RNSplicer：ローカルネットワークサービス連結接続モデルの提案と実装

水上智雄[†,††] 長健二朗[†,†††]

本論文では，ローカルネットワークにおけるサービスディスカバリの技術を用いることにより，VPN各種や SoftEther などとは違うアプローチで，複数のローカルネットワークサービスを連結接続して，リモートネットワークからそれらサービスにアクセスするアーキテクチャモデル「Tunneling with Service Discovery」を提案する．そのアーキテクチャモデルは，サービスディスカバリの連結機能とサービス自体の中継機能から構成される．まず，このモデルでは，サービスディスカバリのリスティング機能を用いて，ローカルネットワークにおけるサービス情報をダイナミックに集約するサーバを用意する．次に，サーバから得るサービス情報を基に，クライアントのローカルホストにサービスをマッピングし，そのサービスをクライアントのローカルネットワークのサービスディスカバリに登録する．これにより，擬似的にほかのローカルネットワークのサービスを，自分のネットワークのサービスとして自動的に利用することが可能になる．このモデルを用いることで，リモートネットワークから VPN でローカルネットワークに接続する場合ではなしえない，1 方向アクセス，マルチプルログイン，多段アクセスが可能になる．また，それにともなったネットワーク構成の変更の必要も生じない．本研究での実装においては，サービスディスカバリの技術として Multicast DNS（mDNS）を用いて，ローカルネットワークサービス連結接続ソフトウェア「RNSplicer」の開発を行った．このソフトウェアにより，Mac OS X で標準的に利用される Bonjour（旧名 Rendezvous）で提供されるローカルネットワークサービスを中継可能にした．

RNSplicer: A Model for Splicing Local Network Services and Its Implementation

Tomoo Mizukami[†,††] and Kenjiro Cho[†,†††]

In this paper, we propose a new architecture model, Tunneling with Service Discovery, for accessing local network services from a remote network. The model allows to splice local network services from multiple remote networks so that users can transparently access the advertised services as if they were local services. This architecture model consists of splicing networks and relaying services, and is different from the existing remote access technologies such as VPN and SoftEther. In this model, a server makes use of Service Discovery, and dynamically collects and updates the information about available local network services. A client on a remote network talks to the server, obtains the service information, maps these remote services onto its local services, and then, registers the mapped services on its local Service Discovery. Therefore, all hosts on the client's network are able to use the relayed services as pseudo local services. The model also allows One-way Access, Multi-Network Access and Multi-Stage Access without requiring any network reconfiguration. As an implementation of this architeture, we have developed "RNSplicer" that splices local network services provided by Multicast DNS (mDNS). RNSplicer works with all types of services provided by Bonjour (Rendezvous), a standard service discovery technology employed by Mac OS X and Darwin.

1. はじめに

近年，リモートネットワークからインターネットを介して別のローカルネットワークへ接続する技術が浸透してきた．これには 2 つの要因がある．1 つは IPv4 アドレスの不足対策として登場した NAT[1)] が一般に広く普及・浸透したことである．もう 1 つは，セキュリティの観点から多くのネットワークでファイアウォー

† 北陸先端科学技術大学院大学情報科学研究科
School of Information Science, Japan Advanced Institute of Science and Technology
†† ソフトバンクパブリッシング株式会社
Softbank Publishing, Inc.
††† IIJ 技術研究所
IIJ Research Laboratory

ルが導入されるようになったことである．その 2 つの要因により，ローカルネットワークとグローバルネットワーク（インターネット）を分離して構成するのが一般的になった．その結果，End-to-End の通信[3] ができなくなり，リモートアクセス技術の需要が増した．

現在，よく用いられるリモートアクセス技術は大別して，NAT[1]，トランスポート・リレー，Virtual Private Network[4]（以下 VPN）などがある．これらの技術はすべて，リレー処理によって，外部ネットワークからの通信を，同じネットワーク内のホストからの通信だと見せかける（偽装する）ことによって成立している．これらは上述した順にモデルが複雑になり，偽装化の質が高まる．

とくにデータリンク層を偽装する VPN では，リモート/ローカルホスト間で仮想ネットワークを構築するときに，全パケットに対してのブリッジ（ルーティング）を双方で設定することで，ローカルネットワークと変わらない接続性がリモートホストにもたらされる．

そして，この場合はローカルネットワーク内のサービスディスカバリをリモートネットワークから利用できる場合がある．個々のサービスの多様化からサービスの設定を手動で管理するのが難しくなってきた現在，このサービスディスカバリをリモートから利用できる利便性は高まってきている．

このような現状で，2 つのローカルネットワークで片側に参加するマシンが，もう一方のローカルネットワークの複数のマシンで提供されるサービスに，サービスディスカバリを用いて自動的にサービスを検出してアクセスする場合を仮定する．そのとき従来のリモートアクセス技術では次のような問題点がある．① NAT Traversal では，内部のサービスのマッピングを外部に提供する形式のため，サービスを意図しないホストに利用されてしまう可能性があり，セキュリティ上問題がある．② トランスポート・リレーを用いる場合は，あらかじめネットワーク先の状況を知っておき手動で設定しないといけないため，現状ではできない．③ Ethernet インタフェースをカプセル化する VPN では，VPN サーバ自体の設定のほかにブリッジ（ルーティング）をリモートホスト/ローカルホスト双方で設定する必要があり，それらの設定は煩雑である．かつ，IP マルチキャストをルーティングしても仕様によりサービスディスカバリを利用できない場合がある．④ MPLS を用いた VPN の場合では異なる 2 つのローカルネットワークで同一のアドレス空間を必要とするため，ネットワークの構築ポリシが異なる事業所間では，ローカルアドレスが重複するような場合が多々あり無前提に接続するようなことができない．⑤ ネットワーク層の仮想化を用いた VPN の場合，PPP での計算量削減とコンフリクトをさけるために IP マルチキャストを通さない実装が多く存在する．その場合にはサービスディスカバリの利用が制限される場合が多い．

そこで本研究では，この仮定においての問題点を克服することを課題とし，かつサービスディスカバリの柔軟性と自動性を用いた応用が容易なモデルとして，トンネリングにサービスディスカバリを組み合わせた「Tunneling with Service Discovery」を考え，その実装である「RNSplicer」を開発した．

2. 関 連 研 究

2.1 リモートアクセス技術

リモートアクセスを可能にする技術の種類は，主に以下があげられる．

(1) NAT[1]
(2) NAPT（Network Address Port Translation）
(3) UPnP（Univarsal Plug and Play）[5]
(4) STUN（Simple Traversal of UDP Through NAT）[6]
(5) トランスポート・リレー（HttpTunnel[8] など）
(6) Secure SHell（SSH2 TCP Port Forwarding[7]）
(7) VPN[4]

これらのうち，前者 4 つが NAT に関連し，UPnP と STUN が提供する外部ネットワークとの接続性は，俗に NAT Traversal と呼ばれる．次の 2 つはトランスポートレベルで，最後の VPN はクライアントホスト単位で接続性を確保する．

これらの技術に共通するのは，ローカルネットワーク内部のサービスをその外側のネットワークから使う目的に，何かしらのリレー処理（トンネリングリレーも含む）を含んでいることである．そして，それらの違いは，ローカルネットワークに接続するリモート側に対して，何を入り口として提供するかに現れている．

2.1.1 NAT とトランスポート・リレー

NAT や NAT Traversal は，ローカル側のアドレスとポートのマッピングをリモート側に提供し，それを通してローカルネットワークへの接続をリレーする．このモデルはシンプルで，IPNL[9]，TRIAD[10),11)]，AVES[12]，IPv4+4[13]，dns-sd[14] などの End-to-End の接続性を確保する研究では，このモデルを基に拡張を行っている．

それに対して，トランスポート・リレーはリレーの内

側と外側をトランスポートレベルで切り離すものである．途中の通信路をトンネリングで確保することもできる．SSH2 の仕様に含まれる TCP PortForwarding もトランスポート・リレーの 1 つの実装である．入り口や通信にセキュリティ上の工夫ができる反面，計算コストは NAT に比べて高くなる．

2.1.2 VPN

VPN は，クライアントの仮想インタフェースのデータリンク層（L2TP using IPsec，SoftEther [15]）か，ネットワーク層（L3TP using IPsec など）を，別ホストの仮想インタフェースの同層に偽装的に埋め込むことで，リモートアクセスを実現している．

VPN は，ネットワーク層以下のレイヤを仮想化するため，ネットワークのトポロジ形成ができる．したがって，1 つのローカルネットワークを複数のネットワークに横断的に存在させる分散ローカルネットワークの構築も可能である．

しかし，VPN には，サーバ側の設定が複雑で，クライアントのネットワークインタフェースの構成変更が必要であり，その導入は容易とはいえない．また，とくに下位層を仮想化して通信する際，必ずしも必要のないデータを送受信するなどの欠点もある．

2.2 サービスディスカバリ

サービスディスカバリは，ローカルネットワーク内でのサービスの提供情報を交換し，サービスを自動的に利用可能にする．サービスディスカバリにはさまざまな実装があり，以下のような仕様が存在する．

- SSDP [16]（UPnP [5] includes）
- mDNS [17]（Zeroconf [18] includes）
- SLPv2 [19]
- Jini [20]
- WINS [21]

これらの仕様内容はさまざまだが，ローカルスコープの IP マルチキャストを利用して，各自が自分の持つサービス情報を広報し，ネットワーク上のサービス情報を交換する仕組みが主流である（**表 1** ☆）．

3. 提案アーキクチャモデル：Tunneling with Service Discovery

2 つのローカルネットワークで片側に参加するすべてのマシンが，もう一方のローカルネットワークの複数のマシンで提供されるサービスにアクセスする場合を仮定したとき，現在では，とくに VPN が利用される場合が多い．それは，VPN が「リモートホストに対して，ローカルネットワーク上のホストと同様の接続性を提供することが可能なモデル」であるからである．しかし，この仮定を解決するとき，本来必要としているのは，「ローカルネットワークにリモートから接続すること」ではなく，「ローカルネットワークにあるサービスをリモートから利用すること」であることは自明である．もし利便性を失うことなくサービスの利用だけを実現できるのであれば，VPN のような「同様の接続性」は必ずしも必要ない．

そこで本研究では，「ローカルネットワークサービス情報を収集し，必要なものをリモートに伝え，それらサービスへの接続のみをリモートに提供するモデル」である「Tunneling with Service Discovery」を提案する．従来なら，このようなモデルを実現するには，手作業で必要なサービスをリストアップする必要があったが，サービスディスカバリを組み合わせることで自動的にサービス提供ができるようにしている．

その「Tunneling with Service Discovery」は，サービスディスカバリを中継する機能をトンネルの入り口と出口に配置し，情報交換させることで，外部ホストがローカルネットワークの内部構成を知らないまま，リモートアクセスをトンネリングさせる（**図 1**）．

このモデルでは，ローカルネットワークで収集されたローカルなサービス情報をリモートに送り，リモート側ではそれらのサービスを自身のサービスとして

表 1 主なサービスディスカバリと使用マルチキャストアドレス
Table 1 The mainstream kinds of Service Discovery and those scopes.

name	offerer	Multicast	Address Class
SSDP	UPnP Forum	239.255.255.250	Site-Local
mDNS	Apple	224.0.0.251	Local Network Control Block
Jini	Sun Microsystems	224.0.1.84-85	Internet Control Block
WINS	Microsoft	224.0.1.24	Internet Control Block

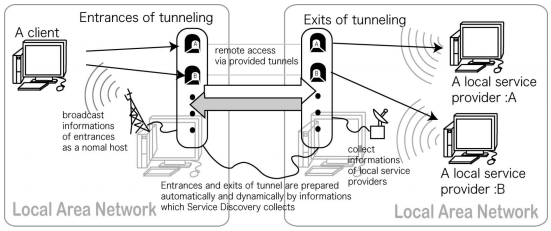

図 1 Tunneling with Service Discovery モデル
Fig. 1 Tunneling with Service Discovery (T-SD).

☆ SLPv2 は，ディレクトリエージェントという仕組みを使うことにより，IP マルチキャストは利用しない．

マップしてサービス中継用テーブルに格納する．同時に，マップされたサービスに対しトンネルの入口を用意する．作成したテーブルを基に，再びサービスディスカバリを用いて，リモート内へマップしたサービスを広報する．

これにより，リモートネットワークにいるクライアントは，サービスディスカバリを通して，透過的に自ネットワークのサービスとして，他ネットワークのサービスも利用できるようになる．ローカルネットワーク側では，ローカルなマシンがサーバとなりサービスを中継するので，サービスを提供するホストからは，通常のローカルアクセスのように見える．

これにより 1 章の仮定においては，意図しないホストに利用されてしまう可能性はトンネリングを用いることで低くなり，サービスディスカバリを利用するためネットワーク先の状況を知っておく必要もなく，トンネリングであるため，同一のアドレス空間を必要としないことになる．

なお，トンネリング通信の部分は，既存のさまざまな技術が利用可能である．実装においては，トンネルの入口と出口に TCPRelay を置き，その間を SSH2 PortForwarding で暗号化して通信している．

サービスディスカバリとリモートアクセス技術が連携動作をするアイデアは，NAT とサービスディスカバリを組み合わせる UPnP にも見られるが，サービスディスカバリがトンネリングと連携するというアイディアが仕様に明記されているものはなく，また，モデルとして認識されてもいない．

3.1 モデルの応用的特徴

このモデルは，あくまで，別のローカルネットワークのサービスに対する接続をあるホスト（クライアント）がローカルネットワーク全体に対して「連結して」提供するモデルであり，リモートホストをローカルネットワークに接続する VPN とは大きく異なる．

このモデルの応用的特徴として，「マルチアクセス」と「多段連結」「サービスの選択的提供」があげられる．マルチアクセスは，複数のローカルネットワークのサービスを 1 つのリモートネットワークに同時に中継する（**図 2**）．多段連結は，一度中継されたサービスをさらに別のネットワークに中継する．中継先で元のサービス情報が通常のローカルサービスと同様に広報されているので，サービスの中継が容易に多段化できる（**図 3**）．

しかし，マルチアクセスと多段連結を組み合わせるとループが起こるため，ループを防止する機構が必要になる．我々の実装においては各サービスに中継回数

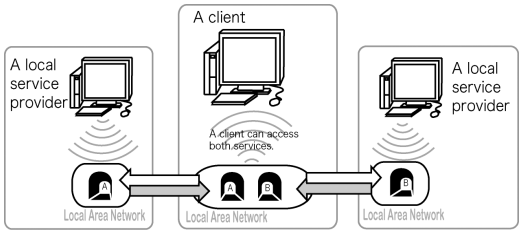

図 2 マルチアクセス
Fig. 2 Multiple access.

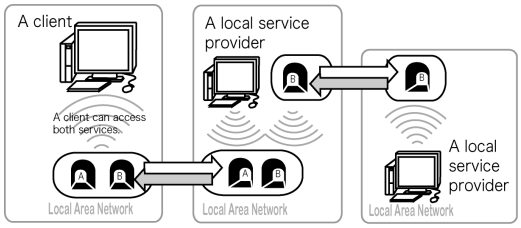

図 3 多段連結
Fig. 3 Multi-stage connection.

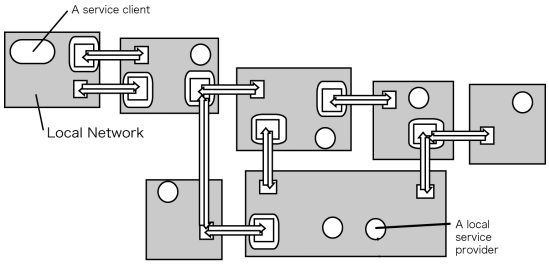

図 4 モデルの応用によるオーバレイネットワークの構築例
Fig. 4 An overlay networking example by T-SD.

のカウンタを持たせ，オリジンが同じサービスはカウンタ数の少ない中継経路を選択することでループを防止している．

これら 2 つの特徴によって，複数のネットワークを任意のトポロジで連結することが可能で，オーバレイネットワーク的なサービス共有クラスタを構築できる．たとえば**図 4** の場合であれば，左上のサービスクライアントは，図上のすべてのローカルサービスを利用できる．

また，サービスディスカバリをいったん中継テーブルに収めて処理をするモデルであるため，その中継テーブルの内容から，部分的に提供する/しないを，ローカル側で決めるといったことも可能である．

3.2 実装上の制約：ブロックアップ

別のローカルネットワークのサービスを「連結して」提供することは，使い方を誤れば，セキュリティ上の問題になる．リモートネットワークがホームオフィスのような場合は問題ないが，第三者が存在するような

ネットワーク全体にサービスを中継するわけにはいかない．

したがって，リモートネットワークの中継ホストのみをローカルネットワークのサービスに接続する利用形態も実現する必要がある．そのために，そのマシン以外は中継したサービスディスカバリのマルチキャストを受信できないようにする「ブロックアップ」を持つ必要がある．具体的なブロックアップの実現方法として我々は，サービスディスカバリが利用する UDP マルチキャストパケットが外部に送信されるところで遮断する手段をとっている．また，このブロックアップのモデルを利用して，送信先をローカルホストに限ることや，特定のホストに対しての送信を制御することによって，サービス対象をコントロールすることも可能になる．

4. 実装例：RNSplicer

上述したモデルを本研究では，MulticastDNS（mDNS）をサービスディスカバリに用い，分散 TCPRelay と SSH2 をトンネリングに利用することで実装した．実装したソフトウェア「RNSplicer」を使えば，ローカルネットワークで mDNS で提供されるサービスをリモートのローカルネットワークのホストから利用できる．中継できるサービスは，mDNS で利用してローカルサービスを提供するサービスで，TCP プロトコルのみを現時点ではサポートしている．とくに Mac OS X 環境では，Bonjour という Zeroconf の実装があり，その中で，mDNS がデフォルトで実装されているため，そこで提供されるローカルサービス（Xcode による分散コンパイル，Xgrid の呼び出し，プリンタ共有，ローカル Web サイト，iPhoto など）を中継できるソフトウェアである．

4.1 構　　造

ローカルで取得できる mDNS のサービスのエントリは，名前，サービスタイプ，アドレス，ポート，TXTRecord（ハッシュテーブル）の写像である．

$$Entry_i = f_{entry}(name_i, servicetype_i, address_i, port_i, TXTRecord_i)$$

これに対して，mDNS では，どの実装でも，サービスタイプを固定して，その写像の部分集合 S を得る，f_{search} 関数が存在する．

$$S = f_{search}(servicetype) \equiv \{Entry_0 \cdots Entry_n\}$$

このことを前提に，RNSplicer の具体的なハンドシェイクを解説する．

このソフトウェアは，先のモデルを忠実に再現した構造を持ちながらも，トンネリングの入り口と出口を，多対 1 のクライアント・サーバ構造と見立てて実装している．そして，入り口（クライアント）が，出口（サーバ）に SSH2 を介してログインし，次のハンドシェイクを行うことで，入り口側のローカルネットワークのホスト上の mDNS 対応アプリケーションは，自ローカルのサービスエントリ情報の 1 つとして得た中継連結ルートを通して，出口側のローカルネットワークのサービスにアクセスできる．

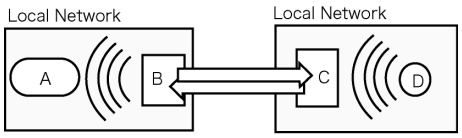

図 5 RNSplicer の接続モデル 1
Fig. 5 A linkup topology of RNSplicer.

このハンドシェイクの主たるシーケンスは，① クライアントが中継したいサービスタイプをサーバに送信する，② サーバは，そのサービスタイプから f_{search} を用いて S を取得する，③ ログインされたときの S のそれぞれのエントリに付加情報 A を追加したスナップショット S' を取得し，これをクライアントに送信する，

$$S' \equiv \{(Entry_0, A_0), \cdots, (Entry_n, A_n)\}$$

④ クライアントでは，送信されてきた S' をエントリ個々に分解し，クライアントアドレス CA と，エントリ 1 つずつに割り当てるポート CAP をアドレスとポートに埋め込んだ $Entry'$ を作り，それを集合 S'' に収める，

$$Entry'_i = f_{convert}(Entry_i) = f_{entry}(name, servicetype, CA, CAP_i, TXTRecord)$$
$$S'' \equiv \{Entry'_0 \cdots Entry'_n\}$$

⑤ 集合 S'' の $Entry'$ を mDNS を用いて 1 つ 1 つクライアント側で，そのサービス情報をブロードキャストする．⑥ CA：CAP_i を入り口とし，サーバを介して $address_i$：$port_i$ 宛の接続を提供するトンネリングを用意する，という手順を踏んでいる．

具体的な利用モデルとしては**図 5** のような接続モデルのときに，ホスト B とホスト C☆に RNSplicer をインストールし，ホスト C で，RNSplicer を起動し，サーバサービスを起動する．そしてホスト B で RNSplicer を起動し，クライアントサービスを起動し

☆ このとき，ホスト C には sshd が内蔵されている必要がある．

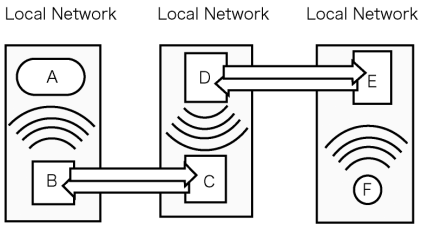

図 6 RNSplicer の接続モデル 2
Fig. 6 A linkup topology of RNSplicer.

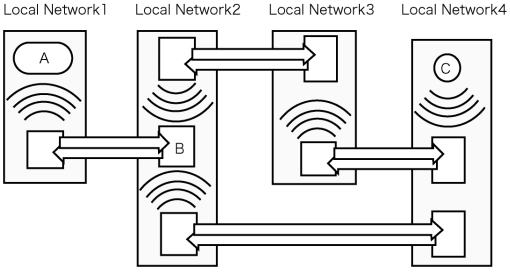

図 7 RNSplicer の接続モデル 3
Fig. 7 A linkup topology of RNSplicer.

て，このクライアントで C に対してログインする☆．

これにより，C 側のローカルネットワークで提供されているローカルサービスを，B 側のローカルネットワークで利用できるようになる．つまり，A から D にアクセスできる．

また，それだけではなく，**図 6** のような中継の中継も可能にしており，図 6 では，B と C を接続し，D と E を接続すると，A で F のサービスが利用できる．ここで RNSplicer がインストールされる必要があるのは，B，C，D，E の 4 つである．

現在の実装においては，別プロセスで 2 つ RNSplicer を起動する必要があるがマルチアクセスも可能で，かつ，**図 7** のように，中継の中継をする際には，B の RNSplicer のサーバが中継をリスティングする際に，ローカルネットワーク 2 内の別の RNSplicer のクライアント 2 つと通信し，サービスの中継回数（metric）を比較することで，A から C に対してのアクセスの場合，ローカルネットワーク 1 →ローカルネットワーク 2 →ローカルネットワーク 4 の順による最短の経路での中継がなされるようになっている．

なお，sshd は必ずしも RNSplicer と同じマシンにある必要はなく，RNSplicer に到達できる WAN 上のホストに ssh でログインできれば中継可能である．また ssh のポートも自由に設定できるため，ブロードバンドルータなどで利用される NAPT などと連携して，外出先から自宅のネットワーク内のリソースにアクセスするといったことも可能である．

☆ C の sshd にログインできるユーザ名，パスワードが必要．

5. VPN との比較

5.1 Mac OS X どうしの VPN 接続の問題点

現在，Mac OS X で利用できる Apple が用意している VPN は，その VPN システム単体で mDNS を中継することはできない．mDNSResponder が ppp 経由で取得したエントリを無視する仕様のためである．その理由は，mDNS の通信はあくまでローカルに限り，その情報を拡張した DNS サーバで集約し，その DNS サーバに対して TCP/IP でアクセスをすることで名前情報を得るというソリューション（dns-sd および Wide Area Bonjour）を Apple は推し進めているからである．

確かに，mDNS エントリとアドレス，ポートの拘束性を解かないこの方法は，グローバルアドレスを多用でき，イントラネット内部にまでそれを利用できる場合には問題は顕在化しない．しかし，VPN の利点の 1 つである「ローカル IP アドレスを利用するローカルネットワークへのアクセス」をしたときには，エントリテーブル内でのローカル IP アドレスの重複などの問題が生じる．NAT ではそれを避けるために，mDNS エントリのドメイン領域を使うことで NAT Traversal を可能にするためのソリューションを用意しているが，VPN への対応は，このソリューションでは不十分である．

また Mac OS X どうしで接続できる仮想イーサーネットインタフェースを用いる L2VPN は，代表的なものに vTun [22] と OpenVPN [23] があげられる．どちらも，tap [24] という仮想 Ethernet インタフェースドライバを利用して，仮想ネットワークを構築できるが，その両者とも Apple が用意している VPN と同様，mDNS エントリを通すことができない．

こちらも mDNSResponder（mDNS エントリの受信部）が仮想デバイスのエントリを無視する仕様が理由の 1 つである．加えて，mDNS のエントリを含む UDP パケットは TTL が 255 で送信されるが，受け取る際にも TTL がチェックされる．そして，TTL が 255，254 以外のときには無視される仕様になっており，リモートとローカルの 2 回ブリッジを通してしまうと TTL が 2 つ減少してしまい，そのエントリを受け取ることができなくなることもあげられる．そして最後の理由としては，公開されている tap ドライバが，IP マルチキャスト/IP ブロードキャストに対応していないということだ．

実際，mDNS のエントリを通す VPN は，一部の VPN 対応ルータに限られ，1 章の仮定は，Mac OS X どうしで接続した場合は，一部の有償ソリューションを除いて☆，ほとんど不可能である．

そこで，VPN との比較をするために本研究では「mDNSrelay」というツールを開発した．これは，ローカルで広報されている受信可能な IP マルチキャスト 224.0.0.251 のポート 5353 で流れるパケットを，リモートに TCP で転送して，再度，リモートで広報するツールである．本研究では，これと vTun を組み合わせることで，VPN でも RNSplicer と同じようにローカルネットワークサービスにリモートからアクセスする方法を作り出し，その性能と RNSplicer の性能を比較した．なお，このツールは Mac OS X 10.3.9 までは対応しているが，Mac OS X 10.4 では，同一のローカルネットワーク以外の mDNS エントリは無視するようになっており，このツールと VPN を組み合わせても実現できなくなってしまった．そこで評価では Mac OS X 10.3.9 を利用している．

具体的な評価環境としては，10Base-T のローカルネットワークに 2 台のマシンを用意し，別のセグメントのネットワークにあるマシンから，ローカルネットワークの片側のマシンに対して，vTun/RNSplicer でログインし，もう片側のマシンが提供しているローカルネットワークサービスに対するアクセスするという形式で比較した．

5.2 スループットの比較

スループットの比較としては，サービスタイプ「_http._tcp」を用い，4 KB〜8 MB までの HTML ファイルを用意して，そのダウンロード性能を httperf[25] を用いて数値化し比較した．なお，L2VPN，Tunneling with Service Discovery と構造上の比較をしたいため，vTun，RNSplicer の両者ともストリーム圧縮機能☆☆を用いていない．結果は，**図 8** のようになった．

この結果から，この条件では 512 KB までのスループット性能は vTun と比べて，RNSplicer のほうがスループット性能は劣るが，それ以降のスループット性能は勝っていることが分かった．これはソケット受信時のコストが vTun に比べて高いことが原因にあると思われる．しかし，このコストはアクセス時のイニシャルコストであるため，送信するデータが大きくなった場合そのコストが占める割合が減る．それに対して L2VPN では，すべての受送信パケットへのデータ操作が必要になるため，データが大きくなった際の転送性能の向上の幅が狭く，3,000 bps 程度で vTun が収束してゆくのに対して，RNSplicer では，4,500 bps 程度までの向上が認められた．この結果，1 セッションで小さいデータをやりとりする場合には，あまり効率的に動作しない点は，このソフトウェアに改善の余地があることが確認された．

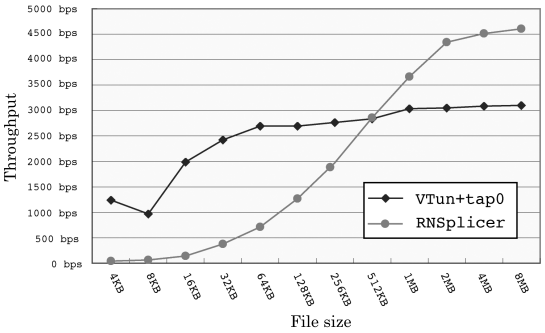

図 8 スループットの比較
Fig. 8 Comparison of throughput.

表 2 mDNS エントリに使用するパケットの転送量の比較
Table 2 Quantity of transfer which mDNS uses.

RNSplicer	vTun + tap0 + mDNSrelay
7.08 KB	102.52 KB

5.3 必要サービス以外の転送量

ここでは，vTun + tap0+mDNSrelay と RNSplicer で，同一サービス利用時に，どれだけリモートからクライアントにそのローカルサービス以外のデータパケットを転送しているかを計測した．具体的には，接続確立以降に，ローカルネットワークでサスペンドしていた 1 台の Mac OS X マシンを復帰させ，再び，サスペンドに移行するまでに，どのくらいリモートネットワークにパケットが転送されるかを比較したものである．サスペンドしていたマシンにおいては，ユーザが意図して別マシンに対してサービスを提供したものは 1 つだけであり，それ以外は Mac OS X のデフォルトのままである．このとき，ローカルにおいてサービスが提供されていない状態から，サービスの提供を開始し，リモートでサービスが発見され，その後，サービスの提供を停止し，リモートでサービスが利用できなくなる過程を確認できたときに，それまでにどれだけのパケット転送があったかを比較した（**表 2**）．

この結果が示すものは，Ethernet を偽装する VPN

☆ RendezvousVPN という 1 サービス 8 ドルという価格設定の有償ソリューションは存在する．構造が不明だが，mDNS をローカルで再送信することで VPN 上でも利用できるようにしているものだと思われる．http://sw.dig2way.net/

☆☆ vTun では LZO 圧縮および ZLib 圧縮をサポートしている．また RNSplicer は ssh との接続時の ZLib 圧縮をサポートしている．

表 3 サービスが起動してからリモートでサービスが発見され，利用可能になるまでの時間
Table 3 Comparison of time which a service turns from invisibility to visible.

RNSplicer	vTun + tap0 + mDNSrelay
1.28 sec	0.27 sec

の場合，実際のアプリケーション接続がなかった場合にも，サービスディスカバリの中で，必要とする以外のサービス情報も転送している．それに対して，RNSplicer は指定されたサービスタイプの更新情報のやりとりをするだけで，それはごくわずかであることを示している．

5.4 サービス情報更新までにかかる時間

上記の同一の条件で，ローカルでサービスを起動させてから，リモートでそのサービスが利用可能になるまでの時間を計測した（**表 3**）．

結果は RNSplicer のほうが遅く，それは RNSplicer の欠点でもある．これは，サービス情報をローカルで一度テーブルに収めて処理をするために，トランザクションの問題が起こり，それを回避するためにテーブルの更新タイミングに冗長性を持たせていることに起因する．なお，この問題は，マシンの性能によって，チューニング可能な問題ではあるが，一度テーブルに収めている以上，VPN を利用した場合を上回ったスピードで情報を伝搬させることはできない．

5.5 VPN との性能比較の考察

このソフトウェアは mDNS エントリを利用することよって，サービスの実体と IP アドレス/ポートの拘束性が解かれることを利用することで，VPN では IP 層で行っているリレー処理をアプリケーション層で行うことが可能になることを示しており，その際に心配されるスループットの減少もそれほど大きくないことを示している．また必要最低限のサービス情報を転送することで無駄な転送もない．サービスの取得までの時間は，それでもチューニングの余地はあるが，実用的なスピードであると判断している．

ただしながら現時点で，Mac OS X どうしで mDNS のエントリをインターネットを介して中継し，トンネリングまで自動的に用意する単一のソフトウェアは存在しないため，正確な意味合いにおいての比較対象がない．また，実装に Java を使用したため，Java では構造上 VPN の実装は難しく，プラットフォームの違いがあり，構造的な優位性の正確な判断ができないという問題があった．

そこで今後の課題とはなるが，現時点での実装をプロトタイプと位置づけ，このモデルを基に C 言語で RNSplicer を書き換える必要があることを認識している．

5.6 操作方法の比較と考察

たとえば，前述の vTun では最低限の設定でも，初めて接続するまでには以下のような操作が必要である．① サーバの設定ファイルを作成する．その内容には受付ポート，パスワード（pass），接続種別（type），転送プロトコル（proto），接続確立時の動作（up の内容）を書き込む．② ファイアウォールの設定を変更し受付ポートを開く．③ 管理者権限で仮想ドライバのカーネル拡張をロードする（kextload）．④ 管理者権限でサーバを起動する（/usr/local/sbin/vtun -n -s -f foo.server.conf）．⑤ クライアントの設定ファイルを作成する．その内容にはパスワード（pass），接続維持設定（persist），接続確立時の動作（up の内容）を書き込む．⑥ 管理者権限で仮想ドライバのカーネル拡張をロードする（kextload）．⑦ 管理者権限でクライアントを起動する（/usr/local/sbin/vtun -n -f foo.client.conf MyConfig ServerAddress）．

これに対して RNSplicer の操作はできるだけ簡単な実装を目指しており，管理者権限を必要とせずユーザモードのみで実行できるようにしている．具体的な操作手順は，① リモートアクセス可能なユーザアカウントをサーバ側で用意する☆，② サービスを提供したいマシンに RNSplicer を入れ，起動し [Start Server] ボタンを押しておく，③ RNSplicer をクライアントマシンで起動し，先のサービス提供マシンのアドレスを入れる（ローカルアドレスでかまわない），④ RNSplicer のサーバが起動しているマシンと同一ネットワークにある sshd が動作するホストのアドレスを入れる，⑤ ssh のユーザ名，パスワード，そしてサーバに要求する相手側のローカルサービスのサービスタイプを入力する（**図 9**），⑥ [Connect] ボタンを押す，というものである．とくに問題がなければ，この手順でクライアント側のローカルネットワークで，接続先のローカルネットワークのサービス情報が mDNS で中継される．

単純な比較はできないが，vTun の設定ファイルには，接続確立時の動作のために ifconfig を使ったインタフェースのアドレスの設定や，route コマンドを使ったルーティングの設定を行わなくてはいけないなど手間も多く，煩雑さという面では vTun のほうが高いといえる．また，VPN ではできないことだが，同一のサービスの選択的な提供も可能であり，こちらも画面上に表示されたテーブルでサービスをクライアントに

☆「システム環境設定」→「アカウント」から設定できる．

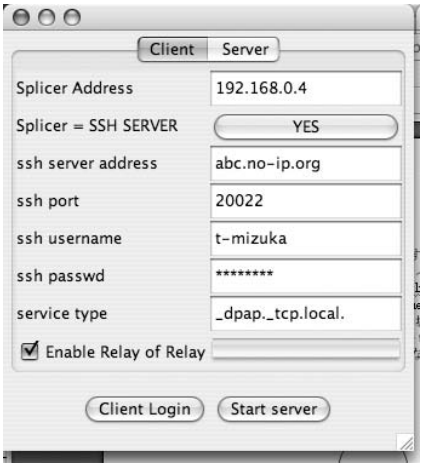

図 9 操作画面（クライアント）
Fig. 9 A screenshot of RNSplicer.

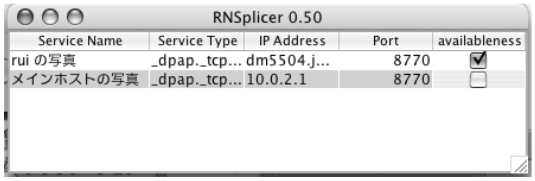

図 10 サーバでの提供サービスの選択画面
Fig. 10 A screenshot of RNSplicer.

提供する/しないを決めることができる（**図 10**）.

5.7 セキュリティ面での比較と考察

本来，このモデルは「分散トンネリング＋サービスディスカバリ告知/受信＋エントリのデータ管理」で成立するモデルであり，本実装において，セキュリティのためだけに SSH2 を利用している．

ここで実現する認証セキュリティは vTun がユーザアカウントの管理をせずパスワードのみでアクセスできるモードがデフォルトであることと比較すれば，安全性は低くなく，それゆえに最低限の認証セキュリティは確保しているとはいえる．しかし，OpenVPN のように大きい共用キーファイルを生成して，それをリモートとローカルの両方に必要とするやり方のほうがパスワードアタックを防げることは確かであり，今後の課題としてその認証方式の導入を考えている．

経路上のセキュリティという面でも，RNSplicer は SSH2 TCP PortForwarding を利用している．分散プロキシトンネル間でのバッファの暗号化手法についても今のところ ssh に任せているが，IPsec や SSL を用いることで分散プロキシトンネル間の安全性を高める形に変更することで，選択的に暗号アルゴリズムを変更できるようにすることにより，絶対にスニッフされないようにする工夫もこれから必要である．

また，実装上の議論の中では，また，ssh Daemon を内蔵する実装も検討したが，これについては，パスワードをわざと設定しなかったり，だれもが勝手に使える共通パスワードをはやらせたりするといった使い方をユーザがしかねないという懸念がある．そのため，ssh Daemon についてはあえて内蔵はせず，アカウント管理は sshd の（つまりオペレーティングシステムの）アカウント管理に任せることによるほうが，安全性が高いと判断している．

なお，このソフトウェアは，リモートのローカルネットワークにアクセス先のローカルネットワークを連結する．それゆえにブロックアップを怠るなどすれば，同じローカルネットワークにいる第 3 者にローカルサービスにアクセスされてしまう可能性が否定できない．それらについては公開するうえで，Web サイトで警告を促すことで対策していきたい．

6. 今後の課題

今後の課題としては以下があげられ，本研究では，これらの解決をしてゆくつもりである．

複数サービスタイプの中継 現時点での実装では，指定して中継できるサービスタイプは 1 つであり，複数にすると利便性が上がる．

複数のポートを利用するサービスへの対応 CVS や POP3 と SMTP など複数ポート，プロトコルを利用するローカルサービスへの対応も図りたい．これは複数サービスタイプの中継と一緒に考えていく．

well-known ポート well-known ポートを利用するアプリケーションは，リモートで再送信されたエントリを受け取ってもポート番号を無視してしまい利用できないものがあった．IP Firewall のフォワード，ディバート機能などと連携することで対応させたい．

双方向通信への対応 このモデルでは，サービスに対してアクセスはトンネリングの出口から行われる．ゆえに，アクセスしてきたアドレスに対して送信（返信ではない）をするチャットなどのアプリケーションでは出口までしかパケットが届かない．両方向にログインしたときには，それを何らかの形で解消されるようにする．

グローバルサービス このソフトウェアではトンネリングでの出口での情報収集によって，サービスに対してアクセスする際に必要な情報を「RNSplicer のアドレス」「ssh のアドレス」および「ユーザアカウント」まで縮めている．しかし，これもユーザアカウントを入れればアクセス可能な RNSplicer ア

ドレス，ssh アドレスの組合せの集合を返す DNS のようなグローバルサービスを用意できれば，さらなる自動化をすることができるため，それを開発したい．

7. まとめ

VPN の利便性の根幹は，本来，サービスディスカバリを利用できることではなく，離れた位置のホストでもローカルと同じ設定で動作させることができることにある．その魅力は，移動するホストは，いちいち設定を変更しなくてもいいことであり，管理者にとってはサービスの告知が一度でいいことにある．その意味においては，サービスディスカバリによるローカルサービスの告知が，メールの設定，社内共有リソースの告知といった用途にまで使われるようにならない限り，つまり IP アドレスやサーバ名，ポート番号などをユーザは知らなくても利用できる時代がこない限り，その VPN の利便性の根幹が損なわれることはない．また逆に，サービスディスカバリを用いずにユーザにアドレスやポート番号を入力させるアプリケーションにおいては，この RNSplicer には利便性がない．

しかしながら，サービスディスカバリを利用するものとしないものが中途半端に混在しており，必ずしも VPN でサービスディスカバリが利用できないといった現状では，サービスディスカバリの普及にともない，このソフトウェアモデルの存在価値も向上する．

そのうえでの，現時点での本研究の意義は，VPN の代替として，このソリューションを提案するのではなく，導入コストが低く，簡易に，そしてユーザが理解しやすく，ssh と VPN の間を埋める選択肢としてのソフトウェアモデルを提案することにある．

本研究では，ネットワークポリシを変更せずに，1 つのローカルネットワークサービスへのリモートアクセスを，ローカルネットワークどうしの接続に変えてしまうモデルを提案した．このモデルでは必要以上の中継をせず，柔軟な接続が可能で，かつサーバあたりの通信転送量は少なくて済む．また，このモデルを実装したソフトウェアを使えば，実際，物理上に組まれたローカルネットワーク以外のローカルネットワークのサービスを，ネットワークインタフェースを追加することなく利用でき，マルチアクセス，多段連結なども可能である．このことにより，以下のような使用をイメージすることができる．

- ネットワークポリシの違う 2 つの会社間の営業所どうしの簡易接続（とくに互いのローカルアドレス空間が同じで，アドレスが重複する場合などに有用である）
- 会社から自宅/別荘など複数の場所にある写真ライブラリの同時利用
- 片方向アクセスが必要な場合の接続（親会社は子会社のリソースにアクセスできるが，子会社は親会社のリソースにはアクセスさせたくない場合など）

また，RNSplicer には半リアルタイムでサービス情報が伝搬される仕組みを内蔵している．これはローカルサービスの提供状態のスナップショットの比較を随時行っていることを意味する．各マシンにエコーサービスとサービスディスカバリのレジストラを組み合わせたソフトをマシン起動時に起動するようにしておけば，終了時/非稼働時にはそのスナップショットから削除され，スナップショットの比較で非稼働マシンを検出できる．このことから，フォールトトレランス分野における非稼働マシンのリモートディテクトなどにも応用することも可能だ．

また Mac OS X 以外にも，GNOME/KDE といった Linux，FreeBSD のデスクトップ GUI ソフトウェアも（mDNS を含む）Zeroconf への対応をアナウンスしており，今後も mDNS 対応アプリケーションが増えてゆくことは間違いなく，これらに対して何も手を加えずに，リモートアクセス対応アプリケーションに変更できることには意味がある．サンプル実装としての RNSplicer は完成しており，修正 BSD ライセンスで 2005 年 2 月より公開している☆．この実装によって，このモデルの有用性が示され，さまざまなソフトウェアでの応用にされることを期待したい．

謝辞 本研究に対して，意見をいただいた北陸先端科学技術大学院大学篠田教授，篠田研究室の皆さん，片山教授/片山研究室の皆さん，落水研究室藤枝助手，ソフトバンクパブリッシング UNIXUSER 編集部の比嘉隆氏，Apple Computer の Dr. Stuart Cheshire 氏に謹んで感謝の意を表する．

参考文献

1) Egevang, K. and Francis, P.: The IP Network Address Translator (NAT), RFC2460 (May 1994).
2) Deering, S. (Cisco) and Hinden, R. (Nokia): Internet Protocol, Version 6 (IPv6) Specification, IETF, RFC2460 (Dec. 1998).
3) Saltzer, J.H., Reed, D.P. and Clark, D.D.: End-To-End Arguments in System Design, *Proc. ACM Trans. Computer Systems*, Vol.2,

☆ http://www.jaist.ac.jp/〜t-mizuka/rns/

No.4 (1984).
4) Ferguson, P. (Cisco), et al.: What is a VPN, *Proc. 5th Workshop on Open Signaling for ATM, Internet and Mobile Networks*, Toronto, October 5-6 (1998).
5) UPnP Forum. http://www.upnp.org/
6) Rosenberg, J., Weinberger, J., Huitema, C. and Mahy, R.: STUN — Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translaters (NATs), IETF, *RFC3489* (Mar. 2003).
7) Lonvick, C. (Cisco) (Ed.): SSH Transport Layer Protocol, IETF Internet-Draft, draft-ietf-secsh-transport-22.txt (Dec. 2004).
8) GNU HttpTunnel. http://www.nocrew.org/software/httptunnel.html
9) Francis, P. and Gummadi, R.: IPNL: A NAT-Extended Internet Architecture, *Proc. ACM SIGCOMM* (Aug. 2001).
10) Cheriton, D.R. and Gritter, M.: TRIAD: A New Next Generation Internet Architecture (Mar. 2000). http://www-dsg.stanford.edu/triad/triad.ps.gz
11) Gritter, M. and Cheriton, D.: An Architecture for Content Routing Support in the Internet, *Proc. 3rd UNENIX Symposium on Internet Technologies and Systems* (*USITS'01*) (Mar. 2001).
12) Ng, T.S.E., Stoica, I. and Zhang, H.: A Waypoint Service Approach to Connect Heterogeneous Internet Address Spaces, *Proc. USENIX Annual Technical Conference* (Jun. 2001).
13) Turanyi, Z. and Valko, A.: IPv4+4, *Proc. 10th International Conference on Network Protocols* (*ICNP2002*) (Nov. 2002).
14) DNS Service Discovery. http://www.dns-sd.org/
15) SoftEther 仮想イーサーネットシステム. http://www.softether.com/jp/
16) Goland, Y.Y. (Microsoft), Cai, T. (Microsoft), Leach, P. (Microsoft), Gu, Y. (Microsoft) and Albright, S. (HP): Simple Service Discovery Protocol/1.0 (Apr. 2000). draft-cai-ssdp-v1-03.txt. http://upnp.org/
17) Cheshire, S. (Apple) and Krochmal, M. (Apple): Multicast DNS (Feb. 2004). draft-cheshire-dnsext-multicastdns-04.txt. http://www.multicastdns.org/
18) The IETF Zeroconf Working Group: Zero Configuration Networking (Zeroconf). http://www.zeroconf.org/
19) Guttman, E. (Sun Microsystems): Attribute List Extension for the Service Location Protocol, IETF, RFC3059 (Feb. 2001).
20) Jini Network Technology. http://www.sun.com/software/jini/
21) Windows 2000 Server Windows Internet Naming Service (WINS) Overview. http://www.microsoft.com/windows2000/techinfo/howitworks/communications/nameadrmgmt/wins.asp
22) vTun (Virtual Tunnel): Maxim Krasnyansky. http://vtun.sourceforge.net/
23) OpenVPN: OpenVPN Solutions LLC. http://openvpn.net/
24) tun/tap driver for Mac OS X: Mattias Nissler. http://www-user.rhrk.uni-kl.de/~nissler/tuntap/
25) Mosberger, D. and Jin, T.: HTTPPERF: A tool for measuring web server performance, HP Labs Technical Report (1998).
26) Jsch (Java Secure Channel, BSD-Like Licence): JCraft Inc. http://www.jcraft.com/jsch/
27) JmDNS (LGPL). http://jmdns.sourceforge.net/

(平成 17 年 1 月 24 日受付)
(平成 17 年 4 月 27 日採録)

水上　智雄（学生会員）

1974 年生．1998 年法政大学法学部法律学科卒業．同年ソフトバンク（株）入社．『DOS/V magazine』編集部を経て，IT 関連書籍の企画，編集，執筆に従事．2004 年北陸先端科学技術大学院大学情報科学研究科博士前期課程入学．現在，ソフトバンクパブリッシング（株）を在籍休職中．

長　健二朗（正会員）

（株）インターネットイニシアティブ技術研究所主幹研究員．北陸先端科学技術大学院大学客員教授．1984 年神戸大学工学部卒業．同年キヤノン（株）入社．1993 年コーネル大学コンピュータサイエンス学科修士課程修了．1996 年（株）ソニーコンピュータサイエンス研究所入社．2001 年慶應義塾大学より博士号（政策・メディア）取得．2002 年より北陸先端科学技術大学院大学客員教授．2004 年より（株）インターネットイニシアティブ勤務．インターネットのトラフィック計測，QoS 通信，オペレーティングシステムのネットワーク機能の研究に従事．