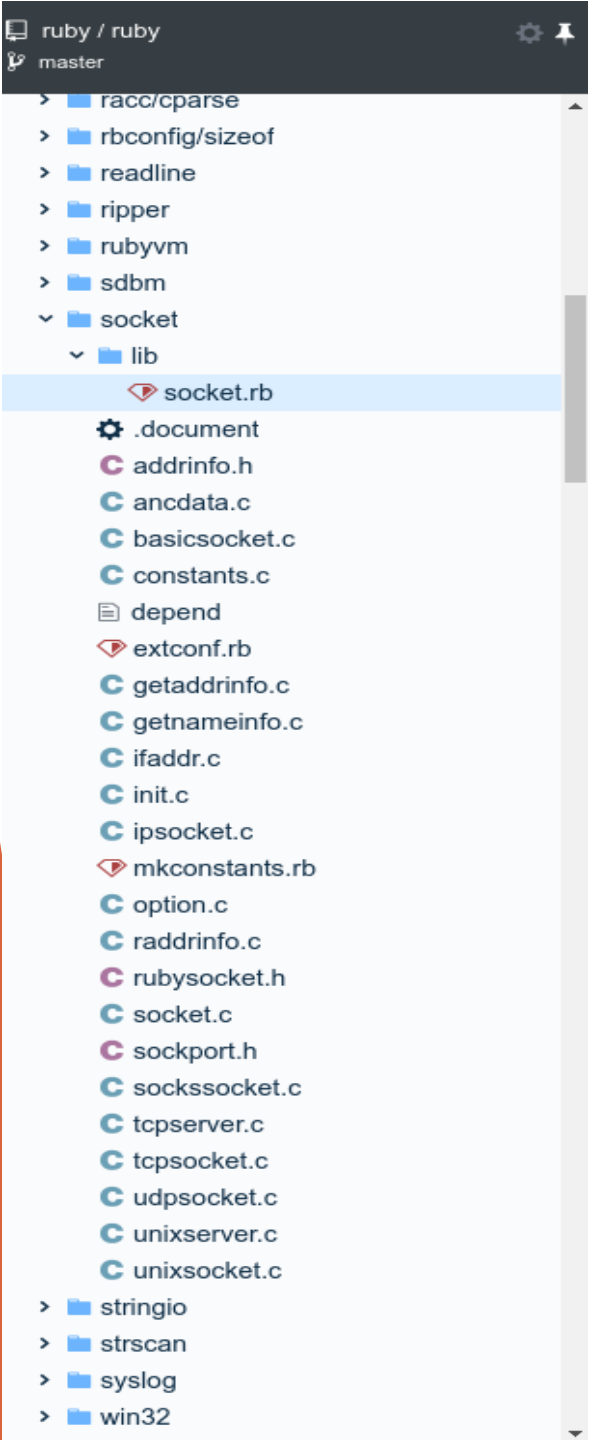
Socket Programming With RUBY

Sushant Bajracharya

/sushant12



require ‘socket’

- Part of ruby standard lib
- Provides thin bindings to C libs
- Includes different classes for TCP, UDP sockets, as well as all necessary primitives

How sockets communicate?

Find each other

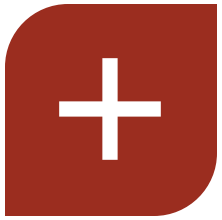
Use IP and Port to relay messages

Ports enable hosts to support multiple sockets

Socket that listens is a “server”

Socket that initiates a connection is “client”

Listener (Server)



CREATE

BIND

LISTEN

ACCEPT

CLOSE

Create, Bind, Listen

```ruby
require 'socket'

class Server
  def initialize(port)
    @server = TCPServer.new(port)
    @connections = []
    puts "Listening on port #{port}"
  end
```

Low-level implementation

```ruby
require 'socket'

# create a socket of type TCP (:STREAM)
# if you wanted to create a socket of type UDP, you
# need to pass :DGRAM
socket = Socket.new(:INET, :STREAM)

socket.bind(Socket.pack_sockaddr_in(3000, '127.0.0.1'))
socket.listen(Socket::SOMAXCONN)

```

Accept connection

```
def start
  Socket.accept_loop(@server) do |connection|
    @connections << connection
    Thread.new do
      loop do
        handle(connection)
      end
    end
  end
end
```

Low-level implementation

```ruby
require 'socket'

socket = Socket.new(:INET, :STREAM)
socket.bind(Socket.pack_sockaddr_in(3000. '127.0.0.1'))
socket.listen(Socket::SOMAXCONN)
loop do
  connection, _ = socket.accept
end
```

Why “close” connection?

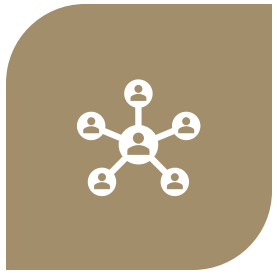
“Socket.accept” returns one connection and then exits

Garbage collect

Open file limit

Initiator (Client)

CREATE



CONNECT

CLOSE

Create and Connect

```
def self.request
  @client = TCPSocket.new(host, port)
  listen
  send
end
```

Low-level implementation

```ruby
require 'socket'

socket = Socket.new(:INET, :STREAM)

remote_address = Socket.pack_sockaddr_in(3000, '127.0.0.1')

socket.connect(remote_address)


```

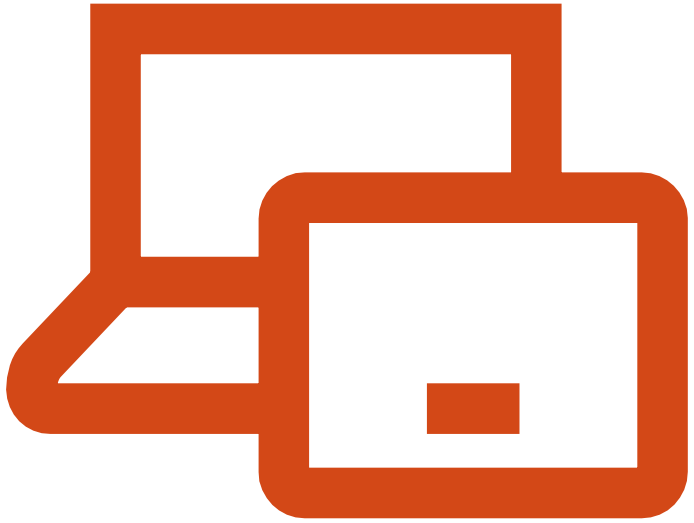
Socket can READ and WRITE

```ruby
require 'socket'

Socket.tcp_server_loop(3000) do |conn|
  conn.write('Welcome ' << conn.read_nonblock(200))
  conn.close
rescue Errno::EAGAIN
  IO.select([conn])
  retry
rescue EOFError
  break
end

```

netcat

A unix utility tool to create arbitrary TCP and UDP connections or servers

```
echo “<your name>” | nc <ip> <port>
```



“write” under the hood

- Big improvements in performance
- Pass the work to OS Kernel
- Kernel will batch many small packets into larger ones
- Kernel decides whether to send data immediately or later

Nagel’s Algo Friend or Foe?

Good for telnet like apps

Bad for protocol like http

Disable Nagel's Algo

```
364
365       # disables Nagle's Algorithm, prevents multiple round trips with MULTI
366       if [:IPPROTO_TCP, :TCP_NODELAY].all?{|c| Socket.const_defined? c}
367         def set_tcp_nodelay
368           @sock.setsockopt(Socket::IPPROTO_TCP, Socket::TCP_NODELAY, 1)
369         end
370       else
371         def set_tcp_nodelay
372         end
373       end
374
```

TCP Chat App

Server

- State to maintain connections
- Pass messages to connections
- Close connections of disconnected client

Client

- Listen for a message
- Send a message

elixir

Elixir is a dynamic, functional language designed for building scalable and maintainable applications

ElixirNepal.org

Build a community

Host workshops / meetups / conferences

ElixirNepal.org

Nepal, the go-to country to find elixir developers