情報ネットワーク学演習 2

課題:ルータの CLI を作ろう

氏名: 佐竹 幸大
学籍番号: 33E16010

課題内容 (ルータの CLI を作ろう)

ルータのコマンドラインインタフェース (CLI) を作ろう。

次の操作ができるコマンドを作ろう。

- ルーティングテーブルの表示
- ルーティングテーブルエントリの追加と削除
- ルータのインタフェース一覧の表示
- そのほか、あると便利な機能

コントローラを操作するコマンドの作りかたは、第 3 回パッチパネルで作った patch_panel コマンドを参考にしてください。

解答内容 (ルータの CLI を作ろう)

ルーティングテーブルの表示

プログラムの説明

ルーティングテーブルを出力させるコマンドを、ShowTable によって定義する。すなわち、以下のコマンドによってルーティングテーブルの出力を行う。

```
ensyuu2@ensyuu2-VirtualBox:~/simple-router-ksatake$ ./bin/simple_router ShowTable
routing table
```

./bin/simple_router に、以下を追記し、ルーティングテーブルの出力を行うコマンドを定義した。

```
# simple_router command
module SimpleRouterApp
  extend GLI::App

  include Pio
  desc 'Show routing tabel'
  arg_name '@routung_table'
  command :ShowTable do |c|
    c.desc 'Location to find socket files'
    c.flag [:S, :socket_dir], default_value: Trema::DEFAULT_SOCKET_DIR

    c.action do |_global_options, options, args|
      print "routing table¥n"
      print "=======================================¥n"
      print "Destination/netmask | Next hop¥n"
      print "---------------------------------------¥n"
      @routung_table = Trema.trema_process('SimpleRouter',
options[:socket_dir]).controller.get_table()
      @routung_table.each do |each|
        each.each_key do |key|
          print IPv4Address.new(key).to_s, "/", @routung_table.index(each), "¥t |
",each[key].to_s, "¥n"
        end
      end
      print "=======================================¥n"
    end
  end
```

また、上記の記述において使用されている get_table メソッドを、 ./lib/simple_router.rb に追記することで定義した。定義箇所は以下である。

```
def get_table()
```

```
    return @routing_table.getDB()
  end
```

また、上記で使用されている getDB は、 ./lib/routing_table.rb によって定義されている。定義文は以下である。

```
def getDB()
    return @db
  end
```

実行結果

simple_router.conf において、起動時のルーティングテーブルは以下のように定義されている。本ファイルは変更を加えずに、元のまま用いている。

```
  ROUTES = [
    {
      destination: '0.0.0.0',
      netmask_length: 0,
      next_hop: '192.168.1.2'
    }
  ]
end
```

実行結果は以下の通りである。

```
ensyuu2@ensyuu2-VirtualBox:~/simple-router-ksatake$ ./bin/simple_router ShowTable
routing table
=========================================
Destination/netmask | Next hop
-----------------------------------------
0.0.0.0/0       | 192.168.1.2
=========================================
```

ルーティングテーブルエントリの追加と削除

プログラム説明

ルーティングテーブルにエントリを追加および削除を行うコマンドをそれぞれ、addTable、deleteTable とした。add Table の引数は、[宛先アドレス] [ネットマスク長] [転送先アドレス] であり、deleteTable の引数は、[宛先アドレス] [ネットマスク長] である。

具体的に、[宛先アドレス] [ネットマスク長] [転送先アドレス] を指定すると、addTable、deleteTable はそれぞれ以下のようなコマンドとなる。

```
ensyuu2@ensyuu2-VirtualBox:~/simple-router-ksatake$ ./bin/simple_router addTable 192.168.1.0 24 192.168.1.1
ensyuu2@ensyuu2-VirtualBox:~/simple-router-ksatake$ ./bin/simple_router deleteTable 192.168.1.0 24
```

./bin/simple_router に以下を追記し、ルーティングテーブルにエントリの追加および削除を行うコマンドをそれぞれ定義した。

```
  desc 'add routing tabel'
  arg_name 'destination netmask nexthop'
  command :addTable do |c|
    c.desc 'Location to find socket files'
    c.flag [:S, :socket_dir], default_value: Trema::DEFAULT_SOCKET_DIR

    c.action do |_global_options, options, args|
      destination = args[0]
      netmask = args[1]
      nexthop = args[2]
      Trema.trema_process('SimpleRouter',
options[:socket_dir]).controller.add_routing_table(destination, netmask, nexthop)
    end
  end

  desc 'delete routing tabel'
```

```
  arg_name 'destination netmask'
  command :deleteTable do |c|
    c.desc 'Location to find socket files'
    c.flag [:S, :socket_dir], default_value: Trema::DEFAULT_SOCKET_DIR

    c.action do |_global_options, options, args|
      destination = args[0]
      netmask = args[1]
      Trema.trema_process('SimpleRouter',
options[:socket_dir]).controller.delete_routing_table(destination, netmask)
    end
  end
```

上述の記述で使用している add_routing_table および delete_routing_table は、 ./lib/simple_router.rb に追記する形で定義した。定義箇所を以下に示す。

```
  def add_routing_table(destination, netmask, nexthop)
    options = {:destination => destination, :netmask_length => netmask.to_i, :next_hop
=> nexthop}
    @routing_table.add(options)
  end

  def delete_routing_table(destination, netmask)
    options = {:destination => destination, :netmask_length => netmask.to_i}
    @routing_table.delete(options)
  end
```

また、上記のプログラムで使用している add、delete メソッドは RouingTable.rb に追記することで定義した。定義箇所は以下である。

```
  def add(options)
    netmask_length = options.fetch(:netmask_length)
    prefix = IPv4Address.new(options.fetch(:destination)).mask(netmask_length)
```

```
    @db[netmask_length][prefix.to_i] = IPv4Address.new(options.fetch(:next_hop))
  end

  def delete(options)
    netmask_length = options.fetch(:netmask_length)
    prefix = IPv4Address.new(options.fetch(:destination)).mask(netmask_length)
    @db[netmask_length].delete(prefix.to_i)
  end
```

実行は以下の手順で行った。

・ルーティングテーブルの追加
・ルーティングテーブルの表示
・ルーティングテーブルの削除
・ルーティングテーブルの表示

その実行結果は以下の通りである。

```
ensyuu2@ensyuu2-VirtualBox:~/simple-router-ksatake$  ./bin/simple_router  addTable
192.168.1.0 24 192.168.1.1
ensyuu2@ensyuu2-VirtualBox:~/simple-router-ksatake$ ./bin/simple_router ShowTable
routing table
=======================================
Destination/netmask | Next hop
---------------------------------------
0.0.0.0/0       | 192.168.1.2
192.168.1.0/24          | 192.168.1.1
=======================================
ensyuu2@ensyuu2-VirtualBox:~/simple-router-ksatake$ ./bin/simple_router deleteTable
192.168.1.0 24
ensyuu2@ensyuu2-VirtualBox:~/simple-router-ksatake$ ./bin/simple_router ShowTable
routing table
=======================================
Destination/netmask | Next hop
```

```
----------------------------------------
0.0.0.0/0       | 192.168.1.2
========================================
```

ルータのインターフェース一覧の表示

ルーティングテーブルエントリの出力コマンドを ShowInterface で定義した。具体的には、以下のコマンドによって実行できる。

```
ensyuu2@ensyuu2-VirtualBox:~/simple-router-ksatake$ ./bin/simple_router ShowInterface
```

./bin/simple_router に、ルーティングテーブルの出力メソッド ShowInterface を追記した。該当箇所を以下に示す。

```
desc 'Show interface'
  arg_name 'interface'
  command :ShowInterface do |c|
    c.desc 'Location to find socket files'
    c.flag [:S, :socket_dir], default_value: Trema::DEFAULT_SOCKET_DIR

    c.action do |_global_options, options, args|
      print "interface list¥n"
      print "==========================================¥n"
      print "port | mac address    ¥t | ip address/netmask¥n"
      print "------------------------------------------¥n"
      interface = Trema.trema_process('SimpleRouter', options[:socket_dir]).controller.get_interface()
      interface.each do |each|
        print each[:port_number].to_s," | ", each[:mac_address], " | ", each[:ip_address]+"/"+each[:netmask_length].to_s,"¥n"
      end
      print "==========================================¥n"
```

```
    end
  end
```

上記のプログラムで使用している get_interface メソッドは、./lib/simple_router.rb に追記して定義した。定義箇所を以下に示す。

```
  def get_interface()
    interfaceList = Array.new()
    Interface.all.each do |each|
      interfaceList << {:port_number => each.port_number, :mac_address => each.mac_address.to_s, :ip_address => each.ip_address.value.to_s, :netmask_length => each.netmask_length}
    end
    return interfaceList
  end
```

起動時のインターフェースは simple_router.conf で定義されている。変更は加えず、元のまま使用している。

```
# Simple router configuration
module Configuration
  INTERFACES = [
    {
      port: 1,
      mac_address: '01:01:01:01:01:01',
      ip_address: '192.168.1.1',
      netmask_length: 24
    },
    {
      port: 2,
      mac_address: '02:02:02:02:02:02',
      ip_address: '192.168.2.1',
      netmask_length: 24
```

```
    }
  ]
```

実行結果は以下である。

```
ensyuu2@ensyuu2-VirtualBox:~/simple-router-ksatake$ ./bin/simple_router
ShowInterface
interface list
================================================
port | mac address       | ip address/netmask
------------------------------------------------
1    | 01:01:01:01:01:01 | 192.168.1.1/24
2    | 02:02:02:02:02:02 | 192.168.2.1/24
================================================
ensyuu2@ensyuu2-VirtualBox:~/simple-router-ksatake$
```

以上の結果より、正しく実装されていることが確認できた。

参考文献

・デビッド・トーマス+アンドリュー・ハント(2001)「プログラミング Ruby」ピアソン・エデュケーション.

・テキスト: 13 章 "ルータ (後編)"

謝辞

プログラム作成にあたり、田中達也さんのものを参考にさせて頂きましたのでここにて謝辞の意を示します。