

Sincronização de Tempo a nível de SO utilizando o protocolo IEEE1588

Peterson Oliveira, Alexandre Massayuki Okazaki e Antônio Augusto Fröhlich

Laboratório de Integração Software/Hardware

Universidade Federal de Santa Catarina, UFSC - Brasil

Email: {peterson,alexandre,guto}@lisha.ufsc.br

Resumo—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.

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 certa tolerância, nos leva ao objetivo principal de estudo deste trabalho, permitindo assim a disponibilização de uma gama maior de aplicações.

Dado à 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 os 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-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), que se enquadra no terceiro tipo, por ser um padrão adotado pela IEEE, pela alta precisão que podemos alcançar e pela sua flexibilidade de configuração e adequação a 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 a aplicações para sistemas embarcados [2]. 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 sensoramento no processo de aquisição de dados e controle. Aplicações em redes de sensores sem fio, similarmente a outros sistemas distribuídos, requerem um serviço de sincronização de tempo escalável. Um exemplo disso são alguns protocolos de localização que podem tirar vantagem do tempo sincronizado dos sensores para melhorar sua precisão.

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 de sincronização, que possuem um *timestamp* associado, entre o mestre e os escravos. 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 PTPd se destina a 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 66MHz 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 implementação para sincronizar tempos de relógios largamente utilizados, 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.

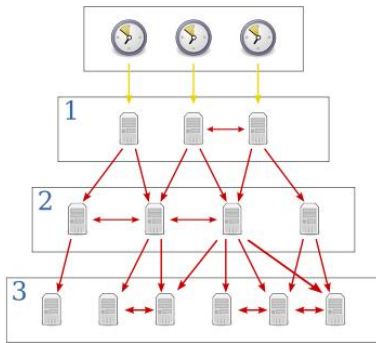


Figura 1. Hierarquia no protocolo NTP

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].

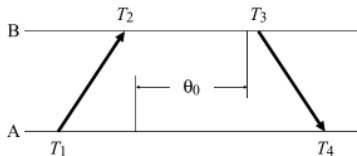


Figura 2. *Timestamps* trocados pelo NTP [6]

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 à 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 a falhas para diferentes relógios na mesma rede. Algumas vantagens deste protocolo são: pouco consumo de banda e pouco consumo de processamento [8]. 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*.

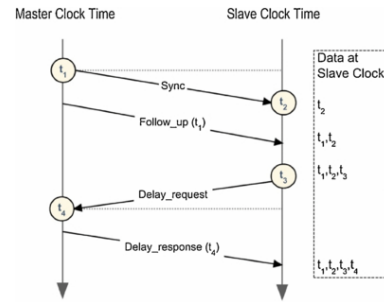


Figura 3. Troca de mensagens efetuadas segundo o protocolo PTP [7]

A figura 2 mostra as mensagens que são trocadas entre os dispositivos da rede. 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 :

$$\beta = t_2 - t_1 \quad (1)$$

$$\gamma = t_4 - t_3 \quad (2)$$

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.

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

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

$$\phi = t_2 - (t_1 + \alpha) \quad (4)$$

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

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

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.

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

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, chega-se à máquina de estados apresentada na Figura 3. Esta foi utilizada na implementação do protocolo PTP utilizando o Sistema Operacional EPOS [2].

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 *timestamps* 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.

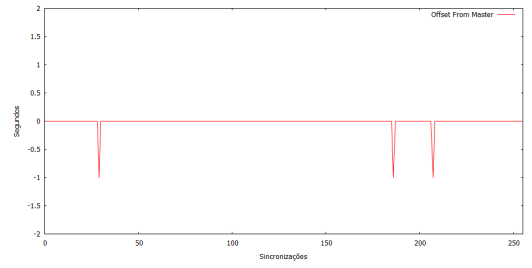


Figura 5. Offset entre dois relógios previamente sincronizados

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 sincronizados 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.

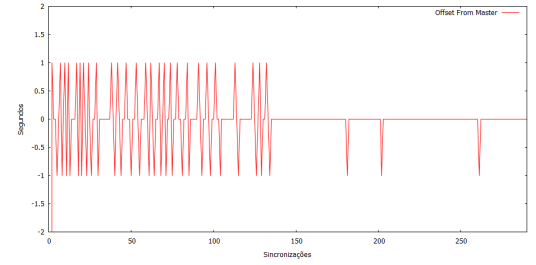


Figura 6. Offset com relógio do escravo atrasado em relação ao relógio do mestre

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

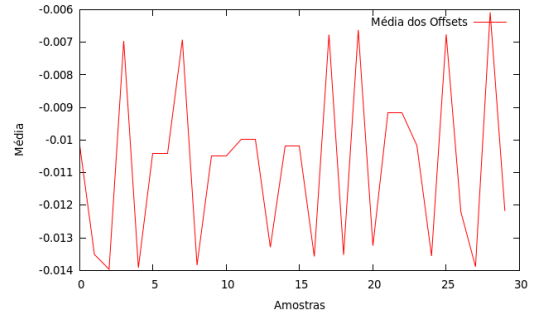


Figura 7. Média do Offset com relógio do escravo atrasado em relação ao relógio do mestre após 30 experimentos

Percebemos que a média se mantém no intervalo [-0,006 ; -0,014]. Assim podemos dizer que o relógio se manteve na

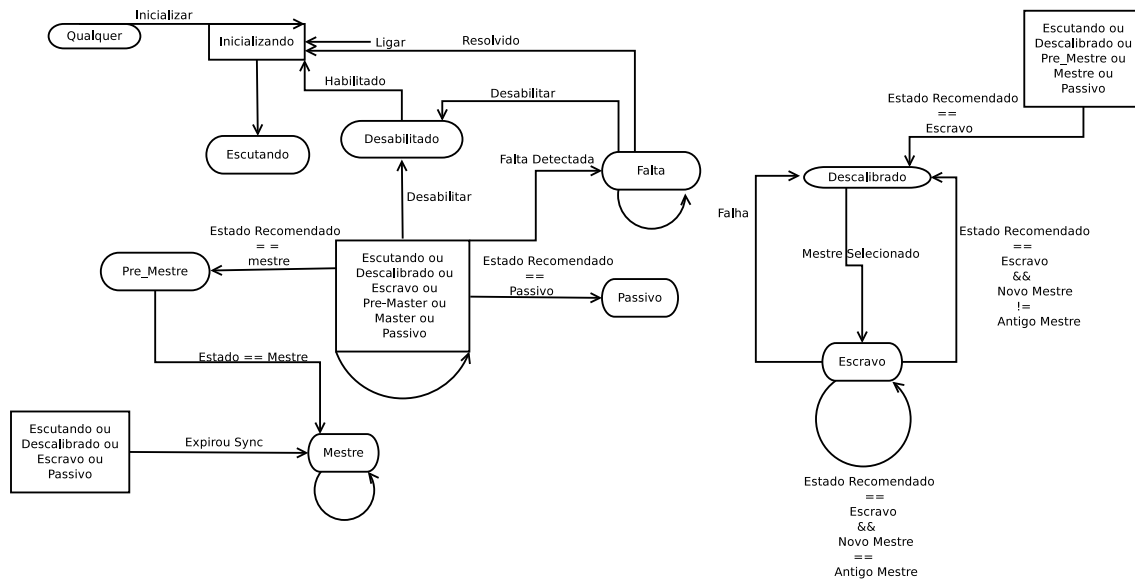


Figura 4. Máquina de Estados do PTP

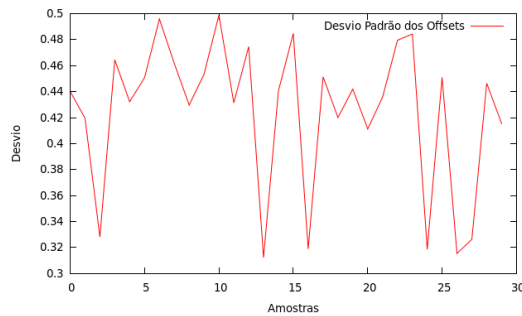


Figura 8. Desvio Padrão do Offset com o relógio do escravo atrasado em relação ao relógio do mestre após 30 experimentos

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 e implantarmos a solução em redes de sensoriamento sem fio. 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.

REFERÊNCIAS

- [1] B. Raton, L. New, and Y. Washington, *Sensors Handbook*.
- [2] A. A. M. Fröhlich, "Application-oriented operating systems."
- [3] LISHA Team, "EPOS Mote," 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. Mânuel, 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.