Codificação em Rede (*Network Coding*) em Redes de Salto de Canal Sincronizado com o Tempo (TSCH)

Marlon da Cruz Carvalho[1], **Luiz F. M. Vieira**[1], **Marcos A. M. Vieira**[1]

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

{marlon.cruz,lfvieira,mmvieira}@dcc.ufmg.br

Abstract. *This article presents the CodeTSCH protocol, which uses Network Coding in TSCH networks (Time-Slotted Channel Hoping) to optimize message delivery and reduce the number of slots used. This process makes it possible for two messages to be combined into just one packet, allowing for a smaller amount of messages circulating on the network, as well as greater robustness to message loss. The CodeTSCH was compared in simulations with the state-of-the-art TSCH protocol. The results demonstrate that CodeTSCH allows for message delivery with an improvement of up to 50% compared to the state-of-the-art TSCH protocol in networks with packet loss.*

Resumo. *Este artigo apresenta o protocolo CodeTSCH, que utiliza a Codificação em Rede (Network Coding) em redes TSCH (Time-Slotted Channel Hoping) para otimizar a entrega de mensagens e diminuir o número de slots utilizados. Esse processo possibilita que duas mensagens sejam combinadas em apenas um pacote, possibilitando uma menor quantidade de mensagens circulando na rede, bem como uma maior robustez à perda de mensagens. Comparou-se o CodeTSCH em simulações com o protocolo estado da arte TSCH. Os resultados demonstram que o CodeTSCH permite uma entrega das mensagens com melhoria de até 50% em comparação com o protocolo estado da arte TSCH em redes com perdas de pacotes.*

1. Introdução

As redes sem fio estão amplamente difundidas atualmente, com milhões de dispositivos conectados e a gama de aplicações aumentando com grande frequência. Uma das aplicações que vem ganhando bastante mercado são os dispositivos de Internet das Coisas (IoT, do inglês *Internet of Things*), conectando eletrodomésticos, dispositivos de automação, sensores, entre outros. De acordo com Santos et al. [Santos et al. 2016], a Internet das Coisas nada mais é que uma extensão da Internet atual, que proporciona aos objetos do dia-a-dia (quaisquer que sejam), que possuam capacidade computacional e de comunicação, se conectarem à Internet. A conexão com a rede mundial de computadores viabilizará, primeiro, controlar remotamente os objetos e, segundo, permitir que os próprios objetos sejam acessados como provedores de serviços.

Um dos problemas dos dispositivos IoT é a limitação de energia desses aparelhos, devido ao seu reduzido tamanho. Por isso, é importante diminuir o consumo de energia desses dispositivos, bem como otimizar a transmissão de dados para minimizar o tempo de utilização do rádio, um grande consumidor de energia. Existe um método de acesso

ao canal de transmissão que vem sendo adotado principalmente para as redes de sensores sem fio, chamado de *Time Slotted Channel Hopping*, ou Salto de Canais por Intervalo de Tempo (TSCH)[Thubert 2019], onde a utilização do canal pelos elementos da rede é dividida por *slots* de tempo e cada elemento somente acessa a rede durante o período de tempo atribuído a ele.

No âmbito das redes sem fio, existe a Codificação em Rede (*Network Coding*) que permite diminuir a repetição de envios devido à perda de mensagens e otimizar o envio de mensagens para vários destinatários (pacotes *multicast*). Para isso, os nós que retransmitem as mensagens realizam uma codificação de mais de uma mensagem em um só pacote e, dessa forma, o receptor consegue decodificar N mensagens se tiver N pacotes, mesmo que não sejam os pacotes específicos das mensagens de 1 a N.

Esse artigo propõe o projeto e a implementação de um algoritmo de Codificação em Rede em uma rede TSCH utilizando o simulador OpenWSN para a diminuição do número de *slots* verificando-se a melhora no desempenho e na vazão da rede. A literatura existente apresenta as vantagens da utilização da Codificação em Rede [Li et al. 2003] [Fragouli et al. 2006] [dos Santos Ribeiro Júnior et al. 2017] e as vantagens da utilização das redes TSCH para Internet das Coisas [Hermeto et al. 2017] [Minet et al. 2017] [Boccadoro et al. 2016]. Esse artigo propõe combinar as duas tecnologias para obter um ganho de desempenho com a Codificação em Rede e além disso ainda manter os benefícios de economia de energia e robustez das redes TSCH. Pressupõe-se que com a codificação de várias mensagens em uma só, o meio de transmissão se torne menos congestionado, permitindo dessa forma diminuir a quantidade de *slots* utilizados, e assim aumentar a vazão geral da rede sem prejudicar sua robustez e alta imunidade a interferências. Essa abordagem foi comparada com o TSCH sem a utilização da Codificação em Rede para verificar se há melhora na vazão e na eficiência da rede, bem como uma diminuição no tempo de entrega das mensagens.

As principais contribuições científicas deste trabalho são as seguintes. Primeiro, é apresentado o projeto e a implementação do CodeTSCH. Em segundo lugar, é estudado o desempenho do CodeTSCH através de extensa simulação com o simulador estado-da-arte OpenWSN. Terceiro, é comparado o CodeTSCH com o TSCH e quantificado qual é o ganho da Codificação em Rede. É mostrado que a Codificação de Rede é útil mesmo para redes TSCH.

A próxima seção desse artigo desenvolve com mais detalhes o problema estudado, seguida pela descrição da solução proposta para resolver tal problema. Em seguida é apresentada a motivação para o projeto e a fundamentação teórica sobre as tecnologias utilizadas na execução da proposta. Na seção 6 encontram-se os trabalhos relacionados e a metodologia do projeto é descrita na sequência. Finalmente, nas seções 8, 9 e 10 encontram-se respectivamente o desenvolvimento do projeto, os resultados encontrados e a conclusão.

2. Problema estudado

O TSCH (*Time Slotted Channel Hopping*) é um método de acesso ao meio utilizado por redes de meio compartilhado [Thubert 2019]. Seu funcionamento é baseado na criação de uma tabela de *slots* onde cada nó acesse o meio apenas nos *slots* designados à comunicação com o nó destino. O problema dessa abordagem é que muitas vezes o

nó não possui dados para enviar e então o *slot* de tempo fica vazio, ou, por outro lado, o nó possui vários dados para enviar e não consegue enviar todos no intervalo de tempo do *slot* alocado a ele, levando um tempo de espera para que chegue o próximo *slot* atribuído a ele e então a comunicação possa continuar.

Esse problema é agravado proporcionalmente à quantidade de nós presentes na rede, visto que a cada nó adicionado, é necessário alocar um novo *slot* de tempo para comunicação desse novo nó e então os *slots* consecutivos de um mesmo nó vão se tornando cada vez mais espaçados temporalmente.

3. Solução Proposta

A solução proposta nesse artigo, chamada de CodeTSCH consiste em utilizar Codificação em Rede para diminuir o número de mensagens circulantes pela rede TSCH e, com isso, diminuir o número de *slots* utilizados, permitindo assim que os pacotes sejam enviados com maior velocidade. A Codificação em Rede é uma estratégia de transmissão onde os dados são codificados e decodificados para aumentar a robustez e velocidade da rede. São aplicado algoritmos aos pacotes de mensagens de forma que alguns pacotes são transmitidos codificados em conjunto com outros. Isso diminui a quantidade de pacotes trafegando pela rede e assim a vazão aumenta, já que mais dados conseguem ser transmitidos em um só pacote. Ressalta-se que a mensagem codificada Mr resultante de quaisquer mensagens M1 e M2 possui o mesmo tamanho de uma única mensagem, caso elas possuam o mesmo tamanho ou o tamanho da maior delas em caso de tamanhos diferentes.

Da mesma forma, com menos *slots* em uso, o tempo entre dois *slots* consecutivos do mesmo nó diminui, o que contribui para aumentar a vazão e a eficiência da rede.

4. Motivação

Com o advento da Internet das Coisas (IoT)[Gubbi et al. 2013], bem como *smart-houses* (casas inteligentes), além de projetos dos chamados Mineração 4.0 e Agricultura 4.0, há a utilização cada vez maior de dispositivos conectados para sensoriamento, telemetria, automação, entre outros. Esse grande número de dispositivos precisa enviar seus dados em ”tempo real”, e uma quantidade muito grande de dispositivo pode causar congestionamento e atrasar a entrega de mensagens. Por isso é importante uma solução que otimize o tráfego na rede para evitar congestionamentos.

O TSCH é um solução que possui a robustez necessária para minimizar interferências [Palattella et al. 2013] (que podem ocorrer em ambientes com muitos equipamentos conectados à rede) como no Bluetooth, porém possui um alcance bem maior que este. Além disso, o TSCH utiliza bem menos energia que o Wi-Fi, por isso ele é uma boa escolha para nós sensores e outros dispositivos IoT onde a economia de energia é um fator preponderante.

Essas características tornam o TSCH adequado para uma solução de IoT, mesmo sendo um protocolo mais antigo, por ainda oferecer uma economia de energia aliada a uma maior imunidade a interferências devido à utilização de canais alternados.

5. Fundamentação Teórica

5.1. TSCH (*Time-Slotted Channel Hoping*)

O TSCH é um método de acesso ao canal utilizado por redes de meio compartilhado ([Duquennoy et al. 2017a]). É utilizado por dispositivos de baixo consumo de energia em comunicações sem fio.

No TSCH, o canal de comunicação é dividido em canais e o tempo é dividido em *slots*. Uma possível tabela de *slots* para uma rede com seis nós (nomeados de A a F) pode ser visualizada na figura 1. Cada *slot* de tempo tem tipicamente a duração de 10ms.

Em cada *slot* de tempo é possível a utilização do canal para a transmissão de dados entre um par de dispositivos. Esses *slots* são sincronizados para todos os dispositivos, e, por isso, estes podem entrar em estado de hibernação para economizar energia e só despertar nos *slots* de tempo destinados à transmissão para aquele nó.

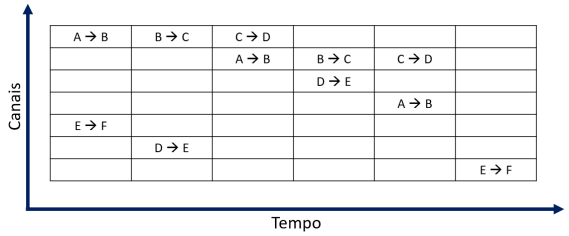


Figura 1. Tabela de divisão de canais por tempo

Além disso, o TSCH também utiliza o salto de canal, conforme observado na figura 1. A sequência de transmissão se inicia em uma determinada frequência (primeira linha da tabela), e, ao se repetir o padrão, é utilizada uma frequência diferente (segunda linha da tabela). Isso permite contornar problemas de transmissão por interferências ou problemas no canal, adicionando robustez ao TSCH.

5.2. Codificação em Rede (*Network Coding*)

Em redes sem fio, os nós funcionam como retransmissores, enviando as mensagens recebidas para os outros nós, até encontrar o nó-destino. De acordo com [Ahlswede et al. 2000], Codificação em Rede é a codificação de várias mensagens em uma, realizado por algum nó intermediário, de forma a minimizar a quantidade de mensagens na rede.

A utilização da Codificação em Rede permite agrupar duas ou mais mensagens por um nó intermediário de forma que este realize a retransmissão de apenas uma mensagem, ao invés das duas ou mais recebidas individualmente. Isso diminui a quantidade de mensagens na rede e, segundo [Fragouli et al. 2006], os benefícios energéticos podem chegar à ordem de $log(n)$, onde n é o número de nós presentes na rede.

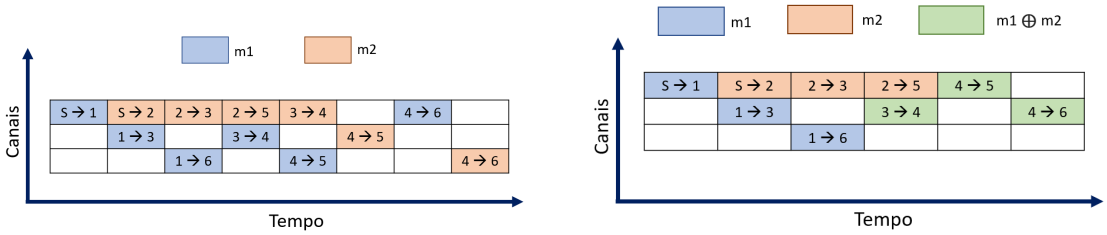
A codificação de rede permite que os pacotes sejam combinados usando um operador lógico. É possível combinar pacotes usando, por exemplo, o operador XOR (”ou exclusivo”ou $\oplus$). Um novo pacote é criado executando um XOR bit a bit de cada bit nos pacotes P1 e P2. O novo pacote é do mesmo tamanho dos pacotes P1 e P2.

Um comparativo simples da utilização da Codificação em Rede pode ser visto na figura 2. A figura 2(a) mostra uma rede sem a utilização de Codificação em Rede, as

(a) Sem NC. (b) Com NC.

Figura 2. Comparativo de mensagens trafegadas com e sem a utilização da Codificação em Rede (NC)

mensagens estão sendo transmitidas individualmente a cada nó. Já na figura 2(b), os nós 3 e 4 utilizam a Codificação em Rede para combinar as duas mensagens m1 e m2 em uma só antes de realizar a retransmissão, conseguindo transmitir os mesmos dados porém com a utilização de menos mensagens. O símbolo $\oplus$ se refere ao XOR (ou exclusivo) entre os bits de m1 e m2.



(a) Sem codificação. (b) Com codificação.

Figura 3. Tabela de divisão de canais por tempo com e sem a utilização da Codificação em Rede

Observa-se na figura 3 que a utilização da Codificação em Rede na configuração da figura 2(b) possibilita que as mensagens sejam entregues utilizando uma menor quantidade de *slots*, permitindo uma maior vazão na rede. Comparando a figura 3(a) com a figura 3(b), nota-se que naquela existem mais colunas (*slots*) utilizados que nesta, demonstrando que sem a codificação são necessários mais *slots* para entregar a mesma quantidade de mensagens. Ressalta-se que essa tabela representa apenas uma iteração do *timeframe*, como um *snapshot*, sendo que a execução desse *timeframe* ocorre de forma cíclica.

Outra possível forma de realizar a Codificação em Rede é através de um esquema chamado *Random Linear Coding* (Codificação Linear Aleatória), que consiste na escolha dos coeficientes lineares pelos próprios nós de forma independente e uniforme sobre todos os elementos do campo finito. O vetor de codificação é então transmitido como um cabeçalho adicional, com o resultado da codificação como *payload*. O fato de escolher os coeficientes aleatoriamente em um campo finito suficientemente grande garante às mensagens codificadas serem linearmente independentes com uma probabilidade alta o suficiente que permita sua utilização na prática. [Ho et al. 2006] demonstraram que essa abordagem permite atingir a capacidade da rede com uma probabilidade que se aproxima

exponencialmente em função do tamanho do código escolhido. Outra vantagem dessa abordagem é sua eficiência independentemente do conhecimento pelos nós da topologia da rede, o que economiza processamento nos nós.

A abordagem proposta por esse projeto é a codificação de mensagens utilizando a operação XOR (*Exclusive-OR*) bit a bit entre duas mensagens. Essa operação é logicamente barata (geralmente apenas um ciclo de *clock*) o que possibilita sua execução em nós com capacidade de processamento e memória limitados.[Li et al. 2003] provaram que a utilização de uma codificação linear é suficiente para atingir a capacidade máxima de uma rede, o que corresponde ao seu fluxo máximo.

5.3. OpenWSN

O OpenWSN [Watteyne et al. 2012] é um projeto criado por um time de desenvolvedores na Universidade Berkeley com a intenção de se tornar um padrão e uma plataforma *Open-Source* para projetos IIoT (*Industrial Internet of Things* - Internet das Coisas Industrial) [Zhang et al. 2008]. O projeto prevê a camada MAC com a utilização nativa do TSCH, além da utilização do padrão 6Tisch, RPL, 6LoWPAN, UDP e/ou CoAP.

Dentro do projeto do OpenWSN, houve também o desenvolvimento de uma família de nós sensores, chamados OpenMote, que implementam o OpenWSN e oferecem um ecossistema de hardware completo e de código aberto para a implementação de projetos IoT ([Vilajosana et al. 2015]). Esse ecossistema inclui também uma *suite* de aplicativos para desenvolvimento e *drivers* para as principais soluções de IoT de código aberto.

Embora existam outras opções de simuladores, inclusive mais modernas, como o TSCH-Sim [Elsts 2020], a escolha do OpenWSN ocorreu pela possibilidade de realizar os testes com nós reais, através da plataforma OpenMote, aproveitando o mesmo código e solução implementada para as simulações, o que será realizado em trabalhos futuros.

6. Trabalhos Relacionados

Em relação ao TSCH, Duquennoy et al. [Duquennoy et al. 2017a] apresentou alguns desafios para a implementação do 6Tisch (implementação de IPv6 sobre TSCH) no Contiki e então avaliou o desempenho do TSCH, comparando-o com o estado-da-arte dos protocolos MAC. A conclusão foi que o TSCH é mais confiável e com melhor desempenho quando o escalonamento é corretamente configurado.

Hermeto et al. [Hermeto et al. 2017] realizou um *survey* onde apresentou os protocolos baseados em salto de canal e divisão de tempo, detalhou os requisitos para os algoritmos de escalonamento, bem como os limites e utilizações ideais desses algoritmos. Cita também que a Internet das Coisas Industrial precisa respeitar garantias estritas, o que leva a um maior cuidado na escolha dos protocolos utilizados para transmissão. O TSCH então se baseia em uma distribuição estrita de transmissões para tornar o protocolo determinístico e eliminar a maioria das colisões.

Duquennoy et al. 2017b[Duquennoy et al. 2017b], em seu artigo *Five-Nines Reliable Downward Routing in RPL*, realizou alguns experimentos com o protocolo RPL para tentar atingir uma taxa de confiabilidade de cinco noves, ou seja, 99,999%. Na camada MAC, foi utilizado o TSCH com apenas um *slot* com um *slotframe* de tamanho 1. Os

nós acordam em todos os *slots* (10 ms) para buscar transmissões. Foi utilizado o Contiki para realizar as simulações, além de testes em *test-beds* (conjunto de sensores disponíveis online para realização de testes).

O TSCH é citado como a opção certa de escolha para controle de acesso ao meio para aplicações industriais, devido à sua resiliência a interferência, por distribuir a comunicação no espectro do tempo e da frequência. ([D. Hauweele and Papadopoulos 2020]). De acordo com o padrão, o TSCH possui um *slotframe* com 101 *timeslots*, cada um com 10ms de duração, e a quantidade de canais varia de acordo com a frequência utilizada, sendo por exemplo 16 canais na banda de 2.4 Ghz.

Palattella et al. [Palattella et al. 2012] apresentou uma abordagem diferente para o TSCH, utilizando os atributos de coloração de grafos aplicados à topologia e ao tráfego da rede para otimizar o escalonamento dos *slots*. Essa abordagem foi chamada de TASA (*Traffic Aware Scheduling Algorithm*). As simulações preliminares demonstraram a viabilidade e a efetividade do algoritmo proposto.

A codificação em rede foi utilizada por dos Santos Ribeiro Júnior et al. [dos Santos Ribeiro Júnior et al. 2017], que demonstrou nesse trabalho um aumento na velocidade de disseminação de mensagens e uma menor quantidade de pacotes transmitidos quando o protocolo Drip, considerado o estado da arte, é utilizado em conjunto com a codificação em rede.

Vieira e Vieira [Vieira and Vieira 2017] demonstraram que a codificação em rede pode otimizar as redes 5G e a comunicação D2D (*device-to-device* - Dispositivo para dispositivo), sendo possível até mesmo dobrar a vazão da rede, além de aumentar a robustez e diminuir o tempo para disseminação das mensagens.

7. Metodologia

A metodologia consiste em implementar um algoritmo para codificar as mensagens usando Codificação em Rede e comparar a melhora de desempenho com a utilização dessa abordagem comparada com a abordagem tradicional (mensagens enviadas sem a utilização da Codificação em Rede).

A proposta é diminuir o número de mensagens circulantes na rede combinando várias mensagens em uma só quando da retransmissão pelos nós intermediários e, assim, diminuir a quantidade de *slots* utilizados, possibilitando com isso um aumento da vazão global dessa rede.

Foi verificado experimentalmente através de simulações qual o percentual de melhora na diminuição do número de *slots*, e se isso permitiria uma melhora global no desempenho da rede, dado que uma menor quantidade de *slots* em uso poderia permitir um menor intervalo entre as transmissões consecutivas do mesmo nó.

Essa implementação foi realizada no simulador OpenWSN, utilizando-se o controlador padrão do TSCH embutido neste e então foi adicionado o algoritmo de Codificação em Rede projetado para codificar as mensagens retransmitidas pelos nós.

Após implementado, foi realizada a comparação da vazão utilizando a mesma configuração de rede mas sem a Codificação em Rede para verificar o ganho da

codificação, se existente. Foi então avaliada a vazão média das duas abordagens para verificar se a Codificação em Rede realmente traz melhorias de vazão conforme citado na literatura.

A abordagem utilizada foi implementar uma topologia com Codificação em Rede e replicar a mesma topologia sem codificação, para que seja possível realizar a comparação dos resultados de ambos os sistemas.

Para combinar mensagens, diferentes protocolos de codificação de rede usam diferentes operadores. CodeTSCH usa o operador XOR ($\oplus$), que é um campo de Galois de 2, em vez de um campo finito mais complexo. Esta escolha permite que o CodeTSCH seja executado com eficiência em nós com restrição de recursos, visto que na maioria das CPUs, o operador XOR é apenas uma instrução de hardware.

8. Desenvolvimento

8.1. OpenVisualizer

O OpenVisualizer é um simulador do OpenWSN, baseado em C e Python. Ele permite simular redes com topologias pré-definidas, bem como atribuir o PDR (*Packet Delivery Ratio* - Taxa de Entrega de Pacotes) entre cada link. Ele possui também um visualizador web que permite a edição, adição ou exclusão de links, bem como a verificação do status de cada nó, a tabela de roteamento, o IP atribuído a cada nó, entre outras informações.

O simulador permite atribuir a quantidade de nós e definir um deles como o nó raiz, que será responsável por gerenciar o escalonamento e a agenda de *slots* da rede TSCH. Para cada nó o simulador abre uma *thread* independente para execução, simulando um nó independente dos demais.

8.2. Implementação da codificação em rede

A codificação foi implementada na camada de aplicação, de forma que todos os nós (com exceção do nó raiz) enviassem um pacote a cada intervalo de tempo definido (definiu-se 30 segundos nos testes realizados.) para cada um dos nós da rede, em um *unicast* múltiplo simulando um *multicast*.

A codificação ou não do pacote era definida a cada mensagem por algumas variáveis, a saber:

- Existe algum pacote em *buffer* para ser codificado junto ao pacote a ser enviado?
- Existe uma probabilidade (cp) de combinar pacotes. Antes de enviar cada pacote, é executado um gerador de números aleatórios para determinar se o pacote será combinado. Nos experimentos, a probabilidade de cp foi variada entre 0 e 50%. Isso foi feito para gerar aleatoriedade na codificação e verificar qual o melhor percentual de codificação, haja vista ser muito difícil definir previamente quais mensagens serão ou não codificadas.

O algoritmo para codificação pode ser escrito da seguinte forma:

```
codificaMensagem(msg){
  se(existe outra msg plana em buffer && probabilidade==sim ){
    msgBuffer = recuperaMsgBuffer;
    payload = codificaXOR(msg, msgBuffer);
```

```
    payload[0] = id(msg);
    payload[1] = id(msgBuffer);
    enviaPacote(payload);
  } senao {
    payload[0] = id(msg);
    enviaPacote(payload);
  }
}
```

Para armazenar as mensagens e as codificações, foram utilizados em cada nó dois *buffers* circulares de oito posições, um para as mensagens planas (sem codificação) e outro para as codificações recebidas. Em conjunto a esses *buffers*, os IDs dessas mensagens também era armazenado para verificação.

O *payload* da mensagem foi modificado de forma que o primeiro byte armazena o ID da mensagem ou de uma das mensagens codificadas, caso seja um pacote codificado. Caso seja uma mensagem codificada, o byte 2 informa o ID da outra mensagem, ou possui valor zero se o pacote não for uma mensagem codificada.

Quando um pacote é recebido, verifica-se o byte 2 para confirmar se é uma mensagem codificada. Em caso negativo, ela é salva no *buffer* apenas, junto a seu ID. Caso seja uma mensagem codificada, os bytes 1 e 2 são lidos para verificar quais mensagens aquele *payload* codifica. São pesquisadas entre as mensagens armazenadas se um daqueles IDs está armazenado e o outro não, possibilitando assim a decodificação da mensagem utilizando-se o XOR entre o pacote codificado e uma das mensagens codificada naquele pacote. Após obter uma nova mensagem plana a partir da mensagem codificada, é verificado se essa nova mensagem permite decodificar mais alguma codificada do *buffer*. Esse processo é repetido até que não seja possível decodificar mais nenhuma mensagem do *buffer* com determinado pacote e seus desdobramentos.

O algoritmo utilizado para a decodificação das mensagens é o seguinte:

```
recebeMensagem(msg){
  se(msg[1] != 0){
    armazenaEmBuffer(msg, bufferMsgCodificadas);
    salvaID(msg[0], listaIDMsgCodificadas);
    salvaID(msg[1], listaIDMsgCodificadas);
    se (existe 1 msg decodificada e outra nao decodificada \\
    entre os ids msg[0] e msg[1]{
      msgNova = decodificaMsg[msgPlana, msg];
      enquanto (msgNova possibilita decodificar mais pacotes armazenados){
        msgNova = decodificaMsg[msgNova, msgCodificadaSalva];
        salvaID(msgNova[0], listaIDMsgPlanas);
      }
    }
  } senao {
    armazenaEmBuffer(msg, bufferMsgPlanas);
    salvaID(msg[0], listaIDMsgPlanas);
  }
}
```

Esse processo de decodificação é formalmente conhecido como Eliminação Gaussiana (*Gaussian Elimination*) [Grcar 2011], a partir do qual é possível obter N mensagens (variáveis) a partir de N mensagens codificadas (equações) através de um sistema linear.

Para poupar espaço útil no *payload*, o ID da mensagem foi concatenado com o ID do nó para ocuparem apenas um campo, de modo que cada mensagem vinda de determinado nó possua uma identificação única dentre todas as outras mensagens circulantes na rede. Essa utilização de um único ID concatenado no cabeçalho permite usar apenas o espaço de um inteiro, ao invés de alocar espaço para salvar os dois separadamente.

8.3. Simulação

As simulações foram realizadas com uma topologia fixa, onde o parâmetro variado foi apenas o PDR (*Packet Delivery Ratio*) de cada link, para simular as perdas de pacotes ocorridas em redes reais. Esse valor vai de 0 a 1, onde 1 significa 100% de entrega (rede sem perdas) e qualquer valor abaixo disso refere-se ao percentual de mensagens entregues (0.7 seria 70% de pacotes entregues, por exemplo).

A aplicação foi configurada para aguardar o nó destino estar sincronizado para só então começar a enviar os pacotes. Devido a uma limitação do simulador OpenVisualizer, que não realiza o tratamento de mensagens *multicast*, cada nó foi configurado para enviar a mesma mensagem para todos os outros, uma por vez, até um número pré-definido de mensagens, quando então eles param de enviar mensagens e a simulação é finalizada.

Foi implementado um contador em cada nó para acompanhar a quantidade de mensagens válidas recebidas por cada nó. Uma mensagem válida é considerada como uma mensagem não codificada ou uma mensagem codificada (A+B) de onde seja possível extrair uma nova mensagem (A ou B), ainda não recebida por aquele nó. Caso ambas as mensagens da codificação já tenham sido recebidas, essa não é considerada uma mensagem válida e é apenas armazenada para posteriores comparações.

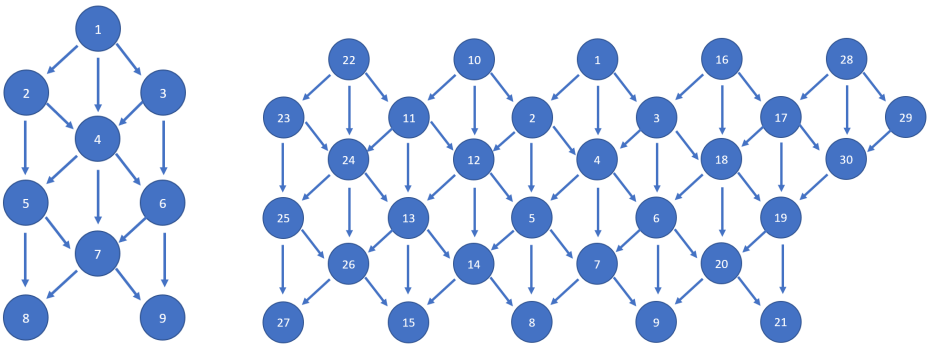
As simulações foram realizadas com e sem Codificação em Rede, com PDR sendo variado de 0.7 a 1.0. Cada iteração da simulação com Codificação em Rede foi comparada com sua correspondente sem a codificação (PDR 0.9 sem Codificação em Rede comparado ao PDR 0.9 com a codificação).

9. Resultados

O resultado esperado ao utilizar a Codificação em Rede é que a vazão da rede aumente, sendo possível entregar um número maior de mensagens nos cenários de múltiplos *unicasts* com perdas na rede.

Para calcular qual o melhor valor para a probabilidade de codificação, foi utilizada uma rede de 9 (nove) nós. O PDR foi fixado em 70% e variou-se a probabilidade de codificação através da fórmula 1/X, onde o X foi variado de 2 a 5, resultando em probabilidades de 20% (1/5) a 50% (1/2), de modo que a única variável fosse a probabilidade de codificação para um mesmo percentual de entrega de mensagens. Para cada probabilidade foi realizada uma bateria de 10 (dez) execuções para obtenção da média de entrega. A topologia da rede utilizada para testar o percentual de codificação em rede pode ser observado na figura 4(a).

O gráfico da figura 5 mostra o desempenho da rede em função da probabilidade de codificação das mensagens. Nota-se que a melhor abordagem para a codificação é de 25% de mensagens codificadas, um valor bem próximo àquele encontrado por dos Santos Ribeiro Júnior [dos Santos Ribeiro Júnior et al. 2017]. Esse valor pode ser explicado pelo



(a) Topologia 9 nós. (b) Topologia 30 nós.

Figura 4. Topologias de rede utilizadas para os testes

fato de que uma probabilidade de codificação muito baixa não traz benefícios perceptíveis por não aproveitar o potencial de recuperação de mensagens a partir dos pacotes codificados, enquanto uma probabilidade muito alta também é prejudicial pois haverá muitas mensagens codificadas na rede e poucas mensagens não-codificadas, tornando difícil a decodificação dos pacotes codificados.

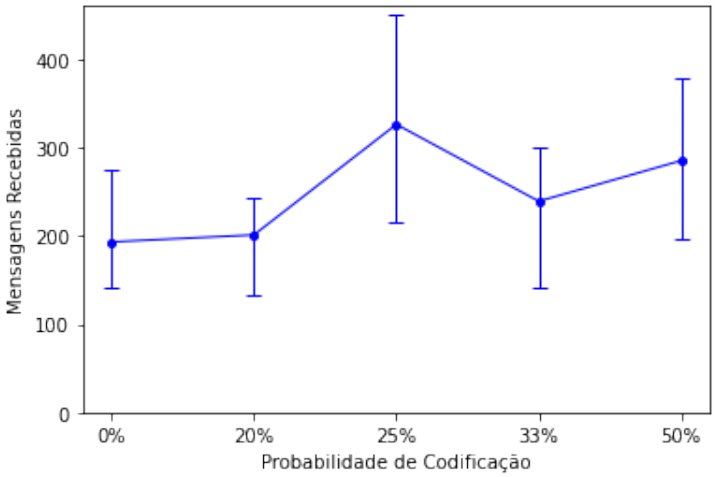


Figura 5. Número de mensagens entregues em função da probabilidade de codificação, em uma topologia de nove nós.

De posse do percentual otimizado de probabilidade, foi criada uma nova topologia com 30 (trinta) nós para verificar a melhora na entrega de pacotes com a utilização da Codificação em Rede comparada ao mesmo PDR sem a codificação. Foram realizadas 10 (dez) baterias de teste para cada PDR, variando de 70 a 100%, com e sem codificação em rede. A topologia utilizada para esses testes é apresentada na figura 4(b).

Importante ressaltar que a quantidade de trinta nós foi utilizada por ser o limite prático da simulação com o OpenVisualizer obtido nos testes. Mesmo com o aumento de memória RAM na máquina virtual utilizada, simulações com mais de trinta nós (trinta e um, para ser mais preciso), o simulador finalizava com erros, impedindo a execução dos testes e consequentemente a obtenção dos valores para comparação.

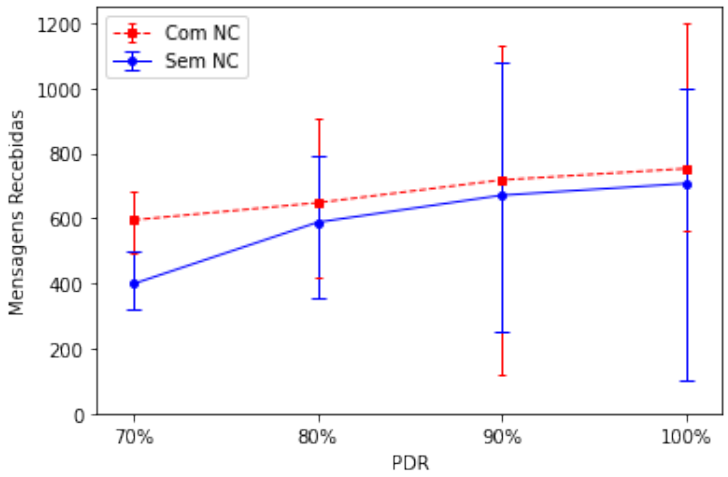


Figura 6. Número de mensagens entregues à medida em que o PDR aumenta, em uma topologia de trinta nós.

O gráfico apresentado na figura 6 mostra o comparativo entre o desempenho da rede com e sem a Codificação em Rede. Nos gráficos exibidos, o ponto representa a média dos valores obtidos nas dez simulações e os limites da barra de erro representam o desvio padrão desse conjunto de dados.

Quando existem perdas de pacotes, muitas vezes a rede mudou sua topologia durante a simulação, pois se algum nó perde o recebimento de um pacote periódico de controle utilizado para manter a sincronia da rede (chamado de pacote DIO - *DODAG Information Object* ou Objeto de Informação do Grafo) ele é dessincronizado e precisa entrar na rede novamente. Esse processo ocorre com mais frequência quanto maior a perda de pacotes na rede. Essas mutações da topologia podem explicar o amplo desvio padrão apresentado pelas simulações.

É importante ressaltar que mesmo com o PDR de 100% haverá perda de pacotes, pois os nós foram configurados para enviarem mensagens *unicast* em sequência para todos os nós. Portanto, até a rede ser toda estabilizada, com todos os nós sincronizados e participando da tabela de alocação, os nós já sincronizados enviarão mensagens para os nós que ainda não estão na rede, e essas mensagens serão perdidas. Como os nós não possuem conhecimento da topologia, na solução proposta não é possível detectar quando o nó destino já está na rede sem enviar um pacote a ele.

Nota-se a partir do gráfico da figura 6 que os maiores benefícios da Codificação em Rede ocorrem quando os valores de PDR são menores. Nesses percentuais a codificação conseguiu trazer maiores benefício de robustez e aumento de entrega de mensagens em comparação ao TSCH sem a codificação.

10. Conclusão

Neste artigo foi apresentada a Codificação em Rede com o protocolo TSCH. O foco principal dessa abordagem é utilizar a Codificação em Rede em redes TSCH para aumentar a vazão dessa rede e aumentar a quantidade de mensagens entregues, tornando-a ainda mais robusta e confiável. Importante ressaltar que a Codificação em Rede também consegue entregar mais mensagens com a mesma quantidade de transmissões, o que resulta em

economia de energia, uma vantagem importante quando se trata se nós com capacidade energética limitada.

Os resultados obtidos nas simulações demonstram que a Codificação em Rede aumenta a capacidade de entrega de mensagens, tornando a rede mais robusta e confiável em cenários onde há perda de mensagens, que é a condição mais comum em aplicações reais.

O custo adicional necessário para implementar a codificação em rede são os identificadores das mensagens codificadas que precisam ser enviados juntamente com o *payload*, para que o nó possa identificar quais mensagens estão codificadas e seja capaz de decodificá-las. Além disso há um custo de memória nos nós para armazenar algumas mensagens decodificadas e algumas codificadas que ainda não puderam ser decodificadas. Na implementação foi definido que seriam oito mensagens armazenadas e oito mensagens codificadas, em dois *buffers* cíclicos. Para aplicações mais restritas em memória, o tamanho desses *buffers* pode ser modificado.

Como possíveis trabalhos futuros, pode-se apontar um estudo mais aprofundado sobre o comportamento da Codificação em Rede, realizando testes com sensores reais e analisando os resultados do aumento do número de mensagens bem como da economia energética com mais precisão. Outra abordagem possível é a utilização de *test-beds* para realizar os testes com uma quantidade maior de sensores e analisar o comportamento do algoritmo proposto nessas condições.

Referências

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

Boccadoro, P., Barile, M., Piro, G., and Grieco, L. A. (2016). Energy consumption analysis of tsch-enabled platforms for the industrial-iot. In *2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*, pages 1–5.

D. Hauweele, R.-A. Koutsiamanis, B. Q. and Papadopoulos, G. Z. (2020). Pushing 6tisch minimal scheduling function (msf) to the limits. pages 327–342. IEEE.

dos Santos Ribeiro Júnior, N., Tavares, R. C., Vieira, M. A. M., Vieira, L. F. M., and Gnawali, O. (2017). Codedrip: Improving data dissemination for wireless sensor networks with network coding. *Ad Hoc Networks*, 54:42–52.

Duquennoy, S., Elsts, A., Al Nahas, B., and Oikonomo, G. (2017a). Tsch and 6tisch for contiki: Challenges, design and evaluation. In *2017 13th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 11–18. IEEE.

Duquennoy, S., Eriksson, J., and Voigt, T. (2017b). Five-nines reliable downward routing in rpl.

Elsts, A. (2020). Tsch-sim: Scaling up simulations of tsch and 6tisch networks. *Sensors*, 20(19).

Fragouli, C., Widmer, J., and Le Boudec, J.-Y. (2006). On the benefits of network coding for wireless applications. In *2006 4th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, pages 1–6. IEEE.

Grcar, J. F. (2011). Mathematicians of gaussian elimination. *Notices of the AMS*, 58(6):782–792.

Gubbi, J., Buyya, R., Marusic, S., and Palaniswami, M. (2013). Internet of things (iot): A vision, architectural elements, and future directions. *Future generation computer systems*, 29(7):1645–1660.

Hermeto, R. T., Gallais, A., and Theoleyre, F. (2017). Scheduling for ieee802. 15.4-tsch and slow channel hopping mac in low power industrial wireless networks: A survey. *Computer Communications*, 114:84–105.

Ho, T., Medard, M., Koetter, R., Karger, D. R., Effros, M., Shi, J., and Leong, B. (2006). A random linear network coding approach to multicast. *IEEE Transactions on Information Theory*, 52(10):4413–4430.

Li, S.-Y., Yeung, R. W., and Cai, N. (2003). Linear network coding. *IEEE transactions on information theory*, 49(2):371–381.

Minet, P., Khoufi, I., and Laouiti, A. (2017). Increasing reliability of a tsch network for the industry 4.0. In *2017 IEEE 16th International Symposium on Network Computing and Applications (NCA)*, pages 1–10.

Palattella, M. R., Accettura, N., Dohler, M., Grieco, L. A., and Boggia, G. (2012). Traffic aware scheduling algorithm for reliable low-power multi-hop ieee 802.15.4e networks. In *2012 IEEE 23rd International Symposium on Personal, Indoor and Mobile Radio Communications - (PIMRC)*, pages 327–332.

Palattella, M. R., Accettura, N., Grieco, L. A., Boggia, G., Dohler, M., and Engel, T. (2013). On optimal scheduling in duty-cycled industrial iot applications using ieee802. 15.4 e tsch. *IEEE Sensors Journal*, 13(10):3655–3666.

Santos, B. P., Silva, L., Celes, C., Borges, J. B., Neto, B. S. P., Vieira, M. A. M., Vieira, L. F. M., Goussevskaia, O. N., and Loureiro, A. (2016). Internet das coisas: da teoria a prática. *Minicursos SBRC-Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos*.

Thubert, P. (2019). An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4. Internet-Draft draft-ietf-6tisch-architecture-28, Internet Engineering Task Force. Work in Progress.

Vieira, L. F. M. and Vieira, M. A. M. (2017). Network Coding for 5G Network and D2D Communication. In *Proceedings of the 13th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, Q2SWinet '17, page 113–120, New York, NY, USA. Association for Computing Machinery.

Vilajosana, X., Tuset-Peiro, P., Watteyne, T., and Pister, K. (2015). Openmote: Open-source prototyping platform for the industrial iot.

Watteyne, T., Vilajosana, X., Kerkez, B., Chraim, F., Weekly, K., Wang, Q., Glaser, S., and Pister, K. (2012). Openwsn: a standards-based low-power wireless development environment. *Transactions on Emerging Telecommunications Technologies*, 23(5):480–493.

Zhang, W., He, B., Zhao, X., and Chen, Q. (2008). An open wireless sensor network platform—openwsn. *Journal of Computer Research and Development*, 45(1):97.