Gerenciamento Auto-organizado de Serviços Distribuídos em Redes de Sensores Sem Fios

Tales Heimfarth[1], Alex G. C. de Sá[2], João C. Giacomin[1], Hewerton E. de Oliveira[1], Jesimar S. Arantes[1], Ariel F. F. Marques[1]

[1] Depto. de Ciência da Computação – Universidade Federal de Lavras (UFLA)
Lavras – MG – Brasil

[2]Depto. de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte – MG – Brasil

tales@dcc.ufla.br, alexgcsa@dcc.ufmg.br, giacomin@dcc.ufla.br,
hewerton@comp.ufla.br, jesimar@bsi.ufla.br, amarques@bsi.ufla.br

Resumo. *Este trabalho apresenta propostas de algoritmos de descoberta e execução de serviços distribuídos em Redes de Sensores Sem Fios (RSSFs). Uma missão contendo ações (sub-tarefas) que devem ser processadas é difundida para todos os nós da rede. Quando a missão é ativada pela ocorrência de um evento específico, um agente de dados é gerado contendo os dados lidos pelo sensor e um plano de processamento, extraído da missão. Esse plano indica quais serviços devem ser chamados para a execução correta das ações contidas na missão. Esses serviços estão distribuídos nos nós da rede. Enquanto o agente de dados migra na direção do nó concentrador, cada sub-tarefa contida no plano de processamento é executada em ordem tão logo seja encontrado o nó que contém o serviço requisitado. Assim, a cadeia de processamento do agente de dados é executada de maneira auto-montável e auto-organizada, reduzindo a complexidade da busca do serviço. Foram testados dois tipos de algoritmos: escolha local e escolha arbitrária, o quais executaram um plano de processamento com seis serviços. O algoritmo do paradigma de escolha local cumpriu o plano de processamento com 14 saltos, enquanto que o algoritmo de escolha arbitrária precisou de 42 saltos.*

Abstract. *This paper presents novel algorithms for distributed service discovery and execution in Wireless Sensor Networks (WSNs). In the proposed system, a mission with actions is diffused to all nodes. When the mission is triggered by an specified event, a data agent is generated with the data read by the sensors and a processing plan, coming from the mission. This processing plan indicates which services should be called for the correct execution of the mission (and in which order). The services are distributed among the nodes. Therefore, when the data agent travels towards the sink node, the tasks of the processing plan are spontaneously executed every time the right service is encountered in the path. The chain of the request's processing is executed in a self-mounted and self-organized manner, reducing the complexity of the service discovery. Two algoritms were experimented to assess the performance: the arbitrary choice and local choice. The local choice algoritm spent just 14 hops to perform a plan with six processing services. However, the arbitrary one spent 42 hops.*

1. Introdução

As Redes de Sensores Sem Fios (RSSFs) são compostas por um grande número de dispositivos autônomos denominados nós sensores. Um nó sensor é um sistema embarcado simples que possui um ou mais sensores conectados a ele. Esses nós podem ser espalhados em uma determinada região geográfica com o objetivo de monitorar as condições ambientais e outros eventos de interesse que nela ocorre [Levis et al. 2005]. As comunicações entre os nós da rede ocorrem normalmente através de pequenos rádios integrados, os quais têm alcance limitado a menos de 100 metros [Polastre et al. 2005]. Assim, as comunicações com o nó concentrador (*sink*) são realizadas por múltiplos saltos passando por vários nós intermediários. Além disso, a taxa de transmissão de dados é pequena, normalmente entre $20kbps$ e $250kbps$ [IEEE 2003].

As RSSFs possuem diversas aplicações, que podem ser agrupadas em classes conforme suas características. Duas das classes de aplicações mais comuns são a de monitoramento ambiental e a de detecção de eventos, cuja principal característica é a direção das comunicações que ocorrem preferencialmente no sentido de nós sensores para o nó *sink* [Li et al. 2007]. Em ambas as classes, os dados podem ser enviados em forma bruta para o *sink* para serem processados em um computador ou podem sofrer um pré-processamento local antes de serem enviados. O pré-processamento local nos nós é vantajoso por tratar a pouca disponibilidade de energia nos nós, considerando que a energia consumida no pré-processamento é largamente compensada pela economia de energia nas transmissões de informações [Estrin et al. 2002]. Além da escassez de energia, os nós sensores têm baixa capacidade de processamento e armazenamento, devido ao fato de utilizarem componentes eletrônicos reduzidos, de muito baixo consumo de energia e de baixo custo.

As restrições de *hardware* limitam a funcionalidade que cada nó sensor pode apresentar. Isso torna o trabalho colaborativo entre essas unidades necessário para o cumprimento de tarefas complexas. Uma tarefa complexa, também chamada de missão, é composta por um conjunto de atividades mais simples (ações), as quais são executadas por serviços instalados em diferentes nós sensores da rede. Cada nó da rede é especializado em um tipo de serviço, não sendo capaz de executar a missão por inteiro. Uma missão contém um plano de processamento, que é a entidade que determina a forma de combinar vários serviços para executar essa tarefa complexa.

O *Grubmi* [Sá et al. 2012] é um *middleware* baseado em agentes, serviços distribuídos e auto-organização que vem sendo desenvolvido nos últimos anos com objetivo de coordenar a cooperação dos diversos nós sensores da rede e oferecer abstrações adequadas que facilitem a programação de aplicações (missões). Nesse contexto, o *middleware* coordena uma missão ou um conjunto de missões. Para isso, ao ser inserida em um ponto arbitrário da rede, uma missão é disseminada para todos os nós da rede que devem possuir a capacidade de iniciar sua execução. Quando surgirem eventos que devem ser monitorados, o *middleware* coordena um nó sensor próximo a esse evento, fazendo ler seus sensores e extrair da missão o plano de processamento para esse tipo de evento, ou seja, a sequência de ações para processar o evento. Para executar uma ação da missão é necessário um serviço específico, que está presente em diversos nós da rede. Assim, o *middleware* executa a missão de forma distribuída, alocando ações presentes em seu escopo para os diferentes nós da rede. O *middleware* também provê os mecanismos de migração, execução e suporte de uma arquitetura de serviços distribuídos baseados em

fluxo de dados que permitem o processamento da missão.

O presente artigo tem como objetivo apresentar dois algoritmos inovadores de descoberta e execução de serviços distribuídos em RSSFs, denominados algoritmo de escolha local e algoritmo de escolha arbitrária. Diferente das abordagens tradicionais, onde serviços com estado são localizados através do uso de um *broker* ou inundação controlada, o presente trabalho propõe o uso de serviços de fluxo de dados sem estados, cuja procura (e execução) é feita de forma espontânea enquanto os dados são enviados para o concentrador (*sink*). Assim, os dados gerados em um nó sensor são automaticamente processados durante sua retransmissão. Os resultados são construídos enquanto os dados fluem na rede de sensores. Após a conclusão do processamento da missão, o resultado final é enviado ao nó concentrador utilizdo-se um algoritmo de roteamento conhecido.

O artigo é organizado da seguinte forma: trabalhos relacionados são introduzidos na Seção 2. Na Seção 3, uma visão geral do *Grubmi* é apresentada. A seguir, a arquitetura de serviços baseada em fluxo é introduzida. Resultados de simulações são relatados na Seção 5. Na última seção são apresentadas as conclusões e perspectivas para trabalhos futuros.

2. Trabalhos relacionados

Como as principais tarefas do gerenciamento de serviços apresentadas no presente artigo são a localização do serviço, sua execução e o repasse do resultado para a próxima etapa, foca-se nesta seção na tarefa de maior complexidade que é a descoberta do serviço. Os protocolos de descoberta de serviços para RSSFs podem ser divididos em três grandes grupos [Ververidis and Polyzos 2008]: arquiteturas baseadas em diretórios, arquiteturas sem diretórios e arquiteturas mistas.

Arquiteturas baseadas em diretórios podem utilizar diretórios centralizados (utilizado em redes infra-estruturadas) ou distribuídos. Os diretórios contêm uma tabela de endereços de fornecedores com seus respectivos serviços oferecidos. Os clientes sempre consultam o nó diretório antes de requisitarem os serviços desejados. Assim, a rede é hierarquizada, ou seja, é dividida em *clusters* e os nós diretórios são também chamados de *clusterheads*. Há um gasto extra de energia na montagem, na divulgação e na manutenção das tabelas de serviços. Em [Jung et al. 2008] é apresentado um mecanismo baseado em DHT (*Distributed Hash Table*) e clusterização. O protocolo define uma descrição do serviço, que é armazenada dentro do nó que o contém. Para esse protocolo, uma rede virtual é construída, considerando que alguns nós sensores específicos (*clusterhead*) cuidam de pequenas regiões com vários nós sensores e têm conhecimento de informações de serviços dessa região. Quando uma requisição de serviço é feita para algum propósito, estabelece-se uma busca local, via tabela *hash*, que determina o nó dentro do mesmo *cluster* que possui o serviço requisitado ou o nó *clusterhead* que sabe o nó que detém esse serviço. Para esse protocolo não é necessário um servidor central, pois o armazenamento da informação do serviço é feito localmente e isso não exige muito tráfego de rede. Entretanto, para utilizar o DHT, redes virtuais de nós sensores devem ser construídas durante a inicialização da rede.

Já arquiteturas sem diretórios são mais simples do que as baseadas em diretórios visto que não há necessidade de criação, manutenção e divulgação de tabelas. Os fornecedores de serviços enviam avisos através de *broadcasts* e os clientes enviam suas

requisições de serviços também através de *broadcasts*. Em alguns casos são usadas tabelas locais, em cada nó sensor, a fim de diminuir o volume de mensagens na rede. Isto conduz a outro problema, o de determinar o tempo que uma informação local permanece útil (*time-to-live*). Outro problema a ser tratado é determinar a frequência em que os provedores devem divulgar seus serviços para a rede, de forma a reduzir o tráfego de informações e evitar transmissões redundantes. Um representante importante das arquiteturas sem diretório é o Konark [Helal et al. 2003]. O Konark é um *middleware* projetado especificamente para descoberta e fornecimento de serviços em redes *ad hoc*, baseado no modelo *peer-to-peer*. Konark foi projetado para redes não hierarquizadas (não há formação de *clusters*) e utiliza uma arquitetura sem diretórios. Todos os nós da rede atuam como provedores e clientes de serviços. São utilizadas comunicações em *multicast* tanto para a descoberta de serviços quanto para o anúncio, quando os provedores enviam avisos para a rede sobre seus serviços. O uso de comunicação em *multicast* no lugar de *broadcast* é uma forma de diminuir o tráfego de informações na rede e poupar energia.

Embora a abordagem para a busca e processamento de serviços apresentada no presente artigo possa ser classificada como arquitetura sem diretório, ela difere em diversos pontos sobre as arquiteturas existentes. Primeiramente, como as demais propostas de arquitetura sem diretórios, o propósito geral da técnica apresentada neste trabalho é mais adequado ao contexto das RSSFs por não sofrer de problemas de escalabilidade e gasto de energia com organização (mensagens de controle). A presente abordagem não requer a criação de *clusters* nem a manutenção de diretórios distribuídos. Quando uma missão deve ser executada, um plano de processamento é criado com todas as ações que devem ser realizadas pela missão e o seu encadeamento. Essa missão será processada enquanto flui em direção ao nó *sink*.

Assim, nossa abordagem é massivamente distribuída, o que é mais apropriado para o contexto de RSSFs. Soluções sem diretório existentes na literatura também podem ser utilizadas em contexto onde centralização é desaconselhável. Porém, diferente da nossa, elas ainda dependem de dispendiosos anúncios de serviços na vizinhança e em mensagens de descoberta. Nossa abordagem propõe a descoberta espontânea dos serviços enquanto os dados e o plano de processamento são enviados para o nó *sink*. Isso parte da observação que, em muitas aplicações, depois de processados, os dados serão enviados para o concentrador, e a ideia aqui é processá-los enquanto isso ocorre. Não é necessário o conhecimento prévio da localização dos serviços.

3. Visão geral do *middleware Grubmi*

Nesta seção é apresentada uma visão geral do *middleware Grubmi*, que contém o algoritmo de processamento de serviços distribuídos que é apresentado no presente artigo.

A ideia principal de um *middleware* é oferecer abstrações e serviços que facilitem a reprogramabilidade da RSSF. No *Grubmi*, uma aplicação é programada como um conjunto de missões que são descritas em pequenos *scripts*. Esses *scripts* são chamados de planos de processamento, que definem o encadeamento de ações (ou chamadas a serviços) necessárias para o cumprimento de tais missões.

Para que a RSSF conheça as características da missão, esta tem de ser inserida em um ponto arbitrário da rede. Após isso, a missão migra automaticamente para todos nós sensores, através de uma inundação controlada. Desse modo, quando um evento

que é pertinente a uma missão é detectado por um nó, a missão do nó que o detectou tem seu processamento iniciado, extraindo o plano de processamento presente na missão para aquele tipo de evento. Esse processamento é auxiliado pela arquitetura de serviços distribuída existente na rede.

Um ponto que merece ser ressaltado é que quando o processamento da missão é realizado em um primeiro passo, a missão é automaticamente interpretada pela máquina virtual presente no *middleware*, iniciando o fluxo de execução de seu plano de processamento. Tudo acontece sem a interferência do usuário. Para que isso ocorra, uma missão é programada como a combinação de diversos tipos de chamadas de serviços (ações), sem focar em detalhes da disposição dos nós e da localização dos serviços. Com esse modelo distribuído foca-se na programação da rede como uma única entidade e evita-se um modo de programação focado no elemento básico da rede, o nó sensor. Essa forma individual e estática de tratar eventos em uma RSSF é geralmente utilizado em sistemas operacionais como o *TinyOS* [Levis et al. 2005], muito utilizado em RSSFs.

A Figura 1 apresenta uma visão geral do funcionamento do *middleware Grubmi*. Nessa figura, uma missão é inserida em um ponto arbitrário da rede pelo usuário, como pode ser visto na Figura 1 (a). A missão é inicialmente disseminada por toda RSSF, fazendo com que todos os nós conheçam-na. Essa disseminação é exemplificada na Figura 1 (b). Quando ocorrerem eventos pertinentes à missão, esta é ativada, iniciando seu processamento. Para isso, uma sequência encadeada de ações, que realizam chamadas de serviços distribuídos, são feitas. O processo de busca e execução dos serviços distribuídos até o nó *sink* é mostrado na Figura 1 (c). As setas tracejadas indicam o fluxo de processamento até o nó *sink* e as setas não tracejadas indicam as ações chamando serviços.

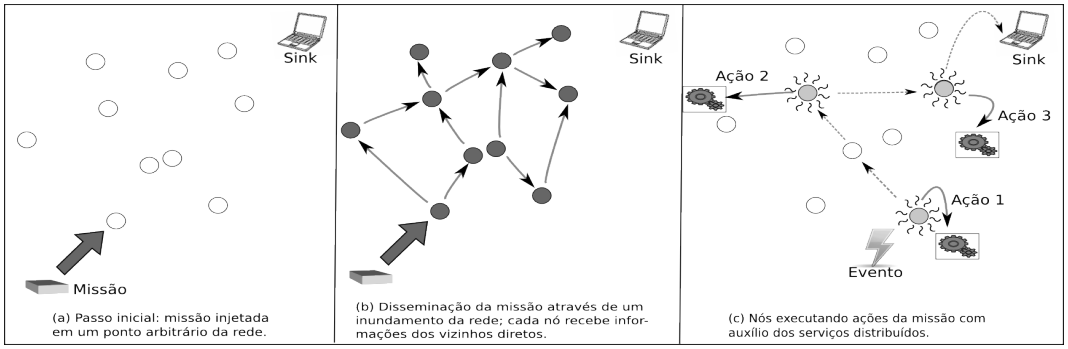

Figura 1. Visão geral do *middleware Grubmi*.

4. Serviços distribuídos baseados em fluxo de dados

O *middleware Grubmi* suporta diferentes tipos de serviços: $T = \{t_1, t_2, ..., t_n\}$. Assim, uma determinada instância de serviço $s \in S$ é de um determinado tipo. S é o conjunto de possíveis instâncias que um serviço pode ter. A função $Y(s) : S \rightarrow T$ faz o mapeamento de uma instância para o respectivo tipo. Para cada nó sensor existe uma quantidade Q de possíveis recursos destinados a esse nó. A função $R(t) : S \rightarrow Q$ determina a quantidade do recurso de memória que um determinado tipo de serviço necessita, realizando seu mapeamento adequado. Cada nó sensor tem uma quantidade limitada desse recurso, assim um nó pode disponibilizar certa quantidade de serviços. Neste trabalho, assume-se que cada nó tem exatamente uma instância de um tipo serviço.

Em arquiteturas de serviços tradicionais, identificam-se como requisitos importantes de gerenciamento: mecanismos de descoberta de serviços e o controle de migração dos serviços. Como será apresentado posteriormente nessa seção, a descoberta de serviços é feita de forma automática enquanto os dados navegam na rede (chama-se aqui de serviços auto-montáveis). Já o controle de posicionamento/migração não será tratado no escopo do artigo.

Conforme já dito, a missão contém o plano de processamento que deve ser executado na ocorrência do evento gatilho (por exemplo, uma medição de temperatura pode iniciar o processamento da missão). Um detalhe importe é que esse plano não pode ser processado somente no nó inicial. Assim, o plano de processamento juntamente com os dados a serem processados devem migrar pela rede a procura dos serviços até o nó *sink*.

Esse plano contém toda a sequência de serviços que deve ser invocada, inclusive representando as dependências entre o processamento dos serviços sobre os dados. Para isso, um grafo $G(V, E)$ orientado e acíclico é utilizado. Os vértices do grafo representam os serviços, já as arestas representam as relações de dependência. Assim, a tupla formada pelos dados e pelo plano de processamento será chamada doravante de agente de dados. A missão inicial, quando ativada, gera um agente de dados que migra para rede executando de forma distribuída o plano de processamento, até chegar no nó *sink*.

O processamento inicia quando os dados do sensor ativam a missão. Após a leitura dos sensores, o *middleware* extrai da missão, um plano de processamento para esse tipo de evento, e envia esse plano junto com os dados (agente de dados) para algum nó sensor vizinho. A decisão sobre qual vizinho deve continuar o processamento será apresentada posteriormente. Ao final, o resultado do processamento é encaminhado para o nó *sink*.

Durante a migração do agente de dados pela rede de sensores, os nós que contêm serviços requisitados no plano de processamento são ativados de forma espontânea, ou seja, processam os dados de algum modo. Dessa forma, após cada nó receber o agente, este verifica se o nó contém o próximo serviço requisitado. Se contiver o serviço requisitado ele será processado, caso contrário o agente é simplesmente passado adiante.

Cada vez que um serviço é executado, o plano de processamento é alterado para indicar que mais uma etapa (ação ou sub-tarefa) está finalizada. Após esse resultado, o agente de dados é novamente mandado adiante. É importante ressaltar a completa ausência de um mecanismo de descoberta de serviços no estilo clássico. Isso aumenta a robustez do sistema e facilita a auto-organização da rede.

Além disso, é possível estabelecer uma quantidade de saltos máxima do agente de dados, que é decrementada a cada salto do agente sobre o nó sensor. Dessa forma, são evitados agentes que circulam por um grande período pela rede.

Uma questão adicional que não será tratada no presente artigo é a distribuição dos serviços nos nós sensores da rede. Cada nó sensor comum contém uma instância s de tipo $t \in T$ de serviço. A escolha do tipo de serviço instalado em cada nó da rede foi feita de forma aleatória, onde para n tipos, cada tipo tem uma probabilidade de $\frac{1}{n}$ de estar presente em um dado nó.

Podemos identificar duas questões adicionais que serão tratadas nas próximas subseções:

- Como é feita a escolha do próximo nó sensor enquanto o agente está sendo processado?
- Qual mecanismo será usado pelo agente na sua migração para o concentrador após a fase de processamento ter sido encerrada?

A seguir são apresentadas metodologias empregadas para responder a essas questões.

4.1. Metodologias para formação da cadeia de processamento

Nesta subseção são apresentados os algoritmos baseados em escolha arbitrária e local, utilizadas na escolha do próximo salto na cadeia de processamento.

4.1.1. Algoritmo de escolha arbitrária

O algoritmo de escolhas arbitrárias faz uso de uma distribuição de probabilidade uniforme como base para efetuar os saltos de processamento. Desse modo, o algoritmo inicia suas funções assim que um evento seja sensoriado por algum nó, desencadeando a execução de uma missão. Nesse instante o nó executando a missão gera um agente de dados que será repassado para a rede. Além dos dados recolhidos, o nó é auxiliado pelo *middleware*, que realiza a cópia do plano de processamento presente na missão para o agente de dados. O plano geralmente possui tarefas que são dependentes umas das outras para a execução completa do mesmo. A ideia proposta do gerenciamento de serviços sobre uma RSSF é que o processamento seja feito de forma distribuída e que ao fim do problema, ou seja, o fim do processamento das tarefas, a rede envie os resultados para o nó *sink*.

A escolha do próximo salto na cadeia de processamento é feita de forma estritamente aleatória entre os nós que levem o agente para mais perto do *sink*: um nó com m vizinhos cujas distâncias do *sink* sejam menores que a sua escolhe um desses vizinhos para repassar o agente. Cada vizinho tem a probabilidade de $\frac{1}{m}$ de ser escolhido.

É importante destacar que com essa estratégia existe uma grande probabilidade do vizinho escolhido não conter o serviço requisitado no plano de processamento. Se esse for o caso, o agente é novamente repassado, utilizando-se da mesma estratégia. Assim, o agente descreve uma caminhada aleatória (*random walk*) pela rede (mas sempre se aproximando do *sink*) em busca de serviços necessários para o processamento de seu plano.

Para redes de pequena dimensão, onde em poucos saltos pode-se alcançar o *sink*, opta-se por uma caminhada aleatória sem o direcionamento para o concentrador, sendo assim todos os vizinhos têm a mesma probabilidade de serem escolhidos.

A Figura 2 (a) mostra um exemplo. Nela pode ser visualizado o retângulo que delimita a área de sensoriamento e os círculos que representam nós sensores sem fios. Nesse exemplo o plano de processamento possui três ações (sub-tarefas). O agente de dados é processado de modo estocástico, ou seja, quando ocasionalmente o agente de dados atinge um nó contendo o próximo serviço especificado pelo plano, esse processamento é realizado. Quando a última tarefa do plano é processada, o resultado final é enviado para o nó *sink*.

Na Figura 2 (a), o sinal em forma de raio representa o evento sensoriado. A letra

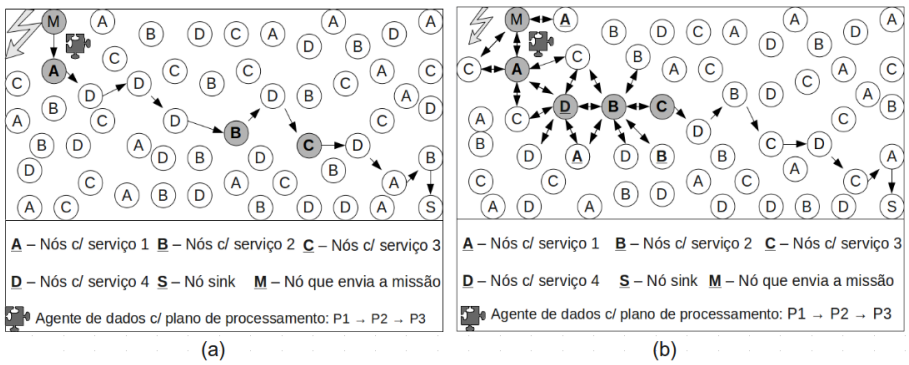

Figura 2. (a) Algoritmo de escolha arbitrária; (b) Algoritmo de escolha local.

"M" em um dos nós significa que esse foi o que iniciou o processo com o sensoriamento do evento, enviando o agente aos nós da RSSF. Os nós com as letras "A", "B", "C" e "D" indicam a sequência de serviços t_1, t_2 t_3 e t_4. "P1", "P2" e "P3" denotam o processamento das ações pelos serviços t_1, t_2 e t_3, respectivamente. O nó denotado com a letra "S" define o nó *sink*. As setas na figura mostram o fluxo de transmissão de dados até o nó *sink*.

4.1.2. Algoritmo de escolha local

O algoritmo de escolha local utiliza uma abordagem de busca local para localizar o próximo serviço do plano de processamento. Para isso, ele envia informações aos nós sensores de sua vizinhança, via *broadcast*, perguntando a disponibilidade do serviço requerido no momento no plano de processamento. Após o recebimento das respostas advindas desses vizinhos, o nó executa as seguintes tarefas:

- Verifica se algum dos nós sensores vizinhos possuem a instância do serviço requisitado para cumprir item atual do plano de processamento. Caso um ou mais vizinhos com o serviço existam, o vizinho mais próximo do *sink* é escolhido como próximo destino do agente de dados. Caso contrário, a escolha é arbitrária como apresentado na seção anterior.
- Repassa o agente de dados para o nó escolhido.

Esse agente de dados é a entidade escolhida para repassar os dados e o plano de processamento ao outro nó, caso o nó sensor não possua o serviço requisitado ou já tenha processado sua parte do plano de processamento.

A Figura 2 (b) exibe a execução do algoritmo para um plano de processamento de tamanho igual a três. Da mesma forma, o algoritmo inicia sua função com a detecção do evento (figura em forma de raio) pelo nó "M". Assim que detectou o evento, um plano de processamento é criado para tratar o dado sensoriado. Uma mensagem de requisição de serviço é enviada para os nós vizinhos perguntando quem detém o serviço necessário para suprir a primeira ação do plano. Se algum vizinho tiver a instância desse serviço, esse será escolhido como próximo salto do agente de dados.

Isso é visto na Figura 2 (b). As setas determinam a troca de mensagens entre os nós sensores e "P1", que indica que esse nó vizinho tem o serviço, fazendo o devido processamento do agente de dados. Caso os nós vizinhos não possuam o serviço, o agente

de dados é enviado para o nó escolhido arbitrariamente, representado na figura por uma letra sublinhada.

Após o resultado final ser gerado, um fluxo até o *sink* é observado. Dois algoritmos de roteamento foram utilizados para enviar os resultados ao nó *sink*. O primeiro foi o *Nearest Closer* (NC) [Stojmenovic 2002], que se utiliza de um roteamento geográfico (Distância Euclidiana) para verificar o custo para o envio de mensagens. E o segundo algoritmo faz uso de uma árvore de difusão que é semelhante à utilizada no *Direct Diffusion* [Al-Karaki and Kamal 2004]. Essa árvore cria gradientes de informação em direção aos nós que tem uma característica específica, nesse caso o *sink*, para realizar um fluxo direcionado para o mesmo.

5. Resultados e discussões

Esta seção apresenta todas as configurações utilizadas para os testes e em seguida os resultados. Juntamente com os resultados é feita uma breve discussão analisando as abordagens propostas.

Os experimentos foram realizados utilizando-se o *GrubiX*, um simulador orientado a eventos discretos derivado do simulador *ShoX* [Lessmann et al. 2008]. Para os testes, a quantidade de nós foi de 250: um nó do tipo concentrador, um nó detector de eventos e 248 nós comuns, com seis tipos diferentes de serviços implementados. Considere que cada nó sensor tem $\frac{1}{6}$ de um único tipo de serviço em sua estrutura. Os nós sensores foram distribuídos aleatoriamente em uma área de $250m$x$250m$. Foi utilizado o modelo de propagação *Free-Space* [Giacomin et al. 2010], com uma distância máxima de transmissão de $30m$. O valor da densidade dos nós de acordo com a área é igual a 0,004 nós/m^2. Com essas configurações, cada nó tem, em média, 11,31 nós vizinhos e geram uma probabilidade da rede estar completamente conectada de 99,69% [Bettstetter 2002]. Assim, cada algoritmo foi executado (simulado) 100 vezes dentro do ambiente *GrubiX* para obtenção de resultados estatísticos confiáveis.

Para os gráficos da Figura 3, as legendas possuem as seguintes definições:

- **Arbitrário1** - O algoritmo utiliza escolha arbitrária para tratar o dado sensoriado e a árvore de difusão para o envio da informação para o nó concentrador.
- **Arbitrário2** - O algoritmo usa escolha arbitrária, mas para o envio dos resultados para o nó *sink*, utiliza o roteamento geográfico (*Nearest Closer*).
- **Local1** - Faz uso do algoritmo com escolha local para processamento e da árvore de difusão para o envio dos resultados finais para o nó *sink*.
- **Local2** - Utiliza o *Nearest Closer* ao invés da árvore de difusão em relação ao método *Local1*.

Para os gráficos da Figura 3 (a) e da Figura 3 (b), que apresentam resultados estatísticos baseados em médias, são considerados intervalos de confiança de 95% para validar os resultados dos mesmos. Isso foi feito para medir a precisão dos algoritmos propostos, ou seja, 95% das vezes que o algoritmo for executado apresentará esse resultado, considerando as margens dos intervalos de confiança.

O gráfico da Figura 3 (a) apresenta os resultados da média da quantidade de saltos necessária para que um plano de processamento com seis ações (sub-tarefas) seja processada pela RSSF. A ordem de execução do plano foi escolhida aleatoriamente e o limite de

saltos para o processamento do mesmo foi de 50 saltos para tentar evitar agente de dados que nunca saem da rede. Caso o algoritmo não complete a execução do plano, resultados parciais são enviados ao nó *sink*. Além do gráfico da Figura 3 (a) expor a quantidade de saltos para o processamento do plano, ele também inclui a quantidade de saltos para o envio para o nó *sink*.

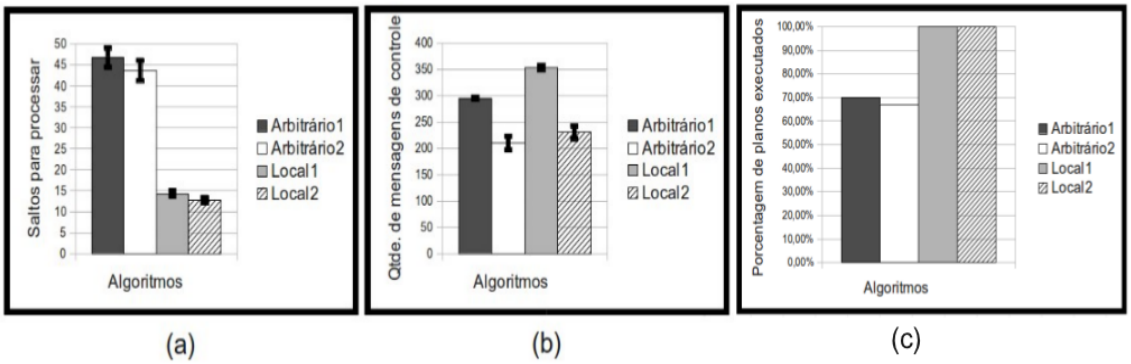

Figura 3. (a) Saltos p/ processar; (b) Mensagens de controle; (c) Porcentagem de planos executados.

Com os resultados presentes na Figura 3 (a), analisa-se que o algoritmo que possui escolha local gasta uma quantidade de saltos inferior para processar e, em seguida, enviar os resultados para o nó concentrador. Isso é justificado pela tomada de decisão do algoritmo de escolha arbitrária, que apenas envia o plano de processamento para o próximo vizinho sem realizar uma busca local, diferente do algoritmo de escolha local, o qual escolhe um vizinho para enviar o plano caso esse vizinho tenha o serviço para processar o primeiro item do plano. Somente em último caso, o plano é enviado para outro vizinho, escolhido arbitrariamente.

É considerada na Figura 3 (b) a média total de mensagens de controle que o algoritmo utiliza para atingir seu objetivo. Pode ser observado através desse gráfico que a utilização de um *flooding* inicial para realizar o algoritmo de roteamento da árvore de difusão aumenta a troca de mensagens entre os nós da RSSF. Esse fator torna desfavorável o uso da árvore de difusão em relação ao roteamento geográfico.

Uma observação importante é que considerou-se nos experimentos que a missão já estava, no momento inicial, presente em todos os nós da rede, não sendo necessárias mensagens para difundi-la. Isso porque procurou-se medir somente o gasto de comunicação com a busca e execução dos serviços distribuídos.

A Figura 3 (a) e Figura 3 (b) simplificam uma comparação de energia entre as abordagens propostas. A energia é avaliada de acordo com número de mensagens trafegadas na rede. Essas mensagens ativam o rádio do nó sensor, responsável em grande parte pelo consumo de energia do nó. É um modelo adequado para representar o consumo de energia dos algoritmos, visto que esses algoritmos são independentes da camada de acesso ao meio (MAC) utilizada. Contudo, esse modelo será melhorado futuramente.

A Figura 3 (c) expõe a porcentagem de planos de processamento realizados para as 100 execuções de cada algoritmo, ou seja, das 100 execuções do algoritmo, quais não concluíram as requisições dos serviços solicitadas pelo plano de processamento. Esse gráfico avalia também qualidade dos métodos utilizados, ou seja seu desempenho. Quanto maior o seu valor (com limite igual a 100%), maior é a qualidade do método.

Com resultados contidos nos gráficos acima, reflete-se que as abordagens que utilizam o algoritmo com escolha local sempre conseguem terminar a missão com um plano de sensoriamento, o que as deixam como primeira opção quando é analisada como fator de escolha qualidade do processamento do evento e seu respectivo envio para o *sink*. E para completar a análise feita, é visto que quando está sendo verificada a quantidade de mensagens de controle trocadas entre os nós, fica visível que as abordagens que se utilizam roteamento geográfico (*Nearest Closer*) são superiores em relação às de fluxo direcionado (árvore de difusão) para rotear o dado. Contudo, o custo do nó que apresenta GPS em sua composição é maior que o nó comum. Portanto, se o custo for um fator limitante da aplicação a ser construída, abordagens que utilizam roteamento geográfico devem ser descartadas.

6. Conclusões e trabalhos futuros

Esse trabalho apresentou uma nova metodologia de busca e execução de serviços distribuídos baseados em fluxo de dados. Quando dados são gerados em um determinado sensor, devem ser primeiro processados para depois serem enviados ao nó concentrador. Esse processamento é feito de maneira distribuída utilizando diversos serviços que são instalados nos diferentes nós da rede. No lugar de utilizar-se de técnicas convencionais baseadas em busca de serviço, os dados juntamente com as tarefas que devem ser aplicadas sobre eles (agente de dados contendo um plano de processamento) são transportadas pela rede, aproximando-se do nó concentrador. Quando um serviço requisitado está disponível em um determinado nó onde o agente se encontra, os dados são processados. Após todas as etapas do plano de processamento serem executadas, o resultado final é enviado ao nó concentrador.

Os dois algoritmos (local e arbitrário), com variações em seus algoritmos de roteamento, foram simulados no ambiente *GrubiX*. O algoritmo que utiliza o paradigma de escolha local mostrou-se superior ao que se utiliza do paradigma de escolha arbitrária no quesito número de passos (*hops*) necessários para completar o plano de processamento (aproximadamente 14 *hops* foram necessários no algoritmo de escolha local contra 42 do algoritmo de escolha arbitrária). Podemos justificar pelo fato da probabilidade de encontrar o serviço necessário (presente no plano) ser maior quando a vizinhança local é consultada. A utilização do roteamento geográfico para o encaminhamento dos resultados para o nó concentrador também reduziu a quantidade de saltos para o processamento.

Como trabalho futuro, pretende-se analisar de forma analítica o comportamento da metodologia proposta. Além disso, para aumentar o desempenho dos algoritmos apresentados e diminuir o número de saltos necessários para execução o plano de processamento (custo), planeja-se combinar a proposta desse artigo com uma estratégia de migração dos serviços. Serviços muito requisitados em uma determinada área serão atraídos para esse local com a utilização de feromônios digitais para esse objetivo.

Agradecimentos

Os autores desse artigo agradecem à Fundação de Amparo à Pesquisa do estado de Minas Gerais (FAPEMIG) e ao Ministério de Ciência e Tecnologia/Conselho Nacional de Desenvolvimento Científico e Tecnológico (MCT/CNPq) pelo apoio financeiro parcial ao projeto de pesquisa.

Referências

IEEE 802.15.4 - wireless medium access control (mac) and physical layer (phy) specifications for low-rate wireless personal area networks (lr-wpans), 2003. Acessado em 10 de março de 2012.

Al-Karaki, J. N. and Kamal, A. E. (2004). Routing techniques in wireless sensor networks: a survey. *IEEE Wireless Communications*, 11(6):6–28.

Bettstetter, C. (2002). On the minimum node degree and connectivity of a wireless multihop network. In *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, MobiHoc '02, pages 80–91, New York, NY, USA. ACM.

Estrin, D., Sayeed, A., and Srivastava, M. (2002). Wireless sensor networks ? tutorial. In *Proceedings of MobiCom 2002, the eighth ACM International Conference on Mobile Computing and Networking*, Atlanta, Georgia, USA.

Giacomin, J. C., Correia, L. H. A., Heimfarth, T., Pereira, G. M., Silva, V. F., and de Santana, J. L. P. (2010). Radio channel model of wireless sensor networks operating in 2.4 ghz ism band. *INFOCOMP - Journal of Computer Science*, 1:98–106.

Helal, S., Desai, N., Verma, V., and Konark, C. L. (2003). A service discovery and delivery protocol for ad hoc networks. In *Proc. 3rd IEEE Conference on Wireless Communication Networks (WCNC)*.

Jung, J., Lee, S., Kim, N., and Yoon, H. (2008). Efficient service discovery mechanism for wireless sensor networks. *Computer Communications*, 31(14):3292 – 3298.

Lessmann, J., Heimfarth, T., and Janacik, P. (2008). Shox: An easy to use simulation platform for wireless networks. In *Proceedings of the Tenth International Conference on Computer Modeling and Simulation*, pages 410–415, Washington, DC, USA. IEEE Computer Society.

Levis, P., Gay, D., and Culler, D. (2005). Active sensor networks. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2*, NSDI'05, pages 343–356, Berkeley, CA, USA. USENIX Association.

Li, Y., Thai, M. T., and Wu, W. (2007). *Wireless Sensor Networks and Applications (Signals and Communication Technology)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Polastre, J., Szewczyk, R., and Culler, D. E. (2005). Telos: enabling ultra-low power wireless research. In *IPSN*, pages 364–369.

Stojmenovic, I. (2002). Position-based routing in ad hoc networks. *Communications Magazine, IEEE*, 40(7):128 –134.

Sá, A. G. C., Heimfarth, T., and Hewerton E. Oliveira, E. P. F. (2012). An algorithm to coordinate missions in wireless sensor networks. *IEEE Latin Americ Transactions*, 10:1595–1602.

Ververidis, C. N. and Polyzos, G. C. (2008). Service discovery for mobile ad hoc networks: A survey of issues and techniques.