組込みシステム用TCP/IPプロトコルスタックの実装と評価

阿部 司[†] 吉村 斎[†] 久保 洋[††]

組込みシステムにおいては，メモリ容量，重量，サイズおよび消費電力等の制約がある．そこで，インターネットに接続するための TCP/IP プロトコルスタックも組込みシステムの制約を考慮しなければならない．ITRON リアルタイムカーネルを用いた組込みシステム用の ITRON TCP/IP API 仕様は，この制約を考慮している．本論文では，ITRON TCP/IP API 仕様による TCP/IP プロトコルスタックの実装を報告する．本実装は，組込みシステムの厳しいリソース制約を満足し，必要不可欠な通信機能だけに厳選している．UNIX 等のオペレーティングシステムで一般的に使用されているソケットインタフェースは，汎用計算機での実装や利用実績が多い反面，高度な抽象化を実現するため多くのリソースを要求する．評価では，ソケットインタフェースによる TCP/IP プロトコルスタックの実装との比較も行い，ITRON TCP/IP API 仕様による実装の有効性を確認した．

Implementation and Evaluation of TCP/IP Protocol Stack for Embedded System

Tsukasa Abe,[†] Hitoshi Yoshimura[†] and Hiroshi Kubo[††]

An embedded system has restrictions, such as memory capacity, weight, size, and power consumption. For this reason, TCP/IP protocol stack for connecting with the Internet must also take restrictions of an embedded system into consideration. The ITRON TCP/IP API Specification for the embedded systems using the ITRON real-time kernel is taking these restrictions into consideration. This paper describes implementation of a TCP/IP protocol stack by ITRON TCP/IP API Specification. This implementation satisfies severe resource restrictions of an embedded system, and is selecting them carefully only to the indispensable communication features. While the socket interface generally used by operating systems such as UNIX has implemented and used in many general-purpose computers, but it requires many resources in order to realize advanced abstraction. In our evaluations, comparison with implementaton of the TCP/IP protocol stack by the socket interface was also performed, and the validity of implemention by the ITRON TCP/IP API Specification was confiremed.

1. はじめに

インターネットは，常時接続の普及等により，生活の中で必要不可欠な媒体になりつつある．パーソナルコンピュータや汎用計算機だけでなく，家電製品，自動車等の各種の装置に組み込まれている組込みシステムにおいても，インターネットに接続するための TCP/IP プロトコルスタックが必要不可欠になっている．

インターネットは，UNIX と相互に発展してきたため，応用層のプログラミングには，UNIX で開発されたソケットインタフェースを使用することが汎用計算機では一般的である．ソケットインタフェースは，汎用的なネットワーク API から構成され，多様なネットワークシステムにおいて統一的で抽象化されたインタフェースを提供するため，インターネットのみのような単一のネットワークシステム上でのプログラミングにはオーバヘッドが大きい．また，物理層も仮想化されているため，物理層にも多数の階層が存在する．

組込みシステムには，メモリ容量，重量，サイズおよび消費電力等の厳しいリソース制約があるため[1),2)]，汎用計算機と同等の高度に抽象化された通信機能を実装することは現実的ではない．そこで，ネットワークシステムをインターネットに限定し，厳しいリソース制約のある組込みシステムへの実装を目的にした ITRON TCP/IP API の仕様[3),4)] が公開されている．

本研究では，当初，FreeBSD をベースに擬似カーネルを用いてソケットインタフェースを実装した[5)] が，組込みシステムの厳しいリソース制約を満足することが困難であることが判明した．そこで，オペレーティ

† 苫小牧工業高等専門学校
Tomakomai National College of Technology
†† 室蘭工業大学
Muroran Institute of Technology

表 1 削除した通信機能
Table 1 Unimplemented networking features.

階層	削除した通信機能
TCP	MSS 以外のオプション，拡張通信機能， 通信相手の各種情報のキャッシング
IP	オプション，経路選択と経路表， 分割と再構成，物理ネットワーク層の選択機能
共通	経路選択と経路表， mbuf およびプロトコル制御ブロックの処理

ングシステムをμITRON 4.0 仕様準拠[6] カーネルに変更したうえで，組込みシステムの厳しいリソース制約を満足し，必要不可欠な通信機能だけに厳選したITRON TCP/IP API 仕様準拠の TCP/IP プロトコルスタックの実装を試みた．ソケットインタフェースによる実装ついては，本研究で提案した実装の有効性を確認するため比較評価に使用した．

2. 組込みシステムに求められる通信機能

組込みシステムの規模には 8 ビットから 64 ビットと幅がある．本研究では，以下に示す規模の組込みシステムを対象としている．

- 必要なメモリ量は，ROM が 128 K バイト，RAM が 32 K バイト程度．
- TCP/IP の各ヘッダは，32 ビット単位で処理すると効率が良いため，C 言語の int が 32 ビットで，レジスタと ALU も 32 ビットが望ましい．
- 仮想記憶システムやメモリ保護機能等の高度なメモリ管理機能と二次記憶装置を持たない．

以上のような組込みシステムに対して，本研究で提案する ITRON TCP/IP API による実装では，次に示す通信機能を実装した．また，ベースの FreeBSD[7] から削除した通信機能を**表 1** に示す．

2.1 応用層プロトコル

現在，インターネットで使用頻度の高い応用層プロトコルは，HTTP である．一方，OpenSOAP 等のインターネットで交換可能な標準的なデータ記述方式として注目されている XML[8] は，HTTP で転送され，組込みシステムにおいても XML によるメッセージ交換が重要と考えている．本研究においても，応用層プロトコルとして HTTP をターゲットとしており，リソース制約から，HTTP のバージョンは，1.0（RFC 1945[9]）で，要求・応答ごとに接続・切断を行う方式を想定している．

2.2 単一リンクの終端ノード

ネットワークにおける組込みシステムは，その性格上，末端の終端ノードとして動作することが求められることが多い．したがって，本研究では対象を末端の終端ノードに限定した．また，隣接ノードとの接続に関しては単一リンクによる接続形態とし，経路選択プログラムと経路表を不要にした．

2.3 単一の物理ネットワーク層

末端の終端ノードに限定すれば，単一の物理ネットワーク層に限定することも可能である．これにより，物理ネットワーク層の選択機能が不要になる．

2.4 IP での分割と再構成

IP データグラムの送信時の分割と受信時の再構成の実装には，プログラムだけでなく，分割されたすべてのデータグラムを保持するためのメモリが必要である．

TCP では，コネクション開設時に最大セグメントサイズ（以下 MSS）のネゴシエーションを終端間で行うことにより，IP での分割を防ぐことができる．ネゴシエーションなしでも，TCP の MSS の規定値は，536 オクテットであり，TCP と IP のヘッダ長を加えた 576 オクテット（RFC 1122[10]）の IP データグラムを物理ネットワーク層で転送できればよい．

UDP は，TCP の MSS のネゴシエーションのような機能を持たないが，UDP を使用する応用層では，DNS のようにデータを 512 オクテット以下（RFC 1035[11]）に制限している場合が多く，問題となることは少ない．

2.5 TCP と IP のオプション

TCP と IP における付加可能なオプションを限定することにより，あらかじめ送信用に固定長のメモリブロックを割り当てることができる．

IP には，各種オプションが存在するが，一般的な通信には冗長なオプションであり，組込みシステムの TCP/IP プロトコルスタックに実装する必要性は低い．

TCP も，オプションに関しては同様であるが，MSS の通知オプションだけは，IP データグラムの分割を防ぐために必要である．また，コネクション開設時のみ使用されるオプションであり，固定的なメモリブロックを割り当てることができる．

3. ソケットインタフェースの問題

ソケットインタフェースは，汎用的な API から構成され，多様なネットワークシステムにおいて統一的で抽象化されたインタフェースを提供するため，組込みシステムにおいては，以下に述べるとおり，メモリブロック割当て，コネクション制御構造体および階層での抽象化によるオーバヘッドが問題となる．

3.1 メモリブロック割当て

各階層のヘッダとペイロードを保持するメモリブロックを，ソケットインタフェースでは，固定長のメ

モリブロック（mbuf）を動的に割り当て，連結リストにより，メモリ間のコピーを行わない方法で解決している．mbuf の大きさは 128 バイト（FreeBSD では，バージョン 4.1 以降は 256 バイト）である．mbuf のヘッダは 20 バイトで，さらに，パケットの先頭に使われる mbuf には 12 バイトのパケットヘッダが付加される．したがって，設定可能なペイロードは 108 バイトか 96 バイトになる．ペイロードの大きさがこれ以上の場合，複数の mbuf を使用すると効率が悪いため，2,048 バイトの大きさの mbuf クラスタを使用している．mbuf の問題点を以下に示す．

- 各階層で，割当て関数を何度も呼び出す必要がある．また，割当て可能になるまで実行がブロックされる可能性があり，リアルタイム性に問題がある．
- 連結リスト内の各階層のヘッダやペイロードへのアクセスを間接的にしか行えない．そこで，mbuf の連鎖を再配置し，連続的にメモリに配置するため動的な mbuf の割当てが行われることがある[12]．
- PPP の標準的なメッセージ転送単位（以下 MTU）である 1,500 オクテット[13] と，プロトコルフィールドの 2 オクテットからなる PPP フレームを構成するためには，それぞれ 1 個の mbuf と mbuf クラスタを使用する．このとき，ヘッダ+ペイロードに対するヘッダの比率は約 2%であるが，使用率は約 69%であり，実装効率が低い．

3.2 コネクション制御構造体

TCP では，以下に示す構造体によりコネクションの状態を管理している．各階層での独立性と汎用性を維持するため，ポインタが多用されており，インターネットのみのような単一のネットワークシステムには不要な項目も多い．7.1 節「ITRON TCP/IP API の評価」に詳細を示すが，ソケットインタフェースのコネクションあたりの制御構造体の合計は 1,256 バイトで，メモリ必要量が多い．

- ファイル記述子
- ソケット
- TCP 制御ブロック
- プロトコル制御ブロック

3.3 階層での抽象化

ソケットインタフェースは，TCP/IP だけでなく各種のネットワークシステムに適合させるため，各階層で高度な抽象化を行っている．

データ送信時に使用する send 関数から呼び出される sendit 関数の主要部分を**図 1** に示す．8 行目で，送信相手のソケットを取り出し，9〜10 行目で，実際に送信を行う so_proto でトランスポート層プロトコルを選択する．次に，その中からソケット層とのインタフェースである pr_usrreqs を選択し，最後に，送信関数 pru_sosend を呼び出している．このように，各種構造体へのポインタを使って間接的にトランスポート層の出力関数を呼び出す必要がある．IP の出力でも，経路選択後，同様の処理を行い，適当な物理ネットワーク層を選択している．

```
 1: static int
 2: sendit(p, s, mp, flags, ssize)
 3:   ...
 4: {
 5:   ...
 6:   error = getsock(p->p_fd, s, &fp);
 7:   ...
 8:   so = (struct socket *)fp->f_data;
 9:   error = so->so_proto->pr_usrreqs
10:             ->pru_sosend(...);
11:   ...
12:   }
```

図 1 sendit 関数
Fig. 1 Function sendit.

4. 実装に使用した組込みシステムの構成

ITRON TCP/IP API による実装，ソケットインタフェースによる実装および擬似カーネルは，FreeBSD バージョン 3.4 をベースにしている．

（社）トロン協会[14] が 2000 年から 2001 年にかけて行った調査では，μITRON 仕様に準拠したリアルタイムカーネル（以下，RTOS）が全体の約 29%のシェアを占めており，組込みシステムの RTOS として定着している．本研究の ITRON TCP/IP API の実装では，RTOS として豊橋技術科学大学，組込みリアルタイムシステム研究室で開発された μITRON 4.0 仕様準拠の TOPPERS/JSP カーネル[15] を用いた．

実装のターゲットには（株）日立製作所製 H8/300H シリーズの H8/3048F[16] を使用した．H8/300H シリーズは，16 ビット CPU であるが，レジスタと ALU が 32 ビットであり，本研究で対象とした組込みシステムの規模に合致している．H8/3048F は，内蔵 ROM は容量に余裕があるが，内蔵 RAM は小容量なため外部に RAM を増設している．

5. ITRON TCP/IP API による実装

ITRON TCP/IP API 仕様の概要を述べる．本仕様は，（社）トロン協会の Embedded TCP/IP 技術委員会により策定され，ITRON 専門委員会により 1998 年 5 月に認定された．仕様では，組込みシステムのプロトコルスタックに求められる以下に示す性質があげられている．

表 2 実装した ITRON TCP/IP API
Table 2 ITRON TCP/IP API.

関数名	処理内容
TCP_CRE_REP	TCP 受付口の生成
TCP_CRE_CEP	TCP 通信端点の生成
tcp_acp_cep	接続要求待ち（受動オープン）
tcp_con_cep	接続要求（能動オープン）
tcp_sht_cep	データ送信の終了
tcp_cls_cep	通信端点のクローズ
tcp_snd_dat	データの送信
tcp_get_buf	送信用バッファの獲得
tcp_snd_buf	バッファ内のデータの送信
tcp_rcv_dat	データの受信
tcp_rcv_buf	受信用バッファの獲得
tcp_rel_buf	受信用バッファの開放
UDP_CRE_CEP	UDP 通信端点の生成
udp_snd_dat	パケットの送信
udp_rcv_dat	パケットの受信

(1) 最小のコピー回数.
(2) 動的メモリ管理の排除.
(3) 非同期インタフェース.
(4) API ごとのエラーの詳細化.

また，その他の設計目標として，以下の項目が設定されている．

(1) ソケットインタフェース互換ライブラリ作成の容易性.
(2) プロトコルの種類ごとの最適な API の定義.
(3) RTOS との適合性.
(4) システム構築の静的な設定の考慮.

TCP では，標準 API が 14 個，拡張 API が 7 個定義されており，UDP では，標準 API が 4 個，拡張 API が 4 個定義されている．また，ソケットインタフェースのコネクション制御構造体に対応する TCP/UDP 通信端点および TCP 受付口が定義されている．

本研究では，組込みシステムの厳しいリソース制約に対応するため，組込みシステムに求められる通信機能に限定した ITRON TCP/IP API 仕様の TCP/IP プロトコルスタックを実装した．実装した機能を以下に示す．

5.1 通 信 機 能

(1) 応用プログラムインタフェース
実装した応用プログラムインタフェースを**表 2** に示す．ソケットインタフェースによる実装と同等の応用プログラムインタフェースを実装した．

(2) TCP/IP プロトコルスタック
4 章に示した ITRON TCP/IP API 仕様のプロトコルスタックに求められる性質中，(1) 最小のコピー回数と，(2) 動的メモリ管理の排除の実装を優先し，(3) と (4) については必要な機能のみ実装した．

(3) 物理ネットワーク層
フレームを折り返すループバックインタフェースを実装した．

1	1	2	0 or 2	64〜ヘッダ+MTU
idix	unit	len	align	buf

図 2 ネットワークバッファ構造体（net_buf）
Fig. 2 Network buffer structure (net_buf).

5.2 メモリブロック割当て

TCP のコネクション開設時に，MSS オプションが付加される以外，IP と TCP のオプションは付加されないため，あらかじめ各階層のヘッダとペイロードを保持できる固定長のメモリブロックであるネットワークバッファ（以下 net_buf）を割り当て，プロトコル処理の途中での動的なメモリ割当ては行わない．

net_buf の構造を**図 2** に，その内容を以下に示す．

(1) idix：固定長メモリプールの ID 番号を保持する配列のインデックスである．
(2) unit：将来の拡張用で，物理ネットワーク層を選択する．
(3) len：データ長で，種類は 64，128，256，512，1,024 と，物理ネットワーク層の MTU にヘッダを加えた長さである．種類と個数は，μITRON RTOS のシステムコンフィグレーションファイルで，静的 API「固定長メモリプールの生成」により指定する．
(4) align：IP ヘッダ以降を 4 バイト境界に整列するためのダミーである．TCP/IP の各ヘッダは，32 ビット単位で処理すると効率が良く，32 ビット境界に整合する必要のある CPU も存在する．
(5) buf：ヘッダとペイロードが入る．受信時には，物理ネットワーク層の MTU のペイロードとヘッダを保持できる大きさの buf から構成される net_buf を割り当てる．PPP では，buf の大きさは，標準的な MTU の 1,500 オクテットにプロトコルフィールドの 2 オクテットを加えた 1,502 オクテットである．

6. ソケットインタフェースによる実装

ITRON TCP/IP API による実装の有効性を確認する比較評価に使用するため，以下に示す通信機能と擬似カーネルを実装した．

6.1 実装した通信機能

(1) 応用プログラムインタフェース
応用プログラムが動作できるだけの関数が必要

表 3 実装したソケットインタフェースとファイル操作関数
Table 3 Socket interfaces and file manipulating functions.

関数名	処理内容
socket	ソケット生成
bind	ソケットへのアドレス設定
listen	接続要求待ち（受動オープン）
accept	接続要求応答の完了（受動オープン）
connect	接続要求（能動オープン）
shutdown	データ通信の終了
recv	データの受信
send	データの送信
recvfrom	データの受信
sendto	データの送信
close	ソケットのクローズ
read	データの受信
write	データの送信

であり，**表 3** に示すソケットインタフェース関数とファイル操作関数を実装した．

(2) 汎用通信機能
プロトコル制御ブロックと物理ネットワーク層の管理，経路選択を実装した．

(3) TCP/IP プロトコルスタック
TCP，UDP，IP および ICMP を実装した．TCP においては，性能向上のため拡張機能であるウィンドウサイズのスケーリング（RFC 1323[17]）とトランザクション（RFC 1644[18]），通信相手ごとの各種情報のキャッシュを実装したが，トランスポート層のファイアウォール機能は削除した．また，IP においては，分割・再構成機能，物理ネットワーク層の選択機能等を実装したが，中継機能とネットワーク層のファイアウォール機能は削除した．

(4) 物理ネットワーク層
ループバックインタフェースを実装した．フレームを折り返す機能，ディバイスの割当て，初期化および IOCTL 機能を実装している．

6.2 擬似カーネル

通信機能と評価用の応用プログラムを実行するためには，オペレーティングシステムが提供する各種サービスが必要である．特に，本研究では FreeBSD をベースにしているため，仮想記憶管理やファイルシステム等がソケットインタフェースに深く関わっている．しかし，全機能の実装は，対象とした組込みシステムの利用可能なメモリの制約から不可能であるため，本研究では，通信機能と評価用の応用プログラムを実行するために最低限必要なオペレーティングシステムの機能のみ実装した．以下にその機能を示す．

(1) プロセス管理と時間管理
クライアントプロセスとサーバプロセスを生成し実行する機能を実装している．スケジューリングは，ノンプリエンプティブ方式で，待ち状態でブロックされるまでプロセスを実行する．

(2) 仮想記憶管理とメモリ管理
仮想記憶管理は，論理的なページ管理機能をシミュレートしている．またメモリ管理は，十分な容量のメモリを割り当てて，ブロックを回避した．

(3) ファイルシステム
ファイルシステムの実体は実装せず，それへのアクセスをバイパスし，ファイル記述子管理と IOCTL 関数の一部を実装した．

7. 評　　価

TCP と UDP の ECHO サーバ・クライアントと同等の機能を実現する応用プログラムと，ループバックインタフェースを用いて，ITRON TCP/IP API による実装と，ソケットインタフェースによる実装のそれぞれについて，評価システムを作成し，メモリ必要量を中心に比較評価した．

7.1 ITRON TCP/IP API の評価

ITRON TCP/IP API の評価を以下に示す．いずれもメモリ必要量に関して効果が認められた．なお，以下に示す (1) から (3) はデータ（RAM）のメモリ必要量に関係し，(4) はプログラム（ROM）のメモリ必要量に関係している．

(1) TCP におけるソケットの複製
ITRON TCP/IP API では TCP 通信端点を渡すため，受動オープン状態で，コネクションが開設されたときのソケットの複製に相当する動的なメモリブロックの割当ては行わない．

(2) 省コピー API
ITRON TCP/IP API の特殊な機能として TCP の省コピー API がある．7.5 節で示す WWW サーバでの使用例を**図 3** に示す．get_char 関数は，バッファから 1 文字読み出す関数である．5 行目で，バッファに文字がなくなったとき，6 行目の tcp_rel_buf 関数を呼び出し，使用済バッファを返却し，7〜8 行目の tcp_rcv_buf で新たなバッファを獲得している．このように TCP 内のバッファを直接アクセスし，応用プログラムと TCP 間のデータのコピーを省いている．

(3) コネクション制御構造体
ソケットインタフェースのコネクションの制御構造体は，ITRON TCP/IP API では，TCP が

```
 1: static int
 2: get_char (ID cepid, T_BUFFER *rbuf)
 3: {
 4:   int len;
 5:   if (rbuf->index >= rbuf->len) {
 6:     tcp_rel_buf(cepid, rbuf->len);
 7:     if ((len=tcp_rcv_buf(cepid,
 8:          (VP*)&rbuf->buf,TMO_FEVR))<=0)
 9:       return EOF;
10:     else {
11:       rbuf->len   = len;
12:       rbuf->index = 0;
13:       }
14:     }
15:   return rbuf->buf[rbuf->index ++];
16:   }
```

図 3 省コピー API の例
Fig. 3 Example of zero copy API.

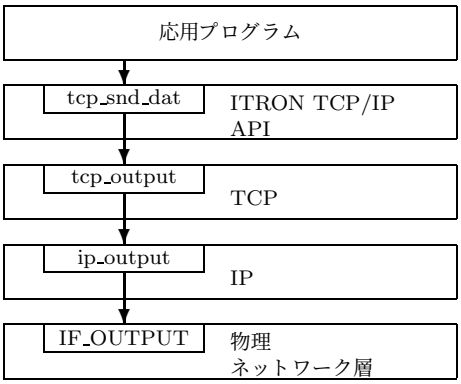

図 4 層間インタフェース
Fig. 4 Interface between layers.

表 4 制御構造体の割当て回数
Table 4 Number of allocations for structures.

構造体	回数 [回]	サイズ [バイト]	小計 [バイト]
ファイル記述子	3	28	84
ソケット	2	160	320
TCP 制御ブロック	3	188	564
プロトコル制御ブロック	3	96	288
合　計			1,256

TCP 通信端点と TCP 受付口，UDP が UDP 通信端点に集約され，簡素化されている．

ソケットインタフェースの TCP の 1 コネクションに対する制御構造体の割当て回数と各構造体のサイズを**表 4** に示す．合計 1,256 バイトになった．これに対して，ITRON TCP/IP API では，同様の機能を持つ構造体の TCP 通信端点が 196 バイト，TCP 受付口が 20 バイト，割当て数はそれぞれ 2 で，合計 432 バイトとなり，約 1/3 のメモリ必要量となった．

(4) 階層での抽象化

ソケットインタフェースでは，図 1 に示したように，各種構造体へのポインタを使って間接的に各層の出力関数を呼び出さなければならないが，ITRON TCP/IP API は，単一ネットワークシステムを対象としており，TCP と UDP ごとに API の関数が別となっている．そこで，**図 4** に示すように，API から直接 TCP および UDP の処理関数を呼び出すことが可能であり，各層の出力関数も直接呼び出すことができる．また，単一の物理ネットワーク層に限定しているため，その選択も不要である．

7.2 net_buf の評価

メモリブロック net_buf に関する評価を以下に示す．

(1) ソケットインタフェースの mbuf の割当て回数

表 5 mbuf の割当て回数
Table 5 Number of allocations for mbuf.

メモリブロック名称	回数 [回]	サイズ [バイト]
mbuf	2	128
パケットヘッダ付き mbuf	13	128

を**表 5** に示す．クライアントからの要求とサーバからの応答の 1 回で，mbuf が 2 回，パケットヘッダ付き mbuf が 13 回の割当てが行われた．それに対して，net_buf は，コネクションの開設時に 3 回と，クライアントとサーバのデータの送信時に，それぞれ 1 回だけ割り当てられ，回数が大幅に減少した．

(2) UDP の送信時には udp_snd_dat 関数内で，応用プログラムのバッファから net_buf へデータのコピーがあり，受信時には udp_rcv_dat 関数内で，逆に net_buf から応用プログラムのバッファへデータのコピーがあるが，物理ネットワーク層，IP および UDP でのデータのコピーはない．

(3) TCP では，UDP と同様に，tcp_snd_dat 関数と tcp_rcv_dat 関数内で応用プログラムのバッファと通信端点内の送受信ウィンドバッファ間のデータのコピーが行われるほか，通信端点内の送受信ウィンドバッファと net_buf 間でコピーが行われる．しかし，図 3 に示したように，TCP での省コピー API により，応用プログラムのバッファと通信端点内の送受信ウィンドバッファ間のデータのコピーをなくすことができる．

7.3 通信機能の評価

通信機能のメモリ必要量を**表 6** に示す．なお，表の値は，データ（RAM）とプログラム（ROM）の合計

表 6 通信機能のメモリ必要量
Table 6 Memory requirement for communication features.

機能	ソケット [バイト]	ITRON [バイト]	比率
API	28,590	4,396	15.4%
TCP	25,722	12,870	50.0%
UDP	3,306	634	19.2%
ICMP	2,174	1,202	55.3%
IP	15,480	976	6.3%
TCP/IP 共通機能	13,692	276	2.0%
汎用通信機能	16,504	442	2.7%
バッファ管理	6,466	426	6.6%
通信機能合計	111,934	21,222	19.0%

メモリ必要量である．また，比率は，ソケットインタフェースに対する ITRON TCP/IP API のメモリ必要量の比を表している．表 6 に示すとおり，ソケットインタフェースの 19%のメモリ容量で ITRON TCP/IP API を効率的に実装できた．

(1) 応用プログラムインタフェース（表では API）の比較では，ソケットインタフェースは，表 3 に示した関数のメモリ必要量が大きいだけでなく，トランスポート層とのインタフェースを実装するためのメモリも必要としている．それに対して，ITRON TCP/IP API は，トランスポート層を直接呼び出すため，層間のインタフェースにおけるメモリ必要量は少ない．

(2) TCP では，拡張機能やオプションの処理，通信相手の各種情報のキャッシング等を省いたことにより，メモリ必要量は減少している．TCP の核となる機能を省くことはできないため，全体の削減効果に比べると，その効果は小さいが，50%の削減となった．

(3) UDP では，ITRON TCP/IP API の関数に UDP 処理機能の一部が移動し，mbuf とプロトコル制御ブロックの処理を省いている．

(4) ICMP では，核となる機能の実装が変わらないため，全体のメモリ必要量の削減効果に比べると，その効果は小さいが，55.3%となった．

(5) IP では，比率が 6.3%と大幅にメモリ必要量を削減できた．その削減要因を以下に示す．
- 単一リンクの終端ノードに限定することで，経路選択と経路表が不要になった．
- 単一の物理ネットワーク層に限定し，選択機能が不要になった．
- 分割と再構成を省き，分割して受信した全データグラムの保持が不要になった．

(6) TCP/IP 共通機能と汎用通信機能では，経路選択と経路表，mbuf とプロトコル制御ブロックの処理が不要になった．そこで，比率が約 2%と大幅にメモリ必要量を削減できた．

(7) バッファ管理では，net_buf の割当てと解放以外の機能が不要になったため，比率が約 6%と大幅にメモリ必要量を削減できた．

表 7 評価システム全体のメモリ必要量
Table 7 Memory requirement for all system.

機能	ソケット [バイト]	ITRON [バイト]	比率
応用プログラム	9,040	4,746	52.5%
ループバック	1,720	40	2.3%
カーネル	43,091	32,666	75.8%
メモリブロック	20,480	16,446	80.3%
通信機能合計	111,934	21,222	19.0%
合　計	165,785	75,120	45.3%

7.4 評価システム全体の評価

表 7 は，評価システム全体のメモリ必要量を示している．なお，表の値は，データ（RAM）とプログラム（ROM）の合計メモリ必要量である．ソケットインタフェースの約 45%のメモリ必要量で ITRON TCP/IP API を実装することができた．

(1) 機能「応用プログラム」では，ソケットインタフェースは，カーネルの初期化やプロセス生成等を行うため，メモリ必要量は多いが，基本的な機能の実装のためのメモリ必要量に差はない．

(2) 機能「ループバックインタフェース（表ではループバック）」では，ソケットインタフェースは，ディバイス割当て，初期化，IOCTL 機能を実装しており，抽象化によるオーバヘッドが大きい．それに対して，ITRON TCP/IP API は，IP の入力関数を直接呼び出すので 2.3%のメモリ必要量となった．

(3) 機能「カーネル」は，75.8%のメモリ必要量となった．ソケットインタフェースは，比較評価用の応用プログラムが実行できるにすぎないのに対し，TOPPERS/JSP カーネルは μITRON 4.0 仕様のスタンダードプロファイルを完全に実装しており，機能差はメモリ容量比以上ある．

(4) 機能「メモリブロック」に関して，ソケットインタフェースでは，mbuf やコネクション制御構造体等を割り当てる擬似カーネルの仮想記憶管理プログラムと割当て用のデータであり．ITRON TCP/IP API では，net_buf の管理プログラムと，net_buf 用のデータである．表の値は，通信機能以外に使用されるデータも含んでおり，通信機能に限定すると 6,880 バイトが割り当てられ，ソケットインタフェースに対し

表 8 WWW サーバのメモリ必要量
Table 8 Memory requirement for WWW server.

機能	RAM [バイト]	ROM [バイト]	計 [バイト]
WWW サーバ	726	8,298	9,024
PPP	1,032	17,490	18,522
カーネル	3,108	29,806	32,914
メモリブロック	22,300	1,418	23,718
通信機能合計	394	21,038	21,432
合　計	27,560	78,050	105,610

ては 33.6%のメモリ必要量となった．

(5) ITRON TCP/IP API は，メモリ必要量の観点からソケットインタフェースに対して優位にあるが，インターネットで使用されている応用プログラムの多くはソケットインタフェースを用いており，ITRON TCP/IP API は移植性の面で不利である．ただし，ITRON TCP/IP API は，ソケットインタフェースとの互換性が考慮されている．また，インターネットで使用されている応用プログラムは，UNIX 等のプロセスモデルや豊富なメモリ量を前提にしており，本研究で対象とした規模の組込みシステムに，そのまま移植することが困難である．したがって，プログラム全体の見直しと同時に ITRON TCP/IP API への変更も考慮すべきと考えている．

7.5 WWW サーバの実装

厳しいリソース制約のため，本研究が対象とする規模の組込みシステムに，ソケットインタフェースを用いて WWW サーバの実現は不可能であった．それに対して，本研究による ITRON TCP/IP API を用いたプロトコルスタックでは，WWW サーバを実現することが可能であり，その実現性を確認することができた．物理ネットワーク層は，シリアルインタフェースとモデムを用いた PPP である．転送可能なページは 2 ページで，トップページの容量が約 1,250 オクテット，動的に生成されるネットワーク統計情報のページが約 3,000 オクテットである．

(1) メモリ必要量を RAM と ROM に分けて，**表 8** に示す．物理ネットワーク層が異なっているため，表 7 とは，カーネル，メモリブロックおよび通信機能合計のメモリ必要量が異なっている．この結果から，対象とした組込みシステムの規模に収めることができた．WWW サーバ自体のメモリ必要量は，RAM が 726 バイト，ROM が 8,298 バイトで，ページ数や機能の追加による必要なメモリの増加は，この部分に限定できる．

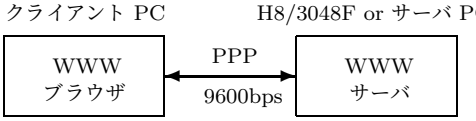

図 5 実行性能試験の構成
Fig. 5 Test configuration for performance test.

表 9 実行性能試験の結果
Table 9 Result of performance test.

項目	ソケット [ms]	ITRON [ms]
コネクション開設	94	81
データ転送	3,773	3,592
コネクション開放	62	102
全　体	3,929	3,775

(2) 本研究では，メモリ必要量を中心に評価したが，参考として，**図 5** に示す構成で，実装した WWW サーバの実行性能の試験も行った．WWW サーバとクライアント PC（IBM PC/AT 互換機，CPU: Pentium4 1.6 [GHz]，OS: Windows 2000，VMware 2.0 上の FreeBSD）は，シリアルインタフェース（9,600 bps）と PPP で直接接続した．比較するソケットインタフェースを用いた WWW サーバに関しては，WWW サーバとして Apache 1.3 を使用し，サーバ PC（IBM ThinkPad X24，CPU: Pentium III 1.13 [GHz]，OS: Windows XP，VMware 2.0 上の FreeBSD）上に構成した．クライアント PC の WWW ブラウザからのアクセスに対する応答時間を**表 9** に示す．全体的には，本研究のプロトコルスタックの性能が良い．コネクション開設の時間が短いのは，7.1 節のコネクション制御構造体の簡素化による効果と考えられる．また，データ転送の差は，ITRON TCP/IP API の平均セグメント数が 13.7 に対して，ソケットインタフェースが 14.0 であり，平均セグメント数の差によると考えられる．コネクション開放では，ソケットインタフェースの方が良い結果が得られたが，ITRON TCP/IP API では，データ転送の一部がコネクション開放時に行われたことによる差と考えられる．

8. 関連研究

本研究でのプロトコルスタックと類似している研究との比較について述べる．

(1) CMU Mach[19]
Machは，OSの機能を仮想記憶システムを中心にしたマイクロカーネルに凝縮し，モノリシック OSではカーネル内にあるネットワーク機能を，マイクロカーネル外のサーバに実装している．本研究のプロトコルスタックも，RTOSと応用プログラムの中間のミドルウェアとして実装しており，この点で Machと同様のアプローチとなっている．

(2) MIT Exokernel[20]
Exokernelは，一般的にはOSで提供されているネットワーク機能を，応用ライブラリ（libOS）として構成し，ハードウェアアーキテクチャに最適化して，ネットワーク性能の向上を目的としている．これに対して，本研究では，汎用的な階層を取り除くことにより，プロトコルスタックに関するメモリ必要量の最適化を行っており，Exokernelと同様のアプローチとなっている．

(3) Cornfed Systems, Inc. Roadrunner/pk[21]
Roadrunner/pkは，ネットワーク性能を含むI/O性能の向上を目的としており，特にデータストリーミングの zero-copyを特徴としている．zero-copyに関しては，ITRON TCP/IP APIにも省コピー APIがあり，本研究のプロトコルスタックにも実装され，応用プログラムとのデータのコピーを省くことが可能である．

(4) RedHat, Inc. eCos[22]
eCosには，μITRON 3.02 仕様[23]の機能と，ネットワーク APIとして，ソケットインタフェースの両方が組み込まれている．これに対して，本研究におけるプロトコルスタックは，μITRON 4.0を対象とし，APIは ITRON TCP/IP APIである点で eCosと異なっているが，機能的な面では同等と考えられる．

9. おわりに

本論文では，組込みシステムに求められる通信機能を示した．応用層プロトコルは，インターネットの標準的なデータ記述方式として注目されている XMLを転送可能な HTTPを採用した．組込みシステムの特性からネットワーク上での配置は，単一リンクの終端ノードで，単一の物理ネットワーク層とした．また，TCPと IPの機能を見直し，IPでの分割と再構成を削除した．

本提案の ITRON TCP/IP APIの実装は，比較評価のために実装したソケットインタフェースに対して，通信機能に関して 19.0%，システム全体に関して 45.3%のメモリ必要量となり，本研究で提案したメモリブロック割当てによる実装と，省コピー API，コネクション制御構造体と層間インタフェースの簡素化等，仕様自体が持っている厳しいリソース制約に対する有効性を確認した．

最後に，応用プログラムとして WWWサーバを実装し，実現性を確認した．また，ソケットインタフェースによる実装では，厳しいリソース制約のため，本研究が対象とする規模の組込みシステムには，この WWWサーバ等の応用プログラムの実装が不可能であることが判明した．

今後の研究課題としては，H8/3048F 以外のアーキテクチャやハードウェアへの移植とイーサネットや bluetooth 等の多様な物理ネットワーク層への応用研究である．本研究では，単一リンクの終端ノードで，単一の物理ネットワーク層に限定したが，複数リンクや複数の物理ネットワーク層への対応も可能と考えており，これらの応用範囲が，きわめて広い．また，今後のインターネットでは，情報家電等への適用のため IPv6への対応が不可欠であるが，メモリ必要量への影響が大きいことが予想される．IPv6への対応にも，本研究と同様の実装方法が適用可能である．

謝辞 ソケットインタフェースの実装に関する研究を支援していただいた財団法人道央産業技術振興機構と，携帯電話・モデム等の通信機器を提供していただいた（株）NTTドコモ北海道苫小牧支店の皆様に，つつしんで感謝の意を表する．

参考文献

1) 高田広章：組み込みソフトウェア開発の現状と課題，*bit*, Vol.32, pp.3–16 (2000).
2) 高田広章：組込みシステム開発技術の現状と展望，情報処理学会論文誌，Vol.42，pp.930–938 (2001).
3) 高田広章（編）：ITRON TCP/IP API 仕様 1.00.01，トロン協会 (1998).
4) 高田広章：TCP/IPプロトコルスタックの問題点と ITRON TCP/IP API仕様，*Interface*, Vol.24, pp.86–97 (1998).
5) 阿部 司：群知能機械組込み用基本通信ソフトウェア，2002 年度精密工学会春季大会論文集，p.233 (2002).
6) 坂村 健（監修），高田広章（編）：μITRON 4.0 仕様 4.01.00，トロン協会 (2001).
7) FreeBSD プロジェクトホームページ．http://www.freebsd.org/
8) 中山幹敏，奥井康弘：改訂版標準 XML 完全解説，上，下，技術評論社 (2001).

9) Berners-Lee, T., Fielding, R. and Frystyk, H.: Hypertext Transfer Protocol — HTTP/1.0, RFC 1945 (1996).
10) Braden, R.: Requirements for Internet Hosts — Communication Layers, RFC 1122 (1989).
11) Mockapetris, P.: Domain Names Implementation and Specification, RFC 1035 (1987).
12) Leffler, S.J., Mckusick, M.K., Karels, M.J. and Quarterman, J.S.: *The Design and Implementation of the 4.3BSD UNIX Operating System*, Addison-Wesley (1989).
13) Simpson, W.: The Point-to-Point Protocol (PPP), RFC 1661 (1994).
14) Itron プロジェクトホームページ. http://www.itron.gr.jp/
15) TOPPERS/JSP プロジェクトホームページ. http://www.ertl.ics.tut.ac.jp/TOPPERS/
16) (株) 日立製作所：H8/3048 シリーズ H8/3048F-ZTAT^{TM}ハード ウェア マニュアル，6 edition, (株) 超 L メディア技術ドキュメントグループ (2000).
17) Jacobson, V., Braden, R. and Borman, D.: TCP Extensions for High Performance, RFC 1323 (1992).
18) Braden, R.: T/TCP — TCP Extensions for Transactions Functional Specification, RFC 1644 (1994).
19) CMUMach プロジェクトホームページ. http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/mach/public/www/mach.html
20) MITExokernel ホームページ. http://www.pdos.lcs.mit.edu/exo.html
21) Roadrunner/pk ホームページ. http://www.cornfed.com/
22) eCos ホームページ. http://sources.redhat.com/ecos/
23) 坂村 健 (監修)：μITRON 3.0 標準ハンドブック改訂新版，パーソナルメディア (1993).

(平成 14 年 6 月 24 日受付)
(平成 15 年 4 月 3 日採録)

阿部 司（正会員）

1960 年生．1984 年豊橋技術科学大学大学院工学研究科情報工学専攻修士課程修了．同年，電電公社入社．1992 年より苫小牧工業高等専門学校情報工学科助教授．2002 年より室蘭工業大学大学院工学研究科博士後期課程在籍中．組込みシステム，情報通信，計算機工学等の教育研究に従事．精密工学会正会員．

吉村 斎（正会員）

1955 年生．1983 年北海道大学大学院工学研究科修士課程修了．同年，朋立技研 (株) 入社．1986 年朋立ロボティクス (株) 出向．1989 年 (学) 桑園学園入社．1995 年室蘭工業大学大学院工学研究科博士後期課程修了．同年，苫小牧工業高等専門学校情報工学科助手，1998 年助教授，1999 年教授．ロボット工学，組込みシステム等の教育研究に従事．博士 (工学)．精密工学会，計測自動制御学会，日本ロボット学会の各正会員．

久保 洋（正会員）

1943 年生．1969 年北海道大学大学院工学研究科修士課程修了．同年，北海道大学工学部助手，1976 年北海道大学大型計算機センター助教授，1984 年室蘭工業大学産業機械工学科教授，1990 年より室蘭工業大学情報工学科教授．インテリジェント CAD/CAM システム，感性のモデル化，Web コンピューティング等の教育研究に従事．工学博士．精密工学会，日本機械工学会，日本感性工学会の各正会員．