ネットワークプロセッサ技術の研究開発動向

河合栄治[†,☆] 門林雄基[††] 山口英[††]

ネットワークの今後のさらなる超高速化を実現する技術の 1 つとして，ネットワークプロセッサ（NP）が注目を集めている．NP とは，プログラム可能なプロセッサを有するネットワーク処理専用のサブシステムであり，従来の ASIC を基礎としたアーキテクチャと比較して，高速化のためのハードウェア構成の変更や，より高度なネットワークサービスの導入に柔軟に素早く対応することができる．一方で，NP 技術はこれまで産業界が主導で開発を行ってきており，そのため非公開な情報も多く，また研究成果の学術的な体系化も不十分である．本稿では，この NP 技術の学術文献およびその研究動向に関してサーベイを行い，今後の研究に資することを目的とする．具体的には，アーキテクチャ，性能評価，プログラミング支援環境，アプリケーション技術の各分野について論じる．

A Survey on Network Processor Technologies and R&D Trends

Eiji Kawai,[†,☆] Youki Kadobayashi[††] and Suguru Yamaguchi[††]

Network processors (NPs) are a promising technology that will support tomorrow's ultra high speed networking. The NP is a sub-system dedicated to network processing with programmable processors, memory buffers, and network interfaces. With NP technologies, developers can provide high-speed networking devices and/or sophisticated network services timely to the market. However, there is little work on the systematization of the NP research achievements because the development of the NP technologies has been conducted mainly by the industry. In this paper, we survey the NP technologies and its research trends to promote research on NPs. The topics of this paper include NP architectures, performance evaluation schemes, programming environments, and its application.

1. はじめに

近年のネットワークプロセッサ技術の発展は目覚ましいものがある．ネットワークプロセッサ（以下 NP と略す）とは，ネットワークトラフィックを処理するための専用のプロセッサであり，ルータなどの中継ノードだけでなく，サーバなどのエンドノードでも利用が可能になってきている．ネットワークプロセッサは，ネットワークインタフェース，パケット処理のためのプロセッサ，チェックサムなどの特定の処理を高速に行うためのコプロセッサ，メモリバッファ，外部接続のためのインタフェース，各要素を制御するためのプロセッサなどから構成され，それらが相互に高速なチャネルで結合されている（**図 1**）．もともと，高レベルなネットワーク処理を外部のコプロセッサで処理するというアイデア自体は新しいものではない[1),2)]．現在，NP が注目されるようになった背景には，次のような点があげられる．

まず，ルータのパケット転送能力の向上や QoS（Quality of Service）機能などの新しい機能の搭載を短時間で実現するためである．近年の高速ルータの実装においては，高速化を実現するために ASIC（Application-Specific Integrated Circuites）を用いたパケット処理プロセッサをネットワークインタフェースの近傍に配置する構成をとることが多い[3)]．しかし，今後の超高速ネットワークをサポートするためには，高性能 ASIC チップの開発に莫大な費用と時間がかかる．そのため，ソフトウエアによるプログラムが可能で並列化による性能向上が容易な NP を用いることで，短期間で安価な製品開発が可能になると期待されている．

次にあげられるのが，アクティブネットワーク技術[4),5)] に代表されるように，より高度なネットワーク

† 科学技術振興事業団さきがけ研究 21
PRESTO, Japan Science and Technology Corporation
†† 奈良先端科学技術大学院大学情報科学研究科
Graduate School of Information Science, Nara Institute of Science and Technology
☆ 現在，奈良先端科学技術大学院大学附属図書館研究開発室
Presently with Digital Library Research Division, Nara Institute of Science and Technology

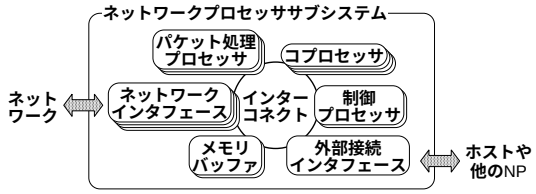

図 1 ネットワークプロセッサの構成要素
Fig. 1 The elements on a network processor sub-system.

サービスの構築が望まれるようになったことである．アクティブネットワーク技術は，クライアントやサーバなどのエンドノードだけでなく，ルータなどの中継ノードにおいても一定のサービス処理能力を持たせ，より動的なネットワークサービスを実現するものである．中継ノードにおける柔軟な処理能力を実現するために NP を応用することが期待されている．

最後にあげられるのが，ゲートウェイやサーバホストなどのエンドノードの性能向上が求められていることである．一般的に，プロセッサの処理能力は 18 カ月に 2 倍向上する（Moore の法則）ことが観測されている．一方で，近年のネットワーク帯域向上の速度は，この計算機の処理能力向上の速度よりもはるかに大きい[6]．そのため，エンドノードにおける TCP/IP 処理やアプリケーション層の処理を一部 NP に移すことで，処理能力の向上が期待できる．

このように大きな期待を集めている NP であるが，その技術研究開発はこれまで産業界が主導して行ってきている[7]．そのため非公開な情報も多く，また研究成果の学術的な体系化も不十分であるのが現状である．本稿では，NP に関する学術論文を中心にサーベイする．特に，現在の NP に関する最先端研究が，どのように相互に関連し，どこに向かおうとしているのかを明らかする．

以下本稿の構成を示す．まず，2 章で NP アーキテクチャに関する研究を概観する．ここでは，マイクロアーキテクチャとシステムアーキテクチャの 2 つの視点から分類する．次に 3 章では，NP における性能評価手法について述べる．ここでは，ベンチマークおよびシミュレーションに分類している．4 章では，NP におけるプログラミングに関する研究について，特に高レベルのプログラミング環境実現に向けた技術開発について述べる．そして 5 章では，現在主流となっている高速レイヤ 3（IP レイヤ）パケット転送以外の機能の NP における実現について研究成果を概観する．最後に 6 章で将来の NP 研究に関する展望をまとめる．

2. アーキテクチャ

NP 技術で最も重要なのがアーキテクチャであり，これまでに多くのベンダから様々なアーキテクチャを持つ NP がリリースされている．それらは，並列処理の実現方法，特定の目的のためのハードウェアの構成，メモリ構成，チップ上の通信機構，ネットワークインタフェースなどの周辺装置との接続方法，などの特徴によって分類することができる[8]．しかし，それらの要素は互いに密に関連するため，個々の議論は環境が変化すると適用できなくなる場合がある．本稿では NP 上に 1 つもしくは複数配置される処理要素（Processor Element，PE）のマイクロアーキテクチャと，NP における PE やメモリ，その他の周辺機器の配置および配線を含めたシステムアーキテクチャに分類し，特に各研究のアプローチの違いに焦点を当てて概観する．

2.1 PE マイクロアーキテクチャ

NP におけるパケット処理の中心的役割を果たすのが PE であり，そのマイクロアーキテクチャは NP のパケット処理能力の多くを決定する．ここでは，PE のマイクロアーキテクチャについて述べる．

2.1.1 PE コア

マイクロアーキテクチャを考える際に重要となるのは NP における処理内容であり，次にあげる様々な特徴を持っている[9]．

(1) データ転送の負荷が高い．
(2) 割込みが非常に高い頻度で発生する．
(3) 複雑な状態管理を必要とする処理が多い．
(4) IP ルーティングなどでは高速な表検索が重要．
(5) ワードやバイトの境界をまたぐ処理もある．

そのため，高速なメモリアクセス機構やマルチスレッドのサポート，パイプライン的な PE の構成，効率的な演算処理を実現するコアの命令セット，高速なイベント管理機構などが重要となる．特にマルチスレッドのサポートは，トラフィック処理の並列度の向上のためだけでなく，メモリアクセス遅延の隠蔽という観点からも必須の機能と考えられている．これは，マルチスレッド環境では，メモリアクセス遅延による実行の中断が発生しても他のスレッドを実行することができるからである．文献 10) では，マイクロアーキテクチャとして Super-scalar Processor（SS），Fine-grained Multithreaded Processor（FGMT），Single Chip Multiprocessor（CMT），Simultaneous Multithreaded Processor（SMT）の 4 種類について，典型的なワークロードに対して SMT が最も良い性能を発揮すると結論づけている．

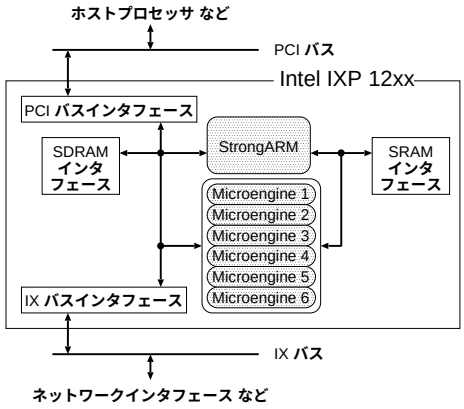

図 2 Intel IXP1200 の基本構成
Fig. 2 Intel IXP1200.

一方で，マイクロアーキテクチャにおけるもう 1 つの重要な点はコストである．NP では，処理の並列度を向上させるために複数の PE を搭載することが多い．そのため，各 PE はできるだけ単純な構造にしておくことが，チップのコストの低減だけでなく，ソフトウエア実装コストの低減にもつながる．たとえば，現在代表的な NP の 1 つとして広く利用が可能な Intel の IXP1200☆では，PE（Intel は microengine と呼んでいる）自体は命令セットの小さい非常に単純な RISC プロセッサを採用しており，メモリアクセス遅延を隠蔽するためにハードウェアによる non-preemptive なマルチスレッド環境をサポートしている．また，アクセス速度やサイズなど特性の異なるメモリを複数搭載し，それぞれの領域を明示的にアクセスする命令が用意されている．IXP1200 の基本構成を**図 2** に示す．

一般に，NP は TCP/IP プロトコルスタックにおけるレイヤ 3 以下のネットワーク処理をターゲットにしていることが多い．一方で，エンドノードにおける TCP off-loading のための NP アーキテクチャの研究もある．TCP では複数パケットにわたる状態情報を管理する必要があるが，こうした処理はレイヤ 3 における処理よりも複雑になる．そのため，レイヤ 3 までの処理を対象とした NP とは別の機構が必要となる．文献 11) では，従来の PE で処理可能なパケット内で完結するデータ処理と，それ以外のデータ処理（たとえば，状態情報の検索や PE の制御やホストプロセッサおよびメモリとの通信など）に機能分離し，後者を処理するプロセッサ（Micro Controller，μC）の要件を検討している．しかしながら，このようなアーキテクチャでは μC が性能のボトルネックになることが多く，ラインスピード（最小のパケットでネットワーク帯域幅を埋める処理速度）を実現することが困難となる．そのため，今後は TCP 処理の並列化の検討が求められる．

2.1.2 キャッシュ

現在の汎用プロセッサにおいては，キャッシュが性能に大きな影響を与える．一方で NP においては，命令キャッシュは有効だが，データキャッシュは有効でない場合が多い．NP で処理を行うデータの大半は，ネットワークインタフェースから入力されたパケットデータであり，パケットは各フィールドを処理した後は他のインタフェースか場合によってはホスト OS へ転送されてしまうからである．そのため，NP においてデータキャッシュが有効となるのは，パケットデータではなく NP が管理する状態情報においてとなるが，特にその中でもルーティングテーブルは時間的にも空間的にも参照の局所性が期待できる．文献 12), 13) では，レイヤ 3 転送におけるルーティングテーブルの高速検索手法に焦点を当て，キャッシュの側面から考察している．メモリキャッシュの構成においては，キャッシュのサイズ，ブロックサイズ，連想記憶の幅（associativity）が重要であり，高速な検索を実現するためにルーティングテーブルのエントリ数の効率的な圧縮技術を提案している．

2.2 NP システムアーキテクチャ

NP の基本的な目標の 1 つに並列処理がある．これには大きく分けて，単一パケット内の処理の並列化と，複数のパケットを並列処理する 2 つの方針がある．NP においては，並列処理における目標をどのように設定するかでシステムアーキテクチャが大きく異なってくる．ここでは，そうした NP システムアーキテクチャの中で特に先進的なものについて述べる．

PE の階層的な接続構造を提案しているのが文献 14) であり，柔軟な QoS を実現するための高度に並列化された NP アーキテクチャを提案している．提案しているアーキテクチャでは，2 つのコアと 1 つのレジスタファイルを有する PE の集合をグループ分割し，1 つの LSI 上に階層構造的に合計で 16 個配置している（**図 3**）．各グループ内は通信バスで直結し，同期動作させることで PE 間の高い相互通信機能を実現している．また，資源アクセス制御については，コンパイル時にあらかじめすべての PE のコードを参照しておくことで静的に解決し，プロセッサコアはレジスタファイル上でのみ演算処理を行い，全体で静的に同期した処理を実現している．

☆ http://www.intel.com/design/network/products/npfamily/ixp1200.htm

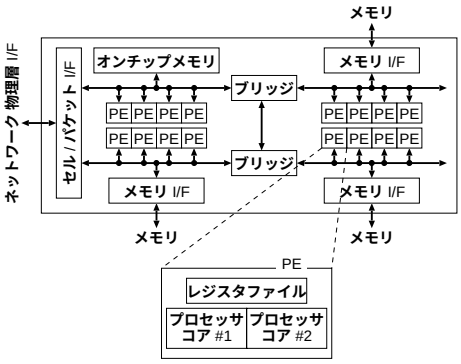

図 3 文献 14) で提案されている NP システムアーキテクチャ
Fig. 3 The NP system architecture proposed by Ref. 14).

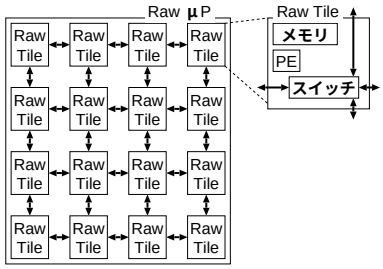

図 4 Raw プロセッサアーキテクチャ
Fig. 4 Raw processor architecture.

文献 15) では，IBM が提唱している SMT をベースにした超並列計算向け BlueGene/Cyclops cellular アーキテクチャ[16] を用いた NP について述べている．各 PE の命令キャッシュの有効性を向上させるために，パイプライン処理を採用し，文献 14) と同様プロトコル処理のプロセッサへの割当てやメモリ管理は静的に行っている．実際のアプリケーションとして Fibre Channel☆と Infiniband☆☆のプロトコル変換機能を実装し，シミュレーションによる評価を行っている．

また，超並列計算機のように非常に多数の RISC ベースのプロセッサをチップ上に配置し，プロセッサ間を縦横に接続した静的なチャネルと宛先を記述できる動的なチャネルによってプロセッサ間通信を実現するアーキテクチャ（Raw Processor[17] と呼ばれる）も提案されている[18]（**図 4**）．本方式は，静的な通信チャネルにおけるコンパイル時の最適化が可能であるという特徴に加えて，動的な通信チャネルを用いてプロセッサのスケジューリングをソフトウエアで制御することも可能であるという特徴を持つ．

☆ http://www.fibrechannel.org/
☆☆ http://www.infinibandta.org/

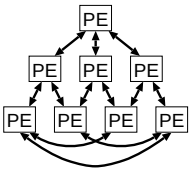

図 5 8 ノードの場合の Octagon 結合網
Fig. 5 Octagon configuration for 8 nodes.

一方で，PE 間の接続トポロジに特化した研究も存在する．文献 19) では，8 個のプロセッサおよびそれらを結ぶ 12 本のポイントツーポイント結合より構成される Octagon と呼ばれる PE 結合網を提案している（**図 5**）．Octagon は，ネットワーククロスバスイッチよりも配線が少なく，8 ノードの場合任意のノード間通信が最悪でも 2 ホップで可能であり，総合的なスループットが高いという特徴を持つ．

さらには，従来のプロセッサアーキテクチャではなく，並列計算機へのアプローチとしてデータ駆動の概念を用いた NP の研究もある[20]．ここでは，データ駆動型プロセッサを用いた終端ホスト向け TCP/IP off-loading エンジンを開発している．アプローチとしては，レイヤ 3 処理およびレイヤ 4 処理をパイプライン構成とし，各パイプラインステージにおける処理時間を考察し，ボトルネックとなる個所を 1 チップ集積したりすることでスループットを向上させる手法を紹介している．

文献 21) では，NP において非常に多数のスレッドをサポートするためのメモリアクセスにおける要件を考察している．スレッド間のメモリアクセスの競合はプログラミングが困難な領域であるため同期問題をハードウェアでサポートすること，それぞれのプロセッサにおけるローカルなメモリアクセス手法に加えてグローバルなメモリ空間も用意することでスレッドの PE 集合における配置問題をプログラミングから切り離すことができることなどが述べられている．

以上，並列計算機的アプローチをとっている NP アーキテクチャについて述べたが，一方で NP ボードのクラスタ化について検討を行っている研究もある．Comet[22] ではボードのクラスタ化によって高性能かつ柔軟なルータの構築を目指している（**図 6**）．Comet は，FPGA（Field Programmable Gate Array）によって実装されたチェックサムやテーブル検索などの機能単位と，それらを制御する汎用プロセッサで構成される簡素なアーキテクチャを採用しており，複数のボードを PCI バスやギガビットイーサネットスイッチなどを介してクラスタ構成にすることができる．ま

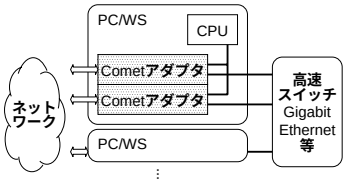

図 6 Comet のクラスタ構成
Fig. 6 Comet cluster architecture.

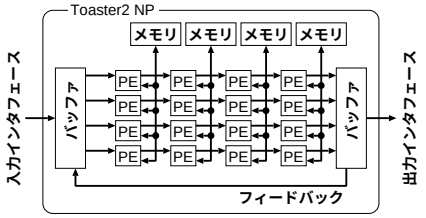

図 7 Cisco Toaster2 ネットワークプロセッサの構造
Fig. 7 Cisco Toaster2 NP.

た，Comet は NP の先駆的な研究としても注目に値するであろう．

3. 性能評価

システムの性能評価には 2 通りの目的がある．1 つはユーザが製品のおよその性能を知ることであり，ワークロードの現実性や他の製品との比較が重要となる．そのため，実際の製品を用いたベンチマークによる評価が一般的である．一方で，開発者がアーキテクチャを決定する際に様々なアーキテクチャについて評価を行いたい場合もある．その場合，各アーキテクチャを実装していては開発コストが高いため，シミュレーションによる評価が行われる．本章では，こうした性能評価手法について述べる．

3.1 ベンチマーク

ベンチマークを行うためには，まずは NP の評価に適したワークロードの選定が必要となる．文献 23) では，NP におけるワークロードとしてパケットのヘッダ処理とペイロード処理に分けて考察している．ヘッダ処理ではルーティングテーブル検索，パケットのフラグメンテーション処理，QoS のためのキュー管理，TCP トラフィックモニタリングを選出している．一方でペイロード処理には，暗号アルゴリズム，データ圧縮，Forward Error Correction（FEC），画像圧縮を選出している．また文献 24) では，これらの分類をさらに一歩進めて，ベンチマークプログラムをマイクロレベル，IP レベル，アプリケーションレベルに分類している．こうした分類により，NP のような複雑なシステムにおける評価対象の範囲を計測のニーズに合わせることが可能となる．たとえば，暗号アルゴリズムの評価を行う場合でも，内部コプロセッサの性能とパケットレベルでのスループットでは評価手法が異なる．

ベンチマークのもう 1 つの視点として，異なるアーキテクチャ間の性能比較がある．異なるアーキテクチャの比較を行うためには，システムレベルでの機能の抽象化が必要となってくる．文献 25) では，ベンチマークプログラムをシステムの階層構造から，システムレベル，機能レベル，マイクロレベル，ハードウェアレベルに分類して定義することにより NP システムのモデル化を行い，様々な NP アーキテクチャに適用可能な評価手法を提案している．さらには，最も抽象度の高い見地からベンチマークの方法論を議論しているのが文献 26) である．そこでは，ベンチマークにおける 3 つの仕様，(1) 機能仕様，(2) 環境仕様，(3) 測定仕様，を定めている．その中で中心的役割を果たす環境仕様では，ネットワークインタフェースやその制御インタフェース，トラフィックや負荷状況などを定義し，多様なアーキテクチャの性能比較を行うためのベンチマークとして汎用性を高めている．

3.2 シミュレータ

NP におけるシミュレーションの場合，ある特定のシステム向けに開発したコードの挙動を詳細に再現するためのものと，特定の実装に依存しない抽象的なフレームワークにより様々な構成のシステム性能を評価するためのものに分類することができる．前者のものは，多くの NP システムの開発環境に付随して配布されている．主に研究が進められているのは後者であり，ここではそれらについて述べる．

3.2.1 シミュレーションフレームワーク

NP システムの抽象的なシミュレーションフレームワークを構築する場合，パケットのデータフローに従ってモデル化を行うのが開発者にとって理解が容易である．文献 27) では，システムを，(1) トラフィック生成部，(2) 入力解析部，(3) 入力インタフェース，(4) プロセッサ部，(5) 出力インタフェース，(6) 出力解析部，にコンポーネント化し，クロック同期を用いた実時間シミュレーションによる正確な評価を実現している．ただし，シミュレータの実装が Cisco の NP である Toaster2 [28] の構造（**図 7**）に依拠しており，他の NP にも適用可能となるような汎用性の向上が求められる．そのためには，文献 29) にあるように，NP の各機能をモジュール化するなどしてアーキテクチャの変更を容易にしなければならない．

また，NP 用シミュレーション環境の構築に，既存

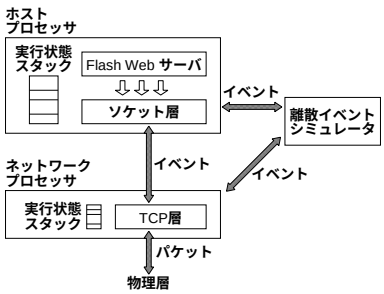

図 8 Countach におけるホストプロセッサと NP のモデル化
Fig. 8 Host procesor and NP models in Countach.

の汎用 OS の環境を有効利用するという手法もある．文献 30) では，Unix 上でユーザレベル TCP スタックを用い，細粒度のフックを埋め込むことでプログラムのメモリ参照の振舞いおよび各種装置におけるイベント処理の正確な同期をシミュレートする手法について述べている．この手法を用いて，エンドノードにおける TCP off-loading エンジン開発をターゲットとしたシミュレーションフレームワーク（Countach と呼んでいる）を構築している（**図 8**）．

3.2.2 設計空間の探索

NP 開発では，様々なシステム構成およびワークロードに対する性能評価を通じて，より高い性能を達成するアーキテクチャの開発が重要となる．ここでは，特にそのような設計空間の探索を実現する技術の研究について述べる．

設計空間における多くのポイントをシミュレートする場合，アーキテクチャの変更を容易にする必要がある．先に紹介した文献 29) では，高度にモジュール化されたソフトウエアルータである Click [31] をベースにした，NP の処理能力モデルを構築し，こうした問題に対処している．Click のフレームワークに NP モデル（プロセッサやメモリ接続チャネルなど）を追加するだけでよく，他のレイヤ 3 機能は Click のモジュールを活用することで，システム性能のシミュレーションによる柔軟な評価を実現している．

また，設計空間の探索は抽象度の高い作業であるため，シミュレーションよりも抽象度の高い解析的なシステムモデルも有用である．文献 32) では，NP システムを以下のように階層的なモデルに分解し，解析モデルを構築している．

(1) NP システムは 1 つの I/O インタフェースと複数の処理エンジンからなるクラスタ構成をとる．
(2) 各処理エンジンには複数の PE および 1 つのメモリインタフェースを搭載する．
(3) 各 PE は命令セットとデータキャッシュを有する．

これらの構成要素においてパラメータ化を行い，ベンチマークワークロードに対する性能を解析的に算出することができる．さらに文献 33) では，このフレームワークを用いて処理能力と消費電力に関する分析も行っている．また，より高レベルなシステム構成に関する研究には文献 34) もある．これは，解析的なフレームワーク [35] を用いて PE 間の接続トポロジに特化した検討を行っている．

一方で，これらの解析的なフレームワークは，抽象度が高すぎるため実際のシステムの特性を十分に反映できないという問題があり，注意が必要である．たとえば，先にあげた文献 34) で議論されている解析モデルでは，PE 集合を用いてパイプライン構成をとるより，各 PE がパケット処理のすべてを行う方式の方が性能的に優れていると結論づけている．しかし，実際には命令キャッシュの制約やチップ面積，通信コストの問題など☆から複数の PE を用いてパイプライン構成をとる方が現実的である場合が多い．

3.2.3 遅延の解析

NP においては確定的な性能の確保（ラインスピードの達成）が重要視されるため，ソフトウエアの機能単位となるコードブロックにおける最悪実行時間の分析が重要となる．文献 36) では，マルチスレッド環境を想定した最悪実行時間の算出法について述べている．具体的には，実行コードの制御フローグラフ（Control Flow Graph，CFG）を元に，それぞれのブロックの実行回数ならびに実行時間に関する制約をすべて列挙し，線形計画法により最もコストの高いパスの実行時間を算出するというものである．マルチスレッドへの対応においては，CFG に中断ノードおよび中断エッジという概念を導入し，スレッド間制御切替えおよび切替えによるメモリアクセス遅延時間隠蔽もサポートしている．一方で，このような解析で得られる最悪実行時間は理論的最悪値であることから悲観的（pessimistic）過ぎる傾向がある．しかし，NP の場合は CFG が比較的単純であり，比較的大きなプログラムでもシミュレーションによる測定時間の 4 倍程度に算出値を収めることができると報告されている．

4. プログラミング支援環境

NP におけるソフトウエア開発では，ハードウェアに関する高度な知識に基づいた低レベルなアセンブリ言語によるプログラミングが要求されることが多く，

☆ たとえば，命令キャッシュの参照の局所性や，ルーティングテーブルのような共有データへのアクセス競合などを考えるとよい．

コストが高いという問題がある．また，並列化による処理レート向上のために，多くのNPでハードウェアによるマルチスレッド環境がサポートされているが，このような複雑なハードウェア構成もプログラミングをより難しくしている．そこで，NPのソフトウエア開発を容易にするプログラミング環境に関する研究がいくつかなされている．それらのうちの1つの方向性は，3章で述べたようなシステムの抽象化を基礎にしたモジュール化である．一方で，このようなプログラムのモジュール化を進めると，実行フローの全体的な最適化が難しくなる．そのため，NPの目的の1つである確定的な性能の確保ができなくなる恐れがあり，トレードオフが存在する．もう1つの方向性としては，マルチスレッドスケジューリングがあげられる．プログラマが全責任を負うnon-preemptive環境における低レベルなスレッドスイッチングや，preemptiveなスレッドスケジューラの導入が考えられるが，ここにもトレードオフが存在する．また，NP用開発言語ならびにコンパイラの研究も重要である．本章では，これらのNPにおけるプログラミング技術を中心に研究成果を概観する．

4.1 モジュール化のためのプログラミングフレームワーク

NPプログラミングにおけるモジュール化という観点からは，先に紹介した高いモジュール性を持つソフトウエアルータClick[31]のフレームワークが有用である．文献37)では，ClickをNPに応用したプログラミングモデルNP-Clickを提案している．Clickのフレームワークを用いることで，NPの様々な機能を抽象化し，それらの上に各種モジュールを提供することを可能にしている．実装においては，ClickはC++で実装されていたのに対して，NP-Clickはこれを IXP1200用の専用開発環境であるMicroengine C（IXP-C）[38]を用いて実装し直している．性能評価では，パケットサイズが十分大きい場合にはIXP-Cを用いた従来の低レベルプログラミングによる実装と比較して，ほぼ遜色のない性能を実現していることを示している．

文献39)では，別のアプローチによるモジュール化支援プログラミング環境NEPAL（NEtwork Processor Application Language）を提案している．先のNP-ClickではNP特有機能の抽象化を言語レベルで静的に実現したのに対して，NEPALではランタイムによる動的な制御機構による支援を利用して実現している．ここでのランタイムは主にコンパイラを指しており，独自の高レベル言語NEPALで記述されたパケット処理のルーチンから，コンパイラによって動的にモジュールを構成する．モジュール構成では，NEPALで記述されたものだけではなく各NPプラットホームの開発環境で実装したものも利用することができる．重要なのは動的な最適化で，モジュールの通信バスの利用を軽減するような最適化や，プロセッサスケジューリングの最適化などを動的に行う．ただし，性能評価においては，比較が提案システムを用いたものだけであり，実際のNPにおいて実用的な性能が達成されるかどうか不明である．一般的にはランタイムを用いた手法はオーバヘッドが大きく[37]，2.2節で述べたような超並列アーキテクチャのように確定的な性能よりもシステムスケーラビリティを重視したアーキテクチャなどで有用であると考えられる．

また文献40)では，言語レベルの静的なモジュール化とランタイムによる動的なモジュール化の中間的なアプローチとして，コンポーネントと呼ぶコードの最小単位を動的にバインドしてモジュールを構成する手法について提案している．各コンポーネントはマシン語によって実装される（もしくはそれぞれの開発環境によりネイティブコードが生成される）という前提のもので，プログラミングを容易にする階層的な抽象化，コンポーネント間のデータの受け渡しのためのレジスタの割当て方式，コンポーネントの動的なバインド手法を設計し実装している．性能評価では，動的なモジュール構成のないモノリシックな実装に対して2%程度と非常に小さいオーバヘッドを実現している．

4.2 スケジューリング

NPでは，高いスループットの達成のためにマルチスレッドをサポートしている場合が多い．マルチスレッドは大きくわけて，マルチプロセッサによるスレッドの並列実行，プロセッサ内におけるマルチスレッドのハードウェアサポート，汎用OSのような擬似的な時分割マルチスレッドのサポート，の3つに分類することができる．NPにおいては，それらのうち前者の2つが主流となっている．

NPにおいては確定的な遅延および帯域幅を実現することが重要であるため，スレッドスケジューリングは比較的静的なものが利用されることが多い．たとえば，IntelのIXP1200では，メモリアクセス遅延隠蔽を目的として，ハードウェアによるnon-preemptiveマルチスレッド環境をPE内で提供している．そのため，プログラムされた自主的なプロセッサ放棄によるスレッド切替えのほかに，メモリアクセスなどでプロセッサがストールする際にも自動的にスレッド切替えがラウンドロビンで行われる．また，ソフトウエアでマルチスレッドを管理する場合でも，各スレッドの実

行コードを参照することで，スケジューリングをあらかじめ決定しておく方が効率が良い．このような静的なスレッドスケジューリングを用いたNPの研究には文献14)，15)などがある．

一方で，超並列アーキテクチャ指向なマルチプロセッサ環境では，より柔軟なマルチスレッド機構が望まれ，ソフトウエアによる支援が必要となる．文献41)では，NPをマルチプロセッサでマルチスレッドをサポートする実時間システムと定義し，公平性を満たすPfair（Proportionate Fairness）スケジューリングと呼ばれる方式のNPへの適用を提案している．一方で，このような動的なスケジューリングを行う場合，確定的な性能保証が難しくなるため，システムスケーラビリティとのトレードオフとなる．

4.3 コンパイラ

NPにおけるプログラミングでは，そのアーキテクチャの特殊性からコンパイラも性能に大きな影響を与える重要なポイントとなる．たとえば，NPではワードやバイトの境界をまたぐようなデータ処理など汎用プロセッサにはない命令も効率良く処理する必要がある[9]．NPに特化したコンパイラの最適化手法の研究として文献42)〜44)などがある．これらのコンパイラは，C言語もしくは制約付きC言語を対象にしていることが多い．また，NPの適用先として有望視されているアクティブネットワークでは，各ルータのアーキテクチャに依存しない実行コードの注入が必要となる．文献45)では，PLAN[46]と呼ばれるアクティブネットワークのためのパケット処理言語から派生したSNAPという言語のNPにおけるJust-in-Time（JIT）コンパイラの可能性について議論している．

5. アプリケーション技術

当初よりNPは高速なレイヤ3ルーティングを対象として開発されてきたという背景を持つが，NPのネットワーキングにおける他の領域への適用がいくつか研究されている．本章ではそれらについて紹介する．

5.1 パケットの分類

パケットの高度な分類はネットワークセキュリティ機能の中心的役割を担っており，NPによる高速処理が期待されている．文献47)では，DDoS（Distributed Denial of Service）攻撃を軽減するためのNetBouncerと呼ばれるパケットフィルタリング機能[48]のNPへの実装について，特にIntelのIXP1200上の実装に関する知見がまとめられている．また，文献49)では，アクセス制御リスト（ACL）を用いたパケットの分類機能を，NP上で実現するための要件について述べている．ACLに基づいたファイアウォールなどでラインスピードを達成するためには，非常に短い時間でフィルタの検索が完了しなければならないが，理論的には検索空間が膨大で，資源の限られた一般的なNPでは実装が難しいとされていた．この研究では，ISPや企業で用いられているファイアウォールのACLを詳細に分析し，実際の検索空間は十分小さいことを示している．そのうえでNPに実装するための要件を，次のようにまとめている．

(1) IPアドレス対（送信者と受信者）に関する検索とトランスポート層の情報に関する検索を分離する．
(2) IPアドレス対の重複する部分は別途管理し，IPアドレス対に対して1つのフィルタを返すようにする．
(3) トランスポート層のフィールド検査ルールはそれほど多くないので特別なハードウェア装置を用いた検索をサポートする．

5.2 レイヤ4機能のoff-loading

NPの適用先の1つとして，エンドノードにおけるTCP/IP処理のNPへのoff-loadingが検討されている．一方で，エンドノードに接続された補助装置によるTCP/IPの処理という方式自体は古くから提案されている[2]．現在のNPの形態に比較的近いアーキテクチャを持つTCP/IP off-loadingエンジンの研究としては文献50)があり，ワークステーション用TCP/IP通信ボードを提案している．実装にあたっては，TCP/IP処理の機能をプロトコルヘッダ処理部と資源管理部に分離し，前者を各種処理機能を高速に実行可能なLSIで実装し，後者を汎用RISCプロセッサ（クロック33 MHzのSPARClite）を用いてプログラム可能にしている．一般的に，TCP処理は複雑な状態情報の管理が必要であり，実現が困難であることが多い．上記の研究でもコネクション多重化に対応していないなどの不備はあるが，先駆的な研究の1つとしてあげられる．ほかにも，これまでに述べた文献11)，20)，30)もTCP/IP処理のoff-loadingを目的としているが，コネクション数やスループットに対してスケーラブルな実装はまだできていないのが実情である．

5.3 ストレージサービス

文献51)では，組み込み用プロセッサを搭載したハードウェアによる，ネットワークストレージサービスのためのシステムを開発している．システム設計では，ボード内の通信バスとしてPCIを用いていたり，ネットワーク処理にLinuxカーネルを用いたりしてお

表 1 NP システム開発における目標と手法
Table 1 The goals and techniques in NP system development.

	決定論的実時間性重視	スケーラビリティ重視
アーキテクチャ	・ハードウェアマルチスレッディング ・マルチプロセッサパイプライニング	・超並列アーキテクチャ ・動的な PE 間通信
性能評価	・ベンチマーク ・エミュレータによる細粒度挙動分析	・抽象化モデルにおける設計空間の探索 ・解析モデルによる評価
プログラミング環境	・アセンブリ言語による実装 ・低レベルマルチスレッディング ・静的な資源アクセス解決および最適化	・高級言語による実装 ・モジュール化による抽象プログラミング ・動的なマルチスレッディング環境

り，正確には NP の応用とはいえない．しかし，ネットワークストレージは NP の格好のアプリケーションとして期待されている[15]．

5.4 レイヤ 7 補助プロセッサ

NP においては，PE で処理するより hard-wired なチップを用いた方が良い性能が得られる場合には，積極的にコプロセッサを採用する傾向がある．文献 52) では，レイヤ 7 の処理を補助する NP 用のコプロセッサのアーキテクチャを提案している．提案しているプロセッサはバレルシフタに似た構造を持ち，HTTP における URL ベースのスイッチなどで必要となるパターンマッチング機能や，IP ルーティングにおける radix-tree 検索機能，セキュリティ機能などで利用される MD5 アルゴリズムなどへの応用例を示し，シミュレーションによる性能評価を行っている．

こうしたコプロセッサの実装プラットホームとして，FPGA も有力な候補である．近年，FPGA の性能が大きく向上したため，実際の NP 製品に搭載されることも多い（たとえば Intel IXP1200）．FPGA を用いたコプロセッサの研究としては，文献 53) があり，先の文献 52) で提案したコプロセッサの FPGA チップにおける実装を検討している．また文献 54) では，移動端末における消費電力とサービス品質を考慮するために，FPGA による DSP（Digital Signal Processing）チップを用いたシステムアーキテクチャを提案している．今後の無線通信の重要性を考慮すると，こうしたシステムも広義の NP システムと考えてよいだろう．また，先に紹介した Comet [22] も FPGA を用いてコプロセッサを実装している．

6. おわりに

本稿では，NP におけるアーキテクチャ，性能評価，プログラミング，アプリケーションの各研究分野について成果を概観した．繰り返し述べているように，NP ではラインスピードの実現が重要視され，そのため決定論的な実時間性が強く求められる．一方で，ネットワークの超高速化にともない，システムのスケーラビリティを考慮したシステム開発も必要である．この 2 つの要件はトレードオフの関係があり，アーキテクチャおよびプログラミング環境について技術を大きくまとめると**表 1** のようになる．

本稿で注意が必要となるのは，学術論文以外の文献（各種製品情報など）についてはあまり触れることができなかった点である．特に 2 章で述べたアーキテクチャについては，多くのベンダから多種多様な NP がリリースされている．それらについては，各社から製品とともにリリースされているホワイトペーパーや，ベンダ向けフォーラムの Network Processing Forum☆（NPF）や，コンファレンスの Network Processor Conference☆☆を参照するとよい．

これから NP は，多種多様なアーキテクチャが考案される初期の研究開発段階から，実用化を含めた次の段階へ移行するだろう．たとえば現在では，ソフトウエアルータとしてデファクトスタンダードの地位を確立している Zebra☆☆☆の商用版である ZebOS Advanced Routing Suite では NP を用いることができ[55]，実用化が進められている．また，より高いスループットを要する基幹ルータへ応用が可能かどうかも NP の当初の目的から 1 つの焦点になると考えられる．さらには，家庭内ネットワーク用ルータなどの低価格なネットワーク製品への応用も進められてきている．そのため，パケットのレイヤ 3 転送だけではなく，様々なエンドノード向けサービスの実装が望まれる．また，製品の低価格化のためには，NP チップを含めたボードの製造コストだけでなく，NP 上で動作するソフトウエアの開発コストの低減化なども重要となるだろう．このように，今後はより実用的な技術の研究開発が求められている．

謝辞 本稿の執筆にあたり，匿名査読者諸氏から数多くの有益なコメントをいただいた．ここに感謝する．

☆ http://www.npforum.org/
☆☆ http://www.networkprocessors.com/
☆☆☆ http://www.zebra.org/

参 考 文 献

1) Coady, Y., Ong, J.S. and Feeley, M.J.: Using Embedded Network Processors to Implement Global Memory Management in a Workstation Cluster, *Proc. 8th International Symposium on High Performance Distributed Computing*, Redondo Beach, California, pp.319–328 (1999).
2) Powers, G.: A front end telnet/rlogin server implementation, *Proc. UniForum 1986*, pp.27–40 (1986).
3) Crowley, P., Franklin, M.A., Hadimioglu, H. and Onufryk, P.Z. (Eds.): *Network Processor Design: Issues and Practices*, Vol.1, chapter 9: An Industry Analyst's Perspective on Network Processors, pp.191–218, Morgan Kaufmann Publishers (2003).
4) Tennenhouse, D.L., Smith, J.M., Sincoskie, W.D., Wetherall, D.J. and Minden, G.J.: A Survey of Active Network Research, *IEEE Communications Magazine*, Vol.35, No.1, pp.80–86 (1997).
5) 山本 幹：アクティブネットワークの技術動向，電子情報通信学会論文誌 B，Vol.J84-B，No.8，pp.1401–1412 (2001).
6) Roberts, L.G.: Beyond Moore's Law: Internet Growth Trends, *IEEE Computer*, Vol.33, No.1, pp.117–119 (2000).
7) Crowley, P., Franklin, M.A., Hadimioglu, H. and Onufryk, P.Z. (Eds.): *Network Processor Design: Issues and Practices*, Vol.1, chapter 1: Network Processors: An Introduction to Design Issues, pp.1–8, Morgan Kaufmann Publishers (2003).
8) Shah, N. and Keutzer, J.: Network Processors: Origin of Species, *Proc.17th International Symposium on Computer and Information Science* (*ISCIS XVII*), Orlando, Florida (2002).
9) Nie, X., Gazsi, L., Engel, F. and Fettweis, G.: A New Network Processor Architecture for High-Speed Communications, *Proc. IEEE Workshop on SiGNAL PROCESSING SYSTEMS* (*SiPS'99*), Taipei (1999).
10) Crowley, P., Fiuczynski, M.E., Baer, J.L. and Bershad, B.N.: Characterizing Processor Architectures for Programmable Network Interfaces, *Proc. 2000 International Conference on Supercomputing*, Santa Fe, N.M. (2000).
11) Nordqvist, U. and Liu, D.: Packet Classification and Termination in a Protocol Processor, *Proc. 2nd Workshop on Network Processors*, Anaheim, California, pp.88–99 (2003).
12) Chiueh, T.C. and Pradhan, P.: Cache Memory Design for Network Processors, *Proc. 6th International Symposium on High-Performance Computer Architecture*, Toulouse, France (2000).
13) Gopalan, K. and Chiueh, T.C.: Optimizations for Network Processor Cache, *Proc. SC2002 High Performance Networking and Computing*, Baltimore (2002).
14) Shimonishi, H. and Murase, T.: A network processor architecture for flexible QoS control in very high-speed line interfaces, *2001 IEEE Workshop on High Performance Switching and Routing* (*HPSR2001*), Dallas, Texas, pp.402–406 (2001).
15) Georgiou, C.J., Salapura, V. and Denneau, M.: A Programmable Scalable Platform for Next Generation Networking, *Proc. 2nd Workshop on Network Processors*, Anaheim, California, pp.1–9 (2003).
16) IBM Blue Gene team: Blue Gene: A vision for protein science using a petaflop supercomputer, *IBM Systems Journal*, Vol.40, No.2, pp. 310–326 (2001).
17) Waingold, E., Taylor, M., Srikrishna, D., Sarkar, V., Lee, W., Lee, V., Kim, J., Frank, M., Finch, P., Barua, R., Babb, J., Amarasinghe, S. and Agarwal, A.: Baring It All to Software: Raw Machines, *IEEE Computer*, Vol.30, No.9, pp.86–93 (1997).
18) Chuvpilo, G., Webtzlaff, D. and Amarasinghe, S.: Gigabit IP Routing on Raw, *Proc. 1st Workshop on Network Processors*, Boston, MA (2002).
19) Karim, F., Nguyen, A., Dey, S. and Rao, R.: On-Chip Communication Architecture for OC-768 Network Processors, *Proc. 38th Design Automation Conference* (*DAC2001*), Las Vegas, Nevada, pp.678–683 (2001).
20) 西川博昭，青木一浩：プロトコル多重処理のデータ駆動型実現法とその実験的検討，電子情報通信学会論文誌 D-I，Vol.J85–D–I，No.7，pp.635–643 (2002).
21) Melvin, S., Nemirovsky, M., Musoll, E., Huynh, J., Milito, R., Urdaneta, H. and Saraf, K.: A Massively Multithreaded Packet Processor, *Proc. 2nd Workshop on Network Processors*, Anaheim, California, pp.64–74 (2003).
22) 陣崎 明，中村 修，村井 純：ギガビットルータ Comet のアーキテクチャとその評価，インターネットコンファレンス'98 論文集 (1998).
23) Wolf, T. and Franklin, M.: COMMBENCH – A Telecommunications Benchmark for Network Processors, *Proc. IEEE International Symposium on Performance Analysis of Systems and Software*, Austin, TX (2000).

24) Memik, G., Mangione-Smith, W.H. and Hu, W.: NetBench: A Benchmarking Suite for Network Processors, *Proc. IEEE/ACM International Conference on Computer-Aided Design* (*ICCAD*), San Jose, CA (2001).
25) Chandra, P.R., Hady, F., Yavatkar, R., Bock, T., Cabot, M. and Mathew, P.: Benchmarking Network Processors, *Proc. 1st Workshop on Network Processors*, Boston, MA (2002).
26) Tsai, M., Kulkarni, C., Sauer, C., Shah, N. and Keutzer, K.: A Benchmarking Methodology for Network Processors, *Proc.1st Workshop on Network Processors*, Boston, MA (2002).
27) Suryanarayanan, D., Marshall, J. and Byrd, G.T.: A Methodology and Simulator for the Study of Network Processors, *Proc. 1st Workshop on Network Processors*, Boston, MA (2002).
28) Crowley, P., Franklin, M.A., Hadimioglu, H. and Onufryk, P.Z. (Eds.): *Network Processor Design: Issues and Practices*, Vol.1, chapter 11: Cisco Systems – Toaster2, pp.235–248, Morgan Kaufmann Publishers (2003).
29) Crowley, P. and Baer, J.L.: A Modeling Framework for Network Processor Systems, *Proc. 1st Workshop on Network Processors*, Boston, MA (2002).
30) Pradhan, P., Xu, W., Nair, I. and Sahu, S.: Efficient and Faithful Performance Modeling for Network-Processor Based System Design, *Proc. 2nd Workshop on Network Processors*, Anaheim, California, pp.125–132 (2003).
31) Kohler, E., Morris, R., Chen, B., Jannotti, J. and Kaashoek, M.F.: The Click modular router, *ACM Trans. Comput. Syst.*, Vol.18, No.3, pp.263–297 (2000).
32) Franklin, M.A. and Wolf, T.: A Network Processor Performance and Design Model with Benchmark Parameterization, *Proc. 1st Workshop on Network Processors*, Boston, MA (2002).
33) Franklin, M.A. and Wolf, T.: Power Considerations in Network Processor Design, *Proc. 2nd Workshop on Network Processors*, Anaheim, California, pp.10–22 (2003).
34) Gries, M., Kulkarni, C., Sauer, C. and Keutzer, K.: Exploring Trade-offs in Performance and Programmability of Processing Element Topologies for Network Processors, *Proc. 2nd Workshop on Network Processors*, Anaheim, California, pp.75–87 (2003).
35) Thiele, L., Chakraborty, S., Gries, M. and Künzli, S.: Design Space Exploration of Network Processor Architecture, *Proc. 1st Workshop on Network Processors*, Boston, MA (2002).
36) Crowley, P. and Baer, J.L.: Worst-Case Execution Time Estimation for Hardware-assisted Multithreaded Processors, *Proc. 2nd Workshop on Network Processors*, Anaheim, California, pp.36–47 (2003).
37) Shah, N., Plishker, W. and Keutzer, K.: NP-Click: A Programming Model for the Intel IXP1200, *Proc. 2nd Workshop on Network Processors*, Anaheim, California, pp.100–111 (2003).
38) Intel Corp.: *Intel Microengine C Compiler Support: Reference Manual* (2002).
39) Memik, G. and Mangione-Smith, W.H.: NEPAL: A Framework for Efficiently Structuring Applications for Network Processors, *Proc. 2nd Workshop on Network Processors*, Anaheim, California, pp.112–124 (2003).
40) Campbell, A.T., Chou, S.T., Kounavis, M.E., Stachtos, V.D. and Vicente, J.: NetBind: A Binding Tool for Constructing Data Paths in Network Processor-Based Routers, *Proc. 5th IEEE Conference on Open Architectures and Network Programming* (*OPENARCH'03*), New York, NY (2002).
41) Srinivasan, A., Holman, P., Anderson, J., Baruah, S. and Kaur, J.: Multiprocessor Scheduling in Processor-based Router Platforms: Issues and Ideas, *Proc. 2nd Workshop on Network Processors*, Anaheim, California, pp.48–62 (2003).
42) Wagner, J. and Leupers, R.: C Compiler Design for an Industrial Network Processor, *Proc. ACM Workshop on Languages, Compilers, and Tools for Embedded Systems*, Snowbird, Utah, pp.155–164 (2001).
43) Kim, J., Jung, S., Paek, Y. and Uh, G.R.: Experience with a Retargetable Compiler for a Commercial Network Processor, *Proc. International Conference on Compilers, Architecture, and Synthesis for Embedded Systems*, pp.178–187, ACM (2002).
44) Wagner, J. and Leupers, R.: Advanced Code Generation for Network Processors with Bit Packet Addressing, *Proc. 1st Workshop on Network Processors*, Boston, MA (2002).
45) Kind, A., Pletka, R. and Stiller, B.: The Potential of Just-in-Time Compilation in Active Networks based on Network Processors, *Proc. IEEE OPENARCH'02*, New York, NY, pp.79–90 (2002).
46) Hicks, M., Kakkar, P., Moore, J.T., Gunter, C.A. and Nettles, S.: PLAN: A Packet

Language for Active Networks, *Proc. 3rd ACM SIGPLAN International Conference on Functional Programming Languages*, pp.86–93, ACM (1998).
47) Thomas, R., Mark, B., Johnson, T. and Croall, J.: High-speed Legitimacy-based DDoS Packet Filtering with Network Processors: A Case Study and Implementation on the Intel IXP1200, *Proc. 2nd Workshop on Network Processors*, Anaheim, California, pp.133–147 (2003).
48) Thomas, R., Mark, B., Johnson, T. and Croall, J.: NetBouncer: Client-legitimacy-based High-performance DDoS Filtering, *Proc. 3rd DARPA Information Survivability Conference and Exposition* (*DISCEX III*), Washington, D.C (2003).
49) Kounavis, M.E., Kumar, A., Vin, H., Yavatkar, R. and Campbell, A.T.: Directions in Packet Classification for Network Processors, *Proc. 2nd Workshop on Network Processors*, Anaheim, California, pp.148–157 (2003).
50) 長田孝彦，東海林敏夫，山下博之，塩川鎮雄：マルチメディアコンテンツ転送向け高性能 TCP/IP 通信ボードの構成と評価，情報処理学会論文誌，Vol.39, No.2, pp.347–355 (1998).
51) 上村哲也，熊谷幸夫：Active Network Storage のネットワーク性能の評価，電子情報通信学会論文誌 D-I，Vol.J85-D-I, No.9, pp.841–849 (2002).
52) Memik, G. and Mangione-Smith, W.H.: A Flexible Accelerator for Layer 7 Networking Applications, *Proc. 39th Design Automation Conference* (*DAC'02*), pp.646–651 (2002).
53) Memik, G., Memik, S.O. and Mangione-Smith, W.H.: Design and Analysis of a Layer Seven Network Processor Accelerator Using Reconfigurable Logic, *Proc. 10th Annual IEEE Symposium on Field-Programmable Custom Computing Machines* (*FCCM'02*) (2002).
54) Smit, G.J., Havinga, P.J., Smit, L.T., Heysters, P.M. and Rosien, M.A.: Dynamic Reconfiguration in Mobile Systems, *Proc. 12th International Conference on Field Programmable Logic and Application* (*FPL 2002*), Montpellier, France, pp.171–181 (2002).
55) IP Infusion Inc.: Teja and IP Infusion IXP1200 Integrated Router Application (2002). http://www.ipinfusion.com/pdf/AN101.pdf

(平成 15 年 5 月 12 日受付)
(平成 15 年 8 月 24 日採録)

河合 栄治（正会員）

1996 年京都大学理学部数学科卒業．1998 年奈良先端科学技術大学院大学情報科学研究科博士前期課程修了．2001 年同大学同研究科博士後期課程修了．2000 年 10 月より，科学技術振興事業団さきがけ研究 21「機能と構成」領域研究員．2003 年 4 月より，奈良先端科学技術大学院大学附属図書館研究開発室科助手．インターネットにおける大規模情報配信システムの開発，高速 I/O 指向オペレーティングシステム，ネットワークプロセッサ，次世代電子図書館システムの研究に従事．博士（工学）．

門林 雄基（正会員）

1996 年大阪大学大学院基礎工学研究科物理系専攻博士後期課程中退．同年大阪大学大型計算機センター助手．1997 年大阪大学大学院基礎工学研究科物理系専攻情報工学分野より博士（工学）取得．1999 年大阪大学大型計算機センター講師．2000 年奈良先端科学技術大学院大学情報科学研究科助教授．WIDE プロジェクトボードメンバ．Content Routing Network Forum 代表．レイヤ 7 での QoS 実現を目標とし，セキュリティ，CDN，マルチキャスト等の研究に従事．著書に岩波講座インターネット第 2 巻『ネットワークの相互接続』.

山口 英（正会員）

1990 年 10 月大阪大学大学院基礎工学研究科情報工学専攻博士後期課程を中退し，大阪大学情報処理教育センター助手として着任．1992 年 10 月奈良先端科学技術大学院大学情報科学センター助教授．1993 年 4 月より，同大学情報科学研究科助教授．2000 年 4 月より，同大学情報科学研究科教授．大規模分散処理環境構築，ネットワークセキュリティ等の研究を行う．また，WIDE Project のメンバとして，広域コンピュータネットワークの構築・研究に従事する．工学博士.