Split-MAC: um protocolo assíncrono de baixa energia e latência reduzida para RSSF

Tales Heimfarth[1], **João Carlos Giacomin**[1]

[1]Dep. Ciência da Computação – Universidade Federal de Lavras (UFLA)
Campus da UFLA – Caixa Postal 3037 – 37200-900 – Lavras –Brazil

{tales,giacomin}@ufla.br

Abstract. *Wireless Sensor Networks (WSN) employ duty-cycles as the main strategy to save energy. The sleep-delay arises as a drawback from this operating mode, increasing end-to-end latency. This paper presents Split-MAC: a low-energy, low-latency asynchronous protocol for WSNs. It combines medium access control (MAC) and routing functions. Some strategies are employed by Split-MAC to achieve low latency: an anycast communication pattern, a medium reservation scheme to transmit data in a pipelined fashion and simultaneous transmission of data segments. Simulations were performed to compare our protocol with others from literature. Split-MAC outperforms all other tested protocols. There is at least* 30% *decrease in latency.*

Resumo. *Redes de Sensores Sem Fio (RSSF) usam ciclos de trabalho como estratégia para economizar energia. O atraso de propagação (*sleep-delay*) é uma consequência negativa desse modo de operação, aumentando a latência fim-a-fim. Apresenta-se aqui o Split-MAC: um protocolo assíncrono de baixa energia e latência reduzida para RSSFs. Ele reúne funções de controle de acesso ao meio e de roteamento, e emprega estratégias para obter baixa latência: comunicação* anycast*, um esquema de reserva de canal para transmitir dados de maneira encadeada e transmissão simultânea de segmentos de dados. Simulações foram realizadas para comparar este protocolo com outros da literatura. O Split-MAC demonstrou superioridade frente aos outros protocolos testados, reduzindo a latência em pelo menos* 30% .

1. Introdução

Uma rede de sensores sem fios (RSSF) é formada por um conjunto de pequenos dispositivos autônomos que se comunicam por um meio não guiado e trabalham colaborativamente para atingir um objetivo comum. Os nós sensores são dotados de um pequeno processador, memória e uma unidade de energia limitada (normalmente uma bateria). Alguns sensores e atuadores podem fazer parte da composição de um nó sensor [Culler et al. 2004]. A principal aplicação das redes de sensores é no monitoramento de ambientes, em que variáveis ambientais como temperatura e umidade são medidas e transmitidas até um centro de coleta de dados. A comunicação é feita por múltiplos saltos, visto que os nós sensores empregam normalmente rádios de baixa potência e pequeno alcance.

Devido à baixa disponibilidade de energia para manter os nós sensores em funcionamento, a economia desse recurso é sempre levada em conta no desenvolvimento dos protocolos de controle de comunicação. O controle de acesso ao meio (MAC) tem

um grande impacto no consumo de energia de uma RSSF. O protocolo MAC é responsável por garantir o acesso exclusivo da mídia compartilhada em um determinado momento [Kumar et al. 2018]. Protocolos MAC desenvolvidos para redes de sensores sem fio que que utilizam ciclos de trabalho controlam os períodos de ativação do rádio como forma de economizar energia [Buettner et al. 2006]. Os nós sensores são mantidos em estado de baixo consumo de energia, com rádios desligados, a maior parte do tempo. Periodicamente são reativados para fazer medições, processamento e comunicação.

Os protocolos MAC podem ser divididos em três grupos [Cano et al. 2011]: protocolos agendados, protocolos com período ativo comum e protocolos assíncronos. Nos protocolos assíncronos, os nós sensores utilizam o mesmo ciclo de trabalho, porém seus períodos ativos são independentes entre si. Portanto, os nós são ativados de forma assíncrona para verificar se há comunicação a ser recebida. A maioria dos protocolos assíncronos utilizam amostragem de preâmbulo. Cada pacote de dados é precedido por um longo preâmbulo para estabelecer a comunicação entre o remetente (transmissor de dados) e o destinatário (receptor) [Bachir et al. 2010]. Protocolos MAC assíncronos são menos complexos e têm um consumo de energia reduzido para redes de baixo tráfego.

Apesar das vantagens dos protocolos de amostragem de preâmbulo, um nó sensor que possui uma mensagem para enviar, deve aguardar que o receptor esteja acordado. Isso resulta em um atraso no envio de dados, denominado *sleep-delay*. Uma maneira de reduzir esse problema é explorar a redundância de caminhos disponíveis na rede, empregando um padrão *anycast* de comunicação [Ashraf et al. 2011]. Nesse método, um conjunto de candidatos para o próximo salto é formado, denominado Forward Candidate Set (FCS). Os elementos deste conjunto são nós que reduzem a distancia ao destino final. Esta informação é fornecida por um algoritmo de roteamento geográfico da camada de rede (NET). O protocolo *anycast* seleciona como próximo encaminhador o nó do FCS que desperta mais cedo. Este procedimento reduz a latência causada pela falta de sincronismo entre os nós.

Embora o uso de ciclos de trabalho seja a melhor maneira de economizar energia e aumentar a vida útil da rede [Du et al. 2007, Hong and ki Kim 2009], escalonar os tempos de ativar os nós no rota em direção ao destino é a melhor opção para reduzir o problema do *sleep-delay* [Lu et al. 2004, Sun et al. 2008, Doudou et al. 2016, Tong et al. 2016]. Neste método, os nós sensores que compõem a rota até o destino programam seus períodos ativos para ocorrerem em sequência, de forma que cada nó desperte imediatamente antes do horário de receber uma mensagem do nó anterior. Assim, a transmissão de dados é feita de forma escalonada (*pipelined*) [Pyeon et al. 2016]. O método é usado pelo PAX-MAC [Heimfarth et al. 2016], nossa proposta de protocolo anterior ao Split-MAC.

O objetivo do Split-MAC é ir além do PAX-MAC para reduzir ainda mais a sua latência, realizando a divisão do pacote de dados em unidades menores, chamadas segmentos, que estão programadas para serem transportadas no canal de comunicação reservado pelo preâmbulo. Em vez da transmissão simultânea da série de preâmbulos e de um pacote de dados logo atrás, como no PAX-MAC, os preâmbulos e uma sequência de n segmentos são transmitidos simultaneamente por nós que fazem parte da rota, localizados em posições distantes entre si de forma que não haja interferências nas comunicações. Isso resulta em uma considerável redução de latência para cenários com longos caminhos de transmissão. Além disso, a mesma técnica pode ser usada para rajadas de comunicação,

por exemplo, para a transmissão de fluxos multimídia. Um mecanismo para controlar essas transmissões simultâneas, evitando colisões entre segmentos e preâmbulos, é introduzido neste trabalho. No trabalho, uma comparação entre o protocolo proposto com outros MACs *anycast* foi realizada. Para os cenários avaliados, obteve-se uma menor latência de comunicação que outros protocolos encontrados na literatura.

2. Trabalhos Relacionados

O protocolo GeRaF [Zorzi and Rao 2003] foi um protocolo assíncrono pioneiro no uso da comunicação anycast, com o objetivo de reduzir a latência em protocolos assíncronos com preâmbulos. Diferente das abordagens que usam o padrão de comunicação *unicast*, o GeRaF considera um conjunto de nós candidatos a encaminhar um pacote, denominado *Forwarding Candidate Set (FCS)*. Para participar do FCS, um nó dentro do alcance de rádio do nó transmissor deve ser capaz de diminuir a distância até o nó destino.

O protocolo CMAC [Liu et al. 2007] é uma melhoria do GeRaF. Ele define um avanço mínimo (r_0) em direção ao *sink* como critério para selecionar os membros do FCS. Os vizinhos do remetente devem fornecer um avanço para o destino maior que r_0 para participar do FCS. r_0 depende da carga de tráfego e da densidade da rede.

O AGA-MAC [Heimfarth et al. 2015] segue o mesmo princípio do CMAC, impondo um avanço mínimo ao destino em cada salto. No AGA-MAC, esse valor limite depende do tamanho da mensagem de dados. Este foi um trabalho pioneiro, considerando o comprimento dos dados para tomar decisões sobre o roteamento em protocolos anycast.

O Any-MAC [Ashraf et al. 2011] é um protocolo do tipo anycast que associa um protocolo de roteamento a um MAC assíncrono com base em amostragem de preâmbulos. O FCS tem tamanho fixo, sendo definido na camada de rede e informado à MAC. O primeiro membro do FCS que responde ao sinal do remetente se torna o próximo salto.

Outra abordagem para obter redução de latência é a reserva de canal, feita em duas fases. Na primeira fase, um pacote de controle é enviado com antecedência para escalonar o horário de comunicação dos nós que participarão da transmissão do pacote de dados. Na segunda fase, pacote de dados é transmitido do nó de origem para o destino de modo escalonado (*pipelined*) [Pyeon et al. 2016], com o menor tempo de entrega. R-MAC [Du et al. 2007] é um protocolo em que os nós têm um período de atividade comum. Durante o período ativo, pacotes PION são enviados para escalonar os nós do caminho escolhido pelo roteamento para receber mensagens de dados. Assim, os nós do caminho serão ativados no horário agendado para encaminhar dados.

O protocolo *cross-layer* PAX-MAC [Heimfarth et al. 2016] associa funções de roteamento geográfico e de controle de acesso ao meio baseado em amostragem de preâmbulo. Utiliza o padrão de comunicação *anycast* do GeRaF [Zorzi and Rao 2003] e a transmissão dividida em fases do R-MAC [Du et al. 2007]. Na primeira fase, preâmbulos são enviados à frente, a fim de descobrir a rota até o nó destino, ao mesmo tempo que sincroniza os horários dos nós da rota. Na segunda fase, o pacote de dados é enviado de forma escalonada, usando o canal de comunicação reservado. O pacote de dados é transmitido alguns saltos atrás do preâmbulo, simultaneamente. Esse paralelismo na transmissão de preâmbulos e dados diminui a latência da comunicação entre o nó fonte e o destino.

O protocolo proposto no presente artigo permite uma menor latência de

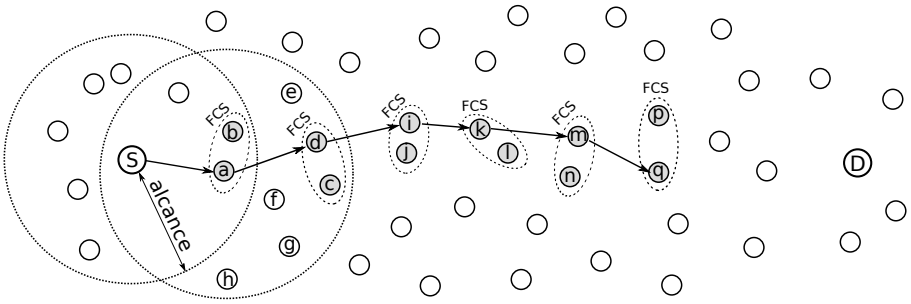

Figura 1. Roteamento com Split-MAC, apresentando os 6 primeiros saltos

comunicação pelo aumento do paralelismo das comunicações na RSSF. A mensagem de dados é dividida em vários segmentos, que são transmitidos de forma escalonada. O preâmbulo e os vários segmentos trafegam em áreas distintas da rede ao mesmo tempo. Essa abordagem é inovadora e não foi encontrada no estado da arte.

3. O protocolo Split-MAC

Split-MAC é um protocolo *cross-layer* assíncrono que integra funções da subcamada MAC e da camada de rede. Este protocolo utiliza comunicação *anycast* e roteamento geográfico com o objetivo de reduzir a latência fim-a-fim.

Os nós sensores usam ciclo de trabalho para economizar energia, permanecendo no estado inativo, com os rádios desligados, na maior parte do tempo. Periodicamente, os nós são reativados para realizar medições, processamento e comunicação. Todos os nós da rede usam o mesmo período de ciclo, mas não há sincronização nos horários de despertar. Quando um nó precisa transmitir uma mensagem, envia uma série de preâmbulos curtos a fim de estabelecer contato com um receptor. Após cada preâmbulo, o nó emissor coloca o rádio no modo de escuta, aguardando um sinal de confirmação (eACK) do receptor. Quando o eACK é recebido, o nó emissor interrompe o envio de preâmbulos.

3.1. Comunicação *anycast*

O Split-MAC emprega o padrão de comunicação *anycast* a fim de reduzir o tempo para estabelecer contato entre transmissor e receptor. Um grupo de nós vizinhos, denominado FCS (*Forwarding Candidate Set*), é composto por possíveis encaminhadores de mensagens. O primeiro nó do FCS a entrar no período ativo envia de volta um eACK ao transmissor, interrompendo a sequência de preâmbulos. Os membros do FCS são escolhidos entre os vizinhos de acordo com métricas pré-estabelecidas, como por exemplo a maior proximidade ao destino. Se for esta a métrica usada, os nós sensores terão conhecimento de suas posições físicas desde que possuam identificador de posição como GPS ou executem algum algoritmo de descoberta de localização no início da operação da rede.

O envio da mensagem de dados do nó de origem S para o destino D é feito por múltiplos saltos, através de uma rota composta por vários nós intermediários que fazem o encaminhamento da mensagem. A Figura 1 exemplifica o roteamento nos seis primeiros saltos. Neste exemplo, o FCS é composto por dois membros. Os nós c e d são membros do FCS do nó a. Os nós e, f e g não foram escolhidos porque estão mais distantes do destino do que os outros dois. A distância do nó h para D é maior do que a do nó a. O nó d se tornou o próximo encaminhador, depois do nó a, porque acordou antes do c e enviou

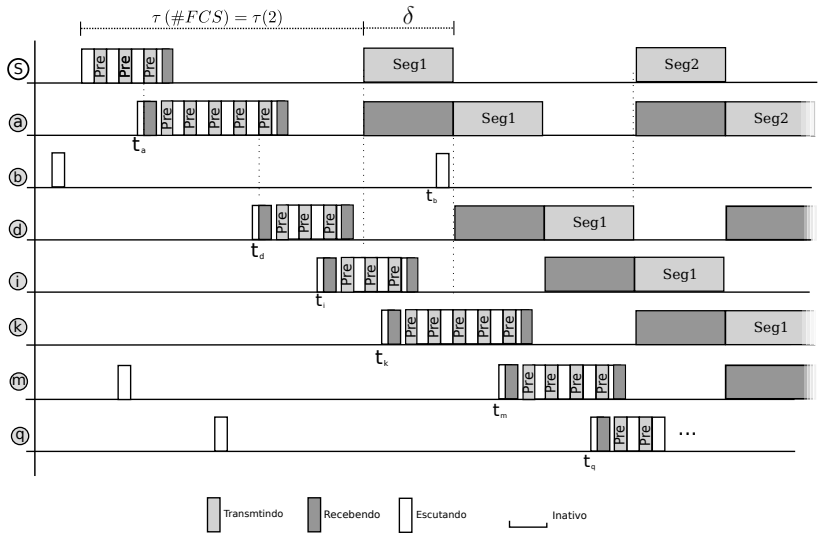

Figura 2. Diagrama de tempo com o Split-MAC, apresentando os 6 primeiros saltos e transmissão simultânea de preâmbulos e dois segmentos de dados

de volta um `eACK`. Depois disso, o nó a entra no estado inativo até o momento de receber dados de S.

3.2. Transmissão em duas fases

Para acelerar a propagação de mensagens, elas são transmitidas de modo escalonado (*pipelined*). Diferentemente de outros protocolos anycast, o Split-MAC não envia dados logo após receber o sinal `eACK`. O processo de comunicação é dividido em duas fases: uma assíncrona e outra síncrona. Na fase assíncrona, preâmbulos são enviados à frente para estabelecer a rota entre S e D e para ajustar os horários de transmissão do pacote de dados. Preâmbulos carregam a informação do horário em que cada nó da rota deverá despertar para receber o pacote de dados. Na fase síncrona, os dados são transmitidos pelos nós da rota com horários escalonados. Cada nó da rota em direção ao D acorda após o anterior, com uma diferença δ entre os horários. Este intervalo é suficiente para a transmissão de um pacote de dados. Ressalta-se que não há sincronização entre todos os nós da rede, é feita apenas uma sincronização momentânea na passagem do preâmbulo. Ambas as fases são executadas ao mesmo tempo na rede, mas em locais diferentes. Enquanto a fase assíncrona é realizada por um nó de roteamento, a fase síncrona ocorre em outro ponto da rede, alguns saltos atrás. Para evitar colisões entre preâmbulo e pacote de dados, o nó de origem S deve separar as transmissões desses pacotes por um período de tempo adequado. Isso é realizado por um intervalo inicial, τ, colocado entre o envio de preâmbulos e o envio de dados. Uma vez que o intervalo entre a transmissão de preâmbulos e a de dados é pequeno, considera-se desprezível o desvio de relógio devido ao *drift*, na fase síncrona.

O Split-MAC não define um pacote `eACK` especial como CMAC [Liu et al. 2007] e X-MAC [Buettner et al. 2006]. Quando um membro do FCS acorda e recebe um preâmbulo, ele começa a enviar sua própria série para estabelecer contato com um dos membros do seu FCS. O primeiro desses preâmbulos é entendido pelo remetente anterior como um sinal `eACK`, fazendo-o interromper suas transmissões.

No exemplo da Figura 1, o FCS do nó origem S é formado pelos nós a e b. A linha do tempo apresentada na Figura 2 mostra que, depois que S começou a enviar sua série de preâmbulos, o nó a acordou em t_a. Isso ocorre antes do nó b, que despertará em t_b. No protocolo anycast Split-MAC, o primeiro nó do FCS que recebe o preâmbulo assume o papel do próximo roteador em direção ao destino D. Imediatamente, ele inicia o envio de uma nova série de preâmbulos para sinalizar sua intenção de transmitir para um membro de seu próprio FCS. Neste exemplo, o nó a está tentando estabelecer contato com c ou d. O primeiro preâmbulo de a atua também como `eACK` para o nó S. Ao mesmo tempo, o nó a está programado para despertar imediatamente antes que o nó de origem S comece a enviar dados no futuro. Este processo continua como apresentado na Figura 2. O nó d se torna o próximo roteador, após o nó a, desde que acorda antes do nó c. Na sequência, o nó d começa a enviar preâmbulos e programa seu tempo para receber pacotes de dados. Como explicado, o primeiro preâmbulo é usado como `eACK` pelo nó a. Após três preâmbulos, o nó i é ativado e assume o papel do próximo roteador, iniciando sua sequência de preâmbulos. Enquanto o nó i está transmitindo seus preâmbulos, o nó S inicia a transmissão do pacote de dados para o nó a, que acabou de acordar para receber dados. A partir desse instante, ocorre uma transmissão simultânea de dados e preâmbulos. É importante observar que essas comunicações não interferem entre si, tendo em vista a distância mantida entre os dois nós transmissores.

3.3. Transmissão segmentada

Para reduzir ainda mais o tempo de entrega dos dados ao destino, mensagens longas são divididas em partes menores, denominadas `segmentos`, enviadas separadamente na fase síncrona do protocolo. Vários `segmentos` da mensagem de dados podem trafegar pela rota entre S e D simultaneamente, em locais distintos da rede. Por isso o protocolo é denominado **segmentado** (*split*). Para que isto seja possível, é preciso manter uma distância maior que dois saltos entre os nós que estiverem transmitindo segmentos consecutivos. Para que essa distância seja mantida, o nó de origem S mantém um intervalo entre os envios de segmentos consecutivos, superior a 2δ.

Como exemplo, podemos ver o diagrama de tempo da figura 2. Nesta figura é apresentada a transmissão de dois segmentos de uma mensagem de dados pelo nó de origem S. O nó k desperta no instante t_k e recebe um preâmbulo do nó i, iniciando imediatamente sua própria transmissão de preâmbulos. Parte dessas transmissões ocorrem simultaneamente com o envio do segmento Seg_1 de a para d. Em seguida, o nó d envia o Seg_1 para o nó i, ao mesmo tempo que m busca o contato com um membro de seu FCS (p ou q). Após isto, o nó S entende que a próxima transmissão do Seg_1 não interfere com sua próxima comunicação, visto que já se passou um intervalo de tempo superior a 2δ. Então S inicia a transmissão do segundo segmento dos dados para o nó a. Neste instante, três transmissões ocorrem simultaneamente, para estabelecer a comunicação entre S e D: o nó q transmite preâmbulos, o nó i transmite o Seg_1 para k e S transmite o Seg_2 para a.

São consideradas mensagens pequenas aquelas que tem tempo de transmissão menor que 33% do tempo de ciclo, enquanto que as grandes ocupam o canal por mais tempo. Essa definição está ligada à cardinalidade do FCS. Uma rede com pacotes pequenos pode também ter seu desempenho melhorado pelo Split-MAC, fazendo uma transmissão onde cada pacote pequeno funcione como um segmento de um pacote maior.

3.4. Colisão iminente

Devido à natureza assíncrona do Split-MAC, o número de preâmbulos gastos em cada salto para fazer contato com o próximo roteador pode variar, conforme apresentado na Figura 2. O nó de origem deve calcular cuidadosamente o intervalo de atraso (τ) para evitar que sua transmissão ocorra em uma região longe de preâmbulos, com uma distância maior que duas vezes o alcance do rádio. O processo de enviar preâmbulos à frente e pacotes de dados alguns saltos atrás continua até atingir o nó de destino D. Uma colisão ainda poderia acontecer se, por exemplo, a série de preâmbulos de d demorasse mais tempo para estabelecer contato com um membro do seu FCS (i ou j). Uma colisão de mensagens ocorreria no nó a envolvendo o segmento de dados Seg_1 vindo de S e o preâmbulo de d. Para esses casos, o Split-MAC possui um método de recuperação denominado *identificação de colisão iminente*. Estando o nó d ciente do horário em que a receberá o primeiro `segmento` de dados vindo de S, d interrompe sua transmissão de preâmbulos e aguarda a chegada de todos os segmentos. Após isto, o nó d reinicia o processo de envio da mensagem tal como o fez o nó S. Esta é uma situação que tem alto custo em latência.

É importante observar que não estamos considerando colisões de outras transmissões de dados, analisamos o comportamento de apenas uma única mensagem sendo transmitida por vez na rede. No caso de fluxos concorrentes em uma rede, outros mecanismos de recuperação devem ser usados, como a adoção de intervalos aleatórios (*back off*) para nova tentativa de transmissão. Consideramos que todo nó remetente encontra pelo menos um vizinho para enviar o pacote na direção de destino. Se essa condição falhar, outros algoritmos devem ser usados, como a *regra da mão direita* [Karp and Kung 2000].

Para evitar colisões iminentes, a velocidade de propagação dos preâmbulos deve ser, em média, igual ou maior que do `segmento`. Pode-se regular o tempo médio necessário para encontrar um nó acordado no FCS ajustando-se sua cardinalidade. Assim, o número de membros do FCS é determinado conforme o tamanho do `segmento` dos dados, quanto menor o tamanho, maior a cardinalidade do FCS. A distância média entre o nó que envia o segmento e o que transmite a série de preâmbulos é constante.

3.5. Seleção do FCS

As decisões de roteamento no Split-MAC são tomadas localmente. Cada nó sensor conhece sua própria posição, as posições de seus vizinhos e a posição para onde o pacote de dados é endereçado, denominada posição destino (D). Com base nessas informações, um nó com uma mensagem de dados na fila seleciona quais de seus vizinhos farão parte do FCS. Os IDs dos membros do FCS são inseridos nos preâmbulos, para que cada um possa estar ciente de sua condição. Quando um membro do FCS acorda, recebe um preâmbulo e começa a enviar seus próprios preâmbulos, dizemos que o preâmbulo fez um progresso. O tempo médio necessário para obter um progresso de preâmbulo depende da cardinalidade do FCS. Esse tempo está associado ao número de preâmbulos que devem ser enviados até que um membro do FCS envie uma resposta. Este número é calculado como [Heimfarth et al. 2016]: $r(v) = \sum_{i=1}^{N_p} \left(\frac{i}{N_p}\right)^v$

N_p é o número máximo de preâmbulos a serem enviados em um tempo de ciclo. Essa equação pode ser usada para obter a cardinalidade (v) do FCS que ajusta o número médio de preâmbulos ($r(v)$). O Split-MAC controla a cardinalidade do FCS para que

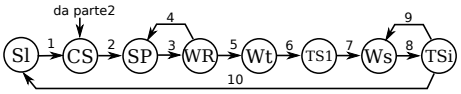

Figura 3. Máquina de estados parte 1, nó remetente.

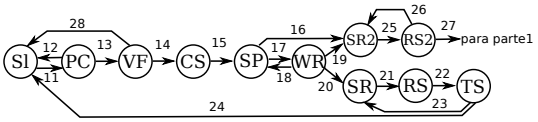

Figura 4. Parte 2, nós intermediários.

um progresso de preâmbulo leve, em média, um tempo igual ou menor que o gasto para transmitir um `segmento` da mensagem de dados.

3.6. Máquina de Estados

Esta seção descreve a máquina de estados do Split-MAC. As Figuras 3 e 4 apresentam o funcionamento do protocolo, do ponto de vista do nó de origem S e de um nó intermediário da rota, respectivamente. A Tabela 1 apresenta o significado dos estados.

Ao receber uma nova mensagem das camadas superiores, o Split-MAC inicia o processo de transmissão através da transição 1 que leva ao estado `CS` (detecção de portadora), Figura 3. Se o canal de comunicação estiver livre, a transição 2 mudará a máquina de estado para `SP` (envio de preâmbulo). Nesse estado, um pacote de preâmbulo é transmitido. Pela transição 3, o estado é alterado para `WR` (aguardando uma resposta), no qual o protocolo MAC aguarda uma resposta de um nó vizinho pertencente ao FCS.

Quando um membro do FCS desperta e recebe o preâmbulo, ele começa a enviar seus próprios preâmbulos adiante. O primeiro preâmbulo funciona como um pacote `eACK`, uma resposta ao nó anterior. Se nenhum membro do FCS estiver no período ativo para receber o preâmbulo e enviar a resposta, a transição 4 retornará o protocolo ao estado `SP`. Se um pacote de resposta (`eACK`) for detectado, o protocolo mudará para o estado `Wt`. Nesse estado, aguarda-se um tempo calculado para enviar o primeiro segmento do pacote de dados, de acordo com a cardinalidade do FCS. Esse tempo é necessário para evitar colisões entre os pacotes de controle (preâmbulos) e o primeiro segmento do pacote de dados. Depois disso, a transição 6 leva ao estado `TS1`. Após o envio do segmento 1, a transição 7 leva o protocolo para o estado `Ws`. Nesse estado, aguarda-se até o segmento anterior ter atingido uma distância segura, evitando interferência entre segmentos consecutivos. Tendo transcorrido esse tempo, a transição 8 leva o protocolo para o estado `TSi`. Nesse estado o próximo segmento do pacote é enviado. Esses dois últimos estados se repetem até que todos os segmentos tenham sido enviados pela rede (transição 9). No final da transmissão, o protocolo retorna ao estado `Sl` pela transição 10.

A segunda parte da máquina de estados, apresentada na Figura 4, detalha o funcionamento do protocolo em nós intermediários no caminho do destino final (saltos intermediários). Na maior parte do tempo, quando não há nenhuma mensagem circulando na rede, os nós repetem o ciclo de suspensão/atividade, conhecido como ciclo de trabalho. Cada nó alterna entre estados `Sl` e `PC` através das transições 11 e 12. O ciclo de trabalho determina o tempo de prospecção em relação ao tempo de ciclo total.

Quando um preâmbulo é detectado no estado `PC`, o Split-MAC vai ao estado `VF` pela transição 13. Nesse estado, o nó sensor verifica se faz parte do FCS. Se for esse o caso, muda o estado para `CS` pela transição 14, caso contrário, o nó retorna para texttt`Sl`, pela transição 28. Após verificar se o canal está livre, o protocolo vai para o estado `SP`

pela transição 15. Se for detectada uma colisão iminente, o envio de preâmbulos deve ser encerrado, todos os segmentos recolhidos e a transmissão reiniciada em momento posterior. A transição 16 é ativada nesse caso. Caso contrário, a transição 17 leva ao estado WR. Nesse estado o protocolo espera pelo eACK (primeiro preâmbulo do próximo salto). Se nenhum nó do FCS responde, o protocolo volta para o estado SP para o envio do próximo preâmbulo (transição 18). A transição 19 tem uma ativação similar à 16: é acionada no caso de colisão iminente. No caso de um eACK recebido, o funcionamento padrão do protocolo deve ser continuado. Para isso, o estado SR é alcançado pela transição 20. Quando o tempo correto para recebimento de um segmento termina, a transição 21 leva o protocolo para o estado RS. Após a recepção do segmento, o mesmo é retransmitido para o próximo encaminhador no estado TS. Se mais segmentos devem ser retransmitidos, a transição 23 é ativada, caso contrário, toda a mensagem foi retransmitida e o protocolo volta para o estado de inatividade pela transição 24.

Tabela 1. Lista de estados da máquina de estados do Split-MAC.

Estado	*Descrição*
Sl - Sleeping	Nó está inativo
CS - Carrier Sensing	Sondagem de canal por comunicação
SP - Sending preamble	Preambulo está sendo enviado
WR - Waiting reply packet	Canal é sondado por um pacote eACK
Wt - Waiting τ	Nó aguarda um período antes de enviar o pacote de dados
TS1 - Transmitting Segment	Primeiro segmento de dados é enviado
TSi - Transmitting Segment i	Segmento i de dados é enviado
Ws - Waiting s	Nó aguarda o período entre segmentos
PC - Probing the channel	Sondagem de canal para detecção de preâmbulo
VF - Verifying FCS	Verifica se nó é membro do FCS
SR - Sleeping for receive segment	Esperando para recebimento de segmento
RS - Receiving segment	Um segmento de dados está sendo recebido

Conforme descrito acima, uma colisão iminente pode acionar as transições 16 ou 19 que levam ao estado SR2. Isso porque o envio de preâmbulos foi interrompido e todos os segmentos devem ser recebidos pelo nó atual, o que é feito nos estados SR2 e RS2. Após isso, o processo de transmissão deve ser iniciado de forma idêntica ao nó iniciador, utilizando a máquina de estados da Figura 3. Isso é descrito na figura como *para parte1*.

4. Resultados Experimentais

A presente seção apresenta os resultados e discussões da avaliação do protocolo proposto neste trabalho. As métricas avaliadas foram latência e consumo de energia. Os nós usam o mesmo ciclo de trabalho com horário de atividade independente. O nosso trabalho foi comparado com diferentes protocolos assíncronos: PAX-MAC [Heimfarth et al. 2016], X-MAC [Buettner et al. 2006], X-MAC Anycast 6 [Ashraf et al. 2011], GeRaF [Zorzi and Rao 2003] e AGA-MAC [Heimfarth et al. 2015]. As simulações foram realizadas com o simulador de redes de sensores sem fio GrubiX. O simulador Grubix é derivado do Shox [Lessmann et al. 2008], contempla a simulação das diferentes camadas de comunicação e contém o protocolo 802.15.4, utilizado em RSSFs.

4.1. Configurações

Para as simulações, um modelo de rádio baseado no padrão IEEE 802.15.4 [IEEE Standard 2003], potência de transmissão fixa, links bidirecionais e modelo de

Tabela 2. Parâmetros utilizados na simulação

Símbolo	Parametro	Valor
R	Raio de alcace (m)	40
P	Potência do transmissor (mW)	60 - baseado no MicaZ
$\overline{SD}$	Distancia entre o nó origem e o destino (m)	650, 1300
η	Densidade de nós $(\frac{\text{nós}}{m^2})$	8×10^{-3}
N_p	Número máximo de preâmbulos	98
t_{ciclo}	Tamanho do ciclo (s)	0.1
t_{dados}	Duração do dado $(\%t_{ciclo})$	25, 33, 50
t_{pre}, t_{eACK}	Duração do preâmbulo e eACK (s)	0.512×10^{-3}
τ	Tempo inicial de espera (Split-MAC, PAX-MAC) (s)	$k \cdot t_{data},\ k = 6$
$Taxa$	Taxa de Transmissão	$256kbps$ - Padrão 802.15.4

propagação de espaço livre foi adotado para difusão isotrópica em um meio de propagação ideal. O alcance do rádio foi definido em 40m para um modelo de rádio de raio unitário.

Os nós dos sensores foram distribuídos de forma aleatória na área de simulação e suas localizações formaram um processo de Poisson. O gráfico de conexão resultante é conhecido como grafo geométrico aleatório [Li et al. 2009]. Os nós são estáticos e a informação de posição é trocada no início da simulação. Cada nó conhece a posição dos seus vizinhos. O protocolo MAC proposto é combinado com um protocolo de roteamento geográfico, usando o protocolo GPRS [Karp and Kung 2000].

Todos os parâmetros em diferentes protocolos foram definidos com os mesmos valores. O protocolo GeRaF foi adaptado para evitar o uso de preâmbulos específicos para cada região, de forma a melhorar seu desempenho. Assim, o comportamento do GeRaF é o mesmo do protocolo CMAC, sem o uso do limite (r_0). Portanto, é denominado aqui como GeRaF/CMAC.

Todos os protocolos são assíncronos e utilizam a técnica de prospecção de preâmbulos. A tabela 2 apresenta os parâmetros utilizado para avaliação. O tamanho de pacote avaliado foi de 25%, 33% e 50% do tempo de ciclo, que nas simulações foi de $100ms$. O nó iniciador da transmissão foi colocado a uma distância de $1300m$ do destino final, e uma mensagem foi enviada em cada simulação. Essa mensagem foi dividida em até 4 segmentos.

Para obter resultados estatisticamente relevantes, os experimentos foram repetidos 60 vezes. Quando o pacote não foi entregue (por motivo de colisão, por exemplo), a execução foi desconsiderada.

4.2. Resultados Encontrados

O primeiro experimento foi realizado para verificar a relação entre a latência obtida pelo protocolo e o número de segmentos utilizados. Cada pacote de dados foi deixado inteiro ou dividido em 2, 3 ou 4 segmentos antes de ser enviado pelo Split-MAC. Os resultados do experimento são mostrados na Figura 5, juntamente com os intervalos de confiança.

Pode-se observar que, para todos os pacotes, uma maior divisão do pacote base em segmentos trouxe uma menor latência na transmissão. Isso pode ser explicado por um maior paralelismo na rede, várias partes do mesmo pacote sendo transmitidas em

posições diversas da rede ao mesmo tempo. Pode-se também observar que, dobrando o número de segmentos, não temos a metade da latência, o que seria de esperar em um caso ideal. Isso acontece devido ao *overhead* do protocolo, com os preâmbulos trafegando à frente e também colisões iminentes. Pode-se também observar que o ganho não escala de forma indeterminada. Por exemplo, para pacotes de tamanho 25%, a latência para 3 e 4 segmentos é aproximadamente a mesma. Isso porque, com maior número de segmentos, a cardinalidade do FCS aumenta, mas o número de nós dentro do raio de alcance do encaminhador é limitado. Assim, o FCS não fica completo e a probabilidade de colisão iminente aumenta. Além disso, saltos curtos aumentam a probabilidade de interferência entre os preâmbulos e os diversos segmentos.

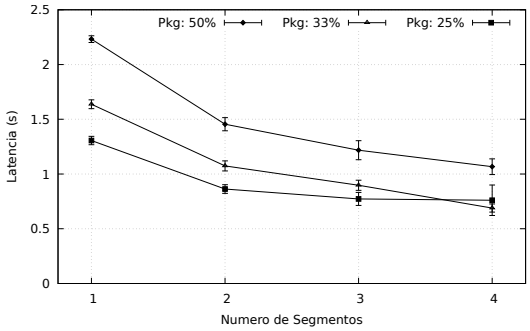

Figura 5. Latência para diferentes números de segmentos. $\alpha = 0.05$

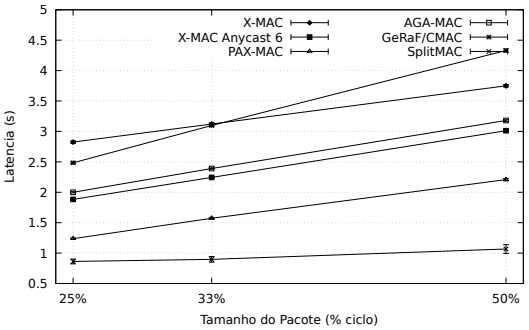

Figura 6. Latência comparativa com outros protocolos. $\alpha = 0.05$

A taxa de sucesso é descrita a seguir. Para pacote com tamanho de 50% do tempo de ciclo, com qualquer número de segmentos, 100%. Para o pacote com tamanho de 33%, as taxas de entrega foram para 1,2,3 e 4 segmentos, 100%, 100%, 98% e 56% respectivamente. Já para o pacote com tamanho de 25% do tempo de ciclo, as taxas de entrega foram aproximadamente, para 1,2,3 e 4 segmentos, 100%, 97%, 38% e 10% respectivamente. Quando o segmento torna-se muito pequeno, a cardinalidade do FCS requerida torna-se muito grande e muitas vezes não existe um número de vizinhos suficiente para preenchê-lo. Assim, o preâmbulo atrasa e colisões iminentes são detectadas. Além disso, o salto torna-se pequeno, fazendo que a distância entre preâmbulos e segmentos possa ser insuficiente, causando colisões com perda de pacote. Pretende-se em trabalhos futuros adaptar as distâncias para evitar perda de pacotes. Em relação as colisões iminentes, o menor número foi encontrado no pacote de tamanho 50% do ciclo e 1 segmento $(0, 13)$ e o maior número no pacote de 25% e 4 segmentos $(1, 67)$. Isso pode ser explicado pelo número de nós no FCS: quanto mais nós no FCS maior a probabilidade de existirem FCS incompletos que atrasam a transmissão do preâmbulo, causando a colisão iminente.

Para o próximo resultado, o Split-MAC foi configurado para utilizar o número de segmentos que traz o melhor resultado para cada tamanho de pacote com taxa de entrega acima de 95%. A figura 6 apresenta a latência obtida para a transmissão de uma mensagem utilizando protocolos assíncronos que utilizam preâmbulos da literatura. Os resultados podem ser comparados com o Split-MAC. Pode-se observar que os protocolos com preâmbulo avançado (Split e PAX MAC) obtiveram os melhores resultados para todos os tamanhos de pacote. Isso pode ser explicado pelo o encadeamento dos nós para recebimento do pacote de dados, que, juntamente com os preâmbulos avançados, permitem

um paralelismo na rede. X-MAC anycast 6 obteve também um bom desempenho devido a característica anycast que reduz consideravelmente o problema do *sleep-delay*. Como é de se esperar, a latência varia com o tamanho do pacote. Pode-se observar também que o Split-MAC demonstrou uma grande redução de latência quando comparado a todos os outros protocolos testados. Essa redução foi de no mínimo 30% da latência quando comparado ao protocolo PAX-MAC.

Para cenários com transmissões esporádicas, o protocolo Split-MAC apresentou uma latência reduzida em comparação aos vários outros protocolos testados. Isso torna possível a utilização de protocolo assíncrono, que economiza energia por não necessitar de sincronização dos nós, em cenários onde a latência da rede tem um papel importante.

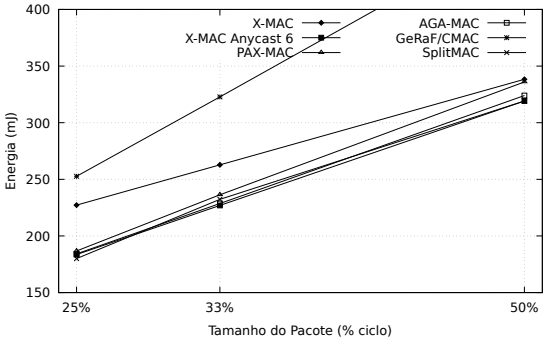

Figura 7. Energia gasta para diversos protocolos e tamanhos de pacotes.

A Figura 7 apresenta o gasto de energia dos diversos protocolos para pacotes de diferentes tamanhos. Somente a energia gasta para transmitir pacotes foi computada (energia gasta normalmente no ciclo de trabalho sem transmissões foi ignorada). Podemos observar que o protocolo anycast GeRaF/CMAC obteve, por larga margem, o maior gasto energético para todos os tamanhos de mensagem. Isso se deve a grande quantidade de nós no FCS, que reduz o tamanho médio de cada salto, aumentando o número total de saltos. Como apresentado em [Heimfarth et al. 2015], um grande número de nós no FCS tem efeito positivo quando pacotes muito pequenos são utilizados. Isso pode ser visto na declividade da curva de gasto energético do GeRaF/CMAC. Em segundo lugar, ficou o protocolo X-MAC. Pode-se observar que todos os outros protocolos tiveram um gasto de energia comparável. Como característica comum, são todos protocolos *anycast* com a cardinalidade do FCS limitada, diferente do GeRaF/CMAC. Dados os resultados alcançados, conclui-se que o Split-MAC obteve, em média, um considerável ganho em latência com custo energético comparável aos outros protocolos com melhor perfil energético para os cenários considerados no presente artigo.

5. Conclusão

RSSFs utilizam ciclos de trabalho para economizar energia, alternando curtos intervalos de atividade com períodos inativos, o que gera um aumento na latência fim-a-fim devido ao fenômeno de *sleep-delay*. Os protocolos assíncronos utilizam uma série de preâmbulos para estabelecer comunicação entre os nós. O aumento na latência ocorre porque o nó transmissor deve esperar até que o receptor esteja acordado para realizar a transmissão dos dados. De forma a amenizar essa desvantagem, a técnica *anycast* é utilizada: em vez de um único encaminhador de mensagem, um conjunto é utilizado, o que reduz, em

média o *sleep-delay*. Uma forma de reduzir ainda mais a latência em protocolos *anycast* é o envio separado de preâmbulos e dados. Preâmbulos trafegam à frente na rede programando o horário de receber o pacote de dados, que trafega ao mesmo tempo alguns *hops* atrasados. No presente trabalho, foi proposto um aprofundamento desse paralelismo: o pacote de dados é dividido em vários segmentos, que trafegam em paralelo em locais diferentes do caminho entre origem e destino, utilizando-se para isso multiplexação espacial. Split-MAC foi desenvolvido para redes com tráfego leve, nas quais as colisões são pouco frequentes, e percursos relativamente longos.

Simulações realizadas mostraram que, para diferentes tamanhos de pacotes, a abordagem apresentada apresentou uma redução de ao menos 30% na latência observada sem aumento no gasto de energia, quando comparado a outros protocolos do estado da arte. Isso demonstra que, para cenários de longa comunicação e baixo tráfego, o Split-MAC pode ser utilizado quando baixa latência é requerida.

Agradecimentos

O presente trabalho foi realizado com apoio da Fundação de Amparo à Pesquisa de Minas Gerais (FAPEMIG), projeto APQ-03095-16.

Referências

Ashraf, F., Vaidya, N., and Kravets, R. (2011). Any-mac: Extending any asynchronous mac with anycast to improve delay in wsn. In *Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2011 8th Annual IEEE Communications Society Conference on*, pages 19–27.

Bachir, A., Dohler, M., Watteyne, T., and Leung, K. (2010). Mac essentials for wireless sensor networks. *Communications Surveys Tutorials, IEEE*, 12(2):222–248.

Buettner, M., Yee, G. V., Anderson, E., and Han, R. (2006). X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, SenSys '06, pages 307–320, New York, NY, USA. ACM.

Cano, C., Bellalta, B., Sfairopoulou, A., and Oliver, M. (2011). Low energy operation in wsns: A survey of preamble sampling mac protocols. *Computer Networks*, 55(15):3351–3363.

Culler, D., Estrin, D., and Srivastava, M. (2004). Guest editors' introduction: Overview of sensor networks. *Computer*, 37(8):41–49.

Doudou, M., Djenouri, D., Barcelo-Ordinas, J. M., and Badache, N. (2016). Delay-efficient mac protocol with traffic differentiation and run-time parameter adaptation for energy-constrained wireless sensor networks. *Wirel. Netw.*, 22(2):467–490.

Du, S., Saha, A., and Johnson, D. (2007). Rmac: A routing-enhanced duty-cycle mac protocol for wireless sensor networks. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 1478–1486.

Heimfarth, T., Giacomin, J., and De Araujo, J. (2015). Aga-mac: Adaptive geographic anycast mac protocol for wireless sensor networks. In *Advanced Information Networking and Applications (AINA), 2015 IEEE 29th International Conference on*, pages 373–381.

Heimfarth, T., Giacomin, J. C., d. Araujo, J. P., and d. Freitas, E. P. (2016). A preamble ahead anycast protocol for wsns. In *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, pages 458–466.

Hong, S.-H. and ki Kim, H. (2009). A multi-hop reservation method for end-to-end latency performance improvement in asynchronous mac-based wireless sensor networks. *Consumer Electronics, IEEE Transactions on*, 55(3):1214–1220.

IEEE Standard (2003). Ieee standard for information technology - telecommunications and information exchange between systems - local and metropolitan area networks specific requirements part 15.4: Wireless medium access control (mac) and physical layer (phy) specifications for low-rate wireless personal area networks (lr-wpans). *IEEE Std 802.15.4-2003*, pages 1–670.

Karp, B. and Kung, H. T. (2000). Gpsr: greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, MobiCom '00, pages 243–254, New York, NY, USA. ACM.

Kumar, A., Zhao, M., Wong, K., Guan, Y. L., and Chong, P. H. J. (2018). A comprehensive study of iot and wsn mac protocols: Research issues, challenges and opportunities. *IEEE Access*, 6:76228–76262.

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

Li, J., Andrew, L. L., Foh, C. H., Zukerman, M., and Chen, H.-H. (2009). Connectivity, coverage and placement in wireless sensor networks. *Sensors*, 9(10):7664.

Liu, S., Fan, K.-W., and Sinha, P. (2007). Cmac: An energy efficient mac layer protocol using convergent packet forwarding for wireless sensor networks. In *Sensor, Mesh and Ad Hoc Communications and Networks, 2007. SECON '07. 4th Annual IEEE Communications Society Conference on*, pages 11–20.

Lu, G., Krishnamachari, B., and Raghavendra, C. (2004). An adaptive energy-efficient and low-latency mac for data gathering in wireless sensor networks. In *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, pages 224–231.

Pyeon, D., Jang, I., Yoon, H., and Kim, D. (2016). Rm-mac: a reservation based multichannel mac protocol for wireless sensor networks. *Wireless Networks*, 22(8):2727–2739.

Sun, Y., Du, S., Gurewitz, O., and Johnson, D. B. (2008). Dw-mac: a low latency, energy efficient demand-wakeup mac protocol for wireless sensor networks. In *Proceedings of the 9th ACM international*, pages 53–62.

Tong, F., Zhang, R., and Pan, J. (2016). One handshake can achieve more: An energy-efficient, practical pipelined data collection for duty-cycled sensor networks. *IEEE Sensors Journal*, 16(9):3308–3322.

Zorzi, M. and Rao, R. (2003). Geographic random forwarding (geraf) for ad hoc and sensor networks: multihop performance. *Mobile Computing, IEEE Transactions on*, 2(4):337–348.