Programming Challenge: Routing Information Protocol

In this lesson, you'll be writing code for the routing information protocol that we looked at previously.

WE'LL COVER THE FOLLOWING

Problem Statement

In this challenge, you will implement the routing information protocol that we just studied in the last lesson! You're given some starter code files.

Starter Code

For this coding challenge, we are providing you with a network simulator written in `python3`. The implementation of our simplified version of RIP is also required in Python. Let's look at the starter code module by module.

`topology_reader.py`

This is the entry point to our code. It takes a network topology in the form of a Python list as input and returns a list of `router` objects that reflect that topology. Here's what the topology looks like:

Sample Input

```
topology = [
  [1, [11, 2, 21, 1], [12, 4, 41, 1]],  # Routers and ports
  [2,[21, 1, 11, 1],[22, 5, 53, 1], [23,3,31,1]], #[IP of router, [port of router, IP of dest
  [3,[31,2,23,1],[32,5,52,1]],
  [4,[41,1,12, 1],[42,5,51,1]],
  [5,[51,4,42,1],[52,3,32,1],[53,2,22,1]]
]
```

The list consists of sublists. Each sublist represents **one** router. So `[1, [11, 2, 21, 1], [12, 4, 41, 1]]`, for instance, represents a router.

- The first element of this sublist is the IP address of the router. It is `1` in the case of the example above.
- The rest of the elements of this sublist are other lists that represent ports and their links to ports on other routers.
- There can be as many of these sub-sublists as there are ports.
- Each port-link sub-sublist is as follows:
  - `[IP of port of router this router, IP of destination router, IP of port of destination router, cost of link]`
- So the topology looks like:
  - `[[IP of router, [IP of port of this router, IP of destination router, IP of port of destination router, cost of link], [IP of router, [IP of port of this router, IP of destination router, IP of port of destination router, cost of link], ...]`
- Note that a link between two routers **has** to be present in both. So a link to a port on a router with IP `2` from a router with IP `1` `[1, [11, 2, 21, 1], [12, 4, 41, 1]]` is reflected in the sublist of router with IP `2`, as follows: `[2, [21, 1, 11, 1],[22, 5, 53, 1], [23,3,31,1]]`

`router.py`

This file contains two classes: `router_base` and `router`.

- The `router_base` class contains the IP address, a list of RIP entries and a list of ports for each router, along with some functions that will help you implement the protocol. The IP address is self-explanatory but we'll get to the other two in a minute.
- The `router` class **inherits** the `router_base` class and is the class **you'll be**

working in. In particular, you'll be writing the functions

`send_RIP_packets()` and `receive_RIP_packets()`.

What Our Test Does?

Our testing code is simply a **network simulator** that supplies a number of network topologies, creates a list of routers with `topology_reader()` and calls `send_RIP_packets()` on each router `steps_to_run` number of times where `steps_to_run` is randomized. It then stores the list of routers returned from the `send_RIP_packets()` function and checks if they are as expected. Note that our testing code is not visible to you.

`port.py`

This file contains the classes `port_link` and `port`.

- The `port_link` class defines the links on each port. This class consists of the destination router's IP address (`dest_IP_address`), the destination router's port's IP addresses (`dest_port_IP`) and the cost of the link (`cost`).
- The `port` class has two attributes: the IP address of the port (`port_IP`), and the link on the port (`link`) which is an object of the class `port_link`.

The `topology_to_routers` function from the `topology_reader.py` file first creates a links, then ports and then finally passes them to router objects upon creation.

`rip_packet.py`

This file consists of two classes: `RIP_entry` and `RIP_packet`.

- `RIP_entry`: each object of this class represents an entry of the forwarding table of the router. Each object of the `router` class consists of a list of these which makes up its forwarding table. Each `RIP_entry` object will have the following attributes: The IP address of the sending port (`port_IP`), the cost of sending on this link (`cost`), the IP address of the destination router (`dest_IP_address`), and the IP address of the next hop router (`next_hop_IP`).
- `RIP_packet`: when a router wants to forward its RIP entries, it does so by creating an object of this class. This class consists of the list of RIP entries

a few useful methods.

Coding Exercise

Great! Now you have some background on the code. Note that we haven't discussed the skeleton code in its entirety so you should read it to understand the methods provided. Try the challenge yourself in the widget below!

> Note that `main.py` is empty. That's okay, don't worry about it.

main.py

port.py

rip_packet.py

router.py

topology_reader.py

```
from rip_packet import RIP_entry
from rip_packet import RIP_packet
from port import port
from port import port_link

class router_base:
  def __init__(self, IP_address, rip_entries, ports):
    self.IP_address = IP_address
    self.rip_entries = rip_entries
    self.ports = ports

  def add_port(self, prt):
    self.ports.append(prt)

  def add_RIP_entry(self, port_IP, dest_IP, cost, next_hop_IP):
    new_rip_entry = RIP_entry(port_IP, cost, dest_IP, next_hop_IP)
    self.rip_entries.append(new_rip_entry)

  def find_RIP_entry(self, destination_IP_to_find):
    for entry in self.rip_entries:
      if(entry.dest_IP_address == destination_IP_to_find):
        return entry
    return None

  def set_RIP_entry_cost(self, destination_IP_to_find, new_cost):
    for i in self.rip_entries:
      if(entry.dest_IP_address == destination_IP_to_find):
        entry.set_cost(new_cost)
    return None

  def delete_RIP_entry(self, destination_IP_to_find):
```

```
    for entry in self.rip_entries:
      if(entry.dest_IP_address == destination_IP_to_find):
        self.rip_entries.remove(entry)


  def print_router(self):
    print("~~~~ Router IP address = " + str(self.IP_address) + "~~~~")
    print("---Ports---")
    print("Port IP | Destination Router IP | Destination Port IP | Cost")
    for p in self.ports:
      p.print_port()
    print("---RIP entries---")
    print("port IP | destination IP address | next hop | cost")
    for re in self.rip_entries:
      re.print_rip_entry()

  def return_router(self):
    r =[]
    r.append("~~~~ Router IP address = " + str(self.IP_address) + "~~~~")
    r.append("---Ports---")
    r.append("Port IP | Destination Router IP | Destination Port IP | Cost")
    for p in self.ports:
      r.append(p.return_port())
    r.append("---RIP entries---")
    r.append("port IP | destination IP address | next hop | cost")
    for re in self.rip_entries:
      r.append(re.return_rip_entry())
    return r

class router(router_base):
  def send_RIP_packets(self, routers):
    # Write your code here
    return routers

  def receive_RIP_packets(self, rip_packet, link_send_on, routers, next_hop_IP):
    # Write your code here
    return routers
```

In the next lesson, we'll look at a detailed analysis of the exercise.