NUNO MANUEL FERREIRA GONÇALVES

DYSTOPOL - POLÍTICAS PARA ARMAZENAMENTO DINÂMICO DE DADOS EM REDES DE SENSORES SEM FIOS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Informática. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.

Orientadora: Profª. Drª. Carmem Satie Hara
Coorientador: Prof. Dr. Aldri Luiz dos Santos

Curitiba

2013

Agradecimentos

Agradeço primeiramente a Deus pela ajuda fundamental para superar todas as barreiras que se levantaram até chegar a este trabalho final. Agradeço profundamente à minha orientadora profª. Drª. Carmem pela infinita paciência demonstrada, pelo incentivo, pela amizade e também pelos puxões de orelhas (sempre merecidos). Foi muito mais do que uma orientadora, guardarei para sempre o seu exemplo e profissionalismo. Agradeço também ao meu coorientador prof. Dr. Aldri que sempre foi um exemplo de dedicação ao trabalho. Agradeço à minha noiva Priscila por todo o apoio durante este mestrado, pelo seu amor e por estar sempre do meu lado. Aos meus pais, José e Maria, que, mesmo estando em Portugal, sempre apoiaram os meus estudos. À minha irmã Dominique, meu cunhado Leonel e minhas sobrinhas Lara e Eva por fazerem parte da minha vida. Aos meus futuros sogros, João e Isamara, que me ajudaram muito nesta aventura por um país diferente e graças a eles tudo foi mais fácil. Agradeço também a todos os colegas da pós que me acompanharam, incentivaram e que serviram de exemplo. Agradeço a todos mencionados do fundo do coração e aos que esqueci de mencionar peço que me perdoem, a todos obrigado por fazerem parte da minha vida. Que Deus vos abençoe grandemente.

"Deus quer, o homem sonha, a obra nasce."

Fernando Pessoa

Resumo

As Redes de Sensores Sem Fio (RSSF) são redes compostas por dezenas, centenas ou mesmo milhares de sensores e por uma ou mais estações base. Os sensores são dispersos por uma área a ser monitorada e comunicam-se via rádio. Na maioria dos cenários não é possível a ligação desses sensores a uma fonte de energia ou a substituição das baterias. Assim a limitação de energia dos sensores influi decisivamente na longevidade da rede e é a característica mais marcante das Redes de Sensores Sem Fio (RSSFs). Além disso por serem dispositivos compactos, os sensores apresentam limitações em termos de capacidade de processamento e memória disponível. Como consequência dessas limitações, em uma RSSF é fundamental a gestão otimizada dos recursos disponíveis. O foco desta dissertação é na gestão do armazenamento dos dados sensoriados. Os dados coletados pelos sensores podem ser armazenados nos próprios sensores produtores, na estação base ou em sensores da rede, chamados de repositórios. A escolha da localização dos dados tem impacto no desempenho do sistema para o processamento de consultas sobre os dados sensoriados. Por exemplo, o armazenamento dos dados em uma estação base requer a transmissão dos valores de cada sensor para a estação. Dessa forma, o custo de atualização do dado é alto, mas o processamento de consultas na estação é baixo. Por outro lado, o armazenamento dos dados nos próprios sensores tem um custo de atualização baixo, mas o processamento de consultas alto, já que é necessário contactar todos os sensores para a obtenção dos valores coletados. Existem diversas propostas na literatura que adotam diferentes modelos de armazenamento. Porém, a maioria adota uma solução estática, na qual o modelo de armazenamento não se adapta aos diferentes contextos da aplicação. Neste trabalho é proposto o DYSTOPOL, um modelo dinâmico e adaptável que permite aplicar a melhor

solução disponível para cada momento do sistema utilizando um índice distribuído e políticas para implementar tais características. As políticas permitem um refinamento do sistema segundo configurações pré-definidas pelo usuário que conferem ao sistema a adaptabilidade resultante da interpretação dos cenários que o sistema por si só não é capaz de estabelecer de forma analítica. Experimentos mostram que a solução proposta pode reduzir até 52% o número de transmissões em relação a outras soluções implementadas.

Abstract

Wireless Sensor Networks (WSN) are networks composed of tens, hundreds or even thousands of sensors and one or more base stations. Sensor devices are scattered over an area to be monitored and communicate via radio. In most scenarios it is not possible to connect these sensors to a power source or replace the battery. Thus, power limitation has a decisive influence on the longevity of the network and is the most striking feature of WSNs. Since sensors are compact devices, they have limited processing power and storage. As a result, for WSN's management of available resources is crucial. The focus of this work is the management of sensed data storage. The data collected by the sensors can be stored in the sensors themselves (producers) in the base station or in designated sensors in the network, called repositories. The choice of the data location has an impact on the system performance for query processing. For example, data storage in a base station requires the transmission of values from every sensor in the field to the station. Thus, the cost of updating the data is high, but the cost of query processing is low since it is executed locally at the base station. On the other hand, the storage of data in the sensors themselves have a low update cost, but the cost of query processing is high since it is necessary to contact all sensors for obtaining the values collected. There are several proposals in the literature that adopt different storage models. However, most of them adopt a static solution, in which the storage model does not adapt to different application contexts. In this work we propose DYSTOPOL, a dynamic model that provides an adaptive solution which adopts the best storage model for a given application context using a distributed index and policies to implement such features. Policies allow the system to be refined under pre-defined settings given by the user. They allow the system to change

storage models based on the interpretation of the scenarios that the system itself is not able to establish analytically. Experiments show that the proposed solution can reduce up to 52 % the number of transmissions compared to other solutions.

Sumário

Lista de Figuras

Lista de Tabelas

Glossário

EB Estação Base 1–3, 8, 9, 14, 15, 30, 35, 38, 39, 42, 44, 46–48

IA Índice de Armazenamento 28, 29, 39, 42, 46, 47

PFDL Policy Framework Definition Language 27

RSSF Rede de Sensores Sem Fios iii, xiii, 1–3, 5, 7–13, 15, 26, 36, 37, 40, 42–44, 47

Scoop *Scoop: An Adaptive Indexing Scheme for Stored Data in Sensor Networks* [11] 3, 27–29, 35, 36, 39, 42, 46, 48

sensor líder é um sensor responsável hierarquicamente sobre um conjunto de sensores. 27

Capítulo

1

Introdução

A recente evolução na tecnologia de micro-processadores e de comunicação levou à criação de novos tipos de redes [2][24], como as Redes de Sensores Sem Fio (RSSFs), que são constituídas por pequenas unidades de sensoriamento, chamados sensores, e por uma ou mais estações base [16]. Os sensores apresentam limitações de recursos computacionais, tanto de processamento como de armazenamento, além de uma bateria com energia limitada. Eles trabalham de forma colaborativa para prover todas as necessidades da rede [14]. A Estação Base (EB) é a interface entre a RSSF e redes externas ou diretamente com os usuários. Em geral ela não tem limitações de energia e tem um poder computacional e de armazenamento superior ao dos sensores. As RSSFs podem ser constituídas por centenas ou até milhares de sensores [3] e a comunicação entre os sensores é realizada através de sinais de rádio. Os dispositivos sensores possuem um tempo de vida limitado devido às restrições energéticas o que limita também a longevidade de toda a rede [2][16][14].

Estudos indicam que o custo de transmissão via rádio é responsável pelo consumo de 2/3 de toda a energia consumida em uma RSSF [7][33][23]. O custo energético com a transmissão de dados é assim um dos principais fatores limitadores da longevidade de uma RSSFs [24]. Como o número de transmissões influencia significativamente o gasto energético, a tarefa de envio de mensagens, seja para o armazenamento ou consulta de dados, precisa ser otimizada de modo a minimizar a quantidade de transmissões da rede.

Existem vários trabalhos [20] [19] [32] [28] [31] que abordam a localização e a agregação de dados para o seu armazenamento em RSSFs. Os modelos de armazenamento existentes em RSSFs podem ser categorizados em três grandes grupos [31]: *(i)* armazenamento *local*, no qual os dados são armazenados localmente em cada sensor e as consultas são disseminadas por inundação na rede; *(ii)* armazenamento *externo*, no qual os dados são enviados para a estação base e as consultas são todas processadas na EB e *(iii)* armazenamento *centrado em dados*, no qual os dados são distribuídos pela rede usando uma estrutura de indexação distribuída sendo armazenados e consultados através do acesso a esse mesmo índice para determinar a localização dos dados.

Para exemplificar a utilidade dos diversos tipos de armazenamento, considere uma RSSF de 50 sensores que tem como objetivo monitorar a área de refrigeração de uma empresa de lacticínios, sendo que deve ser emitido um alarme quando algum sensor detectar uma leitura superior a 10 °C que deve ser amazenada juntamente com o horário do alarme. Neste sistema o alarme pode ser disparado por qualquer sensor com uma leitura superior ao limite. Como não há consultas externas aos valores dos sensores o ideal é utilizar um modelo que implemente armazenamento local nos próprios sensores. Dessa forma as únicas transmissões de dados ocorreriam numa situação hipotética de alarme. Agora imagine que a empresa passa por uma auditoria de qualidade que exige o monitoramento constante da temperatura em todos os pontos da sala. Se o sistema continuar a utilizar um modelo de armazenamento local, seria necessário enviar continuamente mensagens de consultas da EB para os sensores e esperar pelas respetivas respostas dos sensores para a EB. Nesse contexto, a utilização de um modelo externo seria menos dispendioso, em termos de transmissões, pois os dados seriam enviados dos sensores diretamente para a EB e as consultas não teriam um custo associado pois os dados estariam disponíveis na própria EB. Agora suponha que o processo de qualidade permite o monitoramento periódico e com uma margem de erro máxima de 2% nas leituras. A utilização de um modelo de armazenamento externo seria demasiado dispendioso pois os dados não necessitariam estar sempre disponíveis. E o modelo ideal a utilizar seria o modelo centrado em dados em que a localização dos repositórios seria entre os sensores produtores e a EB sendo que

os dados somente seriam enviados para o repositório ao existir uma variação nas leituras superior a 2%. Uma outra possível variante é que a empresa esvazia um setor do seu armazém aos sábados e portanto não precisa controlar essas temperaturas com tanta assiduidade. Dessa forma poderia existir uma política para diminuir a frequência de leituras dos sensores do setor afetado durante todos os sábados.

Os exemplos simples apresentados anteriormente permitem perceber que uma mesma RSSF pode mudar completamente de comportamento dependendo das necessidades. Percebendo tal comportamento é proposto um modelo baseado em um Índice de Armazenamento (IA) que mapeia intervalos de valores sensoriados para um repositório de dados, que pode ser um sensor ou a EB. O repositório é responsável por armazenar as informações dos sensores que coletam dados dentro do intervalo do qual ele é responsável. A localização do repositório não é estática, mas dinâmica, já que novos índices são calculados periodicamente com base na informação colhida e agregada na EB que dá uma visão geral do sistema, assim adequando-se ao contexto atual do sistema. Assim, caso a frequência de consultas aumentar o repositório é atribuido a um sensor mais próximo da EB e caso a frequência de produção de dados aumentar o repositório será atribuído a um sensor mais perto dos sensores produtores. Deve-se salientar no entanto que os dados armazenados durante o período em que um determinado sensor é repositório não são realocados para o novo repositório.

O contexto do sistema é monitorado pela EB através de mensagens enviadas periodicamente dos sensores, denominadas mensagens de sumário. Estas mensagens contêm informações sobre os dados produzidos através de um histograma. Outras mensagens trafegadas no sistema correspondem às mensagens de atualização de leituras do sensor e mensagens de atualização do índice de armazenamento. No modelo proposto a frequência de envio destas mensagens é controlada através de políticas que têm como objetivo reduzir a quantidade geral de transmissões na rede sem, no entanto, diminuir a precisão dos resultados das consultas. Embora possa ser utilizado em diversas RSSFs este trabalho é mais apropriado para cenários de sensoriamento urbano ou onde exista similaridade espacial e temporal de dados.

Este trabalho contém um conjunto de políticas que definem o comportamento do sistema perante alguns cenários. Essas políticas são definidas pelo usuário, antes da inicialização do sistema, e implementadas na EB assim como nos sensores e contêm um dos seguintes domínios: global, se a política influencia todo o sistema, ou local, se influencia somente um sensor ou um conjunto de sensores. A utilização deste sistema de políticas permite definir diferentes comportamentos para diferentes regiões do sistema aumentando a sua adaptabilidade. As políticas têm um papel fundamental no refinamento de parametrizações que sem as políticas seriam fixas, com o objetivo de reduzir o número total de transmissões do sistema.

1.1 Objetivos

A dissertação tem como objetivo definir um modelo dinâmico de gestão de armazenamento de dados através da utilização de políticas que definam o comportamento do sistema perante vários cenários relacionados com a frequência de geração de dados e de coleta dados.

O objetivo do modelo é a otimização do número total de transmissões efetuadas pelos sensores no sistema. Como para o armazenamento e acesso dos dados sensoriados as transmissões de rádio representam uma grande fatia do consumo de energia, ao diminuir as transmissões é reduzido o consumo de energia. Assim o objetivo é diminuir o número de transmissões sem perder a precisão nas respostas a consultas.

1.2 Contribuições

Esta dissertação apresenta as seguintes contribuições:

- utilização de limites dinâmicos e particionamentos por intervalos nos vários cálculos de custos do sistema.
- adoção de *thresholds* definidos pelo usuário para determinar as frequências de transmissão de mensagens contendo informação sobre o monitoramento do sistema.

- utilização de políticas para controlar parâmetros do sistema.
- diversos experimentos, baseados em simulações, mostrando que as estratégias propostas podem reduzir significativamente o número de transmissões do sistema.

1.3 Organização

O restante deste trabalho está organizado da seguinte forma: o Capítulo 2 descreve vários conceitos utilizados no decorrer do trabalho e que são apresentados com o intuito de lançar a base para a organização do Capítulo 3, relativo aos trabalhos relacionados. O Capítulo 3 apresenta uma série de trabalhos relacionados com as diversas abordagens sobre armazenamento presentes na literatura assim como aplicações de políticas em RSSFs. No Capítulo 4 é apresentado um trabalho preliminar com vista à implementação e pré-validação de algumas ideias fundamentais. No Capítulo 5 são apresentados o modelo e a proposta formal do sistema de políticas, assim como alguns experimentos que validam a solução proposta. A dissertação conclui no Capítulo 6 com algumas considerações finais e trabalhos futuros.

Capítulo

2

Conceitos

In order to survive, any enterprise has to change as the world changes around it.

[4]

Neste capítulo são apresentados diversos conceitos sobre as três áreas chave deste trabalho: Redes de Sensores Sem Fio (RSSFs), métodos de armazenamento de dados e políticas.

2.1 Redes de Sensores Sem Fio

As Redes de Sensores Sem Fio (RSSFs) têm atraído a atenção da comunidade científica nos últimos anos. Contribuiu para isso o desenvolvimento recente de diversas tecnologias que as tornaram mais eficientes, compactas, autônomas e baratas [2][24]. Essas redes são compostas por dezenas ou até milhares de nós sensores [3]. Cada sensor tem uma capacidade limitada de processamento, de armazenamento e de energia [3][16]. A comunicação dos sensores é efetuada através de transmissões de rádio de curto alcance. Logo, o roteamento de mensagens é cooperativo entre os sensores. Cada sensor necessita ser autônomo em relação à sua organização e administração [30].

Sensores são dispositivos equipados com três funções fundamentais: sensoriamento, processamento e comunicação. Assim os sensores têm essencialmente as seguintes tarefas: *a)* sensoriamento de um ambiente, ou seja, monitoramento de uma ou mais características do ambiente em que está inserido, por exemplo temperatura de uma determinada rua; *b)* processamento de informações e geração de novas mensagens, que envolve a computação local do sensor com base em dados sensoriados ou informações adquiridas de sensores vizinhos ou da EB; *c)* tarefas associadas com o tráfego em um esquema de retransmissão multi-salto. Os sensores coletam dados através de sensoriamento do ambiente e os processam localmente ou coordenadamente entre seus vizinhos, sendo que os valores resultantes podem então ser armazenados localmente no próprio sensor, enviados para um outro sensor ou para uma EB. A EB é um dispositivo pertencente à rede, embora não tenha as mesmas limitações de recursos dos sensores. Por isso é responsável por diversos serviços da rede considerados pesados para serem efetuados diretamente nos sensores. Geralmente as EBs são o ponto de entrada das consultas na RSSF.

A escolha do caminho a percorrer para entregar as mensagens recai sobre as árvores de repasse que são construídas dinamicamente com base no conhecimento adquirido diretamente através dos sensores vizinhos e de mensagens encaminhadas e que contêm a informação do caminho percorrido até ali. Após uma série de iterações é possível definir uma série de arvores de repasse que permitem a distribuição correta das mensagens.

Cada sensor possui vários componentes, tais como: processador, memória, bateria, transceptor de rádio (dispositivo que combina um transmissor e um receptor de rádio responsável pela comunicação) e o componente responsável pelo sensoriamento. Os processadores atuais são geralmente de 8 bits com frequência de 10 MHz. Os transceptores têm largura de banda de 1 kbit/s a 1 Mbit/s e a capacidade de memória pode ser de 128 Kbytes a 1 Mbyte. Há diferentes tecnologias de fabricação de baterias. A longevidade de uma RSSF é muito variável e depende da energia disponível em cada sensor. Em algumas situações a localização remota dos sensores ou o ambiente hostil não permitem a manutenção e reposição da energia dos sensores. Em consequência, a gestão eficiente da

energia disponível assume um papel fundamental na longevidade e disponibilidade dessas redes.

Tendências atuais sugerem que o custo por bit de transmissão via rádio vai continuar a dominar o custo de armazenamento e leitura de memória [7][33][23]. Por exemplo o custo de escrita de 1 bit num chip de memória *flash Micron Technology 128 Mbit NX25P32* é de $2,8 * 10^{-5}$ mJ [11]. Leituras são substancialmente mais baratas. Em contraste, atualmente os rádios que utilizam a tecnologia 802.15.4 consomem cerca de 15mJ [11] de energia por segundo, fazendo com que o consumo do rádio seja 2500% mais dispendioso que o armazenamento. Assim a redução do número de transmissões é uma forma de reduzir o consumo de energia e assim aumentar a longevidade do sistema.

Algumas RSSFs consideram a existência de uma EB que é um dispositivo, ao contrário do sensor, sem limitações de energia, com maior capacidade de armazenamento e de processamento. Assim a EB é, em muitos modelos, responsável pelo armazenamento de dados da rede, criação de índices, gestão da estrutura de rede entre outras responsabilidades. O ponto de acesso é o elemento através do qual a rede se comunica com outras redes ou com um ou mais observadores. O ponto de acesso pode ser implementado em um sensor ou em uma estação base. Em algumas RSSFs os sensores são organizados de forma hierárquica através da formação de grupos (*clusters*) de sensores onde existe um ou mais sensores líder (*cluster-head*). O sensor líder é eleito entre os sensores do grupo ou designado por uma entidade externa ao grupo. Existem ainda dentro do sistema os *sensores produtores*, ou seja, sensores que produzem um determinado dado e *sensores responsáveis*, ou repositórios, que são responsáveis pelo armazenamento desses dados. Um sensor pode ao mesmo tempo ser um sensor produtor e responsável.

O objetivo mais comum de uma RSSF é o monitoramento de um determinado ambiente do mundo real, mas alguns sistemas podem, além de monitorar esse ambiente, também exercer controle sobre ele. Cada sensor pode ser equipado com diversas modalidades de sensoriamento, tais como: temperatura, pressão e sísmico.

2.1.1 Características

As RSSFs apresentam características distintas das redes cabeadas. Devido à limitação de recursos e à natureza da rede existem algumas características particulares das RSSFs que são descritas a seguir.

- *Escalabilidade* - O sistema deve dar suporte um número elevado de sensores, adaptando-se, assim, a várias áreas potenciais de aplicação desta tecnologia.

- *Sensoriamento Distribuído* - Uma RSSF permite coletar uma quantidade maior de informações se comparado com somente um sensor. Mesmo que utilizando um sensor com um alcance maior, ele poderá ter obstáculos no seu raio de sensoriamento. Assim, o sensoriamento com vários sensores distribuídos tem mais precisão e confiabilidade dos dados coletados.

- *Tolerância a Falhas* - O sistema deve ser robusto em relação às diversas falhas possíveis de sensores, tais como perda total de energia, destruição física e adormecimento. Essas falhas devem ser tratadas como algo inerente do sistema.

- *Auto-organização* - Sensores em uma RSSF podem ser perdidos por causa de sua destruição física ou falta de energia. Sensores também podem ficar incomunicáveis devido a problemas no canal de comunicação sem fio ou por decisão de um algoritmo de gerenciamento da rede. Isso pode acontecer por diversas razões como, por exemplo, para economizar energia ou devido à presença de outro sensor na mesma região que já coleta o dado desejado. A situação contrária também pode acontecer: sensores inativos se tornarem ativos ou novos sensores passarem a fazer parte da rede. Em ambos os casos, é necessário haver mecanismos de auto-organização para que a rede continue a executar a sua função. Essa configuração deve ser automática e periódica já que a configuração manual não é viável devido a problemas de escalabilidade.

- *Limitação de energia* - Em muitas aplicações, os sensores são colocados em áreas remotas, o que dificulta o acesso a esses elementos para manutenção. Neste cenário,

o tempo de vida de um sensor depende da quantidade de energia disponível. Aplicações, protocolos e algoritmos para RSSFs não podem ser escolhidos considerando apenas sua elegância e capacidade, mas também a quantidade de energia consumida.

- *Restrições dos dados coletados* - Indica se os dados coletados pelos sensores têm algum tipo de restrição como um intervalo de tempo máximo para disseminação de seus valores para uma dada entidade de supervisão como um sensor líder.

- *Tarefas colaborativas* - O objetivo principal de uma RSSF é executar alguma tarefa colaborativa na qual é importante detectar e estimar eventos de interesse e não apenas prover mecanismos de comunicação. Devido às restrições das RSSFs, normalmente os dados são sumarizados para melhorar o desempenho no processo de detecção de eventos. O processo de sumarização é dependente da aplicação que está sendo executada.

- *Comunicação multi-salto* - A comunicação direta com o ponto de acesso pode não ser possível a todos os sensores. A solução passa pelo envio da mensagem através de outros sensores até que ela chegue ao ponto de acesso. De acordo com estudos existentes, esta abordagem é mais eficiente que a ampliação de sinal para o contato direto com o ponto de acesso[5].

2.1.2 Classificação

A classificação de uma RSSF depende de seu objetivo e área de aplicação. A aplicação influencia diretamente as funções exercidas pelos sensores da rede, assim como a arquitetura desses sensores. A classificação, segundo [27], pode ser feita baseada em diversos fatores, tais como: a da capacidade de seus componentes (processador, memória, dispositivos sensores, fonte de energia e transceptor), quantidade de sensores que compõem a rede, distribuição dos dispositivos, escolha dos protocolos da pilha de comunicação, tipo de dado que será tratado, tipo de serviço que será provido pela rede e longevidade dessa rede.

Classificação		
Organização	Hierárquica	RSSF em que os sensores estão organizados em grupos e em cada grupo existe um ou mais sensores líderes.
	Plana	Rede em que os sensores não estão organizados em grupos, ou seja, não existe nenhuma distinção hierárquica nem de funções entre os sensores.
Coleta	Periódica	Os sensores coletam dados sobre o(s) fenômeno(s) em intervalos regulares.
	Contínua	Os sensores coletam os dados continuamente.
	Reativa	Os sensores coletam dados quando ocorrem eventos de interesse ou quando solicitado pelo observador.
	Tempo Real	Os sensores coletam a maior quantidade de dados possível no menor intervalo de tempo.
Propagação	Programada	Os sensores propagam os dados em intervalos regulares.
	Contínua	Os sensores propagam os dados continuamente.
	Sob Demanda	Os sensores propagam os dados em resposta à consulta do observador e à ocorrência de eventos.
Cooperação	Infra-estrutura	Os sensores executam procedimentos relacionados à infra-estrutura da rede.
	Localizada	Os sensores executam além dos procedimentos de infra-estrutura, algum tipo de processamento local básico.
	Correlação	Os sensores estão envolvidos em procedimentos de correlação de dados como fusão, contagem, compressão e agregação.

Tabela 2.1: Classificação das RSSFs utilizada nesta dissertação [27]

Na Tabela 2.1 é apresentada uma classificação baseada na proposta de Ruiz, L.B. [27]. Das diversas categorias sugeridas, são apresentadas aquelas relacionadas ao tema desta dissertação, a saber: a organização da rede, a coleta de dados, a propagação dos dados e a cooperação de processamento entre os sensores.

2.2 Modelos de Armazenamento

Existem várias propostas na literatura sobre armazenamento de dados para RSSFs. Nesta seção é apresentada uma classificação destas propostas, enquanto os sistemas são apresentados em detalhe no Capítulo 3. A classificação compreende três grandes grupos [31][10]: armazenamento local, armazenamento centrado em dados e armazenamento ex-

terno. Cada modelo de armazenamento tem vantagens e desvantagens dependendo do cenário de rede e das suas características. Como neste trabalho é considerada uma grande dinâmica entre cenários e configurações, serão abordados em detalhe cada um dos modelos assim como as suas principais vantagens, desvantagens, cenários e configurações ideais.

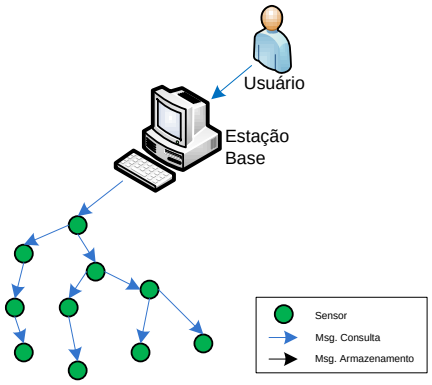


Figura 2.1: Modelo de armazenamento local.

2.2.1 Local

Nesse modelo de armazenamento os dados gerados são armazenados localmente, ou seja, no próprio sensor que realizou a coleta. Todos os dados produzidos ficam armazenados no próprio sensor, sendo que não é enviada qualquer informação para a estação base. A Figura 2.1 apresenta o fluxo de transmissões necessárias para o armazenamento e consulta dos dados do sistema. O custo de transmissão de dados para armazenagem é nulo, pois eles são armazenados no próprio sensor. Já as consultas são geralmente realizadas por inundação da rede, pois a estação base não tem conhecimento da localização dos dados a consultar, o que representa um custo bastante elevado de transmissões. Em sistemas com uma alta frequência de consultas este modelo terá um custo de consultas bastante elevado. Esse modelo de armazenamento é ideal para RSSFs com uma frequência muito baixa de consultas ou com frequências de geração de dados muito elevada. Devido à utilização de inundação para efetuar as consultas é um modelo que se adapta melhor a RSSFs com baixa frequência de consultas.

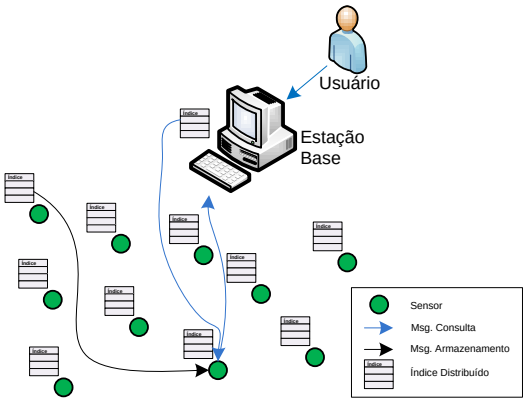


Figura 2.2: Modelo de armazenamento centrado em dados.

2.2.2 Centrado em Dados

Nesse modelo os dados são distribuídos pela rede utilizando estruturas de indexação distribuída ou de sumarização e agregação que mapeiam os dados para localizações de armazenamento específicas. É possível observar na Figura 2.2 que cada elemento da rede tem acesso à localização dos dados, sendo que essa informação é importante para o armazenamento e para a consulta. Assim é possível encontrar de imediato o sensor responsável pelos dados. A manutenção de uma estrutura de indexação representa um custo adicional, pois requer a coleta de informações de contexto dos sensores para a EB e posterior distribuição do índice calculado. A informação existente no índice distribuído deve ser atualizada de forma contínua ou periódica. Para que esta forma de armazenamento seja recomendável, o custo do número de transmissões necessárias para a manutenção dos índices deve ser inferior ao ganho com a otimização dos repositórios. Este modelo apresenta uma maior adaptabilidade do que os demais pois ao coletar informações, ele pode se adaptar a mudanças existentes na rede. No entanto, ele é mais complexo e por isso deve ser organizado hierarquicamente ou utilizar estratégias de redução dos custos de manutenção, principalmente para redes com grande quantidade de sensores.

2.2.3 Externo

No armazenamento externo, os dados são coletados e enviados para uma entidade externa, como a estação base. Em alguns casos todos os dados gerados são enviados ou em outras

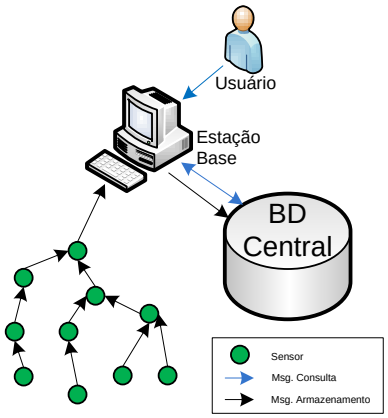


Figura 2.3: Modelo de armazenamento externo.

situações esses dados podem ser enviados com algum tipo de agregação e sumarização ou através de estatísticas. Todas as consultas são processadas diretamente na estação base já que os dados estão disponíveis. Na Figura 2.3 é possível observar que o custo de transmissão de dados é muito elevado pois todos os dados devem ser enviados para a EB e o custo de consultas extremamente baixo. Esse modelo é ideal para RSSFs com frequência baixa de geração de dados ou com um volume muito significativo de consultas.

2.2.4 Resumo

Na Figura 2.4 é possível comparar os custos dos três modelos de armazenamento descritos anteriormente. Como as aplicações possuem comportamentos distintos, torna-se fundamental a definição do modelo de armazenamento mais adequado. Se o cenário se mantiver em constante mudança o modelo com potencial para minimizar o número de mensagens para esse cenário também pode, da mesma forma, mudar. Assim podemos concluir que a escolha do modelo de armazenamento deve depender do cenário atual da aplicação, ou seja, que possa ser ajustado dinamicamente ao longo do tempo.

2.3 Políticas

Políticas são regras que definem as escolhas de comportamento de um sistema [1]. Elas são utilizadas como um meio para implementar sistemas adaptativos e flexíveis para a gestão de diversos serviços, tais como de Internet, de sistemas distribuídos e de segurança. Com

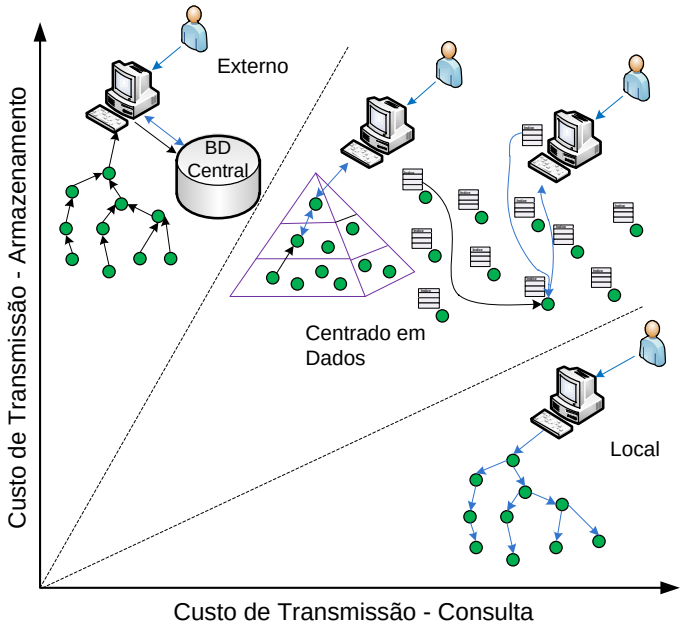


Figura 2.4: Custos de transmissões para os vários modelos de armazenamento.

a contínua integração de novas tecnologias aos sistemas modernos, a gestão de todos esses recursos tem também aumentado a sua complexidade. Surge assim a necessidade de gerir algumas funcionalidades de forma automática, sem a intervenção humana. A utilização de políticas neste contexto tem aumentado, pois permite uma separação entre o comportamento e as funcionalidades de um sistema. Isso significa que é possível adaptar o comportamento de um sistema sem a necessidade de re-codificar sua funcionalidade e as alterações podem ser implementadas sem paradas do sistema. Um comando isolado para executar uma ação não é uma política. Políticas são persistentes e podem ser divididas em 4 grandes tipos [4]:

- *Gestão* - definem as condições em que o agente administrador executa uma ação no sistema para mudar o seu desempenho ou a estratégia de alocação de recursos. Como exemplo de utilização deste tipo de política, podem ser citados: o aumento da frequência de envio dos dados dos sensores de tráfego rodoviário entre 08:00 às 10:00 em todos os dias úteis ou, noutro exemplo, quando devem ser realizadas as políticas de *garbage collection* dos sensores. As políticas de gestão são geralmente implementadas como regras na forma de condição - ações determinadas pela ocorrência de um evento.

- *Comportamento* - definem as regras de interação entre os agentes do sistema. No contexto de RSSF, exemplos de utilização deste tipo de política incluem trocar informações sobre o estado do dispositivo sensor ou a periodicidade com que um sensor deve enviar informações para a estação base.

- *Segurança* - definem regras de acesso dos usuários ou agentes a determinados recursos do sistema. Como exemplo, citam-se: regras de um *firewall*; que tipo de ação tomar após uma falha de login e quais recursos devem ser desligados em caso de ataque ao sistema.

- *Roteamento* - definem condições para a escolha de diferentes caminhos para o roteamento de pacotes. A escolha de um caminho diferente de roteamento para poupar energia em uma RSSF é um exemplo de política de roteamento.

Cada um desses tipos de políticas pode ainda ser dividido em políticas de obrigação(deveres) ou de autorização(direitos). As políticas de obrigação definem uma ação mandatória e incidem sobre objetos da rede. As políticas de autorização definem uma permissão sobre o acesso a um recurso.

Por norma, as políticas são definidas na camada de aplicação que, por sua vez, devem ser transformadas pelo sistema em políticas de níveis mais baixos expressas em termos de operações suportadas pelo sistema. Se não forem suficientes podem ser definidas políticas que interajam com as camadas inferiores. Esse processo é chamado de refinamento de políticas.

Definição 2.1. Uma regra de política é uma quíntupla (S, T, A, C, Tr) [4], onde:

- S - *Sujeito* - componente que define quais os usuários ou agentes são autorizados a efetuar uma ação. Em uma política de obrigação o sujeito é o agente que interpreta a política e executa a ação. Ele pode ser específico a um agente ou grupo de agentes ou inferido pela análise de contexto por parte do sistema.

- T - *Alvo* - define o escopo em termos de recursos ou objetos. Ele age sobre recursos, caso se trate de uma política de autorização, e de objeto se for de obrigação.

- *A - Ações* - especifica o que deve ser efetuado nas obrigações e o que é permitido nas autorizações. Elas podem ter uma ou mais operações ou conter uma série de regras para várias operações.
- *C - Condições* - definem o predicado, ou conjunto de predicados, que devem ser avaliados antes da ação ser tomada. Elas podem estar relacionadas com o tempo ou com atributos do sujeito ou alvo.
- *Tr - Triggers* - definem eventos que iniciam ações em uma política de obrigações. Eles nem sempre são definidos explicitamente na política.

□

É possível, utilizando a definição anterior, criar um conjunto de regras que representam possíveis escolhas de mudanças no comportamento de um sistema. Assim um conjunto de regras de políticas definem a fração adaptável de um sistema.

2.3.1 Classificação

Classificação de Políticas		
Domínio	Global	Política definida para todos os sensores.
	Local	Política definida para um sensor ou um grupo de sensores.
Duração	Estático	Política sempre ativa.
	Dinâmico	Política ativa num determinado intervalo de tempo.

Tabela 2.2: Classificação proposta para políticas.

As políticas, como é possível observar pela Tabela 2.2, podem ser classificadas segundo dois critérios. O domínio define a área de influência de uma política. Uma regra com um domínio global indica que é uma regra que deve ser implementada em todos os sensores. Um exemplo seria uma regra para adormecer durante a noite todos os sensores que monitoram o canto de uma ave diurna. Políticas locais são restritas a um ou a um grupo de sensores. Um exemplo seria o aumento da frequência de envio de dados em caso de aumento significativo de temperatura em uma região, podendo indicar um incêndio.

A duração de uma política pode ser estática ou dinâmica. Uma duração estática indica que a política deve estar continuamente ativa. Uma duração dinâmica indica que a política só estará ativa durante um período de tempo. Esse período de tempo pode ser expresso em intervalo ou em tempo desde o inicio da aplicação. Um exemplo desse tipo seria uma política que define o backup dos dados dos sensores para a estação base em períodos com uma frequência de consultas muito inferior à média.

2.4 Sumário

Neste capítulo foram apresentados os conceitos de base utilizados nesta dissertação. As redes de sensores sem fio representam a estrutura de rede utilizada. Com os recursos limitados dos sensores a gestão eficiente do sistema é um desafio. A escolha do modelo de armazenamento tem uma importância chave pois a escolha errada do modelo pode transformar radicalmente a sua eficiência. Por fim as políticas providenciam adaptabilidade a um sistema pois é possível definir regras que alterem o comportamento de forma dinâmica após mudanças no contexto do mesmo. No próximo capítulo serão descritos em pormenor alguns trabalhos relacionados com os conceitos enunciados.

Capítulo

3

Trabalhos Relacionados

Neste capítulo são apresentados diversos trabalhos propostos na literatura e que são relacionados ao trabalho descrito nesta dissertação. Como indicado no capítulo anterior, os pontos chave deste trabalho são modelos de armazenamento e políticas para RSSFs. Assim nas próximas seções, são apresentados alguns trabalhos relacionados a estes temas.

3.1 Sistemas de Armazenamento

Nesta seção são apresentados diversos trabalhos relacionados com modelos de armazenamento de dados para RSSFs. O objetivo desta exposição é mostrar a diversidade de soluções, tendo cada sistema um conjunto de vantagens e desvantagens que estão intrinsecamente associadas ao cenário da RSSF existente. No final da seção é feito um resumo para agrupar as diversas idéias dos trabalhos relacionados com as idéias propostas nesta dissertação.

3.1.1 CAG

No modelo *The Clustered AGgregation (CAG) technique leveraging spatial and temporal correlations in wireless sensor networks* [32] é proposta uma estratégia de processamento de consultas baseada em duas fases, uma de consulta e outra de resposta. Na fase de

consulta, os agrupamentos são construídos dinamicamente juntamente com a disseminação da consulta pela árvore de repasse. Na fase de resposta, ao invés de enviar todos os dados coletados para a estação base, o sistema abdica da precisão do resultado para economizar energia e transmite somente um valor por agrupamento.

A fase de resposta começa depois de um tempo determinado para que seja possivel que todos os sensores recebem a consulta. Nesta fase, apenas os sensores líderes, transmitem as leituras de volta à estação base. Como a árvore de repasse é baseada nas caraterísticas dos sensores e da rede, os *cluster-heads* podem ser mudados a cada ciclo de consulta e resposta. Isso previne a sobre-utilização de um determinado conjunto de sensores, prevenindo também possíveis situações de gargalo da rede.

O CAG opera em dois modos distintos, interativo e *streaming*. No modo interativo é criada uma única resposta a uma consulta. Já no modo *streaming* são enviadas respostas periódicas. Este sistema agrupa os sensores de forma hierárquica tendo como parâmetro de formação dos grupos os valores sensoreados de cada sensor. A coleta dos dados é feita de forma reativa no modo interativo, já que os dados são coletados quando uma consulta é injetada no sistema, ou de forma periódica no modo *streaming*, para as respostas periódicas. A propagação dos dados é feita sob demanda já que os dados são enviados, somente no modo interativo, em resposta a consultas. Os sensores são envolvidos no procedimento de criação de grupos e disseminação de consultas. O CAG apresenta uma economia de energia na fase de resposta, mas tem como pontos negativos a inundação da rede a cada consulta e o tamanho indefinido dos grupos.

3.1.2 GHT

Similar a uma DHT (*Distribuited Hash Table*) para redes P2P [6][26] que distribui os dados na rede através de uma função que mapeia os dados a um ponto na rede, a *Geographic Hash Table for Data-Centric Storage (GHT)* [25] propõe um modelo de acesso aos dados baseado numa função de distribuição que mapeia valores em coordenadas geográficas. Na GHT um determinado atributo sensoreado é associado a uma chave k e cada sensor do sistema pode armazenar dados dos quais é o responsável. É utilizada uma função de

espalhamento *(hash)* para disseminar os dados pela rede. Essa função de espalhamento mapeia uma chave para uma posição geográfica que terá como repositório o sensor mais próximo dessa posição.

Como resultado da aplicação da função de espalhamento, o sensor mais próximo à coordenada é escolhido como hospedeiro e todos os sensores mais próximos deste são replicadores dos seus dados. Esses sensores mais próximos são denominados de perímetro doméstico. O armazenamento de dados no hospedeiro e a replicação no seu perímetro doméstico formam um repositório de dados centrado na rede. Apresentando uma vantagem de disponibilidade dos dados mesmo quando os sensores saem e entram frequentemente da RSSF. A escolha do hospedeiro que será o repositório da região endereçada pela coordenada Z é feita pelo protocolo *Perimeter Refresh Protocol (PRP)*, que assegura persistência e consistência em caso de falhas ou de mobilidade dos sensores. Como as funções de espalhamento tradicionais, também a GHT tem duas funções básicas:

- *Put(k,v)* - armazena o valor sensoreado v conforme o nome da chave k.
- *Get(k)* - obtém o dado representado pela chave k.

O mais importante neste sistema é o mapeamento de chaves em localizações geográficas. A função de espalhamento garante que chaves iguais tenham a mesma localização. As consultas neste sistema são bastante simples e não necessitam inundar a rede já que a localização do sensor responsável é imediata. No entanto, pode-se perceber que se houver uma grande quantidade de valores para uma mesma chave o sensor responsável pelo armazenamento fica sobrecarregado. Também é importante observar que a atribuição da localização considera somente o nome da chave, que corresponde ao nome de um atributo. Assim, o armazenamento pode ser feito demasiado longe do sensor produtor e mesmo da estação base, aumentando assim o custo para o armazenamento e para a consulta aos dados.

3.1.3 HDMST

Em *Hierarchical Data Management for Spatial-Temporal Information in WSNs (HDMST)*[31] é proposta uma técnica de hierarquização de *cache* de dados. Essa técnica consiste em criar três níveis de disponibilidade compostos por sensores, sensores de *cache* e EB para responder a consultas de forma eficiente e com diversos níveis de qualidade de resposta. Os níveis são: o nível de controle e monitoramento, o nível de cache e de sensor. Na figura 3.1 está representado um exemplo dessa arquitetura. No nível de sensor os dados são mantidos sem qualquer tipo de sumarização. São armazenados também os *timestamps* de cada dado. No nível de cache são armazenados sumários de dados através de linhas de tendência calculadas e enviadas pelos sensores. O nível de controle e monitoramento é composto somente pela EB e não armazena dados. É nesse nível que as consultas são distribuídas para o nível de cache e para os níveis inferiores.

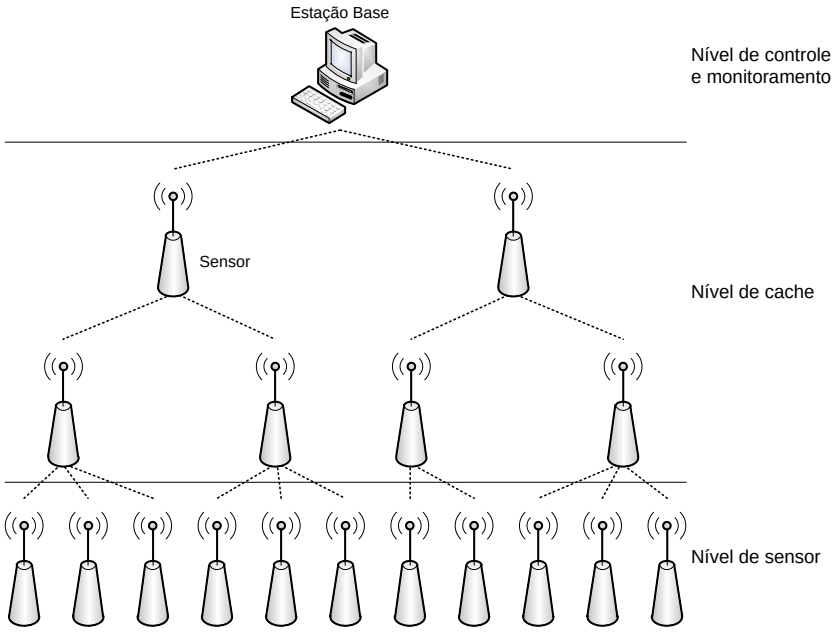


Figura 3.1: Arquitetura de HDMST. [31]

Cada sensor cria parâmetros de previsão e de sumarização dos próprios dados e envia essa informação aos sensores de *cache*. Eles recebem e agregam os vários parâmetros de previsão podendo assim fornecer respostas aproximadas de consultas. Quando a precisão da resposta não satisfizer o intervalo de confiança da consulta esta é encaminhada para o

nível inferior. Ao utilizar linhas de têndencia diminui a necessidade de transmitir dados pela hierarquia no entanto aumenta a probabilidade de responder a consultas com erros que podem ser significativos nas camadas superiores.

3.1.4 DACS

DACS: A Dynamic Approximative Caching Scheme for Wireless Sensor Networks [13] propõe um modelo de armazenamento e consulta independente de topologia. DACS tem como objetivo a redução dos custos de transmissão associados com consultas. Para tal, durante o percurso até o sensor, o sistema tenta responder à consulta utilizando o cache de dados armazenados. Ao percorrer o caminho entre a estação base e o sensor produtor, cada sensor contém um cache sumarizado do dado a pesquisar. Com a aproximação do sensor produtor o cache contém dados cada vez menos sumarizados, e assim mais precisos. A Figura 3.2 apresenta um exemplo deste método, no qual a cada salto a precisão do dado sumarizado aumenta 10%.

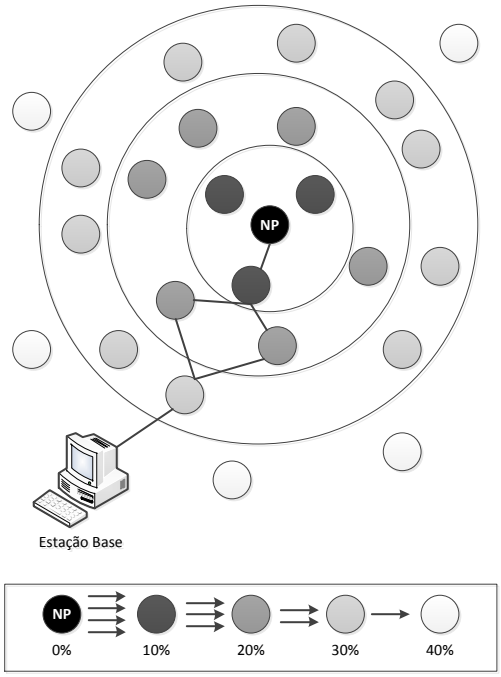


Figura 3.2: Exemplo de um sistema DACS. [13]

Qualquer sensor pode armazenar dados de cache. O sensor armazena não o dado exato, mas um dado aproximado definido dentro de um desvio ou erro máximo. Se esse

valor for ultrapassado é necessário atualizar o cache. O desvio máximo é calculado com base na distância do sensor cache para o sensor produtor.

No DACS a rede é dividida em duas camadas: camada de cache e camada de atualização. A camada de cache indica a distância entre um sensor e a estação base e é utilizada para disseminar as consultas. A camada de atualização indica a distância entre um sensor e o sensor produtor e é utilizada para atualizar os caches. Os sensores são classificados conforme a camada em que se encontram. Cada anel lógico da camada de atualização tem a mesma distância em número de saltos para o sensor produtor e o mesmo valor máximo de desvio ou erro dos dados.

As consultas dos usuários definem uma margem de erro máximo. O sistema utiliza esse erro para escolher as camadas que podem responder à consulta dentro dos limites estipulados e que necessitem do menor número de saltos. A utilização de camadas com diferentes margens de erro permitem a gestão entre o custo e a qualidade de resposta. No entanto a manutenção do sistema de camadas é dispendioso se pensarmos em centenas ou milhares de sensores produtores.

3.1.5 SCOOP

Nesta seção é apresentado o *Scoop: An Adaptive Indexing Scheme for Stored Data in Sensor Networks* [11], um sistema de indexação e consulta de dados armazenados em redes de sensores sem fios. O sistema funciona com base na coleta de estatísticas sobre a frequência de consultas e frequência de leituras dos dados gerados em uma rede de sensores. As estatísticas são usadas para construir um índice que atribui um intervalo de leituras a um sensor, chamado de repositório. Este é responsável pelo armazenamento das informações dos sensores que coletam informações dentro desse intervalo. Usando este índice, as consultas de um usuário sobre os dados armazenados podem ser respondidas de forma eficiente, sem inundar as consultas em toda a rede. O Scoop é também capaz de se adaptar às alterações na distribuição e frequência de dados e consultas.

Como em outros sistemas, dados de sensoreamento são continuamente produzidos pelos sensores. A localidade do armazenamento dos dados coletados é determinado por

um Índice de Armazenamento (IA) adaptativo, que é produzido pela estação base. Os valores são divididos em intervalos e para cada intervalo o IA determina se os dados devem ser armazenados nas memórias flash do próprio sensor produtor, na memória flash de um sensor próximo, ou na Estação Base (EB). Baseado no IA, as consultas para determinar quais sensores possuem valores em um determinado intervalo podem ser respondidas de forma eficiente.

O Scoop se adapta entre os dois extremos de armazenamento: o armazenamento externo e o armazenamento local. O IA é periodicamente atualizado pela estação base e depois transmitido para todos os sensores. Um exemplo de IA é apresentado na Tabela 3.1, que corresponde ao armazenamento de temperaturas por um período de tempo $T1 - T2$. A coluna do lado direito determina o sensor responsável por armazenar todas as leituras de temperatura que se encontrem no intervalo de valores da coluna da esquerda, durante $T1 - T2$. Os intervalos são exclusivos, ou seja sem sobreposição de valores para que seja inequívoco qual o sensor responsável para cada leitura. Sensores podem, no entanto, ser responsáveis por vários intervalos de valores.

Temperatura Tempo:$T1 - T2$	
Valores	Sensor
07 - 12	2
13 - 15	4
16 - 20	7
...	...
30 - 40	13

Tabela 3.1: Exemplo de índice de armazenamento.

Para cada período de tempo e para cada atributo indexado existe um IA. A estação base cria um IA com base em estatísticas coletadas entre cada recálculo do índice. O mapeamento é escolhido para minimizar o número total de mensagens que o sistema envia. Esta abordagem baseia-se na percepção de que valores produzidos recentemente tendem a ser um bom previsor de valores produzidos num futuro próximo. Esta correlação temporal demonstrou estar presente em vários artigos sobre a utilização de estatísticas em modelos de previsão de valores de sensores [15] [8].

Após o cálculo do IA, ele é particionado em vários pacotes e transmitido por *broadcast* a todos os sensores da rede juntamente com o seu identificador (ID) que é incrementado a cada cálculo. Se algum sensor não receber totalmente o IA, ele continua utilizando o mapeamento anterior. Caso nunca tenha recebido um IA os dados produzidos são armazenados localmente e se os valores produzidos estiverem fora dos limites do IA os dados são armazenados nos repositórios com o intervalo mais próximo. Observe que o envio de uma nova leitura para o repositório é uma transmissão multi-salto. Assim se um sensor estiver encaminhando a leitura de um sensor, que não ele mesmo, para um repositório, ele deve validar se o ID do índice de armazenamento que acompanha a mensagem é inferior ao existente localmente. Em caso afirmativo ele deve direcionar, para o repositório responsável pelo valor de acordo com o IA mais recente.

Existem 4 tipos de mensagens no sistema Scoop. Cada tipo de mensagem tem um função especifica que é demonstrada de seguida:

- *Mensagem de Sumário* - É uma mensagem enviada periodicamente dos sensores para a EB. Ela contém informações sobre as últimas leituras do sensor representadas em um histograma, os valores máximo, mínimo e somatório. Ela contém também informação sobre a qualidade de ligação com cada um dos sensores vizinhos e a frequência de envio de mensagens de dados.

- *Mensagem de Dados* - É uma mensagem enviada entre o sensor produtor de determinados dados e o sensor responsável pelo armazenamento dos mesmos. Agrega as últimas 5 leituras efetuadas e contém o identificador do IA utilizado para determinar qual o sensor responsável.

- *Mensagem de Mapeamento* - É uma mensagem enviada da estação base para todos os sensores contendo o último índice de armazenamento gerado. Esta mensagem é enviada logo após o cálculo do índice de armazenamento.

- *Mensagem de Consulta / Resposta* - É uma mensagem enviada da estação base para o sensor responsável por determinado intervalo pedido pelo usuário. Assim

que recebe a mensagem de consulta o sensor envia a mensagem de resposta com os dados solicitados para a estação base.

Um ponto importante a ser observado é que com o conteúdo das mensagens de sumário, a EB consegue prever a probabilidade de cada sensor produzir determinados valores no futuro próximo, baseado em suas últimas leituras. Além disso, como a EB também é o ponto de entrada das consultas, a frequência na qual elas são submetidas também é conhecida. Com base nestas informações é possível a adaptação do sistema a mudanças na rede. Se existirem mudanças na frequência de produção ou de consulta de um determinado valor o sensor responsável pelo armazenamento também pode mudar. Essas heurísticas de adaptação do sistema são apresentadas na Tabela 3.2. Ao existir um aumento na frequência de consulta do dado ele é armazenado mais próximo à EB e se o aumento ocorrer na frequência de geração do dado então esse é armazenado mais próximo ao sensor produtor.

Freq. Consulta	Freq. Dados	Mudança
⇑	=	Aproximar o sensor responsável da EB
=	⇑	Aproximar o sensor responsável do produtor
=	=	Aproximar o sensor responsável do produtor

Tabela 3.2: Dinâmica de armazenamento do Scoop.

3.1.6 Discussão sobre os Sistemas de Armazenamento

É possível observar através dos trabalhos apresentados anteriormente que existem diversas soluções de armazenamento para RSSFs. Cada uma com características específicas com relação à organização, coleta, propagação e cooperação distintas, conforme é possível observar na Tabela 3.3. O sistema *CAG* apresenta uma solução eficiente para consultas em ambientes nos quais existe uma grande similaridade de dados. Os sistemas *GHT* e Scoop apresentam modelos com índices descentralizados que permitem o equilíbrio de custos de transmissões entre armazenamento e consultas. Os dois últimos modelos, *HDMST* e *DACS*, apresentam soluções hierárquicas que utilizam cache de dados com funções de agregação.

Classificação					
Sistema	**Armazenamento**	**Organização**	**Coleta**	**Propagação**	**Cooperação**
CAG	Local	Hierárquico	Reativa	Sob Demanda	Correlação
GHT	Centrado em Dados	Plano	Periódica	Programada	Correlação
HDMST	Centrado em Dados	Hierárquico	Contínua	Sob Demanda	Correlação
DACS	Centrado em Dados	Hierárquico	Periódica	Sob Demanda	Correlação
SCOOP	Centrado em Dados	Plano	Periódica	Programada	Localizada

Tabela 3.3: Tabela comparativa de sistemas de armazenamento.

Os trabalhos apresentados contribuíram para um aprendizado sobre o impacto de diversas estratégias como: a agregação de dados para reduzir transmissões de dados, a utilização de mensagens contendo informação sumarizada sobre os dados locais e enviadas para sensores mais próximos da EB e a utilização da distância do sensor até a EB e ao sensor repositório para escolher melhores repositórios. Estes trabalhos influenciaram a definição do modelo de rede e de políticas que será apresentado no Capítulo 5. No entanto o sistema Scoop utiliza heurísticas que conferem ao sistema adaptabilidade. Essa foi a principal razão para a escolha deste sistema como base. Ao estudar o sistema observou-se que seria interessante introduzir algumas variações em parâmetros fixos do Scoop e testar se essas variações produziriam uma redução no número de transmissões. A implementação feita deste sistema é validada no Capítulo 4. Neste capítulo são apresentados os resultados através dos experimentos efetuados para testar essas variações.

Embora nesta seção o foco seja o armazenamento é possível inferir dos sistemas apresentados uma série de políticas. Muitas vezes de forma implícita, elas podem ser definidas de forma explícita através de um conjunto de regras de políticas. Nos trabalhos apresentados foi possível identificar alguns exemplos de políticas que estavam definidas de forma implícita como pode ser observado na tabela 3.4.

Todos esses exemplos demonstram a versatilidade e a importância das políticas para a eficiência de uma RSSF. As políticas podem ser utilizadas em qualquer camada da RSSF.

Políticas				
Sistema	Sujeito	Alvo	Condição	Ação
CAG	Sensor	Sensor	Leitura do sensor fora do *threshold* do grupo	Migrar para um grupo existente que esteja dentro do *threshold* ou formar um novo grupo
GHT	PRP	Perímetro doméstico	Falha no sensor responsável	Definir um novo responsável e replicar os dados
HDMST	Sensor	Sensor cache	Leitura atual superior ao erro definido	Enviar atualização para o sensor de cache
DACS	Sensor cache	Sensor	Leitura armazenada fora do desvio/erro máximo	Enviar nova leitura e atualizar a cache
SCOOP	EB	Sensor responsável	Aumento da frequência de consultas	Definir um novo responsável mais próximo à EB
DYSTO	EB	Sensores	Variação de leituras mínima	Não enviar nova mensagem de histograma

Tabela 3.4: Exemplo de políticas inferidas dos sistemas de armazenamento.

Esta dissertação tem como objetivo definir um sistema adaptável a diversos contextos nas quais as políticas são responsáveis pela adaptabilidade. Ao definir diferentes políticas para contextos distintos é possível adotar as melhores soluções para o momento atual do sistema. Os cenários de uma RSSF, assim como o mundo real, são um exemplo de contínuo desenvolvimento e mudança.

3.2 Políticas para RSSFs

Embora não tenham sido encontrados trabalhos específicos utilizando políticas para a gestão de armazenamento de dados, existem diversos trabalhos que utilizam políticas no contexto de RSSFs. Nesta seção são descritos alguns desses trabalhos com o intuito de mostrar que as políticas são importantes como um método de dotar os sistemas de adaptabilidade. Da mesma forma que na seção anterior, no final da seção é apresentado um pequeno resumo dos trabalhos expostos e as suas contribuições.

3.2.1 SRM

O artigo *Design and implementation of a policy-based management system for data reliability in Wireless Sensor Networks* [18] descreve o projeto e implementação de um sistema de gerenciamento denominado SRM para controlar a confiabilidade da entrega de mensagens numa RSSF. O SRM é baseado em uma arquitetura hierárquica de gerenciamento e em políticas para a gestão de rede. Ele é composto por quatro módulos: um módulo de especificação de política dos usuários, um módulo de avaliação, um módulo de decisão e um módulo de ação. O módulo de especificação de políticas é composto por uma série de regras que em conjunto definem uma política e por outras ferramentas relacionadas com a introdução, *parsing* e distribuição das políticas. As regras são definidas utilizando como base a linguagem Policy Framework Definition Language (PFDL) [29] com regras do tipo *IF*<*condição*> *THEN*<*ação*> *SCOPE*<*escopo*> expressa em formato *Extensible Markup Language* (XML), onde <*condição*> é uma forma normal conjuntiva de expressões de condição, <*ação*> é uma lista de declarações de ações individuais e <*scope*> é o nó sensor no qual a política é aplicada. Existem três níveis de escopo: *LOCAL* se a política for direcionada a um só sensor; *CLUSTER* se a política for direcionada ao Sensor Líder e *NETWORK* se for de âmbito global da rede. As políticas são descritas no formato XML e compõem a entrada para o módulo de decisão. Na Figura 3.3 é apresentado um exemplo de uma regra. A regra descreve que quando a energia disponível num sensor for inferior a um determinado *threshold* (E_TH) e a razão de entrega for superior ao *threshold* (DR_TH) então o sensor deve diminuir a frequência de transmissões. O módulo de avaliação é responsável por coletar as informações de gestão necessárias para avaliar a confiabilidade da rede. O módulo de decisão é a camada central do sistema e tem dois componentes: o componente de tomada de decisão e um componente de política de gestão. O componente de tomada de decisão decide quais ações devem ser enviadas para o módulo de ação com base na confiabilidade estimada da rede e nas políticas definidas. O componente de gestão de políticas é um repositório de políticas disponíveis que são distribuídas pelo módulo de especificação de políticas. O módulo de ação é responsável por executar a ação atribuída pelo módulo de decisão. Desta forma é possível resumir o

```
<?xml version="1.0" encoding="UTF-8"?>
<rule>
 <condition>
  <variable> ENERGY </variable>
  <operator> &lt;= </operator>
  <reference> E_TH </reference>
  <variable> DELIVERY_RATIO </variable>
  <operator> &gt </operator>
  <reference> DR_TH </reference>
 </condition>
 <action>
  <function> REDUCE_TRANSMISSION_RATE </function>
 </action>
 <scope scope="LOCAL"/>
</rule>
```

Figura 3.3: Exemplo de uma política no sistema SRM[18].

funcionamento do sistema da seguinte forma: as políticas são inseridas através do módulo de especificação de políticas em forma de um arquivo XML; elas são distribuídas para o módulo de decisão, que as armazena num banco de dados especifico; o módulo de decisão valida a aplicação de políticas utilizando para isso os dados coletados pelo módulo de avaliação e ao serem cumpridas as condições envia a ação a realizar para o módulo de ação. De acordo com os resultados relatados a utilização de políticas permitiu a redução de 50% no número de mensagens que necessitam ser retransmitidas dentro de uma RSSF.

3.2.2 PARAW

O artigo *Policy-based Adaptive Routing in Autonomous WSNs* [9] apresenta uma solução de alto nível e flexível para realizar tarefas de gerenciamento relacionadas com o roteamento em RSSFs utilizando políticas, denominada PARAW. As políticas podem ser definidas de forma progressiva com base no conhecimento adquirido do ambiente ou por mudanças nos requisitos da aplicação. Elas são utilizadas para estabelecer normas para ter ações dinâmicas na rede de acordo com o seu estado. Estudos de caso empregando uma política baseada em uma solução híbrida adaptativa permitem a seleção autônoma da melhor estratégia de roteamento em vista das condições de rede e de requisitos da aplicação. Sensores que executam as políticas devem conter um módulo processador de políticas. Este módulo fornece aos sensores da rede a capacidade de armazenar, interpretar e executar políticas específicas.

As políticas devem incluir funcionalidades para monitorar, analisar e agir em uma rede. A fase de acompanhamento utiliza dados e informações coletadas sobre o estado da rede para analisar e utilizar em ações futuras. O perfil (requisitos) de cada aplicativo em execução na rede também pode ser levado em consideração para tomar uma decisão de ação. A fase de análise pode utilizar técnicas de fusão de dados, agentes inteligentes, como a comparação com *thresholds*, técnicas de inferência, e outros, para melhor detectar situações em que uma ação é necessária *(3)*.

```
private counter;
counter=getNumNodes();
if counter > THRESHOLD then
 takeAction(PROACTIVE);
else
 takeAction(REACTIVE);
endif
```

Figura 3.4: Exemplo de uma política de roteamento[9].

Na Figura 3.4 é apresentado um exemplo de uma política utilizada no sistema que define a mudança entre um roteamento proativo ou reativo dependendo do número de sensores existentes na RSSF quando comparado com um valor *threshold* definido pelo usuário. Os resultados demonstram que a utilização de vários tipos de roteamento, dependendo das politicas definidas, leva a uma redução do número de transmissões do sistema e consequentemente do consumo energético. Este é um exemplo de políticas para RSSFs com o intuito de dotar o sistema de adaptabilidade.

3.2.3 PTSN

O artigo *Policy-driven tailoring of sensor networks* [22] apresenta um paradigma baseado em políticas que permite o ajuste fino e otimização do ambiente de tempo de execução por parte dos agentes do sistema. A integração das políticas em vários modelos de programação distintos é analisada. Esse artigo utiliza as políticas para refinar o comportamento do sistema. Várias atividades ocorrem entre a especificação de políticas pelo usuário do aplicativo e a sua aplicação real ou execução no dispositivo de destino. Políticas come-

çam o seu ciclo de vida sendo especificadas pelo usuário em alto nível e, posteriormente, submetidas a vários níveis de transformação e refinamento para acabar como código ou configurações específicas que podem ser implantadas e executadas diretamente nos sensores. O PARAW propõe o seguinte ciclo de vida de uma política:

1. Especificação de políticas por usuários (não-técnicos) utilizando ferramentas de alto nível ajudando-os a criar políticas sintática e semanticamente corretas. Essas políticas são, então, encaminhadas a um administrador juntamente com uma lista de aplicações e sensores de destino onde devem ser aplicadas.

2. A análise de políticas é uma atividade pesada, que deve ser executada na estação base, que possui mais recursos. A análise envolve a verificação de conflitos entre as políticas já aplicadas e verificação se a política é transparente para os outros usuários. Se a política é considerada correta, é marcada para a transformação e distribuição para os sensores.

3. Neste ponto a política se transforma em uma representação mais compacta e otimizada adequada para a divulgação de forma energeticamente eficiente dentro da RSSF.

4. Como as políticas podem controlar todos os tipos de funcionalidades implantadas dentro da rede, a distribuição segura de novas políticas é uma parte essencial do ciclo de vida.

5. A instalação e aplicação de políticas em um sensor acontece após a recepção e verificação da política. A representação compacta da política é interpretada e armazenada localmente em uma estrutura específica.

A Figura 3.5 mostra o exemplo de uma política definida por um usuário que tem como objetivo detectar uma possível inundação quando a pressão aumenta.

```
policy " Flood detection " {
on PRESSURE as p; // PRESSURE contains parameter value
if( p. value > 500 )
then (
allow p; // by default other PRESSURE events are blocked
)
}
```

Figura 3.5: Exemplo de uma política para detecção de inundação [22].

3.2.4 CaPI

No artigo *A Component and Policy-Based Approach for Efficient Sensor Network Reconfiguration (CaPI)* [21] é proposta uma plataforma chamada *Component and Policy Infrastructure (CaPI)*, que é configurável para a construção e gestão de aplicações de RSSFs. A CaPI oferece duas abstrações de programação: componentes e políticas. Os componentes são responsáveis pela reutilização e implementação de novos pedaços de código de baixo nível do sistema. Na análise apresentada nesta dissertação, o foco é nas políticas. As políticas são uma abstração leve, implementada de forma eficiente, numa linguagem independente de plataforma que é projetada para dar suporte à personalização e gestão de questões comportamentais durante o tempo de execução. Durante a execução do sistema é possível a introdução, remoção ou a composição de componentes e políticas. As políticas no CaPI são especificadas utilizando um modelo *Evento-Condição-Ação(ECA)*. Embora o paradigma *ECA* se ajuste à natureza baseada em eventos do CaPI, outros tipos de políticas baseadas em Condições-Ações podem ser definidas. A política exemplificada na Figura 3.6 é aplicada quando o evento *SENSOR_READING* for capturado pelo sistema.

Cada política CAPI é descrita através de um nome, uma descrição informal, e um identificador. Todas as políticas podem incluir a definição e uso de variáveis locais e globais, permitindo especificações mais expressivas e claras. As variáveis locais estão contidas dentro do contexto de execução local da política, enquanto que as variáveis globais correspondem ao estado de todo o sistema compartilhado, mantido pelo componente repositório do estado.

```
policy "EC−SensorData" "Encrypt readings" "SEC_ENCR_XTEA"{
 include security as sec; //security library
 uint8_t id = 0; //local variable

 on ENABLE { id <− sec::initialise("xtea",wsnkey.key); }
 on DISABLE { sec::clear(id); }

 on event SENSOR_READING as s;
 condition ( true )
 action (
  sec::encrypt(id,s);
 )
}
```

Figura 3.6: Exemplo de uma política de segurança de dados [21].

A estrutura de políticas adaptáveis em tempo real do sistema CaPI é dividida em três pontos:

1. *Pontos configuráveis de aplicação de políticas* - as políticas CaPI são aplicadas em um número bem definido de locais dentro do sistema. Como todas as interações entre aplicações distribuídas são efetuadas através de eventos, o canal de distribuição de eventos oferece um único ponto de interceptação, permitindo que as políticas CaPI possam regular todo o comportamento que ocorre em tempo de execução.

2. *Motor adaptável de políticas* - O mecanismo de políticas CaPI implementa uma pequena máquina virtual com um modelo de execução em pilha. Após a interceptação de um evento em um dos pontos de aplicação, o evento é introduzido na máquina virtual que, então, avalia e executa as políticas desencadeadas.

3. *Repositório de políticas* - Após a instalação da política em um sensor, o componente repositório de políticas calcula e aloca a memória necessária e armazena a política junto com alguns metadados. Isso inclui metadados com o status da política (ativado, desativado) e o conjunto de pontos de aplicação em que deve ser utilizada.

3.2.5 Discussão sobre os Sistemas baseados em Políticas

Os diversos sistemas apresentados utilizam políticas com o intuito de modificar uma parte ou todo o sistema de uma forma automática sem a necessidade de intervenção externa, codificação do sistema ou necessidade de parada. Todos os sistemas contribuíram de forma direta ou indireta para o modelo de políticas adotado nesta dissertação. O sistema SRM mostra que a utilização de políticas pode impactar significativamente o número de transmissões utilizando como base a linguagem Policy Framework Definition Language (PFDL) para descrever as políticas. Essa linguagem permite a definição de políticas de uma forma simples e estruturada. No sistema PARAW é possível observar o impacto das políticas para adaptar o roteamento de mensagens e a utilização de *thresholds* em algumas dessas políticas. O sistema PTSN demonstra a utilização de políticas para diversas tarefas no sistema e define um ciclo de vida para cada uma delas. O sistema CaPI apresenta uma solução completa e integrada de políticas e demonstra que é possível ter um sistema complexo de políticas numa RSSF.

3.3 Sumário

Nesta seção foram apresentados diversos sistemas que adotam políticas em RSSFs para a gestão do comportamento de diversos aspectos deste tipo de rede. Estes sistemas apresentam bons resultados na redução do número de transmissões, na adaptabilidade do sistema a novos cenários e na utilização de políticas em várias camadas de rede simultaneamente.

No estudo sobre os sistemas de armazenamento apresentados, observou-se que as políticas são adotadas de forma implícita, o que não possibilita a adequação às necessidades das aplicações. Como não foram encontrados na literatura trabalhos que adotam políticas para a gestão de armazenamento de RSSFs de forma explícita, nos próximos capítulos é apresentado e validado um modelo de armazenamento que utiliza políticas para se adaptar a novos cenários de forma automática.

Capítulo

4

DYSTO

Este capítulo apresenta e valida algumas ideias colhidas dos trabalhos anteriormente descritos. Para tanto, foi desenvolvido o modelo *DYSTO - A Dynamic Storage Model for Wireless Sensor Networks*. No DYSTO diversos parâmetros podem ser definidos pelo usuário, tendo como objetivo de que forma eles podem impactar o número total de transmissões de uma RSSF. O modelo foi desenvolvido na camada de aplicação e é independente do modelo de topologia, isto é, não está ligada a um protocolo de roteamento particular ou topologia da rede. No entanto, necessita de um protocolo de transporte que implemente roteamento horizontal, em que cada sensor mantém caminhos não só para os seus vizinhos diretos, ascendentes ou descendentes, mas também caminhos para sensores no mesmo nível da árvore de roteamento. Essa necessidade existe pois os sensores precisam saber a totalidade ou pelo menos parte do caminho para que possam encaminhar os dados sensoreados para os sensores responsáveis.

O modelo proposto é uma extensão do modelo proposto por Scoop, um sistema que considera a frequência das consultas, geração de dados e condições de rede para estabelecer as configurações de armazenamento e atualizá-los. O Scoop utiliza alguns parâmetros fixos tais como a frequência de mensagens de sumário, ou de envio de dados. O modelo DYSTO testa a variação desses parâmetros através de limites definidos pelo usuário. Além disso, o Scoop não leva em conta os co-ocorrências de dados em consultas e adota um

modelo simples e custoso para obter informações de monitoramento do sistema. Assim, as contribuições do DYSTO são:

- Extensão do modelo dinâmico proposto pelo Scoop, considerando as co-ocorrências de dados em consultas;
- Definição de um modelo de rede.
- Definição de uma função de decaimento para reduzir o número de retransmissões de mensagens de sumário.
- A adoção de limites definidos pelo usuário para determinar a frequência de transmissão de novas mensagens de dados e de sumário;
- Um estudo experimental, a partir de simulações, mostrando que as estratégias propostas podem reduzir significativamente o número de transmissões de mensagens no sistema.

Embora confiar no usuário para definir os limites do sistema possa acrescentar complexidade ao sistema e impactar sua usabilidade, estudos experimentais mostram que a capacidade de mudá-los reduz consideravelmente os gastos de energia em comparação com valores fixos. Neste capítulo é apresentado um estudo sobre os efeitos destes parâmetros sobre o sistema, e é um primeiro passo para um sistema de auto-ajuste que pode ser configurado para atender às necessidades específicas da aplicação.

Este capítulo está organizado da seguinte forma: na Seção 4.1 são apresentadas as asserções utilizadas para o desenvolvimento e teste do DYSTO; na Seção 4.2 é apresentado em detalhes o modelo de rede utilizado para este sistema. A Seção 4.4 descreve um conjunto de experimentos realizados com o intuito de testar o modelo DYSTO comparando-o com o modelo Scoop e apresenta as conclusões sobre a análise experimental.

4.1 Asserções

Foram utilizadas diversas asserções para proceder à modelagem e implementação da proposta desta dissertação. Foram elas:

- Todas as consultas são injetadas e posteriormente distribuídas pela rede através da EB.
- São coletadas somente informações numéricas discretas como temperatura, umidade e luminosidade.
- As consultas são somente consultas de valores, ou seja, para a obtenção dos sensores que possuem leituras dentro de um determinado intervalo de valores.
- Necessita de um protocolo de roteamento que implemente roteamento horizontal.

As asserções aplicadas no trabalho preliminar são utilizadas também na proposta do modelo baseado em políticas, apresentado no Capítulo 5.

4.2 Modelo de Rede

Os componentes do modelo de rede são apresentados na seguinte definição. É considerada a existência de uma sequência de tempo $T = [t_1, t_2, \ldots]$, onde $t_i < t_{i+1}$.

Definição 4.1. A RSSF é definida por uma 10-tupla $W = (EB, G, N, IA, f_{IA}, hR, hS, Th_d, Th_h, Py)$, na qual:

- EB é a estação base;
- $G = (S, A)$ é um grafo, onde S é o conjunto de sensores $\{s_1, \ldots, s_n\}$, tal que $a = (s_i, s_j) \in A$ se o sensor s_i estiver dentro do alcance de s_j. Diz-se que s_i está a um *salto* de distância de s_j e por isso s_i é um *vizinho* de s_j. Cada sensor s coleta novos dados no tempo t, e essa leitura é dada por $val(s, t)$.
- N é o número de entradas no índice de armazenamento. Ou seja, em qualquer tempo t seja $minV$ e $maxV$ os valores mínimo e máximo do conjunto $\{val(s, t) \mid s \in S\}$. O valor de N determina a partição de $[minV, maxV]$ em subintervalos do mesmo tamanho $I = \{iv_1 = (-\infty, v_1), iv_2 = [v_1, v_2), \ldots iv_N = [v_{N-1}, +\infty)\}$;

- $IA[t][iv]$ é o índice de armazenamento, que é uma matriz bidimensional em que cada célula contém o sensor responsável por uma leitura dentro de um intervalo $iv \in I$ no tempo t. Esse local pode ser um sensor de S ou a EB;
- f_{IA} é a frequência com que a estação base computa um novo IA.
- hR é o tamanho do *buffer* de dados de leituras recentes existente em cada sensor.
- hS é o número de partições do histograma de dados calculado em cada sensor.
- Th_d é o *threshold* que determina quando deve ser enviada uma nova mensagem de dados do sensor.
- Th_h é o *threshold* que determina quando deve ser enviada uma nova mensagem de sumário para a EB contendo um novo histograma.
- $Py[s]$ é o tempo de *piggybacking* que cada sensor s tem que esperar por mensagens de sumário de outros sensores descendentes antes de encaminhá-la para a EB. Como cada sensor envia periodicamente mensagens de sumário para a EB, se não for considerada qualquer forma de agregação de mensagens deste tipo, isso representaria um peso enorme para o sistema em termos de consumo de energia. Assim, cada sensor espera durante um certo período antes de transmitir as mensagens recebidas. Se receber novas mensagens durante este período, então estas são concatenadas em uma única mensagem e enviada para a EB, após o tempo expirar. A função que determina $Py[s]$ é dada por $Py[s] = maxPy \times e^{-\lambda \times (dist[BS][s]-1)}$ onde $maxPy$ é o tempo máximo de espera, λ é a constante de decaimento e $dist[BS][s]$ é o número de saltos entre a EB e o sensor s. Tanto $maxPy$ como λ são parametrizáveis.

□

Através do que foi anteriormente discutido e com este modelo de rede pode-se resumir o que foi testado com este sistema em duas perguntas:

- O que acontecerá se os diversos tipos de mensagens não forem enviados num período fixo e sim quando existir uma variação significativa nos dados?

- Se vários dados são consultados em conjunto será que o armazenamento no mesmo sensor poderá representar uma melhoria no número de transmissões?

Algumas das ideias utilizas neste modelo de rede foram posteriormente re-aproveitadas para o modelo baseado em políticas apresentado nesta dissertação. Para descrever com pormenor o funcionamento e o próprio modelo definido são apresentadas na próxima seção diversas etapas do sistema com exemplos específicos.

4.3 Etapas do Sistema

Nesta seção é apresentado o funcionamento do DYSTO de uma forma iterativa e através de exemplos. A ideia é mostrar todas as etapas desde a leitura dos sensores até à consulta e resposta por parte dos usuários.

4.3.1 Leitura de Dados de Sensoreamento

Cada sensor faz uma leitura do objeto sensoreado periodicamente (definido por parametrização). Essa leitura é então colocada num *buffer* que armazena as últimas hR leituras do sensor. Esse *buffer* implementa o método *round-robin*, ou seja a leitura mais recente substitui a leitura mais antiga. Na Tabela 4.1 é apresentado um exemplo do *buffer* com as últimas leituras de temperatura de um sensor. É sobre o valor médio dos valores armazenados no *buffer* que é validado o threshold Th_h de sumário. Mais especificamente, nos tempos t_1 e t_2 de envio de sumários são calculadas as médias m_1 e m_2 a partir dos valores armazenados no *buffer*. Caso $|1 - (m_2/m_1)| \geq Th_h$ então uma nova mensagem de sumário é enviada para a EB.

***Buffer* de Últimas Leituras**															
Valor	8.81	8.51	9.46	9.97	9.51	9.84	9.62	9.42	9.55	9.68	10.81	10.78	10.85	7.78	7.60
N°leitura	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Valor	7.25	8.98	7.09	9.34	8.73	7.30	9.17	7.15	6.97	6.79	9.03	7.61	8.09	8.76	8.31
N°leitura	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30

Tabela 4.1: *Buffer* de Últimas Leituras com $hR = 30$.

```
Prob(s -> v){
larguraBin = (max - min)/nBins
bin = (v - min)/larguraBin
Prob(v|bin) = 1/larguraBin
Prob(bin) = altura(bin)/(Nº Leituras))
return Prob(v|bin) * Prob(bin)
}
```

Figura 4.1: Cálculo da probabilidade de um sensor s produzir um valor v [11].

4.3.2 Geração e envio de mensagens de Sumário

As mensagens de sumário são geradas periodicamente e enviadas de cada sensor para a EB. Para minimizar a quantidade de informações a serem enviadas é criado um histograma contendo as informações necessárias. A mensagem contém dados, necessários à EB, para o cálculo do Índice de Armazenamento (IA). Os dados enviados permitem calcular a probabilidade de cada sensor s gerar um valor v dentro de um determinado intervalo i.

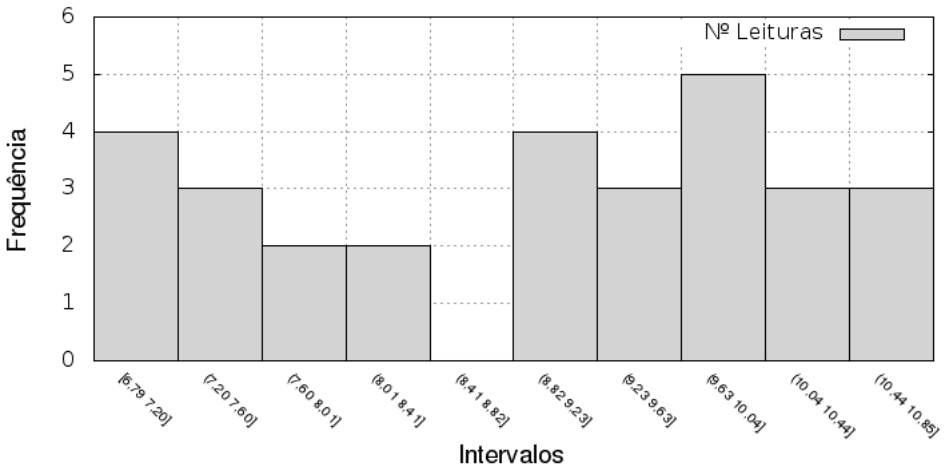


Figura 4.2: Histograma resultante dos dados da tabela 4.1.

Considerando os valores da Tabela 4.1, a Figura 4.2 representa um histograma com 10 classes($bins$), no qual $max = 10,85; min = 6,79; n^o leituras = hR = 30$ e $larguraBin = 0,406$. Cada sensor calcula e envia para a EB uma mensagem de sumário juntamente com cada respectivo histograma calculado. Além disso, é enviada a frequência de envio de mensagens de dados que é também utilizada no cálculo de $Prob(s \rightarrow v)$. Essa informação representa o número de mensagens de dados enviadas desde o recebimento do último IA. Ao receber as mensagens de sumário a EB armazena o histograma e as informações

complementares num banco de dados a ser utilizado no próximo cálculo do IA. A função utilizada na EB para o cálculo da $Prob(s \rightarrow v)$ é apresentada na Figura 4.1. É possível identificar a utilização dos dados enviados pelo sensor para calcular essa probabilidade. Primeiramente é calculado a largura de cada *bin*. Em seguida é identificado qual o *bin* em que se encontra o valor v. Se considerarmos por exemplo o histograma da Figura 4.2, o valor $7,7$ pertence ao 3^o *bin*. Após a obtenção do bin é calculada a probabilidade deste valor no contexto geral do histograma e calculados os valores finais da probabilidade considerando que há uma distribuição uniforme de valores dentro do intervalo.

4.3.3 Piggybacking de mensagens de Sumário

Como cada sensor envia periodicamente mensagens de sumário para a EB, se não considerarmos qualquer forma de agregação de mensagens deste tipo representaria um peso enorme para o sistema em termos de consumo de energia. Assim, cada sensor s tem que esperar por mensagens de sumário de outros sensores descendentes antes de encaminhá-la para a EB. Se receber novas mensagens durante este período, de forma que as mensagens de sumário de outros sensores, que devem ser reencaminhadas, são concatenadas em uma única mensagem e enviada para a EB, após o tempo expirar.

A função que determina $Py[s]$ é dada por $Py[s] = maxPy \times e^{-\lambda \times (dist[BS][s]-1)}$ onde $maxPy$ é o tempo máximo de espera, λ é a constante de decaimento e $dist[BS][s]$ é o número de saltos entre a EB e o sensor s. Tanto $maxPy$ como λ são parametrizáveis. Este tipo de função garante que a espera seja maior se o sensor estiver mais próximo da EB e praticamente sem espera nos sensores folha da rede. Como exemplo considere-se $maxPy = 15s$ e $\lambda = 0,7$. A Tabela 4.2 apresenta na coluna da esquerda o tempo de espera por novas mensagens e na coluna da direita a distância em número de saltos do sensor para a EB. Na Seção 4.4.5 é possível observar um experimento realizado com o intuito de mostrar a redução do número de mensagens que se obtém com esta estratégia.

Tempo Espera (s)	Distância p/EB
15,00	1
7,45	2
3,70	3
1,84	4
0,91	5
0,45	6
0,22	7
0,11	8
0,06	9
0,03	10

Tabela 4.2: Tempos de *Piggybacking* em cada nível de distância da EB.

4.3.4 Cálculo e Distribuição do IA

Da mesma forma que as mensagens de sumário, o cálculo do IA também é periódico. A sua periodicidade deve ser pelo menos duas vezes maior que a de geração de sumários para aumentar a probabilidade de ter informações de todos os sensores já que algumas mensagens podem se perder. O algoritmo do cálculo pode ser observado na Figura 4.3. Para explicar melhor o cálculo é necessário dividi-lo em várias etapas:

1. *Inicialização* - São criadas e limpas as matrizes de custos (linhas 1-2).

2. *Cálculo de limites* - A primeira etapa tem como objetivo definir o limite máximo e mínimo do IA. Para isso são consultadas todas as informações dos sensores, disponíveis na EB, para determinar a leitura mínima e máxima de toda a rede. Após isso é calculado uma largura uniforme para cada intervalo $larguraInt = (maxV - minV)/N$ e calculados todos os intervalos do IA com a mesma largura (histograma *equi-width*). Em um desenvolvimento alternativo, foram realizados experimentos considerando uma distribuição homogênea de sensores por intervalo, ou seja, uma distribuição *equi-depth*. Contudo os resultados obtidos não apresentaram melhorias no desempenho do sistema.

3. *Cálculo da matriz de custos de produção* - Nesta etapa é calculada uma matriz de dimensão $S \times N$ em que S é o número de sensores e N o número de intervalos do IA. O objetivo é calcular o custo de cada sensor produzir um valor dentro de cada intervalo do IA. Como os intervalos do histograma são diferentes dos definidos no IA é necessário estabelecer uma relação entre eles. Para isso é considerada uma

Algoritmo *CalculoIA*
Input: S: conjunto de sensores; t: tempo atual; I: conjunto de intervalos do IA; dist[][]; freqProd[]; probProd[][]; totQry[], corr[][]
Output: $IA[t]$: Índice de Armazenamento no tempo t

1. $custoProd[r][iv] \leftarrow 0$ para todos repositórios r e intervalos iv;
2. $custoQry[r][iv] \leftarrow 0$ para todos repositórios r e intervalos iv;
3. **for** all intervalos iv em I
4. **for** all repositórios candidatos r em $S \cup \{BS\}$
5. **for** all sensores s em S
6. $custoProd[r][iv] += probProd[s][iv] * freqProd[s] * dist[r][s]$;
7. $custoQry[r][iv] = totQry[iv] * 2 * dist[BS][r] * Prob(usuário\ consultar\ iv)$;
8. **for** all intervalos $iv \in I$
9. $minCost[iv] \leftarrow min(\{custoProd[r][iv] + custoQry[r][iv] \mid r \in (S \cup \{BS\})\})$;
10. $minRep[iv] \leftarrow$ repositorio r tal que $(custoProd[r][iv] + custoQry[r][iv]) == minCost[iv]$;
11. $IA[t][iv] \leftarrow null$;
12. $Q = [[iv_1, iv_2], \ldots] \leftarrow$ lista de pares de intervalos em ordem descendente de valores para $corr[iv_1][iv_2]$;
13. **for** all $[iv_1, iv_2]$ em Q
14. **if** $IA[t][iv_1] \neq null$ e $IA[t][iv_2] \neq null$
15. **then** continue;
16. $totComb \leftarrow totQry[iv_1] + totQry[iv_2] - corr[iv_1][iv_2]$;
17. **if** $IA[t][v_1] == null$ e $IA[t][v_2] == null$
18. **then** $minCombCost \leftarrow min(\{(totComb * 2 * dist[BS][r]) + custoProd[r][iv_1] + custoProd[r][iv_2] \mid r \in (S \cup \{BS\})\})$;
19. $minRepComb \leftarrow$ repositorio with $minCombCost$;
20. **else if** $IA[t][iv_1] \neq null$
21. **then** $minRepComb \leftarrow IA[t][iv_1]$;
22. **else** $minRepComb \leftarrow IA[t][iv_2]$;
23. $minCombCost \leftarrow (totComb * 2 * dist[BS][minRepComp]) + custoProd[minRepComb][iv_1] + custoProd[minRepComb][iv_2]$;
24. $costSep \leftarrow minCost[iv_1] + minCost[iv_2]$;
25. **if** $minCombCost < costSep$
26. **then** $IA[t][iv_1] \leftarrow minRepComb$;
27. $IA[t][iv_2] \leftarrow minRepComb$;
28. **for** all intervalos $iv \in I$
29. **if** $IA[t][iv] == null$
30. **then** $IA[t][iv] \leftarrow minRep[iv]$;

Figura 4.3: Algoritmo para calcular o IA.

distribuição uniforme das partições do histograma. Para exemplificar, considere que uma partição do IA é $[2-3)$ e as partições do histograma de um sensor s são $[1,5-2,5)$ e de $[2,5-3,5)$ então a probabilidade de s produzir uma leitura no intervalo $[2-3)$ seria dada por $P(s \rightarrow [2-3)) = 0,5 \times P(s \rightarrow [1,5-2,5)) + 0,5 \times P(s \rightarrow [2,5-3,5))$. Após obter as probabilidades é necessário multiplicar esses valores pela distância do sensor ao sensor candidato a repositório e pela frequência com que os dados são enviados (informação também enviada na mensagem de sumário). Dessa

forma, é calculada a matriz representando o custo potencial de produção para cada sensor candidato a armazenar determinado intervalo do IA (linhas 3-6).

4. *Cálculo de matriz de custos de consulta* - A terceira etapa do cálculo consiste no cálculo de uma matriz de custos de consulta e resposta. O objetivo é quantificar o custo de transmissões de consulta e resposta para cada sensor e intervalo do IA. Assim como a matriz de custos de produção a matriz tem uma dimensão $S \times N$, na qual cada linha representa o custo de cada sensor candidato a repositório e cada coluna um intervalo do IA. Para obter esse custo são multiplicados os seguintes valores: duas vezes a distância do sensor candidato à EB (representado o número de saltos pra enviar a consulta e obter a resposta), pela frequência de consultas desde o ultimo cálculo do IA e pela probabilidade do usuário consultar o intervalo em questão (linhas 3,4,7).

5. *Escolha dos repositórios* - A escolha dos repositórios é feita pelo valor mínimo dos custos totais, resultado da soma dos custos de produção e de consulta, dos sensores para cada intervalo (linhas 8-11).

6. *Cálculo da correlação* - Esta etapa começa por criar uma lista de pares de valores consultados em conjunto ordenada pelo número de consultas de cada par. Em seguida, para cada um desses pares é comparado o custo de armazenar os dois intervalos juntos no mesmo repositório com o custo de manter ambos em repositórios distintos. Caso o custo de armazenar em conjunto seja menor que separado é alterado o IA para que ambos os intervalos sejam armazenados no mesmo repositório (linhas 12-30).

Após o cálculo do IA este é distribuído para todos os sensores. Essas mensagens são chamadas de mensagens de mapeamento. Ao receber um novo IA os sensores passam automaticamente a utilizá-lo para distribuir os seus dados gerados.

4.3.5 Mensagens de Dados

Os sensores enviam mensagens de dados, agrupando os últimos cinco valores produzidos, para os repositórios correspondentes. A frequência de envio é determinada pelo Th_d. Ou seja, quando houver uma variação em comparação com o valor médio enviado anteriormente maior que Th_d, o sensor envia mensagens aos repositórios apropriados de acordo com o IA mais recente.

4.3.6 Mensagens de Consultas

As mensagens de consultas são introduzidas pelos usuários através da EB. O sistema interpreta consultas de valores que sigam um formato similar ao apresentado na Figura 4.4. Como a EB armazena todos os IA previamente calculados a consulta é particionada e enviada diretamente para os sensores repositórios responsáveis. Após recebida a mensagem de consulta e encontrados os dados requisitados esta informação é enviada para a EB em forma de uma mensagem de resposta, caso não sejam encontrados os dados é enviada mesmo assim uma mensagem de resposta.

SELECT $atrib_1, \ldots, atrib_n$
FROM sensores
WHERE $pred_1, \ldots, pred_n$

Figura 4.4: Consulta tipo.

4.4 Experimentos

DYSTO foi implementado na versão 2.34 do simulador de rede NS2[1]. O objetivo do modelo proposto é a redução do consumo total de energia. Dado que o consumo de energia é dominado pelo custo de comunicação [24], nossa métrica de custo é o número total de transmissões enviadas por todos os sensores.

[1] http://www.isi.edu/nsnam/ns/

DYSTO foi inspirada pela estratégia de posicionamento de repositórios proposta pelo sistema Scoop, e portanto, o objetivo de nosso estudo experimental é comparar o DYSTO com o Scoop usando o número de transmissões como métrica de custo para todos os experimentos. Observe que em ambos os modelos, há 4 tipos de mensagens que são transmitidas entre os dispositivos de rede: *mensagens de mapeamento*, contendo um novo IA, que são enviadas a partir da EB para todos os sensores no campo; *mensagens de sumário*, contendo um histograma de informações de dados e informações relacionadas, que são enviadas de cada sensor para a EB; *mensagens de dados*, que contêm dados de sensoriamento, enviados a partir de um sensor produtor para um repositório, e *mensagens consulta/resposta*, enviadas da EB para os repositórios em forma de consulta e voltando com a resposta da consulta para a EB. Em Scoop foi analisado o comportamento do sistema para diversas distribuições de dados, assim em DYSTO focaremos somente num cenário de dados reais e de dados sintéticos com similaridade espacial.

Simulações iniciais demonstraram que o número de mensagens de mapeamento permaneceu constante para ambos os modelos em todos os ambientes, dependendo apenas do número de cálculos do IA. Observe que a frequência de cálculos do IA determina a rapidez com que o sistema reage a mudanças no comportamento monitorado. Diante disso, os experimentos foram realizados para determinar o impacto de cada uma das estratégias propostas pelo DYSTO sobre os demais tipos de mensagens. O primeiro experimento observa o impacto sobre o número de mensagens de dados e de consulta/resposta considerando co-ocorrências em consultas. A segunda foi realizada para mostrar como o *threshold* de dados (Th_d), que determina a frequência de transmissões de atualizações de dados de sensoriamento, afeta o volume de mensagens de dados em comparação com transmissões num intervalo fixo de tempo. A terceira experiência demonstra o efeito do *threshold* para o envio de novos histogramas (Th_h) sobre transmissões de mensagens de sumário, de dados e de consulta/resposta. A quarta experiência mostra o impacto do uso da função de *piggybacking* na redução da mensagem de resumo. A última experiência analisa o impacto das quatro estratégias anteriores consideradas em conjunto com o número total das transmissões de cada tipo de mensagem.

4.4.1 Configurações de Simulação

Foi avaliado o desempenho do DYSTO usando uma base de dados real disponibilizadas pelo Laboratório de dados da Intel[2]. Os dados foram coletados por 54 sensores Mica2 implantados no Laboratório de Pesquisa da Intel durante um período de 35 dias. A localização dos sensores pode ser observada na Figura 4.5.

O conjunto de dados compreende um *timestamp* e informações sobre temperatura, umidade, luz e tensão elétrica. No entanto, apenas a temperatura juntamente com a informação da localização do sensor foram usadas nas simulações. Esta configuração é chamada de *cenário real*.

Para avaliar o modelo numa rede RSSF maior, foi também gerado um *cenário sintético* composto de 500 sensores em uma área de 500×500 metros, com características semelhantes aos dados de sensores utilizados no cenário real. Com base na observação de que as métricas recolhidas no cenário real apresentam leituras altamente correlacionadas espacialmente, foram geradas leituras dos sensores iniciais utilizando a ferramenta Matlab [17]. A ferramenta recebe como entrada um coeficiente de correlação (h) e o tamanho do monitoramento da área (m). Ela gera como saída uma matriz D de dimensão $m \times m$ e as leituras dos sensores são geradas da seguinte maneira: cada sensor s, colocado aleatoriamente em uma posição (x_s, y_s), recebe como leitura o valor em $D[\lfloor x_s \rfloor], \lfloor y_s \rfloor]$. O coeficiente de correlação determina o grau de similaridade. Isto é, se $h = 0$ são gerados dados sem similaridade espacial, e valores maiores de h produzem uma matriz com maior correlação espacial. Todas as simulações descritas nesta seção foram executadas em um cenário gerado com alta correlação ($h = 9$). Cada um dos valores calculados inicialmente foram então atualizados aplicando uma variação aleatória de zero a 30%, aplicada sobre o valor inicial, com base na localização do sensor, a fim de preservar a similaridade espacial dos dados.

Os parâmetros de simulação são apresentados na Tabela 4.3. Os valores dos *thresholds* não estão especificados na tabela uma vez que variam em cada experimento detalhado nas subseções seguintes. Todos os resultados reportados consideram estes valores como

[2] http://db.csail.mit.edu/labdata/labdata.html

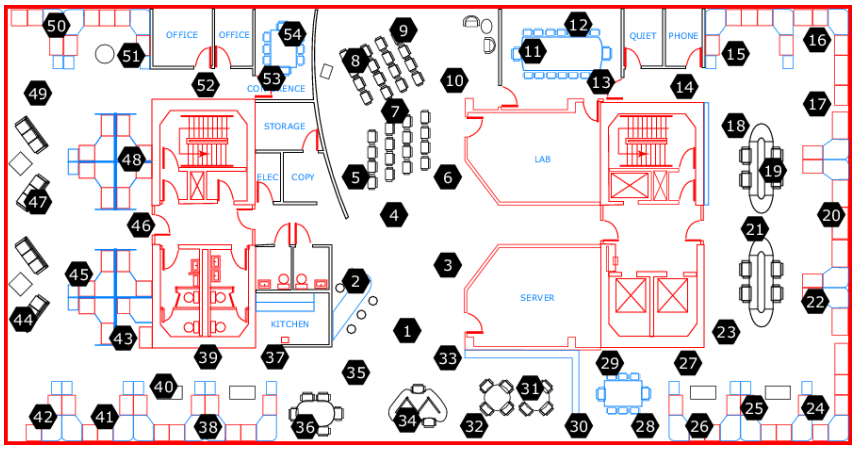


Figura 4.5: Localização dos sensores no laboratório *Intel Berkeley Research Lab*

padrão tanto para o Scoop como para o DYSTO, a menos que especificado de outra forma. Foram utilizados parâmetros similares aos indicados no Scoop para facilitar a validação do sistema. Os resultados dos experimentos consistem de um valor médio recolhido de cinco execuções em cada configuração de simulação.

Tabela 4.3: Parâmetros de simulação

Parâmetro	cenário real	cenário sintético
Número sensores	54 sensores + 1 EB	500 sensores + 1 EB
Fonte de dados	dados reais	dados sintéticos
Alcance da comunicação de rádio	30 metros	
Duração da simulação	40 minutos	
Frequência de amostragem	1 leitura do sensor a cada 15 segundos	
Frequência de consulta	1 consulta a cada 15 segundos	
Frequência de sumário	1 sumário a cada 110 segundos	
Cálculo do IA	1 cálculo a cada 240 segundos	
Número de partições num IA (N)	15	
Número de leituras de um sensor (hR)	30	
Número de entradas no histograma (hS)	10	
Tempo máximo de *piggybacking* ($maxP$)	15 segundos	
Constante de decaimento do *piggybacking* (λ)	0.7	

4.4.2 Experimento 1: Co-ocorrências

O objetivo deste experimento é determinar o efeito de considerar co-ocorrências dos dados de consulta para determinar localidade do repositório. Comparou-se o DYSTO com o Scoop, que segue a mesma abordagem que o DYSTO para escolher repositórios sem considerar esta estratégia. As simulações foram executadas com os mesmos parâmetros

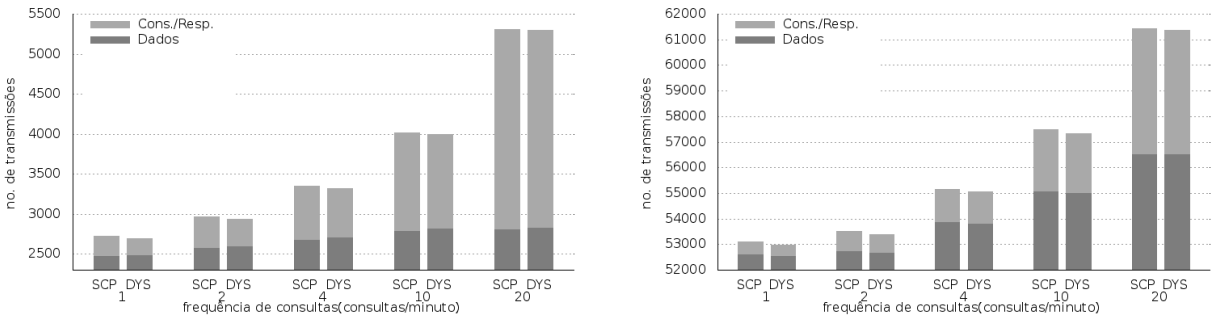
de entrada para ambos os sistemas, conforme apresentado na Tabela 4.3. Foi considerada uma carga de trabalho de consulta em que o valor inicial da faixa de consulta é escolhido aleatoriamente e 80% das consultas são correlacionadas. Ou seja, 80% das consultas de intervalo necessitam o acesso a mais do que um repositório. A Figura 4.6 mostra o número de transmissões para mensagens de dados e de consulta/resposta para as frequências de consulta de 1, 2, 4, 10 e 20 consultas por minuto. O número de transmissões para outros tipos de mensagens não são mostrados porque eles não são afetados pela estratégia de co-ocorrências de consulta.

Os gráficos mostram que, tanto no Scoop como no DYSTO o número de transmissões para mensagens de consulta/resposta aumenta com frequências mais elevadas de consulta, como esperado. Quando aumenta a frequência de consultas o crescimento nas transmissões de mensagens de dados deve-se ao deslocamento dos repositórios para mais perto da EB (e logo mais distante dos sensores produtores). No entanto, essa alteração no posicionamento do repositório tem um impacto maior sobre o número de mensagens de dados no cenário sintético. Isso pode ser explicado da seguinte forma: ao mover um repositório apenas um salto para longe dos produtores pode incorrer num aumento de 500 transmissões, isto é, um para cada produtor do sensor. No cenário real, por outro lado, o impacto é no máximo 54, que é o tamanho da RSSF.

Das simulações do cenário real apresentado na Figura4.6(a), pode-se observar que os resultados da estratégia de co-ocorrências no DYSTO conseguiram uma redução de transmissões de consultas/respostas de 18.75%, 13.28%, 8.18% 3.84% and 0,96% quando comparados com o Scoop, respectivamente para frequências de 1, 2, 4, 10 e 20 consultas por minuto. Para o cenário sintético (Figura 4.6(b)), o número de transmissões de mensagens de dados é praticamente idêntico em ambos os sistemas, e para transmissões de mensagens de consulta/resposta pode-se observar uma diminuição de 15.56%, 9.32%, 6.2%, 3.6% para frequências de consulta de 1, 2, 4 e 10, respectivamente. Com 20 consultas por minuto, os dois sistemas apresentam o mesmo comportamento. A estabilidade do número de consultas de dados neste cenário, em oposição ao seu aumento no cenário real, pode ser explicada da seguinte forma. Em RSSFs maiores podem existir vários sensores

com dados cujos custos estão perto do valor mínimo. Assim, o valor de co-ocorrências de consultas ajuda a seleção do repositório sem afetar o número de mensagens de dados. Em RSSFs menores, por outro lado, tal sensor alternativa pode não existir.

Observe que em ambos os cenários, com mais consultas sendo emitidas para o sistema, o ganho de transmissões de consulta/resposta no DYSTO diminui. Isto acontece porque para a escolha de um repositório, tanto o custo de envio de atualizações de leitura (custo de produção) e o custo da consulta são levados em consideração. Como ambos os gráficos mostram na Figura 4.6, o impacto do custo de produção aumenta com a taxa de consulta. Assim, para maiores taxas de consulta fica mais difícil de encontrar repositórios que quando se fundirem, com base no valor da consulta co-ocorrências, produzam reduções do custo de consulta que superem o aumento do custo de produção. Além disso, o impacto do custo de produção é maior para RSSFs maiores uma vez que cada sensor transmite individualmente e periodicamente uma mensagem de dados para atualizar o seu repositório. Isso explica por que, para o cenário real a estratégia de co-ocorrência apresenta melhores resultados do que para o cenário sintético. Conclui-se que a estratégia de co-ocorrência pode ser considerado um refinamento para o problema de seleção repositório que fornece melhores resultados do que o Scoop para taxas de consulta até 10 consultas / minuto e resultados semelhantes para taxas mais elevadas.



(a) Cenário real (b) Cenário sintético

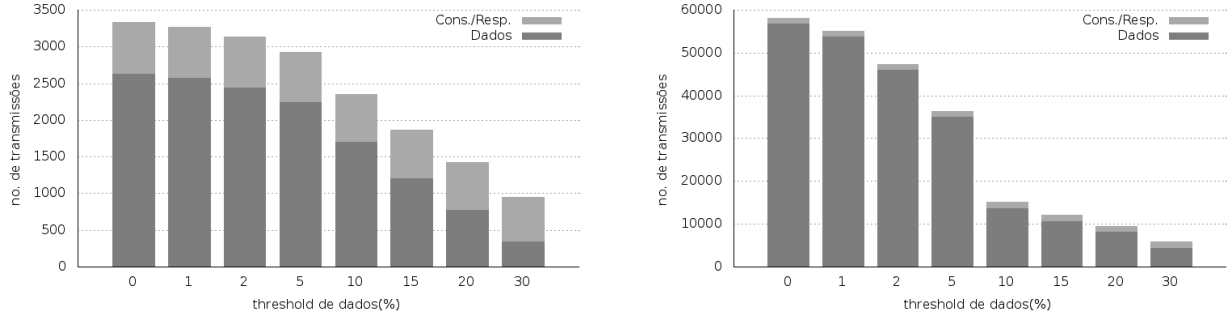
Figura 4.6: Simulação de co-ocorrências

4.4.3 Experimento 2: *Threshold* de Dados Th_d

Entre os quatro tipos de mensagens, as mensagens de dados são as que têm maior impacto no consumo total de energia, como ilustrado na Figura 4.10. Assim, foram conduzidas

simulações para avaliar o impacto da utilização de um *threshold* de dados Th_d para determinar a frequência com que um sensor atualiza as suas leituras no repositório responsável, em oposição a intervalos fixos entre as transmissões de mensagens de dados. Os resultados são apresentados na Figura 4.7, em que os valores relatados para um *threshold* de dados zero corresponde à configuração padrão do Scoop de uma mensagem a cada 75 segundos.

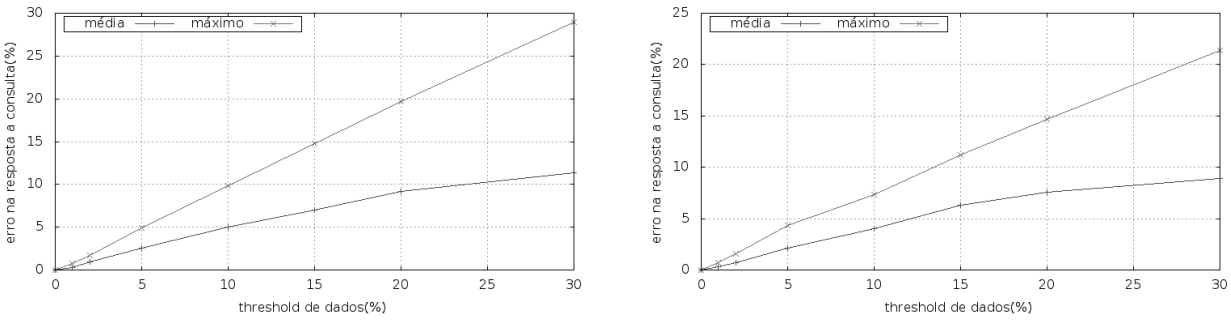
Observa-se que, comparado com intervalos fixos, um *threshold* de dados de 1% diminui o número de transmissões por cerca de 2% para o cenário real e 5% para o cenário sintético, e há uma diminuição constante sobre estes números com valores mais elevados para Th_d. Por exemplo, com $Th_d = 5\%$ eles diminuem em 15% e 38% para os cenários reais e sintéticos, respectivamente. Além disso, há uma pequena redução no número de transmissões de consulta/resposta. Isto acontece porque a redução na transmissão de dados, também reduz a taxa de produção (*prodRate*) do sensor monitorado pela EB. Como consequência, o peso do custo de produção é reduzida, em comparação com o custo da consulta sobre o processo de seleção do repositório. Assim, os repositórios atribuídos a sensores mais perto da EB, reduzindo o número de transmissões para consulta/resposta. Observe-se que, à semelhança da experiência descrita na secção 4.4.2, o número de mensagens de dados no cenário sintético é proporcionalmente muito maior do que as mensagens de consulta/resposta comparando com o cenário real. Isto é uma consequência da dimensão da RSSF, isto é, cada sensor envia uma mensagem de dados com as suas alterações de leituras e, portanto, aumenta o número de mensagens de dados com o tamanho da RSSF. No entanto, em ambos os cenários, manteve-se a mesma frequência de quatro consultas/segundo. Assim, a razão de mensagens de consulta/resposta no número total de transmissões é muito menor no cenário sintético.



(a) Cenário real (b) Cenário sintético

Figura 4.7: Simulação de *threshold* de dados

Ajustar a frequência de transmissão de mensagens de dados com base num *threshold* apresenta um impacto positivo ao reduzir o número de transmissões de dados. No entanto, esta estratégia pode ter um impacto sobre a precisão dos resultados da consulta. Uma vez que em uma simulação podemos constatar o conjunto exato de resultados de consulta, na Figura 4.8 são apresentados os erros médios e máximos relativos, através da comparação do resultado da consulta, com base nos valores armazenados no repositório, com a leituras reais armazenadas em cada sensor. Estes resultados mostram que a média de erro sobre os resultados da consulta não é maior do que a metade do Th_d para ambos os casos, mas pode ser tão elevado quanto o Th_d. O erro relativo para o cenário sintético é um pouco menor do que para o cenário real porque no cenário real podem existir variações bruscas nas leituras, enquanto que no cenário sintético foi aplicada uma atualização de, no máximo, 30% sobre o valor inicial. Assim, em geral, a diferença entre as duas leituras consecutivas não deverá diferir em mais de 20%. O equilíbrio entre o custo de transmissão de dados e da precisão do resultado da consulta deve ser considerado para cada aplicação.



(a) Cenário real (b) Cenário sintético

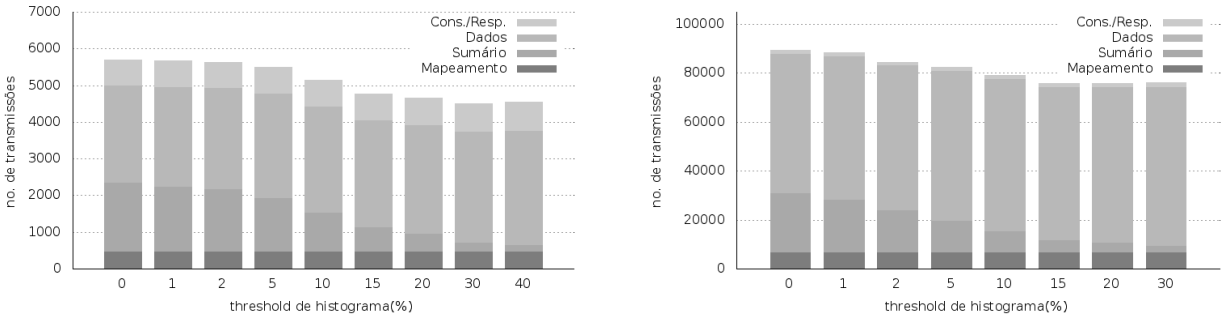
Figura 4.8: Erro relativo de dados no repositório devido ao *threshold* de dados

4.4.4 Experimento 3: *Threshold* de Histograma Th_h

Semelhante à experiência anterior, foram também conduzidas simulações para comparar o efeito de uma abordagem baseada em um *threshold* para determinar a frequência de transmissão de mensagens de sumário, em comparação com intervalos fixos. Vale notar que a quantidade de mensagens de sumário, em comparação com as mensagens de dados é menor por causa da estratégia de *piggybacking* para fundí-las em seu caminho para a EB.

Os resultados da simulação com variação de valores para o *threshold* de histogramas (Th_h) é apresentado na Figura 4.9. O gráfico representa o número de transmissões com um intervalo fixo de 110 segundos (como considerado pelo Scoop), que é considerado como $Th_h = 0$, com transmissões baseadas em valores incrementais de Th_h. Com $Th_h = 1\%$ o número de transmissões de mensagens de sumário é reduzido em 6.65% para o cenário real e 12% para o cenário sintético, quando comparados com o intervalo fixo de envio. A redução é maior para valores mais elevados de Th_h. Por exemplo, com $Th_h = 20\%$, a quantidade reduz 73,5% no cenário real e 83% no cenário sintético. Embora a redução no número de mensagens de sumário é considerável, a mesma redução não se estende para o número total de transmissões. Isto porque mudanças na frequência de mensagens de sumário também tem um impacto sobre as mensagens de dados e de consulta/resposta. Com menos informação proveniente dos sumários, a EB calcula novos IAs com base em dados mais antigos. Assim, atualizações sobre a colocação do repositório podem ser diferentes daquelas com base em informações mais recentes. De fato, valores mais elevados de Th_h aumentam tanto o número de transmissões de mensagens de dados como de consulta/resposta, mostrando que os repositórios são colocados mais afastados da sua localização ideal. Como é mostrado na Figura 4.9, com $Th_h = 30\%$, a redução no número total de transmissões é 21% para o cenário real, mas a redução é somente 20% para $Th_h = 40\%$. Isto mostra que quando $Th_h = 40\%$ o aumento no número de mensagens de dados devido a IAs desatualizados suplanta o ganho obtido na redução de número de mensagens de sumário. O mesmo fenômeno pode ser observado no cenário sintético: a redução geral é de 15% para $Th_h = 20\%$, e 14% para $Th_h = 30\%$. O valor máximo do *threshold* de histograma em que é benéfico para a redução do número total de transmissões é menor para RSSFs maiores porque o volume das mensagens de dados aumenta mais rapidamente para RSSFs maiores.

Observe-se que, ao contrário do *threshold* de dados Th_d, a adoção de um *threshold* de histograma não tem impacto na precisão das respostas a consultas. O impacto principal da estratégia é adiar atualizações dos repositório no IA, mas os repositórios contêm as leituras atuais relatadas pelos dispositivos sensores.



(a) Cenário real (b) Cenário sintético

Figura 4.9: Simulação do *threshold* de histograma

4.4.5 Experimento 4: *Piggybacking*

Esta experiência tem por objectivo avaliar o efeito de várias configurações de *piggybacking* e seu impacto sobre o número global de transmissões de sumário. Como cada sensor envia mensagens de sumário para a EB, se não for considerada qualquer forma de agregação de dados, mensagens deste tipo representam um peso enorme sobre o sistema em termos de consumo de energia. Assim, foi adotada uma abordagem *piggybacking* que funciona da seguinte maneira. Após o cálculo de um novo sumário, cada sensor espera durante um certo período de tempo antes de transmiti-lo. Se durante este tempo receber mensagens de sumário para serem encaminhadas para o seu ascendente através de um caminho para a EB, estas são concatenadas em uma única mensagem, a qual é enviada para a EB, após o tempo de espera expirar. Os resultados da simulação no cenário real, com vários valores de λ, estão presentes na Tabela 4.4. A primeira coluna contém o valor de λ, a segunda contém o número de transmissões de mensagens de sumário durante toda a simulação, e na terceira coluna, o tempo médio necessário para uma mensagem de sumário percorrer o caminho desde o sensor mais afastado s_f até à EB. Para $\lambda = 0.1$, o número de transmissões é reduzido em 16.54% quando comparado com a não utilização de *piggybacking*. O número de transmissões decresce gradualmente até $\lambda = 0.8$, que representa uma redução de 32.78%. Embora o número de transmissões diminua, o tempo médio para rotear uma mensagem do sensor s_f para a EB aumenta 76.64% para $\lambda = 0.1$ até 1142% para $\lambda = 0.9$. Nos experimentos reportados nas seções anteriores foi utilizado um valor de $\lambda = 0.7$ pois representa um boa relação custo/benefício e porque o tempo extra necessário no roteamento de mensagens não afeta a precisão dos resultados a consultas.

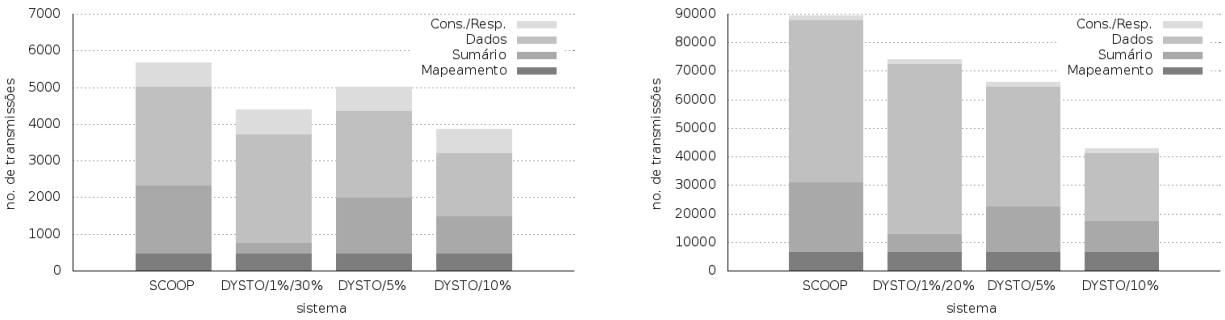
Tabela 4.4: O efeito de *piggybacking*

λ	# Transmissões Sumário	Tempo médio de transmissões(s)
Sem *piggybacking*	2666	15.20
0.1	2225	26.85
0.2	2194	28.81
0.3	2110	31.05
0.4	2009	34.09
0.5	1992	34.88
0.6	1860	40.34
0.7	1816	42.78
0.8	1792	60.23
0.9	1849	188.89

4.4.6 Experimento 5: Geral

A Figura 4.10 apresenta a comparação final entre o Scoop e o DYSTO, considerando a utilização simultânea das estratégias de consultas com co-ocorrências, *threshold* de dados e *threshold* de sumários. O Scoop foi implementado da mesma forma que em experimentos anteriores enquanto o DYSTO teve as seguintes parametrizações: *a*) *threshold* de dados de 1% e *threshold* de histograma de 30% para o cenário real e 20% para o cenário sintético; *b*) ambos os *thresholds* em 5%; e por fim *c*) com ambos os *thresholds* em 10%. A primeira configuração foi escolhida combinando um valor de Th_d que produz resultados de consultas com um erro máximo relativo de 1% e 0,5% em média, com um valor de Th_h que produz a máxima redução no número total de transmissões, com base nos experimentos reportados na Seção 4.4.4.

Comparado com o Scoop, e ainda na Figura 4.10, as reduções totais no número de transmissões do DYSTO para o cenário real são 22.72%, 11.74% e 32.15%, e para o cenário sintético, 17.02%, 26.01% e 52.15% para as configurações a, b e c respetivamente. Estes resultados mostram os benefícios potenciais do DYSTO. Também mostram que mesmo aplicações que necessitem de alta precisão de resultados de consultas podem se beneficiar da utilização de *thresholds*. Embora estes resultados sejam baseados em simulações, eles permitem uma boa previsão do desempenho do sistema em situações reais, como foi reportado em trabalhos anteriores [12].



(a) Cenário real

(b) Cenário sintético

Figura 4.10: Comparação entre o Scoop e o DYSTO

4.5 Conclusão

Neste capítulo, foi proposto o DYSTO, um modelo de armazenamento para RSSFs que coleta informações sobre a produção de dados e taxas de consulta, a fim de alterar dinamicamente o armazenamento de dados que é mais adequado para o contexto atual. O armazenamento de dados pode adotar um modelo local, externo ou centrado em dados dependendo do comportamento do sistema. Foram executados experimentos, com base em simulações, tanto em um conjunto de dados reais e em uma RSSF maior com dados gerados sinteticamente. No cenário real, DYSTO apresentou uma redução nos custos globais de transmissão de 12% e 32% para os *thresholds* estabelecidos para 5% e 10%, respectivamente, em comparação com o modelo SCOOP. Para o cenário sintético, ele reduz o número de transmissões de 26% e 52%, com as mesmas definições de *thresholds*. Estes resultados mostram os benefícios potenciais das estratégias propostas e é com base nestes resultados que avançamos para o sistema descrito no próximo Capítulo 5.

Capítulo

5

DYSTOPOL

Neste capítulo é apresentada a proposta de gestão de políticas para armazenamento de dados em RSSFs denominada DYSTOPOL. Após a validação do DYSTO decidiu-se avançar para um modelo de armazenamento baseado em políticas.

Este Capítulo está dividido em 5 seções. A Seção 5.1 apresenta o modelo de rede utilizado na definição da proposta. A Seção 5.2 demonstra como devem ser escritas as políticas e alguns exemplos de aplicação. A Seção 5.3 apresenta a arquitetura do modelo de políticas e o sistema de gestão. A Seção 5.4 apresenta os resultados obtidos através de simulações. Por fim a Seção 5.5 apresenta as conclusões do capítulo.

5.1 Modelo de Rede

O modelo de rede é similar ao modelo apresentado anteriormente no Capítulo 4. No entanto o modelo tem mais um componente:

- P é um conjunto de políticas $\{p_1, \ldots, p_n\}$ em que cada p representa uma regra na forma de (E, D, A, C) em que E representa o sujeito da política, D o sensor alvo, A a ação a realizar e C as condições para aplicação da regra. Os conjuntos de políticas existem no nível local de sensores e no nível global na EB.

O modelo continua tendo as mesmas funcionalidades e tipos de mensagens do modelo apresentado anteriormente com exceção das particularidades da implementação de políticas que serão descritas nas próximas seções em maior detalhe.

5.2 Definição de Políticas

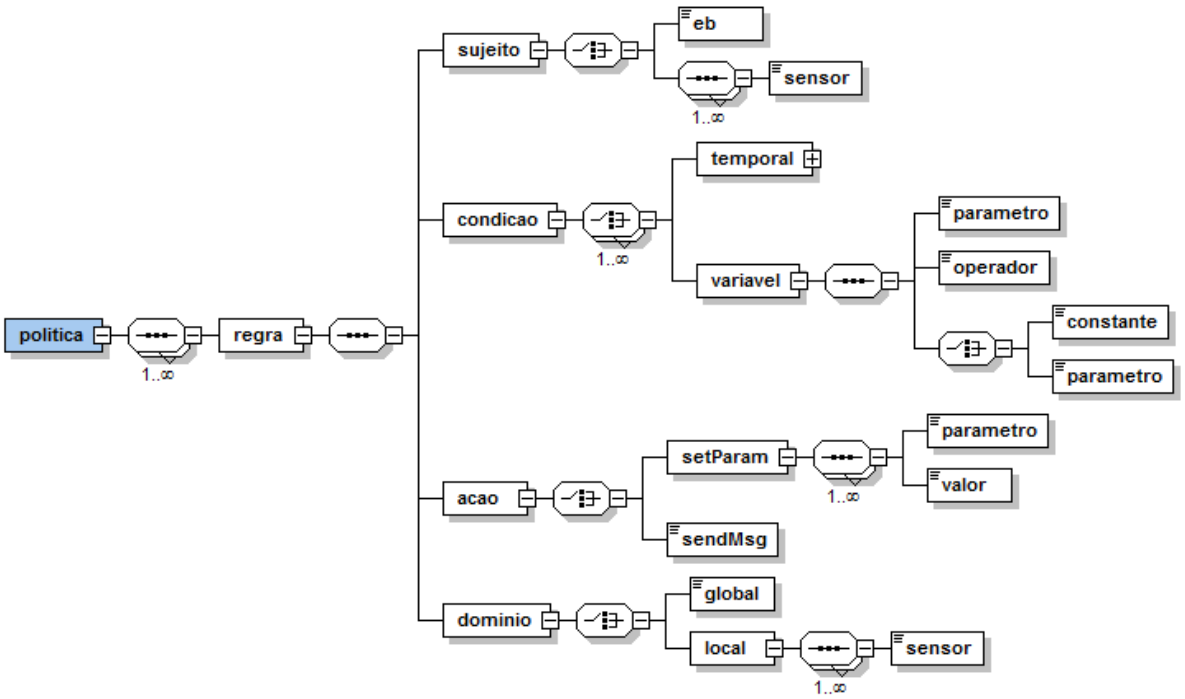


Figura 5.1: Diagrama do XML Schema das políticas utilizadas.

Uma política é um conjunto de regras que regem decisões implementadas para atingir objetivos específicos [1]. No DYSTOPOL as políticas são definidas baseadas na linguagem *Policy Framework Definition Language (PFDL)*. No entanto, para adaptá-la às necessidades do DYSTOPOL foi definido um *XML schema*, como ilustrado na Figura 5.1 através de um diagrama. O documento *XML schema* geral das políticas encontra-se no Apêndice I. O diagrama é composto por: elementos (retângulos), sequencias de elementos (figura com ...) e opções. Nas próximas seções são descritos os detalhes das opções para configurar políticas.

As políticas são constituídas por um ou mais conjuntos de regras e estas são constituídas por sujeito, condição, ação e domínio. Dentro de cada um desses elementos é possível construir políticas distintas com base em vários componentes opcionais.

5.2.1 Condição

Este elemento define as condições necessárias à aplicação da política. As condições podem ser temporais e/ou variáveis. Na Figura 5.1 é possível observar a organização do elemento. As condições variáveis têm como base um parâmetro existente e uma comparação, através de um operador matemático, com outro parâmetro ou constante. Já as condições temporais são, como o nome indica, definidas condições que se relacionam com o tempo de execução do sistema. Elas podem ser definidas com um tempo inicial e final ou somente inicial. Na Figura 5.2 podem-se observar dois exemplos de condições simples. No exemplo

```
...
<condicao>
        <variavel>
                <parametro>var_ult_leitura</parametro>
                <operador>&gt;</operador>
                <constante>50</constante>
        </variavel>
</condicao>
...
```

(a)

```
...
<condicao>
        <temporal>
                <inicio>300</inicio>
                <fim>2600</fim>
        </temporal>
</condicao>
...
```

(b)

Figura 5.2: Exemplos de condições simples

5.2(a) é definida uma condição sobre o parâmetro *var_ult_leitura* que contém a variação em percentagem entre a penúltima e a última leitura obtida pelo sensor. Neste caso a condição será cumprida quando essa variação ultrapassar os 50%. No exemplo 5.2(b) é possível observar uma condição temporal com inicio aos 300 segundos de execução do sistema e fim aos 2600 segundos. Este tipo de condição é particularmente interessante quando existe sazonalidade no sistema.

Na Figura 5.3 o exemplo mostra uma condição composta ou seja uma condição que tem dois elementos conjuntivos, um temporal e outro variável. Neste exemplo imagine-se um aviário que necessita manter uma temperatura elevada durante todos os momentos. Entre os momentos de execução 450 segundos e 800 segundos simula-se o período da noite em que as temperaturas podem baixar mais e não existem funcionários. A condição

```
...
<condicao>
        <variavel>
                <parametro>leitura</parametro>
                <operador>&lt;</operador>
                <parametro>minimo_sistema</parametro>
        </variavel>
        <temporal>
                <inicio>450</inicio>
                <fim>800</fim>
        </temporal>
</condicao>
...
```

Figura 5.3: Exemplo de condição composta.

variável seria então cumprida quando alguma temperatura de qualquer sensor descesse abaixo do mínimo permitido pelo sistema.

5.2.2 Ação

O elemento ação da política identifica a ação ou ações a aplicar quando as condições são cumpridas. Atualmente somente dois tipos de ações, como pode ser visto na Figura 5.1, podem ser realizadas. Pode ser atribuído um novo valor a um parâmetro existente (setParam) e a outra opção é o envio de uma mensagem extraordinária (sendMsg) com uma mensagem específica definida. Na Figura 5.4 é possível observar dois exemplos de

```
...
<acao>
        <setParam>
                <parametro>th_d</parametro>
                <valor>10</valor>
        </setParam>
</acao>
...
```

(a)

```
...
<acao>
        <sendMsg>EMERGÊNCIA!!
                RISCO DE INCÊNDIO
        </sendMsg>
</acao>
...
```

(b)

Figura 5.4: Exemplos de ações possíveis.

ações. No primeiro exemplo 5.4(a) é atribuído um novo valor de 10% ao *threshold* de dados e no segundo exemplo 5.4(b) é enviada uma mensagem de alerta com o texto indicando iminente risco de incêndio.

5.2.3 Domínio

O elemento domínio da política, observável também na Figura 5.1, define sobre quais sensores esta política vai incidir. Pode ser de âmbito global quando a política é definida para todos os sensores ou local, quando restrita a um ou um conjunto de sensores.

5.2.4 Sujeito

O elemento sujeito da política, esquema na Figura 5.1, define quem será o elemento do sistema a fazer a validação da política. Da mesma forma que o elemento domínio pode ser um sensor ou um conjunto de sensores, podendo também ser a EB.

5.3 Arquitetura

Nesta seção é apresentado o modelo de políticas a ser utilizado na proposta. O modelo apresentado na Figura 5.5 tem quatro componentes principais: a definição de políticas, o repositório de políticas, os pontos de decisão de políticas (PDP) e os pontos de aplicação de políticas (PAP). A definição de políticas permite ao usuário definir, utilizando a linguagem apresentada na Seção 5.2, um conjunto de políticas que são interpretadas pela EB e armazenadas no repositório de políticas. O repositório de políticas representa a base de conhecimento de todo o sistema. Ele contém todas as políticas de todos os domínios, fica localizado na estação base e serve somente como depósito de regras não tendo qualquer tipo de intervenção direta no sistema. Os pontos de decisão de políticas são responsáveis pela verificação das condições descritas em cada uma das políticas ativas. Quando essas condições forem satisfeitas os PDP são responsáveis pelo envio da mensagem de ação para o domínio definido na política. Os pontos de aplicação de políticas (PAP) são responsáveis pela implementação das mensagens de ação enviadas pelos PDP.

Os PDPs estão localizados em dois ambientes distintos: na estação base e em cada sensor. O PDP da estação base tem como responsabilidade a verificação de todas as políticas ativas existentes no repositório. Esse ponto de decisão é o único com acesso ao repositório global de políticas. Ao contrário dos PDPs dos sensores, o PDP da estação

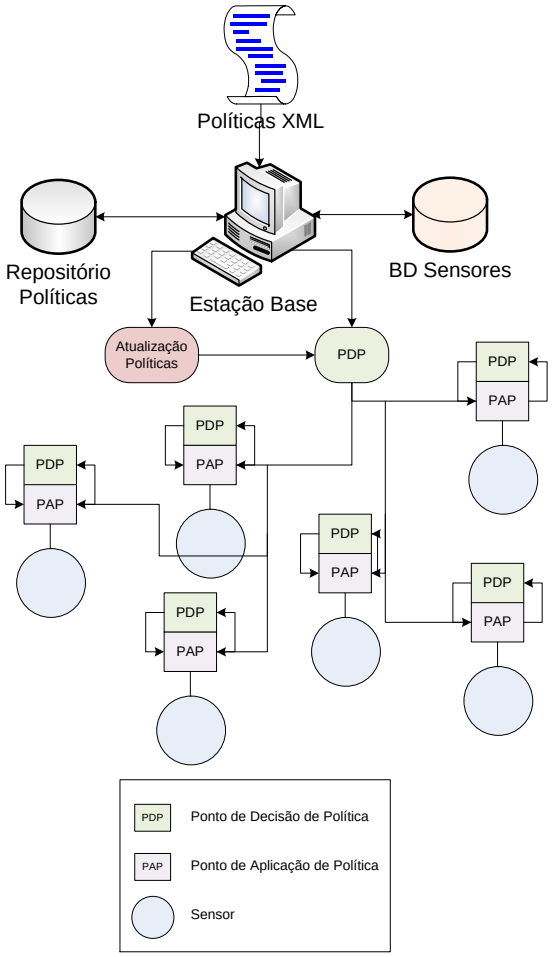


Figura 5.5: Modelo de Políticas.

base pode encapsular regras dentro de mensagens de ações assim podendo atualizar remotamente regras definidas em um ou vários sensores. Os PDPs existentes nos sensores verificam somente as regras locais que têm como sujeito o próprio sensor. São no geral regras simples pois o sensor tem recursos limitados. Os PAPs estão localizados somente

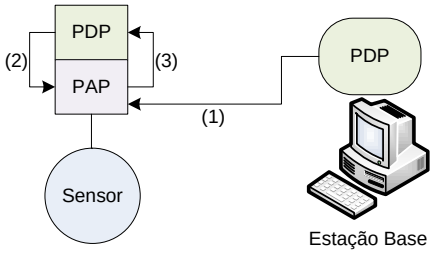


Figura 5.6: Modelo de aplicação de políticas.

nos sensores. A Figura 5.6 mostra a relação entre os agentes PDP e PAP. Os agentes PAP escutam a rede em busca de mensagens de ação enviadas pelo PDP da estação base (1)

ou recebem diretamente a ação a realizar pelo PDP do próprio sensor (2) e implementam a ação diretamente ou se a mensagem contiver uma atualização de políticas para o PDP do sensor, o PAP atualiza o repositório de regras locais do sensor (3).

5.3.1 Tipos de Mensagens

Existem dois tipos de mensagens enviadas através dos PDP para os PAP como se pode observar na Figura 5.7. As mensagens de ação e de atualização de políticas. As mensagens de ação devem conter três informações, são elas: o tipo de mensagem, o sensor alvo e a ação a ser realizada através da alteração de determinados parâmetros. As mensagens de atualização de políticas contêm atualização de regras locais dos sensores. Como foi definido anteriormente as regras de políticas devem conter quatro componentes: sujeito, domínio, ação e condições. Desta forma as mensagens de atualização de politicas são uma particularização das mensagens de ação que já contêm: o sujeito através do destinatário da mensagem, o sensor alvo e a ação através de campos específicos na mensagem. Resta somente a definição das condições para a aplicação de políticas. Essas condições devem ser quantificadas uma única vez na mensagem e todas as condições devem ser tipificadas já que podem ser relativas a uma variação de parâmetros ou uma condição temporal. As condições podem assim incluir o parâmetro, operador e valor a monitorar, ou simplesmente uma data de inicio e fim de politica.

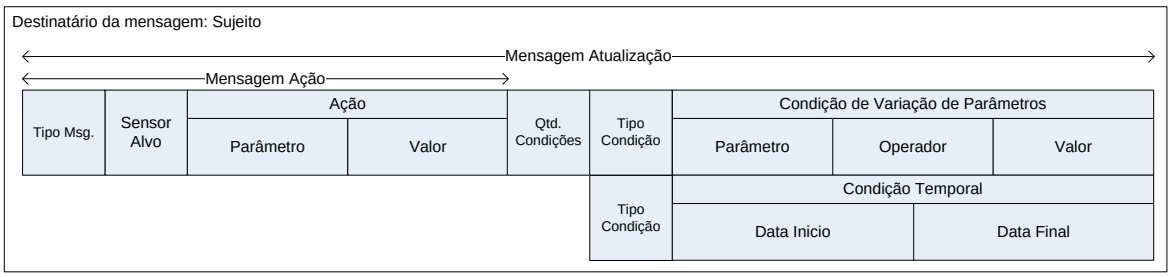


Figura 5.7: Estrutura de mensagens de ação e atualização de políticas.

Considere como exemplo a mensagem da Figura 5.8. Nessa mensagem enviada pela EB para o sensor A, a frequência de sensoreamento para a EB deve aumentar para uma leitura a cada 10 segundos quando a variação entre o último valor sensoreado e o atual representar uma variação de mais de 10%. O PAP do sensor A recebe essa mensagem

e atualiza o repositório local de políticas. A partir desse momento o PDP avaliará as condições a cada nova leitura e quando a condição for satisfeita irá enviar uma mensagem de ação direta para o PAP que simplesmente mudará a frequência de sensoreamento.

Destinatário da mensagem: Sensor A (Sujeito)

Tipo Msg. (MSG_P)	Sensor Alvo (A)	Ação		Qtd. Condições (1)	Tipo Condição (C_P)	Condição de Variação de Parâmetros		
		Parâmetro (FREQ_SENS)	Valor (10s)			Parâmetro (VAR_ULT_L)	Operador (>)	Valor (10%)

Figura 5.8: Exemplo de mensagem de atualização de políticas.

O modelo de políticas apresentado é um modelo global que será utilizado especificamente neste caso para cumprir com os objetivos definidos anteriormente. No entanto a ambição deste trabalho é preparar o sistema para se adaptar a qualquer outro tipo de funcionalidades que possam ser definidas.

5.4 Experimentos

Nesta seção são apresentados os experimentos realizados para validar o DYSTOPOL. Para testar o modelo de políticas foram realizados três experimentos que tem como objetivo determinar se o sistema tem a capacidade de interpretar e aplicar corretamente as políticas.

5.4.1 Configurações de Simulação

Para o experimento 1 são utilizadas as mesmas configurações apresentadas na Tabela 4.3 do Capítulo 4. Nos experimentos 2 e 3 é utilizado somente o cenário real descrito nessa mesma tabela e os *thresholds* passam a ser controlados dinamicamente pelas políticas.

5.4.2 Experimento 1: DYSTO VS DYSTOPOL

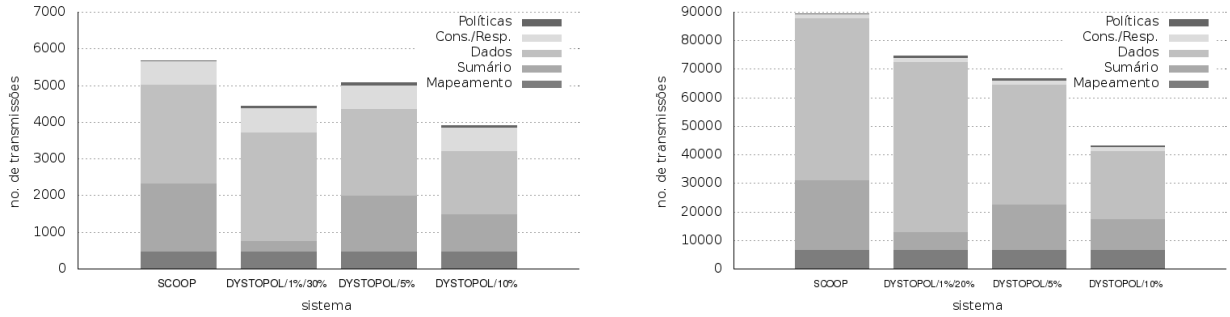
Neste primeiro experimento o objetivo é determinar a corretude do DYSTOPOL tendo em consideração os resultados do DYSTO, definido no capítulo anterior. Para isso ao invés de utilizar um *threshold* codificado no sistema definiu-se uma política que define os *threshold* utilizados no experimento 5 descrito na Seção 4.4.6. Essa política foi definida com a EB

como sujeito, domínio global, condição temporal de início de 1s e ações com a atribuição de *thresholds* correspondentes a cada cenário. A Figura 5.9 apresenta o arquivo XML com a configuração de ambos os *thresholds* em 5%. Na Figura 5.10 é possível observar

```
<?xml version="1.0" encoding="UTF-8"?>
<politica>
  <regra>
    <sujeito>
      <eb />
    </sujeito>
    <condicao>
      <temporal>
        <inicio>1</inicio>
      </temporal>
    </condicao>
    <acao>
      <setParam>
        <parametro>th_d</parametro>
        <valor>5</valor>
      </setParam>
      <setParam>
        <parametro>th_h</parametro>
        <valor>5</valor>
      </setParam>
    </acao>
    <dominio>
      <global />
    </dominio>
  </regra>
</politica>
```

Figura 5.9: Uma das políticas utilizadas no experimento 1.

que os resultados são semelhantes aos apresentados pelo DYSTO tendo no entanto um custo acrescido de menos de 1% em ambos os cenários, resultante do envio de mensagens de ação de política da EB (sujeito) para os sensores (domínio).



(a) Cenário real (b) Cenário sintético

Figura 5.10: Comparação entre o Scoop e o DYSTOPOL

Na Figura 5.10 e em cada um dos cenários existem três políticas distintas: *a*) *threshold* de dados de 1% e *threshold* de histograma de 30% para o cenário real e 20% para o cenário

sintético; *b*) ambos os *thresholds* em 5%; e por fim *c*) com ambos os *thresholds* em 10%. A primeira configuração foi escolhida combinando um valor de Th_d que produz resultados de consultas com um erro máximo relativo de 1% e 0,5% em média, com um valor de Th_h que produz a máxima redução no número total de transmissões, com base nos experimentos reportados na Seção 4.4.4.

5.4.3 Experimento 2: Dynamic Thresholds

Neste experimento definiram-se políticas que alteram o valor dos *thresholds* durante a execução do sistema. A Figura 5.11 apresenta a política definida para o experimento. A ideia é testar a alteração do *threshold* de histogramas Th_h durante a execução do sistema e verificar se ela tem impacto no número de mensagens de histograma produzidas especificamente por um sensor. Neste experimento foi escolhido aleatoriamente o sensor 3. A Figura 5.12 apresenta os resultados da simulação utilizando a política definida. Pode

```
<?xml version="1.0" encoding="UTF-8"?>
<politica>
  <regra>
    <sujeito>
      <sensor>3</sensor>
    </sujeito>
    <condicao>
      <temporal>
        <inicio>600</inicio>
        <fim>1200</fim>
      </temporal>
    </condicao>
    <acao>
      <setParam>
        <parametro>th_h</parametro>
        <valor>10</valor>
      </setParam>
    </acao>
    <dominio>
      <local>
        <sensor>3</sensor>
      </local>
    </dominio>
  </regra>
  <regra>
    <sujeito>
      <sensor>3</sensor>
    </sujeito>
    <condicao>
      <temporal>
        <inicio>1500</inicio>
      </temporal>
    </condicao>
    <acao>
      <setParam>
        <parametro>th_h</parametro>
        <valor>5</valor>
      </setParam>
    </acao>
    <dominio>
      <local>
        <sensor>3</sensor>
      </local>
    </dominio>
  </regra>
</politica>
```

Figura 5.11: Política definida para o experimento 2.

ser observado que políticas diferentes de *thresholds* resultaram em diferentes números de mensagens de sumário transmitidas. Ao aplicar a primeira regra da política, que coloca o th_h com o valor de 10%, o efeito sobre o número de mensagens no período reduziu em 57%. Após o final da vigência da política, o th_h voltou para o valor 0% o que fez com que todas as mensagens fossem enviadas. No final do segundo período o th_h voltou a ser alterado para 5% o que impactou ainda na última mensagem e reduziu o envio de 7 para 6 mensagens no período e o período final com esse th_h representou uma melhoria de 43% no número envios de mensagens de sumário para o sensor 3.

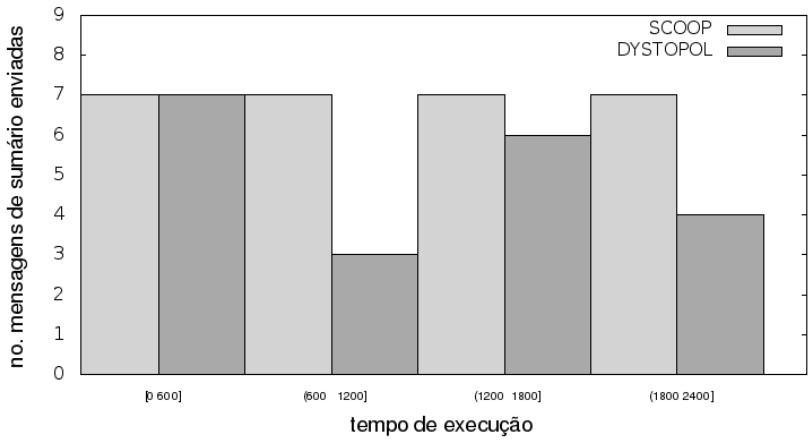


Figura 5.12: Experimento com *Thresholds* de histograma dinâmicos.

5.4.4 Experimento 3: Mensagens de Alerta

Este experimento tem como objetivo demonstrar o envio de mensagens de alerta em certas condições específicas. Neste caso a configuração do experimento implementou uma condição temporal e variável. Para simular uma situação real de quebra de serviço por excesso de temperatura numa sala de servidores durante o período noturno, criou-se uma política com uma condição temporal que representa um período noturno e uma condição variável relativa à temperatura máxima possível. A política resultante desses conceitos é apresentada na Figura 5.13. Foram utilizadas as configurações do cenário real tendo sido introduzidas diversas leituras que excediam o valor máximo dentro e fora do tempo de ação da política. Os resultados podem ser observados na Tabela 5.1. Eles mostram que o sensor interpretou corretamente a política definida e enviou corretamente as mensagens

```
<?xml version="1.0" encoding="UTF-8"?>
<politica>
  <regra>
    <sujeito>
      <global />
    </sujeito>
    <condicao>
      <temporal>
        <inicio>700</inicio>
        <fim>1700</fim>
      </temporal>
      <variavel>
        <parametro>ult_leitura</parametro>
        <operador>&gt;</operador>
        <constante>40</constante>
      </variavel>
    </condicao>
    <acao>
      <sendMsg>Alerta! Temperatura acima de 40°C</sendMsg>
    </acao>
    <dominio>
      <global />
    </dominio>
  </regra>
</politica>
```

Figura 5.13: Política utilizada no experimento 3.

Tempo(s)	Leituras Injetadas	Mensagens Alerta
0-700	12	0
700-1700	20	18
1700-2400	14	0

Tabela 5.1: Resultados do experimento 3.

quando seria necessário, sendo que duas delas embora tenham sido enviadas não chegaram à EB.

5.5 Conclusão

Os experimentos e consequentemente os resultados do DYSTOPOL comprovam que é possível definir políticas que transformem dinamicamente o modelo de armazenamento. A utilização de um *XML Schema* permite a uniformização das regras e facilita aos usuários a introdução de novas políticas. Embora ainda com um alcance limitado, a nível de parâmetros disponíveis, ficou provado através dos três experimentos, que existe potencial para estender as políticas para que seja possível aumentar a complexidade delas. O experimento 1 mostra que é possível transformar os *thresholds* apresentados no experimento final do DYSTO por políticas equivalentes que não aumentam em mais de 1% o valor total de transmissões. O experimento 2 comprovou que o sistema se adapta durante a

execução a novos valores de *thresholds*. Este tipo de mudança seria difícil de implementar no DYSTO sem ter que recorrer à paragem e codificação do sistema. O experimento 3 apresenta uma nova abordagem do sistema de politicas que deixa de ser somente um redutor de tráfego para passar a ser um otimizador de funcionalidades através do envio de mensagens de alerta e outras potenciais aplicações.

Capítulo

6

Conclusões

Esta dissertação teve seu foco no desenvolvimento de um sistema de armazenamento dinâmico para RSSFs. Para isso, foram consideradas as característica do sistema proposto por [11]. Esta dissertação teve como objetivo mostrar as potencialidades de utilizar um modelo de armazenamento dinâmico baseado em políticas. Os sistemas descritos nesta dissertação foram implementados e testados no simulador NS2. O desenvolvimento da dissertação envolveu o projeto de dois sistemas similares, mas ao mesmo tempo diferentes, nos quais foi possível mostrar focos distintos dentro de uma mesma idéia principal.

DYSTO foi o primeiro sistema a ser desenvolvido. Seu objetivo era testar o impacto de mudanças em algumas variáveis do modelo de armazenamento Scoop, proposto em [11]. Foi especificado um sistema de *thresholds* que permitiu tornar alguns parâmetros mais adequados, subsistuindo a geração de mensagens em períodos fixos de tempo. Com isso, passou-se a analisar a necessidade real de envio dos diversos tipos de mensagens trocados entre componentes do sistema e avaliar as vantagens e desvantagens da alteração proposta. Com a apresentação do modelo preliminar DYSTO, foi possível perceber que existem várias parametrizações do Scoop que podem ser alteradas e que resultam em reduções significativas de até 52% no número de transmissões. No entanto, DYSTO implementa soluções estáticas já que as mudanças operadas para reduzir o número de transmissões são fixas durante toda a execução do sistema. Assim a utilização de *thresholds* fixos não é ideal para ambientes onde os cenários estão em contínua mudança.

O DYSTOPOL teve como objetivo explorar os resultados obtidos pelo DYSTO e estendê-los de uma forma que possam ser utilizadas as melhores configurações possíveis em cada cenário existente. A apresentação do DYSTOPOL teve um enfoque não somente no potencial de redução de transmissões mas nas possibilidades que se abrem ao poder adaptar todo o sistema de uma forma automática. A utilização de regras em *XML* para definir políticas simplificou o trabalho de configuração por parte dos usuários já que existe também um conjunto definido de regras. O DYSTO já tinha provado ser possível reduzir o número de transmissões, em comparação com o Scoop, e o DYSTOPOL conseguiu resultados similares utilizando políticas. Assim foi possível explorar outras funcionalidades que não passam somente pela redução de transmissões. Foi possível criar situações de alertas em caso de risco iminente assim como mudar, durante a execução, várias vezes a parametrização de uma mesma variável do sistema.

O resultado dos experimentos dos dois sistemas justificou o desenvolvimento do modelo de gestão de armazenamento dinâmico que provou ser um tema pertinente e desafiador.

Esta dissertação deu origem às seguintes publicações:

- DYSTO - A Dynamic Storage Model for Wireless Sensor Networks
  Nuno M. F. Gonçalves, Aldri L. dos Santos, e Carmem S. Hara
  Journal of Information and Data Management, Volume 3, Número 3, outubro, 2012

- A Policy-based Storage Model for Sensor Networks
  Nuno M. F. Gonçalves, Aldri L. dos Santos, e Carmem S. Hara
  Anais do 2014 IEEE/IFIP Network Operations and Management Symposium, maio, 2014

6.1 Trabalhos Futuros

A partir da definição e implementação dos modelos descritos nesta dissertação, entende-se que cabem como extensão ao modelo os seguintes trabalhos futuros:

- Resolução de conflitos de políticas - atualmente não existem regras definidas acerca de políticas conflitantes. Seria importante definir prioridades dentro dos vários

domínios existentes sendo que por exemplo políticas locais poderiam ter precedência sobre políticas globais.

- Expansão dos parâmetros utilizáveis - criado com o intuito de validar uma idéia o DYSTOPOL não fornece muitas possibilidades de configuração de novas políticas pois o número de parâmetros acessíveis pelas políticas é baixo.

- Estender o *XML Schema* - Estudar a possibilidade de adicionar novas ações e condições para aumentar as possíveis políticas.

- Permitir leituras de múltiplos atributos - Atualmente os sistemas somente consideram o armazenamento de um único tipo de leitura de valor discreto. Seria interessante a coleta de diversos atributos e definir os IAs com base nos vários atributos sensoriados.

- Realização de mais experimentos - O DYSTOPOL pode ser utilizado para realizar mais experimentos que podem trazer novos detalhes sobre a gestão de armazenamento dinâmico.

- Validação dos modelos com sensores dinâmicos - Embora todos os experimentos tenham sido executados num cenário de sensores estáticos, esta não é uma asserção do sistema. Validações dos modelos num sistema dinâmico seriam contribuições interessantes.

Capítulo

7

Apêndice I

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema>
  <xs:element name="politica">
    <xs:complexType>
      <xs:sequence maxOccurs="unbounded">
        <xs:element name="regra">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="sujeito">
                <xs:complexType>
                  <xs:choice>
                    <xs:element name="eb">
                      <xs:simpleType>
                        <xs:restriction base="xs:string">
                          <xs:maxLength value="0" />
                        </xs:restriction>
                      </xs:simpleType>
                    </xs:element>
                    <xs:element name="global">
                      <xs:simpleType>
                        <xs:restriction base="xs:string">
                          <xs:maxLength value="0" />
                        </xs:restriction>
                      </xs:simpleType>
                    </xs:element>
                    <xs:sequence maxOccurs="unbounded">
                      <xs:element name="sensor" type="xs:int" />
                    </xs:sequence>
                  </xs:choice>
                </xs:complexType>
              </xs:element>
              <xs:element name="condicao">
                <xs:complexType>
                  <xs:choice maxOccurs="unbounded">
                    <xs:element name="temporal">
                      <xs:complexType>
                        <xs:choice maxOccurs="2">
                          <xs:element name="inicio" type="xs:positiveInteger" />
                          <xs:element name="fim" type="xs:positiveInteger" />
                        </xs:choice>
                      </xs:complexType>
                    </xs:element>
                    <xs:element name="variavel">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="parametro" type="xs:string" />
                          <xs:element name="operador" type="xs:string" />
                          <xs:choice>
                            <xs:element name="constante" type="xs:integer" />
                            <xs:element name="parametro" type="xs:string" />
                          </xs:choice>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:choice>
                </xs:complexType>
              </xs:element>
              <xs:element name="acao">
                <xs:complexType>
                  <xs:choice maxOccurs="unbounded">
                    <xs:element name="setParam">
                      <xs:complexType>
                        <xs:sequence maxOccurs="unbounded">
                          <xs:element name="parametro" type="xs:string" />
                          <xs:element name="valor" type="xs:int" />
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                    <xs:element name="sendMsg" type="xs:string" />
                  </xs:choice>
                </xs:complexType>
              </xs:element>
              <xs:element name="dominio">
                <xs:complexType>
                  <xs:choice>
                    <xs:element name="global">
                      <xs:simpleType>
                        <xs:restriction base="xs:string">
                          <xs:maxLength value="0" />
                        </xs:restriction>
                      </xs:simpleType>
                    </xs:element>
                    <xs:element name="local">
                      <xs:complexType>
                        <xs:sequence maxOccurs="unbounded">
                          <xs:element name="sensor" type="xs:int" />
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:choice>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Figura 7.1: *XML Schema* do DYSTOPOL.

Referências Bibliográficas

[1] Terminology for policy-based management rfc 3198, 2011.

[2] S. Ahmed, M.R. Eskicioglu, and S. Hussain. Querying sensor networks: techniques, evaluation, and new directions. In *Computer Communications and Networks, 2004. ICCCN 2004. Proceedings of the 13th International Conference on*, page 548, oct. 2004.

[3] I.F. Akyildiz, Weilian Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *Communications Magazine, IEEE*, 40(8):102 – 114, aug 2002.

[4] A. K. Bandara, N. Damianou, E. C. Lupu, and M. Sloman. Policy based management. In J. Bergstra and M. Burgess, editors, *Handbook of Network and System Administration*, pages 507–564. Elsevier, 2008.

[5] M. Bhardwaj, T. Garnett, and A.P. Chandrakasan. Upper bounds on the lifetime of sensor networks. In *Communications, 2001. ICC 2001. IEEE International Conference on*, volume 3, pages 785 –790 vol.3, 2001.

[6] Frank Dabek, Emma Brunskill, M. Frans Kaashoek, David Karger, Robert Morris, Ion Stoica, and Hari Balakrishnan. Building peer-to-peer systems with chord, a distributed lookup service. In *In Proc. of the 8th IEEE Workshop on Hot Topics in Operating Systems (HotOS-VIII*, pages 71–76, 2001.

[7] G. de Meulenaer, F. Gosset, F.-X. Standaert, and O. Pereira. On the energy cost of communication and cryptography in wireless sensor networks. In *Networking and*

Communications, 2008. WIMOB '08. IEEE International Conference on Wireless and Mobile Computing,, pages 580–585, oct. 2008.

[8] Amol Deshpande, Carlos Guestrin, Samuel R. Madden, Joseph M. Hellerstein, and Wei Hong. Model-driven data acquisition in sensor networks. In *Proceedings of the Thirtieth international conference on Very large data bases - Volume 30*, VLDB '04, pages 588–599. VLDB Endowment, 2004.

[9] Carlos M. S. Figueiredo, Aldri L. dos Santos, Antonio A. F. Loureiro, and José M. Nogueira. Policy-Based Adaptive Routing in Autonomous WSNs. *Ambient networks: 16th IFIP/IEEE International*, 2005.

[10] D. Ganesan and P. Shenoy. PRESTO: Feedback-Driven Data Management in Sensor Networks. *IEEE/ACM Transactions on Networking*, 17(4):1256–1269, August 2009.

[11] Thomer M. Gil and Samuel Madden. Scoop: An Adaptive Indexing Scheme for Stored Data in Sensor Networks. *2007 IEEE 23rd International Conference on Data Engineering*, pages 1345–1349, 2007.

[12] Thomer M. Gil and Samuel Madden. Scoop: An Adaptive Indexing Scheme for Stored Data in Sensor Networks. Technical report, April 2007.

[13] Nils Hoeller, Christoph Reinke, Jana Neumann, Sven Groppe, Florian Frischat, and Volker Linnemann. DACS: A dynamic approximative caching scheme for Wireless Sensor Networks. In *Digital Information Management (ICDIM), 2010 Fifth International Conference on*, pages 339–346. IEEE, 2010.

[14] Zhihau Hu and Baochun Li. On the fundamental capacity and lifetime limits of energy-constrained wireless sensor networks. In *Real-Time and Embedded Technology and Applications Symposium, 2004. Proceedings. RTAS 2004. 10th IEEE*, pages 2 – 9, may 2004.

[15] Ankur Jain, Edward Y. Chang, and Yuan-Fang Wang. Adaptive stream resource management using kalman filters. In *Proceedings of the 2004 ACM SIGMOD inter-*

national conference on Management of data, SIGMOD '04, pages 11–22, New York, NY, USA, 2004. ACM.

[16] Qiangfeng Jiang and D. Manivannan. Routing protocols for sensor networks. In *Consumer Communications and Networking Conference, 2004. CCNC 2004. First IEEE*, pages 93–98, jan. 2004.

[17] Apoorva Jindal and Konstantinos Psounis. Modeling spatially correlated data in sensor networks. 2:466–499, 2006.

[18] Tuan D. Le, Wen Hu, Sanjay Jha, and Peter Corke. Design and implementation of a policy-based management system for data reliability in Wireless Sensor Networks. *2008 33rd IEEE Conference on Local Computer Networks (LCN)*, pages 762–769, October 2008.

[19] Fangfang Li, Zhibo Feng, Chuanwen Li, Jia Xu, and Ge Yu. A Data Storage Method Based on Multilevel Mapping Index in Wireless Sensor Networks. *2007 International Conference on Wireless Communications, Networking and Mobile Computing*, pages 2747–2750, September 2007.

[20] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. In *Proceedings of the 5th symposium on Operating systems design and implementation*, OSDI '02, pages 131–146, New York, NY, USA, 2002. ACM.

[21] Nelson Matthys, Christophe Huygens, Danny Hughes, Sam Michiels, and Wouter Joosen. A Component and Policy-Based Approach for Efficient Sensor Network Reconfiguration. *2012 IEEE International Symposium on Policies for Distributed Systems and Networks*, pages 53–60, July 2012.

[22] Nelson Matthys, Christophe Huygens, Danny Hughes, Jo Ueyama, Sam Michiels, and Wouter Joosen. Policy-driven tailoring of sensor networks. In *S-CUBE*, pages 20–35, 2010.

[23] G.V. Merrett, N.M. White, N.R. Harris, and B.M. Al-Hashimi. Energy-aware simulation for wireless sensor networks. In *Sensor, Mesh and Ad Hoc Communications and Networks, 2009. SECON '09. 6th Annual IEEE Communications Society Conference on*, pages 1–8, june 2009.

[24] Gregory J. Pottie and William J. Kaiser. Wireless integrated network sensors. *Commun. ACM*, 43(5):51–58, 2000.

[25] Sylvia Ratnasamy, Brad Karp, Scott Shenker, Deborah Estrin, Ramesh Govindan, L. Yin, and Fang Yu. Data-centric storage in sensornets with GHT, a geographic hash table. *Mobile networks and applications*, 8(4):427–442, 2003.

[26] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *IN: MIDDLEWARE*, pages 329–350, 2001.

[27] Linnyer B Ruiz. Manna: Uma arquitetura para gerenciamento de redes de sensores sem fio. Technical report, 2003.

[28] Haiying Shen, Lianyu Zhao, and Ze Li. A Distributed Spatial-Temporal Similarity Data Storage Scheme in Wireless Sensor Networks. *IEEE Transactions on Mobile Computing*, pages 1–6, August 2010.

[29] John Strassner and Stephen Schleimer. Policy framework definition language, 1998.

[30] Sameer Tilak, Nael B. Abu-Ghazaleh, and Wendi Heinzelman. A taxonomy of wireless micro-sensor network models. *SIGMOBILE Mob. Comput. Commun. Rev.*, 6(2):28–36, April 2002.

[31] Kai-Chao Yang, Yuan-Cheng Yang, Chun-Lung Lin, and Jia-Shung Wang. Hierarchical Data Management for Spatial-Temporal Information in WSNs. *2010 Fourth International Conference on Sensor Technologies and Applications*, pages 435–440, July 2010.

[32] Sunhee Yoon and Cyrus Shahabi. The clustered aggregation (cag) technique leveraging spatial and temporal correlations in wireless sensor networks. *ACM Trans. Sen. Netw.*, 3, March 2007.

[33] Yanjun Zhang, Siye Wang, Zhenyu Liu, Wenbiao Zhou, Xu Yang, Bo Zhou, and Dake Liu. Analysis on energy consumption and transmission delay for wireless sensor network. In *Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), 2012 IEEE International Conference on*, pages 110–113, may 2012.