The ISP Column
A monthly column on things Internet

December 2010

Geoff Huston

Flailing IPv6

Is IPv6 a robust as IPv4? What proportion of IPv6 connections "fail"? And what causes such connection failures?

These questions were raised in a discussion at RIPE 61 in November 2010. The general conclusion appeared to be that using auto-tunnelling techniques to patch up an IPv6 connection was generally thought to be worse than just allowing the connection to be made using IPv4. It has been observed that various forms of mis-configuration and local traffic filters create asymmetric conditions for some client systems. These clients attempt to connect to a dual stack site using IPv6 but time out while waiting for any form of response. It is generally believed that this connection "failure" state is sufficiently prevalent with IPv6 auto-tunnelling that it constitutes sufficient grounds to warn against using 6to4 at all!

So how bad is it?

Emile Aben has investigated this in an article posted on RIPE Labs at the start of this month (http://labs.ripe.net/Members/emileaben/6to4-how-bad-is-it-really), and he has noted a "6to4 failure rate" of some 15% of all 6to4 connections. This seems to be an extremely high failure rate, and in this article I'll be reporting on some results gathered on the measurement systems operated at the potaroo.net web server.

The methodology used here is to search for a particular form of asymmetric failure, where the remote site is capable of getting an IP packet to the measurement server site, but the return packet appears not to reach the remote site. What the measurement site sees in the case of this particular form of asymmetric failure is the initial TCP connection packet (a "SYN" packet) is seen at the measurement server, but when the server sends a SYN/ACK packet in response, the next packet, an ACK packet, is not observed (a longer description of the experimental approach is included at the end of this article).

Observations

In terms of the profile of connections measured in this experiment, on average over the last 7 days in December 2010 this site's web server is contacted by an average of some 22,000 unique source addresses per day. Some 96% of the source addresses are IPv4 addresses, and 4% are IPv6.

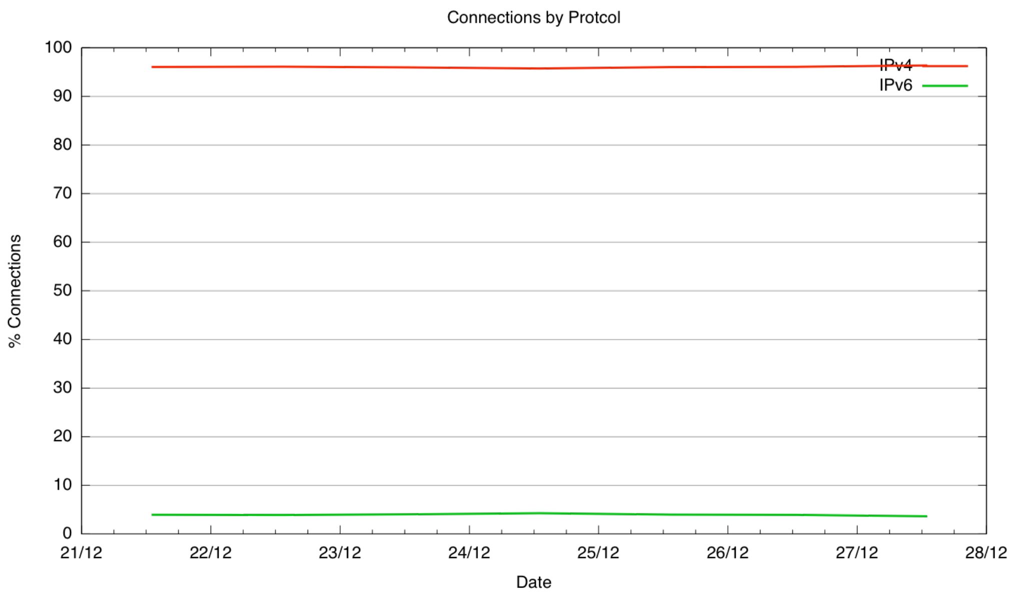

Figure 1 - Connections by Protocol

Of the V6 connections, only some 18% of the V6 connections are non-tunnelled. 78% of the connections, or some 600 unique sources out of an average of 875 IPv6 connections per day, use 6to4. only 4% of the connections, or an average of 40 unique sources, used Teredo.

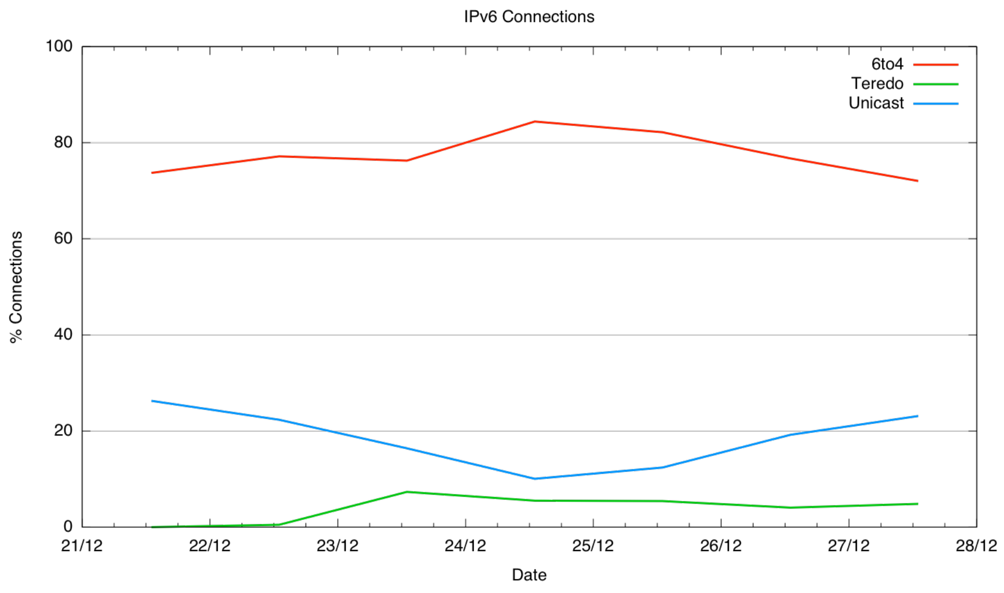

Figure 2 - IPv6 Connections by Access Type

The connection failure rate for all connections is an average of 0.6% of all connection attempts. In terms of counts, the failure rate is an average of 126 connection attempts out of an average total of some 22,000 unique connection attempts per day. The level of failed IPv6 connection attempts, an average of 82 connection failures per day, is approximately double that of the number of failed IPv4 connection attempts, which number an average of 45 connection failures per day.

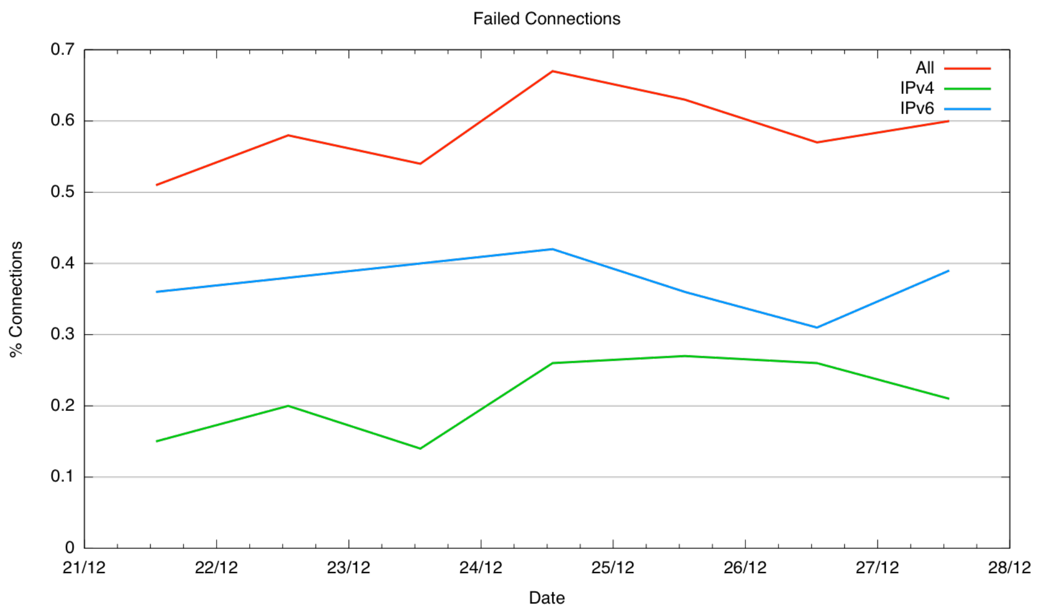

Figure 3 - Failed Connections

The significant level of disparity between IPv4 and IPv6 is quite evident when looking at the relative failure rate of the two protocols. Some 0.2% of the IPv4 connection attempts fail, whereas some 9.5% of the IPv6 connection attempts fail.

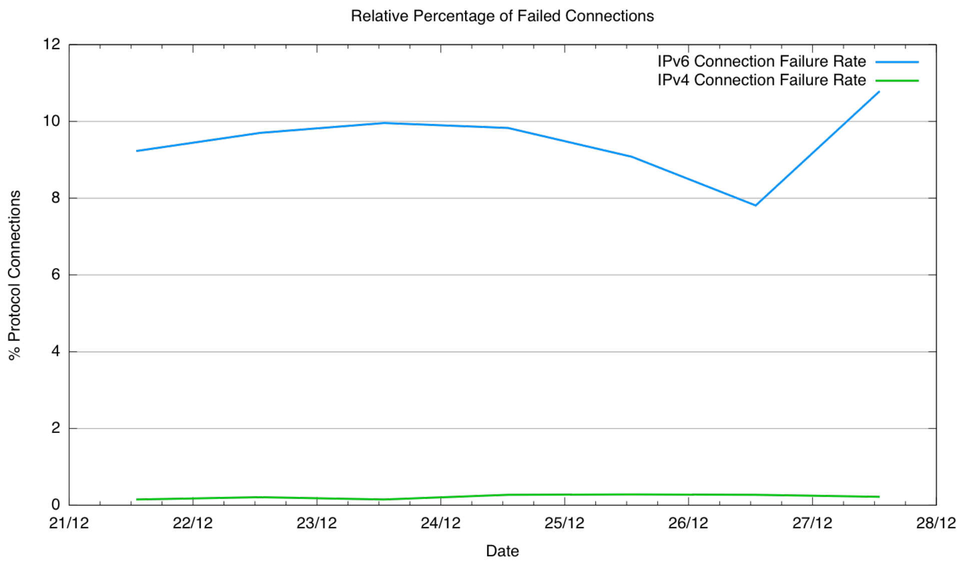

Figure 4 - Relative Percentage of Failed Connections

Within the set of IPv6 connections, the connection failure rate can be further delineated between the various access types. Connections using 6to4 have a failure rate of some 13% on average. Teredo failure rates, on the other hand, are far lower at between 1 to 2%. The non-auto-tunnelled IPv6 connections have a failure rate of 2%, which is 10 times the equivalent
IPv4 average failure rate of 0.2%

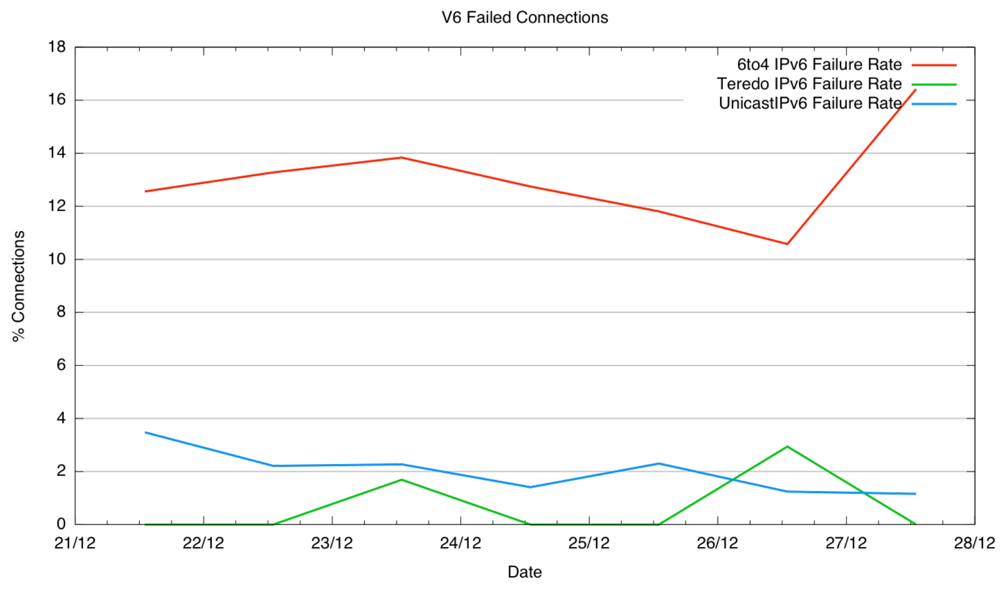

Figure 5 – IPv6 Connection Failure Rate by Access Type

Failure and Recovery

Both 6to4 and Teredo have the local host's IPv4 addresses embedded within their IPv6 addresses. This allows a further question to be posed: What proportion of "failed" connection attempts using 6to4 and Teredo manage to successfully connect to this measurement server using IPv4?

Some 75% of all failed 6to4 connection attempts also log a successful IPv4 connection using the IPv4 address that was embedded in the 6to4 IPv6 address. In other words, three quarters of the 6to4 connection failures are salvaged in a dual stack environment, using a fallback to IPv4 and making a successful IPv4 connection.

In the case of Teredo the "salvage rate" is higher, with some 93% of all failed Teredo connections matching a successful IPv4 connection with the external address of the IPv4 NAT that was embedded in the Teredo address.

What is 6to4 "Failure"?

The 6to4 protocol is an auto-tunnel protocol that uses IPv6-in-IPv4 encapsulation to connect 'islands' of IPv6 to the IPv6 Internet. The encapsulation is a standard form of protocol 41 encapsulation (RFC 2473), where a single 20 octet IPv4 packet header is prepended onto the IPv6 packet. 6to4 relies on 2 relays: one on the 'outbound' path, where a 6to4 gateway takes incoming 6to4 packets (IPv4 packets with an IPv6 payload), strips off the IPv4 header and forwards the IPv6 packet, and one on the 'return' path, which takes incoming IPv6 packets, and adds an IPv4 wrapper, generating the IPv4 destination address from the 6to4 IPv6 address (Figure 6).

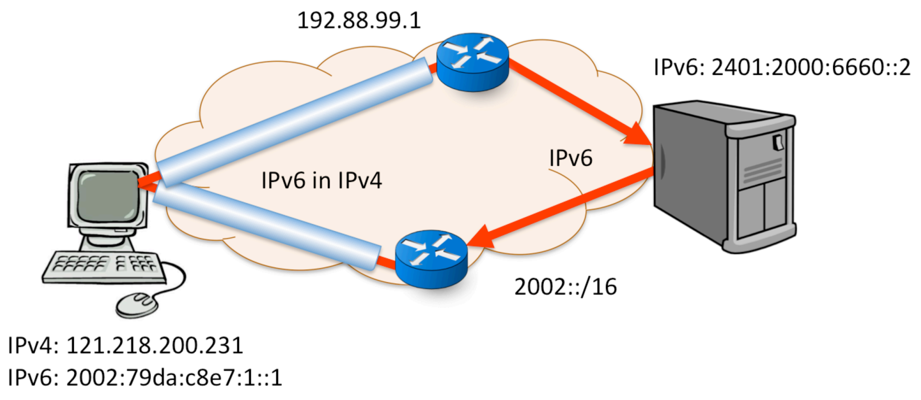

Figure 6 – 6to4 Connection Path

6to4 introduces asymmetry in the packet path. The outbound IPv4 packet is sent to the anycast address 192.88.99.1, while the return packet is first routed to a IPv6 gateway that is advertising a route for 2002::/16.

A 6to4 round trip can get to be pretty impressive. When writing this article I found the following route from my location in Australia to the potaroo.net server, also located in Australia:

```
$ traceroute6 www.potaroo.net
traceroute6 to www.potaroo.net (2401:2000:6660::2) from 2002:79da:c8e7:1::1, 64 hops max, 12 byte
packets
 1 * * *
```

The first hop is a tunnel, so the interior hops are not directly visible in 6to4, but we can see them in IPv4:

```
$ traceroute 192.88.99.1
traceroute to 192.88.99.1 (192.88.99.1), 64 hops max, 52 byte packets
 1  172.18.112.229 (172.18.112.229)  1233.138 ms  859.348 ms  489.917 ms
 2  172.18.71.18 (172.18.71.18)  930.313 ms  379.502 ms *
 3  172.18.242.9 (172.18.242.9)  400.280 ms  109.845 ms  100.067 ms
 4  bundle-ether10.fli8.adelaide.telstra.net (120.151.110.25)  169.501 ms  89.683 ms  309.978 ms
 5  bundle-ether1.fli-core1.adelaide.telstra.net (203.50.11.9)  229.865 ms  229.200 ms  99.634 ms
 6  pos0-4-1-0.win-core1.melbourne.telstra.net (203.50.6.185)  299.045 ms  119.659 ms  520.142 ms
 7  bundle-pos3.ken-core4.sydney.telstra.net (203.50.11.12)  269.926 ms  340.140 ms  569.248 ms
 8  bundle-ether1.pad-gw2.sydney.telstra.net (203.50.6.29)  339.905 ms  320.014 ms  380.272 ms
 9  203.50.13.94 (203.50.13.94)  239.504 ms  319.555 ms  199.950 ms
10  i-9-0-0.sydp-core01.bi.reach.com (202.84.221.89)  319.871 ms  699.752 ms  282.027 ms
11  i-0-0-1-0.1wlt-core01.bx.reach.com (202.84.143.209)  277.761 ms  309.459 ms  259.767 ms
12  i-13-2-1.sjc-core01.bi.reach.com (202.84.143.34)  289.505 ms  310.070 ms  899.668 ms
13  i-3-2.eig-core01.bi.reach.com (202.84.141.141)  349.992 ms  349.962 ms  350.341 ms
```

(Yes, that's across the Pacific Ocean, and continental US, ignoring all forms of 6to4 relays there and heading onward across the Atlantic!)

```
14  i-2-0.ultt-core01.bx.reach.com (202.84.144.82)  469.648 ms  589.352 ms  610.068 ms
15  * * *
16  * * *
17  * * *
18  * * *
```

(We've finally made it to SWITCH in Switzerland, and now we need to to head through a forest of routers to get to their public 6to4 relay service.)

```
19  swic-00-ser0.ce.ip-plus.net (164.128.20.46)  847.290 ms  829.416 ms  460.194 ms
20  swiel2-10ge-1-3.switch.ch (130.59.37.66)  460.068 ms  599.639 ms  450.464 ms
21  swils2-10ge-1-2.switch.ch (130.59.36.69)  429.526 ms  1159.521 ms  450.712 ms
```

```
22  swibe1-10ge-1-1.switch.ch (130.59.37.130)  439.340 ms  439.774 ms  440.156 ms
23  swibe2-v300.switch.ch (130.59.36.198)  459.720 ms  439.574 ms  439.804 ms
24  * * *
25  swifr2-g2-3.switch.ch (130.59.36.105)  667.153 ms  769.497 ms  759.936 ms
```

Now that the packet is through this Swiss gateway we can resume the 6to4 traceroute

```
$ traceroute6 www.potaroo.net
traceroute6 to www.potaroo.net (2401:2000:6660::2) from 2002:79da:c8e7:1::1, 64 hops max, 12 byte
packets
 1  * * *
 2  swibe2-g2-5.switch.ch  815.134 ms  999.826 ms  919.917 ms
```

(That's the 6to4 relay operated by SWITCH in Switzerland)

```
 3  swiba2-10ge-1-2.switch.ch  839.899 ms  859.922 ms  899.888 ms
 4  swiez2-10ge-5-4.switch.ch  890.054 ms  879.706 ms  1140.280 ms
 5  swiix2-10ge-3-1.switch.ch  1194.231 ms  409.655 ms  411.672 ms
```

(The path now traverses Zurich, Paris, Amsterdam, London, New York, Los Angeles and Sydney)

```
 6  10gigabitethernet1-4.core1.zrh1.he.net  398.613 ms  399.806 ms  399.857 ms
 7  10gigabitethernet3-2.core1.fra1.he.net  400.190 ms  399.710 ms  420.094 ms
 8  10gigabitethernet1-4.core1.ams1.he.net  399.710 ms  399.772 ms  409.958 ms
 9  10gigabitethernet1-4.core1.lon1.he.net  409.962 ms  419.895 ms  399.748 ms
10  10gigabitethernet2-3.core1.nyc4.he.net  420.002 ms  399.764 ms  400.021 ms
11  10gigabitethernet5-3.core1.lax1.he.net  400.087 ms  399.705 ms  400.096 ms
12  10gigabitethernet1-3.core1.lax2.he.net  399.905 ms  399.893 ms  399.908 ms
13  ge-1-3-2-140.bdr01.sjc01.nsw.vocus.net.au  779.955 ms  769.599 ms  770.020 ms
14  ge-0-2-2.cor02.syd03.nsw.vocus.net.au  770.146 ms  769.700 ms  770.154 ms
15  ge-0-0-0.bdr01.bne02.qld.vocus.net.au  799.783 ms  779.797 ms  810.187 ms
16  2402:7800:10:1::12  789.789 ms  789.740 ms  810.140 ms
17  ge-0-0-3.bdr01.bne01.qld.vocus.net.au  789.776 ms  789.573 ms  790.090 ms
18  2402:7800:10:1::9  789.883 ms  789.747 ms  789.802 ms
19  apnic.ipv6.brisbane.pipenetworks.com  779.966 ms  780.121 ms  779.596 ms
20  2001:dc0:e002:4608::1  780.204 ms  789.724 ms  779.968 ms
21  www.potaroo.net  789.915 ms  778.933 ms  779.962 ms
```

What about the return path?

```
$ traceroute6 2002:79da:c8e7:1::1
traceroute6 to 2002:79da:c8e7:1::1 (2002:79da:c8e7:1::1) from 2401:2000:6660::2, 64 hops max, 12 byte
packets
 1  2401:2000:6660::254  0.477 ms  0.367 ms  0.360 ms
 2  2001:dc0:e002:4608::2  0.611 ms  0.539 ms  0.482 ms
 3  aarnet.ipv6.brisbane.pipenetworks.com  0.989 ms  0.898 ms  0.990 ms
```

(This packet now enters the IPv4 network, in Brisbane, Australia. The massive elapsed time for the next IPv4 hop, a further 400 ms of delay is enough time to traverse the Pacific Ocean – twice! However, a less prosaic but probably more accurate reason is that the last hop is a 3G mobile broadband connection and while the clamed bandwidth of these 3G networks may be high, the latency and jitter in the radio network are both pretty shocking!)

```
 4  2002:79da:c8e7:1::1  407.762 ms  399.680 ms  390.015 ms
```

The particular mode of failure being observed here in these connection failures is not failure on the outbound path. The initial SYN packet has been passed to the measurement host, so we can be reassured that in these cases the outbound path is functioning. The mode of failure seen here is most likely to be a failure in the return path. (However, is does lead to the observation that there is a second class of 6to4 failure that is not directly measureable using this measurement technique, that of path failure of the outbound path.)

This return path failure that we can observe can be further distinguished into either a failure of the 'return' 6to4 relay, or a path failure closer to the original 6to4 host. In an effort to eliminate the possibility of failure in the return relay, a 6to4 relay has been placed within the measurement system, so that all packets being sent to destinations in 2002::/16 are encapsulated in IPv4 before being forwarded, and are sent via IPv4 from the server directly to the 6to4 host. Presumably, if the 6to4 host's IPv4 address is reachable, then the 6to4 tunnelled packet is also reachable, and we should not see failures in the return path to the 6to4 host.

However, this is not the case. In this experiment an average of some 80 6to4 connections 'failed' in each 24 hour period even with the return path 6to4 relay co-located in the measurement server.

The measurement system is a dual stack system, and the 6to4 hosts are also dual stack hosts, so it is an option for the host to fallback to use IPv4. Because the host's IPv4 address is embedded into the 6to4 IPv6 address its possible to also measure the extent to which we observe 6to4 hosts performing fallback to IPv4. The number of successful 'fallback' connections, using IPv4 numbered an average of 57 connections per day. That implies that some 75% of the 'failed' 6to4 connections successfully managed to fallback to IPv4 and complete a connection, showing that the return IPv4 path was a viable path. So if the return path 6to4 relay is located on the measurement system, why can the server not reach the 6to4 host with IPv6-in-IPv4 packets, yet can each the same host with TCP-in-IPv4 packets?.

Firstly, lets look at the other 25 of the failed 6to4 connections where there is no observed fallback to IPv4. Further investigation shows that the 25% failure rate where there is no observed fallback to IPv4 appears to be a result of using 6to4 from a private environment, where the IPv4 address is not reachable either. The most likely explanation is when IPv4 public address space is used behind a NAT, and a host within this private network configures itself with IPv6 using 6to4. Because all external communication is via some form of NAT, the attempt to direct a return packet to the host's IPv4 address will fail at the NAT. The host itself may not be directly aware of the problem, as, in this case, the fallback to an IPv4 path will use the NAT and complete the connection. Because in this case the connection is made using a different IPv4 address from that used in the 6to4 connection attempt, it is not possible to correlate the connection attempts in each protocol.

What about that other 75% of failures, where the IPv4 address is reachable? Why is the 6to4 connection failing? Here the most likely explanation lies in the use of filters and firewalls close to the 6to4 end host. The IPv4 composition of these 6to4 packets uses protocol number 41, encapsulated IPv6 in IPv4 (RFC 2473), as distinct from the TCP or UDP transport protocols. A common firewall configuration is to block all "unknown" IP protocols, which can include protocol 41. This form of inbound packet filtering would result in the asymmetric condition of being able to send outbound packets, but unable to receive any inbound packets.

What is happening in these clients systems that fail in their attempt to use 6to4 and then have to fallback to Ipv4? The most obvious result is that they experience slow response. It is not unusual to see a 20 second "hold" time in the packet capture between the initial connection attempt and the successful fallback to IPv4.

This is a typical example of a failover as seen at the measurement server:

```
00:00:00.00 IP6 2002:c000:201::c000:201.49645 > 2401:2000:6660::2.80: S
00:00:03.00 IP6 2002:c000:201::c000:201.49645 > 2401:2000:6660::2.80: S
00:00:06.00 IP6 2002:c000:201::c000:201.49645 > 2401:2000:6660::2.80: S
00:00:12.01 IP 192.0.2.1.52211 > 203.133.248.2.80: S
```

The elapsed time from the initial IPv6 connection attempt to the point of fallback to IPv4 is some 19 seconds in this case.

This is obviously not the best of outcomes. It would be good if everyone would review their filter settings and if they are using 6to4 they should verify that their local filters and firewalls permit incoming IPv4 packets using protocol 41. But that's just wishful thinking!

Conclusion

The very high failure rate of 6to4 that we observe in today's dual stack Internet, of some 9.5% of all IPv6 connection attempts and 13% of all 6to4 connection attempts, is certainly a serious concern when contemplating how to transition much of today's Internet infrastructure to IPv6. However, for as long as we can continue to operate in dual stack mode and use IPv4 as a backup, then the vast majority of these failed connection attempts in IPv6 are resolved in IPv4, and the underlying dual stack failure rate is far lower. A related experiment conducted by Tore Andersen points to an underlying failure rate where the end host cannot reach the dual stack server and fallback to IPv4 either fails or does not happen, at around 0.1% (http://fud.no/ipv6/).

The best advice to offer here is that the most useful way to use auto-tunnelling techniques, including 6to4 and Teredo is to put it as a last resort and only use it when all other connection attempts, including IPv4, have failed. That's the default configuration used by the more recent versions of Windows (Vista and 7) and the most recent version of the MAC OS X operating system (10.6.5). Ideally, IPv4 hosts should prefer to use IPv4 to connect to dual stack servers, and should only fall back to try auto-tunnelling approaches when the remote site is an IPv6-only site.

Measurement Methodology

The objective was to measure the success rate of all incoming connections to the web server instance running on TCP port 80 at www.potaroo.net. A packet capture was performed on the server, capturing the first 200 octets of all incoming packets directed to the local port 80, in both IPv4 and IPv6, using the tcpdump (www.tcpdump.org) utility:

```
tcpdump -i em0 -s 100 -w syndump.pcap dst port 80
```

A successful connection will show up in the generated packet logs as:

1. Incoming TCP SYN to initiate the connection, and the following packet will be recorded:

   ```
   04:15:51.680889 IP 192.0.2.1.50378 > 203.133.248.2.80: S
   ```

2. The local host will respond with a SYN ACK TCP packet

3. The remote host will ACK this packet, and the following packet will be recorded:

   ```
   04:15:51.873278 IP 192.0.2.1.50378 > 203.133.248.2.80: . ack
   ```

   Note that the source port addres of this second packet, 50378 in this case, matches that of the initial SYN packet, indicating that this second packet is part of the same TCP session.

   The session will then continue with the data transfer.

If the connection fails the local host will not see the ACK after the initial SYN.

The measurement approach looks at all connections (incoming packets with the TCP SYN flag set) and looks for subsequent TCP ack packets that use the same remote IP address and TCP port address set.

If a subsequent TCP ack packet from the remote end is found then connection is considered to be successful. If there is no further packets following the initial SYN attempts, then connection is considered to have failed.

The potential skew effects of repeated transmissions of the initial SYN, subsequent connections and similar are addressed by collecting connection statistics per unique remote source address per 24 hour period.

The limitation of this approach is that it can only detect connection failures on the return path from the server back to the host. Connection failures on the forward path, from the host to the server are not detectable in this manner.

Disclaimer

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