MPTCP API in Ruby

Grégory Vander Schueren
Raphaël Bauduin
Thibault Gérondal

Option 1: via Ruby standard library

- First, we tried a “pure Ruby” solution
- Ruby standard library exposes getsockopt(), why not reuse it?
- Can easily use getsockopt() to retrieve data (ie. subflows ids).
  - This works, we easily get a stream of bytes...
  - Challenge is to map this stream of bytes to Ruby objects!
  - We use the ‘FFI’ library for this, seems promising, very actives library.

```ruby
class SubStatus < FFI::Struct
  layout  :id,         :uint8,
          :low_prio,   :uint16
end

class SubIds < FFI::Struct
  layout :sub_count,  :uint8,
         :sub_status, [SubStatus, 256]
  # Since we don't know the size of the array in advance, we just set it to
  # the maximum possible size, which is 2^8 (uint8).
end

def self.get_sub_ids(sock)
  # In this case, we can use the getsockopt method exposed by Ruby Socket
  # class. No need to call underlying original C function.
  opt = sock.getsockopt(:IPPROTO_TCP, MPTCP_GET_SUB_IDS)
  memBuf = FFI::MemoryPointer.new(SubIds).put_bytes(0, opt.data)
  SubIds.new(memBuf)
end

# Then, we can do:
# subs = get_sub_ids(sock)
# subs_count = subs[:sub_count]
# sub1_id = subs[1][:id]
```

Option 1: via Ruby standard library CONTD

- Can easily use getsockopt() to retrieve data (ie. subflows ids).
- BUT, the getsockopt() exposed by Ruby only allows to GET data.
  - Seems reasonable... right?
- We CANNOT set data like with getsockopt() like in the MPTCP API
  - We thus cannot use it for opening subflows, closing subflows, etc.
  - Need for another solution...

Option 2: exposing libc function into Ruby

- With FFI, really easy to expose libc functions into Ruby

```
module MPTCP
  extend FFI::Library
  ffi_lib 'c'
  attach_function :getsockopt, [ :int, :int, :int, :pointer, :pointer ], :int
end

MPTCP.getsockopt(...)
```

Exposing libc function into Ruby CONTD

- Seems very promising.
- BUT, mapping from C structs to Ruby objects seems incorrect.
  - Probably our fault…
  - Should dig more into FFI library & read documentation in details.
- In Ruby we can “reopen” classes and add methods to them.
  - No need to patch Ruby itself
  - Could simply write a library that adds methods to the Socket class