情報ネットワーク学演習 2
課題：ルータの CLI を作ろう

秋下 耀介
情報ネットワーク学専攻
学籍番号；33E16001
E-mail: y-akishita@ist.osaka-u.ac.jp

2016 年 11 月 8 日（火）

1 課題：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．

1. ルーティングテーブルの表示
2. ルーティングテーブルエントリの追加と削除
3. ルータのインタフェース一覧の表示
4. 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．

2 解答：ルータの CLI を作ろう

各機能の実装にあたっては，simple_router.rb および routing_table.rb の変更を行った．また，コマンドの実行のために，前回の課題にあった patch_panel を参考にして，bin/simple_router を作成した．

2.1 ルーティングテーブルの表示

ルータのルーティングテーブルの表示では，listRoute コマンドによって simple_router.conf にあるような情報，すなわち

1. destination：対象となるネットワークまたはホスト
2. netmask：対象ネットワークのネットマスク
3. next_hop：ゲートウエイの IP アドレス

の表示を行うものとする．

以下にプログラムに記述した詳細と，動作確認の結果を示す．

2.1.1 simple_router.rb

ルーティングテーブルの表示を行うために，listRoute というメソッドを実装した．

```
def listRoute()
   temp ="\n" + "-----------------------------------\n" + "destination/netmask \t next_hop\n"
   + "-----------------------------------\n"
   @route,@len = @routing_table.listDB()
   @len.downto(0).each do |each|
     @route[each].each do |key,next_hop|
       temp += IPv4Address.new(key).to_s +  "/" + each.to_s + "\t " + next_hop.to_s +  "\n"
     end
   end
   return temp
 end
```

このメソッドでは，ルーティングテーブル表示を行うためのメッセージを作成する．まず，表示させる項目を代入している．そして，ルーティングテーブルは routing_table.rb で定義されているため，そちらから，@db および MAX_NETMASK_LENGTH を取り出す．そして，ネットマスク長の長い順にルーティングテーブルを捜索し，対象となるネットワークのホストと，ゲートウェイの IP アドレスを@route（すなわち@db）から抜き出し，それらとネットマスクを順にメッセージに埋め込む．これを繰り返し，ルーティングテーブルにある全ての情報を一つのメッセージに代入する．そして，それを呼び出し元に返す．

2.1.2 routing_table.rb

ここでは，ルーティングテーブル@db および，ネットマスクの最大長 MAX_NETMASK_LENGTH を呼び出し元に返り値として返す．

```
def listDB()
   return @db,MAX_NETMASK_LENGTH
 end
```

@db は routing_table.rb で定義されているため，simple_router.rb で処理を完結させることはできず，このような実装を行った．

2.1.3 bin/simple_router

ここでは，前回の bin/patch_panel を参考にして，listRoute コマンドを実装した．

```
desc 'List the routing table'
  command :listRoute 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 Trema.trema_process('SimpleRouter', options[:socket_dir]).controller
      .listRoute()
    end
  end
```

ここでは，simple_router.rb の listRoute メソッドを呼び出している．そして，その返り値を表示している．

2.1.4 動作確認

実際に listRoute コマンドを実行した際の表示は以下のようになっている．

```
ensyuu2@ensyuu2-VirtualBox:~/ensyuu2/work5/simple-router-yosuke-akishita$ bundle
exec ./bin/simple_router listRoute

---------------------------------
destination/netmask   next_hop
---------------------------------
0.0.0.0/0  192.168.1.2
```

ここで，設定ファイル（simple_router.conf）を確認すると，

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

のようになっており，コマンド実行による表示は正しいことが読み取れる．

2.2 ルーティングテーブルエントリの追加と削除

ルーティングテーブルエントリの追加は addEntry コマンド，削除は deleteEntry コマンドとして実装した．addEntry コマンドは，

```
bundle exec ./bin/simple_router addEntry [宛先アドレス] [ネットマスク長] [転送先アドレス]
```

のように引数を 3 つ必要とするものとして定義した．また，deleteEntry コマンドは，

```
bundle exec ./bin/simple_router addEntry [宛先アドレス] [ネットマスク長]
```

のように引数を 2 つ必要とするものとして定義した．

2.2.1 simple_router.rb

ここでは，エントリの追加や削除の設定して，routing_table.rb のメソッドを呼び出す．

```
 def addEntry(dest,netmask_len,next_hop)
    options = {:destination => dest, :netmask_length => netmask_len.to_i,
    :next_hop => next_hop}
    @routing_table.add(options)
  end

  def deleteEntry(dest,netmask_len)
    options = {:destination => dest, :netmask_length => netmask_len.to_i}
    @routing_table.delete(options)
  end
```

上記においては，受け取った引数を options として設定し，エントリの追加であれば routing_table.rb の add メソッドを呼び出し，エントリの削除であれば delete メソッドを呼び出す．

2.2.2 routing_table.rb

ここでは，@db にあるエントリの追加や削除を実際に行うメソッドを実装する必要がある．しかし，add メソッドについては初めから実装されていたため，今回はそちらを参考にして delete メソッドの実装のみを行った．

```
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
```

ほとんど実装としては add メソッドと同じであり，最後の追加の部分を削除に変更しただけである．操作としては，与えられたネットマスク長に一致するエントリの中から，宛先アドレスの合致するエントリを探し出し，それを削除する，といったものになっている．

2.2.3 bin/simple_router

listRoute コマンドと同様に実装を行った．ただし，今回は引数を addEntry であれば 3 つ，deleteEntry であれば 2 つ必要とするものとして定義したので，以下のような形となった．

```
desc 'Add the entry'
command :addEntry 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|
    dest = args[0]
    netmask_len = args[1].to_i
    next_hop = args[2]
    Trema.trema_process('SimpleRouter', options[:socket_dir]).controller
    .addEntry(dest, netmask_len, next_hop)
  end
end

desc 'Delete the entry'
command :deleteEntry 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|
    dest = args[0]
    netmask_len = args[1].to_i
    Trema.trema_process('SimpleRouter', options[:socket_dir]).controller
    .deleteEntry(dest, netmask_len)
  end
end
```

2.2.4 動作確認

本節では，2.1 章で実装したルーティングテーブルの表示を用いて，エントリの追加および削除ができているかの確認を行う．

```
ensyuu2@ensyuu2-VirtualBox:~/ensyuu2/work5/simple-router-yosuke-akishita$ bundle
exec ./bin/simple_router listRoute

---------------------------------
destination/netmask   next_hop
---------------------------------
0.0.0.0/0  192.168.1.2
ensyuu2@ensyuu2-VirtualBox:~/ensyuu2/work5/simple-router-yosuke-akishita$ bundle
exec ./bin/simple_router addEntry 192.168.1.2 24 192.168.1.3
ensyuu2@ensyuu2-VirtualBox:~/ensyuu2/work5/simple-router-yosuke-akishita$ bundle
exec ./bin/simple_router listRoute

---------------------------------
destination/netmask   next_hop
---------------------------------
192.168.1.0/24  192.168.1.3
0.0.0.0/0  192.168.1.2
ensyuu2@ensyuu2-VirtualBox:~/ensyuu2/work5/simple-router-yosuke-akishita$ bundle
exec ./bin/simple_router deleteEntry 192.168.1.2 24
ensyuu2@ensyuu2-VirtualBox:~/ensyuu2/work5/simple-router-yosuke-akishita$ bundle
exec ./bin/simple_router listRoute

---------------------------------
destination/netmask   next_hop
---------------------------------
0.0.0.0/0  192.168.1.2
```

まず listRoute コマンドによって設定ファイルにあるエントリしかないことが読み取れる．その後，addEntry コマンドによって，エントリを追加した後，listRoute コマンドを入力すると，先ほど入力したエントリが追加されていることがわかる．そして，deleteEntry コマンドによってエントリを削除した後，listRoute コマンドを入力すると，先ほど入れたエントリが削除されていることがわかる．以上をもってエントリの追加と削除が正しく行えていることがわかる．

2.3 ルータのインタフェース一覧の表示

ルータのインターフェース一覧の表示は，listInterface コマンドとして実装した．ここでは simple_router.conf にあるような情報，すなわち

1. port：ポート番号
2. mac_address：ルータの MAC アドレス
3. ip_address：ゲートウエイの IP アドレス
4. netmask_length：対象ネットワークのネットマスク

の表示を行うものとする．

2.3.1 simple_router.rb

ここでは，インターフェースの一覧を表示するために，listInterface というメソッドを実装した．

```
def listInterface()
  temp ="\n" + "-----------------------------------\n" + "port \t mac_address
  \t ip_address/netmask\n" + "-----------------------------------\n"
  interface = @routing_table.listInt()
  interface.each do |each|
    temp += each[:port_number] +  "\t" + each[:mac_address] +
    "\t" + each[:ip_address] + "/" + each[:netmask_length] + "\n"
  end
  return temp
end
```

このメソッドでは，インターフェースの表示を行うためのメッセージを作成する．まず，表示させる項目を代入している．そして，routing_table.rb から，ポート番号，MAC アドレス，IP アドレスおよびネットマスク長を取り出す．そして，順にメッセージとして代入する．最後に，そのメッセージを呼び出し元に返す．

2.3.2 routing_table.rb

ここでは，Array を用いて Interface に入っている設定を保存する．

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

上記のように，Interface の要素を順に見ていき，それぞれを Array の中に文字列に変換した上で保存していく．そして全ての要素が入った状態の Array を呼び出し元に返す．

2.3.3 bin/simple_router

listRoute コマンドと同様に実装を行った．listInterface メソッドを呼び出し，その返り値を表示する．

```
desc 'List the interfece'
  command :listInterface 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 Trema.trema_process('SimpleRouter', options[:socket_dir]).controller.listInterface()
    end
  end
```

2.3.4 **動作確認**

listInterface の実行結果は以下のようになった．

```
ensyuu2@ensyuu2-VirtualBox:~/ensyuu2/work5/simple-router-yosuke-akishita$ bundle
exec ./bin/simple_router listInterface

---------------------------------
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
```

ここで，設定ファイル（simple_router.conf）を確認すると，

```
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
    }
  ]
```

であるから，コマンド実行によって表示された結果は正しいものであると言える．

参考文献

[1] 情報ネットワーク学演習 2 事前準備 https://github.com/handai-trema/syllabus#事前準備

[2] Trema で OpenFlow プログラミング http://yasuhito.github.io/trema-book/#router_part1

[3] ルーティング・テーブルを表示・設定する http://itpro.nikkeibp.co.jp/article/COLUMN/20060227/230874/