UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

Peterson Clayton de Oliveira

SINCRONIZAÇÃO DE TEMPO PARA REDES DE SENSORES SEM FIO UTILIZANDO O PROTOCOLO IEEE 1588

Florianópolis

2012

Peterson Clayton de Oliveira

SINCRONIZAÇÃO DE TEMPO PARA REDES DE SENSORES SEM FIO UTILIZANDO O PROTOCOLO IEEE 1588

Trabalho de Conclusão de Curso submetida ao Curso de Ciências da Computação para a obtenção do Grau de Bacharel em Ciências da Computação.
Orientador: Prof. Dr. Antonio Augusto Frohlich
Coorientador: Msc. Alexandre Massayuki Okazaki

Florianópolis

2012

Catalogação na fonte elaborada pela biblioteca da
Universidade Federal de Santa Catarina

A ficha catalográfica é confeccionada pela Biblioteca Central.

Tamanho: 7cm x 12 cm

Fonte: Times New Roman 9,5

Maiores informações em:
http://www.bu.ufsc.br/design/Catalogacao.html

Peterson Clayton de Oliveira

SINCRONIZAÇÃO DE TEMPO PARA REDES DE SENSORES SEM FIO UTILIZANDO O PROTOCOLO IEEE 1588

Esta Trabalho de Conclusão de Curso foi julgada aprovada para a obtenção do Título de "Bacharel em Ciências da Computação", e aprovada em sua forma final pelo Curso de Ciências da Computação.

Florianópolis, 23 de Novembro 2012.

Prof. Chefe, Dr. Eng. Bruno Vitório Mazzola
Coordenador do Curso

Banca Examinadora:

Prof. Dr. Antonio Augusto Frohlich
Orientador

Msc. Alexandre Massayuki Okazaki
Coorientador

Bsc. Rodrigo Steiner

Msc. Arliones Stevert Hoeller Jr

Tous les jours, a tous points de vue, on va de mieux en mieux.

Émile Coué

RESUMO

Sincronização de tempo é um campo de estudos pertinente para cumprir os requisitos operacionais de diversas aplicações distribuídas. Neste trabalho, implementamos o protocolo IEEE 1588 - Precision Time Protocol - para o sistema operacional EPOS, com o objetivo de sincronizar o tempo do relógio em redes de sensores sem fio. Para efeitos de testes, escolhemos utilizar uma topologia que consiste em um mestre e um escravo. Foram realizados dois experimentos, sob dois contextos diferentes. O primeiro no contexto das redes cabeadas, onde foram analisadas duas configurações: a primeira com dois relógios previamente sincronizados e uma segunda configuração com o relógio do escravo atrasado em algumas horas para analisarmos o comportamento do protocolo após algumas iterações. O Segundo experimento foi realizado em um contexto considerando as redes de sensores sem fio. Duas configurações também foram analisadas, a primeira, em uma topologia mestre-escravo, dois EPOSMotes foram sincronizados. Na segunda houve o acréscimo de um novo nodo na rede, nem mestre nem escravo, com o objetivo de se manter sincronizado junto a eles, de modo que a rede ficasse sincronizada. Com os resultados obtidos podemos assegurar a viabilidade da implementação do protocolo para o sistema operacional EPOS e consequentemente para sistemas embarcados, inclusive no contexto das redes de sensores sem fio.

Palavras-chave: RSSF. Sincronização de Tempo. PTP. RBS. TDSP.

ABSTRACT

Time Synchronization is a pertinent field of study to fulfill the requirements in many distributed applications. In this work, we implemented the IEEE 1588 protocol - Precision Time Protocol - for the EPOS operating system to deliver the clock time among the network.
For test purposes, we choose a topology consisted of one master clock and another slave clock. We made two configurations: one in which we have two previously synchronized clocks, to evaluate their behavior among time after some synchronizations. On the other configuration, we delayed the slave clock in some hours and analysed the behavior of the protocol after some iterations. With the results obtained we can ensure the viability of the protocol implementation to the EPOS operating system and consequently to embedded systems. With this implementation in hand, we started to change the implementation to fulfill the Wireless Sensor Networks (WSN) context requirements. We changed the testbed to make use of physical motes, embedded devices available and developed at LISHA called EPOSMotes and repeated the experiments with one master and one slave node. During the experiments, we observed that with the addition of more nodes the overhead would increase exponentially. Then, we propose a new model to synchronize all the nodes from a WSN, which we describe in this work. With the results, we observed an offset in the sub-millisecond range. That fulfills almost all the applications requirements for the WSNs, applications in the filed of location estimation, physical sensing measurements, replay attack countermeasures and others.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

SUMÁRIO

1 INTRODUÇÃO

Sincronização de tempo para sistemas distribuídos é um campo de estudos desafiante. Levando-se em conta a topologia de como os dispositivos estão distribuídos, para algumas aplicações específicas, de localização e sensoriamento de eventos físicos, é necessário que todos os nodos da rede estejam sincronizados para se obter precisão nas medições realizadas. Aplicações em redes de sensores sem fio, similarmente a outros sistemas distribuídos, requerem um serviço de sincronização de tempo escalável. Um exemplo disso são protocolos de localização que podem tirar vantagem do tempo sincronizado dos sensores para garantir o tempo exato em que determinado objeto estava no local observado. As redes de sensores constituem uma tecnologia emergente que permite a coleta de informação em diversos ambientes, tais como a monitoração de ambientes, automação industrial e aplicações militares.

Uma rede de sensores sem fio é constituída por um Coordenador, responsável por realizar a comunicação entre os sensores e as aplicações gerenciadoras de tarefas, e por um grupo de sensores que utilizam *links* sem fio para executar tarefas de sensoriamento distribuído (WANNER,). Estes nodos tipicamente possuem um micro-processador embarcado, um restrito poder computacional e pouca capacidade energética. Uma característica presente nas redes de sensores é a cooperação existente entre os nodos, essa cooperação se faz necessária para que o sensoriamento requerido seja feito de forma correta e precisa. Como os sensores possuem pouca capacidade computacional, já que possuem um processador embarcado, ao invés de enviarem o dado bruto coletado, eles são capazes de utilizar seu poder de processamento para enviar apenas os dados necessários e parcialmente processados.

Assim, manter os relógios dos nodos sincronizados, com uma certa tolerância, nos leva ao objetivo principal de estudo deste trabalho. Dado a necessidade de manter-se um tempo de relógio sincronizado, com um nível de tolerância determinado durante o tempo de vida da rede. Alguns fatores podem influenciar na sincronização e devem ser levados em consideração, como: temperatura, ruído de fase, ruído de frequência, atraso assimétrico e falhas no relógio (RATON; NEW; WASHINGTON,). Considerando esses fatores na elaboração de uma estratégia de sincronização, podemos chegar a três classes de técnicas de sincronização (RATON; NEW; WASHINGTON,). A primeira técnica se baseia em servidores de tempo fixo para sincronizar a rede, nela os dispositivos são sincronizados à servidores de tempo precisos. Na se-

gunda técnica, o tempo é traduzido *hop-by-hop*, sendo essencialmente um serviço de tradução do tempo pela rede. E por fim a terceira técnica auto-organização a rede para realizar a sincronização não dependendo de servidores de tempo especializados. Ele automaticamente organiza e determina os nodos mestre como sendo os servidores temporários de tempo.

De acordo com estes três tipos, alguns protocolos foram propostos, para diferentes tipos de ambientes. O mais conhecido e utilizado destes é o *Network Time Protocol* (NTP). Entre as técnicas elencadas, o NTP se enquadra no primeiro tipo. Considerando os protocolos disponíveis, foi escolhido para ser estudado com um maior grau de profundidade o *Precision Time Protocol* (PTP), ou também conhecido pelo acrônimo IEEE 1588, que se enquadra no terceiro tipo. Além de ser um padrão adotado pela IEEE, obtem-se alta precisão e possui flexibilidade de configuração e adequação a diferentes topologias de rede.

Existe uma implementação conhecida deste protocolo para o Sistema Operacional Linux, chamada PTPd. No nosso caso utilizaremos o EPOS, que é um sistema operacional orientado a aplicações para sistemas embarcados (FRöHLICH, 2001). Com o uso de sistema operacional voltado para dispositivos embarcados, entre eles sensores, o sincronismo de dados dos sensores torna-se fundamental para diversas aplicações de sensoriamento no processo de aquisição de dados e controle.

O objetivo do estudo do protocolo PTP se dá na obtenção de sincronismo com precisão na faixa de sub-micro segundo para sistemas embarcados. Neste trabalho iremos implementar o protocolo PTP para o Sistema Operacional EPOS com o intuito de estudar os aspectos e limitações do protocolo. Em seguida temos a possibilidade de iniciar a integração do protocolo ao EPOSMote (LISHA Team, 2011), uma plataforma aberta para redes de sensores sem fio, a fim de obter experimentos de sincronismo utilizando nodos sensores reais.

1.1 OBJETIVOS

Como dito anteriormente o objetivo deste trabalho é realizar a implementação do protocolo IEEE 1588, um protocolo de sincronização de tempo, que tem como características: alta precisão de sincronização, requisitos computacionais mínimos e pouca ocupação da largura de banda da rede, permitindo que ele seja implementado em dispositivos embarcados. A ideia básica é sincronizar os relógios escravos com um relógio mestre, garantindo que os tempos em todos os nodos da rede

sejam os mesmos.

A contribuição esperada está em possibilitar uma gama maior de aplicações para redes de sensores sem fio, que utilizam sistemas embarcados como nodos da rede. Podemos citar uma implementação concreta do protocolo no cenário rede de sensores sem fio no trabalho (RÍO et al., 2012), em que o protocolo é utilizado para distribuir o tempo em uma rede de sensores marinha. Há essa necessidade porque os sinais de GPS não estão disponíveis pela atenuação da água e existem equipamentos, como sismógrafos, que exige uma precisão na faixa de sub-milissegundo.

2 FUNDAMENTAÇÃO TEÓRICA

Esta seção possui a fundamentação teórica consultada para realizar este trabalho. Iniciamos com uma breve explanação sobre um dos protocolos de sincronização de tempo mais utilizados, o NTP. Após o estudo do NTP, seguimos com os protocolos de sincronização propostos para as redes de sensores sem fio. Demonstrando sua abordagem, benefícios e deficiências. Realizando assim, uma visão crítica e captando o estado da arte no campo de pesquisa em sincronização de tempo para redes de sensores sem fio.

2.1 O PROTOCOLO NTP

O Network Time Protocol (NTP) é um dos protocolos de sincronização de tempo mais utilizado. Este protocolo basicamente cria uma rede de hospedeiros na internet que realizam a sincronização do tempo. Realizar esta sincronização é um desafio importante e difícil em sistemas distribuídos, pois, além de obter a correta marcação do tempo entre os nodos, cada relógio possui uma taxa de escorregamento diferente, ou seja, a cada período de tempo o relógio sai de sincronização, o que representa um problema. Os serviços de sincronização NTP estão amplamente disponíveis na internet, a sub-rede pública do NTP em 2011 incluía milhares de servidores em diversos países e em todos os continentes do globo terrestre, no espaço e até mesmo no oceano (MINAR, 1999). Estes servidores dão suporte a uma população de aproximadamente 25 milhões de computadores na Internet.

Esta sub-rede opera em níveis hierárquicos, como pode ser visto na Figura 1, onde cada nível possui uma numeração chamada de *stratum*. Os servidores pertencentes ao *Stratum* 1 estão diretamente conectados e sincronizados ao *Stratum* 0, que é normalmente composto por um Sistema de Posicionamento Global (GPS) ou um relógio atômico. Normalmente o *Stratum* 0 não existe como camada, ele é apenas a referência inicial do tempo a ser sincronizado entre os outros elementos da rede. Os servidores pertencentes ao *Stratum* 2, nível subsequente ao *Stratum* 1, estão sincronizados aos servidores do *Stratum* 1 e assim sucessivamente para as outras camadas subsequentes até a camada 16, onde estão localizados os servidores inoperantes. Normalmente, clientes NTP e servidores com um número relativamente pequeno de clientes não se sincronizam com servidores públicos primários. Existem cente-

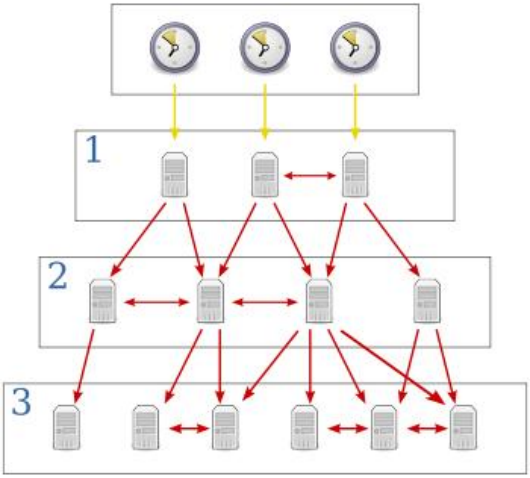


Figura 1 – Hierarquia no protocolo NTP

nas de servidores secundários públicos operando em camadas superiores e são então preferidos ante os primários (MINAR, 1999).

2.1.1 Stratum 0: Relógio Atômico e GPS

A necessidade de medir o tempo com maior precisão levou à criação de relógios baseados em certas propriedades dos átomos. Os assim chamados relógios atômicos permitiram até a criação de um novo padrão para o segundo, constituindo hoje a grandeza física mais bem definida(CEPA, 2012).

Em relação ao GPS, ele é um sistema de navegação baseado no espaço operado pela força aérea americana, e consiste basicamente em um conjunto de 24 satélites. Ele foi desenvolvido inicialmente para uso militar mas devido a sua grande utilidade está sendo utilizado por diversas pessoas ao redor do mundo. Ele provê aos seus usuários informações precisas sobre seu posicionamento, velocidade e horário em qualquer local do mundo.

Considerando o fato do NTP ter sido proposto para redes de larga-escala, assim como é o caso da Internet, os nodos são sincronizados externamente a um tempo global de referência, GPS ou relógio atômico. Este tempo é inserido na rede em diversos lugares, através de

um conjunto de servidores de tempo.

Estes servidores são sincronizados fora da banda, os parentes de cada servidor devem ser especificados nos arquivos de configuração próprios. Estes nodos frequentemente trocam mensagens de sincronização com seus respectivos parentes e utilizam a informação obtida para ajustar seu relógio através de uma incrementação regular do mesmo.

Um dos limitantes para sua aplicação é o fato do NTP manter um sistema de sincronização de relógio através da adição regular de pequenos incrementos no contador do relógio do sistema, através deste comportamento há o impedimento em se trocar o processador para um modo ocioso para economia de energia. Logo o fato dos servidores NTP precisarem estar preparados para receber requisições de sincronização a qualquer momento, impede seu uso para as redes de sensores sem fio, dado suas limitações técnicas.

2.1.2 NTP Arquitetura, Protocolo e Algoritmos

O NTP é construído sobre o *Internet Protocol* (IP), e no *User Datagram Protocol*(UDP) (MILLS, 1994), que provê um mecanismo de transporte sem a necessidade de estabelecer conexões entre os envolvidos. Este foi desenvolvido a partir do protocolo de tempo e da mensagem de *timestamp* ICMP, desenvolvido especialmente para manter a exatidão e redundância. No modelo proposto para o NTP não houve nenhum esforço especial designado para realizar a descoberta de *peers*, configuração ou aquisição, apesar de haver implementações que contemplam estes aspectos. A integridade dos dados é proporcionada pelos *checksum's* do IP e do UDP, nenhum mecanismo de detecção de dados duplicados ou retransmissão são oferecidos. Dado o fato de apenas um formato de mensagem NTP, apresentado na Figura 2, ser utilizado, este protocolo é facilmente implementado e utilizado nos diversos sistemas operacionais existentes. Os servidores pertencentes a rede trocam *timestamps* que são utilizados para determinar os atrasos bidirecionais individuais, os *offsets* dos relógios e estimativas de erros (MILLS, 1994).

2.2 O PROTOCOLO RBS

O Protocolo RBS foi proposto para realizar a sincronização *receiver-to-receiver* nas RSSF (SARI et al., 2008). De modo, geral um nodo

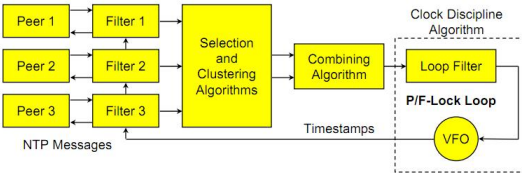


Figura 2 – Mensagens do Protocolo NTP

transmissor envia N mensagens de sincronização e os nodos que receberem as mesmas inserem *timestamps* nestas mensagens. Assim, um *reference broadcast* não contém um *timestamp* explícito, ao invés, esse *timestamp* registrado no recebimento da mensagem de *broadcast* é que será utilizado como um ponto de referência para a comparação entre os tempos de relógio da rede.

Após efetuar o registro do *timestamp* na mensagem, o receptor envia uma mensagem de resposta ao transmissor para que ele calcule o *clock offset*, diferença de tempo entre dois relógios, e o *clock skew*, escorregamento do tempo do relógio, entre os diferentes pares de nodos pertencentes à rede. Através destes passos, dois dos principais problemas em se tratando de sincronização de tempo são eliminados, sendo eles, variação no tempo de envio e no tempo de acesso. A diferença entre os tempos de propagação é insignificante perto da incerteza do tempo de recebimento, que acaba por se tornar a única fonte de erro existente (SARI et al., 2008).

De acordo com Elson em (ELSON; GIROD; ESTRIN, 2002), o principal inimigo para se alcançar precisão na sincronização do tempo é o não-determinismo. Estimativas de latência são confundidos por eventos aleatórios que levam à atrasos assimétricos de ida e volta na entrega de mensagens. Esta assimetria contribui diretamente para a ocorrência de erros na sincronização.

Elson sugere uma decomposição na fonte destes erros com o intuito de obter uma melhor compreensão sobre o que ocasiona cada um deles e como estes erros podem ser tratados. Em (KOPETZ; SCHWABL,), Kopetz e Schwabl caracterizam esta fonte como possuindo 4 componentes distintos.

1. Tempo de Envio.
2. Tempo de Acesso ao Meio.
3. Tempo de Propagação.

4. Tempo de Recebimento.

O tempo de acesso ao meio envolve o atraso referente à espera para ter acesso ao meio de transmissão e é específico do protocolo MAC, conforme consta no apêndice B, em uso. O tempo de propagação envolve o tempo necessário para a mensagem trafegar de um transmissor até um receptor, sendo que este tempo pode ser pequeno caso o receptor e transmissor compartilhem do mesmo meio físico para comunicação, no caso das redes locais.

O tempo de recebimento é o tempo de processamento necessário para que a interface de rede do receptor receba a mensagem do canal de comunicação e notifique o hospedeiro de seu recebimento. Por fim temos o tempo de envio que é caracterizado pelo tempo despendido no transmissor para construir a mensagem. Isto inclui atrasos ocasionados pelo Sistema Operacional, como por exemplo: trocas de contexto e *overhead* de *system call* ocasionadas pela aplicação de sincronização. Tempo de envio também leva em conta o tempo necessário para transferir a mensagem até a interface de rede do transmissor.

Os protocolos de sincronização de tempo se diferenciam principalmente pela maneira como tratam as estimativas e correções das fontes de erros supracitadas. Assim, a propriedade fundamental do RBS é que uma mensagem de *broadcast* seja utilizada apenas para sincronizar um conjunto de receptores uns com os outros. Desta forma, remove-se o tempo de envio e o tempo de acesso ao meio do caminho crítico da sincronização. (MILLS, 1994) atribui a maioria dos erros de fase observados na sincronização de um cliente NTP com um receptor GPS na mesma Rede Local ao *jitter* e às colisões.

Para neutralizar estes efeitos, um *broadcast* de uma mensagem RBS é sempre utilizado como um tempo de referência relativo, nunca para comunicar um valor de tempo absoluto. É exatamente esta propriedade que elimina os erros introduzidos pelo tempo de envio e pelo tempo de acesso ao meio. Cada receptor deve sincronizar com um pacote de referência. A mensagem não precisa conter um *timestamp* gerado pelo transmissor, ou nem mesmo um *timestamp* do momento em que foi enviada a mensagem. Também não precisa de um tipo específico de mensagem para realizar a sincronização dos tempos de relógio. Pois qualquer extensão de um *broadcast* pode ser utilizado para recuperar informações sobre o tempo.

Para efetuar uma sincronização utilizando o protocolo RBS precisamos propagar um sinal de *broadcast* para dois receptores, permitindo que eles estimem os *offsets* relativos à fase. Após esta propagação necessitamos que:

- Cada receptor registre o tempo em que o pacote de referência foi recebido, de acordo com seu relógio local.

- Os receptores troquem suas observações de tempo, enviando mensagens aos seus vizinhos.

De posse destas informações os receptores possuem informações suficientes para formar uma escala local de tempo. Um exemplo de uso desta informação esta no fato de, se um nodo da rede está na posição (0,0), considerando um sistema de coordenadas cartesiano com origem em 0x e 0y, e detecta um alvo no tempo t-4, e j está na posição (0,10) e detecta o mesmo alvo no tempo t-5, podemos concluir que o alvo está se movendo no sentido norte com uma velocidade de 10 unidades por segundo.

2.3 O PROTOCOLO FTSP

Concebido para aplicações que necessitam de uma rigorosa precisão de tempo em uma plataforma sem fio com limitações de recursos(MARóTI; SIMON, 2004). O protocolo baseia-se na técnica de *flooding* onde cada nodo age como um receptor e transmissor de mensagens, e cada mensagem recebida é retransmitida para todos os vizinhos do nodo, exceto pelo nó do qual a mensagem foi originada, assim, o protocolo realiza um *flooding* periódico de mensagens de sincronização e atualização dinâmica implícita da topologia da rede.

O FTSP sincroniza o tempo do transmissor, possivelmente, com múltiplos receptores utilizando uma única mensagem com *timestamp* do transmissor e do receptor. O registro do *timestamp* na mensagem quando realizado na subcamada MAC pode garantir uma melhor precisão e eliminar alguns erros provenientes de outras técnicas (MARóTI; SIMON, 2004).

Tipicamente as RSSF podem operar em áreas de alcance maior do que o raio de *broadcast* de um único nodo. Levando-se em conta esse fato o FTSP oferece suporte à sincronização *multi-hop*. Basicamente a raiz da rede, um único nodo, dinamicamente eleito mantém o tempo global e todos os outros nós devem sincronizar seus relógios em relação ao relógio do nodo raiz. Os nós formam uma estrutura *ad-hoc* para transferir o tempo global a partir da raiz para todos os nós pertencentes a rede.

A mensagem enviada via *broadcast* contém o tempo de relógio do transmissor que é caracterizado como sendo o tempo global estimado no

momento em que é feita a transmissão de um dado byte. Os receptores assim obtêm o tempo local correspondente, a partir do seu relógio local, no momento em que é recebida a mensagem. O *offset* do receptor é estimado pela diferença entre o tempo global e o tempo local de um ponto da sincronização.

Ao contrário do protocolo RBS o *timestamp* do transmissor deve ser embarcado na mensagem atual a ser transmitida. Portanto, o *timestamp* do lado do transmissor deverá ser realizado antes dos bytes que contêm o *timestamp* serem transmitidos.

O funcionamento do protocolo inicia-se com o envio de uma mensagem *broadcast*, que contêm inicialmente o preâmbulo, seguido por bytes de *SYNC*, a seguir um descritor de mensagem seguido dos dados da mensagem e encerrando com os bytes de CRC, apresentados na Figura 3.

Durante a transmissão dos bytes de preâmbulo o radio do receptor sincroniza o mesmo com a frequência do sinal de entrada. A partir dos bytes de *Sync* o receptor pode calcular o bit *offset* necessário para montar a mensagem com o alinhamento de bytes correto. O descritor da mensagem contêm o alvo, o tamanho dos dados e alguns campos contendo o identificador da camada de aplicação, entre outros. Finalmente os bytes de CRC são utilizados para verificar se a mensagem não foi corrompida.

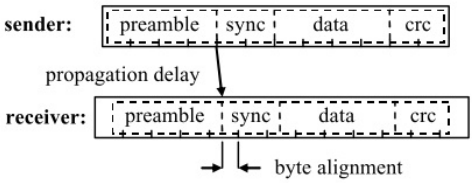


Figura 3 – Pacotes de dados transmitidos pelo canal

2.4 O PROTOCOLO TDP

O TDP é um protocolo proposto para realizar a sincronização de tempo do relógio dos dispositivos pertencentes à rede. Ele permite à RSSF obter um tempo de equilíbrio e mantêm os dispositivos da rede sincronizados com um desvio de tempo pequeno em relação ao tempo de equilíbrio estabelecido (SU; AKYILDIZ,).

O TDP é utilizado para manter o tempo de uma rede sincronizado com uma certa tolerância. Este nível de tolerância pode ser ajustado baseado na aplicação que fará uso desta sincronização. O protocolo foi desenvolvido baseando-se em dois critérios : ser auto-configurável e sensível à requisitos energéticos.

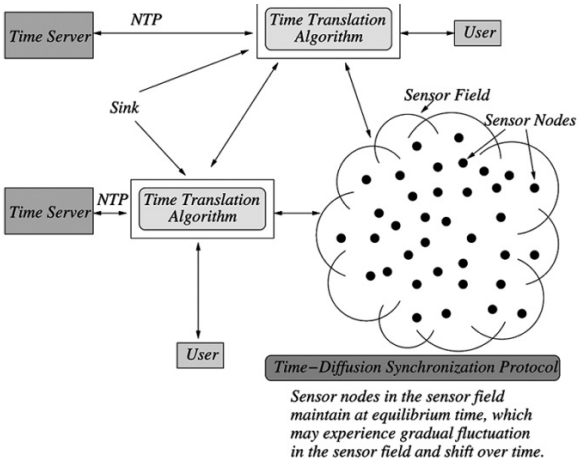


Figura 4 – Arquitetura do protocolo TDP e interação com o mundo externo

Apesar do TDP poder ser utilizado com servidores de alta precisão, é importante destacar a natureza autônoma do TDP. Devido às condições adversas em que são implantados os sensores, os *sinks* podem vir a não ser utilizados como servidores de tempo e em concordância com a natureza autônoma do protocolo, permite que as RSSF's obtenham um tempo de equilíbrio independente dos padrões utilizados, como o UTC ou o TAI .

Dado que o tempo entre os sensores da rede pode variar e também há a possibilidade de ocorrência de um *time drift* em relação à algum padrão adotado internacionalmente é necessário que seja feita uma tradução do tempo dos relógios da rede para algum formato padrão como o UTC. Essa tradução fica a cargo dos *sinks* através do Algoritmo de Tradução do Tempo (SU; AKYILDIZ,).

O funcionamento básico do protocolo inicia com a difusão de mensagens sobre o tempo de relógio para os nodos vizinhos, onde a informação sobre o tempo são posteriormente difundidas por *diffused*

líderes eleitos para *n hops* a partir do nodo mestre. Assim, uma topologia em árvore é criada quando o nodo mestre difunde as mensagens com informações sobre o seu tempo de relógio. No final do processo, a informação sobre o tempo de relógio é utilizada para ajustar os relógios locais.

2.5 PRECISION TIME PROTOCOL

O IEEE 1588 Precision Time Protocol provê um método padrão para sincronizar dispositivos em uma rede com precisão de sub-microssegundos. IEEE 1588 provê não apenas compatibilidade entre sistemas heterogêneos, mas também alta precisão em relação a sincronização dos tempos de relógio.

O processo de sincronização é contínuo, pois diversos fatores podem levar dois relógios idênticos a apresentarem valores diferentes e consequentemente perderem sincronização. Causas como diferenças na temperatura e a taxa de frequência podem afetar a qualidade da sincronização.

IEEE 1588 provê uma sincronização tolerante à falhas para diferentes relógios na mesma rede. Vantagens deste protocolo são: pouco consumo de banda e pouco consumo de processamento (COMMITTEE; TC; SOCIETY, 2008). IEEE 1588 consegue tal feito através do uso do *Precision Time Protocol* (PTP). O PTP é um protocolo baseado na troca de mensagens que pode ser implementado em redes de comutação de pacotes, incluindo, mas não limitado às redes *Ethernet*.

Este protocolo sincroniza todos os relógios pertencentes à uma rede através do ajuste dos relógios em relação ao relógio com melhor qualidade. Para determinar o relógio com melhor qualidade utiliza-se o algoritmo *Best Master Clock* (BMC), que determina qual relógio possui a melhor qualidade de marcação de tempo dentre os pertencentes à rede.

Caso o mestre seja removido da rede ou seja decidido pelo algoritmo de BMC para não continuar sendo o relógio mestre da rede, o algoritmo então redefine o novo mestre e ajusta todos os outros relógios tendo como referência o novo mestre.

Comunicação *Multicast* Bidirecional é utilizada pelos relógios escravos para se sincronizarem ao relógio mestre. Um pacote de *Sync* é enviado do mestre aos escravos de tempos em tempos. Caso seja optado por uma sincronização em um passo, então o *timestamp* é registrado e enviado na própria mensagem de Sync.

Utilizando dois passos um pacote *followUp* também deve ser enviado pelo mestre. Este pacote conterá um *timestamp*, que é o tempo exato que a mensagem de *Sync* deixou o mestre, propiciando assim uma maior precisão, pois o *timestamp* poderá ser registrado em camadas mais próximas ao meio físico, que sofrem menos interferências não determinísticas e permitindo uma maior precisão.

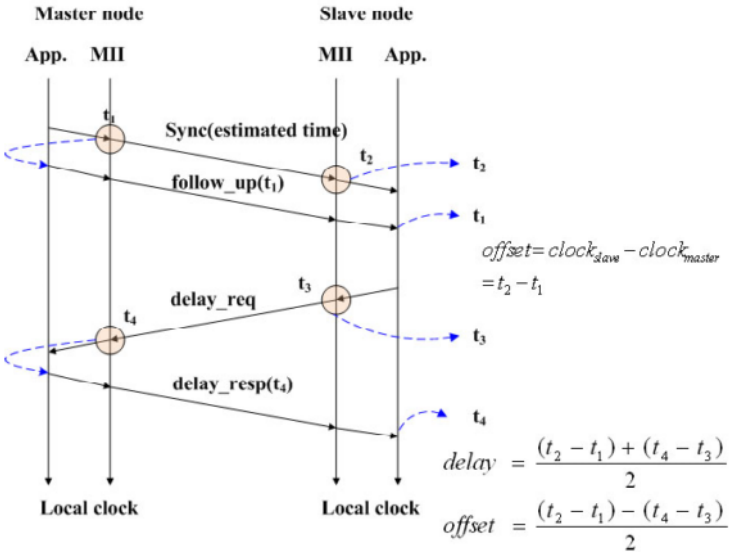


Figura 5 – Troca de mensagens efetuadas segundo o protocolo PTP

O mestre e o escravo trocam pacotes de sincronização em ambos os sentidos e registram o valor do *timestamp* assim que recebem os pacotes, como pode ser observado na Figura 5. A diferença dos tempos de relógio da saída e chegada dos pacotes de *Sync* podem ser calculados através da combinação do *offset* do relógio do escravo em relação ao mestre e do atraso de propagação pela rede. Utilizando o *offset* medido neste ponto, o relógio pode se reajustar e o *offset* entre o mestre e o escravo pode ser reduzido apenas ao atraso de propagação do pacote na rede.

O IEEE 1588 assume que o atraso de propagação na rede é simétrico. E é devido a este fato que o escravo pode determinar e se reajustar para o atraso de propagação. Para isso, o escravo cria e envia ao mestre um pacote de *Delay Request*, registrando o *timestamp*

na saída do pacote para a rede. O mestre, ao receber o pacote, registra o tempo de recebimento e envia um *Delay Response* para o nodo escravo. O atraso de propagação na rede é então determinado através da diferença desses dois últimos tempos registrados pelo escravo e pelo mestre. O processo de envio e recebimento para a sincronização permite aos relógios dos escravos medir, de forma precisa, o *offset* entre o seu próprio relógio em relação ao relógio do mestre.

2.5.1 Passos Gerais do Protocolo

Após uma visão geral do funcionamento do protocolo, exploraremos melhor seus passos para realizar a sincronização entre os nodos da rede. Os passos seguem a seguinte ordem:

1. O nodo mestre envia uma mensagem de Sync ao nodo escravo e registra o tempo de envio t1.

2. O escravo recebe a mensagem de Sync e registra o tempo de recebimento da mesma t2.

3. O nodo mestre transporta ao nodo escravo o tempo t1 de duas formas:

    - Inserindo o *timestamp* t1 na mensagem de Sync (one-step). Este método requer algum processo de hardware para melhor precisão.
    - Inserindo o *timestamp* t1 em uma mensagem de Follow Up (two-step).

4. O nodo slave envia uma mensagem de Delay Req para o nodo mestre e registra o tempo de envio t3.

5. O nodo mestre recebe a mensagem de Delay Req e registra o tempo de recebimento t4.

6. O nodo mestre transmite ao nodo escravo o *timestamp* t4 inserindo-o em uma mensagem de Delay Resp.

Após realizar estas trocas de mensagens o nodo escravo possuirá 4 *timestamps*, e a partir dos mesmos será possível obter o *offset*, diferença entre o tempo registrado no nodo mestre em comparação com o tempo registrado no nodo escravo, e o atraso da rede, tempo

necessário para pacotes trafegarem pelo link entre dois nodos comunicantes.

O atraso no link pode ser calculado da seguinte maneira :

$$\beta = t_2 - t_1 \tag{2.1}$$

$$\gamma = t_4 - t_3 \tag{2.2}$$

β representa o atraso no sentido Mestre-Escravo, enquanto que o γ representa o atraso no sentido Escravo-Mestre. Em cada caso, a diferença nos tempos se refere aos tempos tomados por dois relógios diferentes. No entanto, caso se assuma, que o atraso em uma direção é o mesmo atraso que no sentido oposto, então as duas equações podem ser combinadas na seguinte equação.

$$\alpha = \frac{(\beta) + (\gamma)}{2} \tag{2.3}$$

Assim sendo, temos representado por α o atraso da rede, e podemos obter o *offset* do relógio do nodo escravo através da seguinte equação:

$$\phi = t_2 - (t_1 + \alpha) \tag{2.4}$$

Ou realizando substituições e otimizações, sendo ϕ equivalente ao *offset* a partir do mestre:

$$\phi = \frac{(\beta) - (\gamma)}{2} \tag{2.5}$$

Caso dois conjuntos de mensagens de sincronização e de *Follow Up* sejam enviadas, então o *drift* entre os dois relógios pode ser descoberto através da comparação da variação de tempo entre duas mensagens de sincronização sucessivas, como prossegue.

$$\rho = \frac{(t_{escravo}) - (t_{mestre})}{t_{mestre}} \tag{2.6}$$

ρ representa o *drift*, ou escorregamento, entre os relógios mestre e escravo, sendo os tempos equivalentes aos tempos marcados nos mesmos.

2.5.2 O algoritmo Best Master Clock

O algoritmo de seleção do melhor relógio para ser o nodo mestre é fundamental para a operação do PTP de modo autônomo. Ele especifica o método pelo qual cada relógio determinará o melhor relógio mestre dentro de seu subdomínio, contendo todos os nodos alcançáveis por suas ondas de rádio. A decisão é baseada no número do *stratum*(qualidade do relógio) do relógio local, a precisão do relógio, a estabilidade do oscilador local.

Caso haja alguma combinação entre duas portas em uma subrede então a decisão final é baseada no *Universally Unique Identifier (UUID)* da porta, este algoritmo foi desenvolvido para que não haja negociação entre os relógios, enquanto garante que configurações entre dois nodos mestres, nenhum mestre ou uma oscilação entre dois mestres ocorra.

O IEEE 1588 apenas provê um protocolo padrão para realizar a troca de mensagens entre dois relógios, o beneficio é que relógios de diferentes fabricantes ainda conseguem realizar sincronização uns com os outros.

2.5.3 Hardware

Algumas aplicações podem utilizar soluções *software-only* do protocolo. São aplicações que não demandam alta precisão em comparação ao custo de hardware especializado necessário, porém há casos em que é necessário atingir um nível maior de precisão.(WEIBEL; BéCHAZ, 2004).

Nesses casos os *timestamps* devem ser registrados mais próximos da camada física. Para sistemas assistidos por hardware específico, o *timestamp* pode ser registrado entre o MAC e a camada física, caso seja possível. Para obter os quadros diretamente do cabo, são necessárias algumas funções como, *clock recovery*, *line decoding*, *descrambling*, entre outras de acordo com (WEIBEL; BéCHAZ, 2004).

2.5.4 Versões do Protocolo

O protocolo IEEE 1588 foi proposto inicialmente em 2002, sendo que seu uso começou a ser percebido em 2003 (NISTGOV, 2012) Porém as discussões em relação à elaboração de um *standard* para tratar da sincronização de relógios iniciaram em 2000 (NISTGOV, 2012). Segundo

dados do National Institute of Standards and Technology (NIST), a primeira reunião do comitê ocorreu em Abril de 2001, sendo que o comitê submeteu a primeira aplicação formal ao IEEE, foi aprovada em 18 de Junho de 2001. Porém, o esboço só foi aprovado como um padrão IEEE em 12 de Setembro de 2002, sendo publicado em Novembro do mesmo ano.

Em 2004, o padrão IEEE1588-2002 foi submetido à revisão para preencher os interesses de novas aplicações que surgiram na área de Telecomunicações. Com isso foi criado o projeto P1588, em Fevereiro de 2005, em um comitê IEEE com o objetivo de estender o padrão atual. O resultado deste comitê foi o mais atual padrão IEEE1588-2008 que ficou disponível a partir de Março de 2008, e contou com os seguintes adicionais:

1. Melhor precisão, chegando a sub-nanosegundos.
2. Sincronização mais rápida.
3. Tamanho das mensagens diminuiu para reduzir o uso da banda.
4. Novas classes de mensagens foram criadas.
5. Introdução do *one-step-mode*.
6. Introduzidos os Relógios Transparentes.
7. Introdução de *profiles*.
8. Novo mapeamento para mecanismos de transporte como, *DeviceNet*, *PROFInet*, *ControlNet* e *IEEE802.3/Ethernet*.
9. Introdução de TLV's.
10. Introduzidos aspectos de segurança.
11. Compatibilidade com versão anterior do protocolo

Assim, a Versão 2 do Protocolo obteve uma melhora consistente em relação à versão anterior do protocolo, salientando a usabilidade e a precisão para redes de alta escala.(MESSAGES; MESSAGES; FORMAT, 2012) Para alcançar esses objetivos, houve uma redução no tamanho da mensagem para poupar um pouco da banda a ser utilizada. Além de Relógios Transparentes para previnir a propagação exponencial de erros em redes que possuem uma topologia em cascata, além de outras melhorias já citadas acima.

2.5.5 Tipos de Mensagens

Abaixo iremos discutir algumas mensagens que fazem parte do protocolo PTP, assim como as modificações que ocorreram nas mensagens com a introdução da versão dois do protocolo. Primeiramente iniciaremos com as mensagens de *Management*, Figura 6, que são utilizadas para consultar e atualizar os dados PTP mantidos pelos relógios. Estas mensagens são utilizadas também para customizar o sistema que utiliza o PTP, para gerar determinados eventos, para inicialização e para administração de falhas.

Management 1588 V2

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
Header								34	0
targetPortIdentity								10	34
startingBoundaryHops								1	44
boundaryHops								1	45
reserved				actionField				1	46
reserved								1	47
managementTLV								M	48

Figura 6 – Mensagem de Management da Versão dois do Protocolo

Seguimos com uma das maiores contribuições para a nova versão de mensagens, a mensagem de *Annouce*, Figura 7, que é utilizada para estabelecer a hierarquia de sincronização (COMMITTEE; TC; SOCIETY, 2008).

Announce 1588 V2

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
Header								34	0
originTimestamp								10	34
currentUTCoffset								2	44
reserved								1	46
grandmasterPriority1								1	47
grandmasterClockQuality								4	48
grandmasterPriority2								1	52
grandmasterIdentity								8	53
stepsRemoved								2	61
timeSource								1	63

Figura 7 – Mensagem de Announce da Versão dois do Protocolo

Anteriormente a essa mensagem possuíamos as informações do mestre sendo enviadas a cada mensagem de *Sync* , Figura 8, o que

tornava essa mensagem grande e aumentava o *overhead* na rede, muitas vezes impossibilitando a implementação do protocolo para redes como a IEEE 802.15.4.

Sync & Delay_req 1588 V1									
Bits								Octets	Offset
7	6	5	4	3	2	1	0		
Header								36	0
reserved								4	36
originTimestamp (seconds)								4	40
originTimestamp (nanoseconds)								4	44
epochNumber								2	48
currentUTCoffset								2	50
grandMasterCommunicationTechnology								2	52
grandMasterClockUuid								6	54
grandmasterPortId								2	60
grandmasterSequenceId								2	62
grandMasterClockStratum								4	64
grandMasterClockIdentifier								4	68
grandMasterClockVariance								4	72
grandMasterPreferred								2	76
grandMasterIsBoundaryClock								2	78
syncInterval								4	80
localClockVariance								4	84
localStepsRemoved								4	88
localClockStratum								4	92
localClockIdentifier								4	96
parentCommunicationTechnology								2	100
parenttUuid								6	102
parentPortField								4	108
estimatedMasterVariance								4	112
estimatedMasterDrift								4	116
utcReasonnable								4	120

Figura 8 – Mensagem de Sync da Versão um do Protocolo

Com a nova versão, as tarefas são melhores divididas e apenas a informação necessária para a sincronização é enviada a cada mensagem de *Sync*, Figura 9.

Sync & Delay_req 1588 V2									
Bits								Octets	Offset
7	6	5	4	3	2	1	0		
Header								34	0
originTimestamp								10	34

Figura 9 – Mensagem de Sync da Versão dois do Protocolo

As mensagens de *Delay Resp* também sofreram redução, ficando apenas com as informações do *Header* e do *Timestamp*, como é apre-

sentado nas Figuras 10 e 11.

Delay_resp 1588 V1									
Bits								Octets	Offset
7	6	5	4	3	2	1	0		
Header								36	0
reserved								4	36
associatedSequenceId								4	40
delayReceiptTimestamp (seconds)								4	44
delayReceiptTimestamp (nanoseconds)								4	48
requestingSourceCommunicationTechnology								2	52
requestingSourceUuid								6	54
requestingSourcePortId								2	60
requestingSourceSequenceId								2	62

Figura 10 – Mensagem de Delay Resp da Versão um do Protocolo

Delay_resp 1588 V2									
Bits								Octets	Offset
7	6	5	4	3	2	1	0		
Header								34	0
receiveTimestamp								10	34
requestingPortIdentity								10	44

Figura 11 – Mensagem de Delay Resp da Versão dois do Protocolo

Por último verificamos que o *Header* de todas as mensagens também foi alterado, houve uma reorganização de seus campos com uma pequena redução de 1 byte no tamanho, como mostram as Figuras 12 e 13.

2.5.6 Tipos de Relógio

Na primeira versão do protocolo os únicos tipos de relógios eram o *Ordinary Clock* e o *Boundary Clock*. Os *Boundary Clocks*, apresentados na Figura 14, foram definidos pelo protocolo para oferecer suporte à sincronização em redes que possuem diversas sub-redes. Estes relógios tipicamente possuem mais de duas portas, com uma porta desempenhando a função de um PTP escravo, e as outras portas servindo como PTP mestre para os relógios pertencentes a camada inferior da rede. O *Boundary Clock* é tipicamente utilizado apenas como um elemento da rede, e não é associado com dispositivos de aplicações, como sensores e

Header 1588 V1										
Bits								Octets	Offset
7	6	5	4	3	2	1	0		
versionPTP								2	0
versionNetwork								2	2
subdomain								16	4
messageType								1	20
sourceCommunicationTechnology								1	21
sourceUUID								6	22
sourcePortId								2	28
sequenceId								2	30
control								1	32
reserved								1	33
flags								2	34

Figura 12 – Header da Versão um do Protocolo

Header 1588 V2									
Bits								Octets	Offset
7	6	5	4	3	2	1	0		
transportSpecific				messageType				1	0
reserved				versionPTP				1	1
messageLength								2	2
domainNumber								1	4
reserved								1	5
flagField								2	6
correctionField								8	8
reserved								4	16
sourcePortIdentity								10	20
sequenceId								2	30
controlField								1	32
logMessageInterval								1	33

Figura 13 – Header da Versão dois do Protocolo

atuadores (COMMITTEE; TC; SOCIETY, 2008).

Os *Ordinary Clocks* se comunicam com a rede via duas interfaces lógicas baseadas em uma única porta física. A interface de eventos é utilizada para enviar e receber mensagens de evento, que são *timestamped* por um gerador de *timestamp* baseado no valor do relógio local. A interface geral é utilizada para enviar e receber mensagens gerais do protocolo. Um *Ordinary Clock* suporta uma simples cópia do protocolo PTP e tem apenas um estado PTP. Ele pode ser o relógio mestre do sistema, ou o relógio escravo, em um hierarquia mestre-escravo. Em algumas aplicações, como em automação industrial, um *Ordinary Clock* pode ser associado a um dispositivo aplicativo como um sensor ou um atuador. Enquanto que em aplicações na área de telecomunicações, um *ordinary clock* pode estar associado com um dispositivo marcador de

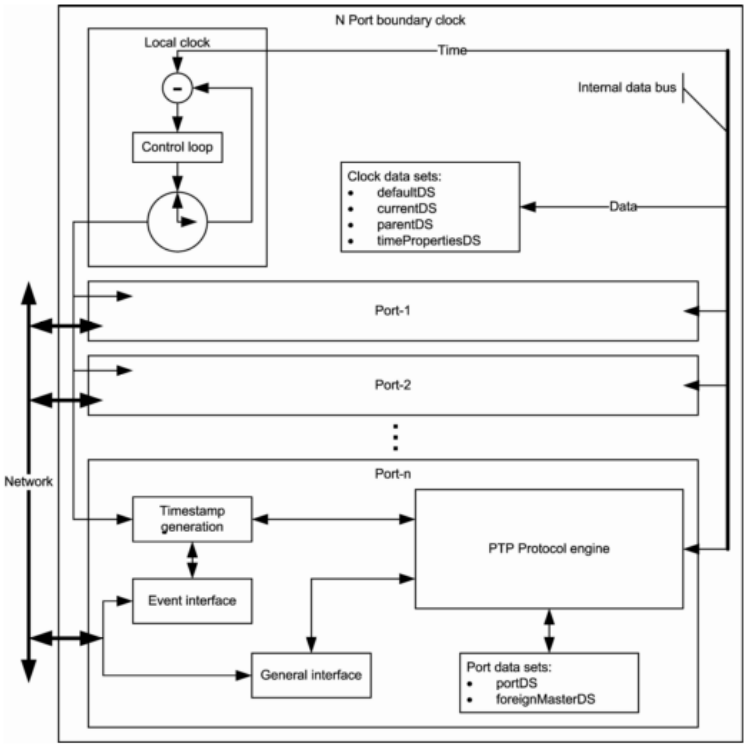


Figura 14 – Modelo do Boundary Clock

tempo.

Por último temos o *Transparent Clock*, responsável por encaminhar todas as mensagens como uma *bridge* normal, um roteador, ou repetidor. Ele mede o tempo que as mensagens levam para atravessar o dispositivo, o tempo de residência é acumulado em um campo especial das mensagens, o *correctionField*. Essa correção é baseada na diferença do *timestamp* gerado quando a mensagem entra e quando ela deixa o *transparent clock*, como mostra a Figura 16.

Os *timestamps* utilizados para computar o tempo de residência são baseados nos *timestamps* gerados pelo relógio local. Dado que esses tempos de residência acumulados são utilizados pelo escravo para ajustar o tempo provido pelo mestre, é importante que qualquer erro que resulte em diferenças nas taxas do mestre e do escravo seja insignifi-

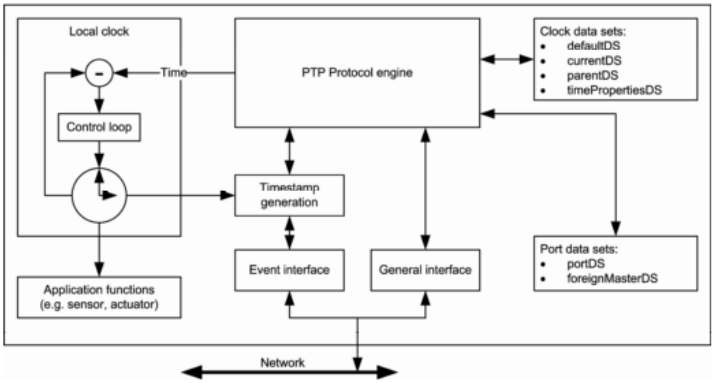


Figura 15 – Modelo do Ordinary Clock

cante para a precisão requerida pela aplicação.

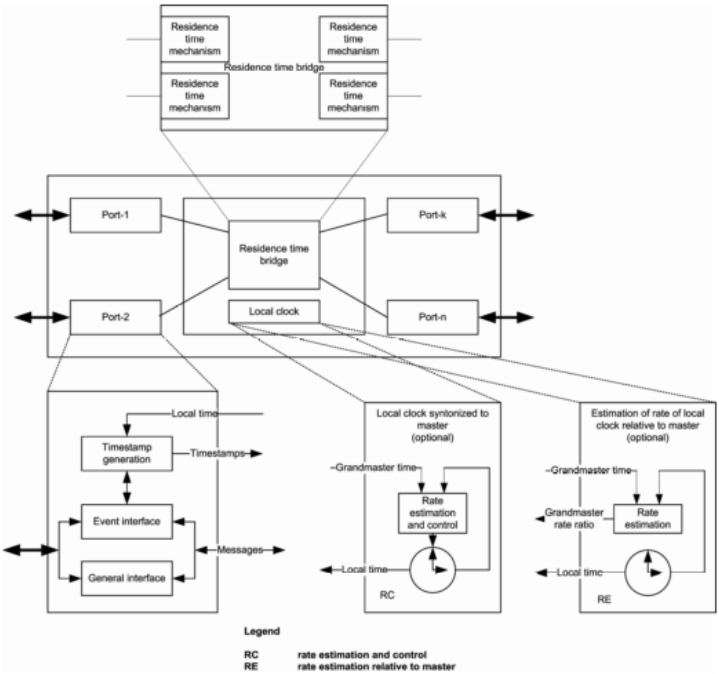


Figura 16 – Modelo do Transparent Clock

3 TRABALHOS CORRELATOS

Esta seção apresenta alguns trabalhos que utilizam o PTP no contexto de RSSFs. Dentre eles vale destacar o estudo e desenvolvimento realizado por Correll, Barendt e Branicky (CORRELL et al.,), em uma implementação software-only, chamada de PTPd, do protocolo PTP. Incluindo também os trabalhos realizados por Wobschal e Ma (MA; WOBSCHALL,) e um segundo feito por Cho, Jung, Cho, Bongrae, Jin Lee e Baek (CHO et al., 2009) que estão mais de acordo com o proposto neste trabalho.

3.1 PTPD

Neste trabalho é proposta uma implementação software-only do protocolo PTP. A sincronização de tempo com PTP se baseia em estimativas precisas ao enviar e receber *timestamps* de mensagens trocadas entre o mestre e os escravos. *Timestamps* de alta precisão podem ser conseguidos com o auxilio de hardware especializado na camada física da rede, no entanto, eles realizam uma abordagem apenas a nível de software.

Estas implementações devem efetuar o *timestamp* em camadas mais altas da rede, o que acaba por introduzir graus de não-determinismo, conhecido como *jitter*. Alcançar a sincronização exata mestre-escravo com *timestamp* sob efeitos do *jitter* é o principal obstáculo nos projetos de implementações *software-only* PTP.

3.1.1 Implementação

O *daemon* foi desenvolvido para sistemas de Teste e Medições. Para tais sistemas o PTP proporciona sincronização de tempo e frequência para o *timestamping* dos dados adquiridos. As necessidades destes sistemas influenciaram significativamente o resultado deste trabalho, mais notavelmente o *clock servo* é otimizado baseando-se na estabilidade da topologia da rede utilizada para testes e medições.

Por ser uma implementação *software-only* ele não apresenta algumas funcionalidades encontradas em implementações assistidas por hardware. Sendo elas, o fato da implementação realizar o registro do *timestamp* em camadas altas do sistema, ao invés de realizar o *times-*

tamp próximo à camada física da rede. O PTPd se destina a sistemas embarcados que possuem recursos computacionais limitados. Isso inclui plataformas com sub-100MHz de CPU. Sendo assim a utilização do *daemon* utiliza cerca de 1% da CPU de um processador de 66 MHz m68k. Além disso, PTPd não requer uma unidade de ponto flutuante(FPU), ou emulação de FPU, pois utiliza somente aritmética de ponto fixo.

3.1.2 Equipamentos

Para a realização dos experimentos foi utilizado o EX1048, que é uma plataforma linux embarcada m68k de 66Mhz. Os testes foram feitos com o PTPd conectado através de um hub *Ethernet*. O relógio mestre utilizado foi um Agilent LXI IEEE-1588 Demonstration Kit, disponibilizado pelo LXI Consortium.

3.1.3 Resultados

O PTPd coordenou o EX1048 conseguindo uma precisão de 10 μs. Esta precisão confortavelmente excede as necessidades da aplicação em que as taxas de amostragem não vão superar 1 kHz. Assim o *daemon* pode preencher as necessidades de aplicações que necessitam de precisão de sub-milissegundo, já que obtiveram um *offset* no intervalo de dez microssegundos em uma plataforma de 66 MHz de CPU. Sendo que essa precisão poderia ser melhorada caso fosse utilizado um dispositivo com um poder de processamento maior. Esta implementação não foi feita para ser implantada em RSSFs e utilizando a primeira versão do protocolo, o que difere da abordagem desse trabalho que fará uso da segunda versão do protocolo, no contexto das RSSFs.

3.2 IMPLEMENTAÇÃO DO PROTOCOLO PTP PARA RSSF'S

Para o trabalho nos interessa uma aplicação do protocolo para redes de sensores sem fio. Os dois trabalhos que realizaram esta abordagem tomaram caminhos diferentes, enquanto um utilizou um protocolo IEEE 1588 modificado o outro optou por um auxilio a nível de hardware para implementar o protocolo.

3.2.1 Synchronization of Wireless Sensor Networks Using a Modified IEEE 1588 Protocol

No primeiro trabalho realizado por Ma (MA; WOBSCHALL,) foi implementado o PTP, descrito por eles como um método de sincronização de tempo precisa, para sincronizar sensores sem fios que empregam um *transceiver* IEEE 802.15.4 e o protocolo 6LoWPAN. Para tal, um módulo de interface sem fio do WTIM foi projetado e fabricado. É utilizado o *transceiver* IEEE 802.15.4, modelo TI CC2430, que permite o acesso a um sinal de sincronismo de hardware. A diferença dos *timestamps* entre dois WTIMs foi medido e os resultados mostram que a precisão da sincronização é melhor do que 10 microssegundos, utilizando intervalos curtos de sincronização, mas aumenta para cerca de 100 μs para intervalos mais longos de sincronização. O método foi testado para o protocolo 6LoWPAN, mas se aplica a outros sensores sem fio com base no protocolo IEEE 802.15.4.

O *transceiver* escolhido por eles foi o modelo TI CC2430 (2.4Ghz) por ser bom para o processo de sincronização, além de baseado no protocolo IEEE 802.15.4. Vale destacar dois aspectos positivos, o fácil acesso a um sinal de sincronização de hardware e a integração com um microcontrolador. O formato do tempo especificado no IEEE 1451.0 é o TAI de 64-bit, o mesmo que é utilizado pelo protocolo PTP, com uma simples diferença na questão dos *leap seconds*, pois o mesmo não possui tais informações. Assim como no PTP, os 32 bits mais significativos dizem respeito ao *epoch*, definido como sendo o número de segundos decorridos desde 1 Janeiro 1970 00:00:00, equivalente ao dia 31 de dezembro de 1969 às 23:59:51.999918 no formato UTC. (COMMITTEE; TC; SOCIETY, 2008). Os 32 bits menos significativos correspondem ao numero de nanosegundos transcorridos.

Segundos eles a precisão necessária para o *timestamp*, e consequentemente a precisão da sincronização, depende muito da aplicação que o utilizará. Para diversas aplicações *wireless*, incluindo controle industrial e monitoramento de processos, um *timestamp* preciso com uma taxa de erro abaixo de 1 milissegundo é suficiente para desempenhar o papel necessário para a maioria das aplicações.

Em contraponto as redes cabeadas providas de um protocolo de sincronização compatível com o *standard* IEEE 1588, são capazes de prover um capacidade de sincronização muito mais precisa. Assim, sincronização para rede de sensores sem fio abaixo de 100 microssegundos é um desafio dado o fato dos tempos de ida e volta das mensagens serem da ordem de 100 μs e possuírem elevada variabilidade em função

da duração da mensagem e do tráfego.

3.2.2 Precision Time Synchronization Using IEEE 1588 for Wireless Sensor Networks

No segundo trabalho consultado, (CHO et al., 2009), os autores afirmam que os dispositivos convencionais utilizados pelo protocolo PTP nas redes *Ethernet* não podem prover um método para entregar mensagens PTP para outras redes. Então, essa implementação não pode ser utilizada no contexto das RSSF's. Assim, o objetivo principal do trabalho foi extender o PTP utilizado nas redes *Ethernet* para as RSSF.

Para alcançar o objetivo proposto, um *gateway* para lidar com as trocas de mensagens entre as duas redes foi desenvolvido, sendo este implementado em uma FPGA, Xilinx Vertex 4, para comunicações em alta velocidade. A arquitetura básica da FPGA inclui um *Ethernet PHY*, um *Ethernet* MAC, um tradutor de mensagens, um conversor paralelo-serial, um controlador Zigbee, um Zigbee PHY e uma unidade de *timestamp*.

No trabalho desenvolvido por eles foi utilizada a primeira versão do protocolo, logo as mensagens de *Sync e Delay Request* tinham um tamanho de 166 bytes, na implementação para *Ehernet*, enquanto que um *frame ZigBee* possui apenas 128 bytes. Assim, a mensagem não poderia ser transmitida por completo e precisava ser fragmentada, tarefa que foi atribuída ao tradutor de mensagens da FPGA, que fragmentava as mensagens vindas da rede *Ethernet* em várias mensagens com tamanho compatível ao *frame* Zigbee.

Resultados apresentados pelos autores mostram que foi obtida uma sincronização entre os tempos de relógio para a rede de sensores sem fio com um *offset*, entre o mestre e os escravos, próximo a 200 nanossegundos. Sendo assim, este resultado é superior à outros protocolos de sincronização, que apresentaram precisão próximas a algumas dezenas de microssegundos.

4 PROTOCOLO PTP ADAPTADO AO CONTEXTO DAS RSSF'S

O objetivo deste trabalho é a implementação do IEEE 1588, um protocolo de sincronização de tempo, para sincronizar dispositivos embarcados em uma rede sem fio. Foi utilizada a versão dois do protocolo, já apresentada na Seção 2, que oferece diversas melhorias em relação à primeira versão do protocolo. A mais significante diz respeito à redução do tamanho das mensagens, que proporcionará uma redução no uso da banda. A ideia é sincronizar os relógios dos dispositivos escravos com o relógio do dispositivo mestre, garantindo assim, que os eventos e registros de tempo em todos os nós sejam os mesmos. Levando em consideração o atraso e *drift* dos relógios, o atraso na transmissão das mensagens e as limitações de hardware inerentes às RSSF's.

A contribuição que se espera envolve boa parte das funcionalidades reais deste tipo de rede, o sincronismo de dados nos sensores é fundamental para o processo de aquisição de dados e controle. Aplicações em redes de sensores sem fio, similarmente à outros sistemas distribuídos, requerem um serviço de sincronização de tempo. Os experimentos envolvem a implementação do protocolo para o contexto das redes de sensores sem fio.

4.1 IMPLEMENTAÇÃO DO PROTOCOLO PARA O SISTEMA OPERACIONAL EPOS

Considerando a especificação do protocolo, (COMMITTEE; TC; SOCIETY, 2008), e os trabalhos relacionados consultados, chegamos à seguinte máquina de estados que representa o funcionamento básico do sistema, Figura 17.

Após a modelagem, foram implementadas todas as funcionalidades para operar o protocolo no contexto *Ethernet*, seguindo toda a estrutura das mensagens definida pelo protocolo. De um modo geral a implementação ocorreu usando-se a linguagem de programação C++, mas alguns pontos devem ser levantados. Para inserir o protocolo no contexto das RSSF algumas modificações nas mensagens foram feitas com o objetivo de otimizar a implementação e o uso da banda.

Conforme especificado no Apendice A, foram utilizados nodos sensores EPOSMotes. As configurações físicas do EPOSMote utilizado, que valem ser destacadas, são: *machine* MC13224V, oscilador com

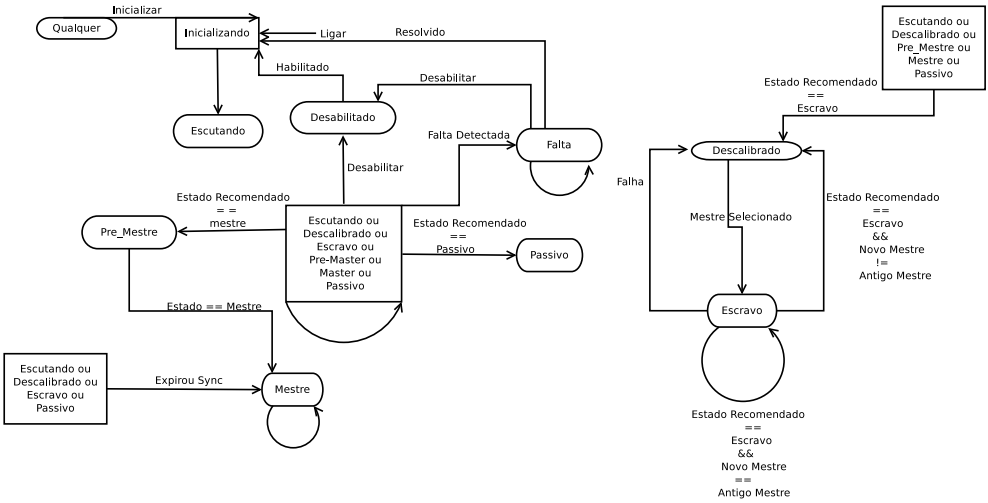
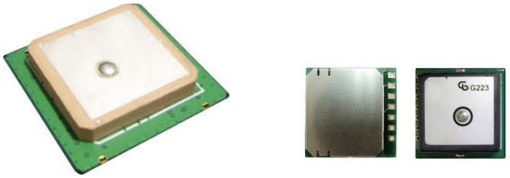


Figura 17 – Máquina de Estados do PTP

frequência nominal de 24.000Mhz e uma taxa de erro de 50 ppm e uma antena de 2.4 Ghz. Para a obtenção do tempo do relógio pelo mestre, já que o protocolo segue uma hierarquia mestre-escravo, foi utilizado um GPS, Apendice C, conectado à entrada serial do nodo mestre e utilizado como fonte primária de tempo. A máquina de estados da implementação do *driver* GPS é apresentada na Figura 19 e o modelo do receptor GPS na Figura 18.



(a) GPS ME1000RW (b) GPS 622R

Figura 18 – Imagens dos receptores GPS

Porém era necessário obter o tempo decorrido no sistema, principalmente durante as trocas de mensagens, de uma forma precisa nos nodos escravos. Para isso foi utilizado o *Timstamp Counter*(TSC), pre-

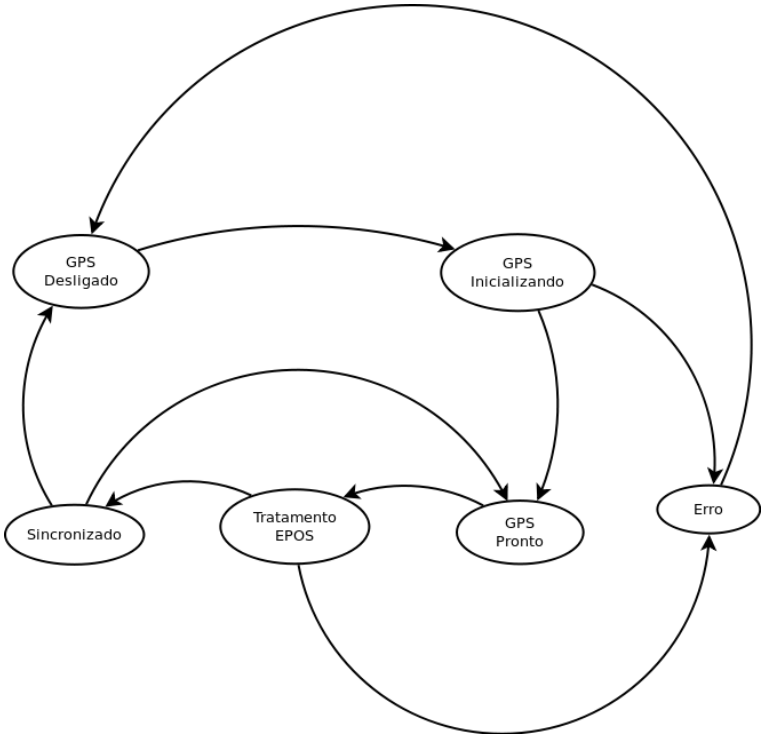


Figura 19 – Máquina de Estados do GPS

sente em todos os EPOSMotes. O TSC é um registrador de 64 bits, presente em praticamente todos os processadores x86 e em outras arquiteturas como ARM e AVR, utilizado neste trabalho para contar o tempo decorrido no dispositivo. Ele basicamente conta o número de ciclos de relógio desde o *reset* do dispositivo.

4.2 RESULTADOS

4.2.1 Metodologia

Conforme apresentado na seção anterior, o trabalho foi dividido em duas partes, sendo a primeira composta pelo desenvolvimento do protocolo para o contexto das redes Ethernet com objetivo de validação

do protocolo para o Sistema Operacional EPOS. Sequencialmente, foi feita a adaptação do protocolo para o contexto das RSSF's. Foi utilizado um receptor GPS para aquisição do tempo preciso, em segundos seguido do tempo fornecido pelo TSC, pelo mestre e esse tempo foi distribuído pela rede para sincronizar os nodos escravos.

4.2.2 Experimento

Para a realização da primeira parte do trabalho foram utilizadas duas máquinas virtuais com configurações distintas. Para a emulação das máquinas foi utilizado o Qemu versão 1.4.0. A comunicação entre as duas máquinas foi feita através da criação de uma bridge entre as duas e instanciação de uma aplicação contendo o Mestre e outra contendo o Escravo.

As mensagens de sincronização ocorreram a cada 3 segundos, sendo assim cada sincronização pertencente aos gráficos a seguir corresponde à troca de três mensagens de sincronização e quatro *timestamps*. Sendo o primeiro *timestamp* equivalente à saída da mensagem de sincronização do nodo mestre, o segundo *timestamp* correspondente à chegada da mensagem de sincronização e o terceiro e quarto *timestamps* representam os envios das mensagens para calcular o atraso da rede.

O primeiro experimento contou com os dois relógios sincronizados desde o início, sendo que a sincronização foi mantida a níveis próximos à 0 segundo durante as 250 sincronizações, como mostra a Figura 20.

Pequenas variações no *offset* foram encontradas, um valor equivalente a -1s em três ocasiões. Isto pode ser explicado através de alguns fatores que influenciam a precisão da sincronização como o atraso assimétrico e falhas no relógio.

Após a realização do primeiro experimento, seguimos com o segundo experimento onde contávamos com as duas máquinas virtuais, sendo que o escravo possuía seu relógio atrasado em relação ao mestre. Foi obtido uma variação do *offset* entre -1 e 1 segundo, porém logo após a centésima sincronização o *offset* começou a se estabilizar, chegando a marca de 0s, mostrando que os mesmos encontram-se sincronizadas na casa de no mínimo Milissegundos, como mostra a Figura 21. Como observado no primeiro experimento obtemos algumas oscilações após obter a sincronização, ou seja, *offset* igual a 0s, que são explicadas da mesma forma.

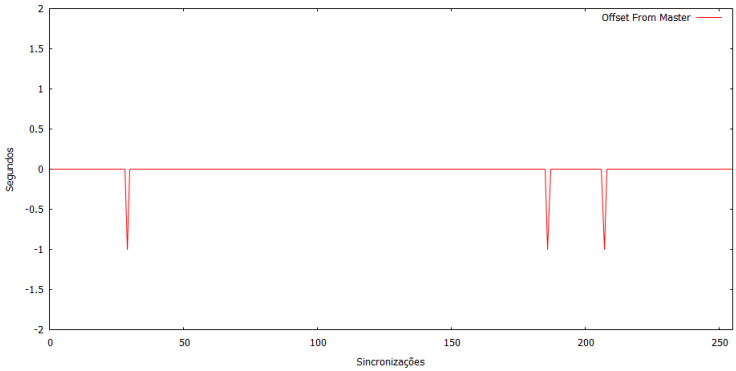


Figura 20 – Offset entre dois relógios previamente sincronizados

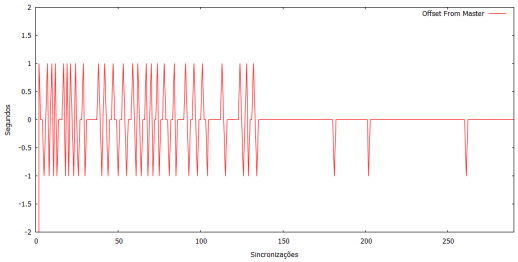


Figura 21 – Offset com relógio do escravo atrasado em relação ao relógio do mestre

O experimento foi repetido, seguindo as mesmas configurações, outras trinta vezes e obteve-se os resultados apresentados nas Figuras 22 e 23, incluindo o desvio padrão dos *offsets* calculados e também a média dos *offsets* calculados.

Percebemos que a média se mantém no intervalo [-0,006 ; -0,014]. Assim podemos dizer que o relógio se manteve na média, quando não sincronizado, à frente do tempo do relógio mestre. Este adiantamento se comprova pelo *offset* negativo. Ao analisarmos o desvio padrão observamos que o mesmo está compreendido no intervalo [0,5 ; 0,3], mostrando assim uma variância entre 0,025 e 0,09 do *offset*.

De porte destes dados e conceitos foi realizada a integração do módulo receptor GPS com o EPOSMote, tornado assim o GPS um

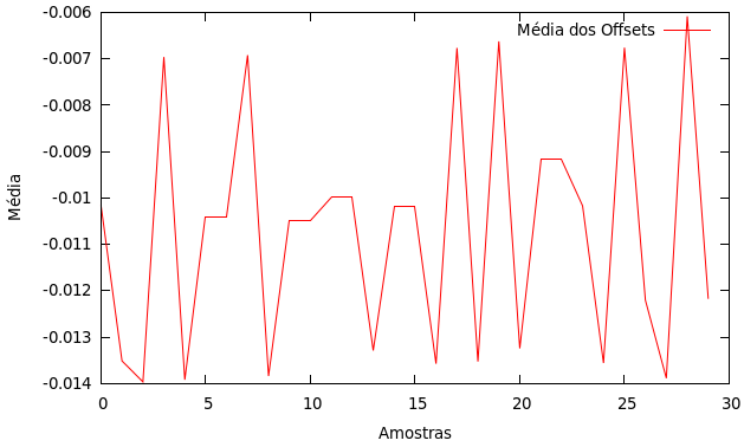


Figura 22 – Média do Offset com relógio do escravo atrasado em relação ao relógio do mestre após 30 experimentos

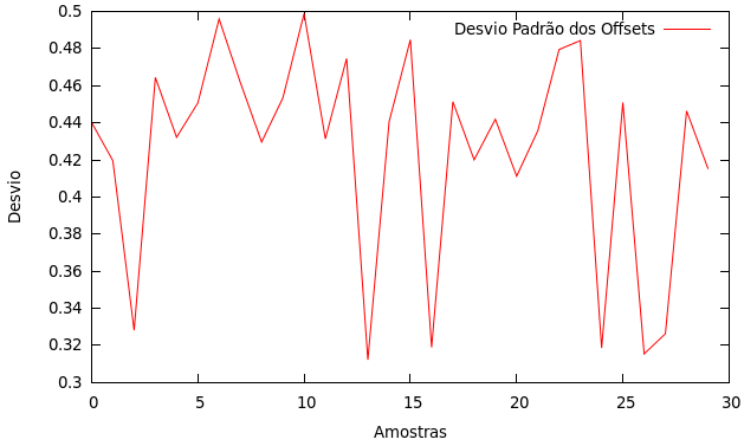


Figura 23 – Desvio Padrão do Offset com o relógio do escravo atrasado em relação ao relógio do mestre após 30 experimentos

Real-Time Clock (RTC) para o Mote. O tempo recebido através do GPS foi utilizado como fonte primária para o Mestre e distribuído pela rede, junto com o tempo fornecido pelo TSC. Similarmente ao experi-

mento anterior foi mantida a topologia mestre-escravo, sendo que neste experimento o mestre e o escravo são dois EPOSMotes, e o cristal que compõe o TSC possui um erro de 50 ppm, que pode ser evidenciada nas Figuras 24 e 25, além da frequência de sincronizações ser de 10 segundos.

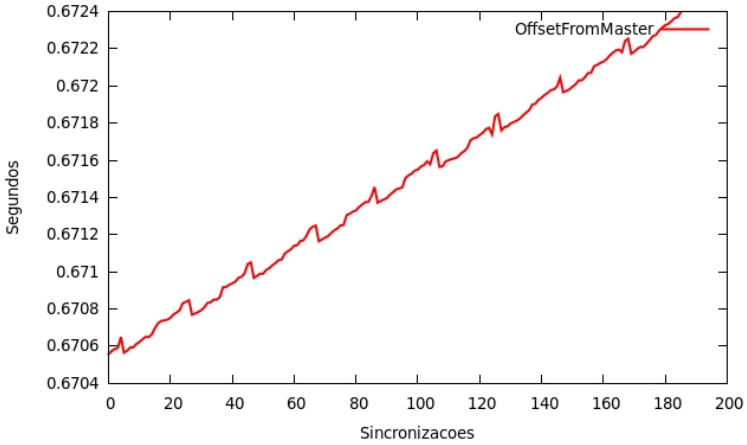


Figura 24 – Offset calculado entre um EPOSMote mestre e outro EPOSMote escravo

Percebemos que em ambas figuras um dos relógios parece "correr"mais rápido do que o outro, tomamos como exemplo a figura 25, onde após 450 sincronizações foi observado uma diferença de 40 milissegundos nos tempos dos relógios. Tal diferença está em conformidade com o que é apresentado pelo fabricante do cristal utilizado no EPOSMote, que prevê tal erro. Este resultado só nos ressalta a importância de possuirmos mecanismos de sincronização de tempo.

Após estes experimentos iniciais, foi realizada a sincronização dos nodos utilizando o protocolo PTP, obtendo os resultados apresentados na Figura 26.

Para analisarmos melhor a figura 26, e consequentemente os resultados obtidos, efetuaremos um zoom na área contendo os dados plotados para verificar os valores dos *offsets*, como é apresentado na Figura 27.

Tal *offset* está situado majoritariamente entre 200 μs e 400 μs, possuindo uma média 337 μs, uma mediana em 261 μs e um desvio padrão em $1.8.3 * 10^{-4}$. Assim, teríamos que o *offset* real seria esse

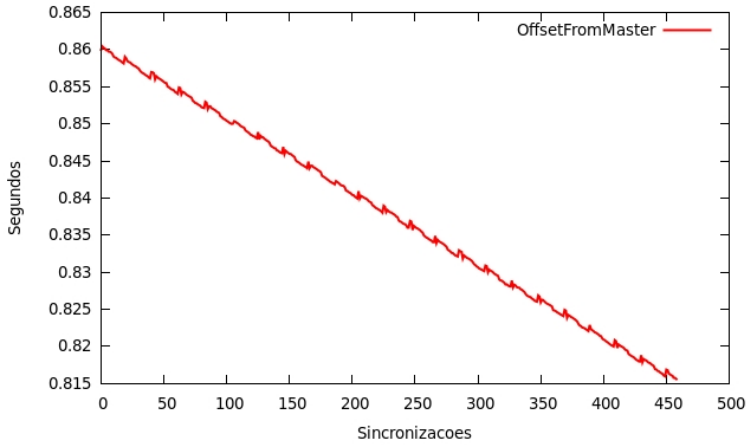


Figura 25 – Offset calculado invertendo-se o escravo e o mestre da figura 24

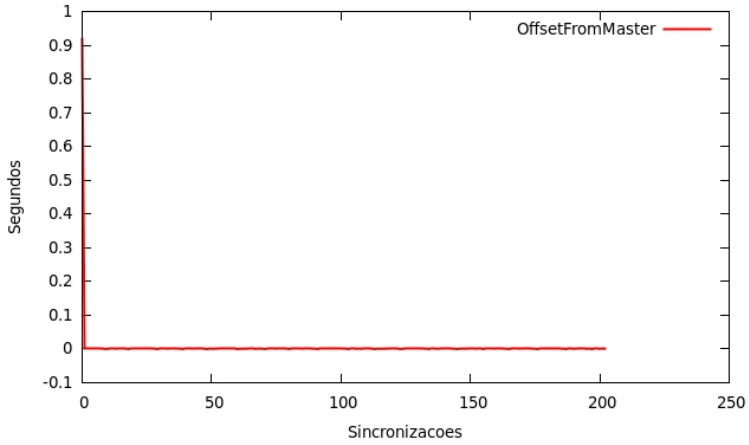


Figura 26 – Offset calculado entre os nodos mestre e escravo utilizando o protocolo PTP

offset mais o escorregamento que o relógio terá com o tempo, o que é conhecido de cada oscilador, assim poderíamos fazer um ajuste de escorregamento do relógio levando-se em conta o pior caso, 100ppm,

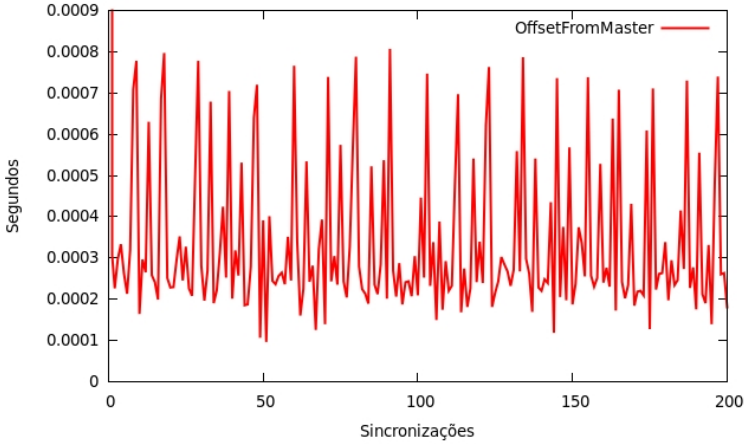


Figura 27 – Zoom no offset calculado entre os nodos mestre e escravo utilizando o protocolo PTP

para cada *offset*. O *offset* obtido nos assegura, segundo (RÍO et al., 2012), a viabilidade de implementação para sistemas de sensoriamento oceânico, já que a precisão alcançada fica abaixo de 1 milissegundo. Vale também analisar o experimento anterior repetido por um período de tempo maior, apresentado na figura 28.

A partir dos resultados obtidos, começamos a investigar a inserção de um terceiro nodo na rede, já que normalmente uma rede é composta por mais de 2 sensores. Assim, considerando que trabalhamos na precisão de microssegundos, optamos por uma abordagem diferente da tradicional, onde são inseridos nodos escravos para se sincronizarem com o nodo mestre. Na nossa abordagem verificamos que tal a maneira tradicional pode gerar um *overhead* de mensagens na rede de acordo com a equação a seguir.

$$\lambda = \epsilon * 3 * \phi \tag{4.1}$$

Sendo, o número de mensagens trocadas na rede a cada ϵ intervalo de tempo pelos nodos, ϕ, igual à λ. Tal *overhead* pode ser reduzido se optarmos por, ao invés de inserirmos nodos escravos, colocarmos nodos que apenas escutem a rede, e consequentemente as mensagens de sincronização trafegadas por ela, seguindo a equação abaixo. A constante 3 é substituída pela constante 4, pois teremos que enviar 1 men-

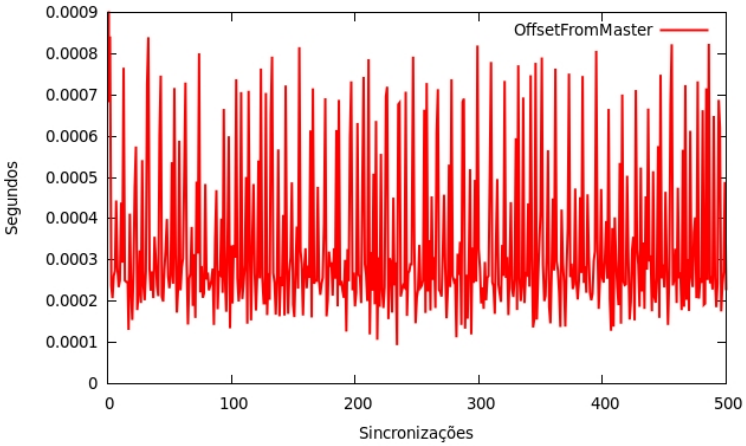


Figura 28 – Zoom no valor absoluto offset calculado entre os nodos mestre e escravo utilizando o protocolo PTP por um tempo superior à primeira

sagem a mais, de Delay Req, para a rede, pois com as 3 mensagens normalmente trocadas entre um mestre e um escravo não temos acesso ao *timestamp* t_2 do escravo, assim precisamos fazer um *broadcast* desse Delay Req que será interpretado apenas pelos ouvintes.

$$\lambda = \epsilon * 4 \tag{4.2}$$

De posse dos *timestamps* trocados o nodo que está escutando a rede pode sincronizar o seu relógio, considerando o *offset* do nodo escravo. Assim, a taxa de erro estaria relacionada ao escorregamento do relógio do nodo que esta escutando em relação ao relógio do nodo escravo. O que considerando o manual do oscilador é conhecido, no nosso caso temos um intervalo de erro de [-50ppm ; 50ppm]. Os resultados preliminares obtidos utilizando esta abordagem seguem na figura 29, sendo que ela apresenta uma visão ampliada dos dados, enquanto que a Figura 30 mostra uma visão em uma escala menor.

Percebemos, ao analisar os resultados que alguns valores de *offset* obtidos pelo nodo escutador possuem valores altos, para fins de representação gráfica setamos que qualquer valor maior que 1 milissegundo será igual à 1 segundo. Isso ocorre pelo não recebimento, pelo escutador, de todas as mensagens que são trocadas pela rede, assim o

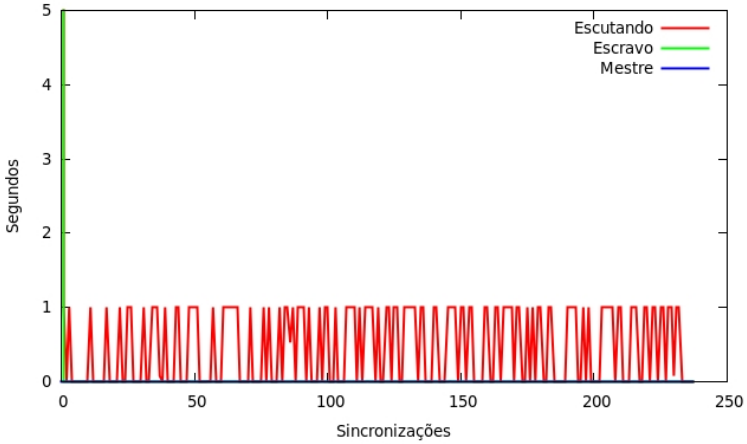


Figura 29 – Valor absoluto offset calculado entre os nodos mestre, escravo e escutador

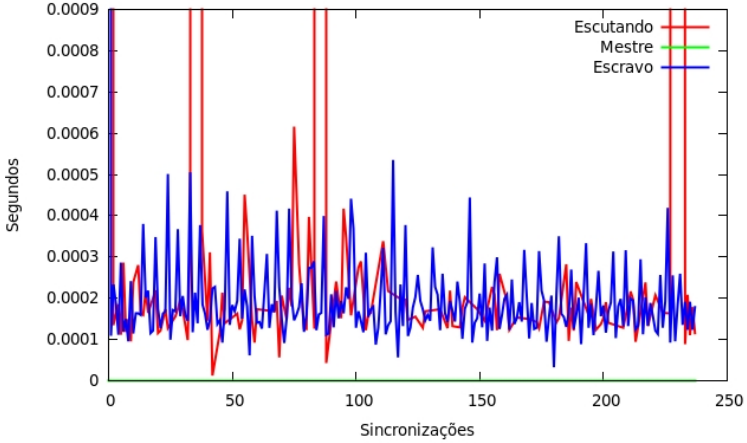


Figura 30 – Zoom no valor absoluto offset calculado entre os nodos mestre, escravo e escutador.

calculo do *offset* fica prejudicado, o que leva à esses resultados. O *offset* volta a ficar próximo ao *offset* do escravo quando ocorre uma nova sincronização, e da maneira como é feito o calculo do *offset*, através da acu-

mulação de *offsets* anteriores, os nodos voltam a ter *offsets* próximos.

5 CONCLUSÃO

Após realizarmos os testes iniciais, com as máquinas virtuais, concluímos a viabilidade da implementação do protocolo PTP para realizar a sincronização dos tempos de relógio em um sistema operacional embarcado, pois conseguimos manter o *offset* próximo a 0 segundo. Isso nos forneceu uma base para trabalharmos a implementação com o intuito de obter um *offset* na faixa de sub-milissegundos e implantarmos a solução em RSSFs. Obtendo essa precisão conseguimos garantir a aplicação, por exemplo, da implementação para sistemas de sensoriamento oceânico, como é abordado em (RÍO et al., 2012). O fato da necessidade de um protocolo deste tipo para tal escopo ocorre pelo fato de sinais GPS não estarem disponíveis nas estações oceânicas devido à atenuação do sinal pela água e requisitos de sincronização de instrumentos marítimos, tais como sismógrafos. Após os resultados obtidos utilizando-se sensores podemos concluir que o objetivo de sincronizar dois sensores sem fio foi atingido, foi obtido um *offset* médio igual a 337 μs, que é suficiente para ser utilizado em aplicações que necessitam de uma precisão de sub-milissegundo. Ressaltamos também que a abordagem que nos pareceu mais interessante para tratar a sincronização de mais nodos na rede, por reduzir o *overhead* causado pelas mensagens que seriam inseridas, chegou à resultados interessantes, podendo ser aperfeiçoada para gerar resultados mais próximos ao *offset* do nodo escravo.

5.1 TRABALHOS FUTUROS

É possível realizar testes em uma configuração contendo mais nodos, seguindo a abordagem que foi feita, levando em consideração todas as variáveis que podem influenciar no atraso e utilizando-as para realizar uma compensação no *offset*, alcançando uma maior precisão. Assim como uma abordagem na questão de segurança para as mensagens trocadas pelo protocolo, tendo em vista que qualquer dispositivo pode se inserir na rede e começar a fazer parte do sistema, trocando mensagens de sincronização, além de poder se passar por um mestre e alterar os relógios dos dispositivos. Atenção também deve ser dada ao algoritmo para selecionar o Mestre, algoritmo do Best Master Clock, no caso de uma implementação autônoma e auto-gerenciável. Uma abordagem que leve em conta outros parâmetros na escolha do mestre

deve ser analisada levando em consideração características específicas do nodo, como a carga de bateria, poder computacional, além da qualidade da fonte de *clock*.

REFERÊNCIAS

CEPA. *CEPA - Centro de Ensino e Pesquisa Avançada*. Brazil: CEPA, 2012.

CHO, H. et al. Precision Time Synchronization Using IEEE 1588 for Wireless Sensor Networks. *2009 International Conference on Computational Science and Engineering*, Ieee, p. 579–586, 2009. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5284255>.

COMMITTEE, T.; TC, T.; SOCIETY, M. *IEEE Std 1588-2008, IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*. [S.l.: s.n.], 2008. ISBN 9780738154008.

CORRELL, K. et al. Design Considerations for Software Only Implementations of the IEEE 1588 Precision Time Protocol. *Design*, v. 1, p. 2–7.

ELSON, J.; GIROD, L.; ESTRIN, D. Fine-grained network time synchronization using reference broadcasts. *ACM SIGOPS Operating Systems Review*, v. 36, n. SI, p. 147, dez. 2002. ISSN 01635980. <http://portal.acm.org/citation.cfm?doid=844128.844143>.

FRöHLICH, A. A. M. *Application-Oriented Operating Systems*. 2001.

KARL, H.; WILLIG, A. *No Title*.

KOPETZ, H.; SCHWABL, W. Global time in distributed real-time systems. *Update*.

LISHA Team. *EPOSMote*. Brazil: [s.n.], 2011. <http://epos.lisha.ufsc.br>.

MA, Y.; WOBSCHALL, D. Synchronization of Wireless Sensor Networks Using a Modified IEEE 1588 Protocol. *Update*.

MARóTI, M.; SIMON, G. The Flooding Time Synchronization Protocol. *Time*, 2004.

MESSAGES, A.; MESSAGES, I.; FORMAT, E. Table of contents. *Clinical infectious diseases : an official publication of the Infectious Diseases Society of America*, v. 54, n. 6, p. NP, mar. 2012. ISSN 1537-6591. <http://www.ncbi.nlm.nih.gov/pubmed/22357815>.

MILLS, D. L. Improved algorithms for synchronizing computer network clocks. *ACM SIGCOMM Computer Communication Review*, v. 24, n. 4, p. 317–327, out. 1994. ISSN 01464833. <http://portal.acm.org/citation.cfm?doid=190809.190343>.

MINAR, N. A Survey of the NTP Network 1 The NTP Network. *Methodology*, 1999.

NISTGOV. *NISTGOV*. EUA: NISTGOV, 2012.

RATON, B.; NEW, L.; WASHINGTON, Y. *Sensors Handbook*. [S.l.: s.n.]. ISBN 0849319684.

RÍO, J. del et al. Precision timing in ocean sensor systems. *Measurement Science and Technology*, v. 23, n. 2, p. 025801, fev. 2012. ISSN 0957-0233. <http://stacks.iop.org/0957-0233/23/i=2/a=025801?key=crossref.9545ac66c505e0a9e95220f461a7b892>.

SARI, I. et al. On the Joint Synchronization of Clock Offset and Skew in RBS-Protocol. *IEEE Transactions on Communications*, v. 56, n. 5, p. 700–703, maio 2008. ISSN 0090-6778. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4524851>.

SILBERSCHATZ. *Operating System Concepts*. [S.l.: s.n.], 2008. ISBN 9780738154008.

SU, W.; AKYILDIZ, I. F. Time Translation Algorithm. *Update*.

TANENBAUM. *Computer Networks*. [S.l.: s.n.], 1993. ISBN 0131668366.

WANNER, L. F. The EPOS System Supporting Wireless Sensor Networks Applications The EPOS System Supporting Wireless Sensor Networks Applications. *Direct*.

WEIBEL, P. H.; BéCHAZ, D. Implementation and Performance of. 2004.

APÊNDICE A – Sistema Operacional EPOS

O Sistema Operacional EPOS nasceu em 1997 no FIRST do GMD como um projeto experimental seguindo os conceitos e mecanismos de um sistema orientado à aplicações. O sistema foi concebido considerando o principal objetivo de pesquisa: embarcar o sistema operacional em uma aplicação paralela. (FRöHLICH, 2001) A estratégia adotada para se atingir este objetivo consistiu em modelar as entidades do domínio com um conjunto de componentes reutilizáveis e adaptáveis.

O EPOS deve muito de sua filosofia ao sistema PEACE(FRöHLICH, 2001), de onde ele herdou a noção de que "genérico"e "ótimo"são adjetivos que não podem ser aplicados simultaneamente. As famílias de abstrações do EPOS resultaram da decomposição do domínio. Diversas entidades pertencentes a este domínio, que envolve os sistemas operacionais, são convencionadas por cientistas da computação e desenvolvedores de sistema. Por isso o EPOS recorreu a livros tradicionais de Sistemas Operacionais, como (TANENBAUM, 1993) e (SILBERSCHATZ, 2008) para obter uma referencia em relação ao vocabulário utilizado neste domínio.

Esta decisão contribuiu em tornar o EPOS um Sistema Operacional amigável, dado que suas abstrações foram nomeadas após o conceito clássico que elas representam, dando a elas um caráter autoexplicativo. Assim, o sistema operacional EPOS é um sistema operacional voltado para aplicações embarcadas. Ele foi programado em C++, utilizando-se de técnicas como orientação a aspectos e metaprogramação estática para alcançar um alto grau de configurabilidade, ao mesmo tempo, que atende requisitos estritos de sistemas embarcados.

Conceitos fundamentais de sistemas operacionais, como escalonador, threads e semáforos, são descritos como famílias de abstrações. Técnicas de programação orientada a aspectos são utilizadas para adaptar essas abstrações às peculiaridades do ambiente em que o sistema irá executar. A unidade responsável por realizar essa adaptação da abstração ao cenário de execução é convenientemente chamada de adaptador de cenário. Um dos ideais do EPOS é que os componentes do sistema operacional possam ser implementados tanto em *software* quanto em *hardware*.

APÊNDICE B – A Subcamada MAC

As redes podem ser divididas em duas categorias: aquelas que utilizam conexões ponto-a-ponto e as que utilizam canais *broadcast.*(TANENBAUM, 1993) em qualquer rede *broadcast*, o fator chave é determinar quem terá o direito de usar o canal de comunicação quando existir uma concorrência para tal.

A subcamada MAC (Medium Access Control) é importante em LAN's, atualmente todas utilizam canais de multiacesso como base para a comunicação, enquanto que nas WAN's são utilizados links ponto-a-ponto. LAN's em geral possuem três características : um diâmetro de apenas alguns kilometros, uma taxa de dados de no mínimo vários Mbps e é propriedade de uma única organização. WAN's em contraste, tipicamente envolve uma distribuição em torno do país inteiro, possui uma taxa de dados abaixo de 1 Mbps e são propriedade de diversas organizações. Podemos citar também que entre as LAN's e WAN's temos as MAN's (Metropolitan Area Network), que nada mais são do que uma rede que envolve uma cidade inteira e utiliza a tecnologia empregada pelas LAN's.

Levando em consideração estas topologias podemos focar na maneira que elas utilizam, em especial as LAN's, para alocarem um simples canal, como um tronco de telefone. Uma das maneiras tradicionais de alocação de um simples canal é utilizando FDM (Frequency - Division Multiplexing). Por exemplo, dada uma rede com N usuários, a banda disponível é dividida em N partes iguais, onde cada usuário é alocado para uma dessas partes. Como cada usuário possui uma única banda de frequência privada, não existe interferência entre os dados trocados pelos usuários. Quando existe apenas um número pequeno e fixo de usuários, e cada um gera uma grande quantidade de tráfego, FDM é um mecanismo simples e eficiente de alocação.

No entanto, quando o número de usuários é grande e varia constantemente, ou o trafego é transferido ou transmitido em pequenas e irregulares rajadas, o FDM apresenta alguns problemas. Se o espectro é dividido em N sub-regiões, e um número inferior a N usuários estão interessados em realizar algum tipo de comunicação utilizando o canal de comunicação, uma grande parte do espectro será desperdiçada. Caso mais de N usuários tenham a intenção de utilizar o canal de comunicação, alguns deles terão esta permissão negada, devido à falta de banda disponível, até mesmo se alguns usuários que foram alocados algumas frequências as utilizarem muito pouco.

O problema básico é que se alguns usuários que foram alocados frequências não as utilizarem, este espaço alocado será perdido, pois nenhum outro usuário poderá utilizá-lo. A baixa performance do

FDM estático pode ser visto como um simples calculo da teoria de filas (TANENBAUM, 1993). Supondo que possuímos o atraso médio, D, para um canal de capacidade C bps, com uma taxa de chegada X quadros/segundo, cada quadro possuindo um tamanho de acordo com uma função exponencial probabilística com média 1/M bits/quadro.

Obteríamos:

D = 1/MC-X

Agora se dividirmos um simples canal em N sub canais independentes, cada um com capacidade C/N bps. A média da taxa de entrada em cada sub canal será agora X/N. Recalculando D teríamos:

DFDM = NT

O atraso médio utilizando FDM é N vezes pior se todos os quadros fossem magicamente ordenados em uma grande fila central. Este resultado nos leva a pensar em uma alternativa, e uma das alternativas propostas foi a alocação dinâmica de canal.

B.1 ALOCAÇÃO DINÂMICA DE CANAL

Algumas premissas chaves sobre o problema de alocação são:

Station Model Single Channel Assumption Collision Assumption Continuous Time Slotted Time Carrier Sense No Carrier Sense

Um dos protocolos que utiliza essa técnica de alocação dinâmica é o Aloha, que foi concebido em 1970 por Norman Abramson junto com sua equipe de pesquisa na Universidade do Havaí. Ele foi proposto como um novo método para resolver o problema da alocação de canal. A ideia básica do protocolo Aloha é simples: deixe os usuários transmitirem a qualquer momento em que eles possuam dados para serem enviados.[Y]

Através deste sistema haverão colisões e os quadros que colidirem serão destruídos. No entanto, devido a propriedade de *feedback* do *broadcast*, o usuário que enviou pode sempre descobrir caso um quadro seu tenha sido destruído, escutando a saída do canal.

Caso o quadro tenha sido destruído, o usuário que enviou aguarda um tempo aleatório e envia o quadro novamente. O tempo de espera precisa necessariamente ser aleatório, pois caso contrário haverá uma nova colisão entre os quadros em questão.

Em 1972, Roberts publicou um método para dobrar a capacidade a capacidade de um sistema Aloha. Sua proposta se baseia em dividir *time up* em intervalos discretos de tempo, cada intervalo correspondendo à um quadro. Uma maneira de obter sincronização entre os usuários seria através de uma estação especial que emitiria um *pip* no

inicio de cada intervalo, como um relógio.

No método proposto por Robert, que acabou sendo conhecido como *slotted* Aloha, o *terminal* não é permitido enviar assim que estiver com a mensagem pronta, ele deve esperar que haja um *slot* disponível. Utilizando este método a melhor utilização do canal que pode ser obtida é equivalente à 1/e. Este resultado não é surpreendente, pois com estações transmitindo à sua vontade, sem prestar atenção ao que está sendo transmitido pelas outras estações, deixa margem à colisões. Em redes locais, este método pode atingir um nível de utilização superior a 1/e.

B.2 CSMA

Protocolos em que uma estação escuta por uma portadora e age de acordo com este fato são chamados *Carrier Sense Protocols*. Alguns exemplos destes protocolos são:

1-*persistent* CSMA (Carrier Sense Multiple Access): Quando um usuário possui dados para enviar, ele primeiro escuta o canal para ver se alguém está transmitindo alguma mensagem. Caso o canal esteja ocupado, o usuário espera até que o mesmo esteja disponível. Quando o usuário detecta que o canal está ocioso, ele transmite um quadro. Caso ocorra uma colisão, o usuário espera uma quantidade de tempo aleatória para transmitir novamente.

O atraso de propagação possui um efeito importante no desempenho do protocolo. Existe uma pequena probabilidade de logo após um usuário iniciar uma transmissão, um outro usuário esteja pronto para começar a ouvir o canal e transmitir. Caso o sinal do primeiro usuário ainda não tenha alcançado o do segundo, o mesmo só escutará o canal ocioso e também começará a transmitir, resultando em uma colisão. Quanto maior for o atraso de propagação, mais importante será o efeito acima, e pior será o desempenho do protocolo.

Mesmo que o atraso de propagação seja zero, mesmo assim haverá colisões. Caso dois usuários estejam prontos para transmitir durante a transmissão de um terceiro usuário, ambos aguardarão este terceiro acabar a transmissão e logo em seguida começarão a transmitir ao mesmo tempo, resultando em uma colisão.

O segundo protocolo proposto foi o CSMA não-persistente, neste protocolo, uma abordagem consciente é feita para torna-lo menos ganancioso que o anterior. Antes de enviar, o usuário escuta o canal. Caso ninguém esteja transmitindo no mesmo, o usuário começa sua trans-

missão. No entanto, caso o canal esteja em uso, o usuário espera por uma quantidade de tempo aleatório e repete o procedimento verificando se o canal está em uso.

O último protocolo proposto foi o p-persistent CSMA, basicamente ele aplica a noção de *slots* e funciona da seguinte maneira. Quando um usuário está pronto para transmitir, ele escuta o canal. Caso o canal esteja ocioso, ele transmite com uma probabilidade p. Com uma probabilidade q = 1-p ele adia até o próximo *slot*. Caso esse próximo *slot* também esteja ocioso, ele transmite e adia novamente, com probabilidades p e q. Esse processo é repetido até que o quadro seja transmitido ou um outro usuário comece a transmitir. No último caso, ele age como se houvesse uma colisão, aguardando uma quantidade de tempo aleatória e realizando o procedimento novamente. Caso o usuário escute o canal e o mesmo esteja ocupado, ele espera até o próximo *slot* e repete o procedimento acima.

Os protocolos CSMA persistentes e não-persistentes são claramente um aperfeiçoamento do protocolo Aloha porque eles garantem que nenhum usuário começará a transmitir enquanto ele o canal estiver ocupado. Uma outra melhora garante que os usuários irão abortar a transmissão assim que eles detectarem uma colisão. Em outras palavras, caso dois usuários escutem o canal de transmissão ocioso e comecem a transmitir simultaneamente ambos detectarão uma colisão quase que imediatamente e devem então cessar a transmissão assim que houver essa detecção.

O protocolo CSMA/CD é amplamente utilizado em LANs, na subcamada de MAC, e utiliza o seguinte modelo:

CSMA/CD pode estar em um dos três estados: contenção, transmissão ou ocioso.

Assim que um usuário acabar sua transmissão. Qualquer outro usuário que possua um quadro para enviar pode agora começar sua transmissão. Caso dois ou mais usuários decidam transmitir simultaneamente, haverá uma colisão. Cada um detectará a colisão, abortará sua transmissão, aguardará um período de tempo aleatório, e tentará transmitir novamente, assumindo que nenhum usuário começou a transmitir no meio tempo.

Portanto o CSMA/CD consistirá de uma alternância entre períodos de transmissão e contenção, com períodos de ociosidade ocorrendo quando todos os usuários estiverem dormindo.

B.3 IEEE STANDARD 802 FOR LOCAL AREA NETWORKS

O IEEE produziu diversos padrões para as redes locais. Esses padrões, coletivamente conhecida como IEEE 802, incluem CSMA/CD, *token bus* e *token ring*. Os padrões são divididos em partes, cada uma publicada em um livro específico. O padrão 802.3 é para o CSMA 1-persistente, diversas pessoas utilizam o nome *Ethernet* de uma maneira genérica para se referir a todos os protocolos CSMA/CD, mesmo que ele refira a um produto específico que implementa o padrão 802.3.

Formato de Quadro do 802.3

Cada quadro inicia com um preâmbulo de 7 bytes, cada um contendo o padrão de bits 10101010. A codificação Manchester desse padrão produz uma onda quadrática de 10 Mhz por 5.6 segundos para permitir que o relógio do destinatário se sincronize com o relógio do remetente. O próximo campo indica o inicio do delimitador de quadro, contém o padrão 10101011.

O quadro contém dois endereços, um para o endereço de destino e outro para o endereço do remetente. O padrão permite endereços de 2 ou 6 bytes, mas os parâmetros definidos para a banda de 10 Mbps segundo o padrão utilizam apenas endereços de 6 bytes. O bit de mais alta ordem do endereço de destino é 0 para endereços comuns e 1 para grupos de endereços. Grupos de endereços permitem à múltiplos usuários escutarem a apenas um endereço. O endereço composto por todos os bits 1's é reservado para o broadcast. Um aspecto interessante do endereçamento é o uso do bit 46 (adjacente ao bit de maior ordem) para distinguir endereços globais de locais.

O campo referente ao tamanho do campo de dados informa quantos bytes estão presentes no campo de dados, com um número mínimo de 0 até um máximo de 1500. Enquanto um campo de dados de tamanho 0 é permitido, ele pode causar um problema. Quando um transceiver detecta uma colisão, ele trunca o quadro atual, que significa que bits perdidos e pedaços de quadros aparecem no cabo a todo o momento. Para facilitar a distinção de quadros válidos para lixo, o 802.3 toma como premissa que quadros válidos devem ter um tamanho mínimo de 64 bytes, contando desde o endereço de destino até o campo de checksum. Se a porção de dados do quadro é menor que 46 bytes, o campo pad é utilizado para completar o requisito de tamanho mínimo. Uma outra razão para o quadro possuir um tamanho mínimo está relacionado com a intenção de prevenir que um usuário complete a transmissão de um quadro pequeno antes que o primeiro bit tenha alcançado o fim do cabo, onde ele pode colidir com outro quadro.

O ultimo campo diz respeito ao checksum. Ele é efetivamente um hash-code de 32 bits dos dados do quadro. Caso alguns bits de dados recebidos estejam errados, o checksum sera incompatível com o esperado, e assim o erro será detectado. O algoritmo do checksum é uma avalização de redundância cíclica. O algoritmo, binary exponential backoff, foi escolhido para adaptar dinamicamente ao número de usuários que querem enviar dados. Se o intervalo aleatório para todas as colisões fosse 1023, a chance de dois usuários colidirem uma segunda vez seria insignificante, mas o tempo médio de espera após a colisão seria de centenas de tempos de *slot*, introduzindo um atraso significante.

O CSMA/CD não oferece reconhecimentos. Dado que a simples falta de colisões não garante se os bits foram modificados por algum ruido no cabo, para comunicação confiável o destinatário deve verificar o checksum, e se estiver incorreto, enviar um frame de reconhecimento ao usuário fonte.

B.4 TDMA

Segundo Karl (KARL; WILLIG,), o TDMA subdivide o eixo de tempo em superframes de tamanho fixo e cada superframe é novamente subdividido em u número determinado de *slots* de tempo. Estes *slots* de tempo são exclusivamente atribuídos aos nodos, portanto, o nodo poderá transmitir periodicamente neste intervalo de tempo em cada superframe.

O TDMA necessita de uma sincronização de tempo precisa entre os nós para evitar a sobreposição de sinais em *slots* de tempo adjacentes, logo se mostra como uma grande área de aplicação para o protocolo estudado neste trabalho.

B.5 PROTOCOLOS PARA RSSF

Dados estes protocolos propostos e as já conhecidas restrições das redes de sensores sem fio, como as baixas faixas de sensoriamento resultando em redes densas, que nos leva a uma necessidade de obter um protocolo eficiente, em termos energéticos, para acesso ao meio. Sendo assim vários protocolos MAC com objetivos diferentes foram propostos para redes de sensores sem fio.

Assim tipos de padrões de comunicação que são observados em aplicações para redes de sensores deveriam ser investigados dado que

esses padrões são utilizados para extrair o comportamento do tráfego na rede que precisa ser manuseado por um determinado protocolo MAC[16].

Alguns dos protocolos propostos foram, o S-MAC [17], o B-MAC [18], o X-MAC, Z-MAC e o C-MAC [22]. O S-MAC inclui abordagens para reduzir o consumo de energia de todas as fontes identificadas de consumo de energia como, idle listening, colisão, overhearing e controle de sobrecarga. Uma característica do S-MAC é que ele dispõe nodos em uma topologia peer-to-peer. Diferente de protocolos que utilizam o paradigma de clusters, S-MAC não necessita de coordenação pelos cluster heads. Uma vantagem disto é que o protocolo se torna mais robusto em relação as mudanças de topologia da rede. A parte negativa é em relação ao incremento da latência devido ao sleeping periódico. Além disso, o atraso pode acumular em cada hop.

O S-MAC propõe uma importante técnica chamada de escuta adaptativa, para melhorar a latência causada pelo sleeping periódico de cada nodo em uma rede multihop. A ideia básica é permitir que o nodo que escuta a transmissão do seu vizinho acorde por um pequeno período de tempo no final da transmissão. Dessa maneira se o nodo é o nodo do próximo hop, o seu vizinho consegue a passar os dados imediatamente para ele, ao invés de esperar o seu tempo de escuta agendado.

O protocolo B-MAC é um protocolo MAC *carrier sense* para redes de sensores sem fio.[18]. Ele provê uma interface que permite reconfiguração on-the-fly, permitindo assim que os serviços de rede ajustem seus mecanismos. Aspectos como, habititar e desabilitar o uso do *clear channel assessment* (CCA) ou reconhecimentos, setando o tamanho do preâmbulo e os intervalos de escuta. Uma limitação do B-MAC é que o destino precisa aguardar até que todo o preâmbulo seja enviado para começar a trocar dados, mesmo que o mesmo estivesse acordado no inicio da transmissão. Isto adiciona ao problema de escuta, onde os destinatários permanecem acordados até o final do preâmbulo e descobrirem que o pacote não era endereçado para eles.

O X-MAC também um protocolo de MAC preocupado com questões energéticas, os objetivos do projeto deste protocolo para redes de sensores sem fio *duty-cycled* foram, a eficiência energética, uma implementação distribuída simples e com pouco *overhead*, pouca latência para dados, alto *throughput*, adaptabilidade e aplicabilidade entre todos os tipos de rádios digitais. Para diversas aplicações *duty-cycling* assíncrono são preferidas para técnicas de sincronização em termos de consumo de energia, latência e *throughput*. [19] Isto porque não

ocorre *overhead* devido à sincronização. Devemos lembrar também que técnicas assíncronas não têm que compartilhar informações de agenda e só ficam acordadas o tempo suficiente para amostrar o meio, a não ser que, estejam recebendo ou transmitindo dados. Por isso o período em que o nodo fica acordado pode ser significativamente menor do que o tempo utilizando técnicas onde há sincronização. Com um tempo acordado menor, os protocolos assíncronos podem acordar com mais frequência enquanto ainda mantêm um *duty-cycle* baixo. Consequentemente, eles experimentam latência reduzida e um maior *throughput*. No entanto, a medida que aumenta a latência, o preâmbulo estendida começa a dominar o consumo de energia para técnicas assíncronas. Em geral, para aplicações com poucos requisitos de latência, abordagens síncronas podem ser mais apropriadas. Foi provado que para uma rede com 10 *hops* o protocolo B-MAC supera o desempenho do S-MAC no que diz respeito à energia com latências abaixo de 6 segundos.[20]

O próximo protocolo é o Z-MAC, que consiste em um protocolo hibrido e combina os pontos fortes do TDMA e CSMA enquanto compensa seus pontos fracos. Assim como o CSMA, o Z-MAC consegue alcançar níveis altos de utilização do canal e baixa latência under low contention e assim como o TDMA, atinge níveis altos de utilização do canal sobre high contention e reduz o número de colisões entre vizinhos de dois hops com um baixo custo. [21] Uma das características do Z-MAC é que seu desempenho é robusto à erros de sincronização, falha na alocação de *slot* e variação de tempo nas condições do canal, sendo que no pior caso, seu desempenho sempre falls back to that of CSMA.

O Z-MAC possui a seguinte ordem para seu funcionamento: descoberta de vizinho, alocação de *slot*, troca local de quadros e sincronização global de tempo.[21] Estas operações ocorrem apenas uma vez durante a fase de setup e não são executadas até que haja uma mudança brusca na topologia da rede. A ideia central consiste de que os custos iniciais para executar essas operações são compensados pela melhora no *throughput* e na eficiência energética durante a transmissão de dados.

Ao utilizar *carrier-sensing* e *congestion backoffs*, Z-MAC se torna mais robusto à erros de clock se comparado ao TDMA. Até mesmo estando sem sincronização, seu desempenho se assemelha ao desempenho do CSMA. Além disso, sob baixa contenção, Z-MAC se assemelha ao CSMA com ou sem sincronização. Então, Z-MAC requer sincronização de relógio sob alta contenção para implementar HCL. Uma funcionalidade importante do Z-MAC é que a sincronização é necessária apenas entre vizinhos remetentes e também quando eles estão sobre alta

contenção. Estes pontos oferecem uma excelente oportunidade para otimizar o *overhead* da sincronização do relógio porque a sincronização é necessária apenas localmente entre vizinhos remetentes e a frequência de sincronização pode ser ajustada de acordo com a taxa de transmissão do remetente para que com isso remetentes com taxas de dados mais altas possam transmitir mensagens de sincronização com maior frequência. Nesse esquema, receptores sincronizam passivamente seus relógios de acordo com o relógio do remetente e não precisam enviar nenhuma mensagem de sincronização.

O C-MAC é um protocolo MAC altamente configurável, concebido como um framework de estratégias de controle de acesso ao meio que pode ser combinado para produzir protocolos específicos à aplicações [22], Ele permite que programadores de aplicações configurem diversos parâmetros de comunicação com o intuito de ajustar o protocolo às suas necessidades específicas.

APÊNDICE C – GPS

Conhecido inicialmente como um sistema de navegação, o Global Positioning System é utilizado também para disseminar o tempo de forma precisa. O GPS transmite sinais gerados por relógios de alta precisão embarcados nos satélites. Esse tempo é utilizado para auxiliar na computação da localização geográfica do receptor.

Vale a pena fazer uma ressalva em relação ao tempo. Tempo e intervalo de tempo são conceitos distintos, sendo, tempo a marcação de um evento com respeito a uma referência de origem. O intervalo de tempo é uma medida de duração. O tempo de um evento é horas, minutos, segundos,enquanto intervalo de tempo pode ser medido pela quantidade de segundos decorridos entre dois eventos subsequentes.

Frequência é a medida do número de eventos que ocorreram em um determinado intervalo de tempo. Assim sendo Universal Time Coordinated (UTC) é um sistema de tempo adotado em diversos países desde 1972. O UTC é coordenado pelo Bureau International des Poids et Mesures (BIPM) na França e é baseado na combinação ponderada dos relógios atômicos espalhados pelo mundo. Ocasionalmente o UTC muda através da adição de saltos de segundos, conhecidos por *leaps*.

O GPS é capaz de disseminar o tempo e a frequência de uma forma global e ubíqua. A precisão da frequência pode ser comparada ao alcançado pelas estações Loran-C, e a precisão do tempo esta no raio de 100 nanosegundos. Alguns receptores GPS possuem um sinal lógico auxiliar chamado de PPS, que basicamente envia um pulso a cada 1 segundo, que indica o início do dado segundo. Este pulso possui uma precisão na faixa de poucos nanosegundos.

Para sincronizar um receptor GPS, o receptor deve encontrar um satélite GPS, ou um Space Vehicle (SV), em uma das duas frequências disponíveis. Intervalos de tempos no receptor podem ser produzidos pela repetição de 1 milhão de vezes do código PRN.

C.1 UTC

O tempo do GPS varia em alguns segundos em relação ao UTC (no ano de 2008 a diferença chegava a 14 segundos) mais uma fração de segundos menor que 1s. A diferença entre o tempo do GPS e o UTC e as características dessa diferença são enviadas pelos satélites (*Subframe* 4). Tempo do Satélite, o *onboard* time para cada satélite, A diferença específica entre o tempo do satélite e o GPS time e suas características são enviadas no *Subframe* 1 da mensagem de navegação. *Receiver Time*, o tempo dentro do receptor de GPS. Este tempo é

determinado através de um oscilador de *quartz* interno e é diferente do GPS time e/ou UTC. A diferença $\boldsymbol{t_0}$ é desconhecida no inicio da operação de um receptor GPS, mas pode ser reduzido após algumas medições.

C.2 DETERMINAÇÃO DO TEMPO DE TRANSMISSÃO

Todo *subframe* da mensagem de navegação começa com um preâmbulo de 8 bits. O preâmbulo na *TLM* é definido pelo seguinte padrão: 10001011. Essa sequência é repetida a cada 6 segundos. O tempo de transmissão (em *Satellite Time*) do preâmbulo é incluído na HOW (Hand Over Word) do *subframe* anterior com os 17-bits da *Time of Week (TOW) Message*.

O receptor GPS realiza uma busca na Navigation Message recebida para o padrão 10001011. Dado que este padrão pode aparecer em outras partes da mensagem de navegação, as condições para os outros parâmetros também precisam ser cumpridas, assim como:

Dois 0's devem aparecer nos bits 51 e 52 após o fim do preâmbulo assumido. A paridade iniciando 16 bits após o preambulo assumido devem estar corretos. Os dois bits antes preambulo assumido devem ser 0. O tempo enviado na mensagem TOW (17 bit) iniciando 22 bits depois do fim do preambulo assumido deve estar aproximadamente correto. Dado que a informação do tempo é repetida a cada 6 segundos, não há nenhuma necessidade de grande precisão para a medida de tempo do receptor.

C.3 GPS TIME E TRANSMISSÃO

A *epoch* GPS inicial é 0000 UT (meia-noite) em 06 de janeiro de 1980. GPS Time não é ajustado e, portanto, é compensado em relação ao *UTC* por um número inteiro de segundos, devido à inserção do *leap second*. O número se mantém constante até o salto do próximo segundo ocorrer. Esse deslocamento também é fornecido na mensagem de navegação (NAV) e o receptor deve aplicar a correção automaticamente. Além dos *leap* seconds, existem correções adicionais dadas na mensagem NAV. O tempo de sistema, por sua vez, é referenciado pelo o relógio mestre (MC) no USNO e guiado para UTC (USNO) a partir do qual o tempo de sistema não irá desviar-se mais do que um microssegundo (requisito PPS). A diferença exata está contida na mensagem

NAV sob a forma de duas constantes, $\boldsymbol{a_0}$ e $\boldsymbol{a_1}$, dando a diferença de tempo e a taxa de tempo do sistema em relação à UTC (USNO, MC). A mensagem de navegação do satélites GPS contém o número de segundos de offset entre o GPS e tempo UTC. Em geral, a inserção de um *leap second* só é decidido cerca de 2 meses antes de ser feito. Apesar de tudo, baseia-se nas anomalias na taxa de rotação da terra, que não é previsível, mas apenas mensurável. Em cada satélite, um derivado de 1,5 *epoch second* fornece uma unidade conveniente para precisamente realizar a contagem e comunicação do tempo. Tempo indicado desta maneira é referido como uma Z-contagem. A Z-contagem é fornecida ao utilizador como um número binário de 29 bits que consiste em duas partes, como se segue: O número binário representado pelos 19 bits menos significativos da Z-contagem é referido como o tempo de semana (TOW) count e é definido como sendo igual ao número de 1,5 segundos que ocorreram desde a transição da semana anterior. Uma versão truncada do TOW, que consiste dos seus 17 bits mais significativos, está contido na palavra *hand-over* (HOW) Os dez bits mais significativos da Z-count é uma representação binária do sequencial número atribuído à semana GPS presente.

O HOW é de 30 bits de comprimento e é a segunda palavra em cada *subframe* / página, imediatamente a seguir à palavra TLM. Um HOW ocorre a cada 6 segundos no quadro de dados. O MSB é transmitido em primeiro lugar. O HOW começa com as 17 MSBs do tempo-de-semana count (TOW). (A contagem total TOW consiste dos 19 LSBs dos 29-bit Z-count). Os outros 10 bits do Z-count dizem respeito ao número da semana desde o *start date*.