LMT-MAC: um protocolo MAC multicanal livre de colisões para Redes de Sensores Sem Fio

Gilson M. Júnior[1], **Ariel F. F. Marques**[1] **Luiz H. A. Correia**[1]

[1]Departamento de Ciência da Computação – Universidade Federal de Lavras (UFLA)
Caixa Postal 3037 – 37200-000 – Lavras – MG – Brazil

{gilsonmj,arielmarques}@posgrad.ufla.br, lcorreia@dcc.ufla.br

Abstract. *The protocol for the Medium Access Control (MAC) layer directly influences the operation of Wireless Sensor Networks (WSN), impacting on multiple metrics as latency, throughput and delivery rate. In current literature, multichannel scheduled protocols use complex mechanisms for time synchronization, channel and slots allocation, unsuitable for devices with limited hardware. In this paper, we present the Lightweight Multichannel Tree MAC (LMT-MAC), an efficient MAC protocol based on Time Division Multiple Access (TDMA), multichannel and contention-free. Results obtained by extensive simulations show that LMT-MAC is more effective than other traditional protocols and its implementation in real sensor nodes demonstrate its feasibility.*

Resumo. *O protocolo da camada de controle de acesso ao meio (MAC) influencia diretamente na operação das Redes de Sensores Sem Fio (RSSF), impactando em diversas métricas como latência, vazão e taxa de entrega. Na literatura atual, protocolos TDMA multicanal usam mecanismos complexos para sincronização de tempo, alocação de canais e slots, geralmente inviáveis para dispositivos com hardware limitado. Neste trabalho, é proposto o Lightweight Multichannel Tree MAC (LMT-MAC), um protocolo MAC eficiente, baseado em TDMA, multicanal e livre de contenção. Resultados obtidos por meio de simulações mostram que o LMT-MAC é mais eficiente que outros protocolos tradicionais, e sua implementação em nós reais demonstram sua viabilidade.*

1. Introdução

As Redes de Sensores Sem Fio (RSSF) têm sido empregadas em diversos segmentos para auxiliar no monitoramento e controle de processos, rastreamento de objetos, avaliação de condições do solo para agricultura, monitoramento de infraestrutura, dentre outras aplicações [Akyildiz et al. 2002]. Cada aplicação possui requisitos específicos, os quais determinam as características dos nós sensores e os protocolos de comunicação que devem ser utilizados.

As primeiras aplicações de RSSF tratavam principalmente de monitoramento ambiental, e apenas poucos projetos levavam em consideração requisitos de qualidade de serviço (QoS) para aplicações de tempo real [Romer and Mattern 2004]. Com o desenvolvimento de dispositivos e protocolos mais robustos para comunicação sem fio, soluções baseadas em RSSF para aplicações que demandam maior confiabilidade começaram a emergir, visando oferecer formas de monitoramento mais flexíveis e com custo inferior ao de soluções cabeadas, tradicionalmente utilizadas [Gungor and Hancke 2009].

Redes para missões críticas (*Mission Critical Networks*) são empregadas para aplicações em que uma ou mais características devem ser priorizadas, a fim de satisfazer requisitos de qualidade de serviço. Essas redes diferem tanto de redes comuns, usadas para a conexão de computadores, quanto entre si. Diferentes aplicações demandam a priorização de diferentes requisitos, os quais são fortemente determinados pelo protocolo MAC. Alguns dos requisitos mais comuns a serem priorizados por uma RSSF incluem: energia, vazão de dados (capacidade de entrega de pacotes para os nós), latência, taxa de entrega, resiliência (capacidade da rede se manter operante quando submetida a interferências) e adaptabilidade (reorganização da rede quando ocorrem mudanças de topologia devido a movimentação ou falhas de nós).

Os protocolos MAC propostos inicialmente para RSSF geralmente operavam em apenas um canal, em muitos casos por decisão de projeto e em outros por limitações dos transceptores usados. Como em várias aplicações o foco principal era em economia de energia, fatores como latência e vazão ficavam em segundo plano. Com o suporte a mais canais de comunicação, protocolos multicanais têm sido propostos como forma de aumentar a quantidade de transmissões simultâneas, e, assim, reduzir a latência e aumentar a vazão das redes. Por outro lado, protocolos multicanais apresentam novos desafios como a coordenação de comunicações, atribuição de canais, prevenção de colisões, dentre outros [Incel 2011]. Em protocolos TDMA, a alocação de canais e a distribuição de *slots* tem custo computacional elevado, sendo comum o uso de heurísticas para fazer essa alocação de forma centralizada, geralmente no nó *sink*, que depois é propagada aos demais nós da rede. Essa solução, no entanto, pode ser inviável para RSSF em que os nós possuem recursos limitados.

Em RSSF, o protocolo MAC é apenas parte dos vários softwares em execução nos nós sensores, que podem executar outras tarefas como leitura de sensores, agregação e compressão de dados, comunicação cabeada com outros equipamentos e acionamento de atuadores. Por isso, os recursos consumidos por esse protocolo devem ser minimizados, a fim de causar o menor impacto possível na execução das demais operações.

Neste trabalho é proposto o *Lightweight Multichannel Tree MAC* (LMT-MAC), um protocolo MAC voltado para RSSF compostas por nós cujos recursos de hardware são limitados. O protocolo é focado para aplicações que requerem baixa latência, principalmente naquelas em que é necessário estimar o tempo máximo em que os dados coletados serão enviados até o nó de destino. A configuração dos nós sensores pode ser feita de modo *offline* (para redes planejadas), salvando as configurações dos nós na memória interna de cada um (dispensando o uso de um protocolo específico para a configuração) ou com o uso de um protocolo para configuração em redes não planejadas ou com mobilidade. Com o LMT-MAC, diferentes nós da RSSF podem se comunicar simultaneamente utilizando canais de comunicação diferentes, permitindo obter uma alta vazão de dados. Além disso, o protocolo emprega *buffers* para que os pacotes sejam armazenados até que os nós tenham oportunidade de transmissão, com o objetivo de oferecer também uma alta taxa de entrega.

O trabalho está organizado da seguinte forma: a Seção 2 apresenta os trabalhos relacionados. A Seção 3 apresenta em detalhes o protocolo LMT-MAC. A Seção 4 traz a comparação do protocolo com o TreeMAC, e os resultados obtidos com a implementação em nós reais. Por fim, a Seção 5 apresenta as conclusões.

2. Trabalhos Relacionados

Com o foco do LMT-MAC em alta vazão, baixa latência e alta taxa de entrega, nesta seção são analisados protocolos que priorizam algum desses requisitos.

Visando obter baixa latência, o *Real Time Hybrid MAC* (RTH-MAC) [Abdeli et al. 2013] emprega um esquema TDMA centralizado, configurado por uma estação base. Após uma fase inicial de descoberta de vizinhos, a estação base executa um algoritmo para alocação de *slots* e canais para cada nó. O algoritmo de agendamento, baseado em coloração de vértices, faz com que vizinhos a dois saltos de distância utilizem uma tupla (*slot*, canal) diferente. A avaliação do protocolo foi realizada por simulações em MATLAB, e os resultados mostram que o protocolo foi capaz de enviar pacotes através de 5 saltos com uma latência fim-a-fim inferior a 20 milissegundos. O tamanho dos *slots* é calculado pela divisão do tamanho do pacote pela taxa de transferência máxima nominal do transceptor, e os nós são considerados perfeitamente sincronizados após a primeira transmissão, sendo dispensado o uso de períodos de guarda para superar os erros de sincronização. Com essas assunções, é possível definir um *slot* de aproximadamente 1,5 milissegundos (no pior caso, quando são usadas mensagens de confirmação), e, portanto, o protocolo atinge uma latência nas simulações que dificilmente seria possível na implementação em nós reais.

Voltado para aplicações industriais, o PriorityMAC [Shen et al. 2014] é um protocolo híbrido de CSMA e TDMA que emprega classificação de mensagens, dando prioridade mais alta para mensagens críticas e usando métodos de acesso diferentes para tráfego de maior prioridade. O protocolo permite obter latências inferiores ao protocolo MAC do padrão WirelessHART [HART 2015], mas ainda possui problemas quando há contenção entre pacotes de alta prioridade. Se pacotes em nós diferentes são classificados com a mesma prioridade, eles serão transmitidos usando o mesmo modo de acesso ao meio, o que causará colisões e irá gerar atraso justamente na transmissão das informações mais críticas. O PriorityMAC não tira proveito dos recursos de multicanal por usar apenas um canal de comunicação para toda a rede.

O MC-LMAC [Incel et al. 2011] estende o protocolo LMAC para usar múltiplos canais de comunicação. Na configuração inicial da rede são determinados os *slots* e canais que cada nó utilizará para transmitir seus dados. A quantidade de *slots* depende da densidade da rede, sendo determinada durante a inicialização. Os *slots* são divididos em duas partes: a primeira contém vários *subslots*, um para cada canal utilizado. Em cada *subslot*, os nós ouvem no canal relativo ao *subslot* e aguardam que o detentor do *slot* envie uma mensagem de controle, a qual contém o endereço de destino do pacote de dados. Na segunda parte do *slot*, o transmissor e o nó de destino se configuram para o mesmo canal, e os pacotes de dados são transmitidos. A previsibilidade do protocolo é prejudicada no caso da ocorrência de disputas, quando dois transmissores tentam enviar dados ao mesmo nó. Ambos transmissores enviam mensagens informando suas transmissões em seus respectivos *subslots*, no entanto, na fase de transmissão de dados, o nó receptor deve escolher de qual transmissor irá receber por meio de algum mecanismo de prioridade. O outro transmissor deverá aguardar seu próximo *slot* para tentar enviar seus dados. Assim, a transmissão desses dados estará sujeita ao mecanismo usado para priorização, o que dificulta a predição de quando os dados poderão ser transmitidos. Além disso, com a divisão da parte inicial dos *slots* em um *subslot* para cada canal, é necessário aumentar o

tamanho do *slot* para acomodar a seção de transmissão de dados, prejudicando a latência, ou essa seção de dados será menor, prejudicando a vazão.

O TreeMAC [Song et al. 2009] é um protocolo TDMA que organiza a rede como uma árvore e divide um ciclo em *frames*, e cada *frame* em três *slots*. Os nós calculam seu *slot* de comunicação dentro de um *frame* de acordo com a sua profundidade na árvore, e a distribuição de *frames* para cada nó é baseada na quantidade total de filhos que o nó possui. Assim, nós com mais filhos possuem mais oportunidades de transmissão. No TreeMAC, contudo, a transmissão de dados é unidirecional, sendo um protocolo apenas para a coleta de dados, e usa somente um canal para toda a rede. A configuração da rede é feita por meio de um protocolo baseado em contenção para a descoberta de vizinhança. Ao obter a topologia, os *frames* são distribuídos de acordo com a necessidade de cada nó. Após a configuração da rede, cada nó tem conhecimento dos períodos em que deve se manter ativo e dos períodos em que pode desativar o transceptor. Isso reduz a necessidade de mensagens de controle, além de aumentar a previsibilidade do protocolo por evitar disputas por *slots* de transmissão.

A análise dos protocolos MAC propostos atualmente para redes críticas indica a viabilidade de um novo protocolo, que integre diferentes mecanismos para oferecer qualidade de serviço, e que use de forma inteligente os recursos disponíveis nos dispositivos. Além disso, deve considerar dispositivos com restrições de recursos, usando métodos simplificados de configuração.

3. LMT-MAC

O LMT-MAC foi desenvolvido com base no TreeMAC [Song et al. 2009]. Porém, emprega mensagens de controle para permitir comunicação bidirecional, tanto coleta de dados, quanto envio de comandos de atuação de um servidor. Além disso, o LMT-MAC permite comunicação em múltiplos canais, utilizando um esquema simplificado de alocação de canais e *slots*, baseado na distância para o *sink* e na quantidade de nós na sub-rede.

A rede é organizada logicamente como uma árvore, tendo o *sink* como raiz. A configuração dos nós pode ser feita de forma manual, especificando diretamente o endereçamento, a relação entre os nós e os *slots* destinados a cada nó, ou por meio de um protocolo de descoberta de vizinhos baseado em contenção quando a rede é iniciada, como ocorre no TreeMAC. O protocolo de descoberta deve ser capaz de identificar a vizinhança de todos os nós e, então, determinar a árvore geradora mínima da rede. Um período de tempo, ou ciclo, é dividido em vários *frames*, e cada *frame* em dois *slots*, como mostrado na Figura 1.

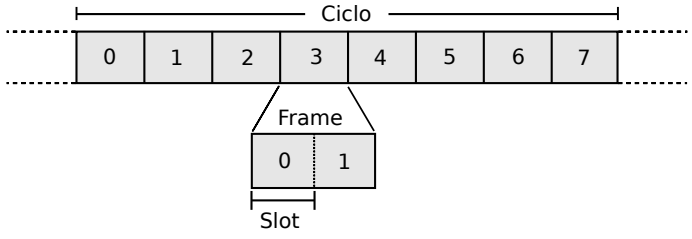

Figura 1. Divisão de tempo usada pelo LMT-MAC.

Para auxiliar na descrição do LMT-MAC, a seguinte notação será adotada:

- N: nós da rede.
- f: canais de comunicação usados.
- F_u: conjunto de *frames* atribuídos ao nó u.
- C_u: conjunto de nós filhos do nó u.
- C_u^k: k-ésimo filho do nó u.
- d_u: profundidade do nó u na árvore (saltos para o *sink*).
- T_{slot}: tempo de duração de um *slot*.
- $F_{LMT-MAC}$: conjunto de todos os *frames* da rede.

Os nós recebem uma quantidade de *frames* proporcional à quantidade de nós em sua sub-rede, ou seja, se um nó tem dois nós filhos em sua sub-rede, ele deve receber dois *frames* (um para cada filho), e um *frame* adicional para a transmissão dos pacotes que não puderam ser transmitidos durante os *frames* anteriores, ou que foram gerados pelo próprio nó. Assim, a quantidade total de *frames* na rede é $|F_{LMT-MAC}| = |N| - 1$, pois o *sink* não precisa de um *frame* adicional. Os *frames* recebidos por um nó i são um subconjunto dos *frames* de seu nó pai u, de modo que $F_{C_u^i} \subseteq F_u$. Para evitar colisões entre nós filhos, cada um recebe um conjunto disjunto de *frames*, de modo que $F_{C_u^i} \cap F_{C_u^j} = \emptyset \ (se\ i \neq j)$. Além disso, os *frames* de um nó são sempre contíguos, de modo que cada nó necessite saber apenas o número do seu *frame* incial, do seu *frame* final e a quantidade total de *frames* na rede. Os nós podem desativar o transceptor para economizar energia quando não estão no seu período ativo (fora de seus *frames*).

Com o conhecimento de seus *frames*, os nós devem definir seus *slots* de transmissão e canais de operação. Um nó u calcula seu *slot* de transmissão usando a Equação 1, e pode calcular o *slot* de seus nós filhos pela Equação 2. O *slot* de transmissão de um nó u é dado por S_u e pode assumir os valores 0 ou 1, indicando um *slot* dos *frames*.

$$S_u \in \{0, 1\} = (d_u - 1) \bmod 2 \tag{1}$$

$$S_{C_u} \in \{0, 1\} = d_u \bmod 2 \tag{2}$$

Após calcular em qual *slot* devem transmitir, os nós determinam em qual canal devem operar em cada *slot*. O número de configurações varia de acordo com a quantidade de frequências utilizadas, tal que $configurações = |f| \times 2$, onde $|f|$ é a quantidade de canais e 2 devido ao uso de dois *slots* por *frame*. Assim, com três canais existem seis configurações diferentes, que os nós usam de acordo com sua profundidade. A Tabela 1 apresenta as configurações para uma rede configurada para usar três canais, de acordo com a profundidade de um nó u. Os valores dos canais podem ser usados como índice para frequências do transceptor, permitindo o uso de canais com intervalos maiores.

Tabela 1. Configurações para uma rede com três canais.

$d_u \bmod 6$	Canal no *slot* 0	Canal no *slot* 1
0	0	2
1	0	0
2	1	0
3	1	1
4	2	1
5	2	2

A transmissão de dados é sempre iniciada pelo nó filho, que verifica em seu *buffer* a existência de pacotes que devem ser encaminhados para o nó pai, e os transmite. Ao

terminar o envio, o nó filho transmite um pacote de controle informando que está pronto para receber pacotes, e o nó pai, por sua vez, transmite os pacotes em buffer destinados ao filho. Ao finalizar as transmissões, o nó pai envia uma mensagem de controle ao nó filho, e ambos podem desativar seus transceptores pelo restante do *slot*.

Nas mensagens de controle trocadas entre os nós são transmitidas as informações necessárias para configuração dos nós e sincronização de tempo. Para configuração dos nós são transmitidas: quantidade de *frames* total, quantidade de *frames* do nó, *frame* inferior e número de saltos para o *sink*. Para sincronização são enviados: tempo global da rede, *frame* atual, *slot* atual e início do próximo *slot*. O LMT-MAC utiliza sincronização global de tempo, na qual o valor do relógio interno do *sink* é tido como referência, e os demais nós da rede devem estimar esse valor. O protocolo de sincronização deve ser capaz de estimar o tempo global a partir das informações do pacote de controle, e sinalizar quando o período de um *slot* se esgotou. Assim o LMT-MAC atualiza seus contadores de *frame* e *slot*, e se configura para o canal apropriado para realizar a transmissão ou recepção de dados. A troca de canais em transceptores comumente usados em nós sensores, como CC2420 e nRF24l01+, é inferior a 1 milissegundo [Instruments 2006][Nordic_Semiconductor 2013], e, portanto, seu impacto pode ser desconsiderado ou acomodado dentro do período de guarda dos *slots*.

O LMT-MAC é desacoplado do protocolo de sincronização, e com protocolos mais sofisticados, pode-se obter uma sincronização mais precisa, permitindo reduzir os tempos de *slot* e guarda, ao custo de algoritmos que requerem mais recursos de hardware. Assim, a escolha do protocolo de sincronização depende dos nós sensores utilizados na RSSF e de decisões de projeto. Ainda assim, no início dos *slots* são definidos períodos de guarda para evitar erros de transmissão devido a imprecisão da sincronização. A avaliação do protocolo de sincronização está fora do escopo deste trabalho.

Cada nó é capaz de estimar a latência máxima de seus pacotes para o *sink*, com base nos valores de T_{slot} (duração de *slot*), d_u (profundidade do nó), $|F_{LMT-MAC}|$ (quantidade total de *frames*) e $|F_u|$ (quantidade de *frames* pertencentes ao nó), usando a Equação 3. A equação considera que o período de um *slot* é suficiente para a transmissão de todos os pacotes em *buffer*. Dessa forma, pode-se planejar a disposição de nós e a distribuição de *frames* de acordo com a latência máxima desejada para determinados nós, ou verificar se nós específicos serão capazes de retornar leituras com intervalos apropriados à aplicação.

$$L_{Max}u = T_{slot} \times (d_u + ((|F_{LMT-MAC}| - |F_u|) \times 2) + 1) \tag{3}$$

4. Resultados e Discussão

Nesta seção, os resultados obtidos com as simulações e os testes em plataforma real são apresentados e discutidos. O LMT-MAC foi comparado ao TreeMAC por meio de exaustivas simulações usando o OMNeT++ 4.6 em conjunto com o *framework* INET 3.00. Além disso, foram realizados experimentos com nós reais que comprovam a viabilidade do LMT-MAC. As seguintes topologias foram usadas para avaliação do protocolo:

1. **Linear:** nesta topologia os nós estão organizados em linha e se comunicam apenas com seus vizinhos imediatos. Essa organização está apresentada na Figura 2.

2. **Árvore binária:** a rede foi organizada como uma árvore binária distribuída regularmente pelo espaço. Essa topologia é apresentada na Figura 3.
3. **Disposição dos nós reais:** disposição usada na instalação real da RSSF, permitindo comparar os resultados obtidos na simulação com aqueles obtidos na RSSF real. Além disso, permite verificar o comportamento da RSSF caso o protocolo MAC utilizado fosse o TreeMAC. A topologia é mostrada na Figura 4.

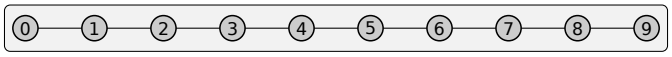

Figura 2. Disposição linear dos nós.

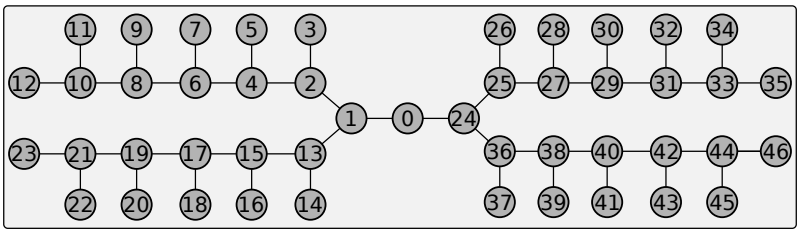

Figura 3. Disposição regular dos nós.

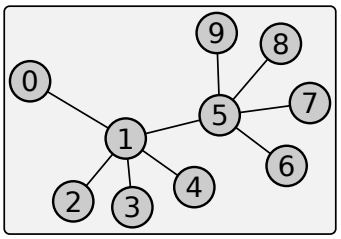

Figura 4. Configuração da rede com a disposição dos nós reais.

As simulações foram feitas com as 3 topologias, enquanto os testes reais foram feitos usando a topologia 3. Os parâmetros das simulações são descritos na Tabela 2. Foram simuladas RSSF com 10, 30 e 47 nós. Cada *slot* tem duração de 20 milissegundos, com tempo de guarda de 1 milissegundo. A taxa de transmissão e a potência foram configuradas para os valores usados nos nós reais, 2 Mbps e 1 mW que são os valores máximos suportados pelo transceptor. Como o protocolo TreeMAC não faz uso de *buffer* para armazenar os pacotes a serem encaminhados, o protocolo foi modificado para usar um *buffer* da mesma forma que o LMT-MAC. Foram usados *buffers* com capacidades para 1, 5, 10 e 20 pacotes.

Um gerador de tráfego constante foi usado para os testes, com configurações de 1, 5, 20 e 40 pacotes por segundo (pps). Cada pacote gerado era armazenado no *buffer* e mantido até que o nó pudesse transmití-lo. Caso a capacidade do *buffer* fosse esgotada, novos pacotes eram descartados e considerados como perdas a serem tratadas pela camada superior.

O tempo de simulação em todos os casos foi de 2000 segundos, com os nós iniciando a transmissão dos pacotes aos 2 segundos para garantir que todos estivessem devidamente configurados, e parando as transmissões aos 1200 segundos, para garantir que todos os pacotes transmitidos pelos nós chegassem ao destino até o final da simulação. Nas simulações foram analisadas, em cada topologia, a taxa de entrega, a vazão e a latência fim-a-fim. Nos testes reais, foram avaliadas a taxa de transmissão ponto-a-ponto e a latência fim-a-fim.

Tabela 2. Parâmetros das simulações.

Parâmetro	Topologia		
	Linear (1)	Árvore (2)	Real (3)
Número de nós	30	47	10
Número de *frames*	29	46	9
Tamanho de *slot*	20 ms		
Tempo de guarda	1 ms		
Taxa de transmissão	2 Mbps		
Potência de transmissão	0 dBm (1 mW)		
Tráfego (pps)	1, 5, 20, 40		
Tamanho de *buffer* (pacotes)	1, 5, 10, 20		
Tempo de simulação	2000 s		

Os testes em plataforma real foram feitos usando 10 nós baseados na plataforma de prototipagem Arduino Uno [Arduino 2015] e no rádio nRF24l01+ [Nordic_Semiconductor 2013], organizados como na topologia mostrada na Figura 4.

Os parâmetros dos testes estão sumarizados na Tabela 3. Foram usados *slots* de 100 milissegundos, tempo de guarda de 20 milissegundos para superar os erros do protocolo de sincronização. Para testes de sincronização essa configuração usou um *frame* adicional. O rádio foi configurado com velocidade de 2 Mbps, potência de 0 dBm (1 mW) e *payloads* de 32 Bytes. O *payload* do quadro transmitido pelo rádio é o espaço reservado para dados das camadas superiores, como os campos de controle usados pelo LMT-MAC. Os nós contavam com espaço em *buffer* para 20 pacotes.

Tabela 3. Parâmetros dos testes reais.

Parâmetro	Valor
Número de nós	10
Número de *frames*	10
Tamanho de *slot*	100 ms
Tempo de guarda	20 ms
Taxa de transmissão	2 Mbps
Potência de transmissão	0 dBm (1 mW)
Tamanho de *payload*	32 Bytes
Tamanho de *buffer*	20 pacotes

Nos testes reais, a taxa de transmissão ponto-a-ponto avaliou a capacidade de comunicação entre dois nós, bem como a quantidade de dados que pode ser trocada em cada *slot*. A latência fim-a-fim foi estimada pela soma dos tempos que os pacotes ficavam armazenados em *buffer* antes de serem encaminhados.

A análise estatística dos resultados foi feita considerando intervalos de confiança de 99% em todos os casos. Nos gráficos de taxa de entrega e vazão os valores para o intervalo de confiança foram omitidos por apresentarem valores demasiadamente pequenos.

4.1. Taxa de Entrega

Para avaliar a taxa de entrega, todos os nós geravam pacotes com destino ao *sink*, e posteriormente foi verificado quantos pacotes foram efetivamente entregues ao destino. Da diferença entre os dois valores obteve-se a quantidade de pacotes perdidos. Com isso, é possível avaliar a confiabilidade do protocolo MAC em RSSF com determinados tipos de tráfego e com diferentes tamanhos de *buffer*.

Os gráficos das Figuras 5, 6 e 7, apresentam a taxa de entrega obtida respectivamente para a topologia linear, árvore binária e disposição dos nós reais. Os resultados

mostram que a taxa de entrega dos protocolos depende diretamente das capacidades dos *buffers* utilizados. Quando os *buffers* são suficientemente grandes para a quantidade de tráfego gerado, ou a rede possui poucos nós, ambos protocolos conseguem entregar 100% dos pacotes.

Com o aumento da quantidade de nós, o tamanho de ciclo também cresce, o que faz com que os nós tenham oportunidades de transmissão em intervalos maiores. Assim, com o gerador de tráfego criando novos pacotes constantemente, e os nós tendo menos chances de transmití-los, a capacidade dos *buffers* é esgotada, e mais pacotes são descartados, como notado nos gráficos das Figuras 5 e 6. Com um ciclo menor e mais comunicações simultâneas, o LMT-MAC permite que os pacotes fiquem retidos em *buffer* por menos tempo que no TreeMAC, liberando espaço para os novos pacotes gerados e reduzindo as perdas por escassez de *buffer*.

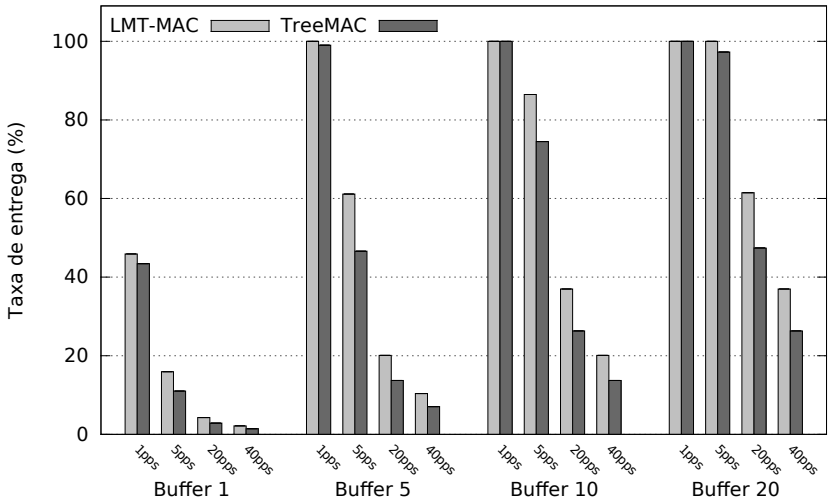

Figura 5. Taxa de entrega para a topologia linear.

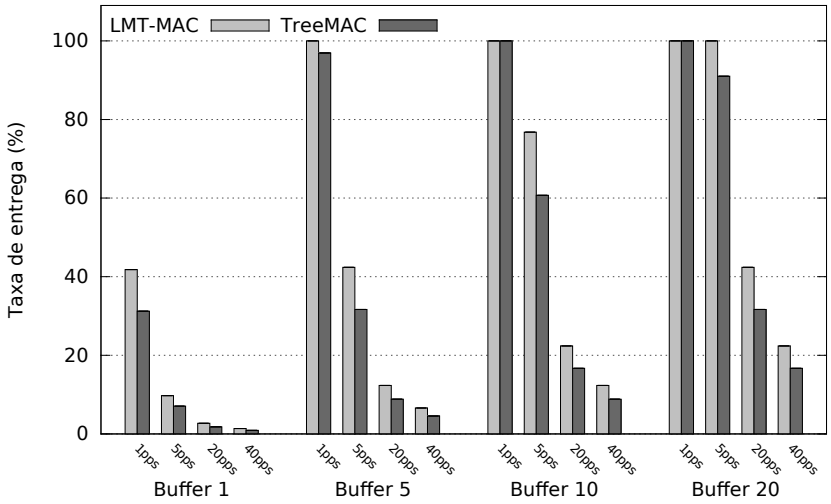

Figura 6. Taxa de entrega para árvore binária.

4.2. Vazão

A avaliação da vazão se deu pela análise da quantidade de pacotes por segundo que chegavam ao *sink*. Nas simulações foram usados pacotes de 321 bits, mesmo tamanho dos

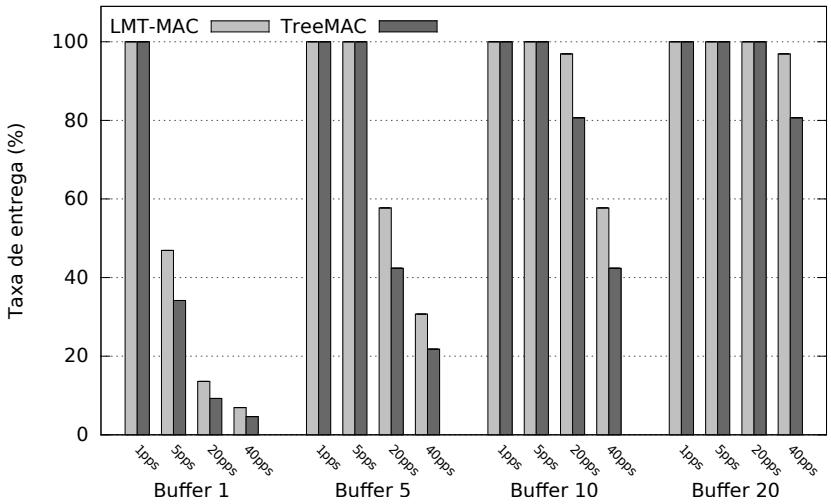

Figura 7. Taxa de entrega para a disposição real.

pacotes gerados pelo transceptor utilizado nos testes reais. Os gráficos das Figuras 8, 9 e 10 trazem os resultados obtidos com os testes de vazão.

A análise dos gráficos mostra que o LMT-MAC apresenta vazão superior ao TreeMAC na maioria dos casos. Quando a quantidade de tráfego gerada é baixa, ambos protocolos apresentam o mesmo comportamento. O LMT-MAC atinge maior vazão quando há mais tráfego sendo gerado, por permitir mais comunicações simultâneas devido ao uso de multicanal. Além disso, a divisão de cada *frame* em apenas dois *slots* permite que as mensagens cheguem ao *sink* em intervalos menores, resultando em maior vazão.

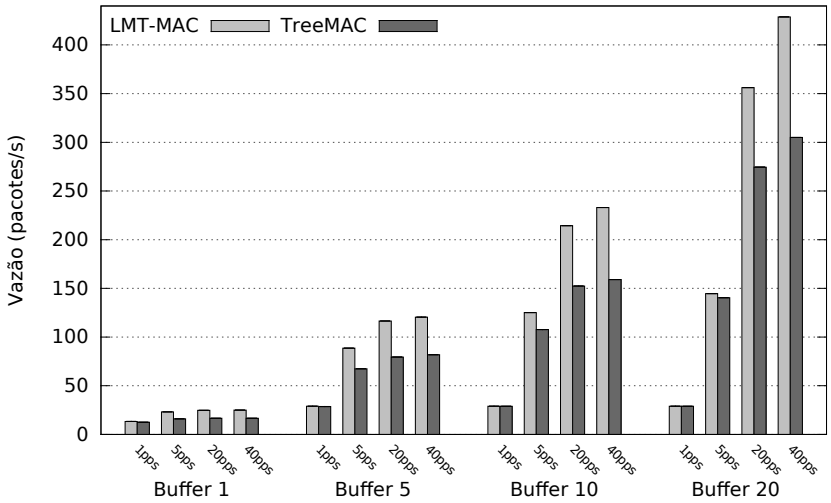

Figura 8. Vazão de dados para o *sink* na topologia linear.

4.3. Latência

Para a avaliação da latência, mediu-se o tempo levado desde a criação de um pacote em um nó, até o seu processamento no nó de destino. Na disposição linear, cujo gráfico é apresentado na Figura 11, pode-se notar o impacto da distância no crescimento da latência nos dois protocolos. Novamente, o LMT-MAC apresenta resultados superiores por ser

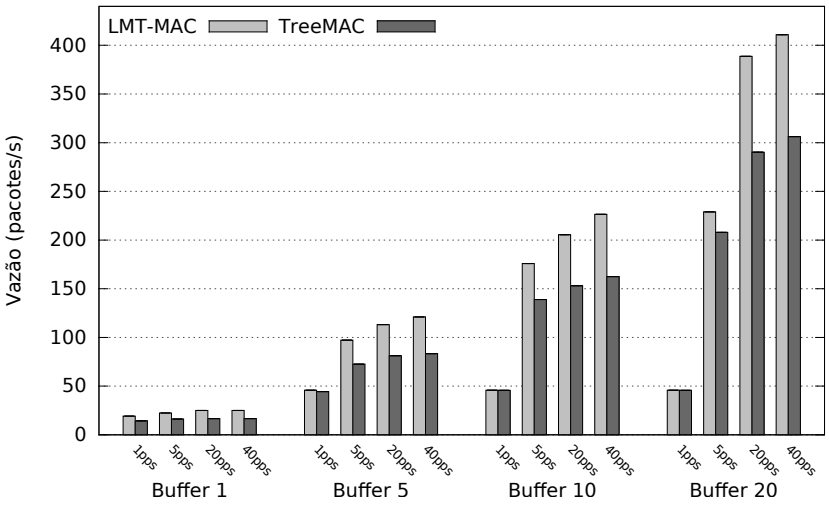

Figura 9. Vazão de dados para o *sink* na árvore binária.

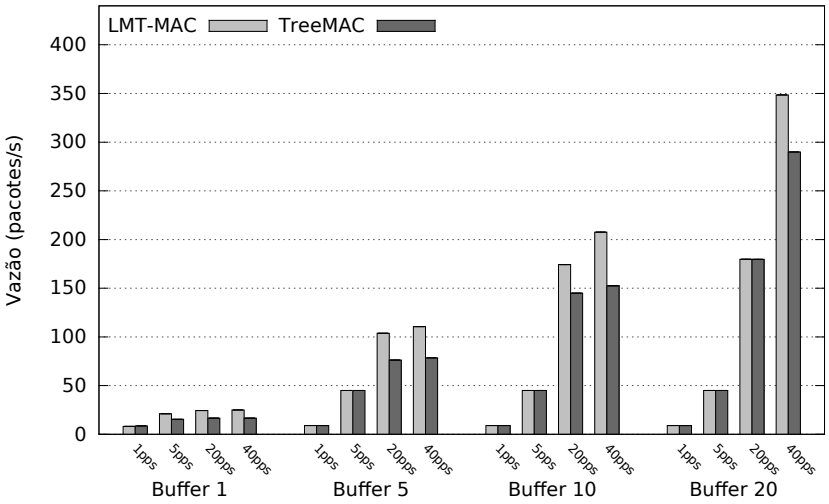

Figura 10. Vazão de dados para o *sink* disposição real.

multicanal e usar menos *slots* por *frame*. Com *frames* menores, os pacotes ficam retidos em *buffer* por menos tempo, dado que os intervalos de transmissão são menores e a quantidade de comunicações simultâneas é maior, devido ao uso de multicanal.

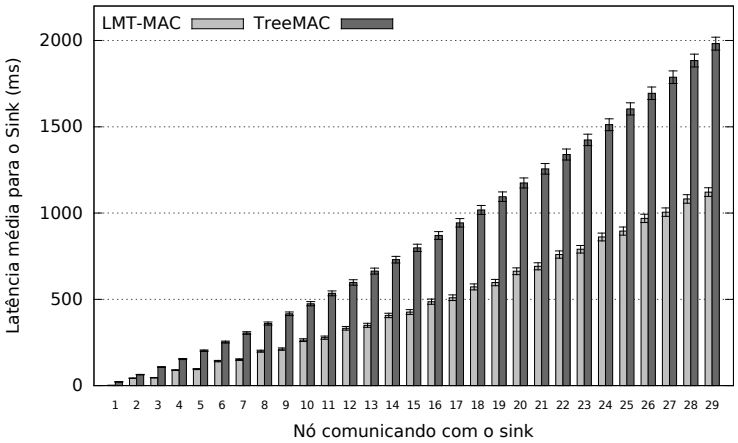

Figura 11. Latência para o *sink* na topologia linear.

O gráfico da Figura 12 apresenta os resultados obtidos para a disposição em árvore binária. Para apresentação dos dados, foi selecionada uma sub-árvore contendo os nós de 1 a 12, e os resultados obtidos para as demais sub-árvores da rede apresentaram o mesmo comportamento. O LMT-MAC obteve latência inferior ao TreeMAC em todos os casos. Em ambos protocolos, os nós que têm mais filhos, e, portanto, têm mais *frames*, apresentam menor latência. No entanto, no LMT-MAC o crescimento da latência é menor, devido à quantidade menor de *slots* por *frame*.

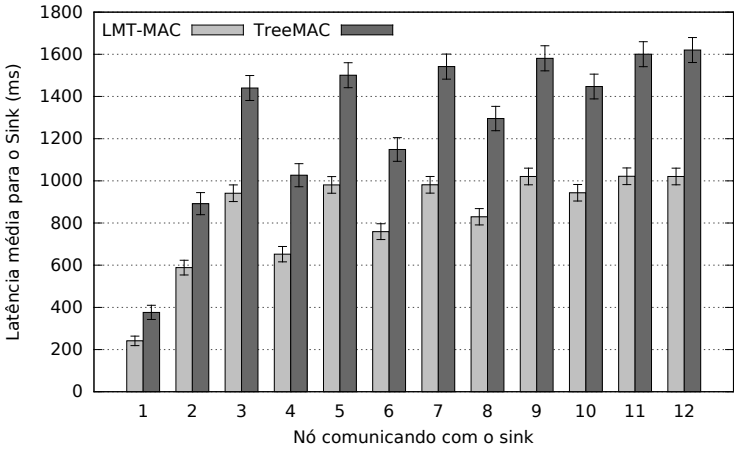

Figura 12. Latência para o *sink* na árvore binária.

No gráfico da Figura 13 são apresentados os resultados para a disposição dos nós reais. A semelhança dos resultados obtidos para nós que possuem a mesma configuração (distância para o *sink* e quantidade de *frames*) em ambos protocolos mostra a possibilidade de predição de tais valores, como é o caso dos nós 2 a 4, e dos nós 6 a 9. O nó 5, por sua vez, está à mesma distância do *sink* que os nós 2 a 4, porém, apresenta latência significativamente inferior por ter mais *frames* de transmissão. Em todos os casos, o LMT-MAC apresentou latência consideravelmente inferior ao TreeMAC, da mesma forma que nos casos anteriores, devido à menor quantidade de *slots* por *frame* e uso de multicanal.

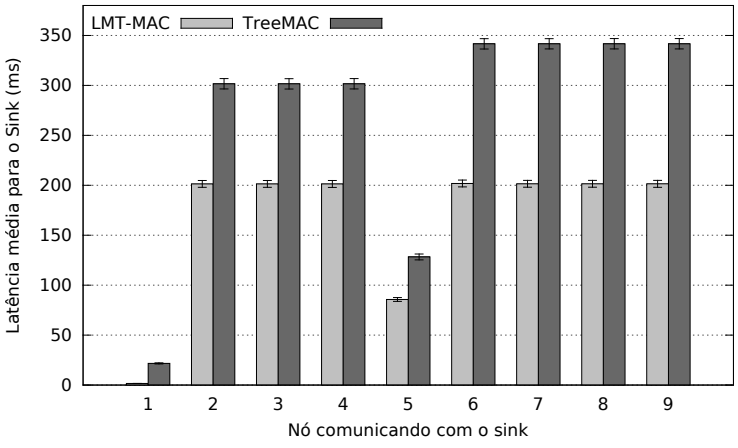

Figura 13. Latência para o *sink* na disposição real.

Nos testes com nós reais, os valores médios obtidos para latência são apresentados na Tabela 4. A tabela traz os tempos mínimo, máximo e médio para transmissão dos pacotes para o *sink* a partir de nós em posições variadas na rede. Pode-se notar que

mesmo estando à mesma distância (em saltos) para o *sink*, a comunicação com o nó 5 tem latência consideravelmente inferior em relação ao nó 2, de modo semelhante ao obtido nas simulações, devido à maior quantidade de *frames*.

Tabela 4. Latência de transmissão para o *sink*.

Origem	Saltos	Tempo mínimo	Tempo máximo	Tempo médio
Nó 1	1	1,66 ms	397,59 ms	115,33 ms
Nó 2	2	115,50 ms	4.097,30 ms	2.091,11 ms
Nó 5	2	103,54 ms	2.495,06 ms	920,86 ms
Nó 6	3	312,90 ms	8.221,24 ms	4.362,53 ms

4.4. Taxa de Transmissão

Nos testes com nós reais, a taxa de transmissão foi testada com o envio de 20 pacotes (capacidade do *buffer*) entre pares de nós. Cada pacote tem um tamanho total de 321 bits, e foram transmitidos em aproximadamente 23 milissegundos, tanto em *upstream* (do nó filho para o nó pai), quanto em *downstream* (do nó pai para o nó filho), obtendo uma vazão média de aproximadamente 270 Kbps, transmitindo um total de 802,5 Bytes. A Tabela 5 sumariza os resultados obtidos.

Tabela 5. Taxa de transmissão entre dois nós.

	Nó filho $\rightarrow$ Nó pai	Nó pai $\rightarrow$ Nó filho
Pacotes enviados/frame	20	20
Tempo médio de transmissão	$23,72\ ms\ \ (\pm\ 0,043\ ms)$	$23,65\ ms\ \ (\pm\ 0,005\ ms)$
Velocidade	$270.608\ bps\ \ (\pm\ 490\ bps)$	$271.411\ bps\ \ (\pm\ 59\ bps)$
Total transmitido/frame	$802,5\ B$	$802,5\ B$

5. Conclusão

Os resultados obtidos nas simulações mostram que com o uso de multicanal, e consequente redução da quantidade de *slots* necessários em um *frame* para evitar colisões, fez com que o LMT-MAC obtivesse resultados superiores ao TreeMAC na vasta maioria dos casos. A necessidade dos nós reterem pacotes em *buffer* até que tenham uma oportunidade de transmissão, faz com que a taxa de entrega dos protocolos dependa diretamente do tamanho do ciclo, do tamanho do *buffer* e da quantidade de tráfego gerada. Por usar um *slot* a menos que o TreeMAC, e empregar multicanal, o LMT-MAC consegue entregar mais pacotes e com menor latência. Pelos mesmos motivos, a capacidade de vazão obtida com o LMT-MAC é superior.

A implementação em nós reais implicou em modificação nos tempos de operação, devido a limitações de hardware dos nós sensores (32 KB de memória de programa e 2 KB de memória RAM), exigindo um método mais simples e menos eficaz de sincronização de tempo. Isso resultou na definição de um *slot* maior (100 ms em nós reais contra 20 ms nas simulações), e em tempos de guarda mais altos (20 ms nos nós reais e 1 ms nas simulações) para superar os erros de sincronização. No entanto, essa implementação foi indispensável para avaliar o funcionamento do protocolo em plataformas reais, e criar uma base para sua aplicação em outros nós sensores mais robustos. O LMT-MAC se mostrou um protocolo previsível, apresentando latência média dentro dos limites calculados analiticamente, e obtendo taxas de vazão e entrega superiores ao TreeMAC.

Como trabalhos futuros, pretende-se desenvolver um mecanismo adaptativo para uso dos *slots*, permitindo reduzir a latência, e implementar o protocolo modificado em uma rede real composta por mais nós.

6. Agradecimentos

Os autores agradecem o apoio financeiro das agências CAPES, CNPq e FAPEMIG.

Referências

Abdeli, D., Zelit, S., and Moussaoui, S. (2013). RTH-MAC: A real time hybrid MAC protocol for WSN. In *Programming and Systems (ISPS), 2013 11th International Symposium on*, pages 153–162.

Akyildiz, I., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002). Wireless Sensor Networks: A Survey. *Computer Networks*, 38(4):393 – 422.

Arduino (2015). Arduino. `http://arduino.cc/`. Acessado em 01-06-2015.

Gungor, V. and Hancke, G. (2009). Industrial wireless sensor networks: Challenges, design principles, and technical approaches. *Industrial Electronics, IEEE Transactions on*, 56(10):4258–4265.

HART (2015). Hart Communication Foundation. `http://www.hartcomm.org/`. Acessado em 01-06-2015.

Incel, O. (2011). A survey on multi-channel communication in wireless sensor networks. *Computer Networks*, 55(13):3081–3099. cited By 28.

Incel, O. D., van Hoesel, L., Jansen, P., and Havinga, P. (2011). MC-LMAC: A multi-channel MAC protocol for wireless sensor networks. *Ad Hoc Networks*, 9(1):73 – 94.

Instruments, T. (2006). CC2420: 2.4 GHz IEEE 802.15. 4/ZigBee-ready RF Transceiver. *Available at Available at http://www. ti. com/lit/gpn/cc2420*, page 53.

Nordic_Semiconductor (2013). *nRF24l01 Product Specification v2.0*.

Romer, K. and Mattern, F. (2004). The design space of wireless sensor networks. *Wireless Communications, IEEE*, 11(6):54–61.

Shen, W., Zhang, T., Barac, F., and Gidlund, M. (2014). PriorityMAC: A Priority-Enhanced MAC Protocol for Critical Traffic in Industrial Wireless Sensor and Actuator Networks. *Industrial Informatics, IEEE Transactions on*, 10(1):824–835.

Song, W.-Z., Huang, R., Shirazi, B., and LaHusen, R. (2009). TreeMAC: Localized TDMA MAC protocol for real-time high-data-rate sensor networks. *Pervasive and Mobile Computing*, 5(6):750 – 765. PerCom 2009.