仮想ネットワークアーキテクチャによるネットワークワイドな保護機構

廣津 登志夫[†] 福田 健介[†] 光来 健一[†]
明石 修[†] 佐藤 孝治[†] 菅原 俊治[†]

本論文では仮想ネットワークアーキテクチャによる保護機構（VNAP）を提案する．VNAPでは仮想LANの技術を用いて目的や用途に応じた複数の通信クラスをイントラネット内に実現する．この各々の通信クラスがOS内の資源やアプリケーションの実行環境と連係することで，イントラネット全域にわたる多層の保護階層が実現される．この通信クラスはネットワーク上の保護ポリシを具現化したものであり，その情報をネットワーク管理者・プログラマ・ユーザで共有することにより，相互の協調によるイントラネット環境の安全性の向上を実現する．本論文では，VNAPの概念や構成とあわせて，プロトタイプシステムの実装と評価についても述べる．

An Intranet Protection Mechanism Based on Virtual Network Architecture

Toshio Hirotsu,[†] Kensuke Fukuda,[†] Ken-ichi Kourai,[†]
Osamu Akashi,[†] Koji Sato[†] and Toshiharu Sugawara[†]

This paper proposes a protection architecture for intranets called a Virtual Network Architecture for Protection (VNAP), in which multiple levels of protection based on the virtual LAN technology are introduced for the whole intranet. Each protection level is bound to a group of resources and kernel functions on each end computer, and then an intranet-wide protection domain is constructed in cooperation with the kernel and the virtual LAN technology. VNAP provides a simple and clear view of protection level for users, administrators and programmers. The performance of the prototype system is also shown.

1. はじめに

インターネットの利用が広まり，ネットワークは単なる情報の収集から個人情報の交換や金銭支払のための決済に至るまで，様々な用途に使われている．そのため，目的や性質が異なり要求される安全性や通信の質も違った多様な通信が1つのネットワーク上に流れている．また，イントラネットと呼ばれる企業などの組織内ネットワークでは，その組織内だけで交換されるべき情報もインターネット向けの通信と同じネットワークの上に流れており，ネットワークのインフラストラクチャは単一であるのが一般的である．

このようなイントラネットは，通常，入口にファイアウォールを構築して外部の攻撃などから内部ネットワークを保護している．しかし，イントラネット内で稼働するソフトウェアや，部分的に外部に公開されたネットワーク（Demiliterized Zone：DMZ）で稼働するソフトウェアの問題などに起因して，意図しない情報の流出や漏洩が頻繁に起きている[1),2)]．ここでは，ホスト上のファイルなどの資源がネットワークからアクセスできるかどうかの制御は，アプリケーションの努力で実現されており，ネットワークやOSからシステムとして支援する枠組はない．

一方ネットワークに注目すると，ネットワークを流れる通信は目的・用途・性質によっていくつかの種類（通信クラス）に分けることができると考えられる．どのような通信クラスがあるかはイントラネットごとに異なるが，一例としては「インターネット向け通信」，「イントラネット内の情報交換」，「決済のための暗号化された通信」などがあげられる．そして，この通信クラスは，ホスト上のファイルなどの情報を流してよいかどうかなどの管理ポリシと密接に関係する．

そこで，ネットワーク上に目的などに応じて複数の

† NTT未来ねっと研究所
NTT Network Innovation Laboratories

通信クラスを実現し，OS の機能やアプリケーションの実行環境とそれらの通信クラスを連係させる仕組みとして，**仮想ネットワークアーキテクチャによる保護機構（VNAP：Virtual Network Architecture for Protection）**を提案する．仮想ネットワークアーキテクチャでは，ネットワークは通信クラスごとに複数の仮想ネットワークに分割・多重化される．そして，OS 内ではそれぞれの仮想ネットワークに適した機能がリソーススペースとして提供され，イントラネットにわたる保護ゾーンが構築される．これにより，通信相手や通信経路の質に関わる管理ポリシはアプリケーションから切り離され，ネットワークやシステムのサービスとして提供することが可能になる．ユーザはアプリケーションを稼働させるリソーススペースを選ぶだけで，システム管理者やプロバイダの提供する安全性のポリシを簡単に使いこなすことができるようになる．本論文では，仮想ネットワークアーキテクチャの概念や構成について述べ，プロトタイプシステムの実装とその性能を示す．

2. 背　　景

TCP/IP によるネットワークは，IP より下位の層（物理層やデータリンク層）には特別な機能は期待せず，単純な通信機能の上に動かすことを想定してプロトコルやソフトウェアが構成されてきた[3]．つまり，ネットワークは通信を担う 1 つの雲のような存在として扱われており，通信プロトコルの大部分やアプリケーションからみるとネットワークには構造がないように見える．このため，通信は IP アドレスで指定された相手との間のエンド–エンドの関係としてのみ扱えばよく，概念的には非常に単純になるという利点があった．

このエンド–エンド通信という性質に起因して，現在のネットワークとソフトウェアの構成では通信の相手を識別するのに IP アドレスかホスト名が使われており，通信の安全性や質の判断は基本的にはアプリケーションの努力により行われている．たとえば，Web サーバでは IP アドレスや逆引きにより得られる通信相手のホスト名を基に資源のアクセス権を設定することができ，また一部の Web クライアントでは URL によって通信相手をゾーン分けし，そのゾーンに応じてブラウザの機能を部分的に可能にしたり不能にしたりすることができる．しかし，これらの機能はあくまでアプリケーションの努力により IP アドレスを用いて通信相手を分別しているのであって，ネットワークのサービスとしてそのネットワークが安全かどうかなどの性質を反映させているわけではない．

一方ネットワークの方に注目すると，VPN や Multicast ネットワークのようなネットワークを中心として，既存のネットワークの上に仮想的に別トポロジのネットワークを構築するオーバレイネットワークと呼ばれる技術がある[4]．オーバレイネットワークでは，異なるセキュリティレベルのネットワークや，特殊用途のネットワークが物理トポロジと独立に構築される．これらのオーバレイネットワークの多くは，IP アドレスの異なるネットワークとして構築するか，経路制御と連動して特定の宛先向けの通信がオーバレイネットワークを経由するように設定することで，アプリケーションや OS から意識することなく利用されている．

さらに，近年，物理的なネットワークにおいてもデータリンクが高機能化し，IEEE 802.1Q Ethernet[5]のように 1 つの物理データリンク上に仮想的な複数のデータリンク（VLAN）を構築したり，IEEE 802.1p[6]のように複数の異なる優先度の通信を実現したりすることができるようになった．現在のところ，これらの機能はスイッチ間つまりネットワーク側の機能として使われることがほとんどである．たとえば IEEE 802.1Q は，スイッチ間の基幹接続で複数のデータリンクを多重化させるために使われており，ホストのインタフェース部分から直接利用することはあまりない．また IEEE 802.1p は OS 内のドライバやスイッチの設定により，トランスポート層のポート番号などのプロトコル情報を覗き見て違う優先度に割り当てる形で使われており，アプリケーションからその機能を直接利用することはできない．

しかし，これらの安全性や通信品質といった通信の性質は，そもそもエンド–エンドの通信に対して直交的であって，同じ相手に対して異なる性質を持った通信が利用できることが望ましいはずである．さらに，これらの繋がるネットワークの性質により OS が提供する機能を変えることで，安全性の低いネットワークを通る通信を通しては重要なデータの処理をさせないなどのよりきめ細かい制御が可能になると考えられる．

3. VNAP

ここでは，「仮想ネットワークアーキテクチャによる保護機構（VNAP）」の概念と構成について述べる．VNAP では実際のネットワークの運用や利用を考慮し，ネットワークで繋がれたコンピュータの世界が次のようないくつかの異なる立場の人々によって成り立っていることを想定している（**図 1**）．

- 企業や学校の内部ネットワークを構築・管理して

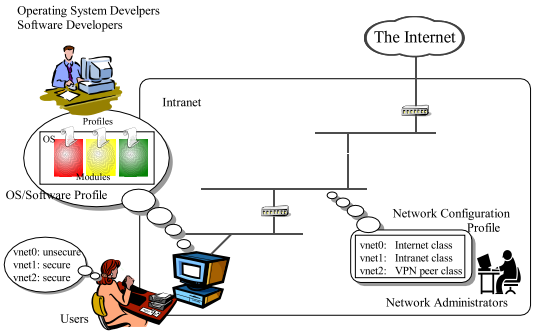

図 1 仮想ネットワークアーキテクチャ
Fig. 1 Virtual Network Architecture.

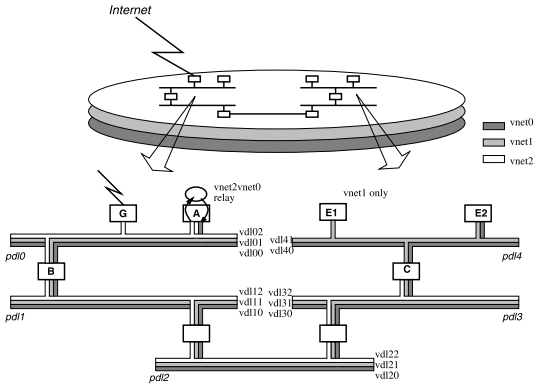

図 2 仮想ネットワーク
Fig. 2 Virtual Networks.

る人や ISP などのプロバイダのネットワークを構築・管理する「ネットワーク管理者」

- コンピュータ上で稼働する OS など基本ソフトウェアを開発する「OS 開発者」
- OS 上で稼働するアプリケーションを開発する「アプリケーション開発者」
- ネットワークとソフトウェアを利用する「ユーザ」

本論文では，このような様々な立場の人々が協調してネットワークを取り扱うためのモデルとして**仮想ネットワークアーキテクチャ**を提案する．仮想ネットワークアーキテクチャでは，1 つの物理的なネットワークの中に複数の仮想的なネットワーク（仮想ネットワーク）が多重化されていると考える．また，ネットワークに繋がる OS の内部も同様に仮想化され，複数の仮想的な OS（リソーススペース）が多重化されているように構成する．

VNAP では，ネットワーク管理者は安全性や通信の品質などの観点に立って 1 つのポリシごとに 1 つの仮想ネットワークを提供する．OS 開発者は各々の仮想ネットワークごとに異なる資源管理ポリシが実現できるように，多様な資源管理機構やアクセス管理機構をリソーススペースとしてグループ化して提供する．ユーザは提供された仮想ネットワークを適切なリソーススペースに繋ぐことで，多様な資源保護の中から適切なものを容易に実現できる．

これらの連係により，安全なネットワークでしかアクセスできないファイルなども実現可能であるし，危険なネットワークでは管理者権限を取得させないといった機能も実現することができ，より安全かつ高度な機能が提供できる．また，アプリケーションやミドルウェアの開発者がリソーススペースの切替えをプログラムすることで，複数の仮想ネットワークをわたって利用するようなアプリケーションも可能になる．

3.1 仮想ネットワーク

仮想ネットワークはルータ（ゲイトウェイ）を越えて複数のサブネットにわたる概念で，1 つの仮想ネットワークが 1 つのポリシを表す．たとえば，VPN で他地点と繋がっている仮想ネットワークであるとか，インターネットに直結した仮想ネットワークや組織内だけに閉じた仮想ネットワークなどが考えられる．仮想ネットワークは次の 3 つの概念から構成される．

仮想データリンク（Virtal Datalink：VDL）
単一の物理ネットワークセグメント内での 1 つの通信クラスを実現する．各々の VDL は VDL 識別子で区別される．この VDL 識別子の例としては，IEEE 802.1Q VLAN タグや，IEEE 802.1p 優先度タグなどがあげられる．

仮想ネットワーク（Virtual Network：VNET）
ルータで繋がる複数の物理データリンクの各々に設定されている VDL を中継することで構築される仮想的なネットワーク．通信を中継点するゲイトウェイでは，特定の通信クラスの VDL どうしの通信の中継を実現する．このとき，VDL 間の対応関係さえ保持されていれば，中継される VDL どうしが同じ VDL 識別子を持っている必要はない．

仮想インタフェース（Virtual Interface：VIF）
1 つの VNET と OS とのインタフェース．つまり VDL へのインタフェースとなる．VLAN タグや優先度タグなどの VDL 識別子に関する情報を保持し，物理インタフェースで受け取ったパケットを VDL 識別子の情報により分別したり，出力するパケットに適切な VDL 識別子を付けたりする．

仮想ネットワークの例を**図 2** に示す．この例では組織内ネットワーク上に 3 つの VNET（vnet_0，vnet_1，vnet_2）が多重化されている．各物理セグメント上には，IEEE 802.1Q VLAN などの機能により複数の VDL

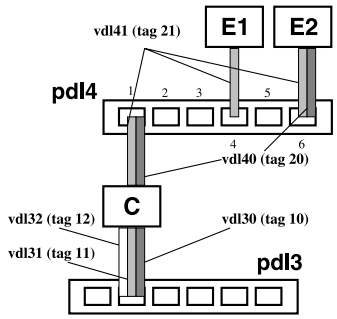

図 3 仮想ネットワーク（詳細）
Fig. 3 Virtual Networks (detailed).

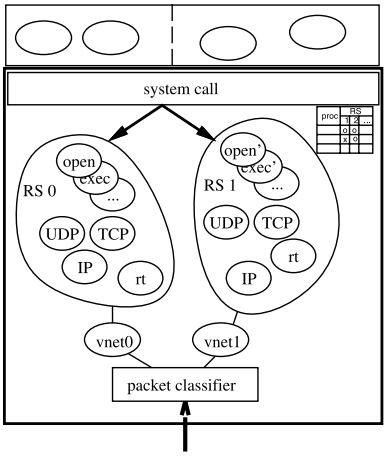

図 4 リソーススペース（RS）
Fig. 4 Resource Space (RS).

が構築されており，ルータなどが VNET ごとに独立に中継することで，組織内全体にわたる VNET を構成する．

ここでは，IEEE 802.1Q VLAN を利用してネットワークが多重化されているとして，少し具体的に説明する．図 2 の仮想ネットワークにおいて，ネットワークは 5 つの物理データリンク（pdl_0〜pdl_4）で構成されており，各物理データリンク上には，2 つから 3 つの VDL（vdl_{00}〜vdl_{02}，vdl_{10}〜vdl_{12} など）が多重化されている．そして，ルータが各々の VDL を個別に中継することで 3 つの VNET（$vnet_0$〜$vnet_2$）が構築されている．ここで，$vnet_0$ は [vdl_{00}, vdl_{10}, vdl_{20}, vdl_{30}, vdl_{40}] から，$vnet_1$ は [vdl_{01}, vdl_{11}, vdl_{21}, vdl_{31}, vdl_{41}] から，$vnet_2$ は [vdl_{02}, vdl_{12}, vdl_{22}, vdl_{32}] から，構成されているが，各々の VNET の全域で同一の VDL 識別子（ここでは IEEE 802.1Q VLAN tag id）を用意する必要はない．**図 3** にルータ C の近辺を詳細に示した．ここでは，pdl_3 のスイッチの 2 番ポートが VLAN id 10，11，12 を提供し，pdl_4 のスイッチの 1 番ポートが VLAN id 20，21 を提供するような tag 付きポートとして設定されている．ルータ C は，VLAN id 10 の vdl_{30} と VLAN id 20 の vdl_{40} を中継することで $vnet_0$ を，VLAN id 11 の vdl_{31} と VLAN id 21 の vdl_{41} を中継することで $vnet_1$ を構成している．このように中継するルータの両側で異なる識別子を利用していても問題なく，ルータが同一 VNET を構成する別セグメントの VDL 間での VDL 識別子の対応付けを正しく処理することだけが重要である．また，pdl_4 のスイッチの 4 番ポートのように特定の VLAN id を提供しないようなポート設定をすれば，特定の VNET への接続をネットワーク側で制限することもできる．たとえば図 2 でインターネットへ接続しているルータ G は，$vnet_0$ にしか繋がっていない．そのため，$vnet_0$ 以外は直接インターネットに接続することができない比較的安全なネットワークになる．ルータ B はファイアウォールを構成するフィルタリングルータで，インターネットから内部への直接のアクセスを排除することで，外部に起因し $vnet_0$ へ向かう通信を遮断している．

VDL の定義やゲイトウェイでの中継の設定によって，あるホストを組織内ネットワークにしかアクセスできないように設定することも可能である（図中 E1）．また，ある物理ネットワーク上からは外部と直接通信できないようにすることも可能である（図中 pdl_4）．

3.2 リソーススペース

VNET の提供するポリシに見合ったサービスを提供し，保護レベルごとに分離された資源管理を可能とするために，OS の内部も仮想化して多重化する必要がある．これにより，たとえばインターネットに直接接続された VNET を利用するプロセスに対しては特権実行を許さないというポリシや，ある VNET に対してのみファイルのアクセスを許すといった保護をシステムのサービスとして実現することができ，アプリケーションによる安全性の確保を補助することが可能になる．

この仮想化された OS 内部の処理群を**リソーススペース（Resource Space：RS）**と呼ぶ（**図 4**）．RS は OS 内部の処理モジュールを構成するルーチン群および，各々の処理モジュールで使われるローカルな状態を保持するデータの集合である．ここでいう処理モジュールとは IP や TCP といった通信プロトコルの入出力処理ルーチンや，システムコールの要求を実際に OS 内部で処理するシステムコール処理ルーチンである．また，各々の RS には RS 識別子**（RSID）**が付与されている．そして，RS は仮想インタフェース

VIF を通じて VNET に繋がれており，ユーザ側からの VNET の使い分けはアプリケーションが RS を切り替えることで実現する．

OS 内部では，

- 実行の単位であるプロセス
- 入出力の抽象化であるファイルエントリ（ソケットはこれに含まれる）

の 2 つが RSID を保持している．プロセスの RSID はそのプロセスのシステムコールがどの RS で処理されるかを決定するもので，この RSID を変更するとその RSID に対応する VNET が利用できる．ファイルエントリの RSID には，そのファイルエントリが生成されたとき，つまりファイルやソケットをオープンしたときのプロセスの RSID を保持している．read(), write() などの入出力処理の際には，それらのシステムコール処理ルーチンで，まず最初にこのファイルエントリの RSID とプロセスの RSID との組合せによって，プロセスの要求した入出力処理を実際に処理するかどうかを判定する．RSID の組合せごとに入出力の許可・不許可を判定する入出力判定表はネットワーク管理者や OS 開発者が定義を与え，システム構成時に静的に組み込まれる．

アプリケーションから呼び出された処理やネットワークからの入力が RS で処理される流れは，次のとおりである．アプリケーション（プロセス）がシステムコールを発行すると，ファイルエントリに関係のないシステムコールに関しては，プロセスの RSID に対応した RS のシステムコール処理ルーチンが呼ばれる．ファイルエントリに関わるシステムコールについては，ファイルエントリの保持する RSID に対応した RS で処理され，前述した入出力判定を経て入出力可能であれば，その RS に対応する VNET 経由の通信が行われる．また，ネットワークからの入力パケットは VLAN タグや優先度タグなどの VDL 識別子に応じて分別し適切な VIF に渡される．VIF には対応する RS があるので，その RS のプロトコル処理ルーチンやシステムコール処理ルーチンで以後のパケット処理が行われる．

RSID の変更はシステムコールを通じて行う．プロセスの RSID の設定には，

- set_rsid(*rsid*)

というシステムコールが用意されている．OS 内部には，入出力判定と同様に RSID 間で変更が可能かどうかを規定した変更判定表が静的に組み込まれており，変更可能な RS 間のみで RSID の変更が成功する．また，ファイルエントリの RSID の設定は，

- fcntl(*fd*, F_SETRSID, *rsid*)
- setsockopt(*s*, SOL_SOCKET, SO_RSID, *rsid*, sizeof(*rsid*))

というシステムコールを呼び出すことで行う．OS 内部では，このシステムコールが呼ばれた際に，前述のプロセスの RSID とファイルエントリの RSID の間での入出力判定表により RSID の変更を行ってよいかどうかを判定し，許可された場合のみ変更が行われる．

リソーススペースの枠組では，システムコールの処理ルーチンが RS ごとに多重化されているので，同じシステムコールでも OS 開発者が様々なシステムコール処理ルーチンを提供することで異なるポリシの処理環境が実現できる．たとえば，exec() システムコールの処理ルーチンで特権実行を無効化したものを用意することで，setuid ビットの立ったコマンドでも特権実行されないような環境が提供され，open() システムコールの処理ルーチンでファイルのアクセス権の確認ポリシを変えたものを提供することで，特定の RS 上で一部のファイルを隠すような環境が提供される．また，RSID を変更するシステムコールを無効化した RS により，プロセスの他の RS への移動を制限し，異なる用途の VNET を使うことを阻止する機能が提供される．

3.3 既存のネットワークアーキテクチャとの中継

仮想ネットワークアーキテクチャでは，IEEE 802.1Q VLAN タグのようなデータリンクの高度な機能を積極的に使うことを狙っている．最近の VPN ルータのように，VPN 接続を特定の VLAN に中継するような機能を持ったアプライアンスはこのアーキテクチャと親和性が高い．しかし，インターネットや既存のネットワークとの接続点などにおいては，データリンクの高度な機能が使えない場合も考えられる．

このような部分においては，他のポリシに従って中継をしなければならない．たとえば，特定のインタフェースから入って来たものをある VNET に送るような構成が考えられる．これは，インターネットからの通信を 1 つの仮想ネットワークに閉じ込めたり，トンネルインタフェース経由で入って来た暗号化通信をすべて 1 つの VNET に中継したりするのに使える．他の例としては，管理者がインターネットからの通信を発信元 IP アドレスを基に振り分けるような構成も考えられる．これは，外部にある自組織に関係のあるネットワークからの通信を，インターネット通信とは異なる VNET に流す場合などである．このように，入って来たパケットを特定のポリシに従って分別して適当な VIF に振り分ける処理については，VIF のパケッ

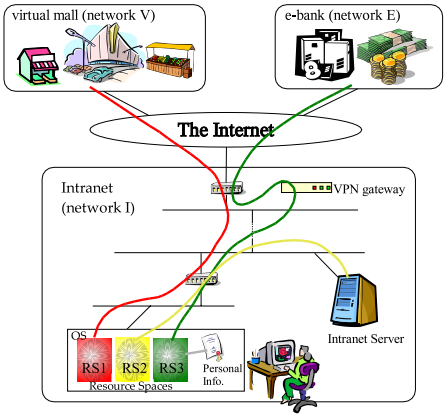

図 5 VNAP の例
Fig. 5 An example of VNAP.

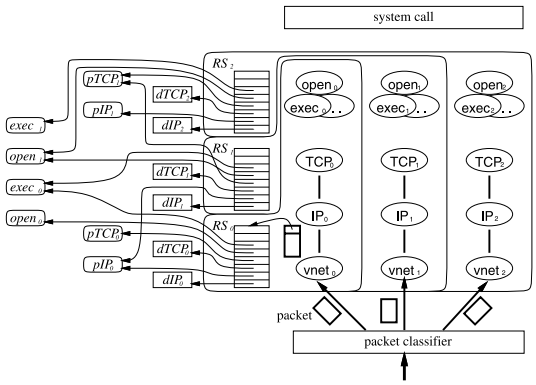

図 6 V-NAOS の実装
Fig. 6 Implementation of V-NAOS.

ト分別器（packet classifier）が，IP アドレス・プロトコル番号・トランスポート層のポート番号といった情報でパケットを分別する機能を持つことで解決する．

3.4 VNAP のもたらすもの

最後に，仮想ネットワークアーキテクチャを導入した場合にどのようなサービスが実現できるかの簡単な例を示す．**図 5** の例は VPN による安全な通信路と一般のインターネットの通信の使い分けについて示している．この例では，ネットワークは VNET1，VNET2，VNET3 の 3 つの VNET に多重化されており，VNET1 はインターネット通信，VNET2 はイントラネット内部の通信，VNET3 は VPN ゲートウェイを経由して銀行の決済ネットワークに繋がっている．ここで，ユーザは仮想ネットワーク対応のブラウザを使ってまず VNET1 上でインターネットショッピングなどの買物をするとしよう．そして，決済をする段階で決済情報だけブラウザに読み込んでから VNET3 に切り替え，決済を済ませる．この際に，決済に必要な個人情報は VNET3 に繋がった RS である RS3 からしか見えないようにしておくことで，重要な個人情報などは漏洩から守られる．

この例で使われるソフトウェアを開発する側について考えてみる．この場合，情報の漏洩を避けるためにブラウザの開発者が注意すべきなのは，VNET の切替えの際に不要な情報を消してから切り替えることだけである．このため，プログラム上で情報漏洩の可能性のある点が限定されて，ソフトウェアの安全性を高めやすいと考えられる．

4. 実　　装

仮想ネットワークアーキテクチャに対応した V-NAOS という OS を実装した．これは，FreeBSD 4.2-RELEASE のカーネルを基に，リソーススペースや仮想インタフェースの機能を実現したものである．V-NAOS の構成を**図 6** に示す．

図の右半分はプロトコルスタックおよびシステムコールの概念的な構造を，左半分の斜字体の各モジュールは，プロトコル処理モジュール（pIP_0，$pTCP_0$）やプロトコル依存データ（dIP_0，$dTCP_0$），システムコール処理ルーチン（$open_0$，$exec_0$ など）といった処理の実体を表している．各 RS はこれらのモジュールの集合として実現しており，その実体はこれらの処理モジュールやデータへのポインタの集合となる．

プロトコル処理モジュールについては，プロトコル依存のパラメータ（たとえば，IP モジュールでは IP forwarding するかどうかの変数など）を RS ごとに独立にすることにより，同一のプロトコル処理モジュールを使いながら，RS により処理を変えることも可能である．たとえば，同一の IP 処理モジュール pIP_0 を用いて，ある VNET では IP forwarding を許すが，他の VNET では許さないといったことが実現可能である．

パケット分別器はネットワークからの入力パケットのヘッダを解析し，

- VLAN ID
- ネットワーク層プロトコルのプロトコル番号（データリンクヘッダ中に存在）
- 発信元，送信先の IP アドレス
- トランスポート層プロトコルのプロトコル番号（IP ヘッダ中に存在）
- 発信元，送信先のポート番号

を抽出して，そのパケットを処理するのに適切な VIF を決定する．現状では Ethernet ドライバのハードウェア独立の部分（if_ethersubr.c）に手を加えて Ethernet（FastEthernet，Gigabit Ethernet も含む）から

の入力のみが分別可能である．このドライバのハードウェア独立部では，MAC アドレスを含む Ethernet パケット全体を入力データとして処理するので，IEEE 802.1Q VLAN tag や IEEE 802.1p priority tag に関する処理も，ハードウェア独立に実装することができた．ネットワークへの出力に関しては，ソケットの保持する RSID に応じた RS のプロトコル処理ルーチンで処理された後，その RS に対応する VIF 経由でネットワークに出力される．VIF は VLAN ID に関する情報を持っているので，ドライバのハードウェア独立部で適切な VLAN ID を埋め込んでパケットを送出する．

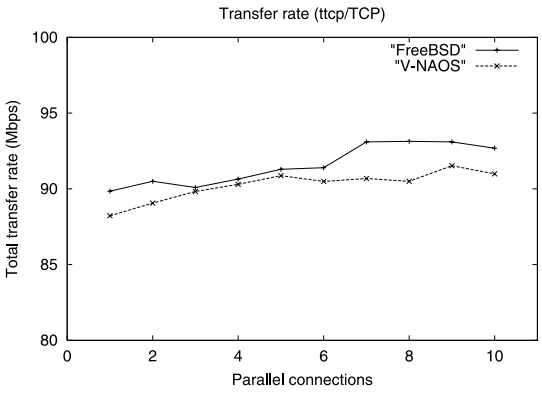

図 7 TCP のオーバヘッド
Fig. 7 Overhead on TCP.

5. 評　　価

V-NAOS の実装について，基本的な性能の側面とシステムコールの機能的な側面について述べる．まず，性能的な側面の評価の目的は仮想ネットワークアーキテクチャのオーバヘッドが実用に耐えうる程度であることを調べることで，ここでは VLAN スイッチに接続した 2 台のマシン間での TCP による転送率を調べた．計測は FastEthernet インタフェースを装着した 2 台の IBM PC-AT 互換機を住友電工 FastStream 2000LX VLAN スイッチに接続して行った．実験機の性能は，送信側は CPU が Pentium III 700 MHz でメモリを 256 MB 搭載しており，受信側は CPU が Celeron 434 MHz でメモリを 128 MB 搭載している．

測定では複数の TCP コネクションを並行に走らせ，各コネクションで 10 MB のデータを転送するようにし，転送速度を計測した．このデータ転送量は，並行な複数コネクションのセットアップ遅延により生ずる測定誤差の影響が十分に小さくなるように定めている．

この実験では，1 つの VNET 上に並行なコネクションを走らせた場合について測定した．ここでは，コネクション数を 1 から 10 に変え，並行なコネクションの転送率の和をグラフにした．10 回測定した平均の結果を**図 7** に示す．比較として FreeBSD 4.2-RELEASE での測定結果も示す．この結果，転送率の差は小さく 3% 程度のオーバヘッドであった（グラフは 80〜100 Mbps の部分だけ表示している）．この測定結果は，仮想ネットワークアーキテクチャのオーバヘッドが実用上問題にならないくらい小さいことを示すものといえる．

次に，VNAP の提供する API でどの程度の機能のアプリケーションが記述可能かという点について述べる．まず，外部に情報を公開したり，ネットワークから取得したデータをファイルに貯めたりする場合などで，1 つのコマンド（*cmd*）を完全に他の RS（RSID *r*）に移して作業してよい場合には，次のようなプログラムで実現可能である．

```
main(...)
{
    cmd にコマンドを設定
    r に RSID を設定
    set_rsid(r);
    exec(cmd)
}
```

この場合，コマンド *cmd* の扱うファイルは RS *r* に置く必要があるが，既存のプログラムがそのまま RS *r* 上で実行されるだけなので，既存のプログラム *cmd* には改造の必要がなく，ユーザからも容易に利用することができる．

ファイルエントリの RSID を変更する場合として，proxy のようなデータ転送の簡単なコードを示す．これは内部ネットワークから要求を受けて外部のネットワークからデータを獲得する場合で，外部に繋がる安全性の低い VNET 用の RS の RSID を *r* とする．

```
proxy(...)
{
    client から要求を受ける
    /* 中継ファイルを作成 */
    fd=open("file", ...);
    fcntl(fd, F_SETRSID, r);
    fork() してプロセス生成
    {
        /*子プロセス */
        set_rsid(r);
        server に要求を出す
        データを fd に保存
    }
```

```
    {
        /*親プロセス */
        wait();
        fdの頭からデータを読む
        client にデータを送る
    }
}
```

これにより，中継ファイルを介してデータだけを安全性の低いネットワークから読み込むことができる．ここで，RS rで稼働する外部と通信する部分のプログラムに buffer overflow のようなバグがあったとしても，その実行コンテキストは内部ネットワークと接続できない RS なので，内部の安全性は守られる．

これらのコードにみられるように，VNAP の枠組では，ユーザが特定のアプリケーションを明示的に異なるセキュリティレベル上で利用することは容易である．また，プログラムを改変する場合でも比較的単純なプログラムの改造で内部ネットワークの安全性を高めることができる．

6. 議　　論

従来，IP ネットワーク上で暗号化や通信品質といった異なる性質のネットワークの機能を導入する場合，それらの機能は IP アドレスの設定やルータの経路制御の設定，スイッチでのプロトコル・ポート番号の snoop（覗き見）などの手法を使って，極力 OS やアプリケーションに見えないように運用されてきた．本論文で提案した VNAP は，これらのネットワークの性質ごとに仮想ネットワークを用意して，OS・アプリケーションやユーザが積極的に使い分けることを目指している．

VNAP では，ネットワーク管理者は VNET としてネットワークのセキュリティポリシの空間を構築してユーザに提示することができるようになる．OS 開発者は，多様な OS 内部の保護機構やアクセス制御機構を RS に抽象化・グループ化して提供することができ，それらをユーザが VIF を通じて繋ぐことで，ネットワークと OS の連係による，より強固で可用性の高いネットワークを構築することができる．さらに，ミドルウェアやアプリケーションのプログラマが複数の RS を使い分けるアプリケーションを構築すれば，これらの多様な複数ポリシのネットワークを柔軟に使い分けつつ高い安全性を持ったアプリケーションを提供できる．

この VNAP の枠組を利用することにより，アプリケーションの脆弱性などに対する攻撃に対しては，攻撃されうるネットワークや実行コンテキストを分離することで対応することができる．また，メールで伝わるウィルスやマクロウィルスなどのアプリケーション層での攻撃に対しても，汚染された可能性のあるデータをアクセスする空間を用意して，そこからのファイルやネットワークに対するアクセスを制限することで被害を最小限に抑えることが可能であると考えている．これらは仮想ネットワークとリソーススペースによる空間分離により提供される機能である．また，VPN などの通信基盤の提供する安全性を高める機能も，ホスト内の資源（たとえば決済情報などの重要な情報）を，それらの安全な通信インフラのみと連係させることでより有効に活用することができる．

一方，VNAP の実装では仮想ネットワークの実現に VLAN 機能を用いているため，VLAN id が分かると他の VNET にアクセスできるようになる．しかし，通常はネットワークのスイッチの ingress/egress filter の設定でホストに利用可能な VLAN は制限することができるので，実用上は問題ないと考えている．また，VLAN を利用可能となったホストが VNET の誤った使い方や悪意のある使い方をした場合には，VNET に危険が及ぶ可能性は回避できない．しかし，この問題は既存のネットワークでもアプリケーションの設定ミスなどで同様に起こりうる問題であり，ここで提案する VNAP に固有の問題ではない．また，問題のあるノードを発見したときには，ネットワーク側でそのノードにアクセスさせるべきでない VNET を遮断することで，問題に関わる機能や情報だけを保護することができる．

次に，本研究に関連するいくつかのシステムとの比較をまとめておく．χ-kernel [7]や Scout [8]は，プロトコルスタックをモジュール化して，プロトコルのグラフ構造を自由に構成する機構を持っている．また Scout では path という概念を導入し，プロトコル処理の下位層でアプリケーション（プロセス）を判定し，必要に応じて低遅延でプロトコル処理する機構も用意されている．Eclipse [9]は，QoS 制御のために OS 内のディスクや CPU の資源の利用量を Reservation Domain という概念により抽象化する機能を持っているが，データリンクなどネットワークの機能については考慮されていない．これに対して本研究は，プロトコル処理モジュールに限らず資源を処理するモジュールまでも自由に構成することが可能で，このモジュールの構成をリソーススペースという概念によって抽象化している．さらに，データリンクの機能による仮想 LAN や優先度制御の技術を含めた多様なネットワークの仮想化の機能を提供しており，この仮想ネットワークと OS

内の処理モジュール群を組み合わせて 1 つの通信クラスを構成する機構を提供している．一方，現状では Eclipse のような資源の QoS 制御の機構は支援していない．将来的にはリソーススペースで資源の QoS 制御についても抽象化することで，本研究の提案する仮想データリンクによる多重通信クラスの機構は，アクセス制御にも QoS 制御にも応用可能な共通のプラットフォームとなると考えられる．

OS による通信クラスの実現としては，dummynet[10]や ALTQ[11]がある．これらはネットワークへの入出力の段階で，IP アドレスや TCP/UDP のポート番号で識別したフローごとに，通信帯域の上限の設定やキューイング処理の差別化により，複数の通信クラスを実現している．しかしこれらのシステムでは，データリンクの提供する機能を直接利用することはできず，また通信クラスごとにシステムコール処理などの OS 内部の処理を多様化することもできない．本研究では，データリンクの機能や通信トポロジによる通信クラスをリソーススペースの概念により OS 内部の処理と関連付けており，通信クラスに応じて多様な OS のサービスを実現できる．一方，帯域制御やキューイング処理については支援していない．これらの機能については，必要があれば仮想インタフェースに組み込むことで，リソーススペースの抽象化の概念とあわせて利用することができる．

また，質の異なる危険性のある通信に対する保護については，ネットワーク越しのプロセス追跡（process trace）[12]に関する研究がある．これは，アプリケーションゲイトウェイなどでのプロセス追跡の結果を汚染度という数字に縮退させて，TCP option としてネットワークを伝えるものである．これは一種の通信クラスを伝えていると見ることもできて，その点では本研究に関連がある．このプロセス追跡は，異常なプロセス起動に関わる通信を受信者側で判定できるという点では本研究の手法より強力な部分があるが，汚染度の情報を利用しないホストでは機能しない．一方，本研究の手法では，データリンクの VLAN tag を使った通信のクラス分けが可能なので，ゲイトウェイで判定した通信クラスを中継の通信機器の設定によってネットワーク全域に強制することが可能である．これにより，きちんとセキュリティ対策をしていないホストは危険度のある VNET に接続させないことが，通信機器側の設定で可能になる．

OS による資源管理ポリシを複数稼働させるという観点に立つと，VMware[14]のようなエミュレーション環境があげられる．これらのエミュレーション環境による複数ポリシの資源管理では，一般に各エミュレータごとに完全に独立した資源環境を構築する．本研究の手法では，システムコールの処理モジュールの挙動により，複数の処理クラス間で資源を共有したり独立させたりと自由に設定可能である．

最後に，VNAP と既存のネットワークや既存の OS の関係について述べる．VNAP では主に高機能化されたデータリンクの機能によりネットワークを仮想化することを想定している．そのため，既存のネットワークは複数の VNET のいずれか 1 つに中継する形で接続することになる．また，VIF がパケット分別の機能を持っているので，接続点で IP アドレスや TCP/UDP のポート番号を基に VNET に分別することも可能である．次に，リソーススペースのない既存の OS が搭載されたホストを VNAP に接続することを考える．この場合，スイッチ側の VLAN の制御をユーザに開放するならば，直接所属する VLAN を切り替えることで VNET を切り替えることも可能である．しかし，多くの場合安全性の観点からスイッチ側の制御は開放しないので，OS は複数ある VNET のうちのいずれか 1 つに接続されることになる．

7. おわりに

本論文では，仮想ネットワークアーキテクチャによるイントラネット全般にわたる保護機構である VNAP を提案した．VNAP では，通信の範囲や用途，また暗号化などの特性ごとに仮想的なネットワーク（VNET）を用意し，OS 内部の機能群をそれらの VNET と対応付けることにより，イントラネット全般にわたり特定の安全性のレベルを保持した保護ゾーンを構築することができる．これにより，アプリケーションの設計や実装から通信相手の安全性に関する判定を切り離すことが可能になり，ユーザは豊富な知識を持つネットワーク管理者の提供する複数の安全性の設定を自由かつ容易に使うことができるようになる．

プロトタイプシステムの実装とその評価から，VNAP を導入しても実用上十分な性能が得られることが確認された．ファイルのアクセスについては，現在はシステムコールのファイルオープンやアクセス権チェックの機構を変えることで多様な機構を提供しているが，将来的にはネットワーク以外のディスクやファイルシステムなどの資源を仮想化（論理化）してリソーススペースに対応づけることを検討している．

参考文献

1）高木浩光，関口智嗣，大蒔和仁：クロスサイト

スクリプティング攻撃に対する電子商取引サイトの脆弱さの実態とその対策，情報処理学会第 4 回コンピュータセキュリティシンポジウム (2001).
2) CERT Advisory CA-2000-02: Malicious HTML Tags Embedded in Client Web Requests. http://www.cert.org/advisories/CA-2000-02.html
3) Stevens, W.R.: *TCP/IP Illustrated, the protocols*, Addison-Wesley (1994).
4) Touch, J.: Dynamic Internet Overlay Deployment and Management Using the X-Bone, *Computer Networks*, pp.117–135 (2001).
5) IEEE Standard 802.1Q-1998, IEEE Standards for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks.
6) IEEE Standard 802.1D-1998, Information technology — Telecommunications and information exchange between systems — Local and metropolitan area networks — Common specifications — Part3: Media Access Control (MAC) Bridges.
7) Hutchinscm, N.C. and Peterson, L.L.: The x-kernel: An Architecture for Implementing Network Protocols, *IEEE Trans. Softw. Eng.*, Vol.17, No.1, pp.64–76 (1991).
8) Mosberger, D. and Peterson, L.L.: Making Paths Explicit in the Scout Operating System, *Operating Systems Design and Implementation*, pp.153–167 (1996).
9) Bruno, J., Gabber, E., Özden, B. and Silberschatz, A.: The Eclipse Operating System: Providing Quality of Service via Reservation Domains, *USENIX 1998 Annual Technical Conference*, pp.235–246 (1998).
10) Rizzo, L.: Dummynet and Forward Error Correction, *Freenix 98* (1998).
11) Cho, K.: A Framework for Alternate Queueing: Towards Traffic Management by PC-UNIX Based Routers, *USENIX 1998 Annual Technical Conference* (1998).
12) 光来健一，千葉 滋，益田隆司：プロセスの依存関係に基づく分散システムのセキュリティ機構，日本ソフトウェア科学会 SPA 2000 (2000).
13) 光来健一，廣津登志夫，佐藤孝治，明石 修，菅原俊治，千葉 滋：複数のオーバレイネットワークを制御するためのプライベートなネットワーク環境，情報処理学会コンピュータシステムシンポジウム論文集，No.18, pp.75–82 (2002).
14) VMware: http://www.vmware.com/.

(平成 15 年 2 月 3 日受付)
(平成 15 年 5 月 13 日採録)

廣津登志夫

1967 年生．1995 年慶應義塾大学大学院理工学研究科計算機科学専攻博士課程修了．同年，日本電信電話株式会社入社．基礎研究所を経て現在，未来ねっと研究所に所属．以来，マルチメディアシステム，分散システム，OS，ネットワーク等の研究に従事．博士（工学）．日本ソフトウェア科学会，ACM，IEEE-CS 各会員．

福田 健介（正会員）

1999 年慶應義塾大学理工学研究科（計算機科学専攻）後期博士課程修了．同年，日本電信電話株式会社入社．現在，未来ねっと研究所に所属．以来，インターネットトラフィックの解析およびモデル化，ネットワーク構造の解析等の研究に従事．この間，2002 年ボストン大学訪問研究員．博士（工学）．ACM 会員．

光来 健一（正会員）

1975 年生．2002 年東京大学大学院理学系研究科情報科学専攻博士課程修了．同年，日本電信電話株式会社入社．現在，未来ねっと研究所に所属．以来，オペレーティングシステム，ネットワークの研究に従事．博士（理学）．日本ソフトウェア科学会，ACM 各会員．

明石 修（正会員）

1964 年生．1987 年東京工業大学理学部情報科学科卒業．1989 年同大学院理工学研究科情報科学専攻修士課程修了．同年，日本電信電話株式会社入社．以来，分散システム，コンピュータネットワーク，マルチエージェントシステム等の研究に従事．現在，NTT 未来ねっと研究所主任研究員．博士（理学）．ACM，日本ソフトウェア科学会各会員．

佐藤　孝治（正会員）

1967年生．1989年慶應義塾大学理工学部数理科学科卒業．1991年同大学院理工学研究科計算機科学専攻修士課程修了．同年，日本電信電話株式会社入社．現在，同社未来ねっと研究所に所属．分散システム，マルチメディアシステム等に興味を持つ．日本ソフトウェア科学会会員．

菅原　俊治（正会員）

1982年早稲田大学大学院理工学研究科（数学専攻）修士課程修了．同年，日本電信電話公社入社（武蔵野電気通信研究所基礎研究部）．以来，知識表現，学習，分散人工知能，マルチエージェントシステム，インターネット等の研究に従事．1992〜1993年，マサチューセッツ大学アムハースト校客員研究員．現在，NTT未来ねっと研究所主幹研究員．博士（工学）．日本ソフトウェア科学会，IEEE，ACM各会員．