Use socket connect_timeout in net stdlib for open_timeout

11/13/2016 10:32 AM - xiewenwei (xie wenwei)

Status: Open
Priority: Normal
Assignee: 
Target version: 

Description
Current net/http and net/pop use Timeout.timeout to tigger open_timeout event. Timeout.timeout is slow. It will create and destroy a thread every time. Timeout.timeout is also dangerous. see [[http://www.mikeperham.com/2015/05/08/timeout-rubys-most-dangerous-api/]]

It is more effective and safe to use socket timeout to accomplish this. Follow is the changes need to do.

1. Replace TCPSocket.open with Socket.new
2. Use socket.connect_nonblock and IO.select to connect and trigger timeout event.

The pull request is here: [[https://github.com/ruby/ruby/pull/1480]]

Related issues:
Related to Ruby master - Feature #12435: Using connect_nonblock to open TCP c... Open

History

#1 - 11/14/2016 05:20 PM - xiewenwei (xie wenwei)
The codes are updated. I use Socket.tcp now. Socket.tcp returns Socket instance. So I need to convert it to TCPSocket instance using TCPSocket.for_fd.

#2 - 11/14/2016 08:41 PM - normalperson (Eric Wong)

xiewenwei@gmail.com wrote:

  net/http, net/pop, net/smtp and net/ftp use Timeout.timeout to calculate connect_timeout. Timeout.timeout is slow. It creates and destroys a thread every time. Timeout.timeout is also dangerous. see Timeout: Ruby's Most Dangerous API

I agree with eliminating Timeout, but I don't think your solution is enough because it does not cover timeouts for DNS resolution (getaddrinfo(3) calls).

For timeouts, we would need to use resolv.rb instead of getaddrinfo(3) provided by libc to do timeouts without a separate thread. I started adding timeouts to resolv.rb last year but can't remember how far I got... I'm not sure if resolv.rb supports all the features of a modern getaddrinfo(3), either, AFAIK, not many people use resolv.rb.

  It is more effective and safe to use socket timeout to accomplish that. Follow is the changes need to do.
  1. Replace TCPSocket.open with Socket.tcp
  2. Create TCPSocket with TCPSocket.for_fd

I don't think this step should be necessary; Socket and TCPSocket should be usable interchangeably for stream sockets (maybe some API calls need to be changed, but I'd rather avoid the extra object entirely).

#3 - 11/15/2016 05:56 PM - xiewenwei (xie wenwei)

- Subject changed from Use socket timeout for net/http and net/pop for open_timeout to Use socket conect_timeout for net stdlib for open_timeout

I changed the codes. Removed TCPSocket.for_fd and used socket directly now.
But I am no idea for DNS resolution timeout. How to fix it?

#4 - 11/15/2016 06:04 PM - xiewenwei (xie wenwei)
- Subject changed from Use socket conec_timeout for net stdlib for open_timeout to Use socket conec_timeout in net stdlib for open_timeout

#5 - 11/16/2016 07:07 AM - shugo (Shugo Maeda)
- Related to Feature #12435: Using connect_nonblock to open TCP connections in Net::HTTP#connect added

#6 - 11/16/2016 04:54 PM - xiewenwei (xie wenwei)
I got it. It is not simple to fix as I expected before. Thank you.

#7 - 12/02/2019 09:20 PM - dylants (Dylan Thacker-Smith)
It looks like we can now use the resolv_timeout Socket.tcp option that was recently added by https://bugs.ruby-lang.org/projects/ruby-trunk/repository/git/revisions/0e9d56f5e73ed2fd8e7c858fdea7b7d5b905bb64.

The only complication with trying to use Socket.tcp now would be that it has two separate timeouts that are specified at the same time, but the net stdlib uses a combine open timeout that include DNS resolving and TCP connecting. Preserving that behaviour would mean using the open timeout for DNS resolution, then using the remaining time for the connect timeout. Doing this outside Socket.tcp would mean duplicating a lot of the code in that method.

Should we add a combined timeout option to Socket.tcp so we can easily use that in the net stdlib's open timeouts?