

Ruby - Feature #21347

Add `open_timeout` as an overall timeout option for `Socket.tcp`

05/17/2025 07:27 AM - shioimm (Misaki Shioi)

<div>Status:Assigned</div> <div>Priority:Normal</div> <div>Assignee:shioimm (Misaki Shioi)</div> <div>Target version:</div>	
<div>Description</div> <div>I propose to add an overall timeout option to Socket.tcp (and TCPSocket.new)</div> <div>Background</div> <div>Currently, TCPSocket.new and Socket.tcp accept two kind of timeout options:</div> <div><ul style="list-style-type: none">• resolv_timeout, which controls the timeout for DNS resolution• connect_timeout, which controls the timeout for the connection attempt</div> <div>With the introduction of Happy Eyeballs Version 2 (as per RFC 8305) in Feature #20108 and Feature #20782, both address resolution and connection attempts are now parallelized.</div> <div>As a result, the sum of resolv_timeout and connect_timeout no longer represents the total timeout duration. This is because, in HEv2, name resolution and connection attempts are performed concurrently, causing the two timeouts to overlap.</div> <div>Example:</div> <div>When resolv_timeout: 200ms and connect_timeout: 100ms are set:</div> <div><ol style="list-style-type: none">1. An IPv6 address is resolved after the method starts immediately (IPv4 is still being resolved).2. A connection attempt is initiated to the IPv6 address3. After 100ms, connect_timeout is exceeded. However, since resolv_timeout still has 100ms left, the IPv4 resolution continues.4. After 200ms from the start, the method raises a resolv_timeout error.</div> <div>In this case, the total elapsed time before a timeout is 200ms, not the expected 300ms (100ms + 200ms).</div> <div>Furthermore, in HEv2, connection attempts are also parallelized.</div> <div>It starts a new connection attempts every 250ms for resolved addresses. This makes the definition of connect_timeout even more ambiguous—specifically, it becomes unclear from which point the timeout is counted.</div> <div>Additionally, these methods initiate new connection attempts every 250ms (Connection Attempt Delay) for each candidate address, thereby parallelizing connection attempts. However, this behavior makes it unclear from which point in time the connect_timeout is actually measured.</div> <div>Currently, a connect_timeout is raised only after the last connection attempt exceeds the timeout.</div> <div>Example:</div> <div>When connect_timeout: 100ms is set and 3 address candidates:</div> <div><ol style="list-style-type: none">1. Connect to address a at t=0ms.2. Connect to address b at t=250ms.3. Connect to address c at t=500ms.4. If all fail, the exception is raised at t=1500ms (1000ms after c, 1250ms after b, 1500ms after a).5. Start a connection attempt to the address a6. 250ms after step 1, start a new connection attempt to the address b7. 500ms after step 1, start a new connection attempt to the address c8. 1000ms after step 3 (1000ms after starting the connection to c, 1250ms after starting the connection to b, and 1500ms after starting the connection to a) connect_timeout is raised</div>	

This behavior aims to favor successful connections by allowing more time for each attempt, but it results in a timeout model that is difficult to reason about.

These methods have supported `resolv_timeout` and `connect_timeout` options even before the introduction of HEv2. However, in many use cases, it would be more convenient if a timeout occurred after a specified duration from the start of the method. Similar functions in other languages (such as PHP, Python, and Go) typically allow specifying only an overall timeout.

Proposal

I propose adding an `open_timeout` option to `Socket.tcp` in the following PR, which triggers a timeout after a specified duration has elapsed from the start of the method.

<https://github.com/ruby/ruby/pull/13368>

The name `open_timeout` aligns with the existing accessor used in `Net::HTTP`.

If `open_timeout` is specified together with `resolv_timeout` and `connect_timeout`, I propose that only `open_timeout` be used and the others be ignored. While it is possible to support combinations of `open_timeout`, `resolv_timeout`, and `connect_timeout`, doing so would require defining which timeout takes precedence in which situations. In this case, I believe it is more valuable to keep the behavior simple and easy to understand, rather than supporting more complex use cases.

If this proposal is accepted, I also plan to extend `open_timeout` support to `TCP Socket.new`.

While the long-term future of `resolv_timeout` and `connect_timeout` may warrant further discussion, I believe the immediate priority is to offer a straightforward way to specify an overall timeout.

Outcome

If `open_timeout` is also supported by `TCP Socket.new`, users would be able to manage total connection timeouts directly in `Net::HTTP#connect` without relying on `Timeout.timeout`.

<https://github.com/ruby/ruby/blob/aa0f689bf45352c4a592e7f1a044912c40435266/lib/net/http.rb#L1657>

History

#1 - 05/17/2025 07:37 AM - shioimm (Misaki Shioi)

Sorry, the example in the sentence "Currently, a `connect_timeout` is raised only after the last connection attempt exceeds the timeout." is incorrect. The correct version is as follows:

Example:

When `connect_timeout: 100ms` is set and 3 address candidates:

1. Start a connection attempt to the address a
2. 250ms after step 1, start a new connection attempt to the address b
3. 500ms after step 1, start a new connection attempt to the address c
4. 1000ms after step 3 (1000ms after starting the connection to c, 1250ms after starting the connection to b, and 1500ms after starting the connection to a) `connect_timeout` is raised

#2 - 05/17/2025 11:18 AM - osyoyu (Daisuke Aritomo)

I am +1 to this feature. As a `Socket.tcp` / `TCP Socket` user, I am usually concerned about the time required to open the connection as a whole, rather than name resolution and `connect(2)` as separate parts. I also like that the mental model aligns with implementations in other languages (`net.DialTimeout` in Go to name one).

In `stdlib`, not only `net/http` but `net/ftp`, `net/pop` and `net/smtp` will all benefit from an `open_timeout` option in `TCP Socket`. They all provide only `open_timeout`, and do not support separate timeouts for DNS resolution and connection.

`Timeout.timeout` is used for implementing timeouts, which is not ideal in terms of resource usage. (See <https://github.com/ruby/net-http/issues/6> for more context)

Just out of curiosity: Is the example given in <https://bugs.ruby-lang.org/issues/21347#note-1> a case when `connect_timeout` is 1000 ms (not 100 ms) ?

#3 - 05/17/2025 02:21 PM - shioimm (Misaki Shioi)

@osyoyu

Just out of curiosity: Is the example given in <https://bugs.ruby-lang.org/issues/21347#note-1> a case when `connect_timeout` is 1000 ms (not 100 ms) ?

It's just a typo. `connect_timeout: 1000ms` is right, sorry.

#4 - 06/05/2025 04:52 AM - matz (Yukihiro Matsumoto)

Accepted. Go ahead.

Matz.

#5 - 06/05/2025 09:27 AM - shioimm (Misaki Shioi)

- *Status changed from Open to Assigned*

- *Assignee set to shioimm (Misaki Shioi)*