広域ネットワークへの適用を考慮したアクティブフロー制御プラットフォーム

柏 大[†1,†2] Eric Y. Chen[†1,†3] 冨士 仁[†1]
重野 寛[†2] 岡田謙一[†2] 松下 温[†2,†4]

ISP（Internet Service Provider）バックボーンなどの広域ネットワーク上でネットワークサービスを実現するための基盤となる「アクティブフロー制御プラットフォーム」を提案する．提案プラットフォームでは，ネットワーク装置上でプログラムを動作させることで，動的な帯域制御や優先転送などの様々な高機能化ネットワークサービスを実現し，広域ネットワークで必要な，高速性と多様性を両立するフロー制御方式，分散プログラムの効率的な管理方式，および，プログラム間の協調動作機構を持つことを特徴とする．本論文ではさらに，提案プラットフォームの応用例として，動的優先転送と DDoS（Distributed Denial of Service）攻撃対策サービスアプリケーションを実装して，提案プラットフォームの実現性を示す．また，フロー制御方式の有効性と提案プラットフォームの実運用への適用について考察する．

An Active Flow Control Platform for Wide-area Networks

Dai Kashiwa,[†1,†2] Eric Y. Chen,[†1,†3] Hitoshi Fuji,[†1]
Hiroshi Shigeno,[†2] Ken-ichi Okada[†2]
and Yutaka Matsushita[†2,†4]

This paper proposes a novel network platform: Active Flow Control Platform, designed for wide-area networks, on which various value-added network service applications can be deployed. It enables various service applications, such as dynamic bandwidth and priority control, by injecting service application programs into network nodes. It provides service application programs on high-speed nodes with flexible flow control mechanisms and environment that allows interoperability with other programs. Moreover, this paper introduces some service applications that allow dynamic priority control and provide countermeasure against DDoS (Distributed Denial of Services) attacks to demonstrate the values of the platform. At the end, this paper also evaluates the effectiveness of the flow control method and application to real use.

1. はじめに

ISP（Internet Service Provider）などが提供するネットワークサービスは，高速化，低価格化に加えて，近年では高機能化が進んでいる．高機能化ネットワークサービスの例として，VPN（Virtual Private Network）や SLA（Service Level Agreement）サービスなどがあげられる．これらのネットワークサービスは，MPLS や DiffServ などの標準化された技術を実装したネットワーク装置によって提供されている．しかし，標準化や実装の過程に長い時間を要する点，利用形態の変化や技術の進歩に応じたカスタマイズが難しい点などが問題となっており，その解決策として，アクティブネットワーク技術が注目されている[1)]．

アクティブネットワークは，ルータなどのネットワーク装置（以下，ノード）に制御用コードを注入することで，ノードがアプリケーション層までの処理を実行するネットワークである[1)]．アクティブネットワーク技術を利用したサービスプラットフォームを広域ネットワークへ適用すると，様々な高機能化ネットワークサービスが迅速に展開可能となる．しかし関連する研

†1 NTT 情報流通プラットフォーム研究所
NTT Information Sharing Platform Laboratories
†2 慶應義塾大学理工学研究科
Graduate School of Science and Technology, Keio University
†3 東京大学理工学研究科
Graduate School of Information Science and Technology, The University of Tokyo
†4 東京工科大学
Tokyo University of Technology

究では，広域ネットワークへの適用に関する検討は十分に行われていない．

そこで本論文では，ISP バックボーンなどの広域ネットワークへの適用を考慮したサービスプラットフォームとして，アクティブフロー制御プラットフォームを提案する．本論文では，ネットワーク上を流れる同一属性を持ったパケット群をフローと呼ぶ．また，サービスプラットフォーム上で提供される応用サービスをサービスアプリケーションと呼び，各サービスアプリケーションを提供するためにノード上に注入される制御用コード群をまとめてプログラムと呼び，サービスアプリケーションを利用する際に，ユーザ端末側で利用するソフトウェアをユーザアプリケーションと呼ぶ．

以下，2 章で，関連研究の動向と広域ネットワークへの適用における要求条件と課題について整理した後に，3 章で，アクティブフロー制御プラットフォームについて詳述する．4 章では，プロトタイプ上でサービスアプリケーションを実現し，5 章でフロー制御方式の有効性と実運用への適用について考察する．最後に，6 章でまとめを示す．

2. 関連研究動向と課題

2.1 関 連 研 究

アクティブネットワーク技術を利用したサービスプラットフォームの研究は，パケット内に格納されたプログラムがキューイングなどのノードの処理内容を制御する「カプセル方式」と，プログラムをノードに注入し，そのプログラムが帯域制御や優先転送などの「フロー制御」を行う「プログラマブルスイッチ方式」とに分類される[2]．カプセル方式では，パケットを単位とする細かいノード制御が可能となる．一方，プログラマブルスイッチ方式では，カプセル方式に比べて単位は粗くなるが，プログラム容量を大きく確保できるために多様なフロー制御が可能になる．また，ノードの制御をパケット内のプログラムに許可するカプセル方式に比べて，セキュリティの確保が容易になる．著者らは，フロー制御の多様性とセキュリティを重視して，プログラマブルスイッチ方式のサービスプラットフォームについて検討を行っている[3]．

これに関連する研究として，文献 4) では，高機能プロトコルソフトウェアをノードに動的に注入するための言語およびシステムを提案している．文献 5) では，商用スイッチ上にプログラムを動作させる環境を構築し，スイッチのフロー制御内容をプログラムから動的に変更するサービスアプリケーションについて紹介している．また，文献 6) では，アクティブネットワークを構成するノードの構成要素と構成要素間のインタフェースについて定義し，様々な OS 上で実装を行ったうえでパケット転送性能などについて報告している．

2.2 広域ネットワークへの適用における要求条件と課題

広域ネットワーク上のサービスプラットフォームでは，ノードが多数のユーザを収容し，多くのフローを同時に処理する必要がある．よって，ノードにおけるフロー制御には高速性が求められる．また，ユーザの様々なニーズに対応するためには，ノードにおけるフロー制御には多様性が求められる．ここで多様性とは，たとえば，ネットワーク状況に適応した帯域制御[7]や特定のユーザアプリケーションに適したパケットの加工[8]などの，フロー制御内容の自由度の高さを指す．

フロー制御に関して，文献 4)，6) などの研究では，すべてのパケットをユーザプロセスレベルまで引き上げ，プログラムの指示により処理を行った後に再送信する方式をとっている．プログラムは Java 言語などの高級言語を用いて記述され，記述内容により，柔軟な処理が可能となる．しかしこの方式では，データコピーやコンテキストスイッチによって処理負荷や処理遅延が増加するため，高速性に問題がある．一方，文献 5) の研究では，ASIC を使ったハードウェアスイッチでパケットを転送し，プログラムがそれを制御する方式をとっている．この方式では高速性は達成できるが，フローの識別条件や制御内容がハードウェアスイッチの仕様に依存し，たとえばフロー識別単位としてパケットヘッダ部以外の属性値を指定できないことや，パケットを加工して再び送信することができないことなど，多様性に問題がある．このように，高速性と多様性を両立するフロー制御方式についてはまだ確立されていない．

また，広域ネットワークのサービスプラットフォームでは，ノード上に多種かつ多数のプログラムが混在するため，プログラムを効率的に管理する必要がある．さらに，広域ネットワークで有効なサービスアプリケーションとして，たとえば，ネットワーク状況適応型品質別マルチキャストルーティング[9]や DDoS 攻撃対策[7]などを実現しようとした場合，大局的なネットワーク状況を分散配置されたプログラムが把握し，プログラムが連携して動作する必要がある．このような場合，すべてのプログラムを集中的に制御することは難しく，プログラム間で自律的に協調動作を行うことが望まれる．しかし，既存の関連研究において，これらを実現するプログラムの管理方法，および，プログラム間の

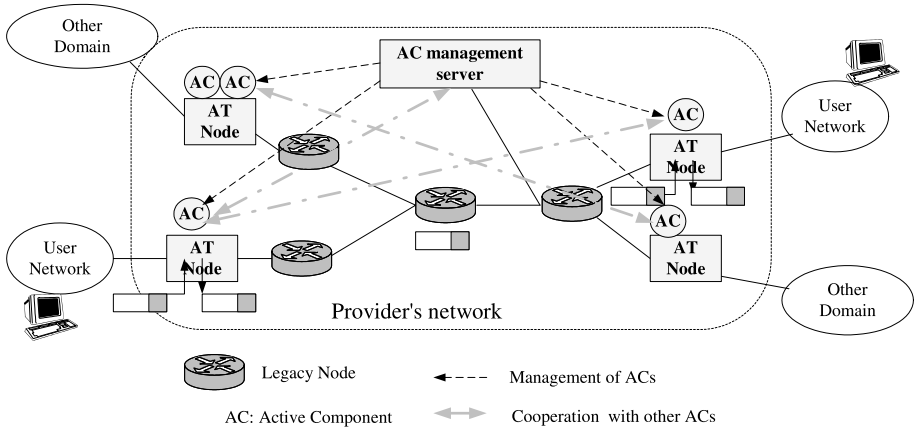


図 1 Active Flow Control Platform 概観
Fig. 1 Active Flow Control Platform overview.

協調動作方法について具体的に示したものはない．

3. アクティブフロー制御プラットフォーム

本章では，広域ネットワークへの適用を考慮したサービスプラットフォームとして，アクティブフロー制御プラットフォーム（以下，AFCP: Active Flow Control Platform）を提案する．AFCP では，高速性と多様性を両立するフロー制御方式を提案する．また，分散配置されたプログラムの効率的な管理方式を提案し，これを用いたプログラム間の協調動作機構を提供する．

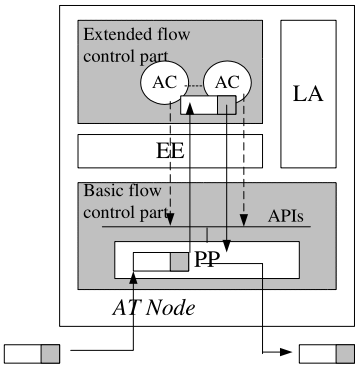
図 1 に，AFCP の概観を示す．AFCP では，広域ネットワーク上に複数の AT ノード（Active Traffic control node）が分散配置される．AT ノードは，プログラム（以下，AC: Active Component）を動作させることで様々なフロー制御機能を提供するノードである．また，AC 管理サーバ（AC management server）は，ISP バックボーンなどの管理ドメイン単位に存在するサーバであり，AT ノード上の AC を管理する．AFCP では，提供するサービスアプリケーションに応じてクラスが設計され，そのインスタンスが AC となって必要なノードに配置される．

3.1 高速性と多様性を両立するフロー制御方式

図 2 に AT ノードの機能モジュール構成を示す．AFCP では，AT ノードにおいて，フロー制御機能を基本フロー制御機能と拡張フロー制御機能とに分離し，各々を PP（Packet Processor）と AC とで処理する方式を提案する．以下では，各機能モジュールの役割を説明する．

PP（Packet Processor）

PP は，カーネルレベルで，基本フロー制御機能を提供するとともに，これを操作するための API（Application Program Interface），および，より複雑な拡張フロー制御を行うための API を上位機能モジュールに対して提供する．基本フロー制御機能としては，**図 3** に示すクラス分け機能（Classification），スケジューリング機能（Scheduling），シェーピング機能（Shaping），マーキング機能（Marking）を提供する．クラス分け機能は，パケットをフローに分類し，さらにフローの集合ごとにクラスに分類するために用いられる．また，拡張フロー制御対象パケットを AC に送信するためにも用いられる．分類のための識別子として，宛先アドレス，プロトコル番号などの IP ヘッダ情報やペイロード情報を指定可能とする．各クラスには複数のフローが入力され，クラス別のスケジューリング処理，シェーピング処理，マーキング処理が行われる．スケジューリング処理では，キューイング方



AC: Active Component, EE: Execution Envorinment,
LA: Local Agent, PP: Packet Processor

図 2 AT ノードの構成
Fig. 2 Structure of AT node.

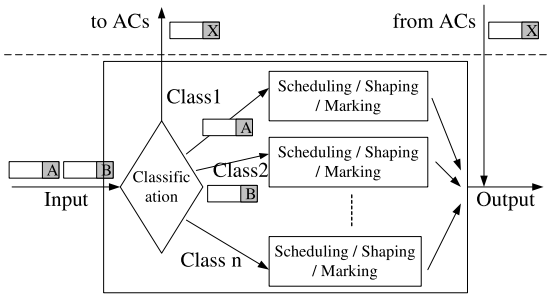


図 3 PP（Packet Processor）におけるフロー制御
Fig. 3 Flow control of PP (Packet Processor).

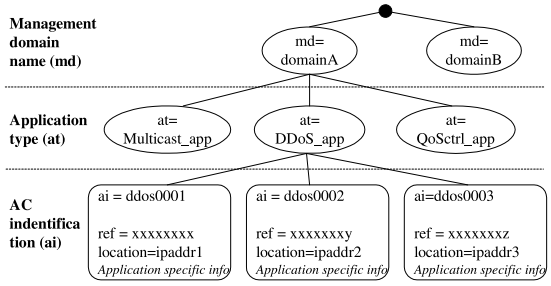


図 4 分散 AC 管理ディレクトリ
Fig. 4 Distributed AC management directory.

式や優先度を指定することによって帯域制御や優先転送などを行う．シェーピング処理では，出力帯域を指定された値に制限する．マーキング処理では，ToS フィールド値を設定することで DiffServ などの外部のフロー制御機構との連携を可能にする．

EE（Execution Environment）

EE は，AC の実行環境であり，AC 管理機能（動作開始/停止，移動，削除など）と，PP のフロー制御 API を AC が呼び出すためのライブラリを提供する．

AC（Active Component）

AC は，高級言語で実装するプログラムであり，EE が提供するライブラリを利用して PP のフロー制御機能を操作し，さらに，拡張フロー制御対象のパケットをキャプチャして，様々な拡張フロー制御処理を実施する．拡張フロー制御には，パケットの中身の精査や加工をともなう，特定のサービスアプリケーションに特化した複雑な処理機能が含まれる．

LA（Local Agent）

AC 管理サーバとのインタフェースであり，AC 管理サーバとの間で AC の送受信を行う．

提案方式では，基本フロー制御として必要な機能をある程度絞り込み，カーネルレベルで処理を行う．これにより，文献 4)，6) などの，ユーザプロセスレベルで提供する方式と比較して基本フロー制御処理を高速化する．また，AC からフロー制御 API を組み合わせて利用することで多様なフロー制御を可能とする．さらに，基本フロー制御機能でカバーできない処理は，AC が拡張フロー制御機能として提供することで多様性を向上させる．

3.2 AC 管理方式と AC 間協調動作

AFCP では，AC の管理を AC 管理サーバで行い，管理方式として**図 4** に示す分散 AC 管理ディレクトリを用いる方式を導入する．

ネットワーク上で動作するプログラムをディレクトリを用いて管理するためのスキーマとして文献 10) が提案されている．文献 10) では，ディレクトリエントリ内でのプログラムリファレンスの表記法を定義し，クラス名やプログラムコード位置などの属性を持ったエントリ構成を提案している．このスキーマを AC 管理に適用することで，分散配置された AC を，ディレクトリを用いて管理することが可能になる．しかし，サービスプラットフォーム上の AC は，管理ドメインやサービスアプリケーションの種別ごとに検索や操作が行われる可能性が高いと考えられるため，この特性に適した情報構造が必要になる．そこで AFCP では，文献 10) のプログラムリファレンス表記法を利用したうえでこのスキーマを拡張し，広域ネットワークで想定される管理形態に則した DIT（Directory Information Tree）とエントリ構成を導入する．この DIT は，管理ドメイン名（md），アプリケーションタイプ（at），AC 識別子（ai）から構成される．また，エントリ構成は，プログラムリファレンス（ref），動作位置情報（location），フロー情報などの AC 固有の情報（Application specific info）の属性を持ち，各エントリが 1 つの AC の情報を格納する．

AFCP では，AC 管理サーバから分散 AC 管理ディレクトリを検索して AC 識別子を取得し，その情報を基に EE の AC 管理機能を呼び出すことで，各 AC の動作開始や停止，他 AT ノードへの移動などの管理を実行する．分散 AC 管理ディレクトリは，管理ドメインごとに md レベルでサブツリーに分割され，各々が AC 管理サーバによって管理される．AC 管理サーバでは，サブツリー間をまたがる検索には DSP（Directory System Protocol）などの標準的なプロトコルを用いる．

また AC 間の協調動作のために，AFCP では，分散 AC 管理ディレクトリを用いた AC 発見機構，AC 間でのメッセージ伝達機構，および，AC の遠隔操作機構を提供する．協調動作にはサービスアプリケーションによって様々な形態が考えられるが，本論文では，

AC 間でスループットや輻輳情報などのフロー情報を交換し，その結果に基づいて，各 AC が自律的または他 AC からの操作によってフロー制御内容を変更する形態を想定する．

AT ノードの EE は，AC が注入されると，(md, at, ai) からなる識別名および各エントリ属性値を分散 AC 管理ディレクトリに登録する．以後，各 AC は分散 AC 管理ディレクトリを検索することで協調対象の AC を発見し，リファレンスを取得することが可能になる．図 4 では，たとえば，検索パスとして "$md = domainA,\ at = DDoS\_app$" を指定して検索を行うことで，domainA 内の DDoS_app サービスアプリケーションを提供する全 AC を発見し，リファレンスを取得することができる．さらに位置情報を示す属性値として "$location = ipaddr1$" を指定することで，ipaddr1 の AT ノード上で動作する AC を発見し，リファレンスを取得することができる．発見した AC に対しては，全 AC が共通で持つコミュニケーションメソッドを用いたメッセージ伝達が可能になる．コミュニケーションメソッドでは，共通のメッセージフォーマットを用いて，情報伝達 (tell)，情報問合せ (ask)，状態通知 (ready) などの動作内容[11]を伝達することで，AC 間でフロー情報や動作状態などを共有する．共有情報に応じて，AC 間でのフロー制御内容の同期や，フロー制御対象の分担などの協調動作が行われる．協調動作の具体的なアクションについては，サービスアプリケーションの要求条件に応じて，AC のプログラムコード内に記述する．さらに AFCP では，AC の実装によりサービスアプリケーション独自の遠隔操作メソッドを公開し，AC 管理サーバまたは他の AC から，フロー制御内容を直接操作することも可能とする．

3.3 セキュリティ対策

AC から分散 AC 管理ディレクトリへの登録および検索要求時や，AC 間でコミュニケーションを行う際には，通信相手の正当性について認証を行う必要がある．これを実現するため，AFCP の LA および AC 管理サーバでは，IPsec[12]により，通信時の相互認証，暗号化処理を行う．これにより，AT ノードに不正な AC が注入されることを防ぎ，不正な AT ノード上からの AC 管理サーバ操作や，不正な AC 間コミュニケーションなどを防ぐ．また，通信の秘匿性も確保する．AFCP では，管理ドメインごとに CA (Certificate Authority) 局を設けて X.509 形式の証明書を各 AT ノードに発行し，通信時にはそれらを用いて認証，および，動的な暗号鍵の作成を行う．広域ネットワークでは多数の AT ノードに対応した証明書管理が必要であり，実運用では十分な性能を持った CA 局を使うことを想定している．

表 1 実装環境
Table 1 Implementation environment.

Hardware	CPU: PentiumIII 866 MHz RAM: 128 Mbyte
Network Device	Ethernet NIC (10/100Base-T)
PP	Linux kernel2.4 + libpcap 0.6.2 + our extension(C++)
EE	JRE(Java Runtime Environment) Ver 1.3 + our extension(Java)
AC	EJB(Enterprise Java Beans)

4. プロトタイプとサービスアプリケーション例

4.1 プロトタイプ

プロトタイプシステムとして AT ノードおよび AC 管理サーバを実装した．AT ノード実装に用いた環境を**表 1** に示す．また，EE と PP 間の制御は JNI (Java Native Interface) を，分散 AC 管理ディレクトリの操作は JNDI (Java Naming and Directory Interface) を利用し，AC 間および AC と AC 管理サーバ間の協調動作のためのインタフェースは RMI-IIOP (Remote Method Invocation over Internet Inter-ORB Protocol) を用いた．

PP としては，Linux カーネルを拡張して基本フロー制御機能を実装した．これらを操作するための主なフロー制御 API を**表 2** に示す．この中で，フロー操作のための関数は基本フロー制御機能の操作を可能にする．また，ネットワーク状況取得のための関数は，ネットワーク状況に応じたフロー制御を可能にし，パケット送受信のための関数は，libpcap[13]ライブラリを用いて AC とのパケット送受信を可能にする．たとえば，*capturePacket* 関数は AC がパケットを捕捉するために用いられ，AC がこの関数を呼び出した時点以降に PP に到着したパケットのうち，引数 (sig) にマッチした最初のものを戻り値として AC に送信する．また，連続して到着するパケットを捕捉する場合には，*captureContinuousPackets* 関数を用いる．この関数を AC が呼び出すと，パケットが到着するごとに，到着順で PP から AC に連続してパケットが転送される．表 2 において引数で使われている "Signature" 型は，IP パケットの先頭からのオフセットバイト位置 ($offset$) と，識別情報を表す 0，1 のビット列 (sig_body) およびビットマスク (sig_mask) から構

表 2 PP（Packet Processor）の提供する主なフロー制御 API
Table 2 Main flow control APIs of PP (Packet Processor).

役割	関数名	内容
フロー操作	Flowclass PP::createFlowClass(Interface *intf*, Flowclass *parent*, Queuing *queuing_method*, int *priority*)	*intf* のインタフェースに *parent* を親として，*queuing_method* で指定されたキューイング方式で，*priority* に優先度を設定したクラスを作成する (Scheduling)
	boolean Flowclsss::classifyPackets(Signature *sig*, Flowclass *class_id*)	*sig* にマッチするパケットを *class_id* クラスに分類する (Classification)
	boolean Flowclass::setShapingValue(int *bandwidth*)	*bandwidth* を最大帯域に設定する (Shaping)
	boolean Flowclass::modifyToS(Signature *sig*, *ToSvalue*)	*sig* にマッチするパケットの ToS フィールド値を *ToSvalue* に変更する (Marking)
ネットワーク状況取得	Throughput PP::getInThroughput(Signature *sig*)	*sig* にマッチするパケットの入力帯域値を取得する (Monitoring)
	Throughput PP::getOutThroughput(Signature *sig*)	*sig* にマッチするパケットの出力帯域値を取得する (Monitoring)
パケット送受信	Packet PP::capturePacket(Signature *sig*)	*sig* にマッチした最初のパケットを取り込み，戻り値としてこれを AC に渡す (Capturing)
	void PP::captureContinuousPackets(Signature *sig*, AC *ac*)	*sig* にマッチしたパケットが到着する度に，*ac* のイベントハンドラを呼び出してパケットを連続的に *ac* に送信する (Capturing)
	Packet PP::copyPacket(Signature *sig*)	*sig* にマッチした最初のパケットのコピーを取り込み，戻り値としてこれを AC に渡す (Copy)
	void PP::copyContinuousPackets(Signature *sig*, AC *ac*)	*sig* にマッチしたパケットが到着する度に，*ac* のイベントハンドラを呼び出してパケットのコピーを連続的に *ac* に送信する (Copy)
	void PP::releasePacket(Packet *pkt*)	*pkt* で指定されるパケットを送出する (Releasing)

表 3 EE（Execution Environment）の提供する主な AC 管理インタフェース
Table 3 AC management interface of EE (Execution Environment).

役割	関数名	内容
AC 注入	boolean EE::deployAC(AC_type *at*)	*at* で指定された AC のコードを AC 管理サーバから EE 上に注入する．
動作開始	boolean EE::startAC(AC_id *ai*)	EE 上で動作中の *ai* の動作を開始する．
動作停止	boolean EE::stopAC(AC_id *ai*)	EE 上で動作中の *ai* の動作を停止する．
AC 移動	boolean EE::moveAC(AC_id *ai*, ATNode *an*)	*ai* で指定された AC を *an* で指定された AT ノードの EE 上へ移動させる．

成され，*offset* 位置から *sig_body* と *sig_mask* との AND 条件にマッチするパケットを識別する．Signature 型ではペイロード部まで含めた任意のフロー指定が可能なため，アプリケーションまでの情報を基にしたフロー識別が可能となる．ただし，IP や TCP ヘッダには可変長の Option フィールドがあるため，各関数では，これらのフィールドが使われているパケットに対して，Signature 型の代わりに文字列型を引数として上位層情報を指定することも可能としている．たとえば，*classifyPackets*(“*tcp dport* = 80″, *class_id*) と指定することで，IP ペイロードの開始位置を自動判断して，TCP プロトコルで宛先ポートが 80 のパケットを *class_id* に分類するようになる．そのほか，“tcp dport=80 http get” などの指定も可能とする．PP では，複数のパケット送受信関数呼び出しが発生した際には，呼び出しの発生した順に引数のマッチング処理を行う．よって，複数の AC が同じパケットにマッチする関数呼び出しを行った場合，先に呼び出しを行った AC のみが処理対象となる．

EE としては，AC の管理機能を Java 実行環境上に構築し，また，PP が提供するフロー制御 API を AC が呼び出すためのライブラリを実装した．AC の管理機能のためのインタフェースを**表 3** に示す．表 3 において，*deployAC* 関数は，引数で指定された種類の AC を AC 管理サーバから EE 上へ注入する．*startAC* 関数および *stopAC* 関数は，引数で指定された AC の動作を開始/停止させ，*moveAC* 関数は，引数で指定された AC を，別の AT ノードの EE 上に Serialized（直列化）して移動させる．また，EE は，AT ノード上で動作する AC の動作状態を管理し，AC が異常終了した場合にその制御対象となっていたフロー分類情報を PP から削除する機能と，分散 AC 管理ディレクトリのエントリ情報を削除する機能を持つ．これにより，AC の異常時には該当フローは通常のルーティング処理が行われるとともに，他の AC および AC 管理サーバの管理者が，分散 AC 管理ディレクトリの検索時に異常を認識できるようになる．

また，AFCP 上で動作するサービスアプリケーションの例として，動的優先転送と DDoS 攻撃対策を実現する AC について実装を行った．これらのサービスアプリケーションや AC の詳細については，4.2 節，4.3 節で述べる．

4.2 動的優先転送

第 1 のサービスアプリケーションとして，動的優先転送を行うための「優先設定 AC」を実装した．以下，優先設定 AC の動作について，表 2 の API と関連付けながら説明する．

本サービスアプリケーションでは，ユーザアプリ

ケーションから「優先転送要求パケット」をフローの経路に従って送信すると，優先設定 AC がこのパケットを *capturePacket* 関数を用いて取得し，その中に含まれる要求を認証する．認証に成功すると，*createFlowClass* 関数を用いて高い優先度のクラスを作成し，*classifyPackets* 関数を用いて要求フローをそのクラスに分類する．優先設定 AC では，ユーザ要求の正当性認証，フローの優先設定，実際に使用したネットワークリソース情報の管理などを行う．

図 5 に示す環境下において優先設定 AC を動作させ，実験を行った．Flow analyzer で観測された各 Flow の出力を**図 6** に示す．図 6 では，時刻 0 で Flow1，時刻 10 で Flow2 の送信を開始している．時刻 20 で優先転送要求パケットを AT ノードに送信すると Flow1 の優先設定が行われ，時刻 40 で優先転送解除パケットを送信すると優先設定が解除されており，ユーザアプリケーションからの要求に応じて動的に優先設定が変更されている．なお，図 6 では時刻 20 や時刻 40 などで出力のピークが出ているが，これはクラス分けの変更などにともなって，一時的に発生しているパケット出力量の揺らぎによるものである．

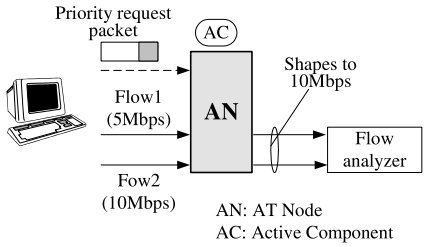


図 5 動的優先転送の動作検証構成
Fig. 5 Testbed settings of dynamic priority control.

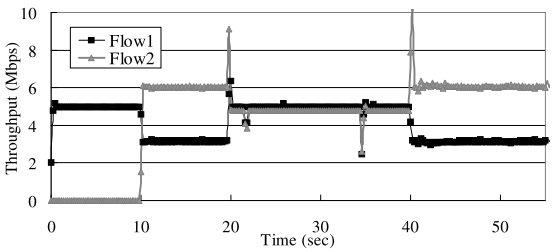


図 6 優先設定 AC によるフロー制御結果
Fig. 6 The result of flow control by priority control AC.

現在の VoIP（Voice over IP）サービスや SLA サービスでは，特定フローを固定的に優先設定している．これに対し，優先設定 AC は AFCP におけるフロー制御の多様性を利用して優先度を動的に変更することで，時間を限定した優先設定やポリシの変更にともなう設定変更などを可能とする．

4.3 DDoS 攻撃対策

第 2 のサービスアプリケーションとして，分散サービス停止（DDoS: Distributed Denial of Service）攻撃対策のための Active Shaping モデル（以下,「AS モデル」）[14]を実装した．DDoS 攻撃は，多数の端末から標的のサーバに大量のパケットを送信する攻撃である．AS モデルは，ノードでの自律分散型の対策により，DDoS 攻撃被害を抑止し，正規利用者を保護する．AFCP 上では「DDoS 対策 AC」を構築してこのモデルを実現した．

図 7 は，正規利用者（Legitimate User）の Flow4 が存在する状態で，攻撃プログラム（Attack Program）から Flow1〜3 によって，防御対象サイト（Protected Site）のサーバが攻撃を受けている状態を示している．以下，DDoS 対策 AC の動作について，表 2 の API と関連付けながら説明する．

(1) 攻撃の検出

防御対象サイト近くの DDoS 対策 AC は，パケット転送量の変化から攻撃を検出する．これは *getInThroughput* 関数によって防御対象サイトへの

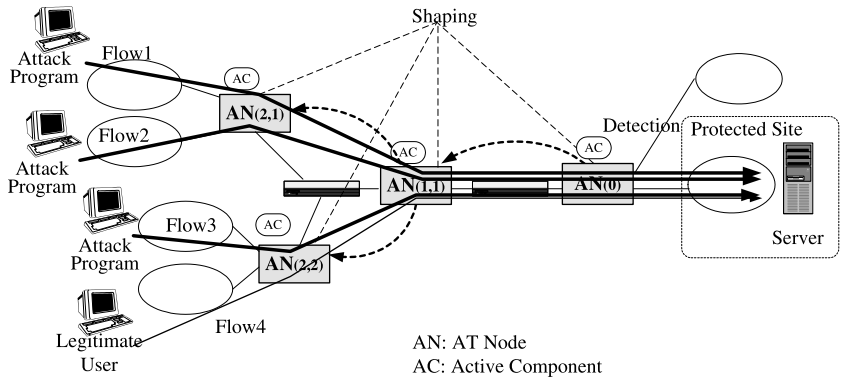


図 7 DDoS 攻撃対策の動作概観
Fig. 7 Overview of the countermeasure against DDoS attacks.

入力帯域を解析することで実現される．解析方法として，たとえば「帯域閾値」と「時間閾値」を用意して，帯域閾値を時間閾値以上，連続して超過した場合に，攻撃の発生を検出する．

(2) 容疑パケットの分散位置での帯域制限

攻撃を検出したDDoS対策ACは，対策として，防御対象サイト宛の「容疑パケット」の帯域を制限する．さらに，攻撃の発生と防御対象サイトのアドレス情報を，隣接するATノード上のACに順次伝達し，伝達されたACがその情報にマッチするフローの帯域を同期して制限することで，ネットワーク全体の分散型で容疑パケットの帯域制限を行う．これを実現するため，DDoS対策ACは図4のエントリ属性のACの固有情報として，隣接ノードの位置情報を保持する．各ACはこの情報を基に分散AC管理ディレクトリを検索して隣接ノード上のDDoS対策ACのリファレンスを取得し，協調動作のための遠隔操作メソッドである帯域制限メソッドを呼び出す．これを再帰的に実施することで，自律分散型での対策が可能となる．帯域の制限は，$createFlowClass$ 関数と $setShapingValue$ 関数により帯域が制限されたクラスを作成し，防御対象サイトのアドレスとそのクラスの識別子を引数として $classifyPackets$ 関数を呼び出すことで実現される．このようにDDoS対策ACでは，容疑パケットの帯域制限をAC間で同期して実施する．

(3) 攻撃発生源の特定と更なる帯域制限

次に，ユーザネットワークを直接収容するATノード（図7の $AN_{(2,1)}$，$AN_{(2,2)}$）上のDDoS対策ACが，収容するネットワークごとのトラヒックを分析し，攻撃プログラムが埋め込まれているネットワークを特定して，そのネットワークからの「有害パケット」の帯域をさらに制限する．攻撃発生源の特定は，ネットワークアドレスごとに不正パターンを引数として $getInThroughput$ 関数によって取得できるパケット転送量を解析することで実現される．たとえば典型的な攻撃手法であるSmurf攻撃では大量のICMPパケットが連続送信される[15]ため，ネットワークごとにICMPパケットの転送量を測定することで，攻撃発生源のネットワークアドレスを特定できる．さらなる帯域制限は，(2)と同様の方法で，$setShapingValue$ 関数の帯域制限値をより小さくすることで実現される．このようにDDoS対策ACでは，攻撃発生源の特定とさらなる帯域制限をAC間で分担して実施する．また，新しく発見された不正パターン情報については，AC間でコミュニケーションメソッドを用いて共有する．

図7において，各アクセス回線を10 Mbpsとしたテストベッド上でDDoS対策ACを動作させ，以下のシナリオに基づいて，**表4**に示す攻撃フロー（Smurf）と正規フロー（TCP）を送信し，防御対象サイトにおける各フローの流入量について調べた．測定結果を**図8**に示す．

表4 入力フロー
Table 4 Input flow.

分類	送信位置	送信量	送信パターン
攻撃フロー	Flow1	5 Mbps	連続送信
	Flow2	10 Mbps	連続送信
	Flow3	3 Mbps	連続送信
正規フロー	Flow4	2 Mbps	バースト送信

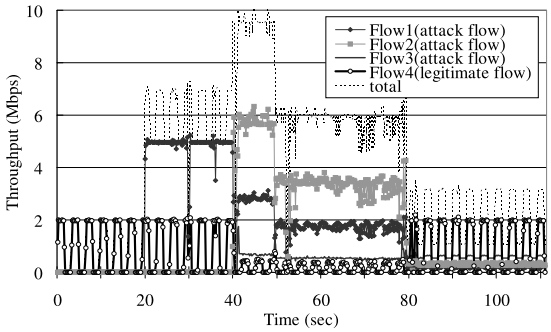


図8 DDoS対策ACによるフロー制御結果
Fig. 8 The result of flow control by DDoS countermeasure ACs.

1) 時刻0からflow4（正規フロー）の送信を開始
2) 時刻20でflow1（攻撃フロー）の送信を開始
3) 時刻40でflow2, 3（攻撃フロー）の送信を開始

図8において，時刻40からの3つの攻撃フローにより，防御対象サイトのアクセス回線が輻輳し，正規フローの出力が低下している．時刻50付近で，DDoS対策ACが攻撃を検出して，防御対象サイト宛の全フローを容疑パケットとして帯域制限することで，輻輳が緩和されている．さらに，時刻80での，攻撃発生源の特定とさらなる帯域制限により，攻撃トラヒックの帯域がより制限されるとともに，正規フローの出力が攻撃発生以前の値に回復している．

DDoS攻撃は広域ネットワークでの対策が有効である[15]．DDoS対策ACは，AFCPにおけるフロー制御の多様性を利用し，さらに，分散的な対策を行うためにAC間の協調動作機構を活用して，自律分散的なDDoS攻撃対策を可能としている．なおDDoS攻撃では，AC管理サーバなどが攻撃対象となりうることを想定する必要がある．これを回避するためには，AC間のコミュニケーションに関する通信路を別網，または，論理的に別網とし，その網上にAC管理サーバ

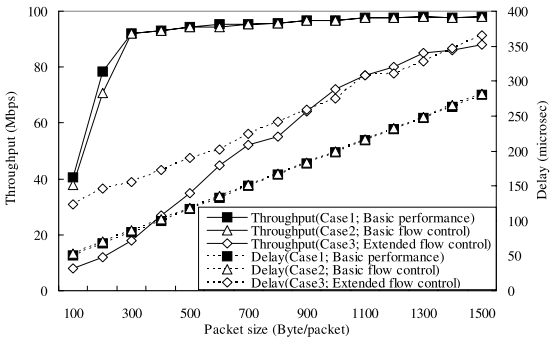


図 9 基本/拡張フロー制御部の処理性能比較
Fig. 9 Performance comparison of basic/extended flow control.

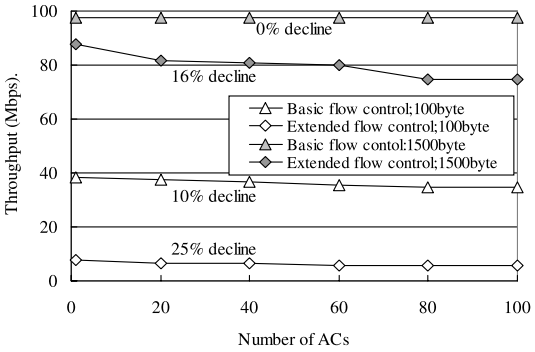


図 10 AC 数を増加させたときの処理性能
Fig. 10 Performance increasing the number of ACs.

を設置するなどの方法が考えられるが，詳細については本論文のスコープを超えるため，これ以上は議論しない．

5. 考　　察

5.1 フロー制御方式の有効性評価

本節では，プロトタイプの AT ノードを用いて，提案フロー制御方式の有効性について評価を行う．

最初の実験では，フロー制御を行わずパケットを転送するケース（Case1），基本フロー制御機能を動作させるケース（Case2），および，拡張フロー制御機能を動作させるケース（Case3）における，パケット転送性能および遅延時間を測定し，各ケースの結果を比較した．ここでの転送性能は，AT ノードへの入力を増加させていったときに，パケット廃棄を発生させずに，全入力パケットをすべて出力できる限界帯域を示したものである．また遅延時間は，入力を 1 Mbps に固定し，パケットが AT ノードに入力されてから出力されるまでの時間を示したものである．Case2 では，DDoS 対策 AC を 1 つ動作させ，PP において全入力パケットを 1 つのフローとして分類して，FIFO（First In First Out）キューイング処理を行った．また同時に，攻撃発生を監視する処理を行った．Case3 では，入力されたパケットの TTL（Time To Live）値を減算する処理を行う簡易 AC を用いた．PP において全入力パケットを拡張フロー制御対象としてこの AC に振り分けた．

図 9 に，パケットサイズをパラメータとして変化させたときの，各ケースにおける測定結果を示す．Case1 と Case2 の結果から，基本フロー制御機能を動作させた場合（Case2）も，動作させない場合（Case1）とほぼ同等の転送性能および遅延時間となっており，特に 300 byte/packet 以上のパケットサイズでは Case2 は Case1 との差がほとんどないことが分かる．一方で，拡張フロー制御機能を動作させた場合（Case3）においては転送性能が低くなり，たとえば 1,500 byte/pkt においても約 88 Mbps 程度となっている．遅延時間についても AC–PP 間での処理遅延により Case3 では Case1，2 に比べて約 70 μs 増加している．

文献 4)，6) などの従来方式では，図 9 の Case3 と同様にすべてのフロー制御をユーザプロセスレベルで行っている．この結果から，提案フロー制御方式では，フロー制御機能を分離することで，すべてのフロー制御をユーザプロセスレベルで行う従来方式と比較して，基本フロー制御機能について大幅な高速化が可能であることが確かめられた．

次に，実運用環境を想定し，AT ノード上の AC 数を増加させたときの，基本フロー制御機能および拡張フロー制御機能の転送性能への影響について調べた．この実験では，前節の実験と同じ AC を AT ノード上で複数動作させ，AC 間で入力パケットを均等に分割して制御を行った．入力パケットとしては，100 byte と 1,500 byte を用いた．**図 10** に，AC 数を 1〜100 に変化させたときの，基本フロー制御機能および拡張フロー制御機能の転送性能を示す．AC 数の増加にともなって CPU 処理に負担がかかるようになるため性能低下が発生しているが，カーネルレベルで処理を行う基本フロー制御機能は，拡張フロー制御機能に比べて転送性能への影響が少なく，低下率が低い．この結果から，提案フロー制御方式の基本フロー制御機能は，ユーザプロセスレベルで全フロー制御を行う従来方式と比較して，AC 数の増加に関するスケーラビリティが高くなることが分かった．

さらに，フロー分類情報を複雑に指定したときの性能への影響について調べた．この実験では，前実験と同様の環境において AC 数を 100 に固定し，フロー分類情報を，宛先アドレスのみを指定した場合

表 5 フロー分類情報ごとの処理性能
Table 5 Performance changing flow identification.

classifyPackets 関数の引数	dst=192.168.1.10/32	dst=192.168.1.10/32 tcp dport=80 http get
基本フロー制御（Throughput）	34.4 Mbps	34.1 Mbps（約 1%低下）
基本フロー制御（Delay）	51.6 μs	52.0 μs（約 1%増加）
拡張フロー制御（Throughtput）	6.1 Mbps	6.0 Mbps（約 2%低下）
拡張フロー制御（Delay）	144.9 μs	145.4 μs（約 0.3%増加）

（dst=192.168.1.10/32）と，アプリケーション層までの情報をあわせて指定した場合（dst=192.168.1.10/32 tcp dport=80 http get）とで比較を行った．後者の場合，PP のフロー分類において，“tcp” の指定により IP ペイロード部の開始位置を，“http” の指定により TCP ペイロード部の開始位置を判断したうえで，宛先ポートが 80 で “GET” 属性を持ったパケットを処理することになる．入力パケットとしては，処理負荷が重くなるショートパケット（100 byte）を用いた．**表 5** に，classifyPackets 関数の引数としたフロー分類情報と，そのときの性能を示す．表 5 の結果では，パケット分類情報を複雑にしても，転送性能の低下，遅延時間の増加ともほとんど発生していない．この結果から，アプリケーション層までの識別情報を使ったフロー分類を行っても，カーネルレベルで処理を行うことより性能への影響はほとんどないことが確かめられた．

5.2 フロー制御方式の実運用システムへの適用性

前節の実験結果では，PC を用いたプロトタイプのため，基本フロー制御機能においてもたかだか 100 Mbps 程度の転送性能となっている．ISP バックボーンに設置する実運用システムでは，最低数 Gbps 程度の転送性能が必要になる．提案フロー制御方式では，基本フロー制御機能については，ソフトウェアによる実装においても数 Gbps 以上の処理が可能な NPU（Network Processor Unit）[16]の利用が可能と考えられ，これにより実運用に耐えうる性能を達成できるもの予想できる．一方，拡張フロー制御機能の処理には多様性が求められるために，CPU の高速化などで改善はされるものの，Gbps クラスの性能を達成することは難しいと考えられる．また，AC–PP 間での処理遅延により特に通過ノード数が増えたときに遅延時間の影響が大きくなる可能性がある．よって，拡張フロー制御機能を利用したサービスアプリケーションは，限られた帯域を利用し，遅延時間をある程度許容するものに限定されるものと考えられる．具体的なサービスアプリケーション例については次節で議論する．

5.3 AFCP の適用領域

AFCP では基本フロー制御機能を利用したサービスアプリケーションで高速性を発揮する．4.2 節，4.3 節で示した以外で高速処理可能なサービスアプリケーションとして，負荷分散型ルーティングや QoS ルーティング[17]などがあげられる．これらは，AC が QoS 情報を基に転送経路を判断したうえで，ルーティング情報を変更することで実現できる．現在のプロトタイプでは，基本フロー制御部にルーティング情報の制御機構は実装されていないため，これらのサービスアプリケーションを実現する際には，PP およびフロー制御 API に拡張が必要である．一方，バッファリングをともなうパケット操作やカプセル化などの処理は，拡張フロー制御部で対応可能である．具体的なサービスアプリケーションとして，信頼性マルチキャストでの NAK パケットの集約や TCP 輻輳制御への適用があげられる．NAK パケットの集約[18]では，拡張フロー制御機能が処理対象とするパケットは，アプリケーションが送受信する全パケット中のわずかな割合となる．また TCP 輻輳制御[8]でも，拡張フロー制御機能としては TCP ウィンドウサイズ調整用パケットを作成して送信するのみであり，拡張フロー制御機能の転送帯域や遅延時間に対する要求条件は低い．よってこれらのサービスアプリケーションでは，拡張フロー制御機能の処理性能でも実用に耐えうると考えられる．

また，AFCP での，AC 間の協調動作機構は，4.3 節で示したような，ネットワーク全体でのフロー制御で効果を発揮する．その他の効果的なサービスアプリケーションとして，ネットワーク全体の QoS 情報を基にした QoS ルーティングやネットワーク状況に応じた品質別マルチキャスト[9]などがあげられる．

6. おわりに

本論文では，様々な高機能化ネットワークサービスを実現するための基盤となる，アクティブネットワークの概念を用いたフロー制御プラットフォーム，AFCP について述べた．AFCP は，広域ネットワークに適用する際に必要な，高速性と多様性を両立するフロー制御方式，分散プログラムの効率的な管理方式，および，プログラム間の協調動作機構を持つこと特徴とし，様々な高機能化サービスアプリケーションを展開可能とする．

また，動的優先設定サービスアプリケーションおよ

び DDoS 攻撃対策サービスアプリケーションを実装し，テストベッド上で実験を行って AFCP の実現性を示した．さらに，フロー制御方式の有効性と実運用への適用について評価し，考察を行った．

残された課題として，LA や AC 管理サーバが乗っ取られたときの不正なコミュニケーションを防ぐ AC 単位での認証処理や，実運用を想定した AC 管理や AC 間協調動作方式のスケーラビリティ評価などがある．今後はこれらの課題について検討を進めるとともに，NPU を用いた高速化システムや更なるサービスアプリケーションの実装を進める予定である．

謝辞 本研究を進めるにあたり貴重なコメントをいただいた慶應義塾大学の森川裕介氏，松本真弥氏に感謝します．

参 考 文 献

1) 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，Vol.J84-B, No.8, pp.1401–1412 (August 2001).
2) 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).
3) 柏 大, Chen, E.Y., 冨士 仁, 岡部恵一：プライベートサービスを実現する広域アクティブネットワークアーキテクチャ，信学技報，Vol.OFS2001-19, pp.51–57 (July 2001).
4) da Silva, S., Yemini, Y. and Florissi, D.: The NetScript Active Network System, *IEEE Journal on Selected Areas in communications*, Vol.19, No.3, pp.538–551 (March 2001).
5) Lavian, T., Wang, P.Y., Travostino, F., Subramanian, S., Hoang, D. and Sethaput, V.: Intelligent Network Services through Active Flow Manipulation, *IEEE Intelligent Networks 2001 Workshop* (*IN2001*), Boston, MA, USA, pp.73–82 (May 2001).
6) Peteron, L., Gottlieb, Y., Hibler, M., Tullmann, P., Lepreau, J., Schwab, S., Dandekar, H., Purtell, A. and Hartman, J.: An OS Interface for Active Routers, *IEEE Journal on Selected Areas in communications*, Vol.19, No.3, pp.473–487 (March 2001).
7) Chen, E.Y.: AEGIS: An Active-Network-Powered Defense Mechanism against DDoS Attacks, *Proc. IWAN2001*, Philadelphia, PA (2001).
8) Faber, T.: ACC: Using Active Networking to Enhance Feedback Congestion Control Mechanisms, *IEEE Network*, Vol.12, No.3, pp.61–65 (May/June 1998).
9) Lavian, T. and Wang, P.Y.: An Active Router Architecture for Multicast Video Distribution, *Proc. IEEE INFOCOM 2000*, Tel-Aviv, Israel, pp.1137–1146 (Apr. 2000).
10) Ryan, V., Seligman, S. and Lee, R.: Schema for Representing Java Objects in an LDAP Directory, RFC2713 (1999).
11) Finin, T., Fritzson, R., McKay, D. and McEntire, R.: KQML as an Agent Communication Language, *Proc. 3rd International Conference on Information and Knowledge Management* (*CIKM'94*), Adam, N., Bhargava, B. and Yesha, Y.(Eds.), Gaithersburg, MD, USA, pp.456–463, ACM Press (1994).
12) Kent, S. and Atkinson, R.: Security Architecture for the Internet Protocol, RFC2401 (1998).
13) McCanne, S., Leres, C. and Jacobson, V.: libpcap, Lawrence Berkeley National Labs Network Research Group. http://ftp.ee.lbl.gov.
14) Kashiwa, D., Chen, E.Y. and Fuji, H.: Active Shaping: A Countermeasure against DDoS Attacks, *Proc. ECUMN2002*, Colmar, France, pp.171–179 (2002).
15) Moore, D., Voelker, G.M. and Savage, S.: Inferring Internet Denial-of-Service Activity, *Proc. 10th USENIX Security Symposium* (2001).
16) Nikolaou, N.A. and Sanchez-P., J.-A.: Application Decomposition for High-Speed Network Processing Platforms, *Proc. ECUMN2002*, Colmar, France, pp.322–329 (2002).
17) Gurin, R.: QoS Routing in Networks with Inaccurate Information: Theory and Algorithms, *IEEE Trans. Networking*, Vol.7, No.3 (June 1999).
18) Lehman, L.-W.H., Garland, S.J. and Tennenhouse, D.L.: Active Reliable Multicast, *Proc. INFOCOM'98*, pp.581–589 (1998).

(平成 14 年 7 月 8 日受付)
(平成 14 年 12 月 3 日採録)

柏　　大（学生会員）

平成 7 年慶應義塾大学理工学部計測工学科卒業．平成 9 年同大学大学院修士課程修了．同年日本電信電話（株）入社．以来，分散処理システム，ネットワークアーキテクチャ，ネットワークセキュリティの研究開発に従事．現在，同社情報流通プラットフォーム研究所研究員．慶應義塾大学大学院後期博士課程在学中．電子情報通信学会会員．

Eric Y. Chen

平成 9 年マギル大学（カナダ）コンピュータ科学科卒業．同年日本電信電話（株）入社．平成 12 年同大学経営修士号取得．現在，日本電信電話（株）情報流通プラットフォーム研究所研究員．東京大学大学院後期博士課程在学中．アクティブネットワーク，モバイルエージェント，E コマース，ネットワークセキュリティの研究開発に従事．平成 12 年情報処理学会全国大会学術奨励賞受賞．IEEE 会員．

冨士　　仁

平成 5 年東京理科大学大学院工学研究科修士課程修了．同年日本電信電話（株）入社．現在，同社情報流通プラットフォーム研究所研究主任．ネットワークセキュリティ，サービスネットワーキングアーキテクチャ，プロジェクト管理の研究開発に従事．情報学博士．電子情報通信学会，日本品質管理学会，プロジェクトマネジメント学会各会員．

重野　　寛（正会員）

平成 2 年慶應義塾大学理工学部計測工学科卒業．平成 9 年同大学大学院理工学研究科博士課程修了．平成 10 年同大学理工学部情報工学科助手（有期）．現在，同大学理工学部情報工学科専任講師．工学博士．無線 LAN の構成法と媒体アクセス制御方式，計算機ネットワークにおけるステーション移動サポート，モバイル・コンピューティング，アクティブネットワーク，遠隔教育システム等の研究に従事．著書「〜ネットワーク・ユーザのための〜無線 LAN 技術講座」（ソフト・リサーチ・センター），「コンピュータネットワーク」（オーム社）等．電子情報通信学会，IEEE，ACM 各会員．

岡田　謙一（正会員）

慶應義塾大学理工学部情報工学科教授，工学博士．専門は，グループウェア，コンピュータ・ヒューマン・インタラクション．「コラボレーションとコミュニケーション」（共立出版）をはじめ著書多数．情報学会誌編集主査，論文誌編集主査，GW 研究会主査等を歴任．現在，GN 研究会運営委員，BCC 研究グループ幹事，日本 VR 学会仮想都市研究会副委員長，電子情報通信学会論文誌編集委員．IEEE，ACM，人工知能学会各会員．1995 年度情報処理学会論文賞，情報処理学会 40 周年記念論文賞，2000 年度情報処理学会論文賞受賞．

松下　温（正会員）

昭和 38 年慶應義塾大学工学部電気工学科卒業．昭和 43 年イリノイ大学大学院コンピュータサイエンス専攻修了．平成 1 年〜14 年慶應義塾大学理工学部教授．現在，東京工科大学教授および慶應義塾大学理工学部客員教授．マルチメディア通信，コンピュータネットワーク，グループウエア等の研究に従事．情報処理学会理事．同学会副会長，マルチメディア通信と分散処理研究会委員長，グループウエア研究会委員長，電子情報通信学会，情報ネットワーク研究会委員長，MIS 研究会委員長，VR 学会，サイバースペースと仮想都市研究会委員長等を歴任．現在，情報処理学会 ITS 研究会委員長，郵政省，通産省，建設省，農水省，都市基盤整備公団，行政情報システム研究所等の委員長，座長，委員を多数歴任．“やさしい LAN の知識”《オーム社》，“201x 年の世界”《共立出版》等著書多数．平成 5 年情報処理学会ベストオーサ賞，平成 7 年，平成 12 年情報処理学会論文賞，平成 12 年情報処理学会 40 周年記念 90 年代学会誌論文賞，平成 12 年電子情報通信学会フェロー，平成 12 年 VR 学会サイバースペース研究賞，平成 13 年情報処理学会功績賞．情報処理学会フェロー．電子情報通信学会，人工知能学会，ファジイ学会，IEEE，ACM 各会員．