Sincronização de Tempo a nível de SO utilizando o protocolo IEEE1588

Peterson Oliveira, Alexandre Massayuki Okazaki e Antônio Augusto Fröhlich,

Laboratório de Integração de Software e Hardware
Universidade Federal de Santa Catarina
Florianópolis, Brasil
{peterson,alexandre,guto}@lisha.ufsc.br

***Abstract*—Time Synchronization is a pertinent field of study to fulfill the OS requirements as 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 a wireless sensor 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.**

***Index Terms*—I.1 [Sistemas Operacionais]: Sistemas Distribuídos**
I.2 [Redes de Computadores] Sincronização de Tempo

I. Introdução

Sincronização de Tempo para dispositivos distribuídos é um campo de estudos desafiante. Levando-se em conta a topologia de como os dispositivos estão distribuídos, para algumas aplicações específicas é necessário que todos os nodos da rede estejam sincronizados para se obter precisão nas medições realizadas. Esta necessidade de manter os relógios sincronizados, com uma certa tolerância, nos leva ao objetivo principal de estudo deste trabalho, permitindo assim a disponibilização de uma gama maior de aplicações.

Dado a necessidade de manter-se um tempo de relógio constante, com um nível de tolerância determinado, durante o tempo de vida da rede. Alguns fatores que podem influenciar na sincronização devem ser levados em consideração: temperatura, ruído de fase, ruído de frequência, atraso assimétrico e falhas no relógio [1]. Considerando esses fatores na elaboração de uma estratégia de sincronização, podemos chegar a três classes de técnicas de sincronização [1]. A primeira técnica se baseia em servidores de tempo fixo para sincronizar a rede, nela os dispositivos são sincronizados à servidores de tempo robustos e extremamente precisos. Na segunda 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-organiza 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 tipo 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), por ser um padrão adotado pela IEEE, pela alta precisão que podemos alcançar e pela sua flexibilidade de configuração e adequação à diferentes topologias de rede.

Temos uma implementação conhecida deste protocolo para o Sistema Operacional Linux, o PTPd. No nosso caso utilizaremos o EPOS, que é um sistema operacional orientado à aplicações para sistemas embarcados [2]. Com o uso de sistema operacionais voltados 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 ou controle. Aplicações em redes de sensores sem fio, similarmente à outros sistemas distribuídos, requerem um serviço de sincronização de tempo escalável. Um exemplo pode ser tomado a partir de aplicações de voz e de vídeo, onde os dados entre nodos sensores podem ser fundidos e exibidos de uma forma significativa na borda da rede. Além disso, alguns protocolos de localização podem tirar vantagem do tempo sincronizado dos sensores.

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 artigo iremos implementar o protocolo PTP para o Sistema Operacional EPOS com o intuito de estudar os aspectos e limitações do protocolo. Após esse trabalho inicial, temos a possibilidade de iniciar a integração do protocolo ao EPOSMote [3], uma plataforma aberta para redes de sensores sem fio, a fim de obter experimentos de sincronismo utilizando nodos sensores reais. Além de nos possibilitar a homologação com outras implementações do mesmo protocolo.

O trabalho foi estruturado em cinco seções, sendo I a Introdução, seguida pelos trabalhos relacionados na seção

II. A seção III conta com o desenvolvimento do protocolo, assim como uma fundamentação sobre o protocolo PTP. O trabalho se encerra com as seções IV e V sendo dedicadas ao experimento e resultados e à conclusão, respectivamente.

II. Trabalhos Relacionados

Para a concepção deste trabalho foram consultados alguns trabalhos relacionados, dentre eles vale destacar o estudo e desenvolvimento realizado por Correll, Barendt e Branicky [4]. Uma implementação *software-only*, chamada de PTPd. A sincronização de tempo com PTP se baseia na troca de mensagens que sincronização, que possuem um *timestamp* associado, entre o mestre e os escravos. Estes *timestamps* de alta precisão podem ser conseguidos com o auxílio de hardware especializado na camada física da rede, no entanto, não foi utilizado nenhum hardware especializado. Implementações *software-only* efetuam o *timestamp* em camadas mais altas da rede, o que acaba por introduzir graus de não-determinismo nas latências, conhecido como *jitter*. Alcançar a sincronização exata entre os dispositivos pertencentes à rede é o principal obstáculo no projeto de implementações deste tipo.

O *daemon* foi desenvolvido para sistemas de Testes e Medições. Para tais sistemas o PTP proporciona sincronização de tempo e frequência. As necessidades destes sistemas influenciaram significativamente o resultado deste trabalho. Por ser uma implementação *software-only* ele não apresenta algumas funcionalidades encontradas em implementações assistidas por hardware. Realizar o registro do *timestamp* em camadas altas das camadas de protocolo, ao invés de realizar tal função próximo à camada física da rede. É utilizado um *software clock*, no entanto, para os experimentos o sistema foi equipado com um relógio de hardware. Isso foi feito para permitir que o relógio pudesse ser lido com *jitter* mínimo.

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

Além do PTPd podemos citar outra grande implementação para sincronizar tempos de relógios largamente utilizado, o NTP, que segundo a definição de Minar o protocolo cria uma rede de hospedeiros na internet que sincronizam o tempo [5]. A topologia do NTP pode ser vista na Figura 1. O NTP é construído sobre o *Internet Protocol* (IP), e no *User Datagram Protocol* (UDP), que provê um mecanismo de transporte sem a necessidade de estabelecer conexões entre os envolvidos [6]. Este foi desenvolvido a partir do protocolo de tempo e da mensagem de *timestamp Internet Control Message Protocol* (ICMP), especialmente para manter a exatidão e a redundância.

No modelo proposto para o NTP não houve nenhum esforço especial 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, sendo que nenhum mecanismo de detecção de dados duplicados ou retransmissão são oferecidos. Dado o fato de apenas um formato de mensagem NTP ser utilizado, este protocolo é facilmente implementado e utilizado nos diversos sistemas operacionais existentes.

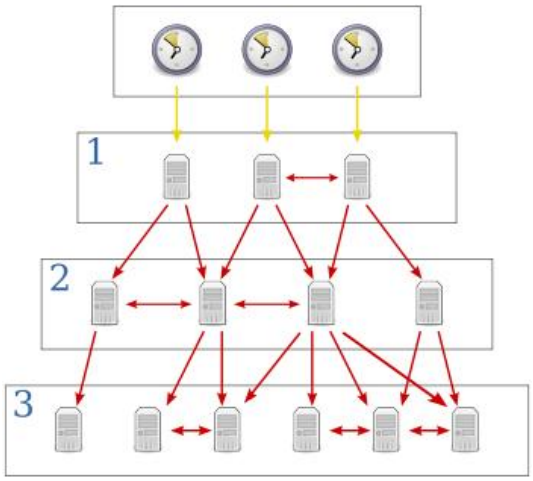

Figure 1. Hierarquia no protocolo NTP

O funcionamento do NTP consiste basicamente na troca de *timestamps* entre os servidores pertencentes a rede, estes *timestamps* são então utilizados para determinar os atrasos bidirecionais individuais, os *offsets* dos relógios e as estimativas de erros [6].

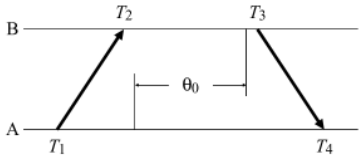

Figure 2. *Timestamps* trocados pelo NTP [6]

De posse destas informações, há a necessidade de uma nova implementação do protocolo PTP, visto que o *daemon* proposto engloba a versão 1 do protocolo, enquanto que na versão 2 houveram melhorias, podendo citar a redução do tamanho da mensagem de sincronização, que facilita a sua implementação para sistemas embarcados. Além disso, o PTP permite obter um nível de precisão maior do que o obtido com o NTP [7].

III. Desenvolvimento

O PTP provê um método padrão para sincronizar dispositivos em uma rede com precisão de sub-micro segundo. Ele 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 sincronia, causas como diferenças na temperatura e frequência podem afetar a qualidade da sincronização.

PTP provê uma sincronização tolerante à falha para diferentes relógios na mesma rede. Algumas vantagens deste protocolo são: pouco consumo de banda e pouco consumo de processamento [8]. Isto é conseguido através do uso do 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 as redes *Ethernet*.

Sua operação inicia-se com o envio de um pacote de sincronização, que 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 sincronização, porém a sincronização também pode ser feita em dois passos. Utilizando dois passos, um pacote *Follow Up* também deve ser enviado pelo mestre. Este pacote conterá um *timestamp*, que possui o tempo exato em que a mensagem de sincronização deixou o mestre, propiciando assim uma maior precisão, já que o *timestamp* poderá ser registrado em camadas mais próximas ao meio físico, o que sofre menos interferências não determinísticas e permite uma maior precisão. A figura 2 mostra as mensagens que são trocadas entre os dispositivos da rede.

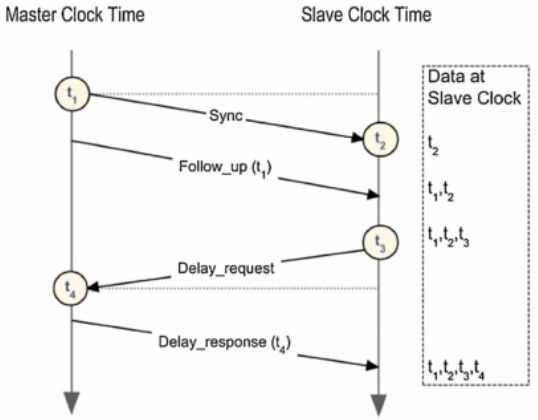

Figure 3. Troca de mensagens efetuadas segundo o protocolo PTP [7]

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 sincronização ao nodo escravo e registra o tempo de envio t_1.
2) O escravo recebe a mensagem de sincronização e registra o tempo de recebimento da mesma t_2.
3) O nodo mestre transporta ao nodo escravo o tempo t_1 registrado através da seguinte forma:
   - Inserindo o *timestamp* t_1 na mensagem de sincronização (um-passo). Este método requer algum processo de hardware para melhor precisão.
   - Inserindo o *timestamp* t_1 em uma mensagem de *Follow Up* (dois-passos).
4) O nodo escravo envia uma mensagem de requisição de atraso para o nodo mestre e registra o tempo de envio t_3.
5) O nodo mestre recebe a mensagem de requisição de atraso e registra
6) O nodo mestre transmite ao nodo escravo o *timestamp* t_4 inserindo-o em uma mensagem de resposta de atraso.

Após realizar estas trocas de mensagens o nodo escravo possuirá quatro *timestamps* e a partir dos mesmos será possível obter o *offset* entre o tempo registrado no nodo mestre em comparação com o tempo registrado no nodo escravo, e assim o atraso da rede, tempo necessário para pacotes trafegarem por pelo link entre dois nodos comunicantes.

O atraso no link pode ser calculado da seguinte maneira :

$$\beta = t_2 - t_1 \tag{1}$$

$$\gamma = t_4 - t_3 \tag{2}$$

O 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.

$$\alpha = \frac{(\beta) + (\gamma)}{2} \tag{3}$$

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

$$\phi = t_2 - (t_1 + \alpha) \tag{4}$$

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

$$\phi = \frac{(\beta) - (\gamma)}{2} \tag{5}$$

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.

$$\rho = \frac{(t_{escravo}) - (t_{mestre})}{t_{mestre}} \tag{6}$$

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

Dado este aspecto teórico, chegou-se a máquina de estados apresentada na Figura 3. Esta foi utilizada na implementação do protocolo PTP utilizando o Sistema Operacional EPOS [2].

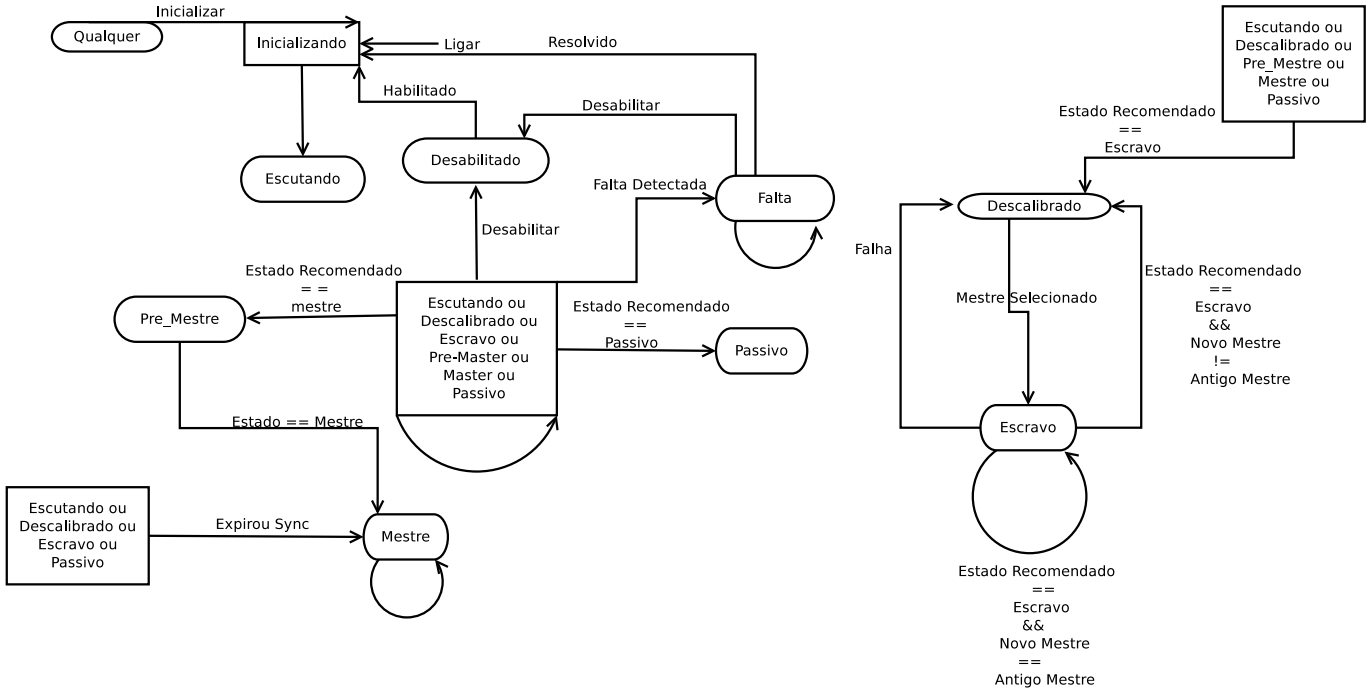

Figure 4. Máquina de Estados do PTP

IV. Resultados

A. Setup do Experimento

Para a realização do experimento foram utilizadas duas configurações diferentes, ambas consistindo de duas máquinas virtuais idênticas para questões de homologação do protocolo. Para a emulação das máquinas foi utilizado o Qemu.

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 *timastamps* representam os envios das mensagens para calcular o atraso da rede.

B. Resultados do Experimento

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 5.

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

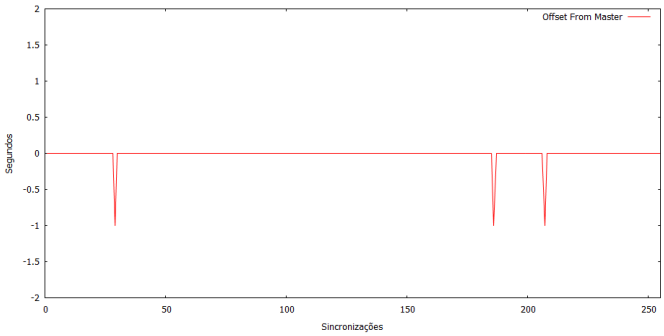

Figure 5. Offset entre dois relógios previamente sincronizados

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 a seguir. Como observado no primeiro experimento obtemos algumas oscilações após se obter a sincronização, ou seja *offset* igual a 0s, que são explicadas da mesma forma.

O experimento foi repetido, seguindo as mesmas configurações, outras cinco vezes e obteve-se os resultados apresentados nas Figuras 7 e 8, incluindo o desvio padrão dos *offsets* calculados e também a média dos *offsets* calculados.

Percebemos que a média se mantêm no intervalo [-0.006, -0.014]. Assim podemos dizer que o relógio se manteve na

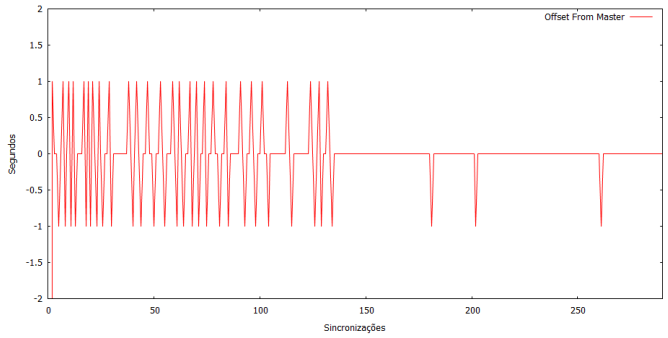

Figure 6. Offset com relógio do escravo atrasado em relação ao relógio do mestre

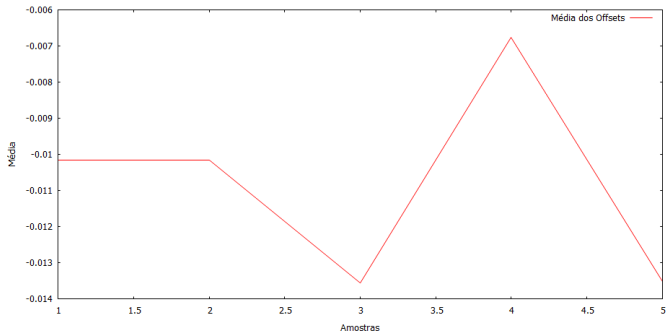

Figure 7. Média do Offset com relógio do escravo atrasado em relação ao relógio do mestre de 5 experimentos

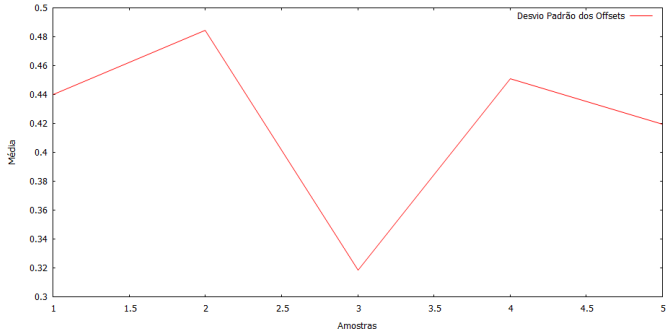

Figure 8. Desvio Padrão do Offset com relógio do escravo atrasado em relação ao relógio do mestre de 5 experimentos

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*.

V. Conclusão

Após realizarmos os testes 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 fornece uma base para trabalharmos a implementação com o intuito de obter um *offset* na faixa de sub-milissegundos. Obtendo essa precisão conseguiríamos garantir a aplicação, por exemplo, da implementação para sistemas de sensoriamento oceânico, como é abordado em [7]. Onde é feita uma abordagem sobre a implementação do protocolo PTP para distribuir os tempos de relógio em uma rede Ethernet de Sensoriamento Marinho(MSN). O fato da necessidade de um protocolo deste tipo para tal escopo se dá pelo fato de sinais GPS não estarem disponíveis devido à atenuação da água no fundo do mar e à requisitos de sincronização de instrumentos marítimos, tais como sismógrafos.

References

[1] B. Raton, L. New, and Y. Washington, *Sensors Handbook*.

[2] A. A. M. Fröhlich, A. Augusto, and M. Fröhlich, "Application-oriented operating systems."

[3] LISHA Team, "EPOSMote," Brazil, 2011. [Online]. Available: http://epos.lisha.ufsc.br

[4] K. Correll, N. Barendt, M. Branicky, and A. Masters, "Design Considerations for Software Only Implementations of the IEEE 1588 Precision Time Protocol," *Design*, vol. 1, pp. 2–7.

[5] N. Minar, "A Survey of the NTP Network 1 The NTP Network," *Methodology*, 1999.

[6] D. L. Mills, "Improved algorithms for synchronizing computer network clocks," *ACM SIGCOMM Computer Communication Review*, vol. 24, no. 4, pp. 317–327, Oct. 1994. [Online]. Available: http://portal.acm.org/citation.cfm?doid=190809.190343

[7] J. del Río, D. Toma, S. Shariat-Panahi, A. Mànuel, and H. G. Ramos, "Precision timing in ocean sensor systems," *Measurement Science and Technology*, vol. 23, no. 2, p. 025801, Feb. 2012. [Online]. Available: http://stacks.iop.org/0957-0233/23/i=2/a=025801?key=crossref.9545ac66c505e0a9e95220f461a7b892

[8] T. Committee, T. Tc, and M. Society, *IEEE Std 1588-2008, IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, 2008, vol. 2008, no. July.