“IPv4 vs. IPv6 – Happy Eyeballs”

Krzysztof Mazepa
IP NGN Systems Architect

EURONOG, Budapest 10-11 September 2012
Abstract

• Increased availability of IPv6 access causes that more and more people turn on IPv6 stack at their laptops and computers. It happens that IPv6 network performance is for some reason worse than one that towards to IPv4 site. A browser or operating system prefers particular address family regardless of network performance and the end site availability very often. That causes connection timeouts and results in an assumption that there is “something wrong” with IPv6.

• This session attempts to explain a mechanism that causes that behavior, it also promotes Happy Eyeballs algorithm that was implemented in particular operating systems …
Most of information included in this presentation comes from Dan Wing and Andrew Yourtchenko documents ...
Agenda

• Dual stack: the problem of address family selection
• The Happy Eyeballs solution
• Implementations of Happy Eyeballs
  Firefox and Chrome
  Apple OSX, iOS, Safari
• Windows 8 IPv6 NCSI
• IPv6 traffic impact
Dual Stack
Address selection problem
Transition Technologies in one Slide
Dual Stack: The Plan

• Dual Stack has been “the plan” for IPv6 migration since ... Forever

• The Plan:
  - Clients get IPv6 address (in addition to IPv4)
  - Servers get IPv6 address (in addition to IPv4)
  - Everyone runs two networks (IPv4 and IPv6)
  - IPv6 is preferred
Reality with Dual Stack

- Reality: IPv4 address might be shared with other hosts (NAT, MAP)
- Hosts prefer IPv6 over IPv4
  Generally necessary to get IPv6 on the network
  Without this preference, IPv4 would persist until IPv4 is turned off
- But what if IPv6 is broken? Overloaded???
  IPv6 peering is down ...
  Tunnel is down ...
  (Microsoft IPv6 NCSI is down.... More on that in a few slides)
IPv6 cannot be slower than IPv4

- Dual-stack client connecting to dual-stack server
- Dual-stack cannot be slower than IPv4
- If slower, users blame IPv6 and disable IPv6!

- IPv6 cannot be slower than IPv4
IPv6 Broken: Problem Description

1) Getaddrinfo(): hostname => address list

2) Try the addresses sequentially
Current Behaviour

<table>
<thead>
<tr>
<th>DNS Server</th>
<th>Client</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. &lt;--- <a href="http://www.example.com">www.example.com</a> A?-----</td>
<td>=TCP SYN, IPv6===&gt;X</td>
<td></td>
</tr>
<tr>
<td>2. &lt;--- <a href="http://www.example.com">www.example.com</a> AAAA?--</td>
<td>=TCP SYN, IPv6===&gt;X</td>
<td></td>
</tr>
<tr>
<td>3. ---192.0.2.1---------&gt;</td>
<td>=TCP SYN, IPv6===&gt;X</td>
<td></td>
</tr>
<tr>
<td>4. ---2001:db8::1---------&gt;</td>
<td>---TCP SYN, IPv4------&gt;</td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

20-30 second
Happy Eyeballs
The Happy Eyeballs Solution

Let the quickest win

A? Wait for the reply

Connect Success?

Get webpage

AAAA? Wait for the reply

Connect Success?
Optimizing Happy Eyeballs

The winner takes it all

Delay the slow one

Demote on failure
Happy Eyeballs with IPv6 broken

<table>
<thead>
<tr>
<th>DNS Server</th>
<th>Client</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>---www.example.com A?-----</td>
<td>300 milisecond</td>
</tr>
<tr>
<td>2.</td>
<td>---www.example.com AAAA?--</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>---192.0.2.1-----------------&gt;</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>---2001:db8::1-----------------&gt;</td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>==TCP SYN, IPv6===&gt;X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>--TCP SYN, IPv4--------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;-TCP SYN+ACK, IPv4-----</td>
<td></td>
</tr>
<tr>
<td></td>
<td>--TCP ACK, IPv4--------&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>==TCP SYN, IPv6===&gt;X</td>
<td></td>
</tr>
</tbody>
</table>
Happy Eyeballs

- Users are happy – fast response even if IPv6 (or IPv4) path is down
- Network administrators are happy
  - Users no longer trying to disable IPv6
  - Reduces IPv4 usage (reduces load on CGN)
- Content providers are happy
  - Improved geolocation and DoS visibility with IPv6

- RFC6555 „Happy Eyeballs: Success with Dual-Stack Hosts”
  (formerly draft-ietf-v6ops-happy-eyeballs)
  - By Dan Wing and Andrew Yourtchenko

Algorithm requirements, RFC6555 section 4

- Delay IPv4
  Avoids contenting for ipv4 resources with IPv4-only devices (see later slides)

- Stateful behaviour when IPv6 fails
  Don’t keep trying IPv6 all the time (optimization)

- Reset on network (re-initialization)

- Abandon non-winning connections
Implementations
Happy Eyeballs Coverage

- Web browsing is the most common application
- First, improve the web browsing experience
- Second, improve other applications
  Instant messaging, email client, etc.
Happy Eyeballs Implementations

- Google Chrome (in current stable channel)
- Mozilla Firefox (current version 13)
- Apple OSX, iOS, Safari

- Microsoft Windows 8
  Not Happy Eyeballs, but worth discussing
Chrome and Firefox Implementation

• Utilizes long-established 250-300ms ‘backup’ thread
  Originally just tried the next IP address
  Happy Eyeballs: tries the next IP address family

• Follows getaddrinfo() address preference
  IPv6 is usually preferred by the Operating System

• Result: IPv6 gets 250-300ms head start

https://bugzilla.mozilla.org/show_bug.cgi?id=621558
http://code.google.com/p/chromium/issues/detail?id=81686
Apple Implementation

- OS X 10.7, iOS 4.3, Safari
- Apple Framework, CFSocketStream
  - A and AAAA queried simultaneously
  - Attempt connection immediately
  - First to connect “wins”
  - Note: this is not a DNS race; it is a connection race
- “Legacy” applications calling getaddrinfo()
  - Addresses sorted based on previous connection success and connection failure
- Learns success/failure for each IPv6/IPv4 subnet
- Result: user connects to fastest of IPv6 or IPv4

OS X debugging details

- **10.7 (Lion)**
  - nettop –n –m route
  - nettop –n

- **10.8 (Mountain Lion)**
  - sudo defaults write /Library/Preferences/com.apple.networkd libnetcore_log_level –int 7
  - syslog –w
  - sudo defaults delete /Library/Preferences/com.apple.networkd libnetcore_log_level

- **tcpdump**
Windows 8

- IPv6 version of their Network Connectivity Status Indicator (NCSI)
  If works, IPv6 is enabled
  If fails, IPv6 is de-prioritized (specific IPv6 routes are prioritized)

- ipv6.msftncsi.com now influences IPv6 connectivity

- Internet Explorer 10 does not do Happy Eyeballs

Your task – check [ipv6.msftncsi.com](http://ipv6.msftncsi.com) availability 😊

Implementation Summary

- Apple OSX, iOS, Safari: prefers whichever is faster
- Firefox & Chrome: slight preference (200-300ms) for IPv6
- Windows 8: depends on IPv6 NCSI and IPv6 Route Advertisements
- Internet Explorer: whatever the OS decides
IPv6 traffic impact
 Reasons to prefer IPv6

<table>
<thead>
<tr>
<th>Before IPv4 address sharing</th>
<th>After IPv4 address sharing</th>
</tr>
</thead>
<tbody>
<tr>
<td>• If IPv4 remains preferred, we don’t learn of IPv6 problems until IPv4 is turned off</td>
<td>• Dual-stack hosts compete with IPv4-only devices</td>
</tr>
<tr>
<td>• Thus, we can never turn off IPv4</td>
<td>• Important: problem exists for all IPv4 address sharing – MAP, A+P, SD-NAT, NAT44, CGN</td>
</tr>
<tr>
<td>• IPv4 is the ‘fallback’ while IPv6 achieves operational parity with IPv4</td>
<td>• Over time, do less IPv4 address sharing</td>
</tr>
</tbody>
</table>
Dual-Stack Competition with IPv4-only

- Dual stack host
- IPv4-only host
- IPv4-only host

Competition

Address sharing device(s)
(MAP, SD-NAT, NAT)

IPv6 Internet
IPv4 Internet
IPv6 Traffic growth

• Claim: Happy Eyeballs harms IPv6 traffic growth

• Reality:
  Windows 8, Firefox, and Chrome all help IPv6 traffic growth (250-300ms delay)
  That is a huge percentage of the market

Apple's implementation:
  If IPv6 is faster, traffic will be on IPv6
  If equal speed, half of Apple-sourced traffic will be IPv6
  If IPv4 is faster, improve your IPv6
  Apple philosophy: user experience is paramount
What Happy Eyeballs says

“Thus, to avoid harming IPv4-only hosts which can only utilize IPv4, implementations MUST prefer the first IP address family returned by the host's address preference policy, unless implementing a stateful algorithm described in Section 4.2. This usually means giving preference to IPv6 over IPv4, although that preference can be overridden by user configuration or by network configuration. If the host's policy is unknown or not attainable, implementations MUST prefer IPv6 over IPv4.”

RFC6555, Happy Eyeballs
An operational issue
Happy Eyeballs Summary

- Implementations are different
  - Slight delay to users with broken IPv6 path: Firefox, Chrome
  - No delay to users with broken IPv6 path: Apple
  - No delay to users after testing IPv6 path to cloud: Microsoft

- Impact on IPv6 traffic patterns depends on client implementation