STELE - Uma Técnica Simples Para Estimativa Local do Atraso de Transmissão em RSSF

Cledson Sousa[1], Ricardo C. Carrano[1], Luiz C. S. Magalhães[1], Celio V. N. de Albuquerque[2]

[1]Departamento de Engenharia de Telecomunicações

[2]Instituto de Computação

Laboratório MidiaCom – Universidade Federal Fluminense (UFF)
R. Passo da Pátria, 156, bloco E, sala 408 – 24210-240 – Niterói – RJ - Brasil

{cledson, carrano, schara}@midiacom.uff.br, celio@ic.uff.br

Abstract. *Traditional techniques for delay calculation between neighbor nodes involve the exchange of frames and assume link symmetry. This approach, besides being costly in terms of energy, is not a good fit for wireless networks, where links are typically asymmetric. On the other hand, MAC layer timestamps eliminate the asymmetry problem, but depend on specific hardware. In this paper, we present a Simple Technique for Local delay Estimation (STELE) which eliminates the need for bidirectional communication, and is therefore applicable to asymmetric links. Experimental results with sensors motes prove this technique advantageous for dense networks.*

Resumo. *As técnicas tradicionais para o cálculo do atraso de transmissão entre nós vizinhos implicam troca recíproca de quadros e pressupõem a simetria dos enlaces. Além de energeticamente custosa, essa abordagem não é adequada para redes de sensores sem fio, onde os enlaces são tipicamente assimétricos. Por outro lado, técnicas como a introdução de marcas de tempo pela camada MAC eliminam o problema da assimetria, mas são dependentes de hardware específico. Neste artigo, é apresentada uma técnica simples de estimativa local (STELE) que elimina a necessidade de comunicação bidirecional, sendo, portanto, aplicável a enlaces assimétricos. Resultados experimentais com sensores reais mostram a técnica como vantajosa para redes densas.*

1. Introdução

O cálculo do atraso de transmissão entre dois nós é fundamental para diversos mecanismos utilizados em redes sem fio, sendo o exemplo mais evidente, a sincronização entre dois dispositivos, que por sua vez, é componente essencial em outros tantos mecanismos, desde o *duty cycling* da interface do rádio [Yick et al. 2008] até as técnicas de redução de colisão [Anastasi et al. 2006]. Diversas aplicações para as cidades inteligentes, como a computação ubíqua, demandarão redes de sensores e seus nós precisarão, eventualmente, estar sincronizados.

Um objetivo básico para a sincronização entre dispositivos em redes com acesso ao meio baseado em contenção, como o CSMA, é a estimativa do atraso de transmissão,

portanto, algo a ser alcançado sem negligenciar requisitos importantes em redes de sensores sem fio (RSSF), como a eficiência energética. Além disso, em dispositivos de capacidades limitadas, como os nós deste tipo de rede (*motes)*, é preciso observar algumas restrições de hardware, como a imprecisão dos relógios, os escassos recursos de memória e *CPU* e a baixa potência dos seus rádios.

A assimetria dos enlaces entre dispositivos munidos de transmissores de baixa potência é um fato conhecido e já caracterizado em [Woo, Alec and Culler 2002], sendo ainda mais severa nas redes de sensores. Esse contraste de condições dos rádio enlaces costuma ser descrito em termos das diferenças de qualidade entre os *links* de ida e os de volta, que mesmo quando simétricos do ponto de vista da probabilidade de recepção de quadros, ainda apresentam tempos de ida e de volta distintos. Assim, estimar a latência como metade do RTT, como no NTP (*Network Time Protocol)* [Mills 1985] se torna impreciso e essa imprecisão se dá por diversos fatores, entre eles: a disputa pelo meio; a necessidade de retransmissões; as diferenças nos tempos de processamento e o enfileiramento entre os nós. Em síntese, a diferença entre os tempos de ida e de volta é função das incertezas nos tempos de transmissão em cada uma das etapas do processo de comunicação, desde o momento em que a aplicação de um nó solicita o envio de uma mensagem, até que esta seja finalmente recebida pelo destinatário. Uma breve análise destas incertezas e suas amplitudes podem ser encontradas na Tabela 1. Desse modo, para obter a sincronização, precisamos então caracterizar, eliminar ou compensar tais incertezas. E são diversas as técnicas propostas para este fim. Vejamos algumas delas.

Ao contrário das redes cabeadas, em redes *ad hoc*, as técnicas de sincronização empregadas pelos protocolos NTP e PTP (*Precision Time Protocol*) [Committee 2008], não apresentam bons resultados, por conta dos obstáculos já mencionados. Por esta razão, novos mecanismos foram propostos: como o RBS [Elson et al. 2002], o FTSP [Maróti et al. 2004] e o PulseSync [Lenzen et al. 2009], mas apesar da precisão alcançada, em geral da ordem de dezenas de micros segundos, alguns destes mecanismos ora precisam de $O(n^2)$ trocas de mensagens, onde n é o número de nós, ora se apoiam em recursos que não estão presentes universalmente em todos os nós sensores, como veremos na Seção 2.

Neste artigo, apresentamos um método de estimativa do atraso de transmissão calculado inteiramente no transmissor, que alcança em média uma precisão da ordem de 50μs, no melhor caso, sem dependência de elementos externos ou características especiais de hardware. Se utilizando apenas de duas mensagens para alcançar a sincronização entre dois nós.

Este trabalho está organizado da seguinte forma: a Seção 2 contém os trabalhos relacionados ao problema da sincronização de relógios e de como eles endereçam o problema do indeterminismo do atraso de transmissão. Na Seção 3, o mecanismo **STELE** *(A Simple TEchnique for Local delay Estimation)* é apresentado, bem como a avaliação dos resultados obtidos com ***motes* reais**. Simulações do consumo de energia são descritas na Seção 4 e típicos cenários de uso, considerações finais e futuros desdobramentos da técnica estão dispostos na Seção 5.

2. Trabalhos Relacionados

A sincronização de relógios é um problema de engenharia que lida com o fato dos relógios internos dos diversos dispositivos em uma rede, terem níveis de imprecisões distintos entre si. Mesmo se inicialmente sincronizados dentro de certos limites

aceitáveis, algum tempo depois, devido ao escorregamento de relógio, em geral diferente para cada nó, eles perderão a sincronia uns com os outros. Mais ainda, especificamente em redes de sensores sem fio, os relógios dos nós são pouco precisos quando comparados àqueles dos computadores pessoais. A frequência de seus osciladores chega a variar em até 40PPM (partes por milhão) [Akyildiz and Vuran 2010] e mesmo a uma taxa de 1PPM, esse desvio causaria um erro de aproximadamente 90ms por dia. Considerando tais fatores, múltiplas soluções têm sido criadas para este problema, algumas mais apropriadas que outras em determinados contextos.

Muitos métodos de sincronização têm sido propostos e usados ao longo dos anos, mas virtualmente todos eles compartilham o mesmo modelo: um servidor troca algumas mensagens com o cliente, que por sua vez, calcula o *offset* e ato contínuo sincroniza seu relógio local. Contudo, em redes de sensores sem fio, com poucos recursos de energia, mais sujeitas a interferências, com enlaces assimétricos e mudanças abruptas da relação sinal ruído, certas estratégias que usam tráfego intensivo de quadros se tornam impraticáveis.

Uma das primeiras abordagens a tratar a sincronização em RSSF foi o ***Reference-Broadcast Synchronization*** **(RBS)**. O RBS emprega a difusão de um quadro de sincronismo enviado de um nó de referência. A partir daí, um conjunto de receptores ao alcance deste quadro trocam seus *offsets* através de *unicast*. Com este método o RBS elimina as incertezas durante o processo de transmissão e produz resultados com precisão da ordem de dezenas de micro segundos. Na prática, no entanto, mensagens em difusão podem ser corrompidas. Além disso, os nós receptores nem sempre estarão ociosos e prontos para gravar a referência, pois podem estar ocupados transmitindo ou com sua CPU executando outro processo.

As vantagens desta abordagem são: eliminar as fontes de erro não determinísticas do lado do transmissor e só realizar os ajustes quando necessário, suprimindo o gasto de energia caso o fizesse periodicamente. E as desvantagens são: o excesso de troca de mensagens entre os nós receptores, pois para redes com mais de um salto com n nós este protocolo requer $O(n^2)$ trocas de mensagens e o tempo de convergência, que passa a ser longo devido ao grande número de mensagens.

O ***Flooding Time Synchronization Protocol*** **(FTSP)** também usa difusão para sincronizar múltiplos receptores, mas seu mecanismo utiliza uma mensagem enviada a partir de um nó raiz, em *broadcast*, que contém sua marca de tempo, usando o conceito de *timestamp* na camada MAC [Maróti et al. 2004] na recepção e na transmissão. Além disso, compensa o escorregamento de relógios através de uma regressão linear.

As vantagens desse protocolo, entre outras, são: ser aplicável a redes sujeitas a falhas ou mudanças de topologia, já que possui um mecanismo de eleição de nó raiz com rápida convergência. E as desvantagens são: em redes multissaltos os erros são propagados exponencialmente [Lenzen et al. 2009]; a falta de coordenação na transmissão das mensagens de sincronização, que causa colisão e perda de quadros e seu *timestamp* na camada MAC necessita de *hardware* específico para sua implementação.

No **PulseSync** [Lenzen *et* al. 2009], os autores argumentam que o limite teoricamente já estabelecido do escorregamento do relógio entre um par de nós é $\Omega(D)$, sendo D o diâmetro da rede, está apoiado em assunções que não descrevem o comportamento real da sincronização entre os relógios dos nós em uma RSSF. Dizem

ainda que ambos, o comportamento aleatório tanto da variação do *jitter* quanto do escorregamento dos relógios não varia arbitrariamente. Lenzen *et* al promoveram então uma análise do protocolo FTSP e constataram que este protocolo apresenta um erro de sincronização que cresce exponencialmente com diâmetro da rede. E a proposta do PulseSync é a de corrigir essas deficiências, através de um algoritmo assintoticamente ótimo. Ele funciona propagando um pulso com a referência de tempo, a partir de um nó raiz, o mais rapidamente possível através da rede, coordenando as transmissões dos nós. Assim, os autores afirmam alcançar com alta probabilidade a diferença entre os relógios como $\Omega(\sqrt{D})$. No entanto, apesar desse mecanismo apresentar vantagens em relação aos já mencionados, é ainda dependente do *timestamp* na camada MAC e essa facilidade não está presente universalmente em todos os sensores.

Na Seção 3 nós apresentamos o STELE - uma técnica simples de estimativa local que elimina a necessidade de comunicação bidirecional e não requer o uso de *hardware* especializado.

3. STELE - Uma Técnica Simples Para a Estimativa Local do Atraso.

Em uma troca de mensagens entre dois nós, um quadro enviado em um tempo T_0 alcançará um determinado vizinho em um tempo T_0+T_X, onde T_X é o atraso na transmissão. Se o quadro em questão contém informações de sincronismo, é necessário então eliminar T_X. O nosso método se propõe a eliminá-lo estimando e compensando seu valor.

A técnica proposta se baseia no fato do nó sensor ser capaz de determinar o momento em que a transmissão de um quadro é concluída, que este tempo pode ser usado para estimar o momento em que o receptor receberá o quadro e que estas grandezas estão fortemente correlacionadas. Desse modo, as incertezas de tempo relativas ao acesso ao meio no momento da transmissão são removidas, restando apenas o atraso de propagação, da ordem de nano segundos (ver Tabela 1) e as componentes de tempo envolvidas no processo de recepção, sendo que estas últimas apresentam valores e variâncias bem menores quando comparadas a fase de transmissão. [Elson et al. 2002].

A Figura 1 ilustra o conceito da estimativa local do atraso sob o ponto de vista do nosso mecanismo, onde T_X consiste no atraso real de transmissão, ou seja, o tempo desde que a aplicação no *mote 'A'* envia uma mensagem, até a recepção final pelo *mote 'B'* e T_E é a estimativa desse atraso, calculado localmente, ou seja, é o tempo entre o envio de uma mensagem pela aplicação do *mote 'A'* até o instante em que o rádio deste *mote* notifique à CPU o seu envio. Em seguida explicaremos em detalhes o passo-a-passo do mecanismo:

1. Em um instante t_0 a aplicação no nó *'A'* comanda o envio de um quadro para o *mote 'B'*;
2. Em t_1 a *CPU* envia o quadro para o rádio;
3. No instante t_2 o quadro finalmente ganha o acesso ao meio;
4. Em t_3 a notificação do envio é remetida de volta a CPU;
5. E em t_4 a aplicação em *'A'* computa esta notificação.

No instante t_4 a aplicação já possui condições de computar a estimativa T_E, que é expressa por t_4-t_0.

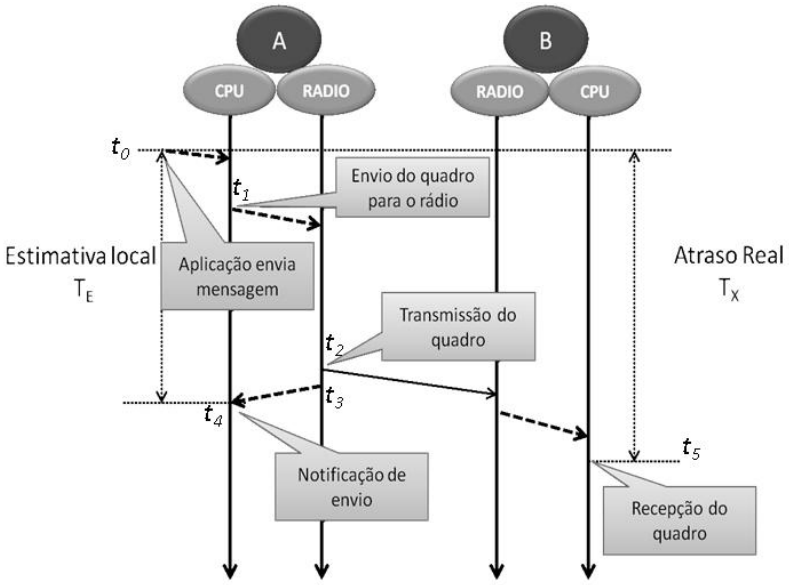

Figura 1. Diagrama de tempo da troca de mensagem do STELE.

Observe que todas as fases necessárias ao mecanismo ocorrem localmente em '*A'*. E note que nossa estimativa será tão boa quanto ela se aproxime do atraso real e sua qualidade se baseará em quatro fatores:

- A forte correlação estatística, que pretendemos demonstrar, entre a grandeza T_E e o atraso T_X;
- Que a nossa estimativa será tão acurada quanto à média da diferença T_X -T_E for próxima de zero e tão precisa quanto menor for sua variância;
- Que o tempo estimado T_E, relativo ao intervalo entre o envio da mensagem para o rádio e o momento real de seu envio, inclui a maior parte do valor esperado para o atraso real T_X;
- E que o tempo de recepção é proporcionalmente pequeno e menos aleatório, podendo ser compensado pela estimativa, sem prejuízo para precisão almejada.

A notificação de envio, na Figura 1, iniciada em t_3 é fundamental para esta técnica e pode ser facilmente obtida em uma série de dispositivos típicos das redes sem fio. Nos sensores que utilizam o sistema operacional TinyOS [Levis et al. 2005] ela é fornecida pelo evento *sendDone()*. Através desta notificação o nó *'A'* poderá computar uma estimativa de quando *'B'* receberá sua mensagem e, em um quadro posterior, enviar uma correção, sincronizando o nó *'B'*. O erro de estimativa, que chamaremos de ε, será a diferença entre T_X e T_E que claramente constituirá uma variável aleatória com sua própria média e variância. E a compensação do atraso será tão eficiente quanto à capacidade do modelo de relacionar estatisticamente o comportamento de T_E ao de T_X.

Na Subseção 3.1 serão apresentados os resultados dos testes com *motes* reais, que mostrarão com clareza:

- A natureza aleatória do atraso;
- A diferença entre as distribuições destes atrasos em ambos os modelos testados, Micaz [Crossbow 2005] e o Iris [Crossbow 2007];
- O grau de correlação entre atraso e estimativa, a eficácia e a precisão da técnica proposta.

Tabela 1. Caracterização das componentes do atraso [Maróti *et al*. 2004].

Tempo	Magnitude	Distribuição
Tempo até o envio	0 – 100ms	Não determinístico, depende da carga do processador.
Tempo de acesso ao meio	0 – 500ms	Não determinístico, depende da contenção no canal.
Tempo de transmissão e recepção	10 – 20ms	Determinístico, depende do comprimento da mensagem.
Tempo de propagação	< 1µs para distância de até 300m	Determinístico, depende da distância entre receptor e emissor.
Tempo para lidar com as interrupções	< 5µs na maioria dos casos, mas pode chegar a 30µs	Não determinístico, depende se as interrupções estão sendo desabilitadas.
Tempo de codificação decodificação	100 – 200µs, com variância < 2µs	Determinístico, depende do *chip set* do rádio.
Alinhamento de *Bytes*	0 – 400µs	Determinístico e pode ser calculado.

3.1 Relação entre o Atraso TX e a Estimativa TE

Para a avaliação da eficácia do nosso mecanismo, montamos o cenário experimental mostrado na Figura 2 e descrito a seguir: o experimento se inicia com o *mote '0'* enviando uma mensagem em difusão que alcança os *motes 'A'* e '*B',* respectivamente em T_{0A} e T_{0B}. Após a recepção destes quadros os sensores *'A'* e '*B'* passarão a usar, respectivamente, as marcas T_{0A} e T_{0B} como sua referência de tempo. Assumindo que o atraso entre estas marcas é menor que 30,5µs (1/32kHz), podemos, a partir desse ponto, considerar os motes *'A'* e *'B'* sincronizados a menos desse erro $\delta < 30{,}5$ µs.

Ao receber a mensagem de *'0'*, o *mote 'A'* dispara seu *clock* e a aplicação ordena o envio de uma mensagem em T_{1A}, que devido ao tempo de *back off*, entre outros fatores já descritos na Tabela 1, ganha o meio apenas em T_{2A}. Depois de algum tempo a CPU de 'A' é notificada do envio pelo evento *sendDone()* e em seguida envia a compensação T_E. Nesse ínterim *'B'* finalmente recebe T_{1A}, no instante T_{1B}. *'B'* em resposta a esse evento*,* envia uma mensagem em difusão contendo o valor de $T_{1B,}$, finalizando um ciclo de tomada de tempo. A função de *'S'* é capturar todos os quadros e servir como relógio que marca o tempo global do experimento. De modo, que o tempo T_X possa ser calculado *offline* para validação da técnica. Assim:

$$T_E = T_{1A} - T_{2A} \tag{3.1}$$

$$T_X = T_{1B} - T_{0B} - (T_{1A} - T_{0A}) \tag{3.2}$$

E o erro ε é calculado *off-line* durante o processamento dos dados e será:

$$\varepsilon = T_X - T_E \tag{3.3}$$

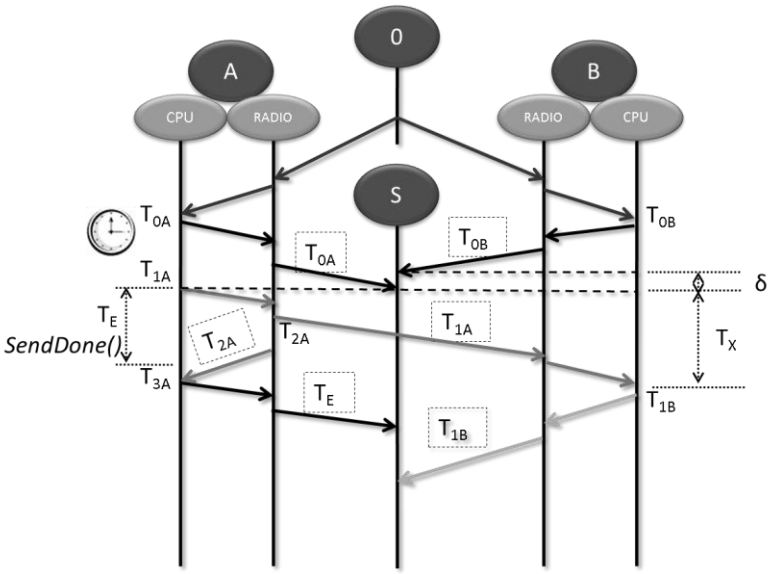

Figura 2. Diagrama de tempo detalhado do experimento.

Para este experimento consideramos o tempo de propagação, da ordem de nano segundos desprezível para a precisão requerida. Um fator importante nos resultados de nossos experimentos em ambos os modelos de *mote*s é que $T_E < T_X$. Isso nos permite predizer que há ainda um componente que poderá ser usado para melhorar a precisão do ajuste.

Tabela 2. Comparação dos parâmetros estatísticos dos atrasos e das estimativas para os diferentes modelos de sensores.

	Atraso MicaZ (ms)	**Estimativa MicaZ (ms)**	**Atraso Iris (ms)**	**Estimativa Iris (ms)**
Valor mínimo	2,01	2,01	1,56	1,49
Valor máximo	11,90	11,47	9,79	9,70
Intervalo	9,89	9,46	8,23	8,21
Mediana	6,98	6,65	4,15	4,08
Média	6,92	6,69	4,64	4,53
Erro Padrão	0,13	0,13	0,11	0,11
Variância	7,45	7,50	4,12	4,08
Desvio Padrão	2,73	2,74	2,03	2,02
Curtose	-1,20	-1,22	-1,23	-1,24
Assimetria	0,02	0,03	0,33	0,33
Coef. de Variação	0,39	0,40	0,44	0,44

O atraso de transmissão medido possui distribuições diferentes para estes dois modelos de *motes*. Confirmamos essa suposição com testes de aleatoriedade [R Development Core Team 2007] que ratificaram com *p-value*> 0,5 que as amostras de T_X em cada um dos dispositivos são aleatórias e possuem diferenças estatísticas significativas entre si, suficientes, para ser útil caracterizar suas distribuições

separadamente. A Tabela 2 evidencia tais diferenças, mas também a proximidade entre nossas principais grandezas: a estimativa e o atraso.

Além da Tabela 2, a sobreposição dos histogramas das distribuições dos atrasos, observados na Figura 3 nos mostra mais claramente que o comportamento das distribuições de T_X nos dois modelos de *motes* testados é significativamente diferente. Os valores da Tabela 2 foram obtidos através de cerca de 3000 ciclos de tomadas de tempo no cenário da Figura 2, para cada par de modelo experimentado. E mesmo com esse contraste de comportamento, que se deve às diferenças de *hardware* existentes entre um e outro modelo, a relação esperada entre o atraso medido e a estimativa persiste, com graus muito próximos de correlação, comprovando que nosso método pode ser aplicado em ambos os modelos.

Depois de demonstrar a aleatoriedade das duas variáveis, a significativa diferença estatística entre elas e determinar que os dois tipos de *motes* testados possuem distribuições dos atrasos diferentes entre si , vamos mostrar a correlação estatística que existe entre essas duas grandezas e analisar sua relevância para o nosso mecanismo.

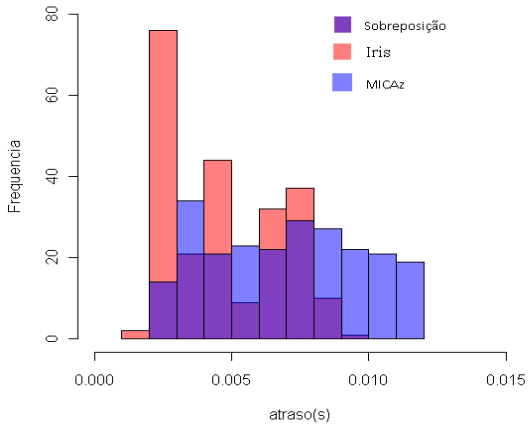

Figura 3. Histograma das distribuições do atraso.

Análises estatísticas dos dados obtidos experimentalmente, nos mostraram através do teste de Pearson, que a relação entre as variáveis aleatórias T_E e T_X, é de $\boldsymbol{\rho}$ = **0.9986** com 95% de intervalo de confiança para o Micaz e $\boldsymbol{\rho}$ = **0.9995** com o mesmo intervalo para o Iris, caracterizando assim ambas as associações como **"positivas fortíssimas"**.

Através dos gráficos de dispersão do atraso em função da estimativa, em ambos os modelos, mostrados nas Figuras 4a e 4b observamos a forte correlação estatística existente entre estas grandezas e a Tabela 2 mostra, além das diferenças entre as variâncias dos diferentes modelos, que nossa tese inicial de que a T_E contém grande parte das componentes do atraso total e que através do nosso método, foi possível alcançar uma estimativa do atraso com erro médio da ordem de 240±30,5µs nos modelos Micaz e de 80 ±30,5µs no Iris.

Esse fator de 30,5µs a que nos referimos foi introduzido pela imprecisão inicial do experimento. Essa imprecisão se deve ao fato de nosso experimento começar com os nós sensores tendo seus relógios fora de sincronia e uma sincronização inicial deve ser alcançada. Para isso utilizamos um *mote* de referência *'0'* (Figura 2), programado especificamente para este fim, que envia através de uma mensagem em *broadcast* um quadro que os dois *motes "A"* e *"B"*, dentro do mesmo domínio de difusão, recebem

quase que simultaneamente. O *mote 'A'* recebe este quadro em T_{0A}, e *'B'* o recebe em $T_{0B} = T_{0A} \pm \delta$. Sendo os osciladores de ambos os dispositivos de 32kHz. O erro δ introduzido será de no máximo 30,5µs.

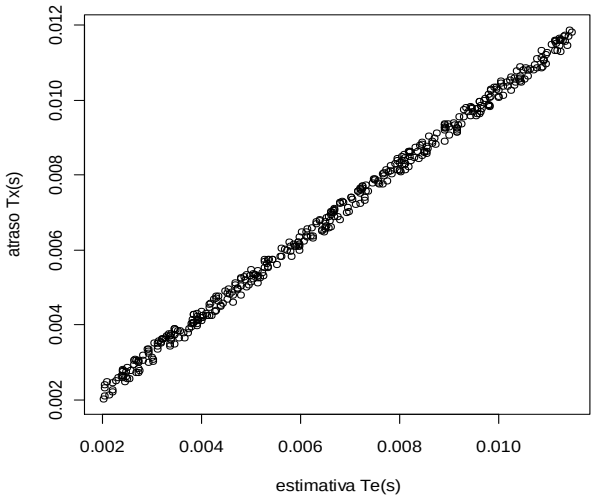

Figura 4a. Dispersão do atraso MicaZ.

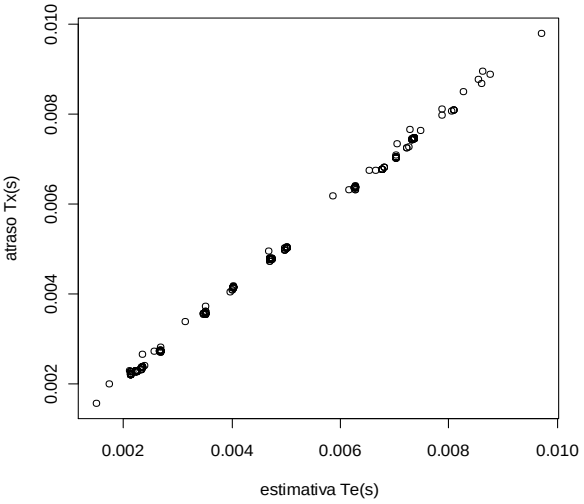

Figura 4b. Dispersão do atraso Iris.

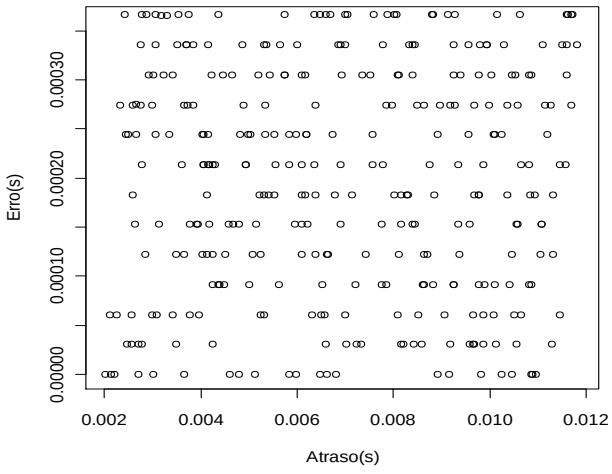

Figura 5a. Erro x Atraso *mote* MicaZ.

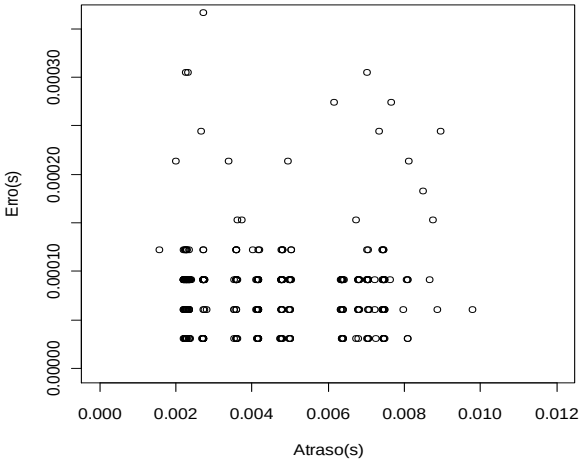

Figura 5b. Erro x Atraso *mote* Iris.

Essa limitação de frequência do relógio desses nós sensores também é responsável pelo comportamento discreto do erro ε em função do atraso apresentado nas Figuras 5a e 5b. Nessa figura podemos observar também a ordem de grandeza e a distribuição do erro ao longo dos atrasos. Verificamos através dos gráficos que a concentração do erro em valores abaixo de 150µs, no Iris, confirma que nesse sensor nosso mecanismo alcançou maior eficiência, Enquanto no Micaz, onde a variância foi maior, esse efeito de concentração não se verifica.

Os testes foram realizados não apenas em um tipo de *mote*, mas em dois dos modelos mais utilizados em RSSF. Deste modo e em virtude dos resultados obtidos não

temos razão para supor que as relações entre o erro ε, a estimativa T_E e o atraso T_X não obedeçam estes mesmos comportamentos em dispositivos de outros fabricantes.

4. Consumo de Energia

Para avaliar o consumo de energia dos *motes* durante o processo de sincronização montamos dois cenários em ambiente simulado usando o Avrora [Titzer et al. 2005]. O primeiro cenário de simulação consistiu de uma rede de sensores com cinco nós em um mesmo domínio de difusão e o segundo de uma rede com vinte nós em um ambiente com condições similares de ruído e interferência e topologia. Evidenciamos com os resultados nesses dois diferentes cenários que nosso método é menos intensivo no envio de mensagens, no uso do rádio e de CPU e sua eficiência aumenta para redes mais densas. Para a comparação, duas técnicas foram simuladas: STELE e FTSP. Em ambas, os nós clientes sincronizam seus *clocks* com um único nó de referência.

Tabela 3. Gasto de energia referente à CPU e ao Rádio com cinco nós.

Consumo em CPU (mJ)			Consumo do rádio em transmissão (mJ)		
Intervalo de Sincronização(s)	**STELE**	**FTSP**	**Intervalo de Sincronização (s)**	**STELE**	**FTSP**
1	178,757	471,0052	1	31,424	90,111
5	138,908	173,8109	5	6,222	13,387
10	133,891	148,5399	10	3,058	6,003
30	130,556	135,1897	30	0,949	1,622
60	129,725	131,340	60	0,422	0,433

Tabela 4. Gasto de energia referente à CPU e ao Rádio com 20 nós.

Consumo em CPU (mJ)			Consumo do rádio em transmissão (mJ)		
Intervalo de sincronização(s)	**STELE**	**FTSP**	**Intervalos de sincronização(s)**	**STELE**	**FTSP**
1	662,549	3939,134	1	31,424	368,017
5	551,303	837,314	5	6,222	124,432
10	532,961	714,532	10	3,058	22,509
30	520,745	563,847	30	0,949	5,111
60	517,693	527,330	60	0,422	1,544

As medidas, feitas em 10 tomadas incluíram apenas o consumo de energia referente ao uso de CPU e à transmissão de quadros pelo rádio. O tempo de cada tomada foi de trezentos segundos, usando como intervalo de sincronização: 1, 5, 10, 30 e 60 segundos para ambos os mecanismos. Os rádios nos dois cenários permaneceram ligados sempre que não estavam transmitindo, jamais sendo colocados em modo *sleep*.

Nessa simulação comparamos nosso mecanismo ao FTSP, descrito na Seção 2. Para isso, usamos uma versão modificada do TestFTSP, distribuído juntamente com a versão 2.1.1 do TinyOS. Essas modificações visaram apenas remover o envio das mensagens de relatório que não são necessárias em um ambiente de sincronização real.

Os dados obtidos nestas simulações podem ser verificados nas Tabelas 3 e 4, onde estão tabulados os gastos de energia com o uso de CPU em miliJoules, assim como os gastos com o envio de quadros. A Tabela 3 tem como referência os dados da simulação com cinco nós enquanto a Tabela 4 mostra os dados simulados em uma rede com vinte nós. Note, nas Tabelas 3 e 4 que o custo energético com o uso do rádio no STELE permanece o constante com o aumento do número de nós enquanto no FTSP há uma grande variação, que se deve, entre outas coisas, a intensa troca de quadros para a montagem da tabela de regressão linear.

Durante os experimentos observamos ainda que o FTSP difunde suas mensagens de em momentos aleatórios dentro do seu intervalo de sincronização, exigindo que os rádios dos receptores permaneçam ligados aguardando estes quadros, dificultando o uso do *dutycycling* para economia de energia. Já nosso método, que envia seus quadros de sincronização em tempos determinados, se adequa bem a essas soluções. Desta forma seu uso pode gerar uma economia expressiva, dado que o consumo de energia do rádio ligado é da ordem de **dezenas** de joules.

5. Cenários de Uso, Conclusões e Trabalhos Futuros

A técnica apresentada pode ser aplicada em um processo de sincronização em duas fases. Na primeira fase, o tempo do nó Mestre é enviado e numa segunda fase apenas o *offset* com a estimativa do atraso é transmitido. Outro cenário de uso seria a previsão de *jitter*. Neste caso o nó Mestre poderia aprender, durante os quadros iniciais, a variação do atraso na rede, levando em consideração sempre a condição do enlace local. E em quadros posteriores a variação pode ser compensada atrasando ou adiantando as transmissões em relação ao dado aprendido.

Neste trabalho apresentamos um mecanismo simples, pouco custoso em energia, que foge da abordagem tradicional que exige trocas de quadros entre servidores e clientes e também não requer enlaces bidirecionais. Essas vantagens somadas ao fato que sua implementação é independente de características especiais de *hardware* confere ao mecanismo STELE a sua universalidade como solução, isto é, este mecanismo pode de ser usado em quaisquer dos modelos de *motes* existentes. Esta técnica, também por enviar os quadros em tempos determinados se adequa bem a soluções que usem *dutycycling* e se ajusta perfeitamente a aplicações que permitam o uso de sincronização *post facto*, onde os nós estão geralmente fora de sincronismo e somente após a ocorrência de um evento externo, um nó inicia a sincronização com seus vizinhos.

Em trabalhos futuros pretendemos verificar a escalabilidade do mecanismo em redes multissalto sujeitas a diferentes níveis de interferência e em cenários de mobilidade.

Referências

Akyildiz, I. F. and Vuran, M. C. (2010). Wireless Sensor Networks. Chichester, UK: John Wiley & Sons, Ltd. p. 243–262.

Anastasi, G., Conti, M., Di Francesco, M. and Passarella, A. (2006). An adaptive and low-latency power management protocol for wireless sensor networks. MobiWAC 2006 Proceedings of the 2006 ACM International Workshop on Mobility Management and Wireless Access, v. 2006, p. 67–74.

Committee, T. (2008). IEEE Std 1588-2008, IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems. Society. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4579760.

Crossbow (2005). Crossbow MicaZ mote specification. http://www.openautomation.net/uploadsproductos/micaz_datasheet.pdf, [acessado em 8/8/2012].

Crossbow (2007). Crossbow Iris mote specification. http://www.dinesgroup.org/projects/images/pdf_files/iris_datasheet.pdf, [acessado em 8/8/2012].

Elson, J., Girod, L. and Estrin, D. (2002). Fine-grained network time synchronization using reference broadcasts. ACM SIGOPS Operating Systems Review, v. 36, n. SI, p. 147–160.

Lenzen, C., Sommer, P. and Wattenhofer, R. (2009). Optimal clock synchronization in networks. Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems SenSys 09, p. 225–237.

Levis, P., Madden, S., Polastre, J., et al. (2005). TinyOS : An Operating System for Sensor Networks. Ambient Intelligence, v. 35, p. 115–148.

Maróti, M., Kusy, B., Simon, G. and Lédeczi, Á. (2004). The flooding time synchronization protocol. In Proceedings of the 2nd international conference on Embedded networked sensor systems SenSys 04. ACM Press. http://portal.acm.org/citation.cfm?doid=1031495.1031501.

Mills, D. L. (1985). Network Time Protocol (NTP). Network. IETF. http://www.ietf.org/rfc/rfc958.txt.

R Development Core Team (2007). R: A language and environment for statistical computing. ISBN 3-900051-07-0. [Austria, Ed.]R Foundation for Statistical Computing Vienna Austria. R Foundation for Statistical Computing. http://www.r-project.org.

Titzer, B. L., Lee, D. K. and Palsberg, J. (2005). Avrora: scalable sensor network simulation with precise timing. 2005, p. 477–482.

Woo, Alec and Culler, D. (2002). Complex Behavior at Scale : An Experimental Study of Low-Power Wireless Sensor Networks. Citeseer, n. 02-0013, p. 1–11.

Yick, J., Mukherjee, B. and Ghosal, D. (2008). Wireless sensor network survey. Computer Networks, v. 52, n. 12, p. 2292–2330.