Middleware Orientado a Serviços para Redes de Sensores sem Fio

Flávia Delicato[1], Paulo F. Pires[1, 2], Alexandre Lages[1]
José Ferreira de Rezende[3], Luci Pirmez[1]

Núcleo de Computação Eletrônica[1], Departamento de Ciência da Computação[2]
Grupo de Teleinformática e Automação[3] – Universidade Federal do Rio de Janeiro

{fdelicato, paulopires, alexandrelages}@nce.ufrj.br, rezende@gta.ufrj.br, luci@nce.ufrj.br

Abstract. *There is a wide range of applications for wireless sensor networks (WSNs) with different needs. The WSN infrastructure and protocols change according to the application needs. To achieve the best performance of the WSN, its operation should be adapted to the application needs. We propose a middleware for WSNs that provides a layer between applications and the network. The middleware offers a standard mechanism for representing user queries, sensor tasks and data. It also provides an automatic choice of the best network configuration and data dissemination strategy. Users are able to access the WSN without worrying about the underlying infrastructure and software. From the WSN perspective, the system provides the best match between communication protocols and application requirements.*

Resumo*. Há uma ampla gama de aplicações para redes de sensores sem fio (RSSF)s, com diferentes necessidades. A infraestrutura e o protocolo de disseminação de dados da rede variam de acordo com a aplicação. Para o melhor desempenho quanto ao consumo de energia e à qualidade do serviço fornecido pela rede, seu funcionamento deve ser adaptado às necessidades da aplicação. Este trabalho propõe um middleware que oferece uma camada entre aplicações e a rede de sensores e oferece um mecanismo padrão para representar consultas, tarefas e dados. Além disso, fornece a escolha automatizada da configuração da rede e da estratégia de disseminação de dados usada, permitindo ao usuário acessar a rede sem tomar conhecimento de infraestrutura e software subjacentes. Do ponto de vista da rede, o sistema visa obter a melhor combinação entre protocolos de comunicação e requisitos da aplicação.*

1. Introdução

A área de redes de sensores sem fio (RSSF) constitui um campo de pesquisa emergente com amplas implicações ao conectar os mundos digital e físico. A estreita conexão com o ambiente físico permite que os sensores forneçam medições locais detalhadas as quais são difíceis de obter através de técnicas de instrumentação tradicionais ou de sensoriamento remoto. Se por um lado as redes de sensores trazem novas e amplas perspectivas para o monitoramento de variáveis ambientais, por outro lado trazem também novos desafios.

Uma RSSF é composta por dezenas a milhares de dispositivos de baixo custo e tamanho reduzido capazes de realizar sensoriamento, processamento e transmissão de informação através de enlaces sem fio. Sensores são alimentados por baterias não recarregáveis e devem operar sem assistência humana por longos períodos de tempo. Como a maior fonte

de consumo de energia nos sensores é a transmissão de dados, os protocolos de comunicação para RSSFs visam propor soluções para minimizar a quantidade e o alcance das transmissões a fim de estender o tempo de vida global da rede. A maioria dos protocolos baseia-se na comunicação de curto alcance em múltiplos saltos (*multi-hop)* e adota algum mecanismo de agregação a fim de reduzir a quantidade de dados a ser transmitida. Além de possuir recursos escassos, tanto computacionais quanto de energia, outra característica das RSSFs é sua estrutura organizacional e topológica altamente dinâmica. Portanto, RSSFs devem possuir algum grau de auto-organização e adaptação.

Há uma ampla gama de aplicações para RSSFs, com diferentes requisitos. Vários trabalhos [7,12] destacam a importância da participação da aplicação no processo de comunicação em RSSFs. Otimizações específicas da aplicação podem reduzir o número de transmissões e, por conseguinte, o consumo total de energia na rede.

Os autores em [12] classificam as RSSFs quanto ao modo de entrega de dados da aplicação em contínuas, dirigidas a eventos, iniciadas pelo observador e híbridas. Cada classe tem suas necessidades específicas e é melhor atendida por um tipo de protocolo de comunicação. Uma questão importante, então, passa a ser a seleção do protocolo mais adequado a cada classe de aplicação. Em [7] observou-se que uma cuidadosa escolha do protocolo pode diminuir a quantidade global de tráfego na rede em até 50%, aumentando a eficiência da rede em termos de consumo de energia. Entretanto, em geral é extremamente difícil para os desenvolvedores de aplicações escolherem o protocolo de rede que melhor se adapte as suas necessidades.

A partir da constatação dos ganhos de desempenho possibilitados pela escolha criteriosa do protocolo de comunicação segundo os requisitos da aplicação e pela dificuldade dos usuários em realizar tal escolha, este trabalho propõe um *middleware* para RSSFs que faça a intermediação e tradução das necessidades das aplicações usuárias, a fim de selecionar e configurar o protocolo mais apropriado. O sistema proposto reside em uma camada intermediária entre a subcamada de comunicação e a aplicação, e é projetado de forma a ser inserido sem impacto na pilha de protocolos adotada em RSSFs. O sistema é genérico o suficiente para ser utilizado por uma ampla gama de aplicações.

Apesar das vantagens comprovadas das tecnologias tradicionais de *middleware*, poucos trabalhos em RSSFs têm considerado essas tecnologias no projeto das redes. RSSFs têm sido construídas com um alto grau de dependência entre as aplicações e os protocolos de comunicação. A justificativa para essa dependência é a necessidade de eficiência em termos de energia. Os trabalhos atuais assumem que deve haver um forte acoplamento entre as camadas da pilha de protocolos a fim de prover tal eficiência. Entretanto, isso gera a construção de sistemas rígidos, com redes de sensores projetadas exclusivamente para aplicações específicas. RSSFs do futuro deverão atender diferentes usuários, com necessidades dinâmicas, sendo necessário um maior grau de flexibilidade no projeto das redes.

A concepção do *middleware* proposto foi baseada nas recentes pesquisas em Serviços *Web* [5]. Neste trabalho é adotada uma abordagem de serviços, na qual a RSSF é vista como uma fornecedora de serviços para aplicações usuárias. Os principais serviços fornecidos pelo *middleware* proposto são a interpretação das necessidades das aplicações e a seleção do melhor protocolo e da melhor configuração da rede segundo essas necessidades, além da realização de operações de agregação de dados. Futuras versões do

sistema podem incorporar vários outros serviços genéricos para redes de sensores, como gerência de energia, controle de localização e segurança.

O projeto do *middleware* proposto adota o esquema de nomeação centrado em dados [10], no qual nós da rede são identificados por atributos que descrevem suas capacidades ou localização. Consultas realizadas pelas aplicações também são descritas em função desses atributos. Na abordagem adotada, atributos dos nós são utilizados para descrever os serviços fornecidos pela rede. O conjunto de serviços fornecido é publicado e acessado através da linguagem XML [14]. Mensagens XML trocadas na rede e com a aplicação são formatadas e empacotadas através do protocolo SOAP [16]. Aplicações que desejam acessar os dados da rede submetem mensagens anunciando seus interesses. Esses interesses são utilizados para enquadrar cada aplicação em um perfil, a partir do qual define-se a missão corrente para a rede. Cada missão conduzirá a escolha da melhor forma de organização topológica da rede e do protocolo de comunicação mais apropriado.

A escolha da linguagem XML foi motivada principalmente pelo seu alto poder de representatividade e sua capacidade de extensão, podendo ser adequada as necessidades específicas do ambiente de redes de sensores. A linguagem XML oferece flexibilidade suficiente para representar consultas e tarefas a serem atribuídas aos sensores. O protocolo SOAP foi escolhido por ser consideravelmente leve e também altamente versátil. Além disso, ambos são padrões de *facto* na web.

Os principais benefícios do *middleware* proposto podem ser avaliados sob dois pontos de vista: do usuário e da rede. Do ponto de vista do usuário, o sistema oferece uma camada de interoperabilidade entre as diferentes aplicações e a rede de sensores. O *middleware* cria a abstração de uma RSSF genérica, que fornece serviços para as várias aplicações. Outro aspecto importante é que, através da utilização da linguagem XML e do protocolo SOAP, o *middleware* proposto naturalmente possibilita a interoperabilidade da RSSF com a Internet. Desta forma, as funcionalidades da RSSF podem ser disponibilizadas como Serviços *Web*, que podem ser acessados por qualquer aplicação usuária com acesso a Internet, de modo independente de linguagem e plataforma. Do ponto de vista da rede, o sistema permite obter a melhor combinação entre protocolos de comunicação e requisitos da aplicação, a fim de atingir uma maior eficiência da rede em termos de consumo de energia. A rede de sensores passa a funcionar como um mecanismo genérico e configurável de extração de dados ambientais, podendo adaptar-se a diferentes missões.

O presente trabalho apresenta a arquitetura do *middleware* proposto e a descrição do seu funcionamento. Além disso, a implementação de uma versão inicial do *middleware* é descrita e são apresentados os resultados de simulações realizadas com o intuito de avaliar o sistema. O trabalho está organizado da seguinte forma: a Seção 2 apresenta os conceitos necessários para o entendimento da proposta. A Seção 3 contém a descrição do sistema proposto. A Seção 4 discute os trabalhos relacionados. A Seção 5 descreve as simulações e o protótipo desenvolvidos para validar a proposta. Finalmente, a Seção 6 contém algumas conclusões.

2. Redes de Sensores sem Fio (RSSF)

A seguir são apresentados alguns conceito utilizados na proposta, incluindo a arquitetura genérica de RSSFs e modelos de comunicação e de entrega de dados adotados nessas redes.

2.1 Arquitetura de Redes de Sensores sem Fio

Uma RSSF possui três principais componentes organizacionais: a infraestrutura, a pilha de protocolos e a aplicação [12]. **A infraestrutura** consiste nos nós da rede e no seu estado atual de instalação no ambiente. Em geral, uma rede possui um ou mais nós de escoamento de dados, chamados de sorvedouros, e diversos nós sensores. Sorvedouros são nós com maior poder computacional e sem restrições de energia. Esses nós fazem a interface entre a aplicação e a rede, servindo de ponto de entrada para a submissão dos interesses da aplicação e de concentrador das informações enviadas pelos nós sensores. Nós sensores contêm uma ou mais unidades de sensoriamento, e possuem limitadas capacidades de processamento e armazenamento. O estado de instalação da rede diz respeito à localização dos sensores no espaço físico e à densidade da rede.

A **pilha de protocolo** usada pela RSSF consiste nas camadas de aplicação, transporte, rede, enlace de dados e camada física. A **aplicação** é responsável por emitir consultas ou interesses que descrevem as características dos fenômenos que o usuário deseja analisar. Interesses devem indicar os tipos de dados desejados, a freqüência de coleta, o atraso máximo tolerado, se os dados devem sofrer agregação, limiares a partir dos quais transmiti-los, ou ainda, indicar tarefas a serem realizadas, como a ativação de atuadores.

Em geral, procedimentos de otimizações da rede cruzam as fronteiras desses componentes organizacionais. Por exemplo, características da aplicação podem influir tanto na infraestrutura da rede quanto nos protocolos utilizados. Portanto, o conhecimento de nível de aplicação deve ser aproveitado pela rede para que ela possa alcançar uma maior eficiência em termos de consumo de energia prolongando, assim, seu tempo de vida.

2.2 Modelos de Comunicação e de Entrega de Dados

Uma RSSF suporta dois tipos de comunicação [12]: de infraestrutura e da aplicação. A comunicação de infraestrutura é toda comunicação necessária entre os nós para configurar, manter e otimizar a operação da rede. Essa comunicação é gerada pelos protocolos de rede e de enlace em resposta aos requisitos da aplicação ou à ocorrência de eventos na rede.

A comunicação da aplicação é responsável pela transferência dos dados coletados sobre o fenômeno em estudo, desde sua origem até o usuário e pela transferência dos interesses do usuário para a rede, a partir da qual a transferência de dados tem início. Idealmente, o interesse da aplicação deve ser especificado em termos do fenômeno em estudo. A adoção de um esquema de nomeação centrado em dados permite a descrição de interesses na forma de conjuntos de atributos de baixo nível, tais como o tipo do nó sensor, a área geográfica que se deseja monitorar, e o intervalo de coleta (taxa de dados).

A partir do interesse definido pela aplicação, a rede começa a coletar e a enviar dados. O modelo de entrega de dados utilizado pela rede irá governar a geração do tráfego da aplicação. A taxonomia apresentada em [12] classifica as redes de sensores quanto ao modelo de entrega requerido pela aplicação em: contínuas, dirigidas a eventos, iniciadas pelo observador e híbridas. No modelo contínuo, sensores comunicam seus dados continuamente a uma taxa pré-definida. No modelo dirigido a eventos, sensores reportam informações somente se um evento de interesse ocorrer. No iniciado pelo observador, sensores reportam seus resultados em resposta a um pedido explícito da aplicação. Finalmente, as três abordagens podem coexistir na mesma rede, gerando um modelo híbrido de entrega. O modelo da entrega de dados dirige a escolha do melhor tipo de protocolo de disseminação de dados a ser adotado.

Os protocolos de disseminação podem ser classificados quanto à abordagem de roteamento em: baseados em difusão, *unicast* e *multicast*. Nos protocolos baseados em difusão, sensores enviam suas mensagens por broadcast para seus vizinhos, que por sua vez as reenviam por broadcast, e assim sucessivamente até alcançarem um sorvedouro. Nos protocolos *unicast*, sensores podem enviar seus dados diretamente para o sorvedouro, através de múltiplos saltos ou, se a rede for organizado em *clusters*, os sensores podem enviar seus dados para o líder de seu *cluster*. Os líderes, por sua vez, enviam seus dados diretamente para o sorvedouro ou através de outros líderes. Nos protocolos *multicast*, os sensores são reunidos em grupos conforme a aplicação e usam multicast para a comunicação com os membros dos grupos.

A interação entre o modelo de entrega de dados da aplicação e o tipo de protocolo de disseminação empregado tem um impacto significativo sobre o desempenho da rede em termos do consumo de energia [12]. Um dos objetivos do *middleware* proposto é promover essa interação, de forma flexível e transparente para o usuário.

3. Sistema Proposto

O termo *middleware* é amplamente usado para denotar uma camada composta por serviços genéricos situada abaixo das aplicações de usuários. O middleware mascara a heterogeneidade das arquiteturas de computadores, sistemas operacionais, linguagens de programação e tecnologias de rede para facilitar o desenvolvimento e a gerência de aplicações.

Um *middleware* para RSSFs deve isolar as aplicações das características de infraestrutura e do protocolo de rede subjacentes. Ele deve permitir a configuração dinâmica da rede de acordo com os requisitos da aplicação e garantir o uso eficiente dos recursos de comunicação disponíveis. Além disso, tem que ser robusto, tolerante a falhas e ter requisitos mínimos de armazenamento e processamento. O *middleware* deve incluir mecanismos para formular tarefas de sensoriamento de alto nível, comunicar as tarefas para a rede, realizar a agregação de dados e coordenar os nós na realização dessas tarefas.

A partir da constatação de que a adaptação da RSSF às características das aplicações promove o melhor desempenho da rede e do fato de que cada vez mais as redes de sensores atenderão a diferentes aplicações, este trabalho propõe um sistema de *middleware* que oferece uma camada flexível para promover a interação entre diferentes aplicações e a RSSF. A interação entre aplicações e a rede é necessária para se obter eficiência em termos de energia. Porém, essa interação deve ser realizada através de uma interface padrão, independente da aplicação, que obtenha os requisitos do usuário e configure a infraestrutura da rede subjacente, e não através da violação da estrutura em camadas da rede.

O *middleware* proposto visa fornecer uma representação padrão, de alto nível, para interesses da aplicação, dados gerados pelos sensores e funções de agregação a serem realizadas na rede. Tal representação é baseada no conceito de serviços e adota XML [14] como a linguagem de descrição e codificação. SOAP [16] é adotado como o protocolo para formatar e transportar as mensagens usando um protocolo de rede subjacente centrado em dados. O *middleware* busca encaixar as aplicações em perfis a fim de ajustar as características da rede às necessidades particulares de cada aplicação.

A escolha da linguagem XML foi motivada principalmente pelo seu alto poder de representatividade e sua capacidade de extensão. A linguagem oferece flexibilidade

suficiente para representar consultas e tarefas de sensores. SOAP foi escolhido por ser um protocolo leve e versátil. Além disso, ambos são padrões de *facto* na *Web*. É importante notar que a maior parte dos usuários de RSSFs não está interessada apenas em obter os dados extraídos da rede através de uma interface gráfica, mas utilizá-los para alimentar suas próprias aplicações. Esse tipo de interface aplicação-aplicação é altamente beneficiado pela abordagem de *middleware* baseado em XML e SOAP.

3.1 Arquitetura do Sistema

A arquitetura do sistema proposto é mostrada na Figura 1. Suas funcionalidades são separadas em dois módulos: adaptação e SOAP. O módulo de adaptação roda apenas em nós sorvedouros e é responsável por traduzir solicitações das aplicações em perfis e por estabelecer a missão da rede. O perfil consiste basicamente em classificar o modelo de entrega de dados, enquanto a missão da rede consiste na escolha da melhor topologia lógica e da estratégia de disseminação de dados a serem adotadas, de acordo com as necessidades de cada aplicação.

O módulo de adaptação é composto de uma unidade de decisão, cujas entradas são fornecidas pela aplicação e utilizadas para decidir a melhor configuração da rede. O módulo de adaptação comunica-se com o módulo SOAP, e possui uma API baseada em XML para a comunicação com o protocolo de rede subjacente. Conhecendo a missão estabelecida, o protocolo de rede gera a comunicação de infraestrutura necessária para configurar a topologia da rede e estabelecer a estratégia de disseminação dos dados. Como as mensagens geradas pela comunicação de infraestrutura são totalmente dependentes do protocolo, elas não são representadas em XML.

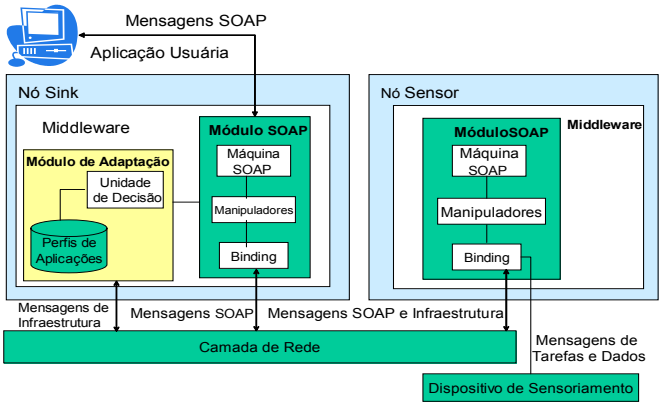

Fig. 1. Componentes dos Nós Sorvedouros e Nós Sensores.

O módulo SOAP roda em nós sorvedouros e sensores e é responsável pela composição e análise de mensagens XML e pela coordenação das operações necessárias para publicar e acessar os serviços do *middleware*. O módulo possui três componentes: a máquina SOAP, um conjunto de manipuladores e um *binding* com o protocolo subjacente. A máquina SOAP atua como o ponto de entrada no módulo SOAP. Ela é responsável por coordenar o fluxo das mensagens SOAP através dos vários manipuladores e por garantir que a semântica do protocolo SOAP é respeitada.

Manipuladores são os blocos de construção básicos do módulo SOAP e representam a lógica de processamento das mensagens. Quatro manipuladores foram definidos: básicos,

`Parse_Interest`, `Matching_Data` e `Matching_Aggregation`. Manipuladores básicos são responsáveis pela análise e composição de mensagens XML e pelo processamento de cabeçalhos. O `Matching_Data` é designado para o envio e a recepção de mensagens através de um protocolo subjacente baseado em interesses e centrado em dados. Ele realiza uma função de comparação entre dados gerados por sensores e interesses recebidos do usuário. O `Parse_Interest` realiza uma função de comparação entre as características de um sensor (tipo e localização) e um interesse recebido. O manipulador `Matching_Aggregation` serve para representar a ativação de funções de agregação dentro da rede. Uma função de agregação é disparada sempre que dados que correspondam a um conjunto de atributos especificados são gerados ou recebidos em um nó.

Bindings são responsáveis por encapsular mensagens SOAP na carga útil do pacote do protocolo de disseminação de dados subjacente. Há duas situações para o *binding*. Se o protocolo subjacente utilizar informações do conteúdo da mensagem para seu algoritmo de roteamento, o modo mais eficiente de *binding* é converter a mensagem SOAP para o formato adotado pelo protocolo (tipo 1). Se o protocolo transportar a carga útil sem tomar conhecimento de seu conteúdo, o *binding* apenas configura os campos necessários do cabeçalho e insere a mensagem SOAP diretamente no corpo do pacote (*binding* tipo 2).

3.2 Funcionamento do Sistema

O sistema proposto adota uma abordagem baseada em serviços. Uma aplicação usuária interessada em consultar dados de uma RSSF desempenha o papel de um solicitante de serviços. Nós sorvedouros atuam como fornecedores de serviços para o ambiente externo. Eles fornecem as descrições dos serviços oferecidos pela rede e oferecem acesso a esses serviços. Ao mesmo tempo, sorvedouros atuam como solicitantes de serviços para nós sensores, requisitando seus serviços especializados a fim de atender às necessidades das aplicações. Nós sensores são fornecedores de serviços. Eles enviam as descrições de seus serviços (tipos de dados fornecidos) para os sorvedouros, que mantêm um repositório com as descrições dos serviços de cada tipo de sensor existente na rede.

Aplicações submetem consultas (interesses) para a RSSF através de sorvedouros. Dois tipos de interação entre uma aplicação e a rede são possíveis: usuário-RSSF, através de uma interface gráfica, ou aplicação-aplicação. Na interação aplicação-aplicação, um *proxy* SOAP comunica-se diretamente com o programa usuário, convertendo chamadas de métodos na linguagem da aplicação para mensagens XML. Da mesma forma, o *proxy* converte mensagens XML de dados gerados pela rede para a linguagem da aplicação.

O sistema proposto funciona segundo quatro etapas principais, descritas a seguir.

Etapa 1: Comunicação da Configuração dos Sensores

Após a instalação da rede de sensores, os nós trocam mensagens SOAP que descrevem os seus serviços. Tais mensagens de configuração (`Publish_SensorDescription`) incluem o identificador do nó, o tipo do dispositivo sensor, localização geográfica, energia residual, graus de confiança máximo e mínimo e taxas de dados máxima e mínima. Mensagens de configuração têm que alcançar pelo menos um nó sorvedouro na rede. Um protocolo multi-hop é utilizado para encaminhar tais mensagens. É importante notar que etapa ocorre apenas uma vez, durante a inicialização da rede, e o protocolo usado não é ainda a solução otimizada para uma aplicação específica.

Etapa 2: Submissão de Interesses pela Aplicação

Aplicações interessadas em consultar os dados de uma rede de sensores podem ou não saber previamente quais os serviços oferecidos pela rede. Caso não saiba, a primeira mensagem trocada entre a aplicação e a rede visa realizar uma consulta sobre os serviços existentes. Essa mensagem é a Query_Sensors. Uma vez conhecendo esses serviços, a aplicação pode submeter seus interesses, através dos diferentes tipos de mensagens SOAP `Subscribe` (Seção 3.3).

Interesses podem ser classificados em *síncronos* ou *assíncronos*. Um interesse síncrono corresponde a uma operação simples de consulta ao estado atual de um fenômeno monitorado pela rede. Exemplo de um interesse síncrono seria: "qual a temperatura máxima da área X ?". Uma mensagem SOAP de interesse síncrono contém o tipo do sensor e a região geográfica alvo. Interesses assíncronos correspondem a consultas de longa duração ou a consulta sobre a ocorrência de algum evento específico. Um exemplo de consultas de longa duração é: "qual a temperatura média nas próximas 24 horas na área A ?". "Avise quando um elefante passar pela área A" é um exemplo de consulta sobre a ocorrência de evento. Mensagens SOAP de consultas de longa duração indicam o tipo do sensor, a área alvo, a duração da consulta e a taxa de aquisição. Mensagens SOAP de consultas sobre a ocorrência de eventos indicam a área, o tipo de sensor e o evento a ser monitorado. Aplicações podem ainda desejar que alguma tarefa seja realizada em resposta ao evento, por exemplo, ativar sensores especializados ou atuadores. Opcionalmente, aplicações podem requerer valores mínimos de precisão ou atraso.

Aplicações podem solicitar a execução de funções de agregação. A solicitação para a execução de uma função de agregação é uma descrição que contém um identificador e uma lista de tipos de dados com seus valores. O identificador é usado para disparar a execução da função apropriada no nó sensor quando tal nó recebe dados que combinem com os valores especificados na descrição.

Etapa 3: Estabelecimento do Perfil da Aplicação e Missão da Rede

Geralmente, o usuário da rede de sensores é especialista em seu domínio de aplicações e não em redes. Portanto, torna-se difícil para ele a escolha do protocolo ou da organização lógica da rede mais adequados a seus objetivos. Por isso, o *middleware* proposto oferece um processo de decisão automatizada, utilizando os parâmetros de entrada do usuário para enquadrar sua aplicação em um perfil e tentar prover a melhor solução (protocolo e topologia) para cada caso. A escolha das melhores soluções baseia-se na literatura e na realização de simulações e é continuamente refinada.

As informações extraídas da mensagem de interesse são utilizadas para selecionar um perfil de aplicação e estabelecer a missão da rede. Um interesse síncrono enquadra uma aplicação no modelo de entrega iniciado pelo emissor. Uma consulta de longa duração é melhor atendida pelo modelo de entrega contínuo, enquanto consultas sobre eventos específicos indicam um modelo de entrega baseada em eventos. Um interesse assíncrono pode especificar a necessidade de monitoramento contínuo e, ao mesmo tempo, a necessidade de ser notificado sobre algum evento específico. Tais requisitos apontam para o uso de um modelo de entrega híbrido. O tipo da consulta, o número de sensores na área alvo, o número estimado de sorvedouros e a taxa desejada de envio de dados são utilizados para decidir entre as seguintes estratégias de disseminação de dados/topologias: difusão direcionada modo *pull* [7]; difusão modo *push* [7]; *unicast* direta com o sorvedouro; *unicast*

com clusterização e hierarquia de 1 nível; *unicast* com clusterização e hierarquia de 2 níveis.

Atualmente, a unidade de decisão do *middleware* adota o algoritmo descrito na Figura 2.

```
Se interesse é Síncrono (consulta simples)
    Se a área alvo é próxima do Sorvedouro
        Se há muitos sensores na área alvo
            Usar difusão direcionada [10]
        Senão
            Usar unicast com comunicação direta com o sorvedouro
    Senão
        Usar difusão direcionada com encaminhamento geográfico (GEAR [19] )
Se interesse é Assíncrono
    Se é consulta de longa duração (envio de dados periódico)
     Se área alvo possui grande quantidade de nós
        Usar protocolo baseado em clusters
        Determinar número de clusters e número máximo de nós dentro de um cluster
        Se todos os líderes de clusters (cluster heads – CH) estão na área de alcance do sorvedouro
            Usar protocolo de hierarquia de um nível (CH transmite direto para sorvedouro)
        Senão
            Usar protocolo baseado em cluster com 2 ou mais hierarquias
     Senão
        Usar protocolo com topologia plana baseado em algoritmo guloso
    Se é detecção de eventos
     Se alta freqüência de envio de dados
        Se área alvo possui grande quantidade de nós
            Usar protocolo baseado em clusters
        Senão
            Usar difusão modo pull com encaminhamento geográfico
     Senão
        Usar difusão modo push
```

Fig. 2. Algoritmo de Decisão do *Middleware*.

As partes em negrito representam valores que devem ser obtidos através de simulações. Alguns parâmetros são explicitamente decididos pelo usuário ao submeter seus interesses, como o número de sorvedouros. As decisões tomadas pelo *middleware* são comunicadas ao protocolo de rede, que inicia a comunicação de infraestrutura necessária. Após a utilização das informações contidas na mensagem de interesse para decidir sobre a configuração da rede, essa mensagem é propagada para os nós sensores, de acordo com a estratégia de disseminação adotada.

Etapa 4: Envio dos Dados

Como o sistema proposto adota uma abordagem orientada a missão e baseada em interesses, um sensor somente envia mensagens de dados se houver recebido previamente uma mensagem anunciando interesses que combinem com seus dados.

Quando um sensor gera um dado, o dado é passado para o módulo SOAP via uma API baseada em XML. O manipulador responsável pela composição XML converte o dado em uma mensagem SOAP e a entrega para o `Matching_Data`. Que verifica se o dado gerado combina com algum interesse recebido. Caso combine, o dado é passado para o `Matching_Function`, que verifica se os atributos do dado combinam com os atributos

especificados para disparar alguma função de agregação. Em caso positivo, os dados são entregues para o manipulador que implementa a respectiva função de agregação. Estão previstos manipuladores para executar 5 tipos básicos de função de agregação: COUNT, MAX, MIN, AVERAGE, SUM. Os dados resultantes são passados para o protocolo de disseminação, via *binding*, como uma nova mensagem SOAP de dados.

Ao receber um pacote proveniente de outro sensor, o protocolo de disseminação verifica através de seu cabeçalho se o pacote é de dado ou controle. Caso seja um pacote de controle (comunicação de infraestrutura), ele é tratado pelo protocolo de disseminação de dados. Caso seja um pacote de dados, ele é passado para o módulo SOAP. Todos os pacotes de dados passam obrigatoriamente pelos manipuladores comuns, que identificam o tipo de *binding* utilizado e analisam o conteúdo do pacote. No caso em que mensagens SOAP são encapsuladas diretamente no pacote de dados do protocolo de disseminação, o analisador XML verifica o tipo da mensagem, isto é, se é de interesse ou de dado. Caso seja uma mensagem de interesse, ela é passada para o Parse_Interest, que verifica se o interesse recebido combina com as características do nó (local e tipo de sensor). Em caso afirmativo, armazena o interesse. Caso contrário, passa-o de volta ao protocolo de disseminação que irá tomar as decisões quanto a reencaminhá-lo ou não. Caso seja uma mensagem de dado, ela é passada para o Matching_data, que verifica se o dado combina com algum interesse recebido pelo nó. Em caso positivo, o dado é passado para o Matching_Function, que verifica se alguma função de agregação deve ser aplicada.

3.3 Operações e Mensagens SOAP

Na abordagem orientada a serviços, são definidas operações para cada fornecedor de serviços. De acordo com as operações definidas, mensagens são trocadas a fim de invocar essas operações. Os fornecedores de serviços do sistema proposto são os Nós Sorvedouros e os Nós Sensores. As operações definidas para Nós Sorvedouros são listadas abaixo.

`Query_Sensors` – usada por uma aplicação para saber os tipos de sensores disponíveis em uma WSN. Esta operação é representada por duas mensagens SOAP: uma mensagem de entrada sem parâmetros e uma mensagem de saída contendo a resposta a consulta.

`Subscribe_Asynch_Interest` - usada por uma aplicação para submeter uma consulta que representa um interesse assíncrono.

`Subscribe_Synch_Interest` - usada por uma aplicação para submeter uma consulta que representa um interesse síncrono.

`Subscribe_Hybrid_Interest` - usada por uma aplicação para submeter uma consulta que se enquadra no modelo de entrega de dados híbrido.

`Subscribe_Triggering_Interest` – usada por uma aplicação para submeter uma tarefa, por exemplo a ativação sob demanda de atuadores.

As operações definidas para Nós Sensores são apresentadas abaixo.

`Publish_SensorDescription` – usada por um nó sensor para criar e disseminar uma mensagem SOAP contendo suas descrições de serviços.

`Publish_Data` - usada por um nó sensor para criar uma mensagem SOAP de dados.

As mensagens SOAP utilizadas para invocar essas operações são mostradas abaixo.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
        <SOAP-ENV:Body>
            <m:QuerySensorIn xmlns:m="http://namespace.example.com">
                <parameter> </parameter>
            </m:QuerySensorIn>
        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Fig. 3. Mensagem SOAP Query_SensorsIn: mensagem de entrada para uma operação Quer_Sensors.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
        <SOAP-ENV:Body>
            <m:QuerySensorOut xmlns:m="http://namespace.example.com">
                <parameter>
                    <m:SensorType>Motion</m:SensorType>
                    <m:Confidence> <m:Max>1.0</m:Max>  </m:Confidence>
                    <m:DataRate unit="mSeconds"> <m:Max>10</m:Max>        </m:DataRate>
                </parameter>
            </m:QuerySensorOut>
        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Fig. 4. Mensagem SOAP Query_SensorsOut: mensagem de saída para uma operação Quer_Sensors.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
        <SOAP-ENV:Body>
            <m:PublishContent xmlns:m="http://namespace.example.com">
                <parameter ID="NODE_MAC_ADDRESS" NetworkID="NODE_NETWORK_ID">
                    <m:TTL unit="Seconds">3600</m:TTL> <m:SensorType>Motion</m:SensorType>
                    <m:DataDomain>
                        <m:Value>Four Legged Animal</m:Value>
  <m:Value>Two Legged Animal</m:Value>
                        <m:Value>Creeper Animal</m:Value>
                    </m:DataDomain>
                    <m:GeographicLocation unit="LatLong">
                        <m:x>35.00</m:x>  <m:y>-23.00</m:y>
                    </m:GeographicLocation>
                    <m:Energy unit="J">1</m:Energy>
                    <m:Confidence>
                        <m:Max>1.0</m:Max>    <m:Min>0.2</m:Min>
                    </m:Confidence>
                    <m:DataRate unit="mSeconds">
                        <m:Max>10</m:Max>     <m:Min>1000</m:Min>
                    </m:DataRate>
                </parameter>
            </m:PublishContent>
```

```
        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Fig. 5. Mensagem SOAP de configuração (usada para invocar a operação Publish_Content).

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope">
        <SOAP-ENV:Body>
            <m:PublishData xmlns:m="http://namespace.example.com">
                <parameter ID="NODE_MAC_ADDRESS">
                    <m:DataValue>Elephant</m:DataValue>
                    <m:Location unit="LatLong">
                        <m:x>35.00</m:x>  <m:y>-23.00</m:y>
                    </m:Location>
                    <m:Intensity>0.6</m:Intensity>
                    <m:Confidence>0.85</m:Confidence>
                    <m:Energy>0.9</m:Energy>
                    <m:TimeStamp>08:16:40</m:TimeStamp>
                </parameter>
            </m:PublishData>
        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Fig. 6. Mensagem SOAP de anúncio de dados (operação Publish_Data).

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" >
        <SOAP-ENV:Body>
            <m:SubscribeSynchInterest xmlns:m="http://namespace.example.com">
                <parameter>
                    <m:SensorType>Temperature</m:SensorType>
                    <m:Area>
                        <m:PointA unit="LatLong">
                            <m:x>35.00</m:x>  <m:y>-23.00</m:y>
                        </m:PointA>
                        <m:PointB unit="LatLong">
                            <m:x>35.02</m:x>  <m:y>-23.03</m:y>
                        </m:PointB>
                    </m:Area>
                    <m:Accuracy>0.1</m:Accuracy>
                    <m:MaxDelay unit="mSeconds">0.5</m:MaxDelay>
                    <m:Constraint>
                        <m:value>35.00</m:value>
                        <m:operation>GT</m:operation>
                    </m:Constraint>
                </parameter>
            </m:SubscribeSynchInterest>
```

```
        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Fig. 7. Mensagem SOAP anunciando um interesse síncrono.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" >
        <SOAP-ENV:Body>
            <m:SubscribeAsynchInterest xmlns:m="http://namespace.example.com">
                <parameter>
                    <m:SensorType>Temperature</m:SensorType>
                    <m:DataRate unit="mSeconds">20</m:DataRate>
                    <m:Duration unit="Seconds">20</m:Duration>
                    <m:Accuracy>0.1</m:Accuracy>
                    <m:Area>
                        <m:PointA unit="LatLong">
                            <m:x>35.00</m:x>  <m:y>-23.00</m:y>
                        </m:PointA>
                        <m:PointB unit="LatLong">
                            <m:x>35.02</m:x>  <m:y>-23.03</m:y>
                        </m:PointB>
                    </m:Area>
                    <m:Constraint>
                        <m:value>35.00</m:value>
                        <m:operation>GT</m:operation>
                    </m:Constraint>
                </parameter>
            </m:SubscribeAsynchInterest >
        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Fig. 8. Mensagem SOAP anunciando um interesse assíncrono (operação Subscribe_Asynch_Interest).

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" >
        <SOAP-ENV:Body>
            <m:InterestTrig xmlns:m="http://namespace.example.com">
                <parameter>
                    <m:Area>
                        <m:PointA unit="LatLong">
                            <m:x>35.00</m:x>  <m:y>-23.00</m:y>
                        </m:PointA>
                        <m:PointB unit="LatLong">
                            <m:x>35.02</m:x>  <m:y>-23.03</m:y>
                        </m:PointB>
                    </m:Area>
                    <m:mainSensor>
                        <m:SensorType>Motion</m:SensorType>
                        <m:DataRate unit="mSeconds">40</m:DataRate>
```

```
                    </m:mainSensor>
                    <m:DataType>Four Legged Animal</m:DataRate>
                    <m:secundarySensor>
                        <m:SensorType>Imagery</m:SensorType>
                    </m:secundarySensor>
                </parameter>
            </m:InterestTrig>
        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Fig. 9. Mensagem SOAP anunciando um interesse de ativação de tarefa (operação Triggering_Interest).

4. Trabalhos Relacionados

Em [3] é apresentada uma arquitetura para RSSFs que fornece um mecanismo para realização de consultas, monitoramento e tarefas. SINA desempenha o papel de um *middleware* que abstrai os nós da RSSF como uma coleção de objetos distribuídos. Usuários acessam informações dentro da RSSF usando consultas declarativas, ou comandam a execução de tarefas usando *scripts* programáveis. A principal diferença entre SINA e o sistema proposto é a adoção de uma abordagem baseada em serviços como modelo de programação e o uso da linguagem XML como linguagem de representação de consultas e tarefas. SINA adota a linguagem baseada em *scripts* SQTL [3] como interface entre aplicações e o *middleware*, enquanto nossa proposta adota padrões *Web de facto*, fornecendo assim um maior grau de interoperabilidade e independência para as aplicações.

O sistema proposto tem similaridades com [2,4,18], que são abordagens de bancos de dados para RSSFs. Em [18] as capacidades de processamento dos sensores são aproveitadas para executar parte do processamento de consultas dentro da rede, através de *query proxies*. Em [2], é proposta uma linguagem declarativa SQL-like para os usuários submeterem consultas para a RSSF. Em [4], é proposta uma arquitetura para RSSFs baseada nos conceitos de bancos de dados virtuais e roteamento centrado em dados. A principal diferença entre tais trabalhos e o sistema proposto é a abordagem baseada em serviços, totalmente distribuída e apoiada na adoção dos protocolos ubíquos SOAP e XML. Do ponto de vista das aplicações usuárias, expor as funcionalidades dos sensores como serviços leva a uma maior flexibilidade quando comparada com a utilização de consultas SQL.

Adicionalmente, de acordo com a literatura pesquisada, a presente proposta é a primeira que adota uma abordagem orientada a missão visando adaptar a configuração e o protocolo da rede aos requisitos das aplicações. [7] demonstra através de experimentos as vantagens obtidas por essa adaptação, dessa forma reforçando nossa proposta.

5. Avaliação do Sistema

Uma das principais vantagens do sistema proposto é facilitar o trabalho do desenvolvedor de aplicações para RSSFs, liberando-o de se preocupar com o software e a infraestrutura da rede. O usuário pode escrever sua aplicação na linguagem de alto nível de sua escolha e conectá-la sem impacto ao *middleware* da RSSF. A avaliação deste tipo de benefício não é possível de medir através de técnicas de simulação, mas deve ser levada em consideração.

Com relação aos ganhos no serviço oferecido pela rede, o trabalho [7] avalia três variações do algoritmo difusão direcionada (DD [10]) e mostra que a escolha da variação

mais adequada pode afetar o desempenho da aplicação em 40 a 60%. Esses ganhos podem ser extrapolados para escolhas mais genéricas entre diferentes protocolos de disseminação.

Como prova de conceito para o sistema proposto, o *middleware* foi implementado na linguagem Java utilizando a plataforma de desenvolvimento de Serviços *Web* Axis [17]. Os serviços para os nós sensores e sorvedouros foram criados como classes Java que implementam todas as operações descritas na Seção 3. A linguagem WSDL (*Web Services Description Language*) [13], uma extensão de XML específica para Serviços *Web*, foi usada para criar um documento descrevendo os serviços fornecidos pela RSSF. Aplicações que desejam utilizar a rede precisam apenas obter o documento WSDL que descreve os serviços fornecidos. Há inúmeras ferramentas [17] capazes de ler documentos WSDL e gerar automaticamente chamadas de métodos na linguagem da aplicação para invocar as operações disponíveis. Uma aplicação baseada em eventos foi implementada como um aplicativo Java que emite consultas para a rede e recebe os resultados. O objetivo da construção desse aplicativo foi validar a interação de alto nível entre aplicações e a rede de sensores, testando as invocações das operações e seu encaminhamento para os respectivos manipuladores SOAP. O código em Java e o documento WSDL criado podem ser obtidos em: `http://www.nce.ufrj.br/labnet/pesquisa/sensores/index.htm`.

A fim de provar a viabilidade de adotar o *middleware* proposto em uma RSSF real, é importante avaliar seu impacto no consumo de energia da rede. Como a maior fonte de consumo de energia em RSSFs é a transmissão de dados, avaliou-se o *overhead* de comunicação ao adotar o *binding* do tipo 2. Dependendo do formato de mensagem adotado pelo protocolo, o uso de XML pode acarretar um aumento no tamanho do pacote. Para avaliar o *overhead* causado por esse aumento, uma versão simplificada do *middleware* foi implementada no simulador de redes NS-2 [11], a qual consiste na máquina SOAP e nos manipuladores básicos. O LEACH [8] foi adotado como protocolo de disseminação. Ele não adota qualquer formato para representação de dados ou interesses e não usa o conteúdo das mensagens nas decisões de encaminhamento. O objetivo da simulação foi comparar a energia média dissipada na rede ao adotar XML/SOAP e ao adotar o tamanho de mensagens do LEACH original. A energia media dissipada é a razão entre a energia total dissipada por nó na rede e o número de eventos distintos entregues na estação-base e relaciona-se diretamente com o tempo de vida útil da rede [10] .

Foram simulados cenários com 50, 100, 150, e 200 nós em uma área inicial de 80x80m e crescendo proporcionalmente com o número de nós. As simulações apresentadas nos trabalhos com o LEACH descrevem dois tamanhos para mensagens de dados: 250 e 500Bytes. Uma mensagem SOAP de dados tem 260Bytes. A diferença entre a energia média dissipada usando tamanhos de mensagens de 250 e 260Bytes foi desprezível. Tamanhos de mensagens de 500B geraram aumento de 30% na energia dissipada em relação aos 260B. Há, no entanto, protocolos que adotam tamanhos de mensagens inferiores a 260 Bytes, e nesses casos o uso de XML pode gerar um aumento na energia dissipada. Uma forma de contornar esse problema é a adoção do formato XML binário compacto dentro da rede [15]. Esse formato reduz o tamanho das mensagens em até 50%, ao preço de uma menor portabilidade, o que não é problema dentro do ambiente controlado das RSSFs. Aplicações continuariam a adotar XML padrão.

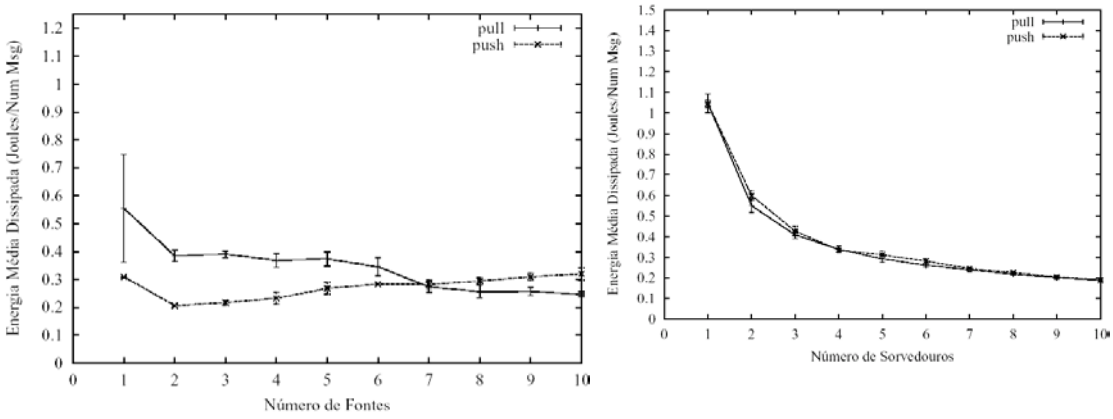

Fig. 10. Variação do Número de Nós Fontes e de Nós Sorvedouros (cenários com 50 nós)

Outra etapa da simulação consistiu em implementar um *binding* do tipo 1 para ser usado com o protocolo de difusão direcionada (DD [10]). O DD usa informações dos dados para configurar gradientes a partir dos quais os dados são encaminhados. O *binding* implementado converte mensagens XML para o formato adotado pelo DD, que consiste em vetores de atributos representados na linguagem C++, e vice-versa. Implementou-se a parte do algoritmo de decisão com a função de decidir entre duas variações do DD: *pull* ou *push*. No *pull*, interesses são enviados periodicamente por nós sorvedouros e os dados gerados são propagados em resposta a esses interesses. Como os interesses são inundados na rede, com um número grande de sorvedouros o protocolo tende a ter um alto *overhead* de comunicação. No *push*, interesses são publicados apenas localmente pelos nós sensores. Mensagens de dados são inundadas a uma taxa inicialmente baixa (dados exploratórios). Se uma mensagem de dado atinge um sorvedouro que tenha anunciado interesse pelo dado, são enviadas mensagens reforçando o caminho formado e os próximos dados são enviados a taxas mais altas. Sendo assim, os fatores que influenciam a escolha entre os modos *pull* e *push* são: número de sorvedouros, número de nós gerando dados (fontes) e taxa de envio de dados desejada pela aplicação. Esses parâmetros foram variados nas simulações e os resultados obtidos foram utilizados para refinar o algoritmo de decisão. Foram gerados cenários com 50 e 100 nós. No modelo de energia adotado a potência dissipada foi configurada para 35mW em modo ocioso, 395mW em modo de recepção e 660mW em modo de transmissão [10]. O alcance rádio foi configurado para 40m. As barras de erro mostradas nas figuras representam o intervalo de confiança de 95% dos resultados.

A variação do número de nós sorvedouros não mostrou diferença significativa entre os modos *pull* e *push* (Figura 10). Para números de fontes variando de 1 a 7, o modo *push* dissipou menos energia. A partir de 7 nós, o modo *pull* passou a apresentar os menores valores de energia dissipada (Figura 10). Para avaliar as duas variações do DD em função da taxa de dados, a taxa de envio das mensagens de interesse e de dados exploratórios foram mantidas conforme a implementação original do protocolo, respectivamente 30 e 60 segundos [7]. O tamanho das mensagens de dados é 64B e de interesses é 36B. Nessas condições, os modos *pull* e *push* dissipam a mesma quantidade de energia apenas em taxas altas de dados (1 dado a cada 5 seg). Com taxas mais baixas, o *pull* dissipa cerca de 25% a mais do que o *push*, ou seja, a rede trabalha mais para entregar informação útil para a aplicação (Figura 11). Como conclusão, o algoritmo de decisão do *middleware* deve basear sua escolha entre *pull* e *push* em dois fatores principais: (i) se há grande quantidade de nós

capazes de gerar dados de interesse para a aplicação, usar o modo *pull*; (ii) se a taxa de dados desejada pela aplicação for muito baixa, usar o modo *push*, ou ajustar as taxas de envio das mensagens de controle de acordo com a taxa de dados.

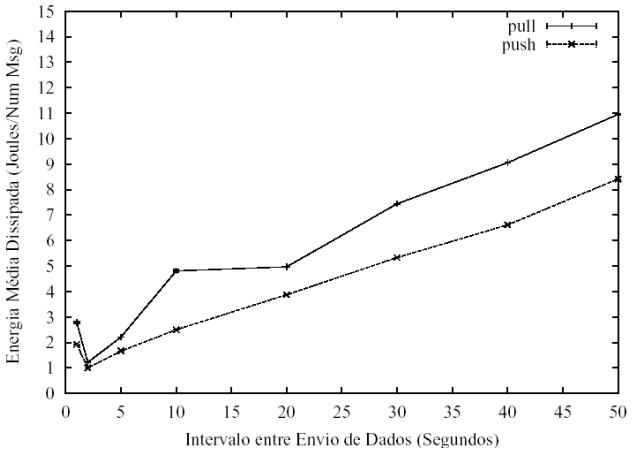

Fig. 11. Variação da taxa de dados (cenários com 100 nós)

É importante ressaltar que as principais vantagens do sistema proposto são obtidas quando mais de uma opção de protocolo estão disponíveis e a opção mais adequada é transparentemente escolhida, com base nas informações extraídas dos interesses submetidos pelo usuário. Tais benefícios serão avaliados com a implementação da versão completa do sistema e de diferentes protocolos de disseminação de dados. Trabalhos futuros apresentarão a avaliação dos ganhos em cada combinação entre aplicação e protocolo.

6. Conclusões

Este trabalho apresentou um sistema de *middleware* para redes de sensores sem fio. O sistema proposto adota uma abordagem baseada em serviços, permitindo a utilização de RSSFs de forma orientada a missões. As principais contribuições são listadas a seguir.

Primeiro, é proposta uma camada entre aplicações e a RSSF, que fornece ao usuário uma visão abstrata da rede como uma fornecedora de serviços. Essa visão é independente da infraestrutura e do protocolo subjacentes, e permite que diferentes aplicações utilizem a mesma rede. O fato das tarefas realizadas pelas RSSF serem bastante dependentes das aplicações tem levado ao projeto de sistemas com arquiteturas estáticas e customizadas para uma única aplicação alvo. Não é possível para um usuário externo, mesmo se ele puder interagir com a rede, utilizá-la de qualquer forma que não seja a pré-estabelecida pela aplicação para a qual a rede foi projetada. Isso gera sistemas essencialmente fechados para usuários e sistemas externos. Tal rigidez de operação e interação consiste em um obstáculo quando se considera que aplicações para RSSFs usualmente possuem longos ciclos de desenvolvimentos e frequentemente atendem a usuários com necessidades variáveis. Com o *middleware* proposto, a rede de sensores pode ser vista como uma entidade que fornece serviços para usuários transtórios com diferentes necessidades.

Segundo, é fornecido um mecanismo de decisão automatizada para estabelecer a melhor configuração da rede de acordo com a missão atual ditada pela aplicação. Esse mecanismo toma decisões sobre mecanismos de baixo nível que frequentemente estão fora do escopo

do desenvolvedor de aplicações usuárias. As decisões tomadas visam fornecer uma utilização eficiente da rede em termos de consumo de energia.

Terceiro, a linguagem XML e o protocolo SOAP são propostos como mecanismos para representar a comunicação da aplicação. Sua utilização viabiliza a interação com a RSSF via Internet, possibilitando acessá-la como um serviço *Web*, e permite a interoperabilidade de diferentes aplicações e diferentes redes.

Uma versão do sistema em Java foi implementada como prova de conceito do funcionamento da arquitetura proposta. Simulações para avaliar o comportamento de partes do sistema foram conduzidas e os resultados alcançados foram satisfatórios. Este trabalho dá um primeiro passo em direção a construção de redes de sensores genéricas, flexíveis, e ainda assim, eficientes em consumo de energia.

Referências

1. Bhattacharya, S., Abdelzaher, T.: Data Placement for Energy Conservation in Wireless Sensor Networks. Disponível em: http://www.andrew.cmu.edu/~weizhang/wsn/ documents/ fin_sagnik_journal.pdf.
2. Bonnet, P., Gehrke, J., Seshadri, P.: Towards Sensor Database Systems. In Proc. of the Second International Conference on Mobile Data Management. Hong Kong, Jan. 2001.
3. Chien-Chung Shen, Chavalit Srisathapornphat, And Chaiporn Jaikaeo - Sensor Information Networking Architecture and Applications," IEEE Personal Communications", Aug. 2001, pp. 52-59.
4. Govindan, R., Shenker, S., Hong, W., Hellerstein, J., Madden, S., Franklin, M.: The Sensor Network as a Database. Disponível em http://www-robotics.usc.edu/~gaurav/CS546.
5. Graham, S. et al.: Building Web Services with Java: Making Sense of XML, SOAP, WSDL, and UDDI. Sams Publishing, 2002.
6. He, T., Blum, B., Stankovic, J., Abdelzaher, T.: AIDA: Adaptive Application Independent Data Aggregation in Wireless Sensor Networks, ACM Transactions on Embedded Computing System Special issue on Dynamically Adaptable Embedded Systems, 2003.
7. Heideman, J., Silva, F., Estrin, D.: Matching Data Dissemination Algorithms to Application Requirements. ISI-TR-571. April 2003.
8. Heinzelman, W., Chandrakasan, A., Balakrishnan, H: Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In Proc. of the 33rd HICSS '00, Jan. 2000.
9. Heinzelman, W., Chandrakasan, A., Balakrishnan, H.: An Application-Specific Protocol Architecture for Wireless Microsensor Networks. IEEE Transactions on Wireless Communications, Vol. 1, No. 4, Oct. 2002, pp. 660-670.
10. Intanagonwiwat, C., Govindan, R., Estrin, D.: Directed diffusion: a Scalable and Robust Communication Paradigm for Sensor Networks. In Proc. of the ACM/IEEE MobiCom 2000 (56-67), Boston, MA, USA, Aug. 2000.
11. NS-2 (The Network Simulator versão 2). Disponível em: http://www.isi.edu/nsnam/ns/.
12. Tilak, S., Abu-Ghazaleh, N., Heinzelman, W.: A taxonomy of wireless micro-sensor network models (2002). Disponível em http://citeseer.nj.nec.com/tilak02taxonomy.html.
13. W3C (World Wide Web Consortium) Note, "Web Services Description Language (WSDL) 1.1". Disponível em: http://www.w3.org/TR/2001/NOTE-wsdl-20010315, March 2001.
14. W3C (World Wide Web Consortium) Recommendation, "Extensible Markup Language (XML) 1.0 (Second Edition)". Disponível em: http://www.w3.org/TR/REC-xml, Oct. 2000.
15. W3C Note, "WAP Binary XML Content Format". Disponível em: http://www.w3.org/TR/wbxml/ (1999).
16. W3C(World Wide Web Consortium) Note on Simple Object Access Protocol (SOAP) 1.1. Disponível em: http://www.w3.org/TR/SOAP/, May 2000.
17. Web Services Project Apache. Disponível em http://ws.apache.org/.
18. Yao, Y., Gehrke, J. : The Cougar Approach to In-Network Query Processing in Sensor Networks. Sigmod Record, Volume 31, Number 3, Sep. 2002.
19. Yu, Y., Govindan, R., Estrin, D.: Geographical and Energy Aware Routing: A Recursive Data Dissemination Protocol for Wireless Sensor Networks. UCLA Computer Science Department -TR-01-0023, May 2001.