Vol.2019-ARC-235 No.1
Vol.2019-SLDM-187 No.1
Vol.2019-EMB-50 No.1
2019/3/17

組込み Linux の Real-Time Ethernet 性能評価

出原 章雄[†1]　水口 武尚[†1]

概要：近年, 汎用 Ethernet デバイスの低価格化, 高機能化から, 機器間の通信に, 独自の専用バスではなく, 汎用 Ethernet を使いたいという要求が高まっている．しかし，Ethernet は一般的にスループット向上を目指すベストエフォート型であり，通信の定周期性が問題となることが多い．

そこで，IEEE は，Ethernet 上で時間制約のある通信を行うことを目的として，TSN (Time-Sensitive Networking)を策定中である．TSN を実現するためには，送信タイミングのスケジューリングが重要であり，この実現に向けて，Linux は，2018 年 10 月に ETF (Earliest TxTime First)機能を導入した．本機能により，Network Interface Controller が搭載するフレーム送信タイミング指定機能を，アプリケーションから利用可能となる．そこで，組込み Linux 上での TSN 通信の実現可能性を確認するため，組込み機器上で ETF 機能を用いた場合のフレーム送信周期のぶれを評価した．

結果，ETF 機能を利用しない通常送信において，1ms 周期のフレーム送信の最大値と最小値の差は 20us 程度となった．対して，ETF 機能を利用した場合，最大値と最小値の差は 55ns となり，送信タイミングのぶれは 10 ナノ秒オーダとなることが判明した. 一般的な TSN 通信のユースケースではマイクロ秒の精度が求められるが, 通常送信の場合, フレーム送信タイミングは最悪 20us 程度ぶれるため，TSN 通信は現実的でない．これに対し，ETF 機能を利用した場合，フレーム送信タイミングは 10 ナノ秒オーダのぶれに収まるため，組込み Linux においても TSN 通信が実現可能となる見込みを得た．

キーワード：Linux, PTP, Ethernet, Earliest TxTime First, ETF, NIC, TAS

Performance evaluation of Real-Time Ethernet on embedded Linux System

Akio Idehara[†1]　Takehisa Mizuguchi[†1]

Keywords: Linux, PTP, Ethernet, Earliest TxTime First, ETF, NIC, TAS

1. はじめに

近年, 汎用 Ethernet デバイスの低価格化, 高機能化から，機器間の通信に，独自の専用バスではなく，汎用 Ethernet を使いたいという要求が高まっている．しかし，Ethernet は一般的にスループット向上を目指すベストエフォート型であり，通信の定周期性が問題となることが多い．

そこで，本稿では組込み Linux(R)上で Ethernet デバイスを用いた場合の Ethernet 通信の定周期性について評価する．次節で関連技術，3 節で課題，4 節で評価方法，5 節で結果と考察について述べる．6 節で今後の展望を述べ，最後に本論文をまとめる．

2. 関連技術

Ethernet デバイスを用いて定周期通信を行う場合には，近年 Time Sensitive Networking (TSN)の適用が検討されている．本節では TSN について記載する．

2.1 TSN の概要

TSN は時間制約のある通信を行う際に使用する標準規格の集合である．現在，IEEE 802.1 のワーキンググループにて TSN に含める規格を策定中である．TSN に含まれる主要な機能を次に示す(参考文献[1]参照).

(1) **時刻同期 (Time Synchronization)**

TSN では，ネットワーク内の全ての機器について，内部クロックを用いた同期を可能とする．このためのプロトコルとしては PTP の利用が想定されている(2.3.1 節参照).

(2) **送信機器とネットワーク間の調停 (Contracts between transmitters and the network)**

TSN では，送信機器とネットワーク間で調停を行うことで，次の 3 つの実現を目指している．

(2-1) 一定範囲内のレイテンシと輻輳によるフレームロスの防止 (Bounded latency and zero congestion loss)

ベストエフォート型のネットワークでは，ネットワーク機器のバッファオーバフローにより，パケットロスが起こることがある．そこで TSN では，パケットの送信量を一定に抑え，十分なバッファ領域を確保することにより，輻輳を防止し，機器間のパケット送信のレイテンシ最悪値を保障する．この実現には，送信元にてフレームの送信タイミングを調整する必要がある．この実現方式として，送信キューの操作アルゴリズム(Credit-Based Shaper，および，Time-Aware Shaper)の使用が想定されている(2.3.2 節参照).

(2-2) 信頼性の高いパケット配送(Ultra-reliable packet delivery)

パケットロスの要因として，機器の故障も挙げられる．TSN では複数の経路に対し，フレームのコピーを複数回送

†1 三菱電機株式会社情報技術総合研究所
Mitsubishi Electric Corp. Information Technology R&D Center

Vol.2019-ARC-235 No.1
Vol.2019-SLDM-187 No.1
Vol.2019-EMB-50 No.1
2019/3/17

信し，受信側にて重複データを除去する機能がある．これにより，機器故障によるパケットロスを防止し，故障の検知や回復のサイクルを不要とする．

(2-3) 柔軟性 (Flexibility)

機器間で TSN 通信を開始・終了することが可能である．この開始・終了の処理中も，通信は維持される．

(3) ベストエフォートサービスとの共存 (Coexistence with best-effort services)

TSN の通信量は一定に抑えられているため，ベストエフォート通信で使用される QoS 機能(優先度スケジュール，重み付き公平キュー，ランダム初期廃棄など)も共存して動作可能である(ただし，使用可能な帯域は減少する)．さらに，TSN 以外の通信も使用可能である．

2.2 TSN と他通信方式の比較

表 1 に従来の専用バスに代表される固定ビットレート通信，Ethernet に代表されるベストエフォート通信，および，TSN の比較を示す(参考文献[1]の Table1 を元に作成)．

表 1 通信方式の比較

比較項目	固定ビットレート (従来バス)	ベストエフォート (Ethernet)	TSN
機器間のレイテンシ	一定	輻輳や機器故障で遅延	一定範囲内に収まる
パケット到達順	In-Order	トポロジ変更時以外は In-Order	パケットロスの回復時以外は In-Order
レイテンシの分布	ほとんど 0 に収束	一定範囲内に収まる保障はない	一定範囲内に収まる
パケットロスの主要因	外乱 (宇宙線, 信号ノイズ)，機器故障	輻輳: 瞬間的なバッファオーバフロー	冗長経路数を超えた機器故障発生時
大量データ送信時のペナルティ	該当データの消失;他の処理へ影響なし	QoS に依存; 他の処理に影響する可能性あり	該当データの消失;他の処理へ影響なし
予約帯域以外の利用	仕様依存	予約の有無に関わらず利用可能	非 TSN フレームが利用可能

2.3 TSN で使用される技術

TSN で使用される技術の内，TSN の通信性能に関連する技術として次のものがある．以降ではこれらの技術について議論する．

- Precision Time Protocol
- Credit-Based Shaper/Time-Aware Shaper

2.3.1 Precision Time Protocol (PTP)

PTP は複数機器間の時刻を同期させるために使用されるプロトコルである．現在, IEEE 1588v2 が最新の仕様となっている．また，IEEE 1588v2 を発展・改良したプロトコルとして, IEEE 802.1AS が存在する. IEEE 802.1AS では, IEEE 1588v2 でオプションとなっていた時刻補正などの一部機能が必須化されている．また，一部フレームの簡素化が行われている．なお，IEEE 802.1AS は PTP (IEEE1588v2)と区別して gPTP (Generalized Precision Time Protocol)とも呼ばれる．

PTP を実現するには，送受信時のタイムスタンプを取得する必要がある．この値を取得する際の遅延をナノ秒オーダに抑えて，高精度な時刻同期を行うためには，Network Interface Controller(以降 NIC)による H/W サポートが推奨されている(参考文献[2]参照).

2.3.2 Credit-Based Shaper/Time-Aware Shaper (CBS/TAS)

Credit-Based Shaper(CBS)は IEEE Std 802.1Qav[3]で用いられている．CBS では，送信キューを操作し，送信量を一定に制限することで，送信帯域の確保を可能とする．

Time-Aware Shaper(TAS)は IEEE Std 802.1Qbv[4]で用いられている．TAS では，複数の送信キューを定周期でスケジュールすることで，特定のタイミングでのフレーム送信を可能とする. ここで, 時間制約のある通信を行う場合は，送信タイミングの指定が可能な TAS の機能が特に重要となる．以降では TAS にフォーカスして記載する．

TAS の機能についても，高精度に送信を実施するには，NIC にて時刻指定送信機能が必要となる．

2.4 TSN の H/W サポート

複数の H/W ベンダから TSN の一部機能を H/W でサポートする NIC がリリースされている. こうした NIC には次のような機能が搭載されている(参考文献[5][6]参照).

(1) PTP 向け機能

・送受信時タイムスタンプ保存
・タイムスタンプ用クロック補正

(2) TAS 向け機能

・フレーム送信タイミング指定
・優先度付き H/W キュー

2.5 Linux における TSN 機能サポート状況

本節では，Linux における TSN 機能のサポート状況を述べる．

(1) PTP

PTP を実装したオープンソースの S/W プロトコルスタックとして，linuxptp[7]，ptpd[8]，および，gPTPd[9]などが

Vol.2019-ARC-235 No.1
Vol.2019-SLDM-187 No.1
Vol.2019-EMB-50 No.1
2019/3/17

存在する．こうしたスタックは，送受信時タイムスタンプ取得機能，および，タイムスタンプ用クロック補正機能といった H/W サポート機能を利用することで, 高精度に機器間の時刻同期が可能である．

(2) **TAS**

TAS については，2018 年 10 月にリリースされた Linux 4.19 にて，ETF (Earliest TxTime First, Time-based Packet Transmission とも呼ばれる)という機能が導入された(参考文献[10]参照)．本機能を用いることで，アプリケーションは NIC に対し，フレームの送信タイミングを絶対時刻で指定可能となる．

3. 組込み Linux における TSN 通信の実現性

TSN で時間制約のある通信を行う場合，性能面で対応が必要なのは下の機能である．

・PTP
・TAS

前節で述べたように，Linux において，PTP はオープンソースのツール類を使用することで，すでに実現可能である．TAS は，最近 ETF が Linux 本体に取り込まれた段階である．ETF の開発者らはデスクトップ機器(Core i5 搭載)を用いた性能評価を実施しており，この結果，受信タイミングの最大と最小の差が 80ns との結果を報告している(参考文献[11]参照).

そこで本稿では，よりマシンパワーの低い組込み機器向けの H/W 構成(Intel Atom)において, ETF 機能を用いた場合の定周期送信のぶれを評価し，今後の組込み Linux での TSN 機能の実現可能性を判断する．

4. 評価

4.1 評価環境

評価環境を表 2，表 3，および，表 4 に示す.

表 2　H/W 環境

H/W: Minnow Board Turbot Dual Ethernet Quad-Core	
CPU	Intel Atom E3845 1.91GHz (4Core)
Memory	2GB
NIC	Intel i210 (2port) PTP, TAS の H/W サポート機能搭載

表 3　S/W 環境

S/W	
OS	Linux 4.19 (CONFIG_NET_SCH_ETF を有効化)
libc	glibc 2.28
gcc	gcc 8.2.1
iproute	4.19 (ETF 対応)

表 4　評価用ツール

LAN アナライザ:
Profishark 1G+ (PROFITAP 社製) Timestamp Resolution: 5ns

ターゲットの接続を図 1 に示す.

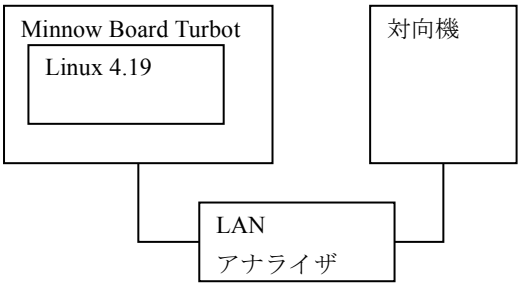

図 1　ターゲット接続

4.2 評価構成

H/W，S/W を含めた評価環境の全体構成を図 2 に示す.

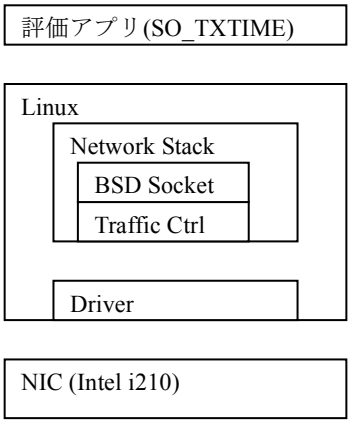

図 2　評価の全体構成

4.3 評価準備

評価の準備として, Linux の ETF 設定手順を次に示す(参考文献[12]参照).

```
phc2sys -s /dev/ptp0 -c CLOCK_REALTIME -O 0 & << (1)

tc qdisc replace dev enp2s0 parent root ¥
 handle 100 mqprio num_tc 3 ¥
  map 2 2 1 0 2 2 2 2 2 2 2 2 2 2 2 2 ¥
  queues 1@0 1@1 2@2 hw 0 << (2)

tc qdisc add dev enp2s0 parent 100:1 etf          ¥
    offload clockid CLOCK_TAI delta 150000 << (3)
```

図 3　評価準備用の ETF 設定

図 3(1)にて，phc2sys コマンドを使用し NIC の時刻(/dev/ptp0)と，システムの時刻(CLOCK_REALTIME)を同期

Vol.2019-ARC-235 No.1
Vol.2019-SLDM-187 No.1
Vol.2019-EMB-50 No.1
2019/3/17

させる．

図 3 (2)にて，tc コマンドを使用し Traffic Control を表 5 に設定する．

表 5　TC 設定 1

引数	意味
qdisc	Queueing Discipline
replace dev enp2s0	NIC 名 enp2s0 の設定を変更
parent root	root を親とする
handle 100	ハンドル値を 100 とする
mqprio	Multi queue priority を設定する
num_tc 3	Traffic Classes(TC)数を 3 に指定
map 2 2 1 0 2 2 ...	SO_PRIORITY と TC#の関係を定義 SO_PRIORITY=2 (2 番目の map)を TC1 SO_PRIORITY=3 を TC0 SO_PRIORITY=0,1,および,4～15 を TC2
queues 1@0 1@1 2@2 hw 0	hw 0 の H/W キューに対して次を実施 TC0 に 1 つのキュー(Q#0)を割り当て TC1 に 1 つのキュー(Q#1)を割り当て TC2 に 2 つのキュー(Q#2,Q#3)を割り当て

図 3 (3)にて，Traffic Control を表 6 に設定する．

表 6　TC 設定 2

引数	意味
qdisc	Queueing Discipline
add dev enp2s0	NIC 名 enp2s0 の設定を追加
parent 100:1	ハンドル値 100:1(TC0)を親とする
etf	Early Txtime First を設定する
offload	H/W オフロード機能を有効
clockid CLOCK_TAI	ETF で使用するクロック源を CLOCK_TAI とする
delta 150000	送信時刻の 150000 ns 前に送信キューに追加する

以上の設定により，フレーム送信時に，ソケットに対し SO_TXTIME，SO_PRIORITY=3，および送信時刻を設定すると Intel i210 の H/W キュー#0 に対し，送信時刻が設定される．Intel i210 は設定された送信時刻を経過すると，該当フレームを送出する．

各種の設定値の関係を表 7 にまとめる．

表 7　TC，SO_PRIORITY，および HW キューの関係

TC	SO_PRIORITY	H/W キュー
0	3	0 (送信時刻指定可)
1	2	1 (送信時刻指定可)
2	0,1,および，4~15	2 or 3 (送信時刻指定不可)

4.4 評価アプリケーション

今回の評価アプリケーションで実施する初期化処理を図 4 に示す．

```
so_priority = 3;
setsockopt(fd, SOL_SOCKET, SO_PRIORITY, &so_priority,
sizeof(so_priority)) << (1)

sk_txtime.clockid = clkid;
setsockopt(fd, SOL_SOCKET, SO_TXTIME, &sk_txtime,
sizeof(sk_txtime)) << (2)
```

図 4　初期化処理

図 4 (1)にて，SO_PRIORITY を 3 に設定する．前節の設定から，Intel i210 の H/W キュー#0 が利用される．

図 4 (2)にて，SO_TXTIME の設定を有効にする．

次に，送信処理を図 5 に示す．

```
msg.msg_control = control;
msg.msg_controllen = sizeof(control);

cmsg = CMSG_FIRSTHDR(&msg);
cmsg->cmsg_level = SOL_SOCKET;
cmsg->cmsg_type = SCM_TXTIME; << (1)
cmsg->cmsg_len = CMSG_LEN(sizeof(__u64));
*((__u64 *) CMSG_DATA(cmsg)) = txtime; << (2)
sendmsg(fd, &msg, 0); << (3)
```

図 5　送信処理

図 5 (1)にて，SCM_TXTIME を設定し，送信時刻指定を有効化する．

図 5 (2)にて，送信時刻 txtime を指定する．

図 5 (3)にて，(1)，(2)の情報を含めた上で semdmsg システムコールを用いて，フレームを送信する．

4.5 評価方法

ETF 評価アプリでは，1ms 周期でフレーム送信を 60 秒(60,000 フレーム)行い，LAN アナライザにて送信タイミングを取得する．

ETF 評価アプリの詳細について，clock_nanosleep システムコールを用いて，送信時刻より前に起床する(送信マージン(今回 400us))．起床後，SO_TXTIME を使用し，送信時刻を指定してフレーム送信を行う．以降，起床時刻および，送信時刻を 1ms ずつ増加させて，この処理を繰り返す(図 6)．

Vol.2019-ARC-235 No.1
Vol.2019-SLDM-187 No.1
Vol.2019-EMB-50 No.1
2019/3/17

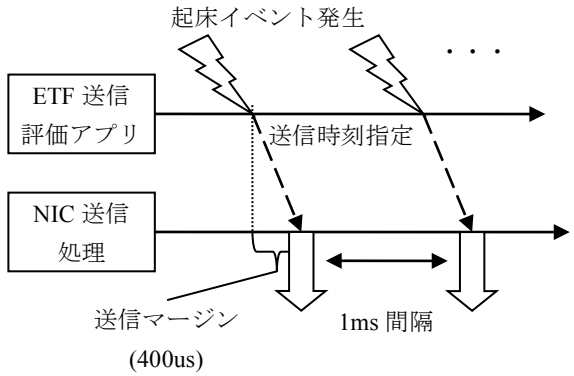

図 6　ETF 評価アプリ動作

また，比較として，SO_TXTIME を使用せず，clock_nanosleep システムコールを用いて 1ms 周期でタスクを起床させ，通常の send システムコールにてフレーム送信した場合の評価を行う(図 7).

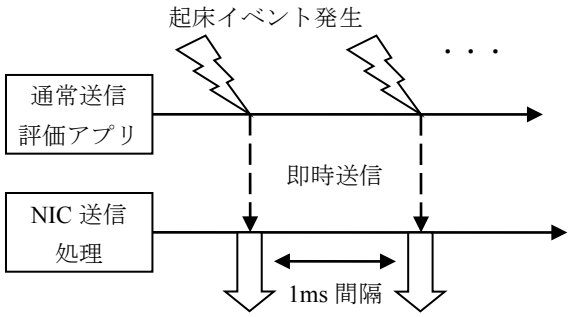

図 7　通常送信評価アプリ動作

5. 結果

LAN アナライザにて送信フレームを取得し，各フレームの通信周期(1ms)に対するぶれを算出した．結果を表 8 に示す.

表 8　評価結果

	ETF	通常
平均 [ns]	6.0	-398.3
最大 [ns]	30.0	12,250.0
最小 [ns]	-25.0	-11,210.0
最大 - 最小 [ns]	55.0	23,460.0
標準偏差	12.7	457.3

また，個々の送信フレームにおける周期からのぶれを図 8，および図 9 に示す.

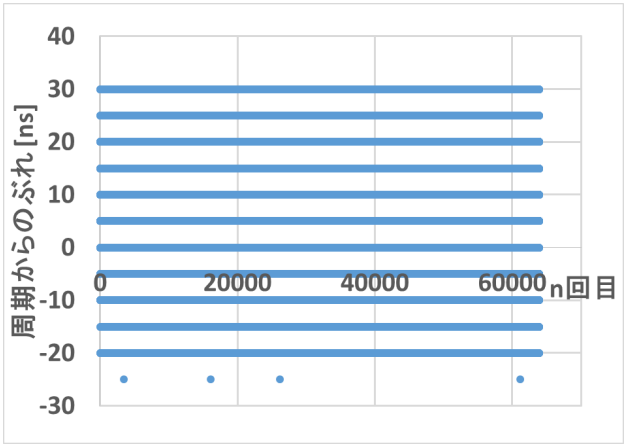

図 8　ETF 送信結果

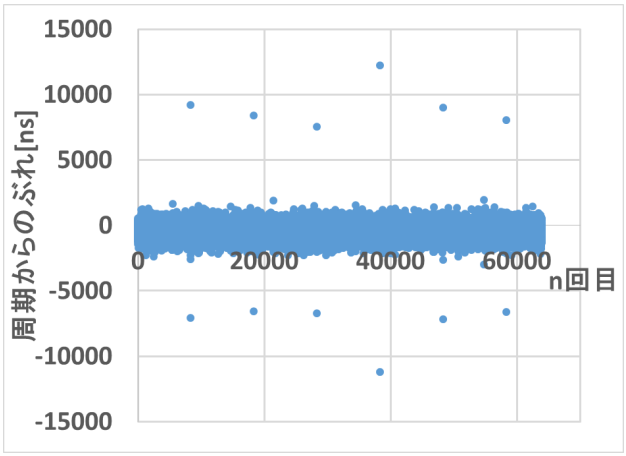

図 9　通常送信結果

評価の結果，ETF 機能を用いた場合，10 ナノ秒オーダのぶれで通信が可能と判明した．特に最大と最小の差は 55ns となった．一般的な TSN 通信のユースケースではマイクロ秒の精度が求められるが，Intel Atom を使用した場合においても，H/W サポート機能のある NIC を用いることで，組込み Linux 上で TSN 通信を実現可能という見込みを得た．

対して，通常の方法でフレーム送信した場合は，マイクロ秒オーダで遅延が生じ，最大と最小の差は 23us となった．一般的な TSN 通信のユースケースでは，マイクロ秒の精度が求められるため，23us の遅延が発生した場合，TSN 通信を実現できない可能性が高い．次節で詳細な考察を行う．

6. 考察

ETF 送信，および，通常送信の性能差異について，次の観点で考察を行う．

- 送信タイミングの遅延
- フレーム構築と送信のタイミング
- 送信マージン

6.1 送信タイミングの遅延

ETF を利用した場合，最大-最小の値は 55ns となった．

これは，Intel i210 の H/W サポート機能を用いて，送信タイミングを正しく設定できたためと想定される．ETF 開発者らの評価も最大-最小の値は 80ns 程度であり，ETF 機能は正しく動作したものと考えられる．

対して，通常送信の場合は 23us 程度の遅延が発生した．この理由について，通常，Linux では，割込み処理等において，10 us 程度の割込み禁止時間が存在する．この場合，タスクの起床時間は 10us 程度遅延する可能性がある．今回観測された遅延は，この割込み禁止時間が影響したと考える．Linux の割込み禁止時間を短くする方法について，PREEMPT_RT[13]などのリアルタイム性能向上機能の適用が考えられる．しかし，過去の評価から，割込み禁止時間を 1us 以下にすることは難しく，送信タイミングのぶれの観点で ETF が有利である．

ETF 送信の場合も，clock_nanosleep システムコールを使用しており，タスク起床の遅延は発生している．しかし，H/W に設定する送信時刻よりも前にタスクが起床しており，送信時刻の設定には間に合っている．このため，ETF ではタスク起床遅延が観測されていない．本件については 6.3 節で考察する．

6.2 フレーム構築と送信のタイミング

ETF の場合，評価アプリがフレームを構築し，socket を介して Linux カーネルに引き渡してから，実際に NIC を介してフレームが外部に送信されるまでに送信マージン分のタイムラグがある．今回の評価では NIC に指定する送信時刻の 400us 前にタスク起床を行い，フレームを構築した上でカーネルにフレームを渡している(図 10)．このタイムラグが問題となる場合，通常のフレーム送信を使用する必要がある．

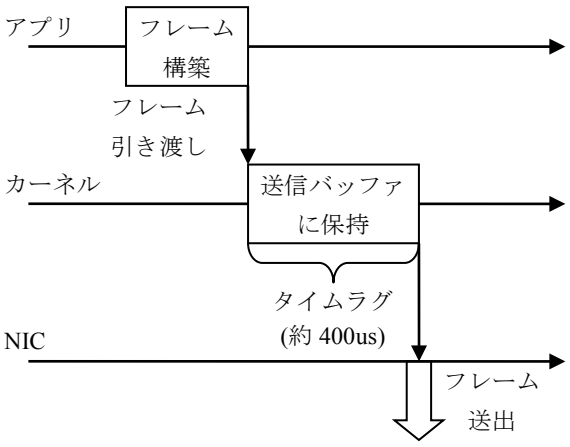

図 10　フレーム構築と送信のタイミング

6.3 送信マージン

6.1 節，6.2 節の考察と関連するが，ETF 機能を利用する場合，システムのタスク起床遅延時間を加味した上で送信マージンを設定する必要がある．例えば，送信マージンが短い場合，NIC に設定すべき送信時刻の後に，タスクが起床してしまい，フレームの送出が遅延する可能性がある(図 11).

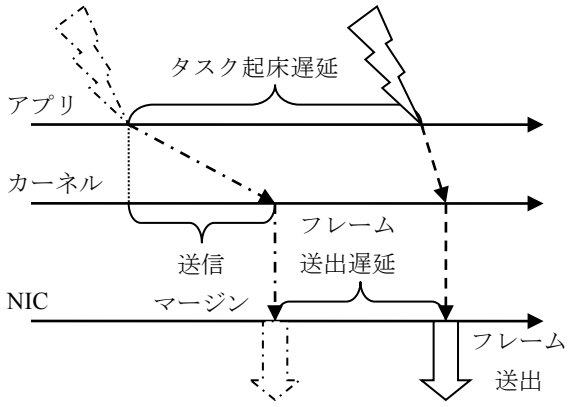

図 11　フレーム送出遅延

一般的に Linux では 10us 程度の起床遅延が発生する．また，System Management Interrupt などが発生した場合，数 100us の遅延が発生することもある．このような遅延が観測されるシステムにおいて，十分な送信マージンがない場合，ETF 機能は正しく動作しない．

以上から，実システムに ETF を適用するためには，あらかじめタスク起床遅延の最悪値を測定した上で，送信マージンを調整する必要がある．

7. おわりに

今回，組込み Linux 上で TSN 通信の実現性を確認するため，Linux に追加された ETF 機能を用いて，組込み機器上でフレーム送信タイミングのぶれを評価した．結果，最大と最小の差は 55ns となった．一般的な TSN 通信のユースケースではマイクロ秒の精度が求められるが，今回の環境はこれを満足可能と判明した．すなわち，組込み機器であっても，H/W サポート機能を持つ NIC を利用することで，TSN 通信を実現可能という見込みを得た．

今後は，複数キューを利用した ETF 機能の確認，および，フレーム受信負荷をかけた状態での ETF 機能の送信タイミングのぶれの評価など，より実システムに近い構成で評価を行う予定である．

参考文献

[1] Norman Finn. Introduction to Time-Sensitive Networking. IEEE Communications Standards Magazine (Volume: 2 , Issue: 2), 2018, 22p
[2] IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems (IEEE Std 1588-2008), 190p.
[3] IEEE Standard for Local and metropolitan area networks—Virtual Bridged Local Area Networks Amendment 12: Forwarding and Queuing Enhancements for Time-Sensitive Streams (IEEE Std 802.1Qav-2009)
[4] IEEE Standard for Local and metropolitan area networks—Bridges

Vol.2019-ARC-235 No.1
Vol.2019-SLDM-187 No.1
Vol.2019-EMB-50 No.1
2019/3/17

and Bridged Networks Amendment 25: Enhancements for Scheduled Traffic (IEEE Std 802.1Qbv-2015)
[5] “Intel(R) Ethernet Controller I210 Datasheet”. https://www.intel.com/content/dam/www/public/us/en/documents/datasheets/i210-ethernet-controller-datasheet.pdf, (参照 2018-01-23)
[6] “AM335x and AMIC110 Sitara™ Processors Technical Reference Manual”. http://www.ti.com/lit/pdf/spruh73, (参照 2018-01-23)
[7] “linuxptp”. http://linuxptp.sourceforge.net/, (参照 2018-01-23)
[8] “ptpd”. https://github.com/ptpd/ptpd, (参照 2018-01-23)
[9] “gPTPd”. https://github.com/AVnu/gptp, (参照 2018-01-23)
[10] “The first half of the 4.19 merge windows”. https://lwn.net/Articles/762566/, (参照 2018-01-23)
[11] “The Road Towards a Linux TSN Infrastructure". https://schd.ws/hosted_files/elciotna18/b6/ELC-2018-USA-TSNonLinux.pdf, (参照 2018-01-23)
[12] “[TSN] Scheduled Tx Tools - Examples and Helpers for testing SO_TXTIME, and the etf and taprio qdiscs”. https://gist.github.com/jeez/bd3afeff081ba64a695008dd8215866f, (参照 2018-01-23)
[13] “The Real Time Linux collaborative project”. https://wiki.linuxfoundation.org/realtime/start, (参照 2018-01-23)