Maximizando a Vazão Através de Múltiplos Caminhos em Plataformas com Dois Rádios

Nildo dos Santos Ribeiro Júnior,
Marcos A. M. Vieira, Luiz F. M. Vieira

[1]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais – Belo Horizonte, MG – Brasil

{nildo,mmvieira,lfvieira}@dcc.ufmg.br

Abstract. *Applications for Wireless Sensor Networks are requiring higher throughput. Platforms with two radios were proposed to achieve it, conserving the energy efficiency. There is a bulk-data transfer protocol optimized for dual-radio platforms proposed in the literature, but it doesn't use all the hardware resources available. In this work we propose a new multi-hop bulk-data transfer protocol for wireless networks that simultaneously uses two disjoint paths, using all available radios. The protocol was implemented using TinyOS and evaluated in the physical world. We compare our proposal with FastForward, the state-of-the-art protocol for dual-radio. Our approach doubled the throughput getting very close to the maximum theoretical limit value.*

Resumo. *Aplicações para Redes de Sensores Sem Fio estão requerendo maior vazão. Foram desenvolvidas plataformas com dois rádios para aumentar a vazão conservando a eficiência energética. Há uma proposta de protocolo de transferência de dados em massa otimizado para essas plataformas, porém ele não aproveita todo o recurso de hardware disponível. Nesse trabalho propomos um protocolo de transferência massiva de dados em vários saltos para redes sem fio que utiliza dois caminhos distintos simultaneamente, aproveitando os dois rádios disponíveis nos nós. O protocolo foi implementado no TinyOS e avaliado experimentalmente no mundo físico. Comparamos nossa proposta com o FastForward, o estado da arte. Nossa abordagem dobrou a vazão chegando próximo ao limite máximo teórico.*

1. Introdução

As Redes de Sensores Sem Fio (RSSFs) consistem em conjuntos de nós sensores que agem tanto como coletores de dados quanto como retransmissores de pacotes na rede. Cada nó contém um ou mais sensores, um microprocessador e um ou mais transceptores de rádio. De acordo com [Akyildiz and Vuran 2010], a comunicação sem fio provê a vantagem de facilidade de implantação, e a capacidade de sensoreamento distribuído faz com que RSSFs tenham potencial para serem um importante componente na nossa vida diária. Elas têm muitas aplicações, como em monitoração ambiental, na agricultura, em assistência médica e em construções inteligentes.

O projeto de uma RSSF depende significativamente da aplicação. O ambiente, os princípios de design da aplicação, o hardware e as restrições de projeto podem ser muito diferentes em cada cenário [Yick et al. 2008]. Por exemplo, o ambiente é importante para

determinar o tamanho, o esquema de implantação e até mesmo a topologia da rede. Porém quando estamos falando sobre o projeto de hardware dos nós sensores, há dois problemas chave levados em consideração: o custo e o consumo de energia.

Como as RSSFs geralmente consistem em uma grande quantidade de nós sensores, minimizar o custo de cada unidade é muito importante para que o custo total da rede não seja muito elevado. Essa restrição faz com que a maioria das plataformas de RSSFs tenham muito pouco poder de processamento e memória. Como um nó sensor geralmente é alimentado por pilhas ou baterias, o consumo de energia deve ser minimizado para prolongar a vida útil deles. Para economizar energia, deve-se minimizar a quantidade de transmissões sem fio, já que o rádio consome uma quantidade muito grande de energia comparada ao resto dos componentes da arquitetura de hardware.

Minimizar o custo e o consumo de energia dos nós sensores faz com que o desempenho da rede seja reduzido. Esse compromisso entre custo e desempenho é muito importante no design de RSSFs. De acordo com [Jurdak et al. 2011], o design de plataformas tradicionais de RSSFs favoreceram o baixo consumo de energia ao custo da vazão ou alcance da rede. Isso fez sentido no contexto em que se deu o início do desenvolvimento de RSSFs, onde a maioria das aplicações era de coletar pequenas quantidades de dados, como valores de temperatura ou iluminação.

Hoje em dia, aplicações também estão sendo desenvolvidas para coletar dados de som e vídeo, que têm uma demanda por alta vazão na rede. Então embora o baixo consumo de energia ainda seja importante no design de RSSFs, a vazão tem ganhado importância nesses novos tipos de aplicações. Levando ainda em consideração que novas tecnologias de captação de energia têm sido desenvolvidas, o projeto das plataformas pôde passar a priorizar um pouco mais outros fatores, além do consumo de energia. Uma boa opção é priorizar a eficiência energética na transmissão, que é a quantidade de energia utilizada para transmitir uma certa quantidade de dados. Dessa maneira, nós podemos ter aplicações com maior vazão que poderão consumir mais energia no total, porém conservando a eficiência energética do sistema, pois são capazes de transmitir uma quantidade muito maior de dados.

Essa mudança de paradigma motivou pesquisadores a desenvolver novas plataformas para RSSFs, com o objetivo de prover maior vazão continuando energeticamente eficiente. Uma técnica que permite alcançar esse objetivo é prover diversidade de rádio, que significa ter mais de um rádio em cada nó sensor. Se cada rádio opera em uma banda diferente, eles não vão interferir um com o outro enquanto estão comunicando, reduzindo a perda de pacotes devido à interferência e permitindo transmissões simultâneas entre nós sensores. De acordo com [Kusy et al. 2011], a diversidade de rádio pode melhorar a taxa de entrega, a estabilidade da rede e os custos de transmissão por um pequeno aumento no custo energético de um único rádio.

Nesse artigo, apresentamos um protocolo de transferência massiva de dados em vários saltos para RSSFs, projetado para plataformas que possuem dois rádios operando em frequências diferentes. O protocolo utiliza dois caminhos distintos para fazer a comunicação entre dois nós na rede. Os pacotes são divididos entre esses dois caminhos e transmitidos simultaneamente, alternando o rádio utilizado em cada salto do caminho e aproveitando os dois rádios disponíveis em todos os nós sensores. Na prática, ele con-

segue aumentar a vazão na rede em até 67%, quando comparado com um protocolo que utiliza apenas um caminho.

O restante do artigo está organizado da seguinte maneira: na seção 2 são apresentados alguns conceitos básicos e os trabalhos relacionados. Na seção 3 é apresentado o protocolo proposto, e na seção 4 é descrito o problema dos caminhos disjuntos com paridade. A seção 5 apresenta a modelagem do problema usando programação linear inteira. Na seção 6 são explicados os detalhes de implementação de teste do protocolo. Na seção 7 são descritos os experimentos realizados para avaliar o desempenho do protocolo e discutidos os resultados. Finalmente, as conclusões e trabalhos futuros são apresentados na seção 8.

2. Trabalhos Relacionados

Os protocolos tradicionais de coleta de dados em RSSFs não suportam alta vazão, pois eles eram projetados para minimizar o consumo de energia dos nós sensores. No cenário tradicional de plataformas com um único rádio, foram desenvolvidos vários protocolos específicos para transferência massiva de dados, por exemplo o Flush [Kim et al. 2007], FlushMF [Tavares et al. 2016] e o PIP [Raman et al. 2010]. Esses protocolos estabelecem uma rota de um único caminho entre um nó fonte e um nó de destino, desabilitam o ciclo de trabalho do rádio e forçam um escalonamento de pacotes ótimo de ponta-a-ponta nesse caminho. Até então o entendimento era que o baixo consumo de energia deveria ser deixado de lado para atingir alta vazão na rede.

Um pouco mais tarde é apresentado o Burst Forward [Duquennoy et al. 2011], uma técnica que permite que o rádio dos nós intermediários no caminho mantenham os seus ciclos de trabalho ativos para economizar energia, sendo necessário que apenas o nó fonte e o nó de destino o desabilitem. Os autores mostram que mesmo assim conseguem atingir uma vazão comparável aos protocolos anteriores, utilizando uma mistura da técnica de transmissão em rajadas e da técnica de armazenamento e encaminhamento. Porém, por terem sido desenvolvidos para plataformas de um único rádio, todos esses protocolos sofrem do problema fundamental em ter apenas um rádio: não é possível transmitir e receber pacotes ao mesmo tempo. Com isso, a vazão total na rede fica limitada a até no máximo 50% da capacidade do canal de comunicação.

Com o desenvolvimento de plataformas com dois rádios, passa a ser possível transmitir e receber pacotes ao mesmo tempo. Visando aproveitar essa nova possibilidade foi desenvolvido o FastForward [Ekbatanifard et al. 2013], até então o único protocolo de transferência massiva de dados projetado para plataformas com mais de um rádio. No FastForward, os pacotes são transmitidos de um nó fonte para um nó de destino através de um único caminho entre eles. Porém, a utilização dos rádios ao longo dos saltos desse caminho é alternada, sendo que cada nó intermediário recebe pacotes por um rádio e, simultaneamente, transmite pacotes pelo outro rádio. Com isso, o limite teórico para a vazão utilizando esse protocolo passa a ser 100% da capacidade do canal.

No FastForward, o fato de utilizar dois rádios em bandas diferentes já ajuda a resolver o problema de interferência entre pacotes. Além disso, ele utiliza a técnica já utilizada pelos protocolos anteriores de alternar canais entre os rádios de uma mesma banda, o que ajuda a diminuir ainda mais o efeito da interferência entre as transmissões. A Figura 1 mostra o esquema de alocamento de rádios e canais utilizado no FastForward.

Nele, duas transmissões sendo feitas pelo mesmo rádio e pelo mesmo canal estarão a pelo menos três saltos de distância um do outro.

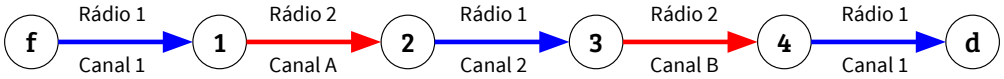

Figura 1. Esquema de alocamento de rádios e canais utilizado no FastForward

Observe que, no FastForward, os nós intermediários utilizam todo o recurso de rádio disponível, pois estão sempre recebendo pacotes por um rádio e transmitindo pelo outro. Porém, os nós fonte e de destino utilizam apenas metade do recurso de rádio disponível. O nó fonte transmite apenas por um rádio, e o nó de destino recebe por apenas um rádio. Para passar a aproveitar os dois rádios também nesses nós, é que foi desenvolvida a abordagem apresentada na próxima seção, que utiliza dois caminhos para a transmissão de dados.

3. Protocolo

Nessa seção, é apresentada a descrição de um novo protocolo de transmissão massiva de dados em vários saltos para RSSFs que utilizam plataformas com dois rádios, e utiliza dois caminhos distintos na rede para transmitir pacotes simultaneamente. O protocolo foi projetado para atuar juntamente com os outros protocolos tradicionais de RSSFs, que utilizam de um modo de economia de energia. Quando a aplicação precisar fazer uma transferência massiva de dados, ela chama o protocolo, que configura os caminhos a serem utilizados e faz com que esses nós saiam do modo de economia de energia, e fiquem no modo de transmissão até que a transferência seja concluída com alta vazão.

O princípio básico do protocolo proposto é utilizar 100% dos recursos de rádio dos nós sensores escolhidos para a transmissão de dados. Como o nó fonte tem dois rádios disponíveis, e ele é responsável apenas por transmitir pacotes, então ele deve transmitir os pacotes pelos dois rádios simultaneamente, de forma que os dois rádios estejam ocupados o tempo todo enquanto existem pacotes a serem enviados. No nó de destino, os dois rádios devem estar simultaneamente recebendo pacotes. Enquanto nos nós intermediários, um rádio deve estar ocupado recebendo pacotes e o outro encaminhando pacotes. Além de alternar entre rádios a cada salto, assim como nos protocolos anteriores, utiliza-se a alternância de canais para rádios que transmitem na mesma banda de frequências. São assinalados quatro canais para cada banda. A figura 2 mostra um exemplo de alocação de rádios e canais nos dois caminhos utilizados. Observe que em todos os nós, são utilizados os dois rádios disponíveis.

No nó fonte, os pacotes a serem enviados vão sendo colocados em uma fila. A cada pacote é assinalado um número de sequência para permitir a sua identificação e garantir o recebimento correto no nó de destino. Primeiramente, dois pacotes são retirados da fila, um é enviado por um rádio e o outro enviado pelo outro. Os destinos do próximo salto dessas mensagens são consultados em uma tabela de roteamento que indica dois endereços diferentes para o mesmo destino final, um para cada rádio. Assim que é sinalizado o fim da transmissão de um dos rádios e a transmissão ocorreu com sucesso,

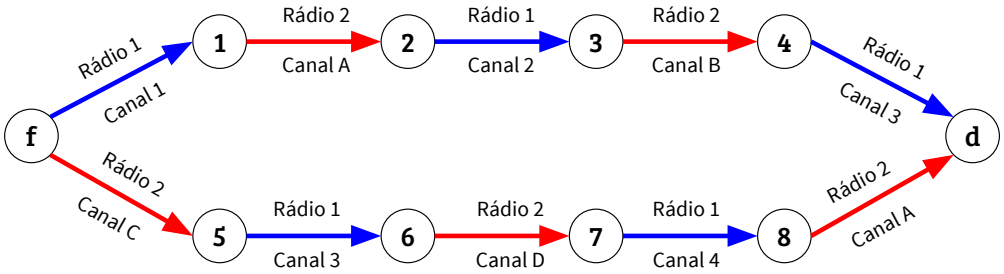

Figura 2. Esquema de alocamento de rádios e canais utilizado no novo protocolo

outro pacote é retirado da fila e transmitido pelo rádio que havia finalizado a transmissão. Como o tempo necessário para a transmissão de um pacote é muito maior do que o tempo necessário para o seu processamento, os dois rádios ficam transmitindo pacotes simultaneamente a maior parte do tempo. O procedimento se repete até que todos os pacotes tenham sido transmitidos.

Em cada um dos nós intermediários nos dois caminhos, ao receber um pacote por um dos rádios, ele consulta a sua tabela de roteamento para determinar o endereço do próximo salto, para o qual ele deve encaminhar essa mensagem. Nos nós intermediários, a tabela de roteamento contém apenas um endereço de encaminhamento para cada destino, pois cada nó intermediário pode fazer parte de apenas um dos caminhos entre o nó fonte e o nó de destino. A mensagem é colocada em uma fila e, sempre que o rádio transmissor fica livre, uma mensagem é retirada da fila e enviada por esse rádio. Novamente, como o tempo de recebimento e envio de pacotes é muito maior do que o tempo de processamento, os dois rádios passam a maior parte do tempo recebendo e transmitindo pacotes simultaneamente.

No nó de destino, os dois rádios são utilizados para receber pacotes. Por um rádio virão pacotes enviados pelo primeiro caminho e pelo outro rádio virão pacotes enviados pelo outro caminho. Os pacotes poderão chegar em ordem diferente da qual foram enviados, por vários motivos: um caminho pode ser mais longo do que o outro, ou um caminho pode ter menos perdas do que o outro. Cada pacote, ao ser recebido pelo nó de destino é sinalizado para a aplicação, juntamente com o seu número de sequência, porém na ordem de chegada. Fica a cargo a aplicação juntar os pacotes recebidos novamente na sequência correta, de acordo com a sua necessidade.

O novo protocolo tenta entregar a maior quantidade de pacotes possível com uma política de melhor esforço. Ele confia em pacotes de confirmação providos pela camada de enlace, e retransmite pacotes que não tenham recebido confirmação até um máximo de cinco retransmissões por pacote. Caso mesmo assim um pacote não consiga ser retransmitido, ele é perdido, e fica também a cargo da aplicação retransmitir ou não os pacotes perdidos de acordo com a sua necessidade. Também é possível configurar o protocolo para desativar o envio e recebimento de pacotes de confirmação da camada de enlace, em caso de necessidade de uma vazão ainda maior ao custo de uma menor confiabilidade.

Para que essa nova abordagem funcione, não basta que sejam utilizados quaisquer dois caminhos disjuntos entre o nó fonte e o nó de destino. A necessidade de alternar os rádios impõe restrições que precisam ser observadas para que tudo funcione. Por isso o

problema de encontrar caminhos em uma rede que sejam compatíveis com o protocolo é discutido na próxima seção.

4. O Problema dos Caminhos

O novo protocolo apresentado utiliza dois caminhos para transmitir dados, porém devido ao fato de cada nó possuir apenas dois rádios distintos que podem operar simultaneamente, esses caminhos devem ser escolhidos de modo que obedeçam a duas restrições. Primeiramente, eles devem ser disjuntos, exceto pelo nó fonte e o nó de destino. Todos os demais nós devem pertencer exclusivamente a um caminho ou outro. Se um nó intermediário for escolhido para os dois caminhos, ele não conseguirá receber e transmitir simultaneamente o fluxo que passa pelos dois, criando um gargalo que faz com que todo o ganho que o protocolo pode oferecer seja perdido.

A segunda restrição é que os dois caminhos devem ter a mesma paridade no seu número de saltos. Precisamos de dois caminhos com número de saltos par ou dois caminhos com número de saltos ímpar. Isso se deve ao fato de que o nó fonte precisa transmitir os pacotes por rádios diferentes e o nó de destino precisa receber os pacotes também por rádios diferentes, para que ambos possam operar simultaneamente. A figura 3 ilustra o porquê dessa restrição. Nela temos uma rede com sete nós, e escolhemos dois pares de caminhos disjuntos diferentes entre o nó fonte s e o nó de destino d. Os caminhos escolhidos em (a) têm paridade diferente. Como o nó fonte precisa enviar por rádios diferentes e os nós intermediários precisam alternar o rádio em que enviam, o nó de destino acaba recebendo os pacotes de ambos os caminhos pelo mesmo rádio, o que será impossível de ocorrer simultaneamente. Já os caminhos escolhidos em (b) têm a mesma paridade. Por isso o nó de destino acaba recebendo pacote pelos dois rádios distintos, o que pode ocorrer ao mesmo tempo.

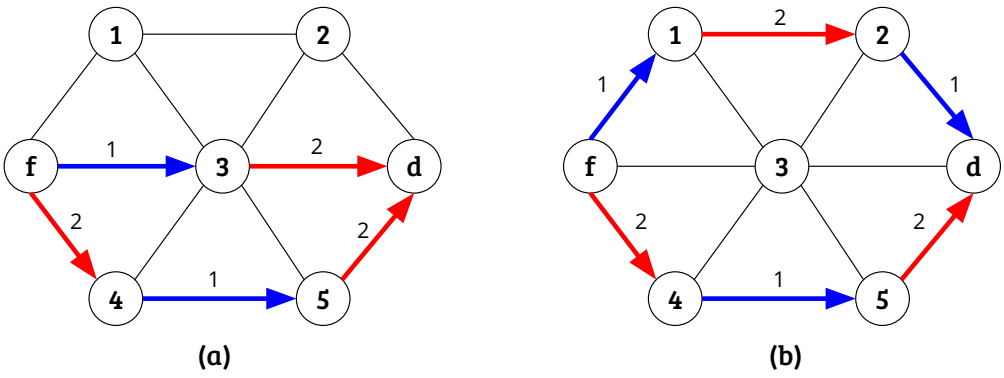

Figura 3. Exemplos de caminhos disjuntos em uma rede. Em (a) com paridades diferentes e em (b) com a mesma paridade.

Um algoritmo de roteamento geralmente estará tentando encontrar o caminho de menor custo entre uma origem e um destino. No caso desse roteamento por dois caminhos distintos, o problema é encontrar o par de caminhos de mesma origem e destino com o menor custo. Caso não houvesse a restrição de que os caminhos tem que ter a mesma paridade, esse seria um problema bem conhecido em teoria dos grafos, que seria facilmente resolvido utilizando o algoritmo de Suurballe [Suurballe and Tarjan 1984]. Porém

tal algoritmo poderia gerar uma solução como a da figura 3 (a), que não serve para ser utilizada com o nosso protocolo.

5. Modelo de Programação Linear Inteira

Para resolver o problema com a restrição de paridade dos caminhos, o modelamos como um problema de programação linear inteira. O modelo completo pode ser lido na figura 4.

$$\text{minimizar} \sum_{(i,j)} c_{ij}^1 x_{ij}^1 + c_{ij}^2 x_{ij}^2$$

sujeito à

$$\text{(i)} \quad \sum_{(i,j)\in S(i)} x_{ij}^1 - \sum_{(j,i)\in E(i)} x_{ji}^2 = \begin{cases} 1, \text{ se } i = f \\ -1, \text{ se } i = d \\ 0, \text{ caso contrário} \end{cases}$$

$$\text{(ii)} \quad \sum_{(i,j)\in S(i)} x_{ij}^2 - \sum_{(j,i)\in E(i)} x_{ji}^1 = \begin{cases} 1, \text{ se } i = f \\ -1, \text{ se } i = d \\ 0, \text{ caso contrário} \end{cases}$$

$$\text{(iii)} \quad \sum_{j\in E(i)} x_{ji}^1 + \sum_{j\in E(i)} x_{ji}^2 \leq 1, \quad \text{se } i \neq d$$

$$\text{(iv)} \quad \sum_{(i,j)\in A} x_{ij}^1 - \sum_{(i,j)\in A} x_{ij}^2 = 0$$

$$\text{(v)} \quad x_{ij}^1, x_{ij}^2 \in \{0, 1\}$$

Figura 4. Modelo de programação linear inteira para resolver o problema de encontrar o menor par de caminhos com a mesma paridade

Representamos a nossa rede como um grafo direcionado $G = (V, A)$. Para cada aresta (i, j) do grafo, associamos dois pesos diferentes, c_{ij}^1 e c_{ij}^2, que representam o custo de enviar uma mensagem do nó i para o nó j. c_{ij}^1 é o custo de enviar a mensagem pelo rádio 1 e c_{ij}^2 o custo de enviar pelo rádio 2. A cada aresta também são associadas duas variáveis binárias, x_{ij}^1 e x_{ij}^2, que assumirão valor 0 se a aresta (i, j) não foi escolhida para nenhum caminho, ou o valor 1, se aquela aresta foi escolhida. x_{ij}^1 representa que o rádio 1 é o rádio escolhido e x_{ij}^2 representa que o rádio 2 foi escolhido. A função objetivo do problema é minimizar o custo total, que é representado pelo somatório do custo de cada rádio multiplicado pela variável x correspondente.

Para modelar as restrições, precisamos fazer mais algumas definições: vamos chamar de f o nó fonte e de d o nó de destino. Para cada nó $i \in V$, vamos definir dois conjuntos de arestas: $S(i)$ é o conjunto de arestas que saem do nó i, e $E(i)$ é o conjunto de arestas que entram no nó i. Vamos então às restrições:

As restrições (i) e (ii) são para garantir a alternância de rádios entre os caminhos escolhidos. Na restrição (i), para cada vértice i, a soma das arestas que saem pelo rádio 1, menos a soma das arestas que entram pelo rádio 2 deve ser 1 se o nó for o nó fonte, -1

se for o nó de destino, ou 0 para todos os outros. Com isso garantimos que o nó fonte não receberá nada pelo rádio 2 e com certeza enviará algo pelo rádio 1. Garantimos que o nó de destino com certeza receberá algo pelo rádio 2 e não enviará nada pelo rádio 1. Para todos os outros nós, se ele recebe algo pelo rádio 2, com certeza ele enviará pelo rádio 1.

Similarmente à restrição anterior, na restrição (ii) para cada vértice i, a soma das arestas que saem pelo rádio 2, menos a soma das arestas que entram pelo rádio 1 deve ser 1 para o nó fonte, -1 para o nó de destino e 0 para todos os outros. Garantimos assim que o nó fonte não receberá nada também pelo rádio 1 e com certeza enviará pelo rádio 2. O nó de destino com certeza receberá algo pelo rádio 1 e não enviará nada pelo rádio 2 e para todos os outros, se o nó recebe algo pelo rádio 1, ele com certeza enviará algo pelo rádio 2.

A restrição (iii) garante que nenhum nó intermediário será escolhido para os dois caminhos, fazendo com que a soma das arestas de entrada em todo nó i, exceto o nó de destino, seja menor ou igual à 1. Como as variáveis são binárias, ou o rádio 1 será escolhido e o rádio 2 não, ou o rádio 2 será escolhido e o rádio 1 não, ou nenhum dos dois será escolhido. Para o nó de destino, de fato, duas arestas de entrada serão escolhidas, uma para cada rádio, como garantem as restrições (i) e (ii). As restrições (i) e (ii) também garantem que não precisamos criar uma restrição para o número de arestas de saída de cada nó, uma vez que ele deverá ser igual ao número de arestas de entrada nos nós intermediários.

Por fim, a restrição (iv) é o que garante que ambos os caminhos terão a mesma paridade. Ela diz que o número de arestas onde o rádio 1 é o escolhido e o número de arestas onde o rádio 2 é escolhido tem que ser iguais. A restrição (v) é para garantir que as variáveis x_{ij}^1 e x_{ij}^2 são binárias.

Com esse modelo de programação linear inteira e dados sobre a qualidade dos enlaces da rede, podemos calcular o menor par de caminhos disjuntos entre um nó fonte e um nó de destino, com caminhos de mesma paridade. Com esses caminhos, podemos utilizar o protocolo proposto nesse trabalho para transmitir dados.

6. Implementação

O protocolo foi implementado usando o TinyOS 2.1.2 para a plataforma Opal [Jurdak et al. 2011]. Essa plataforma dispõe de dois transceptores de rádio 802.15.4 que operam em bandas diferentes: 900 MHz e 2.4 GHz. Os dois rádios compartilham o mesmo barramento SPI, o que cria um gargalo para a transferência de dados entre os rádios e o microcontrolador. Porém, a transferência de dados no barramento é muito mais rápida do que a transmissão de dados pelo rádio. Na plataforma Opal, enviar um pacote SPI para o buffer de transmissão leva menos de 10% do tempo necessário para transmitir o pacote pelo rádio [Ekbatanifard et al. 2013]. Como o protocolo busca manter os rádios sempre ocupados, os dois rádios estarão operando simultaneamente a maior parte do tempo.

A figura 5 mostra uma visão geral da arquitetura implementada para o protocolo, implementada em cima da pilha de protocolos de comunicação para os dois rádios, já disponíveis para TinyOS: o driver dos rádios, e a implementação das camadas MAC e de enlace. Ela oferece opções para configurar a modulação utilizada pelos rádios, a potência

de transmissão, a utilização de verificação de ocupação do canal (CCA, do inglês *Clear Channel Assessment*) e os parâmetros de backoff aleatório utilizados. Ainda há a possibilidade de habilitar ou desabilitar o uso de pacotes de confirmação fornecidos pela camada de enlace.

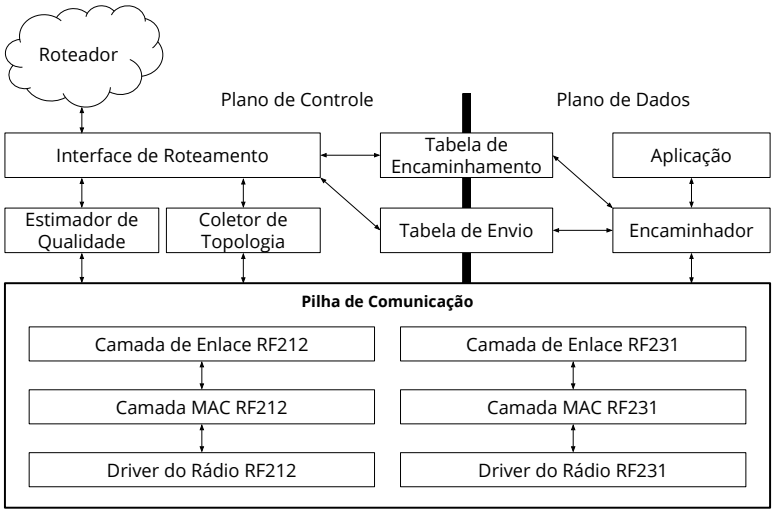

Figura 5. Visão geral da arquitetura implementada

O plano de controle tem a função de encontrar o menor par de caminhos entre um nó fonte e um nó de destino e estabelecer as rotas para os pacotes preenchendo suas tabelas de roteamento. O plano de dados é responsável por enviar e encaminhar os pacotes de acordo com o roteamento pré-definido. Nessa implementação, esse processo se dá em três fases distintas. Primeiro a topologia é coletada, e enviada para o roteador. Segundo, roteamento é calculado offline e as tabelas dos nós sensores escolhidos preenchidas estaticamente. Finalmente, a transmissão dos pacotes é iniciada pelo nó fonte pré-definido no roteamento. A seguir são dados detalhes da implementação de cada uma das etapas.

6.1. Coleta da Topologia

Um estimador de enlace faz com que os nós sensores enviem pacotes periodicamente para que seus vizinhos montam uma tabela que representa a topologia da rede. Ao receber um pacote, o nó sensor adiciona o nó que enviou aquele pacote à sua lista de vizinhos, e passa a contar quantos pacotes recebeu daquele mesmo vizinho. A métrica utilizada para a qualidade do enlace é a razão entre o número de pacotes enviados pelo número de pacotes recebidos entre os nós. Cada nó sensor envia a informação sobre os seus vizinhos pela interface de roteamento, que envia pela porta serial do nó sensor para um computador externo à rede, coletando toda a informação sobre a topologia atual da rede.

6.2. Roteamento

De posse das informações da topologia da rede, o roteamento é feito offline, ou seja, fora da rede de sensores calcula o menor par de caminhos entre determinada fonte e determinado destino na rede. Para isso, utiliza-se o modelo de programação linear inteira descrito na seção 4, implementado utilizando a linguagem GMPL (GNU Mathematical Programming Language) e resolvido utilizando o GLPK (GNU Linear Programming Kit), ferramenta de código aberto para resolver problemas de programação linear. O GLPK levou

em média 107ms para resolver cada instância do problema utilizando o nosso modelo de programação linear inteira, com uma topologia de 100 nós.

A solução encontrada é estaticamente preenchida nas tabelas de roteamento dos nós que fazem parte dos caminhos. Existem duas tabelas: uma tabela de envio e uma tabela de retransmissão. A tabela de envio é utilizada para indicar o próximo salto de cada um dos caminhos para o nó fonte, deve conter duas entradas para cada destino. A tabela de retransmissão é utilizada quando o nó em questão é um nó intermediário no caminho, e pode conter apenas uma entrada para cada destino, que indica o próximo salto. Além do endereço do próximo salto, a tabela indica também o rádio e o canal pela qual a transmissão deve ser feita.

6.3. Transmissão

Depois que as rotas foram estabelecidas, antes de iniciar a transmissão, cada nó altera o canal de operação de seus rádios para os canais indicados na tabela de roteamento. Quando todo o caminho está configurado, o nó fonte inicia a transmissão dos pacotes seguindo o protocolo, descrito na seção 3.

Toda a implementação desse projeto, que inclui o estimador de qualidade, coletor de topologia, o protocolo de transmissão e a modelagem do problema de roteamento usando programação linear são código aberto, e estão disponíveis na internet, em repositório público no GitHub.[1] Além disso também estão disponíveis programas utilizados para realizar os experimentos apresentados na próxima seção.

7. Experimentos e Resultados

Para avaliar o desempenho do protocolo foram realizados experimentos em um testbed de redes de sensores sem fio de larga escala chamado Twonet [Li et al. 2013]. Ele contém 100 nós sensores com dois rádios da plataforma Opal. A interface online provida pelo testbed permite enviar arquivos executáveis compilados para a plataforma Opal, escolher os nós que serão programados, agendar um intervalo de tempo para execução, e como resultado é retornado uma lista de mensagens enviadas para a porta serial de cada nó sensor.

Os experimentos foram feitos para avaliar a vazão e a taxa de entrega de pacotes utilizando o protocolo proposto. Foram feitas várias rodadas de experimentos, e cada rodada consistia nos seguintes passos: coleta da topologia, determinação dos nós de origem e destino, determinação das rotas e medições. O nó fonte e o nó de destino de cada experimento foram escolhidos de acordo com a distância entre eles. Definimos a distância do nó de origem ao destino sendo o número total de saltos dos dois caminhos dividido por 2. Então dois nós em que o par de caminhos entre eles tenham comprimento 5 cada um, estão a uma distância 5. Um par de nós em que os caminhos têm comprimento 4 e 6, também estão a uma distância 5.

Os dois rádios foram configurados para transmitir usando a modulação O-QPSK a 250 kbps e com potência de transmissão de 3 dBm. Nos primeiros experimentos, deixamos habilitadas na camada MAC a verificação da ocupação do canal e os backoffs aleatórios. Depois, essas funções foram desabilitadas a fim de comparar os resultados

[1] `https://github.com/nildo/mpdr`

com os resultados do FastForward. Também realizamos os experimentos com e sem pacotes de confirmação, para analisar o compromisso entre a vazão e a taxa de entrega nos dois casos.

Em cada experimento são enviados 1000 pacotes a partir do nó fonte. Cada pacote tem uma carga útil de 100 bytes, porém o número de bytes efetivamente transmitido pelo rádio para cada pacote foi 127, por causa do custo adicional de vários cabeçalhos de pacote, o que inclui o cabeçalho do protocolo e cabeçalhos da camada de enlace. As métricas utilizadas nos experimentos foram a vazão e a taxa de entrega. Definimos a vazão sendo o número total de bytes recebido pelo nó de destino por segundo, incluindo aqueles não relacionados à carga útil no pacote. Definimos a taxa de entrega como o número de pacotes únicos recebidos pelo nó de destino dividido pelo número total de pacotes enviados pelo nó fonte. Os experimentos foram repetidos 10 vezes para cada instância, e os valores apresentados são a média e o desvio padrão dos resultados obtidos.

A seguir são apresentados gráficos que mostram o desempenho do novo protocolo proposto nesse trabalho que utiliza dois caminhos e comparado com o desempenho da nossa implementação do FastForward. Na figura 6 está o resultado dos testes que foram feitos deixando habilitada a verificação de ocupação do canal (CCA). O gráfico da esquerda mostra a vazão e o gráfico da direita mostra a taxa de entrega. Podemos observar que o novo protocolo conseguiu maior vazão. Em média, o protocolo de dois caminhos conseguiu uma melhora de 60% na vazão nesse cenário. A taxa de entrega do novo protocolo também foi sempre maior ou igual ao FastForward, conseguindo 100% de taxa de entrega em alguns casos devido ao uso de pacotes de confirmação e retransmissões.

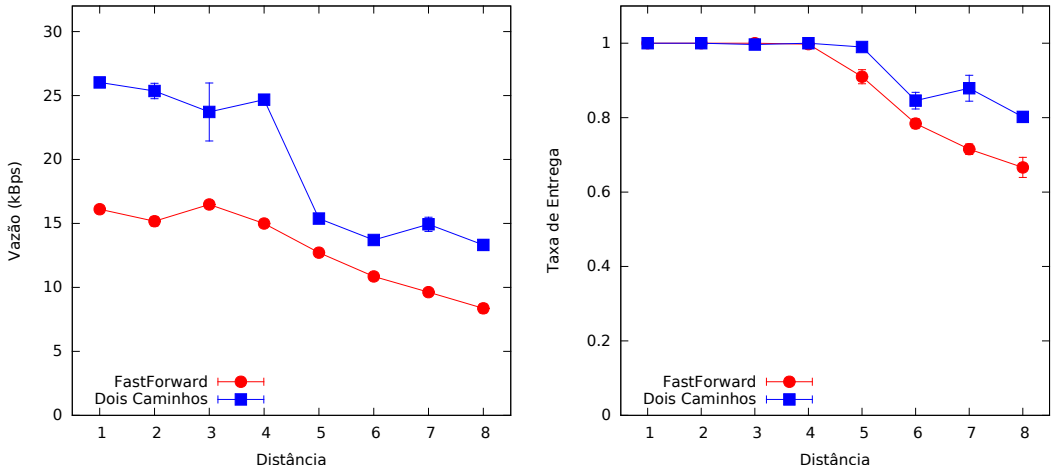

Figura 6. Desempenho com CCA habilitado e com confirmação de mensagens

Ainda observando a figura 6, ambos os protocolos conseguem o máximo de desempenho a até uma distância de 4 entre a fonte e o nó de destino. Como o FastForward usa dois canais em cada rádio, e nossa abordagem utiliza quatro canais, quando o caminho é curto, não há interferência entre canais. A partir de uma distância de 5, passam a ocorrer transmissões no mesmo rádio e no mesmo canal, e essa disputa pelo meio de comunicação faz com que o desempenho dos protocolos piore.

A figura 7 mostra o desempenho dos protocolos quando desabilitamos o CCA na

camada MAC dos rádios. Na prática não é recomendado desabilitar essa função, pois o meio de comunicação pode estar sendo utilizado por várias redes diferentes e uma pode congestionar a outra. O teste foi realizado para avaliar o potencial máximo de transmissão dos protocolos. Nesse cenário também são utilizados os pacotes de confirmação. Podemos observar que nos primeiros casos, obteve-se uma vazão de 50 kBps com o novo protocolo, enquanto o FastForward consegue no máximo 25 kBps, sendo 100% de melhoria, exatamente o limite máximo que se poderia conseguir. Porém o desempenho vai diminuindo a medida em que a distância aumenta. O desempenho nesse cenário cai mais rápido do que o cenário anterior que utilizava o CCA. Nos caminhos mais longos, a vazão é até menor do que antes. Novamente a vazão do novo protocolo foi sempre maior, porém a taxa de entrega entre os dois alternou.

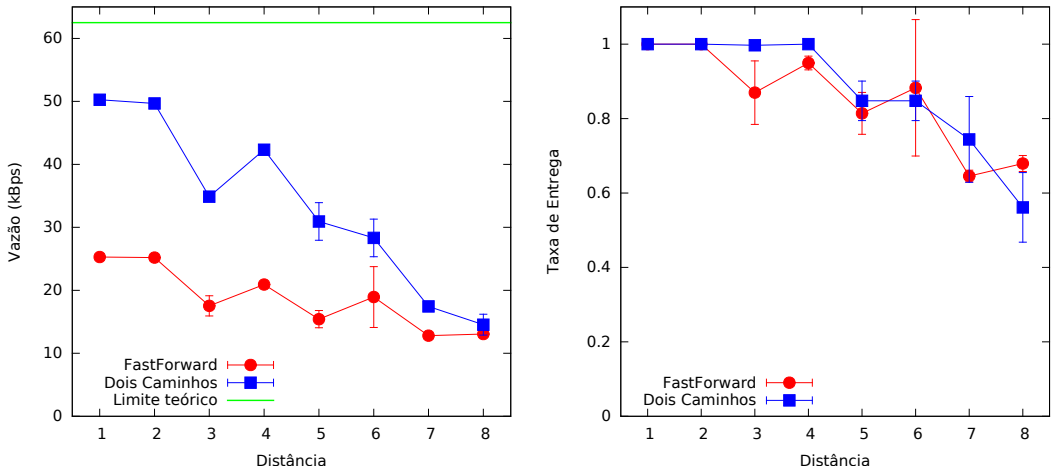

Figura 7. Desempenho com CCA desabilitado e com confirmação de mensagens

Finalmente, a figura 8 representa um cenário onde tanto a verificação de ocupação do canal quanto as mensagens de confirmação foram desativadas. Nesse cenário obtivemos o valor máximo de vazão de 60 kBps nos testes em que a distância era menor ou igual a 4. Esse valor é próximo ao limite teórico máximo de vazão ao utilizar dois rádios. A taxa de transmissão máxima, considerando um ambiente ideal, utilizando apenas um rádio é 250 kbps, ou 31,25 kBps. Então o limite teórico para dois rádios é o dobro, ou 62,5 kBps. Logo, o valor conseguido é 96% da vazão máxima teórica e em um ambiente real. Além disso, esse valor é o dobro do valor conseguido com o FastForward, indicando o máximo aproveitamento dos dois rádios em todos os nós do caminho.

Ainda observando a figura 8, podemos observar uma queda drástica na vazão a partir da distância igual a 5. Essa queda se dá devido à queda na taxa de entrega, que podemos observar no gráfico da direita. Sem CCA, muitos pacotes são perdidos devido à interferência causada entre enlaces que estão compartilhando o mesmo canal, e sem pacotes de confirmação não há a retransmissão de pacotes perdidos. Então apesar do desempenho duas vezes maior para caminhos mais curtos, para caminhos longos o desempenho dos dois protocolos foi praticamente o mesmo, sendo que o FastForward obteve maior taxa de entrega.

Esse último cenário mostra que o protocolo proposto sofre mais com a interferência do que o FastForward, uma vez que ele utiliza mais enlaces. Porém em uma

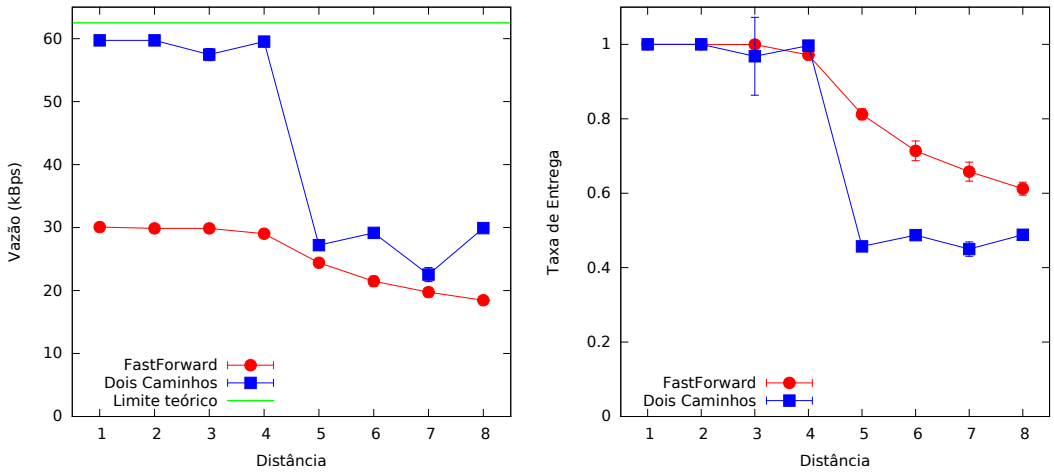

Figura 8. Desempenho com CCA desabilitado e sem confirmação de mensagens

aplicação prática, não seria recomendado desativar a verificação de ocupação do canal, uma vez que é um mecanismo importante para evitar a saturação do meio de comunicação inclusive com outras redes que possam estar presentes no mesmo espaço físico. Como vimos, quando o CCA é habilitado, o desempenho do novo protocolo é melhor do que o do FastForward até para caminhos mais longos.

Uma última questão a ser discutida é em relação ao custo de energia envolvido. Com uma vazão maior, é esperado que o gasto de energia total seja maior. Porém o gasto de energia por byte transmitido se torna menor. Segundo [Jurdak et al. 2011], o Opal consome em média 49 mA de corrente se estiver operando os dois rádios simultaneamente. Como nosso protocolo alcançou uma vazão de 60 kBps, temos um gasto de energético de 0,82 mA/kB. Enquanto com a vazão máxima alcançado pelo FastForward, de 30 kBps, teríamos um gasto energético de 1,64 mA/kB. Logo, nosso protocolo consome metade da energia por byte transmitido em cada nó. Como o número de nós ao utilizar dois caminhos não dobra, pois os nós de origem e destino são sempre os mesmos, o nosso protocolo terá uma eficiência energética total ainda maior do que o FastForward.

8. Conclusão

Nesse artigo foi apresentado um novo protocolo para transferência de dados em massa otimizado para plataformas que utilizam dois rádios. A principal vantagem dessa nova abordagem é a utilização dos dois transceptores disponíveis em cada nó em paralelo, tanto nos nós de origem e destino quanto nos nós intermediários. O protocolo visa atender a demanda por alta vazão em novas aplicações de redes de sensores sem fio que precisam transmitir dados multimídia e em tempo real.

Os experimentos realizados no mundo físico mostram que a abordagem proposta consegue uma vazão de até 60 kBps, o que representa 96% do limite máximo teórico de 62,5 kBps, quando utilizamos dois rádios 802.15.4 utilizando modulação O-QPSK a 250 kbps em paralelo, sem verificação de ocupação do canal. Para uma aplicação que precisa dividir o meio de comunicação e utilizar a verificação de canal ocupado, nosso protocolo consegue uma vazão de até 26 kBps contra 16 kBps do protocolo estado-da-arte FastForward, uma melhora de 60% de desempenho.

Como trabalho futuro, pode-se desenvolver um esquema de roteamento descentralizado para dois caminhos, que possa ser calculado de dentro da própria rede. O problema de encontrar caminhos tem a possibilidade de ser um problema difícil, provavelmente necessitando de boas heurísticas ou soluções aproximadas. Além disso incluir mecanismos que minimizem a interferência cocanal entre os caminhos, uma vez que é o que mais afeta o desempenho do protocolo.

Referências

Akyildiz, I. F. and Vuran, M. C. (2010). *Wireless sensor networks*, volume 4. John Wiley & Sons.

Duquennoy, S., Österlind, F., and Dunkels, A. (2011). Lossy links, low power, high throughput. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, pages 12–25. ACM.

Ekbatanifard, G., Sommer, P., Kusy, B., Iyer, V., and Langendoen, K. (2013). Fast-forward: High-throughput dual-radio streaming. In *2013 IEEE 10th International Conference on Mobile Ad-Hoc and Sensor Systems*, pages 209–213. IEEE.

Jurdak, R., Klues, K., Kusy, B., Richter, C., Langendoen, K., and Brünig, M. (2011). Opal: A multiradio platform for high throughput wireless sensor networks. *Embedded Systems Letters, IEEE*, 3(4):121–124.

Kim, S., Fonseca, R., Dutta, P., Tavakoli, A., Culler, D., Levis, P., Shenker, S., and Stoica, I. (2007). Flush: a reliable bulk transport protocol for multihop wireless networks. In *Proceedings of the 5th international conference on Embedded networked sensor systems*, pages 351–365. ACM.

Kusy, B., Richter, C., Hu, W., Afanasyev, M., Jurdak, R., Brünig, M., Abbott, D., Huynh, C., and Ostry, D. (2011). Radio diversity for reliable communication in wsns. In *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on*, pages 270–281. IEEE.

Li, Q., Han, D., Gnawali, O., Sommer, P., and Kusy, B. (2013). Twonet: Large-scale wireless sensor network testbed with dual-radio nodes. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, page 89. ACM.

Raman, B., Chebrolu, K., Bijwe, S., and Gabale, V. (2010). Pip: A connection-oriented, multi-hop, multi-channel tdma-based mac for high throughput bulk transfer. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, pages 15–28. ACM.

Suurballe, J. W. and Tarjan, R. E. (1984). A quick method for finding shortest pairs of disjoint paths. *Networks*, 14(2):325–336.

Tavares, R., Vieira, M. A. M., and Vieira, L. F. M. (2016). Flushmf: A transport protocol using multiple frequencies for wireless sensor network. In *Proceedings of the 13th IEEE International Conference on Mobile Ad hoc and Sensor Systems (MASS)*. IEEE.

Yick, J., Mukherjee, B., and Ghosal, D. (2008). Wireless sensor network survey. *Computer networks*, 52(12):2292–2330.