CodeDrip: Protocolo de Disseminação de Dados em Redes de Sensores Sem Fio Utilizando Codificação na Rede

Nildo dos Santos Ribeiro Júnior[1], **Luiz F. M. Vieira**[1], **Marcos A. M. Vieira**[1]

[1] Departamento de Ciência da Computação
Instituto de Ciências Exatas
Universidade Federal de Minas Gerais

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

Abstract. *In this paper, we present CodeDrip, a data dissemination protocol for Wireless Sensor Networks that utilizes Network Coding to improve performance. The dissemination goal is to distribute data from the sink node to all nodes in the network. The Network Coding technique consists of combining packets before transmitting them. The dissemination process becomes more robust to packet loss with Network Coding since the lost packets can be recovered by combining the received packets. We simulate CodeDrip in many wireless scenarios, varying the network density and link quality. We compare CodeDrip with the state-of-the-art Drip. Results show that CodeDrip is faster than Drip to disseminate information and transmits fewer packets.*

Resumo. *Este artigo apresenta o protocolo de disseminação de dados CodeDrip para Redes de Sensores Sem Fio utilizando a técnica de Codificação em Rede a fim de obter um melhor desempenho nesse processo. O objetivo da disseminação é fazer com que dados do nó sorvedouro cheguem a todos os nós sensores da rede. Tendo vários pacotes a serem distribuídos, a técnica de Codificação em Rede consiste em fazer com que alguns pacotes sejam combinados antes de serem enviados. O processo de disseminação se torna mais robusto a perda de pacotes com a técnica de Codificação em Rede pois aqueles pacotes que foram perdidos poderão ser recuperados a partir da combinação de pacotes recebidos. Simulamos o CodeDrip em redes com várias densidades e vários níveis de qualidade do canal de enlace. Comparamos o CodeDrip com o protocolo estado da arte Drip. Os resultados mostram que a disseminação do CodeDrip é mais rápida e transmite menos pacotes do que o Drip.*

1. Introdução

Redes de Sensores Sem Fio (RSSFs) são redes com grande número de microsensores compactos com capacidade de comunicação sem fio, chamados de nós sensores [Ruiz et al. 2004]. O objetivo destas redes é sensoriar o meio e coletar dados. Em várias situações existe a necessidade de se monitorar condições ambientais, tanto em locais internos, como um prédio ou uma fábrica, quanto em locais externos, como uma floresta ou nas ruas da cidade. Instalar fios pelo ambiente é inviável em várias situações. A melhor alternativa é utilizar comunicação sem fio. Essa estrutura formada por nós sensores que se comunicam entre si é uma Rede de Sensores Sem Fio.

Em RSSFs, é importante notificar cada nó sensor. Isso permite que administradores de rede possam reconfigurar, criar consultas, comando e tarefas e reprogramar a

rede. Para isso, é necessário fazer com que várias mensagens cheguem a todos os nós sensores da rede. Esse processo é chamado de disseminação (figura 1). Um protocolo de disseminação de dados eficiente precisa transpor algumas dificuldades impostas pela natureza de uma rede de sensores sem fio. A primeira é que a energia disponível em um nó sensor é limitada por uma bateria, então é preciso economizá-la para aumentar o tempo de atividade dos sensores. A segunda é a limitação de capacidade computacional nos nós sensores, que podem não ser capazes de rodar protocolos de rede sofisticados. Por fim, a comunicação é suscetível à erros de transmissão e perdas de pacotes. Para um protocolo de disseminação de dados em uma RSSF, além de consumir pouca energia dos nós sensores, é desejável que a disseminação aconteça rapidamente.

Nesse artigo, apresentamos um protocolo de disseminação de dados em redes de sensores sem fio que utiliza a técnica de Codificação em Rede para melhorar a sua eficiência. Em vez de simplesmente retransmitir os pacotes de dados que recebem, os nós sensores combinam vários pacotes em um só, e transmitem essa combinação para os seus vizinhos. Com isso, a perda de pacotes de dados é atenuada, uma vez que pacotes perdidos poderão ser obtido através da decodificação da combinação de outros pacotes recebidos. Evitando retransmissões, a disseminação dos dados ocorre em menos tempo.

Protocolos de disseminação de dados para redes de sensores sem fio existentes utilizam diferentes técnicas para diminuir o número de mensagens enviadas por período de tempo, diminuindo o consumo de energia do nó sensor. Essas técnicas comprometem o desempenho do protocolo na questão do tempo de disseminação, podendo ocorrer casos em que ela demora muito mais que o esperado por causa de perdas de pacotes durante o processo. O protocolo de disseminação apresentado nesse artigo, ao utilizar a técnica de Codificação em Rede provê mais eficiência ao processo em termos de número de pacotes enviados e tempo de disseminação, pois a perda de pacotes afeta menos a transmissão dos dados.

A próxima seção desse artigo apresenta os trabalhos relacionados e as ferramentas que foram utilizadas para o desenvolvimento desse trabalho. Na seção 3 falamos sobre o princípio de funcionamento e vantagens do método de Codificação em Rede. O algoritmo do protocolo desenvolvido é explicado na seção 4, o processo de simulação de redes de sensores sem fio é descrito na seção 5 e os resultados obtidos destas simulações são mostrados na seção 6. Finalmente na seção 7 são apresentadas as conclusões do trabalho.

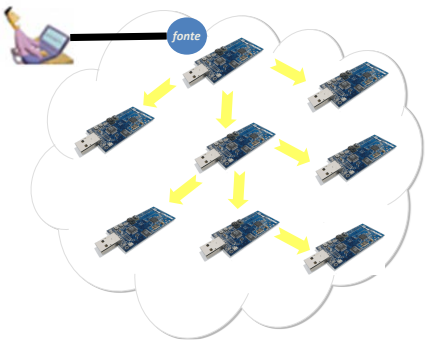

Figura 1. Usuários podem enviar mensagens para todos os nós da rede através do nó fonte utilizando um protocolo de disseminação.

2. Trabalhos Relacionados

Trickle [Levis et al. 2004] é um algoritmo para propagação e manutenção de atualização de código em Redes de Sensores Sem Fio. Nele os nós sensores transmitem periodicamente uma mensagem de aviso para seus vizinhos informando a versão do seu código, mas evitam transmitir se eles receberam recentemente um aviso igual ao seu.

O TrickleTimer é o temporizador utilizado no algoritmo Trickle. Esse temporizador já está implementado como um componente do TinyOS e é utilizado tanto pelo protocolo Drip quanto pelo CodeDrip. Ele incrementa o tempo de espera a cada vez que é disparado. Sendo assim, quanto maior é o tempo decorrido, menor será a frequência que o temporizador disparará um trecho de código a ser executado. A interface TrickleTimer do TinyOS oferece uma função que faz com que o intervalo de espera volte a ser pequeno. Essa função é utilizada quando alguma mensagem nova chega no nó sensor, fazendo com que essa mensagem nova seja retransmitida com mais frequência nesse início, quando a probabilidade de seus vizinhos ainda não a terem recebido é maior.

O Drip [Tolle and Culler 2005] é um protocolo de disseminação de dados para Redes de Sensores Sem Fio implementado como um componente do TinyOS 2.1. Ele provê uma interface de camada de transporte para várias camadas de disseminação de mensagens confiável. O Drip utiliza TrickleTimers para fazer transmissões periódicas do dado que está sendo disseminado para garantir que esse dado eventualmente chegará a todos os nós sensores da rede. Ele é um protocolo estado-da-arte para disseminação de pequenos valores.

O componente implementado no TinyOS para disseminação é o DisseminatorC. Ele implementa o protocolo Drip para fazer a disseminação de um dado na rede. Para cada valor que se queira disseminar na rede, um componente DisseminatorC é instanciado em cada nó sensor. Esse componente é responsável por criar o temporizador que ele precisa, enviar e receber as mensagens para atualizar o dado que o nó sensor possui. Duas interfaces providas por esse componente permitem o nó sensor acessar a mensagem sendo disseminada: DisseminationValue e DisseminationUpdate. A interface DisseminationValue provê um evento que é disparado quando uma nova mensagem chega e o dado que ela contém é novo para esse nó sensor. A interface DisseminationUpdate possui uma função que permite o nó sensor atualizar o seu dado e, assim, começar a disseminação desse dado na rede.

Deluge [Hui and Culler 2004] é outro protocolo de disseminação de dados para RSSFs também implementado no TinyOS. Seu foco é a disseminação de um grande objeto de dados, isto é, que não cabe na memória RAM, de um ou mais nós fonte para o resto da rede. Para isso, esse protocolo representa o objeto de dados como um conjunto de páginas de tamanho fixo, o que fornece uma unidade gerenciável de transferência que permite a multiplexação espacial e suporta atualizações incrementais eficientes. O Deluge também utiliza TrickleTimers para fazer o controle de quando enviar pacotes pela rede.

Como esse protocolo é específico para grandes objetos de dados, as características dele são muito diferentes das do Drip e do CodeDrip, que focam na disseminação de dados pequenos. Por terem objetivos diferentes, o Deluge é um trabalho complementar ao CodeDrip na manutenção de RSSFs.

O trabalho de Ahlswede et al. [Ahlswede et al. 2000] motivou a pesquisa sobre

Codificação em Rede e mostrou que, em geral, a codificação de pacotes poderia resultar em um nível ótimo de aproveitamento da capacidade de transmissão da rede que não poderia ser atingida por outro possível esquema de apenas encaminhamento. Por exemplo, em tráfego multicast, a capacidade de transmissão é definida como a taxa máxima de transmissão em que um nó fonte pode enviar pacotes para todos os membros de um conjunto de receptores. Ela é dada pelo mínimo dos fluxos máximos (s, t) entre o nó fonte s e cada receptor t. Foi mostrado que com Codificação em Rede pode-se atingir uma capacidade de transmissão muito maior que um esquema de somente encaminhamento. Além disso, Li, Yeung and Cai [Li et al. 2003] mostraram que é suficiente que a função de codificação seja linear.

Foi demonstrado que, em ambientes sem fio, a codificação em rede oferece vários benefícios, como diminuição do consumo de energia por meio da redução do número de transmissões, aumento da taxa de transmissão de dados e maior robustez, por causa da possibilidade de um nó receber múltiplas cópias de um único pacote [Deb 2005].

Katti et al. [Katti et al. 2008] demonstraram que o uso de codificação em rede pode melhorar a taxa de transmissão global da rede. Desde então, protocolos têm sido desenvolvidos para cenários específicos. O CodeTorrent [Lee et al. 2006] faz distribuição de conteúdo em VANETs usando codificação em rede.

T. Cui et al. [Cui et al. 2010] consideraram fontes múltiplas com utilitários e codificação em rede em sessões e forneceram um algoritmo para controle de taxa de junção, codificação em rede e agendamento.

Zhang et al. [Zhang and Zhang 2009] fizeram uma formulação para calcular a taxa de transmissão máxima de tráfego unicast que pode ser alcançada com codificação em rede cooperativa. Keller et al. [Keller et al. 2011] investigam experimentalmente o atraso de protocolos baseados na técnica de inundação (flooding) para uma aplicação em detecção de atiradores usando codificação em rede. O DutyCode [Chandanala and Stoleru 2010] combina a ideia de codificação em rede e ciclo de trabalho (duty-cycle) na camada MAC, porém eles não discutem disseminação de dados para redes de sensores sem fio.

A disseminação de dados em RSSFs com codificação em rede também é apresentada em [Wang et al. 2010] porém o foco é bem diferente do nosso trabalho. Eles supõem a camada MAC composta por um protocolo TDMA e focam em um escalonamento de pacotes, determinando quais pacotes a serem combinados e transmitidos dado um cronograma de funcionamento dos nós sensores. Eles provaram que o problema é NP-Difícil e forneceram uma formulação de programação linear. Eles não proveram um protocolo para disseminação e não mencionaram, por exemplo, a questão do uso de temporizadores.

2.1. Ferramentas Utilizadas

Nessa seção apresentamos duas ferramentas utilizadas para o desenvolvimento do nosso trabalho: o TinyOS, um sistema operacional para sensores sem fio e o TOSSIM, um simulador de RSSFs.

O TinyOS é um sistema operacional leve projetado para sensores sem fio de baixa potência. Ele se diferencia da maioria dos outros sistemas operacionais porque o seu projeto foca em operações com baixo consumo de energia. [Levis and Gay 2009]. Ele fa-

cilita o desenvolvimento de aplicações para Redes de Sensores Sem Fio provendo serviços e abstrações como sensoriamento, comunicação, armazenamento e termporizadores. Esse sistema é licenciado sobre a BSD, uma licença de código aberto.

O TinyOS provê um conjunto de componentes de sistema reutilizáveis que podem ser conectados à aplicação. Assim, cada aplicação pode escolher os componentes que necessitar, evitando que serviços não usados sejam compilados e passados para o dispositivo. Então, ao invés de ser um sistema operacional tradicional, o TinyOS permite que um novo sistema específico seja construído durante a compilação, contendo apenas os componentes que interessam à aplicação. Como os recursos computacionais disponíveis em um nó sensor são poucos, esse sitema se torna bastante interessante.

O TinyOS é desenvolvido em nesC, uma linguagem de programação para sistemas de rede incorporados, como nós sensores. [Gay et al. 2003]. Ela é uma linguagem orientada a componentes com um modelo de execução baseado em eventos. O nesC foi desenvolvido tendo em vista superar os desafios que existem na construção de aplicações para Redes de Sensores Sem Fio, sendo que o principal desses desafios é a disponibilidade de recursos computacionais muito limitados, como armazenamento, processamento e energia.

O TOSSIM é um simulador de Redes de Sensores Sem Fio para aplicações desenvolvidas com o TinyOS. Ele consegue simular o comportamento e as interações de milhares de nós sensores em uma rede. [Levis et al. 2003] A arquitetura do TOSSIM é composta pelos seguintes elementos: suporte para compilar os componentes do TinyOS para a infraestrutura de simulação, uma fila de eventos discreta, abstrações de componentes de hardware do TinyOS, mecanismos para modelos de rádio e de conversores analógico/digital, e serviços de comunicação para programas externos interagirem com o simulador.

O TOSSIM permite simular o mesmo código que é executado no nó sensor e usa uma abstração simples porém poderosa para uma rede de sensores sem fio. A rede é representada por um grafo direcionado, onde cada vértice é um nó sensor e cada aresta representa uma possível comunicação com uma probabilidade de erro. Essa abstração permite que sejam feitos testes tanto com perfeita condição de transmissão, quanto com muitas perdas na rede.

3. Codificação em Rede

Codificação em Rede (do inglês *Network Coding* [Ahlswede et al. 2000]) é uma técnica que permite combinar pacotes codificados na rede e provê ganhos para a rede aumentando a vazão desta, diminuindo o consumo de energia, reduzindo o número de mensagens que são transmitidas para disseminar informação, entre outros [Vieira et al. 2007]. O ganho da Codificação em Rede vem da combinação da informação dos pacotes na rede, possibilitando uma disseminação de informação na rede mais eficiente.

Nas transmissões em redes, pacotes podem ser perdidos. A perda de pacotes pacotes exige a retransmissão destes pacotes. Combinando os pacotes de forma inteligente é possível recuperar as informações transmitidas sem ter que retransmitir todos os pacotes perdidos para todos os nós da rede. Utilizaremos um exemplo simples e didático para ilustrar e explicar o funcionamento da Codificação em Rede.

Considere a topologia da figura 2. O nó fonte, no centro da figura, deseja disseminar dois pacotes, chamados de P_1 e P_2. Os nós 1 e 2 estão no raio de alcance do nó fonte e possivelmente podem receber os pacotes. O pacote P_1 é transmitido pelo nó fonte e é recebido apenas pelo nó 1. A seguir o pacote P_2 é transmitido pelo nó fonte e, desta vez, recebido apenas pelo nó 2. Cada nó receptor tem um pacote diferente.

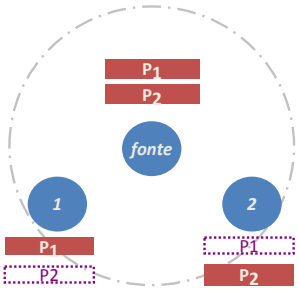

Figura 2. Nó fonte quer disseminar dois pacotes para os dois nós da rede.

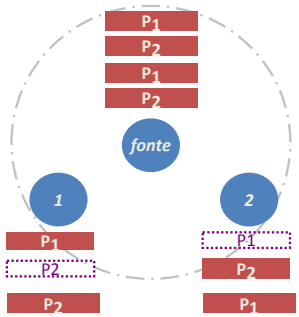

Figura 3. Retransmissão no roteamento tradicional. No total 4 mensagens são transmitidas.

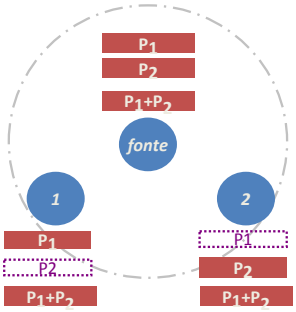

Figura 4. Retransmissão utilizando codificação em Rede. No total 3 mensagens são transmitidas.

No roteamento tradicional, os pacotes perdidos são retransmitidos. No nosso exemplo, o nó fonte terá que retransmitir os pacotes P_1 e P_2 e, se tiver sucesso, o nó 1 irá receber o pacotes P_2 que estava faltando e, de forma similar, o nó 2 irá receber o P_1. No total, considerando que haja sucesso na primeira retransmissão de cada pacotes, serão gastos no total 4 transmissões de pacotes na rede, como mostra a figura 3.

A Codificação em Rede permite que pacotes sejam combinados usando um operador lógico. Podemos combinar os pacotes utilizando o operador XOR (ou exclusivo). Um novo pacotes será criado onde cada bit do novo pacote é o resultado do ou exclusivo de cada bit entre P_1 e P_2. Observe que o tamanho do novo pacote é o mesmo de P_1 e P_2.

O nó fonte, ao invés de retransmitir os pacotes P_1 e P_2, pode retransmitir um pacote que é $P_1 XOR P_2$. O nó 1, ao receber esse pacote é capaz de aprender P_2 aplicando o operador ou exclusivo entre o pacote recebido e P_1, que ele havia recebido anteriormente visto que $P_2 = P_1 XOR (P_1 XOR P_2)$. Da mesma maneira, o nó 2 é capaz de receber P_1 quando receber o novo pacote e aplicar o operador ou exclusivo com o pacote P_2 que ele tinha. $P_1 = P_2 XOR (P_1 XOR P_2)$. Ao final, serão gastos no total 3 transmissões de pacotes na rede, como mostra a figura 4.

O exemplo ilustra bem o ganho de se utilizar Codificação em Rede para uma topologia com um salto. O número de mensagens na rede foi reduzido de 4 para 3. Para uma topologia maior e que são utilizados vários saltos, se faz necessário o desenvolvimento de um protocolo que consiga manter os ganhos introduzidos pela combinação de pacotes na rede.

4. Algoritmo

O CodeDrip surgiu da ideia de aplicar a técnica de Codificação em Rede no processo de disseminação de dados em uma Rede de Sensores Sem Fio e verificar os ganhos que ela traz. Em muitas situações é necessária a disseminação de vários dados na rede, e não apenas um. Nesse caso, essa técnica se torna interessante, pois existirão vários pacotes sendo transmitidos na rede para disseminar diferentes valores, e a combinação de alguns deles tornará o processo mais robusto a perda, uma vez que pacotes perdidos poderão ser recuperados se o nó sensor tiver recebido uma mensagem combinada e uma original.

Assim como o Drip, o CodeDrip utiliza TrickleTimers para transmitir periodicamente mensagens contendo os dados que estão sendo disseminados com o objetivo de que eles cheguem eventualmente a todos os nós sensores da rede. Porém, além das mensagens originais, algumas mensagens combinadas também serão transmitidas. Como o CodeDrip é agnóstico à topologia da rede, não existe custo adicional para adquirir e armazenar informações sobre ela. Por isso a política utilizada para decidir quando transmitir uma mensagem combinada é muito importante e afetará diretamente o desempenho do processo de disseminação.

As mensagens que serão enviadas e recebidas pelos nós sensores tiveram que ser adaptadas para que a técnica de Codificação em Rede pudesse ser aplicada. É necessário indicar no cabeçalho da mensagem qual é a combinação que ela contém para permitir o processo de decodificação. Cada dado a ser disseminado na rede possui um número inteiro de 1 byte como identificador único. Cada mensagem possui um conjunto de identificadores e o conteúdo a ser transmitido. O conjunto de indicadores é o custo adicional necessário na mensagem para que a codificação e decodificação possam funcionar. Uma mensagem original possui apenas um identificador diferente de zero, e seu conteúdo é o dado original a ser disseminado. Uma mensagem codificada possui dois ou mais identificadores diferentes de zero, e seu conteúdo é o resultado da operação XOR entre os dados cujos identificadores estão na lista dessa mensagem. Um número máximo de mensagens a serem combinadas precisa ser definido. Para os nossos testes ele foi definido como duas mensagens.

Cada nó sensor possui um conjunto de mensagens originais e um buffer de mensagens combinadas inicialmente vazios. Ao receber uma mensagem, o nó sensor irá verificar se ela é original ou combinada. Se for uma mensagem original, ele a armazena no seu vetor de mensagens originais e a partir desse momento ele enviará essa mensagem a cada vez que o TrickleTimer correspondente disparar. Se for uma mensagem combinada, o nó sensor verificará se ele consegue decodificar uma nova mensagem a partir das originais que ele já possui. Se isso não for possível, a mensagem será armazenada no buffer de mensagens combinadas até que cheguem mensagens que tornem possível a sua decodificação.

O protocolo utiliza o evento disparado pelos TrickleTimers para fazer o broadcast de uma mensagem. Cada mensagem original armazenada no nó sensor tem um temporizador. Quando este é disparado, o nó sensor enviará uma mensagem contendo o dado correspondente. Porém algumas vezes, em vez de simplesmente enviar esse dado sozinho, o nó sensor o combinará com alguma outra mensagem e enviará essa combinação. A probabilidade de enviar uma mensagem codificada em vez de uma mensagem original irá afetar o desempenho do protocolo.

Tanto no Drip quanto no CodeDrip muitas mensagens redundantes serão recebidas pelos nós sensores. Uma mensagem redundante é aquela que o nó sensor já possui. Um nó da rede receber várias vezes a mesma mensagem é um sinal de que a maioria de seus vizinhos já possuem essa mensagem. Então quanto mais redundância houver no recebimento de determinado dado, menor é a necessidade de enviá-lo em um tempo próximo. Para atrasar o envio desse dado, e consequentemente diminuir o número de mensagens enviadas em um determinado período de tempo, esses protocolos utilizam uma função provida pela interface do TrickleTimer que aumenta o intervalo de espera do temporizador, adiando o envio daquela mensagem. Com a possibilidade de receber mais de uma mensagem com apenas uma transmissão, o CodeDrip suprime o envio de mensagens desnecessárias mais rápido do que o Drip. Por exemplo: ele pode receber uma combinação de duas mensagens redundantes e adiar o temporizador dessas duas mensagens de uma só vez, enquanto o Drip precisaria de receber duas mensagens distintas para fazer a mesma coisa.

5. Simulação

Um conjunto de topologias de redes de sensores sem fio foi gerado para simular a disseminação de dados utilizando o protocolo Drip e o CodeDrip utilizando o TOSSIM. Uma topologia de rede é um grafo direcionado valorado, onde um vértice representa um nó sensor e uma aresta diz que existe comunicação entre os sensores que ela liga. O peso de cada aresta é a probabilidade da transmissão ser feita com sucesso.

O algoritmo que implementamos para gerar uma topologia recebe dois parâmetros: um número n que representa a quantidade de nós sensores na topologia gerada e um número d que representa a distância média de cada nó aos seus vizinhos. O algoritmo sorteia coordenadas x e y para o primeiro nó da rede, e insere o segundo nó a uma distância d do primeiro. Para inserir o restante dos nós da topologia, são sorteadas novas coordenadas x e y para cada um, e essas coordenadas têm que satisfazer duas condições: 1) a distância do novo nó até todos os que já estão na topologia tem que ser maior que $d - 5$; 2) a distância do novo nó até pelo menos dois dos nós que já estão na topologia tem que ser menor que $d + 5$. Se uma das duas condições não for satisfeita, outras coordenadas são sorteadas. Caso contrário, o novo nó é adicionado. A saída do algoritmo será um arquivo de configuração de uma topologia para o TOSSIM. Nela, distância de cada nó a cada um de seus nós vizinhos está no intervalo $[d - 5, d + 5]$. Esse algoritmo garante que a topologia será conexa e que cada nó terá pelo menos dois vizinhos com os quais ele consegue se comunicar. O objetivo disso é que todos os nós da rede possam receber as mensagens que estão sendo disseminadas.

Nós geramos topologias que têm de 10 a 100 nós sensores, variando esse número de dez em dez. Nas simulações que fizemos, consideramos a perda de pacotes sendo uniforme em toda a rede. Isso significa que todo pacote enviado tem a mesma probabilidade de se perder. Fizemos simulações variando o valor dessa probabilidade de 10 a 70 por cento, variando essa probabilidade de dez em dez.

Em cada simulação, considera-se o processo de disseminação iniciado a partir do momento que primeiro nó sensor recebe o primeiro dado a ser disseminado. Conta-se a partir de então toda mensagem que for transmitida na rede, e considera-se esse o tempo de início. A simulação acaba quando todos os nós tiverem recebido todas as mensagens

sendo disseminadas, sendo o momento em que o último nó sensor consegue a última mensagem que o faltava definido como o tempo final. Então são registrados qual foi esse intervalo de tempo e qual o número total de mensagens que foram transmitidas. Em cada simulação três dados diferentes foram disseminados para a rede.

6. Resultados

Utilizando a técnica de Codificação na Rede, o resultado esperado é que processo de disseminação precise enviar menos mensagens para que todos os nós da rede recebam todos os três dados que estão sendo disseminados, e que esse ganho seja maior o quanto pior for a qualidade de transmissão na rede.

Os gráficos apresentados na figura 5 mostram o desempenho dos dois protocolos (Drip e CodeDrip) à medida que a perda de pacotes na rede aumenta em uma topologia com 100 nós sensores. Nos gráficos, cada ponto representa a média dos valores obtidos em 10 simulações, e a barra de erro representa o desvio padrão desse conjunto de dados. Nesses gráficos é possível observar que o CodeDrip é menos afetado pela perda de pacotes durante a disseminação e precisa de transmitir menos mensagens do que o Drip para completar o processo. O tempo gasto para que os dados cheguem a todos os nós sensores também foi, em média, menor no CodeDrip.

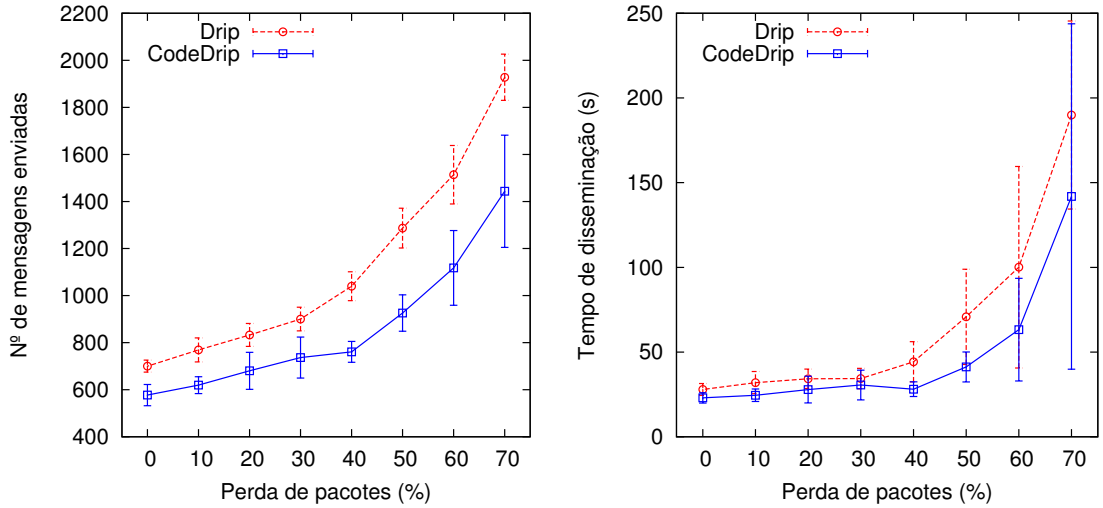

Figura 5. Número de mensagens enviadas e tempo de disseminação para o Drip e o CodeDrip à medida que a perda de pacotes aumenta em uma topologia com 100 nós sensores.

Os gráficos da figura 6 mostram o desempenho dos protocolos quando mantemos fixa uma perda média de 50% dos pacotes transmitidos e aumentamos o número de nós sensores na rede. A adição de nós na rede faz com que o desempenho do CodeDrip em comparação ao Drip melhore cada vez mais. Logo, quanto mais nós a rede tiver, maior será a vantagem em se utilizar o CodeDrip ao invés do Drip para fazer a disseminação de dados.

O gráfico da figura 7 mostra a porcentagem de nós sensores da rede que já receberam todas as mensagens em função do tempo de disseminação em uma rede. Observe que a perda de pacotes causa uma dificuldade para o protocolo Drip logo quando o processo

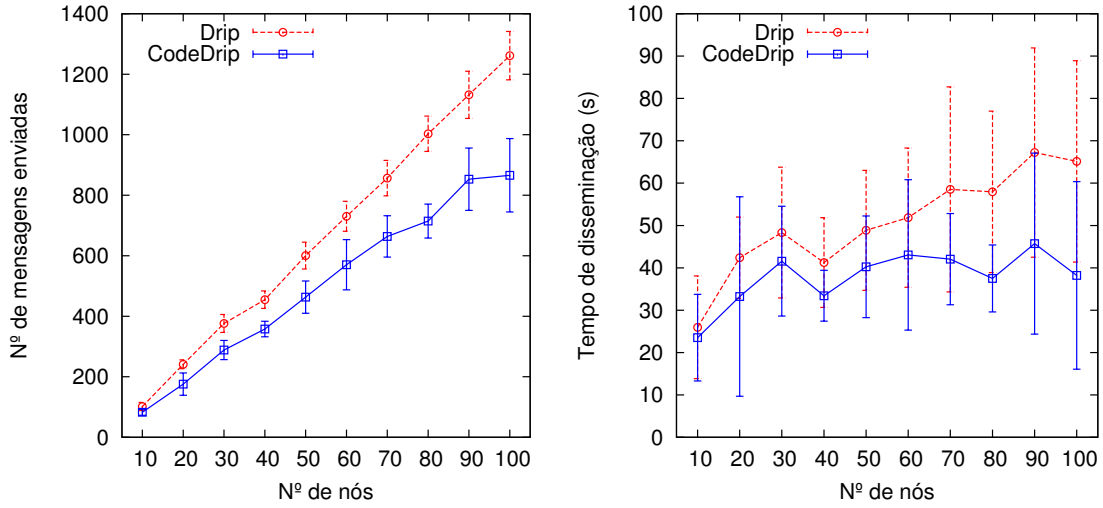

Figura 6. Número de mensagens enviadas e tempo de disseminação para o Drip e o CodeDrip à medida que o número de nós aumenta em uma topologia com perda média de 50% dos pacotes.

de disseminação está começando, pois existem existem poucos nós transmitindo dados e vários deles são perdidos. No CodeDrip mesmo quando a quantidade de nós transmitindo é pouca, os dados fluem pela rede de maneira mais uniforme. Nesse gráfico também podemos observar que o tempo de disseminação se prolonga bastante por causa de apenas um nó sensor que não tenha recebido alguma mensagem, e a combinação de mensagens aumenta a probabilidade desse nó sensor conseguir o dado que lhe falta, terminando a disseminação mais cedo.

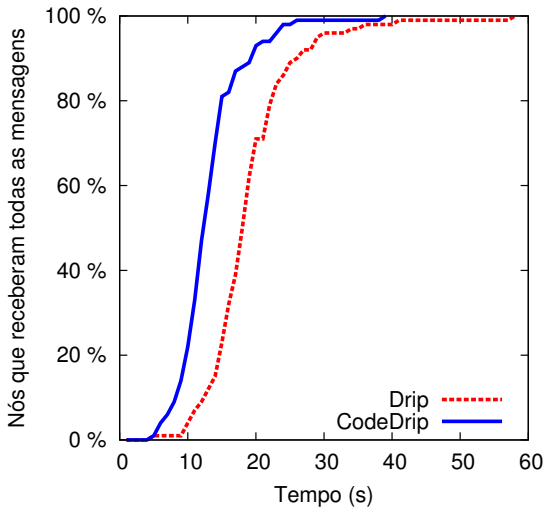

Figura 7. Porcentagem de nós da rede que já receberam todas as mensagens à medida que o tempo passa. Topologia com 100 nós sensores e 60% de chance de perda.

No CodeDrip são implementadas duas políticas de decisão probabilísticas. A primeira é para decidir se a mensagem enviada será original ou combinada. Essa decisão é feita imediatamente antes de enviar a mensagem, ou seja, quando o TrickleTimer corres-

pondente é disparado. A segunda política decide o que fazer quando o nó sensor recebe uma mensagem que ele já possui. Receber com muita frequência uma mesma mensagem indica que seus vizinhos já a possuem. Logo, é razoável que se esse nó sensor não precise ter pressa para enviar essa mensagem, já que provavelmente ela não fará tanta falta e não enviá-la agora prejudicará pouco o processo de disseminação. Então o nó sensor pode decidir suprimir, ou seja, adiar o envio dessa mensagem.

A probabilidade com que essas decisões são tomadas irá afetar diretamente o desempenho do protocolo. Para descobrir qual é a melhor configuração de probabilidades do CodeDrip foram feitas simulações aplicando diferentes valores para as mesmas. Os resultados são mostrado nas figuras 8 e 9.

A figura 8 mostra como o desempenho do CodeDrip é afetado à medida que aumenta-se a probabilidade usada para combinar uma mensagem antes de enviá-la. Quando essa probabilidade passa de 30%, a quantidade de mensagens combinadas enviadas passa a prejudicar o processo de disseminação. Se um nó sensor recebe apenas mensagens combinadas, ele não consegue decodificá-las pois, para isso, é preciso que ele tenha ou receba uma mensagem original. Com essa probabilidade alta, muitos sensores ficarão apenas com combinações armazenadas esperando chegar mensagens originais, que são enviadas com menos frequência. Logo, com base nos gráficos da figura 8, o melhor é que a probabilidade de enviar uma mensagem combinada seja 30%.

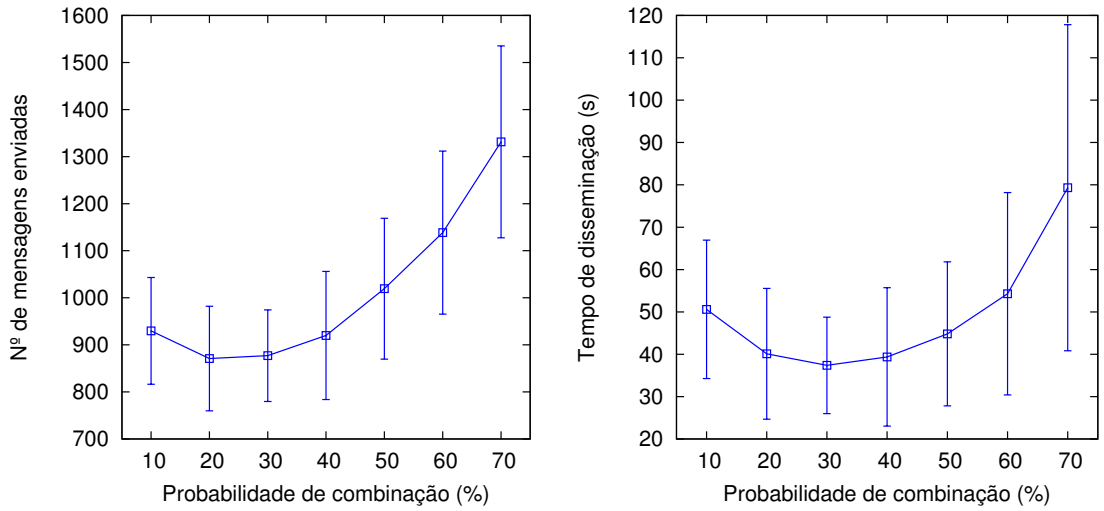

Figura 8. Quantidade de mensagens enviadas e tempo de disseminação em função da probabilidade de combinar pacotes na hora de enviar.

Na figura 9 estão representados resultados dos experimentos feitos aumentando a probabilidade de suprimir o envio de mensagens que são redundantes para aquele nó sensor. O número de mensagens enviadas diminui até a probabilidade de 50% ao preço da disseminação ser um pouco mais lenta. Como essa diferença de tempo não é tão significativa, a economia de mensagens é vantajosa. A partir de 50% o número de mensagens se estabiliza e o tempo de disseminação aumenta. Isso porque mensagens que seriam importantes para um determidado vizinho do nó sensor são suprimidas porque ele recebe essa mesma mensagem de outros vizinhos que a possuem. Sendo assim, a probabilidade que gerará melhores benefícios é de 50%.

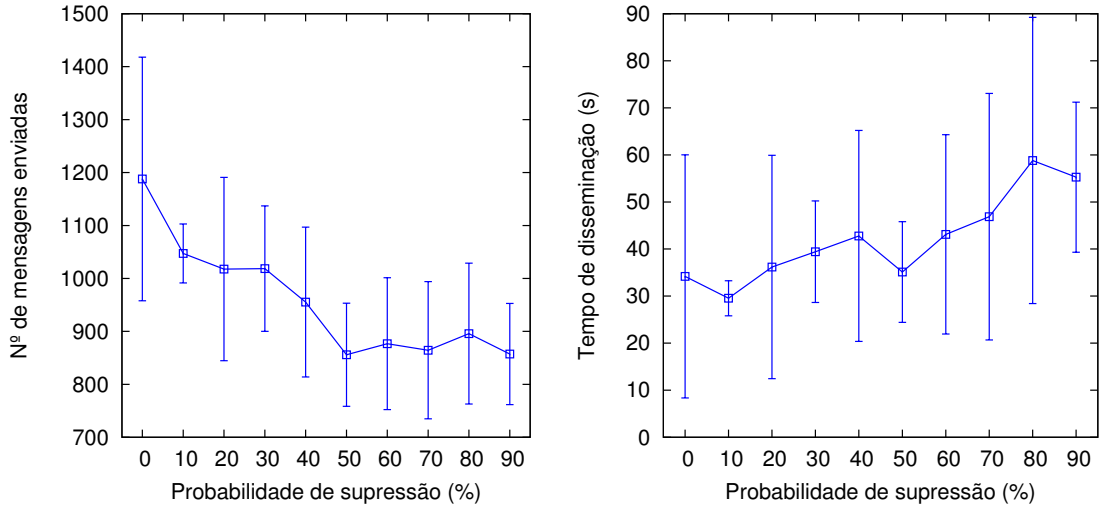

Figura 9. Quantidade de mensagens enviadas e tempo de disseminação em função da probabilidade de suprimir o envio de mensagens redundantes.

7. Conclusão

Neste artigo foi apresentado o CodeDrip, um protocolo de disseminação de dados em redes de sensores sem fio. A ideia principal desse novo protocolo é aplicar a técnica de Codificação em Rede no processo de disseminação, e com isso economizar transmissões de pacotes. Vale lembrar que reduzindo o número de mensagens, reduz-se o consumo de energia da rede, uma importante métrica em RSSFs. Os resultados obtidos nas simulações mostraram que o CodeDrip envia menos mensagens do que o Drip. Além disso, o tempo de disseminação também reduziu, o que mostra que o CodeDrip é mais robusto a falhas de comunicação que o Drip.

O custo adicional necessário para implementar o CodeDrip são identificadores que tem que ser mandados junto com as mensagens para que possa ser possível decodificá-las, e um buffer para armazenar combinações de mensagens que não puderam ser decodificadas. O custo adicional é facilmente controlado definindo um número máximo de mensagens que podem ser combinadas e um tamanho máximo do buffer.

Os ganhos verificados nesse trabalho referem a um custo adicional mínimo no conteúdo de uma mensagem, permitindo a combinação de apenas duas mensagens originais, e um buffer que permite armazenar até três mensagens combinadas. Logo, a economia de energia nos nós sensores por fazer menos transmissões tem o custo de um pouco de memória do dispositivo. Uma aplicação precisa estar utilizando praticamente toda a memória do dispositivo para que esse custo se torne crítico a ponto de não poder usar o CodeDrip por causa disso.

Como possíveis trabalhos futuros, pode-se apontar um estudo mais aprofundado sobre o comportamento do protocolo CodeDrip para casos em que mais mensagens são disseminadas pela rede. Outro possível trabalho é desenvolvimento de novas políticas de combinação de mensagens, fazendo combinações mais complexas a fim de obter um desempenho ainda melhor, analisando as vantagens e desvantagens de cada estratégia.

Agradecimentos

Gostaríamos de agradecer à FAPEMIG, CAPES, CNPq.

Referências

Ahlswede, R., Cai, N., Li, S.-Y., and Yeung, R. (2000). Network information flow. *Information Theory, IEEE Transactions on*, 46(4):1204–1216.

Chandanala, R. and Stoleru, R. (2010). Network coding in duty-cycled sensor networks. In *Networked Sensing Systems (INSS), 2010 Seventh International Conference on*, pages 203–210.

Cui, T., Chen, L., and Ho, T. (2010). On distributed scheduling in wireless networks exploiting broadcast and network coding. *Trans. Comm.*, 58(4):1223–1234.

Deb, S. (2005). Network coding for wireless applications: a brief tutorial. In *International Workshop on Wireless Ad-hoc Networks (IWWAN)*.

Gay, D., Welsh, M., Levis, P., Brewer, E., Behren, R. V., and Culler, D. (2003). The nesc language: A holistic approach to networked embedded systems. In *In Proceedings of Programming Language Design and Implementation (PLDI)*, pages 1–11.

Hui, J. W. and Culler, D. (2004). The dynamic behavior of a data dissemination protocol for network programming at scale. In *In Proceedings of the 2nd international*, pages 81–94. ACM Press.

Katti, S., Rahul, H., Hu, W., Katabi, D., Medard, M., and Crowcroft, J. (2008). XORs in the Air: Practical Wireless Network Coding. *IEEE/ACM Transactions on Networking*, 16(3):497–510.

Keller, L., Karaagac, A., Fragouli, C., and Argyraki, K. (2011). Evaluation of network coding techniques for a sniper detection application. In *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt), 2011 International Symposium on*, pages 327–333.

Lee, U., Park, J.-S., Yeh, J., Pau, G., and Gerla, M. (2006). Code torrent: content distribution using network coding in vanet. In *MobiShare '06: Proceedings of the 1st international workshop on Decentralized resource sharing in mobile computing and networking*, pages 1–5, New York, NY, USA. ACM.

Levis, P. and Gay, D. (2009). Tinyos programming.

Levis, P., Lee, N., Welsh, M., and Culler, D. (2003). Tossim: Accurate and scalable simulation of entire tinyos applications.

Levis, P., Patel, N., Culler, D., and Shenker, S. (2004). Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In *In Proceedings of the First USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI)*, pages 15–28.

Li, S.-Y., Yeung, R., and Cai, N. (2003). Linear network coding. *Information Theory, IEEE Transactions on*, 49(2):371–381.

Ruiz, L. B., Correia, L. H., Vieira, L. F. M., Macedo, D. F., Nakamura, E., Figueiredo, C. M., Vieira, M. A. M., Mechelane, E. H., Camara, D., Loureiro, A. A. F., Nogueira,

J. M., and da Silva Jr., D. C. (2004). Arquitetura para redes de sensores sem fio. *Minicurso, XXII Congresso da SBRC*.

Tolle, G. and Culler, D. (2005). Design of an application-cooperative management system for wireless sensor networks. In *Second European Workshop on Wireless Sensor Networks (EWSN) Istanbul, Turkey, January 31 - February 2, 2005.*, pages 121–132.

Vieira, L., Misra, A., and Gerla, M. (2007). Performance of network-coding in multi-rate wireless environments for multicast applications. In *Military Communications Conference, 2007. MILCOM 2007. IEEE*, pages 1–6. IEEE.

Wang, X., Wang, J., and Xu, Y. (2010). Data dissemination in wireless sensor networks with network coding. *EURASIP Journal on Wireless Communications and Networking*, 2010(1):465915.

Zhang, J. and Zhang, Q. (2009). Cooperative network coding-aware routing for multi-rate wireless networks. *IEEE INFOCOM 2009 The 28th Conference on Computer Communications*, pages 181–189.