User-level Implementation of Precision Time Protocol

Francisco Terra
MIEEC
FEUP
Porto, Portugal
up201604468@fe.up.pt

Mário Cardoso
MIEEC
FEUP
Porto, Portugal
up201303424@fe.up.pt

Pedro Castro
MIEEC
FEUP
Porto, Portugal
up201605469@fe.up.pt

***Resumo*—In distributed systems, clock synchronization is a crucial aspect for time-sensitive applications. This work implements an user-level implementation of both the PTP and the NTP clock synchronization protocols and compares their performance for a set of parameters. For the results, the program simulates adverse network conditions by implementing configurable delays when passing messages while locally simulating several devices using only a single computer.**

***Palavras-chave*—Clock, Network, Protocol, PTP, NTP, Synchronization, Precision, Accuracy**

I. Definição do Problema

No contexto de um sistema distribuído, cada nodo do sistema tem o seu próprio relógio, independente dos restantes. Tipicamente estes relógios têm a sua implementação baseada em circuitos de osciladores de cristal, por vezes imprecisos, instáveis e sensíveis a variações de temperatura [1]. Isto faz com que não seja garantido que dois cristais semelhantes oscilem a frequências idênticas. Devido ao efeito de deriva (*drift*) do relógio, os relógios presentes no sistema divergem no tempo devido a progredirem a velocidades diferentes. Este efeito leva à dessincronização de um relógio que possa ter sido previamente sincronizado. É também possível assumir que os relógios dos vários dispositivos de um sistema distribuído não possuem, à partida, uma noção global coerente de tempo, podendo estar a operar com uma dada diferença temporal, ou *offset*, entre si.

Para contextualizar o problema, a questão da sincronização de relógios resulta da necessidade de existir uma noção global de tempo para que seja possível realizar certas operações como as que envolvem o agendamento temporal de certas acções, o estabelecimento de uma relação temporal entre eventos, a avaliação de idade de dados ou a sincronização para acesso a recursos partilhados. A obtenção de coerência temporal entre os relógios da rede pode ser obtida recorrendo a algoritmos de sincronização capazes de obter factores de compensação para erros de deriva sistemáticos e de erros de *offset*. Esses algoritmos são aplicados em protocolos de sincronização de relógios tais como os mencionados *Precision Time Protocol* e *Network Time Protocol*.

A. Precision Time Protocol (PTP)

PTP é um protocolo de sincronização de relógios de elevada precisão que tem por base o standard IEEE 1588 [2].

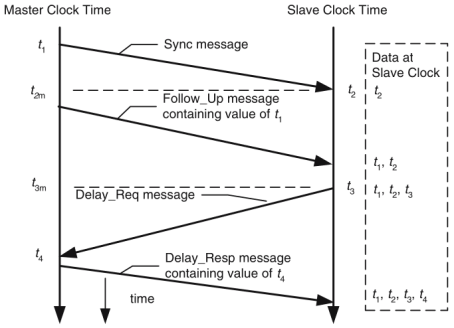

Fig. 1. Diagrama temporal das mensagens de sincronização PTP

Hierarquicamente faz uma divisão entre dispositivos *master* e *slave*, em que o *master* é o relógio fonte e os *slaves* obtém o seu tempo através de uma troca de mensagens com o mesmo. A sequência de troca de mensagens começa quando o relógio *master* transmite uma mensagem de *Sync* aos *slaves*, registando o tempo de envio *t1*. O *slave* recebe e regista o tempo de recepção, *t2*. Numa mensagem *Follow_Up* o *master* envia o tempo *t1* caso este tenha sido obtido imediatamente após o envio da mensagem de *Sync*. O *slave* envia uma mensagem *Delay_Req* ao *master* e regista o instante de envio *t3*. O *master* recebe e regista o tempo de recepção *t4*. Por último, numa mensagem *Delay_Resp*, o *master* comunica ao *slave* o instante *t4* [3].

B. Network Time Protocol (NTP)

O NTP é um protocolo de sincronização de relógios de popular utilização, onde há uma divisão entre dispositivos cliente e servidor. Tem uma estrutura hierárquica onde a informação temporal é passada partindo do relógio de referência [4]. Foi inicialmente documentado pelo IETF no RFC958 [5].

A troca de mensagens, representada na figura 2, inicia quando o cliente faz o pedido inicial e regista *t1*, a *Originate timestamp*. O servidor regista *t2*, a *Received timestamp* na recepção e *t3*, a *Transmit timestamp* do envio do servidor. Por último o cliente regista *t4*, a *Reference timestamp* a quando da recepção da mensagem [5].

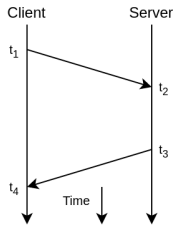

Fig. 2. Diagrama temporal das mensagens de sincronização NTP

II. Descrição da Solução

A. Relógio simulado

A classe *Clock* permite emular o comportamento de um relógio. No momento da sua criação é possível definir o *offset* e o valor da sua derivada. Este relógio possui ainda um parâmetro que é o fator de correcção que é atualizado em cada período de sincronização, o que permitirá alterar a velocidade do relógio. O tempo real é convertido na base de tempo do relógio simulado multiplicando o intervalo de tempo que passou entre invocações do método *GetTime()*, pela derivada atual do relógio (derivada*fator de correção).

B. PTP

O programa do *master* está dividido em duas *threads*, uma que irá enviar as mensagens de *Sync* e respetiva mensagem de *Follow_Up* periodicamente e outra que é responsável por responder aos pedidos de *delay* dos vários *slaves*. O programa do *slave* é um único processo ciclo sequencial que após a receção de todas as *timestamps* volta ao inicio ficando à espera da próxima mensagem de *Sync*. Independentemente do procedimento de sincronização, existe também uma *thread* a correr em cada processo que é responsável pela criação e escrita dos ficheiros de *log* que serão usados posteriormente para analisar os resultados. Estes *logs* registam o tempo real no momento da escrita e também o valor do relógio simulado.

O *slave* após a receção da mensagem de resposta ao pedido de *Delay*, tem todos os *timestamps* necessários para o cálculo do erro de *offset* e do atraso da rede, através das fórmulas (eq. 1) e (eq. 2);

Com estes dois erros e sabendo o intervalo de sincronização, é possível calcular a derivada que o relógio simulado deveria ter de forma a convergir para o valor do relógio do mestre ((eq. 3). O fator de correção foi calculado de forma a que multiplicado pela derivada do relógio seja possível obter o valor de ρ ((eq. 4) .

$$delay = \frac{(t_2 - t_1) + (t_4 - t_3)}{2} \tag{1}$$

$$offset = \frac{(t_2 - t_1) - (t_4 - t_3)}{2} \tag{2}$$

$$\rho = 1 - \frac{(offset + delay)}{t_{Sync}} \tag{3}$$

$$FC = \frac{\rho}{clock.derivada} \tag{4}$$

C. NTP

A implementação do NTP é mais simples do que a do PTP, existe apenas um pedido e uma resposta trocado entre o cliente e o servidor. Por este motivo, para a implementação do processo de sincronização, foi utilizado apenas uma *thread* em cada um. O processo de cálculo do fator de correção é semelhante ao apresentado previamente.

D. Simulação do atraso de rede

Como os testes foram todos realizados localmente foi necessário criar uma forma de adicionar o efeito dos possíveis atrasos causados pela rede. Este atraso é gerado aquando da recessão de um pacote. Quando um pacote é recebido é guardado o *timestamp* desse instante, posteriormente a *thread* entra em *sleep* durante um valor aleatório que é gerado tendo por base um limite máximo e mínimo ajustáveis. Quando a *thread* acorda, soma ao *timestamp* da recessão este tempo de espera, ajustado à base temporal do relógio simulado.

III. Resultados

Como se referiu, a rede foi simulada para o propósito dos testes conseguirem correr no mesmo computador e para ter melhor controlo em alguns dos seus parâmetros.

A. Parâmetros usados

Para o efeito de testes foram usados os parâmetros de referência descritos na tabela I para os processos dos protocolos. Estes serão alterados para diferentes casos por forma de ver o impacto na precisão e na exatidão dos relógios dos *slaves* e dos clientes.

TABELA I
PARÂMETROS PADRÃO PARA OS TESTES

Tempo de sincronização 5 s	Número de processos 4
Delay min 0.05 s	Delay max 0.2 s
Offset 0 s	Drift Rate 2

O *offset* foi considerado pouco relevante para as experiências, visto que já existe um atraso na inicialização de todos os processos. Para além disso, este parâmetro só tem influência no início da execução e será corrigido em poucas transações dos protocolos.

Outro parâmetro que foi pouco variado foi o *Drift Rate*, pois este também vai ter um impacto no início do programa e não ao longo da execução do mesmo.

B. Configuração dos parâmetros

A variação dos parâmetros para as experiências foi maioritariamente feita a nível individual, à exceção dos atrasos mínimo e máximo que atuam diretamente no *jitter*. A nomenclatura que será usada mais tarde para a apresentação dos resultados é a seguinte:

- **Default** → Mostrado anteriormente

- **Drift_Rate[0.5]** → *Drift rate* alterado para o inverso do mesmo parâmetro do **Default**
- **Delay[x; y]** → Com x igual ao *Delay* mínimo e y ao *Delay* máximo (ambos em segundos)
- **Clocks[n]** → Sendo n o número de *slaves* (ou clientes)
- **Sync_T[T]** → Com T segundos de tempo de sincronização

C. Cálculo da exatidão e da precisão

Foi considerado como referência o relógio do mestre/servidor. Ou seja, t é o relógio destes dispositivos. Tendo isto em conta, os cálculos para a exatidão (eq. 5) e precisão (eq. 6) são dados pelas seguintes expressões.

$$\alpha = max\,|Clock_{slave\_i}(t) - t| \tag{5}$$

$$\delta = max\,|Clock_{slave\_i}(t) - Clock_{slave\_j}(t)| \tag{6}$$

O i e o j nas equações representam *slaves* (ou clientes) arbitrários da experiência em causa. Os cálculos consideram então a maior diferença de *Clocks* nas várias amostras de tempo.

D. Resultados das experiências

Os resultados obtidos (Tabela II), tal como algumas figuras das simulações, encontram-se nos anexos. O $\tilde{\alpha}$ e o $\tilde{\delta}$ são os valores médios da exatidão e da precisão respetivamente. Os símbolos com $_M$ são resultados máximos para esse tipo de medição. Para maior resolução, os resultados são mostrados em milissegundos (a virgula é apenas um delimitador dos milhares).

E. Observações

Ao observar os resultados do PTP, é possível tirar algumas conclusões, não só em relação ao protocolo, mas também em relação à implementação. Isto também se aplica para o NTP.

Comparando apenas os parâmetros de referência, conseguimos destacar de imediato que o PTP apresenta melhores resultados. Isto acontece pelos seguintes motivos:

1) O PTP tem mais troca de informação entre os participantes, criando uma melhor exatidão de relógios
2) A implementação do cliente/servidor do NTP faz com que os clientes façam os pedidos de forma dessincronizada entre eles, piorando assim a precisão
3) Os atrasos de rede impostos nas *threads* na implementação do NTP cria congestão dos pedidos dos clientes

Estas justificações também podem ser selecionadas para provar as conclusões a seguir expostas.

Vendo os ensaios de **Delay[x; y]**, não só é possível concluir que o PTP é sensível aos valores de atraso máximo e mínimo, como também ao *jitter*. Isto porque todos estes tipos de teste apresentam piorias nos resultados, para qualquer aumento do atraso mínimo, máximo ou do *jitter*.

Para o caso do **Drift_Rate[0.5]** e as experiências **Sync_T[T]**, estes apresentam por volta da mesma ordem de grandeza dos resultados **Default**, logo, podem ser considerados pouco impactantes no desempenho do algoritmo.

O caso dos 24 *slaves* é inconclusivo, visto que a precisão e a exatidão deveriam piorar como se observa no **Clocks[16]**. Continuando com este último, é relevante apontar a linha *exatidão1* na Figura 3. Isto indica uma imperfeição na implementação do PTP visto que a exatidão dos relógios é degradada com o aumento do número dos *slaves*. Os *slaves* são atendidos por ordem, causando atrasos na resposta dos *slaves* seguintes devido à simulação do atraso de rede ser imposta por desligar o processo do mestre. Logo, irá haver pouca consistência dos clocks, sendo os primeiros *slaves* atendidos, os com melhores resultados.

No NTP, existe em todas as experiências uma tendência ascendente da precisão dos relógios mesmo que a exatidão esteja estável (observável na Fig 4). Isto acontece por causa da dessincronização entre os pedidos dos clientes. Assim, mesmo que a exatidão esteja dentro do esperado, a precisão irá aumentar devido aos atrasos de rede e à espera de mensagens com tempos arbitrários. Apesar de não observável no gráfico, esta tendência para a precisão irá ser limitada pela exatidão máxima, não continuando a crescer indefinidamente.

Tal como o PTP, o NTP aparenta ser sensível ao *jitter* e aos atrasos mínimos e máximos e as experiências **Clock[n]** apresentam piores resultados com o aumento dos clientes, como seria de esperar.

Por último, os ensaios **Sync_T[T]** até conseguem melhorar os resultados, isto porque com tempos de sincronização maiores, os pedidos ao servidor vão estar mais separados entre si, havendo menos atrasos por colisões de pedidos.

IV. Conclusão

Em geral, é possível concluir que o protocolo PTP apresenta uma maior robustez em relação ao NTP, mesmo que as implementação usadas para o provar tivessem espaço para melhoria. No entanto, é de realçar também que o PTP apresenta algumas desvantagens.

Em primeiro lugar, implica a utilização de um *master* na rede que poderá causar congestionamento a esta com pacotes de sincronização. No caso do NTP, cada cliente gere quando é que necessita de nova informação, podendo limitar o tráfego, mesmo que com a perda de *performance*.

Por outro lado, o PTP tem que ter um dispositivo dedicado para a tarefa. A falha do mesmo pode implicar uma falha catastrófica do sistema. Em NTP, a falha de um servidor pode ser condicionada pelo pedido a outros servidores alternativos.

A carga de trabalho foi, em geral, foi bem distribuída entre os membros da equipa.

- Francisco Terra: 33%
- Mário Cardoso: 34%
- Pedro Castro: 33%

ANEXOS

TABELA II
RESULTADOS DAS EXPERIÊNCIAS

Protocolo		Default	Clocks[8]	Clocks[16]	Clocks[24]
PTP	α_M	382	374	1,311	618
	$\tilde{\alpha}$	154	237	1,145	410
	δ_M	299	384	1,214	597
	$\tilde{\delta}$	103	260	1,086	488
NTP	α_M	1,328	1,614	3,845	5,394
	$\tilde{\alpha}$	497	1,214	2,775	4,768
	δ_M	1,365	2,032	5,012	6,366
	$\tilde{\delta}$	909	1628	3,764	5,646

Protocolo		Delay[0.1; 0.2]	Delay[0.2; 0.6]	Delay[0.4; 0.6]
PTP	α_M	532	1,084	1,609
	$\tilde{\alpha}$	138	441	676
	δ_M	409	792	1,163
	$\tilde{\delta}$	168	348	542
NTP	α_M	2,016	4,187	5,440
	$\tilde{\alpha}$	435	1,142	1,444
	δ_M	1,319	5,719	6,196
	$\tilde{\delta}$	918	3,591	4,323

Protocolo		Drift_Rate[0.5]	Sync_T[10]	Sync_T[20]
PTP	α_M	366	248	298
	$\tilde{\alpha}$	168	150	132
	δ_M	255	203	224
	$\tilde{\delta}$	90	92	101
NTP	α_M	862	1,591	2,044
	$\tilde{\alpha}$	249	419	304
	δ_M	1,089	1,413	1,119
	$\tilde{\delta}$	721	911	648

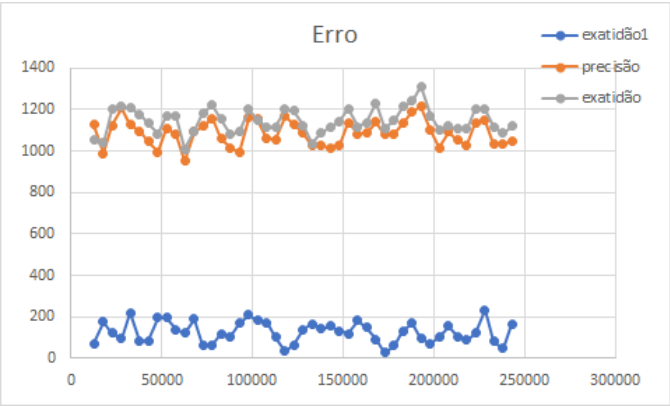

Fig. 3. Gráfico com a exatidão e a precisão da experiência **Clocks[16]** do PTP em *ms*. A linha exatidão1 representa a exatidão apenas do primeiro *slave*.

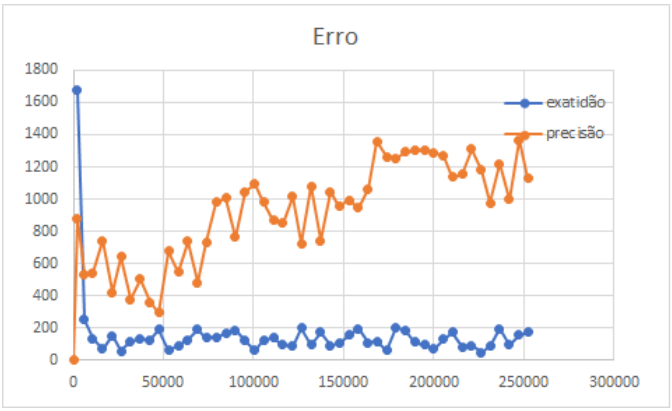

Fig. 4. Gráfico com a exatidão e a precisão da experiência **Default** do NTP em *ms*.

REFERÊNCIAS

[1] Guo, Feng, Mei Song, Yifei Wei, Luigi Sambolino, Pengcheng Liu, Xiaojun Wang, and Martin Collier. "Green Precision Time Protocol Router Using Dynamic Frequency Scaling." In Human Centered Computing, edited by Qiaohong Zu and Bo Hu, 104–15. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016.

[2] "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems." IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002), July 2008, 1–300.

[3] Eidson, John C., ed. "An Overview of Clock Synchronization Using IEEE 1588." In Measurement, Control, and Communication Using IEEE 1588, 35–58. Advances in Industrial Control. London: Springer, 2006.

[4] Idrees, Z., J. Granados, Y. Sun, S. Latif, L. Gong, Z. Zou, and L. Zheng. "IEEE 1588 for Clock Synchronization in Industrial IoT and Related Applications: A Review on Contributing Technologies, Protocols and Enhancement Methodologies." IEEE Access 8 (2020): 155660–78.

[5] RFC958 Network Time Protocol (NTP). D.L. Mills. September 1985.