The ISP Column
A monthly column on things Internet

May 2011
Geoff Huston

Testing IPv6 for World IPv6 Day

Over the past few months in this column I've devoted some time to looking at IPv6. The reason why is pretty simple. It's getting a bit serious with IPv4 exhaustion! At the end of April 2011 the general use IPv4 address pool held by APNIC, the Asia Pacific Regional Internet Registry ran dry. Obviously precise predictions vary, but over the coming months in 2011 its reasonable to anticipate the same "last chance" pressures for IPv4 addresses will drain the available address pools at the RIPE NCC and ARIN.

It's now time to look hard at the practical issues that we will encounter with deployment of IPv6, and in support of this it has been proposed to launch a "test flight" of IPv6 on *World IPv6 Day*. On June 8 2011 a number of web sites, including Google, Facebook, Yahoo! and Bing, will convert their main web pages to be reachable over both IPv4 and IPv6. This 24 hour test, from 00:00 until 23:29 UTC on June 8, is intended to act as focal point for this industry to prepare their service offerings for IPv6 on the day, and, more generally, to accelerate deployment plans for IPv6 in all parts of the Internet.

Why do we need a special day to do this? Why aren't we well prepared already? After all the original discussions about the exhaustion of IPv4 address space and the needs for a successor IP protocol started back in 1990, more than twenty years ago, and the specification of IPv6 was largely completed by 1998. IPv6 can be called many things, but it definitely should not be called a surprise!

But its been evident for many years that IPv6 has been languishing in a quiet corner. For many years IPv4 exhaustion was a distant prospect and many industry actors were content to await a convincing rational to deploy IPv6 that offered some superior performance or capability that was not possible in IPv4. For many years this search for the so-called "killer-app" for IPv6 distracted many folk.

The simple fact is that IPv6 was deliberately designed to be a conservative incremental step for IPv4. Aside from a massive in increase in the address fields, the other changes involved a change to packet fragmentation handling and packet header forwarding. The essential "core" characteristic of IP as a basic unicast datagram protocol that uses hop-by-hop forwarding remained unchanged, and on top of IPv6 was the same TCP and UDP transports. An application that used the DNS to initiate an TCP network transaction would use precisely the same set of calls to the network sub-layer of resolving a name to an address, using the address to initiate a connection, reading and writing to the connection, and then closing the connection. With a suitably abstracted Application Programming Interface (API) both IPv4 and IPv6 can be supported using precisely the same OPEN, READ, WRITE and CLOSE calls. The application has no requirement to be aware of the underlying IP level datagram protocol being used.

There is no "killer-app" that distinguishes IPv6 from IPv4 in a positive sense, and for many this was enough to push IPv6 onto the shelf of "future plans."

In this period IPv6 became the topic of a number of transitional approaches, including "hop over" approaches which attempted to provide end systems with IPv6 capability even when the local network of the local network service provider did not provide any form of native IPv6 service. These have not proved to be very effective, as recent articles on the extremely high failure rates and poor performance of 6to4 and Teredo have shown.

Given the factors of scant IPv6 deployment, with associated problems of reliability and performance, then many content providers saw IPv6 as a liability. From a content provider perspective IPv4 was sufficient for their needs, Even with widespread use of NATs in CPEs, all potential clients could still reach their content, and there was a synergy of interest between the service providers, equipment vendors and software authors to make content delivery as reliable and efficient as possible. These days enterprises such as Netflix are based on the presence of a capable IPv4 infrastructure that can deliver gigabyte content quickly, while others, like Skype relies on highly reliable and efficient connectivity mesh in IPv4. From the perspective of content, the case for IPv6 is hard to make. It dos not reach any new population of clients. It does not improve the content delivery operation. It does not reduce costs. Indeed it has the potential to make the picture slightly worse. By switching a content delivery platform to operate in dual stack mode a certain proportion of clients, variously estimated in recent studies to be around 0.05% of all clients up to 0.5%, will no longer be able to reach the content at all. This may appear small, but if you are looking at, say, 10 million clients per day on your popular web site, 0.05% is still some 5,000 clients who will see an error message, or, even worse, a white page with an endlessly spinning busy icon.

Now that IPv4 exhaustion is a reality, at some stage in the not too distant future the path between a content provider and a growing number of clients client is either a direct path via IPv6 or a twisted path via a number of third party caches, and various forms of attempts at automated protocol translation. If a web site wants to have direct contact with its client base then equipping the web page with IPv6 in a dual stack configuration is now necessary.

But, in spite of this imperative, the "experiments" with IPv6 on the larger web sites so far have been marked by considerable caution by content providers. Some have using an IPv6 variant of the website with a distinct name, allowing the client base to experiment with IPv6 connectivity by using a distinct DNS name for the IPv6 web site. A variant of this has been the use of "DNS white-listing" where the web client will provide a dual stack response to DNS queries when the query originates from a set of so-called "while-listed" client addresses, while maintaining the appearance of being an Ipv4-only site to all others.

However, these experiments are somewhat limited in context and do not really assist the broad population of network service providers and their clients to identify and work toward resolving any issues that clients may experience in an environment where most, if not all of the web services are provided in a dual stack environment. And in the absence of reliable data there is a tendency to populate the space with myths and superstition.

The superstition in the content world is that waiting is still the optimal response. Content providers can already reach their entire client set using IPv4, and moving the content to an IPv6/IPv4 dual stack context only makes the experience worse for some proportion of clients and setting up the content on a dual stack platform will incur additional costs. The proposition to go from IPv4-only content to dual stack content is perceived as a double negative without any positive offset: it increases cost and annoys a few customers who are perfectly happy today, while it does not make life "better" for anyone right now. Many, indeed most, content providers are simply sticking to IPv4 and waiting. But IPv4 address exhaustion has its own inexorable momentum and waiting is not a viable long term strategy for content providers. So it's time to start to tackle these superstitions about cost and performance and replace superstition with some real data. Is operating a dual stack as expensive as they fear? What will be required to set up a dual stack service platform for content? How many of the existing customers will be affected negatively if the content were to be served from a dual stack content platform? What problems will they encounter? Can we understand this and see what we can do to fix it?

In an effort to coordinate testing of dual stack services on a larger basis, the Internet Society is assisting in coordinating an exercise in deliberately using the Internet as an experimental platform and for 24 hours in June conduct the World IPv6 Day experiment. The idea is to allow a large number of content providers to experiment with provision of content over dual stack platforms all at the same time.

More information on this World Ipv6 Day initiative is available at **http://www.isoc.org/wp/worldipv6day.** There you will find details about the websites that will be turning on IPv6 on June 8, how to join, and information for networks and individuals, including an FAQ.

APNIC has been operating its online services in a dual stack mode for many years, so it has nothing to "turn on" in the context of this World IPv6 Day event. However APNIC is contributing a variant of the tool we've been using to measure IPv6 penetration and dual stack performance to this activity. The problem we are addressing with this particular tool is to allow web site operators to measure the IPv6 and dual stack behavior of their clients without having to switch over their own web site, or even any part of it, to a dual stack operation in advance of World IPv6 Day. How many of the web site's clients would access the site using IPv6? How many clients would use auto-tunneling via 6to4 or Teredo? How many clients would resolve a DNS name if the name servers were accessible only via IPv6?

The tool is available at **http://labs.apnic.net**, and in the rest of this article I'll walk though what it is capable of reporting and how it can be customized.

An IPv6 Measurement Tool

The approach used here is to build upon one of the more widespread web site analysis tools, Google Analytics (analytics.google.com), and adding an additional element to the existing Google Analytics code that is embedded into web content.

To use the APNIC IPV6 test, add the text shown in blue below to your existing Google Analytics statements.

```
<script type='text/javascript'>
  var _gaq = _gaq || [];
    // your google analytics tracking account ID
  _gaq.push(['_setAccount', 'XX-YYYYYYYY-Z']);

   // your domain being tracked
  _gaq.push(['_setDomainName', '.my.dom.ain']);
  (function() {
    var ga = document.createElement('script');
    ga.type = 'text/javascript'; ga.async = true;
    ga.src = ('https:' == document.location.protocol ? 'https://ssl' :
              'http://www') + '.google-analytics.com/ga.js';
    var s = document.getElementsByTagName('script')[0];
    s.parentNode.insertBefore(ga, s);
  })();

  // enable the APNIC IPProtoTest and feed google analytics events
  if ('http:' == document.location.protocol) {
    // Your APNIC identifier code
    var ipproto_user = '746616';
    (function() {
      var iga = document.createElement('script');
      iga.type = 'text/javascript';
      iga.async = true;
      iga.src = 'http://labs.apnic.net/ipprototest.js';
      var is = document.getElementsByTagName('script')[0];
      is.parentNode.insertBefore(iga, is);
     })();
    }
</script>
```

What this script will do is perform three basic tests on the client, all performed in parallel and in the background of the browser's activity. The first is a test presented to the client to fetch a tiny (1 pixel) white gif image where the DNS name of the image is only resolvable to an IPv6 address (an AAAA Resource Record (RR)). The second test is where the same image is accessible via IPv6 and IPv4, i.e. a conventional

dual stack image. The third test is a IPv4-only object retrieval of the same image. The client then reports back to the Analytics Server with the success or failure of each test, and the elapsed taken to retrieve each object, as a "Tracked Event."

The advantage of this approach is that you don't need to make any changes to your content platform. This code tests the IPv6 capability of your clients using a separate test set, and does not require you to set up your own systems on a dual stack platform. If you are already using Google Analytics to track the way clients interact with your content, then this is a simple addition that will allow you to understand the extent to which visitors to your web content are capable of using IPv6 and estimate the level of performance impact that this may have.

Google Analytics Reporting and IPv6 Measurement

The dashboard of the Google Analytics report is shown in Figure 1. To get to the IPv6 Test reports you need to open the Content pages.

Figure 1 – Google Analytics dashboard

Then open the Event Tracking page (Figure 2).

Figure 2 – Google Analytics Content Reports

Then open the full table of tracked events with "view all" (Figure 3).

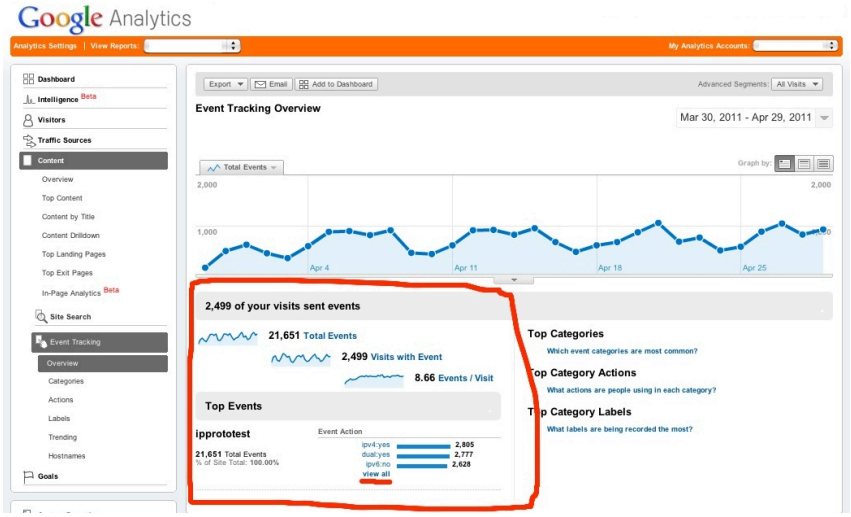

Figure 3 – Google Analytics Tracked Events

This will then display the Event Tracking pages (Figure 4).

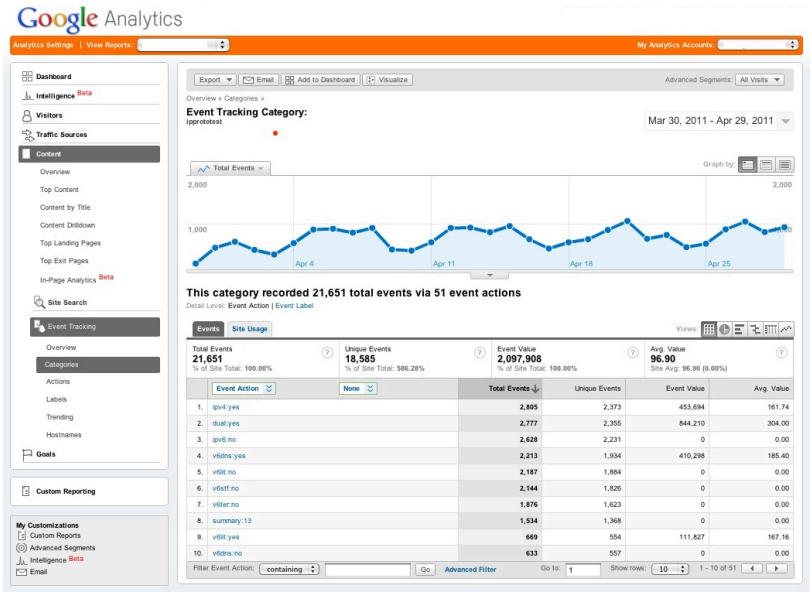

Figure 4 – Google Analytics Tracked Events Table Report

There are two classes of events used in this measurement system: the ***test*** events, which have "yes" and "no" actions, and the ***summary*** event which has a larger set of reported outcomes. We'll look at the summary event later, so for the moment lets concentrate on the test events.

The three core tests are "ipv4", "dual" and "ipv6", and each of these have "yes" and "no" actions.

The event value of the IPv4 test is set to 0.

The event value of the "dual:yes" action is the elapsed time difference, measured in milliseconds, between the retrieval time of the Ipv4 test and the dual stack test. A positive time value indicates that the dual stack test took a longer time. For example, if it takes a client 20 ms longer to retrieve the dual stack object as compared to the IPv4 object, this client will report an Event Value of 20 to the Analytics engine.

Similarly, the event value of the "ipv6:yes" action is the time difference to retrieve the IPv6-only object, as compared to the IPv4 object retrieval time.

The complete set of measureable events is shown in the following table. The first three tests are performed in all cases, while the following four tests can be specifically selected in the javascript code (see the section on "Advanced Scripting" below).

Event Action	Explanation
ipv4:yes/no	The client can/cannot retrieve a IPv4-only URL
ipv6:yes/no	The client can/cannot retrieve a IPv6-only URL
dual:yes/no	The client can/cannot retrieve a dual stack URL
v6dns:yes/no	The client can/cannot retrieve a dual stack URL where the DNS A and AAAA records are served by an IPv6-only DNS nameserver (i.e. V6 DNS resolution is supported)
v6lit:yes/no	The client can/cannot retrieve a IPv6-only URL where the URL is the IPv6 address (i.e. no DNS query is used)
v6stf:yes/no	The client can/cannot retrieve a IPv6-only URL using 6to4
v6ter:yes/no	The client can/cannot retrieve a IPv6-only URL using Teredo

Table 1 – IPv6 Measurement Tracked Events

From the example web sites shown in the above figures, Table 2 shows the event totals for a 3 week period in April 2011.

Event	Count	Avg. Value
ipv4:yes	2034	0
ipv4:no	105	0
dual:yes	2013	-11.83
dual:no	108	0
ipv6:yes	209	65.86
ipv6:no	1914	0

Table 2 – IPv6 Measurement Tracked Event Totals and average values

What these numbers in table 2 are showing is that some 10% of the clients who visited this site were capable of retrieving an IPv6 only object, and the average time to do so was some 65 ms longer than the IPv4 retrieval time. The dual stack retrieval time was some 11 ms faster than IPv4.

This is probably due to the fact that the sequence of tests performed here were fixed in order as IPv6, then dual stack, then IPv4. Many operating systems stagger DNS requests in 10 ms intervals, even if the three tests were started in parallel, an IPv4-only host would emit a DNS query for an A record for the IPv6 object, then wait 10ms, then emit the DNS query for the A record of the dual stack object, then wait a further 10 ms to emit the DNS query for the A record of the IPv4 object. In the case of a dual stack host the DNS sequencing includes both A and AAAA queries, so the retrieval operations are paced in 20 ms intervals. Given that some 10% of the clients have IPv6, then 90% of clients will see a 10ms speed advantage for the dual stack site, while 10% of the clients will see a 20ms speed advantage for the dual stack site. An resultant 11 ms delay between the dual stack and the ipv4 test is not a surprising outcome. The effects of the DNS sequencing could be mitigated by having the javascript set the tests in random order (see the section on Advanced Scripting).

In addition, for each client there is a ***summary*** event, that is the summary of capabilities for that client. This value is a bit vector, where the capabilities of the client as reported as the sum of the values of each individual capability test performed by the client.

Event	ipv4	ipv6	dual	v6dns	v6lit	v6stf	v6ter
Summary val.	1	2	4	8	16	32	64

Table 3 –Summary Event value components

For example, an IPv4-only client with no V6 DNS capability would report a summary value of 5 (ipv4 + dual), and an IPv4 only client with IPv6 DNS resolution capability would report a summary value of 13. (ipv4 + dual + v6dns). A dual stack client with native IPv6 would report a summary value of 31 (ipv4 + ipv6 + dual + v6dns+v6lit), while a Windows client with Teredo enabled would report a summary value of 93 (ipv4 + dual + v6dns + v6lit + v6ter).

The summary record shows a more complete picture of client capability. The data reported in Google Analytics in our example has been reformatted to show the client capabilities recorded in this example.

Event	Count	ipv4	ipv6	dual	dns	lit	stf	ter
summary:13	1,188	■		■	■			
summary:5	268	■		■				
summary:93	167	■		■	■	■		■
summary:85	153	■		■		■		■
summary:31	118	■	■	■	■	■		
summary:0	49							
summary:63	29	■	■	■	■	■	■	
summary:23	26	■	■	■		■		
summary:12	23			■	■			
summary:1	19	■						
summary:9	19	■			■			
summary:29	9	■		■	■	■		
summary:4	7			■				
summary:8	7				■			
summary:95	7	■	■	■	■	■		■
summary:61	5	■		■	■	■	■	
summary:81	5	■				■		■
summary:127	3	■	■	■	■	■	■	■
summary:15	2	■	■	■	■			
summary:16	2					■		
summary:58	2		■		■	■	■	
summary:80	2					■		■
summary:87	2	■	■	■		■		■
summary:20	1			■		■		
summary:21	1	■		■		■		
summary:22	1		■	■		■		
summary:27	1	■	■		■	■		
summary:30	1		■	■	■	■		
summary:62	1		■	■	■	■	■	
summary:69	1	■		■				■
summary:89	1	■			■	■		■
summary:92	1			■	■	■		■
Total (%)	100%	95%	9%	95%	75%	25%	2%	16%

Table 4 –Summary Event value components

The most common configuration, shared by 50% of clients, is IPv4 capability with IPv6 DNS resolution, but no other IPv6 capability. The DNS forwarder provided by their service provider is a capable of generating DNS queries across an IPv6 UDP transport, but the client itself has no IPv6 capability at all. Some 9% of clients have IPv6 capability that allows a fetch of an IPv6-only object, and a further 19% of clients have latent IPv6 capability that is exposed with a literal IPv6 address in the URL (as is commonly observed with Windows Vista and Windows & and their handling of the local Teredo auto-tunneling Ipv6 interface). 16% of clients also have IPv6 capability with Teredo, while2% of clients have IPv6 capability using 6to4.

> It should be noted that there is no single profile of the IPv6 capability of clients when looking at web content. The example site is a site directed to network engineers providing information on the BGP routing table (http://www.cidr-report.org). Other sites will see a markedly different profile of IPv6 capability.

It is also possible to use the Analytics tool to generate some plots of the event data. Figure 5 shows one such plot generated from the example data set that shows a day-by-day breakdown of IPv6 capability of the set of visitors to this web site.

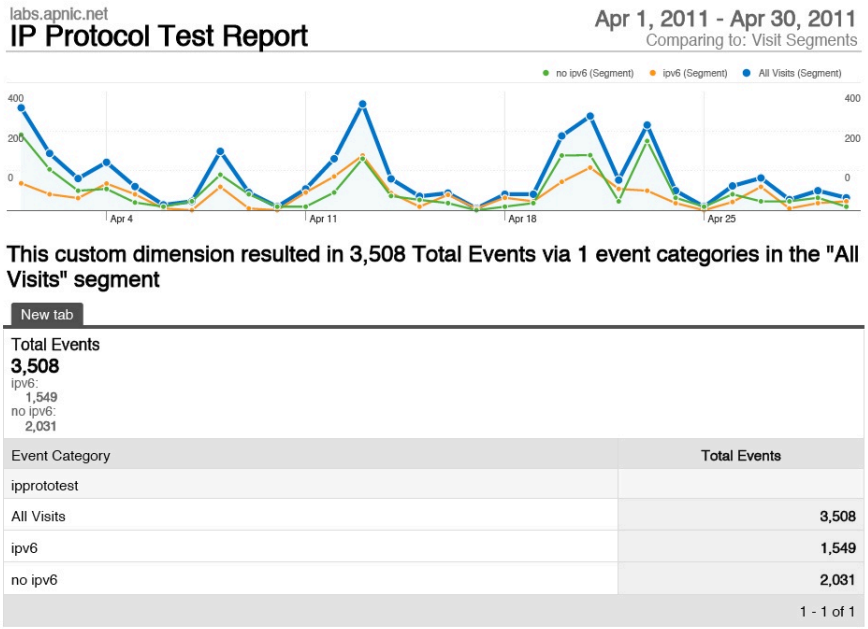

Figure 5 – Google Analytics Tracked Events – IPv6 capability report

Advanced Scripting

It is possible to alter the invocation on the javascript on the web page to invoke additional tests and alter the behavior of the script. The full set of parameters are show in the following alternative script invocation:

```
// enable the APNIC IPProtoTest and feed google analytics events
  if ('http:' == document.location.protocol) {

   var ipproto_user = '111111';         // the user variable

                                        // default state of the options.
   var ipproto_opts = {
     'docookies'        : true,         // test based on noCheckInterval in cookie
     'noCheckInterval'  : 86400000,     // interval to test on, if docoookies is true
     'dov6dns'          : true,         // test v6 dns to a dual-stack URL
     'dov6literal'      : true,         // test a v6 literal URL
     'dotunnels'        : false,        // test for 6to4 and teredo tunnels
     'randomize'        : false,        // by default, sorted test order
     'sampling'         : 1,            // 1/sampling eg sampling=4 1/4th tested
     'userId'           : ipproto_user, // what to log in the collector website
     'callback'         : function_name // prototype function to receive callback
     };

    (function() {
      var iga = document.createElement('script');
      iga.type = 'text/javascript';
      iga.async = true;
      iga.src = 'http://labs.apnic.net/ipprototest.js';
      var is = document.getElementsByTagName('script')[0];
      is.parentNode.insertBefore(iga, is);
     })();
    }
```

docookies If **docookies** is set to "false" the test will be performed every time each client visits this web page. This may bias your analytics reports towards the IPv6 capabilities of the most frequent visitors to the site. If **docookies** is set to "true", then the client is loaded with a controls the frequency of execution of this test for each client. The default value of this variable is "true".

noCheckinterval The default value of the test frequency via the cookie setting is once per day for each client. This can be adjusted by setting **noCheckInterval** to a frequency time. The unit is milliseconds, so that one day (the default value) is 86400000. The default value of this variable is 86400000.

dov6dns If **dov6dns** is set to "true" the script will test if the client is able to cause DNS names to be resolved using DNS queries over IPv6 transport. If this is set to "false", this test will not be performed. The default value of this variable is "true".

dov6literal If **dov6literal** is set to "true" the script will test if the client is able to retrieve an object where the domain name part of the URL is an IPv6 address literal. This test is used to expose latent Teredo capabilities on Windows hosts, where the host will not normally query the DNS for AAAA records of the only Ipv6 interface an a teredo auto-tunnel interface. If this is set to "false", this test will not be performed. The default value of this variable is "true".

dov6tunnels If **dov6tunnels** is set to "true" the script will include a further 2 tests to determine the type of auto-tunnel mechanism is used by the client. The event **v6stf** will record the incidence of 6to4 auto-tunneling by clients, and the event **v6ter** will record the incidence of Teredo auto-tunneling by clients. The default value of this variable is "false".

randomize If **randomize** is set to "true" the order in which the client performs the tests will be randomized. If this variable is set to "false" then the order of the tests

will be V6-only, Dual-Stack, V4-only, V6 DNS (if enabled), V6 Literal (if enabled), V6-only with 6to4 (if enabled), V6-only with Teredo (if enabled).

sampling — If **sampling** is set to a value, then the test will be performed at random at a rate of 1 in **sampling** times. The default value of this variable is 1 (i.e. perform the tests every time the script in invoked, depending on the cookie state)

userid — The **userId** is a value provided by the enrolment page at labs.apnic.net. This allows like web pages to be grouped together. The variable is optional, and the default value is 0 (anonymous).

Callback — The **callback** variable is a function name to be called when the tests have been completed. This can be used to report back to the user on the results of the tests. The default value of this variable is no callback. When using callback, note that This script has a timeout of 10 seconds. This means that the script will wait for 10 seconds before reporting the test results and invoking the callback function.

An example of the use of these settings as a user visible report on their IPv6 capability can be found in the source of the web page http://labs.apnic.net, and in the example code attached to this column.

Acknowledgements

The JavaScript used in these tests was developed by Emil Aben of the RIPE NCC , Byron Ellacott and George Michaelson of APNIC.

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

Example use of the IPv6 test script in a web page.

This is the source code for http://labs.apnic.net, which performs the Ipv6 client test and displays the results back to the user.

```
<html>
 <head>
  <title>Example IPv6 test code</title>
  </head>
<body>
<h2>IPv6 Test Code</h2>


<p>Your IPv6 test results: </p>

<!--This is a rudimentary progress bar for the test interval-->
<blockquote><p>
<table align="left"><tr><td>
<div id="showbar" style="font-size:8pt;padding:2px;border:solid black
1px;visibility:hidden">
<span id="progress1">   </span>
<span id="progress2">   </span>
<span id="progress3">   </span>
<span id="progress4">   </span>
<span id="progress5">   </span>
<span id="progress6">   </span>
<span id="progress7">   </span>
<span id="progress8">   </span>
<span id="progress9">   </span>
<span id="progress10">   </span>
</div>
</td></tr></table></p>
</blockquote>
<pre>


</pre>
<!--This is a rudimentary progress bar -->

<p>
    <dl>
     <dt> IPv4: <dd id='ipprototest_ok_r4td'><i>running</i></dd>
     <dt> IPv4 speed: <dd id='ipprototest_speed_r4td'><i>running</i></dd>
     <dt> IPv6: <dd id='ipprototest_ok_r6td'><i>running</i></dd>
     <dt> IPv6 speed: <dd id='ipprototest_speed_r6td'><i>running</i></dd>
     <dt> Dual: <dd id='ipprototest_ok_rdtd'><i>running</i></dd>
     <dt> Dual speed: <dd id='ipprototest_speed_rdtd'><i>running</i></dd>
    </dl>
</p>
<pre>


</pre>
<p><font color="#E1E1E1">Last Updated: 2011-04-08</font></p>

<!--javascript section-->
<script type="text/javascript">
  var progressEnd = 10;        // set to number of progress <span>'s
  var progressColor = 'blue';  // set to progress bar color
  var progressInterval = 1000; // set to time between updates (ms)
  var progressAt = progressEnd;
  var progressTimer;

function progress_clear() {
  for (var j = 1; j <= progressEnd; j++)
    document.getElementById('progress'+j).style.backgroundColor = 'transparent';
  progressAt = 0;
  }

function progress_update() {
  document.getElementById('showbar').style.visibility = 'visible';
```

```
  progressAt++;
  if (progressAt > progressEnd) progress_clear();
  else
    document.getElementById('progress'+progressAt).style.backgroundColor = progressColor;
  progressTimer = setTimeout('progress_update()',progressInterval);
  }

function progress_stop() {
  clearTimeout(progressTimer);
  progress_clear();
  document.getElementById('showbar').style.visibility = 'hidden';
  }


// for all possible tests, update an _ok and a _speed entry in HTML
function printresults(r) {
  // complete test set.
  var tests = ['r4td','r6td','rdtd'];
  progress_stop();

  for(var i=0;i<tests.length;i++) {
    var x = tests[i];
    var innerDiv;
    if ((innerDiv = document.getElementById('ipprototest_ok_' + x))) {
      if ( r[x] === false ) {
        innerDiv.innerHTML = 'failed';
        }
      else if ( r[x] ) {
        innerDiv.innerHTML = 'OK';
        }
      else {
        innerDiv.innerHTML = 'not tested';
        }
      }
    if ((innerDiv = document.getElementById('ipprototest_speed_' + x))) {
      if ( r[x] === false ) {
        innerDiv.innerHTML = 'failed';
        }
      else if ( r[x] ) {
        innerDiv.innerHTML = r[x] + 'ms';
        }
      else {
        innerDiv.innerHTML = 'n/a';
        }
      }
    }
  }

// Google Analytics Code
var _gaq = _gaq || [];
gaq.push(['_setAccount', 'UA-00000-0']);
gaq.push(['_trackPageview']);

(function() {
  var ga = document.createElement('script');
  ga.type = 'text/javascript';
  ga.async = true;
  ga.src = ('https:' == document.location.protocol ? 'https://ssl' : 'http://www') +
          '.google-analytics.com/ga.js';
  var s = document.getElementsByTagName('script')[0]; s.parentNode.insertBefore(ga, s);
  })();

// enable the APNIC IPProtoTest and feed google analytics events
if ('http:' == document.location.protocol) {
  var ipproto_user = '0';

  // setting the options
  var ipproto_opts = {
    'docookies'        : false,
    'callback'         : printresults
    };

  (function() {
```

```
    var iga = document.createElement('script');
    iga.type = 'text/javascript';
    iga.async = true;
    iga.src = 'http://www.labs.apnic.net/ipprototest.js';
    var is = document.getElementsByTagName('script')[0];
    is.parentNode.insertBefore(iga, is);
    })();
  }

progress_update();// start progress bar

</script>
</body>
</html>
```