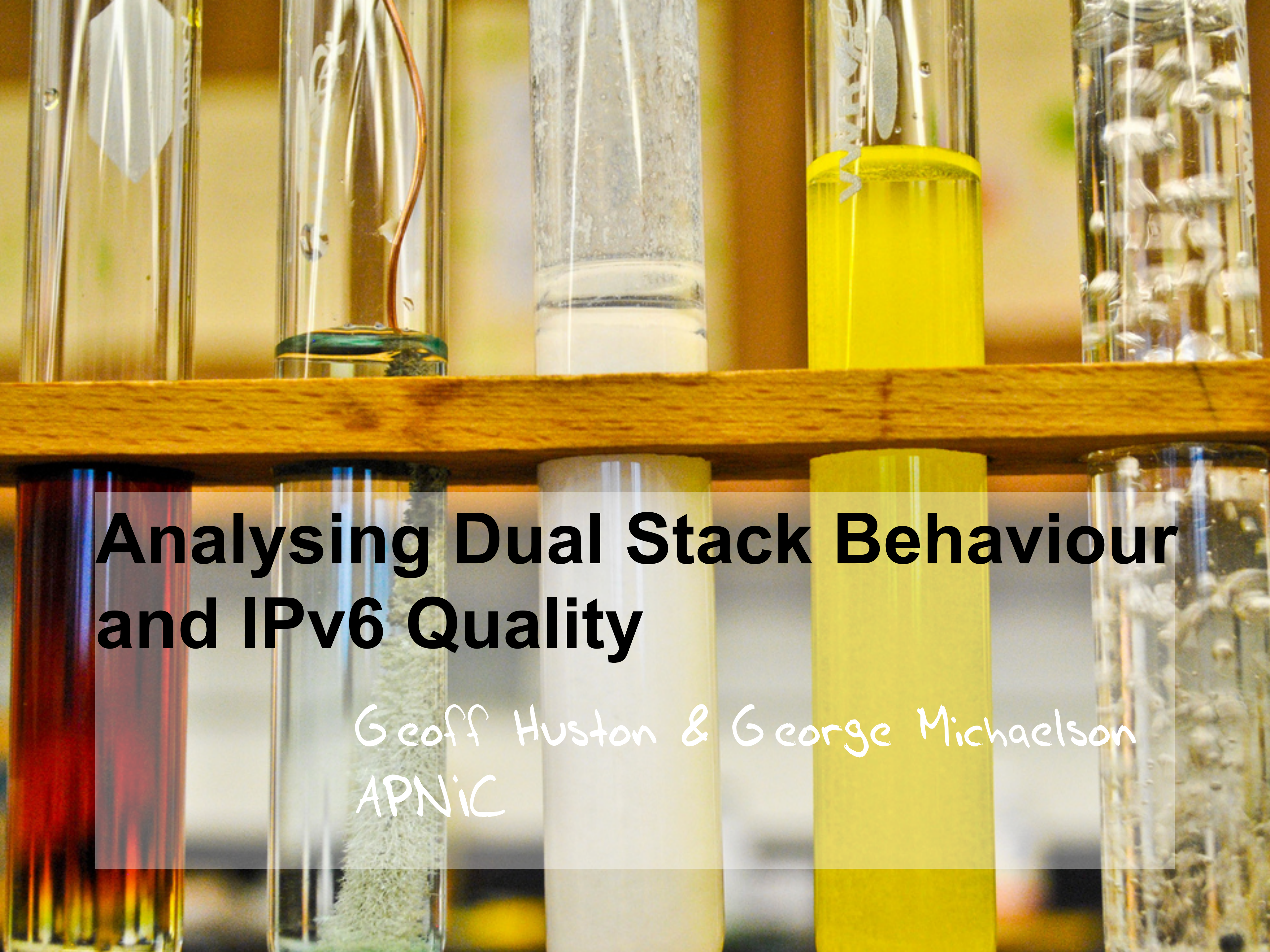
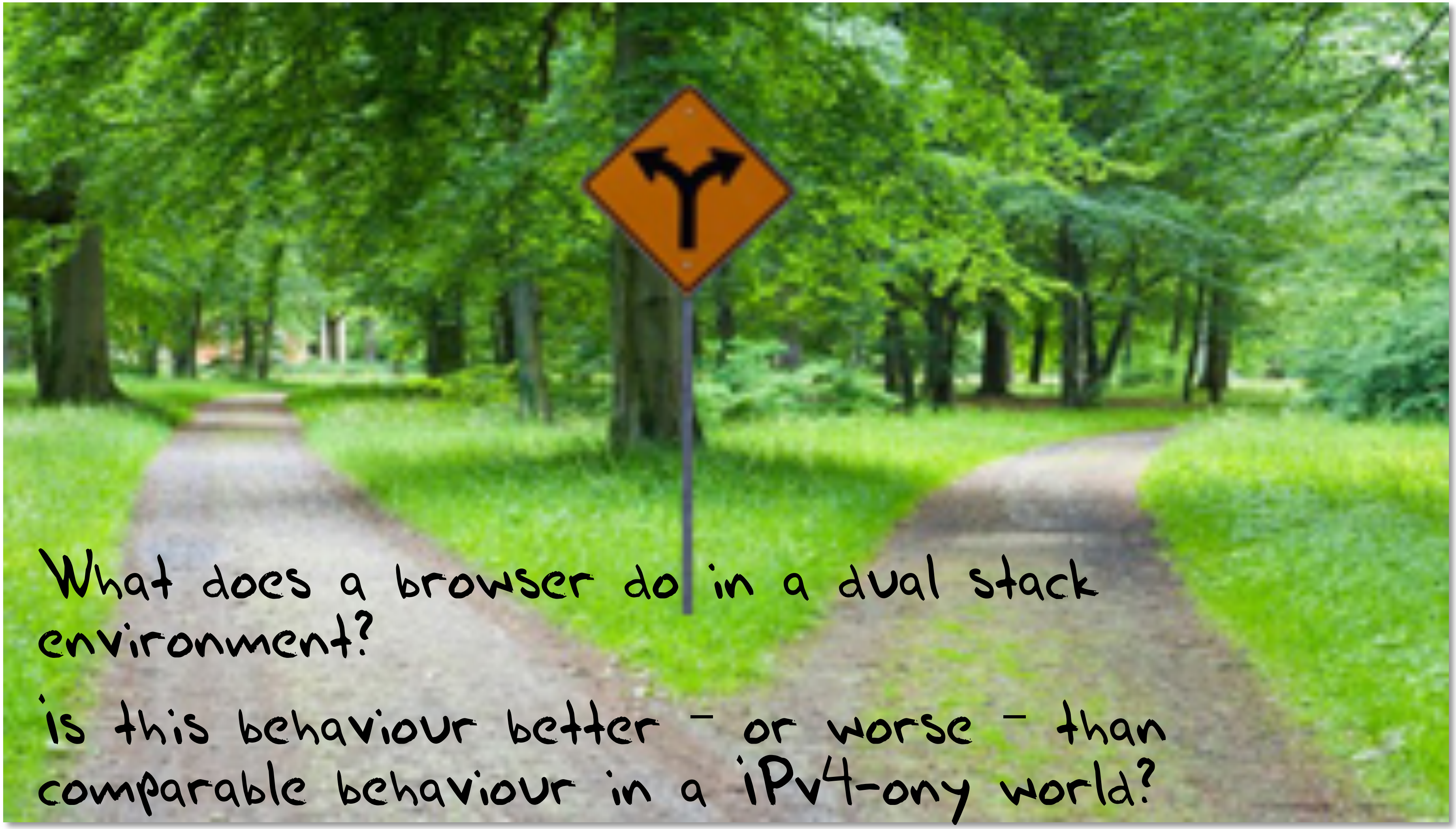


Analysing Dual Stack Behaviour and IPv6 Quality

Geoff Huston & George Michaelson
APNIC



What does a browser do in a dual stack environment?

is this behaviour better – or worse – than comparable behaviour in a iPv4-ony world?

Dual Stack Behaviour: V1.0

IPv6 First:

- Unconditional preference for IPv6 over IPv4

Dual Stack Behaviour: V1.0

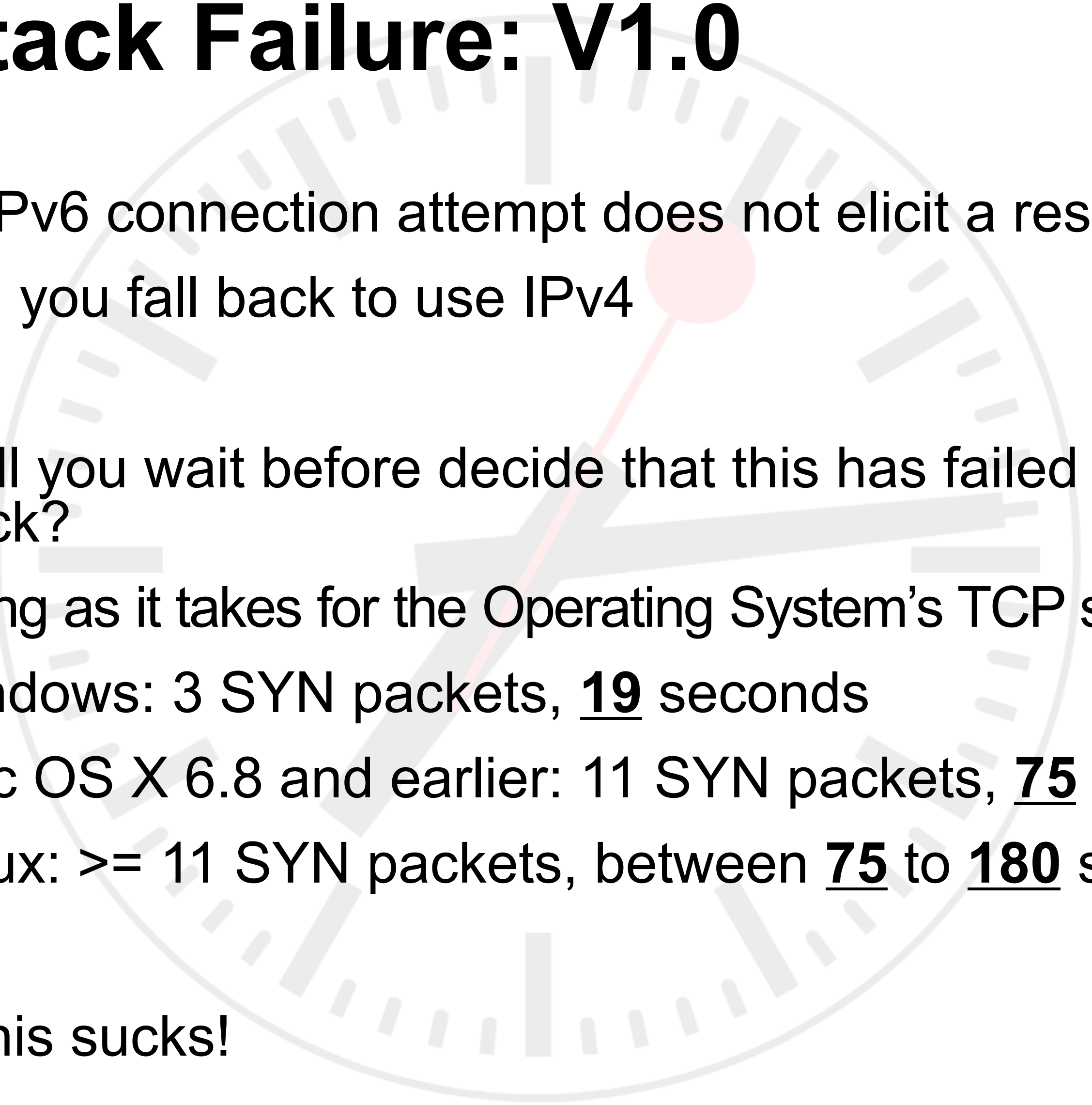
IPv6 First:

Unconditional preference for IPv6 over IPv4

If the local client has an active IPv6 interface then:

- Perform two DNS queries: A and AAAA record queries
- Wait for both to complete
- If the AAAA query succeeds then initiate the browser connection using IPv6
- If there is no AAAA record then initiate the browser connection using IPv4

Dual Stack Failure: V1.0



What if the IPv6 connection attempt does not elicit a response?

Then you fall back to use IPv4

How long will you wait before decide that this has failed and you need fall back?

As long as it takes for the Operating System's TCP system to fail

- Windows: 3 SYN packets, **19** seconds
- Mac OS X 6.8 and earlier: 11 SYN packets, **75** seconds
- Linux: >= 11 SYN packets, between **75** to **180** seconds

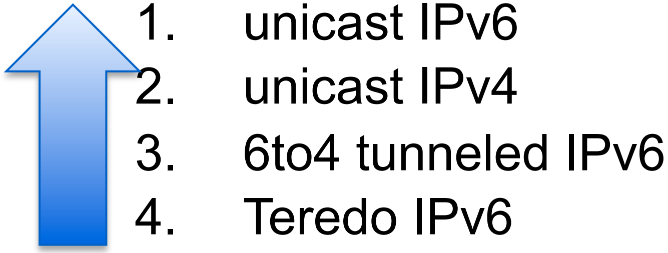
Obviously, this sucks!

Dual Stack Behaviour: V2.0

Native IPv6 First:

Unconditional preference for native IPv6 over IPv4

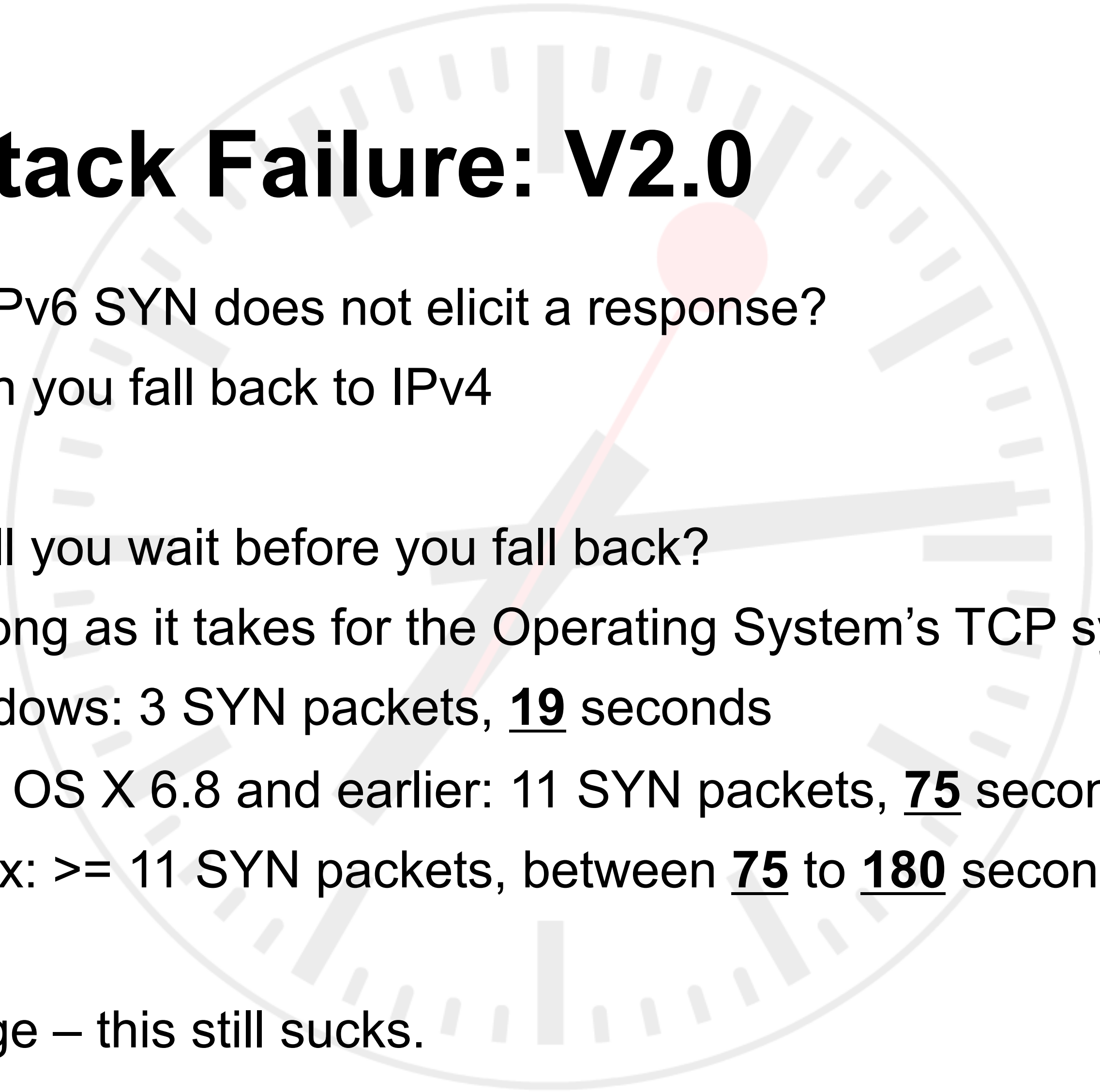
Add Local Preference Rules:



1. unicast IPv6
2. unicast IPv4
3. 6to4 tunneled IPv6
4. Teredo IPv6

The effect of this preference table is that if the local IPv6 interface is an auto-tunneled interface than it will only be used when there is no local unicast IPv6 interface and the remote site is IPv6-only

Dual Stack Failure: V2.0



What if the IPv6 SYN does not elicit a response?

Then you fall back to IPv4

How long will you wait before you fall back?

As long as it takes for the Operating System's TCP system to fail

Windows: 3 SYN packets, **19** seconds

Mac OS X 6.8 and earlier: 11 SYN packets, **75** seconds

Linux: >= 11 SYN packets, between **75** to **180** seconds

i.e. no change – this still sucks.

If you are behind a broken V6 connection, your life is still abject misery!

Dual Stack Behaviour: V2.5
Windows Vista and 7

While Vista and 7 has IPv6 “on” by default, if the system is behind a NAT the IPv6 interface is a auto-configured as a Teredo auto-tunnel interface

The modified behaviour is that these systems will not even query the DNS for a AAAA record if the only local IPv6 interface is a Teredo interface

- – i.e. the Teredo interface is only used when there is no precursor DNS lookup (e.g. use of IPv6 address literal form of URL)

Dual Stack Behaviour: V2.5

Native IPv6 First:

Unconditional preference for native IPv6 over IPv4

(and avoid Teredo)

Add Local Preference Rules:

1. unicast IPv6
2. unicast IPv4
3. ~~6to4 tunneled IPv6~~
4. Teredo IPv6

The effect of this is that if the Windows box is behind a NAT and does not have a unicast V6 connection then it shows IPv4-only behaviours

All this is broken!

- When the network sucks, this form of browser behaviour makes it suck even more!
- These serialized approaches to dual stack connectivity really don’t work well when there is a connection failure.
- The technique used to identify a failure falls back to a timeout – and this can be frustrating to the user if a default OS-provided timeout is used

We need better failures!

We need better failures!

- Altering the local preference rules may alter the chances of encountering a failure, but does not alter the poor method of determining when you have failed

The fine print: The real problem here is that the assumption behind the TCP connection code in most operating systems was that there was no fallback – you either connected to a given address or you report failure. To provide a behaviour that was robust under adverse network conditions the OS connection code is incredibly persistent (up to 3 minutes In the case of Linux default). But to use this same code in the circumstance where you have alternate connection possibilities is just testing the user's patience. So we need to rethink this and use a connection strategy that tests all possibilities in a far shorter elapsed time.

How to conduct a two horse race...

Start with one horse

How to conduct a two horse race...

Start with one horse

if it dies on the way then send off the other horse!

How to conduct a two horse race…

Or…

You can send off both horses at once and go with whichever is fastest…

Dual Stack Behaviour: V3.0
Safari and Mac OSX 10.7 and later

Moderately Happy Eyeballs:

- Determine the preference between IPv4 and IPv6 by maintaining a running performance metric of per-protocol average RTT to each cached destination address
- When DNS queries return both A and AAAA records initiate a connection using the protocol with the lowest current average RTT

Dual Stack Failure: V3.0
Safari and Mac OSX 10.7 and later

- If the connection is not established *within the RTT estimate time interval* then fire off a connection attempt in the other protocol
  - i.e. use a very aggressive timeout to trigger protocol fallback

Dual Stack Failure: V3.0
Safari and Mac OSX 10.7 and later

- ~~If the connection is not established *within the RTT estimate time interval* then fire off a connection attempt in the other protocol~~
  - ~~i.e. use a very aggressive timeout to trigger protocol fallback~~

Dual Stack Failure: V3.0
Safari and Mac OSX 10.7 and later

- ~~If the connection is not established *within the RTT estimate time interval* then fire off a connection attempt in the other protocol~~

Only when you have tried ALL the addresses in the first protocol family, then flip over to the other protocol

Dual Stack Failure: V3.0
Safari and Mac OSX 10.7 and later

- If the connection is not established RTT *estimate time interval the* attempt in the
- C addresses in the over to the other

Multi-addressing a critical service point in dual stack situations can make it look worse to clients, not better!

Dual Stack Behaviour: V3.1 Chrome Browser

Happy***ish*** Eyeballs:

- Fire off the A and AAAA DNS queries in parallel
- It’s a DNS race: Initiate a TCP connection with the first DNS response
- If the TCP connection fails to complete in 300ms then start up a second connection on the other protocol

Yes, 300ms is arbitrary. But assuming that a fast DNS response equates to a fast data path RTT is equally arbitrary!

Dual Stack Behaviour: V3.2 Firefox and Fast Failover

Happ*ier* Eyeballs:

- Fire off the A and AAAA DNS Queries
- Initiate a TCP connection as soon as the DNS response is received
- It's a SYN race: Use the first connection to complete the SYN-ACK handshake for data retrieval
- Close off the other connection

This makes a little more sense – now the data path RTT has some influence over protocol selection, and the user connection will proceed with the protocol that completes the connection in the least time

The bigger picture...

	Firefox
MAC OS X 10.7.2	8.0.1 75s IPv6

Protocol Preference Setting

Failover Timer Values

http://www.potaroo.net/ispcol/2011-12/esotropia.html

The bigger picture...

	Firefox	Firefox fast-fail	Chrome	Opera	Safari	Explorer
MAC OS X 10.7.2	8.0.1 75s IPv6	8.0.1 0ms SYN+ACK	6.9.912.41 300ms DNS	11.52 75s IPv6	5.1.1 270ms RTT	
Windows 7	8.0.1 21s IPv6	8.0.1 0ms SYN+ACK	.0.874.121 300ms DNS	11.52 21s IPv6	5.1.1 21s IPv6	9.0.8112 21s IPv6
Windows XP	8.0.1 21s IPv6	8.0.1 0ms SYN+ACK	.0.874.121 300ms DNS	11.52 21ds IPv6	5.1.1 21s IPv6	9.0.8112 21s IPv6
Linux 2.6.40-3.0	8.0.1 96s IPv6	8.0.1 0ms SYN+ACK		11.60 bets 189s IPv6		
iOS 5.0.1					? 720ms RTT	

http://www.potaroo.net/ispcol/2011-12/esotropia.html

Why?

- Why add all this parallel complexity to browser behaviour?
- What was wrong with the initial concept of “prefer IPv6 if you can, use IPv4 otherwise”?
- Is there really any difference in performance between IPv6 connections?
- Lets see...

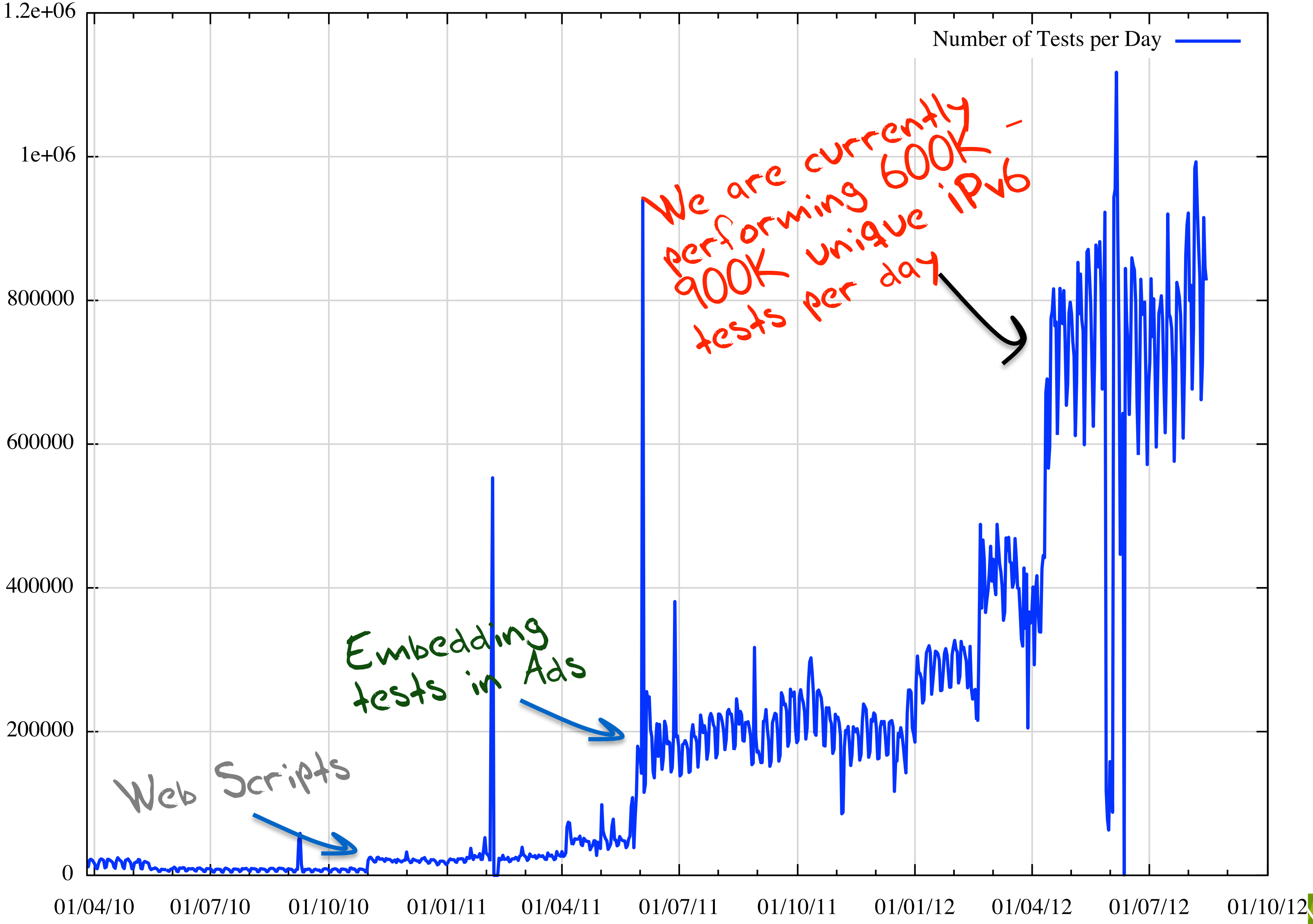
Measuring Dual Stack Quality

Enlist a large set of dual stack clients to connect to an instrumented server using both IPv4 and IPv6

- Equip a number of web sites with a javascript module that poses a number of image-blot retrieval tests
- Extended this using Flash to embed the same tests in a Google Image Ad*

* Thank you to Google, ISOC RIPE NCC & ISC for assistance in conducting this experiment!

Test Volume – Number of unique tests performed per day



Web Scripts

Embedding tests in Ads

We are currently performing 600K – 900K unique iPv6 tests per day

Measuring Dual Stack Quality

Enlist a large set of dual stack clients to connect to an instrumented server using both IPv4 and IPv6

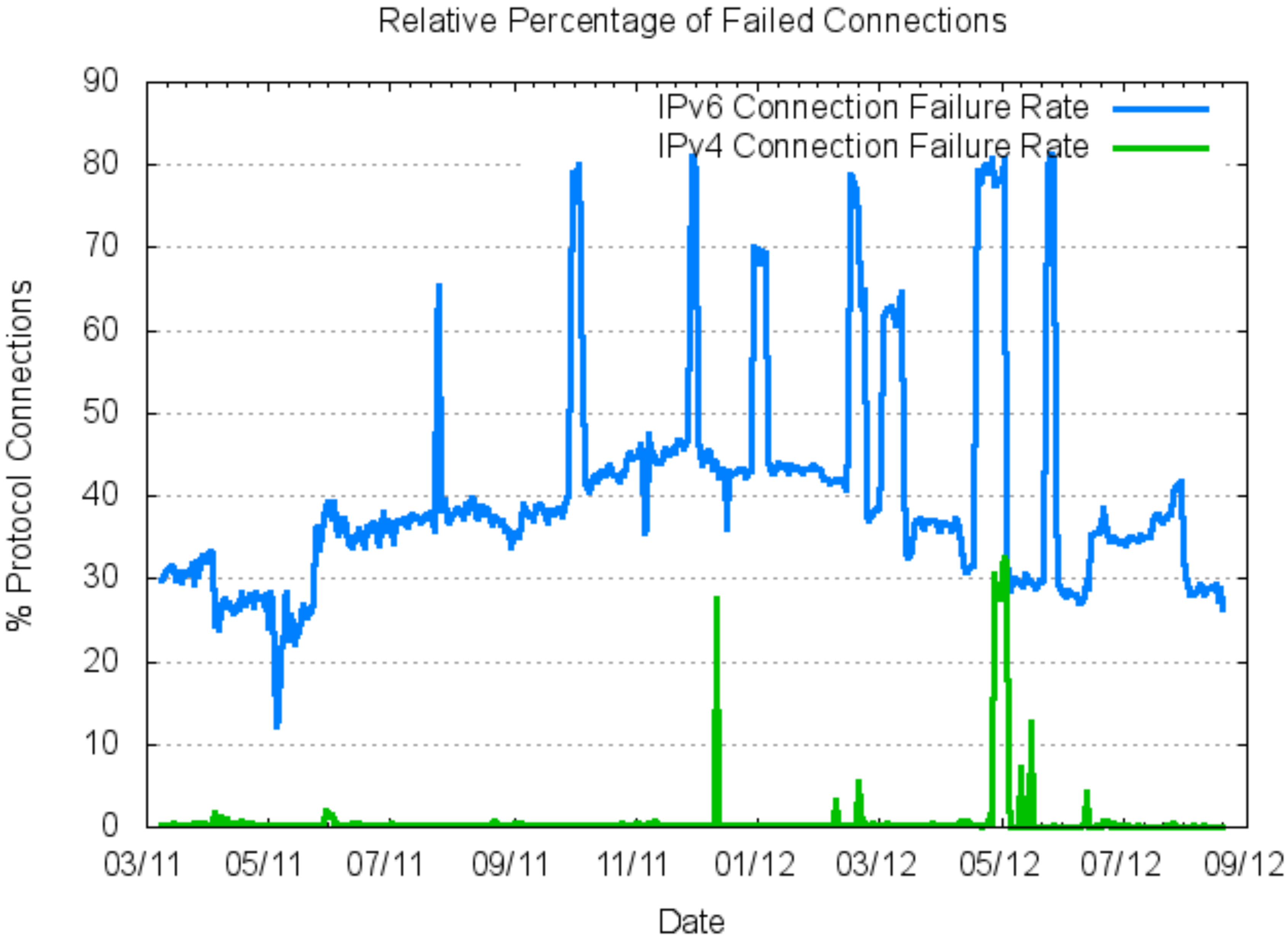
- For each successful connection couplet gather the pair of RTT measurements on the SYN-ACK exchanges
- Gather connection failure statistics (where a “failure” is defined as a received SYN, but no followup ACK)

Connection Failure

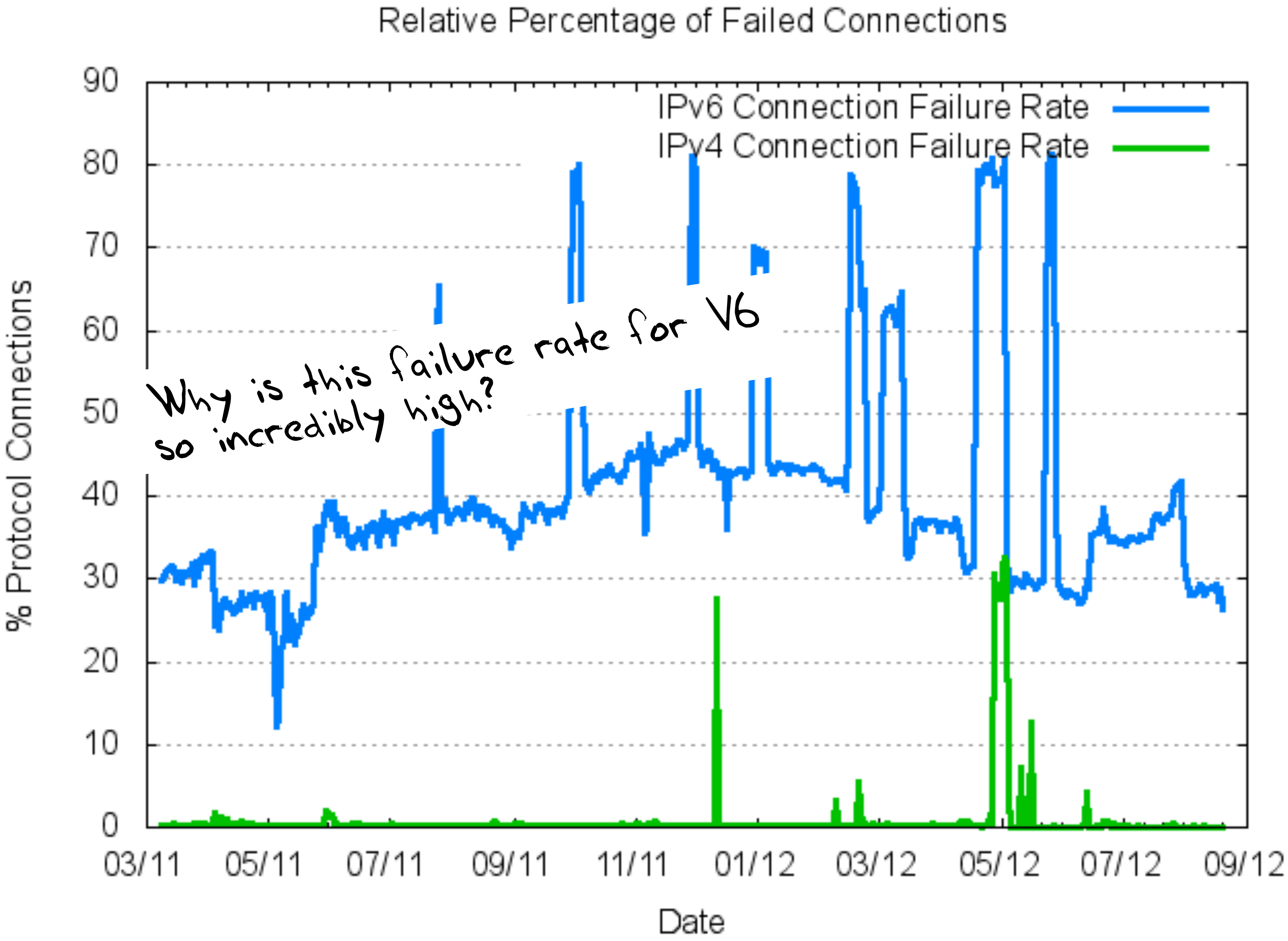
Outbound SYN

Busted SYN ACK
Return path

Measuring Failure

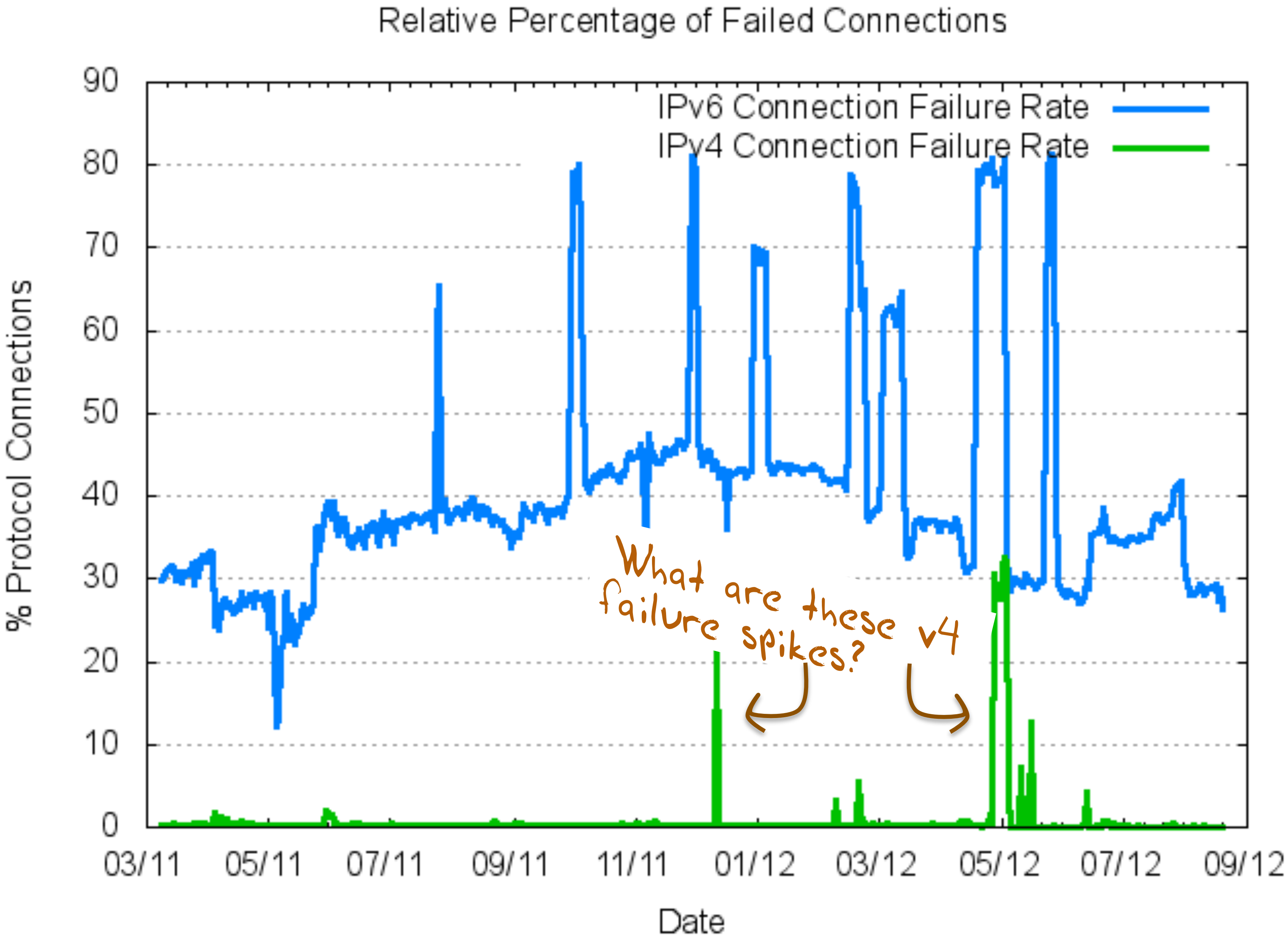


Measuring Failure



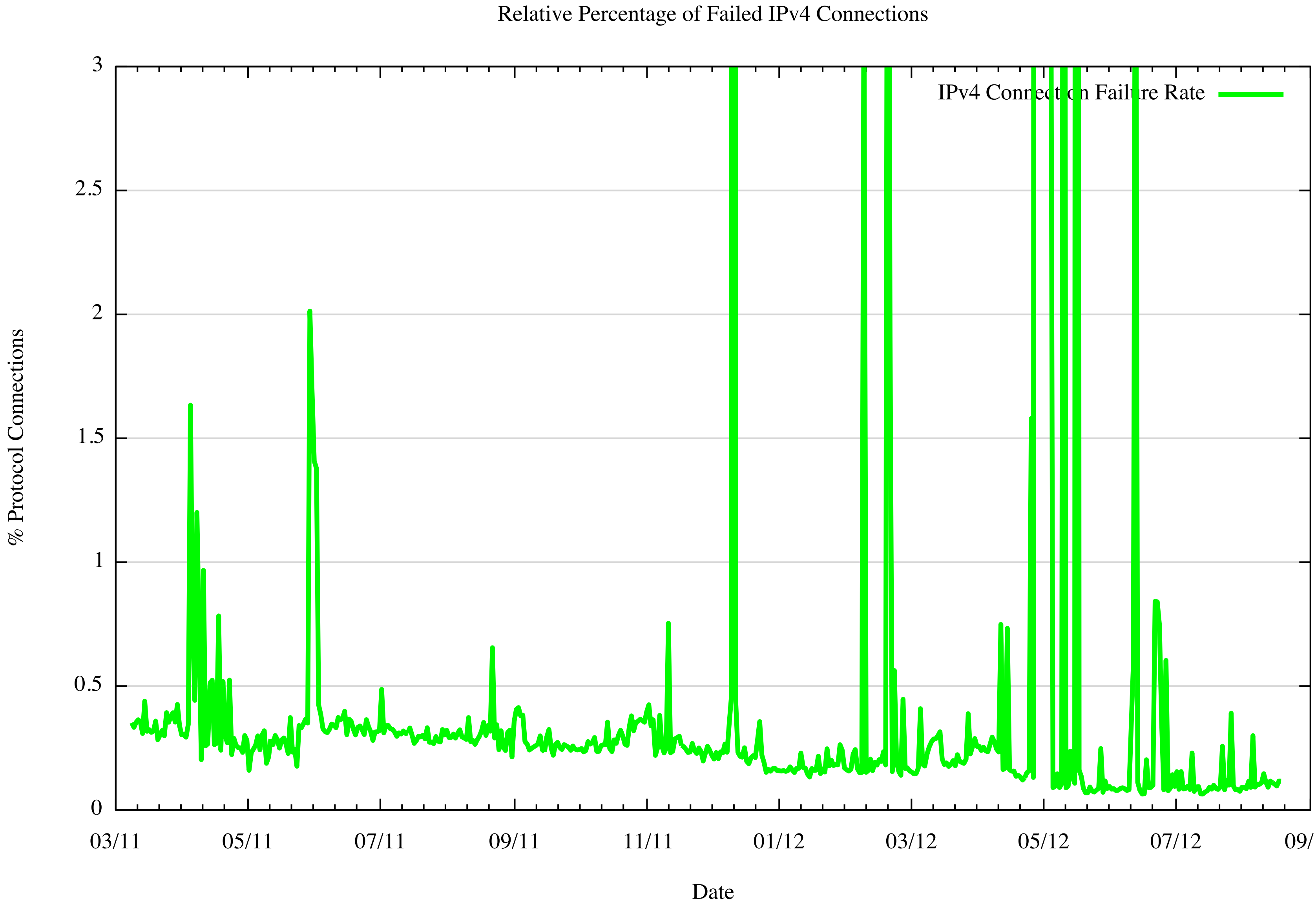
Why is this failure rate for V6 so incredibly high?

Measuring Failure



What are these v4 failure spikes?

What is going on with IPv4?

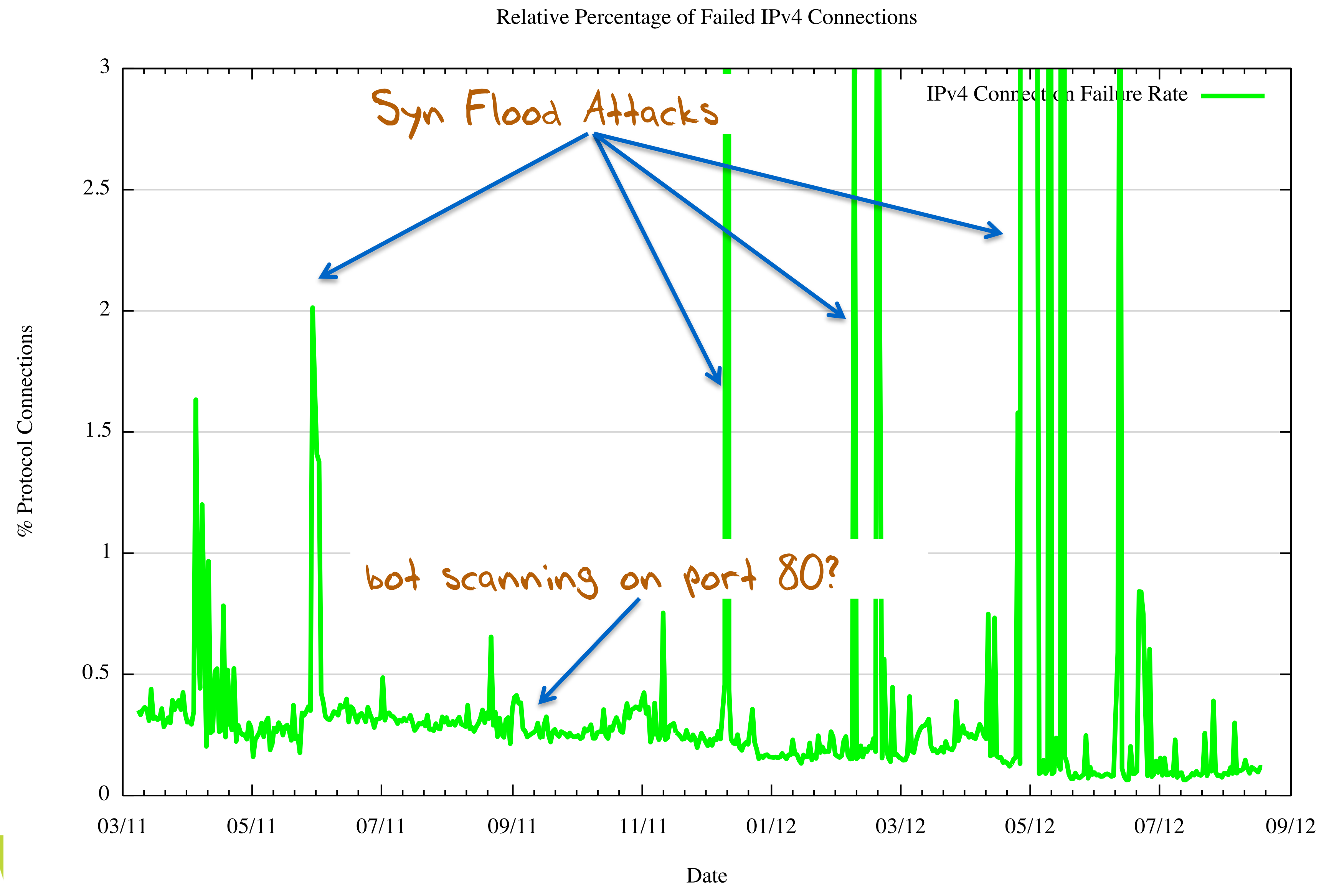


What is going on with IPv4?

The failure rate for V4 decreases as the volume of experiments increases – which implies that the number of “naked SYNs” being sent to the servers is not related to the number of tests being performed.

Aside from residual IPv4 failures in the image fetch due to device resets, connection dropouts, etc, the bulk of the recorded failures here is probably attributable to bots doing address scanning on port 80

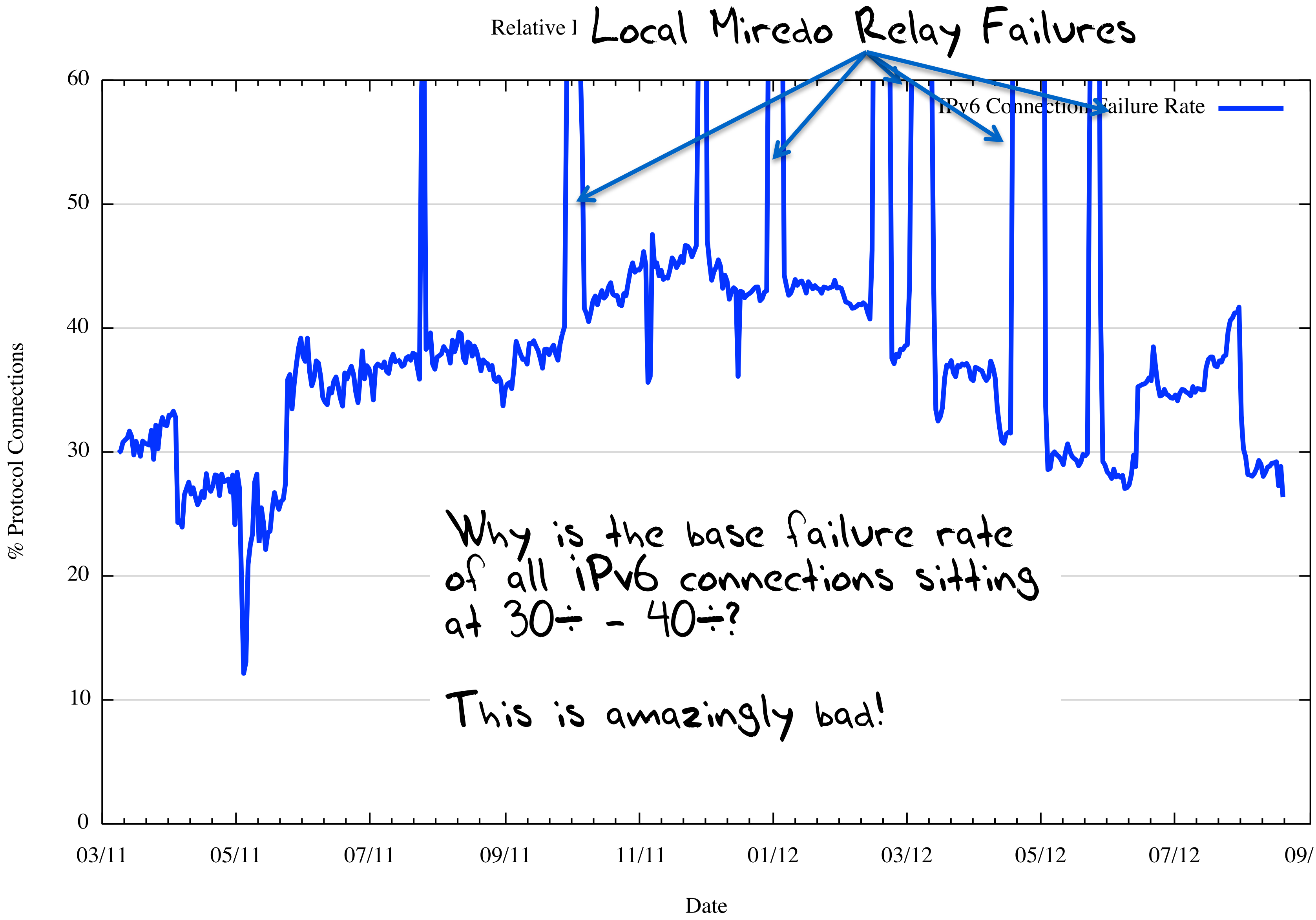
What is going on with IPv4?



Syn Flood Attacks

bot scanning on port 80?

What about IPv6?

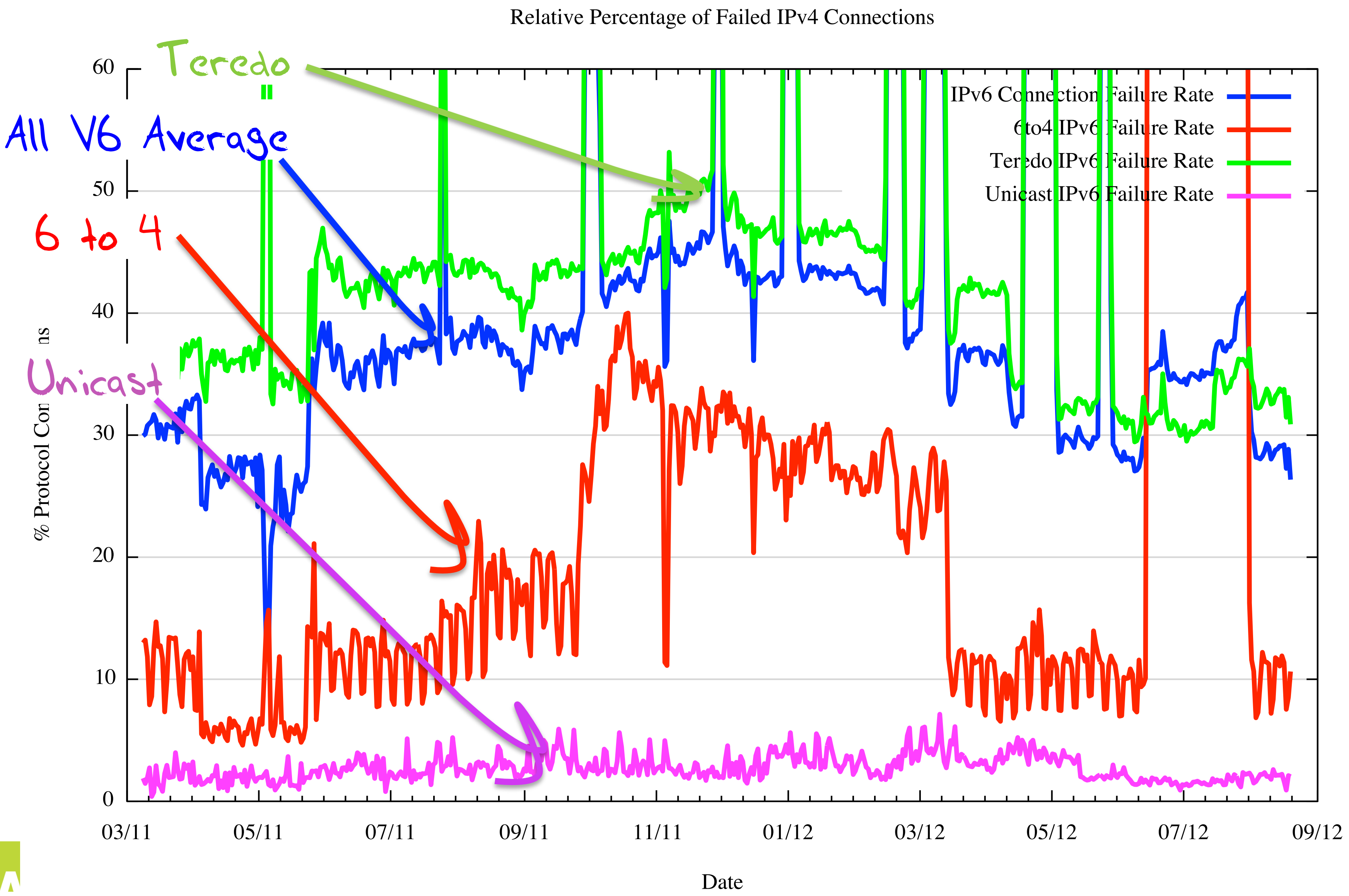


Local Miredo Relay Failures

Why is the base failure rate of all IPv6 connections sitting at 30% - 40%?

This is amazingly bad!

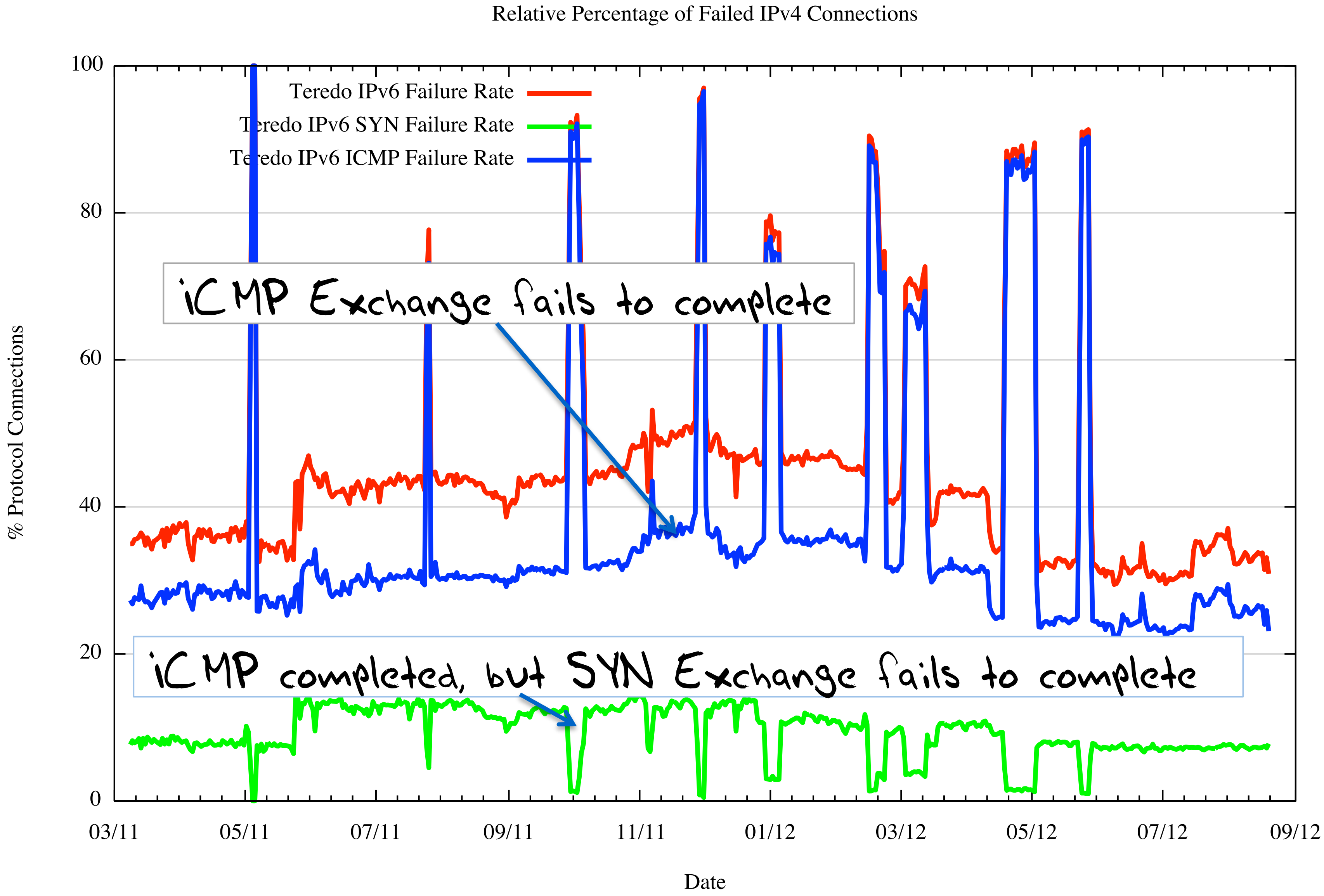
V6 Failure Rate by Address Type



Teredo Failures

- Teredo connections use a 2-step connection process:
  - An ICMP exchange to establish the form of local NAT behaviour (full cone, port restricted cone, ...) and to set up the symmetric path
  - A TCP 3-way handshake
- There are 2 failure modes:
  - ICMP seen, no SYN
  - ICMP seen, SYN seen, no ACK

Teredo Failure Rate



iCMP Exchange fails to complete

iCMP completed, but SYN Exchange fails to complete

It’s NAT Traversal Failure

- Teredo failure is around 35% of all connection attempts
  - Obviously, this is unacceptably high!
  - This is unlikely to be local filtering effects given that Teredo presents to the local NAT as conventional IPv4 UDP packets
  - More likely is the failure of the Teredo protocol to correctly identify the behaviour mode of the local NAT device

Working with Failure

A 35% connection failure is unworkable is *almost* all circumstances

But one particular application can thrive in this environment, and makes use of Teredo addresses - torrents

- Not many DPI interceptors are sensitive to V6 in V4 UDP encap
- The massive redundancy of the data set across multiple sources reduces the sensitivity of individual session failures

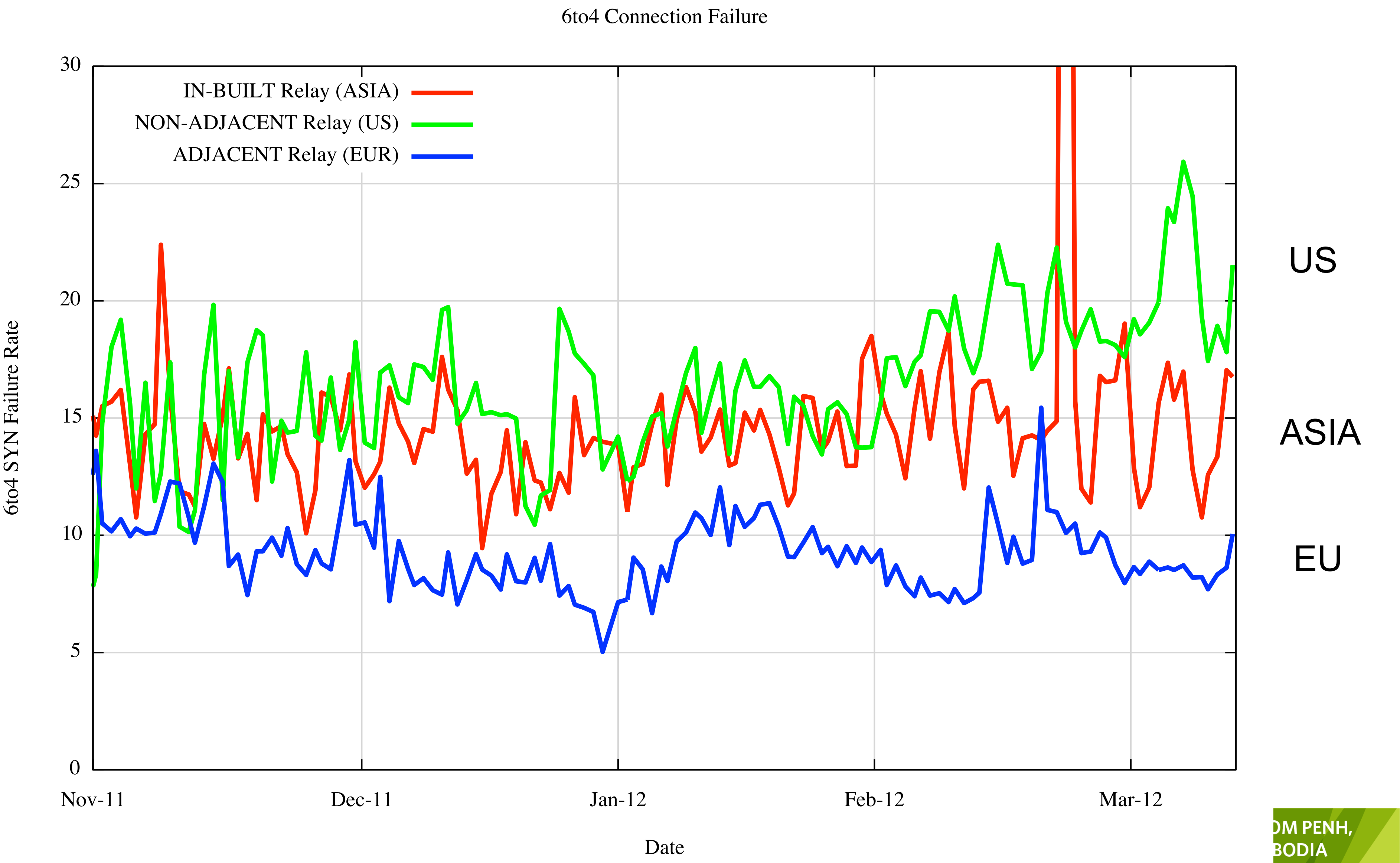
6to4 Auto-tunnelling

6to4 Auto-tunnelling technique

- Cannot operate through IPv4 NATs
- Relies on third party relays in BOTH directions
- Asymmetric traffic paths

- Some of the performance problems can be mitigated by placing the reverse 6to4 relay into the V6 service point

6to4 Failure Rate



6to4 Failure is Local Failure

6to4 failure appears to be related to two factors:

1. The client’s site has a protocol 41 firewall filter rule for incoming traffic (this is possibly more prevalent in AsiaPac than in Europe)
2. Load / delay / reliability issues in the server’s chosen outbound 6to4 relay (noted in the data gathered at the US server)

Even so, the 10% to 20% connection failure rate for 6to4 is unacceptably high!

V6 Unicast Failures

January – August2012:

962,737 successful V6 connecting endpoints
22,923 failures

That's a connection failure rate of 2.3%!

13 clients used fe80:: link local addresses
139 clients used fc00:/7 ULA source addresses
22 clients used fec0::/16 deprecated site local addresses
16 clients used 1f02:d9fc::/16
1 client used 1f01:7e87:12:10ca::/64
1 client used a 3ffe::/16 address
7 clients used :: IPv4 –mapped addresses (10/8, 192.168/16)
7 clients used ::ffff:<IPv4>-mapped addresses

What about the other 22,717 clients?

Unicast IPv6 Failures

38 were using unallocated unicast V6 addresses

150 were using unadvertised unicast V6 addresses

22,529 were using V6 addresses drawn from conventional advertised V6 prefixes!

Local inbound filters appear to be a common problem in IPv6

Where does V6 Fail?

Average - 2.3% of unicast V6 connections fail to complet

However, we saw wide variance across countries:

Highest:

Pakistan - 35%

Hong Kong - 18%

Canada - 12%

Vietnam – 12%

Romania – 10%

Indonesia – 10%

Taiwan – 10%

Malaysia – 7%

New Zealand – 7%

Lowest:

France – 0.3%

UK – 0.3%

Germany – 0.9%

Norway – 0.9%

Australia – 0.9%

Japan - 1%

Greece – 1%

Italy – 1%

Finland – 1%

The “Good” IPv6 AS’s

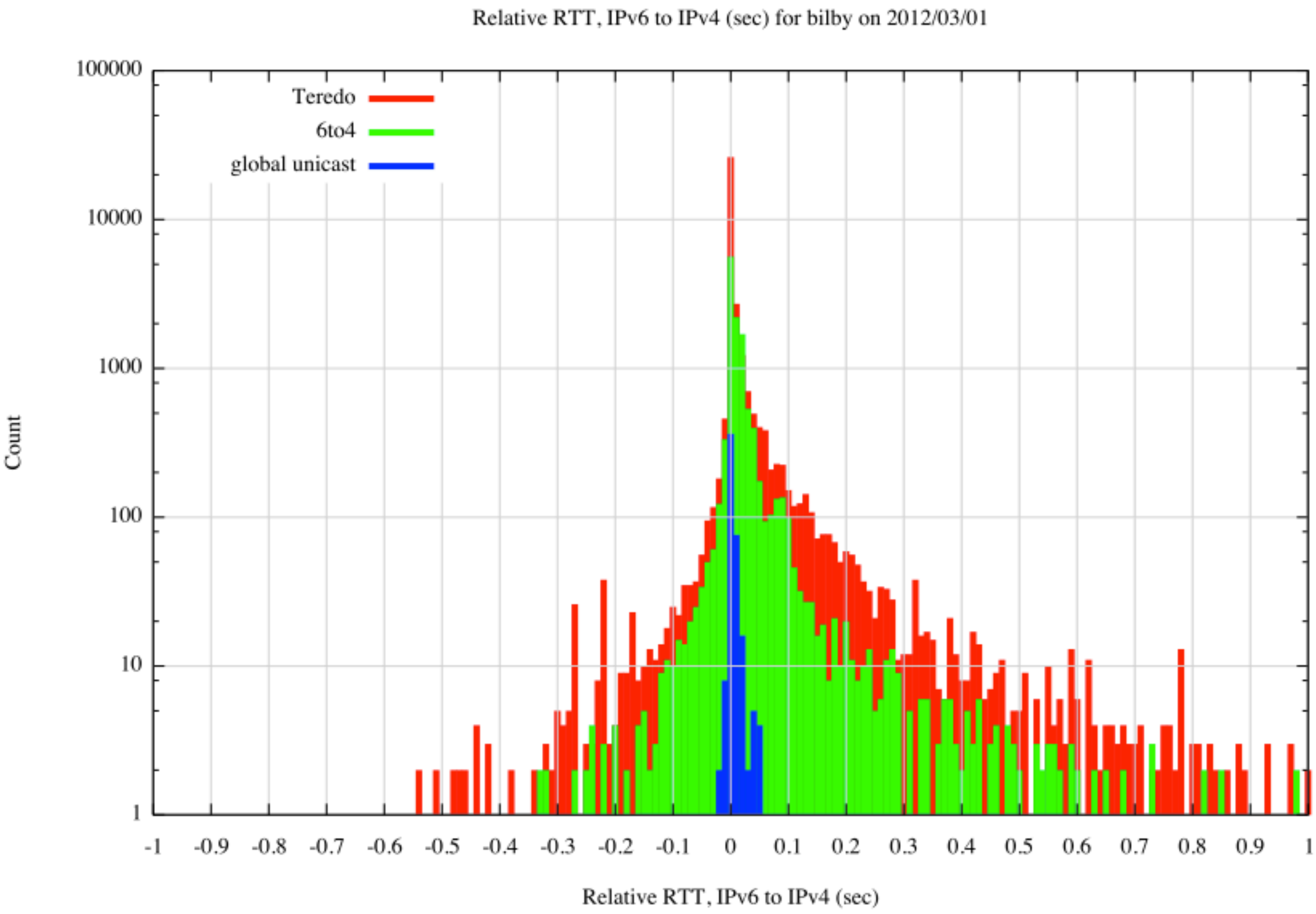
```
AS        V6 connection   AS Description
          Failure Rate
AS38083     0.0%     AU CURTIN-UNI-AS-AP Curtin University
AS24226     0.1%     NZ CATALYST-IT-AS-AP Catalyst IT
AS1312      0.1%     US VA-TECH-AS - Virginia Polytechnic Institute and State Univ.
AS12552     0.1%     SE IPO-EU IP-Only Telecommunication Networks AB
AS31334     0.1%     DE KABELDEUTSCHLAND-AS Kabel Deutschland Vertrieb und Service GmbH
AS237       0.1%     US MERIT-AS-14 - Merit Network Inc.
AS55        0.2%     US UPENN-CIS - University of Pennsylvania
AS17727     0.2%     ID NAPINFO-AS-AP PT. NAP Info Lintas Nusa
AS21453     0.2%     RU FLEX-AS Flex Ltd
AS2516      0.2%     JP KDDI KDDI CORPORATION
AS6661      0.2%     LU EPT-LU Entreprise des P. et T. Luxembourg
AS2107      0.2%     SI ARNES-NET ARNES
AS12322     0.2%     FR PROXAD Free SAS
AS3676      0.2%     US UIOWA-AS - University of Iowa
AS4802      0.3%     AU ASN-IINET iiNet Limited
AS39326     0.3%     GB GOSCOMB-AS Goscomb Technologies Limited
AS53347     0.3%     US PREMIER-COMMUNICATIONS - Premier Communications
AS3333      0.3%     NL RIPE-NCC-AS Reseaux IP Europeens Network Coordination Centre (RIPE NCC)
AS22394     0.3%     US CELLCO - Cellco Partnership DBA Verizon Wireless
AS19782     0.3%     US INDIANAGIGAPOP - Indiana University
AS5661      0.3%     US USF - UNIVERSITY OF SOUTH FLORIDA
AS4608      0.3%     AU APNIC-AP Asia Pacific Network Information Centre
AS3582      0.3%     US UONET - University of Oregon
AS22548     0.3%     BR Comite Gestor da Internet no Brasil
AS8426      0.3%     ES CLARANET-AS ClaraNET LTD
AS2852      0.4%     CZ CESNET2 CESNET, z.s.p.o.
AS57        0.4%     US UMN-REI-UC - University of Minnesota
AS7018      0.4%     US ATT-INTERNET4 - AT&T Services, Inc.
AS1103      0.4%     NL SURFNET-NL SURFnet, The Netherlands
AS55391     0.5%     JP MF-NATIVE6-E INTERNET MULTIFEED CO.
```

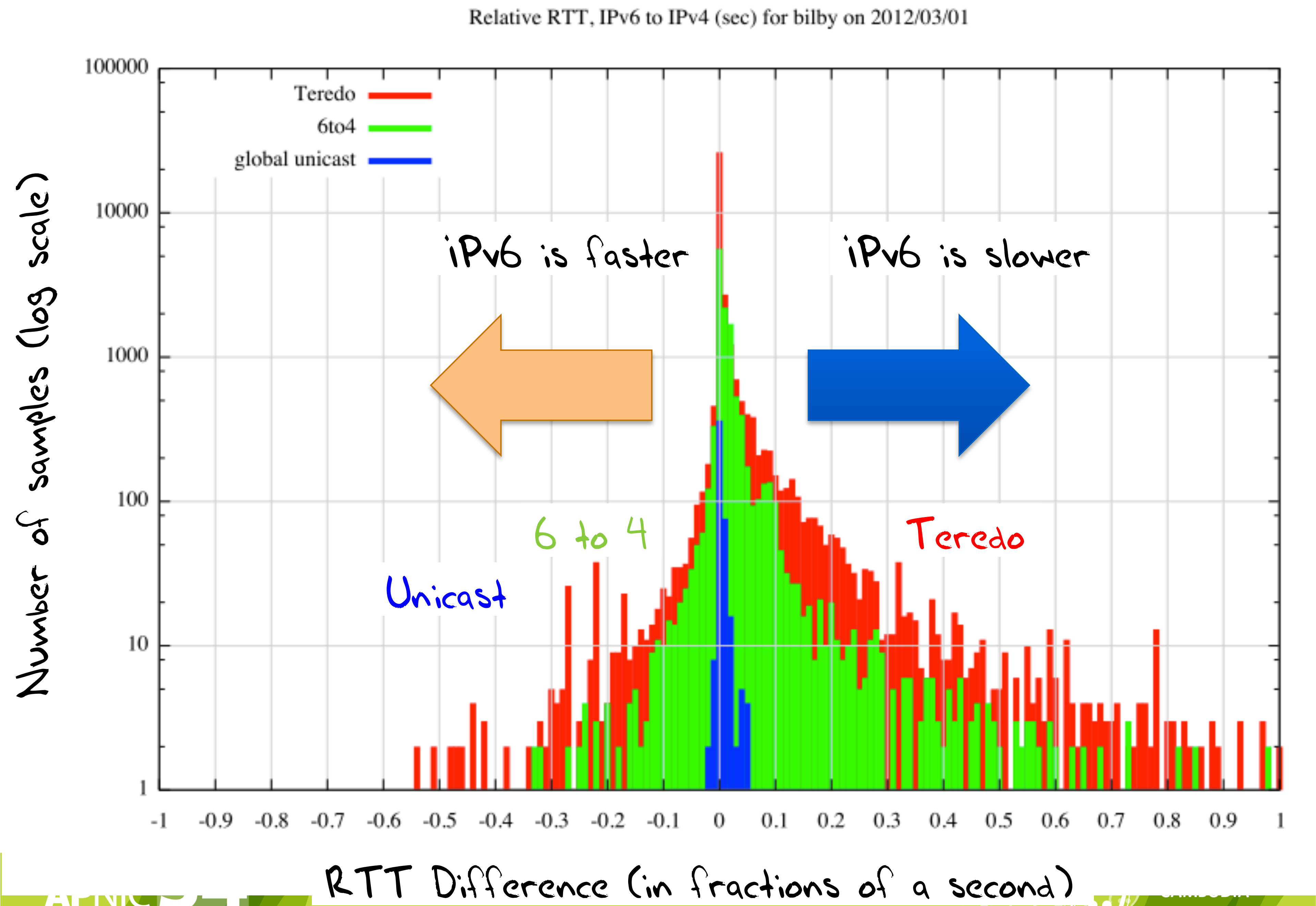
The “Not So Good” IPv6 AS’s

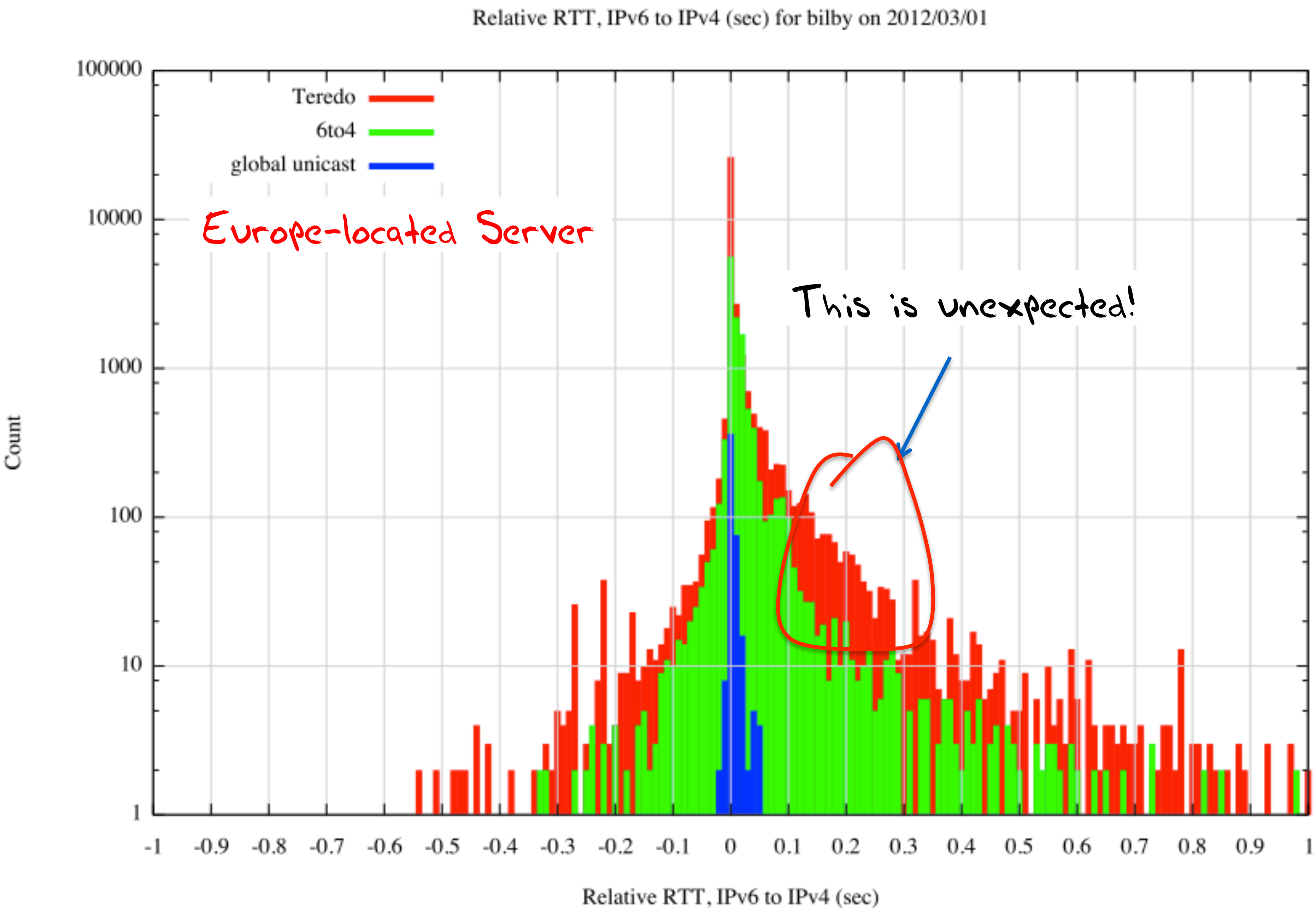
AS	V6 connection Failure Rate	AS Description
AS29113	12.5%	CZ SLOANE-AS UPC Ceska Republica, s.r.o.
AS1659	12.6%	TW ERX-TANET-ASN1 Tiawan Academic Network (TANet) Information Center
AS45230	12.6%	NZ UBERGROUP-AS-NZ UberGroup Limited
AS18119	12.8%	NZ ACSDATA-NZ ACSData
AS17451	13.6%	ID BIZNET-AS-AP BIZNET ISP
AS24173	13.8%	VN NETNAM-AS-AP Netnam Company
AS12271	15.1%	US SCRR-12271 - Road Runner HoldCo LLC
AS17709	16.8%	TW EBT Eastern Broadband Telecom Co.,Ltd
AS11427	18.4%	US SCRR-11427 - Road Runner HoldCo LLC
AS2907	18.4%	JP SINET-AS Research Organization of Information and Systems, National Institute of Informatics
AS8591	19.2%	SI AMIS AMiS
AS812	19.6%	CA ROGERS-CABLE - Rogers Cable Communications Inc.
AS12046	19.8%	MT ASN-CSC-UOM University of Malta
AS3356	20.2%	US LEVEL3 Level 3 Communications
AS4725	20.9%	JP ODN SOFTBANK TELECOM Corp.
AS8970	21.6%	PL WASK WROCMAN-EDU educational part of WASK network, Wroclaw, Poland
AS17579	21.6%	KR KREONET2-AS-KR Korea Institute of Science and Technology Information
AS7539	22.3%	TW TANET2-TW TANet2, sponsored by NSC, TAIWAN
AS3262	22.9%	ES SARENET SAREnet, Spain
AS11537	25.3%	US ABILENE - Internet2
AS16880	32.5%	US TRENDMICRO Global IDC and Backbone of Trend Micro Inc.
AS9431	33.2%	NZ AKUNI-NZ The University of Auckland
AS4528	33.6%	HK HKU-AS-HK The University of Hong Kong
AS45809	34.7%	NZ NZRS-AS-AP ASN for .nz registry content
AS2576	42.0%	US DOT-AS - U. S. Department of Transportation
AS17996	42.3%	ID UIINET-ID-AP PT Global Prima Utama
AS3562	45.5%	PK SNLL-NET-AS - Sandia National Laboratories
AS24514	58.6%	MY MYREN-MY Malaysian Research & Education Network

Measuring Dual Stack Quality

- For each successful connection couplet gather the pair of RTT measurements on the SYN-ACK exchanges
  - Use the server's web logs to associate a couplet of IPv4 and IPv6 addresses
  - Use the packet dumps to collect RTT information from the SYN-ACK Exchange


Relative RTT, IPv6 to IPv4 (sec) for bilby on 2012/03/01
Teredo
6to4
global unicast
Count
100000
10000
1000
100
10
1
-1
-0.9
-0.8
-0.7
-0.6
-0.5
-0.4
-0.3
-0.2
-0.1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Relative RTT, IPv6 to IPv4 (sec)


Relative RTT, IPv6 to IPv4 (sec) for bilby on 2012/03/01
Teredo 🟥
6to4 🟩
global unicast 🟦
100000
10000
1000
100
10
1
-1
-0.9
-0.8
-0.7
-0.6
-0.5
-0.4
-0.3
-0.2
-0.1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
iPv6 is faster
iPv6 is slower
6 to 4
Teredo
Unicast
Number of samples (log scale)
RTT Difference (in fractions of a second)



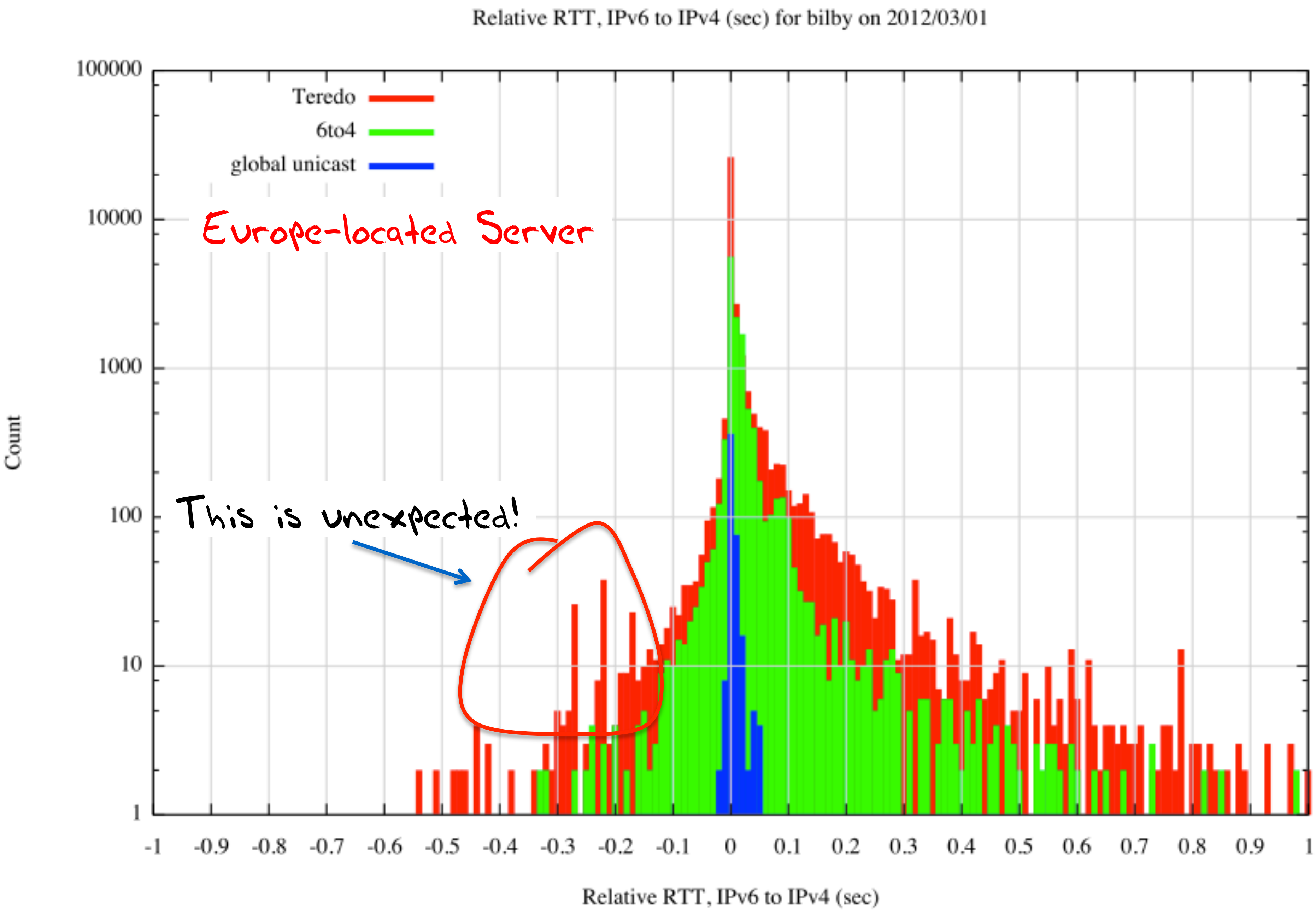
Europe-located Server

This is unexpected!

Why is Teredo slower?

The technique used here is to measure the interval between the first received SYN and the first received ACK

- But something is happening with Teredo
  - we use inbuilt Teredo Relays, so the Teredo RTT should precisely match the IPv4 RTT
    - But we are measuring the initial SYN exchange
    - It appears that there are some major setup delays in Teredo that are occurring in the initial SYN ACK exchange
    - The performance of CPE based NATs has a massive tail of delay, woe and abject misery!

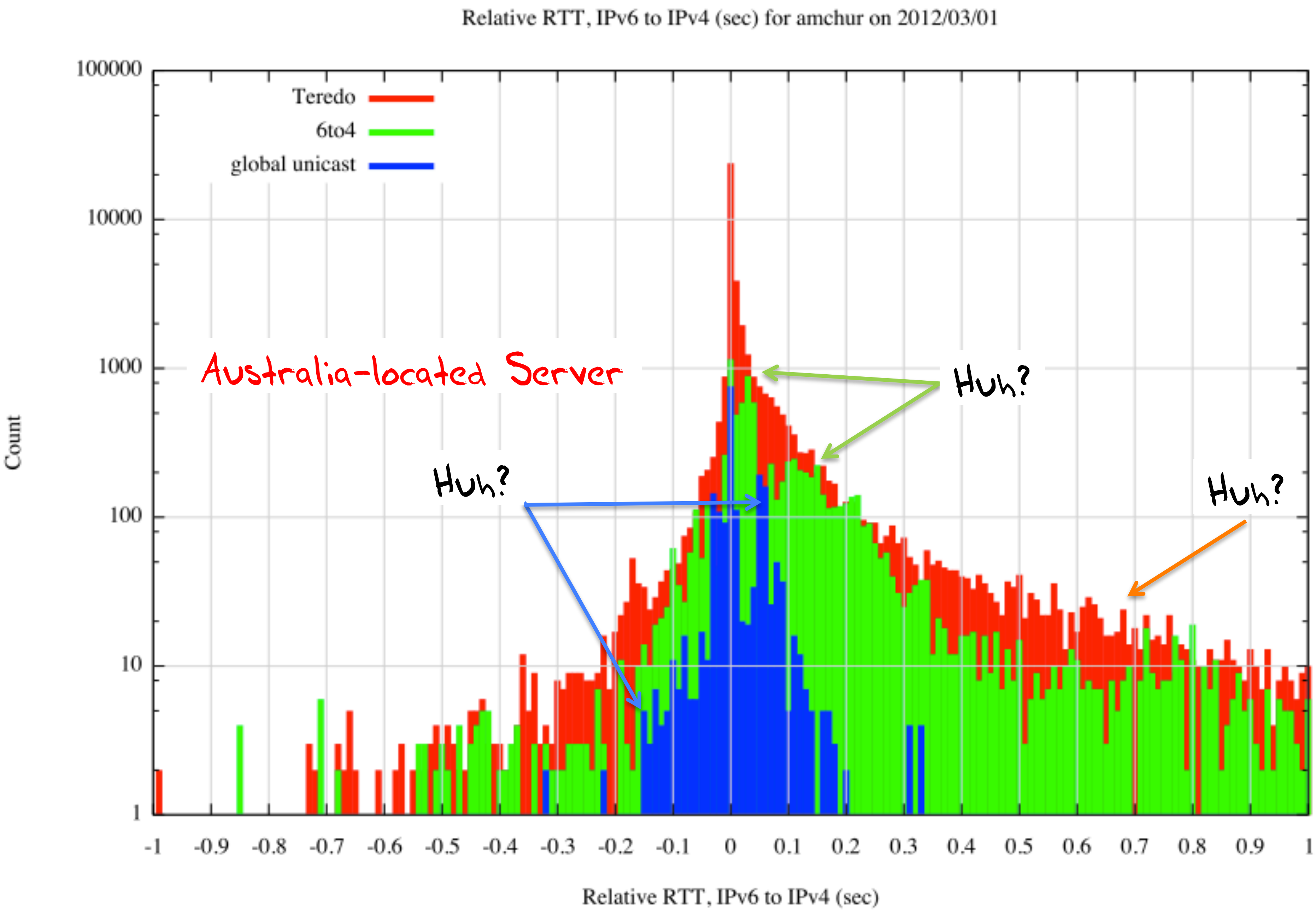


Europe-located Server

This is unexpected!

Why is V6 faster in some cases?

- We see some sessions that have faster V6 RTTs than their paired IPv4 counterpart
  - Because IPv6 is faster?
    - This is possible – there are some strange IPv4 paths out there
    - But why would a Teredo SYN exchange be faster than a native IPv4 SYN exchange?
  - Becuase IPv4 is slower?
    - Is this related to the behaviour characteristics of some CPE based NATs and their handling of NAT bindings during a a SYN exchange?

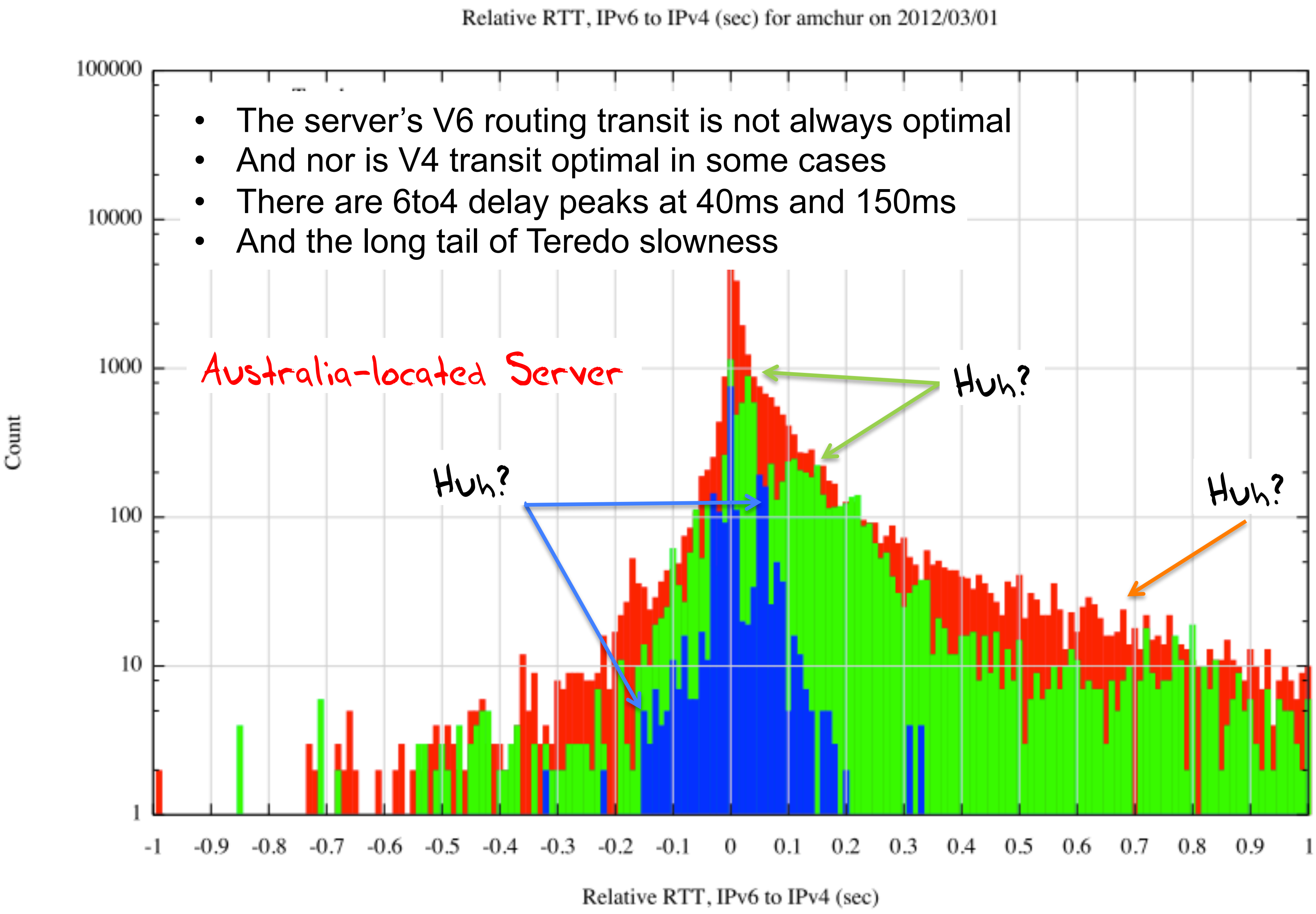


Australia-located Server

Huh?

Huh?

Huh?


Relative RTT, IPv6 to IPv4 (sec) for amchur on 2012/03/01

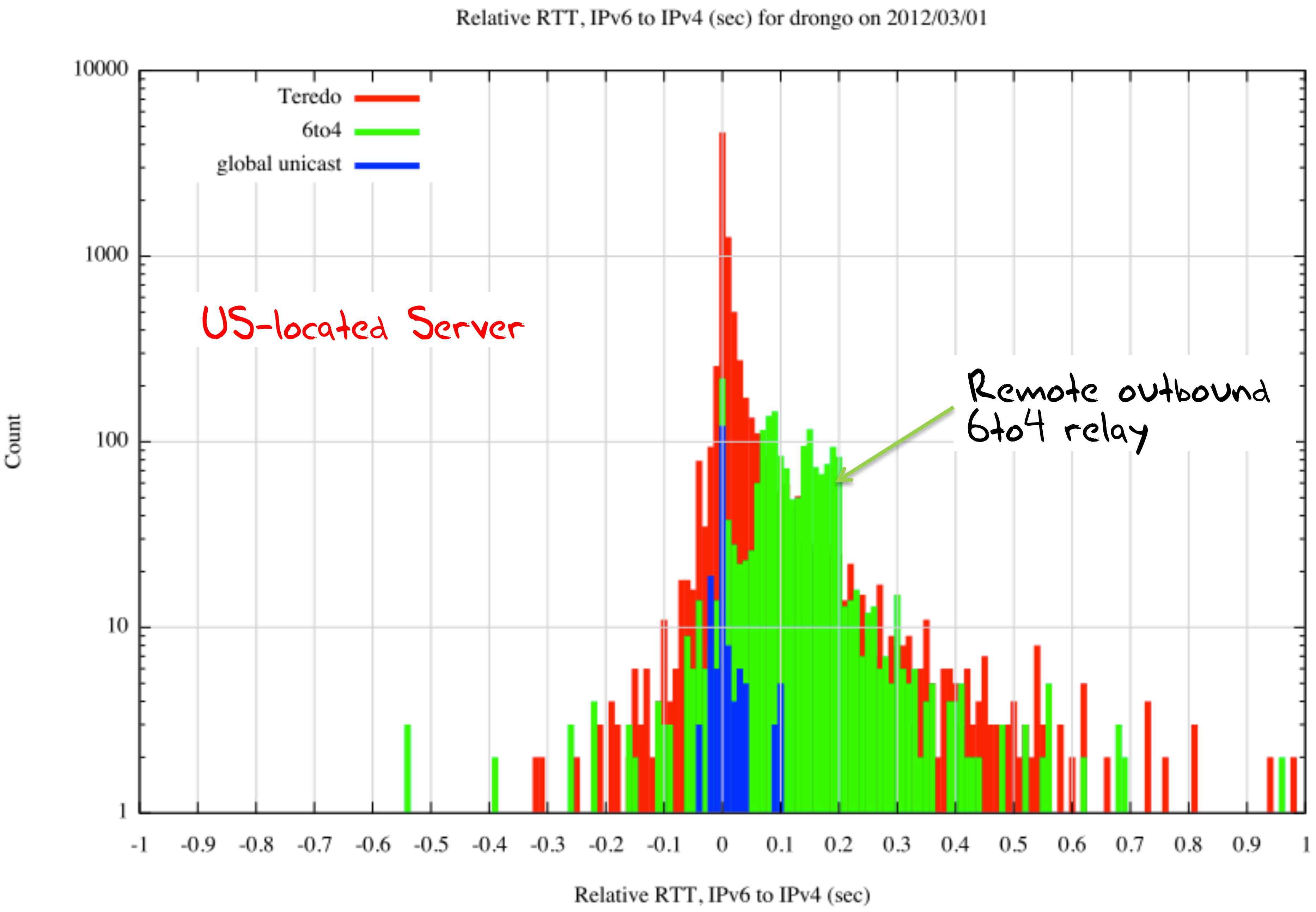
- The server's V6 routing transit is not always optimal
- And nor is V4 transit optimal in some cases
- There are 6to4 delay peaks at 40ms and 150ms
- And the long tail of Teredo slowness

Australia-located Server

Huh?

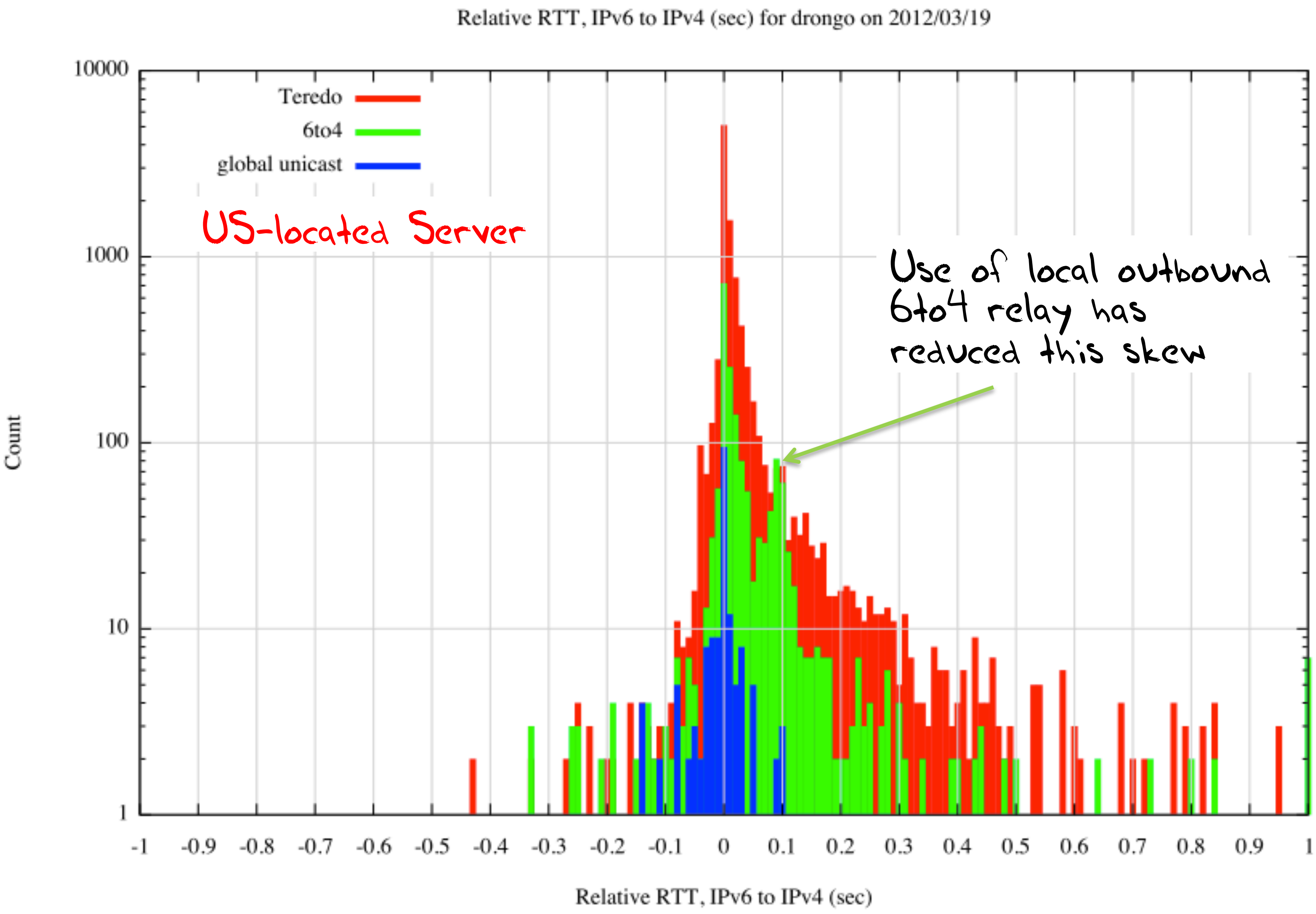
Huh?

Huh?



US-located Server

Remote outbound 6to4 relay



US-located Server

Use of local outbound 6to4 relay has reduced this skew

Observations

Is IPv6 as fast as IPv4?

- If you are native in IPv6, then, yes!
- The use of tunnels and overlays can make this worse in some cases, but, in general, V6 is as fast as V4

Observations

Is IPv6 as robust as IPv4?

- Sadly, on average, No
- The base failure rate of V6 connection attempts at ~2% of the total V6 unicast traffic volume is simply unacceptable as a service platform
- But its not in the core network. It appears that this is mainly self-inflicted with local edge firewall filter settings that trap V6 packets
- But relative robustness is highly variable – some ASes have IPv6 at the same level of robustness as IPv4, while others do no.

How Should Browsers Behave?

One view is to place both protocols on equal footing in a parallel connection environment, using a “SYN-ACK race” with parallel DNS and TCP session establishment

- E.g. Firefox with fast retransmit

Or reduce the server load by using a “DNS race” and take whichever answers first, but prepare for failover using a very aggressive timeout

- E.g. Chrome with 300ms failover timer

Or use local heuristics to estimate which is faster and failover within 1 RTT interval

- E.g. Safari + Mac OS X >= 10.7

How Should Browsers Behave?

One view is to place both protocols on equal footin
connection environment, using a "SYN-A
DNS and TCP session establis

– E.g. Firefox with f

and take
for failover using a very

with 300ms failover timer

None of these are that bad – they are all very fast. The trade off is slightly higher number of connections with the server against speed and robustness of the session

Or use local heuristics to estimate which is faster and failover within 1 RTT interval

– E.g. Safari + Mac OS X >= 10.7

But it’s still not enough!

Many access providers see their immediate future as having to
deploy IPv6 across their infrastructure, and at the same time field CGNs

But how big does the CGN need to be?

> Generically, the CGN needs to be as big as the residual preference for using IPv4 in dual stack scenarios

So how can we help this story along?

How Should Browsers Behave?

- Fire off the DNS queries in parallel
- If the DNS returns AAAA and A records, fire off a V6 connection attempt first
- Use a *reasonably aggressive* fallback timer to trigger V4 connection

    E.g. Chrome with 300ms failover timer

    E.g. Safari + Mac OS X with RTT-derived timer

How Should Browsers Behave?

- Fire off the DNS queries in parallel
- If the DNS returns AAAA and A records, fire off a V6 connecti attempt first
- Use a *reasonably* aggressive fail
  - E.g. Ch over timer
  - Mac OS X with RTT-derived timer

"B iased, but still Pleasantly Amused Eyeballs"!

Thank You

Questions?