```python
"""
Assignment 1: RIP protocol
Team: Bach Vu (25082165), Charlie Hunter (27380476)
Router main program/Daemon.py
"""
########## Header ##########
from daemon_sup import *
import socket, time, select
import sys, random # must use
import traceback # optional features
from router import Router, RTimer

LocalHost = "127.0.0.1"
ROUTER = None # Router Obj
SOCKETS = {} # Enabled Interfaces

########## Body ##########
def createSocket():
    for port in ROUTER.INPUT_PORTS:
        sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        sock.bind((LocalHost, port))
        SOCKETS[port] = sock

def send(mode):
    """ Send forwarding table to neighbour routers """
    for destID, link in ROUTER.OUTPUT_PORTS.items():
        table = ROUTER.get_routing_table(destID, mode)
        if len(table) == 0:
            # entry may got updated while delay & prepare package
            continue
        message = create_rip_packet(table)
        dest = (LocalHost, link[0])
        SOCKETS[link[2]].sendto(message, dest)

    print(f"Routing Table ({mode}) sent to neighbours at {strCurrTime()}.\n")
    if mode == ROUTER.REGULAR_UPDATE:
        ROUTER.reset_timer(RTimer.PERIODIC_TIMEOUT)


def send_periodic():
    mode = "None"
    if ROUTER.is_expired(RTimer.PERIODIC_TIMEOUT, getTime()):
        # Regular update
        mode = ROUTER.REGULAR_UPDATE
    elif ROUTER.has_expired_entry(getTime()):
        # Triggered update. Known NoResponse links don't trigger this
        mode = ROUTER.EXPIRED_UPDATE
    else:
        return

    send(mode)

def receive(timeout = 0.025):
    """ Return True if some data received """
    readable, _, _ = select.select(SOCKETS.values(), [], [], timeout)
    for sock in readable:
        receiver = sock.getsockname()
        data, sender = sock.recvfrom(1024)
```

```
59          if not ROUTER.is_expected_sender(sender, receiver):
60              print(f"Droped message on {sender} → {receiver} link!")
61              continue
62          routes = process_rip_packet(data)
63          print(f"Received ROUTES {str(routes)} at {strCurrTime(getTime())} from {sender}"
    )
64          triggered_update = ROUTER.update_route_table(routes, getTime())
65          if triggered_update:
66              # Next time, the record become Reset Timer
67              send(Router.FAST_ROUTE_UPDATE)
68
69  def garbage_collection():
70      ROUTER.garbage_collection(getTime())
71
72  ########## Program ##########
73  def init_router():
74      global ROUTER # include this if modifying global variable
75      filename = sys.argv[1]
76      rID, inputs, outputs, timeout = read_config(filename)
77
78      #  Router instance with default routing table
79      ROUTER = Router(rID, inputs, outputs, getTime(True), timeout)
80      ROUTER.print_hello()
81
82      # First time notice to neighbours
83      createSocket()
84
85  if __name__ == "__main__":
86      try:
87          init_router()
88          while True:
89              ROUTER.print_route_table(getTime())
90              send_periodic()
91              garbage_collection()
92              receive()
93
94      except IndexError:
95          print("Error: Config file is not provided!")
96      except FileNotFoundError:
97          print("Error: given Config file not found!")
98      except ValueError as v_err:
99          print("Warning:", v_err)
100     except socket.error as s_err:
101         print("Error:", s_err)
102     except KeyboardInterrupt:
103         print("\n******** Daemon exit successfully! Router shuting down... ********")
104     except Exception as e:
105         traceback.print_exc() # Traceback unknown error
106         print("Program exited unexpectedly.\n")
107     finally:
108         print()
109         sys.exit()
```