UNIVERSIDADE FEDERAL FLUMINENSE

HEDER DORNELES SOARES

SINCRONIZAÇÃO EM REDES SENSORES SEM FIO

NITERÓI

2016

UNIVERSIDADE FEDERAL FLUMINENSE

HEDER DORNELES SOARES

SINCRONIZAÇÃO EM REDES SENSORES SEM FIO

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Redes e Sistemas Distribuídos e Paralelos

Orientador:
CÉLIO VINICIUS NEVES DE ALBUQUERQUE

Co-orientador:
RAPHAEL PEREIRA DE OLIVEIRA GUERRA

NITERÓI

2016

HEDER DORNELES SOARES

SINCRONIZAÇÃO EM REDES SENSORES SEM FIO

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Redes e Sistemas Distribuídos e Paralelos.

Aprovada em Janeiro de 2016.

BANCA EXAMINADORA

Prof. Célio Vinicius Neves de Albuquerque - Orientador, UFF

Prof. Raphael Pereira de Oliveira Guerra, Coorientador, UFF

Prof. <NOME DO AVALIADOR>, <INSTITUIÇÃO>

Prof. <NOME DO AVALIADOR>, <INSTITUIÇÃO>

Prof. <NOME DO AVALIADOR>, <INSTITUIÇÃO>

Niterói

2016

Dedicatória(s): Elemento opcional onde o autor presta homenagem ou dedica seu trabalho (ABNT, 2005).

Agradecimentos

Elemento opcional, colocado após a dedicatória (ABNT, 2005).

Resumo

Elemento obrigatório, constituído de uma sequência de frases concisas e objetivas e não de uma simples enumeração de tópicos, não ultrapassando 500 palavras (ABNT, 2005).

Palavras-chave: Palavras representativas do conteúdo do trabalho, isto é, palavras-chave e/ou descritores, conforme a ABNT NBR 6028 (ABNT, 2005).

Abstract

Wireless sensors networks are distributed system composed of battery-powered nodes with low computational resources. Like in many distributed systems, some applications of WSN require time synchronization among their nodes. In the particular case of WSN, synchronization algorithms must respect the nodes computational constraints. The well known FTSP protocol is famous for achieving nanosecond precision with low overhead. However, it relies on MAC timestamp, a feature not available in all hardware. In this work, we propose MAC timestamp independent version in order to extend and adapt FTSP to work on hardware that do not have MAC timestamp while keeping the low overhead and high synchronization precision our results indicate an average synchronization error of $3\mu s$ per hop, while adding a corretion message every three seconds.

Keywords: WSN, Time Synchronization, FTSP, MAC Timestamp.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

RSSF	:	Rede de Sensores Sem Fio;
FTSP	:	Flooding Time Synchronization Protocol;
WSN	:	Wireless Sensor Network;
TI	:	Tecnologia da Informação;
MAC	:	Media access control;
NTP	:	Network Time Protocol;
RBS	:	Reference Broadcast Synchronization;
TDMA	:	Time Division Multiple Access;
TPSN	:	Timing-sync Protocol for Sensor Networks;

Sumário

Capítulo 1

Introdução

- WSN são redes de sensores com recursos ...

As Redes de Sensores Sem Fio (RSSF) são compostas de pequenos dispositivos equipados com uma antena de comunicação sem fio, um ou mais tipos de sensores, e uma CPU de baixa capacidade de processamento de dados [2]. Estes dispositivos geralmente são chamados de *motes*. Devido ao alcance do sinal de rádio limitado e da restrição energética uma RSSF tem limitações e caracteristicas únicas que a difere de redes de computadores e sistemas distribuídos tradicionais [1].

As RSSF tem várias aplicações em diversos campos. A implantação de sensores em aplicações militares foi sempre muito difundido, de modo que a introdução de *motes* era uma incorporação natural para o avanço dos sistemas já utilizados. Aplicações que foram aprimoradas com o uso de RSSF incluem o rastreamento de inimigos e alvos [42], monitoramento de veículos [34], sistema contra atirador [32] e sistemas de vigilância [13]. Monitoramento ambiental também provê oportunidades para aplicação de redes sensores sem fio. Nosso meio ambiente tem uma gama muito grande de informações que desempenham um papel importante em nossa qualidade de vida, como a qualidade do ar, água, som e radiação solar que estamos expostos todos os dias e que diretamente afetam nossa saúde [27, 3]. Com o interesse cada vez maior em coputação verde nos leva as preocupações com o consumo de energia eletrica em instalações de TI [5], e as RSSF têm um papel estratégico no monitoramento e controle destes ambientes [44].

Algumas dessas aplicações requerem mecanismos de sincronização de tempo com boa

precisão e escalabilidade, tudo isso em conformidade com os seus baixos recursos computacionais e disponibilidade energética [44, 32, 42]. Métodos de sincronização tradicionais, como NTP [24], amplamente utilizado em servidores e clientes de rede não se aplicam em RSSF devido a fatores não determinístas relacionados com o acesso ao meio na transmissão sem fio dos sensores. O Flooding Time Synchronization Protocol (FTSP) é o algoritmo de sincronização de relógio mais popular para RSSF [23], ele é tolerante a falhas, consegue alta precisão ($\sim 1,5\mu s$ por salto) utilizando *timestamps* em camadas baixas da pilha de rádio, usa a técnica de regressão linear para compensar o escorregamento do relógio usando poucas mensagens pela rede. Contudo, *timestamps* na camada MAC não é um recurso padronizado, e portanto, não interoperáveis entre diferentes hardware e protocolos da camada física. Há um esforço do Google para padronizar *timestamping* na camada MAC em RSSF [41], mas até agora não há muita conformidade.

Muitos protocolos de sincronização usam MAC *timestamp*: alguns têm menos precisão do que o FTSP, eles se concentram em outros problemas e fazem suposições mais restritivas [11, 39]; outros podem conseguir uma melhor precisão entre os nós distantes [26, 22, 35]. Elson et ai. propôs o Reference Broadcast Synchronization [8] (RBS) para eliminar a incerteza do remetente sem MAC *timestamp* removendo o *sender* do caminho crítico. A idéia é que um terceiro irá transmitir um *beacon* para todos os receptores. O *beacon* não contém qualquer informação de tempo; em vez disso os receptores irão comparar seus relógios um ao outro para calcular o escorregamento de seus relógios. Tem $\sim 30\mu s$ erro por salto e é independente da MAC *timestamp*. Ranganathan e Nygard oferecem uma boa visão geral desses protocolos [31].

1.1 Sincronização em Redes Sensores Sem Fio

Razões e desafios. NTP e GPS, Necessidade de algoritmos de sincronização especificos para RSSF.

- Intro sobre redes sensores sem fio.
- Necessidade da Sincronização em sistemas distribuídos.
- Problemas inerentes a sincronização, fontes de atraso e imprecisão em RSSF

Em sistemas distribuídos como as redes de sensores sem fio, cada nó tem seu próprio relógio e sua própria percepção de tempo. No entanto, uma escala de tempo comum entre nós sensores é importante para identificar relações entre eventos que estejam sendo

monitorados, para apoiar a eliminação de dados redundantes de sensores e para facilitar a operação da rede. Uma vez que cada nó em uma rede de sensores opera de forma independente e conta com o seu próprio relógio, as leituras do relógio de diferentes nós de sensores também será divergente. Além destas diferenças aleatórias, a diferença entre os relógios de um conjunto de sensores vai aumentar ainda mais devido às taxas de escorregamento dos seus osciladores. Portanto, a sincronização dos relógios é necessário para assegurar que os tempos de detecção dos eventos que estejam sendo monitorados possam ser comparados e conseguinte significância.

Embora as técnicas de sincronização de tempo para redes com fios receberam uma quantidade significativa de atenção, esses métodos não são apropriados para uso em RSSF sem modificação, devido aos desafios colocados pelos ambientes sensores sem fio. Estes desafios incluem o tamanho das redes de sensores, a necessidade de auto-configuração e robustez, a mobilidade dos sensores além da necessidade primordial de conservação de energia [7].

Nas RSSF a eficiência energética é uma necessidade básica para seu funcionamento, em uma rede de larga escala não é possível fornecer uma fonte de energia para toda a rede, esses sensores geralmente são acoplados a uma bateria. Devido ao tamanho dos dispositivos a quantidade de energia que eles podem produzir ou armazenar é muito limitada [37].

Sincronização de tempo é um serviço necessário para muitas aplicações e serviços em sistemas distribuídos em geral. Numerosos protocolos de sincronização de tempo têm sido propostos para ambos os sistemas com e sem fio, por exemplo, o Network Time Protocol (NTP) [24] é uma abordagem de sincronização escalável, robusto e auto-configurável amplamente utilizado. Especialmente em combinação com o Sistema de Posicionamento Global (GPS), tem sido utilizado para alcançar a precisão da ordem de alguns microssegundos. No entanto, abordagens como NTP não são adequados para RSSFs devido a ser ineficiente neste contexto pois necessita de maior quantidade de memória. Já o uso de GPS pode elevar o custo de implantação de um sistema de monitoramento, além de requerer vários minutos para sintonizar, GPS também necessita de um uso maior de energia.

1.1.1 Conceitos Básicos

Terminologias. Efeitos do Ambiente. Restrição energética. Mobilidade e acesso ao meio sem fio. Capacidade dos dispositivos.

Os relógios dos *motes* nas RSSF são baseados em osciladores de quartzo, o tempo é mensurado nas oscilações do cristal em conjunto a um contador, este contador tem seu valor decrementado até atingir zero, então ele reinicia novamente para seu valor padrão e o processo continua indeterminadamente, toda vez que o contador chega a zero é gerado uma interrupção e essa interrupção é chamada de *tick* ou *clock tick*, ela incrementa outro contador o do relógio a nível de *software* que é de onde vem tempo que utilizamos no sistema operacional. Com isso, esse relógio de *software* fornece para o *mote* um valor de tempo local (*local time*).

Supondo a existência de dois nós, cada um com o seu próprio relógio R_a e R_b, onde $R(t)$ simboliza o valor do relógio no tempo t, sendo t o tempo de referência. Se verificarmos $R_a(t)$ e $R_b(t)$ podemos encontrar uma diferença entre os tempos, essa diferença é chamada de *clock offset*. Para corrigir esta diferença é necessário realizar uma sincronização para que os tempos sejam iguais ou que a diferença seja tão pequena quanto possível. Podemos definir o *clock rate* como a frequência que um relógio progride, assim a frequência do tempo t de um relógio R_a é $R_a^{'}(t)$, ainda temos o *clock skew* que é a diferença em termo de frequência entre dois relógios [37].

Em um mundo perfeito $R_p(t) = t$ para qualquer instante de tempo t, o relógio perfeito deve ter a variação pelo tempo de referência iguais, como visto na Equação (1.1). Vários fatores influenciam o *clock rate*, como por exemplo, temperatura e efeitos do ambiente, energia, limitação do hardware, impureza do crital entre outros. Esta imprecisão do *clock* faz com que a taxa de alteração dos relógios os afaste do valor de referência, $dR/dt > 1$ resulta em um relógio rápido, já $dR/dt < 1$ apresenta um relógio lento, a Figura 1.1 demonstra esse efeito. Este comportamento é responsavel pela instabilidade das leituras de tempo nos sensores, fazendo com que seja necessário realizar uma sincronização e mesmo após realizada o desvio irá continuar fazendo efeito sobre o relógio, trazendo a necessidade realização de sincronizações periódicas.

$$\frac{dR}{dt} = 1 \tag{1.1}$$

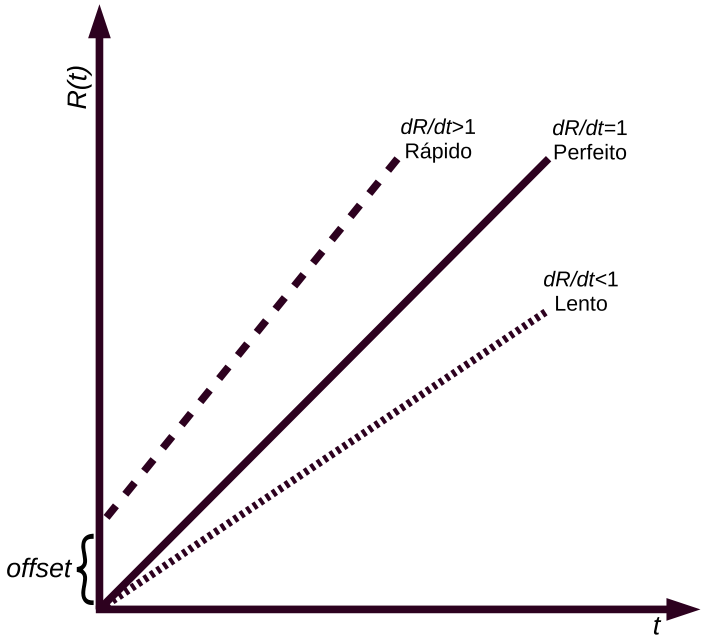

Figura 1.1: Comportamento do relógio em relação ao tempo de referência

A sincronização deve ocorrer de forma cíclica, pois os relógios sempre vão afastar-se dos valores de referência, mas o ajuste do tempo local deve ser realizado de forma gradativa e suave, ou seja, deve-se utilizar uma função para mudar o declive do tempo local ao invés de uma atribuição direta. Um problema de efetuar uma mudança corrigindo o valor do tempo diretamente seria o relógio pulando para frente e para trás, ocasionando problemas como por exemplo, uma tarefa agendada poderia nunca ocorrer devido o tempo de sua execução nunca ter existido por causa do salto dos *ticks*, ou ainda o mesmo tempo pode ocorrer duas vezes em virtude de um salto para trás no relógio.

Podemos utilizar duas formas para classificarmos os tipos de sincronização: externo e interno. No método externo os relógios de todos os nós são sincronizados utilizando uma referência de tempo (*clock reference*) externa a rede, geralmente uma fonte de tempo UTC (Coordinated Universal Time), ou seja, o tempo do mundo real baseado em um relógio atômico. Por outro lado, na sincronização interna todos os nós são sincronizados sem a ajuda de uma referencia de tempo externa, o objetivo é minimizar a diferença entre os relógios locais dos sensores tendo assim uma noção consistente do tempo entre todos os nós da rede [20].

Um conjunto de fatores podem influênciar o funcionamento de sistemas de sensores sem fio alterando seu desempenho, os principais componentes que devem ser observados são listados abaixo:

Efeitos Ambientais: Pressão, temperatura, humidade são algumas das condições de ambiente que alteram o funcionamento normal dos osciladores. Muitas RSSF são criadas para implantação *outdoor*, nestes cenários as condições são bem diferentes das apresentadas em um ambiente estável como um laboratório [19]. Fenômenos climáticos como chuva, neblina e vento também tem influência na performance de sistemas de RSSF, podendo até mesmo impossibilitar seu funcionamento [28].

Limitações Energéticas: Economizar energia é o maior interesse em RSSF, o tamanho pequeno dos nós sensores limita o tamanho de sua bateria, bem como a quantidade de energia que ele pode produzir usando placa solar [9]. A transmissão de mensagens é a base dos algoritmos de sincronização, a energia utilizada para a transmissão de 1Kb a uma distância de 100 metros é aproximadamente de 3 joules, o mesmo necessário para o processador executar 3 milhões de instruções [30]. Transmitir é mais custoso energeticamente do que computar, assim os algoritmos de sincrozação devem minimizar a necessidade de troca de mensagens para serem mais eficientes energeticamente.

Wireless e Mobilidade: A comunicação sem fio apresenta grandes desafios nas redes de sensores, como vimos, é gasto mais energia transmitindo dados do que processando, fenômenos climáticos como chuva, neblina degradam a performance da rede. A característica de transmissão de dados por difusão nas RSSF requer a utilização de um protocolo MAC (Media Access Control) para ordenar o acesso ao meio, dependendo da densidade da rede, faixa de alcance dos nós (entre 20-100 metros) e o tráfego dos dispositivos podem incorrer em interferências na comunicação que levam a perdas de mensagens e conexão intermitente. Grande quantidade de redes sensores são móveis, os nós podem se mover, podem ocorrer falhas na comunicação, a bateria pode esgotar e o nó ficar inativo, causando constantes mudanças na topologia das redes. Por consequência destes problemas os protocolos de sincronização devem ser projetados para respeitar todas essas propriedades.

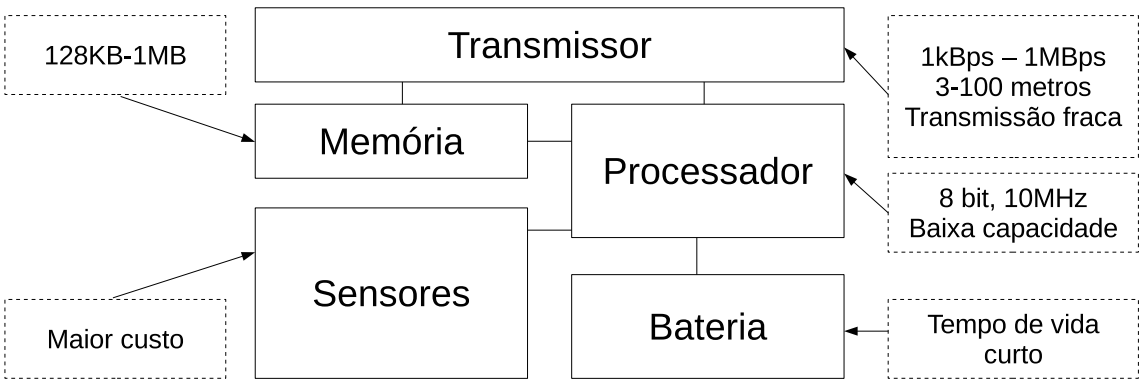

Figura 1.2: Componentes de um mote para RSSF [14]

Limitação de *Hardware*: O tamanho e o preço dos sensores são algumas das razões pelas quais seu *hardware* ser bastante limitado, aumentar o tamanho poderia elevar seu custo e consumo de energia, podendo tornar inviável as aplicações de redes sensores em larga escala [18]. A capacidade computacional destes dispositivos é similar aos computadores pessoais do início da década de 80, a Figura 1.2 ilustra os componentes típicos de um nó sensor, transmissor fraco sujeito a perdas, memória limitada, processador de baixa capacidade aliado a fonte de energia restrita. Os algoritmos para RSSF devem respeitar esse conjunto de desafios e limitações com o objetivo de minimizar o uso dos recursos e manter seus resultados funcionais.

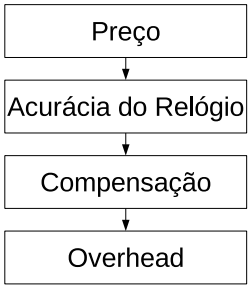

Figura 1.3: Tradeoff custo-beneficio [15]

Os novos avanços nos sistemas integrados nos permitiram desenvolver dispositivos sensores com baixos recursos de energia, rádio e processamento. Visto que uma rede de sensores consista de um bom número de sensores, o valor de um só nó afeta a implantação da rede toda. Alguns preveem que com a grande quantidade de sensores aplicados hoje em dia, seu valor possa chegar a ser menos de $1 dolar [6, 40].

Os *motes* são fabricados com materiais de baixo custo e as vezes de menor qualidade, como por exemplo, um cristal de quartzo de baixa qualidade pode ser um dos fatores que tornam o relógios mais imprecisos. A Figura 1.3 apresenta o *tradeoff* entre o preço e a qualidade, o preço de produção do dispositivo determina a precisão do seu relógio, assim quanto menos preciso o relógio é mais compensação será necessária para corrigir seu tempo, a compensação gera maior *overhead* de comunicação que é gasto no uso de comunicação/energia.

1.1.2 Fontes de Imprecisão em Comunicação

Send delay: Access delay: Propagation delay: Receive delay:

A troca de mensagem com *timestamp* (marca de tempo) é a principal ferramenta dos protocolos de sincronização, porém o não determinismo causado pelas latências no processo de troca de mensagens interfere na precisão que pode ser obtida utilizando este método. Por exemplo, quando um nó envia uma mensagem com seu *timestamp* para outro, os diversos procedimentos pelo qual essa informação vai percorrer até o destino apresentam um custo de computação e tempo para ser realizada, assim quando se dá a recepção do dado no destino já se passou uma quantidade determinada tempo, o relógio do nó origem continuou rodando e o *timestamp* enviado está inconsistente com o valor atual da fonte. A latência deste procedimento pode ser decomposta em alguns componentes, descritos por Kopetz e Ochsenreiter [21] e classificados a seguir:

- Tempo no envio: É o tempo que o sistema leva para montar a mensagem e passar a requisição de transmissão para a camada de acesso ao meio. Fonte de atraso causado principalmente pelo sistema operacional e da carga de uso do CPU.

- Tempo de acesso ao meio: Ocorre devido a janela de contenção do acesso ao meio, é o tempo que o nó leva para ter acesso físico para transmissão esperando o canal estar desocupado para então para usá-lo. Redes sem fio compartilham o mesmo meio de comunicação necessitando de uma forma de coordenar quem pode usar o canal e assim evitar colisões. O protocolo MAC do 802.11 serve como base para as variações utilizadas em RSSF, estes buscam atingir maior eficiência no consumo de energia, algumas opções destes protocolos são: BMAC (*Versatile low power media access*) [29], S-MAC (*Sensor-MAC*) [43], Box-MAC [25], PAMAS (*Power-Aware Multi-Access Protocol*) [33] e ainda os *surveys* [38, 17] tem referências mais atualisadas.

- Tempo de transmissão: Tempo que leva para um nó transmitir uma mensagem para outro nó, varia dependendo do tamanho do pacote e da velocidade da banda.

- Tempo de propagação: Tempo que a mensagem leva para chegar no receptor desde o momento que deixou o emissor. Este tempo é muito pequeno e geralmente insignificante para a análise geral do atraso.

- Tempo de recebimento: É o tempo que o receptor leva para receber a mensagem, sendo igual ao da transmissão.

- Tempo de recepção: Similar ao tempo de envio, é o *delay* de receber a mensagem e encaminha-la para o nível da aplicação.

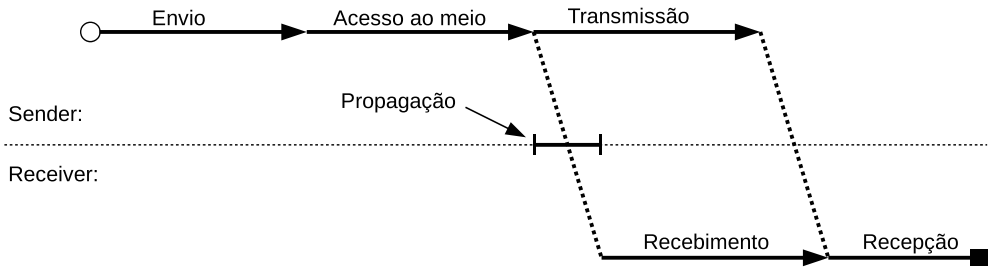

Figura 1.4: Decomposição das fontes de atraso na transmissão de uma mensagem

A Figura 1.4 ilustra as tomadas de tempo que compõem as fontes de atraso na troca de mensagem, as principais técnicas de sincronização concentram-se em eliminar esses *delays* ou ainda computar seus respectivos atrasos para usá-los mais a frente na forma de compensação. Outros trabalhos expandiram ainda mais a compreensão dos componentes de atraso [23, 16, 12], esses componentes estão listados abaixo:

- Tempo de tratamento de interrupções: O tempo que o microcontrolador demora para responder a solicitação do *chip* de rádio.
- Codificação e Decodificação: Codificação é o tempo que o transmissor tira para codificar os *bits* nas ondas de rádio. E decodificação o tempo que o receptor demora até transformar as ondas eletromagnéticas em *bits*.
- Alinhamento de Byte: Diferentes tipos de alinhamento entre o transmissor e receptor, causam a necessidade de reordenar os bytes. Alguns *chips* de rádio não são capazes de resolver o alinhamento de um fluxo de mensagem, o receptor deverá resolver o *offset* da mensagem.

Alguns dos componentes de atraso são de origem não determinístico, outros podem ser calculados e estimados, a 1.1 sintetiza o tamanho e classificação dos componentes já listados. Note que nem todos os transmissores são afetados por todas essas fontes de erro.

1.1.3 Mensagens de Sincronização

Pairwise synchronization, Uniderecional One-Way Message Exchange, Two-Way Message Exchange, Receiver-Receiver Synchronization

Tempo	Magnitude	Distibuição
Enviar e Receber	$0 - 100\ ms$	Não determinístico, depende da carga do CPU
Acesso ao Meio	$10\ ms - 500\ ms$	Não determinístico, depende da contenção do canal
Transmissão e Recepção	$10\ ms - 20\ ms$	Determinístico, depende do tamanho da mensagem
Propagação	$< 1\mu s$ para distâncias acima de 300m	Determinístico, depende da distância
Tratar Interrupção	$< 5\mu s$ na maioria dos casos, mas pode chegar a 30 μs	Não determinístico, depende das interrupções
Codificar Decodificar	$100\ \mu s - 200\mu s$, $< 2\mu s$ de variância	Determinístico, depende o chip de rádio
Alinhamento de Bytes	$0 - 400\mu s$	Determinístico, pode ser calculado

Tabela 1.1: Fontes de atraso na transmissão de mensagens [23]

Os protocolos de sincronização essencialmente são construídos a partir de troca de mensagens entre os nós participantes da rede. O modelo mais simples de sincronização utiliza pelo menos uma mensagem, chamado de *pair-wise synchronization* nele dois nós podem sincronizar apenas trocando uma mensagem, para sincronizar uma rede inteira repete-se a operação entre parelhos até que todos os pares estarem sincronizados, quando isso ocorre damos o nome de *network-wide synchronization*. A seguir vamos discorrer sobre métodos de sincronização.

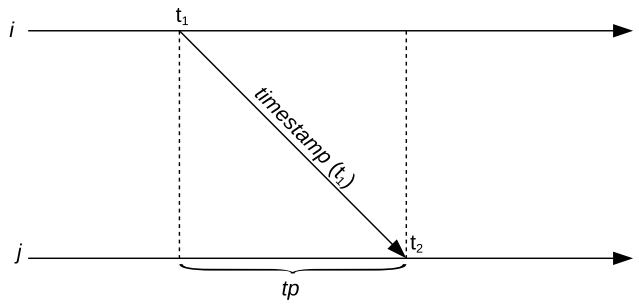

Figura 1.5: Unidirecional *pairwise syncronization*

A sincronização unidirecional ou *one-way* é a forma mais elementar de sincronização *pair-wise*, temos a Figura 1.5 que demonstra o seu funcionamento, nela observamos que o nó i no tempo t_1 envia seu *timestamp* para j, quando o nó j recebe a mensagem seu

relógio local está marcando o tempo t_2, assim o valor de t_2 pode ser calculado como na Equação (1.2):

$$t_2 = t_1 + tp + \theta \tag{1.2}$$

$$t_j - t_i = \theta + tp \tag{1.3}$$

Onde tp é o tempo de propagação, este tempo é muito pequeno podendo ser ignorado. O *delay* de propagação para uma distância de 30 m é 10^{-7}s [19]. O θ é a diferença entre os tempos de i e j, ele indica o *offset* dos relógios. Com essas informações o nó j pode calcular o *offset* e ajustar seu relógio com base no tempo de i, de acordo com a Equação (1.3).

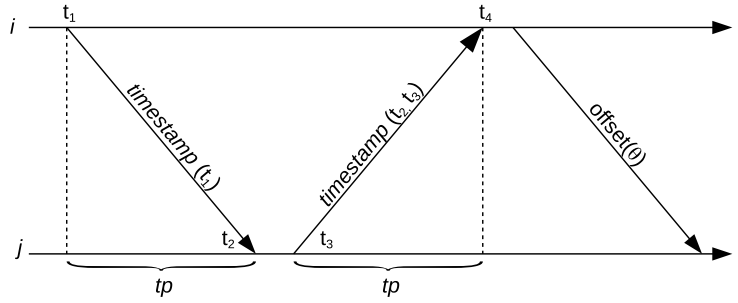

Figura 1.6: Bidirecional *pair-wise syncronization*

A sincronização bidirecional, *two-way* ou ainda *sender-receiver* usa duas mensagens de sincronização. A Figura 1.6 mostra o comportamento desta técnica, primeiro o nó i envia seu tempo t_1 para j, este recebe a mensagem no tempo t_2 e registra seu tempo local. Assim, o tempo t_2 já pode ser calculado como na Equação (1.2). No próximo passo o nó j no tempo t_3 envia uma mensagem para i com os *timestamps* t_2 e t_3, então no instante t_4 tanto o nó i como o j são capazes de calcular o *offset* entre ambos, porém o nó i é capaz de determinar mais precisamente tanto o *offset* quanto o tempo de propagação, como segue:

$$tp = \frac{[(t_2 - t_1) + (t_4 - t_3)]}{2} \tag{1.4}$$

$$\theta = \frac{[(t_2 - t_1) - (t_4 - t_3)]}{2} \tag{1.5}$$

O nó i com uma informação mais acurada, pode enviar uma nova mensagem para j

com o valor do θ. A sincronização bidirecional tem a vantagem de moderar as incertezas na pilha de protocolos e no atraso de propagação usando troca de mensagens e a desvantagem de necessitar de um número adicional de troca de trafego.

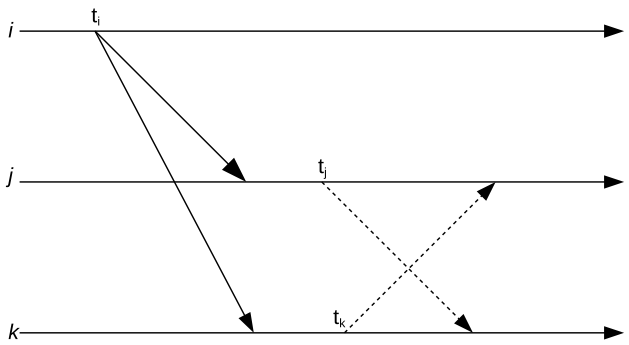

Figura 1.7: Sincronização *Receiver-receiver*

A sincronização *receiver-receiver* toma vantagem da natureza do *broadcast* na comunicação sem fio, onde a difusão de uma mensagem tem o mesmo tempo de recepção em cada um dos receptores, seu tempo de chegada varia muito pouco sendo sensível apenas aos atrasos de propagação e recebimento. No cenário descrito na Figura 1.7, temos um nó que envia *beacons* (não necessariamente *timestamp*) periodicamente, todos nós que recebem a mensagem registram o tempo de recebimento com o seu relógio local. Então, os *receivers* trocam entre si a informação registrada, o que os permite calcularem a diferença entre o tempo de chegada dos *beacons*, ou seja, o *offset*. Esse tratamento pode remover duas das maiores fontes de atraso na comunicação sem fios: o atraso de transmissão e o tempo acesso ao meio [4].

1.2 Definição do Problema

Dependência de hardware, padronização do recurso, esforço para que exista futuramente um padrão que esteja presente em todos os tipos de *motes* e possibilite assim a interoperação de protocolos de MAC *timestamp* em sistemas heterogênios.

1.3 Motivação do Trabalho

- Devido a diversidade de dispositivos e o bom desempenho do FTSP, tona-lo um protocolo independente de recurso de hardware possibilitando o seu funcionamento em *motes* sem o recurso de MAC timestamp.

1.4 Contribuição da Dissertação

- Fazer o FTSP funcionar com timestamp a nível de aplicação.

1.5 Organização do Documento

Esta dissertação é composta por seis capítulos. Descrever no final o que cada capítulo apresenta.

- **Capítulo 2:** Descrever conteúdo aqui.
- **Capítulo 3:** Descrever conteúdo aqui.
- **Capítulo 4:** Descrever conteúdo aqui.
- **Capítulo 5:** Descrever conteúdo aqui.
- **Capítulo 6:** Descrever conteúdo aqui.
- **Capítulo 7:** Descrever conteúdo aqui.

Capítulo 2

Trabalhos Relacionados

Falar sobre os protocolos abaixo:

2.1 RBS

É um método no qual o receptor usa as transmissões da camada física para comparar os relógios. Usa o tempo de chegada do pacote como tempo de referência de sincronização.

2.2 LTS

Lightweight Tree-Based Synchronization.

2.3 TPSN

TPSN não elimina incerteza no sender, apenas minimiza o erro. Ele tenta reduzir este não determinismo utilizando timestamping da mensagem na camada MAC.

2.4 PulseSync

No trabalho [22] é realizado um estudo sobre o FTSP e é observado que o protocolo apresenta um crescimento do erro de sincronização baseado no diâmetro da rede. Então propõe o protocolo PulseSync para melhorar esse cenário. Também é dependente de MAC timestamping.

2.5 FTSP

Descrição de alto nível sobre o FTSP.

Capítulo 3

Revisão detalhada do FTSP

- Explicar a técnica.
- Inserir ilustração do processo de sincronização com o relógio externo.
- Pseudo código
- FTSP+ e subsection

3.1 Flooding Time Synchronization Protocol

This section briefly describes the Flooding Time Synchronization Protocol (FTSP). For more detailed information, refer to [23].

FTSP is a synchronization protocol for WSN that provides high accuracy, consumes few resources, uses little bandwidth and is fault-tolerant. It elects a node as root to provide the time reference for synchronization; if root failure is detected (using timeouts), another root is elected. Root and synchronized nodes send synchronization messages periodically, and receiving nodes use these messages to synchronize. Therefore, FTSP supports multi-hop networks.

Synchronization messages comprise a *sender timestamp* which is the estimated global time and *rootID* which is the network identifier of the root (where the node with the lowest ID is the chosen root). *seqNum* is a sequence counter that is incremented each synchronization round; this field is used to verify the redundancy of messages [23].

All nodes think they are root when the network starts, so they broadcast synchronization messages to the network. When they receive a synchronization message, they

check who has the lowest ID: if the local ID is higher, this node gives up on being root and starts synchronizing. Another important check is the *seqNum*. If it is greater than the local value *highestSeqNum*, it means that this is a new synchronization message and starts the synchronization procedure.

The synchronization procedure consists of computing a linear regression [10] that will provide the clock skew (used to estimate the global time) in relation to the reference node. The last step is to forward its local (synchronized) time to other nodes.

```
1  event Radio.receive(TimeSyncMsg *msg){
2    if( msg->rootID < myRootID )
3        myRootID = msg->rootID;
4    else if( msg->rootID > myRootID
5      || msg->seqNum <= highestSeqNum )
6        return;
7    highestSeqNum = msg->seqNum;
8    if( myRootID < myID )
9        heartBeats = 0;
10   if( numEntries >= NUMENTRIES_LIMIT
11     && getError(msg) > TIME_ERROR_LIMIT )
12       clearRegressionTable();
13   else
14       addEntryAndEstimateDrift(msg);
15 }
```

Figura 3.1: FTSP Receive Routine [23]

Figure 1 shows the routine of receiving synchronization messages. Lines 2 and 3 compare the *rootID* of the synchronization message with the local *rootID*. If the message has a lower *rootID*, the node assumes this *rootID* as root. Lines 4 to 7 ignore messages with higher *rootID* and lower *seqNum*. If *seqNum* is higher, local *highestSeqNum* is updated. Lines 8 and 9 makes a root node give up on being root when it has a *rootID* lower than its own ID. In case the *rootID* is larger it is checked whether the *seqNum* is greater or equal to the value of *highestSeqNum*, this check prevents information redundancy because the message will only be used when the *rootID* is less than or equal to *myRootID* and the number greater than the value of *highestSeqNum*. Lines 10 to 15 verify if the time of a message is in disagreement with the earlier estimates of global time, if applicable clear the regression table, if not, accumulate synchronization messages to calculate the linear regression and synchronize.

Figure 3.2 shows the sending routine. A node decides to be root because has not

```
1  event Timer.fired() {
2    ++heartBeats;
3    if( myRootID != myID
4      && heartBeats >= ROOT_TIMEOUT )
5        myRootID = myID;
6    if( numEntries >= NUMENTRIES_LIMIT
7      || myRootID == myID ){
8        msg.rootID = myRootID;
9        msg.seqNum = highestSeqNum;
10       Radio.send(msg);
11   if( myRootID == myID )
12       ++highestSeqNum;
13   }
14 }
```

Figura 3.2: FTSP Send Routine [23]

received a synchronization message for ROOT_TIMEOUT (lines 3 to 5). A node sends synchronization messages if it is root or has synchronized (lines 6 to 10). If a node is root, it also has to increment its *highestSeqNum*.

3.2 *Timestamp*

Explicar como funciona a marcação de tempo no FTSP

3.3 Escorregamento do Relógio

Descrever os problemas relacionados ao clock drift. Como o FTSP corrige essa fonte de imprecisão.

3.4 Sincronização Multi-saltos

- Formato da mensagem de sincronização.
- Gerenciamento de informação redundante.
- Eleição do nó raiz.
- Evento: enviar/receber mensagem de sincronia.

Capítulo 4

FTSP+

Intro

- Explicar a técnica.
- Inserir ilustração do processo de sincronização com o relógio externo.
- Pseudo código
- FTSP+ e subsection

4.1 Técnica

Descrever o processo de cálculo do tempo de acesso ao meio, e onde foi baseado.

4.2 Modificação

Pseudo código com explicação do passo-a-passo do algoritmo (no sender e no receiver).

In any distributed time synchronization technique, nodes have to tell each other their local time. Figure 4.1 depicts this scenario. The sending node stores its local time $t1'$ in the synchronization message at time $t1$ and orders the message to be sent. Due to medium random access uncertainties, the sending node only gets access to the medium at time $t2$ and starts sending me message. $t2-t1$ is called *medium access time*. The message propagates over the medium for an interval of time tp until it reaches the receiver radio at time $t3$. tp is the *propagation time*. Due to interrupt handling policies and packet header processing, the receiver node timestamps the message receipt at time $t4$ with its local time $t4''$. $t4 - t3$ is the *processing time*.

Synchronization inaccuracy happens because the receiving node thinks that at time $t4$ the sender has time $t1'$ and the receiver has time $t4''$. We can see in Figure 4.1 that this is not true. At time $t4$, the sending node has time $t4' = t1' + \textit{medium access time} + \textit{propagation time} + \textit{processing time}$. MAC layer time-stamping makes *medium access time* and *processing time* equal zero. Since *propagation time* is negligible ($\sim 1\mu s$) [23], some synchronization policies — including FTSP — can achieve very good accuracy. However, without MAC layer time-stamping, these times are non-negligible and have to be calculated.

FTSP+ calculates *medium access time* using an interrupt handler to time stamp the moment that the node gets medium access to send the synchronization message. Although medium access is granted at time $t2'$, *medium access timestamp* equals $t2' + \delta$, where δ is the overhead to process the interrupt handler.

The sender sends a correction message with content $t2' + \delta - t1'$ so the receiver can estimate $t4'$. Let us call this estimation $\bar{t4}'$. The receiver calculates $\bar{t4}'$ as in Equation (4.1).

$$\begin{aligned} \bar{t4}' &= t1' - (t2' + \delta - t1') \\ \bar{t4}' &= t1' + \textit{medium access time} + \delta \end{aligned} \tag{4.1}$$

The difference between $t4'$ and $\bar{t4}'$, which is the estimation error, is:

$$t4' - \bar{t4}' = \textit{propagation time} + \textit{processing time} - \delta \tag{4.2}$$

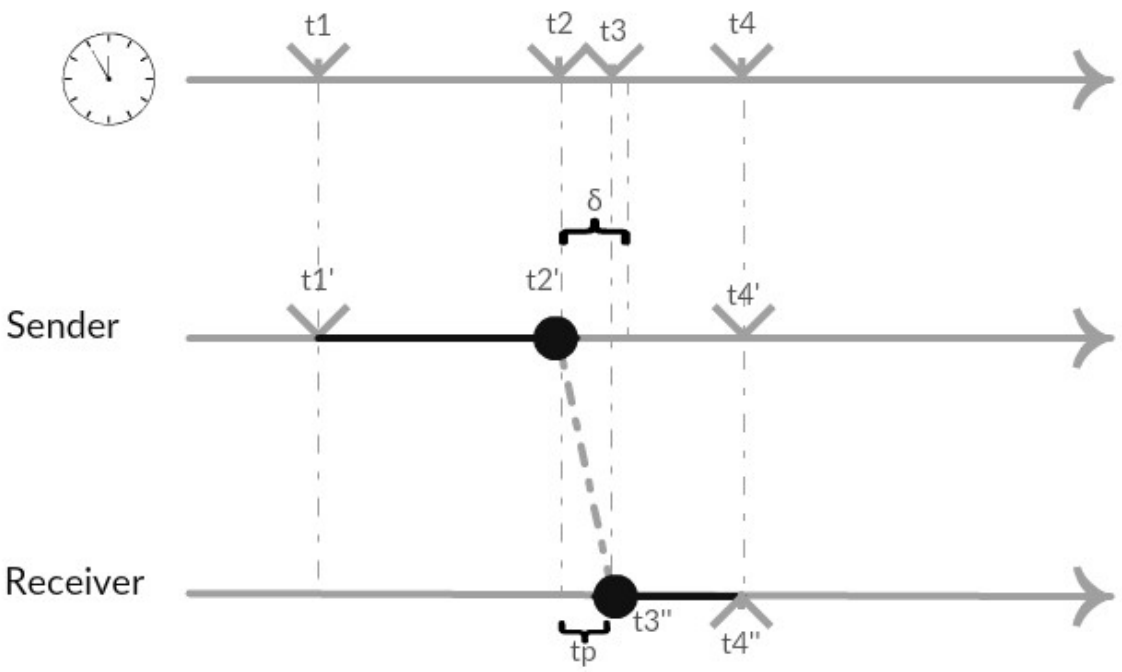

Figura 4.1: Syncronization steps.

Since *processing time* and δ are interrupt handler processing latencies, they tend to cancel out each other. We investigate their values in our experiments in Section **??**. Remember that *propagation time* is negligible.

As can be seen in Figure 4.2, an FTSP+ receiver has routines to handle two different types of income messages: `Radio.receiveSyncMsg(SyncMsg *msg)` handles synchronization messages and `Radio.receiveCorrectionMsg(CorrectionMsg *msg)` handles correction messages. The *correction time*, expressed in Equation (4.3), is the information that is transmitted to the receiver to apply the correction to previous messages.

$$correction\ time = t2' - t1' + \delta \tag{4.3}$$

`Radio.receiveSyncMsg(SyncMsg *msg)` differs from FTSP receive only for lines 13 and 14. These lines store synchronization messages in a list to wait for the correction messages if MAC layer time-stamping is not available.

`Radio.receiveCorrectionMsg(CorrectionMsg *msg)` receives the message with the correction value, use the function `applyCorrection(msg)` that applies the correction and

```
1  event Radio.receiveSyncMsg(SyncMsg *msg){
2    if( msg->rootID < myRootID )
3        myRootID = msg->rootID;
4    else if( msg->rootID > myRootID
5      || msg->seqNum <= highestSeqNum )
6        return;
7    highestSeqNum = msg->seqNum;
8    if( myRootID < myID )
9        heartBeats = 0;
10   if( numEntries >= NUMENTRIES_LIMIT
11     && getError(msg) > TIME_ERROR_LIMIT )
12       clearRegressionTable();
13   else if (MAC_Time==false){
14       addToWaitCorrectionMsgList(msg);
15   }else{
16       addEntryAndEstimateDrift(msg);
17   }
18 }
19
20 event Radio.receiveCorrectionMsg(
21                               CorrectionMsg *msg){
22   applyCorrection(msg);
23   addEntryAndEstimateDrift(msg);
24 }
```

Figura 4.2: Receiver algorithm

removes from the wait list the synchronization message that matches the incoming correction message, corrects the timestamp and calls function `addEntryAndEstimateDrift(msg)`.

```
1  event Timer.fired() {
2    ++heartBeats;
3    if( myRootID != myID
4      && heartBeats >= ROOT_TIMEOUT )
5        myRootID = myID;
6    if( numEntries >= NUMENTRIES_LIMIT
7      || myRootID == myID ){
8        msg.rootID = myRootID;
9        msg.seqNum = highestSeqNum;
10       initialTime = call getLocalTime();
11       msg.timestamp = initialTime;
12       Radio.send(msg);
13   if( myRootID == myID )
14       ++highestSeqNum;
15   }
16 }
17
18
19 event sendDone(SyncMsg *msg, error_t error){
20  finalTime = call getLocalTime();
21  correctionMsg.correction = finalTime-initialTime;
22  correctionMsg.seqNum = msg.seqNum;
23  Radio.send(correctionMsg);
24 }
```

Figura 4.3: Sender algorithm

As can be seen in Figure 4.3, an FTSP+ sender has two routines: `Timer.fired()` periodically sends synchronization messages (like in FTSP) and `sendDone(message_t *msg, error_t error)` sends the correction message.

`Timer.fired()` differs from FTSP only for lines 10 and 11, which collects the timestamp at application level. `sendDone(message_t *msg, error_t error)` handles the interrupt when wireless medium access is granted to collect the medium access timestamp by compute the time that has passed between `send/sendDone` and send out the correction message, this technique is previously discussed by [36] that is a simple technique for local delay estimation in WSN.

Lines 21-22 show how the correction time is computed. `seqNum` is used to identify which message is being adjusted, `finalTime` and `initialTime` refers to times $t2$ and $t1$

in Equation (4.3).

Capítulo 5

Implementação

Descrever o processo de Implementação no tinyos (libs, organização do código e TEPs).

5.1 Sistema Operacional

- Gerencia de memória.
- Gerenciamento energético.
- Redes.
- Linguagem.
- Manipulação de interrupções.
- Programação baseada em eventos.

Tinyos

5.2 Diagramas e Componentes

O Tinyos tem o recurso de componentes para disponibilizar seus mais diversos recursos, descrever os componentes relevantes para o trabalho e os componentes resultantes das alterações.

Capítulo 6

Experimentos

Listar o experimentos de maneira numérica 1,2 e 3. Em cada item explicar os exp em si.

- O que eu quero responder com os experimentos.
- Quais métricas utilizar.
- Definir ambiente.
- Colocar os gráficos e tabelas gerados pelos experimentos.
- Comentar e analizar os resultados.

6.1 Experimento 1

Descrever o ambiente, tipo de sensor, tempo de duração e demais caracteristicas do cenário testado.

6.2 Experimento 2

Descrever o ambiente, tipo de sensor, tempo de duração e demais caracteristicas do cenário testado.

6.3 Experimento 3

Descrever o ambiente, tipo de sensor, tempo de duração e demais caracteristicas do cenário testado.

Capítulo 7

Conclusão e Trabalhos Futuros

...

Referências

[1] Arampatzis, T.; Lygeros, J.; Manesis, S. A survey of applications of wireless sensors and wireless sensor networks. In *IEEE International Symposium on Intelligent Control, Mediterrean Conference on Control and Automation* (2005), IEEE, pp. 719–724.

[2] Baronti, P.; Pillai, P.; Chook, V. W. C.; Chessa, S.; Gotta, A.; Hu, Y. F. Wireless sensor networks: A survey on the state of the art and the 802.15.4 and zigbee standards. *Computer Communications* (2007), 1655–1695.

[3] Cardell-Oliver, R.; Smettem, K.; Kranz, M.; Mayer, K. Field testing a wireless sensor network for reactive environmental monitoring [soil moisture measurement]. In *Intelligent Sensors, Sensor Networks and Information Processing Conference, 2004. Proceedings of the 2004* (Dec 2004), pp. 7–12.

[4] Cho, H.; Kim, J.; Baek, Y. Enhanced precision time synchronization for wireless sensor networks. *Sensors 11*, 8 (2011), 7625–7643.

[5] Chong, F.; Heck, M.; Ranganathan, P.; Saleh, A.; Wassel, H. Data center energy efficiency:improving energy efficiency in data centers beyond technology scaling. *Design Test, IEEE 31*, 1 (Feb 2014), 93–104.

[6] Cuevas, Á.; Cuevas, R.; Urueñ, M.; Larrabeiti, D. A proposal for zigbee clusters interconnection based on zigbee extension devices. In *Wireless Sensor and Actor Networks*. Springer, 2007, pp. 227–238.

[7] Elson, J.; Estrin, D. Time synchronization for wireless sensor networks. In *Parallel and Distributed Processing Symposium., Proceedings 15th International* (April 2001), pp. 1965–1970.

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

[9] Elson, J.; Römer, K. Wireless sensor networks: A new regime for time synchronization. *SIGCOMM Computer Communication Review 33*, 1 (Jan. 2003), 149–154.

[10] Elson, J. E.; Estrin, D. *Time synchronization in wireless sensor networks*. Tese de Doutorado, University of California, Los Angeles, 2003.

[11] Ganeriwal, S.; Kumar, R.; Srivastava, M. B. Timing-sync protocol for sensor networks. In *1st International Conference on Embedded Networked Sensor Systems* (2003), ACM, pp. 138–149.

[12] Ganeriwal, S.; Kumar, R.; Srivastava, M. B. Timing-sync protocol for sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems* (2003), ACM, pp. 138–149.

[13] Gui, C.; Mohapatra, P. Power conservation and quality of surveillance in target tracking sensor networks. In *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking*, MobiCom '04, ACM, pp. 129–143.

[14] Hill, J.; Horton, M.; Kling, R.; Krishnamurthy, L. The platforms enabling wireless sensor networks. *Communications of the ACM 47*, 6 (2004), 41–46.

[15] Holtkamp, H. Decentralized synchronization for wireless sensor networks. *arXiv preprint arXiv:1304.0646* (2013).

[16] Horauer, M.; Schossmaier, K.; Schmid, U.; Höller, R.; Kerö, N. Psynutc-evaluation of a high-precision time synchronization prototype system for ethernet lans. Tech. rep., DTIC Document, 2002.

[17] Huang, P.; Xiao, L.; Soltani, S.; Mutka, M. W.; Xi, N. The evolution of mac protocols in wireless sensor networks: A survey. *Communications Surveys & Tutorials, IEEE 15*, 1 (2013), 101–120.

[18] Kahn, J. M.; Katz, R. H.; Pister, K. S. J. Next century challenges: Mobile networking for “smart dust”. In *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking* (New York, NY, USA, 1999), MobiCom '99, ACM, pp. 271–278.

[19] Karl, H.; Willig, A. *Protocols and architectures for wireless sensor networks*. John Wiley & Sons, 2007.

[20] Kopetz, H. *Real-time systems: design principles for distributed embedded applications*. Springer Science & Business Media, 2011.

[21] Kopetz, H.; Ochsenreiter, W. Clock synchronization in distributed real-time systems. *IEEE Transactions on Computers C-36*, 8 (Aug 1987), 933–940.

[22] Lenzen, C.; Sommer, P.; Wattenhofer, R. Optimal clock synchronization in networks. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems* (2009), ACM, pp. 225–238.

[23] Maróti, M.; Kusy, B.; Simon, G.; Lédeczi, Á. The flooding time synchronization protocol. In *Proceedings of the 2nd international conference on Embedded networked sensor systems* (2004), ACM, pp. 39–49.

[24] Mills, D. Internet time synchronization: the network time protocol. *Communications, IEEE Transactions on 39*, 10 (Oct 1991), 1482–1493.

[25] Moss, D.; Levis, P. Box-macs: Exploiting physical and link layer boundaries in low-power networking. *Computer Systems Laboratory Stanford University* (2008), 116–119.

[26] Nazemi Gelyan, S.; Eghbali, A.; Roustapoor, L.; Yahyavi Firouz Abadi, S.; Dehghan, M. Sltp: Scalable lightweight time synchronization protocol for wireless sensor network. In *Mobile Ad-Hoc and Sensor Networks*, vol. 4864. Springer Berlin, 2007, pp. 536–547.

[27] Oliveira, L. M.; Rodrigues, J. J. Wireless sensor networks: a survey on environmental monitoring. *Journal of communications 6*, 2 (2011), 143–151.

[28] Otero, J.; Yalamanchili, P.; Braun, H.-W. High performance wireless networking and weather. *High Performance Wireless Research and Education Network* (2001).

[29] Polastre, J.; Hill, J.; Culler, D. Versatile low power media access for wireless sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems* (2004), ACM, pp. 95–107.

[30] Pottie, G. J.; Kaiser, W. J. Wireless integrated network sensors. *Communications of the ACM 43*, 5 (2000), 51–58.

[31] Ranganathan, P.; Nygard, K. Time synchronization in wireless sensor networks: a survey. *International journal of UbiComp (IJU) 1*, 2 (2010), 92–102.

[32] Simon, G.; Maróti, M.; Lédeczi, Á.; Balogh, G.; Kusy, B.; Nádas, A.; Pap, G.; Sallai, J.; Frampton, K. Sensor network-based countersniper system. In *Proceedings of the 2nd international conference on Embedded networked sensor systems* (2004), ACM, pp. 1–12.

[33] Singh, S.; Raghavendra, C. S. Pamas-power aware multi-access protocol with signalling for ad hoc networks. *ACM SIGCOMM Computer Communication Review 28*, 3 (1998), 5–26.

[34] Sinopoli, B.; Sharp, C.; Schenato, L.; Schaffert, S.; Sastry, S. S. Distributed control applications within sensor networks. *Proceedings of the IEEE 91*, 8 (2003), 1235–1246.

[35] Sommer, P.; Wattenhofer, R. Gradient clock synchronization in wireless sensor networks. In *International Conference on Information Processing in Sensor Networks* (April 2009), pp. 37–48.

[36] Sousa, C.; Carrano, R. C.; Magalhaes, L.; Albuquerque, C. V. Stele: A simple technique for local delay estimation in wsn. In *Computers and Communication (ISCC), 2014 IEEE Symposium on* (2014), IEEE, pp. 1–6.

[37] Sundararaman, B.; Buy, U.; Kshemkalyani, A. D. Clock synchronization for wireless sensor networks: a survey. *Ad Hoc Networks 3*, 3 (2005), 281–323.

[38] Suriyachai, P.; Roedig, U.; Scott, A. A survey of mac protocols for mission-critical applications in wireless sensor networks. *Communications Surveys & Tutorials, IEEE 14*, 2 (2012), 240–264.

[39] van Greunen, J.; Rabaey, J. Lightweight time synchronization for sensor networks. In *Proceedings of the 2Nd ACM International Conference on Wireless Sensor Networks and Applications* (2003), ACM, pp. 11–19.

[40] Wang, L. *Topology-Based Routing for Xmesh in Dense Wireless Sensor Networks*. ProQuest, 2007.

[41] Wang, S.; Ahn, T. Mac layer timestamping approach for emerging wireless sensor platform and communication architecture, Dezembro 2009. US Patent App. 12/213,286.

[42] Yang, H.; Sikdar, B. A protocol for tracking mobile targets using sensor networks. In *Sensor Network Protocols and Applications, 2003. Proceedings of the First IEEE. 2003 IEEE International Workshop on* (2003), IEEE, pp. 71–81.

[43] Ye, W.; Heidemann, J.; Estrin, D. An energy-efficient mac protocol for wireless sensor networks. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE* (2002), vol. 3, IEEE, pp. 1567–1576.

[44] Zanatta, G.; Bottari, G. D.; Guerra, R.; Leite, J. C. B. Building a WSN infrastructure with COTS components for the thermal monitoring of datacenters. In *Symposium on Applied Computing, SAC 2014, Gyeongju, Republic of Korea - March 24 - 28, 2014* (2014), pp. 1443–1448.