UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

LEANDRO TAVARES BRUSCATO

PROPOSTA DE MÉTODOS DE SINCRONIZAÇÃO DE REDE DE SENSORES SEM FIO

Porto Alegre
2017

LEANDRO TAVARES BRUSCATO

PROPOSTA DE MÉTODOS DE SINCRONIZAÇÃO DE REDE DE SENSORES SEM FIO

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica, da Universidade Federal do Rio Grande do Sul, como parte dos requisitos para a obtenção do título de Mestre em Engenharia Elétrica.

Área de concentração: controle e automação

ORIENTADOR: Edison Pignaton de Freitas

Porto Alegre

2017

LEANDRO TAVARES BRUSCATO

PROPOSTA DE MÉTODOS DE SINCRONIZAÇÃO DE REDE DE SENSORES SEM FIO

Esta dissertação foi julgada adequada para a obtenção do título de Mestre em Engenharia Elétrica e aprovada em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: _____

Prof. Dr. Edison Pignaton de Freitas, UFRGS

Doutor pela Universidade de Halmstad, Suécia e pela Universidade Federal do Rio Grande do Sul– Porto Alegre, Brasil

Banca Examinadora:

Prof. Dr. Walter Fetter Lages, UFRGS

Doutor pelo Instituto Tecnológico de Aeronáutica – São José dos Campos, Brasil

Prof. Dr. João Cesar Netto, UFRGS

Doutor pela Université Catholique de Louvain– Louvain-la-Neuve, Bélgica

Prof. Dr. Raul Ceretta Nunes, UFSM

Doutor pela Universidade Federal do Rio Grande do Sul – Porto Alegre, Brasil

Coordenador do PPGEE: _____

Prof. Dr. Valner João Brusamarello

Porto Alegre, Maio 2017.

AGRADECIMENTOS

Ao Professor Edison Pignaton De Freitas, pela paciência na orientação; oportunidade de trabalhar nesta área de pesquisa; e incentivo que tornaram possível a conclusão desta dissertação.

Aos meus pais, irmãos e a toda minha família que, com muito carinho e apoio, não mediram esforços para que eu chegasse até esta etapa de minha vida, que entenderam todas as vezes que não tive presente em suas reuniões.

Ao Programa de Pós-Graduação em Engenharia Elétrica, PPGEE, pela oportunidade de realização de trabalhos em minha área de pesquisa.

A bolsista, Luciana Tubello Caldas, pelo amparo na redação deste trabalho.

Aos amigos e colegas, pelo incentivo e pelo apoio constantes.

E principalmente a minha esposa, Rafaela Salazar Sikilero Bruscato, pela sua enorme paciência, por todo seu apoio e carinho que tornaram todos os dias mais fáceis.

RESUMO

Os sistemas de monitoramento de ambientes vêm ganhando cada vez mais espaço, conceitos como cidades inteligentes e Internet das coisas demonstram que esta tecnologia está evoluindo em larga escala. Consequentemente, diversas aplicações desta tecnologia estão sendo desenvolvidas, muitas delas são altamente dependentes de redes de sensores sem fio, pois fazem a coleta de dados em ambientes inóspitos ou de difícil acesso. Para que estes dados coletados de diversos dispositivos possam ser analisados em conjunto é preciso que as coletas sejam simultâneas ou em intervalos próximos de tempo, o que implica que toda a rede de sensores tenha uma elevada precisão no sincronismo. Ademais, diversos protocolos de comunicação sem fio utilizam o sincronismo para estabelecer o compartilhamento do meio de propagação, tendo assim uma maior eficiência na troca de dados. Ao observar a importância de atender a essa necessidade de sincronização de tempo entre dispositivos usados em redes de sensores sem fio, este trabalho se concentra na proposta, implementação e teste de um serviço de sincronização de tempo para redes de sensores sem fio de baixa potência usando relógios de baixa frequência em tempo real em cada nó. Para implementar este serviço, são propostos três algoritmos baseados em estratégias diferentes para alcançar a sincronização desejada. O primeiro baseia-se em uma métrica simples de correção adaptativa; o segundo baseia-se em um mecanismo de predição; já o terceiro utiliza um mecanismo mais complexo, a correção analítica. Todos os algoritmos tem o mesmo objetivo: fazer com que os relógios dos nós sensores convirjam de forma ágil, em seguida, mantê-los com a maior similaridade possível. O objetivo deste trabalho é apresentar o melhor método que garanta a sincronização, mantendo o baixo consumo de energia em uma rede de sensores. Os resultados experimentais fornecem evidências do sucesso no cumprimento deste objetivo, bem como fornece meios para comparar estas três abordagens considerando os melhores resultados de sincronização e os seus custos em termos de consumo de energia.

Palavras-chave: Internet das coisas. Sensores de Baixo Consumo. Sincronismo Temporal. Predição de *clock*. Relógio de Tempo Real. Rede de Sensores Sem Fio.

ABSTRACT

Environmental monitoring systems are gaining more and more space, concepts such as smart cities and the Internet of things demonstrate that this technology has been developing a lot. Consequently, many applications of this technology are being developed, many of them are dependent on wireless sensor networks, which collect data in inhospitable or difficult-to-access environments. In order to these collected data from several devices to be analyzed together it is necessary that the data collection be simultaneous or at close intervals, which implies that the entire network of sensors has a high precision in the synchronism. In addition, several wireless communication protocols use the synchronism to establish sharing of medium networks, thus having a greater efficiency in the exchange of data. Observing the importance of time synchronization, this work focuses in proposing, implementing and testing time synchronization protocols for low power wireless sensor networks using real time low frequency clocks. To implement this service, three algorithms based on different strategies are proposed to achieve the desired synchronization. The first is based on the simple metric for self-correction; the second is based on a prediction mechanism; while the third uses a more complex mechanism for analytical correction. All the algorithms have the same goal: to make the clock of the sensor nodes converge in an agile way, then to keep them with the greatest possible similarity. The objective of this work is to present the best method to guarantee the synchronization, keeping the low power consumption in a network, sporadically, transmissions. The experimental results provide evidence of success in achieving this goal, as well as providing means to compare these three approaches considering the best synchronization results and their costs in terms of energy consumption. Keywords: Internet of things.

Keywords: Low Power Sensors. Synchronization. Clock prediction. Real Time Clock. Wireless Sensor Network.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ATS: *Average TimeSync*

BFS: *Breadth-First Search*

CCS: *Continuous Clock Synchronization*

DMTS: *Delay Measurement Time Synchronization*

EB: Estação Base

FCSA: *Flooding With Clock Speed Agreement*

FHSS: *Frequency Hopping Spread Spectrum*

FLL: *Frequency Locked Loop*

FTSP: *Flooding Time Synchronization Protocol*

GPS: *Global Positioning System*

IEEE: *Institute of Electrical and Electronics Engineers*

IoT: *Internet of Things*

IP: *Internet Protocol*

ISO: *International Organization for Standardization*

LAN: *Local Area Network*

LLC: *Logical Link Control*

MAC: *Medium Access Control*

NTP: *Network Time Protocol*

OSI: *Open System Interconnection*

PHY: Camada Física do Modelo OSI

PLL: *Phase Locked Loop*

PPGEE: Programa de Pós-Graduação em Engenharia Elétrica

PPM: Parte Por Milhão

RBM: *Reference Broadcast Message*

RBS: *Reference-Broadcast Synchronization*

RF: Rádio Frequência

RN: *Reference Node*

RSSF: Rede de Sensores Sem Fio

re: *Real-Time Clock*

SCTS: *Self-Correcting Time Synchronization*

TDMA: *Time Division Multiple Access*

TDP: *Time-Diffusion Synchronization Protocol*

TM: *Time Master*

TPSN: *Timing-sync Protocol for Sensor Networks*

UFRGS: Universidade Federal do Rio Grande do Sul

UTC: *Coordinated Universal Time*

VoIP: Voz sobre IP

WLAN: *Wireless Local Area Network*

1 Introdução

Nas últimas décadas houve um avanço tecnológico que permitiu uma evolução no setor das comunicações. Esta melhora resulta em parte do desenvolvimento da microeletrônica, e colabora no melhor atendimento dos requisitos tecnológicos da indústria – que busca maior eficiência e qualidade na produção e, concomitantemente, redução do esforço e da ação humana sobre os processos e máquinas, muitas vezes através da automação e da comunicação sem fio.

Nas últimas décadas, os sistemas de comunicação sem fio têm passado por constantes melhorias que acarretaram na criação de diversos dispositivos focados no baixo custo e consumo. Estes dispositivos são utilizados, hoje em dia, nas redes de sensores sem fio (RSSF) e, em muitas aplicações, esta tecnologia se tornou a melhor escolha. Os aparelhos da RSSF operam em conjunto com outros equipamentos, normalmente são alocados em áreas não muito acessíveis e por esse motivo é fundamental que esta rede seja de baixo consumo, tendo assim uma alta autonomia. A RSSF apresenta diversas soluções para os ambientes fabril, militar, residencial, hospitalar (SUNDARARAMAN; BUY; KSHEMKALYANI, 2005), vigilância, assistência em navegação (LI et al., 2013), Internet das coisas (IoT) (ATZORI; IERA; MORABITO, 2010) e para muitas outras aplicações. Os principais benefícios, comparativamente com as redes cabeadas, são menores custos de instalação, manutenção, além do ganho em escalabilidade e flexibilidade.

Uma tecnologia que cada vez mais ganha espaço é a IoT, normalmente utiliza o ar como meio físico para propagar suas informações (ATZORI; IERA; MORABITO, 2010) –

ambiente em que é comum haver diversos dispositivos transmitindo dados, deixando clara a necessidade de estabelecer uma coordenação de tais transmissões. Para que haja uma coordenação nas transmissões dos dispositivos, o sincronismo apresenta-se como uma importante necessidade, pois através de uma rede sincronizada é possível estabelecer uma coordenação nas transmissões dos dispositivos, reduzindo as colisões e transmissões simultâneas. Através do sincronismo, também, podemos reduzir o consumo energético sem que haja perda na amplitude do sistema, muito pelo contrário, ele aumenta o tempo de vida da rede, o que contribui para a escalabilidade do sistema, pois quanto mais amplo o sistema maior será a coleta de dados e maior será o número de transmissões.

Observando a importância da sincronização temporal nas RSSF, este trabalho propõe o estudo das principais causas dos erros de sincronização e o estudo dos métodos existentes para manter o sincronismo de uma rede. Atualmente, os principais protocolos industriais de comunicação sem fio – como ISA100.11 WirelessHART e Zigbee – utilizam o sincronismo em sua arquitetura (NIXON; ROUND ROCK, 2012) (SKRZYPCZAK; GRIMALDI; RAK, 2009). Baseado neste estudo, este trabalho apresenta uma contribuição à área de redes de sensores sem fio através da proposta técnicas de sincronismo para dispositivos de baixo consumo utilizados em tais redes.

1.1 MOTIVAÇÃO

As aplicações que visam correlacionar eventos previstos para redes de sensores, em sua maioria, requerem manter correlações temporais, em tempo real, entre os nós. Em geral, as leituras dos nós sensores utilizam várias técnicas de processamento de sinais para obter

resultados significativos a partir destes dados brutos. Por exemplo, os aplicativos de rastreamento de destino que utilizam o filtro de Kalman para estimar a posição alvo (ROUMELIOTIS; BEKEY, 1997). Tais técnicas de processamento de sinal requerem a sincronização relativa entre os relógios dos nós de sensor. O tempo de ocorrência de certo evento é em si um parâmetro crítico, para uma gama de aplicações, como detecção de queimadas, fugas de gás, etc., desta forma a sincronização da rede com cada nó mantendo uma escala de tempo global única torna-se primordial.

Além disso, o sincronismo apresenta diversas vantagens, como fazer transmissões agendadas e a ordenação de mensagens. As transmissões agendadas auxiliam na redução das colisões, haja visto que cada dispositivo terá um intervalo de tempo para transmitir sua mensagem. Um protocolo que utiliza esta característica é o TDMA, do inglês *Time Division Multiple Access*, que divide o tempo em slot e cada dispositivo tem o direito de fazer a transmissão em seu respectivo intervalo de tempo. Desta forma, é possível alocar esses slots em links, determinando quem é a fonte e o destinatário da mensagem. Com isso, é possível agendar a tarefa de transmissão para o aparelho fonte e a tarefa recepção para o nodo destinatário, desativando os demais aparelhos e, consequentemente, melhorando o consumo energético. Como pode-se observar na Tabela 1, a atividade de receber e transmitir é a mais onerosa de todas as executadas por um nó sensor – consumindo mais que 4 vezes do modo ativo e mais de 20 vezes do modo ocioso (FRESCALE, 2012).

Tabela 1 Consumo de energia dos aparelhos da família MC 1322x

Modo	Consumo máximo
Reset	0.3 uA
Hibernando	1 uA
Ocioso	0.9mA
Ativo	5mA
Recebendo	20mA
Transmitindo	20mA

Fonte: (FRESCALE, 2012).

Como já citado, outra vantagem do sincronismo é a ordenação de tarefas, mas é importante frisar que é impossível ter uma ordenação total de tarefas em um sistema distribuído, como o deste trabalho, por conta do atraso entre os dispositivos. É fundamental, para o funcionamento de diversos sistemas, que se tenha ao menos uma ordenação parcial, pois beneficia a fusão de dados e dificulta a produção de resultados discrepantes dos processos executados nestes sistemas.

1.2 OBJETIVO

O objetivo deste trabalho é encontrar um método de sincronismo – que consiga manter a rede com os tempos mais similares possíveis e com a menor frequência de transmissão das mensagens de sincronismo, equacionando ambas as variáveis para encontrar a melhor opção. Para isso, foram estudados diversos métodos de sincronismo, entendendo suas diferenças e selecionando suas melhores características, objetivando desenvolver o método que garanta uma divisão temporal que facilite a troca de mensagens na rede e que alcance similaridade dos relógios para estabelecer correlações temporais de eventos em tempo real para uma rede de sensores sem fio de baixo consumo – utilizando componentes de baixa precisão. Após escolher as melhores soluções, foram simulados, implementados e testados estes métodos, a fim de determinar o que alcance coerência temporal com a menor frequência de mensagens de sincronismo.

1.3 ESTRUTURA DO TRABALHO

Este trabalho é constituído por 8 capítulos, sumarizados a seguir. O Capítulo 1 apresenta uma descrição geral do cenário atual das redes de sensores sem fio, sua importância no controle e na atuação dos dispositivos que em conjunto com a sua versatilidade são colocados como as principais motivações para o desenvolvimento deste trabalho. O Capítulo 2 aponta os conceitos teóricos de sincronismo de dispositivo, incluindo os desafios de estabelecer o sincronismo em uma RSSF, apresentando mecanismo para se alcançar este sincronismo e elementos para determinar a defasagem temporal entre os pares. O Capítulo 3 é

dedicado à revisão bibliográfica acerca das metodologias existentes na atualidade que abordam o sincronismo de uma rede, dando enfoque as de sensores sem fio. É efetuada uma análise dos métodos existentes, de modo a destacar as contribuições de cada modelo. O Capítulo 4 apresenta um conjunto de métodos de sincronismo para uma RSSF que sejam de baixo consumo, que ocorram transmissões esporadicamente. Apresentando suas principais diferenças e como eles se comportariam idealmente. O Capítulo 5 expõe a estrutura e as técnicas utilizadas para a implementação dos métodos apresentados no capítulo anterior. O Capítulo 6 é dedicado a comparar os métodos propostos, tanto utilizando um mecanismo de simulação que visa comparar da forma mais justa, quanto utilizando ambientes de testes reais que fazem a coleta de dados em dispositivos reais testados um ambiente controlado. As conclusões acerca do melhor modelo proposto, utilizando a similaridade temporal e a eficiência energética como parâmetros desta comparação, são apresentadas no Capítulo 7.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentadas as características de uma rede, dando ênfase às redes de sensores sem fio, evidenciando em quais aspectos esta tecnologia se diferencia das demais – usando como base a rede entre computadores. Também, faremos um breve estudo sobre métodos de sincronismo, bem como das principais fontes de erro de sincronismo.

2.1 REDES DE SENSORES SEM FIO

Os sistemas distribuídos utilizam redes como meio de estabelecer comunicação. Aspectos como escalabilidade, confiabilidade, desempenho, mobilidade e qualidade de serviço afetam diretamente o projeto de cada rede. Redes de sistemas distribuídos são construídas a partir de uma variedade de meios de transmissão, incluindo a tecnologia cabeada como fibra óptica, Ethernet e tecnologia sem fio. Fazem parte desta rede os componentes de *Hardware* como: Roteadores, *Switches*, pontes, *hubs*, repetidores e interfaces de rede; e componentes de S*oftware* como: Pilhas de protocolos e drivers (COULOURIS; DOLLIMORE; KINDBERG, 2012).

Independente do crescimento futuro dos dispositivos de Internet sem fio, está claro que as redes sem fio e os serviços relacionados à mobilidade, como WLAN e Redes de celulares móveis, estão aqui para ficar (KUROSE, 2013). Rede sem fio refere-se a uma infraestrutura de comunicação que permite a transmissão de dados e informações sem a necessidade do uso de cabos. Isso é possível graças ao uso, por exemplo, de equipamentos de radiofrequência (comunicações via ondas de rádio), de comunicações via infravermelho etc. É

conhecido também pelo anglicismo *wireless network*. O sucesso desta tecnologia deve-se a facilidade na sua instalação, manutenção e mobilidade. Protocolos como IEEE 802.11 (WiFi), Bluetooth, e em telefonia móvel (3G e 4G), apresentam estas características.

O avanço da microeletrônica permitiu a integração de capacidades de sensoriamento, processamento e comunicação sem fio em um único dispositivo de baixo custo e de pequenas dimensões, denominados nós sensores (SOHRABI et al., 2000). Além disso, com a necessidade de monitoramento e a atuação remota, tornaram o desenvolvimento das RSSF inevitável. O baixo custo dos aparelhos possibilitou a elaboração de redes de sensores com um número cada vez maior de nós, fato que implica a necessidade destas redes em serem escaláveis – ocasionando uma maior coleta de informações do ambiente. Outra vantagem é sua mobilidade, posto que esta rede é sem fio. Por se tratar de uma rede que contém, na maioria dos casos, um grande conjunto de dispositivos, é imprescindível que todos os nós tenham a capacidade de transmitir quaisquer informações dos seus nós adjacentes. Portanto, o roteamento, o gerenciamento de energia e a propagação de informação são questões essenciais para RSSF, tendo seus protocolos projetados para privilegiar estas funcionalidades (AL-KARAKI; KAMAL, 2004).

As RSSF têm por objetivo transformar medições em dados relevantes, evidenciando qualquer fenômeno localizado em sua área circunscrita. Normalmente, as RSSF, contêm centenas ou milhares de nós sensores, que tem a capacidade de se comunicar tanto entre si quanto entre uma base externa. A Figura 1 componentes de uma RSSF apresenta a arquitetura de comunicação de uma RSSF e seus componentes. Nela, os nós sensores são espalhados em

uma região e se coordenam visando produzir informações de alta qualidade, sobre o ambiente. Cada componente faz o sensoriamento, processamento e transmissão de suas informações. Cada nó sensor toma suas decisões sobre a missão, as informações e o conhecimento de seus recursos computacionais. Todos os nós tem a capacidade de recolher e encaminhar os dados, tanto para outros sensores adjacentes quanto para a estação base (EB). Esta EB pode ser fixa ou móvel, capaz de se conectar à rede de sensores, ao mesmo tempo em que se comunica a uma infraestrutura externa como a internet (AL-KARAKI; KAMAL, 2004).

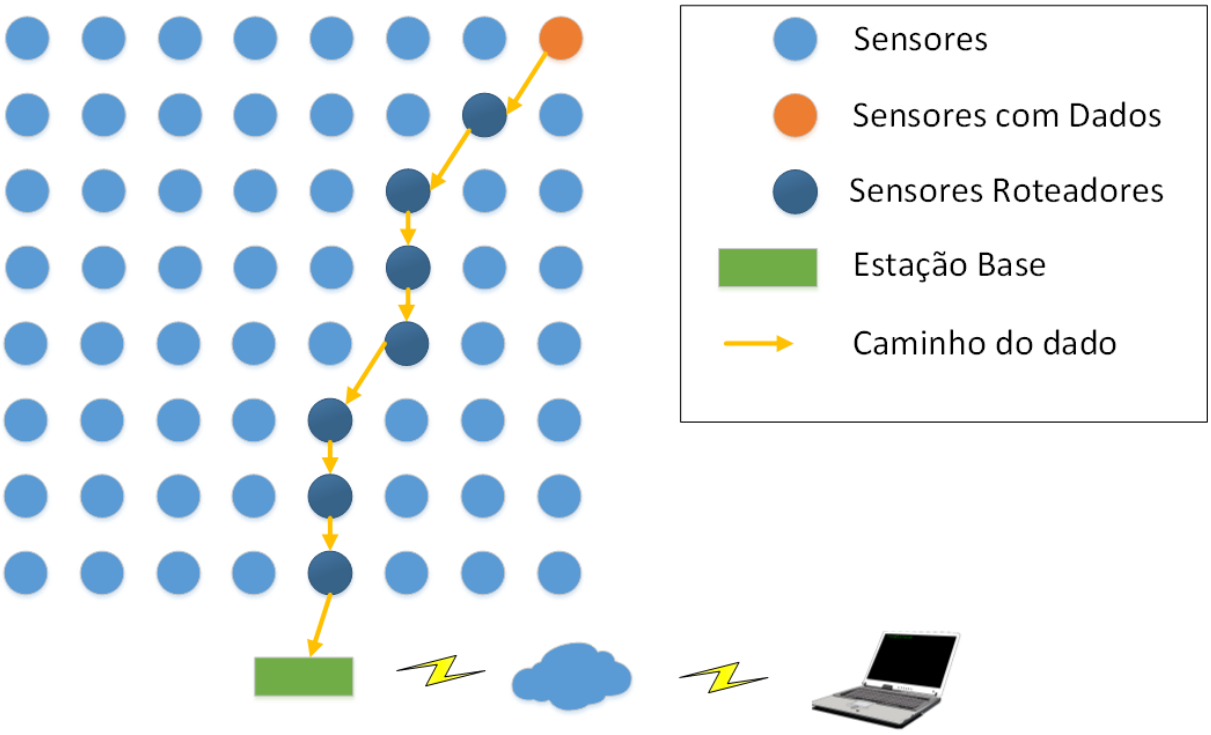

Figura 1 componentes de uma RSSF

2.1.1 Modelo OSI no contexto de RSSF

O modelo OSI (*Open Systems Interconnection*) baseia-se em uma proposta desenvolvida pela ISO (*International Standards Organization*) como primeiro passo em direção à padronização internacional dos protocolos empregados nas diversas camadas de rede. É chamado de modelo de referência OSI, pois trata da interconexão de sistemas abertos à comunicação com outros sistemas, também, segmentados em camadas.

2.1.2 Camada de Aplicação

Aplicações de rede são as razões de existir de uma rede de computadores. Estas incluem aplicações baseadas em texto (e-mail, acesso remoto a computadores); mensagens instantâneas; compartilhamento P2P; Voz sobre IP (VoIP); redes sociais; distribuição de vídeo etc. (KUROSE, 2013). Estas aplicações são o motivo do sucesso da Internet, tornando-a uma parte integrante na vida das pessoas em suas atividades diárias. As aplicações em uma RSSF comtemplam diversas funcionalidade, segundo (STOJMENOVIC, 2005), como: na indústria, onde são extremamente uteis nos monitoramentos de processos. Com as informações coletadas de diversos sensores é possível otimizar processos, fato que gera uma produção mais eficiente e de maior qualidade, reduzindo o custo da empresa com mão de obra. No ambiente militar, as RSSF podem ser usadas tanto para controles logísticos internos quanto para o monitoramento do inimigo. Fornecendo informações que viabilizam melhores decisões, desta forma salvando vidas. Na medicina, acredita-se que nos próximos anos dados como a pressão arterial, o batimento cardíaco, o nível de oxigênio no sangue e muitos outros, poderão ser monitorados por sensores que compõem uma RSSF. Assim, os médicos poderão fornecer um diagnóstico mais preciso e sem a necessidade de estar perto do paciente. As redes

de sensores sem fio são usadas na engenharia civil para monitorar a condição de determinadas estruturas, verificando a ocorrência de danos a uma ponte ou prédio. Nesse caso, os sensores podem ser instalados sobre a superfície de uma construção. Em alguns casos esta tecnologia pode ser muito útil, como em estruturas cuja falha pode resultar em algo catastrófico, usinas nucleares e hidrelétricas; ou em países com grande incidência de terremotos, como o Japão. Atualmente, existem pesquisas no desenvolvimento de sensores que podem se misturar ao cimento, visando fornecer dados ainda mais precisos (STOJMENOVIC, 2005). De maneira geral, as aplicações que podem ser utilizadas as RSSF, são aquelas que necessitam de sistemas que possam medir o ambiente e com isto gerar dados válidos.

2.1.3 Camadas de Sessão e Apresentação

Para a rede de sensores sem fio estas duas camadas são incorporadas na camada de aplicação para simplificar o processamento de dados, dado que os dispositivos desta rede apresenta, sem sua maioria, pequena capacidade de processamento.

2.1.4 Camada de Transporte

Residindo entre as camadas de aplicação e de rede, a camada de transporte é uma peça central da arquitetura de rede em camadas. Ela tem o papel fundamental de fornecer uma comunicação lógica e direta entre os processos de aplicação que se executam em diferentes hospedeiros. Comunicação lógica é uma perspectiva de aplicação, como se os processos que rodam em máquinas diferentes estivessem diretamente ligados, mas na realidade estes computadores podem estar em lados opostos do planeta – ligados através de numerosos

roteadores (KUROSE, 2013). Nas RSSF, esta camada possui a mesma finalidade que nas redes comuns. Ela executa as funções de segmentação dos pacotes, ordenação dos pacotes e a garantia de entrega (AKYILDIZ et al., 2002).

2.1.5 Camada de Rede

A camada de rede é responsável por encaminhar pacotes de roteadores à outros roteadores, a fim de que a mensagem enviada chegue ao destinatário final, estabelecendo uma comunicação fim-a-fim. Para que esta comunicação ocorra é preciso encontrar um caminho para que os pacotes sigam. Desta forma estabelecer o endereçamento lógico para que a troca de mensagens ocorra entre os roteadores. Nas redes convencionais o principal protocolo utilizado é o IP, que é encarregado de fornecer o endereçamento dos dispositivos. Neste caso, a camada de rede é responsável por encontrar um caminho para a mensagem enviada (KUROSE, 2013). Nas RSSF, esta camada foca em reduzir o número de transmissões na rede, valendo-se da confluência de dados, quando não prejudica o esforço colaborativo dos nós de sensores. Ao utilizar esta técnica – confluência de dados – ela otimiza o consumo energético. Além disso, esta camada se preocupa em estabelecer uma malha de caminhos redundantes nas redes de sensores sem fio (AKYILDIZ et al., 2002).

2.1.6 Camada Enlace de Dados

A camada de enlace faz a ligação entre dois dispositivos (*hots*) adjacentes. Entre eles, o datagrama percorre uma série de *links* de comunicação – desde conexões sem fio até conexões transoceânicas –, começando no *host* de origem e passando por uma série de

computadores, até chegar no *host* de destino. Desta forma, a camada de enlace se preocupa com as ligações individuais que compõem o caminho de uma comunicação fim-a-fim (KUROSE, 2013). A camada de enlace é responsável pelo fluxo de dados, meio de acesso e controle de erros dos datagramas, garantindo uma conexão ponto-a-ponto ou ponto a múltiplos pontos (AKYILDIZ et al., 2002). Segundo Mullett (2005), os protocolos da família IEEE 802.x.x – que incluem os protocolos IEEE 802.11 (WiFi), IEEE 802.15 (Bluetooth) e 802.15.4 (alvo deste trabalho) – são restritos a redes que transportam pacotes de tamanho variável, onde a camada de enlace é compreendia como duas subcamadas LLC e MAC, do inglês *Logical Link Control* e *Medium Acess Control*, como a Figura 2 Camadas OSI. O LLC é um protocolo de comunicação de dados que especifica o mecanismo para o endereçamento de pacotes das estações, visando controlar a troca de dados na rede. Já o MAC fornece o conceito de uma rede, gerencia o Superframe, controla o acesso ao canal, valida quadros e envia a confirmação de recebimento (*acknowledgement*) para dispositivos a um salto de distância – visto que o MAC não contempla múltiplos saltos.

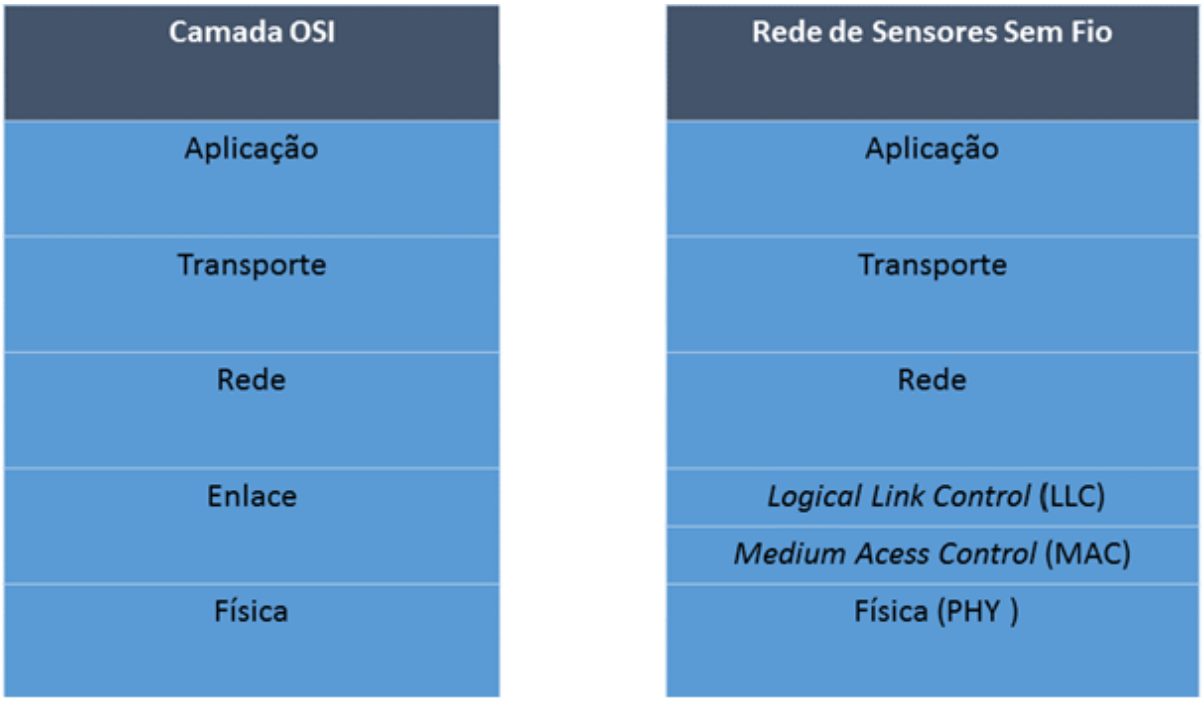

Figura 2 Camadas OSI

2.1.7 Camada Física

A camada física trata da transmissão de *bits* brutos via canal de comunicação. Ao se projetar a rede, é preciso preocupar-se com a representação dos *bits*; com o intervalo de tempo na transmissão de um *bit*; com a possibilidade de haver transmissões simultâneas ou não, *full-duplex* ou *half-duplex*; com a forma em que transmissão se inicia e se encerra; com a quantidade de pinos que o conector deverá ter e a finalidade de cada pino, no caso das redes cabeadas; com a largura de banda que a transmissão ocorrerá; com a quantidade de *bauds* (símbolos) que podem ser transmitidos; e, por fim, com o ruído máximo aceitável (KUROSE, 2013) (TANENBAUM, 2003).

Para o protocolo IEEE 802.15.4 a camada física, também conhecida como PHY, utiliza o ar como meio de propagação que transmite seus dados através do sinal de rádio frequência. Este protocolo é segmentado em três faixas de frequência, como a Figura 3. Onde a primeira corresponde as faixas de 868.0 a 868.6 MHz, a segunda de 902 a 928 MHz e a terceira de 2400 a 2483.5 MHz. A única faixa que pode ser usada em todo o mundo, e alvo deste trabalho é a terceira que apresenta 16 canais com largura de cada canal de 2 MHz e uma distância de 5 MHz entre eles. Esta última faixa de transmissão possibilita uma taxa de 250 kbps e modulação O-QPSK, transmitindo assim 62500 símbolos por segundo (IEEE, 2009).

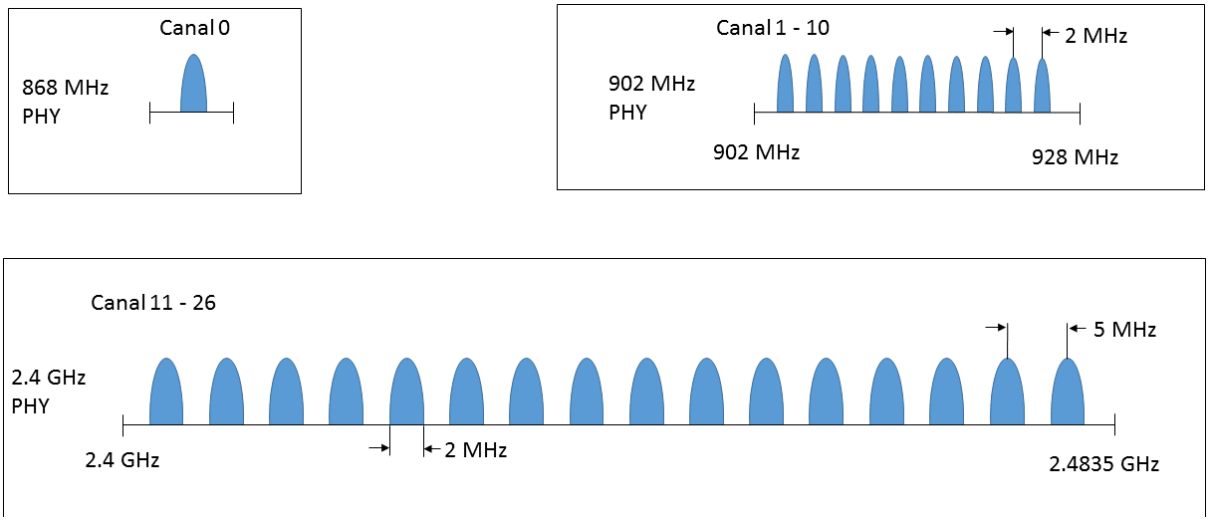

Figura 3 Canais do protocolo IEEE 802.15.4

2.2 DESAFIO EM UMA REDE DE SENSORES SEM FIO

As redes de sensores sem fio tem um grande potencial – devido sua capacidade de escalabilidade, de baixo consumo e muitas outras – que possibilita expandir a quantidade de dispositivos, aumentando o monitoramento e a integração remota com o mundo físico. Sensores inteligentes tem a habilidade de coletar grandes quantidades de dados, facilitando a criação de novos sistemas nas áreas em que as RSSF podem ser usadas. No entanto, para explorar todo o seu potencial, é preciso resolver algumas dificuldades:

1) Limite energético: mesmo que a eficiência energética de computadores esteja aumentando rapidamente, o consumo de energia em uma rede de sensores sem fio ainda é um grande empecilho para o desenvolvimento desta tecnologia. Devido ao baixo custo e as pequenas dimensões, dos dispositivos que compõe estra rede, eles possuem baterias de baixa capacidade. Como uma das principais vantagens de uma rede de sensores sem fio é a mobilidade dos dispositivos, não é uma opção válida conectar esses aparelhos à rede de distribuição de energia elétrica. Além disso, a necessidade de operações não tripuladas estipula que estes aparelhos devam ser alimentados via bateria, logo a eficiência energética é um dos principais desafios dessa tecnologia, visto que a quantidade de energia disponível é bastante restrita.

2) Limite da largura de banda: em uma rede de sensores sem fio o consumo de energia para transmissão de dados é menor que dos dispositivos comuns. Atualmente, a comunicação destes dispositivos sem fio pode transmitir seus dados em uma taxa de 10-250 Kbit/s. Esta limitação afeta diretamente a transmissão de mensagens entre os sensores, posto que o

sincronismo depende desta comunicação, a presente limitação está diretamente relacionada à pesquisa aqui proposta (FRESCALE, 2012).

3) Limite do alcance da rede: a comunicação entre dispositivos móveis é bastante limitada, comumente com transmissões de no máximo 50 metros em ambientes *indoor* e 100 metros para ambientes *outdoor* (FRESCALE, 2012). Essas curtas distâncias tornam as trocas de mensagens, entre os nós distantes, mais difíceis. Desta forma, necessitam utilizar outros nodos como repetidores, para que a rede tenha um maior alcance, aumentando assim o número de transmissões em toda a rede.

4) Limitação de *hardware:* devido as pequenas dimensões dos aparelhos da tecnologia RSSF, o *hardware* destes dispositivos é geralmente muito restrito. Um dispositivo da NIVIS do modelo VersaNode 210 tem um processador ARM7 de 32 bits, CPU que opera normalmente a 2 MHz mas pode alcançar 26 MHz, memórias RAM que operam a 24 MHz e memória *flash* de 128 Kbytes, 96 Kbytes de SRAM e de 80 Kbytes ROM (CDS, 2016). Portanto, as restrições computacionais e o espaço de armazenamento representam um desafio para os desenvolvedores desta plataforma.

5) Conexões de rede instáveis: como a comunicação é transmitida pelo ar, o meio de propagação da informação não é blindado e com isso a interferência externa aumenta o percentual da taxa de perdas de mensagens. Assim, a conectividade intermitente de uma comunicação é outro desafio para os desenvolvedores, pois esta intermitência faz com que a topologia da rede mude frequentemente e sua reconfiguração torna-se necessária, tendo que constantemente estabelecer novas rotas de comunicação. O problema se agrava, ainda mais, com a mobilidade dos nodos.

6) Acoplamento entre os sensores e o mundo físico: RSSF procuram monitorar fenômenos do mundo real. As RSSF são usadas para aplicações críticas como rastreamento militar, acompanhamento de incêndios florestais e vigilância geográfica. Nem sempre existem sensores específicos para cada propósito. Com isso, a rede deve se adaptar as necessidades da aplicação, e esse é um desafio para o desenvolvedor. Por exemplo, os sensores de um dispositivo podem medir apenas a umidade, a luminosidade ou o som, e com isso já é possível fazer um monitoramento de incêndio em florestas, sem necessariamente usar um sensor de temperatura. Esta outra funcionalidade só foi possível por que o desenvolvedor, utilizando apenas os sensores disponíveis, conseguiu identificar o fenômeno da queimada, por outro método que não o de temperatura. Sincronismo de Relógio

Em sistemas centralizados, geralmente não há a necessidade de se preocupar com a sincronização, pois há apenas um elemento temporizador. Neste caso existe um processo que emite o tempo atual para cada evento. Trata-se de um processo responsável por quantizar o tempo, emitindo os valores de forma contínua e incremental. Os sistemas centralizados têm uma ordem clara dos acontecimentos, pois há um relógio referência em que todos os eventos se baseiam, não havendo ambiguidade temporal. Já em sistemas distribuídos não há um relógio global, ou mesmo uma memória compartilhada, que possibilite a distribuição de um tempo único. Cada equipamento de um sistema distribuído tem seu próprio temporizador interno, portanto a própria noção temporal. Deveras, esses relógios podem diferir facilmente, variando poucos milésimos de segundos em um dia, mas que no acumulado de um ano torna-se um erro significativo. Este fato ocorre, até mesmo, em dispositivos que sejam do mesmo

lote de fabricação, problema grave para as aplicações que dependem de uma noção de sincronismo.

O desempenho das aplicações distribuídas é simplificado quando os relógios estão sincronizados. É bastante comum que aplicações distribuídas e protocolos de rede usem tempo limite (*timeout)*, portanto o desempenho de tais aplicações depende de quão bem os dispositivos estão sincronizados. Com o tempo, devido às suas diferentes taxas, os relógios podem divergir, motivando que a sincronização de relógio seja realizada periodicamente, nos sistemas distribuídos, para corrigir sua deformação. Para a grande maioria das aplicações e algoritmos, que funcionam em um sistema distribuído, é necessário ter a informação temporal de um ou mais contextos, como:

• A hora do dia em que ocorreu um evento, de uma máquina específica, da rede.

• O intervalo de tempo entre dois eventos que aconteceram em máquinas distintas, na rede.

• A ordem relativa dos eventos, que ocorreram em diferentes máquinas, da rede.

A sincronização de relógio assegura que os processos distribuídos tenham uma noção de tempo comum auxiliando os sistemas seguros; os diagnóstico e recuperação de falhas; na programação de operações; e nos sistemas de banco de dados.

2.2.1 Sistema síncrono e Sistema assíncrono

Existem diversos sistemas em todo o mundo, em alguns casos o sincronismo não é necessário devido a sua especificidade, e em muitos outros, o sincronismo se faz necessário neste sistema.

Segundo Kshemkalyani (KSHEMKALYANI; SINGHAL, 2011) um sistema assíncrono é aquele que:

1. Não necessita de processo de sincronia.
2. Não há limites temporais na diferença entre os processos, uma vez que não tem sincronismo e é impossível determinar um tempo máximo para qualquer processo, mesmo que este esteja com problema;
3. Não há limites para o atraso, tanto na transmissão quanto na propagação da mensagem;
4. Não há limite temporal para um processo executar qualquer etapa assíncrona.

Kshemkalyani menciona, ainda, as principais características de um sistema síncrono:

1. O tempo para a execução de qualquer etapa de um processo tem limite. Os processos são sincronizados e assim é possível ter um limite para o tempo gasto em cada processo;
2. O tempo gasto na entrega de mensagem pode ser limitado;
3. O tempo para a execução de qualquer processo também pode ser limitado.

Em comparação, é mais fácil desenvolver e verificar algoritmos que se valham da configuração de sistemas distribuídos sincronizados, devido à natureza da ordenação entre os eventos. Com isso, há a possibilidade de identificar possíveis equívocos deste sistema. Não obstante, há obstáculos para se ter uma execução que seja síncrona de fato. É praticamente impossível construir um sistema totalmente síncrono e garantir a entrega de mensagens em um tempo limitado. Logo, fatalmente, haverá atrasos maiores que os aceitáveis e assim alguns processos poderão ficar bloqueados por este atraso.

2.3 TEMPO LÓGICO

Este capítulo trata dos métodos para se estabelecer uma aproximação com o tempo da rede. A partir deste tempo, compartilhado pelos dispositivos, é possível determinar a casualidade dos eventos, mesmo que este tempo não seja exatamente igual a todos, mas sim similar. A causalidade, ou a precedência entre eventos, é um conceito fundamental em muitos projeto ou análises computacionais, paralelas ou distribuídas, em um sistema operacional. Todavia, em sistemas distribuídos, não é possível ter um tempo físico global, sendo possível, apenas, aproximações com este.

O conhecimento da relação causal-precedência entre os eventos de processos ajuda a resolver uma variedade de problemas em sistemas distribuídos, como apontado por Kshemkalyani em (KSHEMKALYANI; SINGHAL, 2008):

- Projetos de algoritmos distribuídos: o conhecimento da relação de precedência causal entre os eventos ajuda a garantir a vivacidade e a justiça em algoritmos de exclusão mútua; ajuda a manter a consistência de bancos de dados e na criação de

algoritmos de detecção de impasse. Conhecer a precedência entre eventos possibilita o agendamento de tarefas e a otimização, tanto do consumo de energia quanto do uso dos recursos deste sistema.

• Rastreamento de eventos dependentes: dentro da depuração distribuída, o conhecimento da dependência causal entre os eventos ajuda a construir um estado consistente para a retomada da execução na recuperação de falhas.

• Conhecimento sobre o progresso: o conhecimento de dependência causal entre os eventos ajuda a medir o progresso de processos na computação distribuída, o que é útil no descarte de informações obsoletas.

• Medir a concorrência: o conhecimento de quantos eventos são causalmente dependentes é útil para medir a quantidade de processos simultâneos; logo, todos os acontecimentos que não estão causalmente relacionados podem ser dissociados.

O conceito de causalidade é amplamente utilizado pelos seres humanos, muitas vezes inconscientemente, no planejamento, na programação e na execução de uma tarefa. No dia-a-dia, existe a hora oficial de Brasília que tem como intuito deduzir a relação de causalidade. Os relógios sincronizados de forma imprecisa, que não são ajustados automaticamente, podem ter defasagem de segundos ou até minutos. Contudo, em um sistema de computação distribuída, a quantidade de tarefas executadas concomitantemente é várias ordens de magnitude maior, e ainda, com um tempo de execução várias ordens de magnitude menor que a capacidade

humana. Consequentemente, a precisão dos relógios de um sistema distribuído necessita ser demasiadamente superior que a dos relógios ordinários.

Na seção a seguir serão estudados os métodos de ordenação temporal, que necessitam desta precisão em sua sincronização para a resolução dos problemas apresentados, por Kshemkalyani, no início deste capítulo.

2.3.1 Modelos de Ordenação Temporal

Como apresentado anteriormente, a ordenação temporal apresenta um conjunto de vantagens que facilitam o desenvolvimento e a depuração de sistemas. Desta forma, foram propostas diversas maneiras para se estabelecer esta correlação temporal entre os dispositivos, visando atender as especificidades de cada aplicação. Com o objetivo de selecionar a melhor opção, para o estudo aqui desenvolvido, segue a descrição de algumas das formas de se estabelecer a correlação temporal, bem como suas respectivas aplicações.

Tempo Global: Com o invento do relógio atômico foi possível estabelecer um padrão de tempo global de alta precisão. Proposto na década de 50 o *Coordinated Universal Time* (UTC) começou a funcionar no primeiro dia de 1960. Este padrão foi o primeiro sistema global com alta precisão utilizado como referência por diversas entidades do planeta. Por apresentar alta precisão, o UTC passou a ser utilizado, também, pelos principais algoritmos de sincronização da Internet (PALCHAUDHURI; SAHA; JOHNSON, 2004). Contudo, este sistema necessita estabelecer uma comunicação aos relógios ligados ao UTC, em muitos casos as RSSF localizam-se em áreas de difícil acesso e não possuem conectividade a Internet, por isto este sistema é inviável para a maioria das redes de sensores sem fio.

Ordenação Física: Em alguns casos a precisão temporal não é fundamental, e sim ter uma ordenação de eventos – como em sistemas que possuem apenas um relógio. Este tipo de ordenação total ou parcial de eventos é significativamente mais simples, sendo assim uma alternativa de baixo custo (PALCHAUDHURI; SAHA; JOHNSON, 2004).

Noção de Ordenação Relativa: O tempo pode ser sustentado entre um conjunto de nós com alguma noção lógica, que não precisa ser similar ao tempo cronológico. Esta temporização é suficiente para estabelecer uma relação temporal entre os dispositivo, desde que a ordenação relativa forneça este tipo de noção temporal (PALCHAUDHURI; SAHA; JOHNSON, 2004).

Ordenação Relativa: Esta é a noção de tempo entre um conjunto de dispositivos, onde cada aparelho é sincronizado em relação a algum valor do relógio de referência – que pode ser completamente diferente do UTC. Em muitos casos, esta função de ordenação é suficiente para estabelecer uma relação temporal entre os dispositivo, como em grande parte das aplicação na rede de sensores (PALCHAUDHURI; SAHA; JOHNSON, 2004). Geralmente, as redes de sensores sem fio não possuem conexão com a internet e em muitos casos utilizam dispositivos de baixa precisão. Assim, este modelo de ordenação se apresenta como uma boa alternativa no desenvolvimento de um método de sincronização, objetivo deste trabalho.

2.3.2 RTC

O relógio de tempo real (RTC ou do anglicismo *real-time clock*) é um dispositivo computacional, geralmente um circuito integrado que indica a evolução temporal. Os RTCs

estão presentes na quase totalidade dos dispositivos eletrônicos, que precisam manter um controle preciso do tempo. Estes circuitos são compostos por um cristal de oscilação que opera em determinada frequência. O tempo deste dispositivo é medido em função do número de oscilações destes cristais, em um determinado intervalo de tempo. Com isso, é possível correlacionar a evolução temporal estabelecida pela humanidade – segundos, minutos e horas – com as oscilações medidas por este sensor. Para estabelecer a ordenação relativa, todos os aparelhos utilizados neste trabalho adotam esta tecnologia para mensurar suas evoluções temporais, conformando no método de sincronização aqui proposto.

2.3.3 Fonte de erro de sincronismo

Existem diversas causas de erros temporais que dificultam a sincronização dos sistemas. As principais fontes de erros, que levam ao assincronismo são as não determinísticas, mas ainda há as determinísticas. Estas fontes de erro são compostas por seis variáveis:

Tempo de Envio: É o tempo utilizado no processo de transmissão de pacotes a partir da camada de aplicação até a camada de enlace, ou para RSSF até a camada MAC. O atraso no tempo de envio é não determinístico, devido a imprevisão no processo de carga dos dados em um sistema operacional (KUROSE, 2013).

Tempo de acesso: É o tempo gasto para encontrar o momento em que possa começar a transmissão. Ele é determinado pelo tráfego da rede atual e sua capacidade de execução na camada de enlace ou na camada MAC (KUROSE, 2013).

Tempo de transmissão e recepção: É o tempo gasto no envio ou na recepção de todos os dados do pacote ao longo do canal. Este tempo pode ser determinado em função da largura de banda e do tamanho do pacote a ser enviado (KUROSE, 2013).

Tempo de atraso de propagação: Este atraso é devido ao tempo gasto para transmitir a mensagem a partir do nó de origem para o seu destino. Se a velocidade do meio de transmissão é estável e a distância entre o emissor e o receptor é conhecida, trata-se de um atraso determinístico. No entanto, em RSSF, a distância entre os nós nem sempre é conhecida e em alguns casos pode até mudar. Para este trabalho a distância é fixa, contudo não é conhecida, ainda assim é possível mensurá-la. Desta forma, no estudo aqui proposto, considera-se como um atraso determinístico (KUROSE, 2013).

Tempo de processamento da mensagem de recepção: É o tempo gasto no envio da mensagem que parte da camada de enlace para a camada de aplicação, no receptor. Assim, como o tempo de acesso, este tempo é não determinístico devido a imprevisão no processo de desencapsular os dados em um sistema operacional(KUROSE, 2013).

2.4 RELÓGIOS DE DISPOSITIVOS

Neste capítulo serão apresentados os conceitos de defasagem de relógios com o objetivo de identificar a diferença temporal dos dispositivos, bem como minimizar esta diferença. Estes conceitos podem ser estabelecidos por dois ou mais aparelhos, simultaneamente. Também, será realizado um estudo das principais causas de imprecisão dos relógios e em como estimar, da melhor forma, o atraso entre estes dispositivos. Para tanto,

partimos das conceituações de David Mill (1998) acerca de tempo; frequência; deslocamento; inclinação; e deriva.

Assumindo que C_a e C_b são relógios, temos:

Tempo: O tempo de uma máquina p é definido pela função $\mathrm{C_p(t)}$, onde $C_p(t) = t$ para um relógio perfeito.

Frequência: é a taxa de progresso de um relógio. A frequência em t do relógio C_a é $C_a'(t)$.

Deslocamento: É a diferença entre o tempo referido pelo relógio e o tempo real. O deslocamento do relógio C_a em relação ao relógio perfeito, é dado por $C_a(t) - t$. O deslocamento do relógio C_a, em relação ao C_b, no tempo $t \geq 0$ é dado por $C_a(t) - C_b(t)$.

Inclinação: É a diferença entre a inclinação da reta $C_a(t)$ e a inclinação da reta $C_b(t)$. Logo, $C_a'(t) - C_b'(t)$.

Deriva: O desvio do relógio $C_a(t)$ é a segunda derivada do relógio, valor em relação ao tempo, ou seja, $C"_a(t)$. A deriva de C_a, relativo a C_b do relógio no tempo t, é dado como:

$$\mathrm{C"_a(t) - C"_b(t)}, \quad (1)$$

2.4.1 Sincronismo entre relógios

Como já mencionado, cada cristal é responsável pela a marcação temporal, em diversos casos ele pode variar e marcar o tempo diferentemente do recomendado. Para isso, é aconselhado que todos os aparelhos se equiparem a o valor do relógio de referência.

A partir da seguinte equação:

$$C_i(t) = \theta + f.t, \quad (2)$$

onde o θ é o deslocamento e f é a inclinação, é possível analisar a evolução temporal de um dispositivo. A Figura 4 apresentada essas definições. Com base na equação anterior, é possível estabelecer a relação temporal entre dois dispositivos, Nodo A e Nodo B, como:

$$C_B(t) = \theta^{AB} + f^{AB}.C_A(t) \quad (3)$$

onde θ^{AB} e f^{AB} representam o deslocamento relativo do relógio e a inclinação do nó B em relação ao nó A, respectivamente. Logo, podemos definir que se $\theta^{AB} = 0$ e $f^{AB} = 1$ os relógios dos dispositivos estão sincronizados (RHEE et al., 2009).

2.4.2 Imprecisões do Relógio

Todos os relógios têm uma defasagem em relação ao tempo, até mesmo os relógios atômicos utilizados pelo UTC, por esta razão é preciso sincronizá-los regularmente. Os relógios tradicionais são sincronizados pelo UTC, no entanto, conforme já discutido, não se recomenda usar o padrão temporal do UTC em redes de sensores sem fio, pois muitas vezes a rede não tem acesso a este relógio e nem os dispositivos comportam o formato temporal do mesmo. Relógios comumente não são muito precisos, aqueles que tem um desvio do ideal de apenas 0,001% de frequência acumulam um erro de segundos em apenas um dia, razão pela qual o desempenho do relógio é muitas vezes medido através de unidades muito finas, como uma parte por milhão (PPM).

Na impossibilidade de coadunar precisão e baixo custo, optou-se pelo baixo custo. Ainda que haja uma maior imprecisão é possível estabelecer um sincronismo na rede

aceitável. Isto posto, estabelecemos que a temporização seja dada no limite aceitável de imprecisão do relógio. Assim, segue a seguinte equação:

$$1 - \rho \leq \frac{dc}{dt} \leq 1 + \rho \quad (4)$$

Onde a constante ρ representa a taxa de imprecisão máxima de inclinação, especificada pelo fabricante. Dentro desta margem de imprecisão há três possíveis casos de relógios: o rápido, o ideal e o lento.

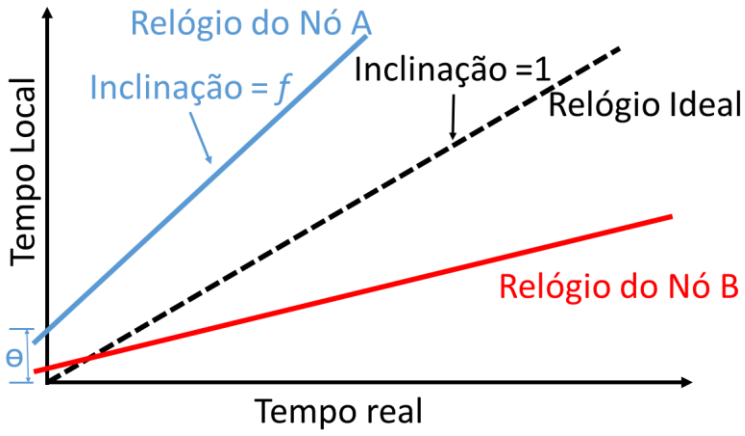

Figura 4 Comportamento do relógio rápido, perfeito e lento.

2.5 ESTIMANDO O ATRASO ENTRE DISPOSITIVOS

Como explicado na subseção 2.3.3, existem diversos fatores que causam o erro de sincronismo dos dispositivos de uma rede, contudo, em muitos casos, é possível estimá-los. Por exemplo, o protocolo de temporização de rede, *Network Time Protocol* (NTP), que é amplamente utilizado para sincronização de dispositivos na internet, usa o método de estimativa do atraso para assim considerar o tempo atual.

Na prática, um nó não consegue estimar, com precisão, o atraso da rede utilizando a diferença entre o tempo de envio e o tempo de resposta. O protocolo NTP utiliza uma prática muito comum: realiza várias medições de atraso e utiliza o atraso mínimo. O método de atraso mínimo, desenvolvido Cristian (1989), propõe uma análise probabilística para a leitura de relógios remotos de um sistema distribuído sujeito a atraso de comunicação aleatória. Ao analisar os atrasos de um grande conjunto de mensagens, temos sua representação em uma distribuição gaussiana, em que a média desta distribuição é muito próxima do valor mínimo. Desta forma, estipula-se que o menor atraso deve apresentar o melhor resultado para um grande conjunto de mensagens. Por essa razão, o NTP utiliza o método de atraso mínimo como estimativa de atraso.

O NTP utiliza o registro temporal de um conjunto de mensagens numeradas entres os nodos A e B, tendo T_1, T_2, T_3 e T_4 como tempos amostrados. A partir da premissa de que os relógios A e B são estáveis e funcionam com a mesma frequência, o NTP estabelece que:

$$a = T_1 - T_3 \tag{5}$$

$$b = T_2 - T_4 \tag{6}$$

Se a diferença entre o atraso de A para B e o atraso de B para A – conhecido como atraso diferencial – for pequeno, o ajuste do relógio é θ e o atraso da transmissão e recepção é δ. A diferença entre o atraso de B em relação a A, no tempo T_4, é dado de forma aproximada pelas seguintes formulas:

$$\theta = \frac{a+b}{2} \tag{8}$$

$$\delta = \mathrm{a} - \mathrm{b} \tag{9}$$

Cada mensagem do protocolo NTP inclui os últimos três registros temporais, a saber: T_1, T_2 e T_3, enquanto T_4 é determinado no momento de chegada da mensagem. Assim, os pares A e B, de forma independente, podem calcular e compensar o atraso utilizando um único fluxo de mensagens bidirecional (KSHEMKALYANI; SINGHAL, 2011). O fluxo das mensagens que estimam o atraso pode ser observado na Figura 5.

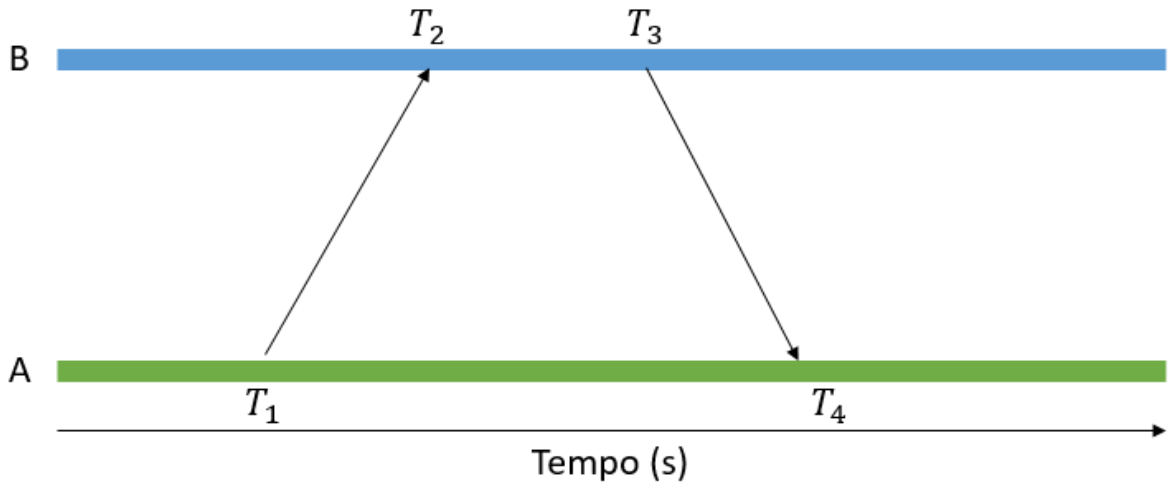

Figura 5 Esboço da estimativa de atraso

3 Revisão da literatura

Neste capítulo é apresentada a análise dos trabalhos relacionados que tratam sincronismo em redes de sensores, dando ênfase às redes de sensores sem fio, junto a comparação da presente proposta com estes trabalhos relacionados.

Estudos sobre sincronismo de relógio em sistemas distribuídos tem sido uma vasta área de pesquisa e vem sendo desenvolvido há muito tempo. Desde 1985, Kopetz já estudava sobre o erro de precisão dos cristais de quartzo nos sistemas distribuídos em tempo real (KOPETZ, 1985). Estes cristais são largamente utilizados para medir o tempo em diversos dispositivos, principalmente nos de baixo custo, objeto deste trabalho.

Em 1987, Kopetz deu continuidade ao seu trabalho anterior, desta vez analisando o comportamento de um sistema distribuído e identificando um limite de erro da similaridade dos relógios, para a implementação eficiente de um sistema tolerante a falhas, baseado no tempo global. Desta forma, desenvolveu um algoritmo de sincronismo de relógio para sistema distribuído em tempo real (KOPETZ; OCHSENREITER, 1987). Este trabalho, ainda, propôs características essenciais, a serem encontradas em um protocolo de sincronização de relógio, para o estabelecimento de um sistema harmônico, como:

1. O protocolo deve lidar com transmissão de rede não confiável e latências ilimitadas.
2. Quando um par de nodos estiver sincronizando, cada nó deve ser capaz de determinar o tempo local de seu equivalente. É importante frisar que esta não é uma questão simples, uma vez que há atrasos não determinísticos em um sistema distribuído.

3. O tempo de cada dispositivo nunca deve ser ajustado para trás. Logo, os relógios devem ser corrigidos gradual e graciosamente, até que o sincronismo seja alcançado.
4. A sobrecarga de sincronismo não pode afetar o desempenho do sistema.

Segundo o autor, estas características são essenciais para que se possa estabelecer o sincronismo da rede de forma harmônica e sem conflito de informações.

Para estabelecer o sincronismo de uma rede, um ponto essencial, é poder mensurar o atraso entre dois dispositivos. Cristian (1989) propôs uma nova abordagem para a leitura de relógios remotos, em que seja possível haver atrasos aleatórios em quaisquer mensagens em uma rede. Este método pode ser usado para melhorar a precisão dos algoritmos de sincronização, de uma rede interna e externa. Desenvolvendo uma abordagem probabilística, porque não garante que seja sempre possível ler a informação do relógio remoto, o processo de leitura é repetido de forma contínua até garantir que o tempo de relógio – de outro dispositivo – obtenha a precisão desejada. Neste trabalho, foi desenvolvido um estudo estatístico baseado na análise de uma amostra, onde demostra que a melhor representação dos atrasos entre *hops* são os de menor valor.

Nas próximas seções será abordado um conjunto de protocolos de sincronismo, focado, primeiramente, no sincronismo de uma rede convencional de computadores, v.g. Internet; depois a redes de sensores, v.g. redes cabeadas de sensores e por final rede de sensores sem fio, v.g. protocolos que são baseados no IEE 802.15.4.

3.1 SINCRONISMO EM REDES DE COMPUTADORES CONVENCIONAIS

O *Network Time Protocol* (NTP) foi idealizado por David L. Mills, sua implementação iniciou-se na década de 80, nele as mensagens de sincronismo são transmitidas via o protocolo *User Datagram Protocol* (UDP) – que é um protocolo não confiável e não orientado à conexão (MILLS, 1981). Este protocolo de sincronismo é um dos mais antigos e é amplamente adaptado às grandes redes, com infraestruturas similares a da internet. O projeto do NTP envolve uma árvore hierárquica de servidor de tempo, em que o servidor primário sincroniza-se com o UTC. Na sequência, os servidores secundários realizam a função de *backup* do servidor primário e, nos demais níveis, a sincronização ocorre como uma relação cliente-servidor. Respeitando a hierarquia entre os servidores, este sincronismo se propaga até o usuário final. O NTP permite manter o relógio de um computador com a hora sempre certa e com grande exatidão, usando uma análise estatística de atraso das mensagens de ida e volta (MILLS, 1992). Embora este protocolo funcione na sincronização dos computadores ligados à Internet, ele não atende aos requisitos de RSSF. Como já mencionado na subseção 2.2, esta rede apresenta desafios diferente das redes convencionais, assim o não determinismo dos atrasos e a separação da internet impossibilita a utilização deste algoritmo para manter a equiparação temporal em uma rede de sensores sem fio.

Em 1997, Veríssimo apresentou seu modelo de sincronismo para sistemas distribuídos de tempo global, para redes LAN ou WAN, chamado CesiumSpray. Baseado em um inovador esquema de sincronização de relógio, que estima o atraso entre os dispositivos de forma determinística, Veríssimo parte dos estudos de Cristian para estimar os atrasos entre as

máquinas em tempo real em uma rede local que utiliza um esquema de sincronização global hierárquico, onde a raiz é fonte de tempo absoluto. O algoritmo de sincronização exibe alta precisão; apresentou ser escalável e tolerante a falhas. Cada nó tem a informação do relógio virtual da rede que é interpretado localmente para correlacionar o seu tempo. Este protocolo de sincronização foi desenvolvido para ser híbrido, juntando o melhor dos dois ambientes: o externo (GPS), o relógio virtual, e o interno de uma rede relógio local, bem como sua eficácia de sincronização, pois todos os dispositivos devem sofrer atrasos próximos e reduzir o não determinismo dos atrasos desta rede (VERÍSSIMO; RODRIGUES; CASIMIRO, 1997).

Um novo campo para o estudo do sincronismo se deu com o desenvolvimento das redes sem fio na década de 90. Nos anos 2000, Mock, apresentou o protocolo de sincronização contínua de relógio em tempo real para rede sem fio, chamado de *Continuous clock synchronization* (CCS); no qual estende o padrão IEEE 802.1.1 de redes locais sem fio. Este protocolo de sincronização proporciona ao protocolo WiFi uma sincronização contínua de relógio, melhor precisão do sincronismo e a tolerância de perda de mensagem de sincronismo – tudo isso utilizando um algoritmo que ajusta o relógio físico em função de um relógio virtual. Como o relógio físico não pode, simplesmente, retroceder ou avançar de forma brusca, pois isso interferiria na ordenação dos processos internos. Desta forma, ajusta-se a frequência do relógio, ou como é conhecido tecnicamente, a inclinação deste relógio. Os resultados dos estudos de Mock foram estender o padrão IEEE 802.1.1 com um sincronismo contínuo de alta precisão (MOCK et al., 2000).

3.2 SINCRONIZAÇÃO EM REDES DE SENSORES SEM FIO

Em 2002, com o desenvolvimento da tecnologia sem fio e a criação das RSSF, Elson apresentou seu trabalho sobre uma nova técnica de sincronismo de relógio para uma rede de sensores, chamado *Reference-Broadcast Synchronization* (RBS), que fornece uma sincronização temporal na rede com uma maior precisão, flexibilidade e eficiência em comparação aos algoritmos tradicionais de uma rede cabeada. No RBS os nós enviam uma mensagem de sincronismo aos seus vizinhos. Esta mensagem não contém nenhum tempo referência. Em vez disso, os nós receptores usam a hora de chegada da mensagem como referência temporal para comparar seus relógios. De posse desta informação, cada dispositivo informa para seus vizinhos o seu registro temporal (*timestamp*), sendo possível que qualquer nó cliente possa traduzir seu tempo local para qualquer tempo de outro nó. Como este algoritmo utiliza somente o tempo de chegada para estabelecer o sincronismo, o não determinismo do lado do remetente é eliminado com o objetivo de alcançar uma maior precisão. O projeto desenvolvido por Elson apresenta como principal diferencial a capacidade dos conjuntos de receptores de sincronizarem-se entre si, em oposição aos protocolos tradicionais, em que os nós receptores se sincronizam em função dos nós remetentes. Neste trabalho, o autor afirma que o RBS pode ser usado sem referências externas, formando uma escala de tempo relativa e de alta precisão entre os nós, ou pode manter a sincronização na escala dos microssegundos para um tempo externo, como o UTC. Além disso, ele apresenta um novo algoritmo de sincronização que utiliza saltos múltiplos, permitindo que o sincronismo seja mantido em domínios de *broadcast*. No entanto, em muitas situações, a rede não conta apenas com um conjunto de dispositivos que tenham uma comunicação direta. Para

a arquitetura de *mult-hops*, a rede se agrupa conforme sua topologia. Os nós que estiverem ligados a mais de um grupo devem transformar os dados temporais entre eles, visando manter uma equiparação temporal de toda a rede (ELSON; GIROD; ESTRIN, 2002).

Em 2003, Su Ping apresentou a técnica *Delay Measurement Time Synchronization* (DMTS). Esta técnica visa atender a rede de *single-hop* e *mult-hops* em uma RSSF e, em comparação com o RBS, ele concluiu que o DMTS apresenta melhor eficiência energética e adiciona baixo tráfego a rede, pois ocorrem menos transmissões de mensagem de sincronismo no DMTS. Entretanto, a precisão do sincronismo é inferior à do RBS. Assim, as vantagens e desvantagens devem ser ponderadas para determinar o melhor protocolo. Neste protocolo um nó raiz é selecionado como mestre, que transmite o seu registro temporal (*timestamp*) para os demais nós da rede. Todos os dispositivos receptores analisam as diferenças temporais e ajustam o seu relógio em função do valor recebido do nó raiz, acrescido do atraso medido entre o nó mestre e os demais nós. Desta forma, todos os dispositivos ficam sincronizados com o nó raiz. A precisão da sincronização de tempo é limitada, principalmente, em função da exatidão nas medições de atraso que foram executadas ao longo do caminho. Este algoritmo foi desenvolvido para ser flexível, robusto, simples e suportar a frequente atualização da topologia da rede. Todas estas características, principalmente a baixa complexidade, visam a atender as particularidades das RSSF (PING, 2003).

No mesmo ano, Ganeriwal propôs uma nova abordagem para a sincronização de tempo, o protocolo *Timing-sync Protocol for Sensor Networks* (TPSN) que visa fornecer sincronização de tempo em uma rede de sensores. Este protocolo é baseado no NTP, porém é

flexível para suportar as características de uma rede de sensores sem fio. Este método é composto por duas fases: descoberta e sincronização. A primeira fase ocorre quando a rede é implementada, para isso é preciso compreender qual é a hierarquia de cada nó em relação a estrutura da árvore. A segunda fase ocorre na sincronização entre os pares, ao longo de toda a estrutura, utilizando a técnica de *handshake* entre receptor e emissor. Este sincronismo ocorre de forma *TOP-DOWN* (de cima para baixo), ou seja, os pares superiores são sincronizados primeiro, em seguida seus filhos, até alcançar toda árvore. Ao contrário do RBS, o TPSN apresenta o não determinismo no atraso do remetente. Desta forma, ele tenta reduzir esse não determinismo do servidor, coletando a informação do registro temporal (*timestamp*) na camada MAC, pois a incerteza do remetente contribui muito pouco para o erro de sincronização total. É importante levar em conta que o TPSN é focado na estrutura de múltiplos saltos e, mesmo assim, obteve um resultado melhor que o RBS que é focado no salto único. Com todas estas características, nos testes desenvolvidos por Ganeriwal, o TPSN apresenta uma precisão duas vezes melhor, se comparado ao RBS (GANERIWAL; KUMAR; SRIVASTAVA, 2003).

PalChaudhuri (PALCHAUDHURI; SAHA; JOHNSON, 2004) publicou um trabalho sobre CesiumSpray – adaptando-o para uma RSSF. Este método utiliza a distribuição Gaussiana para a sincronização de relógio, que atinge uma maior precisão de sincronização de receptor a receptor, ou seja, ele executa um conjunto de medições de atraso e busca a média, quanto menor o desvio padrão maior será a precisão desta medição. Este método baseia-se no RBS, estendendo sua capacidade para fornecer a sincronização de uma rede de dispositivos,

que possua relógios adaptáveis. Ainda, adiciona a caraterística de *mult-hops*, aumentando as possibilidades de sincronização em uma RSSF. Este trabalho foi desenvolvido com sensor Berkeley® e obteve excelentes resultados, novamente, em comparação ao protocolo RBS.

No mesmo ano, Maróti e Kusy (MARÓTI et al., 2004) apresentaram o protocolo de sincronização de relógio, voltado para aplicações que requerem rigorosa precisão em plataformas *wireless* de recursos limitados – chamado de *Flooding Time Synchronization Protocol* (FTSP) – que fornece um mecanismo de sincronização para os relógios locais da rede por inundação, pois, segundo os autores, o TPSN não estima a inclinação do relógio dos nós, limitando sua precisão. Também, não suporta alterações dinâmicas em sua topologia e, desta forma, eles propuseram um protocolo semelhante ao TPSN que solucione estes problemas – o FTSP. Este protocolo é tolerante a falhas de *links*. O FTSP alcança sua robustez através da inundação periódica das mensagens de sincronização e, também, por sua topologia ser atualizada dinamicamente. Nos testes implementados, por Maróti e Kusy, o erro médio de sincronização por salto é melhor do que nos algoritmos RBS e TPSN (MARÓTI et al., 2004).

A semelhança entre o FTSP e o TPSN reside no fato que ambos possuem um nó raiz, em que todos os nós estão sincronizados. O nó raiz transmite as mensagens de sincronismo para todos os receptores de forma simultânea. Estas mensagens contém o registro temporal (*timestamp*) do remetente coletada na camada MAC, como no TPSN. Ao receber esta mensagem, o nó receptor coleta o seu registro temporal, ainda na camada MAC. Em posse destes dois registros temporais, o receptor estima o deslocamento de seu relógio, almejando o

sincronismo com o nó transmissor. Os registros temporais são normalizados, descontando o máximo possível da latência do receptor. Estes descontos ocorrem aos pares, em função do registro temporal e do tempo de chegada da mensagem, para estimar o deslocamento do relógio. A abordagem FTSP utiliza a inundação *ad-hoc* como método de propagação da mensagem de sincronismo, pois foi projetada para suportar grandes redes *multi-hop*, com topologia que se modifica no decorrer de suas funções, em que a raiz é eleita dinâmica e periodicamente – no qual, normalmente, o mesmo nó é reeleito. Todas estas características tornam o FTPS escalável, robusto e tolerante a falha de *links*. Outras propostas para este método baseiam-se na difusão, que visa fornecer o sincronismo interno sem depender de uma hierarquia de rede, de um nó mestre ou de relógios de referência (MARÓTI et al., 2004).

Como as RSSF estão frequentemente sujeitas a falhas, o FTSP é uma ótima escolha para esta rede; visto que esta técnica não se sincroniza com uma fonte externa, ela pode ser particionada em ilhas. Para o caso destas ilhas, deve-se tomar cuidado com a fusão de dados de conjuntos diferentes, pois outra forma de sincronização seria necessária.

Os estudos de Bharath Sundararaman, em 2005, examinaram e avaliaram diversos protocolos de sincronização de relógio. Eles focaram em fatores como precisão, custo e complexidade; mostraram que protocolos de sincronismo tradicionais – de redes com fio – não se aplicam para os de uma rede de sensores sem fio, visto que os protocolos das RSSF requerem a capacidade de se adaptar dinamicamente, de lidar com a mobilidade do sensor e de ser escalável. Ademais, dentre suas limitações, destaca-se seus baixos recursos (poder computacional, alcance da rede etc.) e a pouca energia da bateria. Além disso, os sensores

podem operar em ambientes não confiáveis, como os expostos a muita interferência. O trabalho de Sundararaman teve como diferencial o desenvolvimento de uma estrutura que possibilita a comparação entre protocolos de sincronização, já existentes e futuros (SUNDARARAMAN; BUY; KSHEMKALYANI, 2005).

Ainda em 2005, Weilian Su e Ian F. Akyildiz desenvolveram o protocolo *Time-Diffusion Synchronization Protocol* (TDP). Este protocolo de sincronização foi desenvolvido para que uma rede de sensores alcance um tempo de referência e mantenha uma pequena tolerância de desvio temporal, a partir do valor do nó referência. Este nó é selecionado por meio de votação, em que se elege ou reelege o nó mestre. Após, ocorre uma busca por nós falsos ou inativos para que os mesmos sejam descartados, com isso inicia-se o processo de sincronização com os nós válidos. Este processo funciona entre pares, tendo início no nó raiz e seus adjacentes, se propagando até os nós mais distantes do nó referência. Em todo este processo o nó de sincronização executa o algoritmo de ajuste de relógio, para se equiparar ao nó raiz. Este protocolo caracteriza-se por possuir número de saltos – número de transmissões – fixos, formando assim uma estrutura de árvore radial. O algoritmo de ajuste de relógio pode usar o método híbrido adaptável, que foi desenvolvido para o NTP Versão 4 (MILLS, 1998). O algoritmo híbrido usa uma combinação do *Phase Locked Loop* (PLL) e *frequency lock loop* (FLL), que geralmente são implementados em hardware para minimizar o ruído, e assim melhorar a precisão temporal (XIANG; SUN; LI, 2011).

No ano seguinte, 2006, o *Distributed Time Sync* foi apresentado ao meio acadêmico. Este protocolo visa explorar o maior número de restrições globais possíveis de uma rede. Estas restrições devem ser satisfeitas por uma noção comum de tempo, em uma rede que contém um grande número de dispositivos. Ao impor um grande número de restrições globais, para todos os laços na rede *multi-hop*, a rede terá seus valores de estimativa de atrasos suavizados, e desta forma, tornando-a mais precisa. Trata-se de um protocolo em que os nós vizinhos trocam mensagens de sincronismo, informando seu registro temporal, de forma assíncrona e pareada. Além de conterem o registro temporal, essas mensagens; possuem a inclinação mais recente do relógio – em relação ao nó receptor – e a diferença temporal entre as últimas transmissões. Assim, o nó receptor ajusta a linha de deslocamento, usando uma estimativa de mínimos quadráticos recursiva. Outra característica deste protocolo é que qualquer nó pode tornar-se um nó de referência, basta não ajustar seu relógio durante o procedimento de sincronização, para tanto, este nó precisa ser selecionado pelo administrador da rede (SOLIS; BORKAR; KUMAR, 2006).

Já em 2007, Luca Schenato, Giovanni Gamba e sua equipe (2007) apresentam o *Average TimeSync* (ATS), o qual foi desenvolvido para uma rede de sensores sem fio que emprega uma abordagem de consenso de média. Ele baseia-se no *Distributed time sync* e utiliza dois métodos: i) estima sua taxa de inclinação em relação aos demais nós; ii) os nós transmitem sua estimativa atual da inclinação do relógio virtual. Por fim, os nós receptores combinam a informação de sua inclinação com a do relógio virtual, a fim de ajustar sua

própria estimativa. A ideia consiste em determinar a média de atraso para gerar um tempo virtual, no qual cada nó se adaptará.

O ATS tem três características principais: ele é totalmente distribuído e, portanto, robusto a falha do nó e de *link* suportando uma topologia dinâmica; ele compensa as diferenças de sincronia de relógio entre os nós, mantendo a rede sincronizada por períodos mais longos do que usando a compensação de relógio simples. Finalmente, é computacionalmente leve, pois envolve apenas simples operações de soma e produto. Segundo os autores, ainda é necessário um trabalho extenso para poder comparar o desempenho desta abordagem, em relação a outros protocolos.

Em 2008, Fengyuan Ren, propôs um novo protocolo de sincronização temporal para uma RSSF, denominado *Self-Correcting Time Synchronization* (SCTS). Este protocolo transforma o problema de sincronização de tempo em um processo dinâmico de otimização auto ajustável, utilizando um algoritmo baseado em PLL para fazer a compensação do deslocamento temporal. Ele combina a compensação deslocamento, que é eficaz para a sincronização de curto prazo, com a compensação da inclinação, visando a sincronização a longo prazo. Este método foi desenvolvido para RSSF, pois é de baixa complexidade, robusto, tolerante a perda de *link* e capaz de filtrar algum ruído proveniente do meio; contudo não apresenta escalabilidade (REN; LIN; LIU, 2008). Os testes foram desenvolvidos usando o sensor Berkeley®, o mesmo utilizado no trabalho de PalChaudhuri (2004). Embora os testes fossem limitados a um único domínio de difusão ele pode ser extensível à sincronização de tempo em toda a rede, usando o mecanismo de descoberta e manutenção dos nós raízes. Os

testes desenvolvidos por Ren, Lin, & Liu apresentaram a superioridade do protocolo SCTS, em relação ao TPSN. Além de todas essas vantagens, esta proposta apresenta baixa complexidade e baixo impacto ao tráfego da rede (REN; LIN; LIU, 2008).

No mesmo ano, O. Mirabella, M. Brischetto, A. Raucea e P. Sindoni desenvolveram um algoritmo de Sincronização de Relógio Dinâmico Contínuo (*Dynamic Continuous Clock Synchronization*), para uma RSSF. Este algoritmo fornece um relógio virtual comum entre o nó servidor e seu grupo de nós clientes, com uma abordagem baseada em algumas características de RBS e CCS. Também, baseia-se no protocolo IEEE 802.15.4 e assume que as transmissões sejam instantâneas entre os nós. Este algoritmo utiliza uma topologia em estrela, permitindo que todos os nós sejam conectados através de um único salto. A principal característica, apontada por Mirabella, é a habilidade de permitir que os nós clientes durmam por longos períodos e troquem, somente, raros pacotes de sincronização. O objetivo deste protocolo é sincronizar a rede ao nó raiz (RN do inglês *Reference Node*) periodicamente. O RN transmite uma mensagem de sincronismo, conhecida como RBM do anglicanismo *Reference Broadcast Message*, para os demais dispositivos. Entre estes dispositivos, um é o *Time Master* (TM) e os demais são nós escravos que devem sincronizar-se em função do TM. Periodicamente, o RN transmite sua mensagem aos demais dispositivos, que ao recebê-la coletam seu tempo local. Esta operação é realizada de forma simultânea, no tempo T1. Posteriormente, o TM transmitirá o seu valor de tempo local, coletado em T1, que será usado por todos nós escravos para sincronizarem-se, utilizando seus registros temporais, também

coletados em T1, para executar a equiparação temporal. O objetivo deste processo é compensar o máximo de atrasos possíveis contidos na RBM (MIRABELLA et al., 2008).

Os resultados obtidos através de um banco de testes demonstraram a capacidade de manter uma boa sincronização de relógio, entre uma estação servidor e seus clientes, mesmo com períodos de inatividade muito longos (MIRABELLA et al., 2008). Contudo este algoritmo só pode ser usado em redes que tenham apenas um salto e, também, necessitam de dois dispositivos. Um para estipular a cadência da mensagem de sincronismo e outro para ser a referência temporal, utilizando um aparelho a mais que o RBS, para alcançar o sincronismo.

Em 2013, o trabalho Zhengbao Li, foca na sincronização temporal em redes acústicas de sensores móveis, em um ambiente subaquático. Nele, um grande número de algoritmos de sincronização de tempo foi testado, mas a maioria destes para redes de sensores sem fio que utilizam frequência de rádios (*radio frequency* – RF). Os algoritmos que utilizam a RF, assumem que o atraso de propagação é desprezível. Em uma rede subaquática esta suposição sobre o atraso de propagação é incorreta, pois a comunicação é feita, principalmente, através do canal acústico, e com isso a velocidade da propagação é muito mais lenta do que RF. Isto posto, eles desenvolveram um algoritmo baseado em RF bidirecional, chamado de E²DTS, que na primeira fase estima o relógio e sua inclinação, em função de um conjunto de mensagens de sincronização. Na segunda fase é estimado o atraso entre nós não sincronizados. Desta forma, Li, apresenta um algoritmo de sincronização de tempo distribuído, para redes acústicas de sensores móveis em um ambiente subaquático, com eficiência energética. O E²DTS não estima o atraso de propagação variado pelo tempo, mas

sim, utiliza a diferença entre intervalos de tempo em uma sequência de registros temporais subsequentes, para reduzir o efeito da mobilidade dos nós. Deste forma, este algoritmo apresenta uma maior precisão temporal, para os ambientes subaquáticos, em relação aos demais – uma vez que os protocolos, já apresentados, não foram desenvolvidos para este meio de propagação (LI et al., 2013).

Em 2014, Kasim Sinan Yildirim e Aylin Kantarci, com o intuito de aumentar a sincronização temporal e a escalabilidade de uma rede, propuseram um novo protocolo (*Flooding With Clock Speed Agreement* FCSA). Este protocolo utiliza o método de inundação para forçar todos os nós a funcionarem na mesma frequência, empregando um algoritmo de concordância e sincronizando-os com um nó de referência. Este método emprega um algoritmo que faz pequenas alterações na frequência do relógio, para deixa-lo o mais similar ao seu relógio referência. Como este algoritmo é desenvolvido para iniciar a o sincronismo o mais rápido possível e assim se manter, ele obtém melhora significativa quando comparado com o FTSP (YILDIRIM; KANTARCI, 2014).

No mesmo ano, visando a economia energética de uma rede de sensores sem fio, baseada na norma IEEE-802.15.4, Diedrichs, Ana Laura, Tabacchi, Germán et al. apresentaram uma abordagem híbrida que combina as vantagens do TDMA e do *Frequency Hopping Spread Spectrum* (FHSS). Concorrentemente, foi proposto uma nova abordagem de sincronização que melhora a precisão temporal da rede. Nesta abordagem, Diedrichs, propõe um algoritmo que remove o não determinismo do registro temporal, ao mesmo tempo em que ajusta o relógio local, visando a redução da diferença entre o mesmo e o seu relógio de

referência. Este algoritmo apresenta um erro de sincronização – médio entre os dispositivos – menor do que os algoritmos que são amplamente utilizados, como o RBS e o TPSN, porém, utiliza 25% a mais da largura de banda se comparado à estes algoritmos (DIEDRICHS et al., 2014).

Em 2015, Lenzen, Christoph, Sommer, Philipp e Wattenhofer, Roger propuseram o PulseSync, um protocolo de sincronização de relógio eficiente e escalável, que faz uma série de análises para medir o atraso entre os dispositivos, buscando a média da distribuição normal deste atraso, com isso é possível estabelecer um limite para a precisão, latência e eficiência do sincronismo na rede. A ideia central deste algoritmo é distribuir informações sobre os valores de relógio, o mais rápido possível, enquanto minimiza o número de mensagens necessárias para estabelecer o sincronismo na rede, afim de que a rede estabeleça o sincronismo de forma ágil. Sempre que um nó intermediário recebe o relógio referência ele adiciona o atraso já medido, retransmitindo-o até que a árvore de dispositivos seja alcançada em sua totalidade. É notório que um nó não pode encaminhar nenhuma informação sem recebê-la, logo, o fluxo de informações é direcionado do nó raiz às extremidades da rede. Por outro lado, após a recepção da primeira mensagem, ele deve encaminhar a estimativa do atraso o mais rápido possível, a fim de espalhar a nova informação sobre o atraso para toda a rede. Este algoritmo ainda utiliza a busca em amplitude (*Breadth-First Search* - BFS) em uma árvore, para manter o número de saltos pequenos. Esta busca é executada com o objetivo de manter pequenas distorções do relógio referência, pois em todos os saltos este protocolo faz com que cada nó não apenas minimize seu deslocamento em direção ao nó raiz – ao receber a mensagem de

sincronismo – mas também, emprega uma compensação de deriva. A estimativa de atraso é dada por uma função linear que suaviza as flutuações dos atrasos de mensagens, que normalmente são aleatórios. Deste modo, a qualidade da estimativa de atraso tenta ser o mais preciso possível. Alcançando, assim, um sincronismo de qualidade em um curto espaço de tempo. Nos testes realizados, Lenzen, utilizou dispositivos comerciais de baixo consumo, microcontrolador Atmel SAM3U, que opera sobre a norma IEEE 802.15.4. As experiências demonstram que o PulseSync oferece uma precisão maior no sincronismo e na latência que o FTSP, fato já idealizado em um modelo abstrato desenvolvido pelo autor (LENZEN; SOMMER; WATTENHOFER, 2014).

A Tabela 2 apresenta um resumo dos trabalhos relacionados a sincronismo de rede, destacando suas características e diferenças. Cabe destacar, ainda, que estes protocolos apresentados inspiraram os métodos aqui propostos, contudo estes métodos são um amalgama de princípios destes protocolos, que visa alcançar o sincronismo para uma RSSF; com componentes de prateleira e de baixo custo; reduzindo o não determinismo do atraso e utilizando o mínimo de transições das mensagens de sincronismo.

Tabela 2 Trabalhos relacionados

Tipo	Ano	Autor	Características
Sincronismo geral	1985	Kopetz	Estudou o erro de precisão dos cristais de quartzo em sistemas distribuídos.
	1987	Kopetz	Identificou um limite de erro da similaridade dos relógios e propôs características essenciais para protocolos de sincronização dos mesmos.
	1989	Cristian	Propôs uma abordagem para a leitura de relógios remotos em redes que tenham atrasos aleatórios em quaisquer mensagens da rede.
Sincronismo de redes de computadores	1981	Mills	Idealizou o protocolo NTP, o primeiro para as redes de computadores.
	1997	Veríssimo	Desenvolveu um modelo de sincronismo para sistemas distribuídos (redes LAN ou WAN) de tempo global (CesiumSpray), que combina o sincronismo externo (GPS) e o interno de uma rede.
	2000	Mock	Desenvolveu o primeiro protocolo de sincronização para redes sem fio.

Tipo	Ano	Autor	Observação
Sincronização em redes de sensores sem fio	2002	Elson	Desenvolveu o RBS, primeiro algoritmo de sincronismo de relógio focado para as redes de sensores sem fio.
	2003	Su Ping	Desenvolveu o DMTS, algoritmo de sincronismo de relógio, que visa atender a rede de single-hop e *mult-hops* em uma RSSF. Este método apresenta melhor eficiência energética e adiciona baixo tráfego a rede se comparado com a RBS.
	2003	Ganeriwal	Desenvolveu o TPSN, método de sincronismo de relógio para redes de *mult-hops* que descobre a estrutura da rede automaticamente, além de atingir uma precisão duas vezes maior que o RBS.
	2004	PalChaudhuri	Adaptou o algoritmo CesiumSpray para RSSF, utilizando uma distribuição Gaussiana para a sincronização de *relógio*, atingindo uma maior precisão de sincronização de receptor a receptor.
	2004	Maróti	Desenvolveu o FTSP, que consegue estimar a inclinação dos nós e suporta alterações dinâmicas na topologia da rede.
	2005	Sundararaman	Analisou diversos protocolos comparando-os e desenvolveu uma estrutura que possibilita a comparação entre protocolos de sincronização.
	2005	Weilian Su	Desenvolveu o TDP, que é um protocolo onde não é designado, pelo gerenciador de rede, o nó referência e gera uma estrutura de árvore radial.

Tipo	Ano	Autor	Observação
Sincronização em redes de sensores sem fio	2006	Solis	Desenvolveu o *Distributed Time Sync*, que explora o maior número de restrições globais possíveis de uma rede para deixá-la sincronizada, sem precisar de um nó referência. Sendo assim, é voltado a redes *mult-hops*.
	2007	Schenato	Desenvolveu o ATS, que é baseado no *Distributed Time Sync*, contudo o ATS estima a taxa de inclinação em relação aos demais nós. Desta forma, é projetado um relógio virtual ao qual os *relógios* dos dispositivos se equiparam.
	2008	Fengyuan Ren	Desenvolveu o SCTS, que transforma o problema de sincronização de tempo em um processo dinâmico de otimização auto ajustável, contudo não é escalável.
	2008	Mirabella	Desenvolveu o *Dynamic Continuous Clock Synchronization*, que fornece um relógio virtual comum entre o nó servidor e o seu grupo de nós clientes, combinando o RBS com o CCS.
	2013	Zhengbao Li	Desenvolveu o E^2DTS, protocolo voltado para redes subaquáticas onde a velocidade de transmissão é reduzida. Consequentemente, o deslocamento e outras características devem ser consideradas para estabelecer o sincronismo da rede.
	2014	Yildirim	Desenvolveu o FCSA, que utiliza o método de inundação para estabelecer o sincronismo de todos os nós com o nó referência.

Tipo	Ano	Autor	Observação
Sincronização em redes de sensores sem fio	2014	Diedrichs	Apresentou uma abordagem híbrida que combina TDMA com FHSS, removendo o não determinismo do registro temporal para ajustar o *relógio* local.
	2015	Lenzen	Desenvolveu o PulseSync, que faz uma série de análises para medir o atraso entre os dispositivos. Com esta informação, o nó já inicia a propagação de seu *relógio* ao mesmo tempo que se ajusta ao nó referência.

Este trabalho propôs três métodos de sincronismo que visam atingir os melhores resultados de sincronismo para dispositivos de baixa precisão. Para isto, os conceitos de coletar o registro temporal na camada MAC, estimar o atraso de forma probabilística e utilizar a informação da frequência do relógio com o período de latência foram combinados. Estes três métodos foram desenvolvidos para suportar redes *single-hop* e *multi-hop*. Eles utilizaram um relógio que varia de 2 a 27 kHz e um erro máximo de ± 80 PPM, já o DMTS apresenta uma frequência de relógio de 1MHz e erro máximo de ± 20, mesmo assim, este trabalho, no melhor caso, apresentou resultado, proporcionalmente, melhor que o DMTS com um intervalo de tempo 3 vezes maior de período de latência. Se comparado com o ATS, que combina dois relógios (1MHz e 32KHz) com um erro de 1 PPM (TEXAS INSTRUMENTS, 2004) utilizando o período de latência de 180 segundos (3 minutos), o ATS atinge uma precisão pouco melhor que o melhor caso deste trabalho, contudo, a margem de erro é muito menor

que a utilizada neste artigo. A comparação com os demais protocolos é inviável, uma vez que, as informações para tal comparação foram insuficientes.

4 MÉTODOS PROPOSTOS E APLICADOS

Neste capítulo é apresentado os métodos propostos deste trabalho e como estes foram aplicados para ser implementados nos dispositivos.

O propósito deste trabalho é desenvolver um sistema que possua um algoritmo de sincronismo de relógio que estabeleça a máxima precisão possível, em uma arquitetura baseada no protocolo IEE 802.15.4; que utilize um conjunto de dispositivos de baixo custo; e que possua um baixo consumo energético utilizando uma rede mestre-escravo, na qual os nodos escravos se adaptem (i.e. se sincronizem) ao nó mestre. Por estas características, os equipamentos utilizados foram da antiga marca Freescale® (atualmente NXP®) da família MC1322x. Além destas características, é imprescindível que este novo algoritmo não tenha um grande impacto na rede; não sobrecarregue o sistema; os relógios físicos nunca sejam ajustados para antes da data atual; e que não necessite de uma transmissão confiável, isto é, pode haver perdas de mensagens e mesmo assim a rede permaneça sincronizada. Respeitando estas características, projetamos e desenvolvemos três algoritmos de sincronismo de relógio: correção adaptativa, predição de relógio e ajuste analítico. Estes algoritmos, tanto o nó referência quanto os demais, coletam seus registros temporais na camada MAC, visando eliminar ao máximo o não determinismo do atraso das mensagens, são classificados como do tipo *send-receive* (transmissor receptor). Além disso, o ajuste destes algoritmos é em relação a frequência dos relógios, este ajuste se dá exclusivamente para os nós de sincronização. Estes métodos podem ser separados em dois grupos, os que sempre aguardam o nó referência para ajustar seu relógio (correção adaptativa e ajuste analítico) e aqueles que, em alguns

momentos, esperam a informação do nodo de referência e em outros estimam o valor do relógio do nodo referência. A vantagem do primeiro grupo consiste em aperfeiçoar o comportamento dos nós receptores, ativando-os somente quando for necessário. Por outro lado, estimar o valor do relógio de referência traz a possibilidade de ajustar o relógio do nodo de sincronização sem precisar esperar a próxima mensagem de sincronismo, objetivando atingir uma precisão maior no sincronismo da rede.

4.1 CORREÇÃO ADAPTATIVA

A correção adaptativa é um método no qual o nó de sincronização utiliza o valor de referência temporal, que fora transmitido pelo nó servidor, para aproximar o seu relógio ao valor de referência. Analisando a diferença entre o relógio local e o relógio do nó de referência para aplicar modificações proporcionais.

4.1.1 Proposta

Este algoritmo mede a diferença entre os valores temporais e assim se adapta da melhor forma possível para convergir em um sincronismo, em seguida, manter este sincronismo o mais preciso possível. A Figura 6 Esboço do algoritmo de correção adaptativa.apresenta um esboço deste comportamento. Nela, é possível observar o método de sincronismo adaptativo operando.

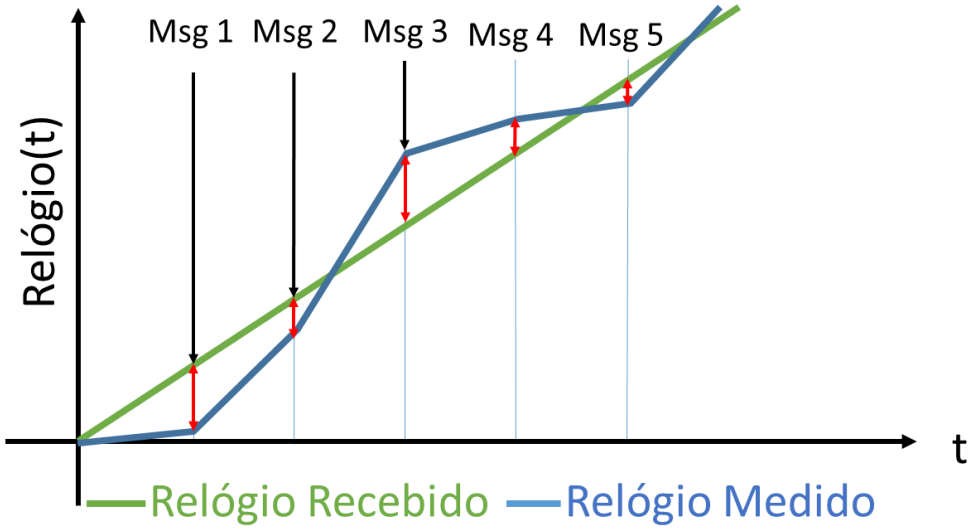

Figura 6 Esboço do algoritmo de correção adaptativa.

A fim de melhorar o sincronismo e evitar que o sistema oscile de forma harmônica, jamais atingindo o sincronismo, foi proposto um método que diminui a oscilação do relógio. Para isso, após a primeira intersecção das retas temporais, as frequências dos relógios são analisadas de modo que elas não tenham uma diferença muito grande, deixando a reta temporal do nó de sincronização o mais similar ao valor do relógio referência. Esta análise é feita pelo ΔTempo de cada dispositivo. ΔTempo é definido por (10), onde K é o número de iterações.

$$\Delta Tempo_k = Tempo_k - Tempo_{k-1} \quad (10)$$

4.1.2 Aplicação

O algoritmo de correção adaptativa tem três decisões principais a serem tomadas. Com o objetivo de fazer o nó de sincronização equiparar seu relógio ao relógio do nó referência o mais rápido possível, analisa-se a diferença entre os relógios. Se esta diferença for muito grande, o nó de sincronização ajusta a frequência do seu relógio para o máximo ou mínimo valor possível. Será também analisado qual dispositivo tem o relógio maior, ou seja, determinar qual nó está adiantado, ajustando assim o valor mínimo para o nó de sincronização – se este estiver adiantado ao nó referência. Esta diferença representa mais de 10 vezes a amplitude das configurações do ciclo do relógio. Após essa análise, verifica-se se a diferença entre os relógios é relativamente grande, usando uma métrica de 1,5 vezes da amplitude das configurações do relógio. Em caso positivo ocorre um ajuste grosso, analisando qual nó está adiantado ou atrasado, desta forma adiantando ou retardando o a frequência do relógio do nó de sincronização. No caso de os dois condicionais serem falsos ocorre o ajuste fino que mensura a diferença entre os relógios executando uma operação em laço, que realiza o ajuste fino utilizando uma métrica de 0,1 vez da amplitude total das configurações, para determinar o número de iterações do laço. Caso o ajuste fino seja usado, a diferença entre os Δrelógios não pode exceder 5%, como é possível ver na Figura 7. Estes valores são específicos para um determinado nó de sensor, representando assim os parâmetros a serem calibrados de acordo com o nó de sensor utilizado. Os valores, aqui indicados, foram empiricamente determinados para os nós de sensores utilizados nos experimentos realizados, que serão apresentados no capítulo 6.

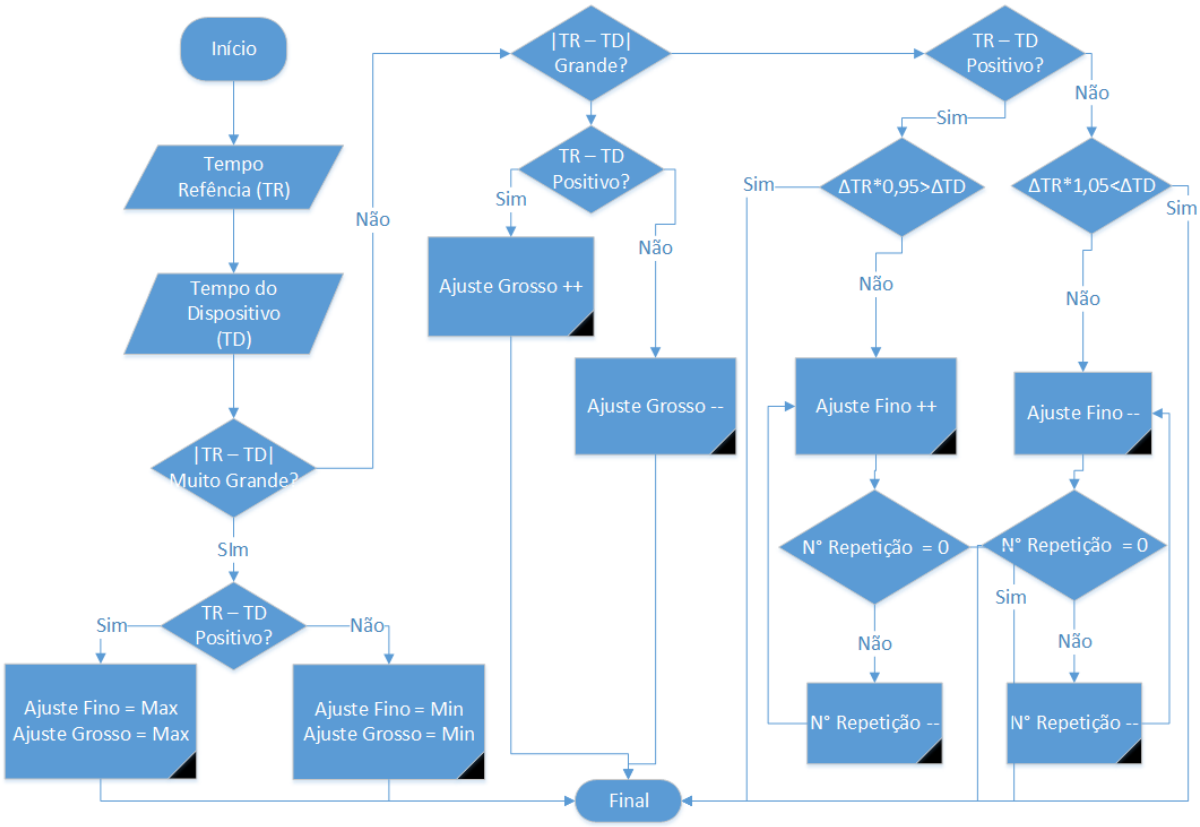

Figura 7 Fluxograma do algoritmo de correção adaptativa

4.2 PREDIÇÃO DE RELÓGIO

A predição de relógio é um algoritmo que tem como objetivo reduzir o número de transmissão de relógio em toda a rede. Inspirado por Panfilo & Tavella (2008), este algoritmo prediz o valor do relógio de referência. Para isso, o nó receptor analisa um conjunto de transmissões anteriores, conforme o registro temporal do nó referência, na tentativa de estimar o valor dos próximos registros temporais.

4.2.1 Proposta

Sabendo que as transmissões realizadas pelo servidor de relógio, para propagar seu relógio e assim estabelecer o sincronismo da rede, tem um grande impacto no desempenho energético, o que demanda processamento e recursos que poderiam ser empregados em outras tarefas. Em muitos casos, o pacote que contém a informação sobre o relógio atual do nó servidor não utiliza o seu tamanho máximo. Desta forma, é possível informar, também, aos nós de sincronização o intervalo de latência (neste trabalho será estabelecido que o intervalo de latência é o hiato entre duas transições da mensagem de sincronismo). Com esta informação, o nó de sincronização pode estimar o momento em que a próxima transmissão ocorrerá. A partir da associação do intervalo de latência com a evolução temporal do nó referência é possível estimar o valor do relógio do nó mestre. Isto posto, o dispositivo receptor, valendo-se deste tempo estimado, aplica o algoritmo de correção adaptativa, algoritmo anterior, para ajustar seu relógio no momento em que não ocorra transmissões da mensagem de sincronismo. É notória a existência de erro nessa estimativa, mas a cada nova transmissão do nó referência os dispositivos de sincronização ajustam seu referencial, deixando sempre uma margem de erro aceitável. A Figura 8 esboça o funcionamento deste algoritmo. A linha em verde com uma parte tracejada refere-se a curva do desenvolvimento do relógio de referência, a parte tracejada é onde ocorre a previsão de relógio. Como é possível observar, ao ocorrer a predição temporal, o nó executa muitas vezes a função de sincronização em um único intervalo de latência e, assim, deixa a unidade com uma maior precisão.

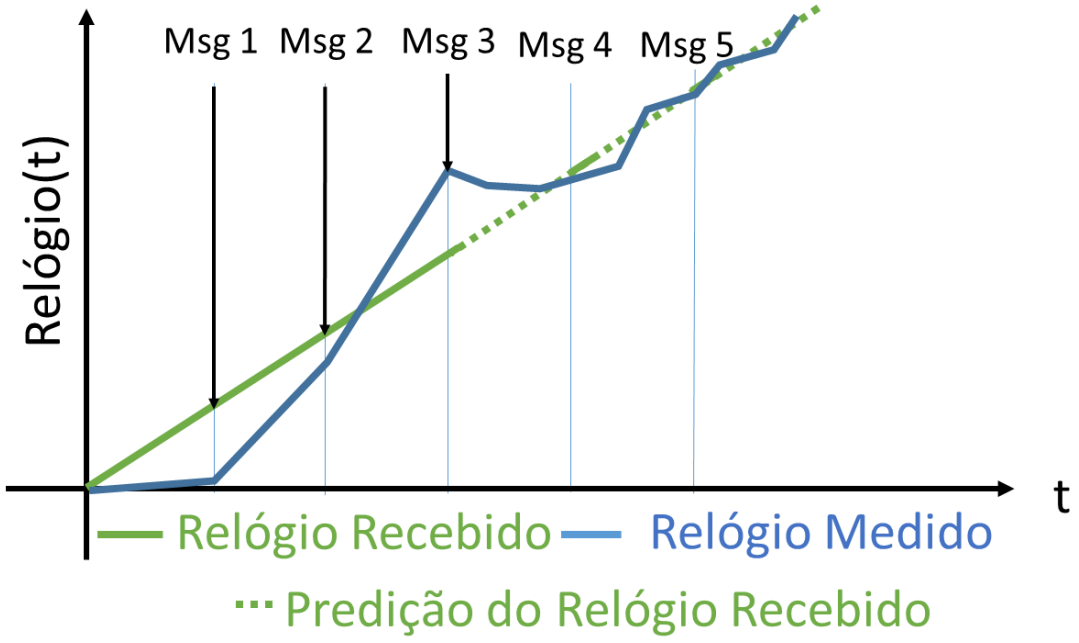

Figura 8 Exemplo de algoritmo de Predição de Relógio.

4.2.2 Aplicação

O algoritmo de predição de relógio analisa as últimas transmissões e estima o próximo valor do relógio do nó referência, observando as variações das transmissões do nó de referência. Esta predição faz uma análise de uma série infinita, em que a última medição da variação do relógio referência tem metade do peso desta série e os demais termos decaem de forma quadrática, como é possível notar nas equações 11. Esta fórmula tem como objetivo suavizar quaisquer medições equivocadas, falhas de transmissões ou perdas de pacotes, dando a mesma importância para a medição anterior e para a soma das demais.

$$Relógio\_Ref_{K+1} = Relógio\_Ref_{K} + \Delta Relógio\_Ref_{k} \quad (11)$$

Onde $\Delta\, Relógio\_Ref$ é definido como:

$$\Delta RTC\_Ref_{k} = \frac{\Delta RTC\_Ref_{k-1}}{2} + \frac{\Delta RTC\_Ref_{k-2}}{4} + \frac{\Delta RTC\_Ref_{k-3}}{8} + \frac{\Delta RTC\_Ref_{k-4}}{16} + \cdots \quad (12)$$

Outra forma de representar esta equação de série infinita é por uma média aritmética do ΔRTC_Ref anterior (ΔRTC_Ref_{k-1}) com os demais.

Para esta abordagem, o nó receptor usa a mesma função de correção adaptativa da seção anterior, a fim de obter o melhor resultado com a mesma latência. Como o sincronismo não é perfeito e os relógios não tem a mesma evolução temporal, o nó receptor termina seu intervalo de latência antes do valor estimado, estando apto a receber novas mensagens. Neste caso, a margem de erro é de pelos menos 5%. Uma vez que neste processo cada ciclo de ajuste é indivisível, optou-se por deixar uma margem fixa para que este não seja motivo para perda de pacotes.

4.3 CORREÇÃO ANALÍTICA

O algoritmo de correção analítica – que combina os protocolos FTSP e ATS (MARÓTI et al., 2004) e (SCHENATO; GAMBA, 2007) – visando melhorar ainda mais a precisão do sistema e reduzir o número de transmissões, pois combina a frequência dos relógios com o período de latência.

4.3.1 Proposta

Este método se vale de transmitir mais informação na mensagem de sincronismo. Também, utiliza as características de propagação de tempo do relógio do FTSP, contudo, sem votação para determinar o nodo referência, visto que, até o presente momento, este é determinado pelo gerente da rede. Este algoritmo, inspirado pelo protocolo ATS, utiliza o conceito de transmitir as informações da frequência do relógio do nodo referência para que os demais nodos se adaptem e o sincronismo seja o mais preciso possível. Como esta informação não é dada pelos fabricantes é preciso, antes de iniciar o processo de transmissão da mensagem de sincronismo, fazer um conjunto de procedimentos que discretizem as configurações do relógio que cada dispositivo possui. Como já visto, a configuração de cada relógio varia por diversos fatores, portanto a função de discretização se torna imprescindível toda vez que a rede seja estabelecida. Conforme a Figura 9, até que o relógio do nó conflua para a linha de relógio de referência, o algoritmo tenta combinar os relógios o mais rápido possível, logo em seguida, tenta obter a mesma inclinação que a referência, visando o melhor sincronismo possível.

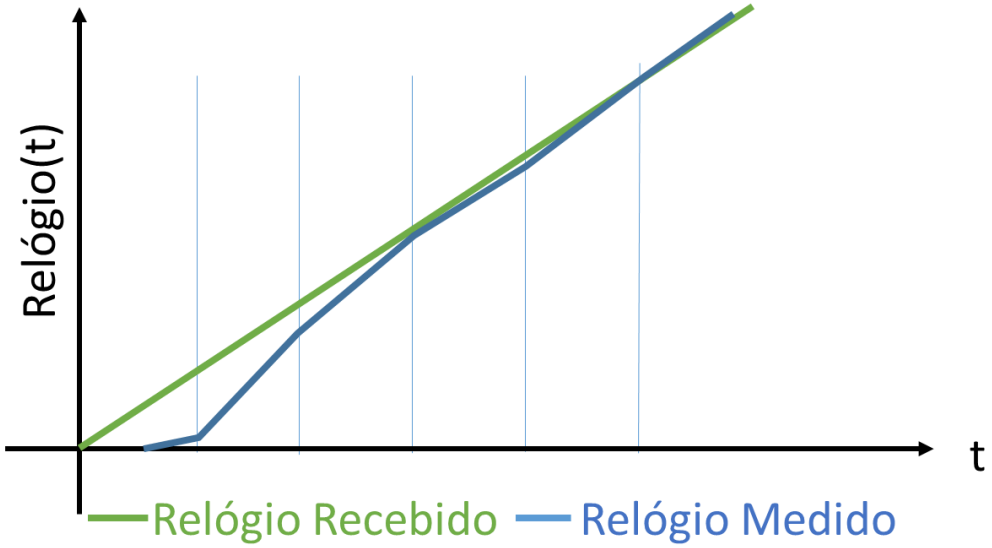

Figura 9 Exemplo de algoritmo de Correção Analítica

4.3.2 Aplicação

O algoritmo de correção analítica utiliza a informação de inclinação da reta temporal do nó servidor para atingir um melhor sincronismo da rede. Como ainda há espaço suficiente para esta informação, no pacote da mensagem de sincronismo, todos os dados são transmitidos na mesma mensagem.

A informação sobre a inclinação temporal não é informada pelo fabricante e depende de diversos fatores, como já citado na seção 2.3.3. Ainda, pode sofrer modificações no decorrer de seu funcionamento. Desta forma, quando este método é usado, o nó raiz e os demais nós fazem uma varredura medindo todas as configurações possíveis de inclinação temporal. Assim, obtém a informação mais atual possível. Esta função gera uma matriz que contempla todas as opções válidas da frequência do relógio. É importante ressaltar que cada dispositivo possui a sua tabela de inclinação e que elas não são iguais. De posse desta tabela, o nó referência pode transmitir sua inclinação atual, afim de alcançar melhor precisão de sincronismo.

Quando o nó de sincronização recebe a informação sobre a inclinação do nó referência ele aplica a (13) para que seu relógio convirja o mais rápido possível. Após, mantém o mais síncrono possível, pois busca a melhor opção entre as disponíveis na tabela.

$$Min(|RTC\_Projected - RTC_{ID} + t * RTC\_Tab[i][j]|) \quad (13)$$

Onde i e j são as opções de ajuste da frequência do relógio do nó de sincronização, então a nova configuração do relógio do nó de sincronização do relógio grosso e fino será igual a i e j respectivamente; RTC_{ID} é o relógio de dispositivo; RTC_Tab é a tabela que contempla todas as possibilidades de ajuste de relógio, t é o tempo de latência; e o $RTC_{Projected}$ é definido por:

$$RTC_{Projected} = RTC_{ID-1} + t * RTC\_Tab_{ID-1}[i][j] \quad (14)$$

No caso de *multi-hops*, o nó de sincronização se adapta ao nó de referência. Neste processo, ele modifica a inclinação do relógio. Após, este nó transmite sua mensagem de sincronização para seus filhos informando esta nova inclinação de relógio, com isso toda rede se sincroniza de forma mais rápida e harmônica que os métodos anteriores.

5 IMPLEMENTAÇÃO DOS MÉTODOS PROPOSTOS

Apresentados os algoritmos de sincronização, neste capítulo será apresentada a estrutura da implementação dos métodos propostos. É fundamental estabelecer uma comunicação estre os dispositivos. Assim, os métodos operacionais dos nós necessitam se corresponderem, visando estabelecer a comunicação necessária para alcançar o sincronismo. Apresentam-se os detalhes sobre os métodos operacionais dos nós e da estrutura em que se aplicam, para que o desenvolvimento e aplicabilidade destes métodos se tornem tangível.

5.1 MENSURANDO O ATRASO ENTRE NÓS SENSORES VIZINHOS

Como estes algoritmos de sincronismo focam em uma rede escalável, com múltiplos saltos e – uma de suas principais características – com capacidade de propagar a informação através de outros nós da rede; é fundamental que se determine o atraso entre os pares de nós, visto que estes atrasos serão somados ao registro temporal do relógio de referência. Esta função ocorre nas primeiras mensagens de sincronização. Deste modo, uma sequência de medidas será efetuada até que seja determinado o menor atraso possível entre o nó referência e o de sincronização – como o proposto por Cristian em (CRISTIAN, 1989).

Para medir o atraso o nó de sincronização espera por uma mensagem de sincronismo, após receber esta mensagem ele envia, ao nó referência, uma mensagem para medir a distância. Ao receber esta mensagem, o nó de referência, responde com uma mensagem de confirmação (*Acknowledgement*). O atraso é determinado pela diferença temporal entre o envio da mensagem, para medir a distância, e o recebimento da confirmação. Todo esse

processo é representado na Figura 10. Este atraso medido é correspondente à troca de duas mensagens e, por esta razão, é inferido que o atraso de propagação do relógio seja metade do valor medido.

Figura 10 Funcionamento do método para medir a distância.

É importante frisar que todos os métodos de sincronismo, desenvolvidos neste trabalho, ocorrem em pares – iniciando com o nó referência e seus adjacentes. Após os nós adjacentes medirem a distância, eles começam a propagar seus registros temporais para seus clientes, até que toda a toda rede esteja em sincronismo.

5.2 ESTRUTURA DA REDE

Como já mencionando, o foco deste trabalho é desenvolver um sistema de sincronismo com múltiplos saltos, isto é, uma rede em que nem todos os dispositivos terão comunicação

direta com o nó principal, por consequência, será preciso que os nós de sincronização também tenham a capacidade de enviar a mensagem de sincronismo para nós mais distantes do nó raiz.

Desta forma, o fluxo de informação sobre os registros temporais se inicia com o nó raiz e se propaga pela rede até os nós mais distantes. Como é possível observar na Figura 11. Antes do nó de sincronização transmitir o seu registro temporal, é preciso ter a informação sobre a distância entre este nó de sincronização e seu nó referência, gerando assim um fluxo lento de informação, como no protocolo FTSP.

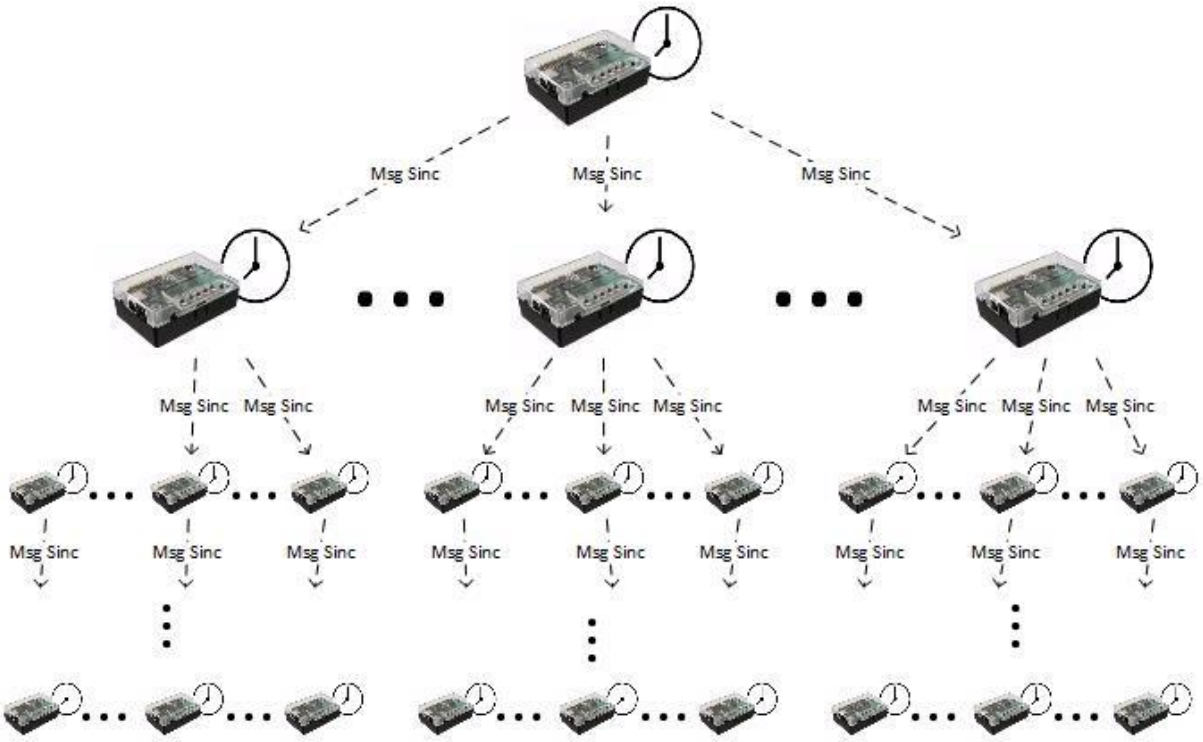

Figura 11 Funcionamento do método para medir a distância.

5.3 NÓ REFERÊNCIA

O nó que corresponde ao servidor temporal tem a função de enviar o seu relógio para os demais dispositivos a fim de estabelecer o sincronismo da rede. Neste trabalho, foi proposto que ocorra apenas uma transmissão por intervalo de latência, contudo este período pode ser modificado em tempo de execução, ou seja, quando o nó referência entender que é preciso melhorar o sincronismo, a frequência das transmissões será modificada. Para o servidor, este intervalo de latência se inicia no início da transmissão, após ele fica ocioso ou executando outra tarefa até o final deste período, como pode ser visto na **Figura** 12, este processo ocorre de forma cíclica e ininterrupta, como também está desenvolvido no Algoritmo 1.

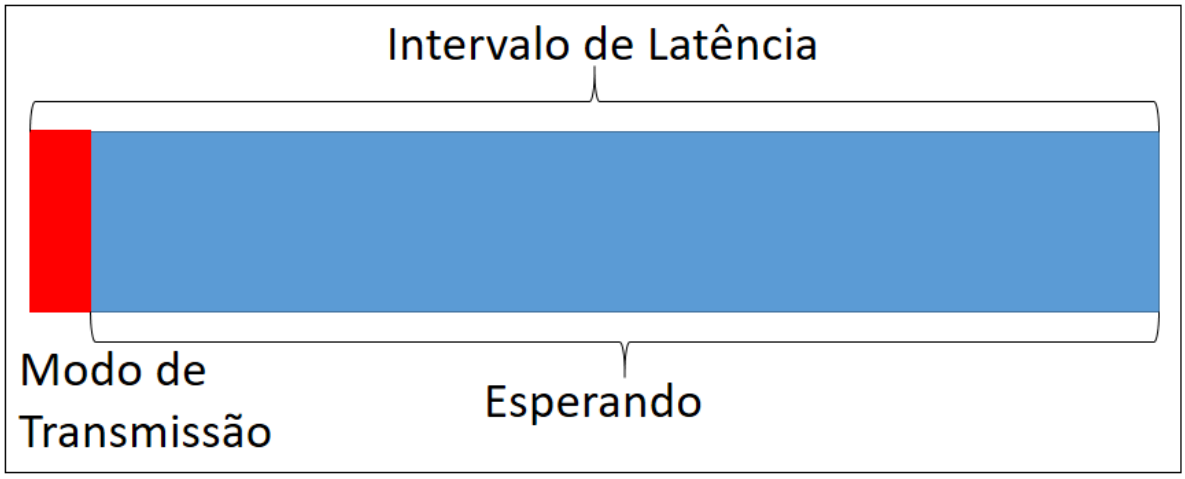

Figura 12 Modelo de operação do nó servidor, para a propagação do relógio.

Algoritmo 1: Operação do nó servidor, para a propagação do seu relógio

```
1.  Latency_period=;  // Intervalo de latência em milissegundos
2.  while(1){
3.     Send_Clock ();
4.     Wait(Next_transmission);
5.  }
```

5.4 NÓ DE SINCRONIZAÇÃO

O nó de sincronização é o dispositivo que irá se adaptar ao valor transmitido pelo nó referência para estabelecer o sincronismo da rede. Com o objetivo de estabelecer uma comunicação entre o nó de sincronização e o nó de referência, três métodos foram projetados.

O primeiro modelo proposto tem como objetivo manter o modo de recepção sempre ativo. Desta forma, ele sempre estará apto a receber novas mensagens de sincronismo. Após receber o registro temporal, o nó de sincronização ajusta seu relógio para se manter o mais próximo do relógio referência, e torna a esperar a próxima mensagem de sincronismo, como é possível notar na Figura 13, e no Algoritmo 2. Neste caso, o consumo de energia é maior que dos métodos seguintes, pois o dispositivo mantém sempre ativo o modo de recepção – um dos modos que mais consome energia. Este modelo de funcionamento do nó de sincronização é viável para os algoritmos de correção adaptativa e correção analítica.

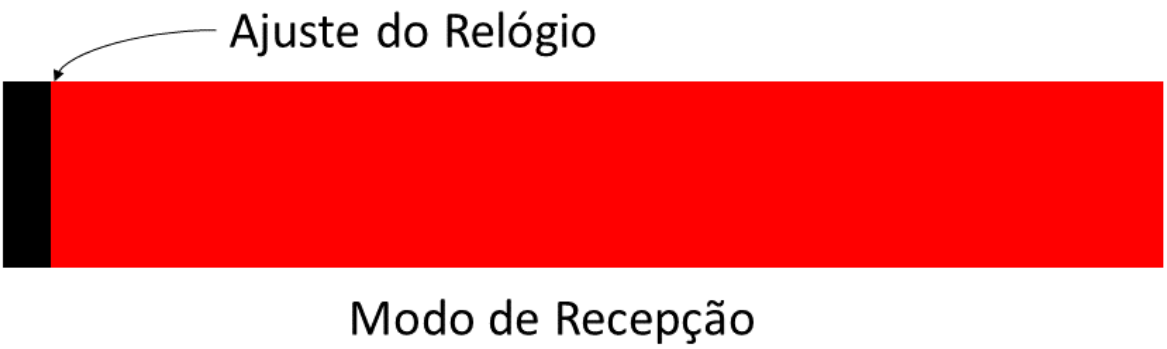

Figura 13 Modelo de operação do nó de sincronização que sempre está apto a receber a mensagem de sincronismo.

Algoritmo 2: operação do nó de sincronização, sempre apto a receber a mensagem de sincronismo

```
1.  while(1)
2.     if(lisent_msg_synch()){
3.        Main_Clock=get_Main_Clock();
4.        Sync_Adjusts_Clock(Main_Clock);
5.     }
6.  }
```

O segundo modelo objetiva consumir menos energia. Neste caso, o nó de sincronização, além de receber a informação sobre o *relógio* do nó referência, também recebe o intervalo de latência; assim é possível estimar a próxima transmissão. Por conseguinte, o nó referência pode permanecer no modo de baixo consumo (*Idle mode*) até o final do intervalo de latência. Uma vez que o tempo medido pelos dispositivos não é perfeito, é adicionada uma margem de erro, assim é considerado que o intervalo é um pouco menor que o verdadeiro. Quando um novo intervalo se inicia, este já começa com o modo de recepção ativo, apto a receber novos registros temporais. Ao receber a mensagem de sincronismo o dispositivo

emprega a função de ajuste de *relógio*, como é possível verificar na Figura 14. No Algoritmo 3, é possível identificar o desenvolvimento deste modelo. Onde, na linha 6, é adquirida a informação sobre o intervalo de latência e, na linha 8, o nó desativa o modo de recepção, ficando ocioso. Da mesma fora que o modelo anterior, este é viável apenas para os algoritmos de correção adaptativa e correção analítica.

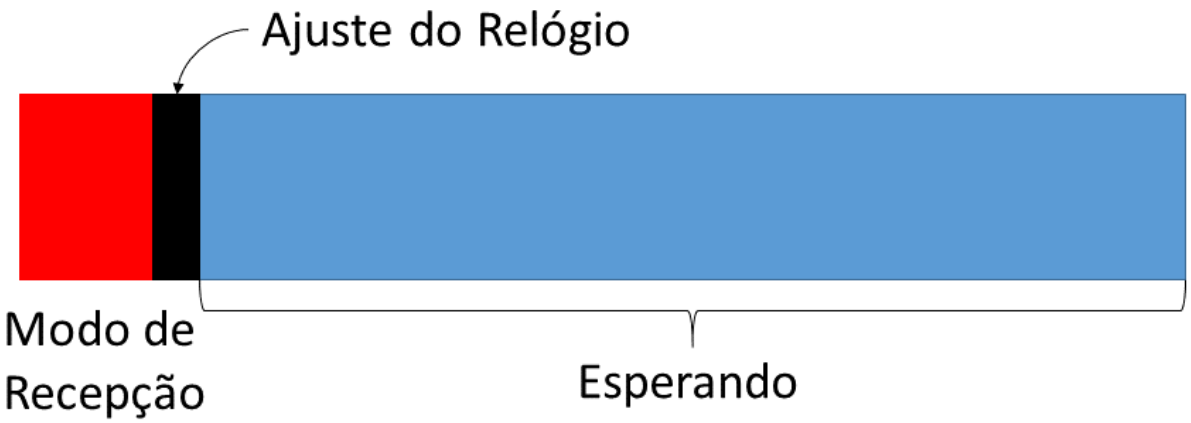

Figura 14 Modelo de operação do nó de sincronização que ligar o modo de recepção quando necessário.

Algoritmo 3: operação do nó de sincronização que ligar o modo de recepção quando necessário

```
1.  Latency_period=;  // Intervalo de latência em milissegundos.
2.  Margin_of_Error;  // Se adapta em função do intervalo de latência.
3.  while(1)
4.     if(lisent_msg_synch()){
5.        Main_Clock=get_Main_Clock();
6.        Latency_period =getLatency();
7.        Sync_Adjusts_Clock(Main_Clock);
8.        Wait(Next_transmission*(1- Margin_of_Error));
9.     }
10. }
```

Neste segundo modelo foi adotada uma margem de erro dinâmica, que se adapta em função do intervalo de latência, como pode ser visualizado na Tabela 2. O percentual da margem de erro diminui quando o intervalo de latência aumenta, tendo uma variação de 20 vezes do seu percentual, no entanto, o tempo da margem de erro aumenta apenas 3 vezes.

Para chegar nestes valores um conjunto de testes foram desenvolvidos, em que cada opção de intervalo de latência foi mensurada mais de mil vezes. O percentual de erro é o valor aproximado do dobro do desvio padrão, nesta análise. Esta margem de erro contempla mais de 99% dos casos medidos, alcançando uma maior segurança na recepção da próxima mensagem.

Tabela 3 Margem de Erro

Intervalo da Latência (s)	Margem de Erro	Tempo da margem de erro (ms)
1	10%	100
2	8%	160
5	4%	200
10	3%	300
15	2%	300
30	1%	300
60	0.5%	300

O último modelo visa apresentar uma melhor precisão em relação aos dois métodos anteriores. Ele tenta reduzir o número de transmissões estimando o relógio de referência, além de não deixar o modo de recebimento sempre ativo. Este modelo usa as mesmas informações sobre a próxima transmissão, como no modelo anterior. Além disso, este terceiro protótipo analisa a diferença entre as últimas transmissões do relógio do nó referência e prevê um valor aproximado para ele, ajustando o relógio com mais frequência, do que a frequência de transmissão do servidor temporal. Este modelo ainda apresenta a margem de erro para o fim do intervalo de latência, o que ocasiona uma espera antecipada da próxima transmissão, conforme apresentado na Figura 15. No Algoritmo 4 é possível identificar o desenvolvimento deste modelo. Onde, na linha 6, é adquirida a informação sobre o intervalo de latência, como no modelo anterior, mas, neste caso, na linha 8, ocorre a verificação se não é a primeira mensagem de sincronismo e se o intervalo de latência não foi alterado em relação a mensagem anterior – para assim, estimar o valor do nó referência. A função de predição ocorre da linha 10 a 12, em que faz a estimativa sobre o valor do relógio do nó referência e utiliza a função de correção adaptativa para ajustar o relógio do nó de sincronização. Por se tratar de um modelo de que aplica diversas correções em um único período de latência. Este modelo de funcionamento do nó de sincronização é viável apenas para o algoritmo de predição de relógio.

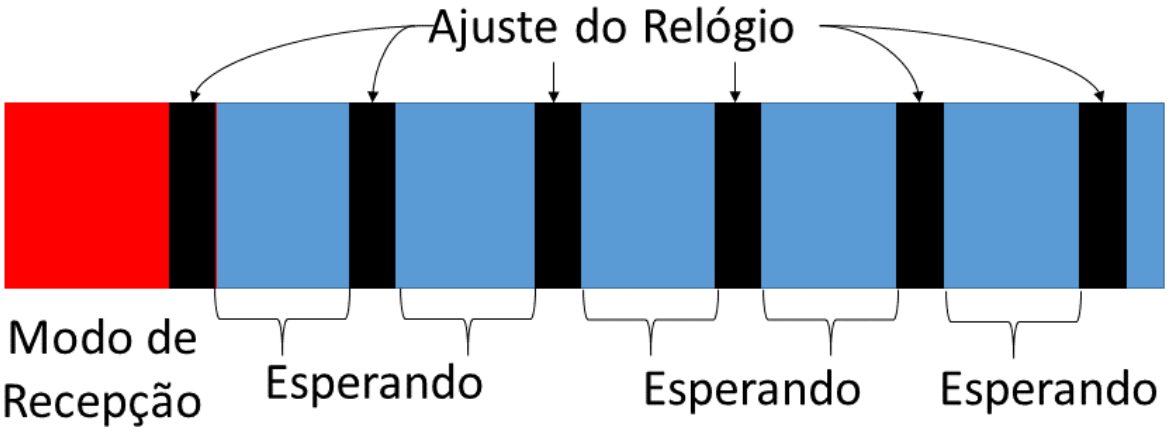

Figura 15 Modelo de operação do nó de sincronização com a predição de relógio.

Algoritmo 4: operação do nó de sincronização com a predição de relógio.

```
1.  Latency_period=;  // Intervalo de latência em milissegundos.
2.  Second_Transmition_Flag=0;//É preciso 2 transmissões para este modo
3.  while(1)
4.     if(lisent_msg_synch()){
5.        Main_Clock=get_Main_Clock();
6.        Latency_period =getLatency();
7.        Sync_Adjusts_Clock(Main_Clock);
8.        if(Latency==OLD_Latency & Second_Transmition_Flag==1){
9.           times_to_repeat=Latency/1000;
10.          for(i=1;i<(times_to_repeat*0.95);i++){
11.             wait(next_ adjustment)//one second
12.             Sync_Adjusts_Clock(New_value_Main_Clock);
13.          }
14.       }
15.    OLD_Latency=Latency;
16.    Second_Transmition_Flag=1;
17.    OLD_Main_Clock=Main_Clock;
18. }
```

Ao se analisar os métodos propostos e projetados para o nó de sincronização, é possível notar a equivalência entre o terceiro modelo, aqui proposto, e o modelo de predição de *relógio*. No entanto, os métodos de correção adaptativa e o de correção analítica não representam diretamente os métodos restantes.

5.5 AJUSTE DE RELÓGIO

Para alcançar o sincronismo dos nós, este trabalho optou por modificar a frequência dos relógios da rede, respeitando as condições apresentadas no capítulo 2. Como já foi mencionado, este trabalho foi desenvolvido com o equipamento MC1322x. Desta forma, a leitura dos valores do relógio é no endereço de memória conhecido como RTC_COUNT, o método de altear a frequência do relógio se dá modificando os valores do endereço de memória conhecido como RINGOSC_CNT. Neste ajuste de relógio, é possível executar dois tipos de ajustes o fino e o grosso. O ajuste fine ajusta um conjunto de capacitor que cada passo altera a capacitância em 160 fF. O ajuste grosso, também altera o mesmo conjunto de capacitores, desta vez, altera a capacitância em 1 pF como a Figura 16 apresenta. Em suma, é possível alterar a frequência deste relógio de 2KHz até 32KHz, segundo o manual, contudo em testes foi possível varias de 2KHz até 27KHz.

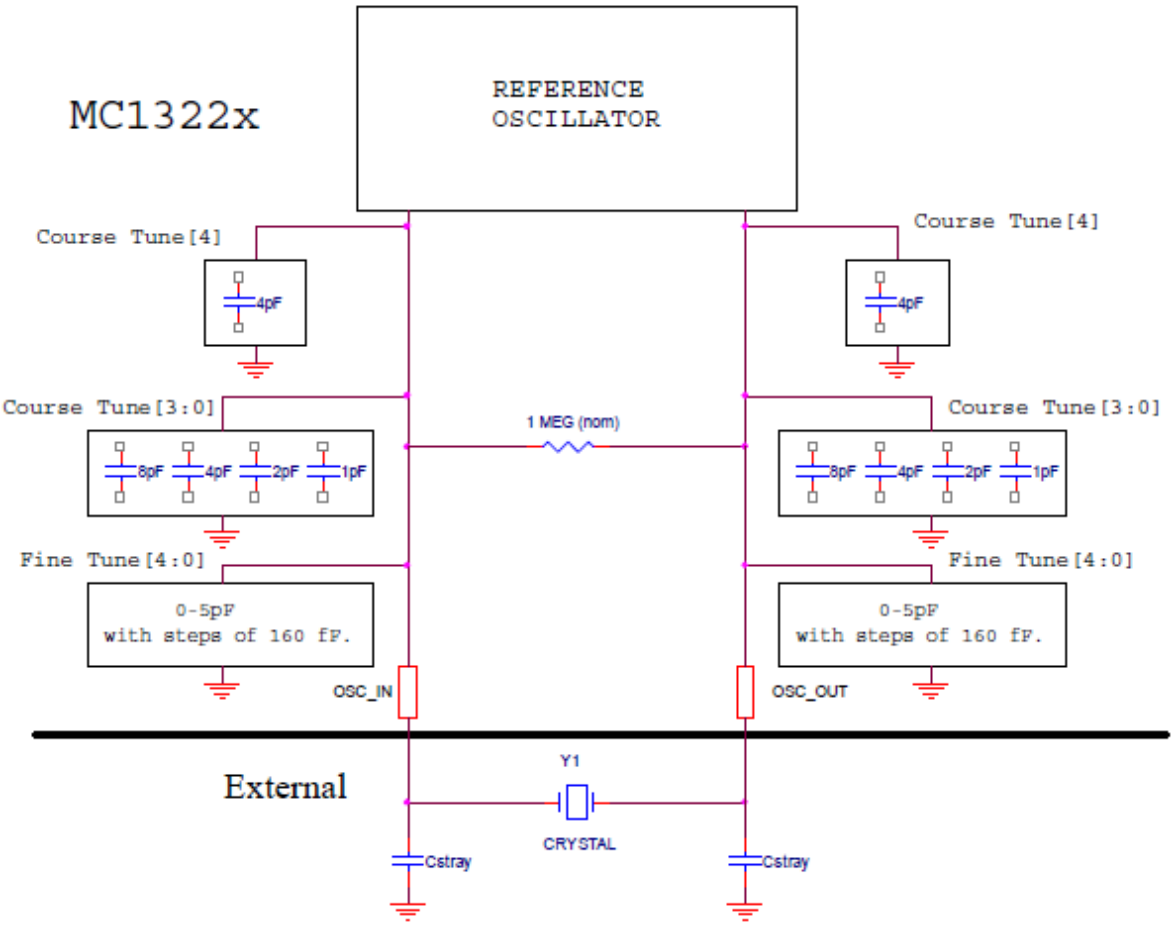

Figura 16 Representação do conjunto de capacitores que modificam a frequência do relógio.

Como há duas opções de ajuste, o ajuste fino e o grosso, e eles são indiferentes é possível representar todas as opções de configuração de relógio por uma matriz 18x7, como mencionado na seção 4.3. Um exemplo de matriz de configuração é a Tabela 4, nela é apresentado quantas oscilações, variações do cristal, ocorrem em um segundo.

Tabela 4 Exemplo de configuração de frequência de relógio

Ajusto grosso / Ajuste fino	0	1	2	3	4	5	6
0	27251	18152	13792	11045	9361	8028	7078
1	25273	17262	13277	10716	9124	7853	6942
2	23558	16449	12794	10401	8898	7684	6812
3	22068	15721	12351	10105	8686	7525	6688
4	20747	15049	11934	9826	8479	7372	6566
5	19589	14433	11548	9562	8286	7226	6450
6	18544	13862	11183	9314	8098	7088	6338
7	17613	13341	10844	9079	7921	6954	6230
8	16759	12845	10519	8851	7747	6819	6123
9	16002	12399	10219	8641	7585	6693	6022
10	15304	11976	9933	8437	7429	6572	5925
11	14669	11585	9666	8243	7279	6454	5831
12	14080	11216	9411	8058	7136	6342	5738
13	13542	10874	9171	7881	6997	6234	5651
14	13035	10551	8941	7713	6864	6128	5565
15	12572	10250	8722	7553	6737	6028	5482
16	12126	9952	8507	7392	6610	5924	5398

6 EXPERIMENTOS E RESULTADOS

6.1 SIMULAÇÃO

Este capítulo apresenta um conjunto de simulações que contemplam os métodos apresentados nos capítulos anteriores, objetivando compará-los em relação ao grau de sincronismo dos dispositivos simulados e o tempo necessário para atingir o sincronismo em um único salto, ou em múltiplos saltos. Essa análise é fundamental para que essa comparação seja a mais justa possível, é preciso que todas as condições iniciais – relógio do servido, relógio dos dispositivos e a configuração dos relógios – sejam iguais em todos os casos.

Visando ser o mais próximo possível ao caso real está simulação foi desenvolvida na linguagem Python, simulando um conjunto de dispositivos (Servidores e Clientes) que, além de simular a evolução temporal de cada dispositivo, também simula um erro de precisão para cada um deles. Este erro é maior que o dobro da variância dos relógios coletados, na mesma análise da precisão temporal descrita no capítulo 5. Foram realizados dois tipos de simulação para os três métodos, uma que analisa com um único salto e outra que analisa com saltos múltiplos.

6.1.1 Único Salto

Esta simulação foi projetada com um único servidor que transmite o seu registro temporal para seus clientes ao mesmo tempo. Cada um utiliza seu algoritmo de sincronismo, ajustando o seu relógio conforme julgue necessário, como é possível notar na Figura 17. O

servidor é inicializado com o valor de 20 segundos e os nós clientes com meio segundo, simulando um servidor que é ativado muito antes dos clientes.

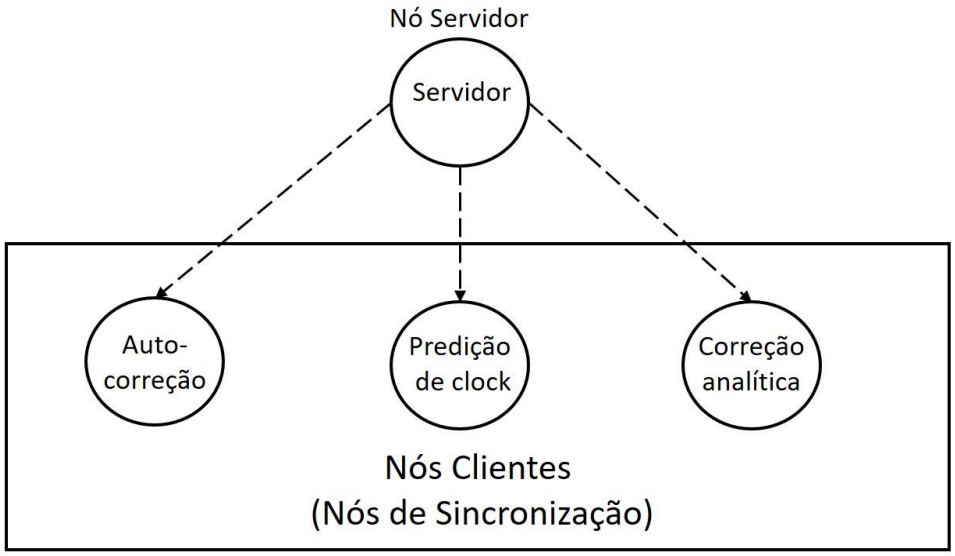

Figura 17 Esboço da simulação de um único salto.

A Figura 18 apresenta a simulação dos valores de relógio de cada algoritmo. Nela é notável que todos os métodos convergem no mesmo instante – 10 iterações ou 100 segundos. A Figura 19 apresenta a mesma figura anterior, mas com *zoom* para identificar melhor a precisão temporal de cada modelo. Nela é possível identificar que a precisão da correção adaptativa é a pior de todas, com defasagem de até 3 segundos. Já a precisão do algoritmo de predição de relógio apresenta um erro de aproximadamente 200 milissegundos, mas o melhor desempenho está na correção analítica. Ela mantém sua defasagem muito próxima de 100 milissegundos.

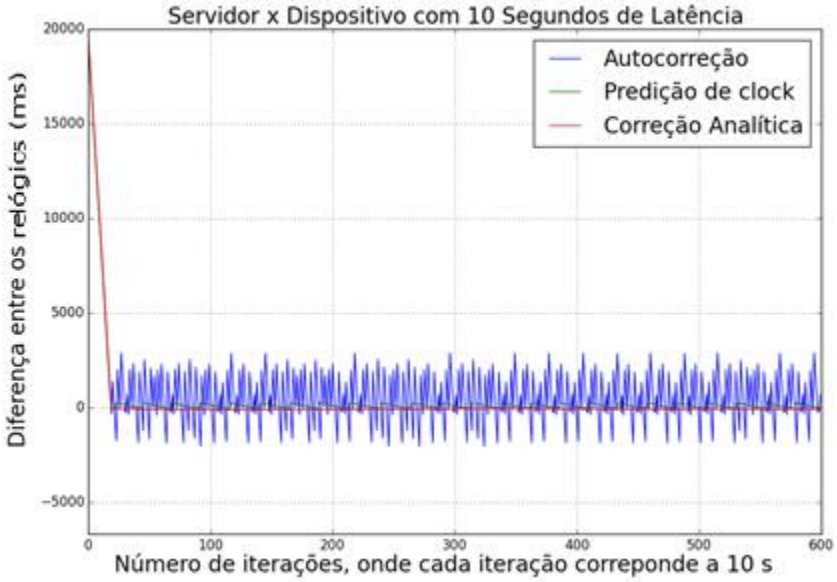

Figura 18 Simulação de todos os métodos com 10 segundos de latência.

Figura 19 *Zoom-in* na simulação de todos os métodos com 10 segundos de latência

6.1.2 Multipolos Saltos

Estas simulações foram projetadas para testar o sincronismo em cascata, desenvolvendo uma simulação para cada algoritmo de sincronismo. Nelas, há um servidor que transmite o seu registro temporal para o primeiro cliente, que em seguida transmite para o segundo, e assim sucessivamente até atingir o décimo dispositivo. Com os mesmos valores iniciais da simulação de único salto, cada cliente recebe o registro temporal de seu nó referência – ou o anterior –, como é possível notar na Figura 20.

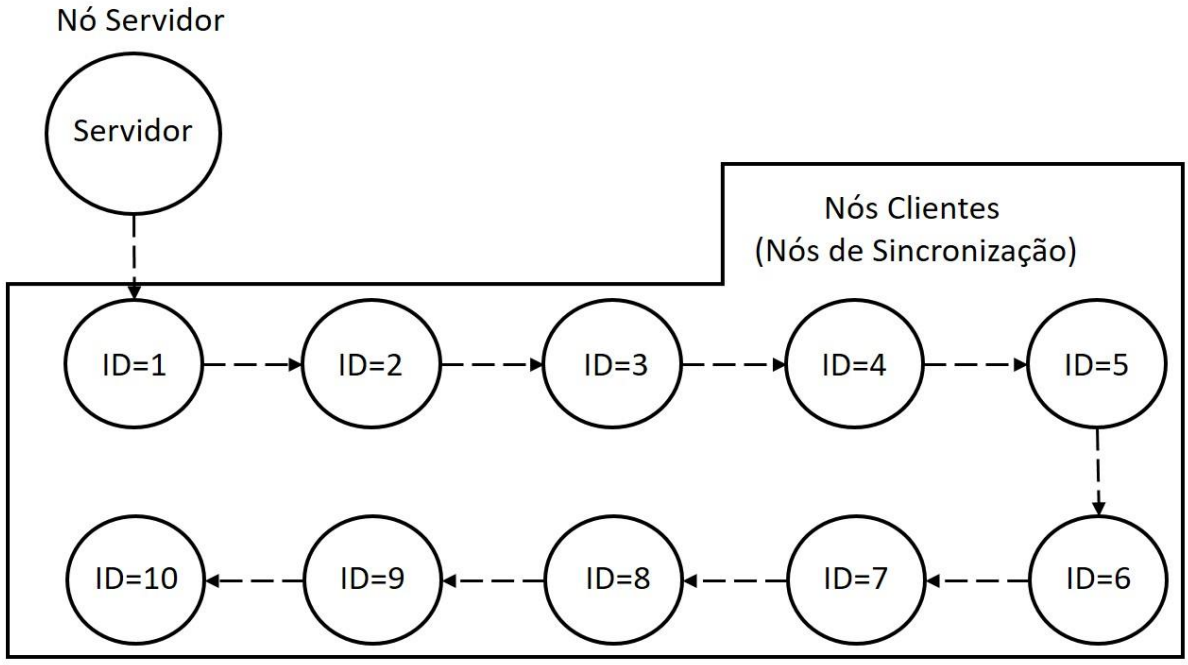

Figura 20 Esboço da simulação de múltiplos saltos

Para facilitar o entendimento, apenas os nós com ID igual a 1, 2, 5 e 10 serão exibidos nos gráficos; já os nós 3 e 4 tem valores intermediários aos nós 2 e 5. Por conseguinte, os nós

6, 7, 8, e 9 possuem valores intermediários aos nós 5 e 10. Esta simplificação pode ser feita, assumindo que os erros são acumulativos – passando de um nó para seus filhos.

6.1.2.1 Correção adaptativa

A Figura 21 apresenta a simulação dos valores de relógio, que executam o algoritmo de correção adaptativa. Nela, é possível notar que, como os dispositivos utilizam a informação do registro temporal de seu antecessor, os erros se acumulam e se reforçam. Como observado, a diferença temporal do décimo nó com o servidor chegou a mais de 80 segundos. Mesmo que anteriormente esta diferença tenha sido muito próxima a zero. Também, é possível notar que em 600 iterações, ou seja, uma hora e quarenta minutos, os dispositivos 5 e 10 não atingem o sincronismo e com isso o sistema, com dez dispositivos, também não.

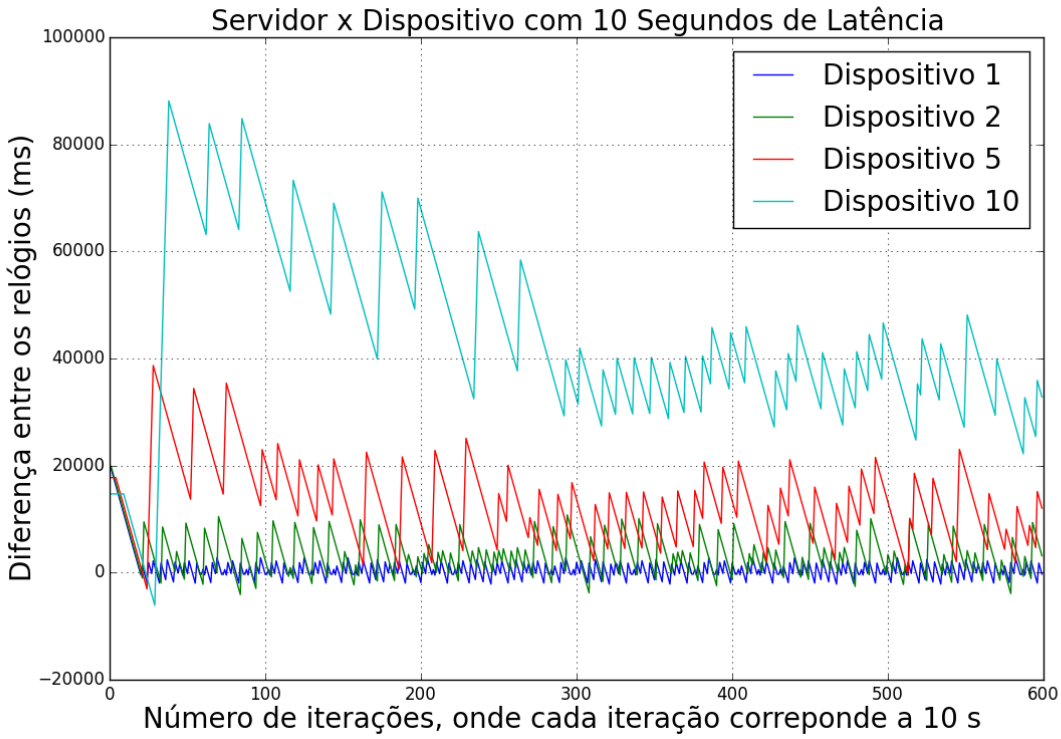

Figura 21 Simulação de sincronismo em cascata do algoritmo de autocorreção com 10 segundo de latência.

6.1.2.2 Predição de relógio

A Figura 22 apresenta a simulação dos valores do relógio que executam o algoritmo de predição de *relógio*. Nela, é possível notar que a precisão, comparando com o caso anterior, é muito maior e que em menos de 40 iterações, ou seja, seis minutos e quarenta segundos, os dispositivos 1 2 3 4 5 atingem o sincronismo. Ainda, neste caso, o dispositivo 10 não atingiu o sincronismo, uma vez que, até o final da simulação, ele apresentou uma defasagem temporal

com o servidor maior que 3 segundos em uma latência de 10 segundos. Mesmo assim, tendo uma precisão muito maior que no caso anterior.

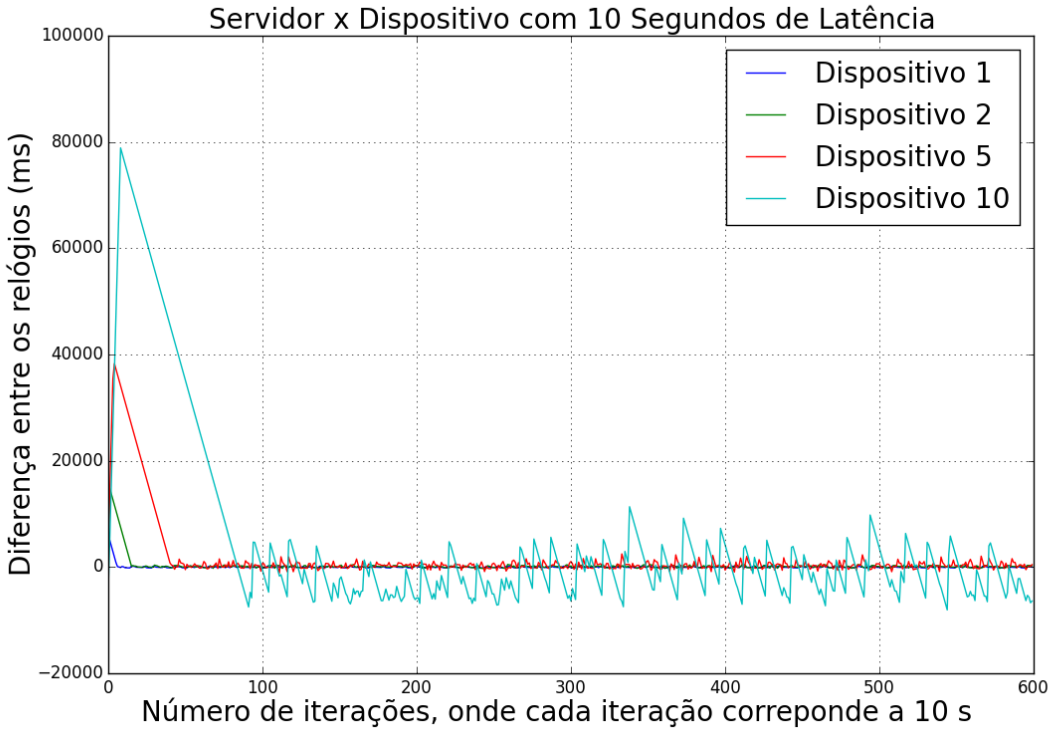

Figura 22 Simulação de sincronismo em cascata do algoritmo de predição de relógio com 10 segundo de latência.

6.1.2.3 Correção analítica

A Figura 23 apresenta a simulação dos valores que executam o algoritmo de correção analítica. Nela, é possível notar que a precisão, comparando com os dois casos anteriores, é muito maior e que em menos de 20 iterações, ou seja, três minutos e vinte segundos, todo o sistema está sincronizada. Ao analisar esta figura com *zoom*, Figura 24, nota-se que a

defasagem temporal do sistema é inferior a 50 milissegundos. Desta forma, conclui-se que este algoritmo teve o melhor desempenho até aqui.

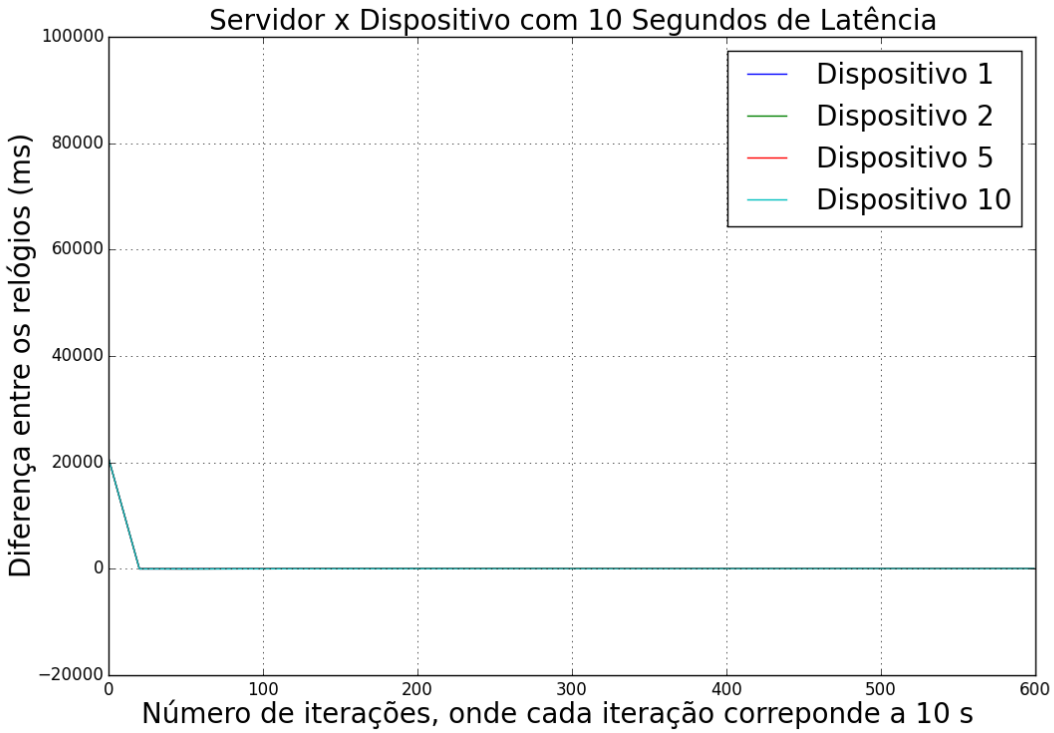

Figura 23 Simulação de sincronismo em cascata do algoritmo de correção analítica com 10 segundo de latência.

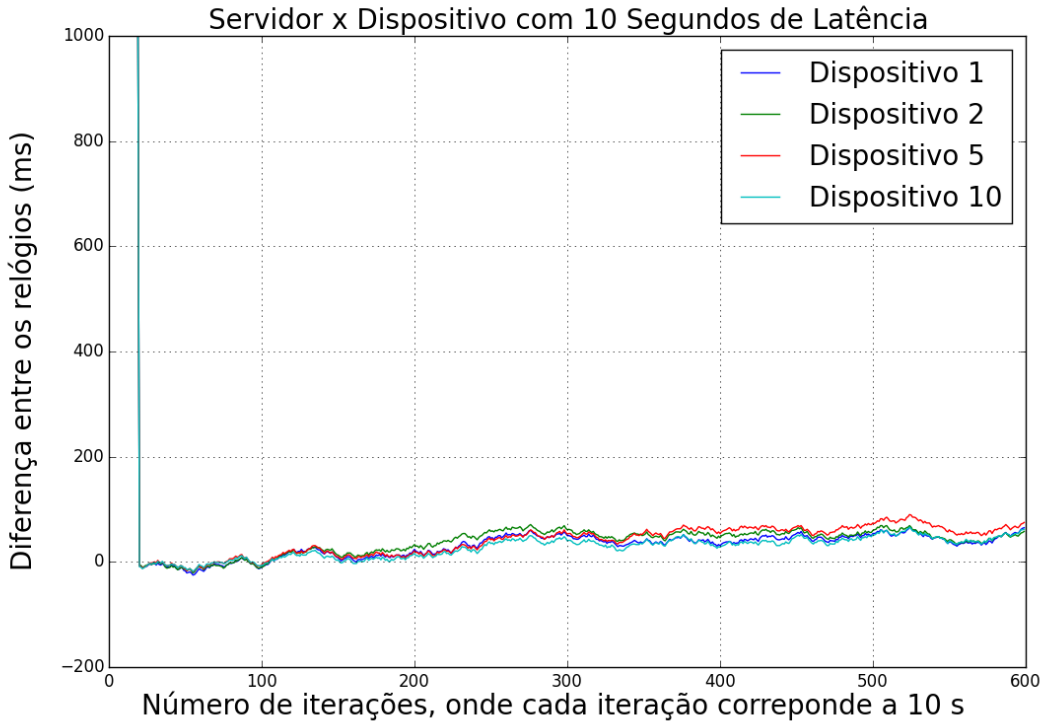

Figura 24 Simulação de sincronismo em cascata do algoritmo de correção analítica com 10 segundo de latência, com zoom.

Em todas estas simulações é importante ressaltar que houve características que não serão notadas em um ambiente real, um exemplo disso, é a opção por utilizar uma única tabela de valores relógios, para todos os dispositivos, visando obter uma simulação mais justa. Este fato não ocorre em ambiente de testes com dispositivos reais. Desta forma, no caso da correção analítica, a vantagem se acentua por alcançar uma maior precisão, diante da igualdade de tabela. A defasagem temporal foi eliminada, mas em um ambiente real existe uma defasagem entre um dispositivo receber a mensagem de sincronismo; ajustar seu relógio;

e transmitir uma nova mensagem de sincronismo. Outra característica que se ressalta – em comparação aos testes reais – é a evolução temporal dos relógios, em um ambiente real esta evolução se dá de forma contínua e linear, mas nesta simulação todas as evoluções temporais se deram de forma brusca e fragmentada, pois havia uma função que atualizava todos os valores de cada dispositivo simulado, contudo, esta simplificação apresenta pouco interferência nas simulações, uma vez que, os erros simulados representam um cenário pior que o real.

6.2 EXPERIMENTOS COM DEMONSTRADOR

Nesta seção serão apresentados e analisados os experimentos para avaliar a precisão e a eficiência energética de cada um dos métodos propostos no capítulo 4. Nestes experimentos foram utilizados um conjunto de nós sensores cujas características serão apresentadas, bem como descritas suas configurações. Na sequência, serão abordadas as configurações dos experimentos realizados.

6.2.1 Nó sensor

Foram utilizados, na realização dos experimentos, nós sensores de comunicação sem fio, tanto no nó raiz quanto para os demais nós. Os nós sensores utilizados foram da família MC1322x, que contém um transceptor integrado de 2,4 GHz baseado no protocolo IEEE 802.15.4. Eles são equipados com uma unidade programável de microcontrolador, de baixo consumo, baseado em ARM7 de 32 bits e que opera em até 26MHz, todavia é tipicamente configurado para operar em 24 MHz. Desta forma, é possível executar programas

desenvolvidos na linguagem C e utilizando o software IAR Embedded Workbench empregado em conjunto com um Segger J-Link, que conformam um sistema de depuração contendo uma interface JTAG.

Este trabalho utilizou um pré-programa desenvolvido pela Freescale®, `connectivity_test`, para os dispositivos da família MC1322x, adaptado aos propósitos deste trabalho para alterar o ciclo de oscilação do cristal de cada dispositivo. Ao alterar este ciclo, também, altera-se o ciclo do relógio, que é o meio utilizado para estabelecer o sincronismo nestes dispositivos. Os aparelhos MC13224V, que pertencem a família MC1322x, utilizam um cristal oscilador que gera uma frequência típica para o relógio, podendo variar de 2 a 25 kHz, consequentemente a maior precisão deste trabalho é 0.04 milissegundos. Este cristal oscilador tem duas maneiras de ajuste: o ajuste grosso e o ajuste fino. O primeiro pode alterar a capacitância de um conjunto de capacitores em 1 pF a cada carga; já o segundo pode alterar a capacitância deste mesmo conjunto de capacitores em 1 fF (FRESCALE, 2012).

6.2.2 Configurações dos experimentos

Com o objetivo de avaliar os métodos propostos e desenvolvidos, visando identificar o quanto a rede permanece sincronizada em diversos casos, os experimentos foram realizados utilizando nós de sensores sem fio – de aparelhos comercializáveis – que fazem parte do *kit* de desenvolvimento da Freescale® (MC13224V). Estes experimentos são divididos em dois tipos: i) cliente-servidor e ii) propagação do sincronismo em cascata. No experimento cliente-

servidor existem dois dispositivos, em que um deles tem a função do nó servidor de tempo, que fornece o relógio de referência; e outro que exerce a função de nó de recepção (nó de sincronização), objetivando ajustar a frequência seu relógio para que o tempo marcado fique o mais similar possível ao tempo do nó referência, tornando assim a rede sincronizada.

No primeiro experimento, cliente-servidor, foram usados quatro intervalos de latências diferentes, ou seja, intervalos de transmissão da mensagem de sincronismo por parte do nó servidor. Para cada um dos métodos desenvolvidos foram testados 1, 10, 30 e 60 segundos de intervalo de latência. Nesses casos, o nó servidor de tempo usou um *loop* para fazer a transmissão da mensagem de sincronismo. O nó de sincronização pode usar três algoritmos de sincronização (Correção adaptativa, Predição de relógio e Correção Analítica). Para todos os casos testados foram executados 120 iterações que correspondem a 120 transmissões de mensagens de sincronismo para o nó servidor, bem como a 120 execuções dos algoritmos para o nó de sincronização.

Para o segundo experimento, propagação do sincronismo em cascata, foi usado um conjunto de quatro dispositivos, entre eles um foi servidor de tempo e os demais foram nós de sincronização. Para este experimento, houve uma hierarquia de propagação de relógio, isto é: o nó servidor transmitiu para o nó de sincronização com ID 1, que por sua vez transmitiu seu registro temporal para o nó de sincronização com ID 2, que também transmitiu sua informação temporal para o nó de sincronização de ID 3. Como representado na Figura 25. Neste caso, a análise é feita em relação à latência da rede, o objetivo desse experimento é

medir a diferença entre os relógios dos nós de sincronização com o relógio do nó servidor, após um longo período de tempo, para verificar se eles ainda estão sincronizados.

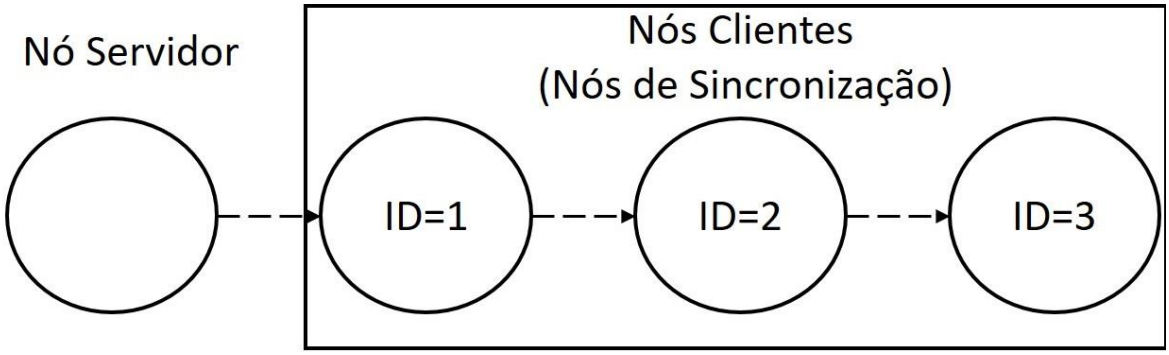

Figura 25 Esboço da simulação em cascata.

6.2.3 Resultados experimentais

Uma vez apresentadas as configurações dos experimentos, passa-se a análise dos resultados com o objetivo de verificar o comportamento de todos os métodos e identificar qual algoritmo melhor se comporta em cada um dos casos. Para isto, foi usado um conjunto de testes que transmitem a mensagem de sincronismo com latências de 1, 10, 30 e 60 segundos, com a configuração de único salto e saltos múltiplos. Como já citado, estes testes se desenvolveram a partir de 120 iterações, para observar se o sincronismo é alcançado e o mesmo se mantém.

6.2.3.1 Cliente-Servidor

Este conjunto de testes foi desenvolvido com uma estrutura de cliente-servidor, onde um nodo propaga seu registro temporal para o nodo de sincronização, o qual se ajusta

conforme o algoritmo selecionado. Nas seções subsequentes será analisado à precisão do sincronismo em relação ao tempo de latência dentro desta estrutura – cliente-servidor.

6.2.3.1.1 Um segundo de latência

O teste executado, para todos os algoritmos, foi com uma latência de 1 segundo, ou seja, uma vez por segundo cada dispositivo recebeu o registro temporal do nó servidor e com esta informação ajustaram os seus relógios. Para facilitar a análise, todos os gráficos desta subseção serão da mesma escala, com os limites superiores e inferiores de cinquenta milissegundos.

A Figura 26 representa os dados coletados da correção adaptativa. Nela é possível observar que, após estabelecer o sincronismo, o algoritmo apresentou um resultado muito próximo de zero, com uma defasagem média de 20 milissegundos com uma latência 1 segundo, mais de 98% de precisão.

Figura 26 Resultado do teste da Correção adaptativa com 1 Segundo de latência

A Figura 27 representa os dados coletados da predição de relógio. Nela é possível observar que o algoritmo apresentou um resultado muito similar ao de correção adaptativa, uma precisão de mais de 98%, com os mesmos 20 milissegundos de defasagem, após este sistema estabelecer o sincronismo. Isso se deve ao fato que com uma latência de 1 segundo a função de predição de relógio não é utilizada, pois para estes testes o mecanismo de predição é ativado com uma frequência de um segundo. Como as transmissões ocorrem nesta frequência, logo este princípio é adotado para que não haja perdas de mensagens do nó referência, com isso a função de predição fica inativa.

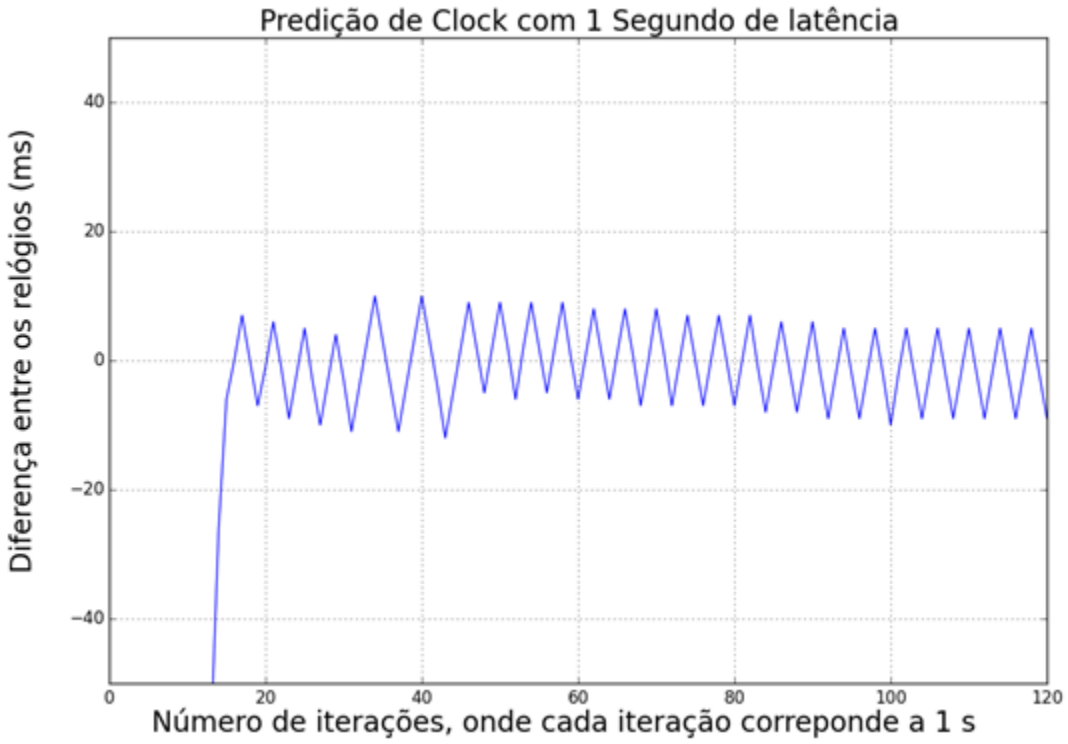

Figura 27 Resultado do teste da Predição de relógio com 1 Segundo de latência

A Figura 28 representa os dados coletados da correção analítica. Nela é notório que o algoritmo apresentou um resultado melhor que os anteriores. Neste caso, a correção analítica apresentou o melhor resultado.

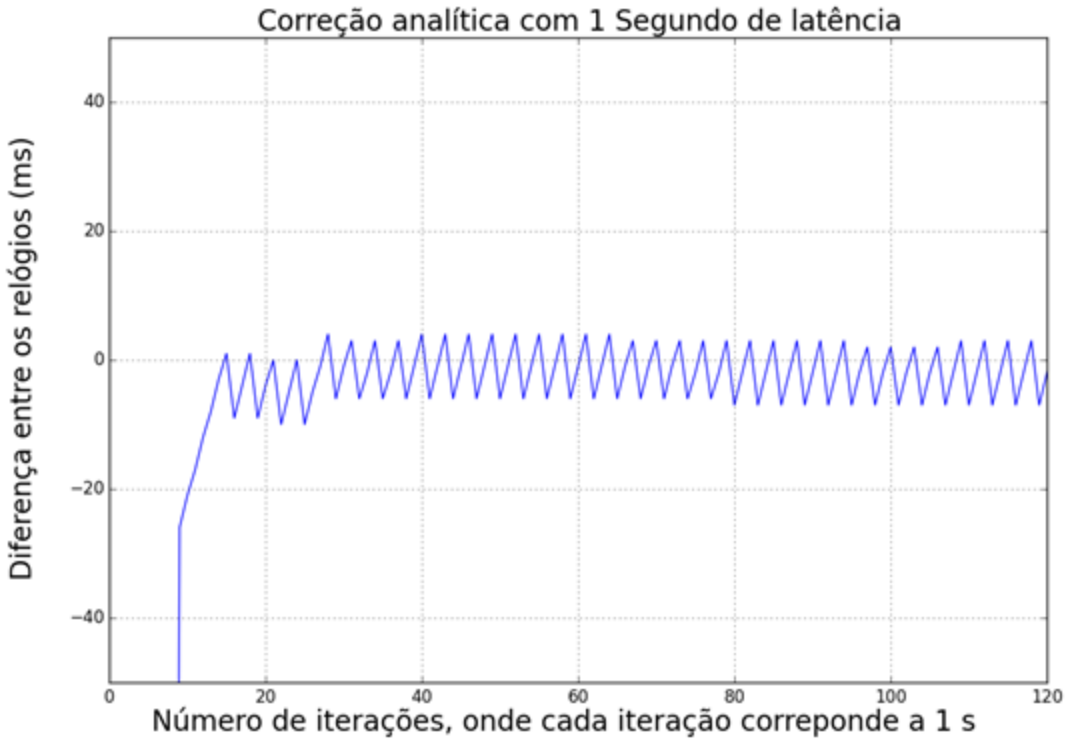

Figura 28 Resultado do teste da Correção analítica com 1 Segundo de latência

6.2.3.1.2 Dez segundos de latência

Com o objetivo de reduzir as transmissões e melhorar o desempenho energético, este teste foi desenvolvido utilizando a mesa configuração do teste anterior, alterando, apenas, o intervalo de latência para 10 segundos e, assim, determinar qual melhor se adapta a esta nova característica. Para facilitar a análise, todos os gráficos desta subseção serão da mesma escala, com os limites superiores e inferiores de setecentos e cinquenta milissegundos.

A Figura 29, representa os dados coletados da correção adaptativa. Após estabelecer o sincronismo, ela apresenta uma defasagem temporal média de 250 milissegundos, contudo aponta quase 600 milissegundos de erro temporal – no pior caso – com uma precisão de aproximadamente de 94%. Como o esperado, este teste apresenta resultados piores em comparação ao mesmo teste realizado com 1 segundo de latência. O fato inesperado é que o erro não se desenvolve de forma linear, pois apresentou um erro quase três vezes pior que o esperado de 200 milissegundos.

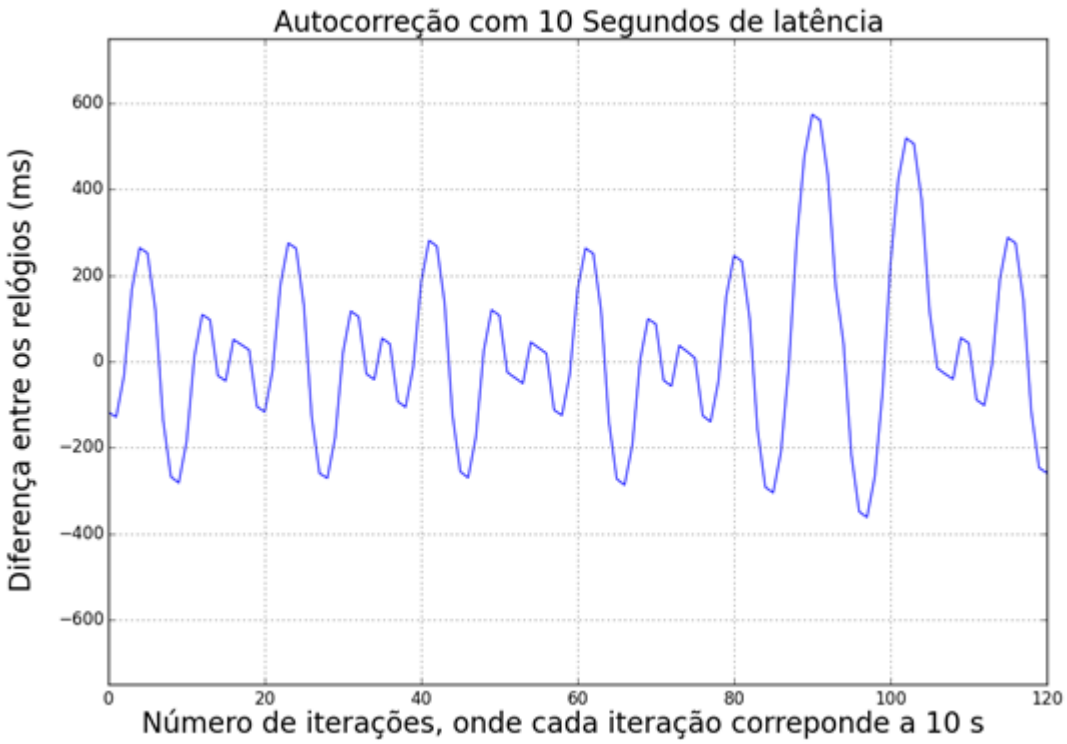

Figura 29 Resultado do teste da Correção adaptativa com 10 Segundos de latência

A Figura 30, representa os dados coletados da predição de relógio. Ela apresenta uma defasagem temporal média de 600 milissegundos, ou seja, uma precisão de aproximadamente de 94%, após estabelecer o sincronismo. Contudo, como é possível observar, a sua equiparação temporal tende a se manter a menos de 500 milissegundos, em relação ao valor do relógio de referência. Este erro pode ser proveniente de uma imprecisão na aferição do atraso entre os dispositivos ou por uma predição equivocada. Desta forma, este método apresenta um desempenho ligeiramente pior que o anterior.

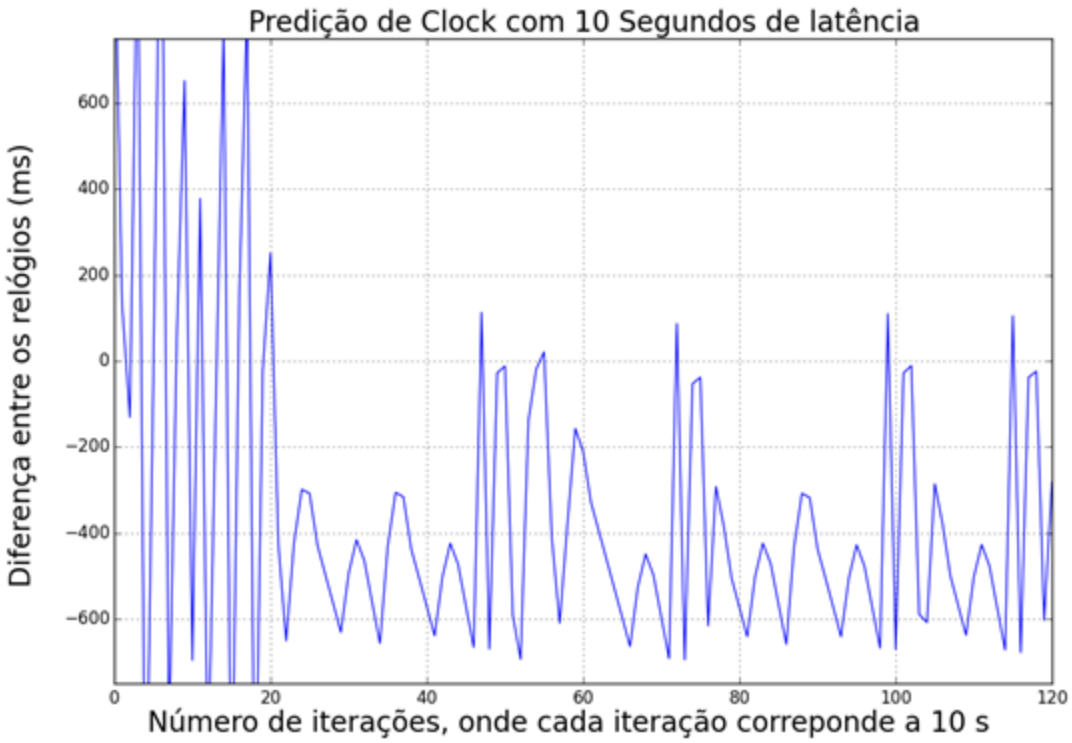

Figura 30 Resultado do teste da Predição de relógio com 10 Segundos de latência

A Figura 31 representa os dados coletados da correção analítica. Após estabelecer um sincronismo, ela apresenta uma defasagem temporal média de 50 milissegundos. É notório que, mesmo tendo uma acréscimo de nove vezes no intervalo de latência, o erro apresentou uma diferença de apenas uma vez e meia, alcançando a marca de 99,5% de precisão. Assim, este método apresenta o melhor desempenho em relação aos anteriores.

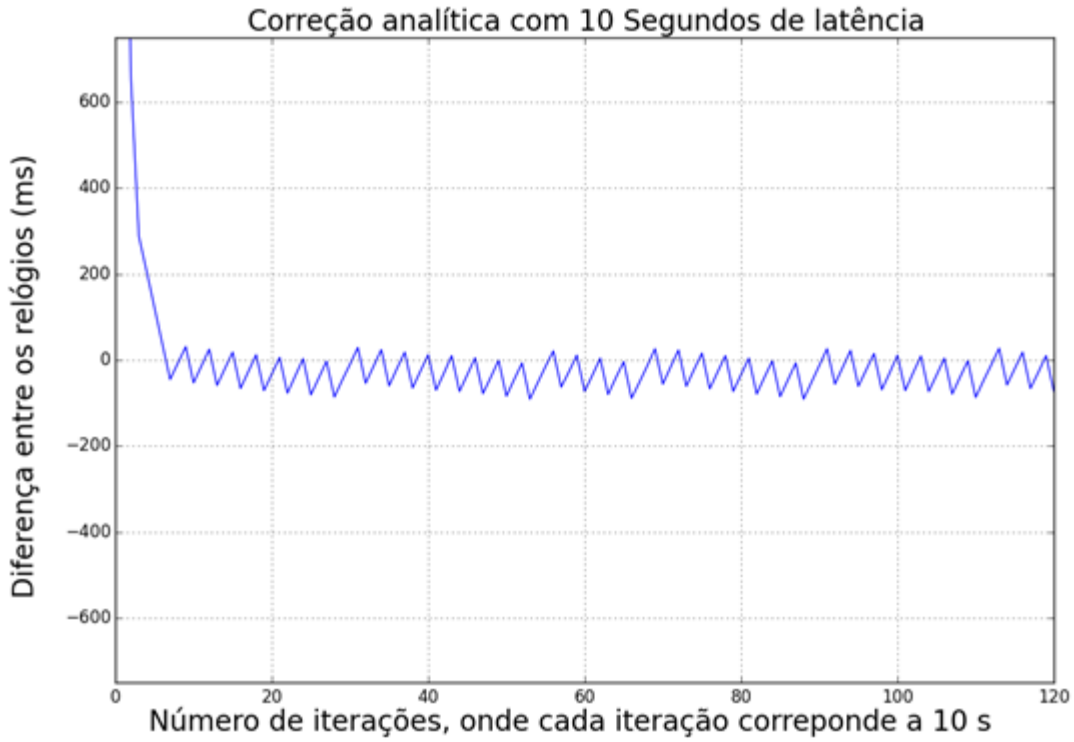

Figura 31 Resultado do teste da Correção analítica com 10 Segundos de latência

6.2.3.1.3 Trinta segundos de latência

Com o objetivo de reduzir ainda mais as transmissões, mirando em aplicações que transmitam informação esporadicamente, este teste foi desenvolvido utilizando as mesmas

configurações dos testes anteriores – só alterando o intervalo de latência para 30 segundos, para determinar qual melhor se adapta e esta nova característica. Para facilitar a análise, todos os gráficos desta subseção serão da mesma escala, com os limites superiores e inferiores de sete segundos e meio.

A Figura 32, representa os dados coletados da correção adaptativa. Nela é possível notar que a diferença entre os relógios é maior que sete segundos, alcançando uma precisão de 75% – erro muito superior ao aceitável. Desta forma, podemos afirmar que este algoritmo, como a latência de 30 segundos, não atingiu o sincronismo.

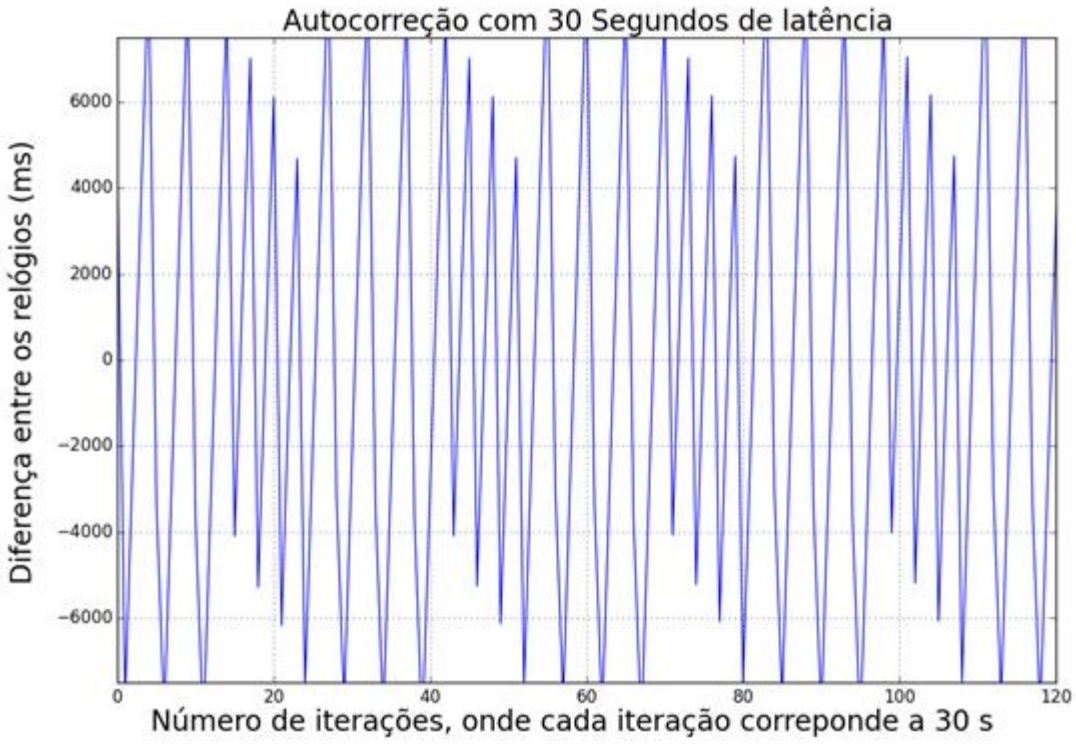

Figura 32 Resultado do teste da Correção adaptativa com 30 Segundos de latência

A Figura 33, representa os dados coletados da predição de relógio. Ela apresenta uma defasagem temporal média de 400 milissegundos, atingindo uma precisão de 98,5%, após estabelecer um sincronismo. E, assim, se mantém até o final dos testes. Este caso apresenta um desempenho 17 vezes melhor que o caso anterior.

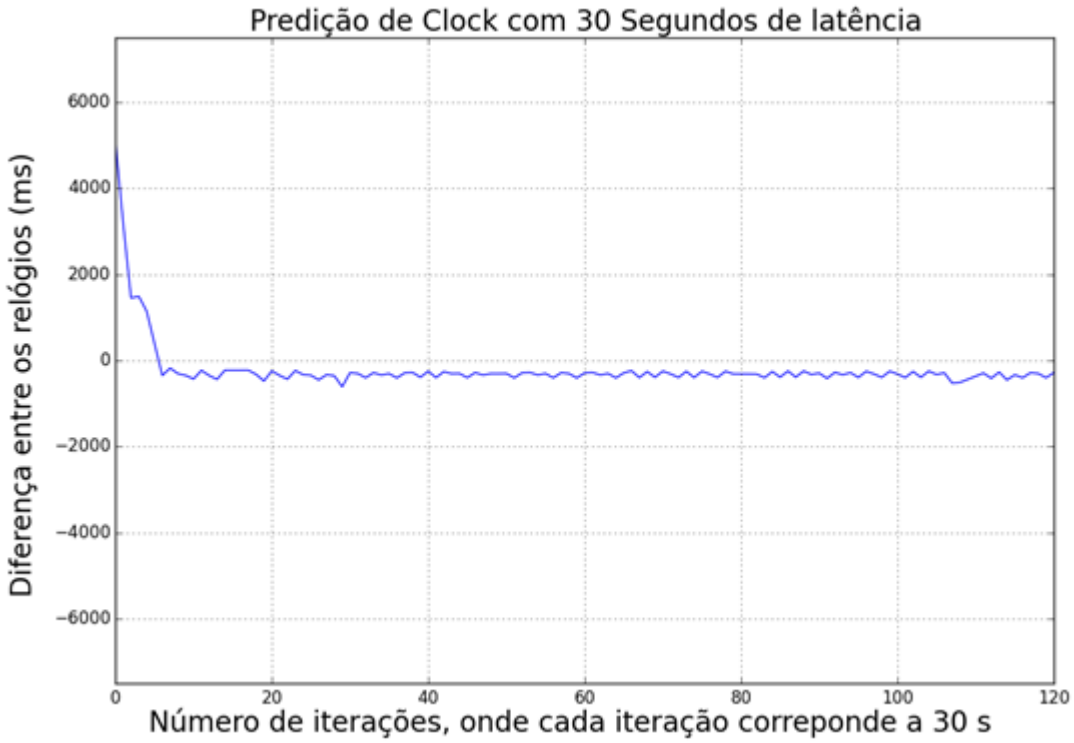

Figura 33 Resultado do teste da Predição de *relógio* com 30 Segundos de latência

A Figura 34, representa os dados coletados da correção analítica. Ela conquista 99,3% de precisão temporal, apresentando uma defasagem de tempo média de 200 milissegundos, após estabelecer um sincronismo. Isto posto, é notório um desempenho duas vezes melhor

que no caso anterior. Esta defasagem está quase imperceptível no gráfico. Desta forma, este método apresenta o melhor desempenho em relação aos anteriores.

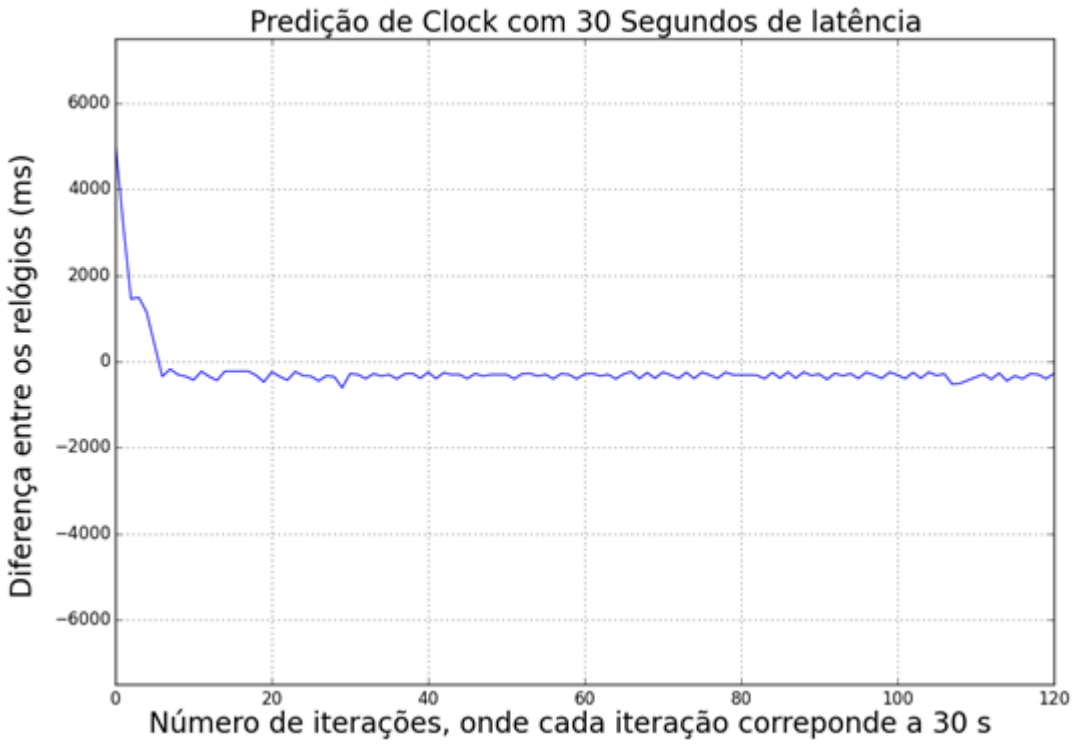

Figura 34 Resultado do teste da Correção analítica com 30 Segundos de latência

6.2.3.1.4 Sessenta segundos de latência

Com a intenção de aumentar ainda mais a vida útil dos dispositivos, ao mesmo tempo estabelecendo ainda menos comunicação nesta rede, foi desenvolvido este teste que contém 60 segundos de intervalo de latência – utilizando as mesmas configurações dos testes anteriores. Foi estabelecido que, nesta subseção, todos os gráficos serão da mesma escala – com os limites inferiores e superiores de dez segundos.

A Figura 35, representa os dados coletados da correção adaptativa. Ela apresenta uma defasagem temporal maior de dez segundos, atingindo uma precisão temporal de 83% – defasagem temporal muito superior ao aceitável. Assim, podemos afirmar que este algoritmo, com a latência de 60 segundos, não atingiu o sincronismo.

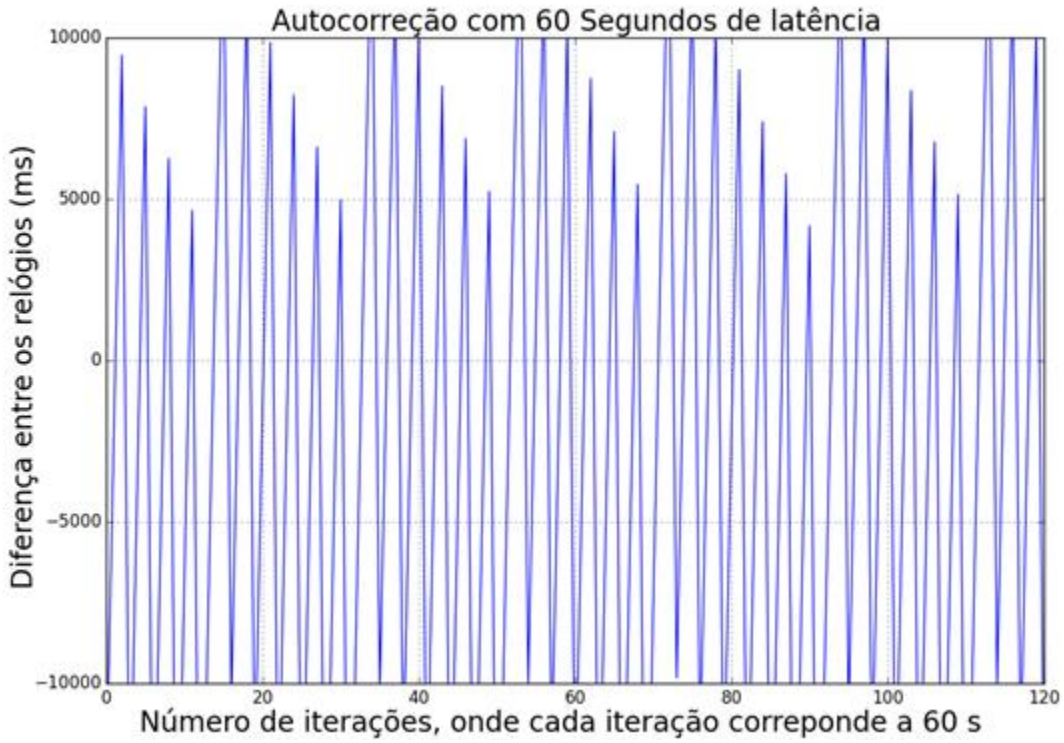

Figura 35 Resultado do teste da Correção adaptativa com 60 Segundos de latência

A Figura 36, representa os dados coletados da predição de *relógio*. Ela apresenta um erro de sincronismo médio de 400 milissegundos, alcançando 99,3% de precisão temporal, após estabelecer um sincronismo. E, assim, se mantém até o final dos testes. Portando, este

caso obteve, em relação ao anterior, uma performance superior – aproximadamente 25 vezes melhor.

Figura 36 Resultado do teste da Predição de relógio com 60 Segundos de latência

A Figura 37, representa os dados coletados da correção analítica. Ela conquista 99,1% de precisão temporal, apresentando uma defasagem média de tempo de 500 milissegundos, após estabelecer um sincronismo. Isto posto, o método que aponta o melhor desempenho, com a latência de 60 segundos, é a predição de relógio. Contudo, na correção analítica, observa-se um comportamento ligeiramente inferior.

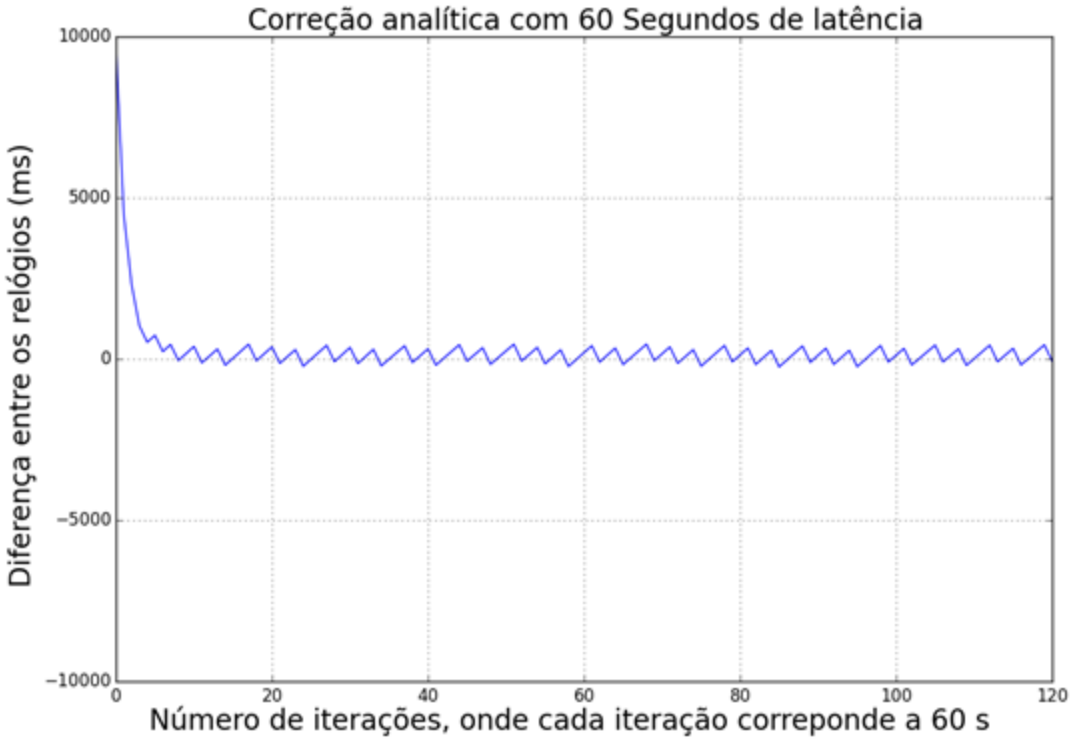

Figura 37 Resultado do teste da Predição de *relógio* com 60 Segundos de latência

Após a análise dos casos cliente-servidor, fica claro que o algoritmo de correção adaptativa apresenta o pior resultando, em muitos dos testes, ele não consegue atingir o sincronismo desejado. Não obstante, os demais casos se comportam de forma muito similar, em que, conforme o caso, algum se destaca se destaca positivamente. Sendo assim, não é possível determinar qual destes métodos se comportam melhor no sistema cliente-servidor.

O próximo conjunto testes tem como objetivo analisar outras características, bem como a acumulação de erros em cada *hop*, para assim decidir quais dos métodos é o mais recomentado.

6.2.3.2 Propagação do sincronismo em cascata

Além de analisar o comportamento dos algoritmos, foram propostos outros testes com o objetivo de investigar o comportamento quando a rede possui diversos caminhos de comunicação, em que o registro temporal seja enviado de um dispositivo a outros através de diversos *hops*. Estes testes foram desenvolvidos utilizando as configurações da propagação em cascata com quatro dispositivos, onde um transmite o seu registro temporal para o primeiro nó cliente – nó de Id 2, na sequência, este envia o seu registro temporal ao segundo nó cliente – Id 3. Por fim, o nó de Id 3 transmite o seu registro temporal para o último nó, denominado de Id 4 – gerando um fluxo de informação do nó servidor até nó mais distante.

6.2.3.2.1 Um segundo de latência

Para todos os algoritmos, este conjunto de testes foram executados com uma latência de 1 segundo, sendo assim, ocorre uma transmissão da mensagem de sincronismo a cada segundo. Desta maneira, cada dispositivo recebeu o registro temporal do nó referência – nó servidor ou nó anterior – e com esta informação ajustaram os seus *relógios*. Para facilitar a análise, todos os gráficos desta subseção serão da mesma escala, com os limites superiores e inferiores de quinhentos milissegundos.

A Figura 38 representa o teste com algoritmo de correção adaptativa, nela é possível notar que a rede apresenta uma defasagem temporal acumulada maior que 1 segundo, ou seja, maior que o intervalo de latência. Desta forma, é possível afirmar que este sistema não está sincronizado. A Figura 39, que representa o teste com o algoritmo predição de relógio, apresenta um resultado muito similar ao anterior, como nos testes do Cliente-Servidor, pois com a latência de um segundo o modo de predição não é ativado. A Figura 40 Resultado do teste em cascata com Correção analítica e 1 Segundo de latência, referente ao algoritmo de correção analítica, aponta uma melhora em relação aos dois testes anteriores, visto que tanto o erro acumulado quanto o erro médio são muito menores. Desta forma, com este intervalo de latência, este algoritmo atinge o sincronismo. Analisando estas três figuras, é notório que o melhor desempenho apresentado foi o da correção analítica. Pois, como é possível observar, os erros acumulados nestes casos são de um segundo, 700 milissegundos e 50 milissegundos para correção adaptativa, predição de relógio e correção analítica respectivamente.

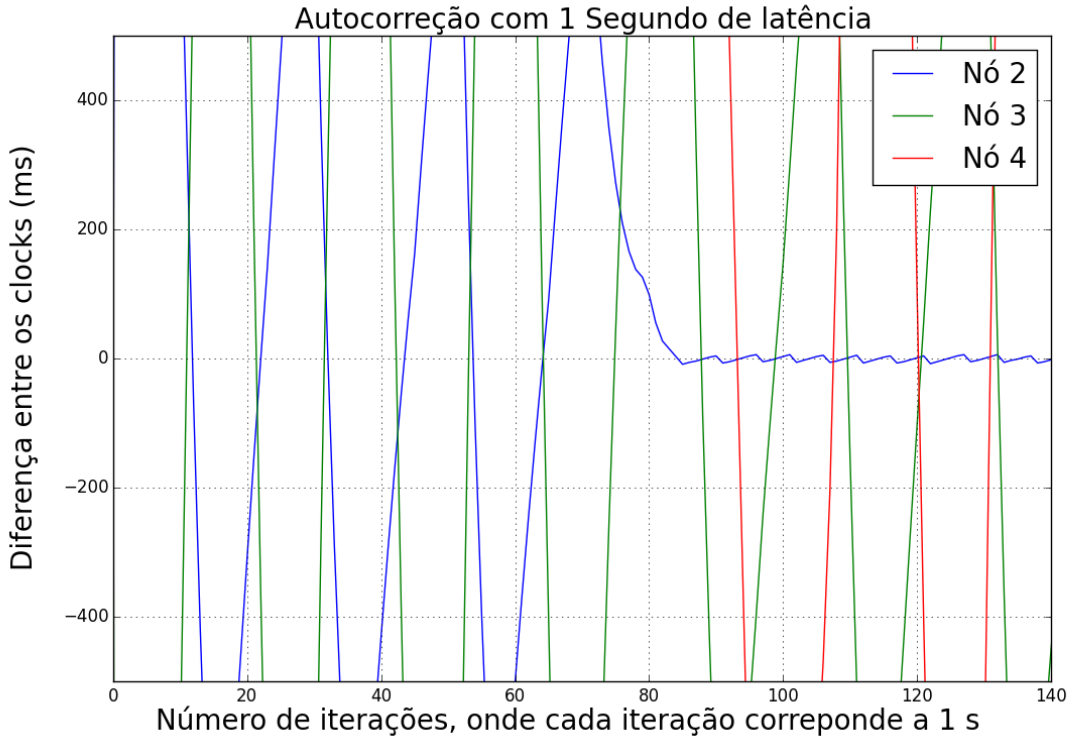

Figura 38 Resultado do teste em cascata com Correção adaptativa e 1 Segundo de latência

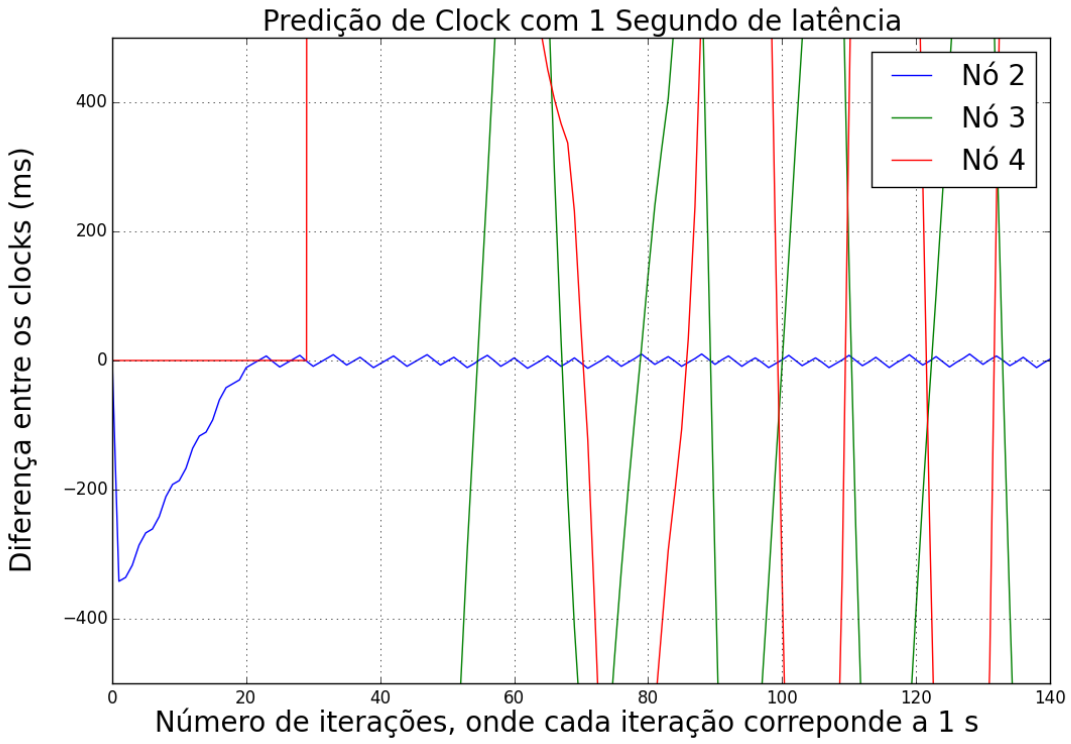

Figura 39 Resultado do teste em cascata com Predição de relógio e 1 Segundo de latência

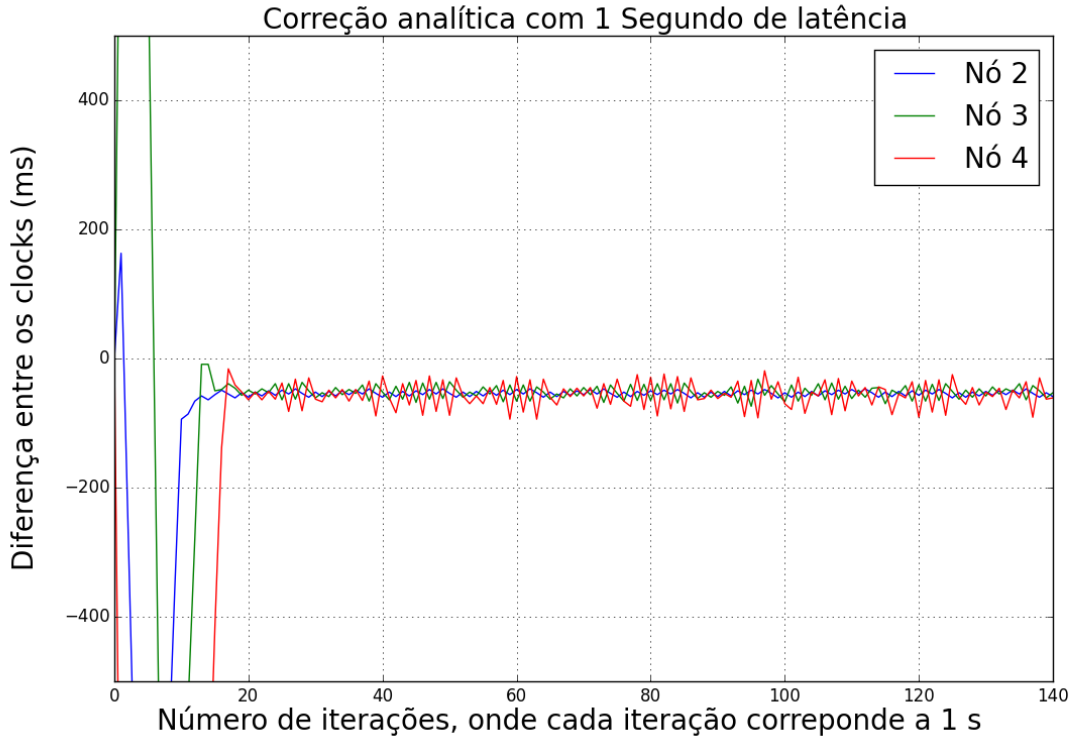

Figura 40 Resultado do teste em cascata com Correção analítica e 1 Segundo de latência

6.2.3.2.2 Dez segundos de latência

Este conjunto de testes foi executado com uma latência de 10 segundos, utilizando as mesmas configurações dos testes anteriores. Logo, a cada dez segundos, ocorre uma transmissão da mensagem de sincronismo, em que cada dispositivo recebeu o registro temporal do nó referência – nó servidor ou nó anterior. Em posse desta informação, os dispostos ajustam os seus relógio. Para facilitar a análise, nesta subseção, os gráficos apresentados terão a mesma escala, com os limites superiores e inferiores de cinco segundos.

A Figura 41 representa os dados coletados com o algoritmo de correção adaptativa, nela é possível observar que o nodo dois apresenta um sincronismo muito próximo a zero. O erro médio acumulado entre os saltos é de aproximadamente um segundo e meio, apresentando um erro acumulado, de aproximadamente três segundos, para o nó mais distante em relação ao servidor de tempo. Desta forma, é possível assumir que a rede, como um todo, não atinge o sincronismo. A Figura 42 referre-se aos dados coletados com o algoritmo de predição de relógio, este teste apresentou uma piora no erro de sincronismo em reação ao caso anterior, mesmo todos os dispositivos conseguido ativar o modo de predição de relógio neste intervalo de latência, este fato deve-se, as perdas da mensagem de sincronismo, o que ocasionaram em uma predição equivocada. A Figura 43 representa os dados coletados pelo algoritmo de correção analítica. O teste apresentou um erro acumulado entre os saltos muito próximo de zero, configurando uma defasagem média acumulada inferior a 100 milissegundos. Isto posto, este teste apresentou um desempenho quarenta vezes melhor que o anterior – sendo o melhor algoritmo.

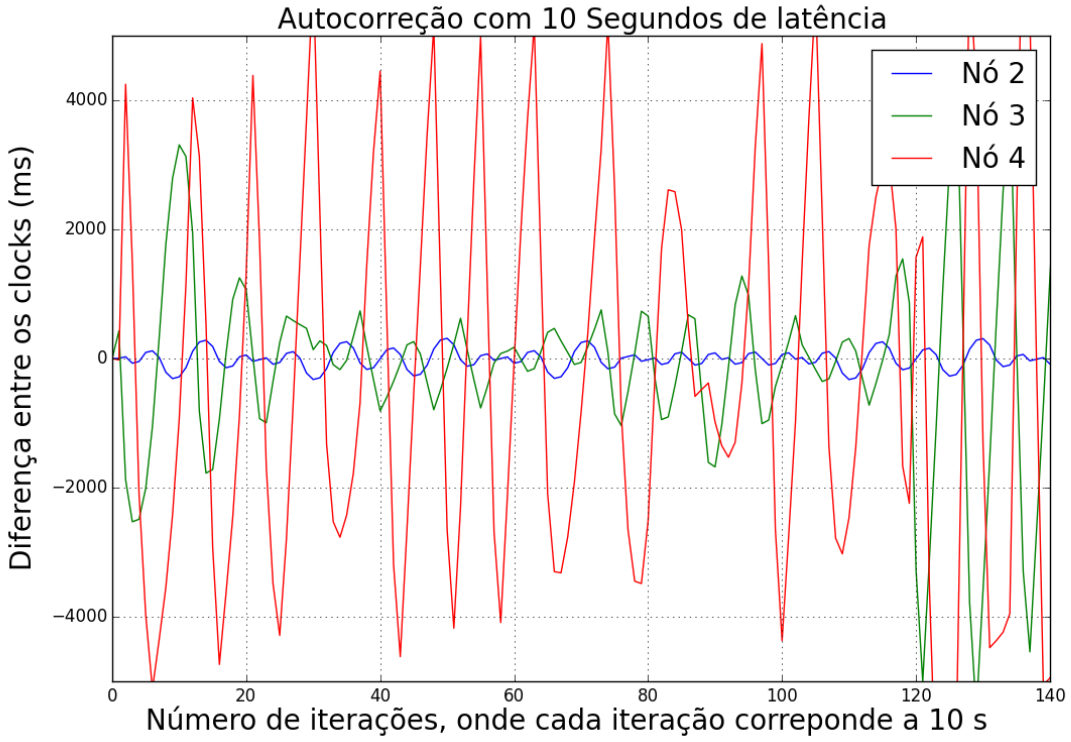

Figura 41 Resultado do teste em cascata com Correção adaptativa e 10 Segundos de latência

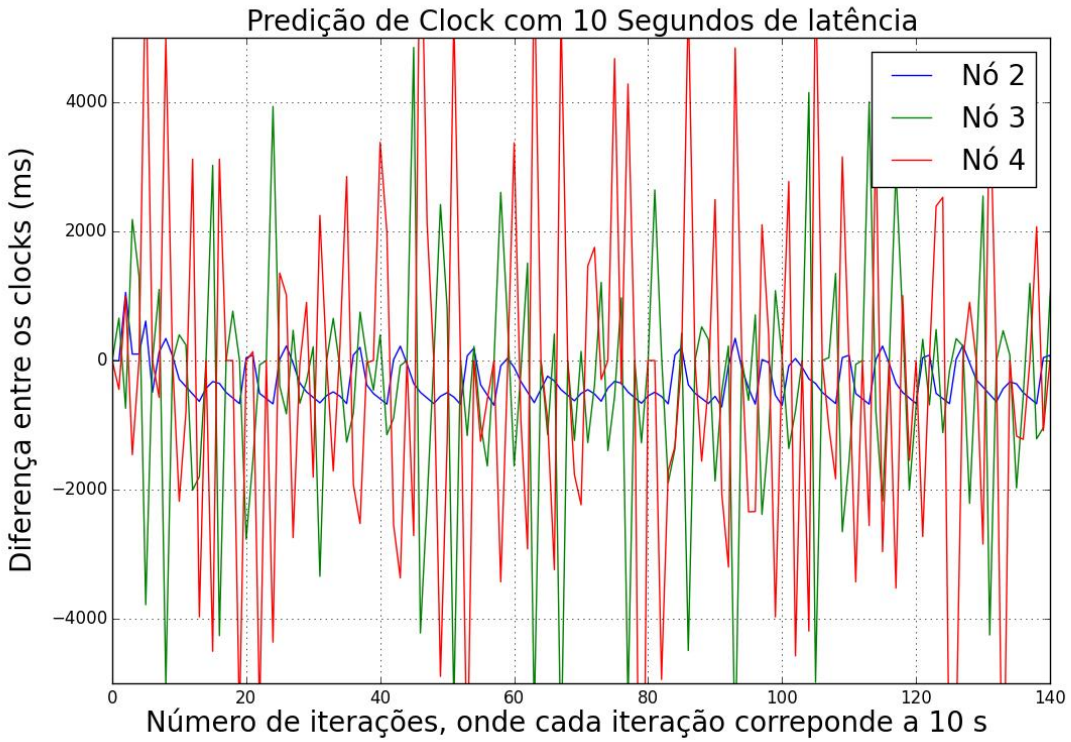

Figura 42 Resultado do teste em cascata com Predição de relógio e 10 Segundos de latência

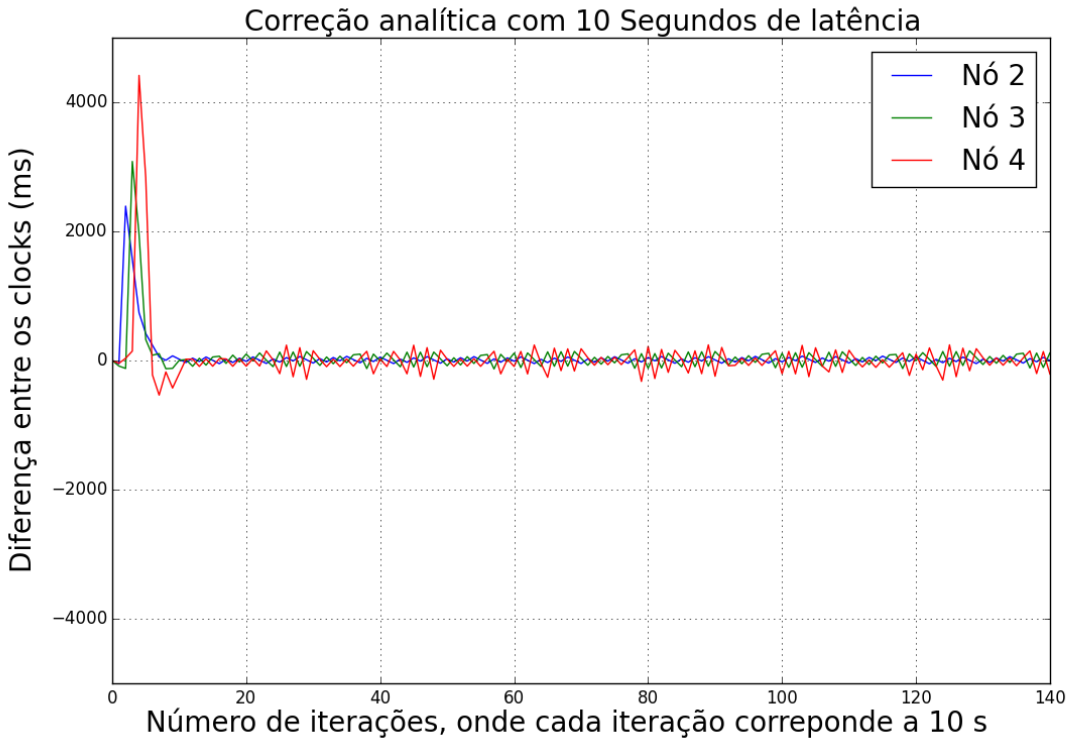

Figura 43 Resultado do teste em cascata com Correção analítica e 10 Segundos de latência

6.2.3.2.3 Trinta segundos de latência

Como já mencionado, para aumentar a eficiência energética é preciso reduzir o número de transmissões. Almejando aplicações que transmitam informação esporadicamente, este teste foi desenvolvido utilizando as mesmas configurações dos testes anteriores – contudo o intervalo de latência foi alterado para 30 segundos, visando determinar qual melhor se adapta a esta nova característica. Para facilitar a análise, todos os gráficos desta subseção serão da mesma escala, com os limites superiores e inferiores de quinze segundos.

A Figura 44 descreve o teste da correção adaptativa, em que o nó dois apresenta uma alta defasagem com um valor médio de nove segundos – como no caso Cliente-Servidor. Este teste, ainda, apresenta um erro médio acumulado – de mais de quinze segundos – do nó quatro. As Figura 45 e Figura 46, apresentam os dados dos testes dos algoritmos de predição de relógio e correção analítica, respectivamente. Estes testes mostraram que o nó mais próximo apresenta alta precisão temporal, assim como no teste Cliente-Servidor. No teste da predição de relógio, o nó 3 apresenta uma defasagem média inferior a dez segundos. Na correção analítica, o nó 3 mostra uma defasagem inferior a cem milissegundos. Quando analisada a defasagem do nó 4, em relação ao servidor, a correção analítica apresenta uma defasagem média pouco maior que duzentos milissegundos, neste teste, é perceptível alguns picos de defasagem. Isto se deve a perdas de mensagem, mas mesmo assim o sistema converge para o sincronismo. Já a predição de relógio indica uma defasagem média maior que sete segundos. Consequentemente, é possível afirmar que a predição de relógio não está sincronizado. Naturalmente, a correção analítica apresenta o melhor resultado nesta análise.

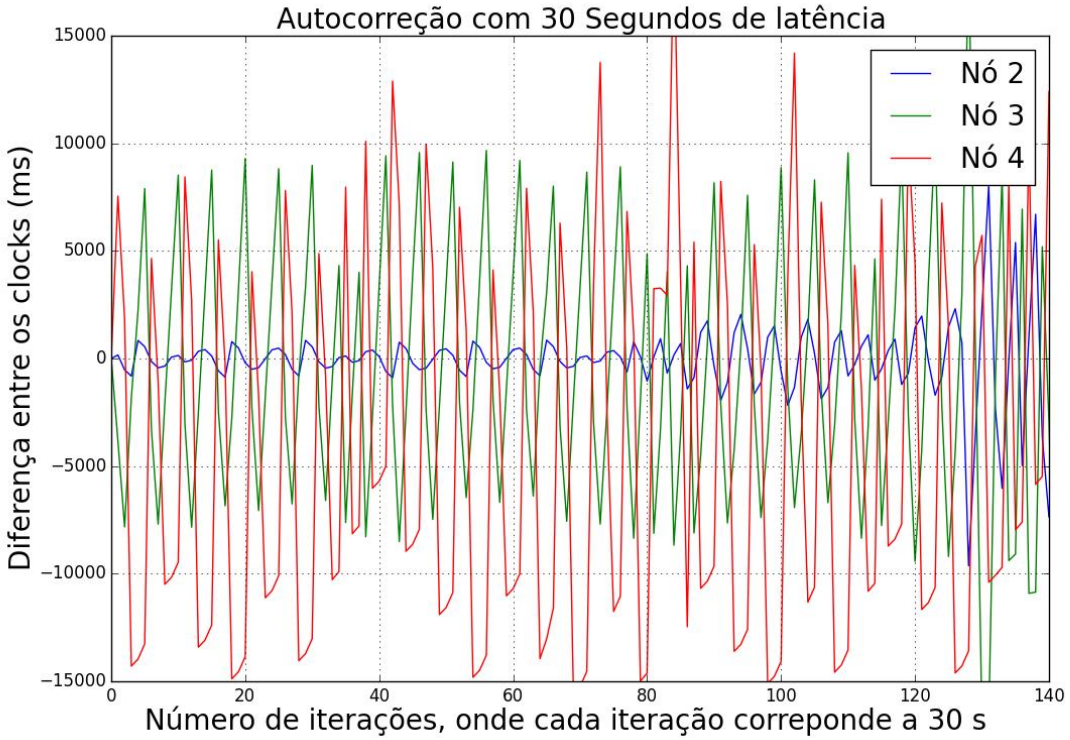

Figura 44 Resultado do teste em cascata com Correção adaptativa e 30 Segundos de latência.

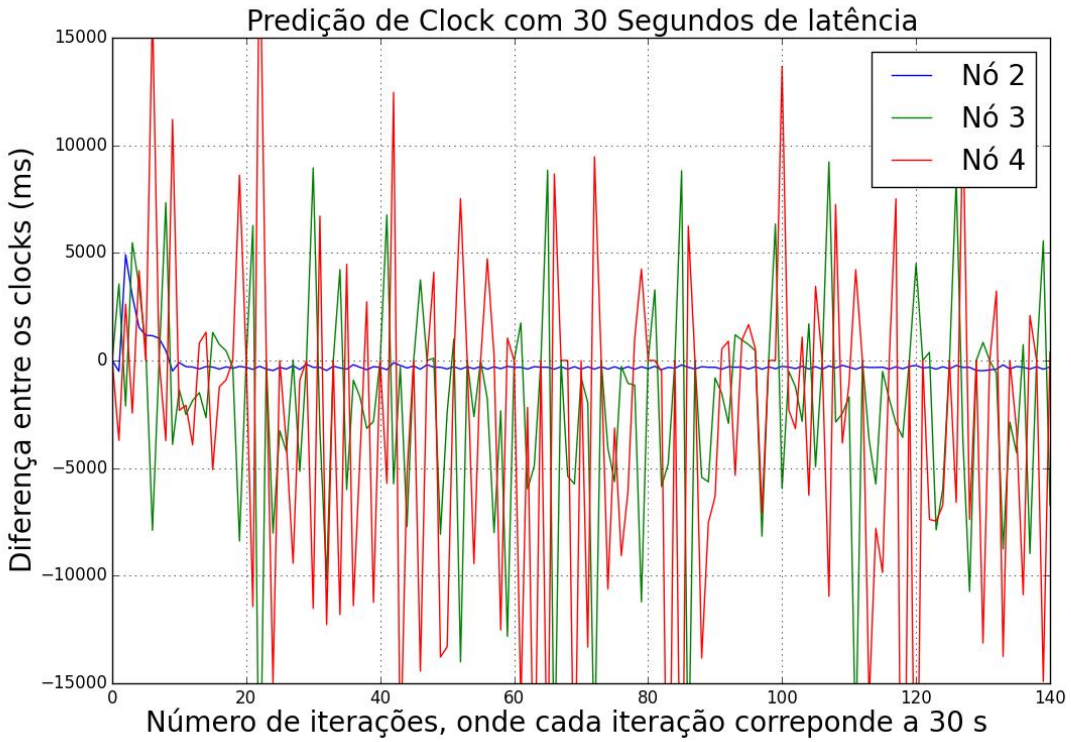

Figura 45 Resultado do teste em cascata com Predição de relógio e 30 Segundos de latência.

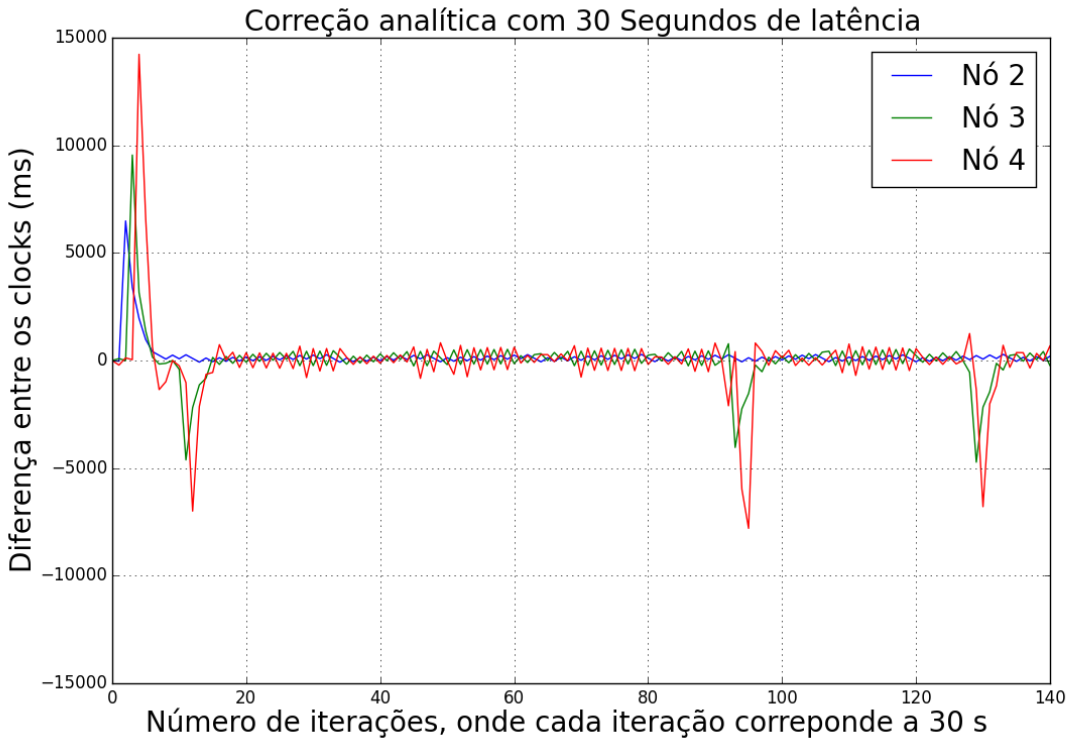

Figura 46 Resultado do teste em cascata com Correção analítica e 30 Segundos de latência.

6.2.3.2.4 Sessenta segundos de latência

Com o objetivo de aumentar ainda mais a vida útil dos dispositivos, ao mesmo tempo estabelecendo ainda menos transmissões da mensagem de sincronismo, foram desenvolvidos estes testes com um intervalo de latência de 60 segundos – utilizando as mesmas configurações dos testes anteriores. Foi estabelecido que, nesta subseção, todos os gráficos serão da mesma escala – com os limites inferiores e superiores de trinta segundos.

A Figura 47 contempla o teste com algoritmo de autocorreção, neste teste é possível notar que a rede apresenta uma defasagem temporal acumulada de mais de 30 segundos. Isto

posto, é possível afirmar que este sistema não está sincronizado. A Figura 48apresenta o teste com o algoritmo de predição de relógio, o qual demonstra um resultado muito superior ao anterior. Nele, o erro médio acumulado é um pouco maior que 20 segundos, no nó 4, Da mesma forma que no teste anterior, este sistema não está sincronizado. A Figura 49, referente ao algoritmo de correção analítica, aponta um desempenho muito superior ao anterior, pois apresenta uma defasagem média acumulada – no ultimo nó – de menos de um segundos. Da mesmo forma que no teste de trinta segundos, neste teste ocorrem perdas, mas o sistema alcança o sincronismo. Por conseguinte, conclui-se que o algoritmo que apresentou o melhor resultado foi a correção analítica.

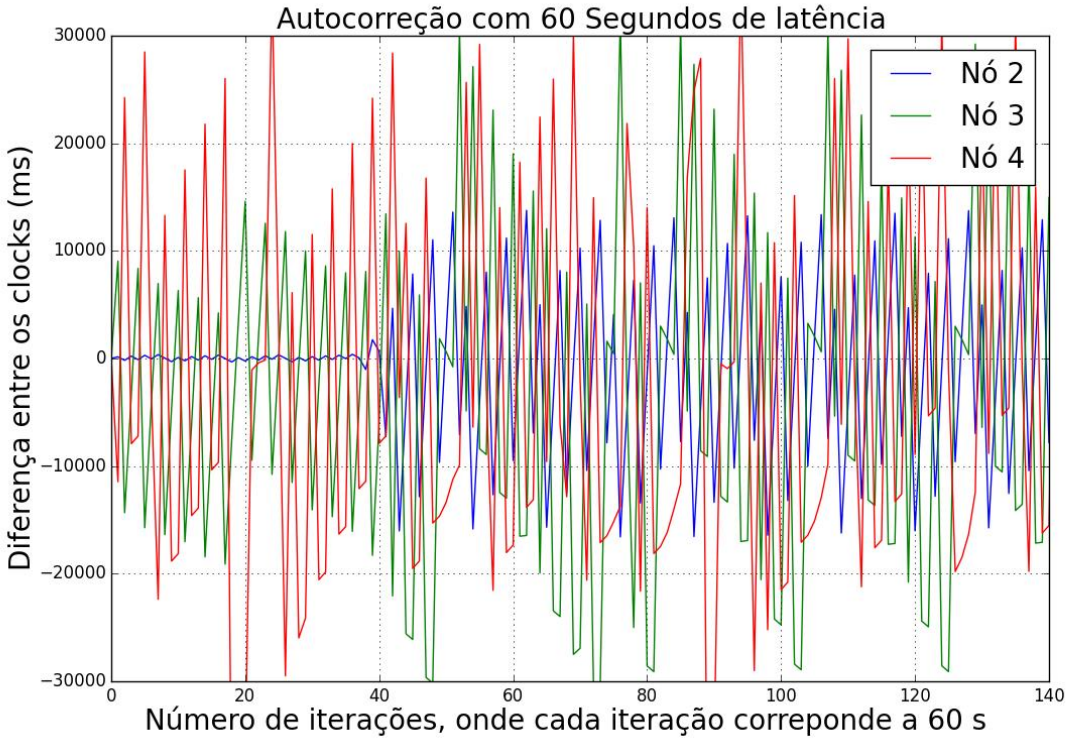

Figura 47 Resultado do teste em cascata com Correção adaptativa e 60 Segundos de latência

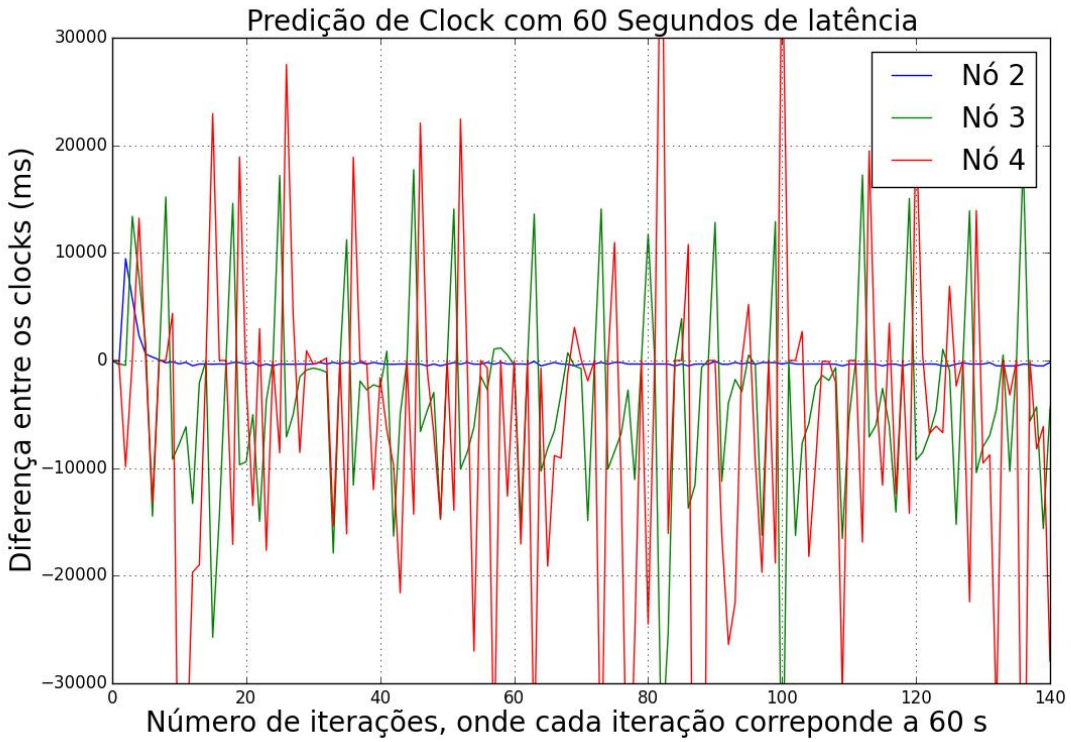

Figura 48 Resultado do teste em cascata com Predição de relógio e 60 Segundos de latência

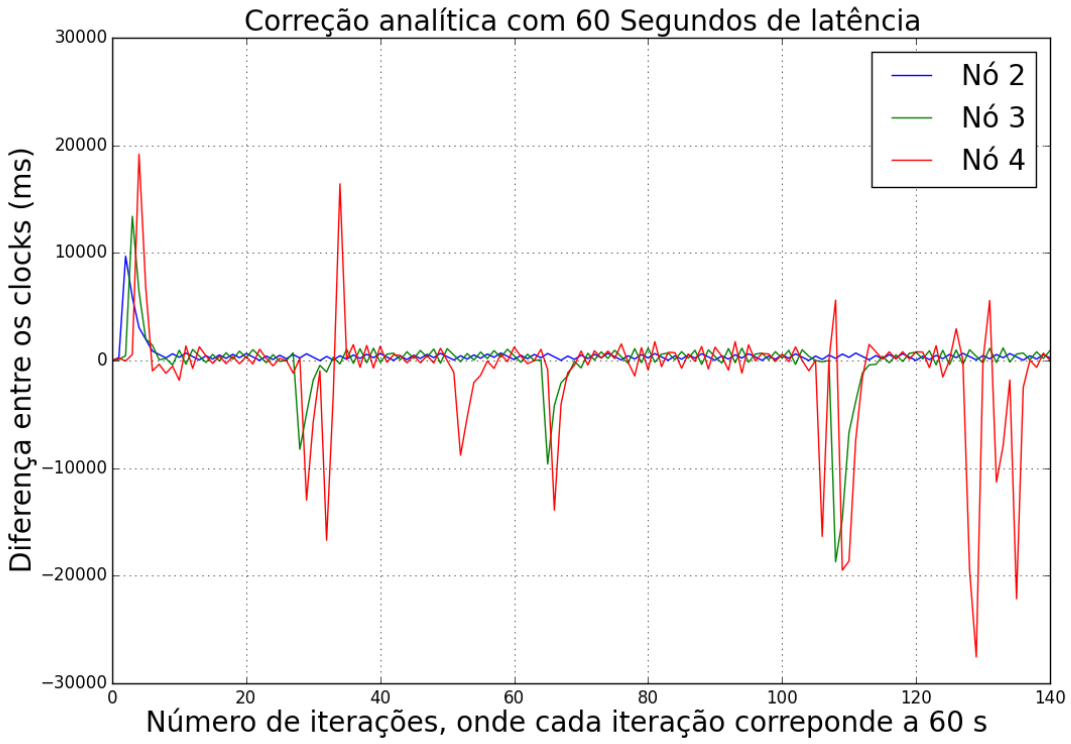

Figura 49 Resultado do teste em cascata com Correção analítica e 60 Segundos de latência

Ponderando todos os testes na propagação em cascata, o algoritmo de sincronismo que melhor se comportou foi o de correção analítica, uma vez que se apresentou como a melhor opção em todos os quatros casos aqui apresentados. A predição de relógio se destacou como o segundo melhor algoritmo, nos testes, mas teve um resultado muito inferior, se comparado ao melhor algoritmo. A correção adaptativa produziu o pior resultado entre todos os algoritmos, pois na maioria dos testes ele não alcançou o sincronismo. Contudo, a precisão temporal não é o único parâmetro a ser considerado, o consumo energético também deve ser levado em conta, o qual será melhor explorado na próxima seção.

6.3 CONSUMO ENERGÉTICO

Como um dos principais motivos para se estabelecer o sincronismo em uma rede de sensores sem fio é a economia energética, foram propostos testes que analisaram este fator. Para identificar a eficiência energética, associada aos três métodos propostos (Correção adaptativa, Previsão de relógio e Correção Analítica), foi elaborado um estudo em que cada método funciona durante dez horas, com as opções de 1, 10, 30 e 60 segundos de intervalo de latência. A Tabela 5 apresenta os resultados obtidos para todos os intervalos testados no cenário de propagação do relógio em cadeia, em que, além do nó esperar pelas próximas transmissões do nó referência, ele retransmite as mensagens de sincronismo. A tabela expõe os valores totais acumulados, no intervalo de teste, na unidade de mAh. Como a menor unidade apresentada no medidor externo é de mAh, podemos considerar que ocorram erros de precisão desta magnitude.

Nestes testes, é possível observar que a Predição de relógio teve um consumo energético maior que a correção adaptativa e a correção analítica. Estes dois algoritmos, obtiveram valores aproximados e se intercalaram como a melhor opção de economia energética. Logo, tem-se duas opções que podem ser determinadas como as melhores. Porém, não é possível eleger uma única opção como a melhor, nestes testes.

Combinando os resultados do consumo energético com a sincronização de relógio, é possível afirmar que a Correção Analítica é a abordagem mais adequada, pois apresenta os melhores resultados em termos de sincronização e consumo de energia. Os outros dois

métodos apresentam um *trade-off* entre as duas métricas, enquanto a correção adaptativa tem menor consumo de energia em relação a predição de relógio, quem apresenta os melhores resultados de sincronização é a predição de relógio.

Tabela 5 Consumo de energia

	Intervalos de latência			
	1	10	30	60
Método de sincronismo	Consumo de energia no intervalo de 10 horas (mAh)			
Correção adaptativa	341	338	331	322
Predição de relógio	457	363	354	349
Correção analítica	344	336	329	319

7 CONCLUSÃO E TRABALHOS FUTUROS

Neste trabalho, foram propostas três abordagens – baseadas no protocolo IEEE 802.15.4 e obedecendo as características específicas de RSSF – para estabelecer, em tempo real, o sincronismo temporal em redes de sensores sem fio de baixa potência, identificando suas semelhanças e destacando suas diferenças. Estas abordagens baseiam-se em três diferentes propostas, com uso de métrica simples, predição e projeção matemática.

A correção adaptativa determina a diferença entre os valores do relógio de referência e o relógio do nó de sincronização. Esta é uma métrica simples, interpretada por um conjunto de condicionais que determinam o quanto será a frequência do seu cristal. A Predição de relógio se vale de todos os valores anteriores do relógio de referência para estimar o próximo valor do relógio de referência – esta estimativa é dada por uma série quadrática infinita. Com o valor estimado, a predição utiliza o mesmo algoritmo da correção adaptativa para ajustar a frequência de seu relógio. A correção analítica emprega um conjunto de informações que são transmitidas pelo nó referência que combina estas informações – valor do relógio atual, latência das transmissões e frequência do relógio– para estimar o valor do relógio na próxima transmissão da mensagem de sincronismo. Isto posto, este algoritmo ajusta a frequência do relógio do nó de sincronização para a menor defasagem possível a este valor estimado. Além dos algoritmos apresentados, foi estudado e desenvolvido um método baseado no trabalho de Cristian (CRISTIAN, 1989) para determinar, de forma mais precisa possível, o atraso entre os dispositivos para alcançar um melhor sincronismo da rede. Desta forma, foi realizado um conjunto de testes, os quais demonstraram que a Correção Analítica obteve o melhor

desempenho. Dado que, ela alcançou um bom resultado em termos de compensação entre a sincronização necessária; de uma sobrecarga aceitável na rede; e ao atingir os melhores resultados em termos de consumo de energia.

A correção analítica se adapta para atingir o sincronismo de forma ágil e, na sequência, busca atingir a maior precisão. Este sistema adaptativo mantém um baixo consumo de energia, enquanto mantém uma alta sincronização entre os nós. Os experimentos realizados para avaliar a eficácia da abordagem proposta, produziram resultados positivos. Conforme apresentado no capítulo 7, o algoritmo de Correção Analítica obteve um desempenho excelente em relação aos outros métodos propostos, tendo o erro de um segundo para um intervalo de latência de 60 segundos. Já o teste de sincronização em cascata obteve um erro acumulado de menos de 200 milissegundos no quarto nó (nó 3). Em outras palavras, o maior atraso da rede foi inferior a 0,2 segundo, em um intervalo de latência de 60 segundos. Analisando os resultados do consumo de energia, as abordagens de Correção adaptativa e Correção Analítica têm resultados muito semelhantes, com uma ligeira vantagem do último algoritmo em relação ao primeiro. Outrossim, quando é combinado aos dados de consumo de energia, com a precisão de sincronização, a correção analítica é definitivamente a melhor abordagem.

Para trabalhos futuros, uma das propostas seria utilizar o melhor algoritmo aqui apresentado – correção analítica – e identificar qual seria o melhor período de latência (intervalo entre as transmissões) que não haja perdas de mensagem de sincronismo em virtude de correlações temporais e aplicar a correção analítica em um ambiente real, no qual ocorram

perdas de dispositivos, fato que modifica a topologia da rede de forma dinâmica. Outra proposta futura é a utilização deste algoritmo em uma rede heterogénea, que possua relógios de diferentes escalas. Desta forma, será preciso identificar qual é a frequência do cristal de oscilação do nó referência e implementar métodos para converter estes valores. Estas características são importantes para tornar este algoritmo – correção analítica – adequado a IoT, que habitualmente necessitam de diversos aparelhos de diferentes marcas.

Atualmente, com o progresso das aplicações das redes de sensores sem fio, o estudo nesta área torna-se crucial, dado que várias aplicações necessitam estabelecer o consenso temporal. Este consenso favorece a confluência de dados, a economia energética, a depuração de sistemas e a identificação de falhas – elementos que facilitam o processo evolutivo destas aplicações.

REFERÊNCIAS

AKYILDIZ, I. F. et al. Wireless sensor networks: a survey. **Computer networks**, New York, v. 38, n. 4, p. 393-422, Mar. 2002.

AL-KARAKI, J. N.; KAMAL, A. E. Routing techniques in wireless sensor networks: a survey. **IEEE Wireless Communications**, New York, v. 4, n, p. 6-28, Dez. 2004.

ATZORI, L.; IERA,A.; MORABITO, G. The internet of things: a survey. **Computer Networks,** Reggio Calabria, v. 15, n. 54, p.2787-2805, Apr. 2010. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1389128610001568>. Acesso em: 07 abr. 2017.

CONTROL DATA SYSTEMS. **Versa Node 210** 2.4 GHz Wireless Radio: Product Specification. 2016. Disponível em: <http://www.cds.ro/Documents/Datasheets/VN210.pdf>. Acesso em: 10 jul. 2017.
COULOURIS, G. F.; DOLLIMORE, J.; KINDBERG, T. **Distributed systems**: concepts and design. 5th. ed. Boston: Pearson Education, 2012. p. 82.

CRISTIAN, F. A probabilistic approach to distributed clock synchronization. In: INTERNATIONAL CONFERENCE ON DISTRIBUTED COMPUTING SYSTEMS,, 9th, 1989, Newport Beach. **Proceedings**… New York: IEEE, 1989. p. 288-296.

DIEDRICHS, A. L. et al. Low-power wireless sensor network for frost monitoring in agriculture research. In: IEEE BIENNIAL CONGRESS OF ARGENTINA (ARGENCON), 2014, Bariloche. **Proceedings** … New York: IEEE, 2014. p. 525-530.
ELSON, J.; GIROD, L.; ESTRIN, D. Fine-grained network time synchronization using reference broadcasts. **ACM SIGOPS Operating Systems Review**, New York, v. 36, p. 147-163, Dec. 2002.

FRESCALE. **MC1322x**: Advanced ZigBee – Compliant SoC Platform for the 2.4 GHz IEEE 802.15.4 Standard Reference Manual, Rev. 1.6. jan. 2012. Disponível em: <http://www.nxp.com/docs/en/reference-manual/MC1322xRM.pdf> Acesso em: 11 jul. 2017.
GANERIWAL, S.; KUMAR, R.; SRIVASTAVA, M. B. Timing-sync protocol for sensor networks. In: INTERNATIONAL CONFERENCE ON EMBEDDED NETWORKED SENSOR SYSTEMS, 1.,, 2003, Los Angeles. **Proceedings**… New York: ACM, 2003. p. 138-149.

INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS. **802.15.4-2006**: standard for information technology: local and metropolitan area networks: specific requirements: part 15.4: wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Low Rate Wireless Personal Area Networks (WPANs). Disponível em: <https://standards.ieee.org/findstds/standard/802.15.4-2006.html>. Acesso em: 03 maio 2017.

KOPETZ, H. Accuracy of time measurement in distributed real time systems. MARS report. Wien: Technische Univ. Wien, Inst. für Technische Informatik, 1985.

KOPETZ, H.; OCHSENREITER, W. Clock synchronization in distributed real-time systems. **IEEE Transactions on Computers**, New York, n. 8, v. 100, p. 933-940, 1987.

KSHEMKALYANI, A. D.; SINGHAL, M. **Distributed computing**: principles, algorithms, and systems. Cambridge: Cambridge University Press, 2008. p. 19-20.

KUROSE, J. F.; ROSS, K. W. **Computer networking**: a top-down approach. 6th ed. New Jersey: Pearson Prentice Hall, 2013. p. 36-40.

LENZEN, C.; SOMMER, P.; WATTENHOFER, R. PulseSync: an efficient and scalable clock synchronization protocol. **IEEE/ACM Transactions on Networking (TON)**, Piscataway, v. 23, n. 3, p. 717-727, Jun. 2015.

LI, Z. et al. E²dts: an energy efficiency distributed time synchronization algorithm for underwater acoustic mobile sensor networks. **Ad Hoc Networks**, Barcelona, v. 11, n. 4, p. 1372-1380, Nov. 2013.
MARÓTI, M. et al. The flooding time synchronization protocol. In: INTERNATIONAL CONFERENCE ON EMBEDDED NETWORKED SENSOR SYSTEMS, 2., 2004, Baltimore. **Proceedings**… New York: ACM, 2004, p. 39-49.

MILLS, D. L. Adaptive hybrid clock discipline algorithm for the network time protocol. **IEEE/ACM Transactions on Networking (TON)**, Piscataway, v. 6, n. 5, p. 505-514, Aug. 1998.

MILLS, D. L. **RFC 1305**: Network Time Protocol (Version 3) specification, implementation and analysis. [S. l.], 1992. Disponível em: <https://tools.ietf.org/html/rfc1305>. Acesso em: 01 ago. 2017.

MILLS, D. L. **IEN-173**: time synchronization in DCNET hosts. DARPA Internet Project Report. [S.l.]: COMSAT Laboratories, 1981.

MIRABELLA, O. et al. Dynamic continuous clock synchronization for IEEE 802.15. 4 based sensor networks. In: ANNUAL CONFERENCE OF IEEE (IECON), 34., 2008, Orlando. **Proceedings**… New York: IEEE, 2008. p. 2438-2444.

MOCK, M. et al. Continuous clock synchronization in wireless real-time applications. In: SYMPOSIUM ON RELIABLE DISTRIBUTED SYSTEMS, 19., 2000, Nürnberg. **Proceedings**… New York: IEEE, 2000. p. 125-132.

MULLETT, G J. **Wireless telecommunications systems and networks**. Springfield: Cengage Learning, 2005. p. 30.

NIXON, M.; ROUND ROCK, T. X. **A comparison of WirelessHART and ISA100.11a**. Round Rock: Emerson Process Management, Missouri, 2012.

PALCHAUDHURI, S.; SAHA, A. K.; JOHNSIN, D. B. Adaptive clock synchronization in sensor networks. In: INFORMATION PROCESSING IN SENSOR NETWORKS, 2004, Berkeley. **Proceedings**… New York: IEEE, 2004. p. 340-348.

PANFILO, G.; TAVELLA, P. Atomic clock prediction based on stochastic differential equations. **Metrologia**, Bristol, v. 45, n. 6, p. S108, Dec 2008.

PING, S. Delay measurement time synchronization for wireless sensor networks. **Intel Research Berkeley Lab**, Cambridge, v. 6, p. 1-10, Jun. 2003.

REN, F.; LIN, C.; LIU, F. Self-correcting time synchronization using reference broadcast. **IEEE Wireless Sensor Network,** Las Vegas, v. 15, n. 4, p. 1-7, Aug. 2008.

RHEE, Ill-K. et al. Clock synchronization in wireless sensor networks: an overview. **Sensors**, Basel, v. 9, n. 1, p. 56-85, Jan. 2009.

ROUMELIOTIS, S. I.; BEKEY, G. A. An extended kalman filter for frequent local and infrequent global sensor data fusion. In: SENSOR FUSION AND DECENTRALIZED CONTROL IN AUTONOMOUS ROBOTIC SYSTEMS, 1997, Pittsburgh. **Proceedings**… Bellingham: SPIE, 1997. p. 14-15.

SCHENATO, L.; GAMBA, G. A distributed consensus protocol for clock synchronization in wireless sensor network. In: CONFERENCE ON DECISION AND CONTROL, 46., 2007, New Orleans. **Proceedings**… New York: IEEE, 2007. p. 2289-2294.

SKRZYPCZAK, L.; GRIMALDI, D; RAK, R. Basic characteristics of ZigBee and SimpliciTI modules to use in measurement systems. In: WORLD CONGRESS FUNDAMENTAL AND APPLIED METROLOGY, 19., 2009, Lisbon. **Proceedings**… Lisbon: IMEKO, 2009. p. 1456-1460.

SOHRABI, K. et al. Protocols for self-organization of a wireless sensor network. **IEEE Personal Communications**, Los Alamitos, v. 7, n. 5, p. 16-27, Oct. 2000.

SOLIS, R.; BORKAR, V. S.; KUMAR, P. R. A new distributed time synchronization protocol for multihop wireless networks. In: CONFERENCE ON DECISION AND CONTROL, 45., 2006. Sandiego. **Proceedings**… New York: IEEE, 2006. p. 2734-2739.

STOJMENOVIC, I. (Ed.). **Handbook of sensor networks:** algorithms and architectures. New York: Wiley-Interscience, 2005. p.32-33.

SU, W.; AKYILDIZ, I. F. Time-diffusion synchronization protocol for wireless sensor networks. **IEEE/ACM Transactions on Networking (TON)**, Piscataway, v. 13, n. 2, p. 384-397, Apr. 2005.
SUNDARARAMAN, B.; BUY, U.; KSHEMKALYANI, A. D. Clock synchronization for wireless sensor networks: a survey. **Ad hoc networks**, Amsterdam, v. 3, n. 3, p. 281-323, May. 2005.

TANENBAUM, A. S. **Redes de computadores.** 4. ed. Rio de Janeiro: Campus, 2003. p.45.

TEXAS INSTRUMENTS. MSP430 LFXT1 oscillator accuracy: MSP430 applications. Dallas, 2004. Disponível em:<http://www.ti.com/lit/an/slaa225/slaa225.pdf>. Acesso em: 10 jul. 2017.

VERISSIMO, P.; RODRIGUES, L.; CASIMIRO, A. Cesiumspray: a precise and accurate global time service for large-scale systems. **Real-Time Systems**, New York, v. 12, n. 3, p. 243-294, May 1997.

YILDIRIM, K. S.; KANTARCI, A. Time synchronization based on slow-flooding in wireless sensor networks. **IEEE Transactions on Parallel and Distributed Systems**, New York, v. 25, n. 1, p. 244-253, Feb. 2013.