教育用簡易 UDP/IP スタック TinyIP の設計と実装

楯岡孝道, 阿部公輝
E-mail: {tate,abe}@cs.uec.ac.jp
電気通信大学 情報工学科

インターネットプロトコル技術は非常に重要な教育テーマであり、広くその動作原理の理解が求められている。このような教育には、実際に動作する実装を用いた手法が有効であるが、現在用いられているプロトコル実装は度重なる改良によって複雑になり、その理解が難しいという問題がある。そこで我々は、通信に必要な最低限の機能のみを持つ教育用簡易 UDP/IP スタック TinyIP を設計、実装した。最低限の機能のみを選択した結果、TinyIP はコメントを含めても 1,700 行以下で実装されており、学生が容易に理解し、改良することが可能となっている。本論文では、TinyIP の設計、実装、および TinyIP を用いた教育や研究成果について述べる。

Design and Implementation of TinyIP: A Limited UDP/IP Stack for Education

Takamichi Tateoka, Kôki Abe
E-mail: {tate,abe}@cs.uec.ac.jp
Dept. of Computer Science, The University of Electro-Communications

Nowadays the Internet protocol technology is one of the most important educational materials in computer science, the principles of which are demanded to be well understood by students. For teaching the materials, it is effective to use real working implementations of the protocols. However, due to repeated improvements on the implementations of the protocols, their current versions are too complex for students to understand. To overcome the problem, we designed and implemented a simple educational version of UDP/IP stack named TinyIP which was specified to have a minimum set of functions required for the communications. The code size of the TinyIP is less than 1,700 lines which is small enough for students easy to understand the principle of the layered communication protocols, to improve the implementation, and to extend the TinyIP itself. In this paper we present the design and implementation of the TinyIP and describe the courses using the TinyIP and the research works based on the TinyIP.

1 はじめに

インターネットの普及により、ネットワークプロトコル、特に実際にインターネットで用いられているインターネットプロトコル技術の教育は非常に重要になりつつある。このような教育には、実際に動作する実装を用いて動作を確認しながらおこなうような手法が有効である。しかしながら、現在ソースコードが入手可能なプロトコル実装は度重なる改良によって巨大で複雑になり、学生がそれを理解することが難しいという問題がある。これは、これらの実装が実用に供するために数多くの機能を含んでいることが原因である。

そこで我々は、通信に必要な最低限の機能のみを持つ教育用簡易 UDP/IP スタック TinyIP を設計、実装した。TinyIP はその実用性よりも、規模の小ささと理解のしやすさに主眼を置いて設計したため、容易に実装全体を読み通し、理解することができる。また、拡張も容易であるため、学生がこれを拡張することによって理解を深めることも可能である。

本論文では、第 2 章で他の IP 実装を教育に用いる際の問題点について述べ、第 3 章で TinyIP の設計と実装について述べる。第 4 章で TinyIP を用いた教育、研究成果について紹介し、第 5 章でまとめる。

2 他の IP 実装とその問題点

ソースコードが利用可能なインターネットプロトコル実装には、BSD UNIX の TCP/IP 実装、Linux の TCP/IP 実装、KA9Q[1]、そして x-kernel[2] がよく知られている。

BSD UNIX の TCP/IP 実装と Linux の TCP/IP 実装は、現実の製品で用いられている完全な実装であり、TCP/IP に必要な機能が全て含まれている。しかしながら、例えば NetBSD 1.5.3 の netinet 部分だけで 45,000 行、Linux kernel 2.4.2 の IPv4 部分だけで 34,000 行と、巨大であり、また効率を上げるための複雑なデータ構造や度重なる改良のため、可読性も低くなってしまっている。また、プロトコルスタック実装に変更を加えた場合、OS 全体の再起動が必要となり、実験が容易にできないという問題点もある。

KA9Q NOS は IBM-PC 互換機で動作する TCP/IP 実装であり、IP 処理主要部が 600 行強、UDP 処理主要部が 300 行弱とコンパクトにまとまっている。しかしながら、実装全体では 70,000 行を超えるうえ、動作させるには IBM-PC 互換機が必要となるなど、制限が多い。また、既にメンテナンスされていないという問題点もある。

x-kernel はオブジェクト指向の考え方を取り入れたネットワークプロトコル実装のためのフレームワークであり、UNIX 上のアプリケーションとしてネットワークプロトコルを動作させることも可能となっている。また、これを用いている教科書 [3] も書かれている。しかしながら、x-kernel ではプロトコル実装が抽象化されており、プロトコル実装全体がどのように動作するかを把握することが難しいという問題がある。また、階層間のインタフェイスが限定されており、インタフェイスを含めた実装の改良などに制限があるという問題もある。

3 TinyIP の設計と実装

3.1 設計

TinyIP の設計にあたり、次のような設計方針をたてた。

- 学生が全体を容易に理解できる大きさと複雑さであること
- 学生が達成感を持てる程度に実用的であること
- 学生が階層化プロトコルの特性や利点を理解できるように教育的であること
- 学生が不足部分や自らのアイデアを追加できるように各階層間のインタフェイスを含めて拡張可能であること
- 学生が様々な実行環境で実験できるように OS やハードウェアへの依存性を最低限にすること

TinyIP を容易で実用的なものとするため、TinyIP には、実際に他の IP 実装とアプリケーションによる相互通信実験を行うことのできる、最低限の機能を持たせることとする。理解の容易さを助け、拡張可能にするために、TinyIP は機能モジュール毎に分割した構造とし、各機能モジュールは定義された関数呼出しのみでデータのやりとりをおこなうことで、モジュール間の独立性を高める。各機能モジュールの概要を表 1 に、データの流れを図 1 に示す。教育的であるために、各モジュールの関係はプロトコルスタックの階層構造と

表 1: 各機能モジュールの概要

udpapi	アプリケーションとのインタフェイスをおこなう。
udp_input	UDP ヘッダの解釈をおこなう。
udp_output	UDP ヘッダの付加をおこなう。
ip_input	IP ヘッダの解釈をおこなう。
ip_output	IP ヘッダの付加をおこなう。
eth_input	Ethernet ヘッダの解釈をおこなう。
eth_output	Ethernet ヘッダの付加をおこなう。
fifo	キューイングに用いる汎用 FIFO 関数群を含む。
util	各ブロックから参照されるユーティリティ関数群を含む。
os	OS 依存部分を吸収する。
hardware	ハードウェアとのインタフェイスをおこなう。そのため、ハードウェアの操作方法毎に異なる。

1 対 1 に対応した構造とし、OS やハードウェアへの依存性を最低限にするために、OS やハードウェアに依存した機能は OS モジュールおよびハードウェアモジュールとしてそれぞれ一つにまとめる。

現実的な処理をおこなうために、受信処理は割込によって駆動するものとし、単純な FIFO バッファによって割込の処理部と通常の処理部のデータのやりとりをおこなう。この FIFO バッファを実現するモジュールは汎用のものを用意することで、他の処理を拡張する際にも利用できるようにする。

TinyIP を容易なものとするために、ネットワーク層としては基本的な IPv4 を、トランスポート層としては最も単純な UDP をそれぞれ採用し、ICMP や TCP は省略する。また、アプリケーション側で対処可能な、IP 層でのフラグメンテーションとリアセンブリも省略する。リンク層としてはリンク層アドレスを持ち非常に一般的なイーサネットを採用し、ARP 要求は通信相手の MAC 層アドレスを静的に設定することで省略し、ARP 応答に関しては代理 ARP もしくは通信相手ホストへの静的な ARP テーブルへの登録することにより省略する。IP フォワーディングに関しても、通信実験には必要ないため省略する。

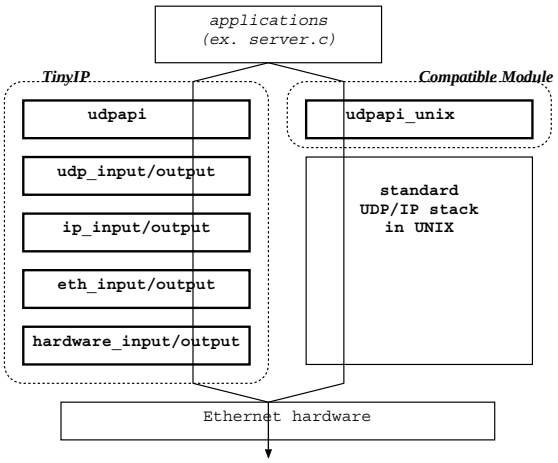

図 2: TinyIP と互換モジュールの関係

3.2 実装

実装は C 言語でおこない、1 つの機能モジュールが 1 つのソースファイルになるように分割した。対象ハードウェアとしては、Linux、BSD 系 UNIX および第 4.1 節で述べる CNP 結合実験における MinIPS[4] コンピュータシステム向けのものを実装した。ただし、MinIPS コンピュータシステム向けのものは Tiny C[5, 6] への移植をおこなった。

Linux 向けのハードウェア依存部は PF_PACKET ソケットを用いて実装し、Vine Linux 2.1.5 において動作を確認した。BSD 系 UNIX 向けのものは BPF[7] (Berkeley Packet Filter) を用いて実装し、NetBSD 1.5.2 および FreeBSD 2.2.8R で動作を確認した。実装はハードウェア依存部を除いて約 1,400 行であり、ハードウェア依存部は PF_PACKET 版が約 150 行、BPF 版が 250 行弱となった。また、MinIPS 版のみアセンブラで記述し 350 行となった。

また、TinyIP と、OS の IP スタックとの動作を比較できるように、BSD ソケットインタフェイスを用いて TinyIP と同じプログラミングインタフェイスを提供する、udpapi 互換モジュールを udpapi_unix モジュールとして実装した。TinyIP と互換モジュールの関係を図 2 に示す。

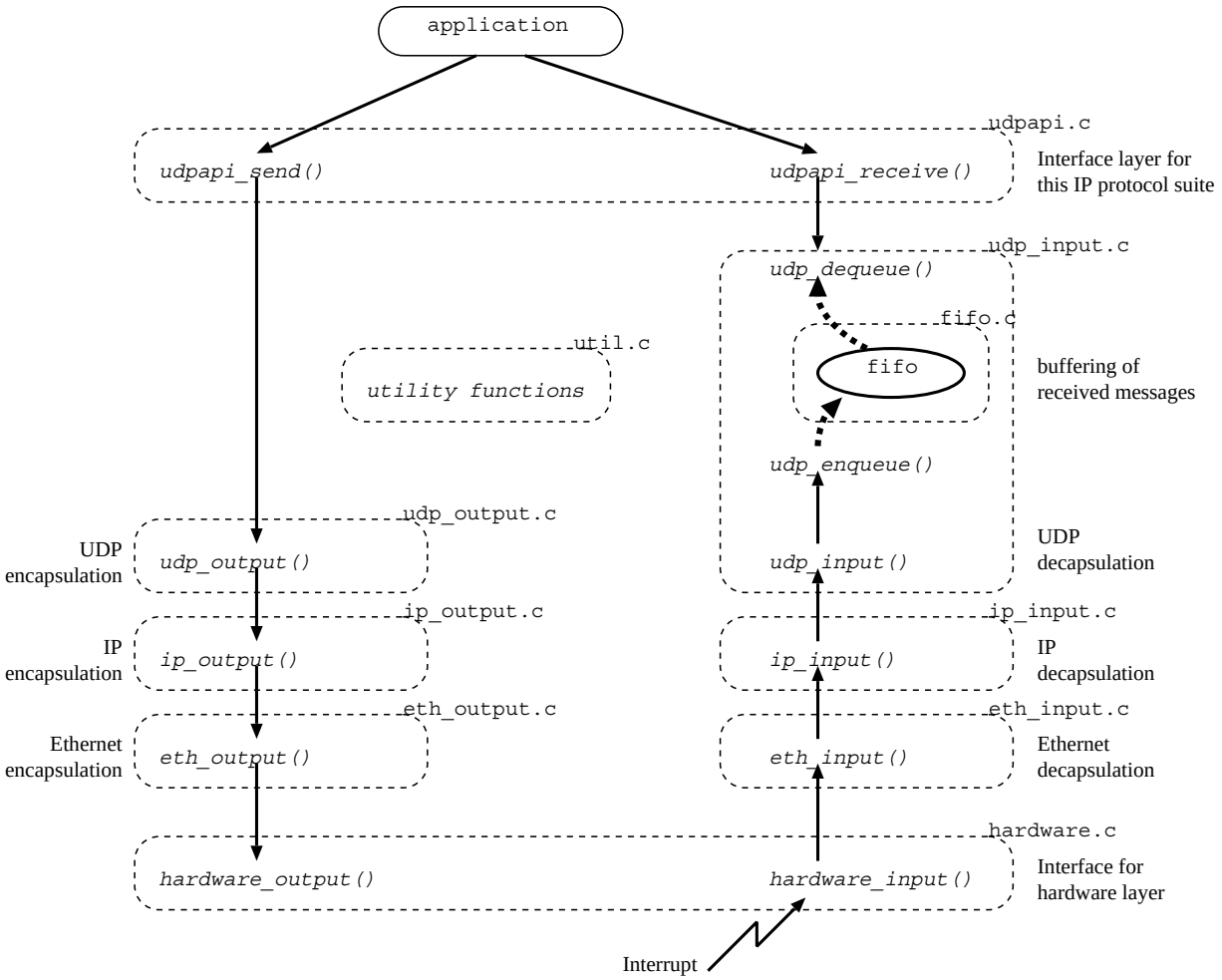

図 1: 機能モジュールとデータの流れ

4 TinyIP を用いた教育・研究成果

これまでに、実装した TinyIP を用いて 2 つの学生実験が開講され、3 つの研究がおこなわれている。本章では、それらの成果について紹介する。

4.1 CNP 統合実験

4.1.1 CNP 統合実験の概要

「CNP 結合実験」[8, 9] は、学生の実装したコンパイラ (C) で、学生の実装したネットワークプロトコルタック (N) をコンパイルし、学生の実装したプロセッサ (P) 上で動作させる学生実験である。本実験は、本学情報工学科昼間主コースの学部 3 年生後学期向けに開講している情報工学実験第二の中の 1 課題であり、3 時間の実習 12 回 (合計 36 時間) で実施している。本課題は選択課題であり、30 名のクラスを前半後半の 2 クラス実施する。クラスへの配属は学生自身の希望をもとに、他の課題とのバランスを考慮しておこなわれる。

本学生実験は教育的であるとともに学生にとって魅力的であるように設計した。すなわち、実験内で扱うのは実際に用いられている今日的テクノロジであり、課題の最終成果は学生が達成感を持てるように大きなものとした。課題は複数の副課題に分割され、複数の学生による共同作業によって完成させられる。この共同作業によって、システム全体を見渡せる広い視野と、サブシステム間のインタフェイスの摺り合わせ等の、共同作業に必要な能力を養う事を目的としている。

この目的のため、本学生実験では学生が UDP/IP スタック、C コンパイラ、32bit RISC プロセッサを副課

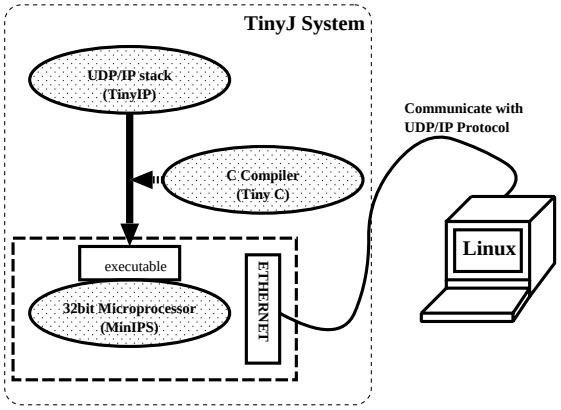

図 3: CNP 実験概要

題として実装し、最終成果としてこれらを組み合わせ、Linux ワークステーションと標準 UDP/IP を用いて通信できるコンピュータシステム (図 3) を構築する。以後、このコンピュータシステムを Tiny J システムと呼ぶ。Tiny J システムの構成要素は全て学生に対して公開し、興味を持った学生はシステムの回路図からアプリケーションソフトウェアのソースコードまでを参照することができるようにした。

4.1.2 CNP 統合実験の実際

クラスの学生は 6 名で構成される 5 つのチームに分割される。各チームのメンバは 2 名づつ副課題のプロセッサ (P)、コンパイラ (C)、ネットワーク (N) のそれぞれを担当し、最終的にチーム内で Tiny J システムを動作させる。以下、各副課題の概要を述べる。

プロセッサ副課題

プロセッサ副課題では、Altera 社 SOPC (System-On-Programmable-Chip) ボード [10] 上に搭載された FPGA 内に 32bit RISC プロセッサ MinIPS を Verilog 言語を用いて設計、実装し、イーサネットインタフェイスを持つワンボードコンピュータ (MinIPS コンピュータ) を構築する。学生はプロセッサ全体の設計はおこなわず、一部の記述が削除されたプロセッサ記述を受けとり、使用するデザインツールの使い方の修得から作業を始め、不足する部分を設計、実装して MinIPS コンピュータを構築する。

コンパイラ副課題

コンパイラ副課題では、プロセッサ副課題担当学生の実装する MinIPS コンピュータをターゲットとした、ネットワーク副課題担当学生の実装するプロトコルスタックをコンパイル可能な、C 言語サブセットのクロスコンパイラを実装する。学生は単純な言語仕様を持つ Tiny C コンパイラのソースコードを受けとり、ソースコードの解析から作業を始め、MinIPS プロセッサへの対応と、プロトコルスタックの実装のために不足しているオペレータとリテラルの追加をおこなう。

ネットワーク副課題

ネットワーク副課題では、MinIPS コンピュータをターゲットとした UDP/IP スタックとそのアプリケーションを Tiny C コンパイラを用いて設計、実装する。コンピュータネットワークについての講義は実験の翌学期に開講されるため、履修者はネットワークに関する専門知識を持たない状態で履修する。そのため、必要なプロトコル等の解説は実験中におこなう。

ネットワーク副課題は、大きく分けて 2 つの段階に別れている。前半段階では Linux をターゲットとした UDP/IP スタック (gcc 版 TinyIP) を実装、改良することで、ネットワークプロトコルスタックについて理解を深める。後半段階では、前半の経験を元に MinIPS をターゲットとしたスタック (Tiny C 版 TinyIP) を Tiny C コンパイラを用いて再度実装し、Tiny J システムを完成させる。学生は gcc 版 TinyIP および Tiny C 版 TinyIP 共にイーサネット、IP、UDP 各フレームの符号化、復号化部分が削除されたソースコードを受けとり、不足する部分を設計、実装して実際の通信実験までをおこなう。

副課題は以下のように進められる。

1. TinyIP のプログラミングインタフェイスを用いた単純なアプリケーションを作成する
2. 作成したアプリケーションを互換モジュール (udpapi_unix) を用いて動作させた際のイーサネットフレームを観測、解析する
3. gcc 版 TinyIP を実装し、Linux 上でアプリケーションを動作させる
4. gcc 版 TinyIP に機能追加や性能向上などの改良をおこなう

5. Tiny C 版 TinyIP を実装し、gcc を用いてコンパイルし、Linux 上で動作させる

6. Tiny C 版 TinyIP を実装し、Tiny C を用いてコンパイルし、MinIPS エミュレータ上で動作させる

7. Tiny C 版 TinyIP を実装し、Tiny C を用いてコンパイルし、MinIPS 実機上で動作させる

実験には学内ネットワークからは完全に独立した Linux ワークステーションを 2 台づつイーサネットクロスケーブルで接続して用いる。学生はワークステーションの管理者権限を持ち、`ifconfig` や `tcpdump` 等の管理者権限が必要なコマンドを用いて実験をおこなう。これらのワークステーションと OS 内 IP 実装は、実験後半段階では Tiny J システムの通信対象としても用いる。

4.1.3 CNP 統合実験の結果

本学生実験は 2001 年度より実施し、各副課題 20 名づつ、合計 60 名が履修した。前半クラスでは、各副課題担当学生間の連携不足によって、各副課題単位では動作しても、それらを Tiny J システムとして結合すると動作しない学生が散見された。しかし、後半クラスでは各副課題担当学生間で毎回ミーティングを行うなどの改善をおこなった結果、全チームが結合に成功した。自分たちの作成した Tiny J システムが、日常的に利用しているワークステーションと実際に通信をおこなう様子は学生に強い満足感を与えた。

ネットワーク副課題担当学生が gcc 版 TinyIP に対しておこなった改良内容と、改良を試行した人数および完成させた人数を、表 2 に示す。学生によっては複数の改良をおこなっており、与えられた実装の不完全さが学生の改良意欲を向上させたと考えられる。

実験終了後にネットワーク副課題担当学生に対して無記名でおこなったアンケート結果を図 4 に示す。この結果より、実験内容が十分に理解され、さほど困難ではないと評価されているにもかかわらず、学生は意欲的に長時間の自習をおこなっている事が読みとれる。

4.2 インターネットプロトコルスタックの実装実験

「インターネットプロトコルスタックの実装実験」は、TinyIP のプロトコル処理部を学生が実装し、さらにそれを改良する学生実験であり、前節で述べた CNP 統合実験のネットワーク副課題の前半の課題とほぼ同一内容である。本実験は、本学情報工学科夜間主コースの学部 3 年生後学期向けに開講している情報工学実験第二の中の 1 課題であり、3 時間の実習 8 回 (合計 24 時間) で実施している。本課題は必修課題であり、情報工学科夜間主コースの全 3 年生が受講する。2001 年度は 16 名が受講した。

学生が TinyIP に対しておこなった改良内容と、改良を試行した人数および完成させた人数を、表 3 に示す。この表より、CNP 統合実験における結果よりも実質的な実習時間は長いにもかかわらず、改良の数が少なく、より単純な改良に留まっていることが読みとれる。これは、学生の希望による選択ではなく全員の履修だったこと、夜間主コースであるために十分な自習時間が取れなかった事などが原因であると考えられる。

表 2: CNP 統合実験: 学生による改良内容と 20 名中の実施人数

内容	学生数 試行者 (完成者)
Optimizing memory usage	8(8)
IP fragment transmission	4(2)
IP fragment reception	3(2)
ARP request	7(6)
ARP reply	5(5)
ICMP echo reply	4(3)
ICMP port unreachable	2(2)

表 3: インターネットプロトコルスタックの実装実験: 学生による改良内容と 16 名中の実施人数

内容	学生数 試行者 (完成者)
Optimizing memory usage	2(2)
ARP request	10(7)
ARP reply	5(5)
ICMP echo reply	4(4)

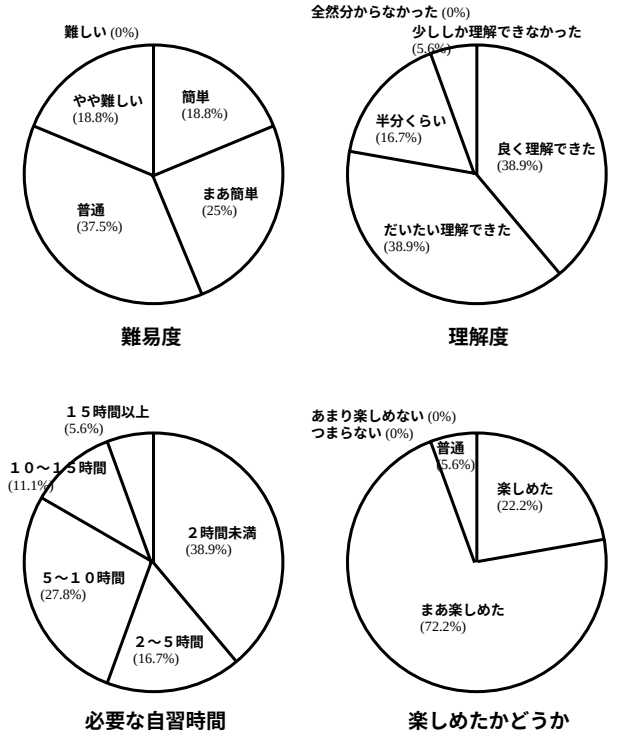

図 4: ネットワーク副課題担当学生へのアンケート結果

4.3 TCP 実装の試み

学部 4 年生の卒業研究として井田、Azwar らによって、TinyIP に TCP を追加する試みがおこなわれた。[11, 12]

この試みでは、TinyIP の IP 層の上に機能を限定した TCP 層を追加し、他の TCP 実装との通信実験に成功した。この実装では、ウィンドウ制御や輻輳制御等の TCP における重要な機能が省略されているが、TinyIP の提供する IP 層以下の機能モジュールや汎用 FIFO モジュールを用いることで、実際に他実装との相互運用が可能な実装が容易に実装できる事を示した。

4.4 UDP/IP スタックのハードウェア実装

森田らによる、UDP/IP スタックを FPGA 上に実装し、その性能評価をおこなう研究 [13] では、そのソフトウェア実装として TinyIP が用いられた。また、ハードウェア化に際しては TinyIP の各機能モジュールを次々とハードウェアに置きかえていく手法が取られた。

TinyIP の単純な構造がハードウェア化を容易にするとともに、機能モジュール単位でハードウェア化することによって、一部分をハードウェア化した段階で動作させることが可能となるなど、容易な開発が可能になった。

4.5 IPv6 スタックのハードウェア実装

出原らによる、IPv6 スタックのハードウェア実装 [14] においては、その実装時、まず TinyIP を IPv6 対応にし、それを元にハードウェア化するという手法が取られた。

これも TinyIP の単純な構造が、IPv6 対応やハードウェア化などの実現を容易にした例であるといえる。

5 おわりに

本論文では、教育用簡易 UDP/IP スタック TinyIP の設計、実装、およびそれを用いた教育や研究成果について述べた。TinyIP はその規模の小ささと単純な構造によって容易に理解可能となっている。ネットワークに関する専門知識のない学生でも僅かな時間でこれを理解し、改良することに成功している事がこれを示している。また、実験を開始できる最低限のフレームワークと実装を持ち、学生が追加した機能をすぐに組込み、他の実装との通信実験ができる環境を提供することで、学生の理解の促進と意欲の向上に役立っていると考えられる。これは、プロトコルの動作を理解するには、完全で複雑な実装を解析するよりも効果的であろう。

なお、TinyIP の実装は以下の URL より可能になる予定である。

`http://wwwhnl.cs.uec.ac.jp/tate/cnp/`

謝辞

TinyIP の仕様策定と活用にあたり、数多くの議論をしてくださった鈴木貢助手と河野健二助手に感謝いたします。

参考文献

[1] Phil Karn, “The KA9Q NOS TCP/IP Package,” `http://www.ka9q.net/code/ka9qnos/`

[2] N. C. Hutchinson and L. L. Peterson, “The x-Kernel: An architecture for implementing network protocols,” IEEE Transactions on Software Engineering, 17(1):64#76, January 1991.

[3] Bruce S. Davie, Larry L. Peterson, and David Clark, “Computer Networks: A Systems Approach,” Morgan Kaufmann Publishers, October 1999.

[4] 葛毅, 大菅大吉, 鶴田三敏, 阿部公輝, “32 ビット RISC プロセッサ MinIPS の設計と実装,” 電気通信大学紀要, vol.10, no.2, pp.71–78, 1997.

[5] 渡辺坦, “コンパイラの仕組み,” 朝倉書店, 1998.

[6] 鈴木貢, 河野健二, 楯岡孝道, 前田洋一, 阿部公輝, 渡邊坦, “計算機システム統合実験,” 第 4 回 組込みシステム技術に関するサマーワークショップ (SWEST4) 予稿集, pp.121–128, July 2002.

[7] Steven McCanne and Van Jacobson, “The BSD Packet Filter: A New Architecture for User-levle Packet Capture,” the 1993 Winter USENIX Conference, January 1993.

[8] 前田洋一, 楯岡孝道, 鈴木貢, 阿部公輝, “MinIPS コンピュータシステムによるプロセッサ/コンパイラ/ネットワーク統合実験, ” 電子情報通信学会論文誌 D-I, vol.J85-D-1, no.10, October 2002. (掲載予定)

[9] Takamichi Tateoka, Mitsugu Suzuki, Kenji Kono, Youichi Maeda, Koki Abe, “An integrated laboratory for computer architecture and networking,” In Proceedings of Workshop on Computer Architecture Education (WCAE2002), pp.110–117, May 2002.
http://www4.ncsu.edu/%7Eefg/wcae2002.html

[10] Altera Co., “System-on-a-Programmable-Chip Development Board User Guide,” September 2001.
http://www.altera.com/literature/ug/sopcug.pdf

[11] 井田久成, “簡易 TCP の設計と TinyIP 上への実装,” 卒業研究, 電気通信大学 情報工学科, 2002.

[12] MOHA AZWAR BIN HASSAN, “簡易 TCP の設計と TinyIP 上への実装,” 卒業研究, 電気通信大学 情報工学科, 2002.

[13] 森田和夫, “UDP/IP プロトコルの FPGA への実装と性能評価,” 情報処理学会全国大会, 8J-04, 2001.

[14] Y. Izuhara, K. Morita, T. Tateoka, and K. Abe, “Specification of TinyIPv6 Protocol Stack for Remote Control and Its Implementation on FPGA,” Trans. IPSJ, Vol.43, No.11, Nov. 2002. (掲載予定)