The ISP Column
A monthly column on things Internet

April 2010

Geoff Huston

Measuring More IPv6

Further studies on the amount of end-to-end IPv6 capability in today's Internet reveals that the extent of full end-to-end IPv6 capability in today's Internet is now at a level of 5% of all end systems, at least within the scope of the systems studied here. This number is now at a level where the level of IPv6 deployment is now passing from mere statistical interest to mainstream commercial importance.

Studies of IPv6 Deployment

Back in April 2008 I looked at the various ways that we could use to measure the deployment of IPv6 (http://www.potaroo.net/ispcol/2008-04/ipv6.html). This work has been further developed by the OECD, and a very well researched OECD report, "Internet Addressing: Measuring Deployment of IPv6" has been published this month (http://www.oecd.org/dataoecd/48/51/44953210.pdf). Before I say any more, I urge you to read through this informative and well researched report, prepared by Karine Perset of the OECD's Directorate for Science, Technology and Industry.

This month I'd like to look at just one aspect of the OECD report, namely using http web server statistics to measure IPv6 deployment. I'd like to quote the relevant section of the OCED report to provide some background here:

> *Proportion of visitors that use IPv6 if given a choice of dual stack service point*
>
> The growth of the proportion of users who connect via IPv6 to reach a dual stack service point indicates end-user capability to complete an IPv6 connection when there is a choice between IPv6 and IPv4 to reach the service. The choice of using IPv6 depends on whether an application on the user side ("client" side) is configured to use IPv6 (often by default) and whether the target application, target operating system, and the connectivity between the two end-points all allow the use of IPv6.
>
> Some organisations with dual-stack web servers collect usage statistics. They record, over time, the numbers of distinct IPv4 and IPv6 query addresses per day on these dual-homed web servers. For example, APNIC, RIPE NCC, and ICANN collect dual stack statistics.
>
> Several caveats warrant noting. Widespread NAT use in IPv4 undercounts IPv4 host counts so that the number comparing IPv6 to IPv4 is a maximum. Visitors to the APNIC and RIPE sites are likely to be more sophisticated technically than average users on the Internet, therefore they are likely to have comparatively more IPv6 clients.
>
> The server will record an IPv6 transaction only if all of the following conditions are met: i) the client has an IPv6 stack; ii) the client's application is configured with IPv6 support; iii) the client's DNS configuration is able to perform an IPv6 address query; and iv) the client and server can communicate end- to-end using IPv6.46 In other words, this measurement will only succeed if all the intermediate components of the connection are configured to support IPv6. Therefore, this metric would be a good indicator of the total level of IPv6 deployment capability across all components of the network.
>
> The data used in this section relate to the use of the APNIC web site, www.apnic.net, and the RIPE web site, www.ripe.net. These web sites have both IPv4 and IPv6 addresses and have been dual homed on both IPv4 and IPv6 networks for over five years. The approach used to measure the relative use of IPv6 to IPv4 was to count the number of unique source addresses visiting these websites each day and to look at the ratio of the number of unique IPv4 source addresses to the number of unique IPv6 source addresses.

Figure 32 shows the daily ratio of IPv6 to IPv4 source addresses that have accessed the APNIC and RIPE websites since 1 January 2004. IPv6 users seem to have started to increase in 2007 and even more so mid-2008, to reach 1% of visits at year-end 2009. Given the considerable variation in the data from day to day, a scatter plot is used to ensure that the trends in the data are visible as well as the day-to-day variation.

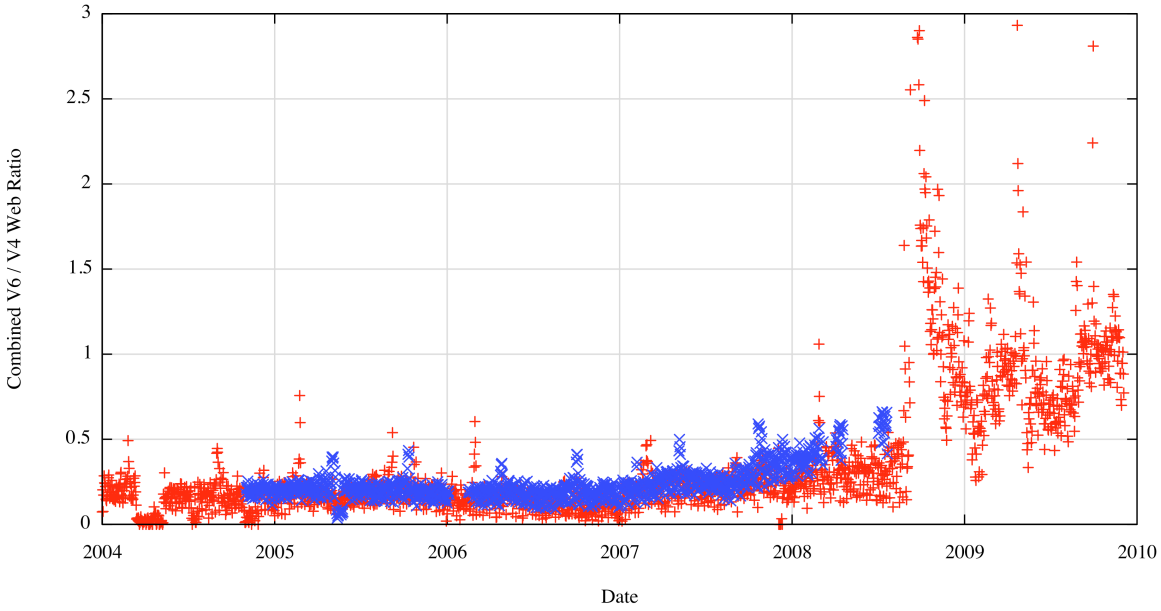

Figure 32. IPv6 / IPv4 Web Access Ratio

"INTERNET ADDRESSING: MEASURING DEPLOYMENT OF IPv6." OECD, April 2010

Approaches to Measurements

The approached described above is an instance of a *passive measurement approach*. In this case the service point id s hosted on a dual stack service platform, and the URL resolves to both an IPv4 A record and an IPv6 AAAA record.

```
$ dig www.apnic.net IN ANY

www.apnic.net.              300     IN      A       202.12.29.230
www.apnic.net.              300     IN      AAAA    2001:dc0:2001:11::211
```

The http server's logs records the source addresses of all transactions, so the data analysis is a simple process of enumerating all distinct IPv4 source addresses and all distinct IPv6 source addresses for each day and generating a ratio of the two values. This is the data set that has been used to generate the figure above.

The question that this measurement approach is attempting to answer is "When given the choice how many end clients will be able to use IPv6 to access this web page?" The reason why this is relevant to measuring IPv6 deployment is based on the general perception that when given a choice an end system will prefer IPv6 over IPv4.

As an example, here is a dump of a browser fetch from a dual-stack site where the IPv6 address of the site is deliberately invalid. The trace shows the local client attempting to use IPv6 for the connection, and attempting 7 times to complete the connection using IPv6, and only then reverting to IPv4.

The client is a MAC OSX 10.6, the browser is Safari

Client attempts to open http://v6broken.v6trans.potaroo.net/1x1.png

1. The client attempts to open a TCP session with the server

```
15:17:50.051687 IP6 (hlim 64, next-header TCP (6) payload length: 44)
2001:44b8:7549:b90:226:b0ff:fef0:5d2a.53810 > 2401:2000:6660::dead.http: Flags [S],
cksum 0x4fda (correct), seq 2341559885, win 65535, options [mss 1440,nop,wscale
2,nop,nop,TS val 517508637 ecr 0,sackOK,eol], length 0
```

2. 1 second later the client resends the IPv6 SYN

```
15:17:51.021520 IP6 (hlim 64, next-header TCP (6) payload length: 44)
2001:44b8:7549:b90:226:b0ff:fef0:5d2a.53810 > 2401:2000:6660::dead.http: Flags [S],
cksum 0x4fd1 (correct), seq 2341559885, win 65535, options [mss 1440,nop,wscale
2,nop,nop,TS val 517508646 ecr 0,sackOK,eol], length 0
```

3. 4 subsequent connection attempts at 1 second intervals

```
15:17:52.022296 IP6 (hlim 64, next-header TCP (6) payload length: 44)
2001:44b8:7549:b90:226:b0ff:fef0:5d2a.53810 > 2401:2000:6660::dead.http: Flags [S],
cksum 0x4fc7 (correct), seq 2341559885, win 65535, options [mss 1440,nop,wscale
2,nop,nop,TS val 517508656 ecr 0,sackOK,eol], length 0
15:17:53.023091 IP6 (hlim 64, next-header TCP (6) payload length: 44)
2001:44b8:7549:b90:226:b0ff:fef0:5d2a.53810 > 2401:2000:6660::dead.http: Flags [S],
cksum 0x4fbd (correct), seq 2341559885, win 65535, options [mss 1440,nop,wscale
2,nop,nop,TS val 517508666 ecr 0,sackOK,eol], length 0
15:17:54.023851 IP6 (hlim 64, next-header TCP (6) payload length: 44)
2001:44b8:7549:b90:226:b0ff:fef0:5d2a.53810 > 2401:2000:6660::dead.http: Flags [S],
cksum 0x4fb3 (correct), seq 2341559885, win 65535, options [mss 1440,nop,wscale
2,nop,nop,TS val 517508676 ecr 0,sackOK,eol], length 0
15:17:55.024624 IP6 (hlim 64, next-header TCP (6) payload length: 44)
2001:44b8:7549:b90:226:b0ff:fef0:5d2a.53810 > 2401:2000:6660::dead.http: Flags [S],
cksum 0x4fa9 (correct), seq 2341559885, win 65535, options [mss 1440,nop,wscale
2,nop,nop,TS val 517508686 ecr 0,sackOK,eol], length 0
```

4. 7th and final IPv6 connection attempt 2 seconds later

```
15:17:57.026221 IP6 (hlim 64, next-header TCP (6) payload length: 44)
2001:44b8:7549:b90:226:b0ff:fef0:5d2a.53810 > 2401:2000:6660::dead.http: Flags [S],
cksum 0x4f95 (correct), seq 2341559885, win 65535, options [mss 1440,nop,wscale
2,nop,nop,TS val 517508706 ecr 0,sackOK,eol], length 0
```

5. Change to IPv4 after a total of 7 seconds of attempting to use IPv6

```
15:17:57.072356 IP (tos 0x0, ttl 64, id 22727, offset 0, flags [DF], proto TCP (6),
length 64) xxx.potaroo.net.53811 > wattle.apnic.net.http: Flags [S], cksum 0x6b52
(correct), seq 1997078952, win 65535, options [mss 1460,nop,wscale 3,nop,nop,TS val
517508707 ecr 0,sackOK,eol], length 0
```

What this example illustrates is that for some dual stack client's environments the client will be quite persistent in attempting to complete the connection with a dual stack server using IPv6, and will only use IPv4 as a second choice.

How widespread is this behaviour? Do other configurations prefer to use IPv4 when there is a choice? Unfortunately this passive measurement approach will not answer these questions. The question that the passive approach can answer is: "What proportion of clients will elect to use IPv6 when given a choice?" The question that is of interest is: "What proportion of clients are capable of using IPv6?"

To answer this second question we need to look at a slightly different approach. This is an *active measurement approach*, that entails enlisting the subject to perform a particular set of tests and then collecting the output from these experiments. The essential difference here is that in a passive measurement approach the various behaviours of the client end systems are being observed when exposed to identical conditions, while active measurement allows the conditions to be altered for each subject, and for the subject itself to be part of the experiment.

The Link Test Experiment

In this experiment we've added the following JavaScript to a number of web pages:

```
<div id="gih-analysis"></div>
<script type="text/javascript">

function gih_analysis() {
 var div = document.getElementById('gih-analysis');
 var iframe = document.createElement('iframe');
```

```
 iframe.setAttribute('src', 'http://www.potaroo.net/linktest-potaroo.php');
 iframe.setAttribute('width', '1');
 iframe.setAttribute('height', '1');
 iframe.setAttribute('frameborder', '0');
 div.appendChild(iframe);

 if (window.removeEventListener) {
   window.removeEventListener(name, gih_analysis, false);
   }
 else if (window.detachEvent) {
   try {
     window.detachEvent('on' + name, gih_analysis);
     } catch (e) {}
   }
 }

try {
 if (window.addEventListener) {
   window.addEventListener('load', gih_analysis, false);
   }
 else if (window.attachEvent) {
   window.attachEvent('onload', gih_analysis);
   }
 } catch (e) {}
</script>
```

The JavaScript wrapper around the original html fragment of:

```
<iframe src=http://www.potaroo.net/linktest.php width="1" height="1" frameborder="0"></iframe>
```

is intended to ensure that the load of this link test PHP script is delayed until after all the visible parts of the web page have been loaded, so that this link test occurs in the background once the page itself has been displayed.

The PHP script that is loaded is:

```
<?php
$source=$_SERVER['REMOTE_ADDR'] ;
$referer = $_SERVER['HTTP_REFERER'] ;
$seq = mt_rand();
$tim = time() ;

echo "<img src=\"http://$seq.ipv6only.potaroo.net/1x1.png?id=6:$tim:$seq
  &src=$source&ref=$referer\" height=\"1\" width=\"1\" alt=\"\">";
echo "<img src=\"http://$seq.dualstack.potaroo.net/1x1.png?id=D:$tim:$seq
  &src=$source&ref=$referer\" height=\"1\" width=\"1\" alt=\"\">";
echo "<img src=\"http://$seq.ipv4only.potaroo.net/1x1.png?id=4:$tim:$seq
  &src=$source&ref=$referer\" height=\"1\" width=\"1\" alt=\"\">";
?>
```

The script uses three domain names, *ipv6only.potaroo.net*, *dualstack.potaroo.net* and *ipv4only.potaroo.net*, and generates a random domain name within each of those domain names. For example, after a client executes the javascript and the server executes the PHP script, the result is that a client would load the following html fragment:

```
<img src="http://567321.ipv6only.potaroo.net/1x1.png?id=6:1271030396:567321
   &src=192.0.2.1&ref=http://www.potaroo.net" height="1" width="1" alt="">
<img src="http://567321.dualstack.potaroo.net/1x1.png?id=D:1271030396:567321
   &src=192.0.2.1&ref=http://www.potaroo.net" height=”1" width="1" alt="">
<img src="http://567321.ipv4only.potaroo.net/1x1.png?id=4:1271030396:567321
   &src=192.0.2.1&ref=http://www.potaroo.net” height="1" width="1" alt="">
```

The use of the random numbers in the domain name is to ensure that clients are not in a position to cache these objects, or cache the DNS resolution of the domain names. Each time the client visits the page it will perform the DNS lookups and perform the fetches of the images because the domain names will change with every execution of this PHP script.

The use of the three distinct images are intended to test the capabilities of the client. The first image is only accessible using IPv6 as the transport, as the domain name only has a AAAA IPv6 address record:

```
$ dig 23423.ipv6only.potaroo.net IN ANY

23423.ipv6only.potaroo.net. 5       IN       AAAA     2401:2000:6660::2
```

The second image is a “conventional” dual stack arrangement:

```
$ dig 23423.dualstack.potaroo.net IN ANY

23423.dualstack.potaroo.net. 5      IN      A       203.119.0.116
23423.dualstack.potaroo.net. 5      IN      AAAA    2401:2000:6660::2
```

The third image is accessible only using IPv4:

```
$ dig 23423.ipv4only.potaroo.net IN ANY

23423.ipv4only.potaroo.net. 5       IN      A       203.119.0.116
```

As a side note, this approach can be further extended to test the capabilities of the client’s DNS resolvers in a similar fashion by adding a fourth image from a sub domain of v6trans.potaroo.net. This domain can only be resolved if the DNS resolver is capable of undertaking DNS queries using IPv6 transport.

$ dig 23423.v6trans.potaroo.net IN ANY

23423.v6trans.potaroo.net. 86400 IN AAAA 2401:2000:6660::2

Some DNS resolvers do not yet have such capability to resolve DNS using IPv6 transport, such as Google’s DNS resolver service on 8.8.8.8:

$ dig 23423.v6trans.potaroo.net IN ANY @8.8.8.8

; <<>> DiG 9.7.0 <<>> 23423.v6trans.potaroo.net IN ANY @8.8.8.8
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 4493
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;23423.v6trans.potaroo.net. IN ANY

;; Query time: 912 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Mon Apr 12 06:38:27 2010
;; MSG SIZE rcvd: 43

What this test does is to force the client to perform three simple fetches, all directed to the same Dual Stack server. The first is IPv6 only, and if the client does not support IPv6 then the fetch will fail. The second is dual stack, leaving the choice of transport to the client. The final image is IPv4 only.

Measuring IPv6 Capability

What results should we expect from this experiment? If the data in Figure 32 is a reliable indicator, and if all clients who can undertake IPv6 will prefer IPv6 then we should expect some 2% of all clients will be capable of fetching the IPv6 only image, and the same 2% will use IPv6 to retrieve the dual stack image, and 100% of all clients will fetch the IPv4 image.

In an experiment conducted over the period from the 20th March 2010 until the 12th April 2010, with some 85,000 individual reachability experiments conducted, the following data was gathered on dual stack preference:

Dual Stack: V4	96.5%
Dual Stack V6	2.1%
Dual Stack: Not Loaded	1.4%

Aside from a slightly higher than anticipated number of “Not Loaded” cases, the 2.1% value appears to be consistent with the trends shown in Figure 32 from the OECD report.

But is this 2.1% value reflective of the population of end systems that are capable of end-to-end IPv6? Not really. More than twice the number of end systems were capable of loading an IPv6 object when the object was only accessible via IPv6, as shown in the following table:

Could load IPv6 only:	4.9%
Could load IPv4 only:	98.1%
Could load Neither:	1.2%

This table shows the critical data point. What it is showing that when end systems are forced to load an http object that is only accessible via IPv6 then 5% of the end systems sampled here were capable of completing the load using IPv6.

Another view of these numbers is to look at those end clients who were able to load both the IPv4 only objects and the ipv6 only objects. What protocol did they use to load the dual stack object?

IPv4 only, IPv6 only, and IPv4 Dual Stack:	2.58%
IPv4 only, IPv6 only, and IPv6 Dual Stack:	1.71%
IPv4 only, IPv6 only, and no Dual Stack:	0.03%

There are also a small set of clients who appeared to be IPv6 only:

No IPv4 only, IPv6 only, and IPv6 Dual Stack:	0.27%

In other words, of those clients who were capable of supporting end-to-end IPv4 and IPv6, more clients opted to use IPv4 when loading a dual stack object than used IPv6. It seems that the amount of end-to-end IPv6 capability is greater than the simple dual stack test would indicate.

Can we learn more from this measurement approach? There are two more aspects of information that this test can show. The first is the IPv6 source address, which can indicate whether the client is using a 6to4 tunnel, a Teredo tunnel, or a native IPv6 address. The second is the operating system signature that the browser collects from the http request header.

The first table here is a breakdown of the IPv6 capability and preference by address type.

V6 Address Type	Could retrieve V6 Objects	Use V6 in Dual Stack
Unicast	41.3%	93%
6to4	50.8%	4.3%
6RD (Free.fr)	0.5%	86%
Teredo	7%	8%

The first column shows the relative ratio of the various address types, showing that the majority of IPv6 access came from 6to4 source addresses (50.8%), and the next most common access form was via the use of a non-tunnelled Unicast IPv6 address. The second column shows the proportion of clients within each address type that use IPv6 when presented with a dual stack object. In other words, while 50.8% of all clients who could retrieve IPv6 objects used 6to4, of those 6to4 clients only 4.3% of them used IPv6 in preference to IPv4 when presented with a Dual Stack object. Interestingly, of the two most popular automatic tunnelling types, 6to4 and Teredo, clients using these access mechanisms generally prefer to use IPv4 when presented with a Dual Stack object, while most of the unicast and 6RD clients will prefer to use IPv6 in the same Dual Stack scenario.

A similar breakdown can be performed by browser type:

Browser Type	Could retrieve V6 Objects	Use V6 in Dual Stack
Explorer	53.6%	34.8%

Firefox	29.5%	46.9%
Chrome	10.3%	42.7%
Safari	4.2%	80.7%
Opera	1.6%	48.5%
Mozilla	0.9%	83.3%

This shows that of the clients that could retrieve IPv6 objects, 53.6% of these clients were tagging their requests with the Explorer browser type. Of these clients who were using Explorer as their browser only one third, or 34.8%, would prefer to use IPv6 when presented with a Dual Stack object. Clients who were using the Safari browser preferred to use IPv6 for Dual Stack objects in 80% of all cases.

The same data can be used to infer the operating system platform.

Browser OS Type	Could retrieve V6 Objects	Use V6 in Dual Stack
Windows	86.1%	34.4%
Mac OSX	8.0%	85.4%
Linux	5.5%	91.2%
FreeBSD	0.2%	70.0%
SunOS	0.1%	0.0%
Symbian	0.0%	-
iPhone	0.0%	-
Blackberry	0.0%	-

Notably it appears that Windows platforms appear to prefer to use IPv4 when given a choice via a Dual Stack scenario when they are capable of supporting IPv6, while the Mac OSX platforms tend to prefer V6 it is can be used. However, only 4% of the 6to4 V6 clients and 8% of the Teredo clients prefer V6 when presented with a Dual Stack choice. Is there a correlation between operating system platform and whether or not the platform is equipped with tunnelled IPv6 access?

Browser OS Type	Unicast V6	6to4	Teredo	6RD (Free.fr)
Windows	34%	56%	9%	0.3%
Mac OSX	76%	21%	1%	1%
Linux	78%	11%	8%	3%
FreeBSD	100%			
SunOS	100%			

What Are We Seeing Here?

There are appear to be three distinct approaches to IPv6 that are visible in the network today.

The first approach is exemplified by the iPhone and apparently other mobile platforms and their support networks, namely not to support IPv6 either as a native capability or as a tunnelled capability. I suspect that the technical nature of the web sites that use this particular IPv6 link test tends not to attract the vast majority of the mobile device population, so this particular class of users is possibly under-represented here.

The second approach is to support native IPv6, and allow the user to optionally load IPv6 tunnelling support using 6to4 or Teredo or similar variants. The Mac OSx and Unix platforms are examples of this approach. Such systems also appear to be configured to prefer to use IPv6 when presented with a Dual Stack choice, although the systems will vary in the amount of time they are prepared to spend establishing an IPv6 connection before falling back to IPv4.

The third approach appears to be to support IPv6 tunnelling, but to use this as a last resort. The Windows platforms appear to use this approach. In this case the platform appears to have tunnelling supported configured, and it will prefer IPv6 in a Dual Stack situation only when there is native IPv6 connectivity. Otherwise the platform will elect to use IPv4, and will only use the IPv6 auto-tunnelling mechanisms of Teredo or 6to4 to reach an IPv6-only destination.

The picture with respect to IPv6 capability is evidently somewhat better than the dual stack access figures would suggest. At present, some 2% of clients of these web pages will use IPv6 end-to-end when presented with a

Dual Stack object to load. However, when presented with an IPv6-only object, a further 2.9% of clients can retrieve the object using IPv6. This latter category of clients appear to use auto-tunnelling for this IPv6-only retrieval, and the reason for the difference between Dual Stack and IpP6 only retrievals is that these auto-tunnel platforms appear to prefer using IPv4 over using auto-tunnelled IPv6.

So if the IPv6 deployment question is: “What proportion of clients will **prefer to use IPv6 end-to-end** when given a choice using Dual Stack?” then today’s answer is **2%** of all clients.

However, if the question is: “What proportion to clients **are capable of using IPv6 end-to-end**?” then today’s answer is significantly higher, at **4.9%** of all clients.

So, where are we with IPv6?

It looks like today’s answer is quite heartening: some **5%** of Internet’s end systems are capable of supporting end-to-end IPv6. And that is surely good news!

Acknowledgement

I'd like to acknowledge Tore Anderson of Redpill Linpro AS, whose reports to the IPv6 Operations list (ipv6-ops@lists.cluenet.de) on "IPv6 brokeness" inspired this work. Tore's approach to testing client systems with Dual Stack services is documented at http://thread.gmane.org/gmane.org.operators.ipv6/2636. I've taken his approach and extended it slightly, so I'd like to thank Tore here for his idea of performing this form of active probing of client capabilities.

Further Reading

In addition to the OECD report noted above, RIPE Labs contains a recent summary of IPv6 measurement efforts at http://labs.ripe.net/content/ipv6-measurement-compilation.

RIPE Labs also contains a good report on their efforts to measure IPv6 capability, including DNS resolution capability over Ipv6. This report can be found at http://labs.ripe.net/content/measuring-ipv6-web-clients-and-caching-resolvers-part-1. In addition, the technique that was used for these measurements is described at http://labs.ripe.net/content/measuring-ipv6-web-clients-and-caching-resolvers-part-3-methodology.

The April 2010 Passive and Active Measurement Conference, PAM 2010, included a paper by Google on the same topic (http://www.pam2010.ethz.ch/papers/full-length/15.pdf).

Disclaimer

The above views do not necessarily represent the views or positions of the Asia Pacific Network Information Centre.

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