Stacking it Up

Experimental Observations on the operation of Dual Stack Services

Geoff Huston, APNIC Labs
If working with one protocol has its problems ...
Then just how much damage can we do by joining two protocols?
Dual Stack End-to-End Service Measurements

Examine IPv6 / IPv4 use from the perspective of a service delivery platform (web server)

- IPv6 is used by clients only when all the various IPv6 infrastructure components support IPv6, otherwise the client will fall back to IPv4 use
- Service metrics for IPv6 are reflective of end-to-end IPv6 capability
Methodology

Test every web client with 3 different retrieval tasks of a 1x1 pixel image:

- V6 only
- Dual-Stack
- V4 Only

Take just one test result for each unique source address per 24 hours

Use server packet dump and web logs as the basis of the analysis

Look at retrieval rates, failure behaviour and transaction times
# Access Combinations

<table>
<thead>
<tr>
<th>Test</th>
<th>Host Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="%E2%9C%94" alt="" /> <img src="%C3%97" alt="" /></td>
<td>V4-Only</td>
</tr>
<tr>
<td><img src="%C3%97" alt="" /> <img src="%E2%9C%94" alt="" /></td>
<td>V6-Only</td>
</tr>
<tr>
<td>✔ ✔ ✔ ✔</td>
<td>V6-Preferred</td>
</tr>
<tr>
<td>✔ ✔ ✔ ✔</td>
<td>V6-Capable (V4-Preferred)</td>
</tr>
<tr>
<td>✔ × × ✔</td>
<td>Dual-Stack Loss</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>V4</th>
<th>V6</th>
<th>Dual</th>
<th>Node Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔</td>
<td>×</td>
<td>✔</td>
<td>V4-Only</td>
</tr>
<tr>
<td>×</td>
<td>✔</td>
<td>✔</td>
<td>V6-Only</td>
</tr>
<tr>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>V6-Preferred</td>
</tr>
<tr>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>V6-Capable (V4-Preferred)</td>
</tr>
<tr>
<td>✔</td>
<td>×</td>
<td>×</td>
<td>Dual-Stack Loss</td>
</tr>
</tbody>
</table>
IPv6: “could” vs “will”

IPv6 Capable

IPv6 Preferred
IPv6: “will” as seen by Google
IPv6: “could” vs “will”
Where are we with IPv6?

The ‘size’ of the IPv6 deployment in terms of end-to-end host IPv6 preference is around 0.3% of the total number of Internet end hosts at present

However, a further 4% of hosts can use IPv6, even though they prefer IPv4 in dual stack mode. These hosts generally use auto-tunnelled 6to4 for IPv6 access
Why is there so much “hidden” IPv6 capability?

Why is the number of client hosts who are capable of performing an end-to-end IPv6 object retrieval 15 times greater than the number of client hosts who prefer to use IPv6 in a dual stack context?
Native vs Tunnels

- Most hosts with unicast IPv6 generally prefer V6 in a dual stack scenario
- Hosts with 6to4 auto-tunnel capability appear to generally prefer V4 in a dual stack scenario
Dual-Stack V6 Preferred by Address Type

Clients who Prefer V6 in Dual Stack by V6 Address Type

- Total
- Unicast
- 6to4
- Teredo

- V6 Unicast
- 6to4
- Teredo
Dual-Stack V4 Preferred by Address Type

Graph showing Dual Stack Clients who Prefer V4 in Dual Stack by V6 Address Type.

- V6 Unicast
- 6to4
- Teredo

Clients percentage over time from November to May.
Native vs Tunnels

• Older versions of dual stack software in hosts preferred IPv6 over IPv4 in all situations, including auto-tunnels
  – This resulted in very slow and erratic performance when accessing some dual stack servers due to the local IPv6 failure timers
    • For example, Windows XP takes 20 seconds to recover a connection if a 6to4 connection is not functioning correctly

• Recent OS releases have de-pref’ed auto-tunneled IPv6 below that of IPv4
Performance Observations
Performance and Tunnels

Avg V6 Load Time Differential (secs) by V6 Type

- Teredo
- 6to4
- V6 Unicast

+4 Secs
+2 Secs
0 Sec
-2 Secs
-4 Secs

Nov Dec Jan Feb Mar Apr May
Performance and Tunnels

• Unicast IPv6 performance is on average equivalent to IPv4 performance for web object retrieval

• Auto-tunnel performance is on average considerably worse
  – Teredo is highly variable with 1 – 3 seconds of additional delay per retrieval
  – 6to4 is more consistent with an average 1.2 seconds additional delay per retrieval
Performance and Tunnels

Two causes of incremental delay:

– Tunnel setup time
  • Stateful Teredo tunnels require initial packet exchanges to set the tunnel up (min 1 x RTT)

– Tunnelling can extend the RTT delay
  • addition of tunnel relays between the source and destination
  • This is exacerbated when the forward and reverse paths are asymmetric
6to4 Packet Path

Client

192.88.99.1 Relay

V4-Only Network

Dual-Stack Network

Dual-Stack Server

2002::/16 Relay
Partial Mitigation of 6to4 Packet Path

Client

V4-Only Network

192.88.99.1 Relay

Dual-Stack Network

Dual-Stack Server

2002::/16 Relay
6to4 Performance

Setup Time

6to4 Setup Delay

% 6to4 clients vs Seconds
6to4 Performance

Tunnel RTT Cost

6to4 RTT Incremental Delay

% 6to4 clients

msecs
6to4 Relative Performance

6to4 adds an average of 1.2 seconds to the object retrieval time

– note this is one-way (as the server has a local 6to4 relay for the response traffic, so the 6to4 response path is the same as the V4 path)
– that’s a very long transit time if this is just added transit time
– There may be a congestion load delay added in here
– But the level of 6to4 traffic is very low, so congestion overload is unlikely
Teredo vs 6to4

What we see:

– 4% of hosts use 6to4 (native V4, auto-tunnel)
– 0.1% of hosts use Teredo (NAT V4, auto-tunnel)
Exposing Teredo

But why so little Teredo?

– Windows Vista and Windows 7 `gethostbyname()` will not query for a AAAA record if the only local IPv6 interface is Teredo
– Can we expose latent Teredo capability?
Exposing Teredo

Use an IPv6 literal as the object URL:
http://[2401:2000:6660::f003]/1x1.png
Exposing Teredo

Client IPv6 Capability by URL Type (***)

- CAN: IPv6 with IPv6-Only Literal URL
- COULD: IPv6 with IPv6-Only URL
- WOULD: Prefer IPv6 with Dual-Stack URL

Graph showing the percentage of clients over time with different IPv6 capability types.
Exposing Teredo

Use an IPv6 literal as the object URL:
http://[2401:2000:6660::f003]/1x1.png

– In the context of the experimental setup it was observed that ~30% of the client base successfully fetched this IPv6 URL using Teredo!

– Conversely, 70% of the clients did not manage a successful object retrieval of this URL
Performance and Tunnels

Teredo adds a further performance penalty in the form of state setup between the Teredo relay and the client.
Teredo Performance

Tunnel Setup Time

Teredo Delay Distribution (Seconds)
Teredo Performance

Tunnel RTT Cost

Teredo RTT Distribution

% of Teredo Clients

Teredo Incremental RTT Delay (msecs)
Teredo Relative Performance

Teredo adds an average of 1 - 3 seconds to the object retrieval time
- Teredo setup takes between 0.6 second to 3 seconds
- Average RTT cost of Teredo is 300ms
- Object retrieval takes ~3 RTT intervals to complete
- Total time cost is some 2 seconds on average
IPv6 Performance

• Unicast IPv6 appears to be as fast as IPv4 for object retrieval

• Auto-tunnelling IPv6 attracts major performance overheads
  – these are strongly context dependent
  – widespread deployment of 6to4 relays and Teredo relays and servers would mitigate this, to some extent
  – Dual Stack servers may want to consider using local 6to4 relays to improve reverse path performance for auto-tunnelling clients
Failure Observations
Dual Stack Failure

How many clients retrieve the V4 only object but DON’T retrieve the Dual Stack objects?

i.e. how many clients exhibit “Dual Stack Failure”? 
Dual Stack Failure Rate

Dual Stack Loss Rate

End hosts

Nov Dec Jan Feb Mar Apr May

0.2%

0.4%

0.6%

0.8%
Dual Stack Failure

This rate of 0.7% of clients is the rate of failure of IPv4 clients to retrieve a dual stack object.
Dual Stack Failure

This rate of 0.7% of clients is the rate of failure of IPv4 clients to retrieve a dual stack object

But this is not a reliable metric of underlying protocol communication failure

– This is the rate of failure of the client to retrieve a dual stack object from within a javascript code object

– The client may:
  • Not execute the javascript at all
  • User reset of the retrieval before completion
  • In addition to the failure to fallback to IPv4 retrieval
Connection Failure

To attempt to look more precisely for some instances of connection failure, let's looking for connections that fail after the initial TCP SYN

Note that this approach does not detect failure of the initial SYN packet, so the results are a lower bound of total connection failure rates.
Connection Failure

Relative Percentage of Failed Connections

% Protocol Connections

IPv6 Connection Failure Rate
IPv4 Connection Failure Rate

Date

18/12 01/01 15/01 29/01 12/02 26/02 12/03 26/03 09/04 23/04 07/05
IPv6 Connection Failure

V6 Failed Connections

% Connections

Date

18/12 01/01 15/01 29/01 12/02 26/02 12/03 26/03 09/04 23/04 07/05
Is Teredo really THAT good?
Teredo Connection Failure

Teredo uses an initial ICMPv6 exchange to assist in the Teredo Server / Relay state setup.

Note that this approach does not detect failure of the initial ICMPv6 echo request, so the results are a lower bound of total connection failure rates.
No.
IPv6 Connection Failure using V6 Literal
IPv6 Connection Failure

• Some **2%-5%** of **IPv6 unicast** connections fail!
  – This rate is better than IPv6 auto-tunnels, but is still 20x the rate of IPv4 connection failure

• Some **12% - 20%** of **6to4** connections fail!
  – This is a very high failure rate!
  – The failure is most likely a protocol 41 filter close to the client that prevents incoming 6to4 packets reaching the client

• Some **40%** of **Teredo** connections fail!
  – This is an amazingly high failure rate!
  – Is STUN just broken? And/or ...?
Can we improve Dual Stack Performance?

We need to understand how client systems behave in a dual stack environment in order to understand how we can improve the situation.
Serialization

Client

DNS Phase

TCP Connection Phase

AAAA Query

A Query

AAAA Response

A Response

V6 SYN

V6 SYN+ACK

V6 ACK
Serialization and Failure

In response to poor performance associated with auto-tunnelling many OS stacks have responded by altering the local protocol preference table to depref 6to4 BELOW V4, and to try and not use Teredo at all!
Can we improve Dual Stack Performance?

Yes!
Parallelization

• In response to an open() call from the application, set off two independent streams (V4 and V6) and perform in parallel:
  – DNS query
  – TCP SYN exchange

• ACK the first TCP SYN+ACK to be received, and present this back to the application as the “working” TCP connection

• RST the other
Parallelization

Client

DNS

A AAAA Query

V4 thread

V6 thread

Web Server

A AAAA Response

V6 SYN

V6 SYN+ACK

V6 ACK ...

Protocol section point

A Query

V4 SYN

V4 RST

V6 SYN+ACK

A Response

V4 thread

V6 thread
Parallelization

Trade offs:

+ Faster client experience
- Higher client state overhead
- Higher server SYN load for dual stack servers

“Happy Eyeballs: Trending Towards Success with Dual-Stack Hosts”

draft-wing-v6ops-happy-eyeballs-ipv6-01
Conclusions

What can we say about the performance and robustness of a Dual Stack network environment as a result of these observations?
For an Online Service...

Converting a service to operate as a Dual Stack service is a viable option in today’s environment

But:

– a small fraction of existing clients will experience a much slower service
– a very small fraction of existing clients will fail to connect to the dual stack service at all
What about IPv6-Only Services?

Is an IPv6-only service a viable option today?

Not really.

– Only ~4% of the existing client base would successfully connect to an IPv6-only service
– And many would experience poor performance relative to IPv4 services
What about Dual Stack Transition?
What about Dual Stack Transition?

End-host auto-tunnelling is not a solution!
What about Dual Stack Transition?

End-host auto-tunnelling is **not** a solution!

- Auto-tunnelling appears to encounter many more performance and reliability problems than it solves in terms of IPv6 connectivity
- Auto-tunnelling is **not** proving to be a useful mainstream transition tool for IPv6
What about Dual Stack Transition?

If we want this transition to operate in a manner where IPv6 operates at least as well as IPv4 then end hosts really need to be connected to a IPv6 Unicast service delivered from their service provider
Thank You

Questions?