Ruby master - Bug #14972

Net::HTTP inconsistently raises EOFError when peer closes the connection

08/07/2018 07:20 AM - joshc (Josh C)

Status:	Open		
Priority:	Normal		
Assignee:			
Target version:	2.7		
ruby -v:	ruby 2.4.3p205 (2017-12-14 revision 61247) [x86_64-darwin15]	**Backport:**	2.3: DONTNEED, 2.4: DONTNEED, 2.5: DONTNEED

Description

If chunked transfer encoding is used, and the peer closes the connection while the caller is reading data, then the Net::HTTP::Response#read_body method will raise EOFError. If chunked transfer encoding is not used (and an explicit Content-Length is used instead), the read_body method swallows the EOFError exception. I would expect read_body to raise EOFError if it reads fewer than Content-Length bytes.

The current behavior is explained by the ignore_eof parameter in https://github.com/ruby/ruby/blob/v2_4_3/lib/net/http/response.rb#L284-L301. However, RFC 7230 section 3.3.3 https://tools.ietf.org/html/rfc7230#section-3.3.3 says:

```
   5.  If a valid Content-Length header field is present without
       Transfer-Encoding, its decimal value defines the expected message
       body length in octets.  If the sender closes the connection or
       the recipient times out before the indicated number of octets are
       received, the recipient MUST consider the message to be
       incomplete and close the connection.
```

As it is now, if chunked encoding is not used, then the caller is unaware when the response body is truncated. In order to detect it, the caller must count the number of bytes read until Content-Length is reached. However, that means you can't use ruby's automatic decompression, because Content-Length is the number of compressed bytes, while read_body yields chunks of uncompressed data.

Here's sample code to reproduce. Run the following http server. Note chunked is currently false, but can be toggled.

```
require 'webrick'

server = WEBrick::HTTPServer.new :Port => 8000
trap 'INT' do
  server.shutdown
end

# toggle this
chunked = false

server.mount_proc '/' do |req, res|
  res.status = 200
  res['Content-Type'] = 'text/plain'

  str = "0123456789" * 10000
  res.body = str
  if chunked
    res.chunked = true
  else
    res['Content-Length'] = str.length
  end
end

server.start
```

Run the following http client code. In order to simulate a closed connection, the block raises EOFError.

```
require 'net/http'
require 'uri'
```

```
uri = URI("http://localhost:8000/")
Net::HTTP.start(uri.host, uri.port) do |http|
  http.request_get(uri.path) do |response|
    response.read_body do |chunk|
      puts "Read #{chunk.length} bytes"
      raise EOFError.new("whoops")
    end
  end
end
puts "EOF was silently caught"
```

When chunked encoding is used, the exception is properly raised. I believe ruby is retrying the request because GET is idempotent:

```
$ ruby --version
ruby 2.4.3p205 (2017-12-14 revision 61247) [x86_64-darwin15]
$ ruby client.rb
Content-Length:
Transfer-Encoding: chunked
Read 16377 bytes
Content-Length:
Transfer-Encoding: chunked
Read 16377 bytes
client.rb:11:in `block (3 levels) in <main>': whoops (EOFError)
    from /usr/local/opt/rbenv/versions/2.4.3/lib/ruby/2.4.0/net/protocol.rb:429:in `call_block'
    from /usr/local/opt/rbenv/versions/2.4.3/lib/ruby/2.4.0/net/protocol.rb:420:in `<<'
    from /usr/local/opt/rbenv/versions/2.4.3/lib/ruby/2.4.0/net/protocol.rb:122:in `read'
    from /usr/local/opt/rbenv/versions/2.4.3/lib/ruby/2.4.0/net/http/response.rb:322:in `read_chun
ked'
    from /usr/local/opt/rbenv/versions/2.4.3/lib/ruby/2.4.0/net/http/response.rb:286:in `block in 
read_body_0'
    from /usr/local/opt/rbenv/versions/2.4.3/lib/ruby/2.4.0/net/http/response.rb:278:in `inflater'
    from /usr/local/opt/rbenv/versions/2.4.3/lib/ruby/2.4.0/net/http/response.rb:283:in `read_body
_0'
    from /usr/local/opt/rbenv/versions/2.4.3/lib/ruby/2.4.0/net/http/response.rb:204:in `read_body
'
    from client.rb:9:in `block (2 levels) in <main>'
    from /usr/local/opt/rbenv/versions/2.4.3/lib/ruby/2.4.0/net/http.rb:1455:in `block in transpor
t_request'
    from /usr/local/opt/rbenv/versions/2.4.3/lib/ruby/2.4.0/net/http/response.rb:165:in `reading_b
ody'
    from /usr/local/opt/rbenv/versions/2.4.3/lib/ruby/2.4.0/net/http.rb:1454:in `transport_request
'
    from /usr/local/opt/rbenv/versions/2.4.3/lib/ruby/2.4.0/net/http.rb:1416:in `request'
    from /usr/local/opt/rbenv/versions/2.4.3/lib/ruby/2.4.0/net/http.rb:1317:in `request_get'
    from client.rb:6:in `block in <main>'
    from /usr/local/opt/rbenv/versions/2.4.3/lib/ruby/2.4.0/net/http.rb:877:in `start'
    from /usr/local/opt/rbenv/versions/2.4.3/lib/ruby/2.4.0/net/http.rb:608:in `start'
    from client.rb:5:in `<main>'
```

When chunked encoding is not used, the exception is not raised:

```
ruby client.rb
Content-Length: 100000
Transfer-Encoding:
Read 0 bytes
EOF was silently caught
```

I verified the behavior exists as far back as ruby 1.9.3p551. It was introduced in https://github.com/ruby/ruby/commit/cdc7602379c9d911983db2c044d69ac417869266#diff-8c2ab8e0fb4f052e1d95ab6334e192c1R949.

History

#1 - 08/07/2018 07:52 AM - joshc (Josh C)

- Description updated

#2 - 08/17/2018 09:00 AM - naruse (Yui NARUSE)

What is the ideal behavior you think? Just below?

```
diff --git a/lib/net/http/response.rb b/lib/net/http/response.rb
index 66132985d9..7c744d02f4 100644
--- a/lib/net/http/response.rb
+++ b/lib/net/http/response.rb
@@ -290,7 +290,7 @@ def read_body_0(dest)

       clen = content_length()
       if clen
-        @socket.read clen, dest, true   # ignore EOF
+        @socket.read clen, dest
         return
       end
       clen = range_length()
```

#3 - 09/20/2018 10:40 PM - joshc (Josh C)

If @socket.read clen, dest reads fully clen bytes then that seems ok. But if it can read fewer than clen bytes, then we should keep reading until we read clen bytes or reach EOF.

#4 - 01/22/2019 11:37 PM - joshc (Josh C)

I submitted a PR against trunk: https://github.com/ruby/ruby/pull/2074

#5 - 01/29/2019 01:57 PM - naruse (Yui NARUSE)

- Backport changed from 2.3: UNKNOWN, 2.4: UNKNOWN, 2.5: UNKNOWN to 2.3: DONTNEED, 2.4: DONTNEED, 2.5: DONTNEED

- Target version set to 2.7

I checked the code again and I noticed I wrote a code which depends current behavior before.
It is to resume with the partially downloaded result.

I consider something like this code with a option or changing the behavior with migration period...

#6 - 02/11/2019 06:32 PM - joshc (Josh C)

> It is to resume with the partially downloaded result.

Doesn't Net::HTTPResponse#read_body raise if called more than once? How can the caller resume?

#7 - 02/12/2019 05:03 AM - naruse (Yui NARUSE)

joshc (Josh C) wrote:

> > It is to resume with the partially downloaded result.
>
> Doesn't Net::HTTPResponse#read_body raise if called more than once? How can the caller resume?

Save first response body, set range HTTP Header and concat the 2nd response body.

#8 - 08/08/2019 11:18 PM - joshc (Josh C)

When a range is requested, the content-length of the response is the number of bytes in the partial response, so I would still expect an exception to be raised if the partial response is truncated:

```
$ curl -s -v -r 0-100 -O https://cache.ruby-lang.org/pub/ruby/2.6/ruby-2.6.3.tar.gz
*   Trying 151.101.65.178...
...
> GET /pub/ruby/2.6/ruby-2.6.3.tar.gz HTTP/2
> Host: cache.ruby-lang.org
> Range: bytes=0-100
> User-Agent: curl/7.54.0
> Accept: */*
>
...
< content-type: application/x-tar
< server: AmazonS3
< accept-ranges: bytes
< age: 1533346
< content-range: bytes 0-100/16784748
< accept-ranges: bytes
< date: Thu, 08 Aug 2019 23:12:49 GMT
< via: 1.1 varnish
< x-served-by: cache-sea1040-SEA
< x-cache: HIT
< x-cache-hits: 0
< x-timer: S1565305970.902169,VS0,VE0
< content-length: 101
<
{ [101 bytes data]
```

#9 - 08/12/2019 09:13 PM - joshc (Josh C)

naruse (Yui NARUSE) wrote:

> joshc (Josh C) wrote:
>
> > > It is to resume with the partially downloaded result.
> >
> > Doesn't Net::HTTPResponse#read_body raise if called more than once? How can the caller resume?
>
> Save first response body, set range HTTP Header and concat the 2nd response body.

If you save the first response body, and make a new request with Range: bytes=X-Y, then the Content-Length header in the second response should specify the number of bytes to expect, or the Content-Length header should be omitted in the case of chunked encoding. For example, given:

```
require 'net/http'
require 'uri'
require 'openssl'

uri = URI("http://cache.ruby-lang.org/pub/ruby/2.6/ruby-2.6.3.tar.gz")

http = Net::HTTP.new(uri.host, uri.port)
#http.set_debug_output($stderr)
http.start
begin
  pos = 0
  req = Net::HTTP::Get.new(uri.path)
  req['Accept'] = '*/*'
  req['Range'] = "bytes=#{pos}-9"

  http.request(req) do |response|
    clen = response['Content-Length'].to_i
    puts "Content-Length #{clen}"
    puts "Content-Range  #{response['Content-Range']}"
    pos += clen
  end

  req = Net::HTTP::Get.new(uri.path)
  req['Accept'] = '*/*'
  req['Range'] = "bytes=#{pos}-#{pos+9}"
  http.request(req) do |response|
    clen = response['Content-Length'].to_i
    puts "Content-Length #{clen}"
    puts "Content-Range  #{response['Content-Range']}"
    pos += clen
  end
  puts "Downloaded #{pos} bytes"
ensure
  http.finish
end
```

Produces:

```
$ ruby range.rb
Content-Length 10
Content-Range  bytes 0-9/16784748
Content-Length 10
```

```
Content-Range  bytes 10-19/16784748
Downloaded 20 bytes
```

In other words, if the Content-Length is specified, it should always specify the number of bytes to read (or drain) from the socket. https://tools.ietf.org/html/rfc7230#section-3.4 specifically says:

> A client that receives an incomplete response message, which can
> occur when a connection is closed prematurely or when decoding a
> supposedly chunked transfer coding fails, MUST record the message as
> incomplete.
> ...
> A message that uses a valid Content-Length is incomplete
> if the size of the message body received (in octets) is less than the
> value given by Content-Length.

So it should never be ok to silently ignore EOF.