Ruby master - Feature #16559

Net::HTTP#request injects "Connection: close" header if #started? is false, wasting HTTP server resources

01/24/2020 03:50 AM - f3ndot (Justin Bull)

Status:	Open
Priority:	Normal
Assignee:	naruse (Yui NARUSE)
Target version:	

Description

Hello,

There appears to be a bug in Net::HTTP#request (and thus #get, #post, etc.) on an instance that isn't explicitly started by the programmer (by invoking #start first, or by executing #request inside a block passed to #start).

Inspecting the source code, it reveals #request will recursively call itself inside a #start block if #started? is false. This is great and as I'd expect.

However in production and in a test setup I'm observing TCP socket connections on the server-side in the "TIME_WAIT" state, indicating the socket was never properly closed. Conversely, explicitly running #request inside a #start block yields no such behaviour.

Consider the following setup, assuming you have docker:

```
docker run --rm -it -p 8080:80/tcp --user root ubuntu
apt-get update && apt-get install net-tools watch nginx
service nginx start
watch 'netstat -tunapl'
```

Running this on your host machine:

```
net = Net::HTTP.new('localhost', 8080)
50.times { net.get('/') } # is bad
```

Will spawn 50 TCP connections on the server, and will all have on TIME_WAIT for 60 seconds (different *nix OSes have different times):

```
Every 2.0s: netstat -tunapl

Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      791/nginx: master
p
tcp        0      0 172.17.0.2:80           172.17.0.1:60772        TIME_WAIT   -
tcp        0      0 172.17.0.2:80           172.17.0.1:60732        TIME_WAIT   -
tcp        0      0 172.17.0.2:80           172.17.0.1:60812        TIME_WAIT   -
tcp        0      0 172.17.0.2:80           172.17.0.1:60778        TIME_WAIT   -
...
```

However running any of these incantations have no such result:

```
50.times { Net::HTTP.get(URI('http://localhost:8080/')) } # is OK

net = Net::HTTP.new('localhost', 8080)
net.start
50.times { net.get('/') } # is OK
net.finish

net = Net::HTTP.new('localhost', 8080)
50.times { net.start { net.get('/') } } # is OK
```

These TIME_WAIT connections matter because a server receiving many HTTP requests from clients using Net::HTTP in this fashion

(as Faraday does[1]) the server will begin to oversaturate and timeout past a particular scale.

I've tested and reproduced this in 2.7 and 2.6.

History

#1 - 01/24/2020 03:20 PM - f3ndot (Justin Bull)

- ruby -v changed from 2.7.0-preview to 2.8.0-dev, 2.7.0, 2.6.5

- File dont-default-connection-close.patch added

I have chased this down to an opportunistic setting of 'Connection: close' header if-and-only-if #request is called when #started? is false.

Attached is a patchfile where I remove this header setting, with the rationale laid out in the git commit body. Posted below for convenience. Hopefully you agree :-)

```
Don't set 'Connection: close' by default if #started? is false

This causes divergent and unexpected behaviour from other ways of using of
Net::HTTP, and sending requests with 'Connection: close' causes a drain on
server resources. Consumers of this Ruby class may unintentionally invoke it in
such a way where they intend to send many many requests to a server, but
unfortunately all of them closing the connection.

For example, this causes 50 TIME_WAIT tcp connections on the server because the
'Connection: close' header was set every time:

```
net = Net::HTTP.new('example.com')
50.times { net.get('/') } # causes many TIME_WAIT TCP sockets server-side
```

Meanwhile neither of these invocations have this behaviour, which (for HTTP
servers >= 1.1) benefit from the default behaviour of a keep-alive/persistent
TCP connection:

```
50.times { Net::HTTP.get(URI('http://example.com/')) } # no TIME_WAIT sockets

net = Net::HTTP.new('example.com')
50.times { net.start { net.get('/') } } # no TIME_WAIT sockets

net = Net::HTTP.new('example.com')
net.start
50.times { net.get('/') } # no TIME_WAIT sockets
net.finish
```

Indeed using cURL, which one can assume represents one-off HTTP requests, does
not set the header either way-- defering to the server which is keep-alive/
persistence on 1.1 and greater.
```

#2 - 03/30/2020 01:47 PM - f3ndot (Justin Bull)

- Subject changed from Net::HTTP#request does not properly close TCP socket if #started? is false to Net::HTTP#request injects "Connection: close" header if #started? is false, wasting HTTP server resources

#3 - 05/28/2020 07:37 PM - jeremyevans0 (Jeremy Evans)

- Backport deleted (2.5: UNKNOWN, 2.6: UNKNOWN, 2.7: UNKNOWN)

- ruby -v deleted (2.8.0-dev, 2.7.0, 2.6.5)

- Assignee set to naruse (Yui NARUSE)

- Tracker changed from Bug to Feature

I don't think this is a bug fix, this just makes a different tradeoff. This can help cases where the computer you are connecting to is overloaded by TIME_WAIT sockets. However, by the same token, it causes TIME_WAIT sockets to accumulate on the computer you are connecting from.

Consider a situation where you have a program that needs to get the root page for hundreds of thousands of websites. It connects to a hundred web servers concurrently:

```
100.times.map do
  Thread.new do
```

```
    loop do
      break unless domain_name = input_queue.pop
      net = Net::HTTP.new(domain_name)
      output_queue.push net.get('/')
    end
  end
end.each(&:join)
```

This approach works fine currently (assuming you add the necessary error handling), because connection: close will be set and the server will close the connection, so the client socket will not end up in TIME_WAIT. With your change, connection: close will not be set, the client will close the connection, and all client sockets will end up in TIME_WAIT until 2MSL expires.

I'm not against the patch, as it would make things more consistent, and the backwards compatibility issues are small. You can always set the connection header manually if you want specific behavior. The net/http maintainer will have to decide if the change is worth it.

Files

dont-default-connection-close.patch	3.77 KB	01/24/2020	f3ndot (Justin Bull)