マルチメディア通信と分散処理 96－8
（2000．2．10）

DNSとDHCPの統合による移動端末ホスト名固定方式の提案とその実装

横田 裕思†，木村 成伴‡，海老原 義彦‡
† 筑波大学 理工学研究科，‡ 筑波大学 電子・情報工学系

概要
TCP/IP ネットワークに接続する際に，接続に必要な IP 番号などの設定を取得するためのプロトコルとして DHCP があるが，IP 番号に対応するホスト名は DNS が管理するため，この方式では接続端末のホスト名を固定することはできなかった．この問題を解決するため，本論文では DNS と DHCP の統合方式を提案する．そして，接続端末から通知させたホスト名を割り当て IP 番号と対応させることで，そのホスト名を固定させる．最後に，本方式を実装し，その評価を行う．

A Proposal and its Implementation of Integration System of DNS and DHCP to Fix Host Name of Mobile Node

YOKOTA Hiroshi†, KIMURA Shigetomo‡ and EBIHARA Yoshihiko‡
† Master's Program in Science and Engineering, University of Tsukuba
‡ Institute of Information Sciences and Electronics, University of Tsukuba

Abstruct
DHCP provides a mobile host with network parameters such as IP numbers to connect up the TCP/IP network. Since a host name corresponded to the IP numbers is managed by DNS, the mobile host may be changed its host name whenever it connects to the network. To solve this problem, this paper proposes an integration system of DNS and DHCP. In this system, the mobile host can notify its own host name to the system to correspond with the assigned IP number. Finally, we implement the integration system and evaluate it.

1 はじめに

近年のインターネットの普及により，これを利用するユーザが急激に増大している．ユーザがネットワークを使用する際には IP アドレス等の設定が不可欠であるが，この作業にはネットワークの仕組みを熟知している必要がある．このため，これらの設定を自動的に行うプロトコル DHCP[1] が用いられる．しかし，DHCP が割り当てる IP アドレスに対応するホスト名は DNS[2] が管理しており，また DNS はホスト名を動的に変更することができないため，ユーザが DHCP により接続した端末に固有のホスト名を与えることができなかった．

これを解決するため，DNS にホスト名情報の動的更新をさせる DNS-UPDATE メッセージ [3] が規定されている．そして，これを用いて DNS と DHCP を連係させ，移動端末のホスト名を変更するプロトコルが提案されている [4]．この方法では DNS サーバ (bind) と DHCP サーバ (dhcpd) が別々に動作しており，これらが所持するホスト名情報を同期させ，これを二重に管理しなければならなかった．そこで本論文では，DNS と DHCP を統合し，これらの情報を一ヶ所にまとめて統合管理する移動端末ホスト名固定方式を提案する．

2 従来方式

本節では，[4] による移動端末ホスト名変更方式についての概要を述べる．この方式では図 1 のように DNS と DHCP のそれぞれがデータベースを所持している．

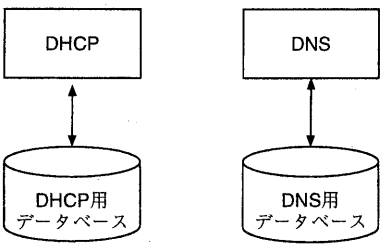

図 1: 従来方式

移動端末であるクライアントがネットワークに接続したとする．このとき，クライアントは図 2 のように，まず DHCP サーバに IP アドレスの要求を行う．これが成功すると，クライアントは DNS-UPDATE メッセージを用いて (もし必要ならば) 自分のホスト名を DNS サーバに通知し，DNS のデータベースを変更する．

クライアントがネットワークとの接続を切断する場合は，図 3 に示すように，アドレス設定とは逆の手順を行い，DNS に登録したホスト名を破棄し，続いて DHCP に登録した IP アドレスを解放する．

このように，従来方式ではアドレス管理データベースが 2 つあり，両者にクライアントが変更の要求を行うことでアドレス情報の同期を保つ．しかし，この同期に時間が要することから，障害が生じる可能性がある．

例えば，クライアントが DNS サーバに登録したホスト名の解放を行う前に，その接続を絶った場合，そのホスト名は DNS サーバに登録されたまま残ってしまう．これを解決するためには，セキュリティ的に問題があるが，DNS サーバに登録された状態でも上書き可能であるように設定するか，もしくは，DHCP サーバと同様に DNS サーバもホストの接続状況を監視する必要がある．

また，クライアントが DHCP サーバから IP アドレスを取得した直後に，このネットワークに接続している第三者のクライアントがこの IP アドレスに対するホスト名を DNS サーバに送信すると，そのホスト名が DNS のデータベースに登録されてしまう可能性がある．これは，DNS サーバは DHCP サーバのデータベースを直接参照できないため，該当 IP アドレスを取得したホストを判別できないことが原因である．

現在，ホストを特定するための認証機構が検討されており，これを用いることでこの現象を防ぐことは可能である．しかし，次節で提案する DNS と DHCP の統合システムにより，DNS と DHCP の双方のデータベースを一本化することで，この問題は容易に回避することができる．

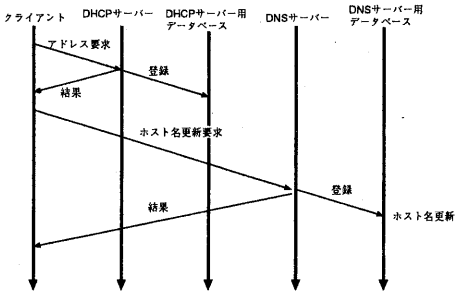

図 2: bind + dhcpd によるアドレス設定

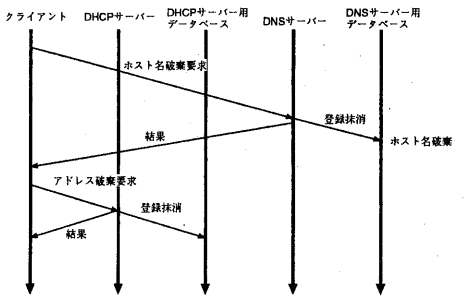

図 3: bind + dhcpd によるアドレス解放

3 提案方式

提案方式では，まず，図 4 に示すように DNS と DHCP を統合し，これらで用いられるデータベースを 1 つにする．これに基づき，本提案方式の動作手順は次のようになる．

まず，クライアントがネットワークに接続すると，図 5 に示すように，DHCP サーバはこれ

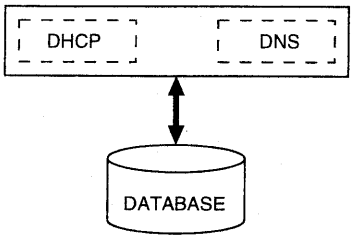

図 4: 提案方式

にIP アドレスを与え，これを統合データベースに登録する．これと同時に DNS サーバはクライアントの仮ホスト名を自動生成し，これをデータベースに登録してクライアントと IP アドレスの関係を対応づける．

更にクライアントが望めば，クライアントは DNS-UPDATE メッセージを用いてホスト名の変更を要求する．これに応じて，DNS サーバは要求されたホスト名をクライアントの IP アドレスに設定し直し，これをデータベースに登録する．もし要求がなかった場合は自動生成された名前がそのまま使われる．この操作は DHCP で得られたホスト情報 (MAC アドレスなど) が IP アドレス 登録直後に DNS サーバに渡るため，第三者が別のホスト名を登録することはできない．

クライアントがネットワークとの接続を切断する場合は，図 6 に示すように，IP アドレスを解放すると同時にホスト名が解放される．このため，クライアントがアドレスを解放せずに接続を切断した場合でも， DHCP によるホスト接続監視機能によって，登録された IP アドレスおよびホスト名はタイムアウト時間後に自動的に抹消される．

以上に基づき，次節では本提案方式を設計し，その実装を試みる．

4 実装

4.1 実装方式の検討

前節で述べた提案方式の実装方法として，次の 2 つが考えられる．

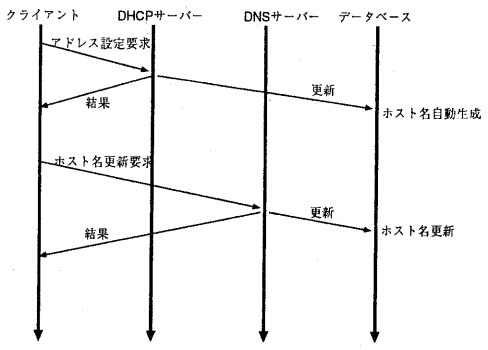

図 5: 提案方式によるアドレス設定

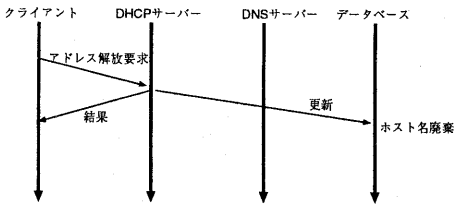

図 6: 提案方式によるアドレス解放

(1) 既存の DNS サーバと DHCP サーバをデータベースシステムでつなぎ，データベースを介して互いのデータを共有する (図 7).
(2) DNS と DHCP の機能を合わせ持つサーバを新規に開発し，データを共有する (図 8).

前者の方式について，現在良く使われている DNS サーバの ISC BIND と DHCP サーバの ISC DHCPD のソースコードを調べたところ，両者のデータベースの内部仕様が大きく異なっており，また，各サーバのデータの保持内容がかなり食い違っていることが分かった．このため，両サーバがデータベースのレコードに同時書き込みすることを調停するレコードロックの機能が複雑になる．以上の理由により，本論文では後者の方式を採用する．

4.2 提案方式の実装

以上の検討を基に，提案方式の実装を行った．本実装には Ruby を用いた．その理由は，Ruby にはプログラムの作成を補助する有用な機能が

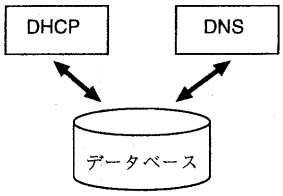

図 7: 既存システムの流用による設計

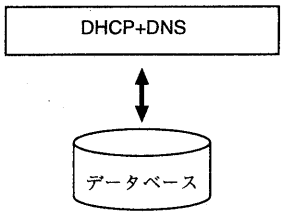

図 8: 新規に開発する設計

数多く用意されており，提案方式の本質とは関わりのない部分に実装の労力を費やす手間が省けると判断したことによる．

Ruby はマルチスレッド機能を持っているため，今回の実装では DNS と DHCP の 2 つのサーバ機能を 2 つのスレッドを用いて実装した．データベース機能はハッシュ(連想配列) を使った．また，それぞれのサーバ機能の通信プロトコルは DNS および DHCP プロトコルに従い，データベースの更新処理に必要なプロトコルのみ DNS-UPDATE プロトコルを採用した．本実装の構成を図 9 に示す．なお，プログラム規模はサーバが約 8k バイト，クライアントが約 2k バイトである．

4.3 実行例

以下に，前項で実装したシステムの実行例を示す．実験で用いた環境を図 10 に示す．

図に示す LAN はコンピュータ内部に構築した仮想 LAN である．そして，デフォルトで用意されているローカルループバック lo (IP アドレス 127.0.0.1) に実装したサーバを接続した．更に，疑似イーサネットインタフェース dummy0 (表面上はイーサネットインタフェースとして振

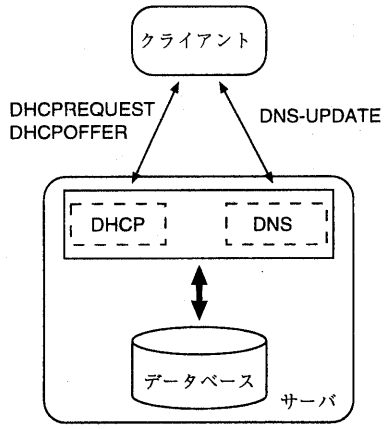

図 9: システム構成図

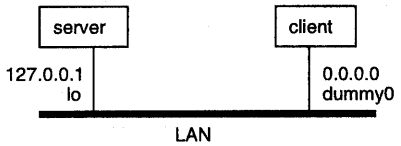

図 10: 仮想 LAN 環境

舞うループバックインタフェース) を設け，これをクライアントとする．

この環境を構築する操作例を図 11 に示す．なお，本システムは Linux 2.2.14 上で動作させている．最初の ifconfig コマンドと route コマンドで dummy0 インタフェースを有効にしている．2 回目の ifconfig コマンドでそのことが確認できる．また，最後の route コマンドではネットワーク状況が確認できる．この時点では，クライアントには IP アドレスが割り当てられていないため，そのアドレスは 0.0.0.0 となっている．また，そのデフォルトルートも 0.0.0.0 に設定されている．

ここから DHCP の操作によって IP アドレスの獲得が行われるが，クライアントはサーバのアドレスなどの情報を知らないため，サーバに対する通信はブロードキャストが用いられる．ところで，クライアントが接続直後に任意のアドレスに対してパケットを送出すると，そのクライアントのルーティングテーブルに情報がないので，そのパケットはそのデフォルトアドレ

```
# ifconfig dummy0 0.0.0.0 up
# route add default dev dummy0
# ifconfig
lo        Link encap:Local Loopback
          inet addr:127.0.0.1
dummy0    Link encap:Ethernet
          unspec addr:[NONE SET]
# route -n
Destination Genmask    Flags Iface
127.0.0.0   255.0.0.0 U     lo
0.0.0.0     0.0.0.0   UG    dummy0
```

図 11: 仮想 LAN の構築

ス 0.0.0.0 に対して送信される．通常，このアドレスは特殊な用途で用いられるため，このパケットがどのホストにも届くことはない．しかし，この疑似イーサネットインタフェースの実態はローカルループバックである．このため，デフォルトアドレス 0.0.0.0 に対するパケットをサーバが疑似的に取得することは可能である．すなわち，この仮想 LAN 環境はおけるルーティングテーブルのデフォルトルートを制御し，デフォルトルート 0.0.0.0 に対するパケットをネットワークデバイス lo に向ける．すると，接続直後にクライアントから発せられた全てのパケットは宛先如何に関わらず lo, すなわちサーバに到着する．今回の実験では実装を簡単にするため，ブロードキャストを用いずに，この仕組みを利用している．もちろん，この実装は本実験で用いた仮想 LAN 環境でのみ有効であり，通常は利用できない．

この環境でサーバとクライアントが通信した動作例を図 12 に示す．なお，読みやすさのため，一部の不要な表示はこの動作例から省いてある．最初の ifconfig コマンドと route コマンドでクライアントのネットワークインタフェース dummy0 の状況が示されおり，IP アドレスがまだ割り当てられていない状態であることが分かる．

次に，dnspnpc コマンドが実行されている．このコマンドは本実装により作成されたものであり，クライアントからの通信をサーバを接続するものである．これにより，サーバの DHCP が機能し，クライアントはサーバから IP アドレスを取得することができる．次の ifconfig コマンドにより，その IP アドレスは 192.168.0.1 であることが分かる．これと同時に，dnspnpc コマンドにより，クライアントはサーバに対して自分のホスト名 “foo.bar.com” の登録要求をサーバの DNS 機能に対して行う．これにより，IP アドレス 192.168.0.1 にホスト名 “foo.bar.com” が割り当てられてたことが，最後に実行された host コマンドによって分かる．

以上の結果から，実装システムにより，IP アドレスとホスト名が正しく設定できることが確認された．

```
client# ifconfig dummy0
dummy0    Link encap:Ethernet
          unspec addr:[NONE SET]
client# route -n
Destination Genmask    Flags Iface
127.0.0.0   255.0.0.0 U     lo
0.0.0.0     0.0.0.0   UG    dummy0
client# ./dnspnpc &
            ↑(クライアントの接続)
client# ifconfig dummy0
dummy0    Link encap:Ethernet
          inet addr:192.168.0.1
client# host foo.bar.com
foo.bar.com  A       192.168.0.1
```

図 12: 実行の様子

5 まとめ

本論文では，DNS と DHCP を統合することにより，移動端末が自分のホスト名を任意に登録することができるシステムを提案した．また，本システムを設計し，これを実装することで，提案方式で IP アドレスとホスト名が正し

く設定できることが確認された．

今後の課題としては，より現実に即した場面での試験を重ね，他の方式と比較検討することにより，提案方式のプロトコルやシステムをより実用的なものに改良していくことが挙げられる．

本論文のシステムは，ネットワークの設定をより簡単にすることを目標として提案したものである．この考えを押し進めるため，提案システムを更に発展させることを検討している．具体的には，サーバを完全になくし，クライアント同士の通信で全てを完結させるシステムを構成する予定である．

このシステムでは，ネットワーク接続に必要な情報は既にネットワークに接続されているクライアントが保持している．ネットワーク接続したクライアントは既接続のクライアントに自分が必要な情報 (IP アドレスやホスト名など) をブロードキャストメッセージを用いて自律的に問い合わせ，返事をもらう．それにより，他のクライアントに使われていない IP アドレスやホスト名を選択し，これを利用する．

DNS による名前解決の問い合わせに対しては，接続している各クライアントが自分のホスト名であるかどうかを確認し，該当するクライアントがあれば，そのクライアントが自発的に回答することでこれに対応する．ただし，既存のアプリケーションとの互換性をとるため，DNS エミュレーションの動作も検討する．

また，外部ネットワークとの通信を行うためには，ルータがそのために必要な情報を提供し，外部ネットワークの DNS 検索も代理で行うことで解決できる．しかし，ネットワークプリンタなど，既存の動作を変更できない装置に対しては，他のマシンが全ての通信を代行する必要がある．

参考文献

[1] Droms, R.: Dynamic Host Configuration Protocol, RFC2131 (1997).

[2] Mockapetris, P.: Domain Names — Implementation and Specification, RFC1035 (1987).

[3] Vixie, P., Thomson, S., Rekhter, Y. and Bound, J.: Dynamic Updates in the Domain Name System (DNS-UPDATE), RFC2136 (1997).

[4] Rekhter, Y. and Stapp, M.: INTERNET-DRAFT Interaction between DHCP and DNS, draft-ietf-dhc-dhcp-dns-10.txt (1999).