Vol.2019-ARC-235 No.2
Vol.2019-SLDM-187 No.2
Vol.2019-EMB-50 No.2
2019/3/17

車載ネットワークに対する時刻同期規格 IEEE 1588 のシミュレーション評価

陳希文[†1]　松原豊[†1]　山崎康広[†2]　本谷謙治[†2]
梶尾和弘[†2]　岩切英之[†2]　高田広章[†1]

概要：自動運転や高度運転支援システム（ADAS）などのタイムクリティカル機能の実現に向けて，車載 Ethernet へ時刻同期規格 IEEE 1588 PTP（Precision Time Protocol）を導入することが検討されている．一方，IEEE 1588 の実装には豊富なオプションが提供されているため，時刻同期精度の平均値や最悪値に関するシステム要件を満たす，PTP のコンフィギュレーションを探索するための DSE（Design Space Exploration）には膨大な時間が必要となる．本研究は，車載ネットワークに対する PTP の適用性を短時間で容易に評価することを目的とし，PTP の DSE を支援するシミュレーションツールを提案する．さらに，このツールを用いた PTP 適用性評価の事例を示す．

キーワード：車載ネットワーク，時刻同期，IEEE 1588，シミュレーション

Simulation Evaluation of the Time Synchronization Standard IEEE 1588 in Automotive Networks

XIWEN CHEN[†1]　YUTAKA MATSUBARA[†1]　YASUHIRO YAMASAKI[†2]
KENJI HONTANI[†2]　KAZUHIRO KAJIO[†2]　HIDEYUKI IWAKIRI[†2]
HIROAKI TAKADA[†1]

Abstract: Toward the implementation of time-critical functions such as Autonomous Driving and ADAS (Advanced Driver Assistance System) in vehicles, the application of time synchronization standard IEEE 1588 PTP (Precision Time Protocol) to automotive Ethernet is under consideration. However, there exists a large amount of implementation options for IEEE 1588, which results that enormous DSE (Design Space Exploration, DSE) efforts are necessary to find suitable PTP configurations which can meet the automotive synchronization requirements. In this research, in order to easily and speedily verify the applicability of PTP in automotive networks, we propose a simulation tool to support the DSE of PTP. Using this tool, we carried out DSE experiments to do the verification.

Keywords: Automotive Network, Time Synchronization, IEEE 1588, Simulation

1. はじめに

将来の自動車には，ADAS（Advanced Driving Assistant System）や自動運転など，さまざまな機能が実装されると予想されている．これらの次世代車載システムは，そのタスクの遂行に，複数の ECU やセンサの協調作業が必要となるタイムクリティカルシステムである．そのため，複数の ECU が同一のタイムベースを保持して，時刻を同期する必要がある．しかし，現在の主流である次世代車載ネットワーク規格 Ethernet [1][2]だけでは，車載システムが求めるハードリアルタイム性と，高精度な時刻同期を実現することが困難である．そこで，Ethernet のリアルタイム性と時刻同期を向上させることを目的として，時刻同期規格 IEEE 1588 PTP（Precision Time Protocol）[3]を，車載ネットワークへ導入することが検討されている．

PTP の実装には，タイムスタンピング方式や時刻差平滑化制御手法（後述）などの面において豊富な設計オプションが提供されている．また，ネットワーク状況（e.g. トラヒック設定）が変わると，最適となる PTP コンフィギュレーションも変わる可能性が高い．従って，ネットワーク設定ごとに，時刻同期精度の平均値や最悪値に関する車載システム要件を満たす，PTP のコンフィギュレーションを探索するためのデザイン・スペース・エクスプロレイション（Design Space Exploration, DSE）には膨大な回数の実験が必要となる．実機で PTP の DSE を行うには，PTP を実現するためのソフトウェアや，PTP 機能を有するネットワークカードやスイッチなどのハードウェアのコストが掛かる．加えて，テストする度に，コンフィギュレーションを変更するセットアップに，多大な労力と時間が必要となる．以上により，正確かつ低コストで PTP の車載ネットワークに対する適用性を評価する手法が求められている．

本研究は，車載ネットワークに対する PTP の適用性を短時間で容易に評価することを目的とし，PTP の DSE を支援するシミュレーションツールを提案する．そして，時刻同

†1 名古屋大学　大学院情報学研究科
Graduate School of Informatics, Nagoya University
†2 トヨタ自動車株式会社　電子制御システム開発部
TOYOTA MOTOR CORPORATION
Electronics Control System Development Div.

Vol.2019-ARC-235 No.2
Vol.2019-SLDM-187 No.2
Vol.2019-EMB-50 No.2
2019/3/17

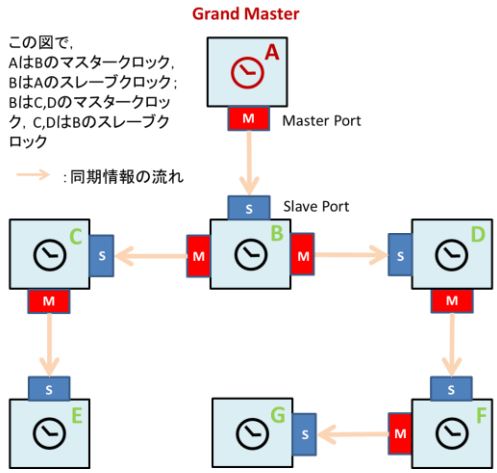


図 1　時刻同期の例

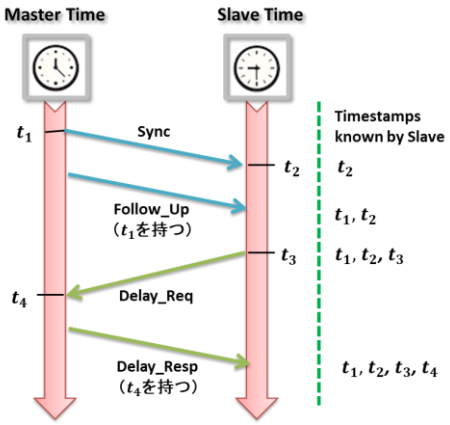


図 2　時刻同期の情報伝送と伝播遅延の計測

期パフォーマンスに影響を与えうる設計オプションの中で，タイムスタンピング方式，時刻差平滑化制御手法をそれぞれ変更し，想定する車載環境に最適な方式を，シミュレーション評価実験によって探索する．

2. IEEE 1588 PTP

2.1 概要

IEEE 1588 PTP は，Ethernet のようなマルチキャスト対応ネットワークにおける，時刻同期のための標準プロトコルである．元の策定目的は主流の時刻同期プロトコル NTP と GPS の弱点を補うためであった[4]．PTP はローカルネットワーク上のクロックデバイスを，マイクロ秒級の高精度で時刻同期させることが可能であり，ミリ秒級精度の NTP より優れた同期性能を持つ．加えて，GPS よりも安価なため，高同期精度および低コストが求められる次世代車載システムには最適と言われている．

PTP は根本的に言うと，時刻同期メッセージの交換を介して，各ネットワークノードのローカルクロックとあるレファレンスクロックの間の時刻差（オフセット）を計算する手法を定義したのである．具体的に，PTP は（1）時刻同期階層構築，（2）時刻同期情報伝送，（3）ノード間伝播遅延計測の 3 つの仕組みから成り立っている．初期化段階に（1）の時刻同期階層が構築され，（2）と（3）は周期的に行われ，常に最新の時刻差を算出する．

2.2 時刻同期の仕組み

2.2.1 時刻同期階層構築：BMCA

BMCA（Best Master Clock Algorithm）というアルゴリズムは，周期的に送信された Announce メッセージを利用し，全てのネットワークノードのローカルクロックデバイスの属性（e.g. クロック種類，基本精度）を比較する．その比較結果を元に時刻同期階層が構築される．一番信頼性の高いクロック（e.g. GPS に同期されているクロック）を持つノードは，ネットワークの根本的な時刻参照ソース，グランドマスター（Grand Master）と指定され，同期階層のルートに置かれる．他のノードも，互いに時刻同期上のマスター・スレーブ関係を持つようになる．

図 1 の例で，複数の隣接ノードを持つノードは，一般的にある隣接ノードのスレーブクロックとなり，他の隣接ノードのマスタークロックとなる．

2.2.2 時刻同期情報伝送

時刻同期の階層と全てのノード間のマスター・スレーブ関係が決まった後，マスターはスレーブに対して周期的に自分の時刻情報を通知し始める．この際に用いられるのが，Sync メッセージと Follow_Up メッセージである．

まず，マスターは Sync の送信された時点のタイムスタンプを記録する．本研究はコストの観点から，送信処理の途中でタムスタンピングを行うハードウェアを使用しないと想定する．Sync 自身の代わりに，Follow_Up に Sync の送信時タイムスタンプt_1を，スレーブへ運ばせる．スレーブは Sync を受け取った時点の受信時タイムスタンプt_2を記録する（図 2）．

ここで，ノード間の通信遅延を$PathDelay$とすると，下記のような式が成立する．

$$OffsetFromMaster = t_2 - t_1 - PathDelay$$

つまり，マスターとスレーブの時刻差を知るには，Sync の伝播遅延$PathDelay$を算出すればよい．伝播遅延の計算は，次の仕組みに任せる．

2.2.3 伝播遅延計測

IEEE 1588 では，2 つの伝播遅延計測メカニズム，Delay request-response（E2E 通信）と Peer delay（P2P 通信）が定義されている．本研究は，Delay request-response を採用する．Delay request-response では，2.2.2 節の時刻同期情報伝送で記録されたタイムスタンプも利用される．スレーブクロックは，周期的に Delay_Req メッセージをマスタークロックに送り，その送信タイムスタンプt_3をローカルで記録する．マスターは，Delay_Req の受信タイムスタンプt_4を記録し，Delay_Resp に載せてスレーブへ返す（図 2）．

Vol.2019-ARC-235 No.2
Vol.2019-SLDM-187 No.2
Vol.2019-EMB-50 No.2
2019/3/17

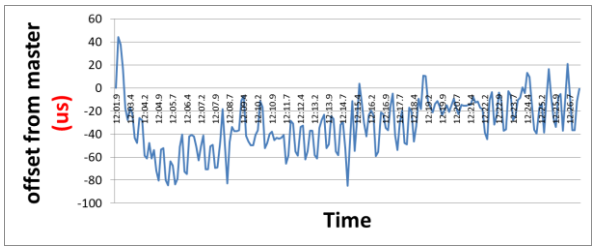


図 3　2 台の PC で測定した$OffsetFromMaster$時系列波形

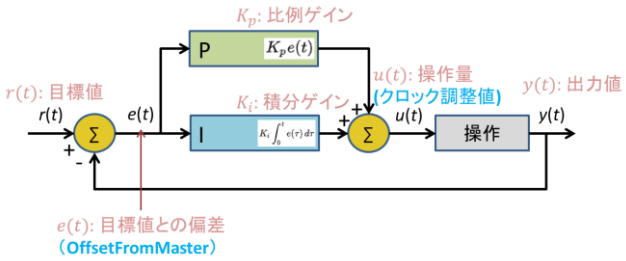


図 4　PTP の時刻差平滑に PI 制御の適用仕組み

Delay_Resp を受け取ったら，スレーブはt_1~t_4のタイムスタンプ値を全て分かるようになる．これらの時刻情報を使い，自身が送信した Delay_Req の伝送遅延を計算する．具体的には，下記の数式でマスターとスレーブ間の平均伝播遅延を計算する．

$$\boldsymbol{PathDelay} = \frac{((t_4 - t_1) - (t_3 - t_2))}{2}$$

ここで算出された PathDelay を 2.2.2 の数式に代入することで，スレーブは自分とマスターの時刻差が計算できる．

2.3 時刻差平滑化制御手法

IEEE 1588 はスレーブクロックとマスタークロックの時刻オフセットを計算する手法のみを定義しており，どのようにこのオフセットを補正するのかは開発者に任される．

直感的に，2.2.2 節で算出された$OffsetFromMaster$を直接クロック修正値として使えばよいと思われがちだが，図 3 で示しているように，実機環境で得られた$OffsetFromMaster$は一般的に変動が激しいため，安定性を求めるクロック調整には適していない．

より良い時刻同期効果を得るために，時刻オフセットに一定の平滑化制御を施すことが勧められている．本研究は，PI 制御と移動平均フィルタの 2 種類の制御方式について検討を行う．

2.3.1 PI 制御

PI 制御（Proportional-Integral Controller）は，制御工学におけるフィードバック制御の一種である．出力値と目標値の偏差，およびその積分によって入力値の制御を行う．

その仕組みを図 4 に示す．　PTP の時刻差平滑化に PI 制御を適用すると，目標値$r(t)$はマスターの時刻，制御量$y(t)$はスレーブの時刻となり，2 つの差は時刻差$e(t)$となる．$e(t)$にそれぞれ P 制御と I 制御をかけ，最終的な結果が今回のクロック調整値，$u(t)$となる．$u(t)$を使いクロックを校正して（図 4 の「操作」ブロックに対応），校正された後のスレーブ時刻をまた次回のマスター時刻に追従させるフィードバックループを行う．

PI 制御は，制御量を安定して目標値に近づけさせることが可能なため，今までの PTP 実装に最も多く使われてきた．

2.3.2 移動平均フィルタ

移動平均とは，系列データの平均を計算することで，そのデータ系列を平滑化する手法である．大別すると移動平均フィルタは下記のような種類がある．

単純移動平均（Simple Moving Average, SMA）: 直近の n 個のデータの，重み付けのない単純な平均．

加重移動平均（Weighted Moving Average, WMA）: 直近の n 個のデータに，異なる重みをつけて計算した平均．

指数移動平均（Exponential Moving Average, EMA）: 全てのデータの重みを指数関数的に減少させて計算した平均（重みの減少度合いは平滑化係数と呼ばれる 0 と 1 との間の値をとる定数βで決定される．時系列上のある時点tの値をθ_tで表し，ある時点tでの EMA をv_tで表すと，EMA の計算式は：$v_t = \beta v_{t-1} + (1-\beta)\theta_t$）．

PTP の時刻差平滑化に移動平均フィルタを使いたいときは，$OffsetFromMaster$の平均を取り，その平均値をクロック調整値として使う．

第 4 章のシミュレーション評価実験では，今回の対象ネットワークに対して PI 制御と移動平均フィルタの制御効果を比較し，どちらが適しているかを明らかにする．

3. 車載向け PTP シミュレーション環境構築

本章は，車載ネットワーク向け PTP のシミュレーション環境構築方法について記述する．行われた 2 つの構築ステップ：基本ソース選定，およびシミュレーションネットワーク作成を説明し，開発されたシミュレータのメリットについて述べる．

3.1 基本ソース選定

開発時間を短縮するため，シミュレータベース，基本ネットワークノード機能，そして時刻同期関連機能の実装という 3 つの方面において，既存のソースを利用した．

シミュレータベース：モデル型のネットワークシミュレーションフレームワーク，OMNeT++[5]を利用．

基本ネットワークノード機能:OMNeT++向けの汎用ネットワークコンポーネント(e.g. スイッチ, TCP クライアント）やプロトコル（Ethernet, PPP, IEEE 802.11, TCP, UDP, IPv4 など）などのモデルセットを提供するオープンソースモデルライブラリ，INET[6]を利用．

時刻同期関連機能の実装：OMNeT++に向け，　PTP 機能を実現したモデルライブラリ libPTP[7]と，現実のクロックノイズを生成するための C++ライブラリ libPLN[7]を利用．

Vol.2019-ARC-235 No.2
Vol.2019-SLDM-187 No.2
Vol.2019-EMB-50 No.2
2019/3/17

表 1　開発に用いられたソフトウェアとそのバージョン

ソフトウェア	バージョン
OMNeT++	4.6 for Linux
INET	2.6.0
libPTP	Github コミットハッシュ：71d4dc6
libPLN	Github コミットハッシュ：c9c77bc

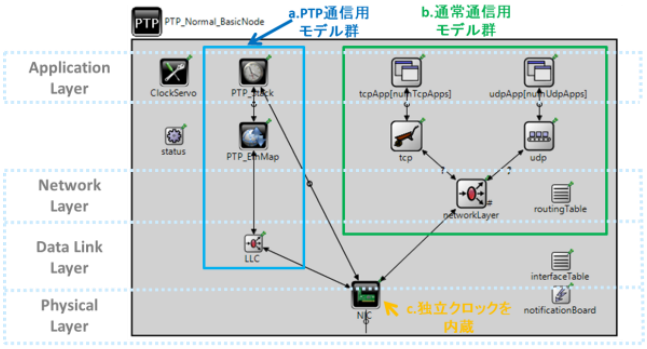
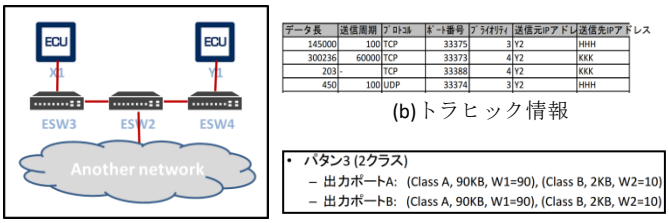


図 5　車載 ECU ノードのシミュレーションモデル



(a)ネットワークトポロジー　(b)トラヒック情報　(c)スイッチ設定

図 6　想定する車載ネットワーク情報

各ソースのバージョンは，表 1 により示す．

3.2 シミュレーションネットワーク作成

PTP 時刻同期機能を有する ECU をシミュレーションするために，INET が提供する UDP などの通常通信機能と，libPTP の PTP 機能を 1 つのネットワークノードモデルに統合した．また，PTP の時刻同期メッセージが識別できるスイッチなど，自動車企業が実際に使っているネットワークコンポーネントのシミュレーションモデルも作成した．本研究が使用している車載向けの PTP シミュレーション環境の ECU モデルは，図 5 に示している．最後に，車載ネットワークを想定して，図 6 のような情報をシミュレータに導入した．

3.3 動作確認およびメリット

構築したシミュレーション環境の全体構成を図 7 で示す．ECU ノード X1 と Y1 が時刻同期を行い，Y1 はマスタークロック，X1 はスレーブクロックとされている．シミュレーションテストで得られた$OffsetFromMaster$の分布特徴と実車のものが合致していることにより，開発されたシミュレータが正確に実機の環境を再現できたと考えられる．

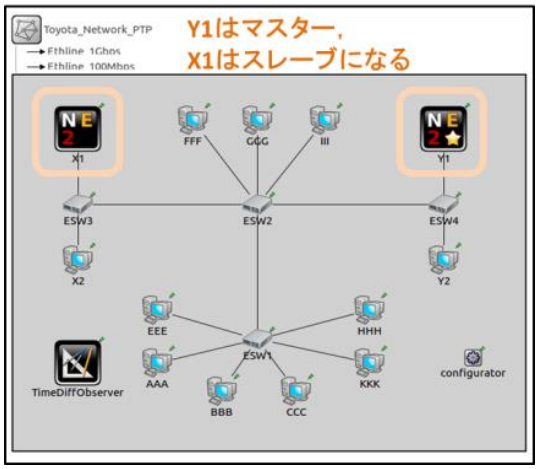


図 7　本研究が使用したシミュレーション環境

表 2　評価実験における時刻同期設定

項目	設定値
クロック誤差度合い	150ppm
遅延計測メカニズム	E2E
Sync 送信周期	125ms
伝播遅延計測周期	1s

実機と比べて，本研究が提案したシミュレーション環境で PTP の適用性を評価すると，下記のようなメリットがあると考えられる．

- 評価環境の毎回セットアップが不要
- モデルを作成することで，実験機器の購入が不要
- 計算された$OffsetFromMaster$の代わりに，各クロックモデルの本当のクロックズレで評価する
- データ自動統計やグラフの自動生成など，便利な結果分析ツールが提供

4. シミュレーション評価実験

この章では，開発されたシミュレータを用い，タイムスタンピング方式と時刻差平滑化制御手法の各コンフィギュレーションの比較・評価を行った．評価結果により，今回想定する車載ネットワーク環境に対する，各方式の最適なコンフィギュレーションを決定した．

4.1 実験における共通設定

時刻同期設定：評価実験の時刻同期における共通設定を表 2 に示す．この中で，クロック誤差は実車が実際に使っている ECU のクロック情報をもとに設定した．Sync の送信周期と伝播遅延計測の周期は，IEEE 1588 のデフォルト設定に従う．

評価基準：車載時刻同期要件を 1.5ms（仮定）以下の時刻同期精度を求めるように定めると，時刻差が 1.5ms を超えた PTP コンフィギュレーションは採用しない．その上，全

Vol.2019-ARC-235 No.2
Vol.2019-SLDM-187 No.2
Vol.2019-EMB-50 No.2
2019/3/17

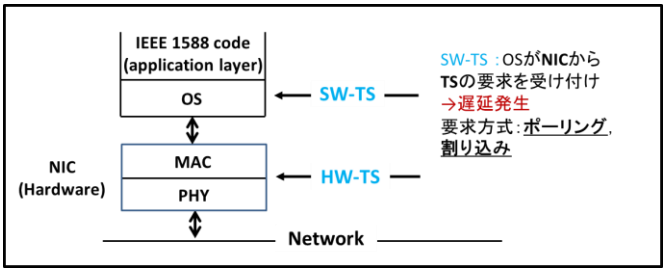


図 8　3 つのタイムスタンピング方式

体の時刻同期パフォーマンスを示す，時刻差の平均値もできるだけ抑えたい．

評価項目：ノード X1 と Y1 のクロックズレをシミュレーション中にログとして残し，その絶対値の平均値，および絶対値の最大値（時刻差最悪値）で評価する．

評価方式：各コンフィギュレーションで 10 回シミュレーションする．毎回の実験時間は 61s に設定し，実験結果として，各コンフィギュレーションの 10 回実験における評価項目の平均値を計算する．

4.2　実験 1：タイムスタンピング方式評価

4.2.1 実験目的

PTP では時刻同期メッセージのタイムスタンプによって時刻差が計算されるため，ECU のタイムスタンピング方式は時刻同期のパフォーマンスに影響を及ぼす．本実験は，3 つのタイムスタンピング（TS）方式：

1. ハードウェア TS（HW-TS）
2. ソフトウェア TS：ポーリング（SW-TS-P）
3. ソフトウェア TS：割り込み（SW-TS-I）

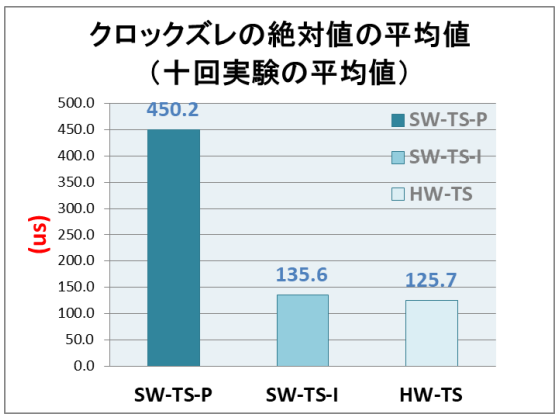
の時刻同期に与える影響を比較し，今回の車載ネットワークにとって最適な TS 方式を探し出す．

図 8 のタイムスタンピング方式モデルを参照すると，ハードウェア TS が NIC で直接タイムスタンプを付けるのに対して，ソフトウェア TS は OS で行われる．つまり，SW-TS はポーリングあるいは割り込みで NIC のタイムスタンピング要求を受け付けなければならず，これにより余分な要求遅延が発生することが分かる．

ただし，HW-TS 機能を有する NIC の購入はコストがかかるため，SW-TS の評価結果が良ければ，できるだけ SW-TS を採用したい．

4.2.2 2 つの SW-TS 方式の請求遅延モデル

OS がポーリングで TS 要求を受け付ける場合，TS 要求が受け付けられるまでの待ち時間は，最短 0 秒（即時受け付け）から最大ポーリング周期までの区間にほぼ均一に分布する．これにより，シミュレータでポーリング方式の要求遅延を一様分布で近似するのが合理的と考えられる．また，TS 要求を割り込み方式で受け付ける場合，一定確率で即時に受け付けることが可能であり，残りの確率では最大$MaxINTDelay$の待ち時間が発生する．ここで，シミュレータでは，割り込み要求の即時受け付け確率をaとし，0 秒



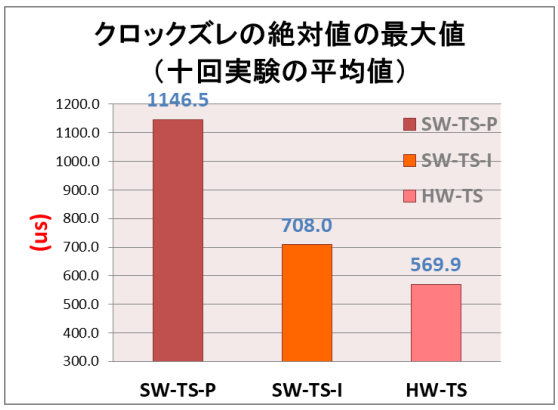


図 9　実験 1：タイムスタンピング方式評価結果

から$MaxINTDelay$までの待ち時間が同じ確率で発生するという形で近似する．HW-TS は，上述の要求遅延が発生しないため，遅延モデルを作る必要がない．

4.2.3 パラメータ設定

4.2.2 節の各遅延モデルのパラメータを決める必要がある．OS によってポーリング周期は様々であるが，ここでは一般的な値である 1ms をポーリング周期とする．割り込み方式の即時受け付け確率と最大待ち時間も決めにくいが，通常の状況を考え，それぞれ 50%と 100us にした．つまり，シミュレーション実験では各方式のパラメータを下記のように設定した．

ポーリング方式：

ポーリング周期が 1ms（一様分布[0, 1ms]）

割り込み方式：

$MaxINTDelay$が 100us，即時受け付け比率aが 50%

4.2.4 実験結果

各タイムスタンピング方式の 10 回実験の結果を図 9 にまとめて示す．

まず，クロックズレの絶対平均値を見ると，SW-TS-P 方式は 450us，SW-TS-I は 136 us，HW-TS は 126 us の平均時刻同期精度が達成できたことが分かった．また，クロックズレの最悪値においては，3 つのタイムスタンピング方式は平均してそれぞれ 1147 us，708 us，570 us の最大時刻差が発生した．

Vol.2019-ARC-235 No.2
Vol.2019-SLDM-187 No.2
Vol.2019-EMB-50 No.2
2019/3/17

$$加重移動平均値=\frac{n\times A[n]+(n-1)\times A[n-1]+(n-2)\times A[n-2]+\cdots+2\times A[2]+1\times A[1]}{n+(n-1)+(n-2)+\cdots+2+1}$$

n:フィルタの段数/リストの数
A[i]:要素のリスト(nが最新、1が最古)

図 10　実験 2 で使用した加重移動平均の計算式

4.2.5 考察

ズレ平均値と最大値両方とも，SW-TS-P > SW-TS-I >HW-TS という結果になった. これは, ポーリング周期（1ms）が最大割り込み時間（100μs）よりもずっと長く設定されていたためと考えられる．また，3 つの方式は全て今回の評価基準（1.5ms 以下の最大時刻差）を満たしているが，SW-TS-P での時刻同期効果は他の 2 方式より顕著に劣っているため，SW-TS-P を採用しないようにする．最後に，ズレ平均値と最大値両方とも, SW-TS-I と HW-TS の結果に大差はなかった.

HW-TS 機能を有する NIC を購入する時のコストを考えると，今回の車載ネットワーク環境においては，時刻同期性能とコストのバランスが良い，割り込み方式のソフトウェアタイムスタンピングを採用することが望ましい.

4.3 実験 2：時刻差平滑化制御手法評価

4.3.1 実験目的

2.3.1 節と 2.3.2 節で，本研究が興味を持つ 2 種類の時刻差平滑化手法，PI 制御と移動平均フィルタを紹介した．PI 制御は，幅広い業界において多くの利用実績があり，平滑化効果がよく認められているが，その実装がやや複雑である. 移動平均フィルタは主にデジタル信号処理に用いられ，制御に用いられることは少ないが，その軽量さと実装のしやすさがメリットである.

対象ネットワークに対して，どちらの手法が適しているかを明らかにするために， PI 制御と移動平均フィルタでの時刻同期パフォーマンスをシミュレータで比較する.

4.3.2 単純移動平均フィルタの時刻差発散現象

2.3.2 節で述べたように，主な移動平均フィルタには，単純移動平均 SMA，加重移動平均 WMA，そして指数移動平均 EMA の 3 種類が挙げられる.

SMA は一番単純で実装もしやすいが, $n = 32$の移動ウインドウで$OffsetFromMaster$を平滑してクロック調整値として使った結果，クロックの時刻差が発散してしまい，同期が出来なくなった．調査や分析によると，これはフィルタの移動ウインドウサイズが大きくなるにつれ，取った平均値の中に過去の情報の比率が高くなるためである．過去の経験で$OffsetFromMaster$がうまく平滑されるが，今回の時刻調整値での最新時刻差の重みが下がってしまう．つまり，新しいデータが平均値に反映されるのが遅くなり，毎回の時刻調整に対する平滑化制御の反応速度（応答性）が足りなくなったためである.

上述の分析に基づくと，すべてのデータに同じ重みを付ける SMA は，平滑効果と応答性を同時的に達成することは困難である．そのため，以降の比較は，データの重みを調整することができる WMA と EMA のみを対象とする.

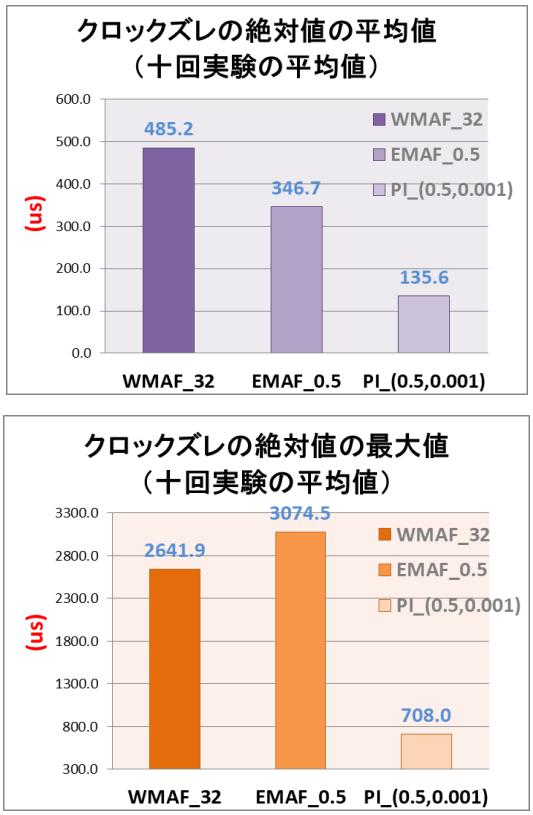


図 11　実験 2 ：時刻差平滑化制御手法評価結果

4.3.3 パラメータ設定

WMA の重み付けは図 10 の数式を使用し，移動ウインドウサイズを$n = 32$に設定する. ここでは, 最新の時刻差$A(n)$の重みが最も大きく，最古の時刻差$A(1)$の重みが最も小さくなるように設定する．新しい時刻差に十分の重みを与えることで, SMA のような時刻発散現象を防ぐことができる.

また，EMA の計算は 2.3.2 節の数式を使う．式中の平滑化係数βと$(1-\beta)$は，それぞれ過去の時刻差データと今回の時刻差の重みとなる．まずβを 0.5 に設定し，過去と現在の$OffsetFromMaster$に同じ重みを付けるようにする.

PI 制御の比例ゲインK_pと積分K_iの設定については，他の開発者の経験を参考した．IEEE 1588 PTP のオープンソース実装 PTPd2[8]では，PI 制御のデフォルトパラメータが（K_p，K_i）=（0.1, 0.001）にされている．また，本研究のシミュレーション環境の構築に用いたモデルライブラリ，libPTP の開発者も，（K_p，K_i）=（0.5, 0.001）の設定でのシミュレーションサンプルを提供している．これらの情報により，10^{-1} 級のK_pと 10^{-3} 級のK_iが通常の PTP のための PI 制御に適すと判断し，今回の比較評価でも，K_pと K_iをまずそれぞれ 0.5 および 0.001 に設定する.

4.3.4 実験結果

各平滑化手法の 10 回実験結果の平均値を図 11 にまとめて示す.

Vol.2019-ARC-235 No.2
Vol.2019-SLDM-187 No.2
Vol.2019-EMB-50 No.2
2019/3/17

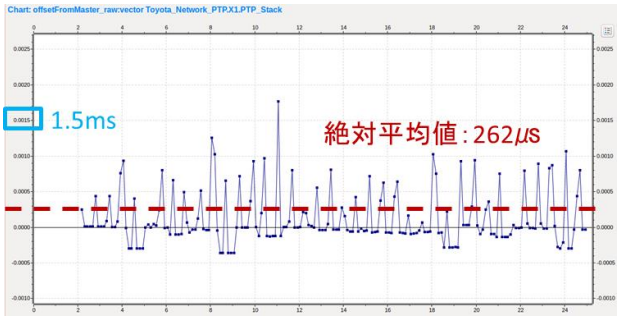


図 12　シミュレーション環境での$OffsetFromMaster$時系列波形

クロックズレの絶対平均値から見ると，ウインドウサイズ$n = 32$の WMA は 485us，平滑化係数$\beta = 0.5$の EMA は 347 us, PI 制御は 136 us の平均時刻同期精度が達成できた．また，クロックズレの最悪値においては，3 つの平滑化方式は平均してそれぞれ 2642us，3075us，708us の最大時刻差が発生した．

4.3.5 考察

クロックズレの絶対平均値において, EMA は WMA より制御効果が良かった．これは，EMA の重みが最新のデータから最古のデータまで指数関数的に減少し, 図 10 の重み付け数式による WMA が最新のデータから最古のデータまで重みを線形に減少させるからである．これにより，EMA は WMA よりも最新のデータを重視し，平滑化制御の反応速度が速いため，全体的に時刻差の平滑化効果が良かった．しかし，今回の車載ネットワークのトラヒック設定では，$OffsetFromMaster$が平均値を大幅に上回った外れ値をとることがある（図 12）．$OffsetFromMaster$が外れ値をとった場合，EMA は今回の時刻調整にその$OffsetFromMaster$から受けた影響が WMA より大きいため，クロックズレの最悪値においては WMA の方の結果が良かった．

しかし，クロックズレの平均最大値が 708us であった PI 制御に対して, 2 つの移動平均フィルタ方式はそれぞれ 2ms と 3ms を超えた時刻差最大値が発生しており，1.5ms という時刻同期精度要求を上回った．クロックズレの平均においても, 移動平均フィルタの制御効果は PI 制御に及ばなかった．

ここで書いてあるパラメータセット以外にも，3 つの平滑化方式に他のパラメータを試してみたが，結果が変わらなかった．これは，PI 制御が簡単な平均値の計算により，複雑な積分計算も行い，より安定性のある時刻差平滑が行えるからと考えられる．

上述の考察により，結論としては，対象車載ネットワークには PI 制御を採用するべきだと考えられる．

4.2 節と 4.3 節の実験結果をまとめると，今回の車載ネットワーク環境には，表 3 に示した PTP コンフィギュレーションが最適と考えられる．

表 3　採用すべき PTP コンフィギュレーション

項目	コンフィギュレーション
タイムスタンピング方式	割り込みのソフトウェア タイムスタンピング
時刻差平滑化手法	PI 制御

5. まとめ

本研究では，車載ネットワークに対する PTP の適用性評価を支援するために, PTP の DSE が容易に行えるシミュレーション環境を開発した．このシミュレーション環境で実際に PTP の DSE を行い，今回の車載ネットワーク環境においては，割り込みのソフトウェアタイムスタンピング方式, および PI 制御の時刻差平滑化手法を採用すべきだという結論を導き出した．

今後の課題として，まずシミュレーション環境の更なる改善が挙げられる. OMNeT++ではモデルのパラメータが設定ファイルで記述され，モデル数が多くなると設定ファイルが読みにくくなり，記述にも時間がかかる．OMNeT++の拡張機能を利用し，パラメータの自動記述プログラムを追加できれば，提案したシミュレータがさらに使いやすくなると考えられる．また，シミュレーション評価実験で，より幅広いパラメータセットを使い評価を行えば，より網羅性のある結論が得られると予想されている．

参考文献

[1] Nolte, Thomas, Hans Hansson, and Lucia Lo Bello. "Automotive communications-past, current and future." In 2005 IEEE Conference on Emerging Technologies and Factory Automation, vol. 1, pp. 8-pp. IEEE, 2005.
[2] Renesas Electronics Corporation. Time Sensitive Network enabling next generation of automotive E/E architecture, 2015.
[3] IEEE 1588-2008, “IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems,” 2008.
[4] Han, Jiho, and Deog-Kyoon Jeong. "Practical considerations in the design and implementation of time synchronization systems using IEEE 1588." IEEE Communications Magazine47, no. 11 (2009).
[5] OMNeT++. http://www.omnetpp.org.
[6] INET Framework. https://inet.omnetpp.org/.
[7] Wallner, Wolfgang, Armin Wasicek, and Radu Grosu. "A simulation framework for IEEE 1588." In Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS), 2016 IEEE International Symposium on, pp. 1-6. IEEE, 2016.
[8] PTPd2. https://sourceforge.net/projects/ptpd2/.