UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

LUCAS RICARDO MARQUES DE SOUZA

ANÁLISE DE DESEMPENHO DE PROTOCOLOS DE SINCRONIZAÇÃO DE RELÓGIOS FÍSICOS

CURITIBA

2022

LUCAS RICARDO MARQUES DE SOUZA

ANÁLISE DE DESEMPENHO DE PROTOCOLOS DE SINCRONIZAÇÃO DE RELÓGIOS FÍSICOS

PERFORMANCE ANALYSIS OF PHYSICAL CLOCK SYNCHRONIZATION PROTOCOLS

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia da Computação do Curso de Bacharelado em Engenharia da Computação da Universidade Tecnológica Federal do Paraná.

Orientador: Profª. Dra. Ana Cristina Barreiras Kochem Vendramin

Coorientador: Profª. Dra. Keiko Veronica Ono Fonseca

CURITIBA
2022

LUCAS RICARDO MARQUES DE SOUZA

ANÁLISE DE DESEMPENHO DE PROTOCOLOS DE SINCRONIZAÇÃO DE RELÓGIOS FÍSICOS

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia da Computação do Curso de Bacharelado em Engenharia da Computação da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 26/maio/2022

Ana Cristina Barreiras Kochem Vendramin
Doutorado em Engenharia Elétrica e Informática Industrial
Universidade Tecnológica Federal do Paraná

Keiko Veronica Ono Fonseca
Doutorado em Engenharia Elétrica
Universidade Tecnológica Federal do Paraná

Luiz Nacamura Júnior
Doutorado em Engenharia Elétrica
Universidade Tecnológica Federal do Paraná

Mauro Sergio Pereira Fonseca
Doutorado em Redes de Informática
Universidade Tecnológica Federal do Paraná

CURITIBA
2022

AGRADECIMENTOS

Meus primeiros agradecimentos aos meus pais, que me incentivaram nos momentos difíceis e entenderam minha ausência enquanto me dedicava à graduação. Agradeço por todo o amor, atenção e carinho oferecidos durante toda a jornada.

Agradeço à minha esposa que acompanhou toda a saga de desenvolvimento deste projeto. Sou grato pelo apoio, carinho e paciência comigo durante este desafio.

Agradeço à orientadora Profª. Dra. Ana Cristina Barreiras Kochem Vendramin, pela paciência, sabedoria e considerações feitas ao longo do trabalho. Um motivo de inspiração no âmbito pessoal e profissional.

Agradeço à coorientadora Profª. Dra. Keiko Veronica Ono Fonseca, pelas ideias e comentários que aprimoraram a qualidade deste trabalho.

Agradeço aos meus colegas do curso de engenharia da computação. Sou grato pelo tempo que passei com vocês e com certeza os levarei na memória e no coração para toda a vida.

Agradeço, por fim, a todos os que por algum motivo contribuíram para a realização desta pesquisa.

RESUMO

O tempo sempre foi uma grandeza física crucial no estudo da engenharia e isso não é diferente em sistemas distribuídos síncronos, onde cada estação deve estar sincronizada temporalmente no sistema. Uma lista de protocolos está sendo desenvolvida por uma equipe de engenheiros do *Institute of Electrical and Electronics Engineers* (IEEE) para garantir os requisitos em qualquer sistema sensível ao tempo baseado em relógios físicos. O intuito deste trabalho é analisar dois desses protocolos: o IEEE 1588-2008 - *Precision Time Protocol* (PTP) e o IEEE 802.1AS - *Generic Precision Time Protocol* (gPTP). A análise de desempenho desses protocolos é realizada no simulador de código aberto *Objective Modular Network Testbed in C++* (OMNeT++). Dois cenários de simulação são considerados: o primeiro considera uma variação no tráfego de mensagens na rede; e o segundo considera diferentes implementações de derivas em relógios físicos. A análise identificou o comportamento dos protocolos no ambiente de simulação, visando auxiliar na escolha entre eles em futuros projetos.

Palavras-chave: Sincronização de relógios físicos; Análise de desempenho; PTP; gPTP.

ABSTRACT

Time has always been a crucial physical quantity in the study of engineering and this is no different in synchronous distributed systems, where each station must be temporally synchronized in the system. A list of protocols are under development by a team of engineers at the Institute of Electrical and Electronics Engineers (IEEE) in order to guarantee the requirements in any time sensitive system based on physical clocks. The purpose of this work is to analyze the performance of two of these protocols: the IEEE 1588-2008 - Precision Time Protocol (PTP) and the IEEE 802.1AS - Generic Precision Time Protocol (gPTP). The performance analysis of these protocols is performed in the open source simulator Objective Modular Network Testbed in C++ (OMNeT++). Two simulation scenarios are considered: the first one considers a variation in message traffic on the network; and the second one considers different implementations of drift in physical clocks. The analysis identified the behavior of protocols in the simulation environment, aiming to assist in choosing between them in future projects.

Keywords: Physical clock synchronization; Performance analysis; Precision Time Protocol; generic Precision Time Protocol.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ACRÔNIMOS

Siglas

AVB	Audio Video Bridging
AWGN	Additive White Gaussian Noise
BMCA	Best Master Clock Algorithm
E2E	End-to-End
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
gPTP	Generic Precision Time Protocol
LTS	Long-Term Support
NTP	Network Time Protocol
P2P	Peer-to-Peer
PTP	Precision Time Protocol
RTT	Round-Trip Time
TSN	Time Sensitive Networking
UTC	Universal Time Coordinated

Acrônimos

ANF	Analysis File
IEEE	Instituto de Engenheiros Eletricistas e Eletrônicos
ISO	Organização Internacional para Padronização
NED	Network Descriptor
OMNet++	Objective Modular Network Testbed in C++
RAM	Random Access Memory

SUMÁRIO

1 INTRODUÇÃO

Um relógio físico é composto por um circuito eletrônico que irá gerar pulsos elétricos usados para definir a taxa de execução de tarefas (COULOURIS; DOLLIMORE; KINDBERG, 2005). Essa geração de pulsos é periódica e os períodos são conhecidos como "ciclos do relógio". Um conjunto de ciclos em uma determinada faixa de tempo é conhecida como "frequência". Com essas definições, um computador consegue assimilar a grandeza física "tempo" e a utiliza como base fundamental para realizar as operações de um sistema.

Apesar do relógio físico ser o dispositivo fundamental para um sistema computacional, ele não está livre de incertezas, ou seja, variações em seus sinais de saída. Como visto em (COULOURIS; DOLLIMORE; KINDBERG, 2005), a incerteza intrínseca na frequência gerada por relógios físicos gera o fenômeno da "deriva". No decorrer do tempo, uma pequena e constante mudança pode gerar um distúrbio na contagem de ciclos, fazendo com que o relógio físico comece a identificar eventos com valores irregulares. Segundo (COULOURIS; DOLLIMORE; KINDBERG, 2005), sistemas que usam cristais de quartzo como gerador de frequência defasam cerca de 1 segundo a cada 12 dias. Quando se usa dispositivos de "alta precisão", a diferença mínima é em torno de 1 segundo a cada 116 dias. Tal fator é ainda mais preocupante quando se consideram sistemas distribuídos, pois como visto em (COULOURIS; DOLLIMORE; KINDBERG, 2005), surge o fenômeno da "inclinação": pode haver diferença entre relógios de uma rede por divergência entre os componentes individuais.

Diante desse cenário, surgem os protocolos de sincronização de relógios físicos. Em meados de 1980, surgiu o *Network Time Protocol* (NTP). Esse protocolo se baseia na arquitetura cliente-servidor e é um dos mais difundidos atualmente. No entanto, o NTP possui uma precisão de milissegundos, como visto em (MILLS, 2006). O NTP não atendia os requisitos exigidos por algumas redes de controle e medição e, por esse motivo, originou-se o *IEEE 1588 - Precision Time Protocol* (PTP), em 2002. Após uma série de revisões, sua versão definitiva surgiu como *IEEE 1588-2008 - Precision Time Protocol* (PTP) (IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002), 2008). O PTP de 2008, que pode ser conhecido também como PTPv2, se baseia na arquitetura mestre-escravo e alcança submicrossegundos de precisão. Em 2011, uma atualização do PTPv2 deu origem ao *Generic Precision Time Protocol* (gPTP) (IEEE Std 802.1AS-2011, 2011). Diferente de seu predecessor, o gPTP busca aprimorar a precisão na sincronização de relógios físicos, a qual pode chegar à nanossegundos, mas para isso demanda alterações de *hardware* nos dispositivos da rede.

Este trabalho tem como objetivo analisar o desempenho dos protocolos PTP e gPTP no simulador OMNet++ (OMNET++, 2022). Para isso, suas semelhanças e diferenças são apresentadas no decorrer do trabalho, bem como seus funcionamentos. São estabelecidos diferentes cenários para simular o funcionamento desses protocolos. Considera-se uma variação de tráfego na rede e um distúrbio nos relógios de cada componente. A análise irá avaliar o desempenho na sincronização dos dispositivos da rede através de métricas utilizadas na

própria comunicação, como o valor médio do atraso entre os dispositivos. Além disso, busca-se incrementar a implementação de simulações de derivas de relógios em cada uma das bibliotecas dos protocolos, contribuindo com cada um dos projetos.

1.1 Objetivos

O objetivo geral do trabalho é analisar o desempenho dos protocolos PTP e gPTP no simulador OMNet++. Para isso, são definidos os seguintes objetivos específicos:

- Analisar o desempenho de precisão dos protocolos PTP e gPTP sob funcionamento no simulador OMNet++;
- Analisar o desempenho de precisão dos protocolos PTP e gPTP sob variação de tráfego na rede;
- Analisar o desempenho de precisão dos protocolos PTP e gPTP sob condições de variação de tempo nos relógios internos de cada dispositivo;
- Contribuir com a análise da deriva de relógios físicos em simuladores.

1.2 Estrutura do documento

Este documento está organizado em 6 capítulos. O presente capítulo resume a temática do trabalho, bem como os seus objetivos. O Capítulo 2 detalha o funcionamento dos protocolos PTP e gPTP, com definições relevantes para o trabalho, e apresenta uma seção sobre os artigos relacionados ao tema deste trabalho. O Capítulo 3 apresenta a arquitetura geral da rede que será usada como base nas arquiteturas específicas de cada biblioteca. O capítulo também descreve os parâmetros utilizados para configurar as simulações, mostra as configurações do computador utilizado, as versões do simulador e bibliotecas. O Capítulo 4 descreve como os testes são realizados, considerando preparativos, procedimento e avaliação. O Capítulo 5 conta com os resultados das simulações, apresentando os gráficos obtidos e diferenças observadas. Encerra-se o trabalho com o Capítulo 6, no qual apresentam-se as percepções e análises realizadas. O capítulo também considera as contribuições geradas para a comunidade e futuras melhorias.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo sintetiza o material de pesquisa utilizado para a elaboração deste trabalho. Na Seção 2.1 é apresentada uma visão geral sobre protocolos de sincronização de relógios físicos. A seguir, na Seção 2.2 é definido um dos conceitos fundamentais na abstração do conceito de tempo em sistemas computacionais. Na Seção 2.3 é apresentada uma visão geral sobre os protocolos, com foco no funcionamento semelhante do PTP e gPTP. As diferenças, que são essenciais no desenvolvimento deste projeto, também são apresentadas e discutidas na Seção 2.4. A Seção 2.5 apresenta o conceito de deriva de relógios físicos e como essa deriva pode ser simulada. Por fim, a Seção 2.6 mostra os trabalhos relacionados.

2.1 Protocolos para sincronização de relógios físicos

Os avanços tecnológicos demandam sistemas de maior complexidade capazes de processar mais rapidamente as informações. Esses sistemas, muitas vezes, se tornam distribuídos, ou seja, distribuem as tarefas entre componentes de *software* e *hardware* conectados entre si em uma rede e que se comunicam através de troca de mensagens, como visto em (COULOURIS; DOLLIMORE; KINDBERG, 2005). Com isso, surgem novas dificuldades, dentre elas o problema de sincronizar o tempo entre os componentes.

A fim de solucionar a sincronização em uma rede, surge o *Network Time Protocol* (NTP) (MILLS, 2006). Este protocolo foi usado pela primeira vez em 1980 e contava com um sistema de sincronismo baseado em uma arquitetura cliente-servidor dividida em segmentos denominados "stratum". O servidor, ou "stratum 0", representava o próprio "tempo primordial", ou seja, o valor de tempo mais confiável. Os níveis seguintes acessavam este valor por meio de requisições às camadas imediatamente superiores. Por exemplo, o "stratum 1" acessava diretamente o servidor, o "stratum 2" requisitava o tempo ao "stratum 1" e assim por diante. O NTP foi o precursor na resposta ao problema de divergência de relógios, entretanto ele foi idealizado para redes de longa distância e não possuía precisão adequada para redes específicas. Com isso, em 2002, surgiu o *Precision Time Protocol* (PTP) (IEEE Std 1588-2002, 2002). Esse protocolo foi criado para alcançar precisão de submicrossegundos atuando em redes locais de controle e medição. Além disso, o PTP baseia-se em uma arquitetura mestre-escravo, diferente do NTP. No PTP, a comunicação entre o mestre e o escravo se baseia na troca dos instantes de tempo de envio de cada mensagem. Basicamente, tem-se o tempo de envio e retorno de cada mensagem e isso faz com que seja possível calcular o atraso gerado pelo meio de comunicação. Após iterações, o valor médio desse atraso é utilizado para ajustar o tempo interno do escravo. Em 2008, após a revisão do protocolo, uma segunda versão foi oficializada. Algumas mudanças nos métodos de correção de tempo interno e cálculo de diferencial foram realizadas e surgiu a versão do PTP utilizada até os dias atuais, como pode ser visto em (IEEE

Std 1588-2008 (Revision of IEEE Std 1588-2002), 2008). Esta versão também ficou conhecida como PTPv2.

Em 2011, o PTPv2 foi transferido para a família 802.1 do conjunto de protocolos *Audio Video Bridging* (AVB), como visto em (ASHJAEI *et al.*, 2021). Neste grupo, todos os padrões são voltados para desenvolvimento de ferramentas audiovisuais, as quais possuem requisitos de alta precisão de sincronização de tempo. Desta forma, surgiu o *generic Precision Time Protocol* (gPTP) (IEEE Std 802.1AS-2011, 2011), com funcionamento semelhante ao PTPv2. No entanto, podemos observar algumas diferenças nos componentes da rede local, que agora devem cumprir requisitos específicos de *hardware* possuindo um circuito individualizado para realizar as medições do tempo. Em 2018, o gPTP teve sua primeira revisão, presente em (IEEE P802.1AS-Rev, 2018) e esta segue vigente até a data de publicação deste trabalho.

2.2 Estampa de tempo

Um conceito relevante no estudo de protocolos de sincronização de relógios físicos é o de "estampa de tempo" ou *timestamp*, que é definido em (ISO 8601-1, 2019) como o registro de um determinado instante de tempo codificado em uma cadeia de caracteres. Esta codificação deve possibilitar a comparação e estabelecimento de uma ordem cronológica entre dois ou mais registros. Sistemas UNIX, por exemplo, utilizam um padrão para o *timestamp* que aloca as marcas temporais em números de 32 bits, como pode ser visto em (ISO/IEC 9899, 2018). O momento inicial da contagem dessas marcas é a data "1970-01-01 00:00:00 UTC".

O *timestamp* exerce um papel fundamental no funcionamento dos protocolos de sincronização de relógios físicos. A noção de tempo no sistema só é possível graças a um padrão estabelecido, o qual possibilita identificar quais componentes apresentam atraso ou avanço em seu relógio quando comparado a outro. Todos os valores de tempo e cálculos presentes nos protocolos apresentados nesse trabalho são baseados no conceito de *timestamp*.

2.3 Funcionamento dos protocolos PTP e gPTP

Esta seção busca apresentar a teoria relacionada aos protocolos PTP e gPTP. Como o gPTP é uma versão adaptada do PTP, seus conceitos são similares e podem ser apresentados sem distinção. Primeiramente, é apresentada uma visão geral dos protocolos na Seção 2.3.1. Na sequência, a arquitetura da rede é descrita na Seção 2.3.2. Os tipos de mensagens e o processo de sincronização são discutidos nas Seções 2.3.3 e 2.3.4, respectivamente. Por fim, são apresentados os modos de sincronização na Seção 2.3.5.

2.3.1 Visão geral

Ambos os protocolos têm seu funcionamento baseado no algoritmo Berkeley (GUSELLA; ZATTI, 1989). Esse algoritmo propõe a ideia de estabelecer sincronismo interno entre relógios físicos pelo estabelecimento de uma estrutura com dispositivos mestres e escravos. O mestre é inicialmente eleito por meio de um algoritmo denominado *Best Master-Clock Algorithm* (BMCA). Essa eleição é feita de acordo com parâmetros que envolvem ordem de prioridade e precisão do relógio, por exemplo, e que são pré-estabelecidos nos dispositivos do sistema. Após isso, o mestre eleito inicia o questionamento individual aos escravos sobre o seu respectivo valor de tempo. Ao receber as respostas, o mestre consegue estimar o tempo no dispositivo analisando o *round-trip time* (RTT), ou seja, o tempo de ida e volta da mensagem, como visto em (COULOURIS; DOLLIMORE; KINDBERG, 2005). O RTT gera o valor médio do atraso entre as entidades, e esse valor é utilizado para realizar o ajuste no dispositivo escravo. É interessante ressaltar que o mestre envia apenas o ajuste e não o valor absoluto necessário para a sincronia.

O que vemos no PTP e gPTP é uma adaptação do algoritmo de Berkeley. Nos protocolos, quem inicia a comunicação continua sendo o dispositivo mestre, no entanto todos os valores de tempo e cálculos são obtidos e realizados pelo dispositivo escravo. Isto serve para evitar uma possível sobrecarga de operações no mestre e promove uma escalabilidade da rede, já que descentraliza as operações. Além dessa diferença, o protocolo ainda adiciona uma série de novos dispositivos, nomenclaturas e mensagens que promovem a real implementação do algoritmo.

2.3.2 Arquitetura de rede

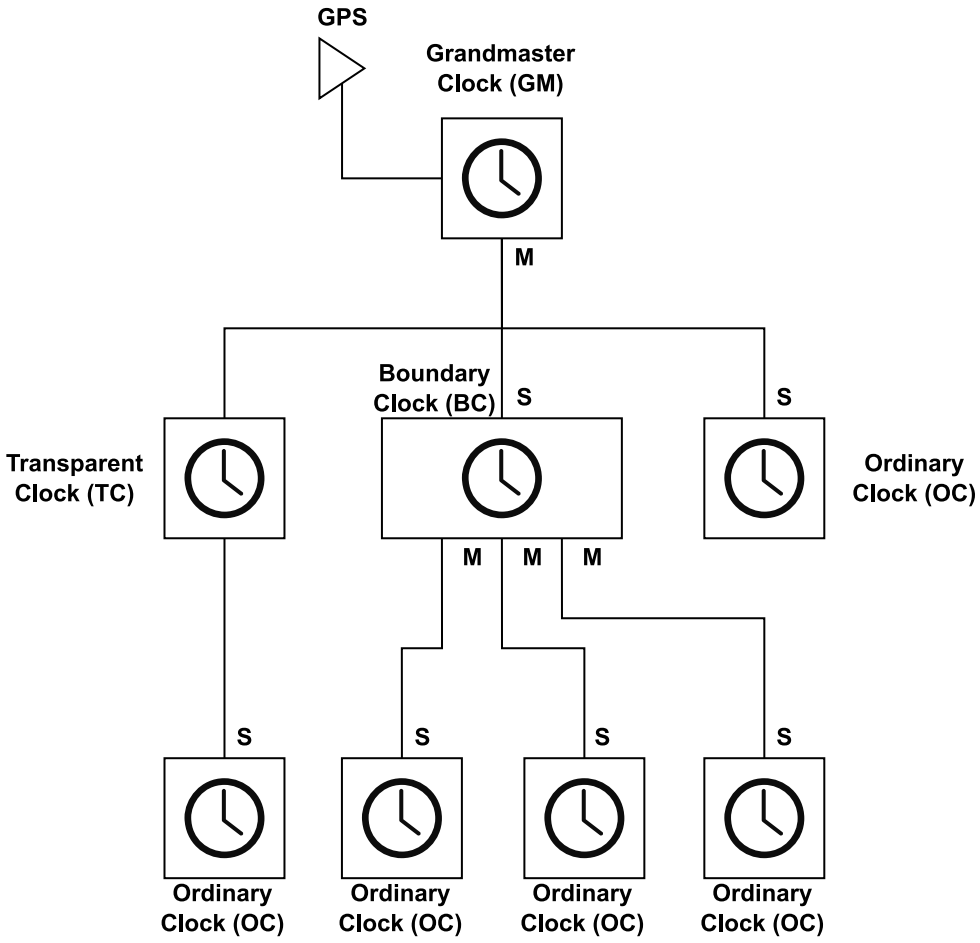
Os protocolos são baseados numa topologia hierárquica mestre-escravo, como mencionado anteriormente. Os dispositivos da rede possuem portas de conexão, que podem ser "Mestre" (M) ou "Escravo" (S). As portas M transmitem as informações de sincronização pela conexão, enquanto as portas S recebem essa informação. Essa estrutura é essencial para a transmissão de mensagens entre os dispositivos. A Figura 1 mostra uma arquitetura que utiliza os quatro componentes básicos que compõe a rede, definidos em (IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002), 2008) e (IEEE Std 802.1AS-2011, 2011), os quais são:

- *Grandmaster Clock* (GM): na tradução literal, o "Relógio Grão Mestre" é o componente que detém o tempo que será a referência de sincronização para os demais dispositivos na rede. Ele possui apenas portas M. A fim de propagar o tempo de forma precisa, a unidade GM deve ser alimentada também com um tempo preciso, para isso pode-se utilizar um dispositivo do tipo *Global Navigation Satellite System* (GNSS),

o qual pode ser um *Global Positioning System* (GPS), como visto em (JUNIPER, 2017);

- *Ordinary Clock* (OC): os "Relógios Ordinários" são os dispositivos que apresentam apenas uma conexão PTP. Eles podem ser tanto o destino das mensagens, como a fonte;
- *Boundary Clock* (BC): os "Relógios de Limites" são os dispositivos que possuem múltiplas conexões PTP. Na Figura 1, pode-se ver o *Boundary Clock* como sendo o dispositivo que possui uma entrada S e múltiplas saídas M;
- *Transparent Clock* (TC): os "Relógios Transparentes" aparecem em estruturas cascateadas e podem ser utilizados em configurações *End-to-end* (E2E) ou *Peer-to-peer*(P2P). Ambos os modos de funcionamento citados são explorados na Seção 2.3.5.

Figura 1 – Exemplo de arquitetura de uma rede PTP.



Fonte: Adaptado de Atahar Haridasula (2021).

2.3.3 Tipos de mensagens

A comunicação nos protocolos PTP e gPTP possui dois tipos de mensagem, como visto em (IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002), 2008): *Timed* (temporizada) ou *General* (genérica). A mensagem temporizada usa *timestamps* de maneira essencial em sua estrutura, ao contrário das genéricas, que servem apenas para carregar informações de configuração e sinalizações.

As mensagens do tipo "temporizada" são:

- *Sync*: é a primeira mensagem enviada no processo de sincronização. Esta mensagem tem seu tempo de envio salvo e este valor é essencial nos cálculos de sintonia;
- *Delay_Request*: após a finalização do processo iniciado pela mensagem de *Sync*, a mensagem de *Delay Request* ou "requisição de atraso" na tradução literal, é originada no escravo e destinada ao mestre a fim de iniciar uma nova fase de medições de tempo, como será visto na Seção 2.3.4.

As mensagens do tipo "genérica" são:

- *Announce*: mensagem utilizada para o estabelecimento de hierarquia da sincronização, ou seja, quem responderá como mestre e como escravo, como visto no algoritmo BMCA na Seção 2.3;
- *Follow_Up*: a mensagem *Follow_Up* é utilizada para enviar informações sobre valores de tempo adquiridos durante a troca de mensagens temporizadas;
- *Delay_Response*: semelhante à mensagem de *Follow_Up*, esta mensagem serve para encaminhar valores de tempo solicitados pela *Delay_Request*;
- *Management*: mensagens são usadas para atualizar os conjuntos de dados entre os relógios, assim como para customizar a rede antes do funcionamento;
- *Signaling*: são usadas para todos os demais propósitos de comunicação, como, por exemplo, negociar a taxa de mensagens *unicast* entre mestres e escravos.

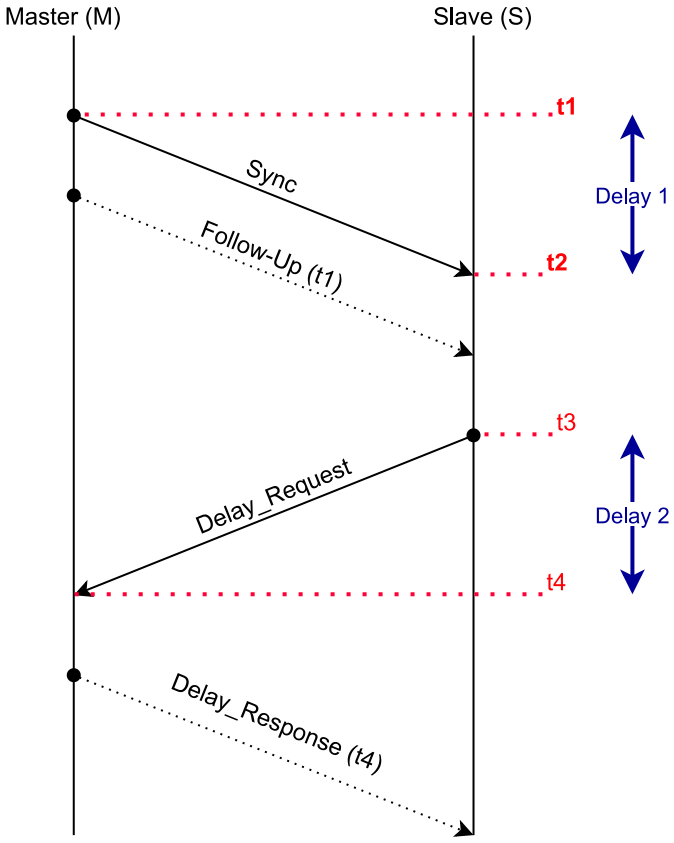
Além dessas mensagens, existem também as que surgem em alguns mecanismos de sincronização, como o *Peer-to-peer delay request*, o qual será abordado detalhadamente na Seção 2.3.5. Algumas mensagens desse mecanismo são: *pDelay_Request*, *pDelay_Response* e *pDelay_Response_Follow_Up*. É interessante perceber que essas mensagens são diferentes, pois possuem uma sinalização "P" no início de suas nomenclaturas, no entanto, funcionam da mesma maneira que as mensagens não sinalizadas.

2.3.4 Troca de mensagens para a sincronização

A troca de mensagens entre os dispositivos, como dito, começa com o GM enviando um pacote denominado *Sync*, como pode-se ver no diagrama da Figura 2. O tempo t1 é gravado e enviado no pacote de *Follow_Up*, no caso de uma comunicação realizada com a configuração *two-step*. Resumidamente, o modo *two-step* permite ao protocolo usar a mensagem de *Follow_Up*, a qual envia o valor do *timestamp* t1. Caso o protocolo esteja configurado para trabalhar em *one-step*, o tempo t1 é enviado no mesmo pacote de *Sync*. O modo *one-step* também é conhecido como *on-the-fly timestamp*. O escravo S agora tem o tempo de envio t1 e o de recebimento t2.

Após o recebimento da mensagem de *Follow_Up*, o escravo S responde a requisição com a mensagem *Delay_Request*. Ao fazer isso, grava em t3 o momento em que a mensagem foi enviada. O Mestre M, então, responde com o pacote *Delay_Response*, com o tempo do recebimento t4.

Figura 2 – Diagrama de troca de mensagens numa rede PTP



Fonte: Adaptado de Atahar Haridasula (2021).

Desta forma, o escravo agora possui quatro valores: t1, t2, t3 e t4. Como apresentado na Figura 2, existem dois diferenciais de tempo formados: "Delay 1" que representa o espaço dado pelo cálculo t2 - t1, e o "Delay 2", que representa t4 - t3. Sendo assim, S pode realizar

o cálculo presente na Equação 1, apresentado em (ATAHAR HARIDASULA, 2021), obtendo assim o atraso médio na comunicação:

$$\begin{aligned} \text{Atraso médio} &= \frac{Delay\ 1 + Delay\ 2}{2} \\ \text{Atraso médio} &= \frac{(t2 - t1) + (t4 - t3)}{2} \end{aligned} \tag{1}$$

O Atraso médio, ou *MeanPathDelay*, é utilizado como referência nos ajustes realizados pelo escravo. Os valores do atraso médio devem se limitar e convergir ao atraso físico entre duas entidades. Por exemplo, se há um atraso de 100 nanossegundos na conexão entre computadores de uma rede, o atraso médio deve convergir para 100 nanossegundos. Desta forma, com o valor do atraso entre as unidades é possível realizar os ajustes para a sintonia entre elas.

2.3.5 Modos de sincronização

Os protocolos compartilham dois modos de sincronização. O PTP se baseia nos dois mecanismos, enquanto o gPTP apenas em um deles. De acordo com os documentos oficiais (IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002), 2008) e (IEEE Std 802.1AS-2011, 2011), tem-se os mecanismos:

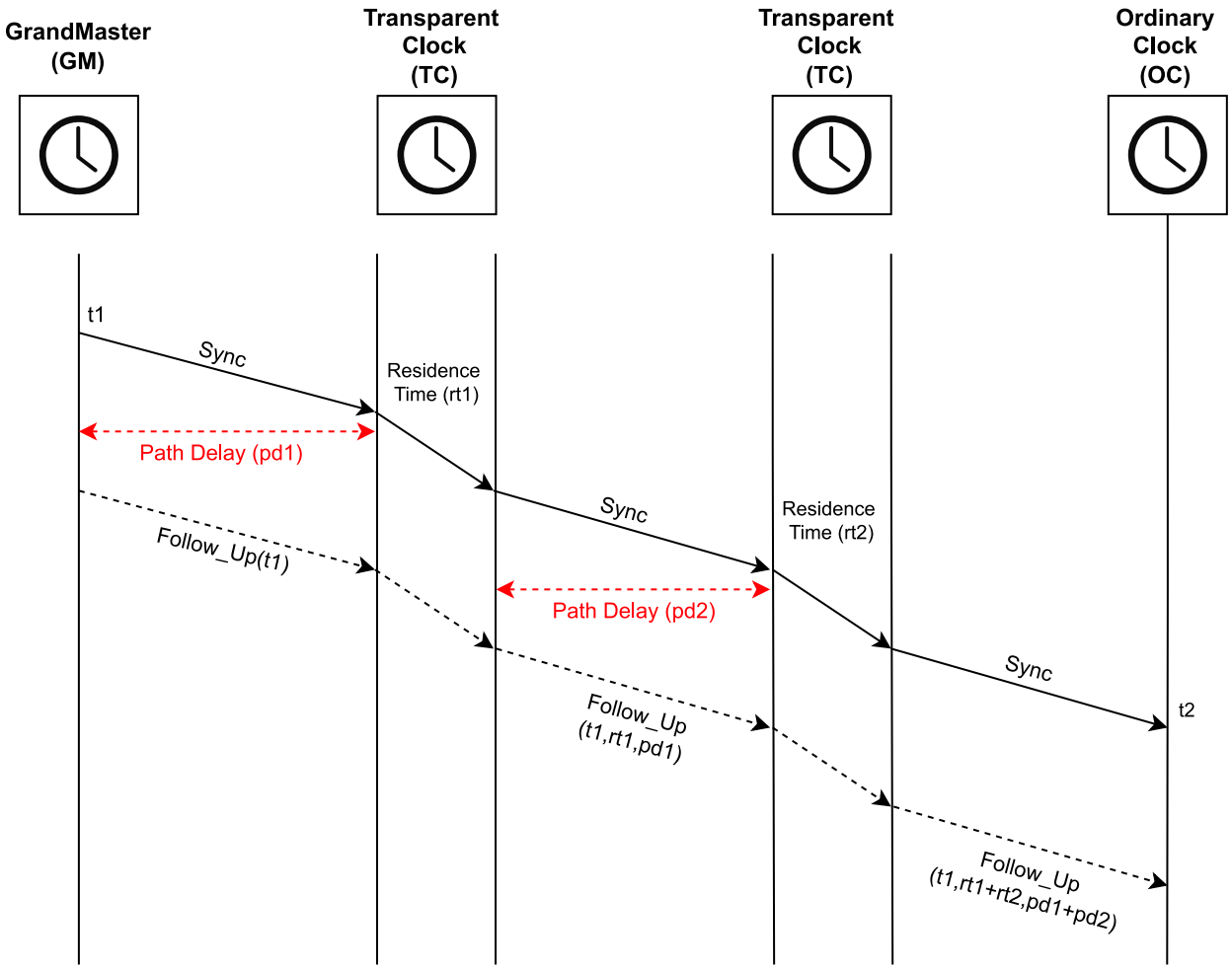
- *delay request-response*: presente no PTP, o qual utiliza as mensagens *Sync, Delay_Req, Delay_Resp* e *Follow_Up*, se necessário;
- *peer delay mechanism*: presente no PTP e gPTP, o qual utiliza *pDelay_Req, pDelay_Resp* e *pDelay_Resp_Follow_Up*, se necessário.

Estes mecanismos refletem o método de comunicação utilizado, que podem ser *end-to-end transparent clock* ou *peer-to-peer transparent clock*.

2.3.5.1 End-to-end transparent clock

Neste mecanismo, a sincronização ocorre com as mensagens de *delay request-response* e acontecem fim-a-fim, ou seja, do GM até o próximo OC. Os dispositivos TC apenas reajustam o tempo do pacote com seu *residence time* (rt), que equivale ao tempo de processamento da mensagem internamente no dispositivo.

Figura 3 – Processo de sincronização fim-a-fim



Fonte: Adaptado de Atahar Haridasula (2021).

Na Figura 3 é possível visualizar quatro dispositivos: um GM, dois do tipo TC e um OC. A estrutura apresentada mostra o percurso da mensagem *Sync* vinda do GM. Na primeira interação, pode-se observar as mensagens *Sync* e *Follow_Up*, bem como o tempo t1 e o *Path Delay* (pd1), o qual é o atraso no caminho entre o GM e o primeiro TC. Vale notar que nesta mensagem de *Follow_Up* existe apenas a informação t1.

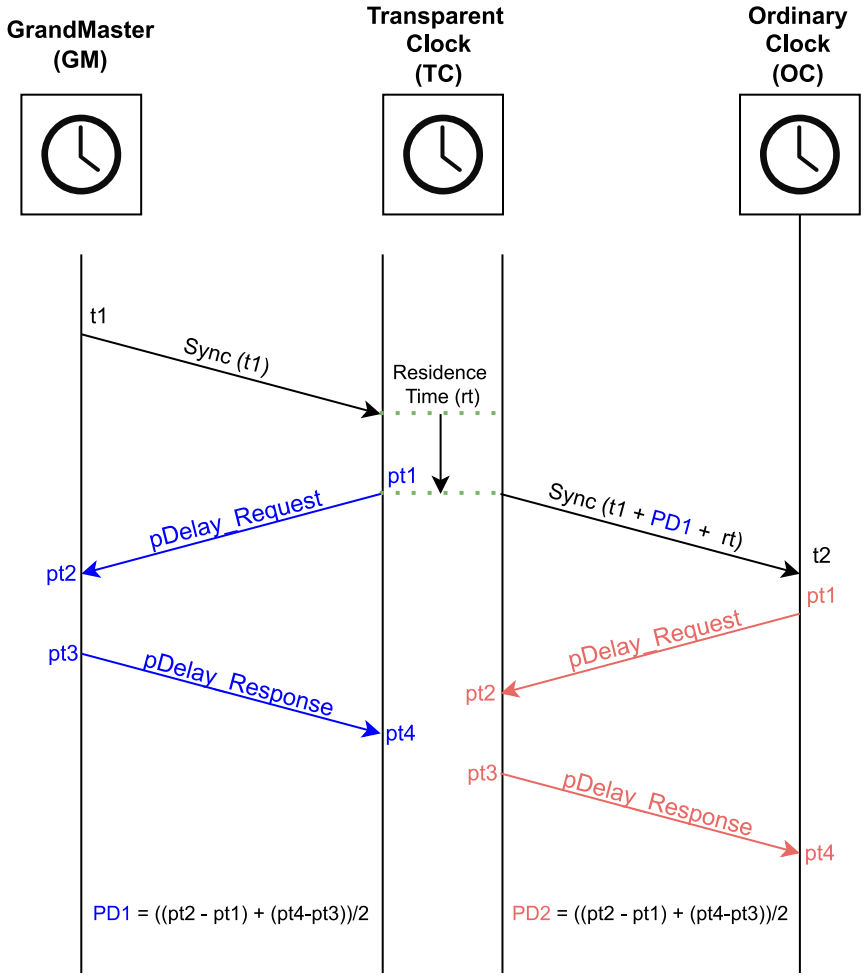
Entre chegar ao primeiro TC e sair para o próximo componente, a mensagem de sincronização é afetada pelo atraso intrínseco ao dispositivo. Na Figura 3, este atraso é representado como rt. Este tempo aparece novamente na mensagem seguinte de *Follow_Up*, juntamente com o valor de pd1. Deste modo, é possível perceber que o papel dos dispositivos TC, em uma comunicação fim-a-fim, é de acumular seus valores de rt e *Path Delay*.

2.3.5.2 Peer-to-peer transparent clock

Este processo usa as mensagens *peer delay mechanism* e a sincronização ocorre entre cada par, ou seja, o GM, ao enviar mensagem de sincronização, recebe as respostas de todos

os pares com as mensagens iniciadas com "P", como visto na Figura 4. Este mecanismo difere do anterior, pois agora todos os componentes trocam mensagens entre si.

Figura 4 – Processo de sincronização entre pares



Fonte: Adaptado de Atahar Haridasula (2021).

Neste caso, a fim de reduzir a quantidade de mensagens na Figura 4, usa-se como exemplo o modo *one-step*, ou seja, a informação do tempo de saída está diretamente no pacote de *Sync*. O dispositivo TC, ao receber a mensagem *Sync*, retorna a mensagem *pDelay_Request*. Isso origina o tempo pt1, que representa o tempo de saída da mensagem do TC, e o pt2, o tempo de chegada no GM. Após isso, originam-se pt3 e pt4 com a mensagem *pDelay_Response*.

Esses valores serão usados para o cálculo de PD1, o qual representa o atraso médio do caminho, ou *MeanPathDelay*, entre os dois dispositivos. Além disso, o valor de PD1 é encaminhado para os dispositivos seguintes e assim por diante até o OC. Vale ressaltar que em cada uma das interações existe o cálculo do atraso médio entre os pares, como visto em PD2. No OC, o cálculo para o ajuste do relógio agora faz uso desses valores de atrasos médios ao longo da comunicação.

2.4 Comparativo entre os protocolos PTP e gPTP

É importante esclarecer que, apesar dos comportamentos essenciais serem semelhantes, os dois protocolos divergem em pequenas configurações ou requisitos específicos. Com os parâmetros básicos de funcionamento dos protocolos em mente, pode-se analisar individualmente quais desses itens estão presentes em cada uma das versões do protocolo. A Tabela 1 mostra, resumidamente, as diferenças e semelhanças entre os objetos de estudo deste trabalho.

Tabela 1 – Comparação de parâmetros entre PTP e gPTP.

Funcionalidades	**IEEE 1588-2008 (PTP)**	**IEEE 802.1AS (gPTP)**
Envio de mensagens	One-step e Two-step	
Intervalo de transmissão de *Sync*	Até 16 mensagens por segundo	
Precisão	Submicrossegundos	
Dispositivos suportados	Independe de hardware	Apenas com hardware específico
Cálculo de atraso	Peer-to-peer ou End-to-end	Peer-to-peer
Protocolos suportados	Camada de enlace, rede e transporte	Camada de enlace

Pode-se observar que o PTP e o gPTP compartilham as seguintes configurações:

- Envio de mensagens: ambos os protocolos funcionam no modo *one-step* e *two-step*. Vale lembrar que essa configuração refere-se ao modo de encaminhamento das mensagens. O modo *two-step* envia duas mensagens para sincronização enquanto o *one-step* envia apenas uma;
- Intervalo de transmissão de *Sync*: ambos os protocolos podem chegar a enviar até 16 mensagens de sincronização por segundo, ou seja, 16 envios de mensagem *Sync* por segundo.
- Precisão: ambos os protocolos podem chegar a submicrossegundos de precisão entre as unidades, no entanto, apenas o gPTP demonstra-se capaz de chegar as dezenas de nanossegundos, como apresentado em (IEEE P802.1AS-Rev, 2018).

Apesar das semelhanças, os protocolos apresentam diferenças substanciais, dentre elas:

- Compatibilidade entre os dispositivos: como dito na Seção 2.1, uma das principais diferenças entre os protocolos ocorre nos dispositivos ao longo da rede. No gPTP, todos os componentes precisam ser *"time-aware"*, ou seja, precisam estar "cientes do tempo", em uma tradução literal. Basicamente, os dispositivos devem possuir mecanismos de *hardware* específicos para operarem numa arquitetura gPTP. No caso do PTP, não é necessária nenhuma capacidade adicional ao longo da rede de sincronização;
- Protocolos suportados: em prol de garantir uma maior precisão na sincronização, o gPTP opta por utilizar mensagens exclusivamente em camadas mais baixas do

modelo OSI (ZIMMERMANN, 1980). Como visto na Tabela 1, o gPTP faz o uso apenas da camada de enlace enquanto o PTP pode utilizar a camada de enlace até a de transporte, passando pela camada de rede.

2.5 Deriva de relógios físicos

O oscilador a cristal é um componente responsável pela geração da frequência de processadores e esta frequência dita o ritmo da realização das atividades. Usualmente, materiais piezoelétricos são utilizados na elaboração desses componentes, os quais baseiam a geração dos sinais na vibração de ressonância do material, como visto em (MALVINO; BATES, 2016). O quartzo é um exemplo de material que compõe grande parte dos osciladores existentes.

Como qualquer componente eletrônico, seu grau de incerteza pode fazer com que os valores da saída flutuem em torno de um limiar estabelecido. No caso dos cristais, há oscilação de valores em torno das suas frequências nominais. Esse fator pode ser desprezível em muitos casos, mas possui relevância quando envolve sistemas sensíveis ao tempo. As divergências nas oscilações geram um fenômeno denominado "deriva". A deriva de um relógio físico acontece devido ao desvio que um oscilador possui em relação à sua frequência nominal, levando a um atraso ou avanço na contagem de ciclos em um microprocessador (COULOURIS; DOLLIMORE; KINDBERG, 2005).

Ao lidar com simuladores, estes problemas são inexistentes e ignorados, já que todos os contadores de ciclo estão sob o mesmo processador e a mesma fonte geradora de frequência. Entretanto, adicionar essa especificidade em simuladores de rede, ainda mais em simulações de redes sensíveis ao tempo, é crucial para garantir a fidelidade de um cenário real em ambiente controlado.

Segundo (MCCLANING; VITO, 2000), uma das abordagens para a geração dos distúrbios em sinais é a utilização do método "*Additive White Gaussian Noise*" (AWGN), na tradução literal "ruído gaussiano branco aditivo". O sinal gerado é "gaussiano" pois utiliza a distribuição de probabilidades gaussiana, é "branco" pois seu arranjo no espectro é uniforme e é "aditivo" por que é adicionado ao sinal de saída. No caso, o sinal ruidoso é aleatoriamente gerado em torno de uma média de valor 0 e desvio padrão estabelecido conforme a amplitude do ruído. A implementação em questão é útil no caso de osciladores, pois a distribuição gaussiana faz com que os valores gerados sejam, em grande maioria, próximos de zero, mas não excluindo picos pontuais de amplitude. Esta abordagem possibilita trazer ao ambiente de simulação comportamentos aleatórios observados em ruídos de fundo.

Implementações de osciladores normalmente levam em consideração uma deriva constante, ou seja, o relógio teria um desvio linear proporcional ao tempo em que está em funcionamento. Isso também acontece na biblioteca gPTP e tal implementação é abordada na Seção 2.6.

2.6 Trabalhos relacionados

O artigo (ASHJAEI *et al.*, 2021) apresenta o estado da arte no desenvolvimento de protocolos para cumprir requisitos de redes sensíveis ao tempo, ou *Time-Sensitive Networking* (TSN). Estes protocolos estão sob o grupo IEEE 802.1 e foram originalmente criados para aplicações em redes voltadas ao desenvolvimento audiovisual. O grupo em questão busca abranger diferentes particularidades dessas redes, dentre elas: sincronização de tempo; adequação de escalonamento de tarefas; adequação das melhores rotas de comunicação; tolerância à falhas na comunicação. O foco na leitura do artigo (ASHJAEI *et al.*, 2021) foi para o aspecto de sincronização de tempo, o qual tem suas funcionalidades implementadas no protocolo IEEE 802.1AS (gPTP) que será objeto de estudo do presente trabalho. O artigo (ASHJAEI *et al.*, 2021) apresenta o estado da arte no desenvolvimento do gPTP, bem como modificações adicionadas pelas recentes revisões e seu funcionamento geral.

O artigo (KOVÁCSHÁZY; FERENCZ, 2012) apresenta as ideias que serviram como base para a realização do presente trabalho. Nele é apresentado uma comparação do IEEE 1588-2008 (PTP) sob diferentes implementações de *timestamp*. As quais são: (i) uso exclusivo de software; (ii) implementação mista entre software e hardware; (iii) uso exclusivo de hardware. O artigo apresenta o cenário para comparar a precisão de sincronização obtida e também a manutenção da sincronia entre as entidades ao longo do tempo. O resultado mais estável e preciso surge ao individualizar a operação de estampa de tempo numa assistência de *hardware*, pois a operação se torna menos suscetível às variações de carga no sistema, a qual foi realizada com uma geração de tráfego na rede.

O artigo (PUTTNIES *et al.*, 2018) mostra uma recente e única implementação do protocolo gPTP no simulador OMNet++. Este projeto teve início na Universidade de Rostock, na Alemanha, e foi apresentado pela equipe criadora no evento *OMNeT++ Community Summit 2018*. O trabalho trata da implementação do protocolo IEEE 802.1AS internamente no simulador. A equipe considerou como motivação a alta demanda em sistemas de rede em tempo real e a não existência de soluções para simulação desse tipo de arquitetura em *softwares* não proprietários. Desta forma, seria de extrema relevância implementá-lo dentro do OMNet++. Ao longo do artigo a equipe pontuou que reduziu as funcionalidades do protocolo, que conta apenas com as funções de sincronização de tempo e propagação de atraso, enquanto o algoritmo BMCA foi ignorado, por exemplo. Um ponto relevante citado pelos desenvolvedores está no fato de adotarem uma deriva de relógio constante. Essa deriva constante é de fácil implementação, mas não exibe as principais variações observadas na prática. Os desvios nos relógios vistos na realidade são, muitas vezes, causados por ruídos de fundo, como visto na Seção 2.5. Os autores pontuaram como trabalhos futuros a necessidade de implementar novos modelos de deriva a fim de validar o projeto em situações mais próximas da realidade. Este trabalho de conclusão de curso busca apresentar os resultados considerando o gPTP com modelos de deriva não constantes.

O artigo (LIM *et al.*, 2011) apresenta uma implementação e análise de desempenho realizado com o protocolo IEEE 802.1AS (gPTP) sob funcionamento no simulador OMNet++. Considera-se uma rede composta por doze componentes de uma rede gPTP, sendo eles um *GrandMaster*, três dispositivos do tipo *Boundary Clock* e oito dispositivos *Ordinary Clock* definidos como escravos da rede. Para análise dos resultados, o artigo faz o uso do parâmetro *Propagation Delay*, o qual é uma outra nomenclatura para *MeanPathDelay* ou Atraso médio, em português.

O presente trabalho tem como objetivo analisar o desempenho dos protocolos PTP e gPTP considerando diferentes derivas de relógio físico e diferentes cargas de tráfego na rede. Esse objetivo foi motivado pelos seguintes fatores: (i) o artigo (KOVÁCSHÁZY; FERENCZ, 2012) não faz menção ao IEEE 802.1AS (gPTP), sendo o gPTP importante para estabelecer uma rede de dispositivos que possuem um *hardware* específico para o tratamento do *timestamp*; (ii) resultados encontrados em (LIM *et al.*, 2011) e (PUTTNIES *et al.*, 2018) consideraram uma deriva linear constante nos relógios físicos de cada dispositivo.

O capítulo a seguir trata da arquitetura geral da rede utilizada com base nas implementações realizadas com as bibliotecas do PTP e gPTP. Além disso, o capítulo mostra quais são os parâmetros configurados no decorrer dos testes.

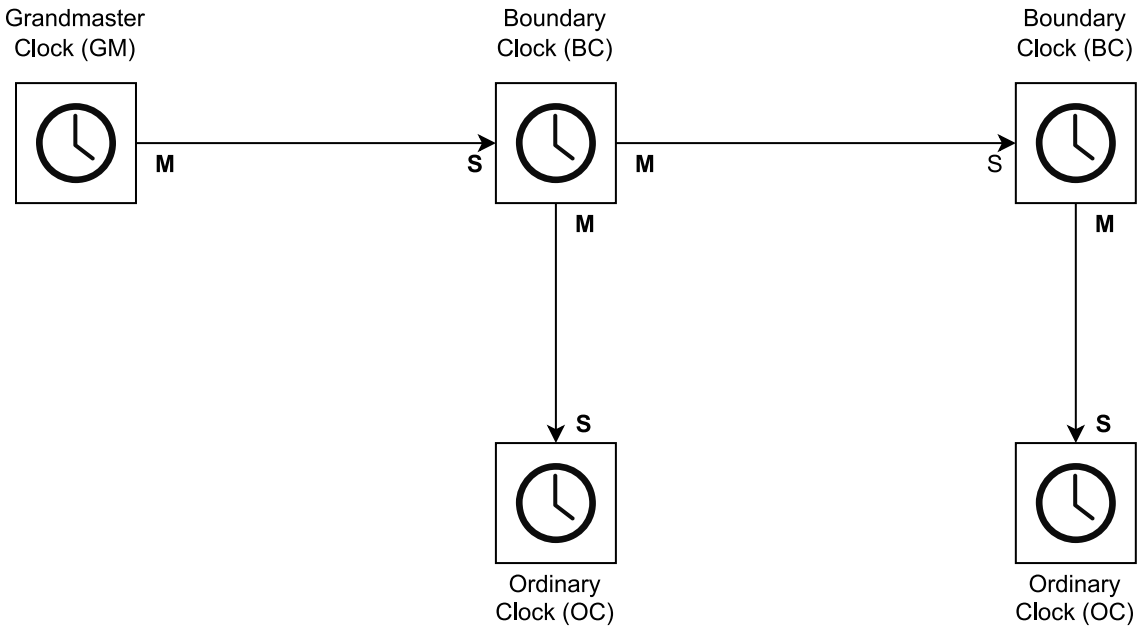
3 AMBIENTE DE SIMULAÇÃO

A Seção 3.1 apresenta a arquitetura geral do sistema, com os dispositivos a serem utilizados. A Seção 3.2 descreve as implementações realizadas. Encerra-se a Seção 3.3 com os parâmetros configurados para realizar a variação de tráfego na rede e a alteração na deriva dos relógios físicos.

3.1 Arquitetura geral do sistema

De modo geral, a arquitetura do sistema para ambos os protocolos contém os seguintes componentes: um dispositivo GM, dois BC e dois OC, os quais são conectados conforme ilustra a Figura 5.

Figura 5 – Arquitetura geral do sistema



Fonte: Autoria própria.

Esta arquitetura fornece os pontos necessários para coleta de resultados e comparações desejadas. Nas implementações, é possível comparar os relógios das unidades OC e BC e assim demonstrar a sincronização entre elas.

A seguir são apresentadas as arquiteturas específicas para cada uma das implementações de protocolo a serem comparadas.

3.2 Implementação dos protocolos PTP e gPTP

Para analisar os desempenhos dos protocolos PTP e gPTP é utilizado o simulador OMNEt++. Ele é um simulador de eventos discretos, modular e baseado em componentes

C++, com grande uso em simulação de redes. O projeto pode ser utilizado gratuitamente para aulas e pesquisas acadêmicas, por exemplo. Por se tratar de um projeto de código aberto, há inúmeras contribuições que fazem do simulador uma ferramenta muito utilizada pela comunidade. É possível citar a biblioteca de código aberto *INET Framework*, que possui diversas funcionalidades implementadas envolvendo redes com e sem fio, além de envolver redes móveis.

No presente trabalho é utilizada a biblioteca *INET*. Vale pontuar que cada protocolo foi desenvolvido em versões diferentes do simulador. As seguintes versões do OMNet++ e INET foram utilizadas: O gPTP funciona com as versões OMNet++ 5.2.1 e INET 3.6.3 e o PTP com as versões OMNet++ 4.6 com INET 2.6.

Houve uma tentativa de utilizar ambos os protocolos na mesma versão do simulador e da biblioteca de desenvolvimento. No Apêndice B constam arquivos do início deste trabalho. Entretanto, não foi possível avançar muito devido a uma grande quantidade de problemas de compilação e a diversas diferenças nas versões.

O sistema operacional utilizado para hospedar a ferramenta de simulação foi o Ubuntu 16.04 LTS. Além disso, o computador possui um processador *Intel Core* i5-8250U e 8 *gigabytes* de memória do tipo *Random Access Memory* (RAM).

3.2.1 Implementação do PTP

A implementação do PTP foi realizada com a *libPTP*, que é uma biblioteca integrada em um projeto de código aberto denominado *ptp-sim*, como visto em (PTP-SIM, 2016). Este projeto busca trazer ao OMNet++ as funcionalidades do IEEE 1588-2008 (PTPv2). Dentre elas, a capacidade de atribuir uma pilha de chamadas PTP a um dispositivo e também placas de rede capazes de realizar transações PTP.

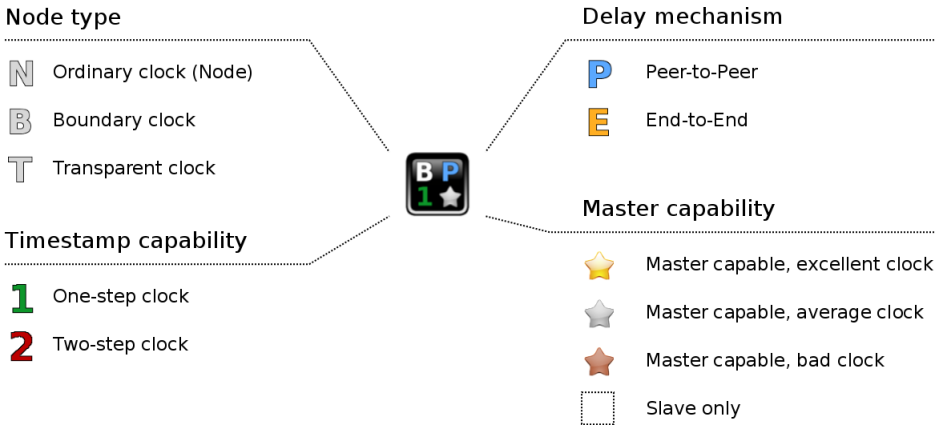
A biblioteca em questão possui uma particularidade, ela faz o uso de um conjunto específico de ícones e figuras para facilitar a compreensão da configuração. A Figura 6 mostra graficamente a posição e nomenclatura das unidades do sistema. A visualização da configuração de cada bloco é definida pelas seguintes simbologias:

- Espaço superior esquerdo mostra o tipo do dispositivo:
  - N - *Ordinary Clock* (OC)
  - B - *Boundary Clock* (BC)
  - T - *Transparent Clock* (TC)
- Espaço superior direito apresenta o modo de sincronização utilizado:
  - P - peer-to-peer (P2P)
  - E - End-to-end (E2E)
- Espaço inferior esquerdo mostra a configuração de troca de mensagens:

  - 1 - *One-step Clock (on-the-fly timestamps)*
  - 2 - *Two-step Clock*

- Diagonal inferior direita apresenta a confiabilidade do relógio, o parâmetro utilizado pelo algoritmo BMCA:
  - Estrela dourada - Relógio confiável.
  - Estrela prateada - Relógio de média confiança.
  - Estrela bronzeada - Relógio de baixa confiança.
  - Sem estrela - Funciona apenas como escravo.

Figura 6 – Esquema de símbolos da *libPTP*.

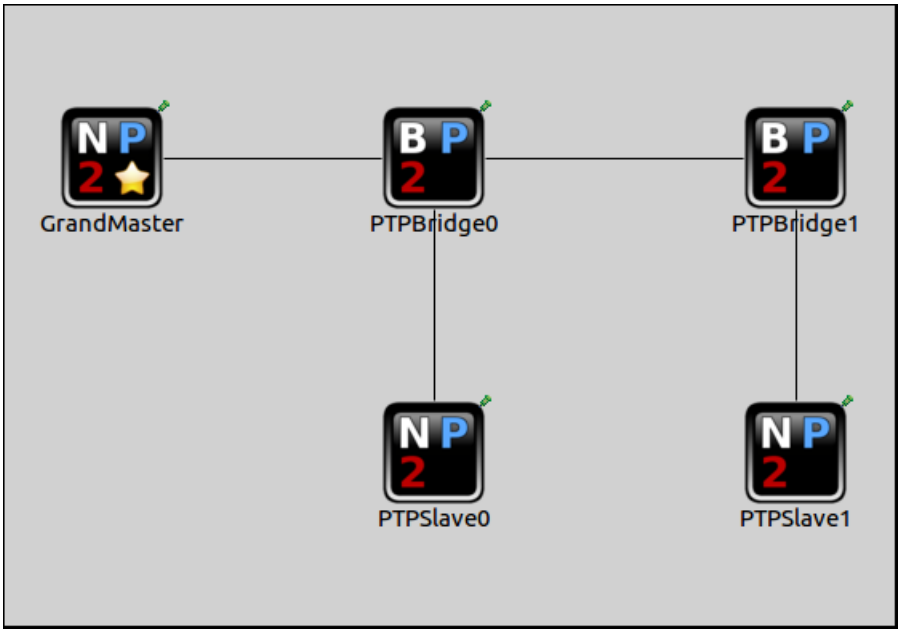


Fonte: (LIBPTP, 2016).

Como dito, a implementação realizada com a *libPTP* busca simular o ambiente apresentado na Figura 5. Ao comparar a arquitetura geral, apresentada na Figura 5, com a implementação apresentada na Figura 7, pode-se perceber que eles respeitam a posição e configuração determinados. Desta forma o dispositivo *GrandMaster*, presente na Figura 7, está configurado para funcionar na forma de um OC, utilizando o *two-step clock* para o envio das mensagens, *peer-to-peer delay* e sendo o relógio mais confiável do sistema para ser eleito como GM pelo BMCA, já que é o dispositivo mestre na rede PTP.

Os componentes nomeados como "PTPBridge" fazem alusão aos dispositivos BC presentes na Figura 5. Eles estão configurados como sendo do tipo BC, utilizando troca de mensagens com o *two-step clock* e no modo *peer-to-peer delay*. Além disso, só podem ser escravos na comunicação pois não possuem o símbolo de uma estrela na representação visual. Os dispositivos do tipo "PTPSlave" possuem configuração semelhante, no entanto sendo dispositivos OC.

Figura 7 – Implementação da arquitetura geral com a libPTP.



Fonte: Autoria própria.

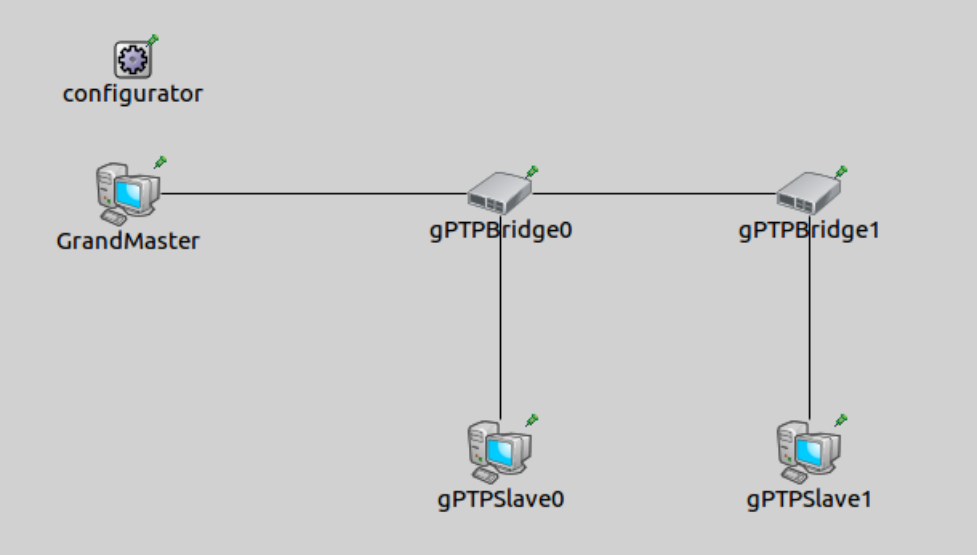
Os códigos e simulações utilizadas na implementação do PTP estão no repositório *github.com/mglucas/TCC-PTP*.

3.2.2 Implementação do gPTP

O projeto *IEEE 802.1AS gPTP for Clock Synchronization* possui o mesmo cunho do *ptp-sim*, no entanto, implementa a versão mais recente do PTP, o gPTP. Este projeto foi idealizado por um grupo de estudantes da Universidade de Rostock, na Alemanha, e é discutido na Seção 2.6.

Diferentemente da biblioteca anterior, a *libgPTP* não possui a representação gráfica para auxiliar na apresentação da configuração. Além disso, os componentes da biblioteca possuem uma menor complexidade quando comparados à *libPTP*. Na Figura 8 estão apresentados os componentes presentes na implementação realizada com a *libgPTP*.

Figura 8 – Implementação da arquitetura geral com a *libgPTP*



Fonte: Autoria própria.

Comparando as nomenclaturas e a posição de cada componente na rede apresentada na Figura 5 com a Figura 8, pode-se perceber que o dispositivo "GrandMaster" faz o papel de GM, as unidades "gPTPBridge" fazem referência aos dispositivos BC e as "gPTPSlave" aos dispositivos OC e estes fazem o papel do escravo. Além disso, é possível verificar um dispositivo denominado "configurator", o qual é responsável por coordenar a inicialização da rede, configurando os endereços e rotas necessárias.

As funções *peer-to-peer delay* e *two-step clock* são inerentes à implementação realizada com a *libgPTP*, diferentemente das opções vistas na *libPTP*. Isto pois, como visto na Seção 2.4, o gPTP funciona apenas no modo *peer-to-peer*, pois todas as suas unidades devem possuir os mecanismos específicos de medição de tempo, sendo capazes de se sincronizar com o mestre. A exclusividade do modo *two-step clock* está relacionada ao fato da equipe criadora da biblioteca a ter implementado desta forma, onde não há parâmetro para a mudança de função.

Os códigos e simulações utilizadas na implementação do GPTP estão no repositório *github.com/mglucas/TCC-GPTP*.

3.3 Parâmetros de configuração

Para garantir que ambos os ambientes funcionem sob os mesmos requisitos, alguns parâmetros devem ser ajustados para garantir maior fidelidade na comparação dos resultados.

3.3.1 Intervalo de envio de mensagens

Um dos parâmetros essenciais na simulação do alto tráfego na rede é o *SyncInterval*. Segundo (JUNIPER, 2013), este parâmetro controla a taxa de envio de mensagens de *Sync* e, como visto na Seção 2.3.4, esta mensagem é a primeira enviada para iniciar a sincronização do mestre com os escravos.

Os valores de *SyncInterval* podem variar de 7 até -4, como pode ser visto em (JUNIPER, 2013), no entanto cada aplicação demanda uma faixa ou somente um valor específico, como visto em (TECH, 2022). Este valor está relacionado a potência de base 2 que controla o intervalo de envio. Por exemplo, se o *SyncInterval* é equivalente a -3, tem-se uma mensagem a cada 0,125 segundos. O cálculo detalhado é apresentado na Equação 2:

$$\begin{aligned} Intervalo &= 2^{Sync\_Interval} \\ SyncInterval &= -3 \\ Intervalo &= 2^{-3} \\ Intervalo &= \frac{1}{2^3} = \frac{1}{8} \\ Intervalo &= 0{,}125s \end{aligned} \tag{2}$$

Desta forma, pode-se controlar o tráfego de mensagens em uma rede PTP e gPTP. Essa é a forma utilizada para simular o aumento do número de mensagens na rede, variando o tráfego de dados.

3.3.2 Atraso nas conexões entre as entidades

Para simular um possível atraso físico de conexão entre os componentes do sistema, o simulador OMNet++ oferece uma série de parâmetros ao conectar os pares de uma rede. No caso, são usados dois parâmetros nas conexões entre os componentes, em ambas as implementações:

- Atraso de comunicação: 25 nanossegundos.
- Taxa de dados: 100Mbps

A configuração é realizada nos arquivos de *Network Descriptor* (NED), que fazem parte do padrão utilizado pelo OMNet++ para o desenvolvimento de redes de simulação. Um arquivo NED define todos os componentes envolvidos em uma rede, bem como suas conexões e demais configurações.

O "Atraso da comunicação" entre as entidades serve de referência para o *MeanPathDelay* (Atraso médio), visto na Seção 2.3.4. A "Taxa de dados" só representa a velocidade

comum de uma conexão *Fast Ethernet* (100BASE-T), como pode ser visto em (IEEE Std 802.3u, 1995). O atraso de 25 nanossegundos foi escolhido com base nos ambientes de teste descritos no artigo (LIM *et al.*, 2011). Segundo os autores, este valor de atraso representa uma conexão de 5 metros entre os dispositivos.

O capítulo a seguir descreve os testes realizados, desde o preparo do ambiente de simulação até como são coletados os resultados. Vale ressaltar que também são detalhadas as formas de deriva dos relógios utilizadas em cada teste, sendo elas lineares ou não.

4 DESCRIÇÃO DOS TESTES

Este capítulo descreve o preparo e procedimento adotados em cada um dos testes construídos com as arquiteturas apresentadas no capítulo anterior. O texto disseca quais parâmetros foram adotados em cada uma das implementações, bem como as modificações realizadas quanto às derivas dos relógios físicos.

4.1 Variação de tráfego de mensagens de sincronização

Os testes de variação de tráfego de mensagens buscam submeter os protocolos a um estresse gerado pelo fluxo de comunicação na rede. Assim, espera-se aferir se a simulação é influenciada pela quantidade de requisições geradas.

4.1.1 Preparo

Para coletar os resultados da simulação com variação de tráfego de mensagens de sincronização ambas as arquiteturas observadas no Capítulo 3 foram submetidas a mudanças no envio de mensagens, utilizando o parâmetro *Sync Interval*, como visto na Seção 3.3.1. Foram escolhidos três valores negativos de *Sync Interval*, isso pois esses números representam um envio de mais de uma mensagem por segundo. As escolhas estão presentes na Tabela 2.

Tabela 2 – Valores adotados para o *SyncInterval*.

Testes	SyncInterval	Equivalente em segundos (s)
Cenário 1	-1	0,500
Cenário 2	-2	0,250
Cenário 3	-3	0,125

O valor -4 não foi considerado, pois a documentação do gPTP mostra que o valor padrão para a conexão é o de -3, como visto em (OMNETPP, 2022). Os relógios físicos no escravo foram submetidos à uma deriva linear constante com diferentes coeficientes angulares e coeficientes lineares nulos. Os dispositivos na rede foram submetidos aos seguintes valores apresentados na Tabela 3.

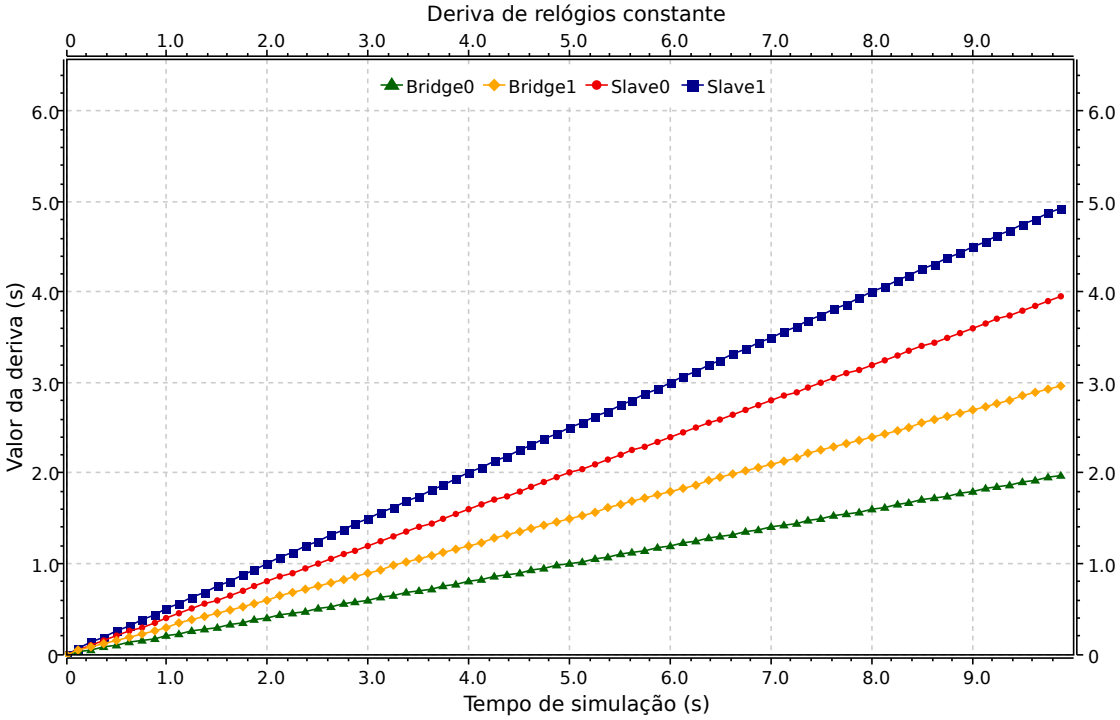
Tabela 3 – Coeficientes angulares da deriva linear.

Unidade do sistema	Coeficientes angulares
GrandMaster	0
Bridge0	2
Bridge1	3
Slave0	4
Slave1	5

Os valores apresentados na tabela 3 foram escolhidos de forma a respeitar os valores pré-definidos no projeto (LIBPTP, 2016) e (JANCHIVNYAMBUU; PUTTNIES; DANIELIS, 2018), ambos realizavam testes de deriva linear com esses números.

Na Figura 9 é possível visualizar um exemplo das derivas lineares geradas pelas implementações.

Figura 9 – Exemplo de deriva dos relógios linear e constante



Fonte: Autoria própria.

4.1.2 Procedimento

Com ambas as arquiteturas configuradas, o valor de *SyncInterval* foi modificado de forma crescente, fazendo com que o tempo entre as mensagens fosse menor a cada teste, desta forma os resultados foram gerados para 0,5, 0,250 e 0,125 segundos entre as mensagens, como apresentado na Tabela 2. Além disso, os testes foram feitos com um tempo de simulação limite de 10 segundos.

4.1.3 Avaliação

Os gráficos e dados foram obtidos utilizando as ferramentas de simulação do OMNet++. O resultado da simulação aparece na forma de vetores e valores escalares, os quais podem ser convertidos em arquivos do tipo *Analysis File* (ANF), o qual é uma extensão específica utilizada pelo OMNet++ para geração de gráficos e análises estatísticas. O simulador também apresenta a média, desvio padrão e variância das medidas obtidas.

Os resultados obtidos envolvem o Atraso médio, ou *MeanPathDelay*. Para atestar o funcionamento do protocolo, ou seja, a sincronização dos dispositivos na rede, as unidades devem ter o valor do atraso médio convergindo para 25 nanossegundos, de acordo com as configurações apresentadas na Seção 3.3.2 e como visto na Seção 2.3.4. Considera-se a convergência se as amostras estiverem sob um limiar de 5% do valor base de 25 nanossegundos.

4.2 Deriva de relógios físicos não constante

Os testes de deriva de relógios físicos não constante buscam submeter os protocolos a valores oscilantes em seus relógios. Deste modo, espera-se aferir se as implementações tem seu resultado influenciado por distúrbios.

4.2.1 Preparo

Assim como a simulação anterior, esta é realizada utilizando as duas implementações vistas no Capítulo 3. O valor da taxa de envio de mensagens de sincronização (*SyncInterval*) foi fixado em -3, representando 0,125 segundos entre as mensagens, o qual é o valor padrão para uma rede gPTP.

O ponto mais importante nessa simulação é a geração da deriva de relógios físicos não constante. Para isso, foi utilizada a função seno com diferentes amplitudes e frequências para cada dispositivo, como pode ser visto na Tabela 4. Além disso, é feito um novo teste com os valores da Tabela 5 considerando as amplitudes do teste anterior multiplicadas por 1000, a fim de averiguar o comportamento dos protocolos com grandes distúrbios.

Tabela 4 – Frequências das senoides.

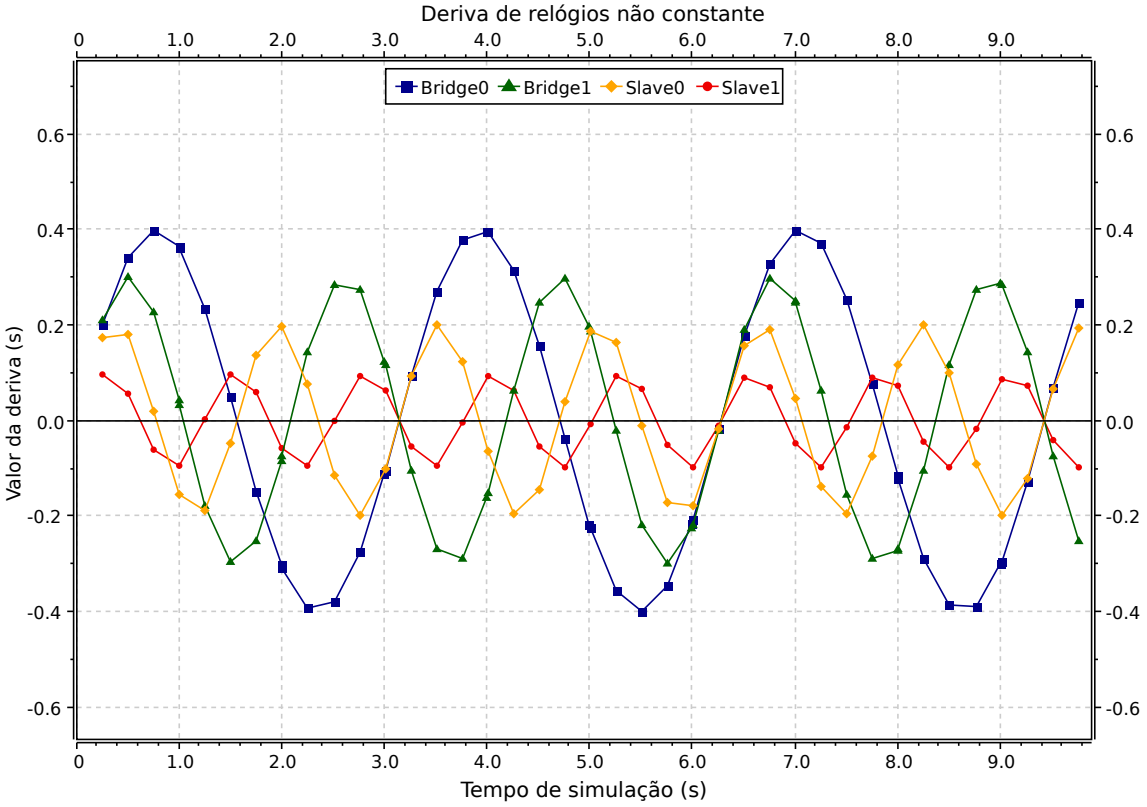
Unidades do sistema	Frequência	Amplitude
GrandMaster	0	0
Bridge0	10	4E-6
Bridge1	20	3E-6
Slave0	30	2E-6
Slave1	40	1E-6

Tabela 5 – Frequências das senoides com amplitude ampliada.

Unidades do sistema	Frequência	Amplitude
GrandMaster	0	0
Bridge0	10	4E-3
Bridge1	20	3E-3
Slave0	30	2E-3
Slave1	40	1E-3

Vale ressaltar que os valores de frequência vistos nas Tabelas 4 e 5 estão em radianos. Desta forma, é esperado que a deriva das unidades seja uma amostragem com base em senoides de diferentes frequências e amplitude, o que é exemplificado na Figura 10.

Figura 10 – Exemplo da deriva de relógio utilizando amostragem em uma onda senoidal.



Fonte: Autoria própria.

Cada ponto apresentado na Figura 10 apresenta um valor que será adicionado ao relógio físico.

4.2.2 Procedimento

Com ambas as arquiteturas estabelecidas, configura-se os valores da Tabela 4. Inicia-se a simulação. Após o término, os valores da Tabela 5 são adicionados e inicia-se novamente a simulação. Ambos os testes duram 10 segundos de tempo de simulação.

4.2.3 Avaliação

Assim como na simulação anterior, os resultados obtidos envolvem o valor de atraso médio. As unidades devem ter uma diferença de 25 nanossegundos entre elas, considerando uma margem de erro de 5%, de acordo com as configurações apresentadas na Seção 3.3.2.

4.3 Deriva de relógios físicos com ruído gaussiano

Os testes de modelagem de deriva de relógios físicos submetem os protocolos a valores oscilantes em seus relógios. No caso, uma distribuição gaussiana com média zero e desvio

padrão variável será utilizada para gerar o ruído gaussiano. Deste modo, espera-se aferir se as implementações tem seu resultado influenciado por distúrbios aleatórios em seus relógios.

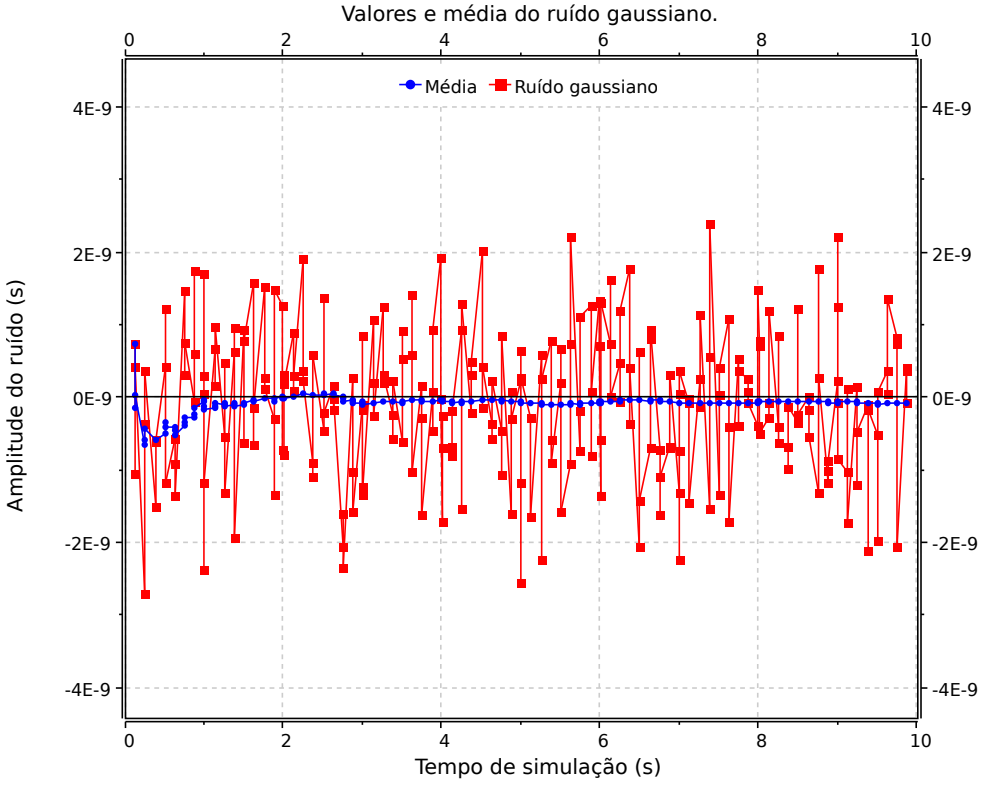
4.3.1 Preparo

Esse teste busca submeter as implementações à um ruído branco como função de deriva de relógios físicos, desta forma, aproximando os resultados ao observado na prática com cristais de quartzo. Para isso, foi utilizada o método *Additive White Gaussian Noise* (AWGN), o qual gera um ruído com base em uma distribuição gaussiana e o soma à um sinal de saída.

No caso do teste em questão, a geração do ruído envolve uma função de distribuição gaussiana com média zero e desvio padrão equivalente à 1×10^{-9}. Com essa configuração espera-se que os ruídos oscilem ao redor de zero, com pequenos distúrbios aleatórios ao longo da simulação, como pode ser visto na Figura 11. Isso acontece pois na distribuição gaussiana, os valores menores do que um desvio padrão ao redor de média equivalem à 68,27% dos valores de saída, enquanto dois desvios padrões ao redor da média equivalem a 95,45%. Desta forma, aproximadamente 95% dos distúrbios serão de até 2×10^{-9}.

Esses valores farão com que o relógio sofra derivas aleatórias positivas e negativas ao longo da simulação. Vale lembrar que o atraso na conexão é de 25 nanossegundos. Deste modo, as derivas causarão uma alteração de até 8%.

Figura 11 – Exemplo da deriva utilizando modelagem de ruído com distribuição gaussiana.



Fonte: Autoria própria.

Assim como nas simulações anteriores, o valor da taxa de envio de mensagens de sincronização (*SyncInterval*) foi fixado em -3.

4.3.2 Procedimento

Com ambas as arquiteturas estabelecidas, configura-se os valores de média e desvio padrão. Inicia-se a simulação e após os 10 segundos de simulação pode-se obter os resultados.

4.3.3 Avaliação

Assim como nas simulações anteriores, os resultados obtidos envolvem o valor de atraso médio. As unidades devem ter uma diferença de 25 nanossegundos entre elas, considerando uma margem de erro de 5%, de acordo com as configurações apresentadas na Seção 3.3.2.

O próximo capítulo apresenta os resultados obtidos com os cenários de teste descritos neste capítulo. Os resultados são mostrados por meio de gráficos e tabelas.

5 ANÁLISE DOS RESULTADOS

Este capítulo mostra os resultados obtidos nos experimentos envolvendo as redes PTP e gPTP. Em cada gráfico são apresentados os valores do atraso médio dos dispositivos da rede, durante 10 segundos de tempo de simulação. Os resultados obtidos entre 0 e 2 segundos de simulação são desconsiderados, pois representam o período de preparação do ambiente de simulação. O capítulo também apresenta tabelas com amostras recolhidas nos segundos 3, 6 e 9, bem como o erro em relação ao valor esperado. O cálculo do erro é feito de forma proporcional e se dá pelo módulo de um menos a razão do valor amostrado pelo valor configurado na conexão. Por exemplo, uma amostra com valor de 24 nanossegundos gerá um erro de 4%, como visto na Equação 3:

$$\begin{aligned} \text{Erro} &= |1 - \frac{\text{Valor amostrado (ns)}}{\text{Atraso configurado (ns)}}| \\ \text{Erro} &= |1 - \frac{24}{25}| \\ \text{Erro} &= 0{,}04 \\ \text{Erro } (\%) &= 4 \end{aligned} \tag{3}$$

Considera-se sincronizado o protocolo mantiver as amostragens realizadas com erro abaixo de 5%, assim como apresentado em (LIM *et al.*, 2011).

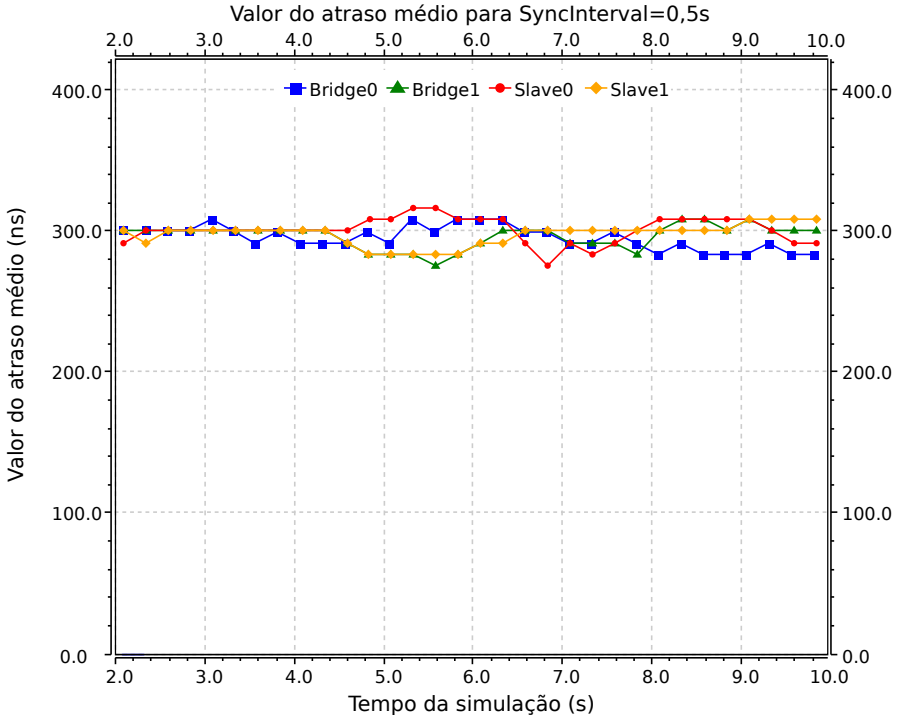
5.1 Resultados para variação de tráfego de mensagens de sincronização

A intenção deste teste é evidenciar se o valor do atraso médio em cada componente da rede é afetado pela variação no número de mensagens enviadas pelo mestre. A deriva dos relógios físicos neste cenário é linear com inclinação positiva, como apresentado no Capítulo 4. Ressalta-se ainda que o atraso entre os dispositivos é de 25 nanossegundos, e espera-se que o atraso médio oscile em torno deste valor.

5.1.1 Resultados do PTP

A Figura 12 apresenta o comportamento do atraso médio considerando um intervalo entre mensagens de sincronização de 0,5 segundos, ou seja, 2 mensagens por segundo. As Tabelas 6 e 7 evidenciam os valores das amostras do gráfico, bem como o erro em relação ao atraso na conexão.

Figura 12 – Atraso médio para SyncInterval de 0,5 segundos em uma rede PTP.



Fonte: Autoria própria.

É possível observar o comportamento do atraso médio ao longo da simulação, o qual se manteve, em grande parte do tempo, oscilando em torno de 300 nanossegundos. Ressalta-se também que pela deriva dos relógios ser linear e ter seu início nulo, nos instantes iniciais, é esperado que essa deriva não afete significativamente os dispositivos da rede.

Tabela 6 – Amostras dos valores do atraso médio e erro para os dispositivos *Bridge* em uma rede PTP sob SyncInterval de 0,5 segundos.

	Bridge0		**Bridge1**	
Instante da simulação (s)	**Atraso médio (ns)**	**Erro**	**Atraso médio (ns)**	**Erro**
3	308,0	1132,00%	300,4	1101,58%
6	308,0	1132,00%	292,0	1068,12%
9	283,6	1034,58%	308,0	1132,00%

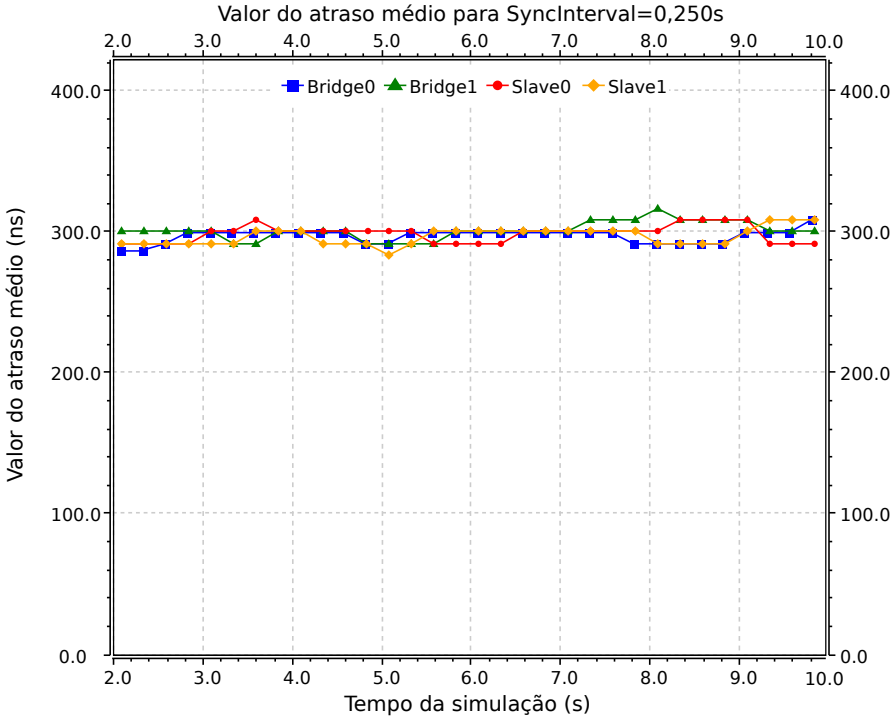
Tabela 7 – Amostras dos valores do atraso médio e erro para os dispositivos *Slave* em uma rede PTP sob SyncInterval de 0,5 segundos.

	Slave0		**Slave1**	
Instante da simulação (s)	**Atraso médio (ns)**	**Erro**	**Atraso médio (ns)**	**Erro**
3	300,6	1102,25%	300,4	1101,58%
6	308,4	1133,56%	291,3	1065,08%
9	308,4	1133,56%	308,0	1132,00%

Pode-se perceber que os seus valores giram em torno dos 300 nanossegundos, apesar do erro ser alto em relação aos 25 nanossegundos, é possível perceber o requisito de submicrossegundos sendo cumprido.

A Figura 13 apresenta o comportamento do atraso médio considerando um intervalo entre mensagens de sincronização de 0,250 segundos (4 mensagens por segundo). As Tabelas 8 e 9 mostram amostras específicas da simulação em questão.

Figura 13 – Atraso médio para SyncInterval de 0,250 segundos em uma rede PTP.



Fonte: Autoria própria.

É possível observar novamente o comportamento do atraso médio ao longo da simulação, o qual se manteve oscilando em torno de 300 nanossegundos. Além disso, visualmente no gráfico é possível perceber uma menor volatilidade nos valores.

Tabela 8 – Amostras dos valores do atraso médio e erro para os dispositivos *Bridge* em uma rede PTP sob SyncInterval de 0,250 segundos.

	Bridge0		Bridge1	
Instante da simulação (s)	**Atraso médio (ns)**	**Erro**	**Atraso médio (ns)**	**Erro**
3	299,6	1098,54%	300,6	1102,25%
6	299,6	1098,54%	300,6	1102,25%
9	299,6	1098,54%	308,4	1133,56%

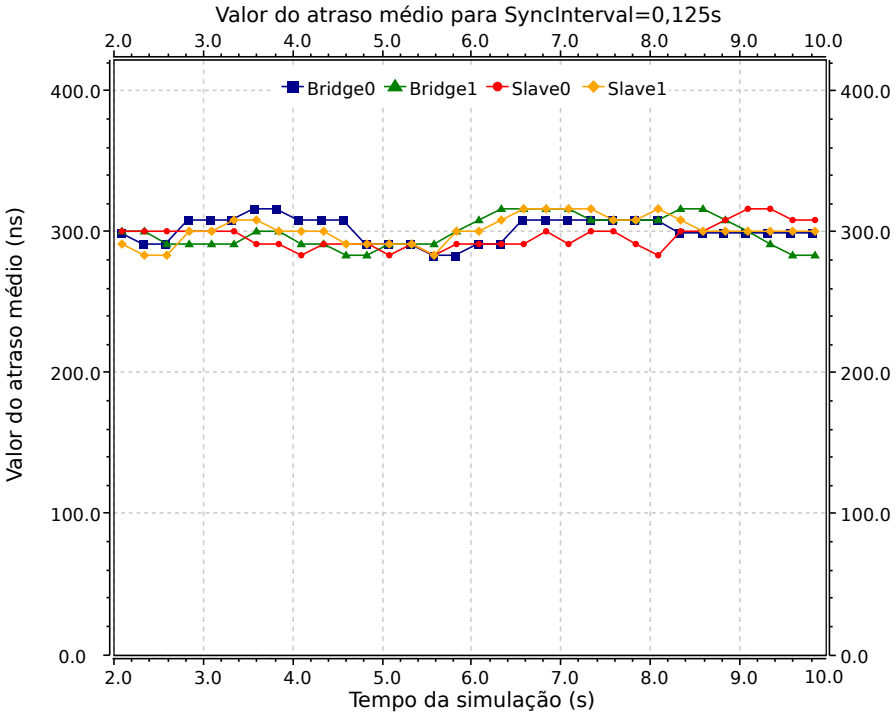
Tabela 9 – Amostras dos valores do atraso médio e erro para os dispositivos *Slave* em uma rede PTP sob SyncInterval de 0,250 segundos.

	Slave0		Slave1	
Instante da simulação (s)	**Atraso médio (ns)**	**Erro**	**Atraso médio (ns)**	**Erro**
3	300,4	1101,58%	291,3	1065,08%
6	292,0	1068,12%	300,4	1101,58%
9	308,0	1132,00%	300,4	1101,58%

As Tabelas 8 e 9 mostram os valores observados nas amostras da Figura 13. Reforçando a estabilidade dos valores em torno dos 300 nanossegundos, apesar do erro ser alto em relação aos 25 nanossegundos.

A Figura 14 mostra o comportamento do atraso médio considerando um intervalo entre mensagens de sincronização de 0,125 segundos (8 mensagens por segundo). As Tabelas 10 e 11 mostram amostras específicas da simulação em questão.

Figura 14 – Atraso médio para SyncInterval de 0,125 segundos em uma rede PTP.



Fonte: Autoria própria.

Como visto anteriormente, os valores do atraso médio para todos os dispositivos continuam oscilando em torno de 300 nanossegundos. As Tabelas 10 e 11 também apresentam o mesmo comportamento.

Tabela 10 – Amostras dos valores do atraso médio e erro para os dispositivos *Bridge* numa rede PTP sob SyncInterval de 0,125 segundos.

	Bridge0		Bridge1	
Instante da simulação (s)	**Atraso médio (ns)**	**Erro**	**Atraso médio (ns)**	**Erro**
3	308,2	1132,79%	292,7	1070,94%
6	291,0	1064,13%	308,4	1133,56%
9	299,6	1098,46%	300,6	1102,25%

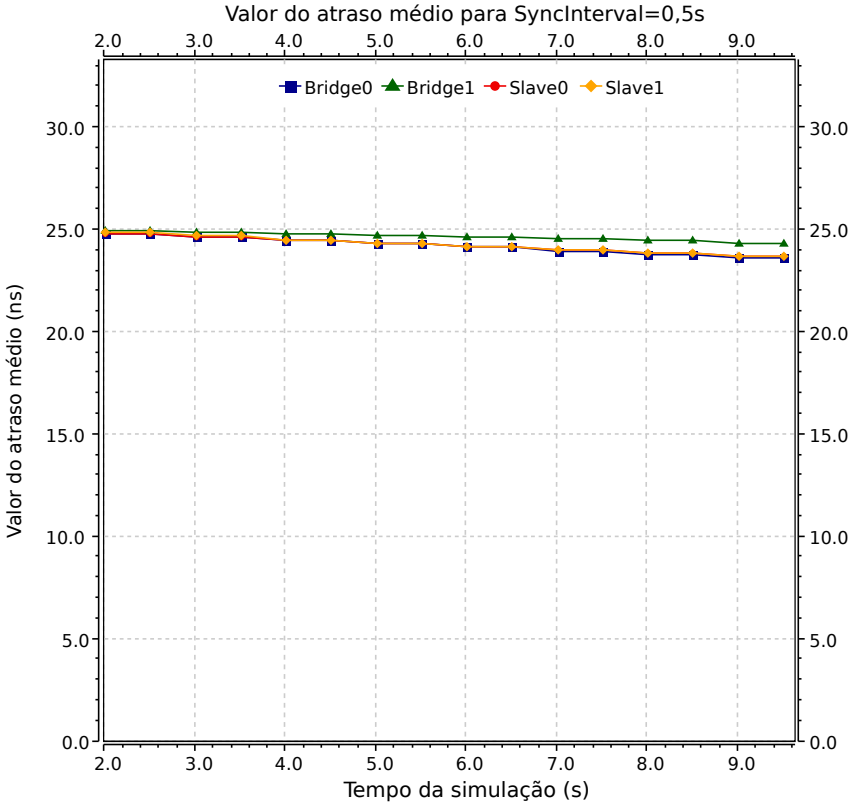
Tabela 11 – Amostras dos valores do atraso médio e erro para os dispositivos *Slave* numa rede PTP sob SyncInterval de 0,125 segundos.

	Slave0		Slave1	
Instante da simulação (s)	**Atraso médio (ns)**	**Erro**	**Atraso médio (ns)**	**Erro**
3	300,6	1102,25%	300,4	1101,58%
6	292,7	1070,94%	300,4	1101,58%
9	317,8	1171,13%	300,4	1101,58%

5.1.2 Resultados do gPTP

O resultado da simulação com o gPTP para um intervalo entre mensagens de 0,5 segundos é apresentado na Figura 12. As Tabelas 12 e 13 evidenciam os valores das amostras do gráfico, bem como o erro em relação ao atraso na conexão.

Figura 15 – Atraso médio para SyncInterval de 0,5 segundos em uma rede gPTP.



Fonte: Autoria própria.

Graficamente é possível perceber uma estabilidade bem maior quando comparado à simulação com o PTP. Os valores oscilam em torno dos 25 nanossegundos, como esperado. Apesar disso, há uma inclinação que faz com que a distância em relação ao valor base de 25 nanossegundos cresça com o tempo, isso se dá pelo fato da deriva ser linear ao longo do tempo, ou seja, com o passar dos segundos os valores da deriva se tornam ainda maiores.

Tabela 12 – Amostras dos valores do atraso médio e erro para os dispositivos *Bridge* numa rede gPTP sob SyncInterval de 0,5 segundos.

	Bridge0		Bridge1	
Instante da simulação (s)	**Atraso médio (ns)**	**Erro**	**Atraso médio (ns)**	**Erro**
3	24,60	1,59%	24,83	0,68%
6	24,10	3,59%	24,58	1,68%
9	23,60	5,59%	24,33	2,68%

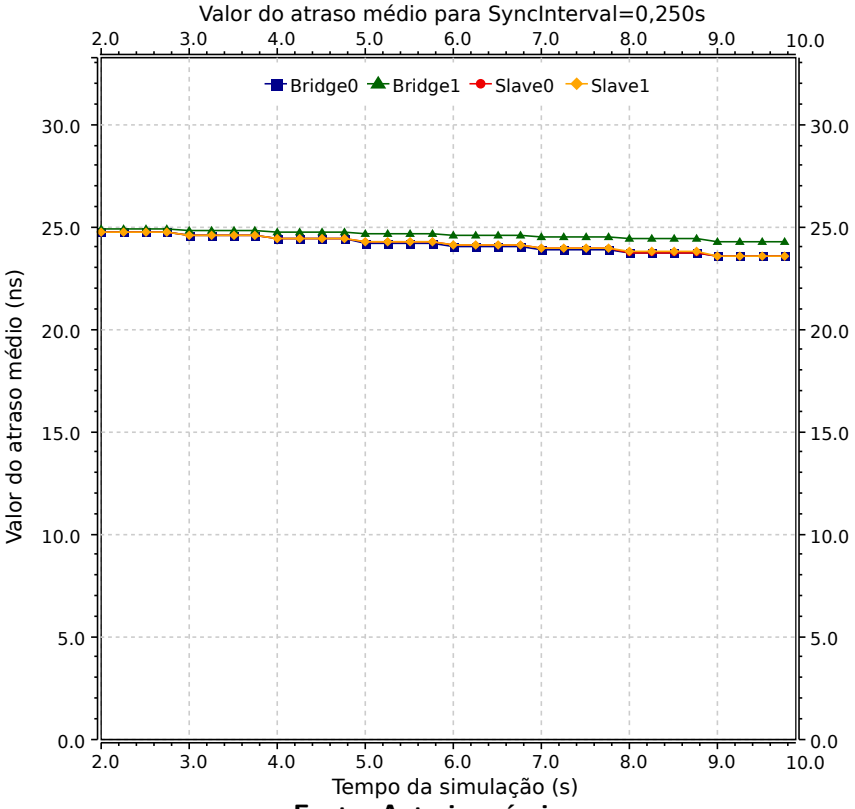
Tabela 13 – Amostras dos valores do atraso médio e erro para os dispositivos *Slave* numa rede gPTP sob SyncInterval de 0,5 segundos.

	Slave0		Slave1	
Instante da simulação (s)	**Atraso médio (ns)**	**Erro**	**Atraso médio (ns)**	**Erro**
3	24,63	1,48%	24,65	1,42%
6	24,13	3,48%	24,15	3,42%
9	23,63	5,48%	23,65	5,42%

É possível verificar essa inclinação nos resultados das Tabelas 12 e 13. O erro cresce quase que linearmente e afasta o valor do atraso médio do definido nas configurações, desta forma é esperado que o erro só tenda a crescer acompanhando a deriva linear implementada.

Para um intervalo entre mensagens de 0,250 segundos tem-se o resultado apresentado na Figura 13. As Tabelas 14 e 15 evidenciam os valores das amostras do gráfico e o erro.

Figura 16 – Atraso médio para SyncInterval de 0,250 segundos em uma rede gPTP.



Fonte: Autoria própria.

Pode-se perceber o mesmo comportamento visto anteriormente. Os valores seguem oscilando minimamente, quando comparado ao PTP, em torno dos 25 nanossegundos, como esperado. A inclinação também está presente, reforçando a influência da deriva linear ao longo do tempo.

Tabela 14 – Amostras dos valores do atraso médio e erro para os dispositivos *Bridge* numa rede gPTP sob SyncInterval de 0,250 segundos.

	Bridge0		Bridge1	
Instante da simulação (s)	**Atraso médio (ns)**	**Erro**	**Atraso médio (ns)**	**Erro**
3	24,56	1,76%	24,81	0,76%
6	24,06	3,76%	24,56	1,76%
9	23,56	5,76%	24,31	2,76%

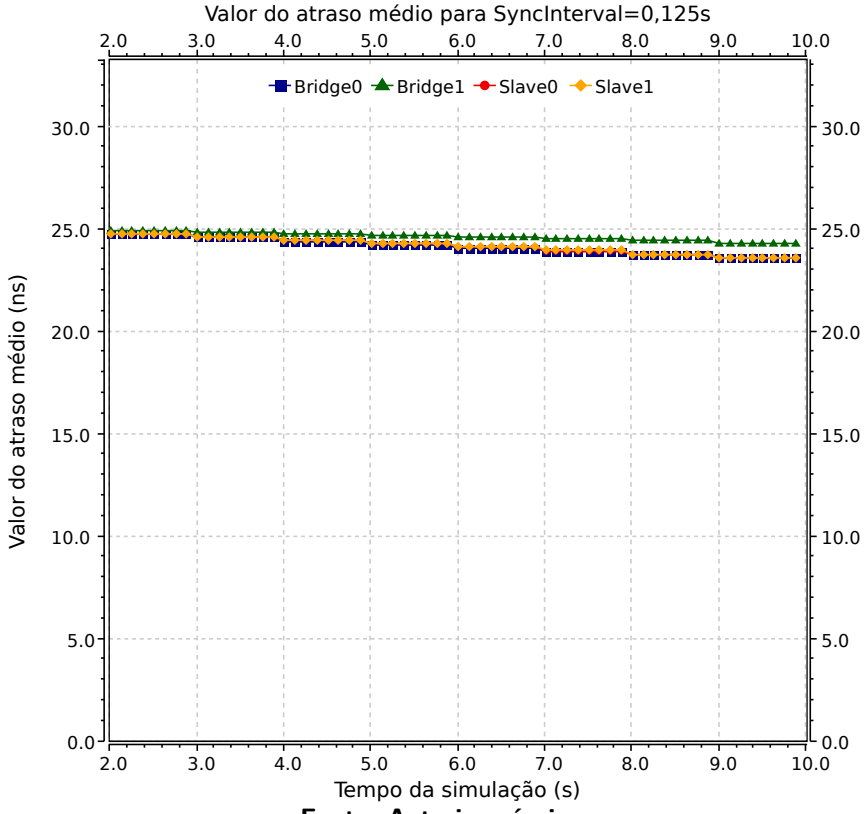
Tabela 15 – Amostras dos valores do atraso médio e erro para os dispositivos *Slave* numa rede gPTP sob SyncInterval de 0,250 segundos.

	Slave0		Slave1	
Instante da simulação (s)	**Atraso médio (ns)**	**Erro**	**Atraso médio (ns)**	**Erro**
3	24,59	1,64%	24,60	1,58%
6	24,09	3,64%	24,10	3,59%
9	23,59	5,64%	23,60	5,59%

É possível verificar essa inclinação novamente nos resultados das Tabelas 14 e 15. O erro cresce na proporção em que o tempo passa. Apesar disso, os valores seguem próximos da configuração previamente realizada.

Enfim, o resultado com o intervalo entre mensagens de 0,125 segundos é apresentado na Figura 14. Novamente, as tabelas 16 e 17 mostram os valores das amostras do gráfico e o erro.

Figura 17 – Atraso médio para SyncInterval de 0,125 segundos em uma rede gPTP.



Fonte: Autoria própria.

Novamente, o mesmo comportamento pode ser observado. Percebe-se que a sincronização é afetada e acaba desviando o relógio do padrão estabelecido.

Tabela 16 – Amostras dos valores do atraso médio e erro para os dispositivos *Bridge* numa rede gPTP sob SyncInterval de 0,125 segundos.

	Bridge0		Bridge1	
Instante da simulação (s)	**Atraso médio (ns)**	**Erro**	**Atraso médio (ns)**	**Erro**
3	24,54	1,84%	24,80	0,81%
6	24,04	3,84%	24,55	1,81%
9	23,54	5,84%	24,30	2,81%

Tabela 17 – Amostras dos valores do atraso médio e erro para os dispositivos *Slave* numa rede gPTP sob SyncInterval de 0,125 segundos.

	Slave0		Slave1	
Instante da simulação (s)	**Atraso médio (ns)**	**Erro**	**Atraso médio (ns)**	**Erro**
3	24,57	1,73%	24,58	1,67%
6	24,07	3,73%	24,08	3,67%
9	23,57	5,73%	23,58	5,67%

Como visto graficamente, os valores dos erros seguem crescendo quase que linearmente, indicando que a rede estará convergindo para longe do valor configurado.

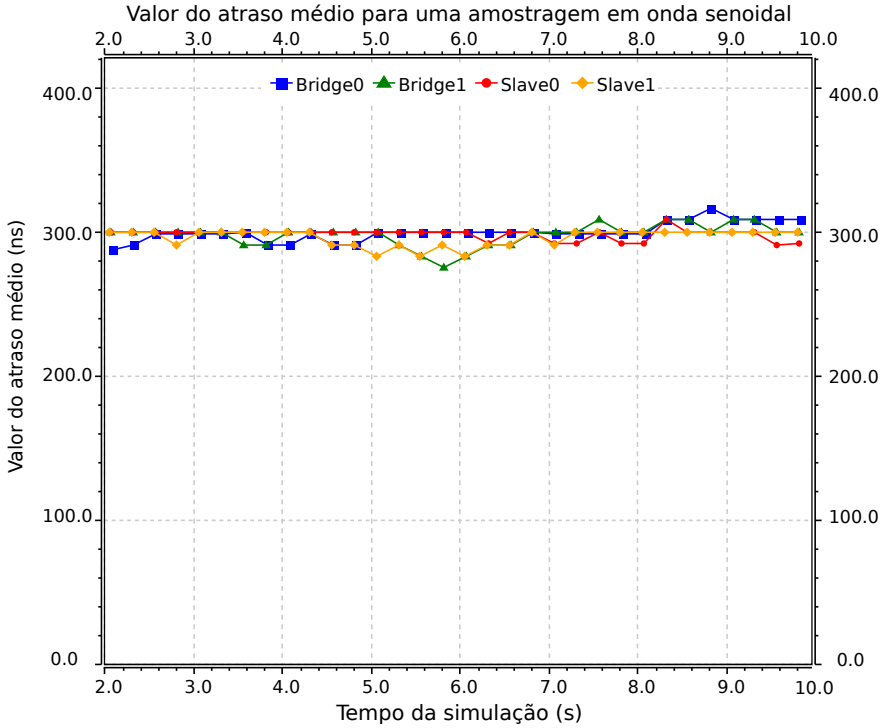
5.2 Resultados para deriva de relógios físicos não constante

O teste da deriva de relógio físico não constante busca evidenciar se o valor do atraso médio em cada componente da rede é afetado por uma deriva não constante. Agora, valores amostrados da função seno são utilizados para induzir os relógios à divergência. Ressalta-se ainda que o atraso entre os dispositivos continua sendo de 25 nanossegundos, e espera-se que o atraso médio oscile em torno deste valor.

5.2.1 Resultados do PTP

O resultado apresentado na Figura 18 considera a deriva dos relógios sendo amostrada de uma senoide com amplitude da ordem de 10^{-6}. As Tabelas 18 e 19 mostram os valores do atraso médio e erro.

Figura 18 – Atraso médio para amostragem de deriva em onda senoidal numa rede PTP.



Fonte: Autoria própria.

Novamente observa-se uma oscilação dos resultados em torno do valor de 300 nanossegundos.

Tabela 18 – Amostras dos valores do atraso médio e erro para os dispositivos *Bridge* numa rede PTP sob amostragem em senoide.

	Bridge0		Bridge1	
Instante da simulação (s)	**Atraso médio (ns)**	**Erro**	**Atraso médio (ns)**	**Erro**
3	299,6	1098,46%	300,4	1101,58%
6	300,4	1101,58%	283,6	1034,58%
9	308,2	1132,79%	308,0	1132,00%

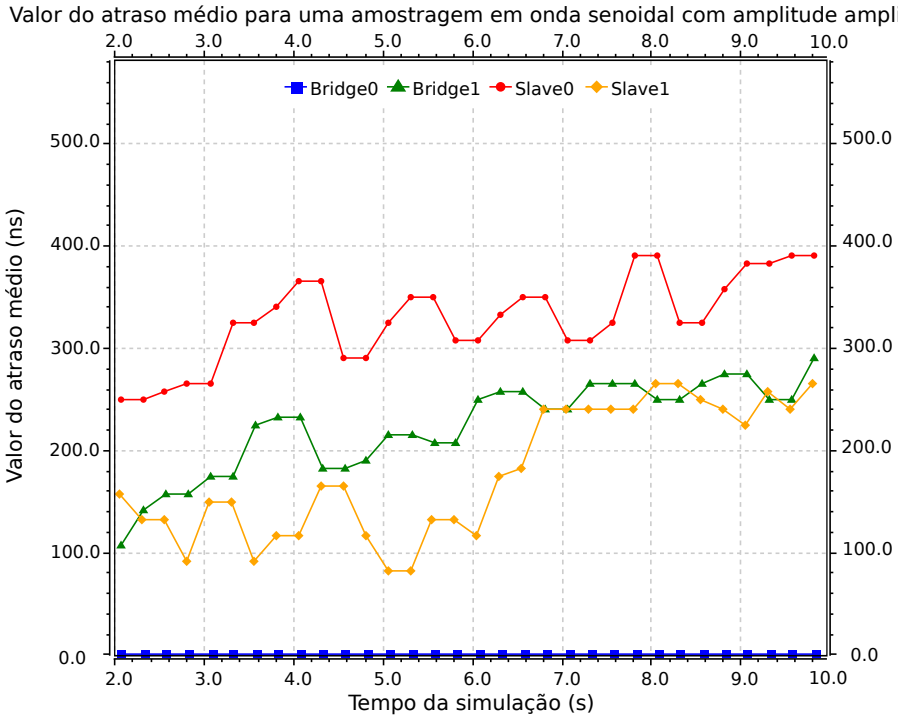
Tabela 19 – Amostras dos valores do atraso médio e erro para os dispositivos *Slave* numa rede PTP sob amostragem em senoide.

	Slave0		Slave1	
Instante da simulação (s)	**Atraso médio (ns)**	**Erro**	**Atraso médio (ns)**	**Erro**
3	300,4	1101,58%	300,0	1100,00%
6	300,4	1101,58%	283,7	1034,80%
9	300,4	1101,58%	300,0	1100,00%

Os valores do atraso médio e erro seguem a mesma tendência vista anteriormente, oscilam em torno do valor de 300 nanossegundos e possuem um grande erro quando comparados ao valor esperado de 25 nanossegundos. No entanto, é possível ver uma maior constância nas amostras obtidas. Na Tabela 7, por exemplo, os valores para o *Slave0* se mantiveram iguais em todas os casos.

O resultado apresentado na Figura 19 considera o mesmo cenário, mas agora com uma senoide com amplitude da ordem de 10^{-3}, ou seja, 1000 vezes maior do que a anterior. As Tabelas 20 e 21 mostram os valores do atraso médio e erro.

Figura 19 – Atraso médio para amostragem de deriva em onda senoidal com amplitude ampliada em uma rede PTP.



Fonte: Autoria própria.

Nesta simulação os valores não oscilam em conjunto em torno de 300 nanossegundos. Pelo contrário, eles divergem entre si. Os resultados levam em consideração a amplitude da senoide, ou seja, com a amplitude de cada amostra de deriva maior, as divergências entre os relógios são suficientes para deixar os valores de atraso médio dispersos. Além disso, pode-se observar que o dispositivo *Bridge0* possui seus valores nulos em todas as trocas de mensagens, o que pode ser influência da proximidade ao *Grandmaster*.

Tabela 20 – Amostras dos valores do atraso médio e erro para os dispositivos *Bridge* em uma rede PTP sob amostragem em senoide com amplitude ampliada.

	Bridge0		Bridge1	
Instante da simulação (s)	**Atraso médio (ns)**	**Erro**	**Atraso médio (ns)**	**Erro**
3	0,0	-%	175,9	603,77%
6	0,0	-%	250,2	900,82%
9	0,0	-%	275,3	1001,36%

Tabela 21 – Amostras dos valores do atraso médio e erro para os dispositivos *Slave* em uma rede PTP sob amostragem em senoide com amplitude ampliada.

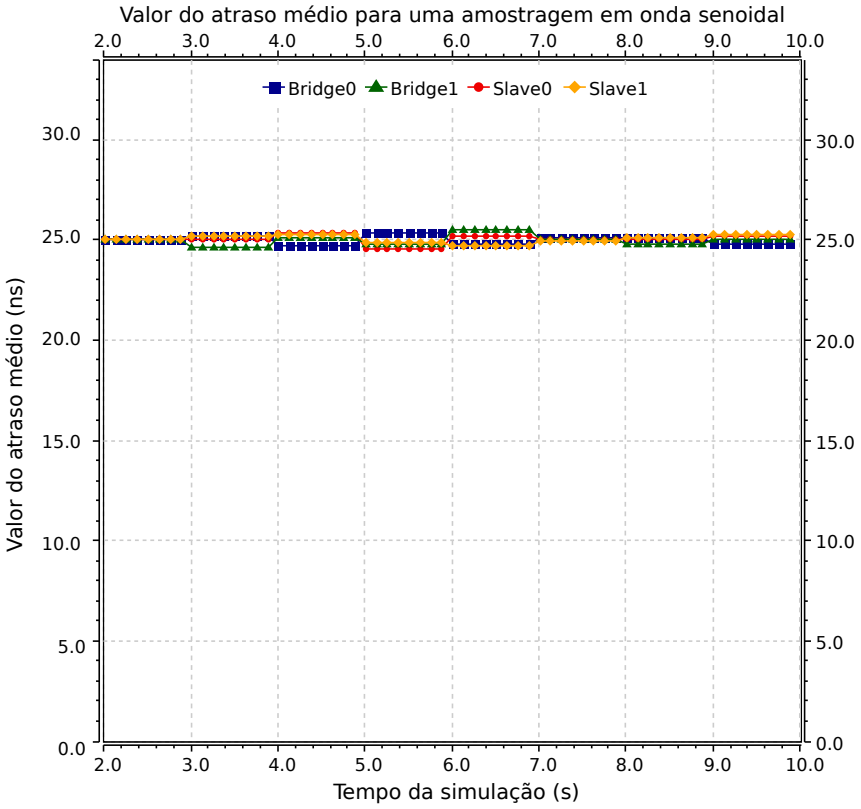
	Slave0		Slave1	
Instante da simulação (s)	**Atraso médio (ns)**	**Erro**	**Atraso médio (ns)**	**Erro**
3	266,2	964,80%	150,8	503,23%
6	308,5	1133,88%	116,5	366,13%
9	383,9	1435,50%	225,1	800,28%

As Tabelas 20 e 21 mostram que os valores dos erros oscilam da mesma forma vista graficamente, ora estão próximos do valor médio de 300 nanossegundos, ora gerando erros em torno de 1500%.

5.2.2 Resultados do gPTP

Essa seção apresenta os resultados do protocolo gPTP, considerando o mesmo cenário anterior. Os resultados são apresentados na Figura 20 e consideram uma amplitude senoidal da ordem 10^{-6}. As Tabelas 22 e 23 mostram os valores do atraso médio e erro.

Figura 20 – Atraso médio para amostragem de deriva em onda senoidal em uma rede gPTP.



Fonte: Autoria própria.

Graficamente é possível perceber que os valores oscilam entre o valor pré-configurado de 25 nanossegundos de forma a acompanhar um movimento senoidal gerado pela deriva dos relógios de cada unidade. Não existem picos ou valores distantes, todos mantém um padrão, mesmo quando fora do valor esperado.

Tabela 22 – Amostras dos valores do atraso médio e erro para os dispositivos *Bridge* em uma rede gPTP sob amostragem em senoide.

	Bridge0		Bridge1	
Instante da simulação (s)	**Atraso médio (ns)**	**Erro**	**Atraso médio (ns)**	**Erro**
3	25,18	0,73%	24,59	1,64%
6	24,75	0,99%	25,50	1,99%
9	24,74	1,03%	25,01	0,05%

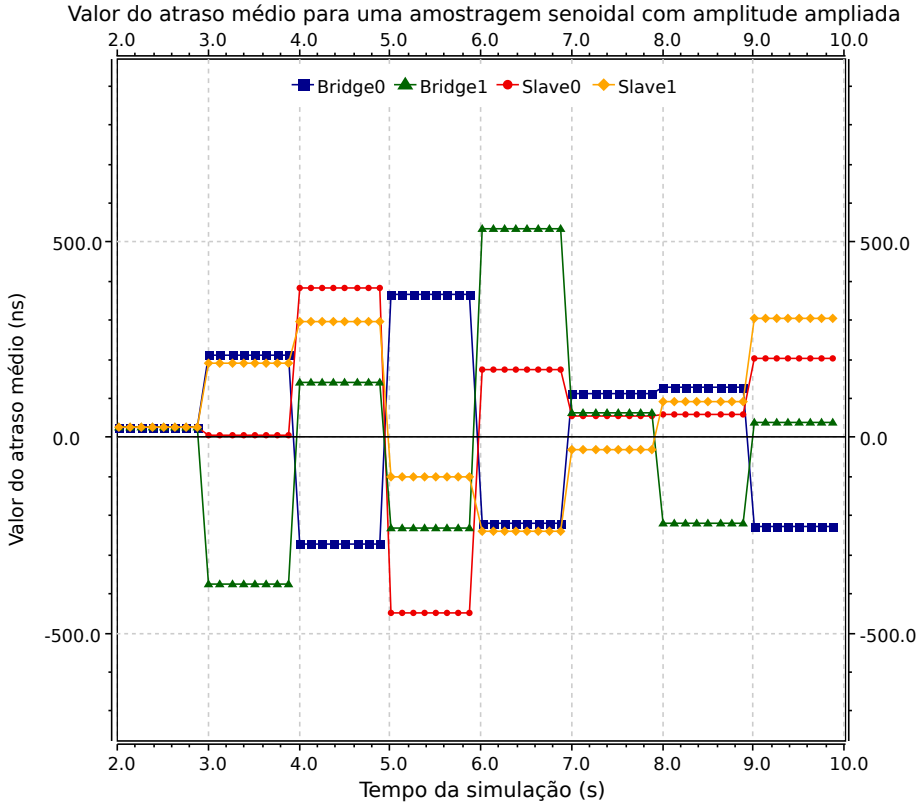
Tabela 23 – Amostras dos valores do atraso médio e erro para os dispositivos *Slave* em uma rede gPTP sob amostragem em senoide.

	Slave0		Slave1	
Instante da simulação (s)	**Atraso médio (ns)**	**Erro**	**Atraso médio (ns)**	**Erro**
3	24,98	0,07%	25,17	0,67%
6	25,15	0,61%	24,73	1,07%
9	25,18	0,72%	25,28	1,11%

As Tabelas 22 e 23 reforçam a análise gráfica. Os valores se mantém próximos do configurado previamente e o erro oscila em torno de um 1%.

O resultado apresentado na Figura 21 considera agora uma senoide com amplitude da ordem de 10^{-3}. As Tabelas 24 e 25 mostram os valores do atraso médio e erro.

Figura 21 – Atraso médio para amostragem de deriva em onda senoidal com amplitude ampliada em uma rede gPTP.



Fonte: Autoria própria.

Assim como observado nos resultados para o PTP, o gPTP apresentou a mesma oscilação de valores, o que dá a ideia da sensibilidade dos protocolos às amplitudes das derivas.

Graficamente na Figura 19 pode-se perceber a flutuação de valores que não mantém uma regularidade, independente da proximidade com o *GrandMaster*.

Tabela 24 – Amostras dos valores do atraso médio e erro para os dispositivos *Bridge* em uma rede gPTP sob amostragem em senoide com amplitude ampliada.

	Bridge0		Bridge1	
Instante da simulação (s)	**Atraso médio (ns)**	**Erro**	**Atraso médio (ns)**	**Erro**
3	209,8	739,39%	-376,4	1605,62%
6	-217,7	970,70%	534,3	2037,18%
9	-227,0	1008,13%	37,80	51,21%

Tabela 25 – Amostras dos valores do atraso médio e erro para os dispositivos *Slave* em uma rede gPTP sob amostragem em senoide com amplitude ampliada.

	Slave0		Slave1	
Instante da simulação (s)	**Atraso médio (ns)**	**Erro**	**Atraso médio (ns)**	**Erro**
3	7,829	68,68%	190,6E	662,50%
6	174,9	599,71%	-241,3	1065,16%
9	204,2	716,93%	303,6	1114,57%

As Tabelas 24 e 25 reforçam a ideia vista graficamente. Os erros e valores não mantém proximidade com os 25 nanossegundos esperados e divergem entre si, como observado pelos valores dos erros indo, aproximadamente, de 60% até 2000%.

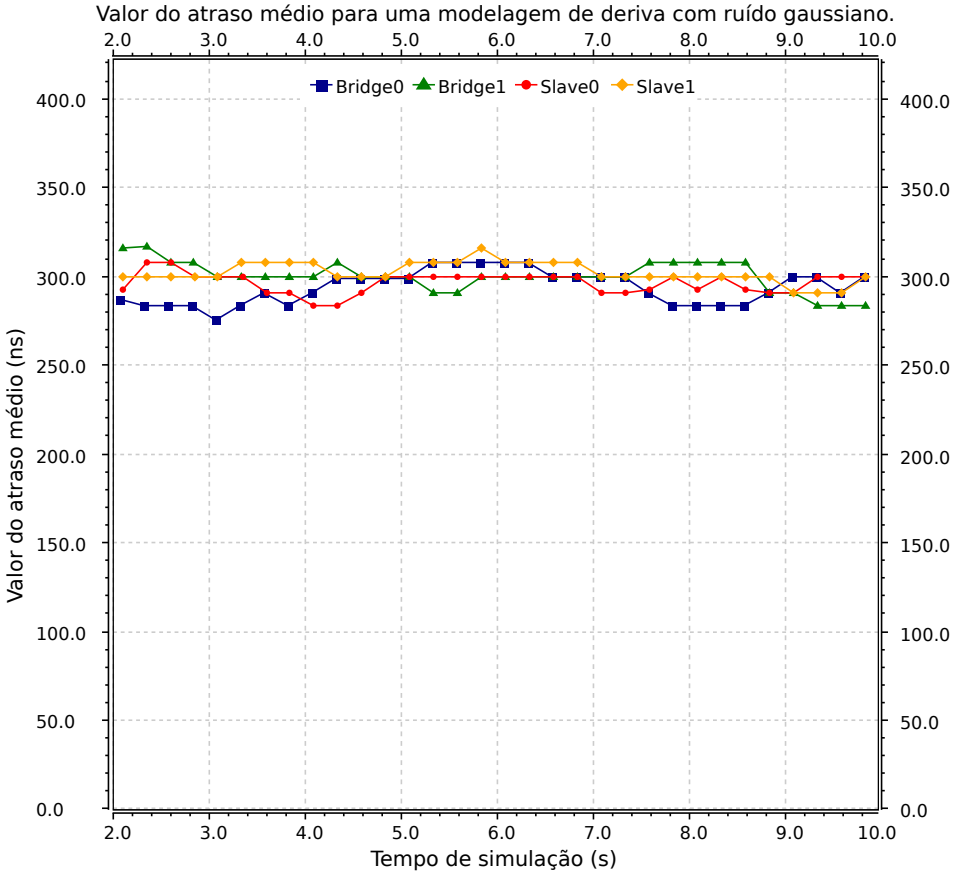
5.3 Resultados para deriva de relógios físicos com ruído gaussiano

Este teste busca evidenciar se a sincronização é afetada pela variação aleatória e ruidosa da deriva nos relógios de cada dispositivo. Ressalta-se ainda que o atraso entre os dispositivos é de 25 nanossegundos e espera-se que o atraso médio oscile em torno deste valor.

5.3.1 Resultados do PTP

Os resultados do PTP são apresentados na Figura 22 e consideram uma distribuição de erro gaussiana. As Tabelas 26 e 27 mostram os valores do atraso médio e erro para algumas amostras.

Figura 22 – Atraso médio para deriva baseada em ruído gaussiano em uma rede PTP.



Fonte: Autoria própria.

Na Figura 22 é possível perceber que os valores continuam oscilando ao redor dos 300 nanossegundos, mas não existem picos de alta amplitude. As Tabelas 26 e 27 acompanham o comportamento do gráfico da Figura 22.

Tabela 26 – Amostras dos valores do atraso médio e erro para os dispositivos *Bridge* em uma rede PTP sob modelagem de deriva com ruído gaussiano.

	Bridge0		**Bridge1**	
Instante da simulação (s)	**Atraso médio (ns)**	**Erro**	**Atraso médio (ns)**	**Erro**
3	275,1	1000,28%	300,7	1102,61%
6	308,4	1133,55%	300,7	1102,61%
9	300,4	1101,48%	291,1	1064,33%

Tabela 27 – Amostras dos valores do atraso médio e erro para os dispositivos *Slave* em uma rede PTP sob modelagem de deriva com ruído gaussiano.

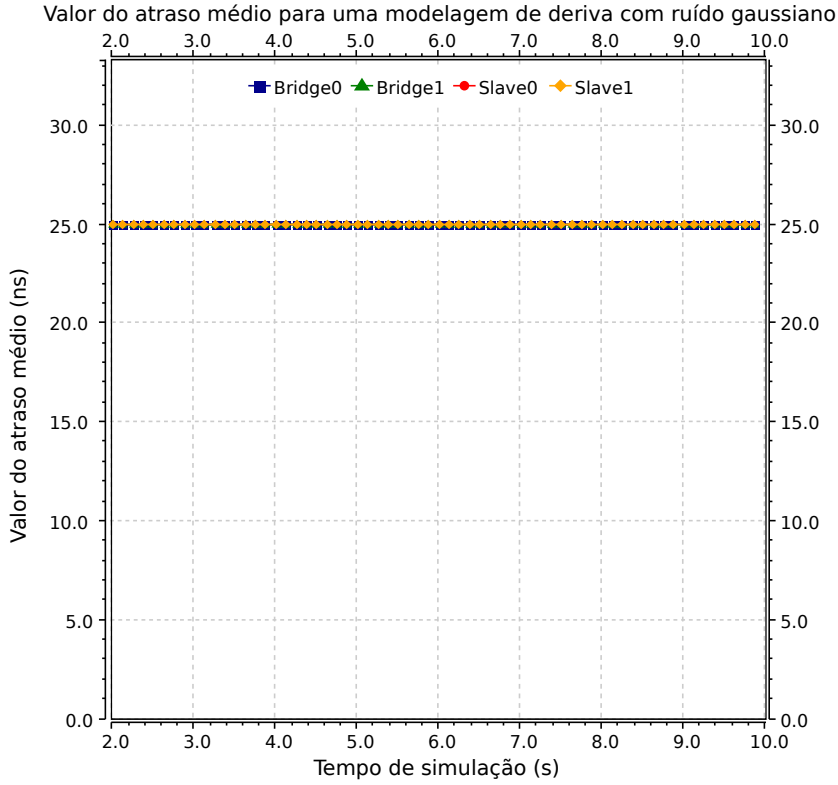
	Slave0		**Slave1**	
Instante da simulação (s)	**Atraso médio (ns)**	**Erro**	**Atraso médio (ns)**	**Erro**
3	300,7	1102,61%	301,3	1105,17%
6	300,7	1102,61%	308,4	1133,79%
9	291,3	1065,36%	291,8	1067,01%

Os erros e valores não mantém proximidade com os 25 nanossegundos esperados, mas mantém um erro próximo quando comparados entre si.

5.3.2 Resultados do gPTP

Os resultados para o gPTP são apresentados na Figura 23 e consideram uma distribuição de erro gaussiana. As Tabelas 28 e 29 mostram os valores do atraso médio e erro para algumas amostras.

Figura 23 – Atraso médio para deriva baseada em ruído gaussiano em uma rede gPTP.



Fonte: Autoria própria.

Visualmente na Figura 22 é possível verificar uma estabilidade exclusiva do gPTP para precisões de dezenas de nanossegundos. Os valores de atraso médio quase se sobrescrevem no gráfico e segue constante até o fim da simulação.

Tabela 28 – Amostras dos valores do atraso médio e erro para os dispositivos *Bridge* em uma rede gPTP sob modelagem de deriva com ruído gaussiano.

	Bridge0		**Bridge1**	
Instante da simulação (s)	**Atraso médio (ns)**	**Erro**	**Atraso médio (ns)**	**Erro**
3	24,99998	0,000072%	25,00019	0,000068%
6	25,00007	0,000276%	24,99997	0,000044%
9	24,99995	0,000204%	24,99980	0,000400%

Tabela 29 – Amostras dos valores do atraso médio e erro para os dispositivos *Slave* em uma rede gPTP sob modelagem de deriva com ruído gaussiano.

	Slave0		**Slave1**	
Instante da simulação (s)	**Atraso médio (ns)**	**Erro**	**Atraso médio (ns)**	**Erro**
3	24,99998	0,000068%	24,99975	0,001008%
6	25,00001	0,000044%	24,99985	0,000612%
9	24,99990	0,000400%	25,00021	0,000856%

O comportamento visto na Figura 23 é confirmado com as Tabelas 28 e 27. O erro é tão pequeno que foi necessário expandir a escala nas casas decimais dos valores para ter a noção da sua magnitude. Além disso, todos os valores de atraso médio possuem mais casas decimais em relação às outras tabelas também pelo mesmo motivo, que é o de evidenciar a estabilidade do gPTP.

5.4 Discussão de resultados

Sobre os resultados apresentados pode-se dizer que o PTP possui como funcionalidade a precisão de submicrossegundos e esta foi alcançada, apesar da alta flutuação dos valores e dos altos valores de erro em todos os cenários de teste. Todavia, é de se compreender, pois este protocolo não foi idealizado para alcançar a casa das dezenas de nanossegundos e isso foi um dos requisitos postos com o valor de 25 nanossegundos como atraso entre dispositivos.

Quanto ao desempenho do PTP sob variação do tráfego de mensagens na rede, pode-se observar que o número de mensagens não afetou o funcionamento do protocolo, o qual manteve todas as suas unidades oscilando em torno dos 300 nanossegundos de atraso médio. Com o valor base configurado como 25 nanossegundos, ele não foi declarado sincronizado no tempo de simulação, mas alterando o atraso para 300 nanossegundos, o PTP conseguiu manter a maioria de seus valores abaixo de 5%, como apresentado no Apêndice A.

Sobre seu desempenho sob derivas não lineares, quando tratando de menores amplitudes e quando considerado um atraso de 300 nanossegundos, é possível considerar que o protocolo manteve-se sincronizado, como pode ser visto nos resultados no Apêndice A. No entanto, com a configuração mantida em 25 nanossegundos, é possível afirmar que ele falhou para ambos os casos. Seu desempenho no ambiente com modelagem de deriva com ruído gaussiano apresentou o mesmo comportamento. Estabilizou-se em torno de 300 nanossegundos, mas como esperado não chegou aos 25 nanossegundos.

Já o gPTP garantiu sua funcionalidade de alcançar os nanossegundos de precisão durante os testes e se mostrou mais estável durante as medições. Além disso, no ambiente de simulação foi verificado que a inexistência de dispositivos auxiliares de *hardware*, pré-requisitos para uma rede gPTP, não são necessários para alcançar a precisão prometida dentro da simulação. A submissão do gPTP às derivas lineares mostrou que o protocolo permaneceu sincronizado até os segundos finais de simulação, no instante em que o erro passa de 5% pode-se entender que a amplitude da deriva é significativa para gerar divergência entre os dispositivos. No entanto, para funções senoidais teve um resultado interessante. Todas as amostras foram obtidas com erros abaixo de 5% e a referência visual mostrou um comportamento estável, quando comparado ao PTP. No entanto, os erros foram quase nulos quando o gPTP foi testado com a modelagem de deriva com erro gaussiano. É importante ressaltar a diferença de amplitude entre os testes, mas ainda assim a amplitude no erro gaussiano foi de

até 8% do valor de atraso de 25 nanossegundos. Deste modo, o protocolo reduziu todos os valores do ruído e manteve os erros amostrados próximos de zero.

Apesar dos resultados particulares de cada protocolo, ambos foram suscetíveis à variações bruscas nas derivas dos relógios, ainda mais considerando altas amplitudes. É importante ressaltar que não é esperada uma variação brusca em um ruído de fundo de um cristal de quartzo, mas submeter os protocolos à essas variações foi importante para entender os seus comportamentos.

6 CONCLUSÕES

Seja no desenvolvimento audiovisual ou no de carros autônomos, sistemas distribuídos síncronos já são e serão cada vez mais utilizados no nosso cotidiano. Com isso, seus requisitos específicos de tempo devem ter garantia de cumprimento e somente protocolos bem definidos podem satisfazer essas regras. O *IEEE 1588-2008 Precision Time Protocol* (PTP) e *IEEE 802.1AS - Generic Precision Time Protocol* (gPTP), buscam solucionar essas especificidades com precisão de submicrossegundos. Para auxiliar na construção dessas soluções, é crucial possuir ambientes de simulação robusto, como o *OMNet++*, e bibliotecas desses protocolos, como a *libPTP* e a *libgPTP*.

Este trabalho analisou o desempenho do PTP e gPTP sob funcionamento no OMNet++, considerando variação no fluxo de mensagens na rede e também derivas de relógios lineares, não constantes e modelagens. As arquiteturas foram construídas buscando estabelecer a maior semelhança possível entre as redes propostas, e com isso, maior fidelidade na comparação. A métrica para averiguar o comportamento dos protocolos foi o valor do atraso médio, o qual é fundamental em ambos na garantia da sincronização.

Apesar do sucesso na comparação, foi evidenciado que o *OMNet++* não possui uma retrocompatibilidade estabelecida e nota-se que é extremamente suscetível à versões de funcionamento. Após tentativas frustradas de garantir a equidade de versão no ambiente, este foi o fator chave na decisão de utilizar diferentes alternativas para garantir o funcionamento dos protocolos e executá-los em versões não idênticas.

Contribuições foram geradas ao submeter a implementação do gPTP a derivas de relógios não lineares, como evidenciado no Capítulo 2. A implementação se mostrou estável e conseguiu alcançar resultados mais promissores quando submetida ao ruído gaussiano do que quando testada com a deriva linear. Além disso, a análise de desempenho contribuiu para auxiliar futuros projetos na escolha de qual dos protocolos é o melhor para cada situação. Ao optar pelo PTP, entende-se que não é necessário uma precisão na casa de nanossegundos e também que os componentes físicos não precisarão ser adaptados. Já o gPTP possui o bônus de ter uma precisão superior, mas tem seu ônus ao exigir dispositivos específicos ao longo da rede.

O objetivo de analisar o desempenho de precisão dos protocolos PTP e gPTP sob funcionamento no simulador OMNet++, sob condições de variação de tráfego na rede, e com variação do método de deriva nos relógios internos de cada dispositivo foi alcançado com sucesso. Apesar da simplicidade nas arquiteturas e medições, encerra-se um trabalho relevante que pode gerar material de base para outros trabalhos. Além disso, pôde-se contribuir com a análise da deriva de relógios físicos em simuladores, analisando fatores como taxas de amostragem e amplitudes. Sem dúvida, o trabalho evidenciou as diferenças de desempenho entre os protocolos e isso contribuirá para escolhas de desenvolvedores em futuros projetos, como dito.

6.1 Trabalhos futuros

Inicialmente, seria importante e relevante garantir que ambos os protocolos funcionem sob as versões mais recentes do simulador *OMNet++* e da biblioteca de desenvolvimento *INET Framework*. Isso, além de ser uma contribuição enorme para ambos os projetos, garante que eles possam ser usados futuramente em novos e recentes trabalhos. Vale ressaltar que, caso implementada com sucesso essa modificação, testes de regressão devem ser realizados para aferir o correto funcionamento dos protocolos. Esses testes consistem em realizar novas medições e comparar com resultados antigos, a fim de checar se as atualizações não afetaram o pleno funcionamento do código.

Modelagens de deriva específicas para cristais já existem e possuem bibliotecas implementadas, como é o caso da *libPLN* vista em (LIBPLN, 2016). Esta biblioteca faz parte do projeto *ptp-sim* e implementou as modelagens de distúrbios que geram deriva em cristais de quartzo, por exemplo. A intenção inicial deste trabalho era adicionar a *libPLN* na geração da modelagem de deriva para a *libgPTP*. No entanto, os métodos de obtenção dos valores de deriva dos relógios são diferentes e não foi possível, a priori, realizar a inserção da biblioteca sem realizar profundas mudanças na arquitetura da biblioteca *libgPTP*. No entanto, foi possível fazer a *libPLN* funcionar na mesma versão de OMNet++ e INET que a *libgPTP*, fator que gera facilidade na integração dos projetos.

No presente trabalho 10 execuções foram realizadas para cada teste de desempenho, mas os resultados se mantiveram. Esse comportamento pode ser explicado pelo fato da arquitetura da rede ser de baixa complexidade. Para evitar esse comportamento pode-se fazer o uso de topologias utilizadas na literatura como a *Daisy chain*, a qual usa um barramento comum e uma série de dispositivos trocando mensagens PTP por meio deste barramento. Como trabalhos futuros, pretende-se executar os mesmos testes em redes com mais dispositivos, envolvendo tráfego de mensagens não relacionadas à conexão de sincronização, fazendo com que haja um aumento no tráfego entre os dispositivos. Uma rede mais complexa e sob influência da variação de tráfego geraria resultados e análises interessantes.

REFERÊNCIAS

ASHJAEI, M. *et al.* Time-sensitive networking in automotive embedded systems: State of the art and research opportunities. **Journal of Systems Architecture**, v. 117, p. 102137, 2021. ISSN 1383-7621. Disponível em: https://www.sciencedirect.com/science/article/pii/S1383762121001028.

ATAHAR HARIDASULA, H. **Conceitos básicos de PTP e SyncE com a configuração do Cisco IOS XR**. 2021. Disponível em: https://www.cisco.com/c/pt_br/support/docs/ios-nx-os-software/ios-xr-software/217579-configure-ptp-and-synce-basics-with-cisc.html. Acesso em: 20 jan. 2022.

COULOURIS, G.; DOLLIMORE, J.; KINDBERG, T. **Distributed Systems: Concepts and Design (International Computer Science)**. 4th rev. ed.. ed. [*S.l.*]: Addison-Wesley Longman, Amsterdam, 2005. ISBN 0321263545.

GUSELLA, R.; ZATTI, S. The accuracy of the clock synchronization achieved by tempo in berkeley unix 4.3bsd. **IEEE Transactions on Software Engineering**, v. 15, n. 7, p. 847–853, 1989.

IEEE P802.1AS-Rev. Ieee draft standard for local and metropolitan area networks - timing and synchronization for time-sensitive applications. p. 1–496, 2018.

IEEE Std 1588-2002. IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems. p. 1–154, 2002.

IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002). IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems. p. 1–269, 2008.

IEEE Std 802.1AS-2011. IEEE Standard for Local and Metropolitan Area Networks - Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks. p. 1–292, 2011.

IEEE Std 802.3u. Ieee standards for local and metropolitan area networks: Supplement - media access control (mac) parameters, physical layer, medium attachment units, and repeater for 100mb/s operation, type 100base-t (clauses 21-30). **IEEE Std 802.3u-1995 (Supplement to ISO/IEC 8802-3: 1993; ANSI/IEEE Std 802.3, 1993 Edition)**, p. 1–415, 1995.

ISO 8601-1. **Date and time — Representations for information interchange — Part 1: Basic rules**. Geneva, CH, 2019. Disponível em: https://www.iso.org/standard/70907.html.

ISO/IEC 9899. **Information technology — Programming languages — C**. Geneva, CH, 2018. Disponível em: https://www.iso.org/standard/74528.html.

JANCHIVNYAMBUU, E.; PUTTNIES, H.; DANIELIS, P. **IEEE 802.1AS gPTP for Clock Synchronization**. 2018. Disponível em: https://gitlab.amd.e-technik.uni-rostock.de/peter.danielis/gptp-implementation. Acesso em: 30 out. 2021.

JUNIPER. **Configuring Sync Interval (PTP)**. 2013. Disponível em: https://www.juniper.net/documentation/en_US/junos/topics/reference/configuration-statement/sync-interval-edit-protocols-ptp-slave.html. Acesso em: 21 abr. 2022.

JUNIPER. **Configuring PTP Grandmaster Clock Using External GPS Receiver**. 2017. Disponível em: https://www.juniper.net/documentation/en_US/junos/topics/task/configuration/ptp-gm-clock-acx-series.html. Acesso em: 06 mai. 2022.

KOVÁCSHÁZY, T.; FERENCZ, B. Performance evaluation of ptpd, a ieee 1588 implementation, on the x86 linux platform for typical application scenarios. **2012 IEEE International Instrumentation and Measurement Technology Conference Proceedings**, p. 2548–2552, 2012.

LIBPLN. 2016. Disponível em: https://github.com/ptp-sim/libPLN. Acesso em: 30 out. 2021.

LIBPTP. 2016. Disponível em: https://github.com/ptp-sim/libPTP. Acesso em: 30 out. 2021.

LIM, H.-T. *et al.* Ieee 802.1as time synchronization in a switched ethernet based in-car network. p. 147–154, 2011.

MALVINO, A.; BATES, D. **Eletrônica - Vol.2: 8ª Edição**. McGraw Hill Brasil, 2016. ISBN 9788580555936. Disponível em: https://books.google.com.br/books?id=BbA0DQAAQBAJ.

MCCLANING, K.; VITO, T. **Radio Receiver Design**. Noble Publishing Corporation, 2000. (EngineeringPro collection). ISBN 9781884932076. Disponível em: https://books.google.com.br/books?id=6hEfAQAAIAAJ.

MILLS, D. **Computer Network Time Synchronization: The Network Time Protocol**. CRC Press, 2006. ISBN 9781420006155. Disponível em: https://books.google.com.br/books?id=pdTcJBfnbq8C.

OMNET++. 2022. Disponível em: https://omnetpp.org/. Acesso em: 03 mar. 2022.

OMNETPP. **Configuring Gptp module**. 2022. Disponível em: https://doc.omnetpp.org/inet/api-current/neddoc/inet.linklayer.ieee8021as.Gptp.html. Acesso em: 21 abr. 2022.

PTP-SIM. 2016. Disponível em: https://ptp-sim.github.io/. Acesso em: 30 out. 2021.

PUTTNIES, H. *et al.* A simulation model of ieee 802.1as gptp for clock synchronization in omnet++. EasyChair, v. 56, p. 63–72, 2018. ISSN 2398-7340. Disponível em: https://easychair.org/publications/paper/Q4kL. Acesso em: 15 out. 2021.

TECH, N. **Specifying the PPS In-Synchronization Limit**. 2022. Disponível em: https://docs.napatech.com/r/Time-Stamping-and-Time-Synchronization/Configuring-the-PTP-Clock. Acesso em: 21 abr. 2022.

ZIMMERMANN, H. OSI reference model-the ISO model of architecture for open systems interconnection. **IEEE Trans. Communication (USA)**, COM-28, n. 4, p. 425–432, abr. 1980. IRIA/Lab., Rocquencourt, France.

APÊNDICE A – Dados da simulação feita com o PTP considerando um atraso entre as conexões de 0,3 microssegundos.

Tabela 30 – Amostras dos valores do atraso médio e erro para os dispositivos *Bridge* numa rede PTP sob SyncInterval de 0,5 segundos.

	Bridge0		Bridge1	
Instante da simulação (s)	**Atraso médio (s)**	**Erro**	**Atraso médio (s)**	**Erro**
1	2,913E-07	2,91%	2,836E-07	5,45%
3	3,080E-07	2,67%	3,004E-07	0,13%
6	3,080E-07	2,67%	2,920E-07	2,66%
9	2,836E-07	5,45%	3,080E-07	2,67%

Tabela 31 – Amostras dos valores do atraso médio e erro para os dispositivos *Slave* numa rede PTP sob SyncInterval de 0,5 segundos.

	Slave0		Slave1	
Instante da simulação (s)	**Atraso médio (s)**	**Erro**	**Atraso médio (s)**	**Erro**
1	2,912E-07	2,94%	3,004E-07	0,13%
3	3,006E-07	0,19%	3,004E-07	0,13%
6	3,084E-07	2,80%	2,913E-07	2,91%
9	3,084E-07	2,80%	3,080E-07	2,67%

Tabela 32 – Amostras dos valores do atraso médio e erro para os dispositivos *Bridge* numa rede PTP sob SyncInterval de 0,250 segundos.

	Bridge0		Bridge1	
Instante da simulação (s)	**Atraso médio (s)**	**Erro**	**Atraso médio (s)**	**Erro**
1	2,913E-07	2,91%	2,755E-07	8,16%
3	2,996E-07	0,12%	3,006E-07	0,19%
6	2,996E-07	0,12%	3,006E-07	0,19%
9	2,996E-07	0,12%	3,084E-07	2,80%

Tabela 33 – Amostras dos valores do atraso médio e erro para os dispositivos *Slave* numa rede PTP sob SyncInterval de 0,250 segundos.

	Slave0		Slave1	
Instante da simulação (s)	**Atraso médio (s)**	**Erro**	**Atraso médio (s)**	**Erro**
1	3,004E-07	0,13%	3,004E-07	0,13%
3	3,004E-07	0,13%	2,913E-07	2,91%
6	2,920E-07	2,66%	3,004E-07	0,13%
9	3,080E-07	2,67%	3,004E-07	0,13%

Tabela 34 – Amostras dos valores do atraso médio e erro para os dispositivos *Bridge* numa rede PTP sob SyncInterval de 0,125 segundos.

	Bridge0		Bridge1	
Instante da simulação (s)	**Atraso médio (s)**	**Erro**	**Atraso médio (s)**	**Erro**
1	2,832E-07	5,59%	2,834E-07	5,52%
3	3,082E-07	2,73%	2,927E-07	2,42%
6	2,910E-07	2,99%	3,084E-07	2,80%
9	2,996E-07	0,13%	3,006E-07	0,19%

Tabela 35 – Amostras dos valores do atraso médio e erro para os dispositivos *Slave* numa rede PTP sob SyncInterval de 0,125 segundos.

	Slave0		Slave1	
Instante da simulação (s)	**Atraso médio (s)**	**Erro**	**Atraso médio (s)**	**Erro**
1	2,912E-07	2,94%	3,004E-07	0,13%
3	3,006E-07	0,19%	3,004E-07	0,13%
6	2,927E-07	2,42%	3,004E-07	0,13%
9	3,178E-07	5,93%	3,004E-07	0,13%

Tabela 36 – Amostras dos valores do atraso médio e erro para os dispositivos *Bridge* numa rede PTP sob amostragem em senoide.

	Bridge0		Bridge1	
Instante da simulação (s)	**Atraso médio (s)**	**Erro**	**Atraso médio (s)**	**Erro**
1	2,910E-07	2,99%	3,004E-07	0,13%
3	2,996E-07	0,13%	3,004E-07	0,13%
6	3,004E-07	0,13%	2,836E-07	5,45%
9	3,082E-07	2,73%	3,080E-07	2,67%

Tabela 37 – Amostras dos valores do atraso médio e erro para os dispositivos *Slave* numa rede PTP sob amostragem em senoide.

	Slave0		Slave1	
Instante da simulação (s)	**Atraso médio (s)**	**Erro**	**Atraso médio (s)**	**Erro**
1	3,004E-07	0,13%	3,000E-07	0,00%
3	3,004E-07	0,13%	3,000E-07	0,00%
6	3,004E-07	0,13%	2,837E-07	5,43%
9	3,004E-07	0,13%	3,000E-07	0,00%

Tabela 38 – Amostras dos valores do atraso médio e erro para os dispositivos *Bridge* numa rede PTP sob amostragem em senoide com amplitude ampliada.

	Bridge0		Bridge1	
Instante da simulação (s)	**Atraso médio (s)**	**Erro**	**Atraso médio (s)**	**Erro**
1	2,913E-07	2,89%	6,284E-08	79,05%
3	0,000E+00	-%	1,759E-07	41,35%
6	0,000E+00	-%	2,502E-07	16,60%
9	0,000E+00	-%	2,753E-07	8,22%

Tabela 39 – Amostras dos valores do atraso médio e erro para os dispositivos *Bridge* numa rede PTP sob amostragem em senoide com amplitude ampliada.

	Slave0		Slave1	
Instante da simulação (s)	**Atraso médio (s)**	**Erro**	**Atraso médio (s)**	**Erro**
1	4,376E-07	45,86%	2,753E-07	8,22%
3	2,662E-07	11,27%	1,508E-07	49,73%
6	3,085E-07	2,82%	1,165E-07	61,16%
9	3,839E-07	27,96%	2,251E-07	24,98%

Tabela 40 – Amostras dos valores do atraso médio e erro para os dispositivos *Bridge* numa rede PTP sob modelagem de deriva com ruído gaussiano.

	Bridge0		Bridge1	
Instante da simulação (s)	**Atraso médio (s)**	**Erro**	**Atraso médio (s)**	**Erro**
1	2,913E-07	2,89%	3,007E-07	0,22%
3	2,751E-07	8,31%	3,007E-07	0,22%
6	3,084E-07	2,80%	3,007E-07	0,22%
9	3,004E-07	0,12%	2,913E-07	2,97%

Tabela 41 – Amostras dos valores do atraso médio e erro para os dispositivos *Slave* numa rede PTP sob modelagem de deriva com ruído gaussiano.

	Slave0		Slave1	
Instante da simulação (s)	**Atraso médio (s)**	**Erro**	**Atraso médio (s)**	**Erro**
1	2,836E-07	5,47%	2,918E-07	2,75%
3	3,007E-07	0,22%	3,013E-07	3,01%
6	3,007E-07	0,22%	3,084E-07	2,82%
9	2,913E-07	2,89%	2,918E-07	2,75%

APÊNDICE B – Erros e soluções na tentativa de fazer a libPTP funcionar nas versões mais recentes do OMNET

#1

Erro:
Description Resource Path Location Type
imported NED type not found: ’inet.nodes.ethernet.Eth1G’Cables.ned
/libPTP/src/Components/Cables line 32 NED Consistency Problem

Solução:
import inet.node.ethernet.Eth1G;
Remover o "s" do "node".

#2

Description Resource Path Location Type
imported NED type not found: ’inet.nodes.ethernet.Eth1G’PTP_NIC.ned
/libPTP/src/Hardware line 47 NED Consistency Problem

Solução:

import inet.node.ethernet.Eth1G;
Remover o "s" do "node".

#3

Description Resource Path Location Type
imported NED type not found: ’inet.linklayer.IEtherMAC’IPTP_MAC.ned
/libPTP/src/Hardware/PTP_MAC line 25 NED Consistency Problem

Solução:

Mudança de caminho de:
import inet.linklayer.IEtherMAC;

para:
import inet.linklayer.contract.IEtherMAC;

#4

Description Resource Path Location Type
imported NED type not found:
’inet.linklayer.IMACRelayUnit’IPTP_MACRelayUnit.ned
/libPTP/src/Hardware/PTP_RelayUnit line 25 NED Consistency Problem

Mudança de caminho de:
import inet.linklayer.IMACRelayUnit;
para:
import inet.linklayer.contract.IMACRelayUnit;

#5

Description Resource Path Location Type
imported NED type not found: ’inet.linklayer.IMACAddressTable’PTP_NIC.ned
/libPTP/src/Hardware line 51 NED Consistency Problem

Mudança de caminho:
inet.linklayer.IMACAddressTable
para:
import inet.linklayer.contract.IMACAddressTable;

#6

Description Resource Path Location Type
’NotificationBoard’: no such module type PtpMacSink.ned
/libPTP/src/Testbenches/PtpMacSink line 56 NED Consistency Problem

Description Resource Path Location Type
’NotificationBoard’: no such module type PtpMacSource.ned
/libPTP/src/Testbenches/PtpMacSource line 58 NED Consistency Problem

Description Resource Path Location Type
imported NED type not found: ’inet.base.NotificationBoard’PTP_NIC.ned
/libPTP/src/Hardware line 50 NED Consistency Problem

Solução:

https://github.com/inet-framework/inet/blob/v3.0.0/WHATSNEW

```
 - Removed NotificationBoard and INotifiable:
 all modules now use OMNeT++ signals for
 publish/subscribe communication.
```

Podemos remover essa parte do código.

#7
Description Resource Path Location Type
imported NED type not found: 'inet.base.IHook'DelayQueue.ned
/libPTP/src/Hardware/DualDelayer/DelayQueue line 25
NED Consistency Problem

Mudança de caminho:
de import inet.base.IHook;
para:
import inet.common.IHook;

#8

Description Resource Path Location Type
type name 'simtime_t' is a reserved word PtpPortConfig.msg
/libPTP/src/Hardware/PTP_NIC_Ctrl/PTP_Config_Msg line 36 C/C++ Problem

Description Resource Path Location Type
type name 'simtime_t' is a reserved word PTPv2.msg
/libPTP/src/Software/PTP_Stack/PTP_Messages line 62 C/C++ Problem

Solução:
remover class nonobject

#9

Description Resource Path Location Type
'cMethodCallContextSwitcher' was not declared in this scope
CallableModule.cc /OMNeT_Utils/src/Callable line 63 C/C++ Problem

Description Resource Path Location Type

‘cMethodCallContextSwitcher’ was not declared in this scope
CallableModule.h /OMNeT_Utils/src/Callable line 53 C/C++ Problem

Description Resource Path Location Type
‘cRuntimeError’ was not declared in this scope CallableSubmodule.cc
/OMNeT_Utils/src/Callable line 63 C/C++ Problem

Description Resource Path Location Type
‘cChannel’ has not been declared VolatileDelayChannel.cc
/OMNeT_Utils/src/Channels/VolatileDelayChannel line 38 C/C++ Problem

Description Resource Path Location Type
‘cMessage’ has not been declared VolatileDelayChannel.h
/OMNeT_Utils/src/Channels/VolatileDelayChannel line 51 C/C++ Problem

Description Resource Path Location Type
‘simsignal_t’ does not name a type DynamicSignals.cc
/OMNeT_Utils/src/DynamicSignals line 58 C/C++ Problem

Description Resource Path Location Type
‘simsignal_t’ does not name a type DynamicSignals.h
/OMNeT_Utils/src/DynamicSignals line 48 C/C++ Problem

Description Resource Path Location Type
‘cRuntimeError’ was not declared in this scope IInitBase.cc
/OMNeT_Utils/src/InitBase line 82 C/C++ Problem

Description Resource Path Location Type
‘class cSubmoduleInitBase’ has no member named ‘pParentModule’
SubmoduleInitBase.cc /OMNeT_Utils/src/InitBase line 61 C/C++ Problem

Description Resource Path Location Type
‘cModule’ does not name a type SubmoduleInitBase.h
/OMNeT_Utils/src/InitBase line 54 C/C++ Problem

Adicionar nos códigos:
using namespace omnetpp;

#10

Description Resource Path Location Type
‘ev’ was not declared in this scope DynamicSignals.cc
/OMNeT_Utils/src/DynamicSignals line 67 C/C++ Problem

Solução:
https://groups.google.com/g/omnetpp/c/gl4zB1_kMq0

Mudança de ’ev’ para ’getEnvir()’.

11

Description Resource Path Location Type
fatal error: CallableModule.h: No such file or directory ModuleInitBase.h
/OMNeT_Utils/src/InitBase line 37 C/C++ Problem

Solução:
Trocar include por
#include "../Callable/CallableModule.h"