社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2003－ＱＡＩ－7 (13)
2003／5／21

ネットワークプロセッサを用いたMPLSラベルスイッチングルータの実装

渡辺 林音[†,††] 宇多 仁[†] 小柏 伸夫[†] 宇夫陽次朗[†††] 篠田 陽一[††††]

† 北陸先端科学技術大学院大学 情報科学研究科 〒923–1292 石川県能美郡辰口町旭台 1–1
†† 現: (株) 日立製作所 IP ネットワーク事業部 〒259–1392 神奈川県秦野市堀山下 1 番地
††† (株) インターネットイニシアティブ 技術研究所 〒101–0051 東京都千代田区神田神保町 1–105
†††† 北陸先端科学技術大学院大学 情報科学センター 〒923–1292 石川県能美郡辰口町旭台 1–1
E-mail: ††rinne.watanabe@itg.hitachi.co.jp, †,††††{zin,n-ogashi,shinoda}@jaist.ac.jp, †††yuo@iijlab.net

あらまし ソフトウェアルータは柔軟性が高く機能の拡張が容易である一方で，ルータとしての性能は専用ルータと比較して非常に低く，適用可能な範囲が限定される場合が多い．専用ルータとの性能格差を低減する技術の 1 つとしてネットワーク処理に特化したプロセッサ (NP) の開発および利用が注目されている．既存のネットワークスタックの一部機能を外部 NP で処理させることで，ソフトウェアルータの柔軟性を損なわずにシステム全体の高速化が可能である．本稿では，汎用 PC 上で動作するソフトウェア MPLS ルータとして設計実装された AYAME に対して，Intel 社のネットワークプロセッサ IXP1200 を外部に接続したハイブリッドシステムを用いた，ソフトウェアルータの高速化の設計と実装を述べる．

キーワード ネットワークプロセッサ，高速化，MPLS ルータ，ソフトウェアルータ

MPLS label switching router implementation enhanced with Network Processor

Rinne WATANABE[†,††], Satoshi UDA[†], Nobuo OGASHIWA[†], Yojiro UO[†††], and Yoichi SHINODA[††††]

† School of Information Science, Japan Advanced Institute of Science and Technology
1–1 Asahidai, Tatsunokuchi, Ishikawa 923–1292, Japan
†† Now: Global IP Network Systems Division, Hitachi, Ltd. 1 Horiyamashita, Hatano, Kanagawa 259–1392, Japan
††† Research Laboratory, Internet Initiative Japan Inc. 1–105 Kanda Jinbo-cho, Chiyoda-ku, Tokyo 101–0051, Japan
†††† Center for Information Science, Japan Advanced Institute of Science and Technology
1–1 Asahidai, Tatsunokuchi, Ishikawa 923–1292, Japan
E-mail: ††rinne.watanabe@itg.hitachi.co.jp, †,††††{zin,n-ogashi,shinoda}@jaist.ac.jp, †††yuo@iijlab.net

Abstract The type of software router has very high flexibility and functional extensibility, whereas it has much lower performance and less wide application than the exclusive hardware router in many case. Development and utilization of the processor optimized for the network processing (NP) has attracted much attention as one of techniques to decrease a performance gap between software and hardware routers. When outside NP module handle a part of existing network stack functions, whole system can be made faster without little damage to the flexibility of software implementations. This paper discusses about the design and implementation of the hybrid router with the IXP1200 intel network processor on generic PC to improve the forwarding performance of the AYAME software MPLS router.

Key words Network Processor, Performance Tuneing, MPLS Router, Software Router

1. はじめに

インターネット上のトラフィックは，利用者数の増加とアプリケーションの要求帯域量の増加に比例して広帯域化している．この要求に対処するために最近のルータの多くは専用 ASIC (Application Specific Integrated Circuit) を用いたハードウェアパケットスイッチング機構による高速/広帯域パケット処理を実現している．一方で専用 ASIC の設計・実装にはソフトウェアによる実装と比較して開発および製造のコストが非常に大きいため，インターネットにおける新規技術標準化への対応が困難である．

このような背景から，パケットスイッチング等のネットワーク処理に最適化された特定用途向けプロセッサであるネットワークプロセッサ (NP) が注目されている．NP を用いることで，高速パケット転送を提供しつつ，新機能への対応に関してもソフトウェア実装並みの柔軟性が可能となる．

著者らは，次世代インターネット技術として注目されている MPLS の研究開発プラットフォームである AYAME [1] の開発を進めている．AYAME は NetBSD オペレーティングシステム上で動作するソフトウェア MPLS (Multiprotocol Label Switching) [2] 研究開発プラットフォームである．AYAME は基本的な MPLS LSR(Label Switching Router) の機能を実装しつつ，MPLS を用いた新規技術に関する研究に用いるための拡張性を重視しており，既存シグナリングプロトコルの実験的拡張などが常に進められている．本稿では，NP を用いた AYAME における高速パケットスイッチングの実現に関して報告する．

2. ネットワークプロセッサ

ネットワークプロセッサ (NP: Network Processor) は，パケットスイッチング等のネットワーク処理に最適化された特定用途向けプロセッサである．NP は様々なアーキテクチャが提案されているが，一般的にはネットワーク関連の処理を高速に行うためのハードウェアおよび命令セットで構成される．専用のハードウェアでしか実現できなかったネットワーク処理速度をプログラム可能なプロセッサで実現できるため高性能かつ拡張性が高い機器の開発が可能である．

本稿ではソフトウェアルータ実装を拡張するうえで Intel 製の IXP1200 プロセッサ (注1) を搭載した横河電機 (株) 製の “ネットワークプロセッサカード NAPPI1200 (注2)” を採用した．NAPPI は比較的安価かつ入手性に優れる NP 開発環境である．

2.1 IXP1200 アーキテクチャ概略

IXP1200 は，6 個のパケット処理特化プロセッサ MicroEngine と StrongARM アーキテクチャの汎用プロセッサで構成される．

MicroEngine: 高速なパケット処理に特化した命令セットをもつパケット処理専用プロセッサである．ハードウエアレベルでマルチスレッドをサポートするため低レイテンシでコンテキストスイッチが可能であり，専用のパケット書き換え機構などによる高い転送能力を発揮するが，汎用性は低く複雑な処理は実現できない．

StrongARM: 主に MicroEngine をサポートする目的で利用される汎用プロセッサ．一般には Linux や VxWorks といった OS が稼動しており，MicroEngine の動作を制御する補助システムを提供するために用いられる．また，MicroEngine で処理できなかった例外パケットの処理にも用いられる．

2.2 NAPPI1200 概略

ネットワークプロセッサカード NAPPI1200 は，IXP1200 と付随する周辺チップやネットワークインタフェイスおよびメモリ等が搭載された NP 評価用ボードである (図 1)．ボード上には，

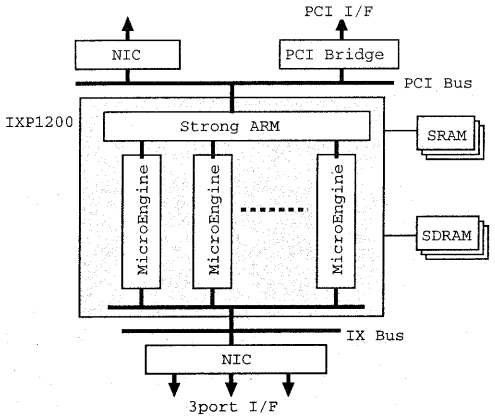


図 1 NAPPI1200 の構成

IXP1200 に接続された FastEthernet(FE) インタフェイスが 3 系統と補助用の FE が 1 系統実装されている．前者は，IXP1200 の MicroEngine と直結されており，本系統からの入出力は MicroEngine による高速処理が可能であるが，後者は StrongARM に接続されており，StrongARM 上の OS から操作される．以下，本稿では StorngARM に直接接続されているインタフェイスを，他のインタフェイスと区別し制御用 FE ポートと呼ぶ．また，NAPPI1200 上には 8Mbyte の SRAM と 256MByte の SDRAM が搭載されている．これらのメモリは，全ての MicroEngine および StrongARM から直接参照が可能であり，

- プログラムの格納
- パケット転送時のパケットバッファ
- パケット転送時に用いる FIB の格納

などに用いられる．実際に MicroEngine 上で動作するプログラムを記述する際は，動作速度・容量など特性の異なるこれらのメモリを，その用途に応じて使い分けることになる．

3. 設　　計

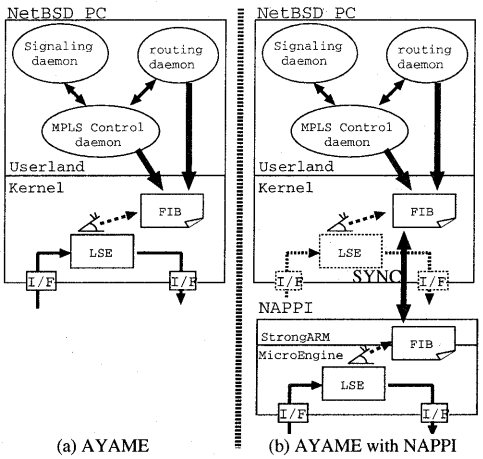
MPLS 実装 AYAME は，NetBSD のカーネルを拡張して実装されているラベル付きパケット転送機構 (LSE: Label Switching Engine) [3] と，ユーザ空間で動作するシグナリングデーモン等の制御機構で構成される．制御機構については，常に開発が進められている AYAME の新たな機能との整合性を維持するために特に手を加える事なく NetBSD 上で動作させながら，AYAME のパケット転送機構を NP を用いて高速化させる拡張を行った．既存 AYAME 実装と NP を用いた拡張の概略を図 2 に示す．

転送機構を NP 上で動作させることにより転送機構と制御機構が別のプロセッサ上で動作することとなる．このため，既存のソフトウェア実装においては，単一のテーブルとして保持していた FIB(Forwarding Information Base) 情報が，

- 制御機構の動作する PC 上の FIB
- 転送機構の動作する NP 上の FIB

と複数になり，その同期の手法等を考慮する必要が生じる．以下に，NP および PC における本実装の設計について述べる．

(注1)：http://www.intel.co.jp/design/network/products/npfamily/ixp1200.htm
(注2)：http://www.netstar.co.jp/products/NAPPI/



(a) AYAME　(b) AYAME with NAPPI

図2　既存 AYAME 実装と NP を用いた拡張

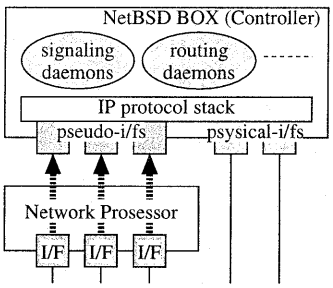


図3　実インタフェイスと NP 上のインタフェイスの同一視

入力ラベル (20bit)	始点 MAC (48bit)	終点 MAC (48bit)	出力ラベル (20bit)	処理 (2bit)	出力 I/F (4bit)
:	:	:	:	:	:
:	:	:	:	:	:

図4　NP 上の NHLFE テーブルの構成

3.1　NP 上のパケット転送機構

NP 上には AYAME LSE を実装した．MPLS LSR における LSE の処理は以下の手順で行われる．

（1）　ネットワークインタフェイスからパケットを受信し，
（2）　ヘッダ内のラベル値を取り出し，
（3）　NHLFE テーブルをを参照し，
（4）　PUSH/POP/SWAP など目的の処理をし，
（5）　適切な I/F から適切なノードに対して出力する．

今回の実装においては，LSR 上で最も一般的な動作であるラベルスワップの機能のみを NP 上のパケット転送機構で実現することとした．この場合，パケット転送には，以下のエントリが格納されたテーブル (FIB) が必要である．

- 入力パケットのラベル値 (検索鍵)
- 出力先インタフェイス
- 次ホップノードの MAC アドレス
- 出力パケットに付けるべきラベル値

なお，今回対象としている NAPPI1200 ボードは，イーサネットインタフェイスのみを提供しているため，フレーム中のラベル格納手法として SHIM ヘッダを用いたもののみを扱うこととし，ラベル空間としては一般に多く用いられている Global Label Space のみをサポートすることとした．

3.2　PC 上の制御機構

先述の通り，制御機構は NetBSD が動作する PC 上で実現する．シグナリングや経路制御など，既存の AYAME 実装に対して手を加えることなく NP 上のインタフェイスを取り扱える設計とすることが重要である．

これは，NetBSD 上で動作する各制御システムに対して，NP 上のインタフェイスをあたかも “PC に直接搭載されているインタフェイス” のように見せることで実現できる (図 3)．このような構成とすることで，各制御システムは NP 上のインタフェイスを一般のインタフェイスと同等に制御できることとなり，それぞれの制御システムに対して全く拡張を施すことなく NP 上のインタフェイスを扱うことができる．

3.3　PC – NP 間における FIB の同期

NP 上でパケット転送を実現する際に必要となる FIB 情報は，3.1 節で述べた通りである．PC 上では，3.2 節で述べた通り，経路制御・シグナリングなどの制御機構が NP の存在にかかわらず動作しており，NetBSD のカーネル空間内に FIB 情報として格納されている．この PC 上の FIB 情報のうち，NP 上で必要とされるものを NP との間で同期する必要がある．

制御機構の置かれた NetBSD PC 上で保持している FIB 情報は，先に挙げた NP における FIB 情報とほぼ同等なものである．ただし，PC 側では，MPLS ラベル付きパケットに対して SWAP のみならず，PUSH/POP などの処理も行うため，若干情報量が多い．また，PC 側では，次ホップの情報は IP アドレスとして保持しており，インタフェイスからのパケット送出時に ARP テーブルを参照することで MAC アドレスを求めている．今回の実装においては，転送処理に最低限必要な情報のみを NP 上に置くこととしたため，ARP 解決後の MAC アドレスを次ホップノードの情報として NP に伝達することとした．

4.　実　　装

前節で述べた設計に基づき，実際に実装を行った．

4.1　NP 上での LSE の実装

LSE は，ネットワークインタフェイスから入力されたラベル付きのパケットに対し，NHLFE (Next Hop Label Forwarding Entry) の情報にもとづくラベルの SWAP,PUSH,POP の操作を行い次ホップノードに対しパケットを送出する機構である．

まず，NP 上での NHLFE テーブルの構成に関して述べる．3.1 でも述べた通り，転送処理に必要となる情報は，入力パケットのラベル値を検索鍵とし，ラベルに対する処理，出力ラベル値，出力インタフェイスと次ホップノードの MAC アドレスである．さらに，出力時には各出力インタフェイスに対応する MAC アドレスを出力イーサフレームの始点アドレスとして格納する必要がある．この情報は，出力インタフェイス番号を鍵に別テーブルから検索し取得することも可能であるが，出力処

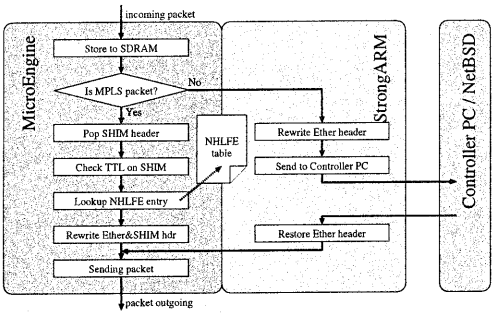


図5　NP 上の LSE におけるパケット転送動作

理の簡素化・高速化のため，今回は NHLFE テーブルにも出力時の始点 MAC アドレスとして格納することにした．今回用いた NHLFE テーブルの構造を図 4 に示す．なお，運用形態によっては NHLFE エントリ数は比較的大きな数となる場合も考えられるため，今回の実装では，この NHLFE テーブルは SDRAM 上に格納することとした．

次に実際の入力パケットに対する MicroEngine の処理について述べる．その概要を図 5 に，詳細を以下に示す．

まず，MicroEngine は，ネットワークからのパケット到着を確認すると，パケットを SDRAM に格納する．そして，転送処理において参照・書き換えが行われるイーサネットヘッダと SHIM ヘッダ部分 (計 18byte) を，SDRAM 転送レジスタ (32bit) に格納する．ここで，IXP1200 の MicroEngine における SDRAM の転送単位は 8byte なので，連続した 6 つの SDRAM 転送レジスタを用いて計 24byte を読み込む．

次に，該当パケットに対応する NHLFE を検索するために，SDRAM 転送レジスタに格納されたパケットデータから SHIM ヘッダ部分を汎用レジスタに取得し，その汎用レジスタ中のラベル値を検索鍵とし前述した NHLFE テーブルを検索する．ここで獲得した NHLFE に応じて，パケットに対する処理を行うわけだが，今回はラベルスワップのみを想定しているので，NHLFE の情報にもとづいて，

- SHIM ヘッダ内の TTL 値の減算
- SHIM ヘッダ内のラベル値の書き換え
- イーサネットヘッダの始点・終点アドレスの書き換え

を行う．なお，この書き換え処理は，先ほど SDRAM 転送レジスタ上に格納したフレームヘッダ部分に対して行う．

この書き換え処理の後，先に NHLFE で指定された出力インタフェイス番号に従い，パケットを送出する．なお，今回の実装にあたり，パケット送出部分を始めインタフェイスの直接操作に関わる部分に関しては，IXP1200 開発環境とともに配布されているサンプルコードを用いた．

4.2 NP 上のインタフェイスの PC 側での扱い

3.2 節で述べたように，NP によるパケット転送機構を用いた環境で AYAME のシグナリング機構や経路制御機構を手を加えることなく利用するためには，NP 上のネットワークインタフェイスを制御 PC 上に直接搭載されたものと同等に見せる必

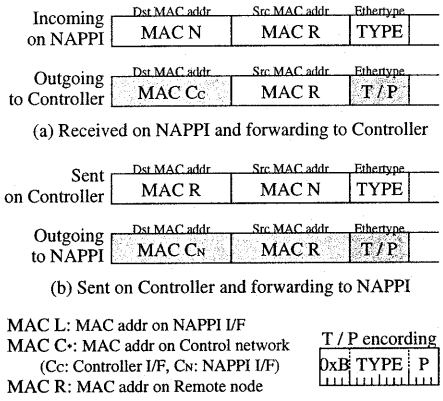


図6　フレーム回送時のイーサネットヘッダ書き換え

要がある．そこで，制御 PC 上の NetBSD において，NP 上のインタフェイスを扱うための仮想インタフェイスを実装した．NP 側では，先に述べたようにラベルスワップを行うパケット転送のみを扱っており，自ノードを終点とするパケットは制御 PC 側に回送する．制御 PC 側では，この NP から回送されて来たパケットを仮想インタフェイスから入力されたパケットと扱い，また，仮想インタフェイスから出力したパケットを NP 側へ回送し NP のインタフェイスから出力する．

ここで，NP と制御 PC 間でのパケットの回送手法について述べる．NAPPI1200 ボードは制御 PC の PCI バスに挿入されており 2. 節で示した PCI ブリッジを介して接続されている．しかし，現時点でこの PCI ブリッジを制御 PC 側で扱うためのドライバ等が用意されていないため，今回は NP と制御 PC の間の通信に PCI バスを用いることを断念した．それに代わり，NAPPI 上の制御用 FE と制御 PC 上の FE を接続し，このネットワークを介して NP と制御 PC 間の通信を行うこととした．

4.2.1 制御 PC – NP 間フレーム回送

まず，NP で受信した自ノード宛のパケットを制御 PC 側に回送する手法について述べる (図 6(a))．この回送には，NP と制御 PC 間のネットワークを用いるわけだが，制御 PC に対し受信パケットを正しく届けるためには，NP において受信したイーサネットフレームをそのまま PC 側に届ければよいわけではなく，宛先 MAC アドレスを制御 PC の MAC アドレスに書き換える必要がある．さらに，NP 上には転送に用いる 3 つのポートがあり，回送したパケットがどのポートから入力されたパケットなのか制御 PC 側で認識する必要がある．また，制御 PC 側では，受信したイーサフレームが実際には NP 上のポートから入力されたフレームであることを認識し，仮想インタフェイスからの入力と扱わなければならない．そこで，NP において受信した自ノード宛パケットは，その *ethertype* フィールドを書き換えて NP 上のインタフェイス番号を識別できる情報を埋め込んだ上で，制御 PC に送出することとした．ここで送出するフレームの *ethertype* は以下とした．

- *15–12bit*: NP で受信したフレームを指す (0xB)

- *11–4bit*: payload の種類を示す
- *3–0bit*: NP 上のインタフェイス番号を示す

また，*11–4bit* のペイロードタイプは，我々が扱うもののみを IP フレームを 0x01，ARP フレームを 0x02 のように独自に規定した．該当フレームを受信した制御 PC 側では，*ethertype* の *15–12bit* が 0xB であることで NP から回送されたフレームであることを検出し，上述の逆変換によりフレームを復元した後，仮想インタフェイスからの入力として扱う．

次に，自ノード発のパケットを NP 上のインタフェイスからの送出とするためのパケット回送手法について述べる (図 6(b))．制御 PC から送出されるパケットは，仮想インタフェイスを介して送出され，制御 PC – NP 間のネットワークを介して回送され，NP 上のインタフェイスから送出される．このようなパケットも，制御 PC – NP 間のネットワークで正しく NP に届け，また，NP が複数のポートのいずれから送出すべきかを判別できるようにするため，フレームの操作を行う必要がある．まず，該当フレームを正しく NP に届けるために，宛先 MAC アドレスを NP 上の制御 FE ポートの MAC アドレスに書き換える．元々の宛先 MAC アドレスは始点 MAC アドレスとして格納する．さらに，先に述べた NP から制御 PC へパケットを回送する場合と同一の手法で *ethertype* の書き換えを行い，フレームを NP へ回送する．このようなフレームを制御 FE ポートから受信した NP は，逆変換を行い元フレームを復元したのち，*ethertype* に指定された送出インタフェイスからフレームを送出する．一見，この手法では元フレームの始点 MAC アドレスの情報が失われているようにも見えるが，NP 側で，送出ポートの情報から始点 MAC アドレス情報が復元できる．

ここで挙げた手法とは別の手法として，NP – 制御 PC 間のフレーム回送の際には Ethernet over Ethernet のトンネリング手法を用いるという手段もある．この場合，構造が容易かつ回送できる *ethertype* に制限が加わらないという利点もあるが，回送できるフレーム長に対する制限やパケットサイズ増大によるオーバヘッドの影響を考慮し，今回は上述した手法を採用した．特に，現状のインターネットでの運用においては，MTU 値が 1500 未満となることは致命的である．

4.2.2 制御 PC 上での仮想インタフェイスの実装

制御 PC 上で，NAPPI1200 上のインタフェイスを直接接続インタフェイスと同等にユーザ空間プログラムに見せるための仮想インタフェイス *veth* を実装した．この実装は，制御 PC 上で動作する NetBSD カーネルに対する拡張である．

仮想インタフェイス *veth* ドライバは，インタフェイスタイプなどは一般の直接接続されているイーサネットインタフェイスと同一のものとした．通常のインタフェイスドライバでは，ハードウェアからの割り込み要求によりパケット入力処理が駆動され，また，パケット出力ではハードウェアに対してパケット送出処理を行う．これに対し，我々の実装した仮想インタフェイス *veth* は，特定の *ethertype* を持つパケットが NP と接続している実インタフェイスに入力されると *veth* の入力関数に渡され *veth* インタフェイスからの入力として扱い，また，*veth* の出力関数は出力フレームの書き換え処理を行った上で NP と接続している実インタフェイスから出力する．以下にその実装の詳細を述べる．

まず，NP から回送されてきたパケットの処理を行う部分について述べる．一般に，NetBSD においては，イーサネットインタフェイスからの入力フレームはイーサネット共通入力処理関数である *ether_input* 関数によって，その *ethertype* からネットワーク層プロトコルを決定しネットワーク層入力関数に渡される．我々は，この *ether_input* 関数を拡張し，先に述べた，*ethertype* の *15-12bit* が 0xB であるフレームを検出し，このフレームを *veth* ドライバの入力関数 *veth_input* に渡すこととした．*veth_input* では，前節で述べた手法により NP で受信した状態のフレームに復元した上で，入力インタフェイス情報を *vethN* (*N* は NP 上のインタフェイス番号) に書き換えた上で，一般のイーサネットインタフェイスの処理と同様にこのフレームを *ether_input* 関数に渡す．これにより，NP で受信したフレームはあたかも *vethN* インタフェイスで受信したように見える．

次に，制御 PC 側から送出する際の処理について述べる．制御 PC から *veth* ドライバを介した送出パケットは，一般のイーサネットインタフェイスにおける処理と同様に *ether_output* 関数によりイーサネットフレームが作られた上で，*veth_output* 関数へと渡される．*veth_output* 関数では，前節で述べた手法によりイーサネットヘッダの書き換えを行う．その上で，NP が接続されている実インタフェイスの出力関数 *xxx_output* を，*ether_output* 関数からの呼び出しと同等の方法で呼び出し，実インタフェイスのパケット出力処理を駆動する．これにより，*veth* インタフェイスから出力されたパケットは実インタフェイスを介し NP へと回送されることとなる．

4.3 FIB 情報の同期機構

AYAME ではラベルスイッチング処理内容を示す FIB 情報は，制御機構によって管理されており NetBSD カーネル中で保持されている．カーネル内の LSE では FIB 情報を直接参照できるが，NP を用いて外部 LSE を動作させる場合には NP 側で参照可能でなければならない．本実装では 4.1 節でも述べたように，NP 上の LSE はパケット転送処理の度に制御 PC 側の FIB 情報を参照せず，必要な最低限の FIB 情報を NP 側で保持する．このような，制御 PC 側の FIB 情報をマスタ，NP 側の FIB 情報をキャッシュとした動作を実現するためには制御 PC 上の FIB 情報を NP 上の FIB 情報へ反映して同期を取る必要がある．

NP 上に配置する FIB 情報は，4.1 節で述べた通り図 4 に示すものである．これらの情報のマスタは，制御 PC 上の NHLFE テーブルおよび ARP テーブルである．今回の実装では NetBSD のユーザ空間に kvm(kernel virtual memory interface) を用いた FIB 同期のデーモンを作成し，カーネル内の NHLFE テーブルおよび ARP テーブルを定期的に読み出し，NP における格納形態に加工した上で NP に対して通知することとした．NP 側では，制御 PC からの FIB 同期情報を StrongARM プロセッサで動作する FIB 同期のデーモンで受信し，MicroEngine が参照可能な SDRAM に格納する．このような仕組みで，制御 PC 上の FIB 情報がパケット転送処理に実際に用いられる NP 上の FIB 情報テーブルに反映される．

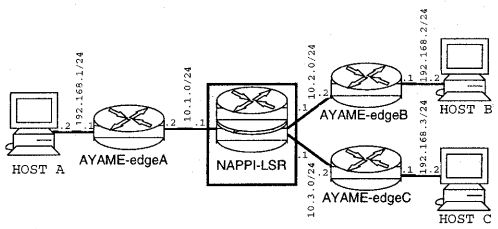


図7　実験ネットワーク

5. 動作実験と課題

実装の動作を検証するため，図7に示す実験ネットワークを構築した．実験により得られた知見と本実装の問題点およびその解決策について述べる．

5.1 動作検証

本実装ではMPLSのコアルータ機能のみを提供しているため，NP搭載AYAME LSRをコアルータとし，エッジルータにソフトウェア実装のAYAME LSRを配置した(図7)．各LSRではシグナリングプロトコルLDP[4]によるラベルシグナリングを行う*ldpd*を動作させた．

今回実装した仮想インタフェイス機能の検証としてLDPピアリングの正常動作の確認を行った．またtelnetやftp等のアプリケーションを用いてNP搭載AYAME LSRの制御PCと隣接ノード間で正常にデータ通信が行えていること確認した．さらにラベルスワッピング機能の動作を検証するためにHost A, Host B, Host Cの間でTTL値を調節したpingパケットを送受信した各LSR間のセグメントではパケットモニタを接続してLSPを介してpingパケットが転送されていることを確認した．また，telnetおよびftpなどのTCPアプリケーションにおいても同様にHost A, Host B, HostC間の通信がLSPを介した転送されることを確認した．

5.2 実装における課題

一部リンクを切断した際の挙動や*ldpd*によらず手動でラベルスイッチング規則を設定を行った際の挙動調査などから，いくつかの課題が得られた．以下にそれらの課題を挙げる．

a) FIB情報の同期

実験では，制御PC側のFIB情報が更新された場合，NP上のFIB情報がその更新に追従するまでに若干の時間を要した．原因は制御PC側のFIB情報を非同期ポーリングで取得する設計となっている点である．FIB情報の更新の遅れは，障害時等のパケット到達性回復までの時間に影響を与えるため，FIB情報更新をトリガとして同期更新する方式に変更する必要がある．

b) ARPエントリ期限

上述の実験ネットワークにおいて，*ldpd*を停止し手動でラベルスイッチング規則を設定した場合に，隣接ノードのARPエントリが期限切れとなり転送処理が停止する現象が確認された．*ldpd*動作中は制御PCから隣接ノードに対する定期的通信が行われているためそれらに対するARPエントリは維持される．しかしそのような通信が存在しない場合，NP上で転送処理が完結すると該当パケットの転送処理を行った事実が制御PCに伝わらず，制御PC上のARPテーブルが更新されない．

c) その他実装上の課題

今回の実装では実現を見送ったものとして以下が挙げられる．

- NPでのSWAP以外のラベル処理
- NP上のFIB情報の最適な配置の検討
- NP－制御PC間のPCIを用いた通信

まず，NP上でのSWAP以外のラベル処理が実装されていないため本実装はコアルータ以外の動作はできない．エッジルータ分野におけるNP搭載AYAME LSRへの要望も大きく，SWAP以外のラベル処理の実現が強く求められている．またIPパケットの転送処理については制御PC側で行ったが，エッジルータとしての動作を考慮した場合，IP転送機能もNP上で動作させることが強く求められると考えられる．

次に，NP上でのFIB情報の配置については，今回の実装ではメモリ容量の問題からその全体をSDRAMに配置した．SDRAMは読み書きにSRAMに比べ2倍程度の時間を要することを考慮すると，テーブルサイズが比較的小さな場合の配置手法や，大きな場合でも頻繁に用いられるものをキャッシュとしてSRAMに持つなどのより最適な配置を検討する価値がある．

また，NP－制御PC間の通信方法について見ると，NAPPI1200ボードには非対称PCIブリッジが用意されていたが，ホストPC側のOSおよびNP側OSのサポートの双方が必要なため利用を断念した．これらを用いることで効果的な接続が可能であるため今後の対応が検討すべきである．

6. まとめ

今回の論じた実装はコンセプトを実証するためのプロトタイプであり，設計自体の評価を行うことはできないが，ソフトウェア実装で実現されたネットワークスタックのパケット処理部分を外部接続されたNPを用いて並列処理することで，ソフトウェア実装の利点を維持した高速ルータの実現が可能であることを示すことができた．

今後は，AYAMEにおけるパケット配送機構として利用可能なNP実装を実現し，AYAMEの適用範囲を拡大していく予定である．

謝辞

本研究は，北陸先端科学技術大学院大学と横河電機(株)との共同研究として進められた．関係諸氏に深謝する．

文献

[1] Y. Uo, S. Uda, N. Ogashiwa, S. Ohta, and Y. Shinoda, "AYAME: A design and implementation of the CoS capable MPLS layer for BSD network stack," Proceedings of INET2000, no.438, Yokohama, Japan, July 2000, Internet Society.

[2] E. Rosen, A. Viswanathan, and R. Callon, RFC 3031: Multiprotocol Label Switching Architecture, IETF, January 2001.

[3] 宇多仁，宇夫陽次朗，篠田陽一，"MPLS実装AYAMEにおけるパケット転送機構の設計および実装，" 情報処理学会シンポジウムシリーズ，vol.2000, no.15，長野県下伊那郡，12 2000，情報処理学会．

[4] L. Andersson, P. Doolan, N. Feldman, A. Fredette, and B. Thomas, RFC 3036: LDP Specification, IETF, January 2001.