Vol.2022-ARC-248 No.33
Vol.2022-SLDM-198 No.33
Vol.2022-EMB-59 No.33
2022/3/10

パーティショニング OS 向け ユーザモード TCP/IP プロトコルスタック

手塚　湧太郎[1]　本田　晋也[2,1]　大谷　寿賀子[1,3]　枝廣　正人[1]

概要：
自動車や航空宇宙機などの分野の組込みシステムでは，一般的な組込みシステムに比べ高い信頼性が求められている．そこで，システムを複数のパーティションに分割する機構を持つパーティショニング OS が使用されつつある．パーティショニング OS において，TCP/IP プロトコルスタックは規模が大きく，外部とやり取りするため不具合が発生する可能性が高いため，ネットワーク機能がシステムの安全性の維持には必要ない場合は，ユーザモードで動作させたいという要求がある．本論文では，TCP/IP プロトコルスタックのユーザモード化の実現方法について提案する．実現手法としては，ドライバを仮想化してパーティション毎に TCP/IP プロトコルスタックを実行する方法と，TCP/IP プロトコルスタックを実行する専用のパーティションを用意して複数のパーティションから共有する方法があるが，組込みシステムは，一般的にリソースが厳しいため後者の方法を検討する．この方針に従い，実現したい性質や機能要求を定義し，要求を満たすユーザモード TCP/IP プロトコルスタックを提案し，実装した．実装した TCP/IP プロトコルスタックについて，要求をどの程度満たしたか定性的な評価と定量的な評価を示した．

1. はじめに

自動車や航空宇宙機などの分野のシステムでは，一般的な組込みシステムに比べ高い信頼性が求められている．そこで，システムを複数のパーティションに分割して空間的かつ時間的にお互いの影響を無くすパーティショニング機構を持つリアルタイム OS（パーティショニング OS） が使用されつつある．

現状のパーティショニング OS の多くはデバイスを操作するミドルウェアを特権で動作するカーネルモードで実行させる．これは，リアルタイム性が求められるシステムではメモリ保護に MMU ではなく MPU 用いられ，MPU は MMU と比較して設定可能なメモリリージョン（アクセス権を設定可能な連続したメモリ空間）の数に制約があり，ユーザモードからデバイスアクセスが出来ないためである．デバイスアクセスのみを特権モードとすることは可能であるが，ミドルウェアの変更や呼び出しオーバヘッドが増加する．また，ミドウェアの API がユーザアプリケーションからの関数呼び出しを前提に設計されていることもカーネルモードで動作させる要因となっている．一方，カーネルモードで実行されるミドルウェアは，システムの最高水準の安全度レベルで開発する必要がある．

組込みシステムで広く使われているミドルウェアとして，ネットワーク機能を実現する TCP/IP プロトコルスタック（TCP/IP スタック）が挙げられる．TCP/IP スタックは状態遷移が複雑であることや外部とやり取りをするために不具合の原因となりやすい [1]．また，外部から調達することが一般的であるため，品質保証が難しいという課題もある．そのため，ネットワーク機能がシステムの安全に関わらないシステムの場合，TCP/IP スタックをユーザモードで動作させたいという要求がある．

本研究では，ユーザモードで動作する TCP/IP スタック（ユーザモード TCP/IP）を提案する．まず，TCP/IP スタックをユーザモード上で実行する際に実現するべき性質や機能要求を定義する．次に，定義した要求を満たすようなユーザモード TCP/IP の API 仕様とアーキテクチャを提案する．最後に実現したユーザモード TCP/IP について，定めた要求をどの程度満たしたか定性的な評価と定量的な評価を実施する．

対象とするパーティショニング OS は，次世代宇宙用マイクロプロセッサ [2] での採用が決まっている ITRON 仕様をベースとした TOPPERS/HRMP3 カーネルのシングルプロセッサ版の TOPPERS/HRP3 カーネル（HRP3 カーネル）とした．また，TCP/IP スタックとしては，ITRON

[1] 名古屋大学 情報学研究科
[2] 南山大学 理工学部
[3] ルネサスエレクトロニクス株式会社

TCP/IP 仕様に準拠した TINET を用いた．

本研究の具体的な研究課題は次の 4 点とする．

RQ1：TCP/IP スタックを構成するモジュールのユーザモード化出来ない箇所とその規模や処理内容を明らかにする．

RQ2：API 仕様の制限や変更点を明らかにする．

RQ3：帯域制御のための機構を明らかにする．

RQ4：発生するオーバヘッドを評価する．

2. パーティショニング機構

本章では，本研究で対象とする HRP3 カーネルの仕様と実装について説明する．HRP3 カーネルは，ITRON 仕様をベースとして，メモリパーティショニングと時間パーティショニング機能を拡張している．

2.1 メモリパーティショニング機能

HRP3 カーネルでは，パーティションに相当する保護ドメインと呼ぶカーネルオブジェクトのグループを定義することが可能である．保護ドメインはユーザドメインとカーネルドメインの二種類に分けられる．ユーザドメインは，複数定義することができ，オブジェクトはプロセッサのユーザモード（非特権モード）で実行され，ユーザアプリケーションなどが所属する．カーネルドメインは，システムで 1 つしか設定することができない．オブジェクトはプロセッサのカーネルモード（特権モード）で実行される．割込みハンドラや周期ハンドラなどハンドラは，カーネルドメインにしか属せない．

ある保護ドメインから他の保護ドメインのメモリにアクセスできないように設定できる．この機能は，MPU ないし MMU によって実現される．デバイスドライバは，割込み処理を使用していることや，MPU を用いる環境では作成可能なメモリリージョン数に制限があるため，カーネルモードで実行されることが基本である．メモリリージョンとは連続したアドレスのメモリの集合であり，ユーザドメインからのアクセス権を制限できる．

デバイスドライバのようにユーザドメインからカーネルモードで実行したい関数を呼び出す方法として拡張サービスコール（拡張 SVC）という機能がある．拡張 SVC で呼び出された関数は，カーネルモードで実行されるが，割込みは呼び出し前の状態を維持し，自動的に割込み禁止にはならない．

2.2 時間パーティショニング機能

時間パーティショニングは，あるパーティションの時間的な振る舞いの変化が，他のパーティションに影響を及ばさないようにする．我々が知る限りでは，TDMA スケジューリングにより各パーティションを実行する方法が一般的である．TDMA スケジューリングは，時間枠にシステム周期を定める．そして，システム周期をいくつかのタイムウィンドウに分割する．タイムウィンドウには，いずれかのパーティションが割り当てられ実行される．パーティションに属するタスクは，割り当てられたタイムウィンドウ内でスケジューリングされ，CPU 利用時間，実行順序，実行タイミングなどを保証する．

システム周期の長さや，その中でのタイムウィンドウの実行順序や実行時間はユーザが設計時に設定する．HRP3 カーネルでは，タイムウィンドウに保護ドメインが割り当てられ，所属するタスクはその時間内で優先度ベースのスケジューリングで実行される．また，TDMA スケジューリングを拡張して割込みを受け付け可能としている．割込みの処理の実行時間分，タイムウィンドウを後方にシフトすることになり，CPU 利用時間を保証する．タイムウィンドウがシステム周期を超えてシフトすると問題となるため，アイドルウィンドウと呼ばれるどの保護ドメインも割り当てられていないタイムウィンドウをシステム周期の最後に設定する．そして，タイムウィンドウが後ろずれた時間分アイドルウィンドウが短くなることで，タイムウィンドウがシフトしたとしても，システム周期内に収まるようにする．

3. 実現方針と要求

研究対象とした TCP/IP スタックである TINET の仕様と実装について説明した後，ユーザモードで TCP/IP スタックを動作させる 2 種類の手法について述べ，本研究で採用した手法について説明する．

3.1 TINET

本節では，TINET[3] の仕様と構成について説明する．

3.1.1 ITRON TCP/IP 仕様

TINET は，ITRON TCP/IP 仕様 [4] に準拠している．本仕様は，保護機構を持たない RTOS 向けに設計された仕様であり，パーティショニング OS での実行は考慮されていない．

送受信 API は，ソケット通信とは異なる仕様となっており，接続要求を待ち受けるための受付口と接続の端点として用いる通信端点を用意している．データ送信時は通信端点のみを使用する．データ受信時は接続を受付口で待ち，接続後に通信端点を使用してデータを送受信する．

組込みシステムへの利用を考慮して，通常の送受信 API に加え，バッファコピーの回数を減らす，省コピー送受信 API を用意している．省コピー API では，バッファを取得する API を呼び出して TCP/IP スタックからバッファを取得してそのバッファを直接アクセスすることでコピー回数を減らすことができる．

待ち状態に入る可能性のある API においては，ノンブロッキングコールが用意されている．ノンブロッキング

Vol.2022-ARC-248 No.33
Vol.2022-SLDM-198 No.33
Vol.2022-EMB-59 No.33
2022/3/10

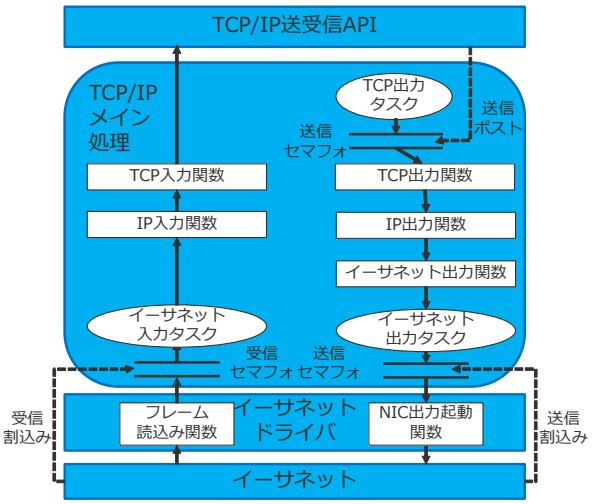

図 1 TINET の実装部分

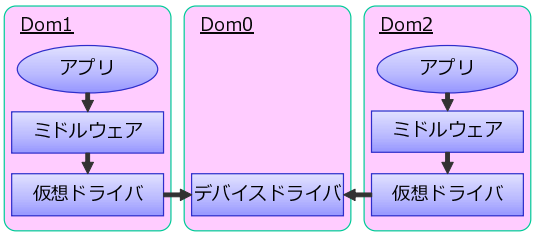

図 2 デバイス共有方式

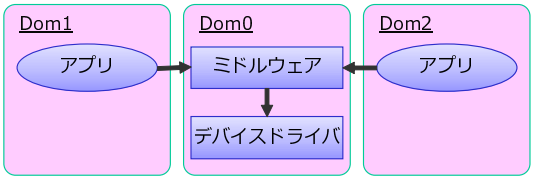

図 3 ミドルウェア共有方式

コールは API の中でブロックされる状況になった場合に，処理を継続したまま API からリターンする手順である．処理が完了した時点で，TCP/IP スタックからコールバック関数を呼び出すことでアプリケーションプログラムに処理完了を通知する．コールバック関数は通信端点ごとにユーザアプリケーションで定義可能である．

3.1.2 TINET のシステム構成

TINET のシステム構成を**図 1** に示す．送信の場合，TCP/IP 送信 API が呼び出されるとセマフォによって TCP 出力タスクが起動し，タスクルーチンで関数呼び出しが実行される．その後，データキューによってイーサネット出力タスクが起動し，セマフォを獲得してデバイスドライバを呼び出す．送信が完了すると，送信割込みでセマフォが返却され，送信完了となる．受信の場合，TCP/IP 受信 API が呼び出されるとセマフォによって，受信待ちとなる．データを受信をすると，受信割込みでセマフォが返却され，イーサネット入力タスクが起動する．

送受信どちらの場合でもノンブロッキングコールなら，セマフォ返却でなく，通信端点に登録されたコールバック関数が呼び出され処理完了を通知する．

3.2 共有する階層

パーティショニング環境で TCP/IP を動作させる 2 種類の方法を**図 2** と**図 3** に示す．

図 2 は，デバイス仮想化を用いる方法であり，デバイスを仮想化して，ユーザドメイン毎に独立した TCP/IP スタック及び仮想ドライバを実行する．仮想ドライバは物理ドライバを動作させる処理単位に送受信を依頼する．図 2 では，専用のドメインを用意しているが，デバイスドライバをカーネルドメインでしか動作させることが出来ない場合はカーネルドメインに依頼する．

図 3 は，TCP/IP スタックを共有する方法であり，専用のドメイン（Dom0）で TCP/IP スタックを実行し，ユーザドメインは，TCP/IP スタック送受信 API レベルで送受信を依頼する．

小規模組込みシステムではリソース制約が厳しいため，ドメイン毎に TCP/IP スタックを用意するのは難しいケースが多い．また，TCP/IP スタックより信頼性高いドメインでは TCP/IP スタックを同じドメインで実行すると安全性が確保できない．そのため，本研究ではミドルウェア共有方式で設計することとする．

3.3 実現したい性質及び機能

前述のミドルウェア共有方式により，ユーザモード TCP/IP を実現する上での要求について説明する．

まず, TCP/IP スタックを専用のユーザドメイン（Dom0）で実行することが挙げられる．ユーザドメインで実行するために，以下を満たす必要がある．

- 要求 1-1：TCP/IP スタックを構成するモジュールをユーザモードで動作させること．
- 要求 1-2：TCP/IP スタックを構成するモジュールを割当てたタイムウィンドウ内で動作させること．

要求 1-1 により，TCP/IP スタックに不具合が発生した場合でも，他のドメインに影響を及ぼさないメモリ保護が実現可能となる．要求 1-2 は，TCP/IP スタックを専用ドメインで動作させると，HRP3 カーネルでは**図 4** に示すように，ドメインに割り当てたウィンドウのみで実行されることにより，時間パーティションが実現可能となる．

次に，TCP/IP スタックのドメイン間による共有に関する要求について整理する．

- 要求 2-1：ドメイン毎に通信のためのオブジェクトやメモリがパーティショニングされていること．
- 要求 2-2：ドメイン毎の通信帯域が保証できる機構を持つこと．

要求 2-1 は，各ドメインはパーティショニング OS によりカーネルオブジェクトやメモリのアクセスが制限され，

Vol.2022-ARC-248 No.33
Vol.2022-SLDM-198 No.33
Vol.2022-EMB-59 No.33
2022/3/10

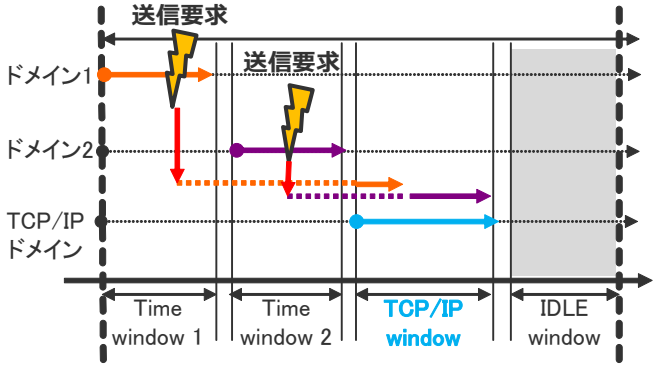

図 4 ユーザモード TCP/IP の TDMA スケジューリングでの振る舞い

お互い干渉しない環境となっている．TCP/IP スタックを用いた通信においても同様のパーティショニングを実現する必要がある．要求 2-2 は，パーティショニング OS による時間保護に相当する．

最後に，TCP/IP スタックのユーザドメインとしての実行とドメイン間による共有に共通にする要求として次が挙げられる．

- 要求 3-1：実行オーバヘッドが大きく増加しないこと
- 要求 3-2：対応可能なハードウェアアーキテクチャが広いこと

要求 3-1 は，ハードウェアコストの増加を抑制するために定めた．要求 3-2 は，提案手法の適用範囲を広げるために定めた．小規模組込み用プロセッサでは MPU が主に使われているため，MPU を前提として設計を行う．

3.4 評価項目

前述の要求に従って，ユーザモード TCP/IP の設計・実装を通じて明らかにしたい項目について説明する．

RQ1：TCP/IP スタックを構成するモジュールのユーザモード化出来ない箇所と規模や処理内容を明らかにする．例えば，イーサネットドライバは，**要求 3-2** から，MPU 環境ではユーザモードで動作させることができず，**要求 1-2**，**要求 1-2**，を満たすことが出来ないことは明確である．TCP/IP スタック中のこの様な箇所の処理内容や処理時間を明らかにすることにより各要求をどの程満たすか評価する．なお，MMU を用いることにより改善可能な点については明確化する．

RQ2：API 仕様の制限や変更点を明らかにする．API 仕様は変更しない方がユーザアプリケーションにとっては，パーティショニング OS を利用しないシステムと互換性が保てるため都合が良い．一方，**要求 1-1** や**要求 1-2** より，省コピー API やコールバック関数の実現は困難であることは明確である．また，**要求 2-1** から通信端点に対する API に何らかのアクセス制限を追加する必要がある．

RQ3：帯域制御のための機構を明らかにする．**要求 2-3** を実現するための機構を実現してその効果を評価する．本研究では，送信に絞って帯域制御を実現する機構について検討する．一方，受信は，データを受け取るまでどのドメインに対するデータであるか判断できないため，検討しない．

RQ4：発生するオーバヘッドを評価する．各要求を満たすための変更により**要求 3-1** がどの程度となるか評価する．

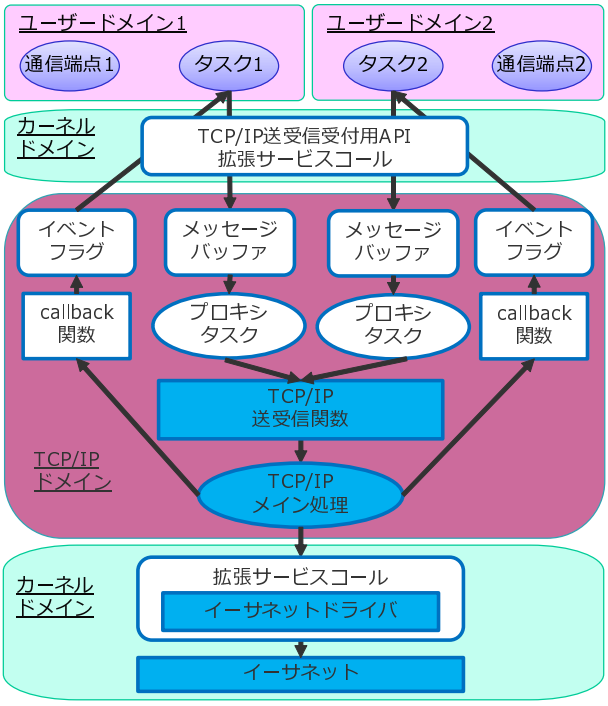

図 5 ユーザモード TCP/IP のシステム構成図

4. ユーザモード TCP/IP

実現したユーザモードで動作する TCP/IP スタックの構成を，**図 5** に示す．TCP/IP スタックは専用の TCP/IP ドメインで動作させる．ユーザドメインは拡張 SVC として実現された TCP/IP 送受信受付用 API を呼び出して，送受信を行う．本章では，まず API 仕様について説明し，続いてアーキテクチャの詳細について説明する．

4.1 API 仕様

API 仕様について，ITRON TCP/IP 仕様に対して機能制限を設けた箇所と追加した機能について説明する．

4.1.1 機能制限

省コピー送受信 API はサポートしない．省コピー送受信 API に関しては，専用のドメインで動作する TCP/IP スタックが管理するバッファにユーザアプリがアクセスすることになる．これは**要求 2-1** を満たすことができないためである．

ノンブロッキングコールはサポートしない．ノンブロッキングコールをサポートすると，TCP/IP ドメインから

ユーザドメインへドメインをまたいで関数を呼び出すことになる．ドメイン間での関数の呼び出しは，HRP3 カーネルはサポートしていないため対応しない．

4.1.2 追加機能

各 API に関して，通信オブジェクトである受付口及び通信端点を利用できるユーザドメインを 1 つに制約するアクセス制御を追加した．複数のユーザドメインがアクセスできると，他のドメインが使用する通信端点に不正なデータ送受信を発生させることが可能となり，**要求 2-1** を満たせない．ユースケースとしても異なるユーザドメイン間で同じ受付口や通信端点を共有することは考えられにくいため使用上の制約とはならないと考えている．

各 API において，バッファへのポインタを受け取るものは，呼び出されたドメインからバッファへのアクセスが可能であるかをチェックする機能を加えた．これは後述するように各ユーザドメインが呼び出す送受信 API を拡張 SVC として実現すると，不正なアドレス（他のドメインのメモリ領域や存在しないアドレス）を渡された場合，不正なアクセスや特権モードでの例外が発生するために定めた．送受信 API をユーザモードで動作する関数で実現できる場合は，この機能を無効にすることができる．

4.2 アーキテクチャ

4.2.1 デバイスドライバ

TINET のイーサネットインタフェースでは，NIC に依存しない汎用イーサネット制御と，NIC に依存するイーサネットドライバから構成されている．このうち汎用イーサネット制御は IP 層から直接呼び出し可能であり，デバイスに直接アクセスしないためユーザモードで実行することが可能である．したがって，汎用イーサネット制御は TCP/IP ドメイン内で実行する．

一方，イーサネットドライバはデバイスアクセスを行い，NIC からの割込みによって割込みハンドラが実行され送受信セマフォを返却する処理を行っている．**要求 3-2** より，MPU を前提として設計するため，TCP/IP ドメインからは直接デバイスアクセスは行えず，イーサネットドライバをユーザモード化することは出来ない．そのため，拡張 SVC として実現し，汎用イーサネット制御から呼び出す．

イーサネットドライバの規模が大きいと**要求 1-1** が満たせないことや，割込み禁止区間や割込みハンドラの実行時間が長いと**要求 1-2** が満たせなくなる．これらの点について，TINET のイーサネットドライバに対する評価は 5 章で実施する．

4.2.2 API 受付処理

ユーザドメインから送受信処理受け付ける方法としては，次の 2 種類の方法が考えられる．

- ユーザ関数方式
- 拡張 SVC 方式

ユーザ関数方式はユーザドメインとして動作する関数を用意する方式で，拡張 SVC 方式は拡張 SVC として動作する関数を用意する方式である．

要求 1-1 からは，ユーザ関数方式が適しているが，以下に挙げる問題がある．TCP/IP 送受信受付用 API は，送受信データが格納されているバッファのポインタを受け取る．送信の場合，バッファのポインタを TCP/IP スタックへ送ると TCP/IP スタックではウィンドウバッファへ書込む．ユーザモード TCP/IP では，TCP/IP スタックは送信元のアプリケーションとは異なるドメインで動作するため，バッファのポインタが指し示すメモリにアクセスできない場合がある．この問題を解決する方法としては，ドメイン間で通信可能なキューを作成して送信データを全てコピーする方法があるが，コピー回数が増えていしまい，**要求 3-1** を満たすことが出来ない．また別の方法として，送受信したいユーザドメインと TCP/IP ドメインのみアクセス可能な共有メモリを用意する方法もあるが，MPU はメモリリージョンの数に制限あるため実現できない．MMU を用いて共有メモリ領域が作成できる場合であっても，送信元のタスクと TCP/IP ドメインからアクセス可能か確認する必要がある．HRP3 カーネルでは，prb_mem() という API を用いてメモリアクセス権をチェックすることが可能である．prb_mem() は指定したタスク ID のタスクから指定したメモリ領域へのアクセスが可能であるかチェックする．しかしながら，TCP/IP ドメインでは送受信 API を呼び出したタスクのタスク ID を知ることができないためこの方法を使うことは出来ない．

一方，拡張 SVC 方式では，呼び出したタスクの延長として動作するため，タスク ID を取得することができ，この問題を解決することができる．ただし**要求 1-1** を逸脱することになるが，拡張 SVC 方式として実行する機能を少なくすることでその影響を小さくすることを目指す．

実現した機構を送信 API（tcp_snd_dat）を例にとり説明する．送信 API の処理は大きくエラーチェック処理と，TCP/IP メイン処理への送信要求処理に分けることができる．エラーチェック処理は，パラメータのチェックと通信端点の状態の確認でありメモリ参照のみのシンプルな処理である．そこで，**図 6** に示すように，TCP/IP 送受信 API をエラーチェック処理を行う TCP/IP 送受信受付用 API と TCP/IP 送受信関数に分割し，TCP/IP 送受信受付用 API を拡張 SVC としてカーネルモードで実行し，エラーチェックが完了した送信要求を TCP/IP ドメイン内に用意したプロキシタスクへ送る．プロキシタスクは，受け取った要求を元に TCP/IP 送受信関数を呼び出し送信を行う．

プロキシタスクの構成方法としては，次の 3 種類が考えられる．

- 全体でプロキシタスクが 1 つ
- ユーザドメインごとにプロキシタスクが 1 つ

Vol.2022-ARC-248 No.33
Vol.2022-SLDM-198 No.33
Vol.2022-EMB-59 No.33
2022/3/10

- 通信端点ごとにプロキシタスクが 1 つ

全体でプロキシタスクを 1 つとする場合は，リソース使用量が少なくてすむが，後述する帯域制御機能の実現に課題がある．ユーザドメインごとにプロキシタスクを 1 つとする場合，プロキシタスク数が多くなるが，ユーザドメイン単位での帯域制御を実現する基盤を提供できると考えられる．通信端点ごとにプロキシタスクを 1 つとする場合，プロキシタスク数が一番多くなるが，通信端点単位での帯域制御が実現可能となる．しかしながら，プロキシタスク数が多くなると，リソースの制約を受けやすくなるとともに，タスク切り換えによる処理時間の増加が懸念される．本研究ではユーザドメインごとにプロキシタスクが 1 つの方式を採用した．

TCP/IP 送受信受付用 API とプロキシタスク間の引数の受け渡しは，HRP3 カーネルの機能であるメッセージバッファを使用した．メッセージバッファのバッファ数は最低限 1 回の API 呼び出しで使用する分で良いが，1 ドメインから並列に要求を出す場合は，複数の呼び出し分用意することも可能である．

プロキシタスクはメッセージバッファにより受け取った要求により，TCP/IP 送受信関数を呼び出す．その際，ブロッキングコールの API を用いると，通信が完了するまでプロキシタスクはブロックされる．前述の通り，プロキシタスクはドメイン毎に用意するため，複数の通信端点を処理する必要がある．複数の通信端点（複数のタスク）から同時に要求が発生する可能性があり，この場合，ブロッキングコールの API を使うとスループットが悪化する．そのため，プロキシタスクが呼び出す TCP/IP 送受信関数はノンブロッキングコールとする．

プロキシタスクがノンブロッキングコールを呼び出すため，通信が完了すると TCP/IP ドメイン内でコールバック関数が呼び出される．このコールバック関数から，TCP/IP 送受信受付用 API でブロックしているユーザドメインのタスクを起床させることとする．この起床は，イベントフラグで実現する．

本節では，送信 API を例にユーザモード化について説明したが，受信 API についてもプロキシタスクを用いた手法によるユーザモード化を実現した．また，他の API についてもプロキシタスクを用いた手法によってユーザモード化実現可能と考える．

4.2.3 バッファのコピー

前述の通りデータのコピー回数を増加させないため，TCP/IP 送受信受付用 API は，送受信データが格納されているユーザドメインのバッファのポインタを受け取り，メッセージバッファにポインタを渡す．メッセージバッファをプロキシタスクが受け取り，TCP/IP 送受信関数にバッファポインタを引数の一つとして渡す．TCP/IP 送受信関数はバッファポインタを参照し，データを送受信ウインドウバッファにコピーする．この際，ユーザドメインのバッファには TCP/IP ドメインからは直接アクセス出来ないため，コピー処理は拡張サービスコールとして実現した．

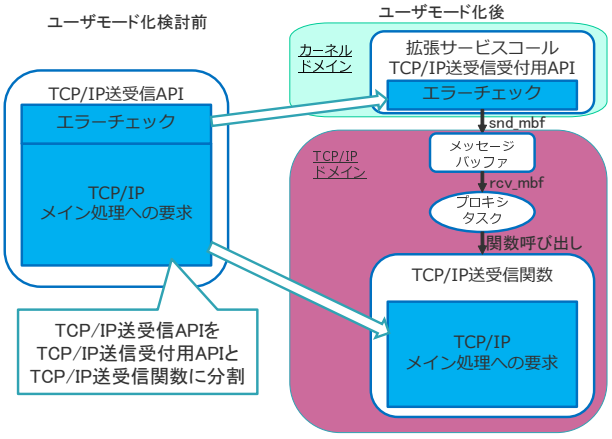

図 6 TCP/IP 送信関数のユーザモード化

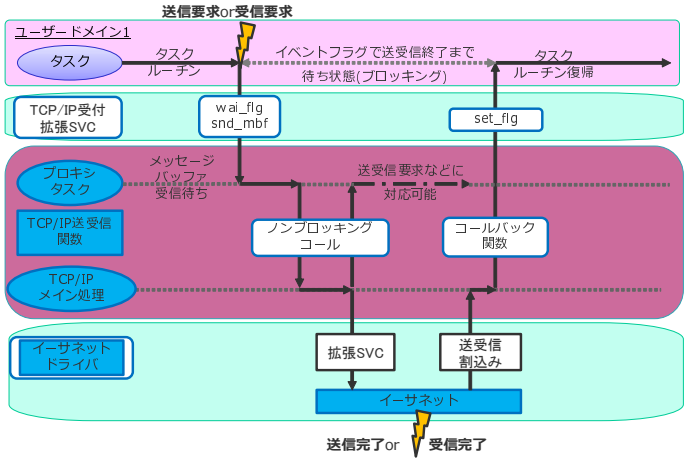

図 7 ユーザモード TCP/IP のシーケンス図

4.2.4 API 処理のシーケンス

実装したユーザモード TCP/IP の送受信 API のシーケンスについて**図 7** に示す．

TCP/IP 受付用 API はユーザアプリケーションから渡された引数をメッセージバッファへ snd_msg（メッセージバッファ送信）で送信する．その後，wai_flg（イベントフラグ待ち）を発行し，待ち状態とする．メッセージバッファへ送信すると，メッセージバッファ受信待ち状態のプロキシタスクが起動し，TCP/IP 送受信関数を実行する．TCP/IP 送受信関数は，TCP/IP のタスクを起動させた後，イーサネットドライバを拡張 SVC で呼び出し実行する．イーサネットでデータの送受信が完了すると，送受信割込みが発生し，TCP/IP のタスクを起床させ，そこからコールバック関数が呼び出される．コールバック関数では，set_flg（イベントフラグのセット）を発行する．イベントフラグ待ちであった，ユーザアプリケーションのタスクは待ち状態が解除され，タスクルーチンに復帰する．

Vol.2022-ARC-248 No.33
Vol.2022-SLDM-198 No.33
Vol.2022-EMB-59 No.33
2022/3/10

4.2.5 MMU 向けの構成

MMU を前提とすると，ユーザモードで実行可能な箇所を増やすことが可能であり，**要求 1-1** をより高いレベルで満たすことが可能である．

イーサネットにアクセスするメモリを HRP3 カーネルでメモリオブジェクトとして登録し，アクセス権を TCP/IP ドメインのみに設定することで TCP/IP ドメインのタスクから関数呼び出しが可能になりユーザモードで実現することができる．

また，前述の TCP/IP 送受信関数における指定されたバッファとウィンドウバッファへ間のコピーに関しても，送受信要求元のユーザドメインと TCP/IP ドメインのみアクセス可能なメモリリージョンを作成することで，拡張 SVC を用いずにコピーが可能となる．ただし，送受信要求元のタスクは送受信するデータをそのメモリリージョンに必ず格納する必要がある．

5. 評価

本章では，提案したユーザモード TCP/IP の評価について説明する．

5.1 実行オーバヘッド（RQ4）

エコー応答時間をカーネルモードで動作させた TINET と提案手法でユーザモード化した TINET とで比較する．エコー応答時間は，クライアントからデータを送信して，そのデータを受信しデータを送り返して，クライアントが受け取るまでの時間を計測した．なお，TDMA スケジューリングは設定していない．TDMA スケジューリングを設定すると TCP/IP ウィンドウが実行されるまでの時間に影響を受けてしまうためである．

実験環境は，ルネサスエレクトロニクス社製の RX64M（120MHz），コンパイラは CC-RX を使用した．計測はクライアント側の windows が動作する PC で wireshark を用いた．

エコー応答を 100 回試行した結果を示す．はじめに，クライアントからの送信データ量 1Byte，ホストからの送信データ量 1Byte で測定した結果を図 8 に示す．次に，クライアントからの送信データ量 1KByte，ホストからの送信データ量 1KByte で測定した結果を図 9 に示す．

図 8 の場合，カーネルモード実行に対してユーザモード実行は，実行オーバヘッドが増加しているのが確認できる．図 9 の場合，カーネルモード実行に対するユーザモード実行の実行オーバヘッドの割合が少ない．これは，ユーザモード化すると，TCP/IP スタックの呼び出し処理の実行オーバヘッドは増加するが，その後のパケット処理自体は変わらないためであると考えられる．つまり，実行オーバヘッドは追加された機能によるもので，そのオーバヘッドが大きく増加しないと考えられる．そのため，**RQ3** は満

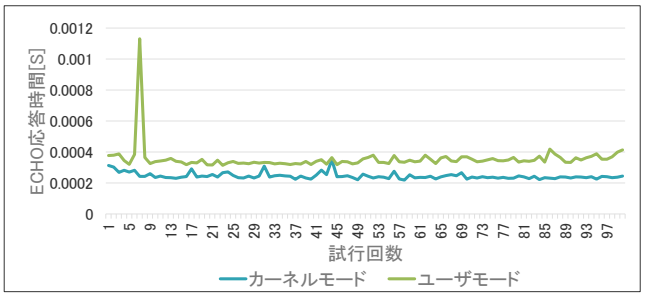

図 8 エコー応答オーバヘッド（Request 1Byte, Response 1Byte）

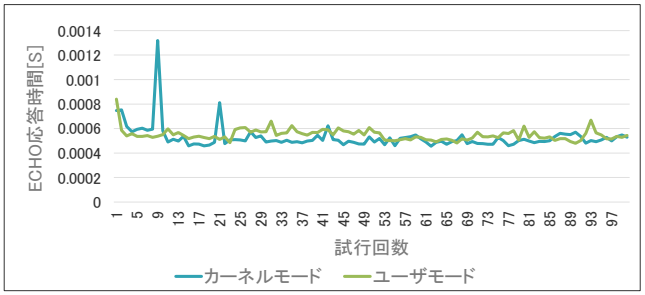

図 9 エコー応答オーバヘッド（Request 1KByte, Response 1KByte）

たすことができたと考えられる．なお，外れ値についてはネットワーク通信であることや，ログ出力のための RTOS の機能によって発生していると考えられる．

5.2 ユーザモード化出来ない箇所（RQ1）

5.2.1 API 呼出し処理

TCP/IP 送受信受付用 API はカーネルモードで動作し，コードサイズは送受信ともに 50 行程度である．さらに，パラメータチェック等の読み込み処理とパラメータのメッセージバッファへの送信の処理であるため，検証コストは容易と考える．また，割込みは禁止しないため，TDMA スケジューリングへの影響はない．

5.2.2 イーネット送受信割込み

イーサネット送受信割込みは，ユーザドメインのタイムウィンドウを中断して実行され，タイムウィンドウの切り換えの遅延も発生させる可能性がある．そこで，イーサネット送受信割込みの実行時間を評価するとともに，割込みの実行時間を踏まえた TDMA スケジューリングの設定方法を提案する．5.1 で実行したエコー応答時間の評価シーケンス（request 1KByte, response 1KByte）を 100 回試行した際の割込みの実行時間を調査する．

イーサネット送受信割込みの頻度多い区間を通信区間とし，その通信区間の中でイーサネット割込みが発生した時間の解析結果を**表 1** 示す．

全体の通信時間に対する送受信割込み実行時間の割合，つまり TDMA スケジューリングに与える影響は，0.018% であることがわかった．送受信割込みの実行時間の最大値は 2.3μsec である．送受信割込みは一度発生すると次に

表 1 イーサネット送受信割時間込み時間

項目	結果
通信区間全体時間 [$msec$]	3,042.5
送受信割込み実行時間（合計）[μsec]	564.8
割込み実行時間の割合	0.018 %
送受信割込み実行時間（最大値）[μsec]	2.3
割込み発生回数 [times]	299

TCP/IP スタックを割り当てたタイムウィンドウが実行されるまで発生することはない．そのため，システム周期中の n 個のタイムウィンドウに TCP/IP スタックを割り当てると，最悪で送受信割込みにより $2.3 * n\,\mu sec$，タイムウィンドウの切り換えが遅延される．文献 [5] より，HRP3 カーネルのタイムウィンドウ切り替えの時間が $4.6\mu sec$ であるため，大きな値でないと言える．また，発生する遅延の最大数も見積ることが可能であり，その分アイドルウィンドウを確保すれば，安全系への影響はないと言える．

5.2.3 イーサネットドライバ

イーサネットドライバを拡張 SVC としてカーネルモードで実行する影響を評価する．

コード規模に関しては，500 行程度であり，システムの最高の安全度水準で作成する必要がある HRP3 カーネルのコード行数 39,000 行と比較して十分小さいため，HRP3 カーネルと同様にシステムの最高の安全度水準で開発することは可能と考える．

次に，イーサネットドライバの各関数の実行時間を計測した．実行時間の最大値はフレーム読込み関数の $45.4\mu sec$ であった．フレーム読込み関数は，読込み前にイーサネット PHY のリンクアップ状態を確認する．今回の実験環境ではこの状態を低オーバヘッドで取得する方法がなく，1Bit 単位の通信でイーサネット PHY に問い合わせる必要があり実行時間が増加した．この様に実行時間が長いが，フレーム読込み関数は割込みを禁止せずに動作するため，TDMA スケジューリングへ影響を及ぼさない [5]．一方，NIC リセット関数はイーサネットドライバで割込み禁止を使用している．NIC リセット関数は接続時に一度だけ呼び出される処理であるため，システムへの影響は少ないと考えられる．

以上のように，イーサネットドライバはコード規模が大きくないこと，リセット処理を除いて TDMA スケジューリングへ影響を及ぼさないことを確認した．

6. 関連研究

イーサネットドライバのユーザモード化に関する研究として，汎用 OS では，virtio ドライバと呼ばれる準仮想化ネットワークドライバの vhost-user モードを利用して，ユーザモードでパケット処理を行う Intel DPDK というフレームワークが存在する [6]．しかし，このフレームワークはカーネルモードを介さずに通信することによる高速化を目的としており，パーティショニングという今回の目的とは異なる．また，DPDK フレームワークでは使用している NIC も特殊なものであり，本研究では採用できないと判断した．

専用ドメインにより TCP/IP スタックを用意する手法は，車載システム向けの仮想マシンを対象とした手法が提案されている [7]．提案手法では，各 VM は共有メモリを持たないため，専用の FIFO を用意している．また，各 VM はコアを専有しており，ブロック等が発生するかは説明されていない．本研究ではパーティショニング OS を用いるため，ドメイン間の通信に OS 間通信機能が使えるため，先行研究と比較して，OS の待ち状態を用いた効率的な通信が可能である．

7. まとめ

パーティショニング OS でミドルウェアを低コストで利用するために，TCP/IP スタックを題材として，ミドルウェアのユーザモード化と共有を目的とした TCP/IP スタックの実現方法について提案した．設計・実現可能なユーザモード TCP/IP スタックを実現したうえで，その性能の評価や性質の評価を行った．その結果，可能な限りユーザモードで動作する TCP/IP スタックを実現することができた．今後の課題としては，帯域制御の実現が挙げられる．また，本研究では MPU を対象として実装を行ったが，MMU を対象とする場合より高信頼で効率的な TCP/IP スタックを提案できると考えられる．

参考文献

[1] 伊藤 弘将，松原 豊，高田 広章，”ミドルウェアに対する Coverage-based Greybox Fuzzing の適用”，情報処理学会論文誌（2021）．
[2] 三菱重工株式会社，”三菱重工技報 Vol.58 No.4 （2021） 航空宇宙特集”，入手先 ⟨http://www.mhi.co.jp/technology/review/pdf/584/584070.pdf⟩（2022.01.31）．
[3] 阿部 司，吉村 斎，久保 洋，”組込みシステム用 TCP/IP スタックの実装と評価”，情報処理学会論文誌（2003）．
[4] Embedded TCP/IP 技術委員会，トロン協会 ITRON 専門委員会: ”ITRON TCP/IP API 仕様 Ver 1.00.01”，（1996）．
[5] 手塚湧太郎，本田晋也，大谷寿賀子，枝廣正人，”時間パーティショニング機構を持つリアルタイム OS の性能評価手法”，第 55 回組込みシステム研究発表会（2021）．
[6] Gabriele Ara, Tommaso Cucinotta, Luca Abeni, Carlo Vitucci, *Comparative Evaluation of Kernel Bypass Mechanisms for High-performance Inter-container Communications* ,CLOSER 2020: Proceedings of the 10th International Conference on Cloud Computing and Services Science,（2020）．
[7] Dominik Reinhardt，Maximilian Güntner，S. Obermeir: *Virtualized Communication Controllers in Safety-Related Automotive Embedded Systems* ARCS 2015， pp 173-185,（2015）