Ruby trunk - Bug #15773

Net::HTTP doesn't try next IP address in case of timeout

04/16/2019 11:12 PM - nicolasnoble (Nicolas Noble)

Status:	Rejected		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:		**Backport:**	2.4: UNKNOWN, 2.5: UNKNOWN, 2.6: UNKNOWN

Description

This example requires to have a working IPv6 address. Since IPv6 is used in first priority, I am using it to demonstrate the problem, but it exists with plain IPv4, which will be more round-robin-style, so less deterministic to show a reproduction case. I have made two URLs that have both IPv4 and IPv6 addresses: http://ipv6.grumpycoder.net/ and http://bad-ipv6.grumpycoder.net/ - both URLs should work in an IPv6-enabled web browser, as well as curl or wget for instance. The difference is that the bad-ipv6 subdomain doesn't have an IPv6 that will actually connect. Therefore, browsers, curl and wget will fallback to using the IPv4 when the initial IPv6 connection attempt failed:

```
$ wget -T1 http://bad-ipv6.grumpycoder.net -O - # demonstrating using wget because its output is m
ore clear than curl's verbose
--2019-04-16 15:56:52--  http://bad-ipv6.grumpycoder.net/
Resolving bad-ipv6.grumpycoder.net (bad-ipv6.grumpycoder.net)... 2001:bc8:3690:200::2, 62.210.214.
144
Connecting to bad-ipv6.grumpycoder.net (bad-ipv6.grumpycoder.net)|2001:bc8:3690:200::2|:80... fail
ed: Connection timed out.
Connecting to bad-ipv6.grumpycoder.net (bad-ipv6.grumpycoder.net)|62.210.214.144|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 45 [text/html]
Saving to: ‘STDOUT’

-                                     0%[
              ]       0  --.-KB/s               <html><body><h1>It works!</h1></body></html>
-                                   100%[=========================================================
=============>]      45  --.-KB/s    in 0s

2019-04-16 15:56:53 (6.55 MB/s) - written to stdout [45/45]
```

However, in Ruby, using Net::HTTP (or OpenURI), that's not going to be the case:

```
$ ruby bad-ipv6.rb
http://ipv6.grumpycoder.net
<html><body><h1>It works!</h1></body></html>
http://bad-ipv6.grumpycoder.net
Net::OpenTimeout: execution expired
```

Contents of my test file:

```
require 'open-uri'

['ipv6', 'bad-ipv6'].each do |s|
  url = 'http://%s.grumpycoder.net' %[s]
  puts url
  begin
    puts open(url).read
  rescue StandardError => e
    puts "#{e.class}: #{e.message}"
    next
  end
end
```

The proper behavior should be to retry the next IP address and exhaust all of the IPs in the DNS resolution results before throwing out an error.

History

#1 - 04/19/2019 01:32 AM - naruse (Yui NARUSE)

- Status changed from Open to Rejected

In general resolving DNS is done by libc (getaddrinfo) and Ruby just uses their result.
Therefore it is not a bug and won't be a small feature.

#2 - 04/24/2019 03:31 PM - nicolasnoble (Nicolas Noble)

I thought long and hard about how to reply to this.

Let's put it this way: the fact that you are using glibc isn't the problem at all. The fact that you are badly using its results is.

Any other big piece of code out there that is doing any sort of abstraction is going to be establishing an internet connection will retry on all the entries returned by the DNS, in a round robin fashion. Ruby is the only one that I know of that doesn't do that. In fact, if you run my reproduction case on jruby, it will work correctly. If anything, the bug should be that jruby and ruby are acting differently.

It is important to realize that when one is attempting to write any abstraction layer, there's an expectation of correctness. In this one case, the current chosen model of abstraction means that the http downloader system will catastrophically fail to work in certain conditions without any sort of recourse by the user of the abstraction.