

Networking

Low level

tcp/udp

High Level

http/ftp/smtp etc....

Low Level

TCP vs UDP

UDP

a time server

UDPSocket

```
require 'socket'

###
# Server Code
#
server = UDPSocket.new
server.bind nil, 12321

loop do
  text, sender = server.recvfrom(1)
  server.send(Time.new.to_s + "\n", 0, sender[3], sender[1])
end
```

UDPSocket

```
require 'socket'
require 'timeout'

###
# Client code
#
socket = UDPSocket.new
socket.connect('localhost', 12321)

socket.send('', 0)
timeout(10) do
  time = socket.gets
  puts time
end
```

TCP

TCPServer

```
require 'socket'

###
# Server code
#
server = TCPServer.new(12321)
while session = server.accept
  session.puts Time.new
  session.close
end
```

TCP Socket

```
require 'socket'

###
# Client code
#
session = TCPSocket.new('localhost', 12321)
time = session.gets
session.close
puts time
```

A Threaded Server

Threaded Server

```
require 'socket'

###
# Server code
#
server = TCPServer.new(12321)
while session = server.accept
  Thread.new(session) do |my_session|
    my_session.puts Time.new
    my_session.close
  end
end
end
```

We can do better

GServer

```
require 'gserver'

###
# Server code
#
class TimeServer < GServer
  def initialize(port=12321, *args)
    super
  end
  def serve(io)
    io.puts(Time.now.to_s)
  end
end

# Run the server with logging enabled (it's a separate thread).
server = TimeServer.new
server.audit = true # Turn logging on.
server.start
server.join
```

High Level

net/telnet

Web Server

```
require 'net/telnet'

tn = Net::Telnet.new('Host'      => 'google.com',
                    'Port'      => 80,
                    'Timeout'   => 60 )
tn.write("GET / HTTP/1.0\r\n\r\n")
puts tn.read
```

Checking Your Email

I can't respond to any emails today.

Something has crashed on my computer...

net/pop

```
require 'net/pop'  
  
pop = Net::POP3.new('pop.gmail.com')  
pop.start('username', 'password')  
pop.mails.each do |msg|  
  puts msg.header.grep /^Subject/  
end
```

net/pop

```
require 'net/pop'

pop = Net::POP3.new('pop.gmail.com')
pop.start('username', 'password')
pop.mails.each do |msg|
  msg.delete if msg.all =~ /. *spam!.* /
end
```

net/imap

```
require 'yaml'  
require 'net/imap'  
  
credentials = YAML.load_file('/Users/aaron/f.txt')  
  
imap = Net::IMAP.new('imap.gmail.com', 993, true)  
imap.login(credentials['username'], credentials['password'])  
imap.examine('INBOX')  
p imap.responses  
imap.logout
```

Sending An Email

Net::SMTP.start()

- server
- port
- domain
- username
- password
- auth type

net/smtp

```
require 'net/smtp'
```

```
Net::SMTP.start('smtp.example.com') do |smtp|  
  smtp.sendmail('hello world!', 'from@example.com', 'to@example.com')  
end
```

Surfing the Web

net/http

Net::HTTP.new

- host
- port
- proxy address
- proxy port
- proxy user
- proxy password

Random Number

```
require 'net/http'

connection = Net::HTTP.new('random.org')
response, data = connection.get('/integers/?
num=1&min=1&max=100&col=1&base=10')

case response.code
when '200'
  data =~ /<pre class="data">(\d+) .*</pre>/m
  puts $1
else
  raise 'Crap... some other response'
end
```

Posting

```
require 'net/http'

connection = Net::HTTP.new('tenderlovmaking.com')
response, data = connection.post('/index.php', 's=Aaron')
case response.code
when '200'
  puts data
else
  raise "Crap... some other response #{response.code}"
end
```

Request Object

```
require 'net/http'

connection = Net::HTTP.new('tenderlovmaking.com')
get = Net::HTTP::Get.new('/index.php?s=aaron')

connection.request(get) do |response|
  response.read_body { |part|
    puts part
  }
end
```

open-uri

open-uri

```
require 'open-uri'

open("http://www.ruby-lang.org/",
     { 'User-Agent' => 'awesome!' })
{|f|
  f.each_line {|line| p line}
}
```

mechanize

mechanize

- fetch pages
- click links
- submit forms

finding text

```
require 'rubygems'  
require 'mechanize'  
  
agent = WWW::Mechanize.new  
page = agent.get('http://google.com/')  
  
form = page.form('f')  
form.q = 'Aaron Patterson'  
page = form.submit(form.buttons.first)  
page.search('//a[@class="l"]').each do |a_tag|  
  puts a_tag.inner_text  
end
```

clicking a link

```
require 'rubygems'  
require 'mechanize'  
  
agent = WWW::Mechanize.new  
page = agent.get('http://google.com/')  
  
form = page.form('f')  
form.q = 'Aaron Patterson'  
page = form.submit(form.buttons.first)  
new_page = page.links.first.click
```

Distributed Ruby

DRb

Server

- Starts TCPServer and listens
- Binds an object to the server instance
- Accepts connections from clients
- May provide access control

Client

- Establishes a connection to the server
- Binds a local object to the remote server
- Sends messages to the server object
- Receives responses from the server

DRb Server

Server

```
require 'drb'

class MyObject
  def message
    "hello world"
  end
end

DRb.start_service('druby://:1234', MyObject.new)
DRb.thread.join
```

Client

```
require 'drb'  
  
ro = DRbObject.new(nil, 'druby://:1234')  
puts ro.message
```

Redux

```
require 'drb'

class MyObject
  attr_reader :list
  def initialize
    @list = []
  end

  def append(something)
    @list << something
  end
end

DRb.start_service('druby://:1234', MyObject.new)
DRb.thread.join
```

Chat Server

```
require 'drb'
require 'drb/observer'

class ChatServer
  include DRb::DRbObservable

  def speak(who, message)
    changed(true)
    notify_observers(who, message)
  end
end

DRb.start_service('druby://localhost:9000', ChatServer.new)
DRb.thread.join
```

Chat Client

```
require 'drb'

class ChatClient
  include DRbUndumped

  attr_accessor :name
  def initialize(name, service)
    @name = name
    @service = service
    service.add_observer(self)
  end

  def hello!
    @service.speak(self, "Hello!")
  end

  def update(who, message)
    puts "#{who.name} said #{message}"
  end
end

DRb.start_service
server = DRbObject.new(nil, 'druby://localhost:9000')
chatter = ChatClient.new('Aaron', server)
chatter.hello!
```

Rinda

a tuplespace

Tuple Server

```
require 'drb'  
require 'rinda/tuplespace'  
  
ts = Rinda::TupleSpace.new  
DRb.start_service('druby://localhost:9000', ts)  
DRb.thread.join
```

Tuple Client

```
require 'drb'  
require 'rinda/tuplespace'  
  
DRb.start_service  
ts = DRbObject.new(nil, 'druby://localhost:9000')  
  
...
```

Tuple Client

- read

Tuple Client

- read
- read_all

Tuple Client

- read
- read_all
- write

Tuple Client

- read
- read_all
- write
- take

Tuple Client

- read
- read_all
- write
- take
- notify

```
ts.read([:Sum, nil])
```

```
ts.read_all([:Sum, nil, nil])
```

```
ts.take([:Sum, nil])
```

```
ts.write([:Sum, 10])
```

```
ts.write([:Sum, 10], 30)
```

```
ts.notify('write', [:Sum, nil])
```

```
require 'drb'
require 'rinda/tuplespace'

ts = Rinda::TupleSpace.new

kittens = ts.notify('write', ['kittin', nil])
puppies = ts.notify('take', ['puppy', nil])

th1 = Thread.new do
  kittens.each { |op,k| puts "#{op}: #{k.join(' ')}" }
end

th2 = Thread.new do
  puppies.each { |op,k| puts "#{op}: #{k.join(' ')}" }
end

sleep 1
```