修　　士　　論　　文

柔軟なアクセス制御を実現するための
ユーザ主導仮想閉域ネットワーク

指導教官　　青山 友紀 教授
森川 博之 助教授

東京大学大学院新領域創成科学研究科
基盤情報学専攻

氏 名　　47-36317 飛岡　良明

提 出 日　　平成 17 年 1 月 31 日

目 次

図 目 次

表 目 次

第1章　序論

1.1 研究の背景と目的

計算機の小型化や低廉化，及びインターネットのグローバル規模での広帯域化や常時接続化を背景に，ユーザ個人が PC や携帯電話などの複数の端末を所持するようになった．このことは，昨今流行の言葉となったユビキタス環境 [1] を実現する一因となっている．身の回りのあらゆるものに計算能力が組み込まれ，それらがインターネットを介してコミュニケーションできる環境が浸透するに伴い，ユーザは多様な端末やデバイスだけでなく，様々なサービス（アプリケーション）を個人で保有し，利用するようになる．

このように，インターネットがグローバル規模で目覚しい発展を遂げた要因は，インターネットプロトコル（IP） の設計指針としてシンプルさとロバストさを優先させ，ネットワークそのものにおいて必要不可欠となるメカニズムだけを組み合わせた [2] ことにあると言われる．そのため，現在のインターネットアーキテクチャは，任意の IP アドレスへの到達性の効率化を第一指針として構築されており，ネットワークトポロジに強く依存したかたちで管理されている．

一方，ユーザにとって関心があるのは IP アドレスへの到達性ではなく，実端末や実際に提供されるサービスである．ネットワークを利用する際のユーザの要求は,“ ネットワークにいつでもどこでも繋がり，全てのものに平等にアクセスしたい ”ということではなく,“ 自分が利用できるサービスをネットワークを通じて使用したい ”ということである．ユーザにとってみれば，自分が利用できるサービスや端末が存在しなければネットワークに繋がることに意味はない．また，サービスを提供する側にしても，ネットワークに接続されている全ての端末に対して平等にサービスを提供するのではなく，特定のユーザや端末だけにサービスを提供したい．

しかし，現状のインターネットには管理者主導の枠組みしか存在せず，ユーザは管理者が管理する IP アドレスなどのネットワーク情報しか用いることができない．これにより，インターネットはユーザにとって利便性の低いものとなってしまっている．このことは，ユーザの要求に十分に応えるファイアウォールを設定することが困難であることからも見てとれる．ファイアウォールを用いて実現したいことは，あるポリシ（例：同一ユーザが所有している）を満たす端末からのみ，提供するサービスへの通信を認めるということである．ところが，そのようなユーザポリシを表現する枠組みが存在せず，管理者が管理するネットワーク情報を用いるしかないところに様々な不都合が生じる．たとえば，DHCP [3] を用いて IP アドレスを管理している場合，ユーザの取得する IP アドレスが一意ではないため，他の端末は要求を実現するファイアウォールの記述を行うことができない．

このような不都合さは，管理者による到達性の効率化と，ユーザによるインターネットの利便性向上という異なるポリシを管理者主導の枠組みで実現しようとすることに原因がある．よって，ユーザの利便性を向上させるためには，ユーザと管理者の要求を切り離し，管理者が構築している既存のインターネット上にユーザ主導の枠組みを再構築することが求められる．

このようなユーザ主導の枠組みを再構築することによる，ユーザの利便性の向上を図るためにさまざまな研究がなされており，それらについて簡単に説明を行う．

Region Project [4] では，大規模，広分散，かつヘテロなネットワークのためのアーキテクチャデザインで鍵となるのは region と呼ぶグループ分け，区分けを行う機構にあるとし，ネットワーク上のエンティティは必ず何かしらの region に属するものとしている．region はその内部のエンティティが同一の技術や管理ポリシを持つものとして定義され，その区分には関しては Autonomous System（AS）やサブネットなどの到達性を確保するためものだけでなくより汎用的なものを目指しており，例として “課金モデル”，“セキュリティポリシ” などを挙げている．Region Project では，グローバル規模で汎用的にネットワークを区切ることにより IP ネットワークをよりユーザが使いやすく管理することを目的としている．

新しい名前空間の適用によって，既存のネットワークトポロジに強く依存する IP 空間をよりユーザが使いやすく整理する研究として，Layered Naming Architecture [5]，Internet Indirection Infrastructure [6] などが挙げられる．双方とも名前の解決には分散ハッシュテーブル [7, 8, 9] を用い，ネットワークトポロジに依存しない名前を用いてのモビリティの解決などを提案している．特に，Layered Naming Architecture では，ネーミングに着目し，‘User-level Descriptor’，‘Service Identifier（SID）’，‘Endpoint Identifier（EID）’，‘IP Address’ という 4 層に階層化することによって IP にかかる負荷を減らすなど，興味深い提案がなされている．EID と IP Address 間の名前解決と，解決後の通信方法に関しては，実際に DOA というシステムを実装している [10]．DOA システムにおいては，ユーザはロケーションに依存した IP アドレスを用いて通信相手を指定するのではなく，端末に固有の EID を用いて通信を行う．これにより，ネットワークロケーションの移動に伴う IP アドレスの変化に関係することなく，ユーザは固定した識別子をもって通信を行うことができる．また 現状 NAT [11] の存在などにより IP 層で直接通信を行うことができない端末に対しても，EID を用いることにより階層化を壊すことなく通信可能であることを示している．

また，Virtual Internet [12] は，インターネットをトンネリングによって仮想化することによって，利用形態にあわせてネットワークトポロジを定義する．この定義されたネットワークは，実際のインターネット上にオーバレイとして実現される．このために，仮想ホスト，仮想ルータ，仮想リンクといった概念を導入している．Virtual Internet では，実際のインターネットにおける到達性の確保と，ユーザがネットワークを利用する際の利便性の確保を分離を目指している．

上記の研究以外にも，ユーザの利便性向上のために，物理ネットワークトポロジに依存しないルーティング [13, 14]，ネットワーク上に存在するサービスの連携 [15, 16, 17] などの提案もなされている．

前述したユーザ指向の枠組みにおいて，本研究では“ サービスには提供領域が存在する ” ということに着目する．ユーザがインターネット上にサービスを公開するとき，全ての IP アドレスを平等に考え，何処から，誰からでもサービスへのアクセスを許すのではなく，端末の所有者や物理位置などのメタ情報に依存した特定のグループだけに対してサービスを提供したいという要求は強い．具体的にユーザの関心があるメタ情報としては，“ 誰が所有する端末であるのか ”，“ 物理的にどの場

所に存在するのか ”,“ サービスへの課金を済ませた端末か否か ”，などが挙げられる．ユーザがこれらの情報を用いてネットワークリソースを管理することで，同一部屋内部に存在する端末だけにサービス利用を許す，家族が所有する端末からはファイルへのアクセスを許す，などといった柔軟なネットワークの利用が可能となる．

このような点に鑑み，本研究では，共通のメタ情報によって括られるサービス提供領域をアーキテクチャとして提供し，その領域を用いて柔軟なサービスへのアクセス制御を可能にする枠組みを構築することを目的としている．そのために，共通のメタ情報によって括られるサービス提供領域である仮想閉域ネットワーク MyNetSpace（MNS）を提案する．

MyNetSpace システムの特徴として以下の 2 点が挙げられる．1 つ目は，共通のポリシを持つ端末だけで構成されるネットワークを構築する点である．2 つ目は，このポリシによって構築されたネットワークが外部のネットワークとの間で自由な通信を許さない閉域ネットワークである点である．このようなシステム上で，端末は認証を通して単一，あるいは複数の MNS に同時に参加し，専用の仮想ネットワークインタフェース（VNI）を介して MNS 内部に存在する通信を許可された端末と通信することができる．これらより，ユーザは特定の MNS 用の VNI を通しての通信と，提供するサービスを結びつけることにより，メタ情報に依存したサービス提供領域を実現できる．

このようなネットワークトポロジに依存しない仮想ネットワークの構築に関して，Virtual Private Network（VPN）を提供する技術が既存のものとして挙げられる [18, 19, 20, 21, 22]．しかし，これらの技術には複数の VPN に端末が参加するとき，個々の VPN に対してアクセス制御の記述ができないことや，アドレス空間が衝突する可能性があるなどの問題が存在する．MNS では，VNI に割り当てられるアドレスをインターネット上でのルーティングには用いず，端末内部で識別子としてのみ機能させることにより，これらの問題を現状のインターネットとの親和性を高く保ったまま解決することを目指す．

1.2 本論文の構成

本論文は，以下の各章によって構成される

第 1 章 序論

第 2 章 柔軟なアクセス制御

第 3 章 MyNetSpace を用いたアクセス制御

第 4 章 MyNetSpace システムアーキテクチャ

第 5 章 実装と動作検証

第 6 章 結論

まず第 2 章では，特定の条件を充たす端末だけへのサービスの提供に関する想定シナリオを示し，既存のアクセス手法では想定シナリオを十分に実現することはできないことを述べる．

第 3 章では，仮想ネットワークの性質を使うことにより，既存のアクセス制御では実現できなかった柔軟なアクセス制御が可能になるが，既存の VPN を構築する技術では，柔軟なアクセス制御を実現するのにいくつかの問題があることを説明する．そして，そのような問題を解決できる，ユーザ主導仮想閉域ネットワーク “MyNetSpace” を提案する．次いで，MyNetSpace に要求される機能を挙げ，MyNetSpace を用いてのアクセス制御の概要を示す．

第 4 章では，第 3 章で提案した MyNetSpace のシステムアーキテクチャを示す．まず，MyNetSpace を実現する上で必要な機能とその実現手法を述べ，システムの動作概要，構成要素を説明する．

第 5 章では，Windows XP SP1 上に行った MyNetSpace システムの初期実装について説明し，動作検証を示す．

最後に第 6 章で本研究の成果についてまとめ，今後の展望を記す．

第2章　柔軟なアクセス制御

2.1 はじめに

今日，非常に多数の端末がインターネット接続性を持ち，物理的に離れた場所に有るサービスを容易に使用できるようになった．そのような環境において，本研究では“ サービスには提供領域が存在する ”ということに着目し，ユーザの柔軟なサービスへのアクセス制御を目的とする．本章では，2.2 節で柔軟なアクセス制御の必要性を述べ，2.3 節で想定しているシナリオを挙げる. そして，2.4 節では既存技術をもって要求するアクセス制御を実現することが困難であることを示す .

2.2 柔軟なアクセス制御

現状のインターネットアーキテクチャにおいては，ユーザがインターネットに接続している端末でサービス（アプリケーション）を公開すると，インターネット上の膨大な数の端末からそのサービスへアクセスが可能になる．ウェブに代表される全世界のユーザに公開したいサービスでは，このような何処からでも，誰からでもアクセスできることが重要であった．

しかし，インターネットを用いてより多様なサービスの提供を考えると，“自分が所有してる端末”，“同じ部屋に存在している端末”，“サービスへの課金を済ましている端末” など，特定の条件を充たす端末だけに提供したいサービスの実現が重要になる．例えば，個人データへのアクセスは，非常に限定された端末だけに許可したいサービスの典型といえる．

現在のインターネットが，任意の IP アドレスの 2 点間で対等な到達性を実現するために設計されている [2, 23] ため，インターネットでのサービス提供は前者の形になっている．そこで後者のような限られた端末だけにサービスを提供するためには，条件を充たさない端末からのアクセスを拒否するアクセス制御技術が必要である．

既存のアクセス制御の技術では，ユーザが柔軟にアクセス制御を実現することは難しい．本研究では，このような限られた端末だけへのサービス提供を実現するのに必要になる，ユーザが柔軟にアクセス制御できる枠組みの構築を目的としている．

2.3 想定シナリオ

ユーザの要求として，以下のようなシナリオを想定している．

良明は研究室からノートパソコンを渡されており，研究室や自宅，またホットスポットなどよりインターネットに接続している．そのような状況下で良明は以下のようなことを実現したいと考えている．

a. 良明は複数の端末を所持しており，用途によって使い分けている．その際にデータが複数の端末に分散してしまい，必要なときに利用できない状態は非常に困る．従って，自分が所持して

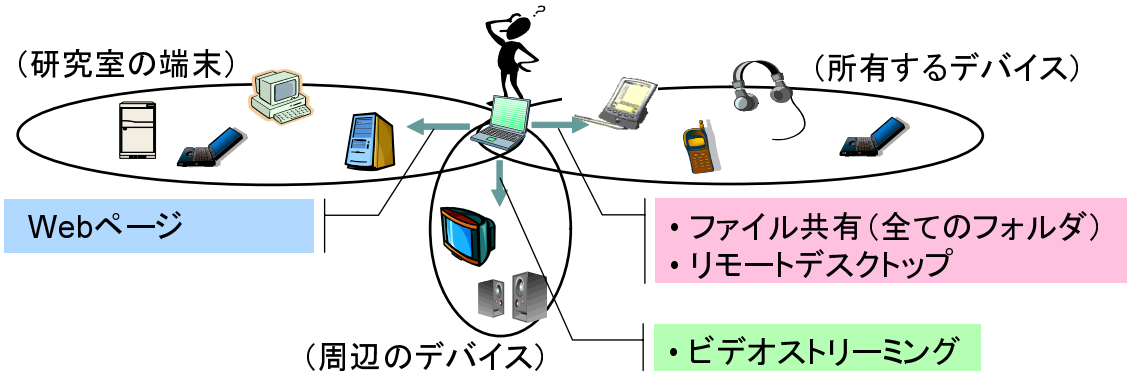

図 2.1: アクセス制御のシナリオ図

いる端末からは，ノートパソコンのデータへ常にフルアクセスできる状況にしておきたい．

b. 良明は Web サービスとして，研究室の空調を操作できるサービスを開発し，ノートパソコン上で実行している．研究室にいない人間が自由に空調を操作できては困るので，研究室内部にある端末からのみ Web サービスへアクセスできるようにしておきたい．

c. 良明は，ノートパソコンを用いてビデオストリーミングサービスを行いたいが，その際にデータへのアクセスは身近にある端末だけ許すようにしたい．つまり，大学にいるときは大学のスピーカ，ディスプレイからはアクセス可能であるが，家へ戻った際には，家のスピーカ，ディスプレイからのみアクセス可能としたい．

ここで重要なのは，全てのシナリオにおいてノートパソコンが提供するサービスは，ユーザが要求する領域へしか提供されないということである．シナリオ a においては “ユーザが所有している端末”，シナリオ b では “研究室内部にある端末”，シナリオ c では “ユーザの身近にある端末” というのがサービス提供領域になり，その領域外部にある端末からのアクセスは制限されなければならない．例えばシナリオ b において，同僚が端末を使用しながら研究室外部へと出て行った場合，研究室内部にある端末というサービス提供領域からは外れてしまうので，Web サービスへのアクセスを拒否できなければならない．

2.4 既存のアクセス制御技術

既存のサービスアクセス制御手法は，アプリケーションレイヤで行う手法とネットワークレイヤで行う手法の 2 つに大別できる．以下で両者とも想定する環境に十分に応えるものではないことを述べる．

2.4.1 アプリケーションレイヤでのアクセス制御

アプリケーションレイヤのアクセス制御とは，サービスを提供するアプリケーションにユーザ認証機構を組み込み，サービスを受ける（通信を行う）度にパスワード入力を要求し，正しいパスワードを持っているユーザにのみサービスを許すというシステムである．アプリケーションレイヤで全て認証を行えば，サービス単位で細かく認証を行うことができ，上記のシナリオを充たすことは可能であろう．

しかし，この手法では，サービスを提供する全てのアプリケーションで認証を実装しなければならない．これは，アプリケーション作成を非常に煩雑なものにしてしまう．また，認証機構が組み込まれていない大多数の既存のアプリケーションでは，アクセス制御を実現できない．さらに，同じサービス提供領域（例：研究室内部（b））に対して提供したいと考えるサービスが複数ある場合，本質的に同じ認証を複数のアプリケーションで実装する必要があり，認証の冗長化を招く．以上のような点に鑑み，アプリケーションレイヤで全てのアクセス制御を行うことは好ましくない．

2.4.2 ネットワークレイヤでのアクセス制御

ネットワークレイヤでのアクセス制御技術は，パケット送信者の IP アドレスなどのネットワーク情報を用いて，通信そのものの到達許否によってアクセス制御するものであり，現在 NAT [11] やファイアウォールを構成する技術として広く用いられている．この技術では，既存のアプリケーションに変更を加えることなくアクセス制御を実現できる点が，前述のアプリケーションレイヤでのアクセス制御技術と異なる．

しかし，この手法はユーザにとって柔軟性に欠ける．その理由は，IP アドレスはユーザが自由に決めることができるものではなく，ネットワーク管理者によりグローバルスケールでの到達性を効率よく達成するために決められるという点にある．これにより，IP アドレスは物理ネットワークトポロジに強く依存したものとなっている．一方，一般的にユーザが要求するサービスの提供範囲（アクセスを許可する対象）と物理ネットワークトポロジは一致しないことが多い．このために，ユーザは要求するアクセス制御を柔軟に実現することが難しい．

具体的には以下のようにネットワーク管理者とユーザとの要求が異なり，ユーザの要求するアクセス制御が実現できない．シナリオ a では自らの所持する端末というサービス提供領域を設けたが，ユーザが DHCP サービス [3] を利用している場合は，ユーザが所有している端末を表す IP アドレスは固定されたものではなくなってしまう．同様に，シナリオ b においては，研究室内部に存在する端末というサービス領域を設けた．その際に，研究室のネットワーク環境が無線 LAN を採用している場合，一つの端末が特定の IP アドレスを設定したまま部屋の出入りをすることが可能になる．ユーザがアクセス制御を行う際に注目しているのは，あくまでも部屋の内部にあるのか，ないのかという点であるが，これは IP アドレスから把握できない情報となってしまい，ユーザは要求するア

クセス制御を実現することができない．このように,“ サービス提供領域に存在する端末か否か ”を IP アドレスを用いて表現することは難しく，サービスを提供する端末で要求するアクセス制御を記述することは困難である．

以上のような議論をもって，ネットワークレイヤを用いてのアクセス制御でも，十分にユーザの要求に応えることができないと言える．

2.5 おわりに

本章では,“ サービスには提供領域が存在する ”というユーザの要求に応えるためには，ユーザが柔軟にアクセス制御を行うことが必要であると述べた．そのうえで想定するシナリオを挙げ，それを実現するためには既存のアクセス制御技術には問題があることを述べた．第 3 章では，これらの問題点を解決するために提案するユーザ主導仮想閉域ネットワーク MyNetSpace（MNS）について述べる．

第3章　MyNetSpaceを用いたアクセス制御

3.1 はじめに

第 2 章では，既存のアクセス制御技術では想定するシナリオを十分に実現することが困難であると述べた．本研究ではそれらの問題を解決するた仮想ネットワークの技術を用いることを提案する．本章では，3.2 節で仮想ネットワークの性質を用いたアクセス制御について述べる．次に，3.3 節で，既存の仮想ネットワーク構築技術である VPN における問題点を述べ，3.4 節で，その問題点を解決できるユーザ主導の “仮想閉域ネットワーク”MyNetSpace（MNS）の提案を行う．そして，3.5 節で，MyNetSpace へ要求される機能を説明した後，3.6 節で MyNetSpace を用いたアクセス制御実現の概要を示す．

3.2 仮想ネットワークを用いたアクセス制御

前章で，既存のアクセス制御の技術では，ユーザが要求する柔軟なアクセス制御を実現することができないと述べた．仮想ネットワークを用いることにより，そのようなアクセス制御を実現することができる．本節では，仮想ネットワークの特徴を挙げ，その特徴を利用してのアクセス制御の実現について述べる．

3.2.1 仮想ネットワークの特徴

仮想ネットワークの特徴として以下の点が上げられる．

- ネットワークトポロジに非依存
- ネットワーク内部に存在するエンティティの管理

まず前者であるが，実際の IP ネットワークは到達性を重視するために，ネットワークトポロジに強く依存した形で構築されている．一方仮想ネットワークは，到達性が確保された上で仮想的に構築されるネットワークであるので，トポロジに依存せずにユーザが自由に構築できる．

次に，後者の意味するところは，ある条件を充たす仮想ネットワークを構築する際には，その内部には条件に合致する端末しか存在しないことを保証できるということである．つまり，端末や，IP アドレスに関係せず，内部に存在するエンティティがポリシに反すれば仮想ネットワークから排除し，逆にポリシを充たせば参加させるといようなエンティティ管理ができるということである．以降仮想ネットワークの構成要素を “エンティティ”，仮想ネットワークに参加するための条件を “ユーザポリシ” と呼ぶ．

上記の 2 点はネットワークを実ネットワークトポロジから切り離し，仮想化したことにより実現されたことで，仮想ネットワークの大きな特徴といえる．特に，ネットワーク内部に存在するエン

ティティの管理に関しては，初期に Darpa がインターネット（IP）を構築する際に必要であることは認めながら，到達性を優先するために犠牲にした目標であった [2].

3.2.2　仮想ネットワークを用いてのアクセス制御

前述の仮想ネットワークの特徴を用いることにより，ユーザの要求する柔軟なアクセス制御が実現できる．ユーザはサービスを提供する端末を特定の仮想ネットワークへ参加させる．その仮想ネットワーク内部には，ユーザの要求するポリシに合致する端末しか存在しないという点は，システムとして保証されている．2.3 節のシナリオ b の例だと，“研究室内部にある端末” というユーザポリシをもって構築された仮想ネットワークが存在する．その仮想ネットワーク内部には，研究室外部に存在する端末が存在しないことは，システムが保証する．つまり，端末が部屋の内部から外部へと移動した際には，他の端末へ影響を与えることなく仮想ネットワークから除外され，また部屋の内部に入ってくると仮想ネットワークに参加する．この特徴を利用して，ユーザはその仮想ネットワーク内部の他のエンティティからのサービスへのアクセスを許可するよう記述することで，ユーザの要求するアクセス制御が実現できる．

3.3　既存の仮想ネットワーク技術の問題点

既存の VPN [20] の技術を用いて仮想ネットワークを構築することは可能である．しかし，VPN は離れたネットワークを仮想的に接続し，アクセスできなかったサービスに対してアクセスを可能にすることを主目的として研究，開発されている技術であり，本研究とは目的が異なる [18, 19]．そのため，既存の VPN の技術を用いて我々が想定するアクセス制御を実現するには，“端末が複数の VPN に参加するときのアドレス空間の衝突” や，“特定の VPN に対してだけにサービスを提供することが困難” などの問題がある．双方ともに，仮想ネットワーク用のアドレス（仮想 IP アドレス）が，ネットワーク上での端末の識別子として用いられているおり，ユーザが自由に設定，変更できないところに原因がある．以降で各々の問題について説明を行う．

アドレス空間衝突

既存の VPN において，仮想 IP アドレスが VPN 内部でパケットを転送する宛先として用いられてるため，端末が同じアドレス空間を使用している複数の VPN に参加する場合，端末はどちらの VPN に対してパケットを送信すればよいのか判断できないという問題が生じる．VPN のアドレス空間を決定し，端末に仮想 IP アドレスを配るのは VPN 管理者であり，複数の VPN が存在する場合に，VPN 管理者間で通信を行いアドレス空間の割り当てを行うことは困難である．よって，現状の VPN においては，端末が複数の VPN に参加することは難しい．

特定の VPN に対するサービス提供

既存の VPN において特定の仮想ネットワークからの通信に対してサービスを提供することは困難である．この問題はインターネットで通信するアプリケーションで一般的に用いられるソケット API の仕様による．ソケット API では，アプリケーション内部で，ソケットを受信する IP アドレスにバインドする必要がある．端末の IP アドレスがパケットを転送する際の宛先として用いられる場合，アドレス空間を含めてユーザが自由に決定，変更することは不可能である．したがって，通常サービス提供アプリケーションでは，自分宛に送信されたパケット全てを受信するように $INET\_ANY$ をバインドする．既存の VPN においても仮想 IP アドレスが VPN 内部でのルーティングに用いられるため，サービス提供アプリケーションは $INET\_ANY$ をバインドする．これにより，端末が複数の VPN に参加している場合，どの仮想 IP アドレス宛に来たパケットであってもアプリケーションが受信してしまうので，特定の仮想ネットワークに対してサービス提供を行うことは困難である．

3.4 MyNetSpace の提案

前述のような考察をもって，仮想ネットワークを用いてアクセス制御を行うことを主張し，既存の VPN における問題点を解決するために，ユーザポリシを充たす端末から構成される仮想閉域ネットワーク MyNetSpace（MNS）を提案する．MyNetSpace システムの特徴として以下の 2 点が挙げられる．1 つ目は，共通のポリシを持つ端末だけで構成されるネットワークを構築する点である．2 つ目は，このポリシによって構築されたネットワークが外部のネットワークとの間で自由な通信を許さない閉域ネットワークである点である．次節で，MNS に要求される機能について述べる．

3.5 MyNetSpace システムへの要求機能

3.5.1 ユーザポリシによるエンティティ管理

MNS を用いてのアクセス制御を実現するためには，MNS 内部のエンティティには，ユーザが定めるポリシに反する端末は存在しないことが必要である．よって，MNS を実現するためには，統一されたポリシをもって内部エンティティを管理することが要求される．たとえば，“同一部屋内部に存在する端末” というユーザポリシをもって MNS を構築する際には，MNS 内部に部屋の外にある端末が存在してはならない．よって，たとえ IP アドレスが変わることなく端末が部屋の外部へと出て行った場合にも，その端末が MNS から排除されることが要求される．具体的な処理としては，端末が MNS に参加する際に，その端末が MNS の定めるポリシに適っているか充たしているかの認証を行うこと，ポリシに反するエンティティが存在する際に MNS から脱退させることが必要である．

3.5.2 閉域ネットワークの実現

MNS 内部にはユーザのポリシに適った端末しか存在しないということが保証されたとしても，内部のエンティティがポリシに合致しない MNS 外部の端末と自由に通信ができるとアクセス制御は実現できない．そこで，MNS は内部と外部の自由な通信を許可しない“ 閉域ネットワーク ”であることが要求される．

3.5.3 端末の仮想化

ユーザが MNS の中でサービスを展開するためには，端末を MNS のエンティティとして参加させなければならない．このとき，端末は複数の MNS に属するということが十分に考えられるため，端末を仮想的に扱うことが要求される．すなわち，1 つの端末が複数の仮想端末を作成し，各々が異なった MNS のエンティティとして扱われることになる．PC を例に考えてみると，特定の部屋に存在するという面も，あるユーザの管理下にあるという面もあり，各々の側面で端末が果たすべき役割

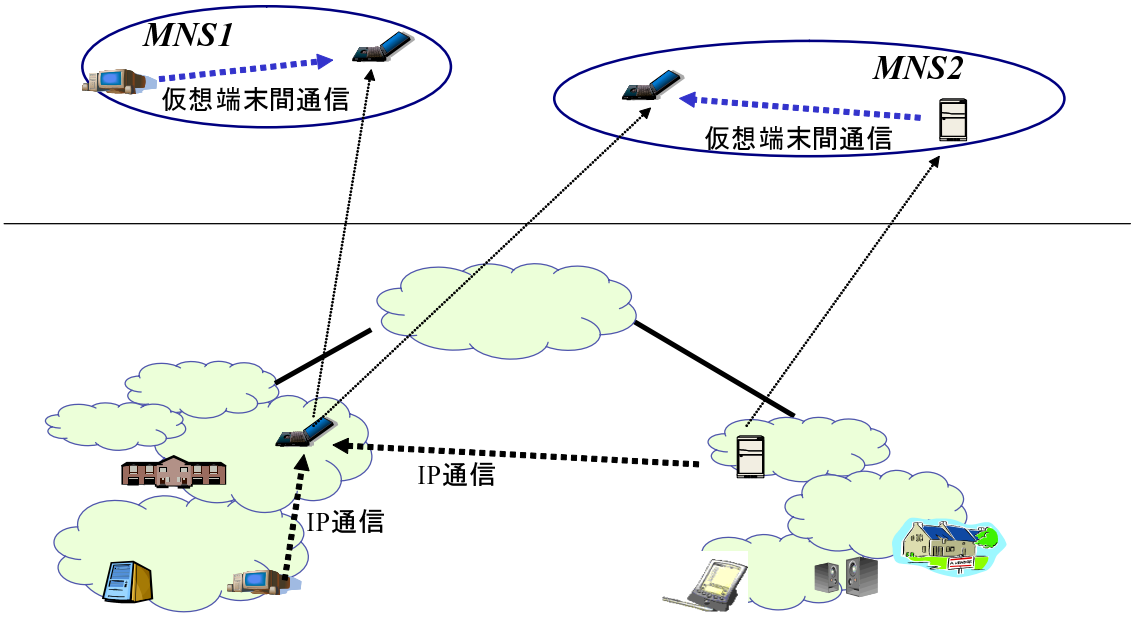

図 3.1: 通信の仮想化概念図

（提供するサービス）は異なる．異なったポリシによって構築される MNS に対して各々に仮想端末を作成することで，ユーザポリシに適ったサービス提供が実現できる．

3.5.4 通信の仮想化

MNS 内での通信は仮想端末間の通信として認識されることが要求される．このことは，図 3.1 で示すように，端末が複数の MNS に属しているとき，物理的に同一端末宛の通信であるとしても，それぞれの通信がいずれの MNS 内部で行われているのかを区別し，異なった仮想端末宛の通信であると認識されるということである．これにより，サービスを提供する端末は受信したパケットがどの仮想端末宛てであるかを判断することが可能になり，仮想端末単位でのサービス提供・享受が実現される．また，3.3 節で挙げた問題点を解決するために，仮想端末間の通信を実現するときに，通信相手の識別子として送信先の仮想端末を用いてはならない．

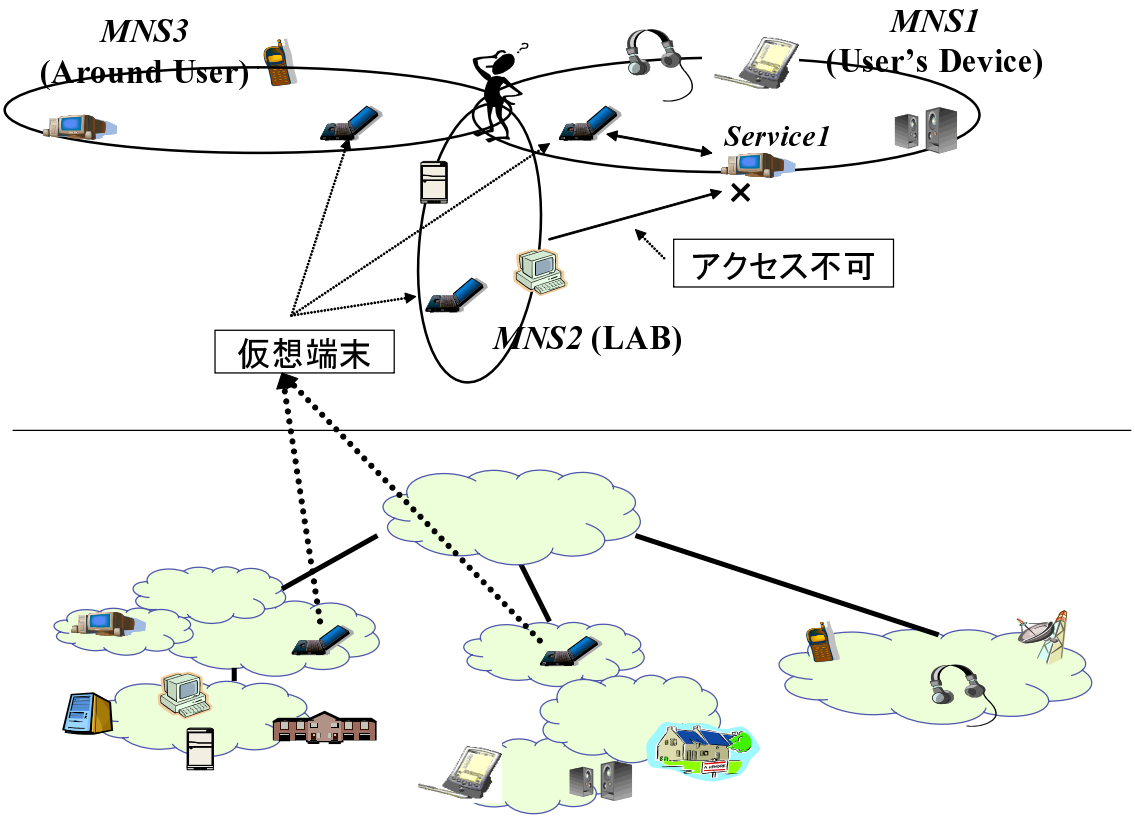

図 3.2: MyNetSpace 構築の概略図

3.6 MyNetSpace を用いてのアクセス制御概略

3.5 節で述べた機能をもった，“ユーザ主導閉域仮想ネットワーク” である “MyNetSpace” 構築の概略図を図 3.2 に示す．現在の物理ネットワークトポロジに依存したインターネットを下側に，MNS を構築した様子を上側に表現している．図 3.2 では，“ユーザが所有する端末” というユーザポリシによって管理される $MNS1$，“研究室内部にある端末” というユーザポリシで管理される $MNS2$，“ユーザの近隣にある端末” というユーザポリシで管理される $MNS3$ と 3 個の MNS が構築されている．

まず，端末は仮想端末という形で複数の MNS に同時に参加している．図 3.2 ではモバイル端末であるノートパソコンが同時に 3 個の MNS に参加している．そして，$MNS1$ 内部では，ユーザが所有するデスクトップ PC が，自分の所有している端末だけにサービス（$Service1$）を提供している．これは，$MNS1$ 内部の仮想端末だけからこのサービスを提供する仮想端末へのアクセスを許可し，$MNS1$ 外部からはアクセスを拒否する “閉域ネットワーク” の性質によって実現される．また，IP

アドレスはネットワークトポロジに依存するので，ノート PC が自宅にあるときと大学で使用されているときとで当然異なる．しかし，“ユーザが所有する端末” というユーザポリシを充たすかが，$MNS1$ に仮想端末として参加できるかできないかの判断基準となるので，このユーザの所有するノート PC は自宅にあろうが，大学にあろうが $MNS1$ へ仮想端末を作成し，$Service1$ を利用することができる．一方 $MNS2$ は，“研究室内部にある端末” というユーザポリシで構築されているので，ノート PC が研究室にあるときには $MNS2$ へ仮想端末を作成することができるが，無線 LAN を使いながら研究室から出ると，IP アドレスに変更がなくても仮想端末を作成することができなくなる．

3.7 おわりに

本章では，まず仮想ネットワークを用いることによって既存のアクセス制御では実現できなかった柔軟なアクセス制御が可能になることを述べた．一方，既存の仮想ネットワークを構築する VPN の技術は，アクセス制御を目的として研究開発されているものではないので，柔軟なアクセス制御を実現するには問題があることを述べた．それらの問題を解決するために，“ユーザ主導仮想閉域ネットワーク” である MyNetSpace を提案し，MyNetSpace を構築するうえで要求される機能を説明した．

第4章　MyNetSpaceシステムアーキテクチャ

4.1 はじめに

本章では，第 3 章で述べた要求される機能を実現し，柔軟なアクセス制御を実現するための MyNetSpace システムを構築するためのシステムアーキテクチャについて述べる．システムを設計するにあたり重要視したのは，“既存のインターネットとの高い親和性” をもったシステムであるということである．特に，既存のアプリケーションに変更を加えることなく導入できるシステムを目指した．

4.2 節では，機能要求を充たすシステムを構築するために実現しなければならない動作とそれを実現するための設計について述る．そして，4.3 節で MyNetSpace システムの概要を述べ，4.4 節でそれを実現するために必要となる構成要素の詳細を記す．

4.2 システム実現手法

4.2.1 MNS への参加・脱退

MyNetSpace システムは 3.6 節で述べたように，サービスを提供・享受する端末が特定の MNS へ参加し，その内部でアクセス許す端末同士の閉じられた通信を行うことによりアクセス制御を実現するシステムである．これを実現するためには，MNS への “参加・脱退” を表現する機構が必要である．

MNS を“ 閉域ネットワーク ”であると定義したことにより，端末が MNS に参加するということは，MNS 内部の仮想端末と通信可能になるということである．既存の仮想ネットワーク技術である VPN などの手法では，仮想ネットワーク上でのデータ転送をつかさどるサーバが存在し，VPN 内部の通信はそのサーバを経由して行われる．そこで，VPN サーバの認証を通り，他のエンティティと正しく通信を行えることをもって，端末の VPN への “参加” を表現し，逆にサーバによって通信を制限することによって “脱退” を実現している．この手法の欠点としては，全ての通信が VPN サーバを通らなくてはならないという点が挙げられる．これにより内部に存在するエンティティの増大に伴い，VPN サーバの負荷が増大し，通信の速度が低下してしまうなどの問題がある．

この問題を解決するために，MNS システムでは通信の許否の判断をサーバで行うのではく各々の端末で行う．具体的な仕組みとしては，MNS への参加が認証されると 4.2.4 節で述べる MNS に正しく参加したことを証明する $MNSID$ を端末に配布し，正しい $MNSID$ をもっている端末だけが，MNS に存在する他の仮想端末と通信できることとする．つまり，通信に利用できる正しい $MNSID$ の配布を持って，端末の MNS への “参加” を実現する．同様に，MNS からの “脱退” に関しては，配布してある $MNSID$ を用いては他の仮想端末と通信できなくことで実現する．この手法により，通信が一点に集中することを回避でき，内部に存在する仮想端末が増加にも耐えうるシステムとなる．

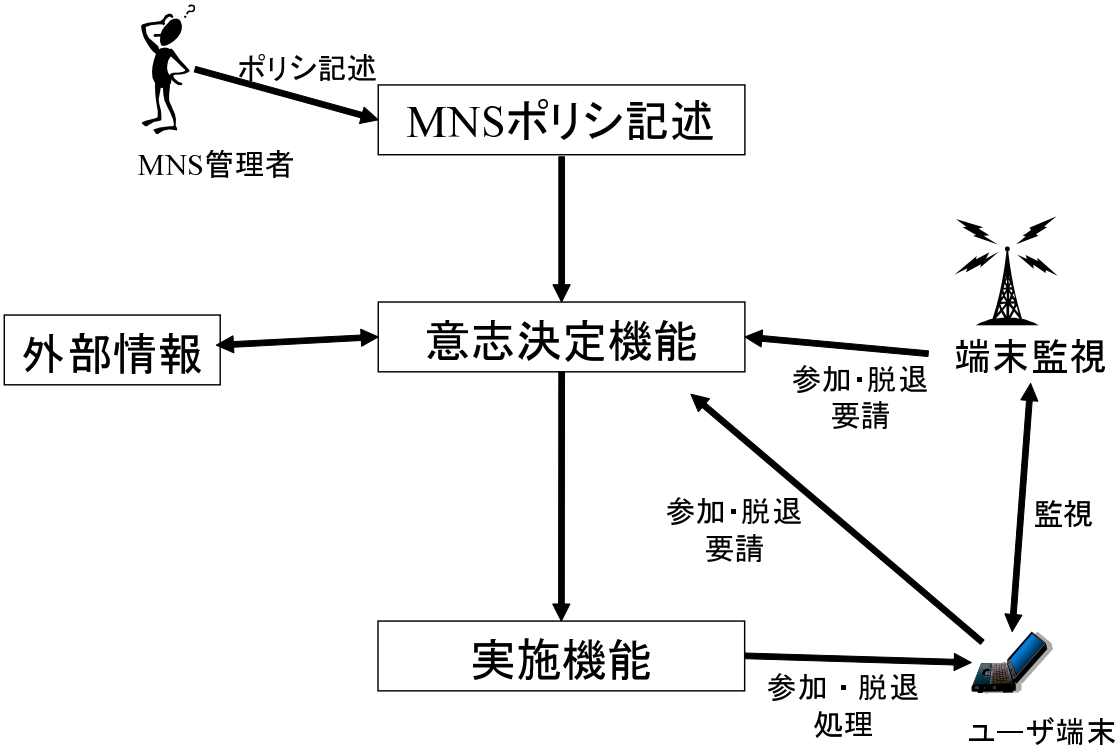

図 4.1: 内部仮想端末管理機構

4.2.2　内部仮想端末のポリシ管理

3.5.1 節で述べたように，MNS を構築するためには，内部にはユーザポリシに適った仮想端末しか存在しないことをシステムとして保証しなければならない．4.2.1 節で，MNS への参加・脱退の手法は述べた．よって，ここではどのようにユーザポリシを充たす端末のみを “参加” させ，反する端末が存在する場合は “脱退” させるのかを述べる．

端末の参加・脱退は，MNS の性質によって能動的・受動的の 2 種類に大別される．能動的な場合とは，ある端末側が自ら MNS への参加・脱退を要求するものであり，“自分の所有する端末” などというユーザポリシによって MNS を構築し，ユーザ自らが MNS への参加・脱退を行うものである．一方，MNS には端末自らが参加・脱退の要求を出すのではなく，“特定の部屋に存在する端末” などのような MNS のおいて，端末が定められた条件を充たすと MNS より参加要請を受け，逆に条件を充たさなくなると強制的に MNS から脱退させられるような受動的なものも考えられる．

図 4.1 を用いて，これら双方を実現するためのシステムを説明する [24]．まず，MNS 管理者が内部の仮想端末が充たすべきユーザポリシを記述する “MNS ポリシ記述部” がある．次に端末が記述

されたポリシを充たしているかの判断を行う “意思決定機能” が存在する．意思決定をするタイミングは，ユーザ端末自らが能動的に参加・脱退要請を送信してきた場合と，“端末監視部” が決められたポリシを充たした（反した）端末を発見して，参加・脱退要請を “意思決定機能” へ通信してきた場合である．また，意思決定を行う際には，外部データベースなどの “外部情報” を参照する場合も考えられる．そして，“意思決定機能” が参加・脱退どちらかの判断を行うと，その結果が “実施機能” へと伝えられ，参加・脱退処理が行われる．

4.2.3 仮想端末の作成・削除

端末内部に対して仮想端末を構成するための手法の一つとして，ライブラリの中で新しいアプリケーションプログラムインタフェース（API：Application Program Interface）を定義することが考えられる．この API は，特定の MNS を指定することができ，その MNS 内部の通信を送受信する．相手の実端末までの通信は IP を用いて行うことを想定しているので，既存のソケット API を参照し機能を追加する形で新しい API を作成することになる．そして，その API を通してネットワークサービスを提供するようにアプリケーションを作成する．しかし，この手法では既存のアプリケーションに対して変更を必要とするので，本研究の設計指針に反する．

そこで，MNS では，既存の VPN を実現する技術と同様に，仮想ネットワークインタフェース（VNI:Virtual Network Interface）を作成する．そして，端末内部で自分が参加している MNS を明示的にバインドすることにより仮想端末を実現する．この手法では新しい API を用いる必要がなく，既存のソケット API を用いることができ，アプリケーションに変更を加える必要がない．

以上より，認証を通して MNS への参加が決定すると，端末はその MNS 内部の通信用の VNI を作成する．そして，その MNS 内部へ提供すべきサービスをその VNI で待ち受けることによって “仮想端末の作成” を実現する．同様に “仮想端末の削除” は作成してある VNI を削除することによって実現される．

4.2.4 仮想端末間通信

仮想端末間通信の様子を図 4.2 に示す．MNS3 内部で仮想端末 A から仮想端末 B へと通信を行う場合，端末 A の MNS3 に参加するときに作成した VNI3 から，端末 B の VNI3 へと送信される．この通信は，MNS2 内部の通信（端末 A の VNI2 と端末 B の VNI2 間の通信）と IP 層では同一の通信であるが，仮想端末間通信では別の通信と認識される．

4.2.1 節で述べたように，端末は参加している MNS ごとに正しく参加していることを証明する $MNSID$ を所持している．そこで，仮想端末間の通信を，$MNSID$ を用いてどの MNS 上での通信であるかを示すことにより実現する．

詳細なパケットの扱いを図 4.3 を用いて述べる．送信端末は，送信元を通信する仮想端末の VNI

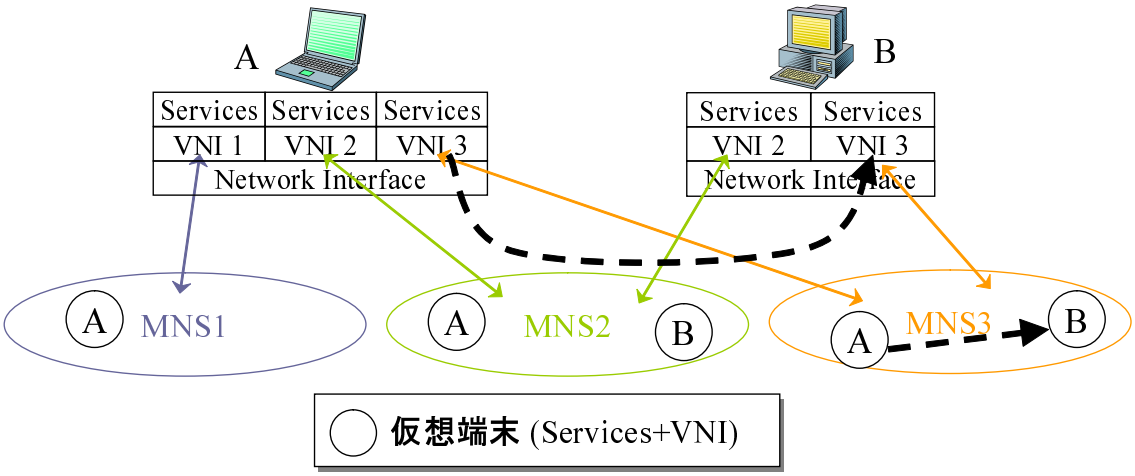

図 4.2: 仮想端末間通信

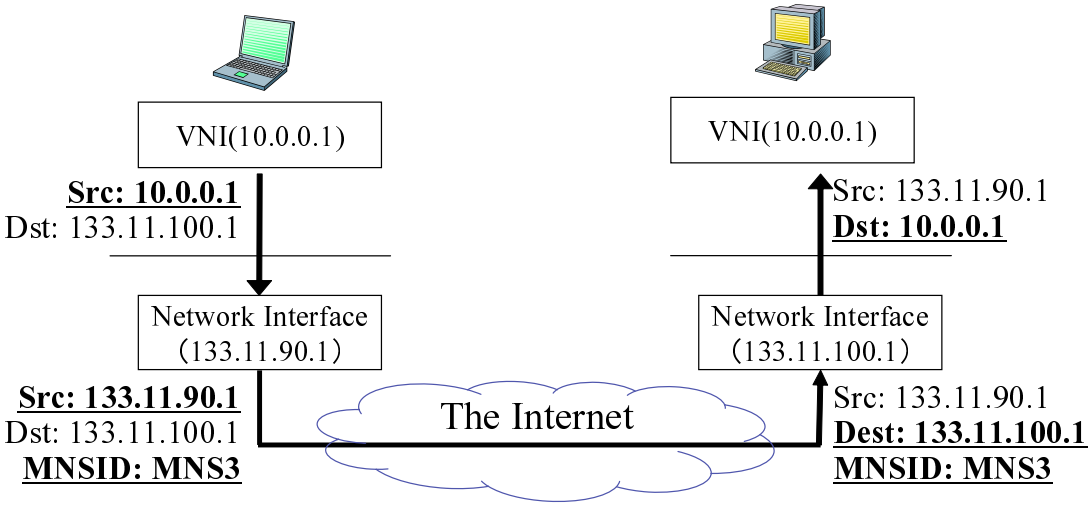

図 4.3: 仮想端末間パケット

のアドレス（仮想 IP アドレス），宛先を通信相手が実際にインターネットで用いているアドレス（実 IP アドレス）に設定する．VNI を通過したパケットがインターネットへ送信される前に，送信元が仮想 IP アドレスから自端末の実 IP アドレスへと変換され，$MNSID$ が付加される．すなわち，インターネット上では，通信を行う双方の実 IP アドレスと MNSID がパケットに記載されている．受信端末は $MNSID$ を解析し，宛先を該当する仮想 IP アドレスに変更してから $MNSID$ を削除する．正しく MNS に参加していることを示す $MNSID$ が記載されていないパケットは VNI へと転送されない．これによって，3.5 節で述べた “閉域ネットワーク” であることが実現される．

また，この手法の特徴は仮想 IP アドレスが端末内部のみで用いられ，インターネット上でのルーティングには利用されないということである．つまり，仮想 IP アドレスはインターネット上での位置を表すものではなく，純粋に端末内部で特定の仮想端末を示す識別子として用いられている．3.3 節で述べたように，既存の仮想ネットワーク技術を用いてアクセス制御を行うときの問題になるのは，“仮想 IP アドレスがネットワーク上で端末の識別子として用いられている” ことであった．そのために，ユーザが自由に設定・変更することが難しく，それにより “アドレス空間の衝突”，“特定の仮想ネットワークへのサービス提供が困難” などの問題が生じていた．本手法では，仮想 IP アドレスは端末内部だけでしか用いられなく，インターネット上で用いられるのは到達性の保障された端末の実 IP アドレスであるので，複数の MNS に参加したとしてもアドレス空間が衝突してパケットの転送ができなくなるという問題は生じない．また，仮想 IP アドレスが端末内部でしか用いられないということは，ユーザが自由に設定することができるので，ある特定の MNS 内部の通信のみを待ち受けるようにアプリケーション内部で記述することが可能になり，全てのパケットを受信するという問題を解決できる．

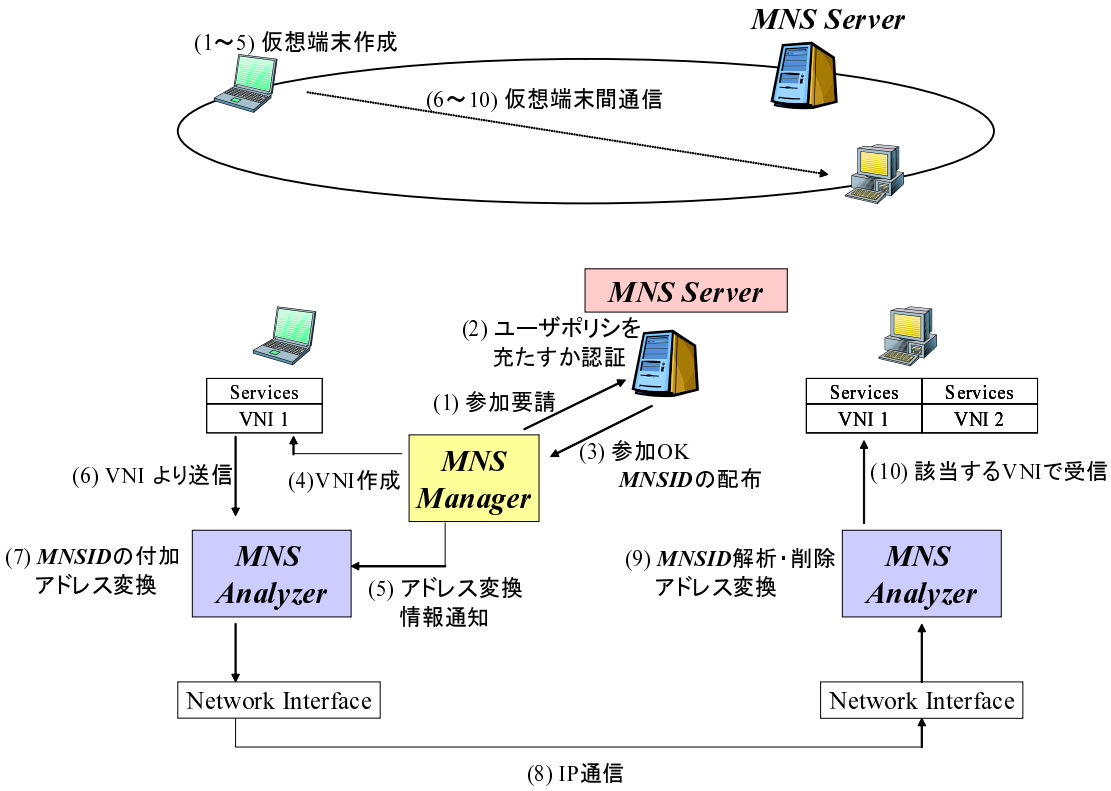

図 4.4: MNS システム概要

4.3　システム概要

MNS システムの概要と，ユーザ端末が能動的に MNS へ参加し，仮想端末間通信を実現するまでの動作を図 4.4 に示す.

構成要素

MNS システムは以下の構成要素からなる.

1. MNS Server
2. MNS Manager
3. MNS Analyzer

まず，MNS Server は MNS ごとに一つ存在し，MNS 内部の仮想端末が従うべきポリシの管理や，端末の MNS への参加・脱退要請の受信，それに対応する処理，MSN 内部に存在することを証明する $MNSID$ の作成・配布など，MNS の構築・維持を行う．

次に，MNS Manager は個々の端末に一つ存在する．MNS Manager は MNS Server との間で，参加・脱退要請や $MNSID$ の取得などの制御メッセージを通信する．また，MNS への参加に伴い対応する VNI の作成，VNI に設定する仮想 IP アドレスの設定，自端末が提供するサービスの VNI での待ち受けによる” 仮想端末の作成” を実現する．さらに，“仮想端末間通信” を実現するために必要なアドレス変換を行うための情報を次に述べる MNS Analyzer へ通知する．

最後に，MNS Analyzer も端末に一つ存在し，“仮想端末間通信” を実現するために必要なアドレス変換，$MNSID$ の付加・削除を行う．MNS Analyzer は，MNS Manager から受信したアドレス変換のための情報をもとに，4.2.4 節で述べたようにアドレス変換を行う．各々の構成要素については次節で詳細に述べる．

動作概要

ここでは，端末が能動的に MNS へ参加要請を行い，認証を経て内部に存在する他の仮想端末との間で仮想端末間の通信を行いサービスを享受するようすを述べる．この動作は，仮想世界から見ると

i. MNS へ新しい仮想端末の参加

ii. 仮想端末間の通信

の二つの動作になる．各々について実際にどのような処理が行われているのかを記す．

i. MNS へ新しい仮想端末の参加

(1) 端末は MNS Manager を通して MNS Server へと参加要請メッセージの送信

(2) MNS Server は，参加要請メッセージ送信端末を自らが定めるユーザポリシにより認証

(3) MNS Server は認証後，参加 OK のメッセージとともに，参加を証明する $MNSID$ を端末の MNS Manager へ送信

(4) MNS Manager は $MNSID$ を受信後，参加した MNS 用に VNI の作成，仮想 IP アドレスの設定

(5) MNS Manager は MNS Analyzer へアドレス変換に必要な情報の送信

ii. 仮想端末間の通信

(6) 送信端末は仮想端末用の VNI よりデータの送信

(7) 送信端末の MNS Analzyer が MNS Manager より受信したアドレス変換情報を元にアドレス変換，$MNSID$ の付加

(8) 受信端末のネットワークインタフェースに向けて IP 通信によるデータ送信

(9) 受信端末の MNS Analyzer が $MNSID$ を解析し，アドレス変換，$MNSID$ の削除

(10) 受信端末が該当する MNS に対応する VNI を用いてデータの受信

以上のような動作をもって，仮想端末間通信の通信が実現される．

脱退要請を行うときは MNS Manager が仮想端末間通信を用いて MNS Server への脱退要請を行う．正しく脱退要請が認められると，MNS Manager は VNI の削除，MNS Analyzer に該当するアドレス変換情報を削除を命じることにより仮想端末の脱退を実現する．

上記の処理は全てが端末側が能動的に MNS への参加・脱退を行う場合であった．受動的な処理となる場合には，4.2.2 節で述べたような “端末監視” を行う機能が MNS Server に対して参加・脱退を行うべき端末の情報を通知し，それに伴い MNS Server 側より端末の MNS Manager に対して参加・脱退要請を出すよう通知する．

4.4 システム構成要素

本節では，前節で述べたシステム構成要素について詳細に述べる．

4.4.1 MNS Server

MNS Server の役割は，図 4.5 で示すように

- ポリシ管理部
- 処理部
- 通信部

の 3 種に大別される．以下で，各々の役割について詳細を記す．

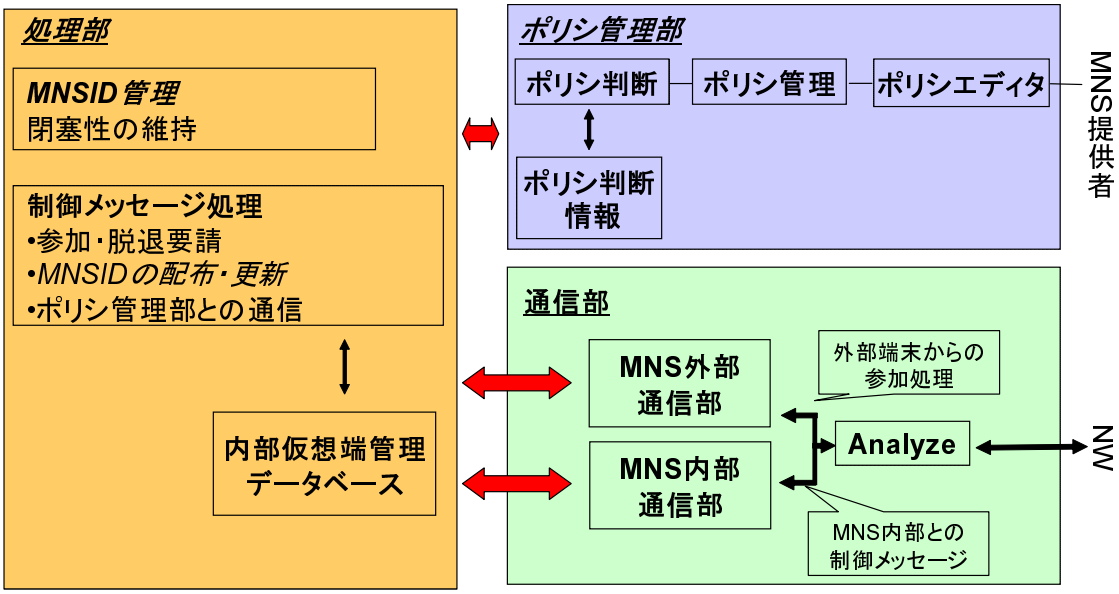

図 4.5: MNS Server コンポーネント図

ポリシ管理部

ポリシ管理部は，4.2.2 節で述べたように，MNS 管理者が定めるユーザポリシを記述する “ポリシエディタ”，そのポリシを保存する “ポリシ管理”，端末がユーザポリシを充たしているか否かの判断を行う “ポリシ判断”，さらにポリシ判断を行うときに参照すべき “ポリシ判断情報” からなる．“ポリシ判断” で決定された内容は，“処理部” へ送信される．

ここで，“ポリシ判断情報” に関しては，MNS Server 内部で所持することも可能だが，別の端末が持ち，それを参照する形態も十分に考えられる．たとえば “サービスに課金を済ました端末” というユーザポリシであれば，実際にサービスを提供している端末，組織が課金を済ました端末のデータベースをもっており，MNS Server はこれを “ポリシ判断情報” として参照することになるであろう．さらに，ユーザポリシに関しては，2.3 節で例を挙げただけでなく，多種多様なものが考えられ，また，ユーザが自由に追加できなくては柔軟なシステムということはできない．それらの多様なポリシを充たすかの判断を下すための “ポリシ判断情報” には，端末の物理位置監視や，ユーザ情報データベースの利用など様々な機構が必要になると考えられる．そのため MNS Server のシステムとしては，特定のポリシに特化した認証を組み込むのではなく，なるべく汎用的に判断できるフレームワークを構築し，各々の実際の認証に関しては別個組み込むことができる形態とする．

処理部

処理部は MNS を構築，維持するために必要な処理を行う．

まず，“ポリシ管理部” と通信を行い，端末の参加・脱退処理を行う．動作例として参加要請を受信した場合を挙げると，参加要請を受けると送信元の端末情報を “ポリシ管理部” へ送信する．そして “ポリシ管理部” より参加を許すか否かの判断を受信し，参加を許す場合には $MNSID$ を配布する．

また $MNSID$ を作成，閉塞性の維持も処理部で行う．ここで，$MNSID$ の閉塞性について少し説明する．今まで述べてきたように，$MNSID$ は MNS に正しく参加していることを証明するものである．そこで，もし外部の端末が $MNSID$ を所持できると，内部に存在するように偽証し，MNS 内部にしか提供されないサービスを利用できてしまう．従って MNS に正しく参加している端末以外が利用可能な $MNSID$ を所持することは許されなく，この性質を “$MNSID$ の閉塞性” と呼ぶ．$MNSID$ の閉塞性を実現するために重要になるのは，一度正しく MNS に参加していた端末が脱退したときの配布済みの $MNSID$ への対応である．過去に MNS に正しく参加してた端末は，その時は当然ながら正しく利用できる $MNSID$ を所持していた．脱退した後もその $MNSID$ が利用できるようであると，外部端末が利用可能な $MNSID$ を所持していることになり $MNSID$ の閉塞性が破られてしまう．そこで，“処理部” では閉塞性を保つための処理を行う．具体的には，脱退する端末が存在すると新しい $MNSID$ を生成し，内部に残ってい端末全てに $MNSID$ の更新を要求するなどの手法が考えられる．

また，MNS Server は常に MNS に参加している仮想端末のデータベース “内部仮想端末管理データベース” を保持している．このデータベースを利用し，MNS Server より様々な制御用のメッセージを送信することができる．そのような制御メッセージの例としては前述した $MNSID$ の再配布や，ポリシを充たさなくなった端末に対する脱退勧告などが挙げられる．

通信部

MNS Server の制御メッセージなどを扱うサービスは，MNS 内部にだけ提供されるべきものである．そこで，MSN Server は仮想端末として MNS へ参加し，仮想端末間の通信として制御メッセージの通信を行う．一方，外部端末からの参加要請を受ける必要もあり，このメッセージはインターネット全体から受けなければならない．これより，MNS Server の “通信部” には MNS 内部での通信を行うための” 内部通信部” と，外部との通信を行う “外部通信部” の 2 つからなる．

4.4.2 MNS Manager

MNS Manager は個々の端末に一つ存在し，前述の MNS Server と，次に述べる MNS Manager と，端末を利用しているユーザとをつなぐ役目を担う．MNS Manager の役目について図 4.6 を用いて説明する．

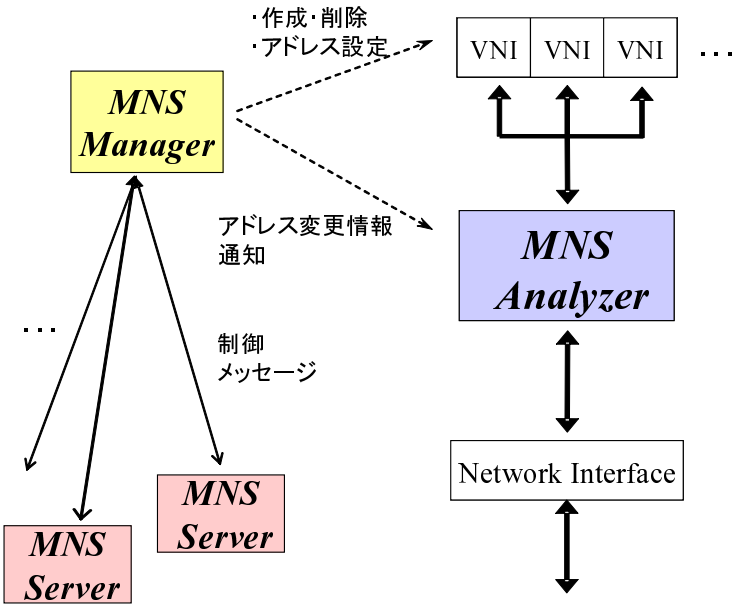

図 4.6: MNS Manager

まず，MNS Manager は，MNS Server との間で制御メッセージのやり取りを行う．制御メッセージの例としては，参加・脱退要請，それに伴う $MNSID$ の受信，変更要請などが挙げられる．そのとき，端末は複数の MNS に同時に参加可能であるので，各々の MNS に対応した MNS Server を管理するためのデータベースを所持しておかなければならない．

次に，MNS Manager は MNS へ参加したときに，その MNS 内部の通信用の VNI を作成し，その仮想 IP アドレスを設定する．そして設定した仮想 IP アドレスと，対応する MNS Server より受信した $MNSID$ の対応表を作成する．これによって，参加している MNS 用の仮想端末が実現される．同様に MNS から脱退した場合には，VNI を削除し，対応表より該当する仮想 IP アドレスを削除する．

最後に，上記の対応表は以下で説明する MNS Analyzer がアドレス変換をするときに必要になるので，対応表の情報を MNS Analzyer へと通知する．

4.4.3 MNS Analyzer

MNS Analyzer は仮想端末間の通信を実現するために必要なアドレス変換，$MNSID$ の付加・削除を行う機能を有する．仮想端末間での通信を実現するために図 4.3 で述べたようなアドレス変換が必要である．MNS Analyzer の役割は図 4.7 のようになる．

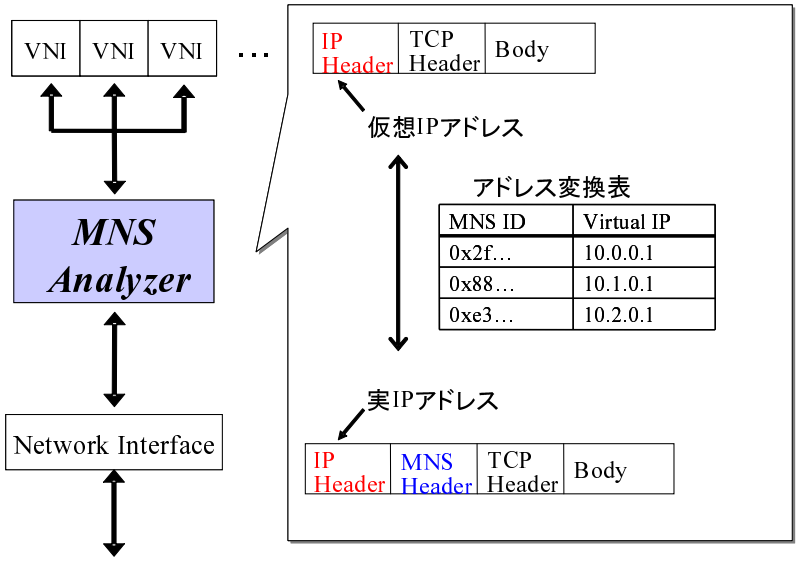

図 4.7: MNS Analyzer

まずは，MNS Manager よりアドレス変換に必要となるアドレス変換表を構成する仮想 IP アドレスと $MNSID$ を受信する．このアドレス変換表を用いて，送信端末では送信元アドレスを仮想 IP アドレスから実 IP アドレスへと変換し，$MNSID$ の付加．逆に受信端末では宛先アドレスを実 IP アドレスから仮想 IP アドレスへと変換し，$MNSID$ を削除することが MNS Analyzer が果たさなければならない役割である．ここで，MNS Analyzer は，受信したパケットの $MNSID$ をアドレス変換表より探し，該当する仮想 IP アドレスを判断するのだが，自分の所持しているアドレス変換表に受信した $MNSID$ が記載されていない場合が考えられる．アドレス変換表は，前述したように MNS Manager が $MNSID$ と該当する仮想 IP アドレスとの対応表の情報を受信し作成するものであり，アドレス変換表に記載されていない $MNSID$ とは，MNS Manager が MNS Server から受信した正規の $MNSID$ では無いことを意味する．そこで，その通信は自分が所属している MNS 外部からの通信であると認識し，VNI へデータを転送することなく MNS Analzyer の時点でパケットを破棄する．これにより，VNI 間，仮想端末間の通信は正しい $MNSID$ をもったもの，つまり同じ MNS に属している仮想端末間でしか確立することができなく，また仮想端末で特別なアクセス制御機構を必要としないシステムが構築される．

MNS Analyzer を設計するときに考えなければならない点は，$MNSID$ をパケットのどこに挿入するかである．本研究では，仮想端末間の通信に TCP を利用可能とすることに着目した．既存のインターネットのサービスの多くは，TCP を利用しており，信頼性の高い通信を実現するための非常に優れた手法である．よって，既存のアプリケーションを利用するためには，仮想端末間の通信にお

いて TCP を利用可能とすることが必要である．これを実現するためには，アドレスを変換する鍵となる $MNSID$ は，トランスポート層よりも下位層で処理することが要求される．一方パケットの転送に関しては既存の IP 層を利用するので，$MNSID$ の処理は IP 層とトランスポート層との間に挿入する．これにより，仮想端末間での通信は “自端末の対応する仮想 IP アドレスと通信先端末の実 IP アドレス” という IP アドレスのセットをもって TCP が管理される．これにより，一つの端末が複数の仮想端末を所持する場合，同じポート番号を用いて全ての仮想端末が TCP を張ることができるようになり，既存のアプリケーションが十分に利用できる柔軟なシステムが構築できる．

4.5 おわりに

本章ではまず，前章で述べた MyNetSpace システムを構築する際に必要となる動作として “MNS への参加・脱退”，“内部仮想端末管理”，“仮想端末の作成・削除”，“仮想端末間通信” の 4 種を挙げ，各々について実現手法の概要を示した．次に，MyNetSpace システムの概要と，それを利用する際の動作概要を示し，システムの構成要素である “MNS Server”，“MNS Manager”，“MNS Analyzer” について，各々役割を説明しシステム設計を行った．

第5章　実装と動作検証

5.1 はじめに

本章では第 4 章で述べた MyNetSpace システムの初期実装について述べ，動作検証を報告する．まず，5.2 節で，仮想端末間通信を実現するために必要な実装を，5.3 節で内部仮想端末管理を行うため制御プロトコルの詳細を述る．そして 5.4 節で，実装したシステムの結果を報告し動作検証とする．システムの実装は全て Windows XP SP1 上で行った．

5.2 仮想端末間通信

5.2.1 動作概要

図 5.1 は “MNS3” に参加している仮想端末間の通信において，MNS Analyzer が行うべき動作概要を示したものである．双方の端末の MNS Manager は，MNS Server から “MNS3” という MNSID を受信し，仮想端末用の VNI を作成，そのアドレス（仮想 IP アドレス）を 10.0.0.1 と設定している．従って，双方の端末のアドレス変換表には，$MNSID$ が “MNS3”，対応する仮想 IP アドレスが “10.0.0.1” と記載されている．

送信端末が VNI（10.0.0.1）からパケットを送信すると，ネットワークインタフェースからインターネットへと送信される前に，MNS Analyzer がそのパケットを受信する．そして，アドレス変換表を参照し，送信元アドレス（10.0.0.1）に対応する $MNSID$ (MNS3）を探し出す．そして，送信元アドレスを実 IP アドレス（133.11.236.10）へと書き換え，$MNSID$ （MNS3）をパケットに付加し，通常の IP 通信としてネットワークインタフェースからインターネットへと送信する．

次に，インターネットよりパケットを受信した端末は，MNS Analyzer でアドレス変換表を参照し，記載されている $MNSID$ (MNS3) に対応する仮想 IP アドレス（10.0.0.1）を探し出す．そして，あて先アドレスを仮想 IP アドレスに書き換え，MNSID を削除することで，対応す VNI へと転送する．

5.2.2 システム設計

システムを設計する際に，“既存のインターネットとの高い親和性” を重視した．そのためには，既存のインターネットサービスで多く利用されている TCP が，仮想端末間通信で利用可能でなければならない．これを実現するための要素機能を以下で述べる．

MNS Analyzer（アドレス変換）

4.4 節で述べたように MNS Analyzer の主な機能は，仮想 IP アドレスと実 IP アドレスを端末内部で変換することである．仮想端末間では，送信元を自らの仮想 IP アドレス，宛先を相手の実 IP アドレスとして TCP セッションを構築する．これを実現するために，仮想 IP アドレスと実 IP ア

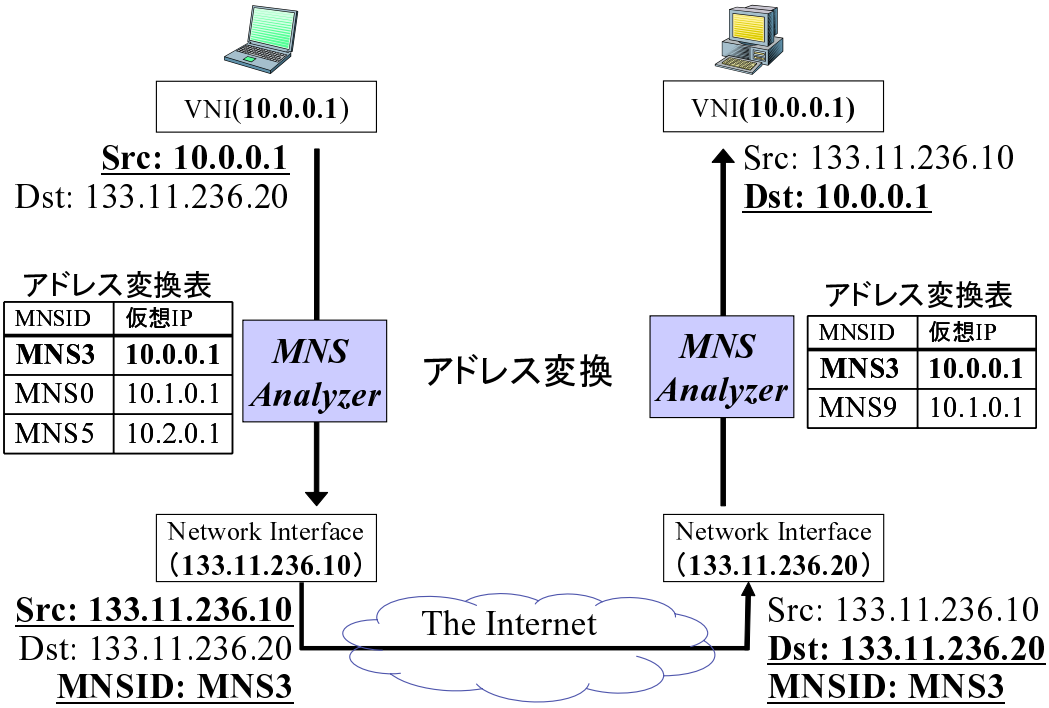

図 5.1: 仮想端末間通信

ドレスとの変換，それに伴う MNSID の付加・削除が必要である．仮想端末間で TCP を実現する際に，独自に TCP を管理する機構を構築するのでなく，OS の TCP スタックを用いることで効率のよい処理が実現できる．そのため，MNS Analyzer は OS の TCP スタックよりも下位層で動作する必要があり，カーネルモードのプログラムとなる．

アドレス変換情報の通知

アドレス変換を行うために，MNSID と仮想 IP アドレスを MNS Analyzer に通知するための機能である．アドレス変換を行うには，図 5.1 で示すように MNSID と仮想 IP アドレスを対応付けるアドレス変換表が必要である．ここで，4.4 節で述べたように，“MNS Server より MNSID の受信”，“VNI の作成・管理” を行うのは MNS Manager であり，ユーザが直接操作可能なユーザモードで動作するアプリケーションである．一方前述の通りに，OS の TCP スタックを利用し，効率のよい仮想端末間通信を実現するために，MNS Analyzer はカーネルモードのプログラムとなる．これより，MNS Manager から MNS Analyzer へアドレス変換情報を通知することが必要になるのだが，ユーザモードとカーネルモード間通信が増加すると仮想端末間通信の処理効率が低下するという問題が

生じる．これを解決するために，なるべく MNS Manager から MNS Analyzer への通信回数を減らすことが必要である．

そこで，MNS Analyzer もカーネルモード領域にアドレス変換表を持ち，MNS Manager の変換表に変更が生じたときのみ差分を通知することで，双方の変換表の同期を取る手法を採用する．これにより，両者の通信回数を抑え，仮想端末間の通信を全てカーネルモードで処理できる．

SYN パケットの送信元アドレスの設定

OS の TCP スタックよりも上位層で，SYN パケットの送信元アドレスを仮想 IP アドレスに設定する機能である．SYN パケットの送信元アドレスは，OS が経路表を参照し設定することが一般的な仕様であるため，送信端末の実 IP アドレスが設定される. そして，SYN パケットは MNS Analyzer を通る前に TCP スタックを通過してしまう．これより SYN パケットの送信元アドレスを設定しないと，“送信元を自らの仮想 IP アドレス，宛先を相手の実 IP アドレス” という MNS の要求する TCP セッションを構築することができない.

まとめ

仮想端末間の通信を実現するために重要な構成要素はアドレス変換を行う MNS Analyzer である．そして，システムの設計として，

- MNS Analyzer はカーネルモードで動作する
- MNS Analyzer はカーネルモード領域にアドレス変換表を確保する
- MNS Manager から MNS Analyzer へアドレス変換表の差分を通知する
- SYN パケットの送信元アドレスは，TCP スタックよりも上位層で設定する

ということが挙げられる．次節で，この条件を充たすよう行った実装の詳細について記す.

5.2.3 実装詳細

本節では前述したシステム設計を充たすように行った実装の詳細について記す．実装を行ったコンポーネントは図 5.2 である．各々について以下で述べる．

MNS Analyzer

5.2.2 節で述べたように，アドレス変換を行う MNS Analyzer は TCP スタックよりも下位層で動作するカーネルモードのプログラムでなくてはならない．図 5.2 に Windows のネットワークスタックも

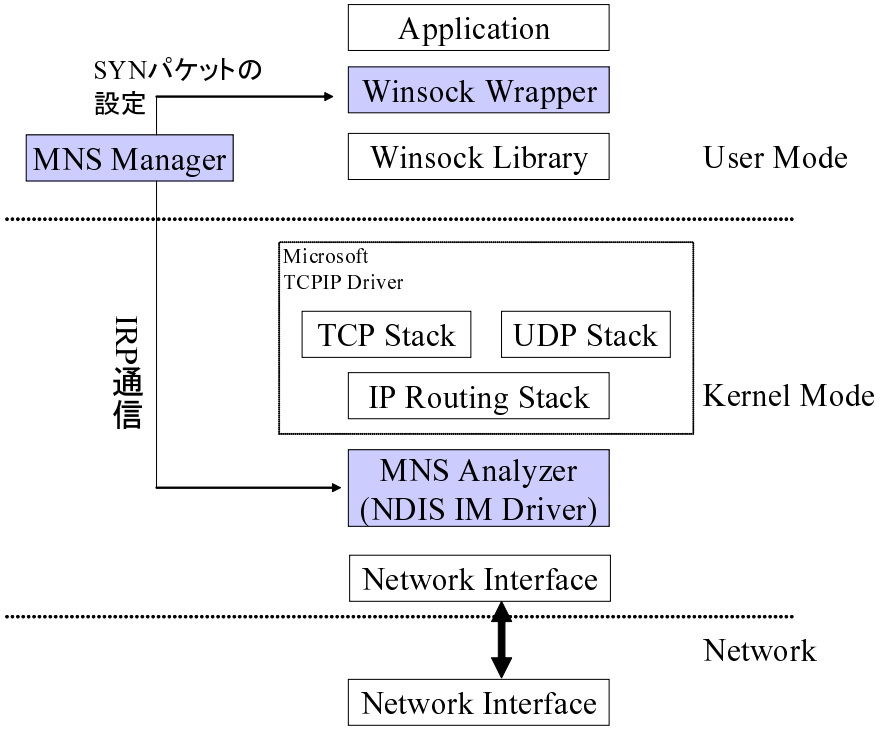

図 5.2: 実装コンポーネント図

示しており [25]，TCP よりも下位層という条件より Network Driver Interface Specification（NDIS）ドライバ [26] に MNS Analyzer を実装した．

NDIS は Microsoft が制定した，ネットワークドライバを作成するときに生じるハードウェアの差異を吸収するための API 群である．NDIS の詳しい構成を図 5.3 に示す [27]．NIC ドライバは直接ネットワークインタフェースカードを操作し，フレームの送受信などを行う．Transport ドライバはデータを受信し，それを TCP などのプロトコルスタックへと伝える役割を持つ．各々の間を NDIS API を通して通信する．たとえば，NIC から実際にフレームを送信する場合には，$MiniportSendPackets$ 関数を用いる．図 5.3 で各々のドライバの下に書いてある $Miniport/ProtocolXxx$ 関数がそのドライバに通信するインタフェースである．

MNS Analyzer は NDIS Intermediate ドライ（NDIS IM ドライバ）バとして実装した．NIDS IM ドライバは，ネットワークドライバへミドルウェアとして機能を追加できるように準備されている．NDIS IM ドライバは，上位層にある Transport ドライバからは NIC ドライバのようにみえ，逆に下位層に存在する NIC ドライバからは Transport ドライバのように見える．そのため，NDIS IM ドライバを作成する場合には，本来の NIC ドライバ，Transport ドライバが提供していた $Miniport/ProtocolXxx$ 関数全てを提供する必要がある．そのうえで，ミドルウェアとして機能追加をすることで，MNS

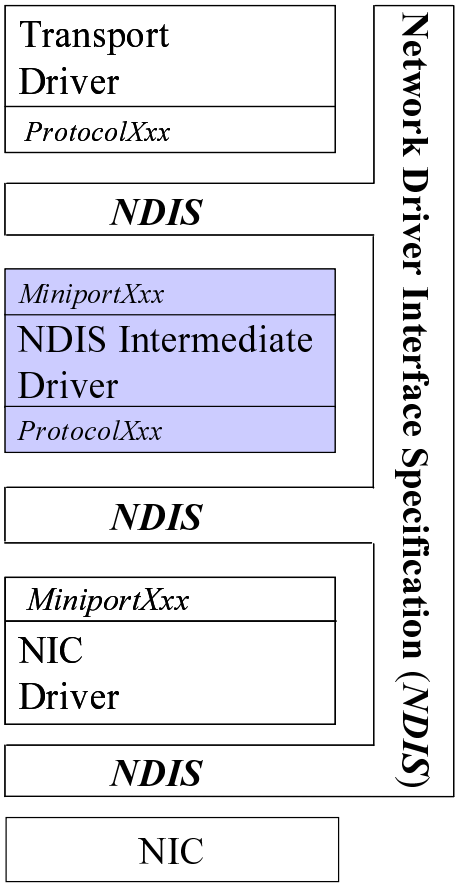

図 5.3: NDIS Driver

Analyzer を実現した.

まず，パケットを送信する場合について説明する．MNS Analyzer が行う処理は 5.2.2 節で述べたように，送信元アドレスを確認し，自らが所持しているアドレス変換表を参照して適合する $MNSID$ をパケットに付加することと，送信元アドレスを実 IP アドレスへと変換することである．このとき，存在しない VNI からパケットを送信できないように，仮想 IP アドレスがアドレス変換表に記載されていなければそのパケットは破棄する（図 5.4）．前述したように，パケットを送信するときは上位の Transport ドライバより $MiniportSendPackets$ 関数が呼ばれる．そこで，NDIS IM ドライバ内部の $MiniportSendPackets$ 関数で機能追加をし，パケット処理をしてから NIC ドライバへと送信する．まず，$MiniportSendPackets$ 関数から送信するデータを受信して，そのデータ長に $MNSID$ のデータ長を（本実装では 4Byte）加えたサイズのデータメモリを $NdisAllocateMemoryWithTag$ 関数を用いて確保する．次に，元の送信データを新しく確保した領域にコピーするのであるが，そ

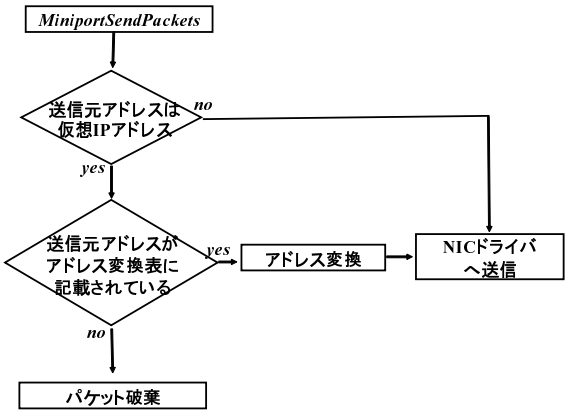

図 5.4: パケット送信処理

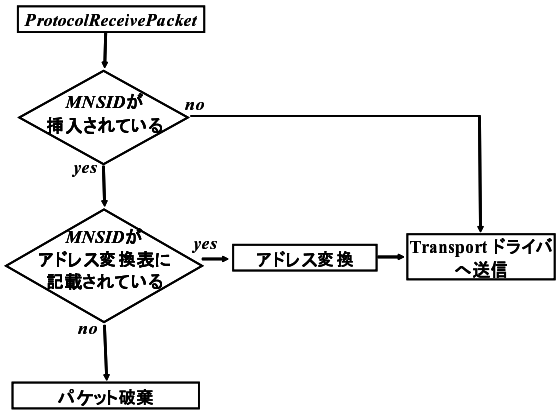

図 5.5: パケット受信処理

の際に送信元アドレスの変換，$MNSID$ の挿入を行う．最後に，新しく作成した送信データの IP チェックサムと TCP/UDP チェックサムを再計算し代入する．これにより，要求するアドレス変換を行ったデータが作成できるので，元の送信データの領域を開放し，$NdisSend$ 関数を用いて新しいデータを NIC ドライバへと送信する．以上がパケット送信時の処理の流れになる．

次に，パケットを受信する場合について説明する．MNS Analyzer が行う処理は 5.2.2 節で述べたように，パケットに付いている $MNSID$ を確認し，自らが所持しているアドレス変換表を参照して適合する仮想 IP アドレスへとパケットの宛先アドレスを変換することと，$MNSID$ を削除することである．このとき，自分のアドレス変換表に登録されていない $MNSID$ が記載されているパケットは自分が所属している MNS 外部からの通信と認識しパケットを破棄する（図 5.5）．具体的には，まず $ProtocolRecvPackt$ 関数から受信したデータを取り出し，$MNSID$ が挿入されているかの判断を行う．$MNSID$ が挿入されていた場合は，受信したデータ長から $MNSID$ のデータ長を減らしたサイズのデータメモリを確保する．受信したデータから $MNSID$ の削除，宛先アドレスの変換を行いデータをコピーし，新しく作成した送信データの IP チェックサムと TCP/UDP チェックサムを再計算し代入する．最後に，新しいデータを $NdisMIndicateReceivePacket$ 関数を用いて Transport ドライバへと送信する．以上が受信時の処理の流れとなる．

アドレス変換情報の通知

5.2.2 節で述べたように，効率のよい仮想端末間通信を実現するためには，ユーザモードのアプリケーションである MNS Manager と，前述したカーネルモードで動作する MNS Analyzer の間で通信する必要がある．これを実現するために Windows でドライバ間の通信の為に規定されている I/O Request Packet（IRP）を用いた．Windows では，カーネルモードへのアクセスは，Win32 シンボリックリンクという形でユーザモードに提供される．そこで，MNS Manager はそのシンボリックリ

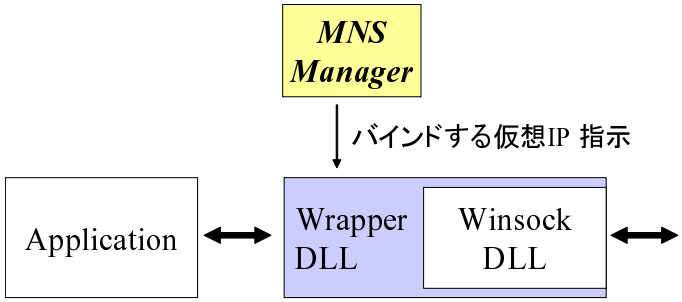

図 5.6: Winsock Wrappr DLL

ンクを用いて write 命令 (Win32 の実装では WriteFile API を使用）を呼び出し MNS Analyzer にデータを送信した.

SYN パケットの送信元アドレスの設定

5.2.2 節で述べたように，仮想端末間で TCP を利用するためには，TCP を開始する SYN パケットの送信元アドレスだけは，OS の TCP スタックよりも上位層で設定する必要がある．これを実現する手法は，複数考えられ，簡単な方法としてはクライアントアプリケーションが TCP を張るときに，ソケットを仮想 IP アドレスに対してバインドをして，仮想 IP アドレスを送信元アドレスとして明示的に指定することが考えられる．しかし，既存のクライアントアプリケーションは，ソケットをバインドせずに書かれているので，この手法では既存のアプリケーションに変更を加える必要のないシステムという設計指針に反する.

そこで本実装では，Windows のソケットライブラリである，Winsock のラッパダイナミックリクライブラリ（Wrapper Dll）を作成し，機能を追加をする手法をとった．具体的には，Winsock ライブラリの中で connect 関数が呼ばれる時に bind 関数を呼び MNS Manager が指示する仮想 IP アドレスをバインドにした．Winsock は dll の形で提供されており，ファイル名は “*ws*2_32.*dll*” である. この dll のファイル名を変更し（*ws*2_32*O*.*dll*），新たに wrapper dll として *ws*2_32.*dll* というファイルを作成する. そして新しく作成した *ws*2_32.*dll* 内部で，元々の Winsock dll である *ws*2_32*O*.*dll* を参照し，外部に提供されていた関数は全て新しい dll 内部でもそのまま提供し，connect 関数だけ内部で bind 関数を呼ぶように記述することによって要求する wrapper dll が作成できる. 以上の処理を図解したものが図 5.6 である．図 5.6 より分かるように，本手法では，既存のアプリケーションに変更を加えることなく，SYN パケットの送信元アドレスを設定することができる.

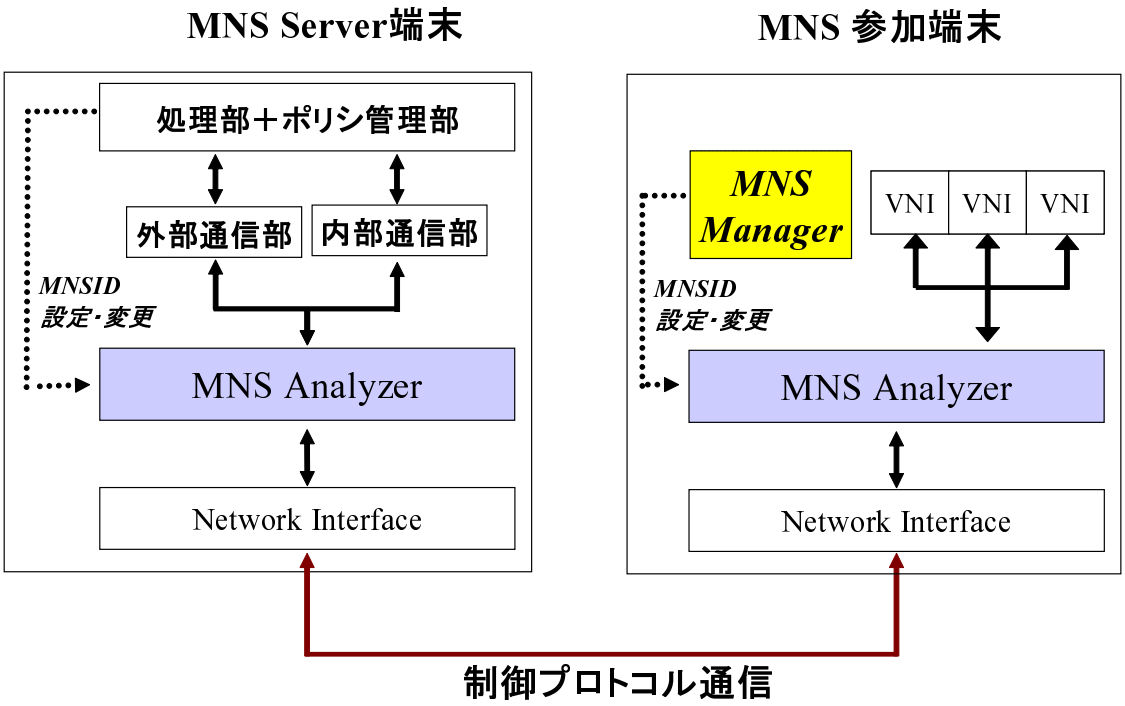

図 5.7: 内部仮想端末管理システム概要

5.3 内部仮想端末管理

5.3.1 システム設計

4.2 節で述べたように，MNS を構築するためには，内部にはユーザポリシに適った仮想端末しか存在しないことをシステムとして保証しなければならなく，そのためにはユーザポリシを充たす端末のみを “参加” させ，適わない端末が存在する場合は “脱退” させるシステムが要求される．そのためのシステム概要を図 5.7 を用いて説明する．

端末の参加・脱退は MNS 内部での通信に使用可能な $MNSID$ を所持しているかどうかで実現される．$MNSID$ を作成・配布するのは MNS Server が行う．また，4.4 節で述べたように，MNS Server には MNS 内部に存在する仮想端末から制御メッセージを受信する “内部通信部” と，MNS 外部の端末から参加要請を受ける “外部通信部” を持つ．一方 MNS に参加する端末は，自端末の MNS Manager を通して MMS Server との間で制御メッセージの通信を行う．

内部仮想端末管理を考えるうえで重要になるのは，4.4 節で述べた “$MNSID$ の閉塞性” の確保である．もし外部の端末が内部仮想端末間通信で利用可能な $MNSID$ を所持できると，内部に存在するように偽証し MNS 内部にしか提供されないサービスを利用できてしまう．このため，MNS に

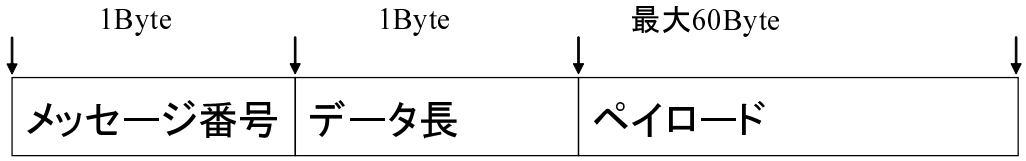

図 5.8: 制御メッセージのデータフォーマット

正しく参加している端末以外が，内部で利用可能な $MNSID$ を所持することは許されない．そのため，一度正しく MNS に参加していた端末が MNS から脱退した場合，そのとき使用可能であった $MNSID$ をどのように使用できなくするかが問題となる．本研究では，MNS から脱退する端末があるたびに MNS Server が新しい鍵を作成し，内部に残っている他の仮想端末すべてに対して $MNSID$ の更新を命じることで，この問題を解決した．MNS Analyzer は異なる $MNSID$ を持つパケットは同一 MNS 内部の仮想端末からの通信であるとは認識しないので，内部の仮想端末間で $MNSID$ の更新時刻に差が生じると仮想端末間の通信ができなくなる．したがって，$MNSID$ 更新の時刻は同期をとる必要がある．そのため，$MNSID$ の更新を要求するメッセージには，新しい $MNSID$ とともに，MNS Manager が $MNSID$ の更新を MNS Analyzer へと通知する時刻を付加する．これにより MNS 内部の仮想端末間で $MNSID$ を更新する時刻を一意に保つ．

5.3.2 制御プロトコル詳細

MNS Server と MNS Manager は TCP 上で制御メッセージの通信を行い，MNS の構築・維持に関する制御メッセージの通信を行う．MNS を“ 閉域ネットワーク ”であると定義したことにより，端末が MNS に参加するということは，MNS 内部の仮想端末と通信可能になるということである．以下で制御メッセージのプロトコルを説明する．

図 5.8 は制御メッセージのデータフォーマットである．制御メッセージはメッセージ番号，メッセージ長，ペイロードの 3 つのフィールドからなり，全体で 62Byte の固定長である．メッセージ番号は 1Byte の符号無し整数でメッセージの種類を表す．メッセージ長は 1Byte の符号無し整数でペイロード部分の長さを Byte 単位で表す．ペイロード部分は最大 60Byte 確保し，$MNSID$ などの各メッセージに必要な情報を付加する．表 5.1 に制御メッセージの種類とペイロード部に付加されるデータを示す．以下で端末参加・脱退の制御プロトコルの詳細を述べる（図 5.9，図 5.10）．

端末の参加には能動的参加と受動的参加がある．能動的参加とは端末側から参加要求メッセージを送信する場合である．“自分が所有する端末” などのユーザポリシをもつ MNS において，ユーザは端末から MNS Server に “$LOGIN$” を送信して端末を MNS に参加させる．“$LOGIN$” を受信した MNS Server は，ユーザ端末がポリシを充たしているか認証を行い，端末の参加を承認するメッセージ “$LOGIN\_OK$” を端末へ送信する．受動的参加とは MNS Server 側から端末へ参加要請メッセー

表 5.1: 制御メッセージ

メッセージ番号	メッセージ名	ペイロード	送信端末	受信端末	内部/外部通信
10	*LOGIN*		外部端末	MNS Server	外部通信
20	LOGIN_OK	MNSID	MNS Server	外部端末	外部通信
30	LOGIN_DENYED		MNS Server	外部端末	外部通信
40	LOGIN_INVITE	MNS名 MNS Server ポート番号	MNS Server	外部端末	外部通信
50	LOGOUT		内部仮想端末	MNS Server	内部通信
60	LOGOUT_OK		MNS Server	内部仮想端末	内部通信
70	LOGOUT_OFFER		MNS Server	内部仮想端末	内部通信
80	KEYCHANGE	MNSID MNSID 更新時刻	MNS Server	内部仮想端末	内部通信
90	KEYCHANGE_OK		内部仮想端末	MNS Server	内部通信

ジを送信する場合である．たとえば “特定の部屋に存在する端末” というユーザポリシで構築される MNS で，新しい端末が部屋に入ることを “端末監視部” が発見するとその端末情報を MNS Server へと送信する．その情報より，MNS Server が MNS に参加すべき端末であると認識すると，該当する端末に対して “$LOGIN\_INVITE$” を送信する．“$LOGIN\_INVITE$” には MNS 名，MNS Server が待ち受けるポート番号が記載される．“$LOGIN\_INVITE$” を受信した端末は，MNS へ参加する場合は MNS Server へ “$LOGIN$” を送信する．さらに，“$LOGIN$” メッセージを受信した MNS Server は，その端末がユーザポリシを充たすか再度判断を行い，問題が無い場合には “$LOGIN\_OK$” を端末へ送信する．“$LOGIN\_OK$” には $MNSID$ が記載されており，端末はこの $MNSID$ を MNS Analyzer にセットすることにより VNI を用いてその他の仮想端末との通信が可能となる．

端末の参加と同様に，脱退についても能動的な場合と受動的な場合がある．能動的脱退とは端末側から脱退要求メッセージを送信する場合である．たとえば，“自分が所有する端末” というユーザポリシで構築される MNS において，ユーザが使用していた端末を他人に譲るときなどは，MNS Server に “$LOGOUT$” を送信して端末を MNS から脱退させる．受動的脱退とは内部の仮想端末の情報が変化してユーザポリシを充たさなくなったときに，MNS Server 側から強制的に脱退させるメッセージを送信する場合である．たとえば，“特定の部屋に存在する端末” というユーザポリシによって構築される MNS では，部屋の内部に存在した端末が部屋の外部へ出ていったときに，その端末は定められたポリシを充たさなくなる．これを “端末監視部” が発見すると，その端末の情報は MNS Server へと送信される．そして，MNS Server は該当する端末に対して “$LOGOUT\_OFFER$” を送信する．

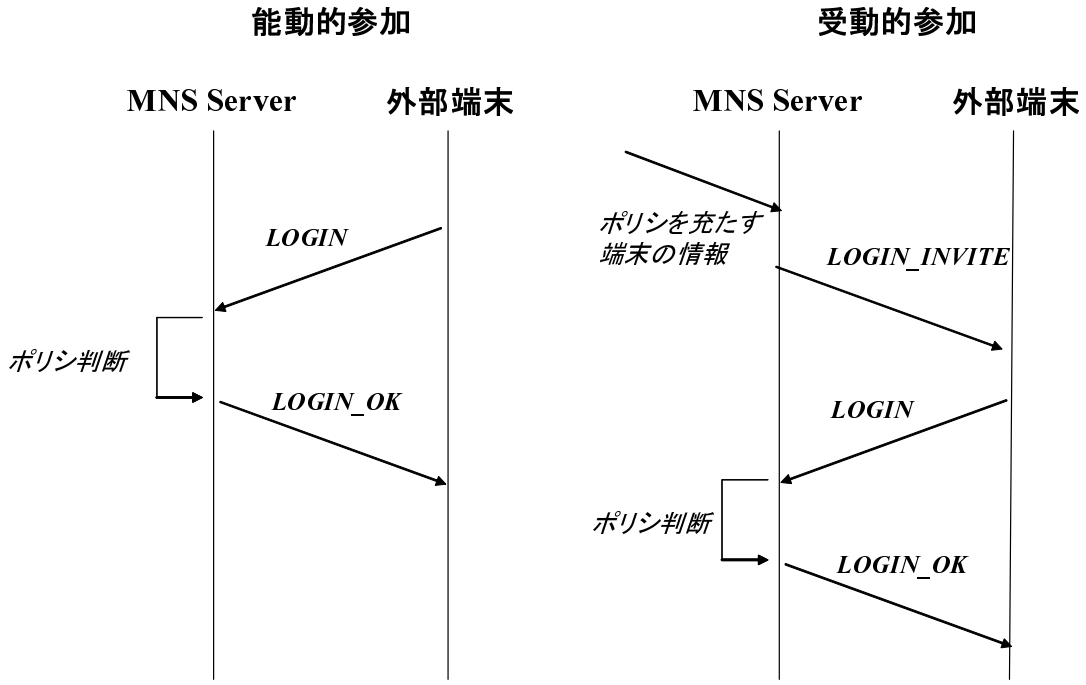

図 5.9: 端末参加時における制御プロトコル

前述したように，本実装では MNS に参加している端末が脱退を行うたびに，内部に残るその他すべての仮想端末の $MNSID$ を変更する．つまり，MNS Server は端末の脱退が決定した後，すなわち能動的脱退時においては “$LOGOUT\_OK$” を送信した後，受動的脱退時においては “$LOGOUT\_OFFER$” を送信した後に新しい $MNSID$ を生成し，その他の全ての内部仮想端末に対して $MNSID$ の更新を要求するメッセージ “$KEYCHANGE$” を送信する．前述のように，内部に存在する仮想端末で $MNSID$ の一意性を保つために，“$KEYCHANGE$” には，更新を行う時刻をペイロード部に記載しており，各端末の MNS Manager は指定された時刻に $MNSID$ の更新を行う．

5.4 動作検証

前述した実装をもって MNS システムの試作を Windows XP SP1 上に行い，MNS を利用したアクセス制御，仮想端末間通信における TCP 通信の実現を確認した．本節では実際に作成したアプリケーションのスクリーンショットや，仮想端末間通信を行っているパケットをキャプチャした画像をもって動作検証を行う．

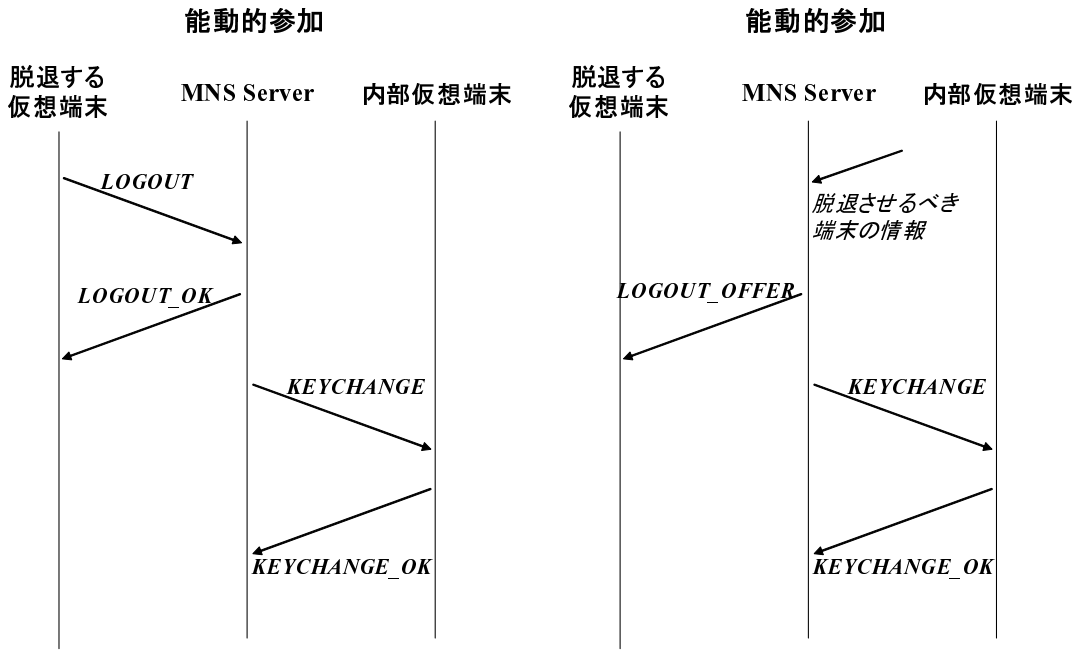

図 5.10: 内部仮想端末脱退時における制御プロトコル

作成したソフトウェアの紹介

ユーザ端末（192.168.17.8）が “研究室の中にある端末”,“自分の所有する端末” という 2 つの MNS に参加している状態を実験した. 図 5.11 に，作成した MNS Manager のスクリーンショットを示す. 参加している MNS 名が “LAB”，“MyDevice” であり，それぞれへの仮想 IP アドレスが 10.0.0.1, 10.0.0.2 であることが上側のコントロールより分かる．このように同時に複数の MNS へ参加が可能になっている．また，下側の “Key，VNI IP” のセットは，MNS Analyzer に通知をするアドレス変換のための $MNSID$ と仮想 IP アドレスの情報である．図 5.11 においては，“LAB MNS” 内部での $MNSID$ が “83887311” であり，“MyDevice MNS” 内部では “17106127” であることを示している.

図 5.13 は，“LAB MNS” 内部の様子を示したものである．これより，“LAB MNS” の MNS Server の実 IP アドレスが 192.168.17.7 であり，内部にはユーザ端末以外に 192.168.17.15，192.168.17.100, 192.168.18.101 の 3 端末が仮想端末として存在していることが分かる.

また，MNS Manager のユーザインタフェースには，“Logout ボタン” がついており，MNS を指定してこのボタンを押すことにより能動的に脱退することができる. 5.3 節で述べたように，MNS から脱退するときには，$MNSID$ の閉塞性を維持するために内部に残っている他の仮想端末に対しては $MNSID$ の更新を行う．図 5.12 は，“LAB MNS” から 192.168.17.100 の端末が脱退した後のユーザ端末の MNS Manager の様子である. ユーザ端末の仮想 IP アドレスなどは変化が無いが，“LAB

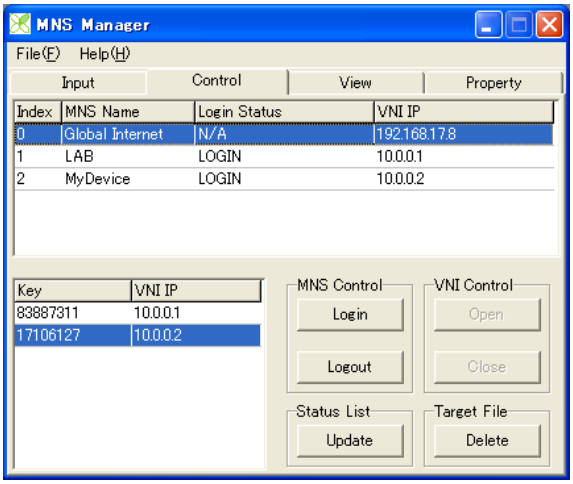

図 5.11: MNS Manager 鍵更新前

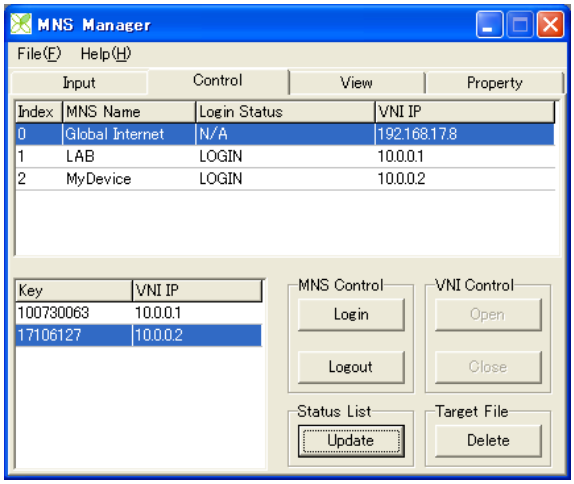

図 5.12: MNS Manager 鍵更新後

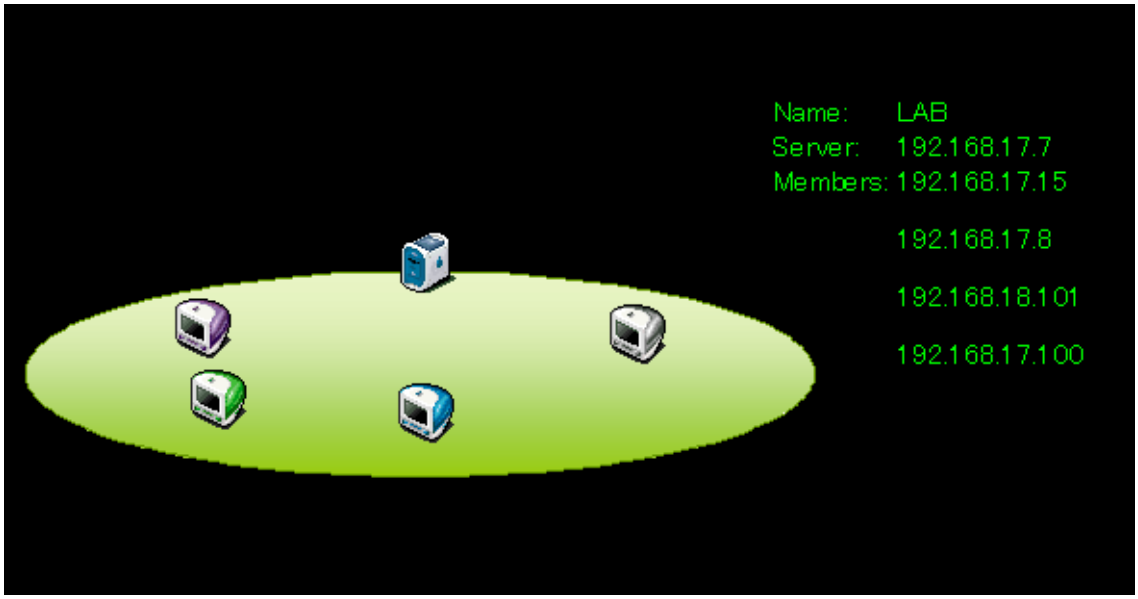

図 5.13: MNS 構築を示すスクリーンショット

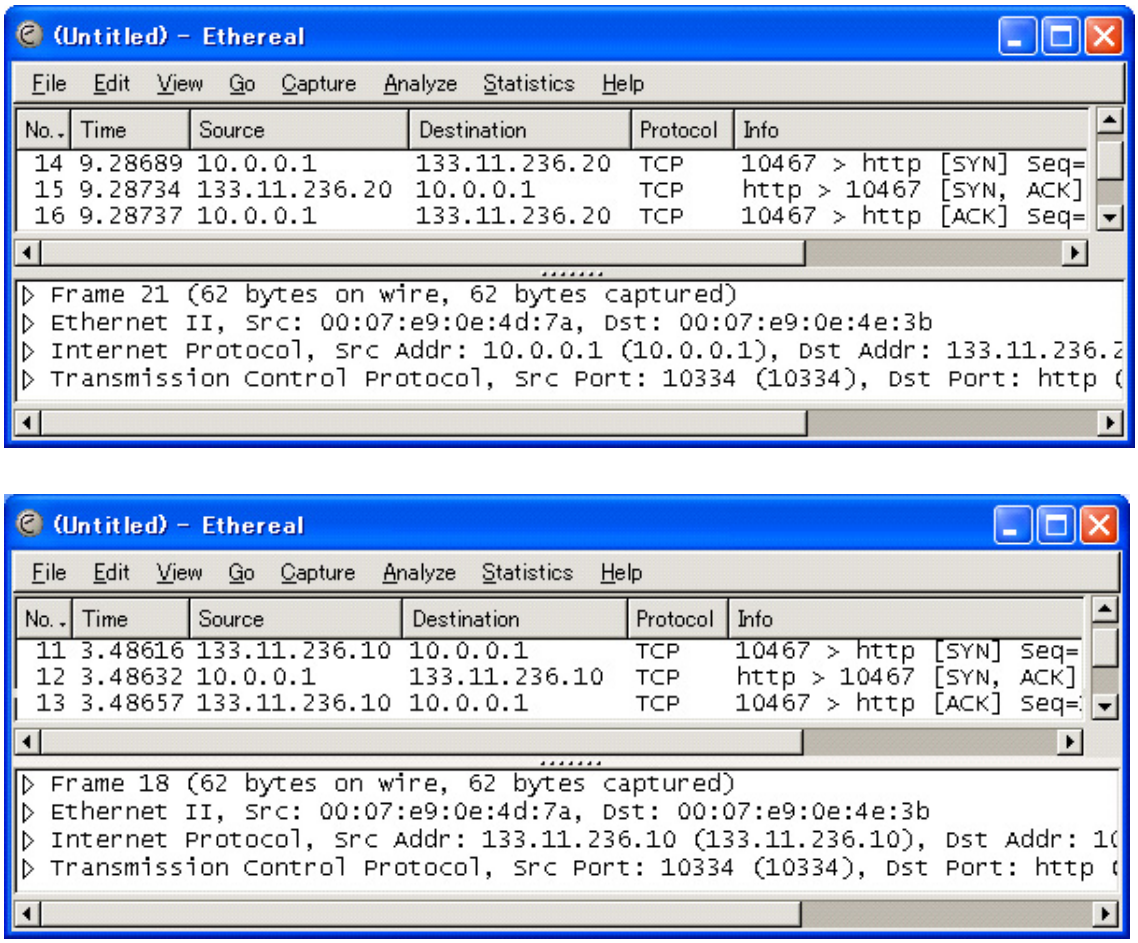

図 5.14: 仮想端末間通信パケットキャプチャ図

MNS” 内部での $MNSID$ が図 5.11 のときの “83887311” から “100730063” へと更新されていることが分かる.

仮想端末間通信の検証

図 5.14 を用いて仮想端末間通信における TCP 通信の利用について説明する．図 5.14 は，同じ MNS に参加している実 IP アドレスが 133.11.236.10 の端末から 133.11.236.20 の端末へと，仮想端末間通信として HTTP（ポート番号 80）の TCP セッションを張るときのパケットをキャプチャしたものであり，上側が 133.11.236.10 の端末でのキャプチャ画像であり，下側が 133.11.236.20 の端末でのキャプチャ画像である．このとき HTTP セッションを張るためのアプリケーションとしては,

一般に用いることができる Opera Web Browser [28] を用いた．双方の端末は，該当する MNS 用の仮想 IP アドレスとして 10.0.0.1 を割り当てている．また，パケットは，Windows の NDIS ドライバと TCP スタックとの間の階層でパケットをキャプチャする Ethereal [29] を用いてキャプチャした．つまり，送信端末ではアドレス変換を行う前の TCP スタックを通ったパケットであり，受信端末ではアドレス変換を行った後のパケットになる．

まず TCP の SYN パケットの送信元アドレスが，正しく仮想 IP アドレス（10.0.0.1）に設定できていることが分かる．前述したようにこの TCP セッションは，既存のアプリケーションである Opera Web Browser を変更することなく用いたものであり，“既存のアプリケーションに変更を必要としない” で SYN パケットの送信元アドレスの設定が実現されている．さらに，SYN パケット以外の通信において，送信側では送信元アドレスが仮想 IP アドレス（10.0.0.1）に，受信側では宛先アドレスが仮想 IP アドレス（10.0.0.1）へと変換されていることが分かる．これより，133.11.236.10 の端末から見ると，自端末の仮想 IP アドレス（10.0.0.1）と通信相手端末の実 IP アドレス（133.11.236.20）とで TCP がセッションが確立しており，同様に 133.11.236.20 の端末でも，自端末の仮想 IP アドレス（10.0.0.1）と通信相手の実 IP アドレス（133.11.236.10）とで TCP セッションが確立されていることが分かる．また，最終的に TCP スタックへと届けられるパケットは，対応する仮想 IP アドレスへとアドレス変換がなされている以外は既存のパケットと変わるところがないので，既存のアプリケーション（ソケット API）が変更なく利用できることが分かる．実験においても，Opera Web Browser を用いて仮想端末間通信として通常の Web ページを見ることが可能であった．また，実験においては Web（HTTP:80）以外にも，Mail（SMTP:25,POP:110）に関しても一般に用いられているアプリケーションに変更を加えることなく仮想端末間通信として TCP セッションを張り，サービスを利用できることを確認した．

アクセス制御の検証

MNS 外部の端末からのパケットが MNS Analzyer によって破棄されることによって，MNS 外部の端末からは MNS 内部に提供されているサービスを利用できないことを確認した．これを上記と同様に Web，Mail 双方のサービスでアプリケーションに変更を加えることなく実現した．また，一度 MNS に正しく参加していた端末が脱退したときにその時利用可能であった $MNSID$ を用いても通信ができないことを確認して $NSID$ の閉塞性が実現されていることを確認した．

5.5 おわりに

本章では，前章までで述べてきた MNS システムに対する初期実装について報告を行った．まず，仮想端末間通信の実現に関しては，“MNS Analyzer”，“アドレス変換情報の通知”，“SYN パケットの送信元アドレスの設定” が必要であると述べ，各々の設計指針と Windows XP SP1 上での実装について詳細を説明した．次に，“内部仮想端末管理” について，制御プロトコルの詳細を説明した．また，その際に “$MNSID$ の閉塞性” が重要であると述べ，その解決手法である端末が脱退するたびに内部に存在する仮想端末すべてに $MNSID$ の更新を促す手法について説明した．最後に実際に行った実装の検証を行った．

第6章　結論

6.1 本研究の主たる成果

本論文では，“同一部屋内部に存在する端末”，“自分が所有する端末” などのような限られた端末だけにサービスを提供するために必要な柔軟なアクセス制御を実現するために，ユーザ主導仮想閉域ネットワーク MyNetSpace を提案し，初期実装を通してその有用性を示した．

まず，サービスへのアクセス制御に対して筆者が想定するシナリオを示し，既存のアプリケーションレイヤ，ネットワークレイヤ [11] を用いたアクセス制御技術では，十分に対応できないことを述べた．この問題を解決するには仮想ネットワークの技術を用いればよいことを主張する一方，既存の VPN 技術 [18, 19, 20, 21, 22] でアクセス制御を行うには問題が存在することを述べた．以上のような問題点を解決し，柔軟なアクセス制御を実現する技術が MyNetSpace である．

第 3 章で MyNetSpace に必要な要求機能を挙げ，第 4 章で MyNetSpace のシステムアーキテクチャについて述べた．最後に第 5 章で，Windows XP SP1 上に行った初期実装を通して，MyNetSpace を用いることにより，既存のアプリケーションには変更を加える必要なく柔軟にアクセス制御ができることを示した．

インターネットの研究はこれまで，いかに相手端末までの到達性を確保するか，いかにグローバル規模で平等なエンドツーエンドの関係を構築するかに注力をしてきた [2, 23]．その成果により，ユビキタスコンピューティングという世界が提案され [1]，“いつでも，どこでも” 遠隔地に存在するネットワークリソースを利用できる環境が整いつつある．しかし，“いつでも，どこでも” 到達可能であるだけでは，ユーザにとって十分に利便性の高いインターネットは実現できないと考えられる．ユーザの利便性向上のために，ユーザが使いやすいネーミング [30, 6, 5]，物理ネットワークトポロジに依存しないルーティング [13]，ネットワーク上に存在するサービスの連携 [15, 16, 17] などが提案されている。筆者は，それらの中で，サービスには物理ネットワークトポロジには依存しない提供領域が存在し，ユーザが柔軟にサービスへのアクセスを制御できることが，ユーザの利便性向上に不可欠であると考え本研究を行った．本研究がユーザ利用しやすいインターネット構築 の一助となれば幸いである．

6.2 今後の展望

MyNetSpace システムにおいて $MNSID$ は，MyNetSpace 内部に存在することを証明する非常に重要な要素である．そのため第 4 章で述べたように，$MNSID$ の閉塞性を保障することが不可欠である．本研究では，一度正しく MyNetSpace へ参加していた端末が脱退した後で，所持していた $MNSID$ を使用できなくさせるために，内部に残っている仮想端末すべてに対して $MNSID$ の更新を命じるという手法をとった．この手法の欠点としては，参加・脱退が頻繁に生じる MyNetSpace では，仮想端末の $MNSID$ の更新が頻繁に生じるところにある．内部の仮想端末間通信は $MNSID$ の一意性が保障されなければ成立しないので，$MNSID$ の更新の間は通信が中断されてしまう．したがって，頻繁な $MNSID$ の更新は仮想端末間通信の性能を低下させる恐れがある．このような問題を解決できる $MNSID$ の管理手法や，作成アルゴリズムは今後の課題といえる．また同様に，通信を盗み見られることによって生じる $MNSID$ の漏洩などのセキュリティ面を含めた $MNSID$ の管理も今後の課題である．

また，第 4 章で述べたように，本研究では MyNetSpace 内部のポリシを実際に判断する機構は個々に対処することとしており明示していない．初期実装として “部屋内部に存在する端末” というポリシ判断には RFID タグを使用するアプリケーションを作成したが，その他にもユーザの位置測定や，プロファイル情報などを用いたポリシ判断を実現させる予定である．

さらには，本研究では初期実装として Windows XP SP1 上へと実装を行ったが，OS，実装手法を含めて最適な実装であるかの十分な検証はできていない．このような検証を行った後，実装の最適化を図り，スループットの測定を行うことなどで定量的な有用性を示すことも今後の課題である．

謝辞

本研究を進めていくに過程で直接ご指導いただき，さまざまな知識や研究生活に対する心構えを御教授くださり，また研究室内の打ち合わせのみならず常日頃から適切かつ有益な御指導，御鞭撻，叱咤激励をいただいた青山 友紀教授，森川 博之助教授に深く感謝いたします．また，研究環境の整備に尽力くださった渡邊 廣次助手，王 溪助手，事務処理のみでなく，円滑な研究室生活を支えてくださいました秘書の川北 敦子さん，宮島 史子さん，昨年度まで秘書として在籍された松本 みどりさんに感謝いたします．

博士課程 2 年の三村 和さんは多大な労力を惜しまず，本研究のすべての面において様々な助言をくださいました．ここに深く感謝の意を述べたいと思います．本研究室を昨年度卒業された川田 雅人さんには，青山森川研究室での研究生活のすごし方を一から教えいただき，また卒業されてからも幾度となく励ましの言葉をかけていただきました，深く感謝いたします．共に本研究をテーマとし，僕の我侭な要求にも付き合ってくれた修士課程 1 年の平井 肇君，卒論生の林 辰也君に心から感謝します．本研究を進めていく上で，日頃から励まし，そして議論に参加してくださった博士課程 3 年の川原 圭博さん，博士課程 2 年の金子 晋丈さん，博士課程 1 年の川西 直さん，猿渡 俊介さん，松本 延孝さん，本研究の実装を行うにあたりに多数の助言をくれた修士課程 1 年の小森田 賢史君に深く感謝します．博士課程 3 年の橋口 知弘さん，修士課程 1 年の丸山 達也君，卒論生の神田 敦君，鈴木 誠君には，研究室での日常生活において大変お世話になりました，感謝いたします．

青山森川研究室で過ごした 2 年間は非常に忙しく厳しいものではありましたが，その中でも月に最低 1 度は飲み会を行ったり，猿渡さんがサウナ，ダーツなどの大人の世界に僕を導いてくださったり，丸山君を筆頭に研究室のメンバと一緒にサッカーを行うなど楽しいこともたくさんあり，充実した 2 年間を過ごせました．青山森川研究室所属の皆様に感謝いたします．

また，修士課程 2 年間を同期として苦楽を共にした，小野 孝之君，鹿島 拓也君，鯉江 尚央君，林 智天君，ヴー クァン ミン君，本当にお世話になりました．この 2 年間，プレッシャーに押しつぶされそうになったことも何度もありましたが，みんなと一緒だったから乗り切れたと思います．飲み会などのイベントがあるとなぜかうちの学年だけ一杯いるような団結力とノリのよさが最高でした．今後別々の道へと歩みだすことになりますが，お互い頑張りましょう．そしてこれからも宜しくお願いします．

最後に今まで私を支えてくれた友人や，家族に深い感謝の意を述べたいと思います．
皆様ありがとうございました．

平成 17 年　1 月 31 日

参考文献

[1] Mark Weiser. The computer for the twenty-first century. *Scientific American*, pp. 94–104, Sep. 1991.

[2] D. D. Clark. The design philosophy of the darpa internet protocols. *Proc. ACM,SIGCOMM*, pp. 106–114, Aug. 1988.

[3] R. Droms. Dynamic host configuration protocol. *RFC 1531,Internet Engineering Task Force*, Oct. 1993.

[4] Karen R. Sollins. Designing for scale and differentiation. *Proc. Workshop on Future Directions in Network Architecture (FDNA-03)*, Aug. 2003.

[5] Hari Balakrishnan, Karthik Lakshminarayanan, Sylvia Ratnasamy, Scott Shenker, Ion Stoica, and Michael Walfish. A layered naming architecture for the internet. *Proc. ACM SIGCOMM 2004*, Sep. 2004.

[6] Ion Stoica, Daniel Adkins, Shelley Zhuang, Scott Shenker, and Sonesh Surana. Internet indirection infrastructure. *Proc. ACM SIGCOMM 2002*, Aug. 2002.

[7] D. Karger M. F. Kaashoek I. Stoica, R. Morris and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *Proc. ACM,SIGCOMM*, Aug. 2001.

[8] A. Rowstron and P. Druschel. Scalable, distributed object location and routing for large-scale peer-to-peer systems. *Proc. IFIP/ACM Middleware*, Nov. 2001.

[9] J. D. Kubiatowicz B. Y. Zhao and A. D. Joseph. Tapestry:an infrastructure for fault-tolerant wide-area location and routing. *U. C. Berkeley Technical Report*, pp. UCB//CSD–01–1141, Apr. 2000.

[10] Michael Walfish, Jeremy Stribling, Maxwell Krohn, Hari Balakrishnan, Robert Morris, and Scott Shenker. Middleboxes no longer considered harmful. *Proc. 7th Symposium on Operating Systems Design and Implementation (OSDI '04)*, Dec. 2004.

[11] P. Srisuresh and M. Holdrege. Ip network address translator (nat) terminology and considerations. RFC 2663, IETF, Aug. 1999.

[12] Joseph D. Touch, Yu-Shun Wang, Lars Eggert, and Gregory G. Finn. A virtual internet architecture. *Proc. Workshop on Future Directions in Network Architecture (FDNA-03)*, Aug. 2003.

[13] Bryan Ford. Unmanaged internet protocol. *Proc. 2nd Workshop on Hot Topics in Networks (HotNets-II)*, Nov. 2003.

[14] Christian Tschudin and Richard Gold. Network pointers. *Proc. 1st Workshop on Hot Topics in Networks (HotNets-I)*, Oct. 2002.

[15] Microsoft Corp. Universal plug and play device architecture, version 1.0, Jun. 2000. http:/ / www.upnp/ org/ download/ UPnPDA10_20000613.htm.

[16] Sun Microsystems. Jini^{TM} technology core platform specificatin,v.1.1, Oct. 2000. http:// www.sun.com/jini/space.

[17] 南正輝, 森川博之, 青山友紀. ユビキタス環境におけるサービス合成支援のためのインタフェース指向ネームサービス. 電子情報通信学会論文誌,, Vol. J86-B, No. 5, pp. 777–789, May. 2003.

[18] W. Townsley, A. Valencia, A. Rubens, G. Pall, G. Zorn, and B. Palter. Layer two tunneling protocol (l2tp). RFC 2661, IETF, Aug. 1999.

[19] K. Hamzeh, G. Pall, W. Verthein, J. Taarud, W. Little, and G. Zorn. Point-to-point tunneling protocol (pptp). RFC 2637, IETF, Jul. 1999.

[20] B. Fox. Virtual private networks identifier. RFC 2685, IETF, Sep. 1999.

[21] 藤田範人, 小出俊夫, 石川雄一, 塚本明. ピアツーピア型レイヤ 2 仮想ネットワークの提案. 電子情報通信学会ソサイエティ大会, No. B-6-59, Sep. 2004.

[22] 小出俊夫, 藤田 範人雄一, 塚本明. P2p 型レイヤ 2 仮想網におけるトポロジ構成手法の提案. 電子情報通信学会ソサイエティ大会, No. B-6-60, Sep. 2004.

[23] Jerome H. Saltzer, David P. Reed, and David D. Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, Vol. 2, No. 4, pp. 277–288, Nov. 1984.

[24] B. Moore, E. Ellesson, J. Strassner, and A. Westerinen. Policy core information model – version 1 specification. RFC 3060, IETF, Feb. 2001.

[25] Windows Network Stack.
http://www.ndis.com/papers/winpktfilter.htm.

[26] NDIS : Network Driver Interface Specification.
http://www.ndis.com.

[27] MSDN.
http://msdn.microsoft.com/.

[28] Opera Web Browser.
http://www.opera.com/.

[29] Ehtereal.
http://www.ethereal.com/.

[30] William Adjie-Winoto, Elliot Schwartz, Hari Balakrishnan, and Jeremy Lilley. The design and implementation of an intentional naming system. *SOSP*, Dec. 1999.

発表文献

[1] 飛岡　良明 , 三村　和 , 森川　博之 , 青山　友紀. “ユーザポリシを反映した仮想端末によるネットワーク空間の構築”. 電子情報通信学会ソサイエティ大会，September 2004.

[2] 三村　和 , 飛岡　良明 , 森川　博之 , 青山　友紀. “MyNetSpace を実現するためのネットワーク空間の管理・制御機構”. 電子情報通信学会総合大会，September 2004.

[3] 飛岡　良明 , 三村 和 , 森川 博之 , 青山 友紀. “MyNetSpace：柔軟なアクセス制御を実現するためのユーザ主導仮想閉域ネットワーク” 電子情報通信学会技術研究報告, 信学 IN, March 2005.（投稿済）

[4] 飛岡　良明 , 三村　和 , 森川　博之 , 青山　友紀. “MyNetSpace における仮想端末間通信機構の設計と実装”. 電子情報通信学会総合大会，March 2005.（投稿済）

[5] 三村　和 , 飛岡　良明 , 森川　博之 , 青山　友紀. “柔軟なサービスアクセス制御を実現するユーザ主導仮想閉域ネットワーク”. 電子情報通信学会総合大会，March 2005.（投稿済）

[6] 林　辰也 , 飛岡　良明 , 三村　和 , 森川　博之 , 青山　友紀. “MyNetSpace を構築するための端末管理機構”. 電子情報通信学会総合大会，March 2005.（投稿済）