*) é responsável por estimar a disponibilidade de recursos da rede e avaliar o impacto da adição de uma nova instância de roteamento em parâmetros como largura de banda e consumo de bateria dos nós.



**Figura 3.2 - Criação e Gerenciamento de Instâncias RPL**

Em (GADDOUR et al., 2014), é proposta a utilização de lógica fuzzy para cálculo da função objetivo para o protocolo RPL. Na especificação do RPL, o componente responsável por selecionar os caminhos é a função objetivo (OF *Objective Function*). A função permite selecionar o nó pai preferencial dentro um conjunto de candidatos. No entanto, a especificação não define quais conjuntos de métricas se deve utilizar na seleção dos melhores caminhos, deixando este item aberto aos implementadores.

Os autores propõem uma função objetivo, chamada por eles de OF-FL (*Fuzzy Logic*), a fim de computar o melhor caminho ao nó sink considerando um conjunto de parâmetros. Os parâmetros escolhidos foram: Número de Saltos (*Hop Count*), Tempo de Atraso (fim-a-fim), Taxa de Perda de Pacotes, e Taxa de Mudanças na Rota Default.

Esses parâmetros são representados como variáveis linguísticas e combinadas com um conjunto de regras fuzzy, e a avaliação destas regras origina a classificação de um nó vizinho (em termos de qualidade como "excelente", "muito bom", "bom", "ruim", etc).



Hop Count	End-to-end delay	Battery level	ETX	Quality
Near	low	high	short	Excellent
Near	low	high	average	very good
Near	low	average	short	very good
Near	average	high	long	good
Vicinity	low	average	average	good
Vicinity	average	high	short	low good
Vicinity	high	low	average	bad
Vicinity	average	average	long	bad
Far	high	high	average	low bad
Far	high	average	short	low bad
Far	average	low	average	awful
Far	high	low	long	awful

**Figura 3.3 - Regras Fuzzy**

Em (SHARKAWY et al., 2014) é proposta uma função objetivo sensível ao contexto (CAOF *Context Aware Objective Function*) que leva em consideração a quantidade de recursos do nó sensor e suas mudanças temporais. A abordagem é proposta inicialmente nos protocolos de roteamento CAR e SCAR, e foi incorporada no RPL neste trabalho (SHARKAWY et al., 2014).

A função proposta (Figura 3.4) habilita os nós a selecionarem os nós pai (*rota default*) avaliando o grau de conectividade do nó sensor (*cdc - change degree of connectivity*), o nível de bateria disponível (*batt - battery level*) e a sua localização relativa na árvore topológica (*colloc - colocation with sink*). Para isso, é efetuada uma soma ponderada de alguns termos.

Cada um desses atributos é representado por uma função "U". Cada nó sensor calcula localmente os seus atributos. A probabilidade de entrega "P" (*Delivery Probability*) de um nó sensor  $i$ , é a soma das funções individuais de contexto, e "W" são os pesos que refletem a importância de um determinado atributo na definição de contexto do nó.

$$P(n_i) = W_{cdc} * U_{cdc}(n_i) + W_{coloc} * U_{coloc}(n_i) + W_{batt} * U_{batt}(n_i)$$

**Figura 3.4 - Função de Soma**

### 3.2. Outros Trabalhos Relacionados

Em (DE JONG et al., 2009) é apresentado o componente de middleware MoMi, que utiliza um sistema baseado em regras para detectar nós falhos em uma rede de sensores sem fio. Em cada nó, são feitas observações sobre informações locais e a respeito de nós vizinhos. Essas informações são comparadas umas com as outras, seguindo regras pré-estabelecidas, e as observações conflitantes são enviadas ao gateway, que gera uma predição dos nós potencialmente sujeitos a falha. As conclusões, no entanto, não conduzem à reconfiguração efetivamente dos nós, mas devem ser utilizadas em conjunto com outras solução de reconfiguração.

Em (TERFLOTH et al., 2006) é proposto o middleware FACTS, e é adotada uma abordagem utilizando sistemas especialistas (*expert systems*) para prover raciocínio sobre reconfiguração. Nos sistemas especialistas, a informação necessária é representada por fatos. Um repositório de regras é utilizado para inferir novas informações ou determinar que ações são adequadas. As regras são definidas em um formato próprio (RDL), que são compiladas pelo ambiente de execução FACTS e o produto é um bytecode que é interpretado pelo nó sensor.

Em (RFC 6206) é definido o protocolo Trickle. Este é um protocolo para propagação e disseminação de atualização de código nas RSSF, e utiliza *beacons* para propagar a versão corrente de código disponível, e detectar quando uma nova versão está disponível. O intervalo em que o *beacon* é enviado, é adaptado de acordo com as atualizações. Por exemplo, atualizações frequentes resultam em pequenos intervalos, atualização não-frequentes resultam em grandes intervalos (intervalo mínimo e máximo, há outros tipos de intervalos).

O mecanismo de *radio duty cycle* tem o objetivo de efetuar economia de energia nos nós sensores, desligando o rádio em momentos de ociosidade da rede. O protocolo X-MAC (BUETTNER et al., 2006) é um exemplo em que, assincronamente, os nós periodicamente ligam o rádio e verificam se há alguma transmissão a ocorrer. A frequência com que o rádio é ligado determina a eficiência de energia e latência na rede. Em (BUETTNER et al., 2006) é proposto uma versão adaptativa do X-MAC em que é

possível adaptar o intervalo para ligar o rádio de acordo com a carga da rede, definindo intervalos mínimos e máximos.

Em (DONG et al., 2010) é apresentado o DPLC (*Dynamic Packet Length Control*), um esquema com o intuito de melhorar o desempenho da rede de sensores, ajustando de forma dinâmica o tamanho do pacote de dados que o rádio enviará. Para avaliar o impacto que o tamanho dos pacotes tem no desempenho de uma rede de sensores, foram efetuados testes de comunicação entre dois nós sensores variando o tamanho do pacote de dados. Segundo os autores, à medida que a carga do pacote aumenta, o desempenho da rede aumenta até um valor considerado ótimo (no experimento em questão este valor corresponde a cerca de 30 bytes), e após isto, o desempenho diminui. Além disso, variando-se a distância e a potência da transmissão, o valor ótimo também varia. Assim, os autores advogam que o ajuste dinâmico de tamanho do pacote de dados de acordo com as condições do meio físico (qualidade do enlace de comunicação) é uma otimização benéfica.

O DPLC funciona de forma que, ao receber uma mensagem da aplicação no nó sensor, é feita uma avaliação do tamanho desta mensagem. Se a mensagem a ser enviada é pequena, é utilizado um serviço de agregação, caso contrário é utilizado um serviço de fragmentação (mensagem maior do que o tamanho máximo de pacote suportado pelo rádio do nó sensor - 128 bytes para rádios CC2420). Um módulo chamado *Link Estimator* dinamicamente calcula o tamanho do pacote (tamanho ótimo) para transmissão. Segundo este cálculo, o DPLC pode decidir quantas mensagens devem ser agregadas, ou em quantos pacotes uma mensagem deve ser fragmentada. Quando pacote está pronto para transmissão, ele é enviado à camada MAC. O processo inverso ocorre no receptor, que desagrega ou desfragmenta o pacote a fim de obter a mensagem original.

Em (ALI et al., 2006) é proposto um esquema de roteamento denominado RTPA (*Real Time Communication with Power Adaptation*), que busca gerenciar o compromisso entre vazão (*throughput*) e atraso (*delay*) sacrificando a vazão em favor de baixos atrasos, com o objetivo otimizar aplicações de tempo real em redes de sensores. O intuito é realizar a entrega de pacotes que possuam prazo de entrega fim-a-fim (requisito temporal), enquanto minimiza o consumo de energia da rede de sensores. Por exemplo, transmitir um pacote utilizando uma potência de rádio alta pode aumentar a

área de cobertura da comunicação sem fio e/ou melhorar a qualidade do enlace, e também reduzir o atraso de comunicação, porém ao custo de aumentar o consumo de energia da rede. O RTPA propõe a adaptação da potência de transmissão dos rádios de forma dinamicamente, de acordo com os requisitos de tempo das aplicações e qualidade dos enlaces de comunicação entre as origens e os destinos, de forma que atenda aos requisitos temporais sem consumir excessiva a energia dos nós sensores.

Em (JAYARAMAN; DELIR HAGHIGHI, 2013) é proposto uma abordagem de reconfiguração sensível a situações (*situation-aware*), que adapta a operação do nó sensor de acordo com as situações correntes, situações estas que são depreendidas a partir dos dados obtidos na rede de sensores. Por situações pode-se entender uma conjuntura de acontecimentos, um cenário do mundo real que ocorre dentro do ambiente monitorado, como chuva, umidade e temperatura.

A abordagem proposta, denominada SA-A-WSN (*Situation-Aware Adaptation Approach for Energy Conservation in Wireless Sensor Network*) tem por objetivo maximizar o tempo de vida útil dos nós sensores, ao mesmo tempo em que mantém um nível aceitável de exatidão dos dados sensoreados, dependendo dos requerimentos da aplicação. Por exemplo, em uma situação crítica (ameaça de incêndio), uma aplicação pode necessitar de uma maior taxa de amostragem dos dados sensoreados, enquanto que em uma situação não crítica (sem ameaça de incêndio) a aplicação pode funcionar com uma taxa de amostragem inferior. A situação crítica (hipotética) do exemplo poderia ser caracterizada como: temperatura alta, velocidade do vento alta, baixa umidade do ar. Desta forma, o SA-A-WSN utiliza os dados sensoreados para inferir uma situação potencial e assim, controlar a operação do nó sensor e reduzir o consumo de energia da rede.

A Figura 3.5 retrata a arquitetura do SA-A-WSN, que é composta por dois componentes principais: SPOT-SAA (que executa no nó sensor) e SINK-AA (que executa no nó sink). O SPOT-SAA realiza o sensoreamento e envia os dados para o nó sink. O componente *SPOT-Scheduler* é um dos principais e é responsável por controlar os módulos de sensoreamento (*sensing module*) e de rádio (*radio module*), agendando o *duty cycle* do nó sensor de acordo com as informações enviadas pelo nó sink. O SINK-AA é responsável por receber os dados oriundos dos nós sensores da rede. Alguns destes dados (que são utilizados como informações contextuais) são passados ao

módulo *Context Collector*, que efetua uma pré-processamento desses dados, agrega, e os passa ao módulo de inferência (*Situation Inference Engine*). Este por sua vez, realiza raciocínio a fim de deduzir situações. Por fim, as regras que são utilizadas pela máquina de inferência são definidas utilizando lógica fuzzy (*Situation Rules*).

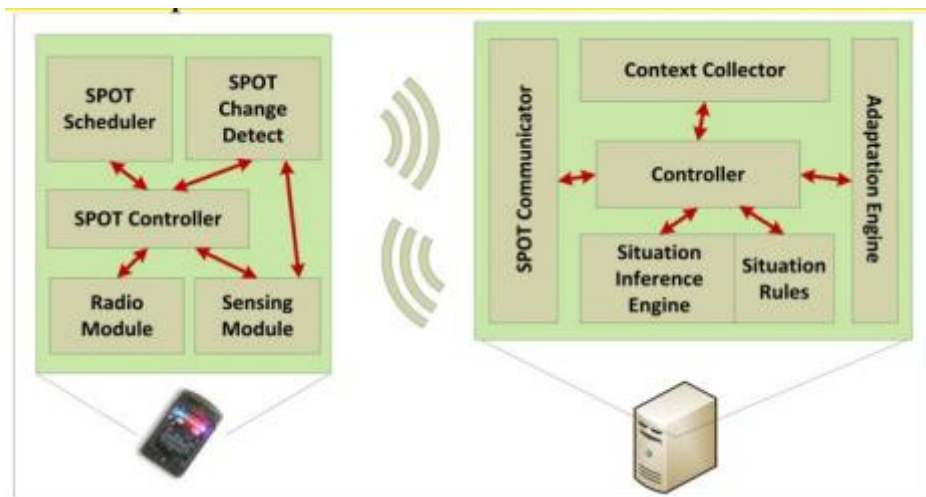


Figura 3.5 - Arquitetura SA-A-WSN

Neste capítulo foram apresentados trabalhos da literatura que propõem algum tipo de mecanismo de reconfiguração, seja ele através de middleware ou de lógica inserida no próprio algoritmo/sensor. No que se refere à provisão de infraestruturas para reconfiguração de redes de sensores sem fio, levando-se em consideração o tema sensibilidade ao contexto e o protocolo de roteamento RPL, percebe-se, pela análise da literatura da área, uma carência de ferramentas para atingir estes objetivos, principalmente quando se trata do tema middleware.

A análise desta amostra de trabalhos permitiu reforçar a importância de infraestruturas como a proposta nesta dissertação. Esta análise permitiu ainda avaliar opções de metodologias de reconfiguração e tecnologias de sistemas baseados em regras, algumas das quais foram empregadas na implementação da arquitetura conceitual, a qual será apresentada em detalhes nos próximos capítulos.

## 4. Arquitetura Proposta

Este capítulo apresenta o projeto do serviço de reconfiguração dinâmica de roteamento em RSSF proposto neste trabalho. O capítulo se inicia com um breve resumo dos problemas que motivaram a proposição da infraestrutura (Seção 4.1) seguido de uma descrição dos requisitos que nortearam a definição da arquitetura conceitual (Seção 4.2). A Seção 4.3 apresenta os componentes da arquitetura conceitual proposta.

### 4.1. Introdução

As aplicações para redes de sensores sem fio geralmente são projetadas com objetivos específicos bem definidos, tais como tempo de vida, vazão (*throughput*), latência e confiabilidade, os quais devem ser monitorados e atendidos de alguma forma. Há também a possibilidade destes requisitos mudarem ao longo do tempo de vida da aplicação, ou mesmo por um curto período de tempo, o que vai exigir da infraestrutura montada alguma forma de ajuste ao novo cenário – em outras palavras, alguma ação de reconfiguração da rede. Assim, a reconfiguração da RSSF tem por objetivo (i) atender aos objetivos da aplicação; e (ii) maximizar a desempenho da rede.

Conforme mencionado em capítulos anteriores, uma abordagem comum ao desenvolvimento de aplicações para RSSF é utilizar configurações estáticas nos nós, de acordo com as características esperadas para o funcionamento da rede; porém, em muitos casos, esta pode não ser a melhor abordagem pois os requisitos podem mudar no decorrer do tempo de vida da rede. Além disso, é difícil prever quais os parâmetros adequados para operação da rede. Parâmetros considerados ótimos e utilizados em simuladores e testbeds podem não refletir a realidade de uma RSSF.

Com base nessas e outras considerações acerca da dinamicidade do comportamento da rede e suas aplicações, as próximas seções descrevem a infraestrutura criada para dar suporte à reconfiguração dinâmica em RSSF. O projeto da arquitetura conceitual proposta tem por base um conjunto de princípios e requisitos funcionais e não funcionais, os quais são descritos nas próximas seções. O foco está em especificar um serviço de middleware para a categoria de Gerência de Dispositivos,

mais especificamente a reconfiguração dinâmica de parâmetros de comunicação do nó sensor. O trabalho concentra-se no projeto e implementação de um serviço de middleware que provê raciocínio (*reasoning*) sobre o momento oportuno para efetuar a reconfiguração de roteamento dos nós sensores, de acordo com algumas características da rede.

Observa-se que uma arquitetura de middleware genérica para RSSF, pode focar em uma série de funcionalidades, por exemplo, conectividade e comunicação, gerência de dispositivos, coleta de dados (análise e atuação), segurança, etc. Uma arquitetura de middleware modular e escalável, que suporte a adição e subtração de funcionalidades de forma estruturada e, conseqüentemente, vários casos de uso de RSSF, se torna uma ferramenta valiosa. O trabalho aqui proposto se insere neste contexto mais amplo, constituindo um dos serviços de uma arquitetura genérica de middleware, com várias outras funcionalidades que vem sendo vislumbrada no escopo dos trabalhos do grupo de redes de sensores do LPRM/UFES. Desta forma, deve ser destacado que as informações coletadas pelo serviço proposto serão de utilidade também para outros componentes do middleware em questão, diluindo-se, assim, o custo das informações de controle trafegadas na rede introduzidas pelo novo serviço.

## 4.2. Princípios e Requisitos

A arquitetura proposta está centrada nos seguintes princípios:

- **Visão holística da rede.** O serviço proposto deve se basear em uma visão holística da rede, ou seja, a rede deve ser vista como um todo, uma entidade que possui características e modo de funcionamento particular. Este paradigma está em oposição à visão usualmente adotada, em que cada nó sensor decide certos aspectos de configuração com uma visão limitada (a visão do próprio sensor inserido na rede).
- **Uso de uma abordagem de alto nível para representação das configurações e eventos em um ambiente de middleware.** A abordagem de alto nível deve possibilitar ao projetista da rede a definição de regras e suas respectivas ações. O fraco acoplamento entre as regras (middleware) e o nó sensor permite o envio de

ações ao nó e a sua reconfiguração, sem a necessidade de recompilação de código.

Este trabalho concentra em desenvolver um serviço que provê raciocínio (*reasoning*) sobre o momento oportuno de efetuar a reconfiguração de roteamento dos nós sensores, de acordo com algumas características da rede. Além desses princípios, a arquitetura proposta deve atender aos seguintes requisitos (alguns deles compartilhados com o middleware genérico):

- **R1.** *Adoção de padrões:* Utilizar tecnologias padrão IETF.
- **R2.** *Uso de regras para a especificação do comportamento da rede:* A construção de regras deve ser realizada por meio de um paradigma de programação flexível, que viabilize a sua descrição em alto nível de abstração. Deve ser possível a definição de um repositório de regras e eventos do domínio. O processamento das regras deve ser escalável e possuir um mecanismo capaz de mudar o comportamento da rede, por meio de interfaces padronizadas. O ambiente deve ainda, suportar uma forma de subscrição de dados obtidos da RSSF.
- **R3.** *Comunicação RSSF-Middleware:* Deve existir uma interface capaz de estabelecer e manter conexões seguras com o ambiente de RSSF. Esta interface deve ainda ser capaz de formatar os dados recebidos, para que possa ser armazenado e utilizado por outros serviços do middleware.
- **R3.** *Qualidade de Contexto:* É desejável que seja realizado o tratamento dos aspectos referentes à precisão dos dados recebidos das redes de sensores (gerenciamento de QoC - *Quality of Context*).
- **R3.** *Segurança:* O serviço de middleware deve suportar conexões seguras com a rede de sensores, sendo desejável que exista algum mecanismo de autenticação.

### 4.3. Arquitetura Conceitual

Considere o cenário apresentado na Figura 4.1. Durante o processo de desenvolvimento, o desenvolvedor determina os requisitos – por exemplo, que tipo de aplicação a rede executará e o que a rede deverá prover para que esta aplicação funcione satisfatoriamente – e define o desenho básico da rede – como será feito o roteamento e



que tipo de protocolos e métricas de roteamento serão utilizados. Isso resulta nos componentes que farão parte da rede, incluindo aqueles que poderão ser reconfigurados.

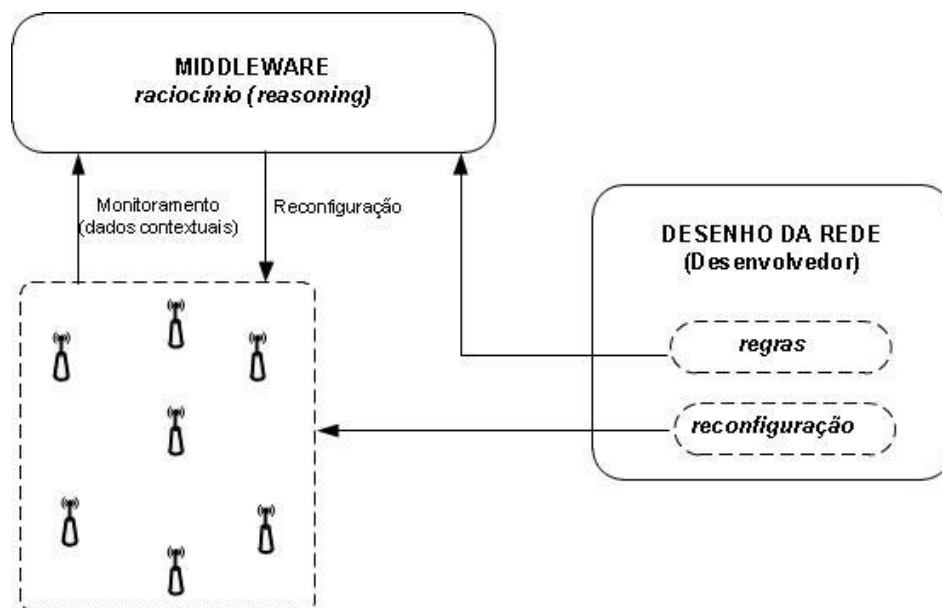


Figura 4.1 - Projeto de uma RSSF

A fim de efetuar raciocínio sobre quando reconfigurar quais componentes, o serviço de middleware requer conhecimento sobre a rede, como o domínio da aplicação, requisitos e características. Esse conhecimento pode vir do desenvolvedor, que providencia uma modelagem dessas informações, e também da própria rede, através dos nós sensores, os quais estão continuamente monitorando o seu contexto e aplicação, e alimentando o serviço de middleware em tempo de execução. Assim, o serviço pode combinar essas informações e inferir se a reconfiguração é necessária/desejável ou não.

Um requerimento para qualquer tipo de gerenciamento da rede é que este não deve consumir demasiadamente recursos dos nós. Isto se traduz diretamente em memória limitada, baixo poder de processamento e energia dos nós. Uma maneira de se realizar um serviço de reconfiguração é através de uma solução não centralizada, isto é, a informação é coletada e processada localmente no próprio nó sensor, o que permite a reconfiguração ser realizada sem depender de comunicação de rádio com outros nós. No entanto, isto significa que a reconfiguração somente poderá ser efetuada levando-se em conta uma visão limitada da rede, isto é, a visão que um determinado nó sensor tem de si mesmo e seu contexto, o que pode representar uma solução não ideal.

Neste trabalho é proposta uma solução de serviço de reconfiguração centralizada. Este tipo de solução oferece uma visão da rede como um todo e possibilita o compartilhamento de informações de tempo de execução entre os nós sensores, permitindo tomar ações de reconfiguração do comportamento da rede em si, em contraste com a reconfiguração de nós individuais da abordagem não centralizada.

É inegável que há um custo associado a este tipo de abordagem, pois os nós sensores terão que efetuar comunicação via rádio para prover o serviço de middleware com informações contextuais. No entanto, como já ressaltado, as informações coletadas serão de utilidade também para outros serviços do middleware genérico, diluindo o custo absoluto da abordagem proposta.

A Figura 4.2 mostra a arquitetura conceitual do serviço proposto. São identificados os componentes do serviço e os fluxos de dados da interação RSSF-Middleware, representados como Coleta de Dados e Aplicação de Política.

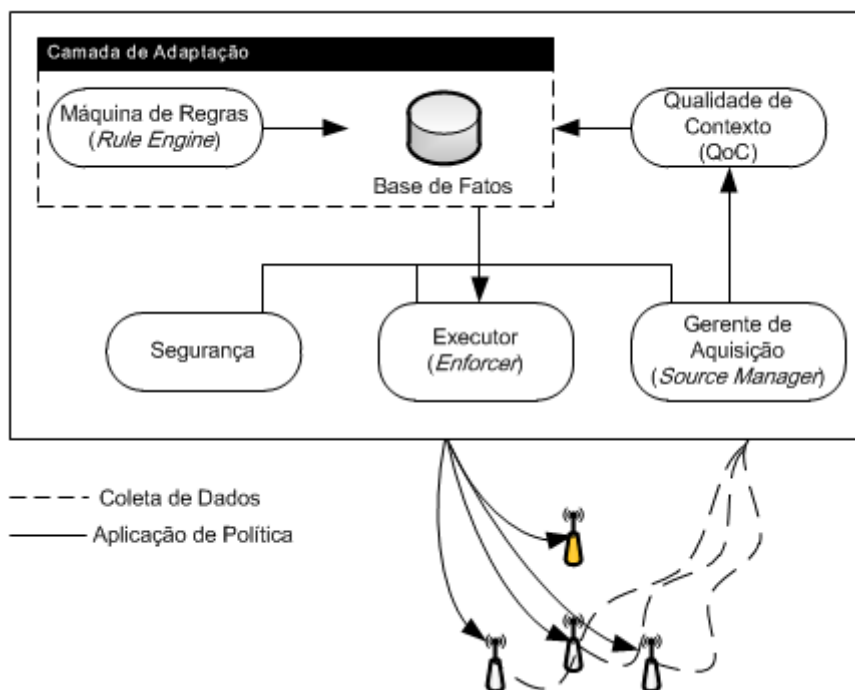


Figura 4.2 - Arquitetura Conceitual

Alguns aspectos relacionados a cada componente estão descritos a seguir.

- **Segurança**

- Este componente trata de questões relacionadas à segurança na comunicação entre o middleware e os nós sensores. Como se trata de aspectos de reconfiguração da rede, este componente exerce um papel importante, por exemplo, para evitar ataques do tipo *spoofing*. Desta forma, este componente deve prover um mecanismo de autenticação básica, para validar as mensagens que chegam ao serviço de middleware, e para que o nó sensor possa verificar a legitimidade de uma solicitação de reconfiguração.
- **Gerente de Aquisição** (*Source Manager*)
  - Este componente gerencia a comunicação entre o middleware e as fontes de dados contextuais (sensores).
- **Executor** (*Enforcer*)
  - Este componente realiza a “imposição” das políticas de configuração nos nós sensores. Deve prover mecanismos de versionamento (para atualização das configurações nos nós sensores).
- **QoC** (Qualidade de Contexto)
  - Este componente provê mecanismos para garantir a qualidade do dado coletado (informações contextuais). Idealmente, a rede de sensores pode ser formada por diferentes tipos de nós (hardware). Como forma de reduzir o custo, componentes baratos e de baixa qualidade podem ter sido utilizados na construção dos nós sensores, e como resultado, cada nó poderá ter sua leitura de sensoriamento ligeiramente diferente de outros nós. A tarefa deste componente é atuar como um filtro, e minimizar a influência de leituras incorretas na interpretação dos dados contextuais.
- **Camada de Adaptação**
  - Através de uma máquina de inferência, este componente utiliza dados de contexto a respeito da rede de sensores para inferir o comportamento da mesma. Regras de adaptação podem ser ativadas quando determinadas situações ocorrem (ou acabam), mudando os parâmetros de roteamento em tempo de execução.

### **4.3.1. Representação do Conhecimento**

A fim de efetuar raciocínio sobre quando reconfigurar quais componentes, o serviço de middleware requer conhecimento sobre a rede (domínio da aplicação, requisitos e características). Este conhecimento pode vir em tempo de execução da rede, ou no momento do desenho da rede.

De acordo com a Figura 4.1, informações sobre o comportamento da rede podem ser providas durante a modelagem da rede, através de um conjunto de regras que o serviço de middleware utilizará no raciocínio sobre reconfiguração. Outras informações são, por exemplo, quais tipos de métricas de roteamento os nós sensores suportarão (e, conseqüentemente, possuirão implementados no software). Essas informações são definidas antes da criação da rede de sensores, mas dependem diretamente das informações que serão providas em tempo de execução.

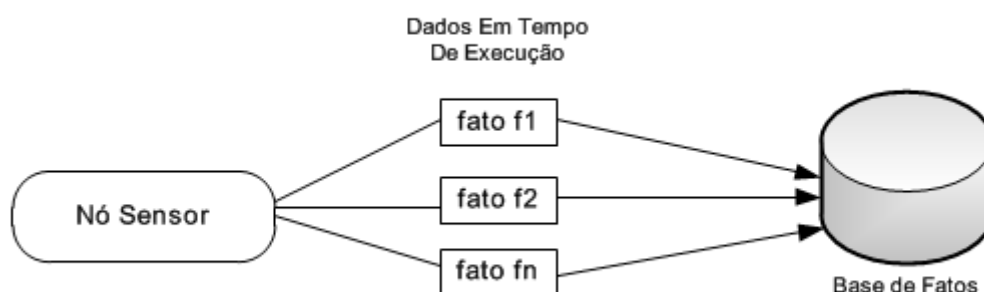
Durante a execução da rede, o serviço de middleware requer informações contextuais sobre a rede e os nós. Esse fluxo de informação é representado na Figura 4.2 pela seta nomeada monitoramento. Esses dados contêm detalhes sobre o ambiente sensoreado e o estado do hardware dos nós.

### **4.3.2. Dados Contextuais**

O serviço de reconfiguração depende dos dados contextuais da rede. Como esses dados são dinâmicos e englobam muitos elementos que são desconhecidos durante a fase de desenho da rede, eles devem ser monitorados em tempo de execução. O serviço deve casar essas informações coletadas em tempo de execução com as regras definidas pelo desenvolvedor.

Para que os dados contextuais sejam representados de uma maneira uniforme, é proposto que cada informação contextual de cada nó seja representado por uma variável. Essas variáveis poderão conter informações sobre o próprio nó sensor (e.g. nível da bateria) ou características da rede (e.g. quantidade de vizinhos de um determinado nó). Desta forma, o estado de um nó (ou da rede) em um determinado momento pode ser capturado através de um conjunto de variáveis.

Cada uma dessas variáveis está relacionada a um dado contextual de um nó sensor. Como forma de unificar essas informações, o serviço de middleware mantém um repositório, em que os nós sensores podem armazenar os dados através de uma interface padronizada. Os valores de cada elemento de contexto são armazenados como dados e serão referenciados como fatos, e o repositório em que a coleção de fatos estará armazenada será referenciada como Base de Fatos. A Figura 4.3 ilustra uma coleção dessas informações.



**Figura 4.3 - Base de Fatos**

O exemplo a seguir ilustra como um dado contextual de um nó é armazenado. Cada fato contém um nome representativo (tipo do dado), a identificação do nó sensor, o valor da variável e um metadado (*timestamp* da última atualização). Cada nó pode armazenar e atualizar fatos a qualquer momento, possivelmente sobrescrevendo valores prévios.

```
sensor = (sensor1, bateria, 60mAh, 15:50 – 08/08/2014)
```

```
wireless = (sensor2, RSSI, 63db, 15:50 – 08/08/2014)
```

**Figura 4.4 - Representação de Fatos**

### 4.3.3. Reconfiguração Condicional

Além do conhecimento dos dados contextuais da rede, o serviço também requer conhecimento de que tipo de reconfiguração deve ser feita e em que contexto. O desenvolvedor deve especificar o comportamento de reconfiguração usando uma linguagem de programação declarativa, pois facilita a descrição das regras. As regras são uma relação entre uma ou mais condições e uma ação. Se a conjunção lógica das condições for verdadeira, uma ação é lançada.

$$\text{condição1} \wedge \text{condição2} \rightarrow \text{ação}$$

**Figura 4.5 - Regra Condição-Ação**

A condição é uma função booleana que opera em um elemento contextual de um nó sensor (representado por um fato) e um valor. O operador de comparação pode ser binário (e.g. < ou >), possivelmente utilizar metadados (e.g. *later than*) ou mesmo ser unário.

Quando todas as condições de uma regra foram satisfeitas, uma ação é executada. A ação consiste em transmitir um conjunto de parâmetros de configuração de roteamento para que o nó sensor execute.

### 4.3.4. Processo de Inferência

Quanto todo o conhecimento necessário é colocado no middleware, este pode efetuar raciocínio sobre o contexto de nós sensores específicos ou da rede como um todo, e determinar as ações requeridas em termos de reconfiguração.

Utilizando os fatos descritos na Seção 4.3.2, o serviço de middleware recebe os dados de contexto e insere os mesmos na base de fatos. A máquina de regras compara então os dados da base de fatos com as regras armazenadas na base de regras, e sinaliza quando todas as condições para uma ação específica forem satisfeitas, de forma que a ação correspondente seja realizada.

A atualização ou criação de fatos na base de fatos aciona o processo de inferência, que em teoria, deveria iterar pelas condições de todas as regras. Após as regras terem sido analisadas, as ações para as regras satisfeitas seriam executadas. No entanto, existem várias otimizações para que o processo de analisar todas as regras não seja necessário. Uma destas otimizações é o algoritmo Rete, um algoritmo de casamento de padrões, introduzido por (FORGY, 1982), que é utilizado pela maioria das implementações de sistemas de inferência que usam o *forward chaining* como modelo de raciocínio.

Apesar de prover otimizações, o algoritmo Rete é bastante conhecido pelo consumo excessivo de memória. Para abordagens de reconfiguração por regras, onde a máquina de regras é implementada no próprio nó sensor, este algoritmo é inviável. Como este trabalho tem enfoque de middleware (em uma entidade fora da RSSF) não há ressalvas quanto a este item.

#### **4.3.5. Qualidade de Contexto**

As RSSF são consideradas importantes fontes de contexto. No entanto, falhas nos nós sensores decorrentes de problemas nas plataformas de hardware e de características particulares dos ambientes em que as RSSF são instaladas, impõem uma série de desafios quanto à gestão eficiente da informação de contexto, por exemplo, a necessidade de tratamento de eventuais imperfeições nos dados coletados.

De acordo com (HENRICKSEN; INDULSKA, 2004), as aplicações sensíveis ao contexto geralmente assumem que a informação contextual é completa e acurada; entretanto, como resultado de ruídos, falhas nos sensores e outros fatores, os dados sensoreados podem estar sujeitos a erros.

Desta forma, essas imperfeições podem ocasionar comportamentos inadequados e influenciar o processo de avaliação das informações contextuais. Com o objetivo de mitigar este problema, é proposto que o serviço de reconfiguração possua um componente para avaliação de qualidade da informação contextual (QoC *Quality of Context*), e é sugerida a utilização do modelo proposto em (HOFFMAN et al., 2013), em fase de desenvolvimento no LPRM/UFES.

O citado modelo define cinco parâmetros de QoC, sendo eles:

- *Coverage*: Representa o escopo aceitável de valores (min e max) esperado para o tipo de contexto.
- *Up-to-dateness*: Denota relevância temporal.
- *Frequency*: Indica se a geração do dado ocorre na periodicidade esperada.
- *Accuracy*: Corresponde ao grau de correção entre o valor do dado adquirido e seu valor real. Os autores (HENRICKSEN; INDULSKA, 2004) ressaltam que é difícil determinar o valor real de um contexto e sua abordagem, e estimam um intervalo de confiança entre leituras consecutivas de um mesmo contexto.
- *Significance*: mede a relevância ou importância de um dado contextual, ou seja, indica se um dado pertence a um subconjunto de valores críticos definidos para o seu tipo.

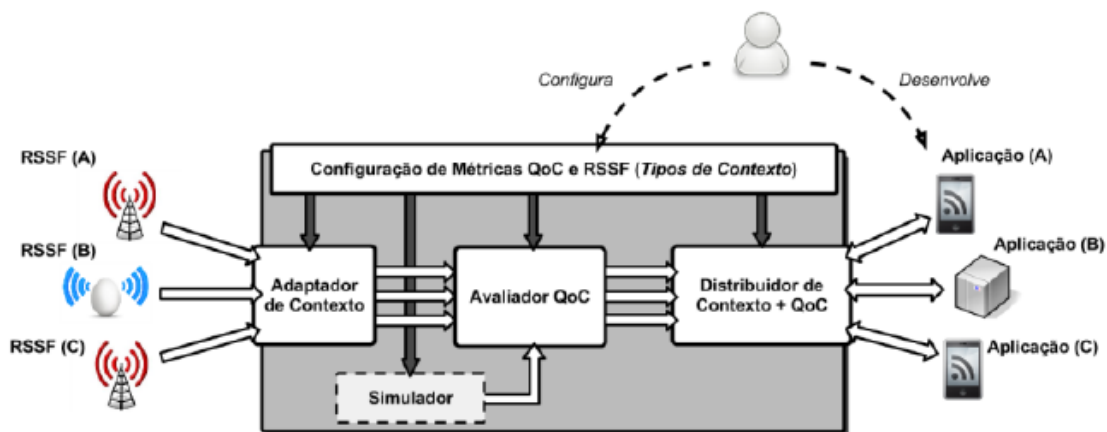


Figura 4.6 - Arquitetura Conceitual QoC

A arquitetura conceitual é constituída de quatro componentes principais: (i) Configuração de Métricas QoC e RSSF; (ii) Adaptador de Contexto; (iii) Avaliador de QoC e (iv) Distribuidor de Contexto e QoC.

O componente de configuração gerencia a parametrização da rede onde é definido o tipo de contexto e as medidas de qualidade desse tipo.

O componente Adaptador de Contexto é responsável por receber os dados das fontes de contexto cadastradas, classificá-las por tipo de contexto, e instanciá-los na



forma de uma entidade dado de contexto; e encaminhá-lo ao módulo avaliador. Uma entidade dado de contexto possui um tipo de contexto, um valor escalar e um *timestamp* de geração, que é a data e hora em que o contexto foi coletado ou gerado pelo simulador.

O componente Avaliação de QoC recebe os dados de contexto, realiza o tratamento com base nas parametrizações do respectivo tipo de contexto, gera e agrega um *timestamp* de avaliação e medidas de QoC repassando-o ao componente de distribuição de contexto.

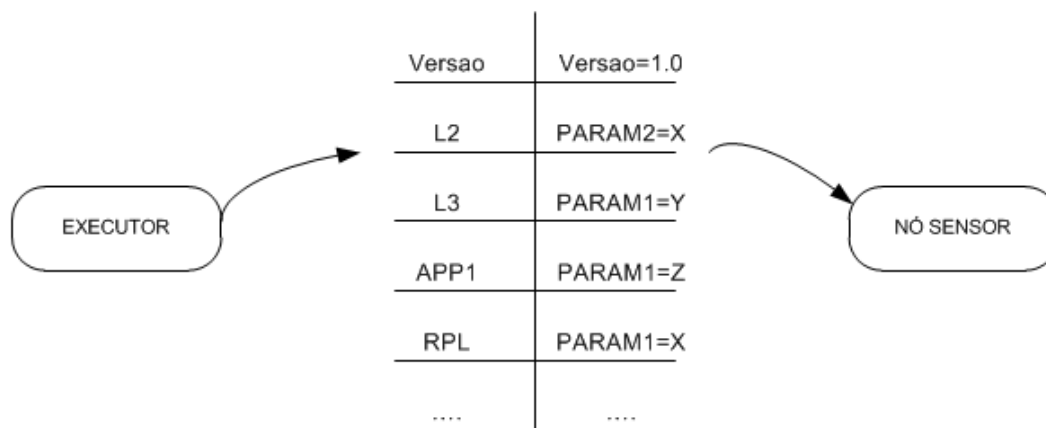
O componente Distribuição de contexto por sua vez, realiza a entrega dos dados de contexto e QoC agregado à aplicação. No caso do serviço de middleware proposto, esses dados são entregues à Camada de Adaptação.

#### **4.3.6. Atuação**

Uma vez que a máquina de inferência tenha uma regra satisfeita, o serviço de middleware deve efetuar a reconfiguração especificada. Conforme mencionado na Seção 4.3.2, o componente sujeito à reconfiguração deve compartilhar uma interface com o serviço. Sempre que uma regra for satisfeita, o serviço de reconfiguração envia um conjunto de parâmetros de configuração através desta interface. Cada aplicação no nó sensor que possua interface com o serviço pode assim, executar sua reconfiguração.

É importante mencionar que o serviço de middleware somente indica que tipo de reconfiguração deve ser feita, ou seja, quais parâmetros de configuração devem ser modificados. Na versão atual da proposta, não há operadores básicos de comunicação Middleware-Sensor, como “atualizar”, “apagar” ou “modificar configuração”, etc.

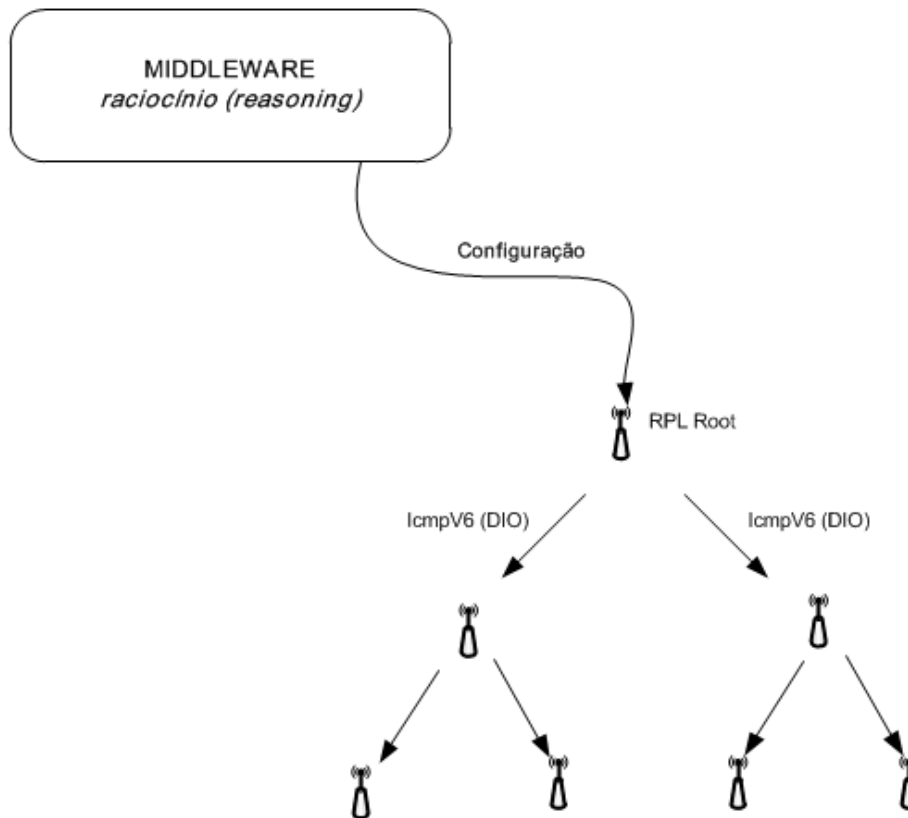
Como pode ser observado na Figura 4.7, o componente “Executor” do serviço de middleware envia a configuração do nó sensor em formato de uma tupla com dois elementos, onde o primeiro corresponde ao componente de software endereçado (aplicação, roteamento, camada de enlace – L2, etc), e o segundo aos parâmetros (chave, valor) para reconfiguração.



**Figura 4.7 - Parâmetros de Reconfiguração**

Como dito anteriormente, o serviço de middleware somente indica *que* tipo de reconfiguração deve ser feita (parâmetros), sem especificar *como* deve ser feita. Isto provê um serviço de reconfiguração genérico que pode ser utilizado em várias arquiteturas de sistemas. Por exemplo, embora o foco deste trabalho seja a reconfiguração de parâmetros de roteamento, seria possível definir uma aplicação reconfigurável, bastando que a mesma crie uma interface com o serviço de middleware para receber os parâmetros de reconfiguração mantendo-se, desta forma, uma separação entre a lógica (código) e a configuração (parâmetros).

Em relação ao roteamento, foi escolhido o protocolo RPL para implementação, validação e testes do serviço proposto. A Figura 4.8 apresenta a reconfiguração de uma rede de sensores executando o RPL. As principais informações de roteamento (construção do DODAG e métricas utilizadas) são disseminadas na rede através do protocolo ICMPv6 (DIO), que são geradas periodicamente pelo nó raiz. Assim, no caso do roteamento RPL, não se faz necessária a reconfiguração de cada nó sensor individualmente, bastando somente a reconfiguração do nó sensor raiz da rede RPL.



**Figura 4.8 - Reconfiguração do RPL**

Também vale ressaltar que, mesmo com a reconfiguração de um nó sensor, novos fatos continuam sendo gerados e enviados para base de fatos, em um processo de retroalimentação. Esse processo introduz dinamicidade à rede e deve ser examinado com cuidado durante o desenho das regras.

Este capítulo apresentou uma proposta de um serviço de middleware para reconfiguração dinâmica de Redes de Sensores em Fio com suporte de alto nível para definição de regras. O Capítulo 5 apresenta aspectos tecnológicos da arquitetura proposta e alguns testes de desempenho da rede com relação à reconfiguração de roteamento.

## 5. Implementação e Testes

Este capítulo apresenta os aspectos tecnológicos envolvidos na infraestrutura de reconfiguração para redes de sensores sem fio proposta no capítulo anterior. O capítulo se inicia descrevendo a arquitetura de implementação (Seção 5.1), ressaltando as escolhas tecnológicas adotadas para a realização do protótipo. A Seção 5.2 descreve um cenário de uso do serviço proposto e como a reconfiguração é estabelecida. Na Seção 5.3 são efetuados alguns testes de desempenho, seguindo a abordagem de reconfiguração dinâmica para roteamento RPL. Por fim, a Seção 5.4 apresenta uma discussão crítica de alguns aspectos do serviço proposto.

### 5.1. Introdução

A arquitetura de implementação é mostrada na Figura 5.1, onde se pode destacar a estrutura tecnológica utilizada para viabilizar e atender os requisitos da arquitetura conceitual definida no capítulo anterior.

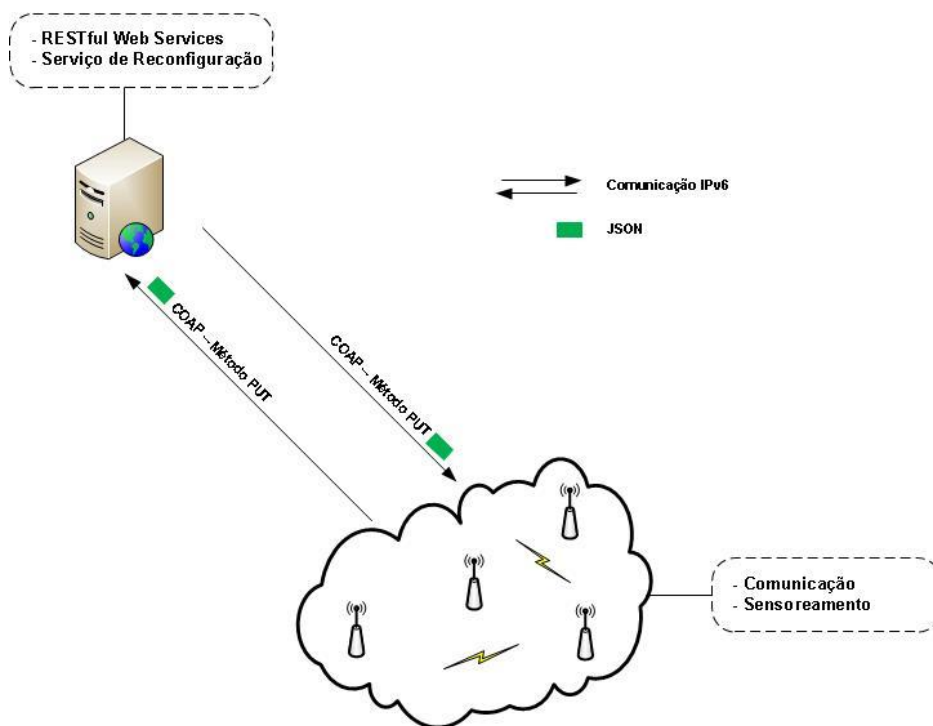


Figura 5.1 - Comunicação RSSF-Middleware

Segundo a abordagem de reconfiguração usada neste trabalho, o especialista de RSSF inicialmente constrói as aplicações que serão gravadas nos nós da rede de sensores e no nó *sink*, usando alguma linguagem de programação adequada à plataforma de sensores do respectivo sistema operacional. Em um segundo momento o especialista, através do serviço de middleware de reconfiguração, define as regras para descrever e caracterizar o comportamento da rede, bem como quais tipos de ação tomar, caso a reconfiguração seja desejada. Para a definição das regras e ações ele faz uso da linguagem declarativa disponibilizada pela infraestrutura.

Para que as redes desejadas possam começar a disponibilizar os dados para o serviço de middleware é necessário que ocorra a execução de uma aplicação específica nos nós sensores. Tal aplicação (agente) é responsável por estabelecer uma conexão com a infraestrutura de middleware e enviar um conjunto de dados, que representam informações específicas da rede que poderão ser utilizadas para caracterizá-la. O conjunto de dados enviados e o intervalo de envio são definidos pelo especialista no momento de projeto da rede. Esta etapa representa o monitoramento da rede, e é ilustrado na Figura 5.1 pela seta de comunicação partindo da RSSF em direção ao middleware. Outra responsabilidade desta aplicação é, receber dados oriundos do middleware (parâmetros de configuração) e realizar, caso necessário, a reconfiguração de determinado parâmetro com o valor indicado pelo middleware (seta de comunicação partindo do middleware em direção à RSSF).

### **Comunicação**

A comunicação da aplicação responsável por efetuar a interface entre o nó sensor e o middleware é efetuada através do protocolo IPv6. Desta forma, é possível prover comunicação fim-a-fim entre os dispositivos (sensor-middleware) sem necessidade de protocolos intermediários de tradução (*proxies*), fazendo com que seja possível, inclusive, a execução do middleware em um ambiente de nuvem computacional de forma facilitada.

O envio dos dados monitorados e a recepção dos parâmetros de configuração são realizados de forma semelhante, através do método PUT do protocolo CoAP (*Constrained Application Protocol*) (SHELBY, 2012), utilizando a arquitetura REST (*Representational State Transfer*) (FIELDING, 2000). O protocolo de transporte utilizado é o UDP. Conforme destacado em (FILHO, 2012), o CoAP - *Constrained*

*Application Protocol* “é um protocolo da camada de aplicação que foi desenvolvido com o objetivo de se ter um protocolo Web genérico, alternativo ao HTTP, adequado aos requisitos especiais do ambiente de dispositivos de baixa potência para o qual é desenvolvido, considerando, principalmente, a questão do consumo de energia”. O Working Group CoRE (*Constrained RESTful Environments*), do IETF, é o responsável pela maior parte do trabalho de padronização do CoAP. O objetivo é que o CoAP seja apropriado ao ambiente de Internet das Coisas e às comunicações M2M (*machine-to-machine*).

Quanto ao formato do dado utilizado no intercâmbio de dados entre os nós sensores e o middleware, foi adotado o formato JSON (VASSEUR, 2010), por exigir menos recursos computacionais para seu processamento, se comparado com o formato XML, estando alinhado com as limitações de hardware inerentes às RSSF.

Os nós sensores foram programados utilizando o sistema operacional Contiki (DUNKELS; GRONVALL; VOIGT, 2004). Este S.O. disponibiliza uma implementação da pilha 6LoWPAN, denominada SICSLowPan, e é um dos pioneiros no suporte ao IP e na implementação da camada 6LoWPAN, já tendo inclusive recebido o selo do programa de conformidade e interoperabilidade *IPv6 Ready* (IPv6 Ready). A implementação da pilha 6LoWPAN no Contiki é baseada originalmente nas especificações RFC 4944 (*Transmission of IPv6 Packets over IEEE802.15.4 Networks*) (MONTENEGRO, 2007), draft-hui-6lowpan-interop-00 (*Interoperability Test for 6LoWPAN*) (HUI, 2007), e draft-hui-6lowpan-hc-01 (*Compression format for IPv6 datagrams in 6lowpan Networks*) (HUI et al., 2010). A implementação do RPL disponível no Contiki é chamada de ContikiRPL, e tem por base a RFC 6550 (*RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks*) (WINTER, 2012).

### **Ambiente para Regras**

O ambiente Drools (BALI, 2009) foi adotado como plataforma de regras na implementação realizada. A escolha de Drools atende ao requisito (ii) da arquitetura conceitual, ou seja, “Uso de regras para a especificação do comportamento da rede”. Drools oferece uma linguagem específica de domínio, denominada *Drools Rule Language (DRL)*, de alto nível de abstração, que proporciona flexibilidade e agilidade para alterações das regras de negócio e dos eventos de interesse da aplicação.

### **Reconfiguração do RPL**

Muitas métricas de roteamento foram definidas e recomendadas para as RSSF. Entretanto, a especificação do RPL se refere à construção de uma árvore topológica de roteamento, mas não impõe nenhuma métrica de roteamento e deixa este item aberto aos implementadores. As funções objetivo propostas pelo IETF apresentadas em (RFC 6719) e (RFC 6552) definem algumas recomendações sobre como implementar as funções objetivo, mas sem especificar que tipo de métrica utilizar.

Em (RFC 6552) é descrito a função objetivo OF0. É um simples mecanismo em que um nó decide o seu nó pai de acordo com o menor *rank*, o qual é definido como um inteiro representando a posição individual do nó dentro do DODAG. Este parâmetro aumenta o seu valor à medida que se desce na árvore topológica. Esta OF representa um denominador comum e tem propósitos de interoperabilidade entre diferentes implementações. Além disso, não especifica nenhuma métrica de roteamento, apesar de, por ter como critério escolher o menor *rank* e representar assim o menor caminho, ser citada também como *Hop Count*.

Em (GNAWALI, 2012), o MRHOF (*Minimum Rank with Hysteresis Objective Function*) é proposto. Esta função objetivo é baseada em *containers*, que são cabeçalhos para métricas localizados no pacote ICMPv6 DIO. Assim, é possível que em um pacote DIO especificando a função objetivo MRHOF, vários tipos de métricas de roteamento sejam indicadas.

Em (KARKAZIS, 2012) os autores propõem uma combinação de duas métricas de roteamento entre as opções: (i) *hop count*, (ii) ETX, (iii) nível energia, (iii) RSSI no processo de decisão do RPL. Por exemplo, caso seja especificado duas métricas  $m1$  e  $m2$ , o nó pode computar o caminho utilizando funções de soma ( $m1 + m2$ ) ou dando prioridade à alguma delas (a que for maior ou menor).

No caso de se utilizar uma métrica única (1 único *container*), isso pode ser ineficiente e degradar o desempenho da rede, já que a métrica pode não atender a todos os requisitos da aplicação. Por exemplo, caso a métrica seja *Hop Count*, pode levar a alguns nós esgotarem a bateria muito mais rápido do que outros, já que a métrica não leva em isso em conta. Caso a métrica seja ETX, pode ocasionar latência mais alta na rede. Percebe-se, assim, que nenhuma métrica isolada atenderá a todos os requisitos, já que cada métrica possui características únicas.

No caso da combinação de duas métricas (BRACHMAN, 2013) (2 *containers*), apesar de ser uma abordagem mais flexível, também há argumentos contrários. A combinação de duas métricas é insuficiente para atender a todos os

requisitos das aplicações que podem rodar em uma rede de sensores. Além disso, a utilização de duas métricas pode melhorar o desempenho, mas ao custo de degradação de algum outro parâmetro. Por exemplo, combinando *Hop Count* e *ETX* serão selecionados caminhos curtos e confiáveis, mas pode levar ao uso em demasia de recursos em alguns nós, causando esgotamento de bateria.

Outras questões são que apenas duas métricas são insuficientes para cobrir os vários requisitos possíveis das aplicações e, também, os requisitos das aplicações podem mudar com o tempo, e a mesma não deveria ficar amarrada a um único tipo de métrica ao longo da vida útil.

Na abordagem de reconfiguração adotada neste trabalho, o middleware envia nos parâmetros de reconfiguração (Figura 5.2), todas as possíveis métricas de roteamento que o nó sensor deve utilizar (duas ou mais). A ordem das métricas nos parâmetros de configuração direciona qual delas será utilizada para o cálculo. Como reconfiguração (por exemplo, através da avaliação das características da rede através das informações contextuais), basta enviar um novo conjunto de parâmetros com a nova ordem. É importante ressaltar que o software do nó sensor deve suportar as métricas que são enviadas como parâmetro (deve possuir as métricas implementadas no software).

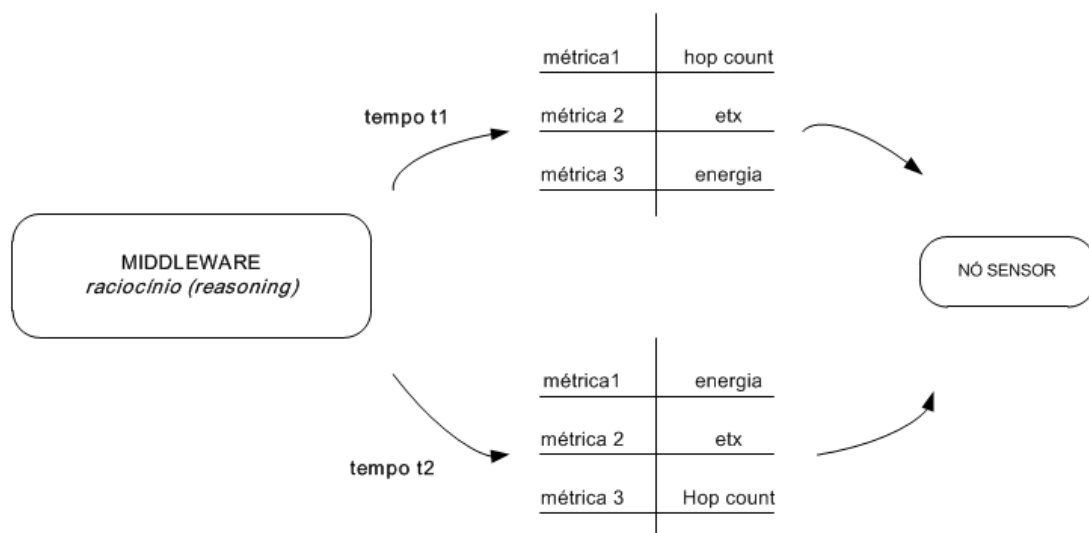


Figura 5.2 - Reconfiguração de Métrica de Roteamento



## 5.2. Cenário de Exemplo

Como forma de exemplificar o funcionamento do serviço de middleware, é apresentado um cenário de exemplo. Neste cenário, considera-se uma aplicação de monitoramento ambiental. Esta aplicação coleta informações como presença de fumaça no ambiente monitorado. É desejável que, em caso detecção de fumaça, a rede seja reconfigurada para aumentar a taxa de amostragem do evento, a fim de ter uma visibilidade maior do que está ocorrendo, e mudar o modo de funcionamento da rede para o modo “alta velocidade e baixa latência”. Ao fim do evento, a rede deve voltar ao modo normal de operação “baixa velocidade e economia de energia”. A tabela 5.1 ilustra os parâmetros que são alterados em cada caso.

<b>Alta Velocidade</b>		<b>Economia de Energia</b>	
<b>Métrica de Roteamento</b>	<i>hop count</i>	<b>Métrica de Roteamento</b>	<i>energy</i>
	<i>energy</i>		<i>hop count</i>

**Tabela 1 - Parâmetros de Reconfiguração**

O parâmetro “Métrica de Roteamento” indica qual tipo de métrica será utilizado na rede. A Figura 5.3 exemplifica as regras (em formato Drools – DRL) criadas para atender à reconfiguração. É importante notar que, conforme a Seção 4.3.6, para configuração da métrica de roteamento, basta a configuração do nó *sink*, pois o mesmo propaga as informações de métricas de roteamento para os demais nós da rede através de mensagens ICMPv6.

```

1 import com.reconfiguracao.*;
2
3 Boolean $FlagTP
4
5 rule "Verifica existencia de Fumaca"
6     when
7         exists( Fumaca() )
8     then
9         $FlagTP = true;
10    end
11
12 rule "Configura a rede para o modo baixa-velocidade e economia de energia"
13     when
14         eval( ! $FlagTP )
15     then
16         Configuracao config = new Configuracao();
17         Configuracao config_sink = new Configuracao();
18         Executor exec = new Executor();
19
20         config_sink.setMetric("energy, hop count");
21
22         System.out.println("Configurando a rede para o modo baixa-velocidade");
23         exec.send("sink", config_sink );
24    end
25
26
27
28 rule "Configura a rede para o modo alta velocidade e baixa latencia"
29     when
30         eval( $FlagTP )
31     then
32         Configuracao config = new Configuracao();
33         Configuracao config_sink = new Configuracao();
34         Executor exec = new Executor();
35
36         config_sink.setMetric("hop count", "energy");
37
38         System.out.println("Configurando a rede para o modo alta-velocidade");
39         exec.send("sink", config_sink);
40    end

```

**Figura 5.3- Regras Drools**

É possível também a especificação de regras utilizando informações contextuais dos próprios nós sensores e a adequação das métricas de roteamento de acordo com esses valores. Na Figura 2.4 são exemplificados vários tipos de métricas de roteamento, cada qual com suas particularidades. Assim, é possível criar regras do tipo: “Se 70% dos nós da rede estiverem com bateria abaixo de  $x$  mA, então configure métrica RE”, ou “Se nível de número de retransmissões na camada de enlace estiver baixo, então configure métrica ETX”.

### 5.3. Testes

Para avaliar o desempenho da reconfiguração dinâmica das métricas de roteamento do RPL, foi utilizado o ambiente de simulação Contiki em todos os experimentos. O ambiente é definido em duas partes: (i) simulador de rede COOJA (OSTERLIND, 2006), o qual permite simulação em larga escala com diferentes modelos de rádio; e (ii) emulador de motes (nós da rede) MSPSim, o qual combina a emulação dos ciclos de execução de motes baseados no microcontrolador MSP430 (ex., TmoteSky e TelosB) com a emulação do transceiver de rádio CC2420.

A realização de experimentos simulados não reproduz 100% de uma situação real de execução; por outro lado, permite a repetição fiel dos cenários e o controle total sobre os parâmetros usados nos experimentos. Assim, são explorados esses aspectos do ambiente simulado com o objetivo de isolar certas interferências (embora reais) para focar as avaliações em aspectos específicos de operação da rede.

A topologia empregada nas simulações é a mesma em todos os experimentos, e é mostrada na Figura 5.4. Foram utilizados um nó sink e 40 nós que assumem papéis de roteador e/ou emissor de informações, de acordo com os cenários de avaliação descritos abaixo. A área simulada é de 240x240 metros, com o alcance de rádio para comunicação de 50 m, e mais 50 m de interferência, gerando as duas áreas em volta do nó *i* mostradas na Figura 5.4. Todas as mensagens trocadas entre os nós são implementadas na camada de aplicação e usam o protocolo de transporte UDP.

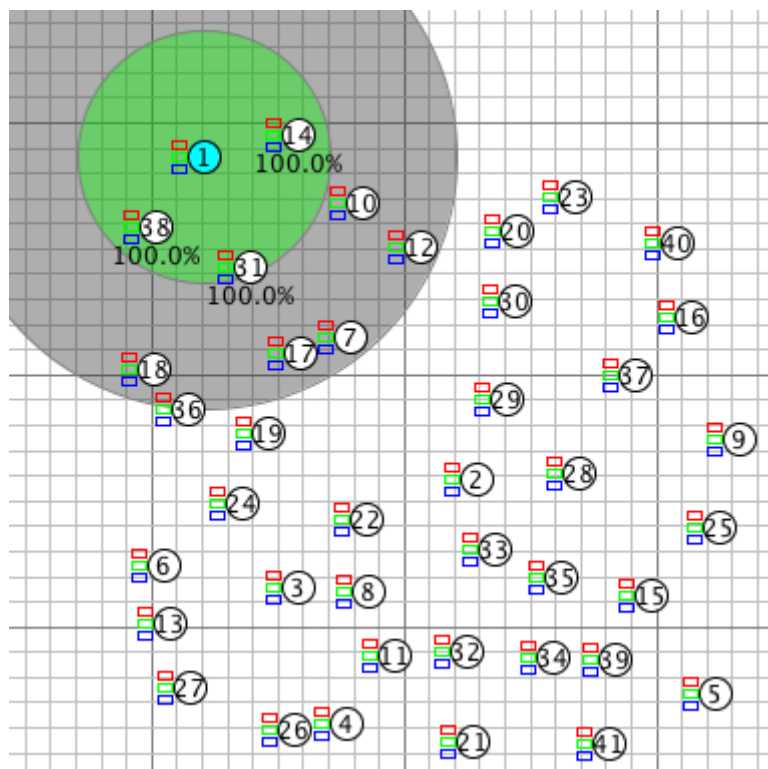


Figura 5.4 - Topologia da Simulação

Três funções objetivo foram avaliadas. A primeira, OF0 (métrica *Hop Count*) leva em consideração o número de saltos na topologia de rede, para o cálculo da rota *default*. A segunda é a função MRHOF (métrica ETX – *Expected Transmission Count*), que efetua a decisão de roteamento de acordo com a qualidade dos enlaces de comunicação. A terceira é a função Híbrida, que suporta reconfiguração e representa a possibilidade de chaveamento entre *Hop Count* e ETX em tempo de execução.

Devido a limitações na implementação do ContikiRPL, o mesmo ainda não suporta uma variedade de métricas de roteamento, e as únicas disponíveis são *Hop Count* e ETX. Este cenário impõe empecilhos para uma simulação utilizando as características de várias métricas citadas na literatura. Além disso, o ambiente de simulação COOJA possui poucas informações que podem ser obtidas do ambiente (informações contextuais), o que dificulta o teste do ambiente de reconfiguração proposta em um cenário completo. Desta forma, como forma de avaliar a possibilidade de reconfiguração do nó em tempo de execução e o resultado desta ação, o chaveamento foi feito de forma manual durante a simulação (em cada simulação, o nó chaveia de HopCount para ETX após decorrido metade do tempo da simulação).

### 5.3.1. Perda de Pacotes

O objetivo deste experimento é verificar a taxa de perda de pacotes com a variação do número de mensagens trafegando na rede. Para isso, diversificou-se o número de nós emissores e o intervalo de tempo entre as transmissões das mensagens em cada nó. Todas as mensagens são destinadas ao nó *sink* que apenas recebe e contabiliza as mensagens. Neste experimento, todas as mensagens possuem um payload de 12 bytes, para simular, por exemplo, o envio de três medidas sensoreadas de 4 bytes cada (como temperatura, pressão e umidade).

Quatro cenários foram experimentados com relação ao percentual de nós da rede que enviam pacotes: 25% dos nós são emissores/roteadores, 75% são apenas roteadores; 50% dos nós são emissores/roteadores, 50% são apenas roteadores; 75% dos nós são emissores/roteadores, 25% são apenas roteadores; e 100% dos nós são emissores/roteadores. Além disso, dois cenários foram montados com relação ao intervalo de tempo das mensagens enviadas (4 ou 10 segundos) de forma a emular as características de diferentes aplicações.

A escolha dos nós emissores e roteadores para cada um dos 8 casos de teste (4 cenários com 2 frequências de emissão) foi feita de forma aleatória. Foram realizadas 5 simulações de 1 hora para cada caso. Os valores obtidos são mostrados na Figura 5.5.

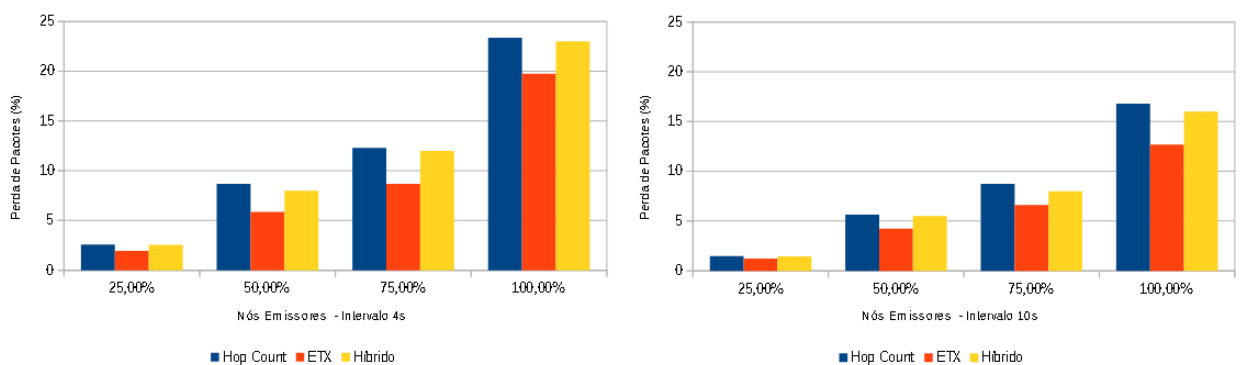


Figura 5.5 - Gráfico de Perda de Pacotes

Observa-se que a perda de pacotes é baixa quando a densidade (número de nós emissores) é baixa. A medida que o número de emissores aumenta, há uma perda maior de pacotes. Isso porque há menos nós com a tarefa dedicada de roteamento, e estes

também estão tentando enviar dados. Nota-se também que a taxa de perda aumenta com o diminuição do intervalo de envio (aumento da frequência de envio).

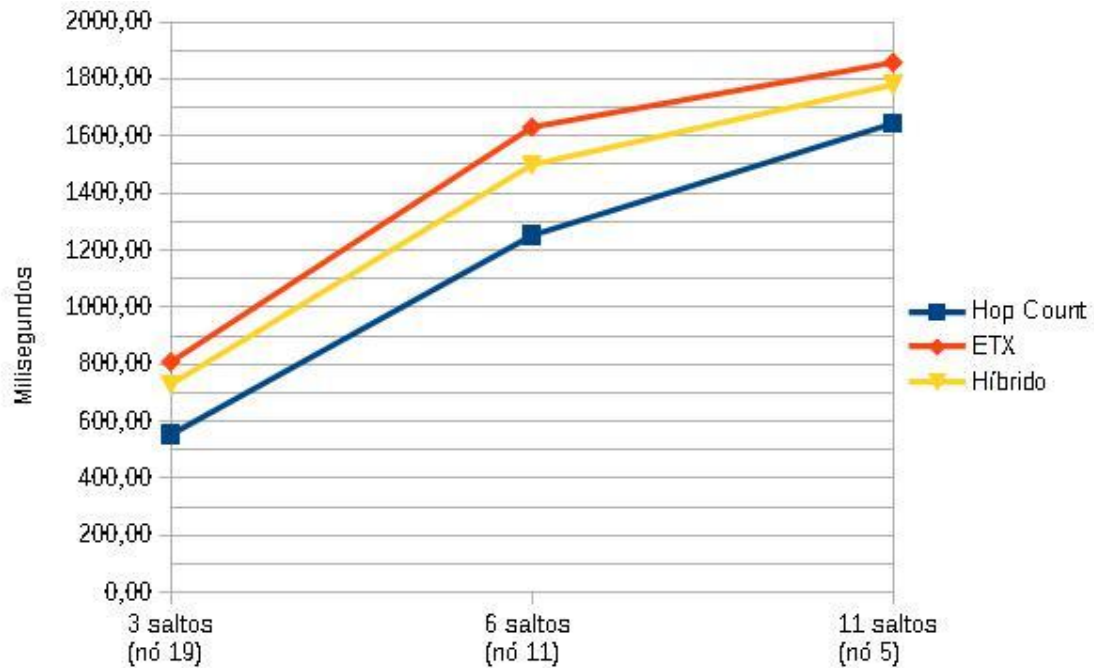
Outra observação é que em todos os cenários é possível obter uma taxa de perda menor na abordagem da função Híbrido do que com a função *Hop Count* (exceção 25%), e uma taxa de perda ligeiramente superior à da função ETX (exceção 100%, em que a diferença entre Híbrido e ETX é mais acentuada).

A razão para a alta porcentagem da função *Hop Count* é que esta não prioriza os links com alta qualidade na seleção da rota *default*, e esta rota pode estar congestionada e assim perder pacotes. Já a função ETX, que tem as menores taxas de perda de pacotes, prioriza os links com as menores taxas de perda.

### 5.3.2. Round-Trip Time

Neste segundo experimento mediu-se o tempo de comunicação entre pares de nós da rede (o segundo nó é sempre o nó *sink*). Escolheu-se três nós emissores para realizar a medição de acordo com as suas distâncias, em saltos, até o nó *sink*. O objetivo nesse caso, foi analisar o impacto mínimo do roteamento no tempo de comunicação de acordo com o número de saltos.

Os nós escolhidos para o experimento foram o 19, 11 e 5 (vide Figura 5.4). O primeiro se encontra a 3 saltos do *sink*, o segundo a 6 saltos e o último 11 saltos. Foram realizadas 5 simulações para cada nó (1 hora para cada simulação). O nó envia um pacote de 1 byte ao *sink*, que responde com o mesmo pacote. Após receber a resposta, o nó repete o processo. Isso se dá durante toda a simulação. Mediu-se então o tempo gasto desde o envio até o recebimento para cada mensagem.



**Figura 5.6 - Gráfico de RTT (Round Trip Time)**

De acordo com o gráfico, as três funções objetivo mantêm um atraso de ida e volta abaixo de 2 segundos.

É perceptível que, à medida que o nó emissor se afasta do nó receptor, o tempo de ida e volta do pacote aumenta, pois soma-se os atrasos de armazenagem e reenvio de cada nó pelo caminho. Além disso, a função *Hop Count* possui os menores valores, já que representa o menor caminho do nó emissor ao nó receptor. No entanto, o menor caminho não significa o caminho que possui a menor latência, já que alguns nós poderiam estar congestionados no menor caminho (o que não foi o caso neste cenário).

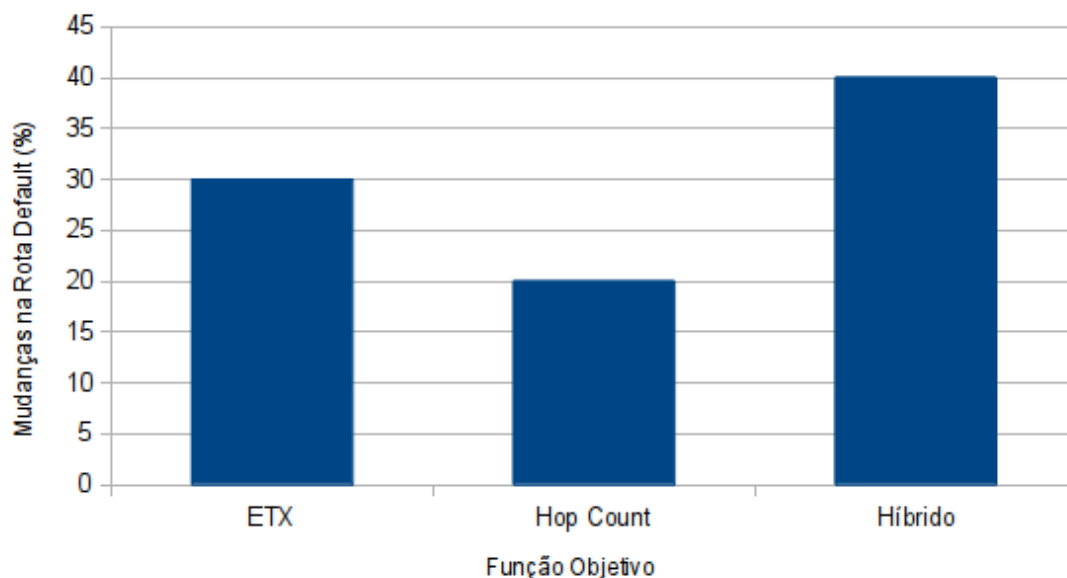
A função Híbrido possui um tempo atraso muito próximo aos valores da ETX. Isso se deve em parte ao atraso no chaveamento da função *Hop Count* para a função ETX (convergência de todos os nós da rede).

### 5.3.3. Estabilidade da Rede

Neste terceiro experimento mediu-se a quantidade de vezes que qualquer nó da rede (exceção do nó *sink*) altera o seu nó pai (*rota default*) na árvore RPL. O objetivo nesse caso foi analisar a estabilidade da rede de acordo com cada função objetivo.

Como aplicação exemplo foi utilizada o envio de pacotes UDP ao nó sink, em intervalos de 10 segundos.

Todos os nós foram monitorados. Foram realizadas 5 simulações (1 hora para cada simulação), e ao final foi calculado a média de todos os nós.



**Figura 5.7 - Gráfico de Mudanças na rota Default**

A Figura 5.7 apresenta o número de mudanças na rota *default* das três funções objetivo. Enquanto as funções objetivo ETX e *Hop Count* permanecem abaixo de 30%, a função Híbrido possui um valor mais alto. Este número de certa forma é esperado, já que com a mudança da função objetivo dinamicamente, as rotas *default* devem ser recalculadas. No entanto, o maior número de mudanças na rota *default* para a função Híbrido representa o custo de uma maior flexibilidade na escolha das métricas, de acordo com os experimentos 1 e 2.

#### 5.4. Discussão

Conforme descrito nas seções anteriores, as informações contextuais oriundas da rede de sensores são enviadas ao serviço de middleware com o intuito de alimentarem uma base de dados de conhecimento acerca da rede. Com esse conhecimento, é feito o casamento de regras a fim de executar ações de reconfiguração na rede. É válido neste ponto, realizar uma análise de alguns pontos da arquitetura proposta.



Existem na literatura dois tipos de enfoque quanto ao tema reconfiguração. Um é focado em soluções de middleware (que executam fora do nó sensor). Evidentemente é necessário que haja comunicação entre o middleware e os nós sensores para que o middleware efetue a reconfiguração desejada. Outro enfoque é em soluções que determinam a implementação de agentes, que são aplicações implementadas nos nós sensores responsáveis por realizar a reconfiguração (i) recebendo parâmetros de uma entidade externa (como um middleware) ou (ii) realizando a reconfiguração por meio de regras implementadas no próprio software (*hard-coded*). Assim, este trabalho priorizou a especificação do serviço de reconfiguração, deixando para trabalhos futuros a especificação de um agente para a execução da reconfiguração.

Em relação à escalabilidade, há um grande debate acerca de soluções distribuídas (cada nó sensor responsável por sua própria reconfiguração) e centralizadas (middleware). Muitos autores advogam que o uso de middleware traz consigo problemas de escalabilidade. No caso da reconfiguração de métricas de roteamento para o protocolo RPL, descrita na seção 5.1, este problema é minimizado, já que o mecanismo de reconfiguração já está embutido na própria lógica do protocolo (através das suas mensagens ICMPv6). Além disso, a mensagem de reconfiguração é enviada para apenas um nó (*sink*). No entanto, para outros parâmetros de configuração (como em um cenário hipotético em que se deseja reconfigurar parâmetros da camada de enlace por exemplo), este problema fica mais evidente. Caso haja 500 nós na rede, é necessário enviar os parâmetros para todos os 500 nós, o que acarreta um enorme gasto energético.

Uma solução seria o estudo de mecanismos de roteamento *multicast* para propagação dessas informações de configuração ou mesmo a utilização de alguma variante do protocolo *Trickle*, de forma semelhante ao implementado no RPL para divulgação de métricas. É necessário assim, um estudo aprofundado a esse respeito.

## 6. Conclusões

De uma maneira geral, pode-se afirmar que a qualidade de serviço das RSSF depende muito do contexto em que elas estão inseridas. Assim, como forma de otimizar a RSSF e enriquecer os serviços prestados, estamos observando um crescimento de serviços de rede que incorporam a sensibilidade ao contexto ao seu desenho, adaptando o seu comportamento a partir do uso controlado de informações contextuais da própria rede e/ou outras fontes de contexto de interesse, inclusive externas. Porém, atualmente, na maioria dos casos, a lógica de adaptação está embutida no código, e não há maneiras simples alterá-las, sem que seja necessário a reprogramação (código) dos nós sensores. Também são utilizadas informações contextuais limitadas ao entendimento do nó sensor referente ao ambiente em que ele se encontra, já que os nós sensores não compartilham essas informações.

Tomando por base este cenário, este trabalho apresentou um serviço de reconfiguração dinâmica para Redes de Sensores sem Fio, que faz parte de uma arquitetura genérica de middleware que vem sendo vislumbrada no escopo de trabalhos do grupo de redes de sensores do LPRM/UFES. Utilizando um serviço de reconfiguração dinâmica como o proposto neste trabalho, são criadas regras que adaptam o comportamento da rede em tempo de execução, em função das informações contextuais. Assim, o serviço proposto implementa a configuração mais adequada na rede de acordo com o contexto, provendo uma abordagem estruturada de reconfiguração, em oposição à soluções específicas e implementadas diretamente no código das aplicações.

Além disso, foi especificado um mecanismo de reconfiguração de métricas de roteamento para o protocolo RPL, e alguns testes de desempenho foram efetuados. Embora não sejam conclusivos, os testes indicam que há potencial de desenvolvimento, principalmente se estas reconfigurações forem efetuadas conjuntamente com outros parâmetros da rede, como proposto nesta dissertação.

Apesar da sua importância, algumas questões não foram deliberadamente tratadas nesta primeira versão da proposta. Por exemplo, embora segurança e privacidade sejam tópicos de extrema importância, mecanismos desta classe de funções de middleware não foram especificados nesta versão do serviço de reconfiguração, devendo este ser objeto de estudos futuros. Outras questões não menos importantes, tais

como a especificação de um agente responsável por efetuar a interface com o serviço de middleware para receber os parâmetros de reconfiguração e aplicá-los no nó sensor também não foram aqui tratadas.

Essas e outras questões constituem temas naturais de trabalhos futuros, alguns deles de concepção mais simples e imediata; outros de complexidade mais elaborada, exigindo, portanto, estudos mais aprofundados. Como exemplos, destacamos:

- Especificação de um agente: a especificação de um agente de software para o nó sensor permitirá que o mesmo realize a interface com o middleware, receba os parâmetros de configuração e aplique-os efetivamente.
- Implementação do módulo de segurança: devido a sua relevância, faz-se necessário a especificação de mecanismos para garantir aspectos de segurança como criptografia e autenticação.
- Realização de testes de desempenho: neste trabalho foi descrito um caso de uso para o serviço de middleware proposto. Logo, é importante a realização de testes de desempenho reais e simulados para aferição dos potenciais ganhos de desempenho que esta arquitetura pode proporcionar.
- Escalabilidade: na abordagem proposta, é necessário que o serviço de middleware envie os parâmetros de reconfiguração para cada nó sensor (no caso de uma reconfiguração que não seja de roteamento RPL). É necessário investigar formas eficientes de distribuição dos parâmetros de reconfiguração para rede.

## 7. Referências

ABOWD, Gregory D. et al. Towards a better understanding of context and context-awareness. In: **Handheld and ubiquitous computing**. Springer Berlin Heidelberg, 1999. p. 304-307.

ADVANTECH. [http://www.advantech.eu/it/edm/2011\\_WSN/index\\_flash.htm](http://www.advantech.eu/it/edm/2011_WSN/index_flash.htm), 2014.

AKKAYA, Kemal; YOUNIS, Mohamed. A survey on routing protocols for wireless sensor networks. **Ad hoc networks**, v. 3, n. 3, p. 325-349, 2005.

AKYILDIZ, Ian F.; VURAN, Mehmet Can. **Wireless sensor networks**. John Wiley & Sons, 2010.

AL-KARAKI, Jamal N.; KAMAL, Ahmed E. Routing techniques in wireless sensor networks: a survey. **Wireless communications, IEEE**, v. 11, n. 6, p. 6-28, 2004.

ALI, A. et al. Real time communication with power adaptation (RTPA) in wireless sensor network (WSN). In: **Computing & Informatics, 2006. ICOCI'06. International Conference on**. IEEE, 2006. p. 1-7.

ATZORI, Luigi; IERA, Antonio; MORABITO, Giacomo. The internet of things: A survey. **Computer networks**, v. 54, n. 15, p. 2787-2805, 2010.

BALI, Michal. **Drools JBoss Rules 5.0 Developer's Guide**. Packt Publishing Ltd, 2009.

BARRENETXEA, Guillermo et al. The hitchhiker's guide to successful wireless sensor network deployments. In: **Proceedings of the 6th ACM conference on Embedded network sensor systems**. ACM, 2008. p. 43-56.

BRACHMAN, Agnieszka. RPL Objective Function Impact on LLNs Topology and Performance. In: **Internet of Things, Smart Spaces, and Next Generation Networking**. Springer Berlin Heidelberg, 2013. p. 340-351.

BUETTNER, Michael et al. X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks. In: **Proceedings of the 4th international conference on Embedded networked sensor systems**. ACM, 2006. p. 307-320.

CHEN, Dazhi; VARSHNEY, Pramod K. QoS Support in Wireless Sensor Networks: A Survey. In: **International Conference on Wireless Networks**. 2004. p. 1-7.

CHEN, Guanling et al. **A survey of context-aware mobile computing research**. Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College, 2000.

CHONG, Chee-Yee; KUMAR, Srikanta P. Sensor networks: evolution, opportunities, and challenges. **Proceedings of the IEEE**, v. 91, n. 8, p. 1247-1256, 2003.

DARGIE, Walteneagus; POELLABAUER, Christian. **Fundamentals of wireless sensor networks: theory and practice**. John Wiley & Sons, 2010.

DE COUTO, Douglas SJ et al. A high-throughput path metric for multi-hop wireless routing. **Wireless Networks**, v. 11, n. 4, p. 419-434, 2005.

DE JONG, Adriaan; WOEHRLE, Matthias; LANGENDOEN, Koen. MoMi: model-based diagnosis middleware for sensor networks. In: **Proceedings of the 4th International Workshop on Middleware Tools, Services and Run-Time Support for Sensor Networks**. ACM, 2009. p. 19-24.

DONG, Wei et al. DPLC: Dynamic packet length control in wireless sensor networks. In: **INFOCOM, 2010 Proceedings IEEE**. IEEE, 2010. p. 1-9.

DRAVES, Richard; PADHYE, Jitendra; ZILL, Brian. Routing in multi-radio, multi-hop wireless mesh networks. In: **Proceedings of the 10th annual international conference on Mobile computing and networking**. ACM, 2004. p. 114-128.

DUNKELS, Adam; GRONVALL, Bjorn; VOIGT, Thiemo. Contiki-a lightweight and flexible operating system for tiny networked sensors. In: **Local Computer Networks, 2004. 29th Annual IEEE International Conference on**. IEEE, 2004. p. 455-462.

FIELDING, Roy. Representational state transfer. **Architectural Styles and the Design of Network-based Software Architecture**, p. 76-85, 2000.

FILHO, Fernando Antonio Marques et al. Uma Proposta de Framework para Integração de Objetos Inteligentes a Redes Sociais. In: X Simposio Brasileiro de Sistemas Colaborativos. SBSC. 2013.

FINKENZELLER, Klaus. **RFID handbook: radio-frequency identification fundamentals and applications**. New York: Wiley, 2008.

FORGY, Charles L. Rete: A fast algorithm for the many pattern/many object pattern match problem. **Artificial intelligence**, v. 19, n. 1, p. 17-37, 1982.

GADDOUR, Olfa et al. OF-FL: QoS-aware fuzzy logic objective function for the RPL routing protocol. In: **Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt), 2014 12th International Symposium on**. IEEE, 2014. p. 365-372.

GADDOUR, Olfa; KOUBÂA, Anis. RPL in a nutshell: A survey. **Computer Networks**, v. 56, n. 14, p. 3163-3178, 2012.

GNAWALI, Omprakash. The Minimum Rank with Hysteresis Objective Function. 2012.

GRAZIOSI, F.; POMANTE, L.; PACIFICO, D. A middleware-based approach for heterogeneous wireless sensor networks. In: **WSEAS International Conference. Proceedings. Mathematics and Computers in Science and Engineering**. World Scientific and Engineering Academy and Society, 2008.

HAN, Chih-Chieh et al. Sensor network software update management: a survey. **International Journal of Network Management**, v. 15, n. 4, p. 283-294, 2005.

HENRICKSEN, Karen; INDULSKA, Jadwiga; RAKOTONIRAINY, Andry. Modeling context information in pervasive computing systems. In: **Pervasive Computing**. Springer Berlin Heidelberg, 2002. p. 167-180.

HENRICKSEN, Karen; INDULSKA, Jadwiga. Modelling and using imperfect context information. In: **Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on**. IEEE, 2004. p. 33-37.

HOFFMAN, Joéver Silva et al. Um Serviço de Gerenciamento de Qualidade de Contexto em Redes de Sensores Sem Fio. In: **Proceedings of the IADIS Conferencia Ibero Americana Computação Aplicada**. CIACA, 2013

HU, Peizhao et al. Context-aware routing in wireless mesh networks. In: **Proceedings of the 2nd ACM international conference on Context-awareness for self-managing systems**. ACM, 2008. p. 16-23.

HUI, Jonathan et al. Compression format for IPv6 datagrams in 6LoWPAN networks. **draft-ietf-6lowpan-hc-13 (work in progress)**, 2010.

HUI, Jonathan. Interoperability Test for 6LoWPAN. 2007.

IEEE 802.15.4 (IEEE Computer Society 2003).

IPv6 Ready Logo Program, <https://www.ipv6ready.org/>, accessed in 19/08/2014.

JAYARAMAN, Prem Prakash; DELIR HAGHIGHI, Pari. SA-A-WSN: Situation-aware adaptation approach for energy conservation in wireless sensor network. In: **Intelligent Sensors, Sensor Networks and Information Processing, 2013 IEEE Eighth International Conference on**. IEEE, 2013. p. 7-12.

KARKAZIS, Panagiotis et al. Design of primary and composite routing metrics for RPL-compliant Wireless Sensor Networks. In: **Telecommunications and Multimedia (TEMU), 2012 International Conference on**. IEEE, 2012. p. 13-18.

KARL, Holger; WILLIG, Andreas. **Protocols and architectures for wireless sensor networks**. John Wiley & Sons, 2007.

KIM, Seung-Ku et al. Tiny module-linking for energy-efficient reprogramming in wireless sensor networks. **Consumer Electronics, IEEE Transactions on**, v. 55, n. 4, p. 1914-1920, 2009.

KO, JeongGil et al. Industry: beyond interoperability: pushing the performance of sensor network IP stacks. In: **Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems**. ACM, 2011. p. 1-11.

KORTE, Kevin Dominik; SEHGAL, Anuj; SCHÖNWÄLDER, Jürgen. A study of the RPL repair process using ContikiRPL. In: **Dependable Networks and Services**. Springer Berlin Heidelberg, 2012. p. 50-61.

KULKARNI, Koustubh et al. Dynamic Reconfiguration of Wireless Sensor Networks. **IJCSA**, v. 6, n. 4, p. 16-42, 2009.

LELIGOU, Helen C. et al. Reconfiguration in Wireless Sensor Networks. In: **Developments in E-systems Engineering (DESE), 2010**. IEEE, 2010. p. 59-63.

LEVIS, Philip; CULLER, David. Maté: A tiny virtual machine for sensor networks. In: **ACM Sigplan Notices**. ACM, 2002. p. 85-95.

LIU, Juan; ZHAO, Feng; PETROVIC, Dragan. Information-directed routing in ad hoc sensor networks. In: **Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications**. ACM, 2003. p. 88-97.

LOSHIN, Peter. **IPv6: Theory, protocol, and practice**. Morgan Kaufmann, 2004.

MASCOLO, Cecilia; MUSOLESI, Mirco. SCAR: context-aware adaptive routing in delay tolerant mobile sensor networks. In: **Proceedings of the 2006 international conference on Wireless communications and mobile computing**. ACM, 2006. p. 533-538.

MCCARTHY, John; BUVAC, Sasa. Formalizing context (expanded notes). 1997.

MONTENEGRO, Gabriel et al. Transmission of IPv6 packets over IEEE 802.15. 4 networks. **Internet proposed standard RFC**, v. 4944, 2007.

MURALIDHAR, P.; RAO, Chunduri B. Rama. Reconfigurable wireless sensor network node based on NIOS core. In: **Wireless Communication and Sensor Networks, 2008. WCSN 2008. Fourth International Conference on**. IEEE, 2008. p. 67-72.

MUSOLESI, Mirco; HAILES, Stephen; MASCOLO, Cecilia. Adaptive routing for intermittently connected mobile ad hoc networks. In: **World of wireless mobile and multimedia networks, 2005. WoWMoM 2005. Sixth IEEE International Symposium on a**. IEEE, 2005. p. 183-189.

NARTEN, Thomas et al. Neighbor discovery for IP version 6 (IPv6). 2007.

OSTERLIND, Fredrik et al. Cross-level sensor network simulation with cooja. In: **Local Computer Networks, Proceedings 2006 31st IEEE Conference on**. IEEE, 2006. p. 641-648.

PORTILLA, Jorge et al. Adaptable security in wireless sensor networks by using reconfigurable ECC hardware coprocessors. **International Journal of Distributed Sensor Networks**, v. 2010, 2010.

**RFC 4944:** Transmission of IPv6 packets over IEEE 802.15. 4 networks

**RFC 6206:** The Trickle Algorithm

**RFC 6550:** RPL IPv6 Routing Protocol for Low-Power and Lossy Networks

**RFC 6552:** Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL).

**RFC 6719:** The Minimum Rank with Hysteresis Objective Function

SATYANARAYANAN, Mahadev. Pervasive computing: Vision and challenges. **Personal Communications, IEEE**, v. 8, n. 4, p. 10-17, 2001.

SCHILIT, Bill N. A context-aware system architecture for mobile distributed computing. **Unpublished PhD, Columbia University**, 1995.

SHAH, Rahul C.; RABAEY, Jan M. Energy aware routing for low energy ad hoc sensor networks. In: **Wireless Communications and Networking Conference, 2002. WCNC2002. 2002 IEEE**. IEEE, 2002. p. 350-355.

SHARKAWY, Bassam; KHATTAB, Ahmed; ELSAYED, Khaled MF. Fault-tolerant RPL through context awareness. In: **Internet of Things (WF-IoT), 2014 IEEE World Forum on**. IEEE, 2014. p. 437-441.

SHELBY, Zach; BORMANN, Carsten. **6LoWPAN: The wireless embedded Internet**. John Wiley & Sons, 2011.

SHELBY, Z. et al. Constrained Application Protocol (CoAP), draft-ietf-core-coap-13. **Orlando: The Internet Engineering Task Force–IETF, Dec, 2012.**

SUBRAMANIAN, Anand Prabhu; BUDDHIKOT, Milind M.; MILLER, Scott. Interference aware routing in multi-radio wireless mesh networks. In: **Wireless Mesh Networks, 2006. WiMesh 2006. 2nd IEEE Workshop on.** IEEE, 2006. p. 55-63.

SUN, Jun-Zhao. Os-based reprogramming techniques in wireless sensor networks: A survey. In: **Ubi-media Computing (U-Media), 2010 3rd IEEE International Conference on.** IEEE, 2010. p. 17-23.

TERFLOTH, Kirsten; WITTENBURG, Georg; SCHILLER, Jochen H. FACTS-A rule-based middleware architecture for wireless sensor networks. In: **Comsware.** 2006.

VASSEUR, Jean-Philippe; DUNKELS, Adam. **Interconnecting smart objects with ip: The next internet.** Morgan Kaufmann, 2010.

WALTERS, John Paul et al. Wireless sensor network security: A survey. **Security in distributed, grid, mobile, and pervasive computing**, v. 1, p. 367, 2007.

WINTER, Tim. RPL: IPv6 routing protocol for low-power and lossy networks. 2012.

YANG, Yaling; WANG, Jun; KRAVETS, Robin. Designing routing metrics for mesh networks. In: **IEEE Workshop on Wireless Mesh Networks (WiMesh).** 2005.

ZAHARIADIS, Theodore et al. The implications of Service Virtualisation on the routing procedure in Wireless Sensor Networks. 2012.