2020 年度 Ruby アソシエーション開発助成
最終報告書

ruby の標準ライブラリ Socket への
Happy Eyeballs Version 2 (RFC8305) の導入

松下　正樹

2021 年 3 月 15 日

1　プロジェクトの概要

ruby の標準ライブラリ Socket には、ruby を実行するホストが IPv6 アドレスを持つにも関わらず接続先と IPv6 での通信を行えない場合、接続を試みたまま数十秒程度プログラムが停止してしまう問題がある[*1]。この問題により、意図せずプログラムが停止してしまう場合があるほか、RubyGems でのパッケージ取得にも支障が生じる場合があることが報告されている[*2]。 本プロジェクトでは、ruby の標準ライブラリ Socket に Happy Eyeballs Version 2 (RFC8305) で規定される接続試行アルゴリズムを導入することにより、IPv6 での通信が確立しない場合は速やかに IPv4 へフォールバックさせることで上記の問題の解決を目指す。

2　本プロジェクトで解決を目指す課題

2.1　名前解決

Socket では getaddrinfo 関数を利用して DNS による名前解決を行っている。getaddrinfo 関数では処理のタイムアウトを指定することができないため、名前解決を試みた際に DNS サーバーから応答が返ってこないと数十秒〜数分間処理が停止してしまう。ま

[*1] https://bugs.ruby-lang.org/issues/15628
[*2] https://github.com/rubygems/rubygems/pull/2662

た、getaddrinfo 関数のような C レベルで停止してしまう処理は Timeout などで中断させることもできない。

2.2 接続試行

Socket では、名前解決された IP アドレスの一覧に対して順に connect 関数を呼び出して接続試行を行う。IP アドレスの一覧は Default Address Selection for Internet Protocol Version 6 (RFC6724) に従いソートされており、IPv6 アドレスへの接続試行が先に行われる。ruby を実行するホストが IPv6 アドレスを持つにも関わらず接続先と IPv6 での通信を行えない場合、接続を確立できない IPv6 アドレスに対する接続試行がタイムアウトするまで IPv4 アドレスに対する接続を試行できず、IPv4 で接続を確立できるまで数十秒程度待たされてしまう。

3 本プロジェクトの経過

3.1 実装方針の検討

Socket において TCP ソケットを生成する方法は C で実装された TCPSocket.new と Ruby で実装された Socket.tcp の 2 種類がある。メンターの田中哲氏と相談した結果、Ruby で実装されているため改変が容易で Thread も利用可能な Socket.tcp から実装を進めることとした。

3.2 Socket.tcp への Happy Eyeballs Version 2 の実装

中間報告までに IPv6 と IPv4 に対する getaddrinfo 関数の呼び出しを異なる Thread から行い、結果を Queue で返すことで非同期に名前解決を行う実装を作成し、ruby 開発コミュニティに提案した。しかし、メンターの田中哲氏によるレビューにおいて、この実装では名前解決を待っている間に接続が確立した場合に無駄な待ち時間が発生することが判明した。例えば、IPv6 の名前解決が終わり接続を試行したあと、IPv4 の名前解決を待っている間に IPv6 の接続が確立した場合、IPv4 の名前解決が終わるまで IPv6 の接続を利用できない実装となっていた。

このため、名前解決の終了を Queue ではなく pipe で通知する実装へと変更した。pipe による通知は select 関数で待つことが可能であり、接続の確立と名前解決の終了を同時に待つことが可能である。これにより、名前解決を待っている間に他の接続が確立した場合

でも無駄な待ち時間が生じることなく処理を行えるようになった。

3.3 TCPSocket.new への Happy Eyeballs Version 2 の実装

C 実装である TCPSocket.new については、中間報告までに Happy Eyeballs Version 2 を実装することができなかった。大きな要因として、非同期の名前解決 API である getaddrinfo_a 関数を ruby 3.0 に導入できなかったことが挙げられる。筆者はかねてから本プロジェクトとは別に ruby への getaddrinfo_a 関数の導入に取り組んでおり、実装をマージしていた[*3]。しかし、fork 関数と組み合わせて呼び出した場合に不具合を生じることが ruby 3.0 のリリース直前に判明した[*4]ため、やむを得ずこの実装を revert することとなった。Happy Eyeballs Version 2 では名前解決を非同期で行うべき (SHOULD) と規定されており、TCPSocket.new では getaddrinfo_a 関数を利用して名前解決を非同期で行うことを想定していたが、方針の変更を迫られることとなった。このため、Happy Eyeballs Version 2 のうち接続試行を行う部分のみを実装することとした。

4 本プロジェクトの成果

本プロジェクトでは github.com 上の ruby 開発リポジトリに 4 件の機能追加を pull request として提案し、成果物を公開した。

- Socket.tcp への Happy Eyeballs Version 2 の実装[*5]
- SocketError にエラーコードを保持させる機能追加[*6]
- TCPSocket.new への Happy Eyeballs Version 2 の一部実装[*7]
- 中断可能な rb_getaddrinfo の実装[*8]

それぞれの機能追加の詳細は下記の通りである。

[*3] https://github.com/ruby/ruby/commit/0e9d56f5e73ed2fd8e7c858fdea7b7d5b905bb64
[*4] https://bugs.ruby-lang.org/issues/17220
[*5] https://github.com/ruby/ruby/pull/4038
[*6] https://github.com/ruby/ruby/pull/4247
[*7] https://github.com/ruby/ruby/pull/4262
[*8] https://github.com/ruby/ruby/pull/4268

4.1 Socket.tcp への Happy Eyeballs Version 2 の実装

Socket.tcp に Happy Eyeballs Version 2 を実装する patch を作成した。本実装では、IPv6 と IPv4 に対する getaddrinfo 関数の呼び出しを異なる Thread から行い、呼び出しの終了を pipe で通知することで非同期に名前解決を行う。これにより、指定されたタイムアウトを超過した場合は getaddrinfo 関数の呼び出しで停止している Thread を終了させることができ[*9]、2.1 に示した問題を回避することができる。また、本実装では解決されたアドレスに対してそれぞれ 250 ミリ秒（RFC8305 の推奨値）の停止を挟みつつノンブロックで connect 関数を呼び出して複数の接続を同時に試行し、最も早く接続の確立した Socket を返す。これにより、ある IP アドレスへの接続を速やかに確立できなかった場合は他のアドレスへの接続試行に移り、2.2 に示した問題を回避して最も早く接続の確立した Socket をユーザに返却することができる。

4.2 SocketError にエラーコードを保持させる機能追加

これは 4.1 の実装に必要な機能追加である。4.1 の実装では、エラー発生時に発生したエラーの種類に応じて異なる処理を行う必要がある。しかし、既存の SocketError ではエラーの種類を識別するためのエラーコードを取得する方法がなく、発生したエラーの種類を知るにはエラーメッセージを調べるしか方法がない。エラーメッセージは変更される可能性があり、エラーの種類を識別する上で信頼できる情報とはいえない。そこで、SocketError にエラーコードを保持・取得できる機能の追加を提案した。

4.3 TCPSocket.new への Happy Eyeballs Version 2 の一部実装

TCPSocket.new に Happy Eyeballs Version 2 の一部を実装する機能追加を行った。ここでは、Happy Eyeballs Version 2 のうち Connection Attempts[*10]を実装している。4.1 と同様にノンブロックで connect 関数を呼び出して複数の接続を同時に試行し、2.2 に示した問題を回避して最も早く確立した接続から TCPSocket を生成する。

[*9] 現在の ruby は Thread の中断を行うことができないため、Thread を直ちに終了させることはできない。Thread は getaddrinfo 関数から処理が戻った後に終了する。

[*10] https://tools.ietf.org/html/rfc8305#section-5

4.4 中断可能な rb_getaddrinfo 関数の実装

これは、3.3 で導入を断念した getaddrinfo_a 関数の代替として Thread と Queue を用いた中断可能な rb_getaddrinfo 関数を実装したものである。rb_getaddrinfo 関数は、Socket 内で getaddrinfo 関数を呼び出すために用意されている関数である。Addrinfo.getaddrinfo メソッドがこの関数により実装されているほか、TCPSocket.new や Socket.tcp もこの関数を利用して名前解決を行っている。本実装では、getaddrinfo 関数をスレッドプールを介して呼び出すことで、rb_getaddrinfo 関数を中断させることが可能である。これにより、2.1 に示した問題を C レベルでも回避することができる。

5 今後の課題

本プロジェクトの成果物を ruby にマージすることが今後の課題である。Socket.tcp への Happy Eyeballs Version 2 の導入については、pipe を利用することによる制約から一部のプラットフォームでテストに失敗する問題があり、これを解決する必要がある。TCPSocket.new への Happy Eyeballs Version 2 については段階を踏む必要があると考えており、まず 4.3 で提案した Happy Eyeballs Version 2 の一部の実装をマージし、それから 4.4 の中断可能な getaddrinfo 実装を使って Happy Eyeballs Version 2 の完全な実装を目指すこととしたい。

6 謝辞

本プロジェクトに取り組むにあたり、メンターの田中哲氏には実装方針の検討や pull request のレビューなど、様々な協力をいただいた。この場を借りて深く感謝申し上げる。