March 2013
Geoff Huston

Literally IPv6

As many who have worked with computer software would attest, software bugs come in many strange forms. This month I'd like to relate a recent experience I've had with one such bug that pulls together aspects of IPv6 standard specifications and interoperability.

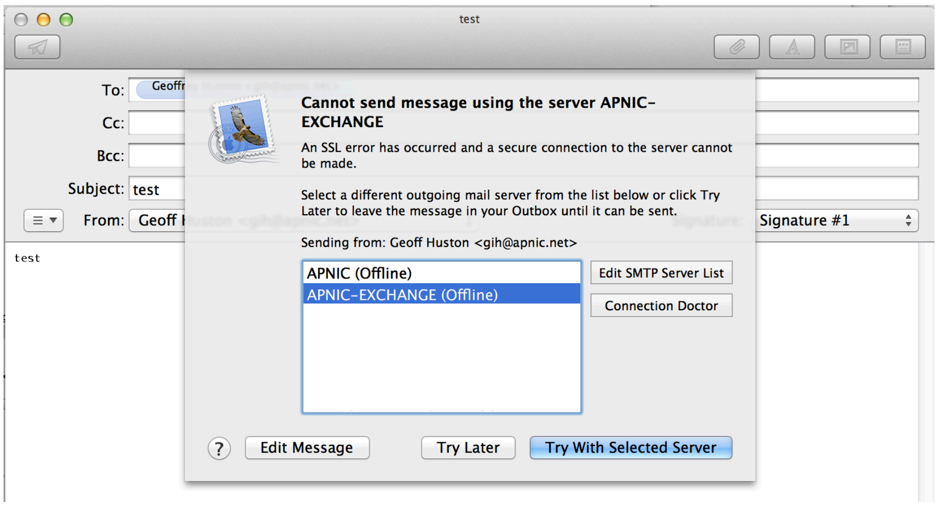
At APNIC we have changed our mail server configuration in the past few months, and are now using a mail server from Microsoft, namely the Exchange Server. These days for mail I use a Mac laptop, and use Apple's Mail application for my mail. To collect my mail I use IMAP, and to send my mail I use STMP, both standard protocols. These days security can't be ignored so we use TLS for channel encryption. All pretty standard stuff.

There are underlying standards that define the operation of these protocols, and the Microsoft and Apple software in question has been deployed on millions of systems, so its completely expected to see that all of this simply Just Works.

Well not quite.

It works most of the time. But sometimes things just don't work. Seemingly at random my efforts to send mail just fail.

Here's an example



Subject: test

Signature #1

test

Cannot send message using the server APNIC-EXCHANGE

An SSL error has occurred and a secure connection to the server cannot be made.

Select a different outgoing mail server from the list below or click Try Later to leave the message in your Outbox until it can be sent.

Sending from: Geoff Huston <gih@apnic.net>

APNIC (Offline)
APNIC-EXCHANGE (Offline)

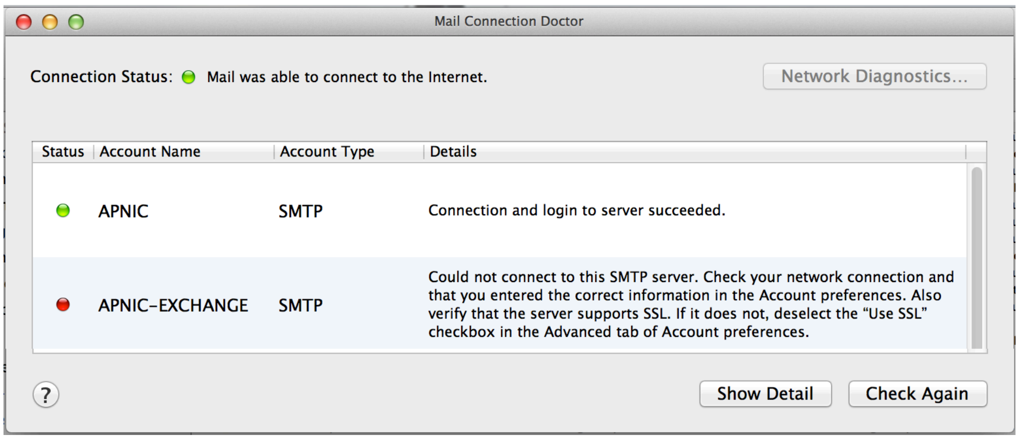
However, most of the time, my outgoing mail just works.

At this point I'm scratching my head a bit. Am I using the right port? In Apple's Mail preference panel I turn off all other options and specify port 587 as the only possible port, as the remote server is using a secure channel, listening on port 587. The problem still persists. Then move to a different wireless network, and the problem goes away. Now I'm getting very confused. The connection problem appears to be completely random. Sometimes I can send mail. Sometimes I can't.

Nothing annoys me more than a random problem. In dealing with computers, applications and networks I've always maintained the view that problems manifest themselves in deterministic ways, and when a problem appears random it means that I've not managed to recognise an underlying pattern.

The first task was to find a set of circumstances where I could reliably find failure. In the last week of February I attended the APRICOT conference in Singapore, and I found that the conference network was consistent. Whenever I used the conference network my mail client was unable to send outgoing mail.

When I see these kind of connectivity faults my first suspicion is that the network path has failed. Indeed, the connection diagnostic screen from the Apple Mail applications says: "check your network connection in the Account preferences"



Connection Status: ● Mail was able to connect to the Internet.

Status	Account Name	Account Type	Details
🟢	APNIC	SMTP	Connection and login to server succeeded.
🔴	APNIC-EXCHANGE	SMTP	Could not connect to this SMTP server. Check your network connection and that you entered the correct information in the Account preferences. Also verify that the server supports SSL. If it does not, deselect the "Use SSL" checkbox in the Advanced tab of Account preferences.

So, like all network engineers my first response is to check the network path, and to do this I turn to ping:

```
$ ping smtp-server.apnic.net
PING smtp-server.apnic.net (203.119.101.xx): 56 data bytes
64 bytes from 203.119.101.xx: icmp_seq=0 ttl=241 time=205.859 ms
64 bytes from 203.119.101.xx: icmp_seq=1 ttl=241 time=203.568 ms
64 bytes from 203.119.101.xx: icmp_seq=2 ttl=241 time=203.093 ms
64 bytes from 203.119.101.xx: icmp_seq=3 ttl=241 time=205.800 ms
64 bytes from 203.119.101.xx: icmp_seq=4 ttl=241 time=202.995 ms
^C
--- smtp-server.apnic.net ping statistics ---
5 packets transmitted, 5 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 202.995/204.263/205.859/1.294 ms
```

I'm in Singapore, and the server is in Brisbane, so a 200ms round trip time is ok. The server is up, the DNS is responding, and the end-to-end path is obviously working with no loss, minor jitter and no reordering. The problem is probably not the network.

Then a bell about dual stack starts clanging in my head. We make an effort to ensure that we use IPv6 in these conferences, and one of the major differences between most hotel networks and this conference network is the presence of IPv6. At APNIC we've also been working to ensure that all our services are dual stack with IPv4 and IPv6. So perhaps this might be something associated with IPv6. Maybe `ping` was the wrong command. Maybe this is a dual stack server, and I should be testing with `ping6` as well.

This is a good example where dual stack makes the debugging exercise just that little bit more complicated. At every stage you have to be aware that even though things look ok in one protocol, that does not necessarily mean that they look good in the other protocol. No visible problems in IPv4 certainly does not mean that the same is true in IPv6. It also widens the set of potential issues. Is this one of protocol selection, or poorly working protocol failover? In this case where I'm using a Mac, then I have to also consider if this application using Apple's version of "happy eyeballs" to manage the TCP connection, resulting in what is in effect a non-deterministic choice between using IPv4 and IPv6?

Let's check the DNS to see if there is a IPv6 address on this server as well:

```
$ dig ANY smtp-server.apnic.net.

; <<>> DiG 9.8.3-P1 <<>> ANY exch-dual.rand.apnic.net.
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 58834
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;exch-dual.rand.apnic.net.       IN      ANY

;; ANSWER SECTION:
exch-dual.rand.apnic.net. 3233   IN      A       203.119.101.xx
exch-dual.rand.apnic.net. 3600   IN      AAAA    2001:dd8:9:2::101:xx

;; Query time: 83 msec
;; SERVER: 220.247.159.2#53(220.247.159.2)
;; WHEN: Thu Feb 28 12:26:29 2013
;; MSG SIZE  rcvd: 86
```

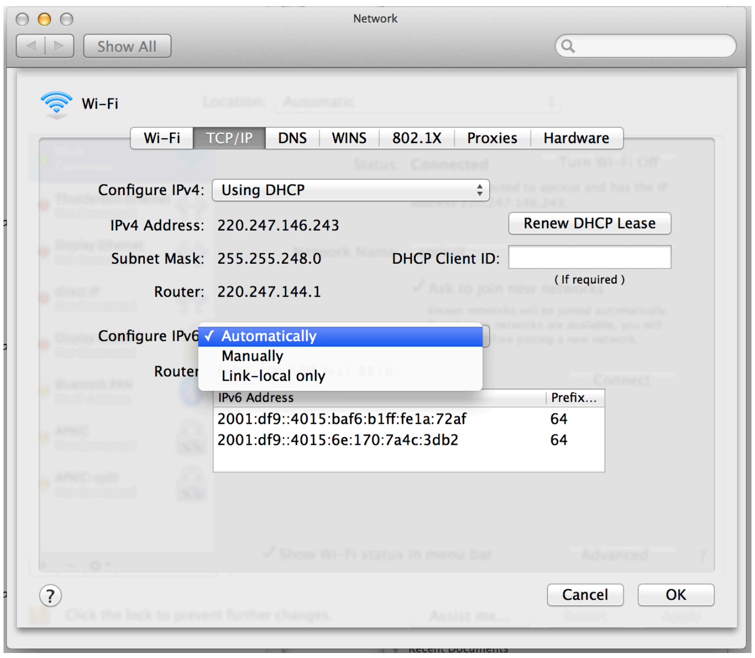
So there IS an IPv6 address behind this server, and this IPv6 address is visible to my local system. Does the mail server respond in IPv6, or has it gone offline? So lets see what `ping6` has to say:

```
$ ping6 exch-dual.rand.apnic.net.
PING6(56=40+8+8 bytes) 2001:df9::4015:6e:170:7a4c:3db2 --> 2001:dd8:9:2::101:xx
16 bytes from 2001:dd8:9:2::101:xx, icmp_seq=0 hlim=50 time=204.387 ms
16 bytes from 2001:dd8:9:2::101:xx, icmp_seq=1 hlim=50 time=189.769 ms
16 bytes from 2001:dd8:9:2::101:xx, icmp_seq=2 hlim=50 time=198.532 ms
16 bytes from 2001:dd8:9:2::101:xx, icmp_seq=3 hlim=50 time=190.296 ms
16 bytes from 2001:dd8:9:2::101:xx, icmp_seq=4 hlim=50 time=191.732 ms
16 bytes from 2001:dd8:9:2::101:xx, icmp_seq=5 hlim=50 time=192.263 ms
^C
--- exch-dual.rand.apnic.net ping6 statistics ---
6 packets transmitted, 6 packets received, 0.0% packet loss
round-trip min/avg/max/std-dev = 189.769/194.496/204.387/5.271 ms
```

Aside from showing me that the IPv6 path to and from the server is 10ms faster than IPv4, I'm no closer to understanding what the problem may be. The server is up on IPv6 as well, its responding and the network is working. Again no loss, minimal jitter, no reordering. All looks just fine.

I'm still no closer to isolating the cause of the problem, but perhaps its worth digging about IPv6 a bit deeper. Which protocol is Apple Mail using to reach the SMTP server? IPv4 or IPv6?

That are a number of ways to do this, but I perhaps the fastest way would be to just turn off IPv6 on my local system and see if I can send my mail using IPv4. But that's easier said than done. But I'm running OSX 10.8 and the dropdown panel in the Network Preferences no longer allows me to turn IPv6 off. Gee, thanks Apple!



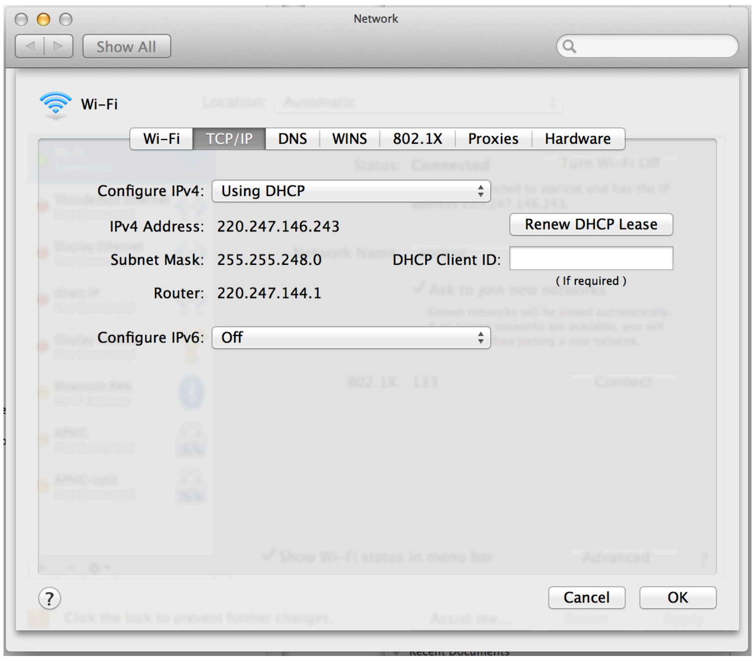
Configure IPv4: Using DHCP
IPv4 Address: 220.247.146.243
Subnet Mask: 255.255.248.0
Router: 220.247.144.1
DHCP Client ID:
Configure IPv6: ✓ Automatically
Router

IPv6 Address	Prefix...
2001:df9::4015:baf6:b1ff:fe1a:72af	64
2001:df9::4015:6e:170:7a4c:3db2	64

Google to the rescue - there IS an undocumented command line method to turn off IPv6 on my Mac:

```
# networksetup -setv6off Wi-Fi
```

And it does the job:



Configure IPv4: Using DHCP
IPv4 Address: 220.247.146.243
Subnet Mask: 255.255.248.0
Router: 220.247.144.1
DHCP Client ID:
Configure IPv6: Off

And the network interface now has no IPv6 binding.

```
# ifconfig en0
en0: flags=8963<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
        ether b8:f6:b1:1a:72:af
        inet 220.247.146.243 netmask 0xfffff800 broadcast 220.247.151.255
        media: autoselect
        status: active
```

Now my laptop will only use IPv4.

Can I now reliably send my mail?

Well, yes!

So it's a pretty good guess at this stage that this now looks like some kind of IPv6 problem.

I'm aware of the IPv6 Path MTU problem that can create connection black holes. A previous article from 2009, "A Tale of Two Protocols: IPv4, IPv6, MTUs and Fragmentation" (`http://www.potaroo.net/ispcol/2009-01/mtu6.html`), described a similar problem when trying to connect to web pages using IPv6, where the attempt of a dual stack browser to retrieve a web page froze without displaying anything.

The essential problem with Path MTU discovery in IPv6 is that if the sender, in this case my local system, manages to open a TCP session with the mail server, then, either as part of the connection setup process, or within the established session, sends a packet that is too large for the end-to-end path to the server then I may encounter a problem. If the packet is too large for the path there will be a router on the path where the IPv6 packet cannot be forwarded onward. The IPv6-defined response is to send an ICMP6 Packet Too Big control response to the sender of the packet, and then discard the offending packet. In the normal course of events the sender, my laptop on the conference network, would receive the ICMP packet, adjust its sending size and the connection would resume. But if the conference network is blocking incoming ICMP6 packets then there would be a problem. If my local system has sent a packet, but it's not getting either an ACK in response, or an ICMP6 response, then the local TCP session will simply hang. The connection will fail. So this looks like a plausible explanation for the problem.

So I'll turn on IPv6 again (`# networksetup -setv6automatic Wi-Fi`), and now try to see if this is a IPv6 Path MTU problem. The way I'll do this is by using the `tcpdump` application on my laptop, and record the entire packet exchange with the remote mail server. What I am looking for is a successful connection establishment up to the point of the exchange of a large packet that is supporting some form of certificate exchange to support the TLS channel.

```
1085.983669 IP6 2001:df9::4015:7c2e:f7fb:d46:88d8.65028 > 2001:dd8:9:2::101:xx.587: Flags [S], seq 123000360, win 65535,
                options [mss 1440,nop,wscale 4,nop,nop,TS val 962855584 ecr 0,sackOK,eol], length 0
1086.172576 IP6 2001:dd8:9:2::101:xx.587 > 2001:df9::4015:7c2e:f7fb:d46:88d8.65028: Flags [S.], seq 2478234559, ack 123000361,
                win 4320, options [mss 1440,nop,wscale 0,nop,nop,TS val 596451933 ecr 962855584,sackOK,eol], length 0
1086.172753 IP6 2001:df9::4015:7c2e:f7fb:d46:88d8.65028 > 2001:dd8:9:2::101:xx.587: Flags [.], ack 1, win 8211,
                options [nop,nop,TS val 962855772 ecr 596451933], length 0
1086.363845 IP6 2001:dd8:9:2::101:xx.587 > 2001:df9::4015:7c2e:f7fb:d46:88d8.65028: Flags [P.], seq 1:96, ack 1, win 4320,
                options [nop,nop,TS val 596452124 ecr 962855772], length 95
                220.NXMDA1.org.apnic.net.Microsoft.ESMTP.MAIL.Service.ready.at.Fri,.1.Mar.2013.11:24:44.+1000..
1086.363929 IP6 2001:df9::4015:7c2e:f7fb:d46:88d8.65028 > 2001:dd8:9:2::101:xx.587: Flags [.], ack 96, win 8205,
                options [nop,nop,TS val 962855962 ecr 596452124], length 0
1086.373069 IP6 2001:df9::4015:7c2e:f7fb:d46:88d8.65028 > 2001:dd8:9:2::101:xx.587: Flags [P.], seq 1:48, ack 96, win 8205,
                options [nop,nop,TS val 962855971 ecr 596452124], length 47
                EHLO.[IPv6:2001:df9::4015:7c2e:f7fb:d46:88d8]..
1086.561695 IP6 2001:dd8:9:2::101:xx.587 > 2001:df9::4015:7c2e:f7fb:d46:88d8.65028: Flags [.], ack 48, win 4367,
                options [nop,nop,TS val 596452323 ecr 962855971], length 0
1086.563043 IP6 2001:dd8:9:2::101:xx.587 > 2001:df9::4015:7c2e:f7fb:d46:88d8.65028: Flags [P.], seq 96:127, ack 48, win 4367,
                options [nop,nop,TS val 596452324 ecr 962855971], length 31
                501.5.5.4.Invalid.domain.name..
1086.563198 IP6 2001:df9::4015:7c2e:f7fb:d46:88d8.65028 > 2001:dd8:9:2::101:xx.587: Flags [.], ack 127, win 8203,
                options [nop,nop,TS val 962856160 ecr 596452324], length 0
1086.564210 IP6 2001:df9::4015:7c2e:f7fb:d46:88d8.65028 > 2001:dd8:9:2::101:xx.587: Flags [F.], seq 48, ack 127, win 8203,
                options [nop,nop,TS val 962856161 ecr 596452324], length 0
```

If I strip out the IPv6 packet headers from the `tcpdump` log, and also strip out most of the TCP details, what is left is the following picture of the protocol exchange I observed for a failed effort to send mail:

```
Time (ms)     Size    Client      Mail Server

    0          60     TCP: SYN
  187          60                 TCP: SYN + ACK
  187          60     TCP: ACK
  373         156                 SMTP: 220 IAMDA2.org.apnic.net Microsoft ESMTP MAIL
                                           Service ready at Thu, 28 Feb 2013 10:53:09 +1000
  373          60     TCP: ACK
  385         107     SMTP: EHLO [IPv6:2001:df9::4015:1430:8367:2073:5d0]
  569          60                 TCP: ACK
  570          91                 SMTP 501 5.5.4 Invalid domain name
  570          60     TCP: ACK
  571          60     TCP: FIN
```

The largest packet here was 156 octets in size, so it's clear that the failure here is not one based about IPv6 path MTU discovery problems. However, the packet dump is helpful in another way: it also illustrates the nature of the actual problem with my unsuccessful efforts to send mail.

If we remove the TCP part of the conversation we have the mail sending conversation This conversation uses the Simple Mail Transfer Protocol, as defined in RFC5321. The conversation looks like:

```
Client                         Mail Server

open a session to port 587
on the Mail Server
                               220 IAMDA2.org.apnic.net Microsoft ESMTP MAIL
                               Service ready at Thu, 28 Feb 2013 10:53:09 +1000

EHLO [IPv6:2001:df9::4015:1430:8367:2073:5d0]

                               501 5.5.4 Invalid domain name
```

SMTP was one of a small number of IP application protocols that eschewed the use of binary-packed control protocols, such as the later-defined SNMP and its use of ASN.1, and instead used simple lines of ASCII text as the base language of the protocol exchange. It may not have quite the same compactness of expression, but it certainly makes debugging the operation of the protocol, and even hand-cranking the protocol fantastically easy!

What is going on here is that the mail server opens up a channel with the client (220 response), the client then responds with a hello, with the use of the EHLO variant of this response to indicate that it recognises extended SMTP comments. The EHLO response has an associated value, namely the domain name of the client system. If the client has no domain name it is allowed to send its IP address as an address literal. In the case above the client is sending its IPv6 address, `2001:df9::4015:1430:8367:2073:5d0`, as an address literal.

But the SMTP server is declaring this to be an invalid domain name!

Lets see what it thinks when I try the same thing using IPv4:

```
$ telnet smtp-server.apnic.net 587
Trying 203.119.101.xx...
Connected to exch-smtp.apnic.net.
Escape character is '^]'.
220 IAMDA1.org.apnic.net Microsoft ESMTP MAIL Service ready at Thu, 28 Feb 2013 16:42:42 +1000
EHLO [220.247.146.243]
250-IAMDA1.org.apnic.net Hello [203.119.101.xx]
```

That's strange. It worked!

Lets try the IPv6 connection again, this time using a deliberately invalid domain name:

```
$ telnet -6 smtp-server.apnic.net 587
Trying 2001:dd8:9:2::101:xx...
Connected to exch-v6only.rand.apnic.net.
Escape character is '^]'.
220 NXMDA1.org.apnic.net Microsoft ESMTP MAIL Service ready at Thu, 28 Feb 2013 16:45:22 +1000
EHLO donkey
250-NXMDA1.org.apnic.net Hello [2001:dd8:9:2::101:xx]
```

It liked that!

What about an unreachable IPv6 address?

```
EHLO [IPv6:2001::1]
250-NXMDA1.org.apnic.net Hello [2001:dd8:9:2::101:xx]

EHLO [IPv6:::1]
250-NXMDA1.org.apnic.net Hello [2001:dd8:9:2::101:xx]
```

It liked those too!

What about badly formatted IPv6 addresses?

```
EHLO [IPv6:1:2:3:4:5:6:7:8:9]
501 5.5.4 Invalid domain name
```

So it seems that the Microsoft mail exchange server is thinking that `2001:df9::4015:1430:8367:2073:5d0` is a badly formatted IPv6 address.

Let's put that to the test by going to another system and once more getting out the handy `ping6` tool:

```
$ ping6 2001:df9::4015:1430:8367:2073:5d0
PING6(56=40+8+8 bytes) 2001:388:1000:120:d267:e5ff:x:y --> 2001:df9:0:4015:1430:8367:2073:5d0
16 bytes from 2001:df9:0:4015:1430:8367:2073:5d0, icmp_seq=0 hlim=53 time=355.244 ms
16 bytes from 2001:df9:0:4015:1430:8367:2073:5d0, icmp_seq=1 hlim=53 time=342.355 ms
16 bytes from 2001:df9:0:4015:1430:8367:2073:5d0, icmp_seq=2 hlim=53 time=347.857 ms
16 bytes from 2001:df9:0:4015:1430:8367:2073:5d0, icmp_seq=3 hlim=53 time=340.488 ms
16 bytes from 2001:df9:0:4015:1430:8367:2073:5d0, icmp_seq=4 hlim=53 time=340.604 ms
^C
--- 2001:df9::4015:1430:8367:2073:5d0 ping6 statistics ---
5 packets transmitted, 5 packets received, 0.0% packet loss
round-trip min/avg/max/std-dev = 340.488/345.310/355.244/5.644 ms
```

It worked! At least the `ping6` tool thinks that this is a validly formatted address.

But there is something a little odd here. While I entered the V6 address as `2001:df9::4015:1430:8367:2073:5d0`, the `ping6` program reports this address in a slightly different manner as `2001:df9:0:4015:1430:8367:2073:5d0`

Strange.

Is this always the case? Lets try `ping6` with another IPv6 literal that contains "::"

```
$ ping6 2001:df9::baf6:b1ff:fe1a:72af
PING6(56=40+8+8 bytes) 2001:388:1000:120:d267:e5ff:x:y --> 2001:df9::baf6:b1ff:fe1a:72af
```

Evidently not.

Let's try the alternate address format on the Exchange mail server where the "::" is expanded to "`:0:`".

```
220 IAMDA1.org.apnic.net Microsoft ESMTP MAIL Service ready at Thu, 28 Feb 2013 17:03:46 +1000

EHLO [IPv6:2001:df9::4015:1430:8367:2073:5d0]
501 5.5.4 Invalid domain name

EHLO [IPv6:2001:df9:0:4015:1430:8367:2073:5d0]
250-IAMDA1.org.apnic.net Hello [2001:dd8:9:2::101:249]
```

Now we've managed to identify the problem. It appears that the Microsoft exchange server will not accept IPv6 addresses when the :: notation is used instead of a SINGLE 0-value nibble.

Bug? Standards Compliance? Or something else? Now let's do a bit of RFC browsing.

I had always thought that the last word on IPv6 address formats was RFC4291, published in 2006, which said:

```
2. Due to some methods of allocating certain styles of IPv6
   addresses, it will be common for addresses to contain long strings
   of zero bits.  In order to make writing addresses containing zero
   bits easier, a special syntax is available to compress the zeros.
```

```
      The use of "::" indicates one or more groups of 16 bits of zeros.
      The "::" can only appear once in an address.  The "::" can also be
      used to compress leading or trailing zeros in an address.
```

Its a compact specification, but if you think about it its a complete and unambiguous specification. And `2001:df9::4015:1430:8367:2073:5d0` conforms to this specification.

However, its not the last word in specification of IPv6 addresses. RFC5952, published in 2010, updates RFC4291, and it adds some further, and arguably quite spurious, additional constraints:

```
    4.2.2.  Handling One 16-Bit 0 Field

      The symbol "::" MUST NOT be used to shorten just one 16-bit 0 field.
      For example, the representation 2001:db8:0:1:1:1:1:1 is correct, but
      2001:db8::1:1:1:1:1 is not correct."
```

When we combine that specification with that in RFC5321:

```
   4.1.3 Address Literals

      [...]

      For IPv6 and other forms of addressing that might eventually
      be standardized, the form consists of a standardized "tag" that
      identifies the address syntax, a colon, and the address itself, in a
      format specified as part of the relevant standards (i.e., RFC 4291
      [8] for IPv6).
```

These days the "relevant standard" for IPv6 address literals is no longer RFC4291, but RFC5952.

So it appears that the Microsoft's Exchange Mail Server has taken an accurate view of the relevant standards in accepting an IPv6 literal that precisely conforms to RFC5952, and rejecting all other IPv6 literal address forms, including `2001:df9::4015:1430:8367:2073:5d0`. And it appears that Apple's Mail product is the problem here in offering an IPv6 literal in its SMTP conversation that is no longer compliant with current standards.

But in so saying that, I can't help but think that this is still an unsatisfactory outcome. I can't help thinking that the words of Jon Postel in RFC1122, from October 1989, offer some extremely helpful advice here:

```
     1.2.2  Robustness Principle

         At every layer of the protocols, there is a general rule whose
         application can lead to enormous benefits in robustness and
         interoperability [IP:1]:

                "Be liberal in what you accept, and
                 conservative in what you send"

         Software should be written to deal with every conceivable
         error, no matter how unlikely; sooner or later a packet will
         come in with that particular combination of errors and
         attributes, and unless the software is prepared, chaos can
         ensue.  In general, it is best to assume that the network is
         filled with malevolent entities that will send in packets
         designed to have the worst possible effect.  This assumption
         will lead to suitable protective design, although the most
         serious problems in the Internet have been caused by
         unenvisaged mechanisms triggered by low-probability events;
         mere human malice would never have taken so devious a course!
```

I'd like to express my thanks to Ben O'Hara and George Michaelson of APNIC for their persistence in pursuing this problem through, and working with me in delving through numerous logs to follow the trail described here.

Afterword

I've received a couple of comments about this article from Simon Leinen and Seiichi Kawamura (who is one of the co-authors of RFC5952) soon after I published this article, and it seems that the standards part of this story is a little more confused than it would appear at first. So perhaps I should make a couple of additional comments, to either clarify the situation, or possibly to add to the confusion!

In this article I wrote that:

> "So it appears that the Microsoft's Exchange Mail Server has taken an accurate view of the relevant standards in accepting an IPv6 literal that precisely conforms to RFC5952, and rejecting all other IPv6 literal address forms, including 2001:df9::4015:1430:8367:2073:5d0."

But does it really do this?

RFC5952 says that if there is a `"::"` instance it should be maximal, so that constructs such as `":0::"` or `"::0:"` are invalid. Does Microsoft's Exchange Mail Server agree?

```
EHLO [IPv6:1:0::0:2]
250-NXMDA1.org.apnic.net Hello [203.119.101.xx]
```

Evidently not.

RFC5952 says that if there are two 'runs' of consecutive 0-valued 16 bit nibbles, the longer run is to be replaced by a `"::"` symbol. Does Microsoft's Exchange Mail Server agree?

```
EHLO [IPv6:1::4:0:0:0:8]
250-NXMDA1.org.apnic.net Hello [203.119.101.xx]
```

Evidently not.

The writing in RFC5952 is incredibly sloppy for a Proposed Standard, but one interpretation of the text is that RFC5952 says that if there is a 'run' of 2 or more consecutive 0-valued 16 bit nibbles then it must be replaced by a `"::"` symbol. Does Microsoft's Exchange Mail Server agree?

```
EHLO [IPv6:0:0:0:0:0:0:0:0]
250-NXMDA1.org.apnic.net Hello [203.119.101.xx]
```

Evidently not.

Maybe Microsoft's Mail Exchange Server is not actually implementing the provisions of RFC5952 at all. Maybe its implementing a simpler set of constraints as specified in RFC2821 which is slightly more constraining than RFC4291 (even though RFC2821 was published earlier), in so far as it contains a formal grammar form of the IPv6 literal:

```
IPv6-comp = [IPv6-hex *5(":" IPv6-hex)] "::" [IPv6-hex *5(":"IPv6-hex)]
      ; The "::" represents at least 2 16-bit groups of zeros
```

So lets test this. Does Microsoft's Exchange Mail Server implement a IPv6 literal parser that implements the syntax of RFC2821?

```
EHLO [IPv6:1::2::3]
250-NXMDA1.org.apnic.net Hello [203.119.101.xx]
```

Evidently not.

I have no idea what rule set the Microsoft Exchange Mail server is following – it seems to be related to RFC2821, but it also accepts other IPv6 literal forms that are not valid IPv6 literals.

So what is the story about the `"::"` symbol anyway?

RFC2821 references RFC2373, which obsoleted RFC1884, the original IPv6 Address Architecture specification. Both RFC1884 and RFC2373 contain the following text:

```
The use of "::" indicates multiple groups of 16-bits of zeros.
The "::" can only appear once in an address.  The "::" can also be
used to compress the leading and/or trailing zeros in an address
```

My understanding of the term "`multiple`" in this context is that a `"::"` symbol can be used to represent two or more groups of 16 bits of zeros in this document.

But this document was obsoleted by RFC3513, and here the text was changed to read:

```
The use of "::" indicates one or more groups of 16 bits of zeros.
The "::" can only appear once in an address.  The "::" can also be
used to compress leading or trailing zeros in an address.
```

Subsequently, this document was obsoleted by RFC4291, but this section of the document was not altered by RFC4291. A `"::"` symbol can be used to represent one or more groups of 16 bits of zeros.

But this document is updated by RFC5952, so the text in RFC5952 is now the normative text.

It appears to be that at present this is all about trying to interpret RFC5952 and its use of so-called "normative" text. RFC5952 is a Internet Standards Track document, and makes reference to a set of terms that have a defined meaning. These days many RFCs include the following text as part of its "boilerplate" introductory text, and RFC5952 is no exception:

```
1.1.  Requirements Language

   The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
   "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
   document are to be interpreted as described in [RFC2119].
```

Lets go to RFC2119 and check out what meaning some of these terms are intended to convey:

```
1. MUST   This word, or the terms "REQUIRED" or "SHALL", mean that the
   definition is an absolute requirement of the specification.

2. MUST NOT   This phrase, or the phrase "SHALL NOT", mean that the
   definition is an absolute prohibition of the specification.

3. SHOULD   This word, or the adjective "RECOMMENDED", mean that there
   may exist valid reasons in particular circumstances to ignore a
   particular item, but the full implications must be understood and
   carefully weighed before choosing a different course.

4. SHOULD NOT   This phrase, or the phrase "NOT RECOMMENDED" mean that
   there may exist valid reasons in particular circumstances when the
   particular behavior is acceptable or even useful, but the full
   implications should be understood and the case carefully weighed
   before implementing any behavior described with this label.
```

You would think that's all pretty straightforward. "MUST" means "it really should happen," and "MUST NOT" is a case of "do not ever do this."

But I find the nesting of the MUST, MUST NOT and SHOULD terms in RFC5952 to be extremely confusing. In retrospect, I believe that the additional specifications in RFC5952 are spurious and its served only to take a situation that was loosely specified in parts, but allowed interoperation, to a situation that is now over-specified in a manner that breaks interoperation.

Lets review RFC5952 and the specifications in Section 4 of that document:

```
4.2.1.  Shorten as Much as Possible

   The use of the symbol "::" MUST be used to its maximum capability.
   For example, 2001:db8:0:0:0:0:2:1 must be shortened to 2001:db8::2:1.
   Likewise, 2001:db8::0:1 is not acceptable, because the symbol "::"
   could have been used to produce a shorter representation 2001:db8::1.
```

That first sentence is woefully imprecise for a standards document. Does this mean that the `"::"` is to be used whenever it could be used? The example in the second sentence seems to suggest this, but its use of the word "must" is not a use of the normative language of "MUST". Are IPv6 address constructs such as `"2001:db8:0:0:0:0:2:1"` valid because they choose not to use the symbol `"::"` at all?

Was that first sentence intended to state: "*The use of the symbol `"::"` MUST be used whenever it could be used, and, when used, it MUST be used to its maximum capability*"? Or was it intended to state: "*The use of the symbol `"::"` MAY be used whenever it could be used, and, when used, it MUST be used to its maximum capability*"?

The next section appears to be a little less ambiguous in meaning:

```
4.2.2.  Handling One 16-Bit 0 Field

   The symbol "::" MUST NOT be used to shorten just one 16-bit 0 field.
   For example, the representation 2001:db8:0:1:1:1:1:1 is correct, but
   2001:db8::1:1:1:1:1 is not correct.
```

This appears to be revising RFC4291, and clearly reverting back to the specification of RFC1884 and RFC2373, where a `"::"` symbol replaces two or more groups of 16 bit zeros. This is also consistent with the specification in RFC2821.

So here Apple's Mail, and indeed other elements of Apple's IPv6 implementation on the current version of OSX, seems to have a problem with compliance with this part of the specification. i.e. this seems pretty clear that Apple's continued use of `"::"` to replace `":0:"` is not compliant, and Microsoft is simply following a literal interpretation of the document's normative text in rejecting this. This is a pretty clear interpretation of MUST NOT.

But maybe it's not as clear as this.

Just above this text, RFC5952 also has the following text:

```
4.  A Recommendation for IPv6 Text Representation

   A recommendation for a canonical text representation format of IPv6
   addresses is presented in this section.  The recommendation in this
   document is one that complies fully with [RFC4291], is implemented by
   various operating systems, and is human friendly.  The recommendation
   in this section SHOULD be followed by systems when generating an
   address to be represented as text, but all implementations MUST
   accept and be able to handle any legitimate [RFC4291] format.  It is
   advised that humans also follow these recommendations when spelling
   an address.
```

What does this mean? The word "recommendation" does not appear to be capitalized, so it appears not to refer just to those actions described in this section that are qualified by SHOULD or RECOMMENDED. Instead the use of a normative SHOULD appears to say that all actions subsequently described in this section, including the MUST NOT in section 4.2.2, are not necessarily absolute requirements. In other words, the MUSTs in the following subsections of section 4 are actually SHOULDs, and the MUST NOTs are actually SHOULD NOTs.

Perhaps Apple does not have a problem with compliance with RFC5952, in so far as it's software SHOULD NOT have substituted a `"::"` for `":0:"`, but in so doing it did not violate strict compliance with RFC5952. And in the same manner perhaps Microsoft's Exchange Mail Server, who enforces the MUST NOT provisions in section 4.2.2 of RFC5952 when parsing an IPv6 literal address, is actually in violation of strict compliance with RFC5952, as in it fact MUST accept any IPv6 literal form that complies with RFC4291, including the use of `"::"` to stand for `":0:"`.

One interpretation of the combination of RFC2119, RFC4291 and RFC5952 is that is it possible for an implementation to use the `"::"` construct in place of `":0:"`, and there was no need to make changes to existing code following the publication of RFC5952, and other implementations of IPv6 should accept this as a valid IPv6 literal. i.e. no need to change.

Another interpretation of RFC5952 is that a MUST NOT is as defined in RFC2119 in all and every instance where it appears in a document, and the use of this term in section 4.2.2. implies an absolute requirement in all cases, so that implementations of IPv6 that previously accepted the use of `"::"` to represent `":0:"` must be changed in order to comply with RFC5952. i.e. a change is required.

No wonder that the software engineers can read precisely the same words and came to different conclusions.

Indeed, I note that RFC5952 has two authors, and as RFC5952 itself points out:

```
3.4.2.  Preference in Documentation

   A document that is edited by more than one author may become harder
   to read.
```

I can't help but think that this continual effort to tinker with working specifications can be detrimental rather then helpful. Sometimes, as in this case, the result of such tinkering with the specifications is a result that breaks things for us poor dual-stack users.

Many years ago the IETF used to say that their standards were pragmatic documents that were based on working interoperable implementations, and this pragmatic concern with interoperation differentiated them from some other standards-making bodies. The IETF, they claimed, did not specialize in paperware about vaporware. Instead, the IETF produced technical specifications based on working interoperable code.

These days it's a very hard case to sustain that all, or possibly even any, of the IETF's outputs are based primarily in documenting working, interoperating code, and this particular topic is a case in point. I would contend that the victim here of this apparent shift in the way the IETF produces technical specifications is the basic and fundamental reason for technical standards in networking: interoperation. I don't think that this kind of confusion reflects well on the IETF and it's levels of review of its standards specifications.

Finally, let me note that the solution we used to correct our problem of interoperability was very simple. The pragmatic solution to this problem that did not require waiting for an indefinite period for a patch from either Apple or Microsoft was to turn off V6. We removed the AAAA record for the mail server's name in the DNS, and the problem as simply disappeared for me.

So I'm both happy and disappointed with this outcome. Yes, happily, I can now send mail in dual stack networks without encountering frustrating errors. However I'm very disappointed that the solution is another instance that supports that unofficial axiom in networking operations: when you encounter a problem in dual stack network services, first turn off IPv6.

Disclaimer

The views expressed are the author's and not those of APNIC, unless APNIC is specifically identified as the author of the communication. APNIC will not be legally responsible in contract, tort or otherwise for any statement made in this publication.

Author

Geoff Huston B.Sc., M.Sc., is the Chief Scientist at APNIC, the Regional Internet Registry serving the Asia Pacific region. He has been closely involved with the development of the Internet for many years, particularly within Australia, where he was responsible for the initial build of the Internet within the Australian academic and research sector. He is author of a number of Internet-related books, and was a member of the Internet Architecture Board from 1999 until 2005, and served on the Board of Trustees of the Internet Society from 1992 until 2001.

www.potaroo.net