モバイル端末からのIPv6接続実験と評価

広瀬　雄二*

概　要

次世代インターネット接続プロトコルとしてIPv6が声高に叫ばれるようになって久しい。常時接続の固定ホストに限れば、既に商用ネットワークサービスでのIPv6接続は実用段階に入っている。いっぽう非固定であるモバイル端末からのIPv6接続は技術的に確立されているものの未だ実験段階と呼べるまでには至っていない。本稿ではDTCPを用いたIPv6トンネリング接続の適用範囲を広げる拡張を実装した上で、実際に移動端末からの公衆ダイヤルアップ回線を利用したIPv6トンネリング接続の実験を行ないその可用性を確認した。

An Experiment and Evaluation of IPv6-Connection from the Mobile terminal

HIROSE, Yuuji

Tohoku University of Community Service and Science

Abstract

It has been years since the necessity of IPv6 Internet connection has emphasized. Now, we can connect to the IPv6 bakbone via Internet Service Provider (ISP). However, these IPv6 connection services are practical only for static network terminals today. Although the method of IPv6 connection from mobile terminal has been established, it is still not applicable widely.

In this paper, we implemented the extension of authentication database model to the DTCP server/client system and experimented on the extended system by establishing the connection from the mobile terminal via public dial-up line.

＊東北公益文科大学　yuuji@koeki-u.ac.jp

1．背　　景

現在インターネットをむすぶプロトコルとして主流となっているIPv4[1]はアドレス空間の大きさが不十分で、間もなく枯渇すると言われている。枯渇問題を含め、現在抱えている問題を解決するため、IPv6[2]が次世代プロトコルとして設計され、既に実用段階に入っている。とはいえ、現時点でIPv6によるインターネット接続を利用している比率はまだ小さい。今後の普及をさらに進めるためには、現在IPv4を利用しておこなっているインターネット接続形態の全てをカバーしていく必要がある。

現時点でIPv6を使う場合、常時接続、もしくは固定端末での利用が一般的である。本来IPv6は接続端末が移動しても移動先のネットワークに簡単に繋げる設計になっているが、それは移動先のネットワークがIPv6ネイティブ接続であることが求められる。そこで本研究では、移動先での接続手段としてIPv4しか確保できない現状を踏まえ、IPv4でのダイヤルアップ接続を利用したIPv6ネットワークトンネリングをおこなうシステムを構築・運用し検証した。

2．モバイル端末のIPv6利用

現在では多くのインターネット接続業者（以下ISP）がダイヤルアップサービスを供給している。しかしながら現状ではいずれのISPもダイヤルアップサービスはIPv4のみである。この状況下で、IPv6での接続性を確保するために、IPv4ダイヤルアップ接続とDTCP［2］を組み合わせる。

2．1　DTCPの応用

DTCP[3]は、2点間のトンネル接続を確立するときの認証プロトコルである。

---

[1]Internet Protocol Version 4
[2]Internet Protocol Version 6
[3]Dynamic Tunnel Configuration Protocol

プロトコル自体はIPv6トンネル接続のためだけに限らず、IPv4トンネルなどにも応用できる枠組みとなっている（図1）。

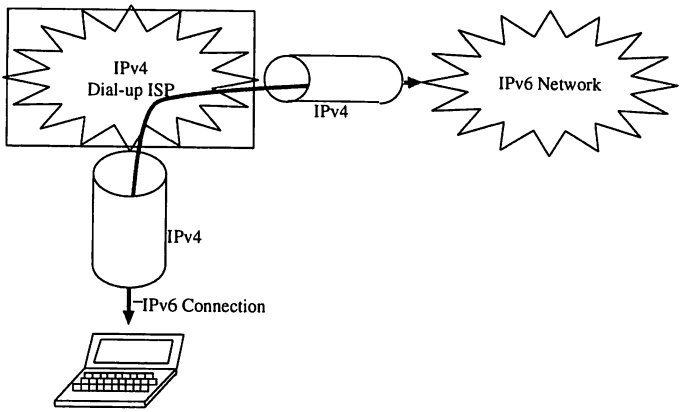

図1　IPv4 ISPを経由したIPv6トンネル接続

本研究ではDTCPによるIPv6トンネル確立のための実装であるdtcps. rb, dtcpc. rb [3] をベースに改良を加え、実験環境に即した認証データベース・認証方式を取るように変更した。これを利用しダイヤルアップ環境からIPv6接続を確立し、LAN外部から自由にアクセスできるようにした。このLANはIPv4的にはほぼ全てがIPv4プライベートアドレス空間に位置するので外部からは直接アクセスできない。しかし、このLANに同時に振ったIPv6アドレス空間を利用すると外部からのアクセスが可能となる（図2）。

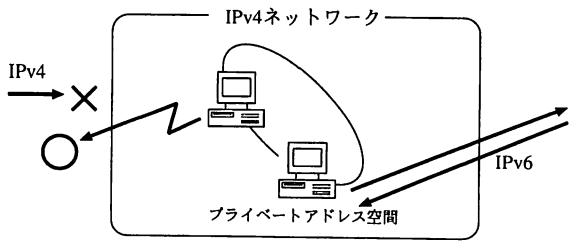

図2　IPv4で構成されたプライベートネットワーク

2．2　認証方法の拡張

DTCPで利用する認証方式は、APOP [4] と同様のChallenge and Response方式を用いたものである。実際、実装のひとつである dtcps. rb, dtcpc. rb では認証情報をAPOPの実装のひとつであるQpopper [5] の管理下にあるデータベースから引き出すようになっている。

本研究で利用しているネットワークでは、APOP実装としてワシントン大学によるIMAPD実装 [6] に、筆者らが独自のAPOP認証データベース管理方法を追加実装したUW-IMAPD Extensions [7]（以下imapext）を利用している。DTCPサーバにおいて、imapextの認証データベースを利用する拡張をおこなった。これにより得られる利点はimapext自体の利点と同じく以下のものとなる。

・パスワードをユーザー自身の権限（UID）でおこなえる。

・パスワード情報の保存ファイルの形式変更が容易である。

imapextでは、各ユーザの認証パスワード情報をホームディレクトリにある .apop ファイルに格納することになっており、ここからパスワードを取り出すために /usr/local/sbin/deapop コマンドを利用するようになっている[4]。本実験では dtcps. rb, dtcpc. rb のサーバ側の実装となる dtcps. rb に改良を施し、.apop を認証データベースファイルとして利用するようにした。この改良点は本稿末の付録としてまとめた。修正リストはGNU diffによるunified diff形式となっている。

[4]いずれも既定値であり、各パラメータは変更可能。

3. 実験環境

本実験に用いたネットワーク、およびホストの概略は図3のようになっている。このうち、接続の両端とゲートウェイとなるホストはIPv6スタックを実装したシステムで動作している必要がある。各ホストの主要環境は以下の通りである。

Mobile Client	CPU	Intel Pentium III 700MHz
	OS	NetBSD 1.6
Gateway	CPU	AMD Athlon 900MHz
	OS	FreeBSD 4
Internal Server	CPU	Pentium III 1000MHz
	OS	NetBSD 1.5ZB

また、Mobile ClientからGatewayホストへのIPv4接続を確立するための、基本ダイヤルアップ接続はDDI Pocket[5]社によるAir H" サービスを利用した。

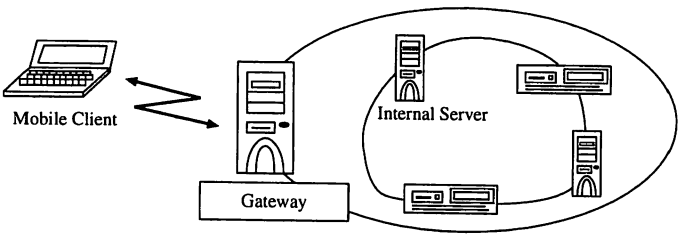

図3　接続実験環境の概略

[5] http://www.ddipocket.co.jp/

4. 接続実験

4. 1 実験手順

本実験で構築したシステムを利用した、モバイル端末からのIPv6トンネル接続は以下の手順による。

1. DTCPサーバ（Gatewayホスト）でdtcpsを起動する（常時起動）
2. クライアント（Mobile Client）が、ダイヤルアップ接続（IPv4）をおこなう
3. クライアントが、dtcpc を起動しトンネルを確立する

具体的な手順は以下の通り。

1. dtcps の起動（Gateway）

   Gatewayホストで以下のコマンドラインを起動する。

```
# ruby dtcps -i 'gif[1-9][0-9]' -d 2001:240:34:40f0::
```

   （これを自動化するスクリプト start. dtcps を付録に掲載）

2. ダイヤルアップ（Mobile Client）

   クライアントシステムに導入されているダイヤルアップ用のアプリケーション（任意）を利用してIPv4接続をおこなう。

3. トンネル確立（Mobile Client）

   クライアントホストで以下のコマンドラインを起動する。

```
# ruby dtcpc -t host gateway-host.koeki-u.ac.jp
```

   （これを自動化するスクリプト v6connect を付録に掲載）

4．2　実験結果

Mobile Clientがトンネル確立に成功すると以下のメッセージが出力される。

```
logging in to 211.120.119.69 port 20200
tunnel to 211.120.119.69 established.
add net default: gateway fe80::203:47ff:fe8a:9035%gif0
default route was configured.
true
```

これにより、Mobile ClientがIPv6で内部ネットワークに繋がった。確認のため、ネットワーク内部にあるInternal Server（**図3**）に対して **ping6** 導通試験を行なった。

```
% ping6 -c 5 Internal-Server
PING6(56=40+8+8 bytes) 2001:240:34:40f0::2e --> 2001:240:34:4010::1
16 bytes from 2001:240:34:4010::1, icmp_seq=0 hlim=63 time=304.979 ms
16 bytes from 2001:240:34:4010::1, icmp_seq=1 hlim=63 time=824.963 ms
16 bytes from 2001:240:34:4010::1, icmp_seq=2 hlim=63 time=1299.68 ms
16 bytes from 2001:240:34:4010::1, icmp_seq=3 hlim=63 time=1289.94 ms
16 bytes from 2001:240:34:4010::1, icmp_seq=4 hlim=63 time=359.675 ms

--- Internal-Server ping6 statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/std-dev = 304.979/815.849/1299.682/430.820 ms
```

このように、導通が確認できた。

5．検　　討

今回の実験によりダイアルアップサービスを用いた一般的なIPv4接続を利用した場合でも問題なくIPv6ネットワークへの接続が確保できることが確認できた。また、認証情報データベースをimapextと共通化したことにより、POPプログラムにimapextを利用しているサイトのユーザがIPv6トンネリング接続を行なうことが容易になった。

これらの結果をふまえ今後は、より設定の容易なIPv6接続環境を提供するこ

とが課題である。

なお、今回作成した認証部分の拡張を加えたDTCP実装は、anonymous cvsで配付しており、

% cvs -d : pserver : anonymous @ yatex. org : / cvs co -r dot-apop dtcp

により入手可能となっている。

参考文献

[1] KAME Project ; http://www.kame.net/

[2] Peter R. Tattam, Trumpet Software International Pty Ltd. ; Yet another Dynamic Tunnel Configuration Prototocol ;
http://jazz-1.trumpet.com.au/ipv6-draft/dtcp-draft-prt-13-may-1999.htm

[3] 梅本　肇；ダイナミックトンネル；
http://www.imasy.or.jp/~ume/published/dtcp/

[4] IETF RFC 1939 ; Post Office Protocol-Version 3 ; May 1996 ;
http://jazz-1.trumpet.com.au/ipv6-draft/dtcp-draft-prt-13-may-1999.htm

[5] QUALCOMM ; Qpopper ;
http://www.eudora.com/qpopper/

[6] Univ. of Washington ; IMAP Toolkit Environment ;
http://www.washington.edu/imap/

[7] 広瀬雄二；UW-IMAPD Extensions ;
http://www.gentei.org/~yuuji/software/imapext/

付　録

DTCPサーバプログラム　dtcps. rb に対する修正点（Unified diff)。

リスト 1 dtcps.rb.diff

```
RCS file: /cvs/dtcp/dtcps.rb,v
retrieving revision 1.1.1.1
retrieving revision 1.1.1.1.2.1
diff -a -u -r1.1.1.1 -r1.1.1.1.2.1
--- dtcps.rb 17 Mar 2003 14:04:16 -0000 1.1.1.1
+++ dtcps.rb 17 Mar 2003 17:05:20 -0000 1.1.1.1.2.1
@@ -265,7 +265,7 @@
   begin
     open(ROUTETABLE) do |f|
       f.each_line do |l|
- l.chop!.sub!('\s*#.*$', '')
+ l.chop!.sub!('\s*#.*$', '') #'
   next if l =~ /^\s*$/o
   if l =~ /^#{user}\s+(.*)$/
     prefixes = $1
@@ -513,7 +513,7 @@
     end
     user = t[1]
     type = (t[3] == 'tunnelroute') ? 'network' : t[3]
-   pass = getpopauth(user)
+   pass = ($dot_apop ? getapopauth(user) : getpopauth(user))
     if pass == nil
       logmsg("client #{s} has no password in database for #{user}\n")
       debugmsg("#{s}: sent <-ERR authentication failed.>\n")
@@ -663,6 +663,48 @@
   return p
 end

+# get ~user/.apop
+DEAPOP = "/usr/local/sbin/deapop"
+def getapopauth(user)
+  pw = Etc.getpwnam(user)
+  if pw == nil
+    debugmsg("no user named #{user}\n")
+    return nil
+  end
+  origuid = Process.euid
+  # XXX begin seteuid(pop)
+  Process.euid = pw[2]
+  if (homedir = pw.dir) == nil
+    debugmsg("no home dir for #{user}\n")
+    return nil
+  end
+  apopfile = homedir+"/"+".apop"
+  if (st = File.stat(apopfile)) == nil
+    debugmsg(".apop not found for #{user}\n")
+    return nil
+  end
```

```
49  +  if st.mode&0066 != 0
50  +    debugmsg(".apop of #{user} readable for others\n")
51  +    return nil
52  +  end
53  +  #p Process.euid
54  +
55  +  word = nil
56  +  begin
57  +    open("| #{DEAPOP} #{apopfile}", "r") do |apop|
58  +      word = apop.gets.chomp!
59  +    end
60  +  rescue
61  +    debugmsg("Cannot exec #{DEAPOP}\n")
62  +    return nil
63  +  end
64  +  Process.euid = origuid
65  +  # p Process.euid
66  +  debugmsg("ok, relevant password database item found\n")
67  +  return word
68  +end
69  +
70  +
71   #---------------------------------------------------------------
72
73   port = 20200
74  @@ -672,7 +714,7 @@
75   $prefix = nil
76   $network_with_peeraddr = nil
77
78  -if !getopts('acdDo', 'g:', 'i:', 'p:')
79  +if !getopts('aAcdDo', 'g:', 'i:', 'p:')
80     usage()
81     exit 0
82   end
83  @@ -684,6 +726,7 @@
84   $tunif = $OPT_i if $OPT_i
85   $create_only = $OPT_o
86   port = $OPT_p if $OPT_p
87  +$dot_apop = $OPT_A
88
89   case ARGV.length
90   when 0
```

DTCPクライアントプログラム　dtcpc. rb に対する修正点(同じくUnified diff)。

リスト 2 dtcpc.rb.diff

```
 1  RCS file: /cvs/dtcp/dtcpc.rb,v
 2  retrieving revision 1.1.1.1
 3  retrieving revision 1.1.1.1.2.6
 4  diff -a -u -r1.1.1.1 -r1.1.1.1.2.6
 5  --- dtcpc.rb 17 Mar 2003 14:04:16 -0000 1.1.1.1
 6  +++ dtcpc.rb 17 Mar 2003 17:44:04 -0000 1.1.1.1.2.6
 7  @@ -96,7 +96,7 @@
 8   #ROUTE_METHOD = ROUTE_GATEWAY
 9
10   def usage()
11  -  $stderr.print "usage: #{File.basename($0)} [-cdDln] [-i if] [-p port] [-t
tuntype] [-u username] [-A addr] [-R dest] [-P prefix-delegation] server\n"
12  +  $stderr.print "usage: #{File.basename($0)} [-cdDln] [-i if] [-p port] [-t
tuntype] [-u username] [-A addr] [-R dest] [-P prefix-delegation] [-s sleeptime]
server\n"
13   end
14
15   class PDInfo
16  @@ -648,7 +648,7 @@
17
18     private
19
20  -  def initialize(dst, port, username, password, tuntype, behind_nat)
21  +  def initialize(dst, port, username, password, tuntype, behind_nat, sleepti
22       @dst = dst
23       @port = port
24       @username = username
25  @@ -656,6 +656,7 @@
26       @tuntype = tuntype
27       @behind_nat = behind_nat
28       @tunnel = nil
29  +    @sleeptime = (sleeptime || 60).to_i
30     end
31
32     def connect_to(dst, port)
33  @@ -706,8 +707,8 @@
34     def keep_alive(sock)
35       begin
36         while TRUE
37  - debugmsg("sleep(60)\n")
38  - sleep 60
39  + debugmsg("sleep(#{@sleeptime})\n")
40  + sleep @sleeptime
41    sock.print "ping\r\n"
42    debugmsg(">>ping\n")
43    t = select([sock], [], [sock], TIMEOUT)
44  @@ -753,7 +754,7 @@
45   # end
46   # exit 0
47
48  -if !getopts('acdDlno', 'A:', 'f:', 'i:', 'p:', 'P:', 'r:', 'R:', 't:', 'u:')
49  +if !getopts('acdDlno', 'A:', 'f:', 'i:', 'p:', 'P:', 'r:', 'R:', 't:', 'u:',
```

```
's:')
50      usage()
51      exit 0
52    end
53  @@ -777,6 +778,7 @@
54    static_routes = $OPT_R if $OPT_R
55    tuntype = $OPT_t if $OPT_t
56    username = $OPT_u if $OPT_u
57  +sleeptime = $OPT_s
58    dst = ARGV[0]
59
60    trap("SIGTERM", "EXIT")
61  @@ -803,7 +805,7 @@
62        route = Route.new(route_type, static_routes)
63        pd = PrefixDelegation.new(prefix_delegation.split(/\s*,\s*/),
64           rtadvd_disable)
65  -     dtcpc = DTCPClient.new(dst, port, username, password, tuntype, behind_nat
66  +     dtcpc = DTCPClient.new(dst, port, username, password, tuntype, behind_nat
sleeptime)
67        while TRUE
68          interrupt = nil
69          begin
```

DTCPサーバスタートスクリプト　Gatewayホストで起動。

リスト 3 start.dtcps

```
1  #!/bin/sh
2  for i in gif10 gif11 gif12 gif13 gif14 gif15 gif16 gif17 gif18 gif19
3  do
4    ifconfig $i create
5  done
6  ruby dtcps -i 'gif[1-9][0-9]' -d 2001:240:34:40f0::
```

DTCPクライアントスタートスクリプト　Mobile Clientで起動

リスト 4 v6connect

```
1  #!/bin/sh
2  ruby dtcpc -i gif0 -t host gateway-host.koeki-u.ac.jp
```