Peer Pool: DNS クエリによって構成されるオーバレイネットワーク
Peer Pool: Overlay Network Configured by DNS Query

中野悦史 *
Etsushi Nakano
ets@ubi.cs.ritsumei.ac.jp

西尾信彦 †
Nobuhiko Nishio
nishio@cs.ritsumei.ac.jp

概要

スマート環境間の連携を可能とするためには NAT の背後にあるノード間で接続性を得る必要がある．このとき，必要なノードにのみ接続性を提供し他のノードへの影響を最小としたいが，現在一般に利用されている手法では難しい．そこで我々は，DNS クエリをインタフェースとして IP レイヤのオーバレイネットワークを動的に制御・構成する「Peer Pool」の研究をしている．Peer Pool によれば，ネットワークにただ 1 台のホストを追加することで，既存のネットワークやアプリケーションへの変更を最小限としながら，任意のノード群のみに他拠点との接続性を提供することが可能となる．本論文では Peer Pool の要件と設計について論じる．また，実装を行い応答時間と通信の遅延を測定した結果から実現性を示し，その接続性を関連研究と比較し評価する．

1. はじめに

オフィスや研究室だけでなく，一般家庭においても，ネットワークを介して様々な機器やアプリケーションを連携しユーザがサービスを享受できる環境 [1][2][3][4] が普及しつつある．それらは一種のスマート環境と呼ぶことができる．スマート環境の普及に伴い，ユーザは複数の環境において機器やサービスを利用する権限を持つようになることが想像される．もしユーザが，権限を持った機器やサービスのある自宅やオフィスなどの環境 (以下，拠点) の間で機器やサービスの連携をすることが可能ならば，ユーザが拠点を移動しても場所に限らず同様のサービスを受けることが可能となるだろう．また，拠点間で連携できることによる新たなサービスの創出も可能となるだろう．

スマート環境間で連携するためには，NAT の背後にあるサービスや機器などのノード間の接続性を得る必要がある．そのためには，Virtual Private Network(VPN) でネットワークを結合する手段が一般的である．しかし，この方法では IP アドレスを統一的に管理し，IP アドレスの重複や誤りに対応する必要がある．さらにネットワーク内の全てのノードに一様に他の拠点との接続性を提供してしまう点に注意しなくてはならない．複数のスマート環境間の連携において，これらが大きなコストとなることが想像に難くない．この問題を回避するためにノード単位で End-to-End の接続性を得る STUN[5] や TURN[6] などの NAT Traversal 技術も存在するが，利用するためにはアプリケーションがその技術に対応する必要がある．アプリケーションが市販のものであったり組込みソフトウェアである場合，対応は困難となる．

上記の問題を解決するため，我々は既存のシステムやアプリケーション，ネットワークに大きな変更を加えずに，任意のノード群のみに複数拠点間の接続性を提供する **Peer Pool** というシステムを設計し実装している．Peer Pool は IP レイヤのオーバレイネットワークを DNS クエリをインタフェースとして即時に構成・管理するシステムである．基本的な設計は，ネットワークに設置された 1 台のホストが NAT Traversal やアクセス制御，ノードの管理を行うことで，既存のノードへの変更を最小限に任意のノード群のみに接続性を提供するというものである．Peer Pool は Peer Pool を利用しないノードに対して影響を与えない．また，IP レイヤで接続性を提供し DNS クエリを制御に利用する設計であるため，既存のアプリケーションや，情報家電，スマートフォンなどの組込みシステムも接続性を得ることが可能である．加えて，DNS クエリによるノードの名前解決を利用することも可能である．つまり，Peer Pool を利用することで既に構築されたネットワーク間で IP レイヤの接続性を得ることを容易とし，スマート環境間の連携をすることが容易となる．

本稿では Peer Pool の設計と実装および評価を述べる．2 章では Peer Pool の要件とその背景を述べ，3 章で Peer Pool の設計の概要，続く 4 章で DNS クエリのインタフェースや内部動作を述べる．5 章では Peer Pool の実装，続く 6 章で評価を述べ，最後に今後の展望について述べる．なお，本文では特に明示的な記述がない限り，アドレスおよびネットワークは IPv4 のものを，NAT は NAPT(IP マスカレード) を含む広義の NAT を意味することとする．

2. Peer Pool における要件

現在のスマート環境の多くは，単一の拠点で利用することを想定して各ネットワークで閉じており，IPv4 のプライベートネットワークに構築されていると考えられる．場

* 立命館大学大学院理工学研究科
Graduate School of Science and Engineering, Ritsumeikan University
† 立命館大学情報理工学部
Department of Computer Science, Ritsumeikan University
本研究は総務省戦略的情報通信研究開発推進制度 (SCOPE)「異種スマート環境間をセキュアに動的接続・構成する基盤技術」の支援を受けて実施されている．

合によっては複数の NAT の背後に存在するプライベートネットワークであることも考えられる．つまり，ローカルの NAT がグローバルアドレスを持っている保証はされていないため，グローバルアドレスを必要とする技術は利用できない可能性を考える必要がある．したがって，拠点間でノード間の接続性を得るためには **(1)NAT Traversal の技術を利用する必要がある．**IPv6 で環境が構築されていればこの必要はないが，IPv6 に対応していない機器が存在する上，単一のネットワーク内に閉じていれば IPv4 で問題ないため，このような環境が長期に渡って存在し続けることが想定されている [7]．なお，NAT Traversal の技術を利用することは既存の NAT を置き換える必要がないというメリットも生む．

ところで，スマート環境を構成している機器は多種多様である．PC だけではなく，ネットワーク接続機能を有した PC の周辺機器や情報家電，ゲーム機，PDA やスマートフォンなどの非 PC の組込みシステムのデバイスも構成要素である．スマート環境間の連携を考えた場合，**(2) 非 PC のデバイスに対応する必要がある．**これらのデバイスはソフトウェアの更新が難しく，単独で NAT Traversal の技術を利用することはできないと考えるべきである．つまり，アプリケーションに変更が必要な STUN や TURN などの NAT Traversal 技術をそのまま利用することは望ましくない．また，ポートフォワーディングによってレイヤ 4 で接続性を提供する方法もあるが，この場合 Well-known ポートや固定ポート番号を使用するアプリケーションでポート番号の重複が発生しやすい．したがって，宛先ノードを指定するためにはアドレスだけでなくポート番号も予め知り，ポート番号を指定して通信を開始する必要がある．特殊なシステムが必要となり要件 (2) を考慮すると不向きである．以上より，**(3) 接続性は IP レイヤ (レイヤ 3) 以下で提供する必要がある．**レイヤ 3 以下の接続性であれば，上記のデバイスやアプリケーションも基本的にそのままで利用可能となる．

現状，レイヤ 3 以下で複数拠点間を連携可能とする技術として一般に利用されているのは VPN である．VPN でネットワークを結合することで，ネットワーク内全てのノードに他の拠点のノードとの接続性を提供することができる．しかし，この手法には問題がある．ネットワークを結合するためには，ネットワークアドレスおよびアドレスの管理方法を全拠点で統一し，重複や矛盾に注意を払う必要があることである．また，この方法は他の拠点との接続性を望まないノードに対しても一様に接続性を提供してしまう．これらの問題は結合する拠点数やノード数が多ければ多いほど大きな問題となる．**(4) 接続性が必要なノードにのみ接続性を提供する必要がある．**しかし，非 PC のデバイスを考慮した上で任意のノード群のみの VPN を構築するためには，物理的に，あるいは論理的にネットワークを分離する必要がある．これは一般に煩雑な作業を必要とする．また，どちらの場合においても VPN に参加したノードは既存のネットワークから隔離され，既存のネットワーク上のノードから通信ができなくなる問題を生じる．他のネットワークへの接続性を持ったノードが既存のネットワークから隔離されないように，**(5) ノードの既存のネットワークとの接続性を維持する必要がある．**

なお，モバイルデバイスなど出入りするノードの多いネットワークでは，設定の変更が頻繁に必要であり静的な設定は有効とは言い難い．このようなノードに対応するためにも，**(6) 適時にかつ容易に接続性を変更できる技術が必要である．**その方法として，既存のシステムやアプリケーション，非 PC のデバイスが変更なく利用できるように，標準的に利用される既存技術を利用する方法が望ましい．

3. Peer Pool の概要と構成

本章では，前章で述べた要件を実現するために我々が設計・実装した Peer Pool について，その概要とシステム構成について述べる．

本文以下では，あるノードから見た場合に，そのノードの所属するネットワークを『ローカルネットワーク』，他拠点のネットワークを『リモートネットワーク』と呼ぶ．また，ローカルネットワーク内のノードを『ローカルノード』と呼ぶ．

3.1 Peer Pool の概要

Peer Pool は, 各ネットワークに設置された **Pool Gateway**(以下，**PoolGW**) と呼ばれるホスト間で構築される IP レイヤのオーバレイネットワーク (**Pool ネットワーク**) を，Peer Pool を利用するノードからの DNS クエリをインタフェースとして即時制御・構成するシステムである．NAT Traversal やノード単位のアクセス制御，アドレス管理や DNS レコードの管理など，ネットワーク間連携に伴う煩雑な設定を PoolGW が一元管理することで，選択的にノードとオーバレイネットワークの間の接続性を制御することが可能である．また，この設計により DNS クエリを利用して Pool ネットワーク内のノードの探索 (名前解決，逆引き) を可能としている．

Pool ネットワークは PoolGW 間で構成される IPv4 の L3VPN である．その構築には NAT Traversal が可能であり各 PoolGW にアドレス空間を配布可能なレイヤ 3 以下の VPN 技術を利用する．また，既存のネットワークから分離するために Pool ネットワークごとに特殊なネットワークアドレスを割り当てる．必要ならば通信路の暗号化も行う．このとき，PoolGW 以外の Peer Pool を利用するノードには，Pool ネットワークへの参加時に PoolGW の持つアドレス空間のアドレスを利用してノード単位の双方向 NAT(*Pool* ネットワークのアドレス⇔実アドレス) が PoolGW に設定され，Pool ネットワークとの接続性が実現される．したがって，ノードが Pool ネットワークに参加した場合でもそのノードの保持する実アドレスに変更は加えられないため，ノードはローカルネットワークから隔離されることはない．また，Peer Pool を利用しないノードへの影響はなく，Pool ネットワークから通信を受け取ることはない．(図 1) このとき，PoolGW によって Pool ネットワークとの接続性を得たノードを **Pool ノード**と呼び，PoolGW に割り当てられるアドレスを **Pool アドレス**と呼ぶ．

PoolGW はノードからの DNS クエリをインタフェースとして各種の制御を行う．ノードの Pool ネットワークへ

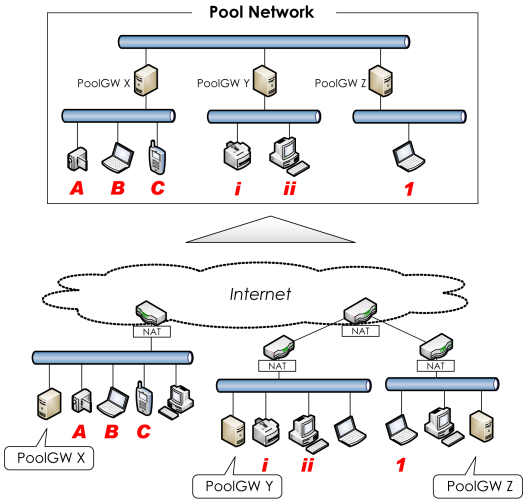

図 1　Peer Pool の概念図

の参加や脱退はノード単位で行うことが可能である．このとき，各ノードは Pool ネットワークへの参加の際に自分の名前を定義する．Peer Pool はこの名前を管理する機構 (**PoolDNS**) を持ち，Peer Pool を利用するノードは従来の DNS 宛と同様の DNS クエリによって宛先ノードを探索することが可能である．この機構により，Peer Pool を利用するノードは Pool ノードとの通信において，通信対象の Pool ノードの Pool アドレスを知る必要がない．また，Pool ノードが自分自身に割り当てられた Pool アドレスを知るためにこの機構を利用できる．このような設計により，IPv4 の通信が可能であり DNS クエリを発行できるノードは本システムを利用することが可能である．また，新たに本システムを利用するソフトウェアを開発するにあたって，特別な技術を実装する必要もない．

以上の設計により, Peer Pool を利用するノードは, Pool ネットワーク宛のパケットのネクストホップあるいはデフォルトルートをローカルネットワークの PoolGW に変更し，DNS として PoolGW を指定する必要がある．しかし，その変更によって従来の通信に影響がないように，PoolGW は Pool ネットワークに無関係なパケットをネットワーク本来のデフォルトルートや DNS に転送する設計としている．したがって，DHCP などを用いてネットワーク上の全てのノードの設定を変更しても問題とはならない．また，物理的に NAT を置き換える必要はない．

3.2　システム構成

3.2.1　Peer Pool の構成要素

Peer Pool には大きく分けて 3 種のノードが存在する．

■Pool Gateway　他の拠点の PoolGW 間でオーバレイネットワーク (Pool ネットワーク) を構築しており，Pool ネットワークとローカルネットワークの接続点となるホスト．各ネットワークに最小で 1 台設置され，ローカルネットワーク内の Pool ノードを管理する．DNS クエリをインタフェースとして双方向 NAT を設定し，ローカルノードの Pool ネットワークへの接続性を制御する．また，Pool ノードの名前と Pool アドレスの対応を PoolDNS に登録し，他のノードから名前解決要求を受け取ると PoolDNS を参照し応答する．なお，Pool ネットワークに関係のない通信はローカルネットワークの基本設定のデフォルトルートへ転送するルータとしても振舞う．

■PoolDNS　Pool ノードの名前とアドレスの対応 (DNS レコード) を管理するホスト．PoolDNS は PoolGW と分離可能であるが，PoolDNS の管理する DNS レコードが PoolGW の管理する NAT ルールと同期するように，基本的には PoolGW が PoolDNS としても動作する．したがって Pool ネットワークには複数の PoolDNS が存在する．PoolDNS 間はブロードキャストやマルチキャスト，アプリケーションレイヤ・マルチキャストなどの手法によって通信し，Pool ネットワーク内の全ノード探索を可能としている．なお，PoolDNS は PoolGW 同様に，Pool ネットワークに関係しない DNS クエリをローカルネットワークの基本設定の DNS に転送する．

■Pool ノード　PoolGW に Pool ネットワークへの参加要求を行うことによって，実アドレスと Pool アドレスの双方向 NAT ルールが設定されたノード．

3.2.2　ネットワーク構成

Pool ネットワークは IP レイヤのオーバレイネットワークである．PoolGW 間のレイヤ 3 以下の VPN 技術によって構成され，それぞれ既存のネットワークと異なるネットワークアドレスを持つ．Pool ネットワークでは各 PoolGW がアドレス空間を保持しサブネットを構成する．このアドレス空間のアドレスを用いて，Pool アドレスと実アドレスの双方向 NAT が構成される．

各 PoolGW は自分のローカルネットワーク内の Pool ノードを管理する．したがって，Pool ネットワークは 2 層のツリー構成と見ることができる．1 層目は PoolGW 群であり，2 層目は各 PoolGW の下に配置される，PoolGW と同一ネットワーク内の Pool ノード群である．

3.3　Pool ネットワークへのアクセス

あるノードが Pool ネットワーク内のノード (Pool ノード) に対してアクセスする方法は 2 つ存在する．(1) 自分も Pool ノードとなって通信対象の Pool アドレスでアクセスする方法と，(2) 通信対象の Pool ノードを仮想的にローカルノードとして自分の所属するネットワークに配置し，ローカルネットワークのアドレスでアクセスする方法である．後者は，経路表の設定を変更せずに利用できる唯一のアクセス手段となる．

3.3.1　Pool アドレスによるアクセス

Pool ノード間では Pool アドレスを用いて双方向の通信を行うことができる．このとき，どちらのノードにおいても通信相手のアドレスは Pool アドレスであり，実アドレスを知る必要はない (図 2)．なお，Pool アドレスを利用して通信をするためには，経路表において Pool ネットワークのネットワークアドレス宛のパケットのネクストホップを PoolGW に設定している必要がある．

ここで，ノード A(実アドレス A : Pool アドレス A') からノード B(実アドレス B : Pool アドレス B') への通信を例に挙げる．(図 3)

1. 前提として，ノード A とノード B は Pool ネットワー

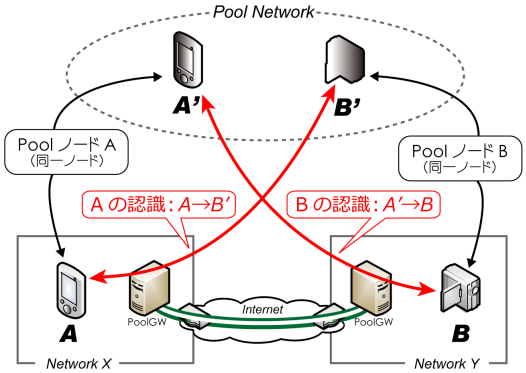

図 2　Pool アドレスによるアクセスの概念図

クへ参加している Pool ノードである．

2. ノード A は，Pool ノード B の Pool アドレス B' 宛のパケット『A to B'』を経路表に基づいて PoolGW X にルーティングする．
3. PoolGW X はこのパケットの送信元アドレス A を Pool ノード A の Pool アドレス A' に書き換え，『A' to B'』として宛先の Pool ネットワークにルーティングする．
4. 宛先 Pool ノード B は，リモートネットワークの PoolGW Y の背後に存在する．PoolGW Y は双方向 NAT のルールにしたがってパケットの宛先 B' を Pool ノード B の実アドレス B へと変換し，『A' to B』として Pool ノード B の実アドレスへルーティングする．
5. リプライのパケットは，先ほどとは逆方向に『B to A'』⇒『B' to A'』⇒『B' to A』という 2 度の双方向 NAT を経て，ノード A に到達する．

以上より，Pool ノード A が送信したパケットは『A to B'』であり，Pool ノード B が受信したパケットは『A' to B』である．Pool ノード間の通信においては，互いに宛先は Pool アドレスである．

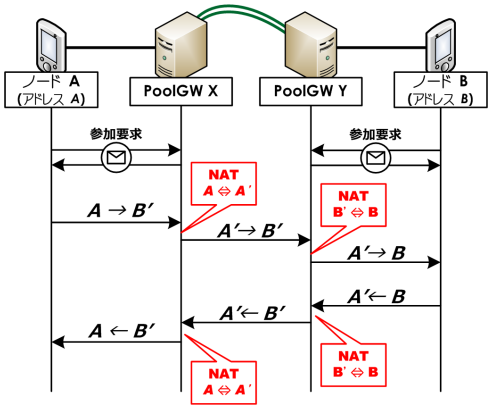

図 3　Pool アドレスによるアクセス

3.3.2　仮想ローカルノード化によるアクセス

PoolGW は，Pool ノードの Proxy となり，ローカルネットワーク内の通信で Pool ノード宛の通信を実現することが可能である (図 4)．このとき PoolGW はローカルネットワーク内の空きアドレスを利用して Proxy を構成・動作するため，あたかも通信対象の Pool ノードをローカルノードとして配置したかのように通信が可能となる．この方式では，ノードは経路表に変更を加えなくても Pool ノードとの通信が可能であり，また，Pool ネットワークに参加せずに通信が可能である．この通信方法において PoolGW が構成する Proxy を仮想ローカルノードと呼ぶ．

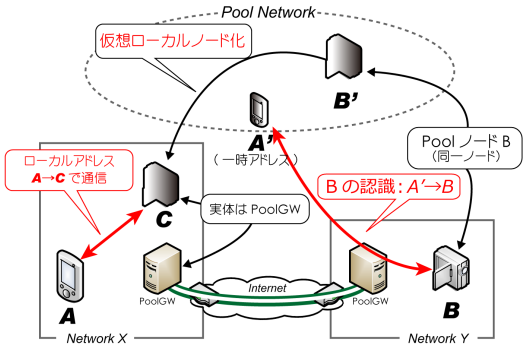

図 4　仮想ローカルノード化の概念図

ここで，ノード A(実アドレス A) から Pool ノード B(実アドレス B：Pool アドレス B') への通信を例に挙げる．(図 5)

1. まず，ノード A から Pool ノード B の仮想ローカルノード化要求を PoolGW X に送信する．
2. PoolGW X は Pool ノード B の Pool アドレス B' を確認し，Pool ノード B の Proxy となる仮想ローカルノード C(アドレス: C) を生成し，双方向 NAT ルール『C ⇔ B'』を設定する．アドレス C を保持するのは PoolGW X である．このとき同時にノード A の一時的な Pool アドレス A' が用意され，双方向 NAT ルール『A ⇔ A'』が設定される．処理が終了すると，PoolGW X はノード C のアドレスを名前解決結果としてノード A に返す．
3. ノード A はノード C に対して通信を開始する．ノード A からノード C へのパケットは，アドレス C を保持する PoolGW X に達し，NAT ルールに基づいて『A to C』から『A' to B'』へと書き換えられ，Pool ネットワークへとルーティングされる．
4. Pool ノード B を管理する PoolGW Y により，Pool ノード B 宛のパケットは『A' to B』へと書き換えられて Pool ノード B の実アドレスに到達する．
5. リプライにおいても 2 段階の双方向 NAT を通過する．『B to A'』⇒『B' to A'』⇒『C to A』という変換が行われ，ノード A に到達する時にはノード C からのリプライパケットとして振舞う．

以上により，ノード A は『A to C』というパケットを送信し，『C to A』というリプライを受け取る．ノード A は，仮想ローカルノード C を介してローカルネットワーク内の通信として Pool ノード B と通信することが可能である．

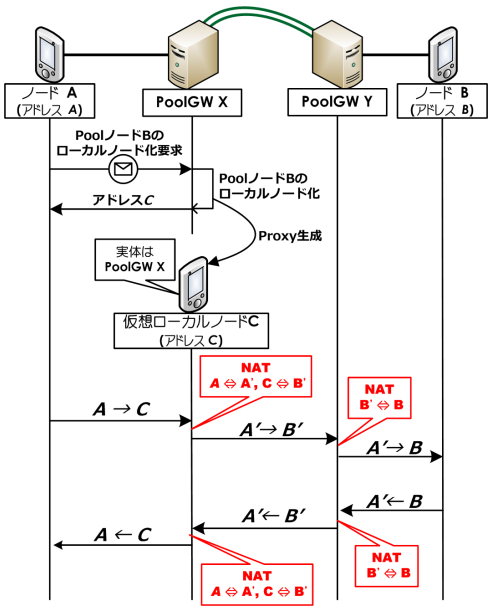

図 5　仮想ローカルノード化によるアクセス

4.　Peer Pool の仕様

前章では Peer Pool の概要およびシステム構成を述べた．本章ではユーザの立場から，Peer Pool の制御を DNS クエリによってどのように行うか，インタフェースの仕様を述べる．また，その時 PoolGW がどのように要求を処理するのか述べる．

4.1　Pool ノードの名前付け規則

まず Pool ノードの名前付け規則について説明する．現在の設計では Pool ノードの名前は完全修飾ドメイン名 (FQDN) で以下のように表わされる．

⟨HOST⟩.⟨GROUP⟩.pool
*⟨HOST⟩：ホスト名．数字以外で始まる文字列．
*⟨GROUP⟩：グループ名．”.”も含む文字列．

Pool ノードの FQDN は Pool ネットワークへの参加時に自由に定義できる．このとき，ホスト名だけでは利用や管理の際に不便となるため，ホスト名の後にグループ名を定義することができる．グループ名に関しては，現在予約済みである *in*，*out*，*local*，*admin* 以外の名前を自由に定義することが可能である．最後に Pool ネットワーク固有のサフィックスの *.pool* を付与する．なお，単純化のために FQDN は Pool ネットワーク内でユニークに設定される．

4.2　制御インタフェースの仕様

4.2.1　基本仕様

PoolGW は，PoolGW に対する DNS の A レコード (IPv4 アドレスのレコード) の名前解決要求クエリをインタフェースとして，各種の制御要求および Pool ノードの探索要求を受付可能である．PoolGW は標準的な DNS と同様に，基本的に UDP の 53 番ポートを要求の待受ポートとしている．PoolGW は要求に対する処理が成功した場合，何らかの名前解決結果を返す．要求に対して処理が失敗した場合は，レコードが存在しないことを示す「NXDOMAIN」が返される．PoolGW は名前解決結果を要求に対する応答とするため，名前解決結果がキャッシュされないように有効期限は非常に短く (例えば 1 秒) 設定される．

本文以下では明記しない限り，DNS クエリは A レコードに対する名前解決要求クエリを意味し，PoolGW への要求は DNS クエリによって行われるものとする．

4.2.2　Pool ネットワークへの参加要求

Pool ネットワークへの参加要求の DNS クエリは以下である．

⟨HOST⟩.⟨GROUP⟩.in.pool
*成功の返答: 割り当てられた Pool アドレス
*失敗の返答: NXDOMAIN

PoolGW において参加が許可され処理が成功すると，PoolDNS に「⟨HOST⟩.⟨GROUP⟩.pool ＝ ノード A の Pool アドレス」という A レコードが登録される．なお，予約されたグループには参加できない．名前が不適当 (予約済グループへの参加要求，名前が重複しているなど) の場合や，割り当て可能なアドレスがない場合など処理に失敗した場合は，NXDOMAIN が返される．

4.2.3　Pool ネットワークからの脱退要求

Pool ネットワークからの脱退要求の DNS クエリは以下である．

⟨HOST⟩.⟨GROUP⟩.out.pool
*成功の返答: 解放された Pool アドレス
*失敗の返答: NXDOMAIN

このホスト名とグループ名は，参加した時設定した名前と同一である．PoolGW において脱退処理が成功すると，解放された Pool アドレス (Pool ノード A が利用していたアドレス) が名前解決結果として返される．脱退処理に失敗した場合，対象ノードが Pool ネットワークに存在していない場合，あるいは対象ノードが DNS クエリの送信者と同一ではない場合，NXDOMAIN が返される．

4.2.4　通信対象の仮想ローカルノード化要求

Pool ノード *⟨HOST⟩.⟨GROUP⟩.pool* を仮想ローカルノード化して通信する場合，その Pool ノードと通信をするノードから以下の要求を行う．

⟨HOST⟩.⟨GROUP⟩.local.pool
*成功の返答: 仮想ローカルノードのアドレス
*失敗の返答: NXDOMAIN

PoolGW が対象の Pool ノード *⟨HOST⟩.⟨GROUP⟩.pool* を発見し，PoolGW における仮想ローカルノード化処理が成功した場合，仮想ローカルノードに割り当てられたローカルアドレスが名前解決結果として返される．処理に失敗した場合，対象のノードを Pool ネットワークから発見できなかった場合は，NXDOMAIN が返される．なお，このとき配置された仮想ローカルノードは次の DNS クエリによって撤去さ

れる．

⟨**HOST**⟩.⟨**GROUP**⟩**.local.out.pool**

また，一定時間この仮想ローカルノードに対して通信が発生しなかった場合も撤去される．この一定時間についてはユーザの利用環境や用途によって最適な値が異なるため，ユーザが任意に設定できるようすべきである．

4.2.5 代理要求

ユーザが操作するのに十分なインタフェースを備えていないアプリケーションや機器の場合，各種の要求を送ることが困難である．そのため，他のノードがそのノードに代わって代理要求を送ることができる．

代理要求の場合，各要求の前に次のプレフィックスを付与する．

⟨**TARGET**⟩**.**
*⟨TARGET⟩：アドレスの「.」を「_」に置換.

例えば，IP アドレス *192.168.0.1* のノードを *node.test.pool* という FQDN で Pool ネットワークに参加させる代理要求を行う場合，以下の DNS クエリを PoolGW に送信する．

192_168_0_1.node.test.in.pool

ただし現在の設計では，この要求が利用できるのは PoolGW か，静的に設定ファイルに記述されているノードか，*admin* グループに参加している Pool ノードだけである．

なお，*admin* グループは代理要求以外で参加できないグループであり，参加しても Pool ネットワークへ参加とはならず，PoolDNS にも登録されない特殊なグループである．PoolGW が代理要求を可能なノードを識別するためのグループとして管理し利用する．

4.2.6 Pool ノードの探索要求

Pool ネットワーク内のノードの探索要求は以下の DNS クエリである．

⟨**HOST**⟩.⟨**GROUP**⟩**.pool**
*成功の返答: 対象 Pool ノードの Pool アドレス
*失敗の返答: NXDOMAIN

また，逆引き要求の DNS クエリも Pool ノードの探索要求と判断される．この名前解決要求や逆引き要求，返答は，従来の DNS へのクエリおよび返答と同様である．

4.3 PoolGW における処理

この節では，Pool ネットワークへの参加や脱退，あるいは仮想ローカルノード化の要求に対し，PoolGW がどのように処理をするのか述べる．

4.3.1 Pool ネットワークへの参加

参加要求は DNS クエリのサフィックスが *.in.pool* であることから判断される．そのようなクエリを受け取った PoolGW は以下の処理を行う．

1. AAAA レコード (IPv6 アドレスのレコード) の問い合わせの場合，A レコードの問い合わせを行うよう返答をする (フォールバック処理).
2. 代理要求かどうかの確認．権限を確認し対象のアドレスを抽出する．
3. クエリの送信元アドレス/代理対象のアドレスが IPv4 であるか確認.
4. 定義している名前の確認．重複や禁止グループ名でないか確認.
5. 登録済の Pool ノードではないか確認．未登録の場合は空き Pool アドレスを確認し，双方向 NAT ルールを設定.
6. PoolDNS にレコードを追加.
7. DNS のリプライを生成，返答.

この過程においてエラーが発生した場合は，すぐに失敗の返答が行われる．

4.3.2 Pool ネットワークからの脱退

脱退要求は DNS クエリのサフィックスが *.out.pool* であることから判断される．脱退要求は Pool ノード自身からの要求か権限を持ったノードの代理要求以外では拒否される．脱退要求のクエリを受け取った PoolGW は以下の処理を行う．

1. AAAA レコードの問い合わせの場合，フォールバック処理.
2. 代理要求かどうかの確認．権限を確認し対象のアドレスを抽出する．
3. クエリの送信元アドレス/代理対象のアドレスが IPv4 であるか確認.
4. 脱退要求している対象 Pool ノードの存在確認.
5. 脱退要求している対象 Pool ノードの実アドレスと，クエリの送信元アドレス/代理対象のアドレスが同一であるか確認.
6. PoolDNS から該当レコードの削除.
7. 対象 Pool ノードの全てのレコードを削除しているなら，双方向 NAT ルールの解除.
8. DNS のリプライを生成，返答.

この過程においてエラーが発生した場合は，すぐに PoolGW は失敗の返答をする．

4.3.3 Pool ノードの仮想ローカルノード化

Pool ノードの仮想ローカルノード化要求は，DNS クエリのサフィックスが *.local.pool* であることから判断される．このクエリを受け取った PoolGW は以下の処理を行う．

1. AAAA レコードの問い合わせの場合，フォールバック処理.
2. 代理要求かどうかの確認．権限を確認し対象のアドレスを抽出する．
3. クエリの送信元ノードのアドレス/代理対象のアドレスが IPv4 であるか確認.
4. 対象の Pool ノードの確認.
5. 設定済の仮想ローカルノード化ではないか確認．設定済の場合，そのローカルアドレスを返答．そうでなければ引き続き処理.
6. 仮想ローカルノード化に利用する空きローカルアドレ

スの確認.
7. 要求ノードの一時アドレスとなる空き Pool アドレスの確認.
8. 対象ノードの宛先 NAT ルールと，要求ノードの双方向 NAT ルール設定.
9. PoolDNS に仮想ローカルノードのレコード追加.
10. DNS のリプライを生成，返答.

この過程においてエラーが発生した場合は，すぐに PoolGW は失敗の返答をする.

4.3.4 Pool ノードの探索

Pool ノードの探索は PoolGW が前述の各種の要求に該当しないクエリを受け取ったとき，あるいは，各要求の処理において PoolGW が PoolDNS にノードの存在を確認する場合に行われる．名前解決においては，PoolGW および PoolDNS は以下の処理を行う.

1. PoolGW は受け取った DNS クエリを元に, PoolDNS に DNS クエリを発行する．これは従来の DNS クエリと同様である.
2. PoolDNS は問い合わせのサフィックスが.*pool* の場合，あるいは逆引き要求の場合，自分自身の DNS レコードを確認する．該当しなければ従来の DNS に転送する.
3. 自分自身で解決できない場合，他の PoolDNS に問い合わせる．逆引きの場合は従来の DNS にも問い合わせる.
4. 名前解決結果や逆引き結果が得られた場合, PoolDNS は PoolGW に返答を転送する．NXDOMAIN しか得られない場合，あるいはタイムアウトが発生した場合，NXDOMAIN を返答する．それ以外のエラーを得られた場合エラーを転送する.
5. PoolGW は要求元に DNS のリプライを転送する.

この過程においてエラーが発生した場合は，PoolGW および PoolDNS はすぐに失敗の返答をする.

5. Peer Pool の実装

Peer Pool は現在，Linux Fedora Core 6 を OS とする PC を動作環境として，Java で実装されている．本章ではその実装について述べる.

5.1 Pool ネットワークの実装

PoolGW 間のオーバレイネットワークを構成する技術として OpenVPN[8] を採用し，L2VPN を構成している．この技術により，OpenVPN サーバさえグローバルネットワークに設置されていれば，PoolGW は NAT の内側に存在していても VPN を構築可能である.

本実装における Pool ネットワークは標準では 169.254.0.0/16 のネットワークアドレスを持ち，各 PoolGW に 169.254.x.0/24 のネットワークアドレスを配布する．各 PoolGW はこのアドレス空間を利用して Pool ノードを設定する．なお，各 PoolGW は 169.254.255.0/24 のネットワークに所属しており，例えば 169.254.255.1 のアドレスを持つ PoolGW は，169.254.1.0/24 のネットワークアドレスを, 169.254.255.2 のアドレスを持つ PoolGW は，169.254.2.0/24 のネットワークアドレスを管理する．この構造のため，各 PoolGW が接続性を与えられる Pool ノードは最大 254 台となる．なお，このネットワークアドレスは変更可能である.

5.2 双方向 NAT の実装

双方向 NAT を行う技術には iptables[9] を採用している．iptables は再起動なしに適時 NAT ルールを追加・削除することが可能なソフトウェアである．iptables に適時 NAT ルールを適用することで双方向 NAT を実現し，ノードの Pool ネットワークへの参加や Pool ノードの仮想ローカルノード化を実現する.

なお，Pool ノードの仮想ローカルノード化の実現には，仮想ローカルノードのアドレス宛のパケットを PoolGW が受け取る必要がある．本実装では ARP の代理応答を行うことで，仮想ローカルノードのアドレス宛のパケットを PoolGW が受け取っている．なお，仮想ローカルノードに利用するアドレスはローカルネットワークの管理者の指定した範囲の未使用アドレスから選択される.

5.3 PoolDNS およびインタフェース

PoolDNS およびインタフェースは Java で実装されている．本実装では PoolDNS は PoolGW と同一のホストで動作するように実装されており，PoolDNS は Pool ネットワーク内に PoolGW の数だけ存在する．各 PoolDNS はローカルネットワークの Pool ノードのレコードのみを管理し，PoolGW のセグメントに IP のブロードキャストを行うことで他の PoolDNS の管理するレコードを参照する．本実装では PoolDNS はブロードキャストに対し，該当レコードが存在しない場合は返答を行わない．そのため，PoolDNS は一定時間 (1000ms) 返答が得られない場合，レコードが存在しないと判断する．なお，名前解決結果がキャッシュされないように名前解決結果の有効期間は 1 秒に設定されている.

6. 評価

本章では Peer Pool が有効に機能することを，前章で述べた実装の動作・処理時間の計測結果に基づいて評価する．また，Peer Pool と関連研究について利用可能なアプリケーションを考察し比較する.

6.1 応答時間とオーバヘッド

6.1.1 測定環境について

まず，ネットワークの Round-Trip-Time(RTT) について, 100bytes の ICMP Echo Request (Ping) を 30 回行って測定した．ローカルネットワークにおける PoolGW とノードの間の RTT は，最小 0.217ms，平均 0.250ms，最大 0.293ms であった．また，Pool ネットワークにおいて PoolGW 間は最小 0.691ms，平均 0.859ms，最大 2.248ms の RTT であった.

なお, PoolGW は CPU: Intel®Pentium®M processor 1.73GHz，RAM: DDR2 SO-DIMM 400 512MB，NIC: Gigabit Ethernet の PC で動作し，それぞれ 100Mbps の

インタフェースを有する NAT の背後に配置されている．

6.1.2 要求への応答時間

Peer Pool のインタフェースは，既存の DNS と同様の振る舞いをする必要がある．つまり，応答までにタイムアウトを起こさない速度が必要となる．ところで，DNS レコードを参照する代表的なプログラムの dig および nslookup は，それぞれ 5 秒，2 秒がデフォルトのタイムアウト値となっている．したがって，応答時間は 2 秒が目標値となる．ここでは，本実装における要求への応答速度を元に，インタフェースが有効に動作することを評価する．

代理参加要求は，現在の実装において最も時間を必要とする処理である．特に，参加の処理で PoolGW が必ず行う名前の重複確認は，Pool ネットワーク内の全 PoolDNS に確認を行う時間が必要である．現在の実装においては，回答が得られない場合必ず 1000ms 待つ．つまり，この処理は要求への返答に 1 秒以上の時間を確実に必要とする．

この測定では，dig によって総計 254 ノード (1 台の PoolGW が持つ全 Pool アドレス) について単一の PoolGW に対し秒間 2 ノードずつ代理参加要求を行い，その応答時間を測定した．その要求はそれぞれ異なる名前で行うため，必ず成功する．その応答時間の測定結果は図 6 の通りである．この図では横軸に要求数，縦軸に応答までに要した時間を記している．なお，応答に要した時間の最小値は 1011ms，最大値は 1423ms，平均値は 1034ms であった．

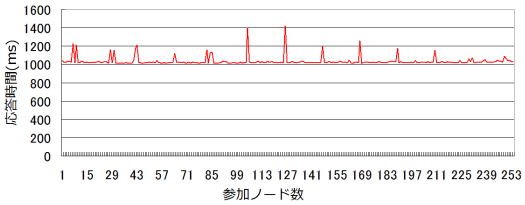

図 6　代理参加要求への返答時間

実装の問題であると思われるが，極端に返答が遅くなることが不定期に発生している．しかし，それ以外の値は安定して平均値付近を推移していることがわかる．また，その傾向はノード数が増加しても変化が見られない．いずれも目標値以内である．

なお，脱退の処理においてはローカルネットワーク内および単一の PoolGW で処理が完結するため，応答は代理参加要求より高速である．また，仮想ローカルノード化の処理においては，対象ノードが存在する限り応答が得られるため，応答待ちの 1000ms 以下になることが想像される．なお，この 1000ms という値は，Pool ネットワークの RTT を考えると余裕のある値である．したがって，この値は RTT を考慮してより小さな値に設定することが可能であり，インタフェースが設計通り利用できることが予測される．

6.1.3 アドレス変換によるオーバヘッド

Peer Pool は PoolGW において Pool ノードごとに双方向 NAT を設定するため，NAT ルールが増加し通信性能に影響することが懸念される．そのアドレス変換によってどの程度の遅延が発生するか，仮想ローカルノード化によるアクセスを例に，仮想ローカルノード化の処理を行った PoolGW が管理している Pool ノード数が 1 台の場合と 254 台の場合 (含む仮想ローカルノード時の一時的な Pool アドレス) でそれぞれ測定した．設計では，仮想ローカルノード化によるアクセスは片道で 3 つの NAT ルールを適用する．これは本システムにおいてアドレス変換回数が最多の通信である．その影響を測定し，拡張性を確認する．

100bytes の ICMP Echo Request (Ping) を 30 回行って測定した結果は，管理する Pool ノード数が 1 台の場合に最小 1.2ms / 平均 1.6ms / 最大 4.8ms，管理する Pool ノード数が 254 台の場合に最小 1.4ms / 平均 1.7ms / 最大 2.2ms となった (図 7) .

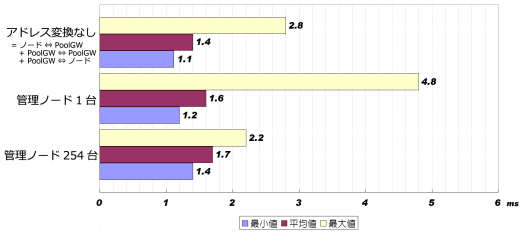

図 7　アドレス変換による遅延 (RTT)

また，TCP のスループットを 60 秒間に渡って測定した結果，管理する Pool ノード数が 1 台の場合に 50.1Mbps (転送量 359MBytes)，管理する Pool ノード数が 254 台の場合に 49.1Mbps (転送量 351MBytes) となった．PoolGW 間のスループットを基準とした場合，NAT ルールの増加によるスループット低下は約 2% である (図 8).

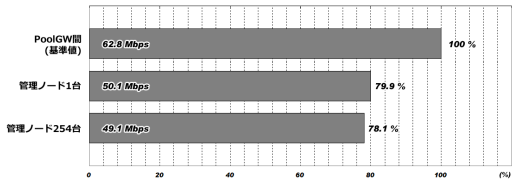

図 8　アドレス変換による遅延 (スループット)

これらの結果は実装に使用している iptables の性能に依存している．どちらにおいても管理ノード数の差による大きな差異は見られず，RTT では逆転が起きている点からも，管理ノード数の増加による性能への影響は小さいことがわかる．

6.2 提供される接続性の有用性

Peer Pool によって既存のアプリケーションが利用できるか，様々なプロトコルや通信方法のアプリケーションについて関連研究や技術と比較しながら考察する．それにより，Peer Pool の有用性を評価する．

なお，前述のとおり，Peer Pool は (A)Pool ネットワークに参加して宛先の Pool アドレスによりアクセスする方法と，(B) 宛先 Pool ノードを仮想ローカルノード化してアクセスする方法が存在する．それぞれについて考察する．

6.2.1 関連研究・技術

■(1) NATS　NATS[10] は NAT の拡張技術である．既存の NAT を置換して利用する．NATS では，IP パケットのヘッダあるいはペイロード中に宛先ノードの識別子を含めることで IP アドレスを拡張し，NATS の背後のノードを宛先として指定する．この識別子により，NATS と NATS の間ではプライベートネットワークのノード間に接続性が与えられる．

NATS は DNS クエリフックという機能を有している点に特徴がある．この機能は，NATS 背後のノードに対する名前解決要求に対し，NATS に向けてパケットを送るような仮想のアドレスを返答するものである．この機能により，特殊なパケットを生成できない既存のノードも NATS を利用することができる．

本研究とは接続性を得る手法に共通点があるが，NATS は接続性や DNS レコードを即時設定するという柔軟性に関しては考慮されていない．加えて，NATS 非対応の既存ノードが接続性を得るためには，必ず事前の名前解決を行う必要がある点に差異がある．本研究では名前解決は必ずしも必要ではない．

■(2) Teleshare　Teleshare[11] は，一般ユーザが異なるネットワークのユーザに，ネットワークに接続されているノードに対して一時的にアクセスを許可することを容易とする研究である．操作端末がゲートウェイ (GW) となりノード間のレイヤ 3 の接続性を提供し，通話と同期する一般ユーザに理解しやすいアクセス制御を実現している．Teleshare の提供する接続性は Peer Pool における仮想ローカルノード化と共通点が見られ，ローカルネットワークのアドレスを利用しアドレス変換を行うことでリモートノードとの通信を実現する．

本研究との差異は接続性の制御方法にある．Teleshare は GW そのものが操作端末であり，通話と接続性が連動する．Peer Pool では通信を行うノード自身が接続性を変更可能である．また，Teleshare では宛先ノードの指定にアドレスを指定する必要があるが，Peer Pool は名前解決を利用して通信可能である．

■(3) RNSplicer　RNSplicer は Tunneling with Service Discovery[12] という，複数ネットワーク間のサービス連携手法の実装である．ローカルネットワークにサービス情報の集約サーバを設置し，リモートネットワークのサービスをローカルのサーバにマッピングすることで，リモートネットワークのサービスを利用可能とする．RNSplicer は Bonjour(Multicast DNS) を対象としている．

本研究との差異は接続性を提供するレイヤにある．本研究は IP レイヤの接続性のみを提供することを目標としているが，この研究では複数レイヤで変換を行い接続性を提供する．

■(4) UPnP NAT Traversal　NAT の静的ポートフォワーディングを，UPnP を利用して自動的に設定する技術である [13]．一般ユーザ向けに一部の市販 NAT に実装されており，ソフトウェアでは Linux 向けに linux-igd(Linux UPnP Internet Gateway Device)[14] という実装がある．

本研究とは，得られる接続性がポートフォワーディングによるものである点に差異がある．この方法ではポート番号固定のサービスを複数のノードが利用しようとした場合，待受ポート番号の重複が起こってしまう．また，多段 NAT の環境下では利用することができない．なお，この機能を利用するためには，NAT に設定を依頼するための UPnP メッセージをアプリケーションに実装する必要がある．

6.2.2 想定する環境と通信

それぞれの研究で得られる接続性を比較するために，複数のネットワークに存在する複数のノード間で様々な通信を行う状況を想定し考察する．このときのノードは，特殊な技術を実装していないノードと想定する．

通信として，ICMP の (a)ping，TCP で宛先ポート番号指定が可能な (b)VNC，UDP で基本的に宛先ポート番号固定の (c)DNS を，それぞれクライアントとして利用することを想定する．続いて，特殊な通信方法を想定し，データ中にクライアントのアドレスが含まれるアクティブモードの (d)FTP を，クライアントとして利用することを想定する．また，マルチキャストのディレクトリサービスの (e)Bonjour を実装するサービスを利用できるか考察する．最後に，実際に通信を開始するノードにおいて，(f) 宛先ノードの名前解決が提供されるか考察する．

6.2.3 提供される接続性の考察

考察結果は表 1 となる．なお，(f) に関しては，実際に通信を開始するノードに，宛先ノードの DNS による名前解決が提供される場合は○，非 DNS の名前解決の場合△，どちらも提供されない場合は×としている．

表 1　利用可能なアプリケーション

	(a)	(b)	(c)	(d)	(e)	(f)
本研究 (A)	○	○	○	△	×	○
本研究 (B)	○	○	○	×	×	○
(1)	○	○	○	×	×	○
(2)	○	○	○	×	×	×
(3)	×	×	×	×	○	△
(4)	×	○	×	△	×	×

本研究 (A) の場合，(a)(b)(c) に関しては IP レベルの接続性により通信可能である．(d) に関しては，アプリケーションが自分自身のアドレス (データに埋込まれるアドレス) を指定することが可能な場合，Pool アドレスを指定することで通信可能となる．その設計となっていない場合は実アドレスを利用するため通信不可能である．(e) の通信は Pool ネットワーク宛の通信を意味するアドレスではないため，利用不能である．(f) の名前解決は設計に含まれている．

本研究 (B) も基本的に本研究 (A) と変わらない．ただし，(d) に関してはリモートネットワークで割り振られるアドレスが各々のネットワークで異なるため，利用不能である．

(1) の NATS に関してはレイヤ 3 の接続性のため，基本的に本研究 (A) と同様である．ただし，(d) は自分のアドレスのデータへの埋込みが URL で行われる場合のみ利用可能であり，一般的には利用不能である．(f) について，NATS は DNS クエリフックの機構を持ち名前解決がで

きる．

(2) の Teleshare は，基本的に本研究 (B) と同様である．(e) について，マルチキャストを転送することは設計上可能かもしれないが，実際にサービスを利用するためにはデータ中のアドレスの変換も必要であり，利用不能といえる．(f) に関し，名前解決は設計に含まれていない．

(3) の RNSplicer の場合他の技術と目的が異なり，(f) を利用可能とするものである．(a)(b)(c)(d) は UPnP などのアーキテクチャを利用しないため通信不可能であり，宛先を指定することもできない．ただし，(b) に関しては Bonjour のサービスとして登録すれば通信可能となる可能性はある．(f) について，DNS クエリによる名前解決は提供されないが，Bonjour のサービスの利用において宛先アドレスを意識する必要はない．

(4) の UPnP NAT Traversal の場合，(b) は利用可能である．また，(d) は NAT の持つアドレスを設定することができれば利用可能である．ポート番号の存在しない (a) や，ポート番号の重複が発生する (c) は，宛先ノードの指定ができないため利用できない．(e) に関しては，マルチキャストパケットがインターネットを通過する必要があり，一般には利用不能である．なお，NAT が UPnP に対応していること，NAT が他の NAT の背後に存在しないこと，など，様々に制約がある．また，名前解決は仕様に含まれておらず，宛先ノードに対応するポート番号を知る方法も仕様には含まれない．

以上より，Peer Pool は他の技術と比較しても有用な接続性を提供している．IP レイヤの接続性により，ポートフォワーディングでは利用できないアプリケーションも利用可能である．なお，Peer Pool を利用する場合，DNS の名前解決が利用できる点は既存のシステムとの親和性という点で有用といえる．その DNS レコードの設定は手動ではあるが，特殊な技術は必要なく，即時に適用される．一方で，Peer Pool は IP レベルのアドレス変換しか行わないため，データ中にアドレスを含むアプリケーションに関しては，データ中のアドレスと Pool アドレスが異なるため問題が発生しやすい．この問題は同様に，コンテンツやサーバの設定にアドレスや URL が含まれる場合も起こり得る．

7. まとめと今後の課題

本稿では，既に構築されているスマート環境間で，既存のネットワークやアプリケーションに与える影響を最小限に任意のノード群に IP レイヤの双方向の接続性を提供し，機器やサービスの連携を可能とする Peer Pool について，要件と設計，実装を述べた．その評価として応答時間とオーバヘッドを測定し，インタフェースが有効に動作すること，ノード毎に適用される双方向 NAT の影響が小さいことを示し，関連研究と比較して IP レイヤの有用な接続性が提供されることを示した．

現在の Peer Pool は IPv4 を想定して設計されている．しかし，その構造は IPv6 であっても利用可能なものである．今後は IPv6 を含めたオーバレイネットワークを構築できるよう設計と実装を進めていきたい．さらに，現実装と異なる VPN 技術を使用した場合や PoolGW の数が増加した場合の性能評価を取ることを目標としたい．また，Peer Pool はグループの扱いや PoolDNS の利用方法がまだ単純である．加えて，セキュリティ面に関してはほとんど考慮されていない．今後はスマート環境の連携シナリオを深く検討し，有効な設計を検討していきたい．

参考文献

[1] Yu Enokibori, Nobuhiko Nishio “Realizing A Secure Federation of Multi-Institutional Service Systems”, System Support for Ubiquitous Computing Workshop (UbiSys2004), Part of UbiComp2004, Nottingham, UK, Sept., 2004.

[2] 河口信夫, 稲垣康善, “cogma:動的ネットワーク環境における組み込み機器間の連携用ミドルウェア”, 情報処理学会コンピュータシステム・シンポジウム, pp.1-8, 2001.

[3] Digital Living Network Aliance, "http://www.dlna.org/", Aug 2007.

[4] Networking Bonjour Guides, "http://developer.apple.com/documentation/Networking/Bonjour-date.html", Aug 2007.

[5] J. Rosenberg, J. Weinberger, C. Huitema, R. Mahy, “STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)”, IETF RFC3489 [Standards Track], March 2003.

[6] J. Rosenberg, R. Mahy, C. Huitema, “Traversal Using Relay NAT (TURN)”, ’’http://tools.ietf.org/id/draft-rosenberg-midcom-turn-08.txt’’, Sep 2005.

[7] 社団法人 日本ネットワークインフォメーションセンター番号資源利用状況調査研究専門家チーム, “IPv4 アドレスの枯渇に向けた提言”, Mar 2006

[8] OpenVPN project, “OpenVPN”, "http://openvpn.net/", Aug 2007.

[9] netfilter core team, “netfilter/iptables homepage”, "http://www.netfilter.org/projects/iptables/index.html", Aug 2007.

[10] Kuniaki Kondo, Intec NetCore, Inc., “Capsulated Network Address Translation with Sub-Address(C-NATS)”, "http://www.nats-project.org/", Dec 2002.

[11] 鯉江尚央, 猿渡俊介, 水野浩太郎, 森川博之, 青山友紀, “Teleshare: 通信をメタファに用いたアクセス制御の設計と実装”, 電子情報通信学会技術研究報告, IN2004-258, Mar., Mar 2005.

[12] T.Mizukami, K.Cho “Tunneling with Service Discovery:A romote access model and Implemantation”,2006 3rd IEEE Consumer Communications and Networking Conference (CCNC2006), MP-1-03-5, CD-ROM Proceedings, Jan 2006.

[13] Microsoft Corporation, “Windows XP の NAT Traversal とユニバーサル プラグ アンド プレイによる最適化”, "http://www.microsoft.com/japan/technet/prodtechnol/winxppro/deploy/nattrnsv.mspx", Aug 2007.

[14] danielblueman, ewirt, krazydime, “Linux UPnP Internet Gateway Device”, "http://linux-igd.sourceforge.net/", Feb 2007.