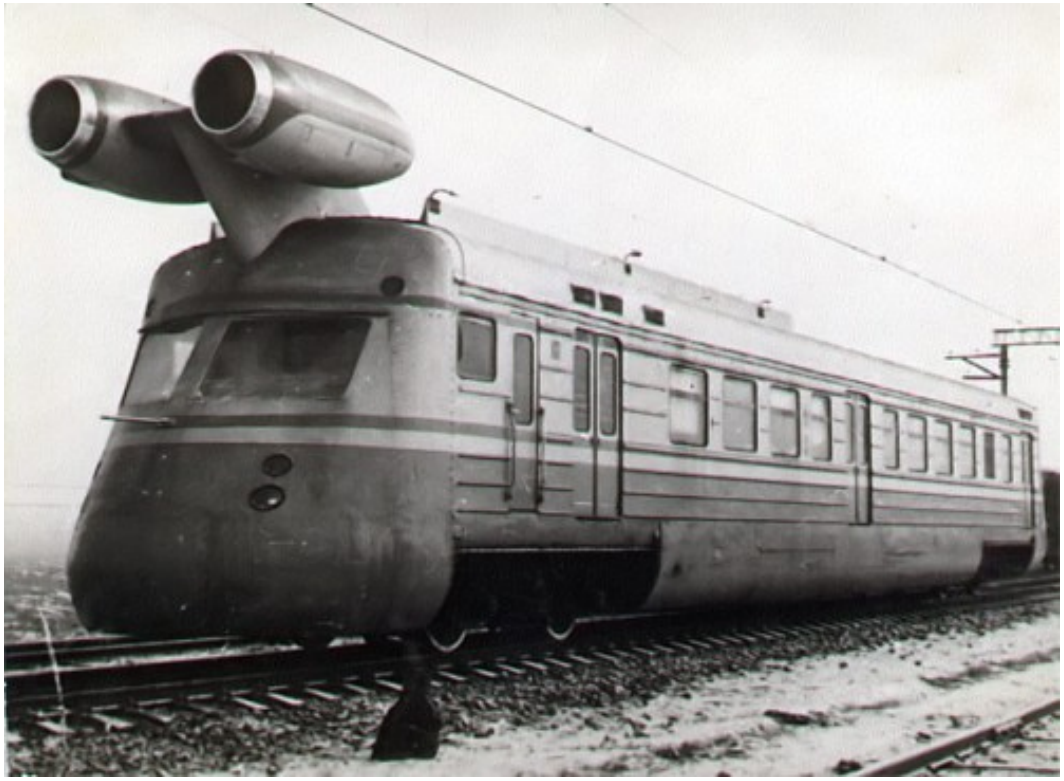
Stacking it Up

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 fun can we have by using two protocols at once?



Some Dual Stack Questions

- How many clients are capable of IPv6 access?
- What forms of IPv6 access are they using?
- Is their experience over Dual Stack better or worse than IPv4?

Setting the scene

- Adding IPv6 to your website may have risks
  - Will your clients still be able to ‘see’ you?
  - What % of clients will experience issues?
- Finding out in advance what to expect is useful
  - A way to measure end-user behavior
  - Without affecting your own website investment
- Measuring failure is hard!
  - Website logs only measure successful connections

Adding IPv6 may have risks

- Older Windows XP hosts experience problems with dual-stack (IPv4, IPv6) DNS records
  - May refuse to connect to the IPv4 address
- Some hosts cannot process IPv6 DNS properly
  - Not supported in all DHCP backed configurations
- 'Partial IPv6' problems
  - Locally IPv6 enabled, no IPv6 route to global Internet
- Loss of eyeballs = Loss of revenue?
  - When your core business presents via the web, what risks to loss of web access are you willing to take?

Finding out in advance what to expect

- Measure client's IPv6 behavior without having to add IPv6 to your website
  - Leverage cross-site URL fetches
- Integrate these measurements into existing tracking methods, and analytics framework
  - No new tools needed

Measuring failure is hard!

- Web logs record completed TCP/IP events
  - Even 4xx and 5xx responses in logs are completed valid TCP/IP sessions
- What about the people who fail to complete the connection?
  - Not in access- or error- logs
- Only partially visible on-the-wire
  - Characteristic missing 'SYN/ACK' sequence in TCP signals failure to complete a 2-way handshake
- But (inside a time limit) client knows what worked or failed: and can report back.

APNIC's web measurement system

http://labs.apnic.net

- Built on google 'analytics' method
  - Javascript, highly portable
  - Asynchronous, runs in the background
    - after page render already complete
  - Uses DNS wildcards, uncacheable
- Data integrated into google analytics reports
  - Graphs of 'events' to monitor IPv4, IPv6 and dual-stack
- Configurable by website manager
  - Sample or every connection, extra tests etc

Measuring by 1x1 invisible pixels

- Javascript requests sequence of 1x1 pixel images
  - Images fetched but not included in the DOM so not displayed
  - Image fetches take place after DOM render,
  - does not add delay to page view, invisible
    (may be seen in browser status bar, error report windows)
  - Javascript callback records success/time
- Image fetches from unique DNS names
  - Every client is a fresh name, no cached state
- Client reports timing, connect failures
  - to your analytics report as a results/summary field
  - Can account for 'unable to connect' TCP/IP failure

What is tested?

- Basic test set is dual-stack, IPv4, IPv6
  - Dual stack enabled DNS behind all fetches
- Additional (optional) tests
  - IPv6 literal

    (bypasses many Windows Teredo IPv6 suppression settings)
  - IPv6 DNS

    (can be visible to user, stress-tests DNS)
  - Auto-Tunnel detection

    URLs only reachable from Teredo and 6to4 source IP addresses
- Results reported over IPv4-only URL

Additional Measurements

We extended this technique into Flash, and created an anonymous banner ad

The IPv6 capability test is built into the Flash code

Banner Ad Fun

No clicks needed

(indeed we would prefer that clients did NOT click the ad, as it costs us more for a click!)

Impressions are really cheap

$25 per day buys around 25,000 impressions

Every impression carries the complete IPv6 test set

But many users are ad-intolerant

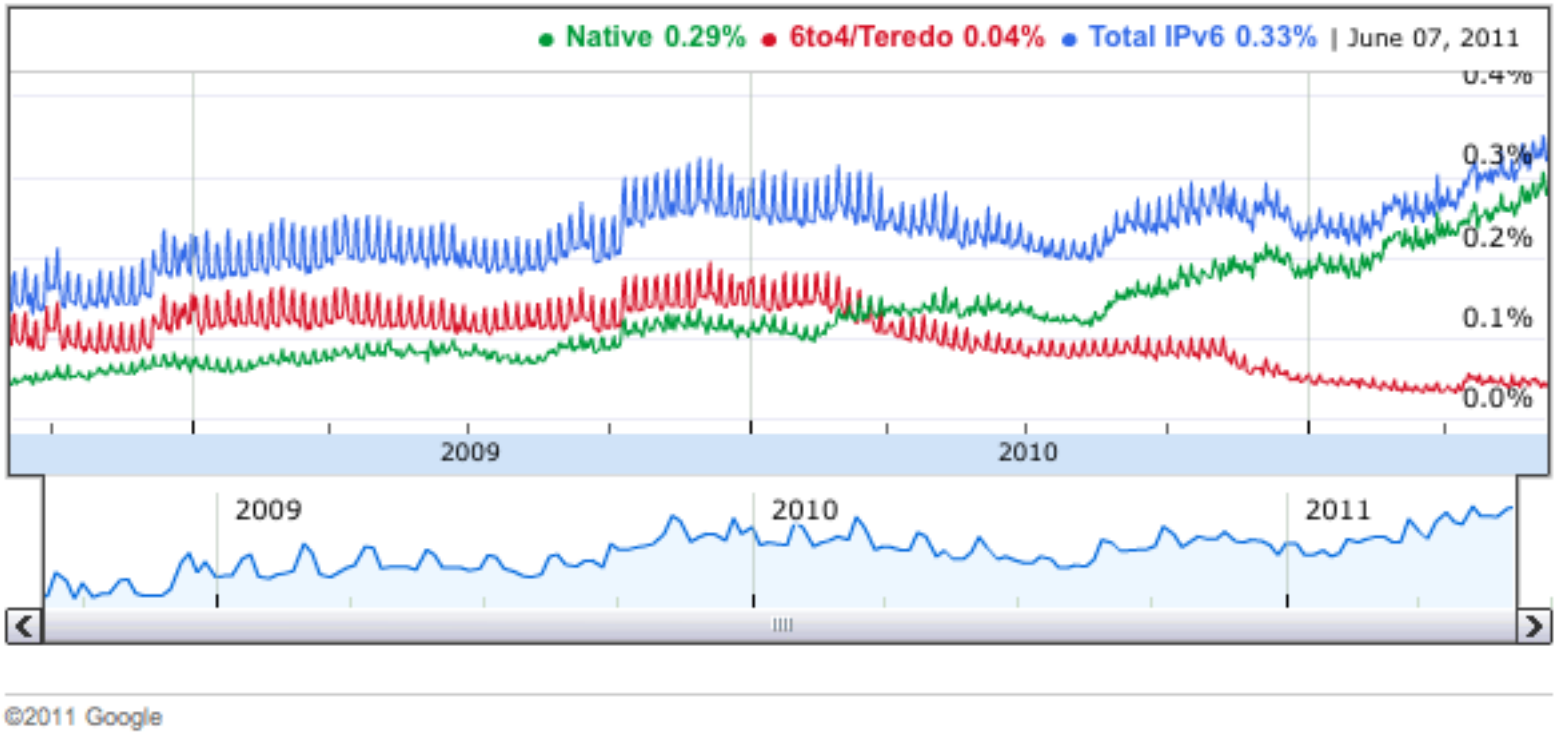
Users tend to browse away from pages containing the ad in a far shorter time interval

We see a higher number of aborted test runs with the ad

Some Results

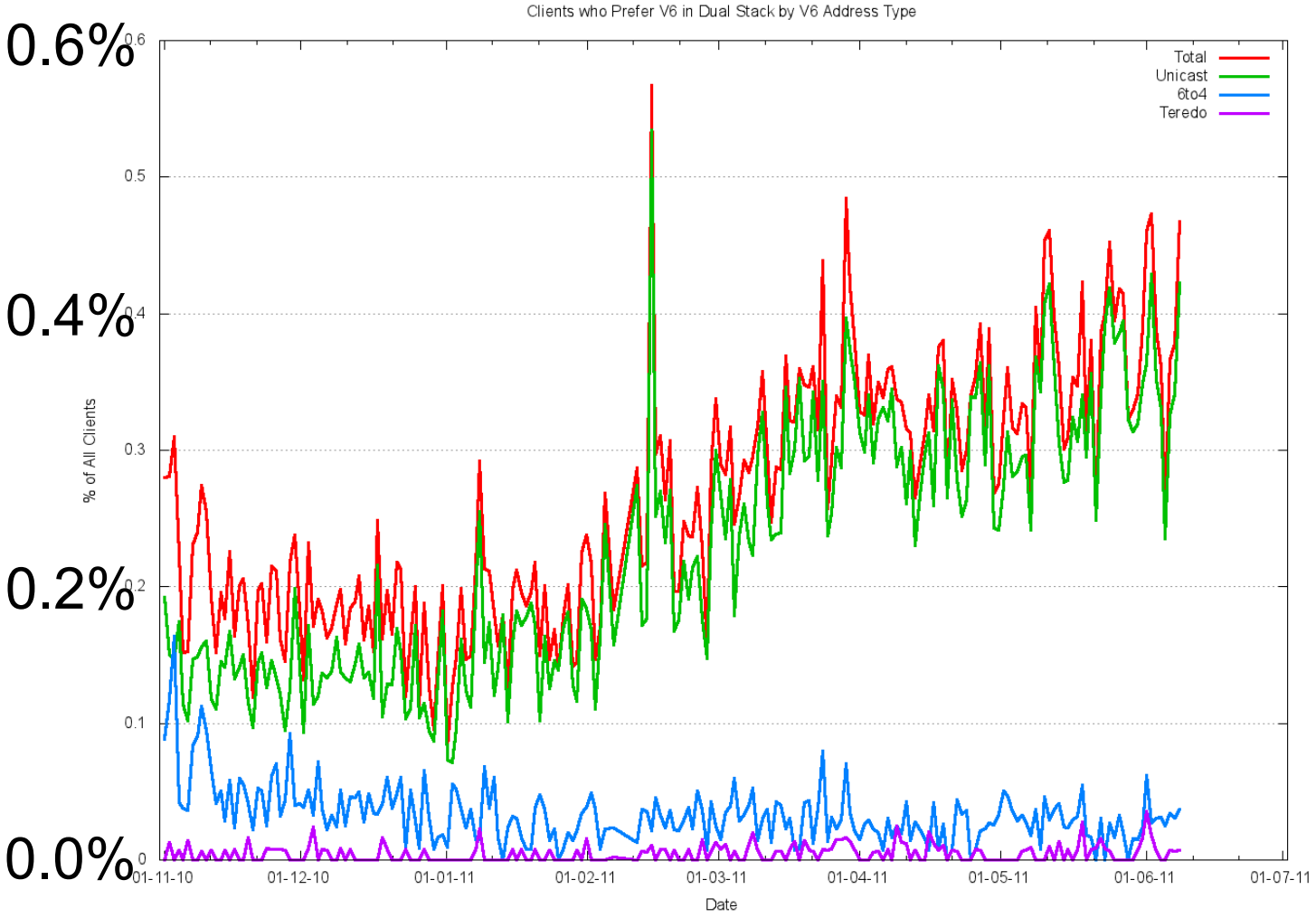
- How much IPv6 is out there in terms of end host capability?
- What forms of IPv6 access are clients using?

IPv6 capability, as seen by Google



http://www.google.com/intl/en/ipv6/statistics/

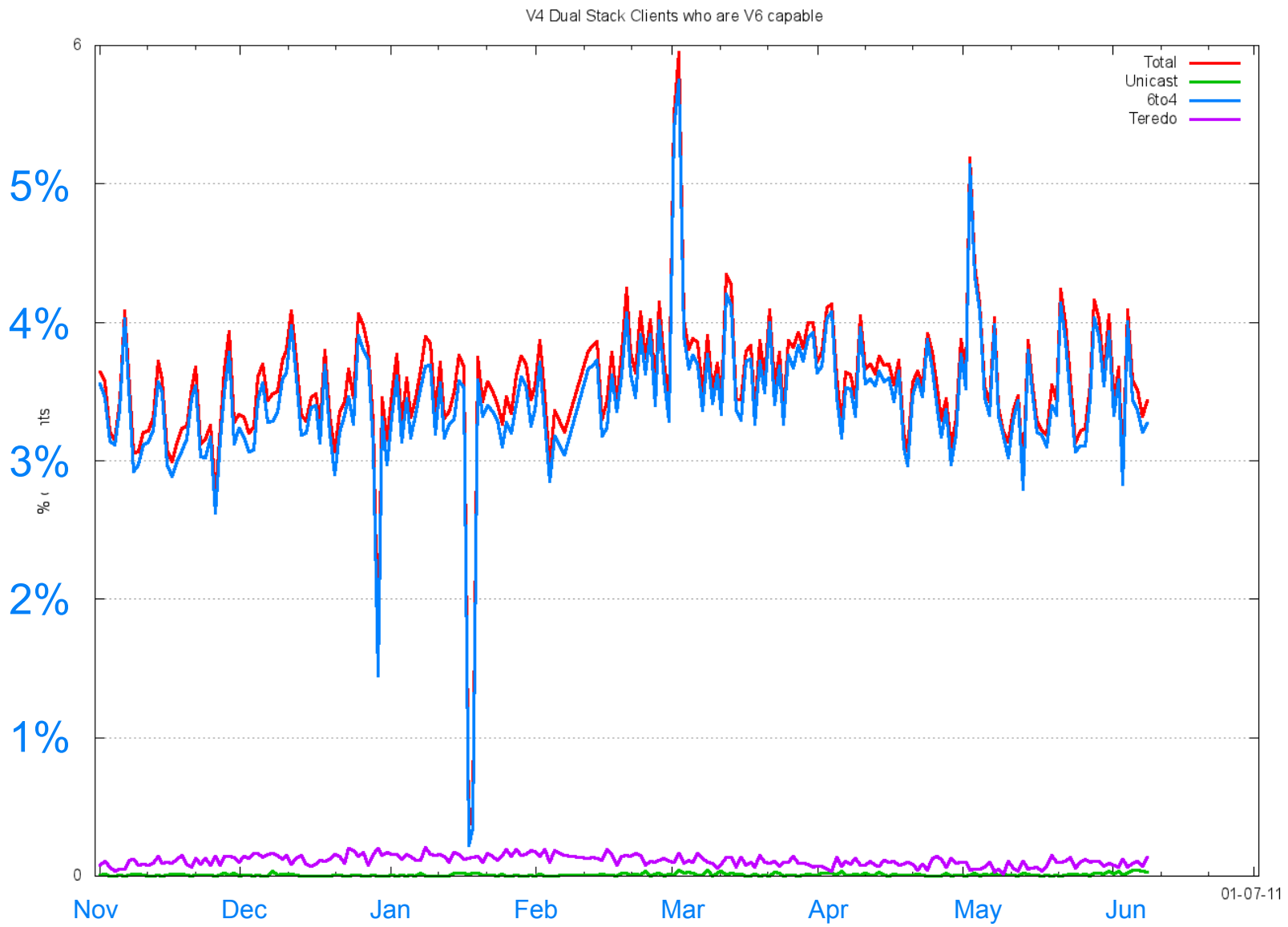
IPv6 capability, as seen by APNIC



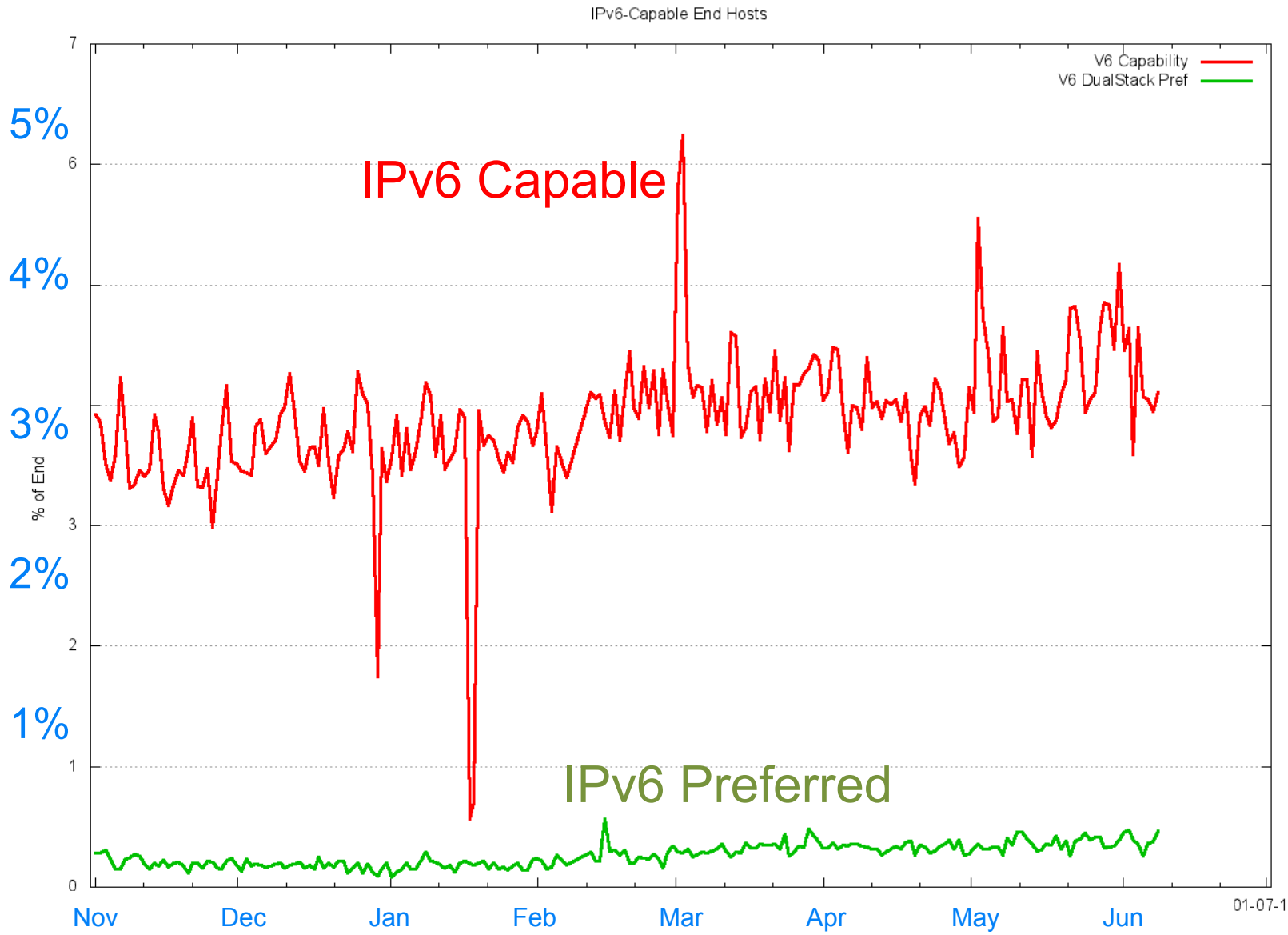
Is This All There Is?

- 0.3% – 0.4% of clients is a very low number
- And most of the IPv6 access we see here is using unicast IPv6
- Where are all the 6to4 and Teredo auto-tunnels?
- Lets look harder by testing with an IPv6-only image

IPv6 ONLY, as seen by APNIC



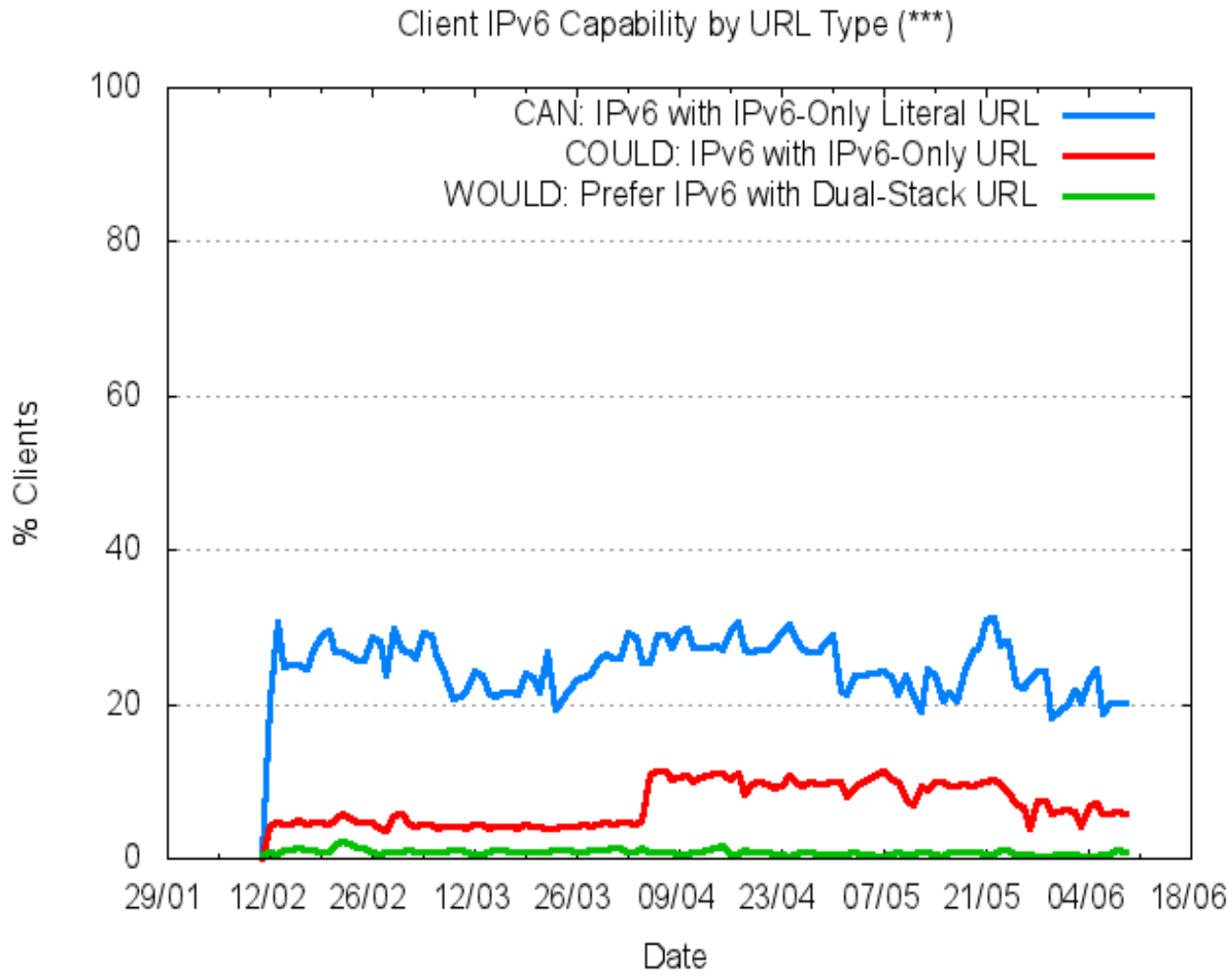
IPv6: “could” vs “will”



Is This All There Is?

- 3% - 4% of clients is still a very low number
- Most of the access in IPv6-only is via 6to4 auto-tunnelling
- Where is Teredo?
- Lets look harder by testing with an image that does not require a DNS lookup: http://[2401:2000:6660::f003]/1x1.png

IPv6: “can” vs “could” vs “will”

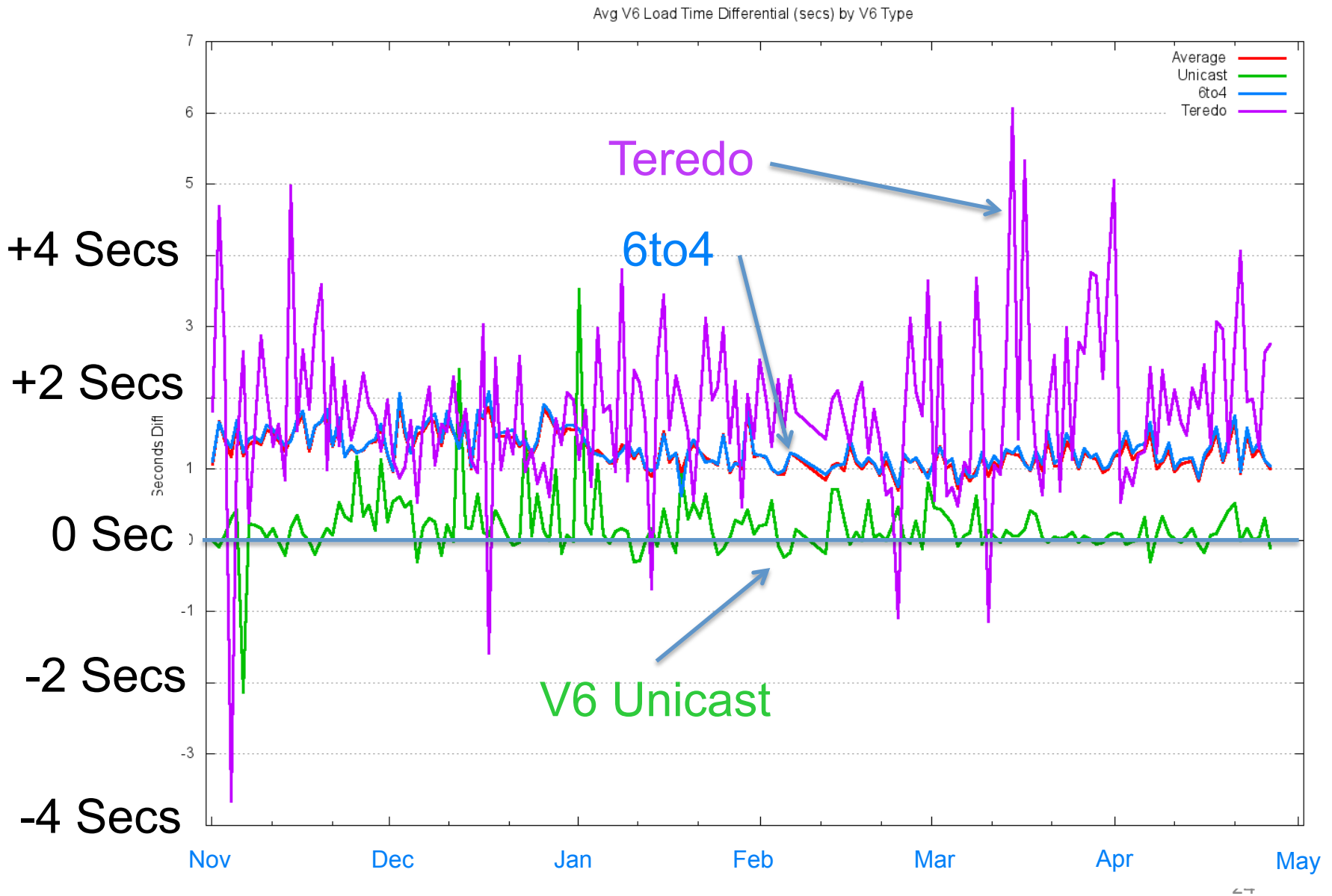


How Much IPv6 is Out There?

- Around **0.4%** of the Internet's clients can and will use IPv6 in a Dual Stack scenario
  - And these clients are generally using a "native" IPv6 service
- Around **4%** of the Internet's clients can use IPv6 in an IPv6-only scenario
  - And the additional clients are generally using 6to4 auto-tunnelling
- Around **35%** of the Internet's clients are equipped with IPv6 capability that can be exposed
  - And the additional clients are using Teredo auto-tunnelling

Performance Observations

Performance and Tunnels



Performance and Tunnels

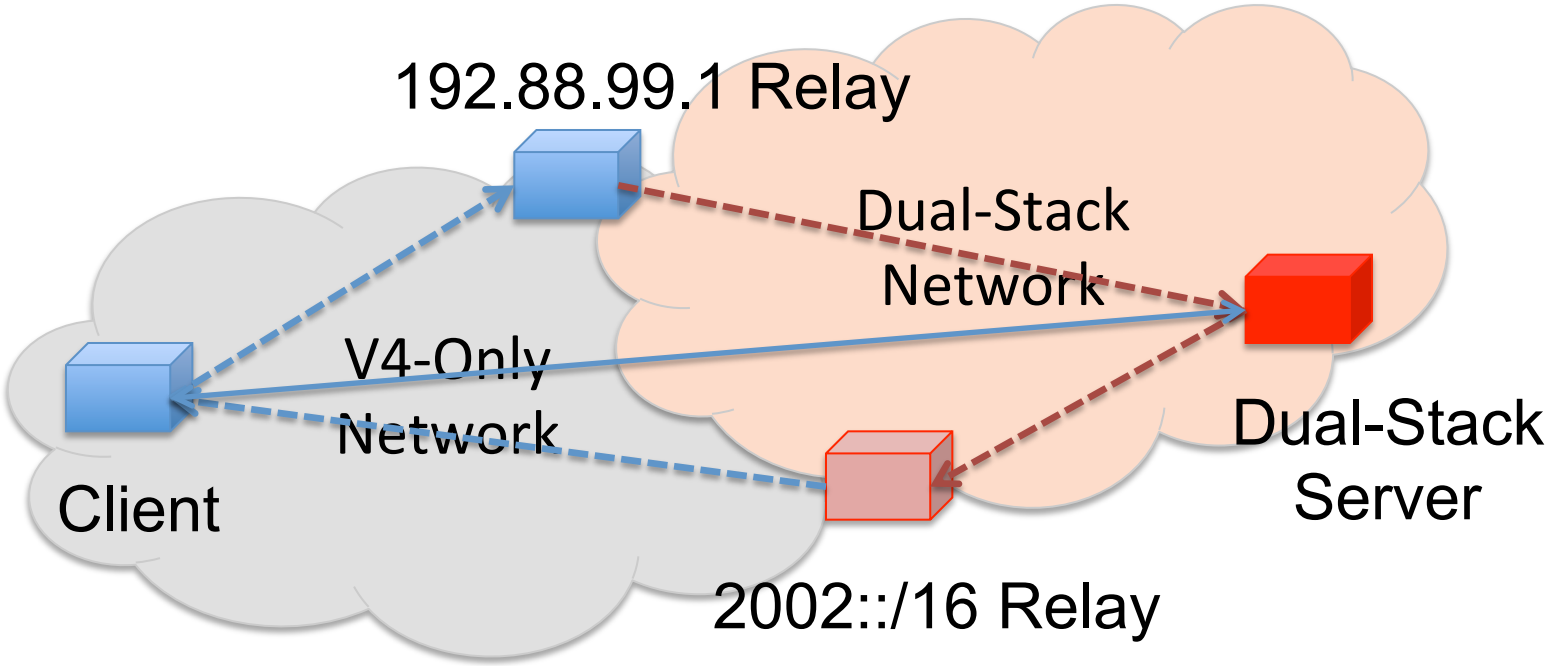
- Unicast IPv6 performance is on average equivalent to IPv4 performance for web object retrieval
- Auto-tunnel performance is on average considerably worse
  - Teredo is highly variable with **1 – 3 seconds** of additional delay per retrieval
  - 6to4 is more consistent with an average **1.2 seconds** additional delay per retrieval

Performance and Tunnels

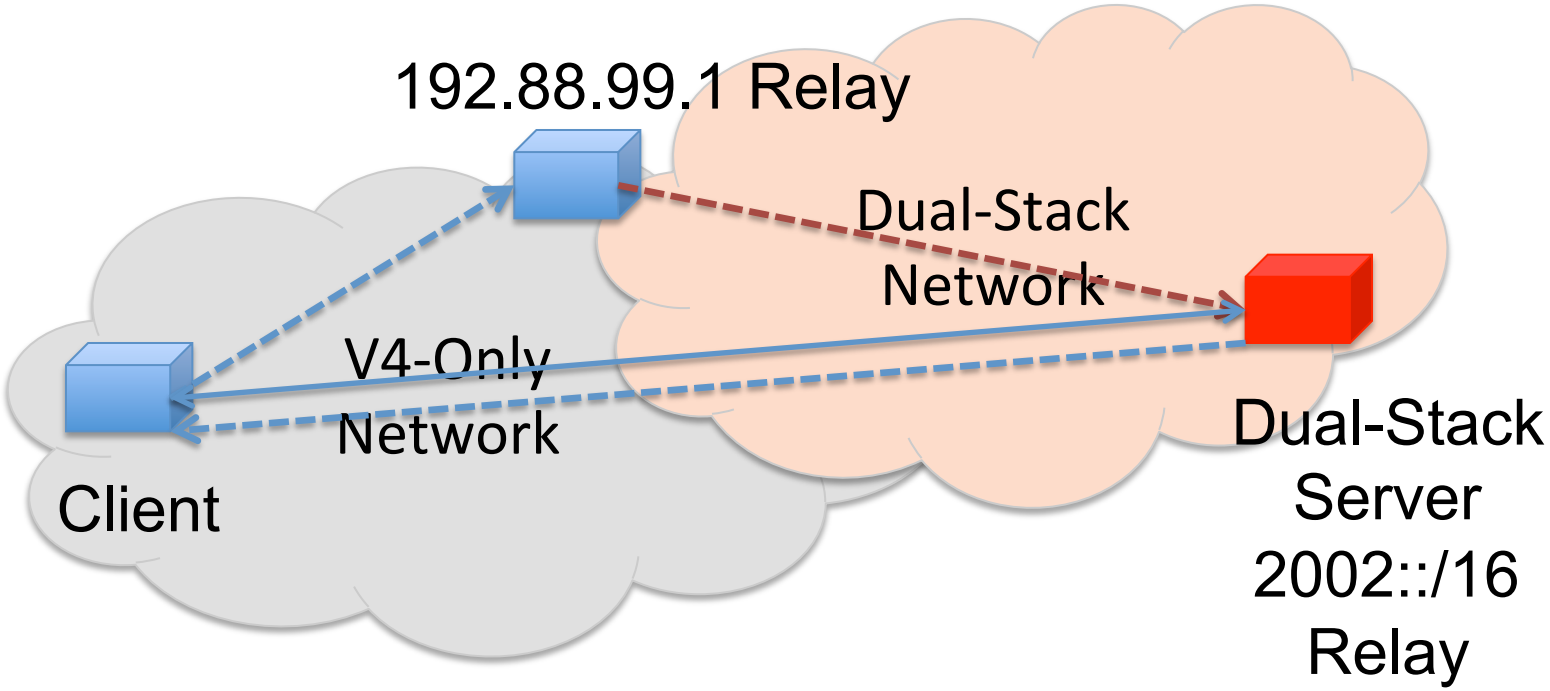
Two causes of incremental delay:

- 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 asymmteric

6to4 Packet Path

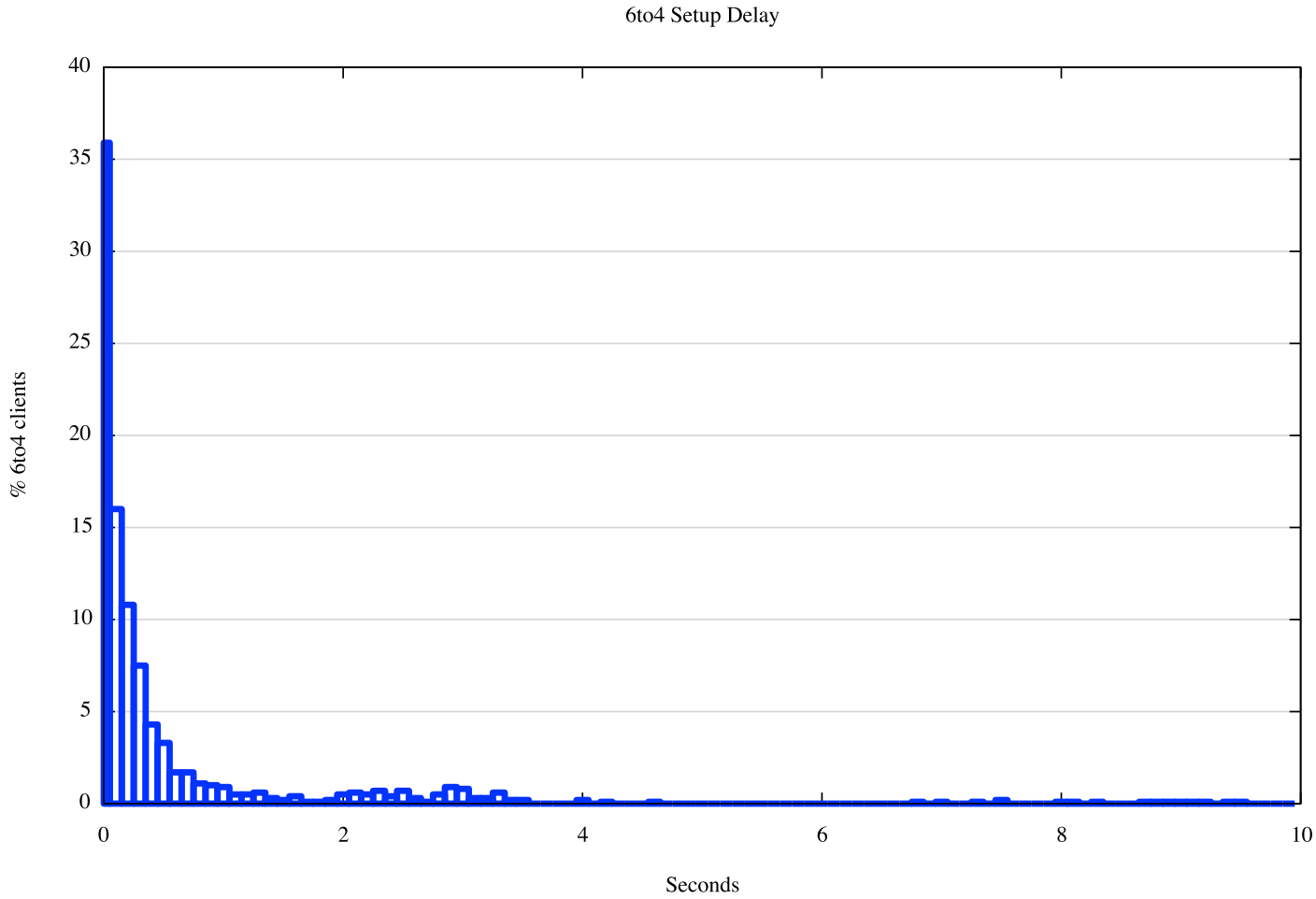


Partial Mitigation of 6to4 Packet Path



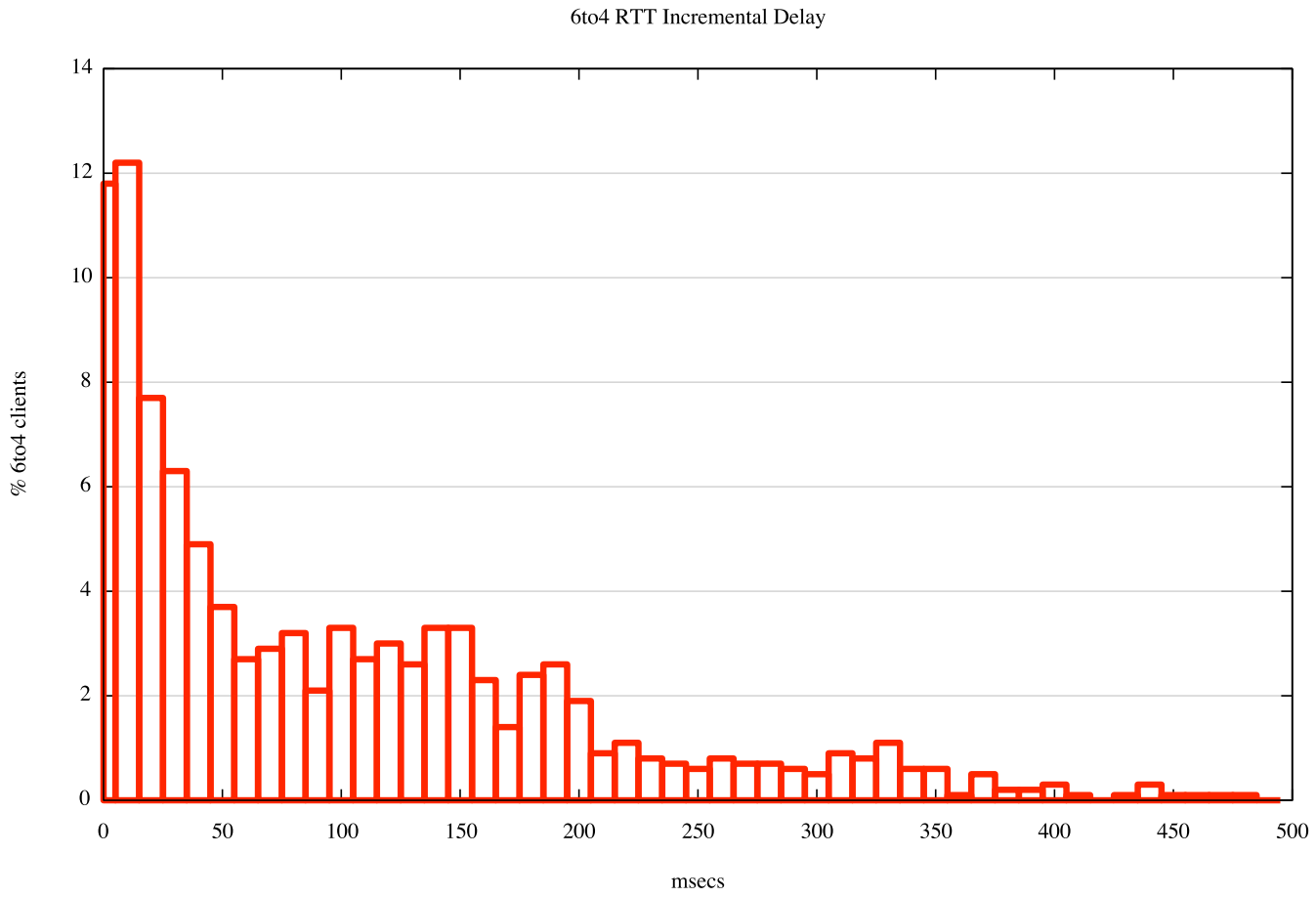
6to4 Performance

Setup Time



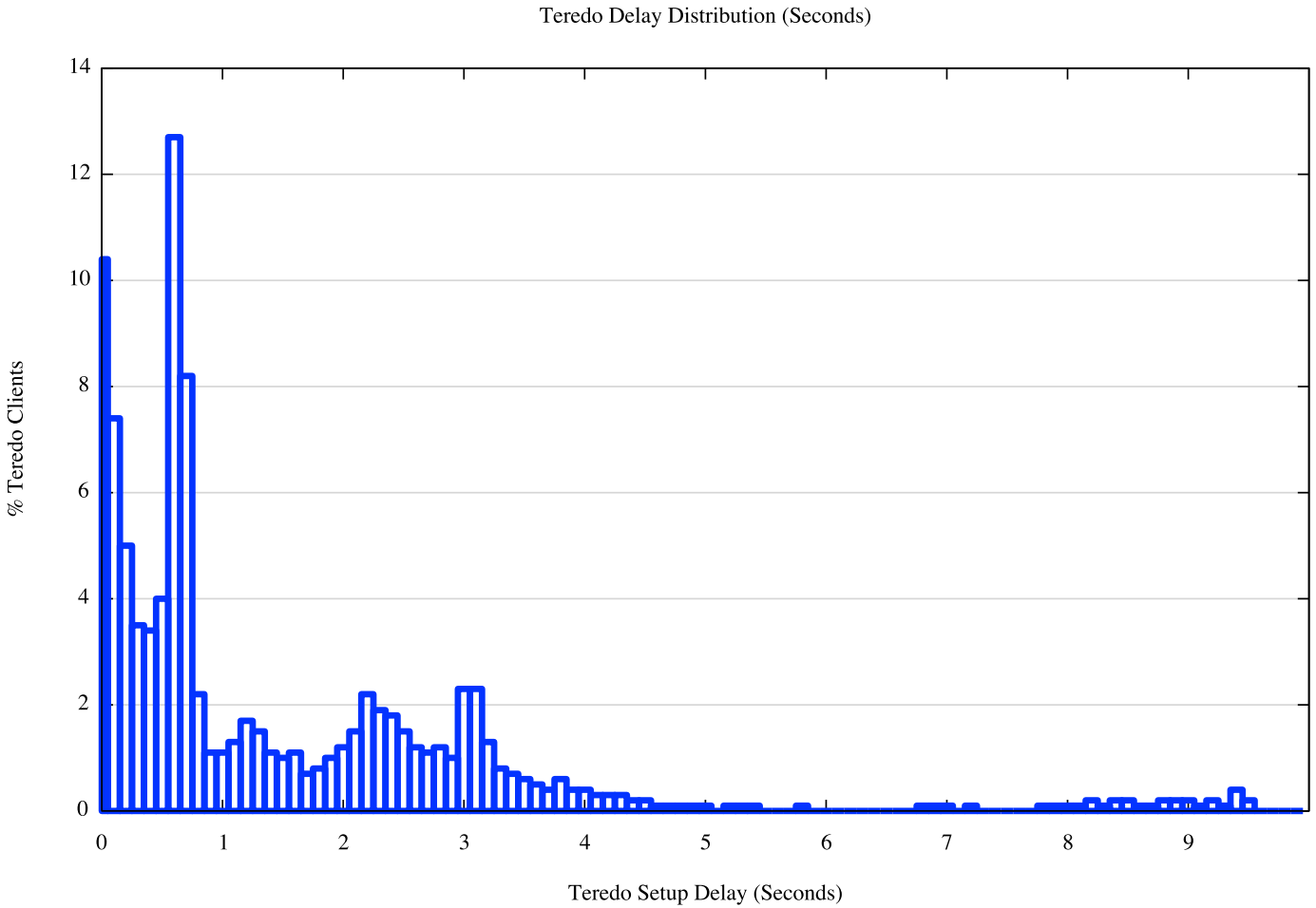
6to4 Performance

Tunnel RTT Cost



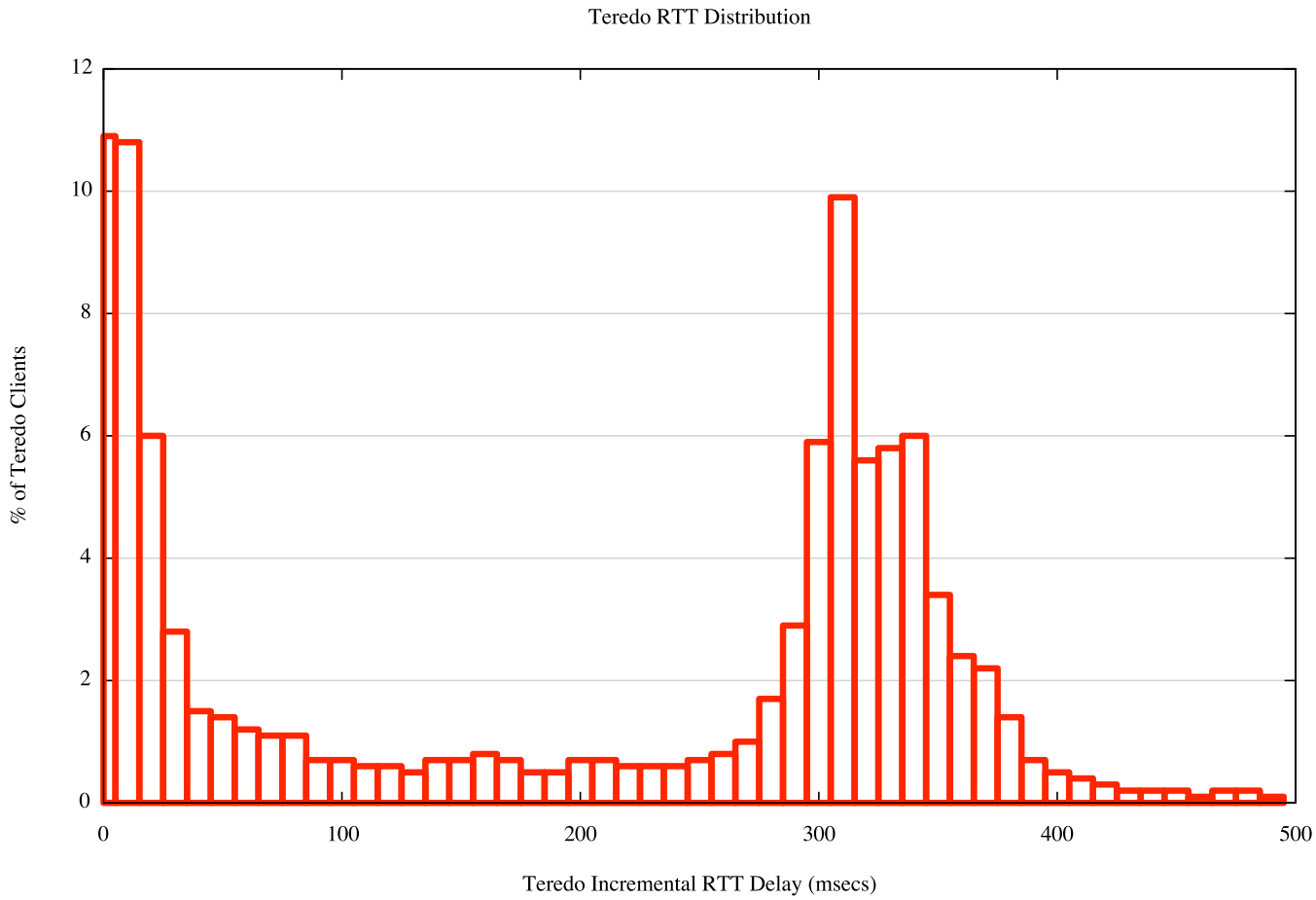
Teredo Performance

Tunnel Setup Time



Teredo Performance

Tunnel RTT Cost



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

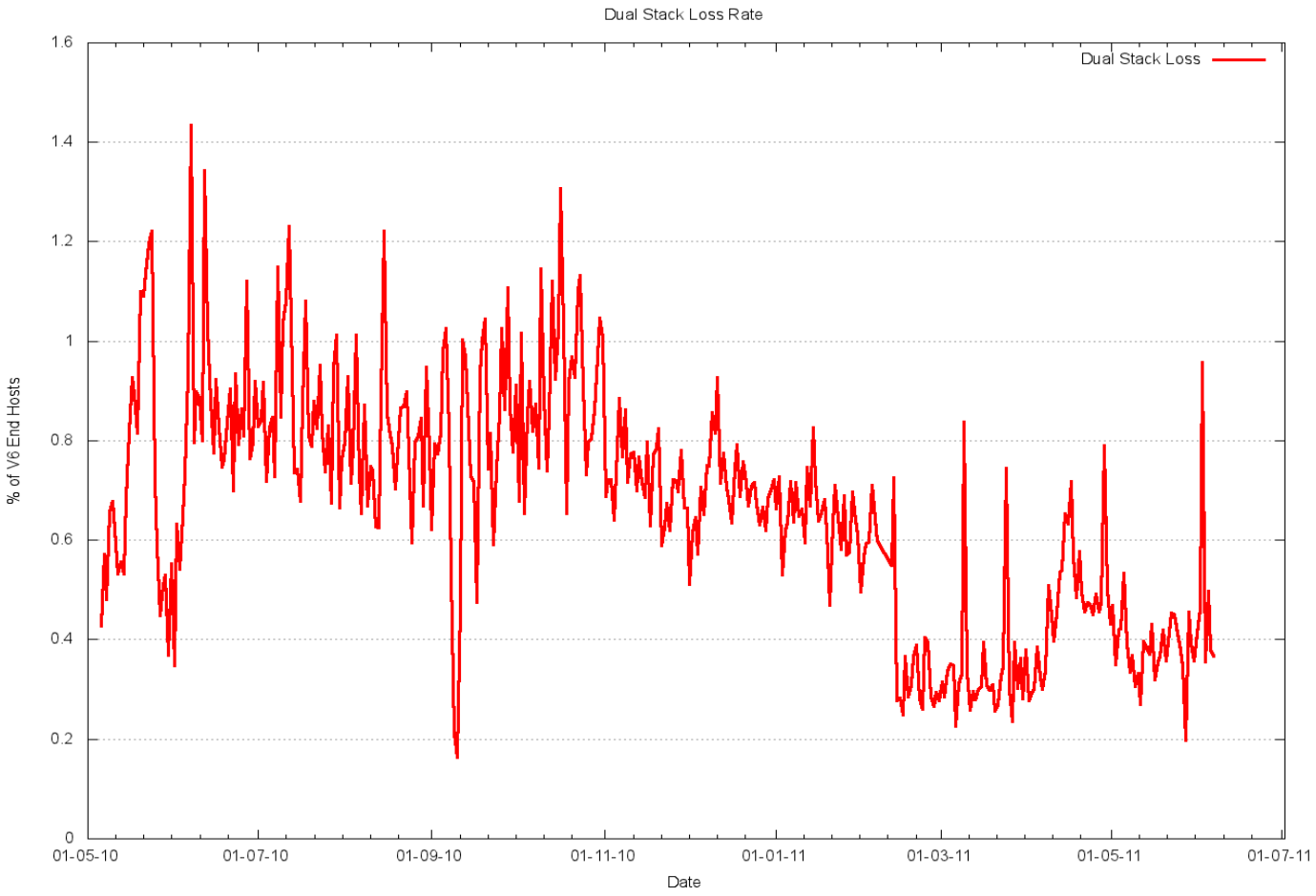
Failure Observations

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 Loss Rate

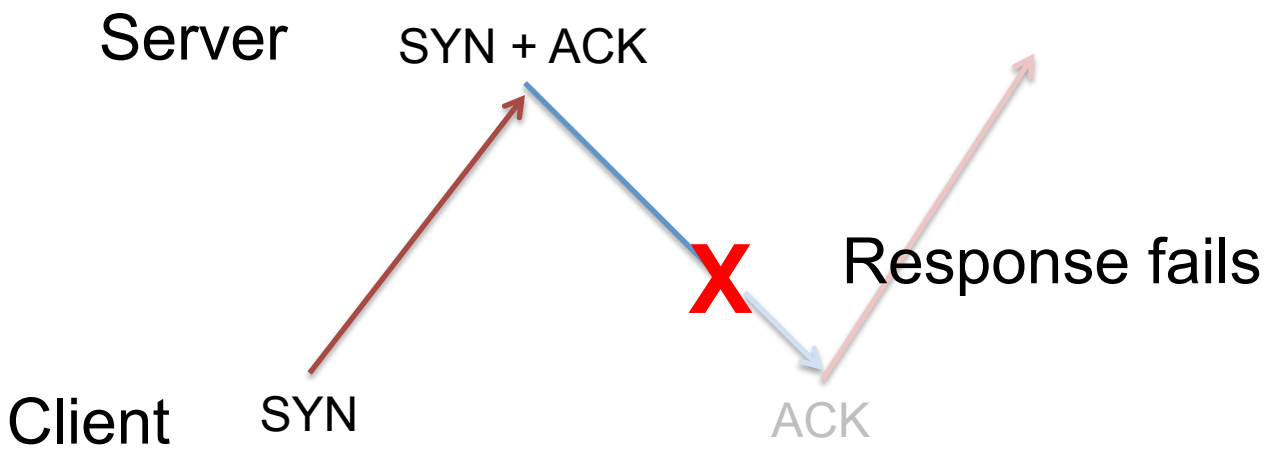


Dual Stack Loss

- 4 in 1000 clients are unable to fetch a web URL if presented with a dual-stack DNS name
- Older (windows XP) hosts, browsers

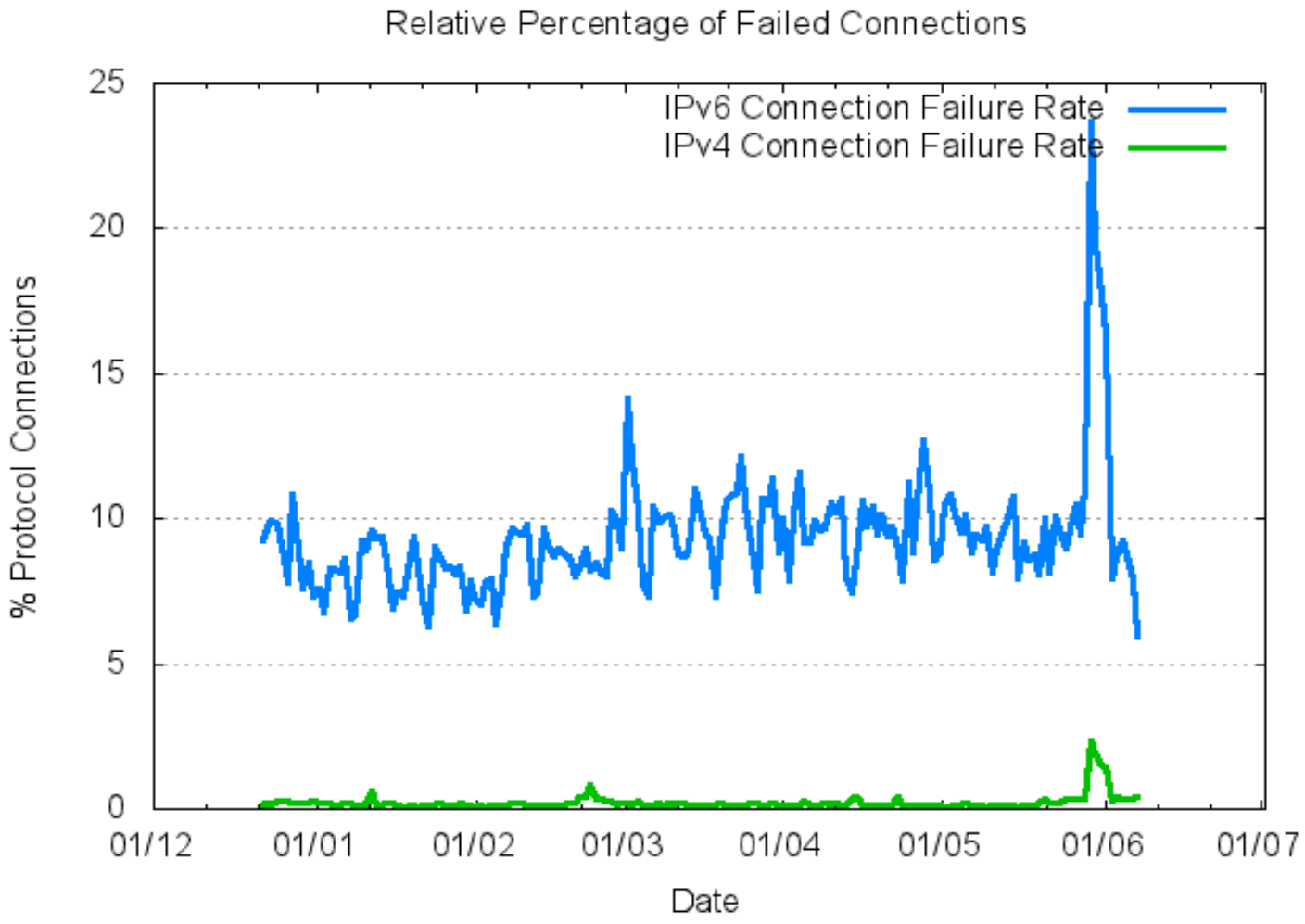
Connection Failure

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

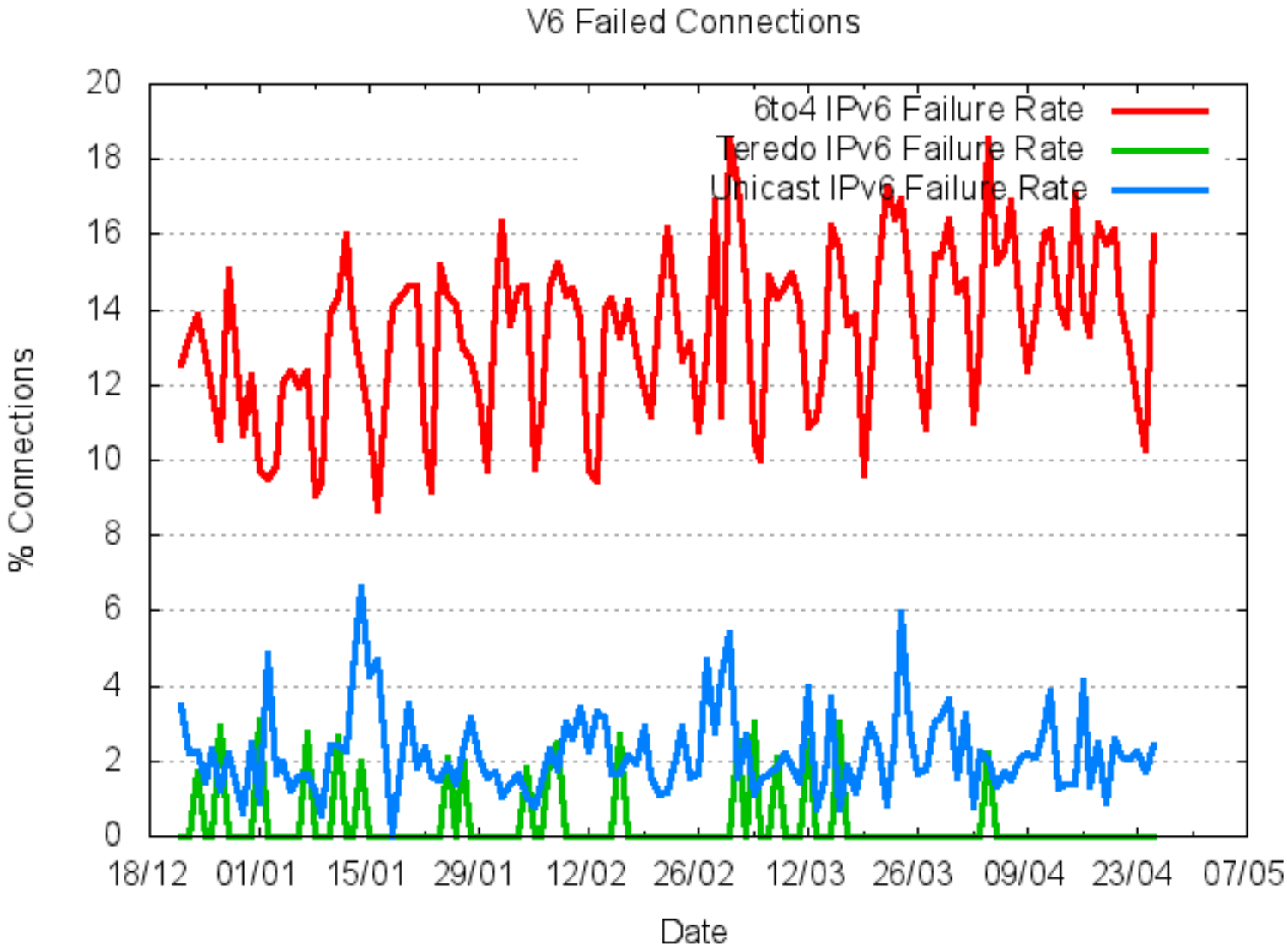


Note that this approach does not detect failure of the initial SYN packet, so the results are a lower bound of total connection failure rates

Connection Failure



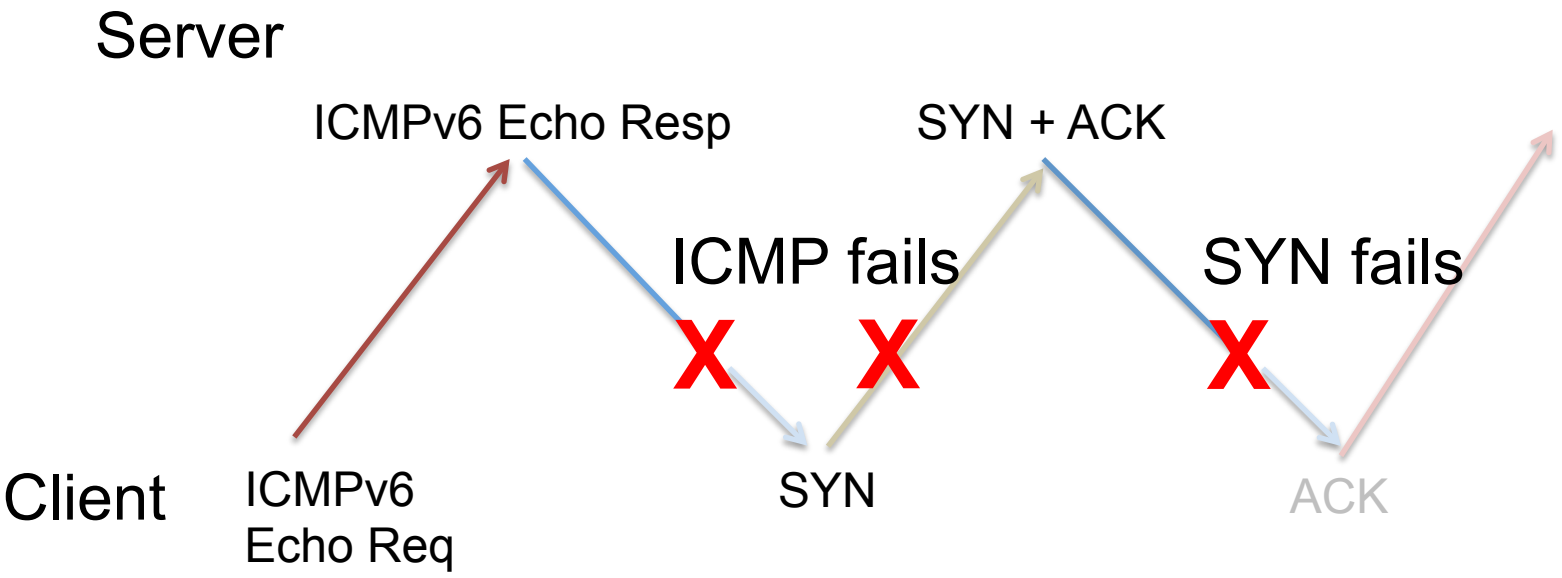
IPv6 Connection Failure



Is Teredo really THAT good?

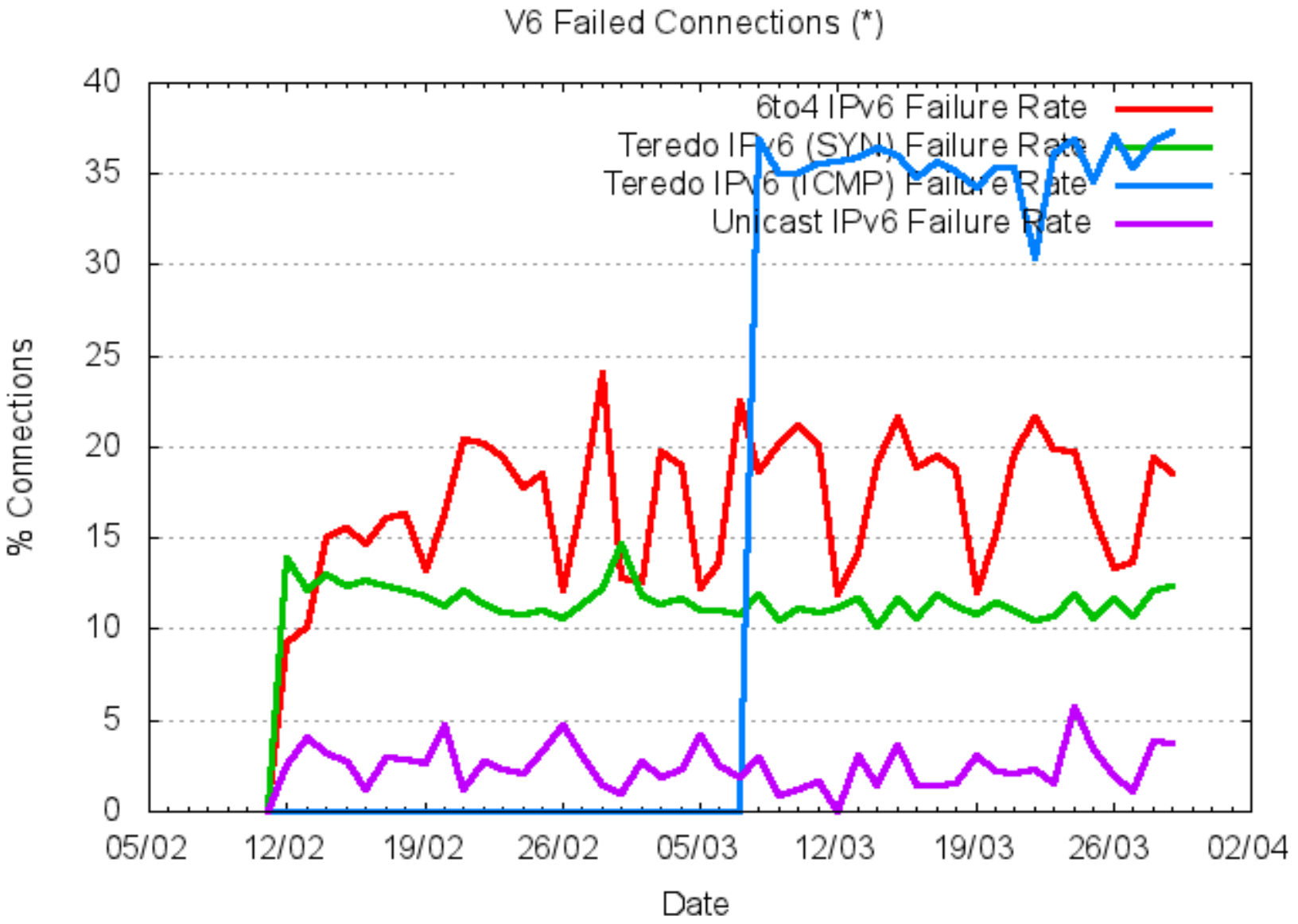
Teredo Connection Failure

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

IPv6 Connection Failure



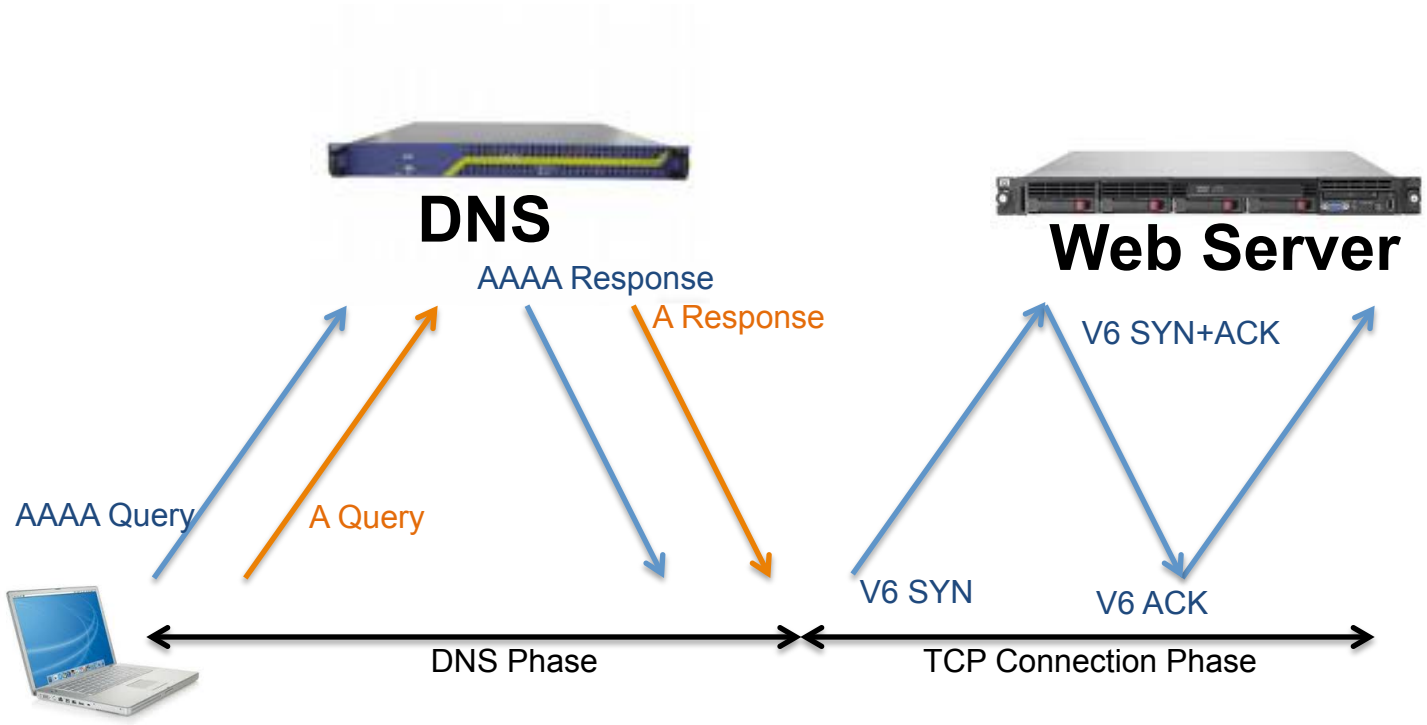
IPv6 Connection Failure

- 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 **35%** of **Teredo** connections fail!
  - This is an amazingly high failure rate!
  - Is STUN just broken? And/or …?

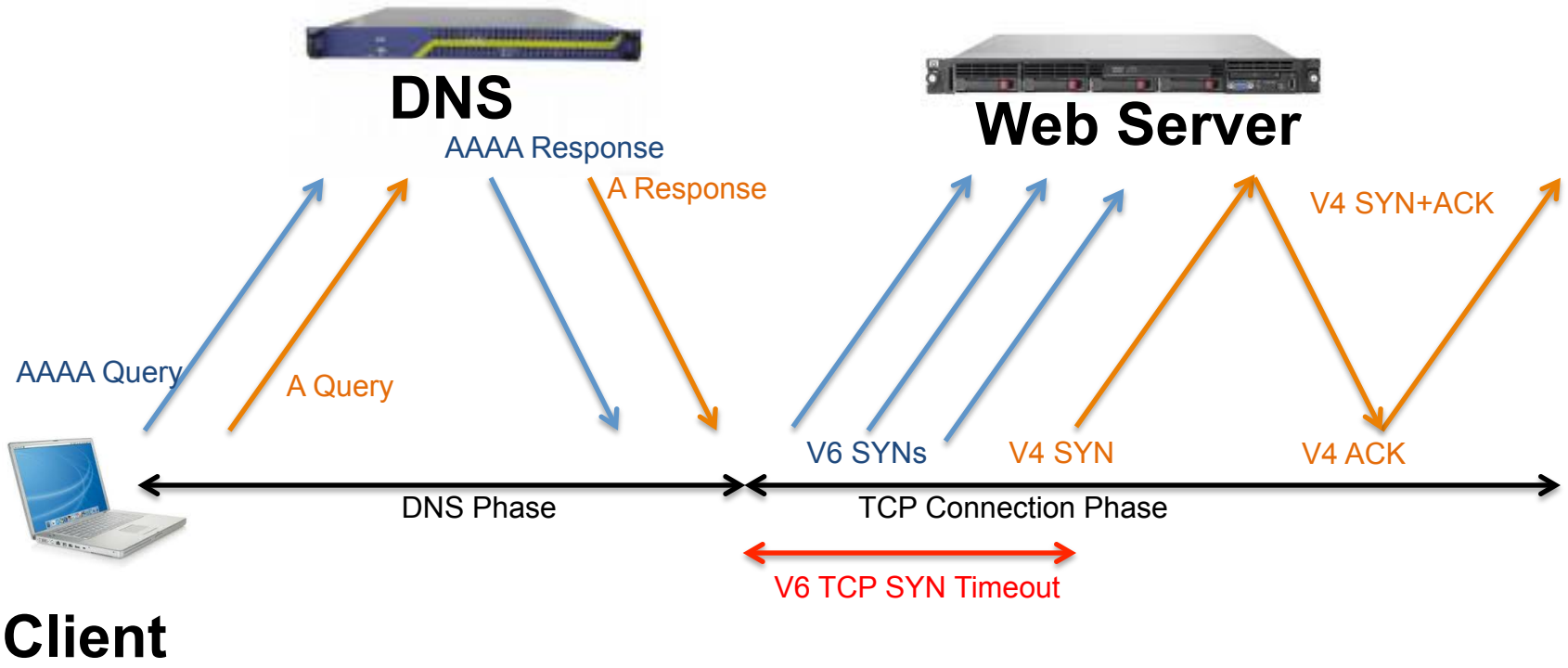
Can we improve Dual Stack Performance?

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

Serialization



Serialization and Failure



Serialization and Failure

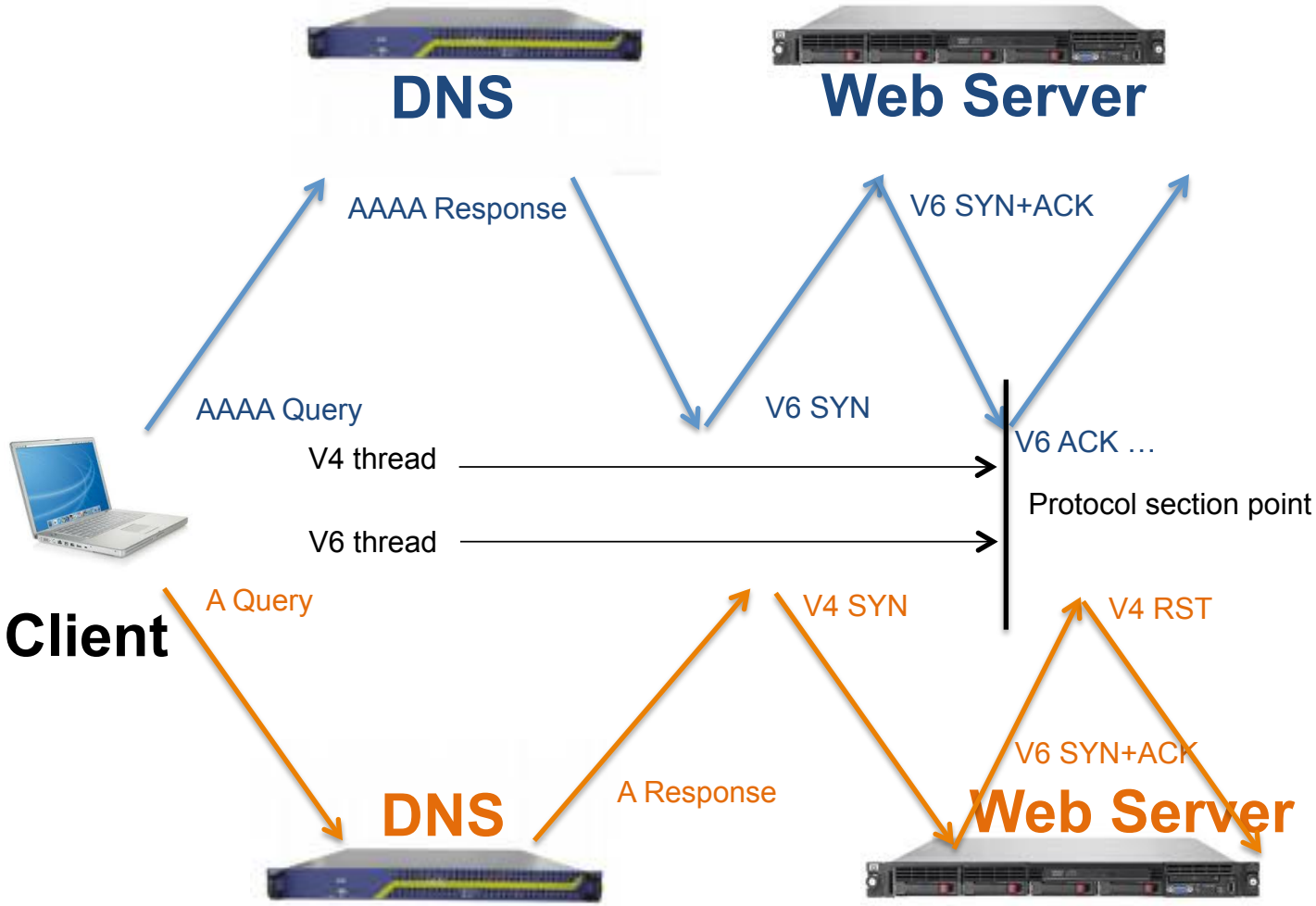
In response to poor performance associated with auto-tunnelling many OS stacks have responded by altering the local protocol preference table to depref 6to4 BELOW V4, and to try and not use Teredo at all!

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



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

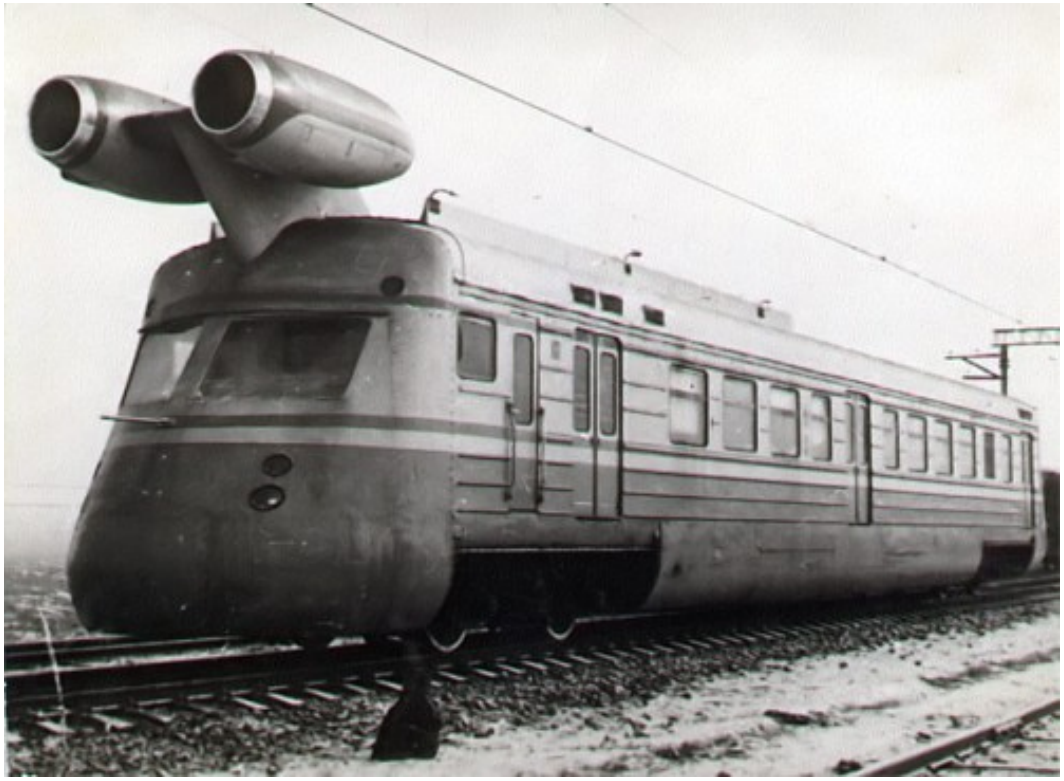
What about IPv6-Only Services?

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?