

```
1  '''COS364 RIP Assignment
2  Jordan Chubb
3  Vincent Zong
4  29/04/2025
5  '''
6
7  import argparse
8  from functools import partial
9  import json
10 import selectors
11 import socket
12 import time
13
14 from configmanager import read_config_file
15 from ripmanager import RipManager
16
17
18 MAX_PACKET_SIZE = 4 + 20 * 25 # header + rip entry * max number of rip entries
19
20
21 parser = argparse.ArgumentParser()
22 parser.add_argument("config", help="filename of the configuration file")
23 parser.add_argument("-d", "--debug", help="print debugging information", action= ↵
24     "store_true")
25 parser.add_argument("--autotesting", help="for automatic testing", action= ↵
26     "store_true")
27 args = parser.parse_args()
28
29 if args.autotesting:
30     """Force all print calls to flush immediately. Required for
31     automatic testing's reading of stdout.
32     """
33     print = partial(print, flush=True)
34
35 def main():
36     config = read_config_file(args.config)
37     debug(config)
38
39     sockets = get_sockets(config)
40     selector = selectors.DefaultSelector()
41     for sock in sockets:
42         selector.register(sock, selectors.EVENT_READ)
43
44     rip = RipManager(debug, config, sockets[0])
45
46     next_print_time = time.time()
47     while True:
48         next_print = max(0, next_print_time - time.time())
49         next_timeout = min(next_print, rip.next_timeout())
```

```
50         events = selector.select(timeout=next_timeout)
51
52         for key, _ in events:
53             sock = key.fileobj
54             message = sock.recv(MAX_PACKET_SIZE)
55             rip.incoming_message(message)
56             rip.send_any_updates()
57
58             if time.time() >= next_print_time:
59                 next_print_time = time.time() + 1
60                 if args.autotesting:
61                     print(json.dumps(rip.table_list()))
62                 else:
63                     print(rip)
64
65
66 def debug(line):
67     if args.debug:
68         print(line)
69
70
71 def get_sockets(config):
72     """Return a socket for each input port."""
73     sockets = []
74     for port in config.input_ports:
75         sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
76         sock.bind(('127.0.0.1', port))
77         sockets.append(sock)
78     return sockets
79
80
81 if __name__ == "__main__":
82     main()
83
```