LiTE: Um Algoritmo de Localização Temporal e Ordenação de Eventos em Redes de Sensores Sem Fio Compostas por Nós Dessincronizados

Leonardo L. Guimarães[1], Horácio A.B.F. Oliveira[1], Rômulo T. Rodrigues[2], Edjair S. Mota[1], Antonio A.F. Loureiro[3]

[1]Depto. de Ciência da Computação – Universidade Federal do Amazonas, Brasil
[2]Depto. de Engenharia Electrotécnica e Computadores – Universidade do Porto, Portugal
[3]Depto. de Ciência da Computação – Universidade Federal de Minas Gerais, Brasil

{horacio,leonardo,edjair}@dcc.ufam.edu.br
ee09264@fe.up.pt, loureiro@dcc.ufmg.br

Abstract. *Wireless Sensor Networks are basically designed to monitor and detect events of interest. Two key aspects of this task are the identification of the exact time of occurrence of an event and, mainly, the ordering of several events in the network. Current solutions propose different algorithms for clock synchronization for sensor nodes. However, these solutions require constant executions to keep the network synchronized, since the sensor clocks quickly get unsynchronized (up to 3 seconds per day). In this work, we propose the LiTE algorithm, a novel, simple, and efficient algorithm for temporal localization and ordering of events in these networks. The proposed algorithm does not require clock synchronization of the sensor nodes. Laboratory experiments with real sensor nodes prove the applicability of the proposed algorithm and extensive simulation experiments show the scalability and efficiency of the proposed solution.*

Resumo. *Redes de Sensores Sem Fio são redes basicamente projetadas para monitorar e detectar eventos de interesse. Dois aspectos chaves desta tarefa são a identificação exata do tempo de ocorrência de um evento e, principalmente, a ordenação e o sequenciamento da ocorrência de diversos eventos na rede. Soluções atuais propõem diferentes algoritmos de sincronização de relógios dos nós sensores. Entretanto, tais soluções requerem constantes execuções para manter a rede sincronizada, já que os relógios dos sensores rapidamente se dessincronizam (até 3 segundos por dia). Este trabalho propõe o algoritmo LiTE, uma nova abordagem, simples e eficiente, para localização temporal e ordenação de eventos em tais redes, que não requer sincronização dos nós sensores. Experimentos práticos em laboratório com nós sensores reais comprovam a aplicabilidade do modelo e simulações extensivas mostram a escalabilidade e robustez da solução proposta.*

1. Introdução

Redes de Sensores Sem Fio (RSSFs) são compostas por nós sensores que cooperam entre si a fim de monitorar uma área de interesse comum [Akyildiz et al. 2002, Estrin et al. 2001, Loureiro et al. 2003]. Esta tecnologia pode ser empregada nas mais diversas situações: monitoração de ambientes inóspitos, instalações médicas, urbanas,

militares, industriais, etc. À medida que ocorrem avanços tecnológicos nas áreas de sensores, nanotecnologia, circuitos integrados e comunicação sem fio, a utilização das RSSFs nas mais diversas aplicações se torna uma possibilidade revolucionária, por se tratar de uma ferramenta de coleta e processamento de informação, que tende a ser escalável e de baixo custo.

Tais RSSFs são voltadas basicamente à detecção e monitoração de eventos. Eventos possuem duas características principais: a primeira é qualitativa (*critério causal*), diz respeito à variável monitorada (e.g., temperatura, luminosidade, som, pressão); a segunda característica é referente à localização, tanto espacial quanto temporal (*critério espaço-temporal*), e indica quando e onde um evento ocorreu [Oliveira et al. 2009, Davidson 1980]. Enquanto que o primeiro critério é facilmente identificado usando o dispositivo sensorial que o detectou, o critério espaço-temporal só pode ser identificado usando-se o posicionamento dos nós sensores e também, em geral, os seus relógios. Considerando que os nós sensores de uma RSSF são basicamente estáticos, o critério espacial, uma vez identificado, permanece o mesmo ao longo do tempo.

Entretanto, manter os relógios dos nós sensores sincronizados é um desafio muito grande uma vez que estes se dessincronizam a uma taxa de $40\,\mu s/s$ [Maroti et al. 2004]. A essa taxa de dessincronização, os nós sensores precisarão ser sincronizados a cada $25\,s$ para manter uma sincronização na faixa dos milissegundos. Tendo em vista esta problemática, diversos trabalhos propõem algoritmos de sincronização leves e passíveis de serem executados continuamente, dentre os quais podemos citar: *Reference Broadcast Synchronization* [Elson et al. 2002], *Flooding Time Synchronization Protocol* [Maroti et al. 2004], *Delay Measurement Time Synchronization* [Ping 2003] e *Post-Facto Synchronization* [Elson and Estrin 2001].

Neste trabalho, uma nova abordagem, simples e eficiente, para localização temporal e ordenação de eventos em RSSFs está sendo proposta. Nesta abordagem, implementada no algoritmo LiTE (Localização Temporal de Eventos), não se procura sincronizar os nós sensores entre si, mas sim sincronizar o evento com o relógio do nó sink, responsável por coletar e agregar todos os eventos da rede. Tal algoritmo se baseia no cálculo preciso dos atrasos dos pacotes enviados em múltiplos saltos a partir do nó que detectou o evento até o nó sink. Experimentos reais conduzidos em laboratório através de osciloscópios e nós sensores atestam a possibilidade do cálculo destes atrasos enquanto que simulações extensivas usando o simulador NS-2 demonstram a escalabilidade, a eficiência e a robustez da solução proposta.

O restante deste trabalho está organizado como segue. Soluções atuais encontradas na literatura são discutidas na seção 2. Na seção 3, são introduzidos alguns conceitos relevantes à compreensão deste trabalho. Em seguida, na seção 4, é apresentado o algoritmo proposto, o LiTE. Na seção 5, são feitas avaliações de aplicabilidade e de desempenho. Na seção 7, alguns aspectos sobre a aplicabilidade e possível substituição dos algoritmos atuais pelo proposto neste trabalho são discutidos. Por último, na seção 6, são apresentadas conclusões relativas aos resultados obtidos e possíveis aspectos que deverão ser tratados em trabalhos futuros.

2. Trabalhos Relacionados

O *Reference Broadcast Protocol* (RBS) [Elson et al. 2002] é um protocolo de sincronização que utiliza um *broadcast* de referência, o qual é originado a partir de nós especiais (*beacons*) que possuem o tempo de "referência". Os nós *beacons* realizam o *broadcast* de referência e, em seguida, seus nós vizinhos fazem um *broadcast* informando o tempo de recebimento deste pacote, possibilitando a criação de uma tabela com os atrasos (*offsets*) relativos à cada vizinho. Este algoritmo apresenta a vantagem de eliminar muitas fontes de erro no processo de sincronização. Entretanto, seu custo computacional é elevado se comparado com as demais soluções – $O(2 * n)$, onde n é a quantidade de nós na rede.

No protocolo *Flooding Time Synchronization Protocol* (FTSP) [Maroti et al. 2004], o nó sink (que possui o tempo de referência) faz um *broadcast* dando inicio ao *flooding* na rede. Os demais nós, ao receberem esse pacote, fazem um *timestamp* na camada MAC (*Media Access Control*), calculam os atrasos do tempo de transmissão em relação ao sink, e repassam o pacote com as devidas correções, dando continuidade ao *flooding*. Ao final, todos os nós alcançáveis da rede terão realizado um *broadcast* e toda a rede estará sincronizada com uma determinada precisão. O custo de comunicação do FTSP é de um pacote enviado por cada nó da rede – $O(n)$.

O *Delay Measurement Time Synchronization* (DMTS) [Ping 2003] pode ser utilizado para sincronização local (assim como o RBS), ou global (como o FTSP). A sincronização local ocorre como segue. Em uma determinada região é eleito um nó líder (referência), o qual faz um *broadcast* com o seu tempo. Ao contrário do RBS, os vizinhos não trocam pacotes entre si. Eles se sincronizam com o tempo do líder, calculando o *atraso de transmissão do pacote* (detalhado na seção 3). A sincronização por múltiplos saltos funciona da mesma forma, porém após cada vizinho se sincronizar, ele deve realizar um *broadcast* contendo o seu tempo sincronizado, ou seja, funciona como um algoritmo de inundação (*flooding*) com custo $O(n)$. Existe uma forte semelhança entre o DMTS e o FTSP, mas o que os difere é basicamente a forma de calcular o *atraso*.

O algoritmo *Post-Facto Synchronization* [Elson and Estrin 2001] é um algoritmo de sincronização instantânea voltada especificamente para eventos. Assim como o RBS, é necessário que hajam nós de referência espalhados ao longo da rede. No entanto, esses nós *beacons* só realizam o *broadcast* de referência caso algum vizinho detecte um evento e solicite o tempo real [Elson and Estrin 2001]. Desta forma há uma ordenação precisa dos eventos, porém se vários eventos são detectados praticamente todo o tempo, várias solicitações de sincronização serão realizadas.

Como pode-se observar, existem diferentes propostas buscando soluções cada vez mais eficientes na área de sincronização. Embora existam diversas vantagens na utilização de algoritmos de sincronização, é importante destacar que isto implica em consumo extra de energia da rede, que é escassa em RSSFs. Considerando que em nós sensores Mica2 há uma taxa de dessincronização de $40\,\mu s/s$ [Maroti et al. 2004], isso acarretará em uma dessincronização de aproximadamente $3,5s$ por dia e a rede precisará ser ressincronizada frequentemente, o que resulta em mais consumo de energia. Neste trabalho, uma nova abordagem está sendo proposta: não se preocupar com a sincronização dos nós sensores, mas sim dos eventos. Nesta abordagem, nenhum pacote será trocado para sincronizar nós com relação a alguma referência. Além disso, a sincronização do evento é realizada com

o próprio pacote que o nó sensor envia ao sink informando a respeito da ocorrência do evento, o que gera uma solução com custo de comunicação praticamente nulo.

3. Definição do Problema

Definição 1 (Rede de Sensores Sem Fio) *Uma RSSF pode ser representada formalmente como um grafo Euclidiano $G = (V, E)$ como segue:*

- $V = \{v_0, v_1, \ldots, v_{n-1}\}$ *é o conjunto de nós sensores (vértices do grafo), sendo que v_0 é o nó sink;*
- $\forall v_i \in V$, r *é o raio de comunicação de v_i;*
- $Q = [0, x] \times [0, y] \times [0, z]$ *a região de sensoriamento em três dimensões;*
- $\langle i, j \rangle \in E$ *se, e somente se, a distância entre v_i e v_j for menor que r, i.e., v_i alcança v_j e vice-versa;*
- t *é o tempo global da rede; pode ser baseado no UTC (Coordinated Universal Time, e.g., GPS) ou em um tempo relativo (e.g., do nó sink).*
- $\forall v_i \in V$, $t_i(t)$ *é o tempo local em que v_i se encontra no instante t.*

Conforme mencionado, redes de sensores são basicamente voltadas à monitoração, detecção e notificação de eventos.

Definição 2 (Eventos e Histórico Local e Global de Eventos) *Um evento pode ser definido como "algo que acontece em um dado lugar e tempo" ou "um fenômeno localizado em um único ponto do espaço-tempo" [Fellbaum 1998]. Do ponto de vista temporal em RSSFs, um evento pode ser detectado por um ou mais nós e pode ser definido como:*

- e_i^t *é o evento e sendo detectado pelo nó v_i no seu tempo local t_i;*
- $h_i = \{e_i^{t_1}, e_i^{t_2}, \ldots\}$ *é o histórico ordenado de eventos do nó v_i;*
- $\{h_1 \cup h_2 \ldots \cup h_n\} \rightsquigarrow H$ *é o histórico global de eventos ordenado pelo tempo global t;*

Em RSSFs, o *histórico local* pode ser facilmente calculado ordenando-se os eventos detectados usando os relógios locais dos nós sensores. Entretanto, para que esta informação seja útil, ela precisa ser convertida em um *histórico global* que é o histórico de todos os eventos da rede que, neste trabalho, é a definição de *ordenação de eventos* (figura 1):

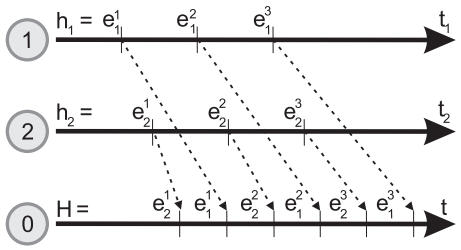

Figura 1. Ordenação de eventos.

Definição 3 (Ordenação de Eventos) *Conversão de $\{h_1 \cup h_2 \ldots \cup h_n\}$ em H.*

Uma solução para este problema de ordenação de eventos é manter todos os nós da RSSF sincronizados com o tempo global t:

Definição 4 (Sincronização de Nós e Erro de Sincronização) $\forall v_i \in V$ *atualizar $t_i(t) \approx t$. Diga-se aproximado, pois nenhum algoritmo de sincronização é perfeito por se basearem em técnicas que geram erros. O erro de sincronização de um nó i é definido como $t_i(t) - t$.*

Quando os nós sensores estão com seus relógios sincronizados, H é facilmente gerado ordenando-se os eventos por seus tempos locais t_i (que são aproximações do tempo global t). Em RSSFs, por seus protocolos serem baseados em múltiplos saltos, uma das técnicas comumente utilizadas em sincronização é a estimativa de atraso de um pacote em um salto (figura 2).

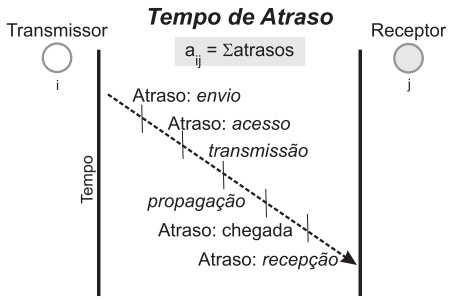

Figura 2. Atraso de um salto.

Definição 5 (Estimativa de Atraso de um Salto – a_{ij}^p)
Estimativa de atraso (delay measurement [Ping 2003, Oliveira et al. 2009]) consiste em calcular, ou estimar, todos os possíveis atrasos existentes durante a transferência de um pacote p, em um único salto, do nó transmissor i para o nó receptor j: $a_{ij}^p = a_{env}^p + a_{mac}^p + a_{trans}^p + a_{prop}^p + a_{cheg}^p + a_{recep}^p$, *onde:*

- a_{env}^p *é o atraso de envio*; onde ocorre a montagem da mensagem, e cabeçalho. Este atraso é variável e não determinístico pois o processo de envio concorre com outros processos e é dependente do sistema operacional;
- a_{mac}^p é o *atraso da camada MAC*; está diretamente relacionado ao estado do canal, ou seja, neste momento o nó está disputando com os demais sensores um momento para enviar seu pacote;
- a_{trans}^p *é o atraso de transmissão*; é determinístico, pois está relacionado ao tempo decorrido durante a transmissão bit a bit do pacote, dependendo principalmente do tamanho do pacote;
- a_{prop}^p *é o atraso de propagação*; dado que a velocidade de propagação de uma onda eletromagnética é de aproximadamente $3 \times 10^8\, m/s$, basta relacionar essa velocidade com o espaço percorrido;
- a_{cheg}^p *é o atraso de chegada*; tempo de recebimento do pacote completo; determinístico e depende do tamanho do pacote;
- a_{recep}^p *é o atraso de recepção*; tempo decorrido durante a montagem do pacote e interrupção do sistema operacional; este tempo varia dependendo do SO, portanto é um tempo não determinístico.

4. LiTE - Localização Temporal de Eventos

Nesta seção, é apresentado um novo algoritmo para sincronização e ordenação de eventos em RSSFs: o LiTE (Localização Temporal de Eventos).Conforme mencionado, o ponto chave do algoritmo LiTE está em reconhecer que manter nós sensores sincronizados gera um custo elevado para apenas determinar o tempo e ordem de ocorrência de eventos em uma RSSF. Desta forma, o algoritmo LiTE procura apenas sincronizar o tempo em que o evento foi detectado pelo nó sensor em relação ao tempo do nó sink. Para isso, o algoritmo consiste em calcular o *atraso de roteamento do pacote* (figura 3).

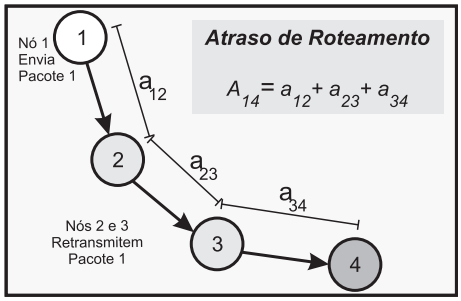

Figura 3. Atraso de roteamento.

Algoritmo 1 - LiTE

▷ **Entrada:**
1: Nó sensor v_i detecta o evento $e_i^{t_i}$.
Ação:
2: $tmEv_i \leftarrow t_i$; {Salva o tempo de detecção do evento}
3: $proxSalto_i \leftarrow calculaProxSalto()$; {Calcula o próximo salto em direção ao sink}
4: $A_i \leftarrow t_i - tmEv_i$; {Atualiza o atraso de roteamento}
5: Envia $pacote(e_i^{t_l}, A_i)$ para $proxSalto_i$. {Envia consulta para a rede}

▷ **Entrada:**
6: $msg_i = pacote(e_k^{t_k}, A_k)$ tal que $a_i = calculaAtrasoSalto(msg_i)$.
Ação:
7: **se** $i \neq 0$ **então** -{SE: este nó não for o sink ...}-
8: $tmPac_i \leftarrow t_i$; {Salva o tempo de chegada do pacote}
9: $proxSalto_i \leftarrow calculaProxSalto()$; {Calcula o próximo salto da resposta}
10: $A_i \leftarrow A_k + a_i + (t_i - tmPac_i)$; {Atualiza o atraso de roteamento}
11: Envia $pacote(e_i^{t_l}, A_i)$ para $proxSalto_i$. {Envia consulta para a rede}
12: **senão**
13: $A_i \leftarrow A_k + a_i$; {Atualiza o atraso de roteamento}
14: $t_k = t_i - A_i$; {Calcula o tempo global do evento}
15: $H \leftarrow H \cup \{e_k^{t_k}\}$; {Registra o evento no histórico global}
16: **fim se**

Definição 6 (Atraso de Roteamento do Pacote – A_{ij}^p) *Tempo total que o pacote p levou para deixar o nó v_i e chegar, em múltiplos saltos, ao nó v_j (em geral, o nó sink). Esse cálculo é possível somando-se todos os atrasos e todos os tempos de processamento dos nós intermediários (i.e., que repassaram o pacote) até o momento que este atingiu o nó de destino, conforme ilustrado na figura 3. Logo, $A_{ij}^p = a_{ik}^p + \ldots + a_{lj}^p$, onde $\{v_k, \ldots, v_l\}$ são nós intermediários que repassaram o pacote.*

O atraso de roteamento pode ser calculado usando qualquer protocolo de roteamento, uma vez que qualquer atraso introduzido pelo roteamento será calculado nesta fase do algoritmo LiTE. É importante ressaltar que tanto o processo de cálculo de atraso dos saltos quanto o de roteamento não requerem sincronização de relógios entre os nós sensores.

O algoritmo 1 descreve o funcionamento do algoritmo LiTE proposto neste trabalho. O algoritmo é simples e eficiente, e não requer nenhuma configuração inicial (i.e., troca de mensagens para sua configuração). Neste algoritmo, quando um determinado nó sensor detecta um evento (linha 1 do algoritmo 1), este irá registrar o tempo local de detecção do evento (linha 2) e, em seguida, irá solicitar ao protocolo de roteamento o próximo salto do pacote (linha 3). Como este último passo pode demorar dependendo do protocolo de roteamento (reativo, pró-ativo, híbrido), o atraso de roteamento é atualizado com este tempo de processamento local (linha 4). Em seguida, o nó encaminhará o pacote com o registro do evento para o próximo nó sensor em direção ao sink (linha 5).

Cada nó sensor intermediário irá calcular o atraso do salto ao receber o pacote (linha 6), registrar o tempo de recebimento do pacote (linha 8) e calcular o próximo salto do pacote (linha 9). Em seguida, o nó intermediário irá acrescentar ao atraso de roteamento do pacote o seu atraso de salto mais o seu tempo de processamento (linha 10) e, por último, encaminhar o pacote para o próximo salto (linha 11). Quando o pacote chegar

ao nó sink, este irá também calcular o atraso do salto e acrescentá-lo ao atraso de roteamento (linha 13). Por último, o nó sink irá calcular o tempo real de ocorrência do evento como sendo o tempo atual subtraído do atraso do pacote e, então, registrar o evento em sua ordem correta no histórico global de eventos.

A implementação do cálculo de atraso de um salto pode ser realizada apenas na camada de aplicação ou pode tirar proveito de informações de tempo introduzidas na camada MAC. Para isso, duas variações do LiTE foram implementadas: o *LiTE Apl*, implementado apenas na camada de aplicação (conforme o algoritmo apresentado), e o *LiTE Mac*, implementado introduzindo-se marcações de tempo na camada MAC.

No *LiTE Mac*, um código é introduzido logo após o nó sensor obter acesso livre ao meio e logo antes do pacote ir para o *driver* de rede para ser enviado (antes da camada física). Esse código obtém o atraso da camada de aplicação até o momento de acesso livre ao meio e adiciona esse atraso ao atraso de roteamento. Outro código no nó receptor é responsável por armazenar uma marcação de tempo no instante em que o *driver* de rede receber o pacote. Essa marcação de tempo, junto com o tempo de recebimento na aplicação, será adicionado ao atraso de roteamento. Nesta versão do LiTE, grande parte dos tempos não determinísticos de envio e recepção de pacotes podem ser eliminados, gerando um cálculo de atraso mais preciso.

5. Avaliação de Desempenho

Nesta seção, o desempenho do algoritmo LiTE será avaliado sob três aspectos: aplicabilidade, escalabilidade e robustez. O primeiro aspecto, experimentado em nós sensores reais, avalia a técnica de cálculo de atraso de um salto e é apresentado na seção a seguir.

5.1. Experimentos em Nós Sensores

O objetivo destes experimentos é analisar o impacto dos erros não determinísticos na técnica de cálculo de atrasos quando implementada em nós sensores reais, mais especificamente, nos nós sensores SunSPOT [SunLabs 2009]. Apesar de experimentos similares terem sido aplicados em outros trabalhos [Maroti et al. 2004], este é o primeiro a experimentar tal técnica em sensores SunSPOT e o primeiro a comparar dados obtidos tanto na camada MAC quanto na de Aplicação. Além disso, como será mostrado a seguir, resultados diferentes foram obtidos por tal técnica quando implementada nestes nós sensores.

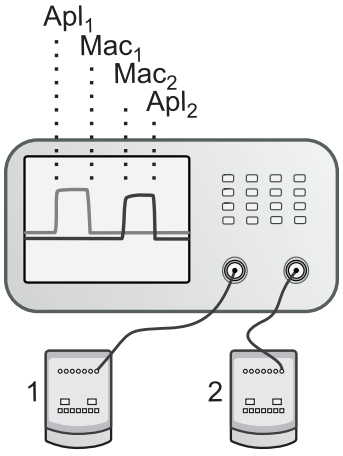

Figura 4. SunSPOTs.

5.1.1. Metodologia

Para calcular o tempo de envio e recepção de um pacote e, mais importante, calcular a variação deste tempo (tempos não determinísticos), um relógio externo, com tempo global, foi necessário. Para isso, dois nós sensores, um transmissor e um receptor, foram conectados a um osciloscópio de alta precisão (MS06032A, *Agilent Technologies*, com precisão de 25 μs – figura 4). Pacotes iguais com tamanho de 52 bytes são então enviados pelo transmissor. Na camada de aplicação antes de solicitar o envio do pacote, o sinal da

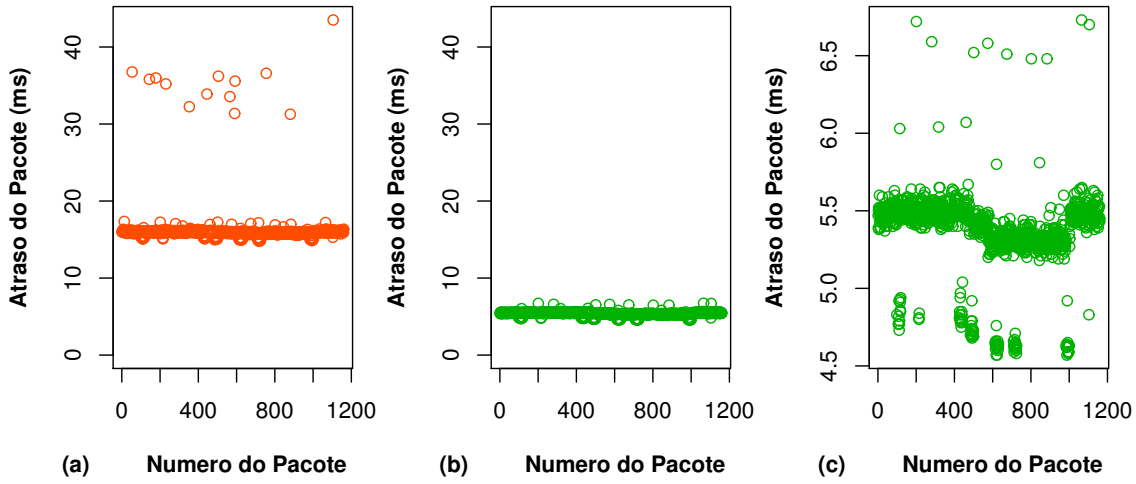

Figura 5. Atrasos dos pacotes na camada de aplicação (a); e MAC (b,c).

saída digital $D0$ sobe para nível lógico 1 (marcação Apl_1, da figura 4), permanecendo assim até que todas as verificações da disponibilidade do meio sejam feitas e, finalmente, o pacote esteja pronto para ser enviado, tornando ao nível lógico inicial 0 (marcação Mac_1). Após a chegada do pacote no receptor (atraso de chegada), o nível lógico da saída digital D0 deste nó sobe para 1 (marcação Mac_2) permanecendo assim até que, na aplicação, após a finalização do processo de recebimento do pacote, o nível volte ao seu valor inicial 0 (marcação Apl_2). Para cada pacote enviado e recebido, pode-se calcular o tempo de um salto tanto na camada de aplicação ($atrasoApl = Apl_2 - Apl_1$) quanto de acesso ao meio ($atrasoMac = Mac_2 - Mac_1$).

É importante notar que, neste experimento, não se procura calcular todos os atrasos do pacote, mas sim identificar atrasos não determinísticos, ou seja, os que variam inesperadamente de um pacote para outro, uma vez que tais atrasos não determinísticos são os responsáveis pela imprecisão da técnica.

5.1.2. Análise dos Resultados

Os gráficos da figura 5 ilustram os atrasos obtidos por diversos pacotes em um salto. Como pode-se observar na figura 5(a), mesmo sem concorrência de acesso ao meio, ainda assim alguns pacotes obtiveram atrasos bem maiores do que a média, indicando uma variação grande dos atrasos quando esta técnica é implementada na camada de aplicação. As figuras 5(a) e (b) ilustram os atrasos obtidos na camada MAC, sendo que esta última com uma visão mais detalhada. Como pode ser observado, tais atrasos variam cerca de $1\,ms$, principalmente abaixo da média.

Nas figuras 6(a,b), são mostrados os histogramas de densidade dos atrasos na camada de aplicação e MAC, respectivamente. Observando os gráficos, pode-se notar que, nestes sensores SunSPOT, os atrasos não seguem uma distribuição Gaussiana, conforme considerado por grande parte dos trabalhos que simulam algoritmos de sincronização. Isso é uma observação muito importante, pois a utilização de modelos errados pode gerar conclusões incorretas a respeito da eficiência dos algoritmos propostos. Para confirmar tal observação, os gráficos quantil-quantil da figura 7 comparam os quantis amostrais com os teóricos, indicando mais uma vez a não-normalidade dos dados pois vários pontos não estão próximos à reta de mínimos quadrados plotada.

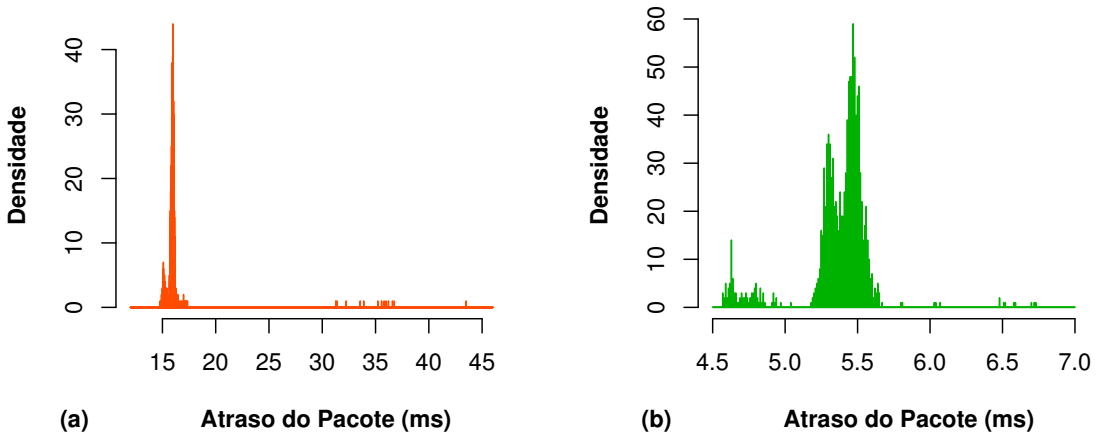

Figura 6. Histograma de densidade dos pacotes: aplicação (a); e MAC (b).

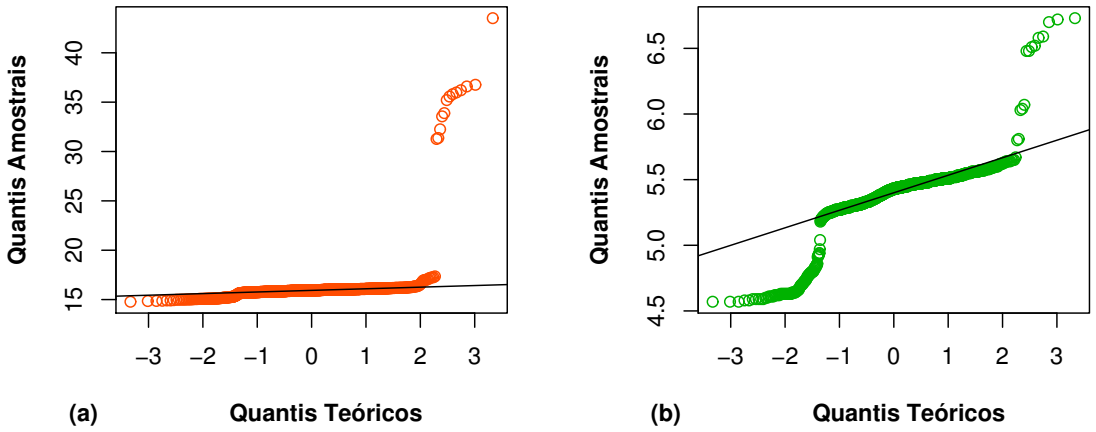

Figura 7. Quantis amostrais e teóricos dos atrasos: aplicação (a); e MAC (b).

Os gráficos Q-Q, apesar de serem bastante poderosos para verificar desvios de normalidade, não constituem um teste formal, servindo apenas para uma análise exploratória dos dados. Testes de adequação formais, tais como o *Chi-quadrado* e *Kolmogorov-Smirnov*, permitem uma análise mais profunda da questão. Desta forma, tais testes foram aplicados para um nível de confiança de 95% nos dados obtidos na camada de aplicação e na camada MAC e indicaram valor de prova $p\text{-}value < 0.01$ para ambos os testes. Esse $p\text{-}value$ é a medida do grau de concordância entre os dados e a hipótese nula H0 (no caso, que a distribuição de probabilidade dos dados é normal). Quanto menor o $p\text{-}value$, mais forte é a evidência contra H0. Uma regra prática de decisão é rejeitar a hipótese nula se $p\text{-}value \leq \alpha$, onde α é a taxa de erro. Como está-se procurando uma margem de confiança de 95%, então, $\alpha = 1 - 0.95 = 0.05$. Logo, com base nos testes aplicados com os dados coletados nas medições, deve-se rejeitar a hipótese de normalidade.

Do ponto de vista de aplicabilidade, pode-se observar pelos gráficos mostrados que os atrasos não determinísticos geram um erro de aproximadamente $1\,ms$ ao ser utilizar a camada MAC e um erro de aproximadamente $5\,ms$ na camada de aplicação. Uma vez que observa-se pouca variação nos atrasos, pode-se concluir que a técnica é passível de ser implementada em nós sensores reais e, mais especificamente, nos nós sensores SunSPOT.

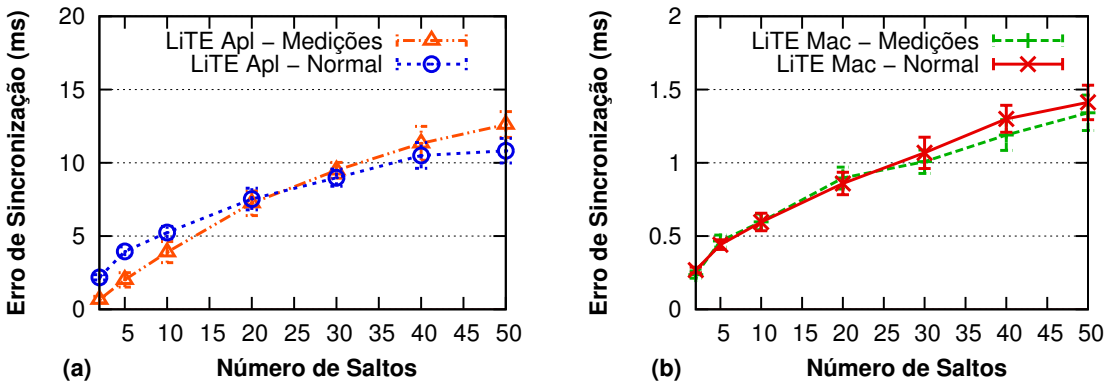

Figura 8. Erro de sincronização por número de saltos: aplicação (a); e MAC (b).

5.2. Experimentos de Escalabilidade e Robustez

O objetivo destes experimentos é avaliar o comportamento do algoritmo quando executado em múltiplos saltos em uma RSSF.

5.2.1. Metodologia

A avaliação de desempenho foi realizada utilizando o *Network Simulator 2* [McCanne and Floyd 2005]. Em todos os resultados, as curvas representam uma média das execuções, enquanto que as barras de erro, o intervalo de confiança para 95% de confiança a partir de 33 execuções diferentes (sementes aleatórias).

A tabela 1 apresenta valores padrões para os parâmetros de simulação. Os nós sensores são distribuídos no campo de monitoramento de acordo com uma grade perturbada, i.e., os nós tendem a ocupar a área uniformemente, mas sem formar uma grade regular. Para simular os erros de cálculo de atraso de um salto, foram utilizadas *simulações baseadas em medições* [Kashyap et al. 2008]. Nestas simulações, medições reais obtidas experimentalmente (neste caso, os atrasos calculados na seção anterior) são alimentadas ao simulador que irá utilizá-los quando necessário. Nesta abordagem não se tem os erros estatísticos observados ao se utilizar um modelo probabilístico.

Parâmetro	*Valor*
Campo de sensores	758 m × 758 m
Numero de nós	576 nós
Densidade	0.001 nós/m^2
Raio de comunicação	50 m
Atraso de um pacote	Medições
Erro de atraso	Medições

Tabela 1. Valores

5.2.2. Análise dos Resultados

Em termos de escalabilidade, o principal fator que afeta o algoritmo LiTE é a quantidade de saltos que o pacote percorre saindo do nó sensor que detecta o evento até o nó sink. Os gráficos da figura 8 mostram este impacto que a quantidade de saltos que o pacote percorre tem sobre o erro de cálculo de atraso e, consequentemente, na sincronização do evento. Como pode-se observar, os erros obtidos quando o algoritmo LiTE é implementado na camada de aplicação (figura 8(a)) são maiores e crescem mais rapidamente com o aumento do número de saltos do que quando implementado na camada de acesso ao

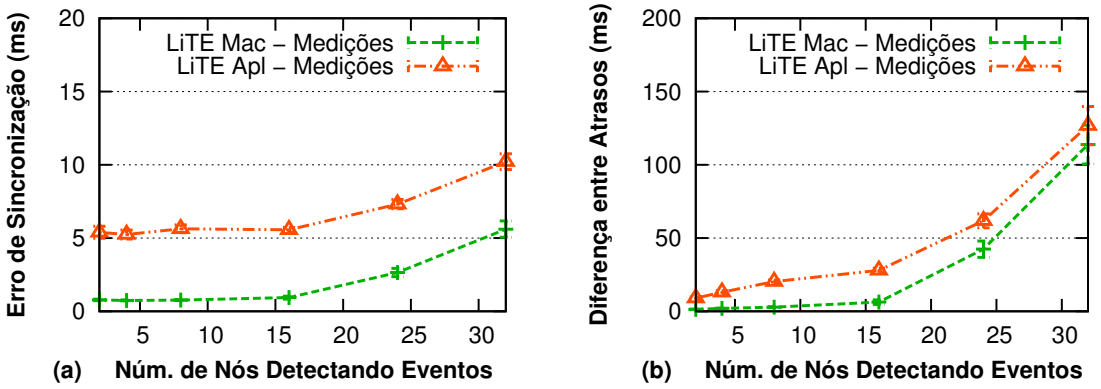

Figura 9. (a) Erro e (b) diferença de sincronização de eventos simultâneos.

meio (figura 8(b)). Uma outra observação importante é o erro de apenas 1.5 ms após 50 saltos quando o LiTE é implementado na camada MAC. Isso se deve ao fato do erro de cálculo de atraso de um salto poder ser anulado pelo erro de cálculo de atraso do salto seguinte. Ainda nesses gráficos, estamos comparando os resultados obtidos experimentalmente com a simulação dos erros usando uma distribuição normal. Pode-se observar uma certa diferença entre os resultados, em especial quando nos dados obtidos pela camada de aplicação. Nos gráficos seguintes, apenas as simulações baseadas em medições serão apresentadas.

Nos gráficos da figura 9, está-se avaliando a robustez do algoritmo para sincronizar eventos quando diversos eventos são detectados na rede. Para isso, diversos nós na rede, escolhidos aleatoriamente, detectaram um evento exatamente no mesmo instante. O gráfico da figura 9(a) mostra o comportamento do erro de sincronização dos eventos quando estes chegam no nó sink, enquanto que o gráfico da figura 9(b) mostra a diferença entre o menor e o maior tempo estimado do evento. Pode-se observar que em ambos os casos, o erro de sincronização dos eventos começa a crescer quando muitos eventos são detectados ao mesmo tempo, devido a atrasos maiores no envio e encaminhamento dos pacotes.

Foi avaliado também a capacidade do algoritmo LiTE de ordenar eventos na rede. Para isso, nos gráficos das figuras 10(a,b), 10 eventos foram gerados em ordem e em intervalos de tempo iguais (eixo X). Tais eventos foram então sincronizados no sink e ordenados usando o *LiTE Apl*, *LiTE Mac* e também usando a ordem de chegada dos pacotes no sink como a ordem dos eventos. Dois cenários foram avaliados. No primeiro cenário (figura 10(a)), os eventos estão próximos um do outro (e.g., um som alto sendo detectado por diversos sensores) e, no segundo cenário (figura 10(b)) os eventos estão espalhados aleatoriamente na rede (e.g., animais se movimentado em diversas partes). Como pode ser observado nos dois gráficos, o algoritmo *LiTE Mac* é capaz de acertar 100% da ordem dos eventos quando o tempo entre estes é maior do que $5\,ms$, mesmo no caso em que os eventos se encontram espalhados pela rede. Pode-se observar ainda, pelos gráficos, que a ordem de chegada dos pacotes no sink não e uma boa fonte de referência, principalmente no segundo cenário.

Por último, foi avaliada a capacidade do algoritmo LiTE de ordenar eventos à medida que a quantidade destes aumenta e mantendo-se o tempo entre eventos fixado

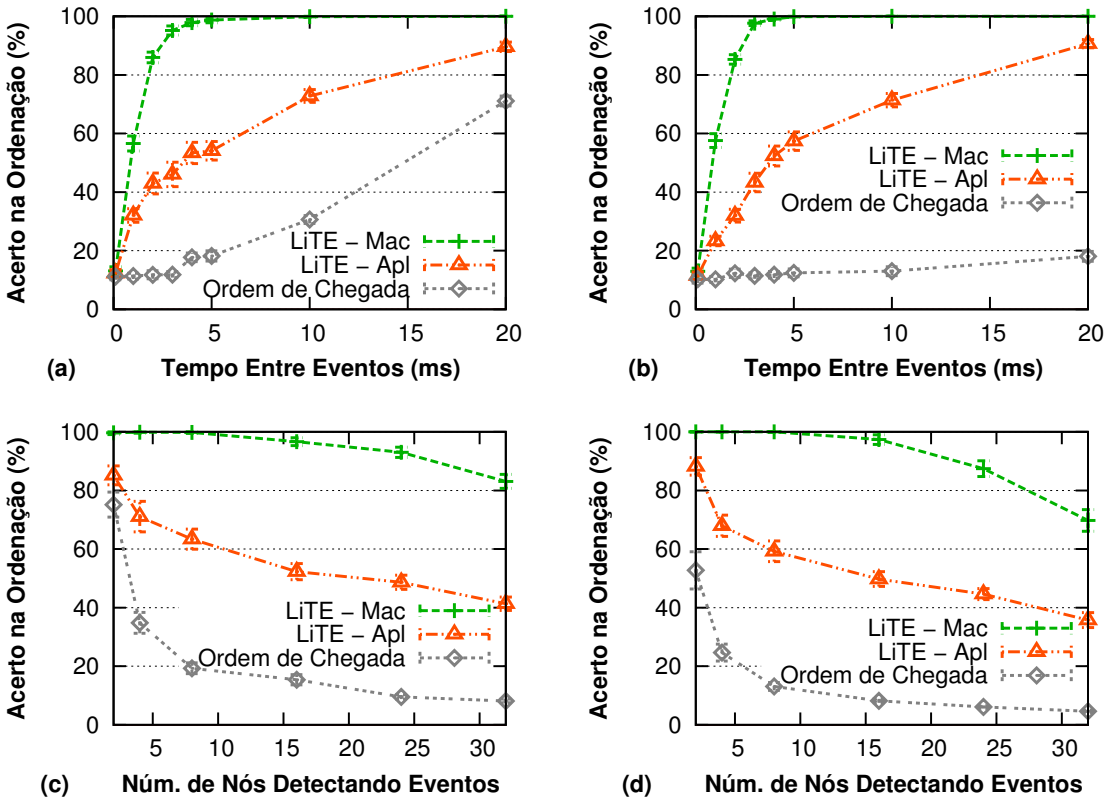

Figura 10. Acerto na ordenação de eventos próximos (a) e aleatórios (b).

em $5\,ms$ (figuras 10(c,d)). Novamente, foi-se avaliado o cenário em que eventos estão próximos (figura 10(c)) e espalhados (figura 10(d)). Pode-se observar que o algoritmo *LiTE Mac* foi capaz de ordenar corretamente mais de 90% de 20 eventos que estavam separados por apenas $5\,ms$ de tempo. Além disso, observa-se um comportamento com queda bem mais lenta da precisão deste último algoritmo em relação ao algoritmo *LiTE Apl* e na ordem de chegada dos pacotes no nó sink.

6. Aplicabilidade e Extensibilidade da Solução Proposta

Os resultados obtidos, em especial pelo algoritmo *LiTE Mac*, indicam que o mesmo é capaz de sincronizar e ordenar diversos eventos separados entre si por apenas $5\,ms$. Nesta precisão, uma aplicação poderia, por exemplo, identificar facilmente a localização de um som que estivesse a uma distância de pouco mais que $1.7\,m$ dos sensores.

A solução proposta no algoritmo *LiTE* pode ser facilmente utilizada nas mais diversas aplicações dos algoritmos de sincronização tradicionais sem a necessidade de modificação:

- *localização temporal de eventos*;
- *ordenação temporal de eventos*;
- *rastreamento de objetos*;
- *localização de sons*;
- *geração de mapas de energia*; etc.

Em outros casos, o algoritmo *LiTE* pode ser facilmente estendido para ser utilizado em algoritmos que precisam de processamento temporal distribuído como:

- *fusão de dados*: para combinar dados relacionados no tempo, cada nó intermediário pode sincronizar os eventos que este for combinar/encaminhar usando a mesma técnica executada pelo nó sink no algoritmo LiTE;
- *localização de sons e rastreamento* in site: nós vizinhos podem trocar pacotes com seus tempos locais e usarem o atraso do pacote para sincronizar seus relógios. Após o processamento distribuído, o dado processado poderá ser sincronizado globalmente no sink.

Além disso, o algoritmo *LiTE* não precisa se preocupar com questões como re-sincronização de nós devido ao *drift*, nós em modo *sleep*, tempo de convergência, complexidade computacional, tolerância a falhas, dentre outros; problemas estes que afetam todos os algoritmos de sincronização (apesar de poucos trabalhos levarem em consideração todos ao mesmo tempo).

7. Conclusão

Este trabalho propôs uma nova abordagem para o problema de sincronização e ordenação de eventos em RSSFs: o algoritmo LiTE – Localização Temporal de Eventos em RSSFs. Foi mostrado que, em muitos cenários, sincronizar todos os relógios da rede não apenas é um processo custoso como também desnecessário. Para solucionar tal problema, o algoritmo LiTE propõe a sincronização apenas dos eventos, e não dos nós sensores.

O desempenho do algoritmo LiTE foi avaliado tanto em experimentos práticos, em laboratório com nós sensores reais, que comprovaram a aplicabilidade do modelo, como também foi avaliado em simulações, mostrando a escalabilidade e robustez da solução proposta. O algoritmo *LiTE Mac* obteve o melhor desempenho nos experimentos realizados, tanto práticos quando simulados, e foi capaz de sincronizar eventos a 16 saltos de distância com erros próximos a apenas $1\,ms$ e foi capaz ainda de ordenar corretamente vários eventos espalhados pela rede e distantes apenas $5\,ms$ no tempo uns do outros. Pode-se dizer que o custo de comunicação do algoritmo é nulo, pois ele não requer troca de pacotes para configuração, aproveitando os pacotes utilizados no roteamento dos dados para o sink para enviar os dados de sincronização.

Os resultados obtidos são promissores. Algumas vantagens e limitações serão exploradas em trabalhos futuros como, por exemplo, identificar claramente quais os procedimentos que geram erros não determinísticos nos sensores e, então, modificá-los para reduzir tais atrasos e adaptá-los a uma rede que requer sincronização de eventos, uma vez que a implementação dos sensores atuais não levam isso em consideração. Pretende-se ainda implementar a técnica proposta em algoritmos de fusão de dados que necessitam de informações temporais.

Referências

Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cyirci, E. (2002). Wireless sensor networks: A survey. *Computer Networks*, 38(4):393–422.

Davidson, D. (1980). *Essays on Actions and Events*. Clarendon, Oxford.

Elson, J. and Estrin, D. (2001). Time synchronization for wireless sensor networks. In *IPDPS'01: Proceedings of the 15th International Parallel & Distributed Processing Symposium*, pages 1965–1970, Washington, DC, USA. IEEE Computer Society.

Elson, J., Girod, L., and Estrin, D. (2002). Fine-grained network time synchronization using reference broadcasts. *SIGOPS Operating Systems Review*, 36(SI):147–163.

Estrin, D., Girod, L., Pottie, G., and Srivastava, M. (2001). Instrumenting the world with wireless sensor networks. In *ICASSP'01: Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 2033–2036, Salt Lake City, Utah.

Fellbaum, C. (1998). *Wordnet: An Electronic Lexical Database*. Bradford Books, 01 edition.

Kashyap, A., Ganguly, S., and Das, S. R. (2008). Measurement-based approaches for accurate simulation of 802.11-based wireless networks. In *MSWiM '08: Proceedings of the 11th international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 54–59, New York, NY, USA. ACM.

Loureiro, A. A. F., Nogueira, J. M. S., Ruiz, L. B., Mini, R. A. F., Nakamura, E. F., and Figueiredo, M. S. (2003). Redes de sensores sem fio. *SBRC'03: Proceedings of the 21st Brazilian Symposium on Computer Networks*, pages 179–226. Belo Horizonte, MG, Brasil.

Maroti, M., Kusy, B., Simon, G., and Ledeczi, A. (2004). The flooding time synchronization protocol. In *SenSys'04: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, pages 39–49, Baltimore, MD, USA. ACM Press.

McCanne, S. and Floyd, S. (2005). ns network simulator. [Online] Available: http://www.isi.edu/nsnam/ns/.

Oliveira, H. A. B. F., Boukerche, A., Nakamura, E. F., and Loureiro, A. A. (2009). Localization in time and space for wireless sensor networks: An efficient and lightweight algorithm. *Performance Evaluation*, 66(3-5):209–222.

Ping, S. (2003). Delay measurement time synchronization for wireless sensor networks. Technical Report IRB-TR-03-013, Intel Research.

SunLabs (2009). Sun small programmable object technology (sun spot) developer's guide. [Online] Available: http://www.sunspotworld.com.