Problem #1

Subject: UDP Socket Programming
Modules: socket, time, random, threading, protocol
Filename: network.py

WORK
The script implements UDP socket programming, initiating by setting IPv4 addressing and preparing locks for thread safety. The program consists of key functions: **find_manager**, **join_network**, and **router**.
find_manager listens on a local IP for a beacon to determine the manager's address.
join_network requests a node configuration from the manager, establishing a connection based on received port and node details.
The core **router** function activates each router in its thread. Routers continuously listen for messages, updating their forwarding tables when directed by manager messages or peer exchanges, like "NEIGHBOR" or "DUMP". Routers respond to "LINK" or "EXCHANGE" messages from peers by adjusting paths and broadcasting updates, maintaining network integrity and efficiency.
The setup supports multiple routers operating concurrently, dynamically managing network configurations and communications to reflect a robust, self-updating network system.

OUTPUT
Running on MacOS (Output was copy & pasted from MacOS Linux Terminal)

Manager.py vvvvv

```
jakobwest@Jakobs-MacBook-Air ~ % sudo python3
swe/academia/cs3080/hw07_jwest21/code/manager.py
MANAGER is listening on  127.0.0.1 20600
BEACON is broadcasting from: 127.0.0.1 21951
BEACON is broadcasting on  : 127.0.0.1 27319
Waiting for port request.
000: RECEIVED: MANAGER UNKNOWN {'HELLO': ''}
MANAGER: TO ('127.0.0.1', 23518) MESSAGE: {'USENAME':
'goqiiyqu', 'USEPORT': '40909'}
001: RECEIVED: MANAGER UNKNOWN {'HELLO': ''}
MANAGER: TO ('127.0.0.1', 23125) MESSAGE: {'USENAME':
'mpptxwpk', 'USEPORT': '45794'}
002: RECEIVED: MANAGER UNKNOWN {'HELLO': ''}
MANAGER: TO ('127.0.0.1', 23917) MESSAGE: {'USENAME':
'rxjyfryr', 'USEPORT': '48360'}
003: RECEIVED: MANAGER UNKNOWN {'HELLO': ''}
MANAGER: TO ('127.0.0.1', 23843) MESSAGE: {'USENAME':
'zijilvam', 'USEPORT': '49261'}
004: RECEIVED: MANAGER UNKNOWN {'HELLO': ''}
MANAGER: TO ('127.0.0.1', 23763) MESSAGE: {'USENAME':
'ukavkcjz', 'USEPORT': '48207'}
005: RECEIVED: MANAGER UNKNOWN {'HELLO': ''}
MANAGER: TO ('127.0.0.1', 23413) MESSAGE: {'USENAME':
'cgbirvmv', 'USEPORT': '48176'}
```

```
006: RECEIVED: MANAGER UNKNOWN {'HELLO': ''}
MANAGER: TO ('127.0.0.1', 23454) MESSAGE: {'USENAME':
'nncdakcv', 'USEPORT': '49589'}
007: RECEIVED: MANAGER UNKNOWN {'HELLO': ''}
MANAGER: TO ('127.0.0.1', 23018) MESSAGE: {'USENAME':
'kqvbvesw', 'USEPORT': '45333'}
008: RECEIVED: MANAGER UNKNOWN {'HELLO': ''}
MANAGER: TO ('127.0.0.1', 23694) MESSAGE: {'USENAME':
'tnzzqlub', 'USEPORT': '43914'}
009: RECEIVED: MANAGER UNKNOWN {'HELLO': ''}
MANAGER: TO ('127.0.0.1', 23595) MESSAGE: {'USENAME':
'qecirkmk', 'USEPORT': '41505'}
010: RECEIVED: MANAGER UNKNOWN {'HELLO': ''}
MANAGER: TO ('127.0.0.1', 23878) MESSAGE: {'USENAME':
'dyjqrdft', 'USEPORT': '42444'}
011: RECEIVED: MANAGER UNKNOWN {'HELLO': ''}
MANAGER: TO ('127.0.0.1', 23909) MESSAGE: {'USENAME':
'ahsdceoc', 'USEPORT': '47627'}
012: RECEIVED: MANAGER UNKNOWN {'HELLO': ''}
MANAGER: TO ('127.0.0.1', 23560) MESSAGE: {'USENAME':
'mvabcjvf', 'USEPORT': '45453'}
013: RECEIVED: MANAGER UNKNOWN {'HELLO': ''}
MANAGER: TO ('127.0.0.1', 23669) MESSAGE: {'USENAME':
'ickbywyy', 'USEPORT': '48793'}
014: RECEIVED: MANAGER UNKNOWN {'HELLO': ''}
MANAGER: TO ('127.0.0.1', 23376) MESSAGE: {'USENAME':
'alxelwhd', 'USEPORT': '49152'}
015: RECEIVED: MANAGER UNKNOWN {'HELLO': ''}
MANAGER: TO ('127.0.0.1', 23899) MESSAGE: {'USENAME':
'jaqyawrh', 'USEPORT': '43901'}
016: RECEIVED: MANAGER UNKNOWN {'HELLO': ''}
MANAGER: TO ('127.0.0.1', 23084) MESSAGE: {'USENAME':
'alaswayb', 'USEPORT': '43121'}
017: RECEIVED: MANAGER UNKNOWN {'HELLO': ''}
MANAGER: TO ('127.0.0.1', 23283) MESSAGE: {'USENAME':
'bdpzgthp', 'USEPORT': '43094'}
018: RECEIVED: MANAGER UNKNOWN {'HELLO': ''}
MANAGER: TO ('127.0.0.1', 23112) MESSAGE: {'USENAME':
'gayurzba', 'USEPORT': '45093'}
019: RECEIVED: MANAGER UNKNOWN {'HELLO': ''}
MANAGER: TO ('127.0.0.1', 23701) MESSAGE: {'USENAME':
'xgavkvxt', 'USEPORT': '45868'}
020: RECEIVED: MANAGER UNKNOWN {'HELLO': ''}
MANAGER: TO ('127.0.0.1', 23345) MESSAGE: {'USENAME':
'wmxssjws', 'USEPORT': '43389'}
021: RECEIVED: MANAGER UNKNOWN {'HELLO': ''}
MANAGER: TO ('127.0.0.1', 23840) MESSAGE: {'USENAME':
'tyabyxru', 'USEPORT': '40630'}
```

```
022: RECEIVED: MANAGER UNKNOWN {'HELLO': ''}
MANAGER: TO ('127.0.0.1', 23494) MESSAGE: {'USENAME':
'avjctclu', 'USEPORT': '49449'}
023: RECEIVED: MANAGER UNKNOWN {'HELLO': ''}
MANAGER: TO ('127.0.0.1', 23020) MESSAGE: {'USENAME':
'ctitxnpm', 'USEPORT': '47662'}
024: RECEIVED: MANAGER UNKNOWN {'HELLO': ''}
MANAGER: TO ('127.0.0.1', 23904) MESSAGE: {'USENAME':
'rbkgkwni', 'USEPORT': '43671'}
025: RECEIVED: MANAGER UNKNOWN {'HELLO': ''}
MANAGER: TO ('127.0.0.1', 23176) MESSAGE: {'USENAME':
'mtdtsinx', 'USEPORT': '40940'}
026: RECEIVED: MANAGER UNKNOWN {'HELLO': ''}
MANAGER: TO ('127.0.0.1', 23163) MESSAGE: {'USENAME':
'soacaaqq', 'USEPORT': '41168'}
027: RECEIVED: MANAGER UNKNOWN {'HELLO': ''}
MANAGER: TO ('127.0.0.1', 23591) MESSAGE: {'USENAME':
'onfwbzix', 'USEPORT': '40496'}
028: RECEIVED: MANAGER UNKNOWN {'HELLO': ''}
MANAGER: TO ('127.0.0.1', 23157) MESSAGE: {'USENAME':
'osszaddq', 'USEPORT': '49494'}
029: RECEIVED: MANAGER UNKNOWN {'HELLO': ''}
MANAGER: TO ('127.0.0.1', 23256) MESSAGE: {'USENAME':
'mltxkcgw', 'USEPORT': '45935'}
030: RECEIVED: MANAGER UNKNOWN {'HELLO': ''}
MANAGER: TO ('127.0.0.1', 23885) MESSAGE: {'USENAME':
'yfutvjvt', 'USEPORT': '46379'}
031: RECEIVED: MANAGER UNKNOWN {'HELLO': ''}
MANAGER: TO ('127.0.0.1', 23276) MESSAGE: {'USENAME':
'bmvjakrn', 'USEPORT': '48948'}
Network appears to have settled with 32 nodes.
Configuring network
Ensure network is fully connected.
Pick random nodes to connect to.
Configuring nodes
Gathering Route Tables
032: RECEIVED: goqiiyqu MANAGER {'45794': "{'distance': 1}",
'47627': "{'distance': 1}"}
032: RECEIVED: rbkgkwni MANAGER {'46379': "{'distance': 1}",
'40940': "{'distance': 1}"}
032: RECEIVED: zijilvam MANAGER {'49589': "{'distance': 1}",
'48207': "{'distance': 1}"}
032: RECEIVED: nncdakcv MANAGER {'49152': "{'distance': 1}",
'45333': "{'distance': 1}"}
032: RECEIVED: cgbirvmv MANAGER {'49589': "{'distance': 1}",
'48207': "{'distance': 1}"}
032: RECEIVED: kqvbvesw MANAGER {'43914': "{'distance': 1}",
'47627': "{'distance': 1}"}
```

```
032: RECEIVED: ahsdceoc MANAGER {'49261': "{'distance': 1}",
'45453': "{'distance': 1}"}
032: RECEIVED: ickbywyy MANAGER {'49152': "{'distance': 1}",
'48948': "{'distance': 1}"}
032: RECEIVED: mvabcjvf MANAGER {'48793': "{'distance': 1}",
'49494': "{'distance': 1}"}
032: RECEIVED: alaswayb MANAGER {'48360': "{'distance': 1}",
'43094': "{'distance': 1}"}
032: RECEIVED: bdpzgthp MANAGER {'45093': "{'distance': 1}",
'43671': "{'distance': 1}"}
032: RECEIVED: jaqyawrh MANAGER {'43121': "{'distance': 1}",
'40940': "{'distance': 1}"}
032: RECEIVED: xgavkvxt MANAGER {'49449': "{'distance': 1}",
'43389': "{'distance': 1}"}
032: RECEIVED: wmxssjws MANAGER {'47662': "{'distance': 1}",
'40630': "{'distance': 1}"}
032: RECEIVED: avjctclu MANAGER {'47627': "{'distance': 1}",
'47662': "{'distance': 1}"}
032: RECEIVED: ctitxnpm MANAGER {'48207': "{'distance': 1}",
'43671': "{'distance': 1}"}
032: RECEIVED: mtdtsinx MANAGER {'41168': "{'distance': 1}",
'48793': "{'distance': 1}"}
032: RECEIVED: onfwbzix MANAGER {'42444': "{'distance': 1}",
'49494': "{'distance': 1}"}
032: RECEIVED: mltxkcgw MANAGER {'49449': "{'distance': 1}",
'46379': "{'distance': 1}"}
032: RECEIVED: osszaddq MANAGER {'40909': "{'distance': 1}",
'45935': "{'distance': 1}"}
032: RECEIVED: rxjyfryr MANAGER {'49261': "{'distance': 1}",
'47662': "{'distance': 1}"}
032: RECEIVED: bmvjakrn MANAGER {'40909': "{'distance': 1}",
'45935': "{'distance': 1}"}
032: RECEIVED: mpptxwpk MANAGER {'48360': "{'distance': 1}",
'43094': "{'distance': 1}"}
032: RECEIVED: ukavkcjz MANAGER {'48176': "{'distance': 1}",
'41505': "{'distance': 1}"}
032: RECEIVED: tnzzqlub MANAGER {'41505': "{'distance': 1}",
'47627': "{'distance': 1}"}
032: RECEIVED: alxelwhd MANAGER {'43901': "{'distance': 1}",
'47662': "{'distance': 1}"}
032: RECEIVED: gayurzba MANAGER {'45868': "{'distance': 1}"}
032: RECEIVED: tyabyxru MANAGER {'43121': "{'distance': 1}",
'49449': "{'distance': 1}"}
032: RECEIVED: soacaaqq MANAGER {'40496': "{'distance': 1}",
'42444': "{'distance': 1}"}
032: RECEIVED: dyjqrdft MANAGER {'47627': "{'distance': 1}",
'47662': "{'distance': 1}"}
```

```
032: RECEIVED: qecirkmk MANAGER {'42444': "{'distance': 1}",
'43671': "{'distance': 1}"}
032: RECEIVED: yfutvjvt MANAGER {'45794': "{'distance': 1}",
'48948': "{'distance': 1}"}
```

network.py vvvvv

```
jakobwest@Jakobs-MacBook-Air ~ % python3
swe/academia/cs3080/hw07_jwest21/code/network.py
00 [goqiiyqu] PORT: 40909
01 [mpptxwpk] PORT: 45794
02 [rxjyfryr] PORT: 48360
03 [zijilvam] PORT: 49261
04 [ukavkcjz] PORT: 48207
05 [cgbirvmv] PORT: 48176
06 [nncdakcv] PORT: 49589
07 [kqvbvesw] PORT: 45333
08 [tnzzqlub] PORT: 43914
09 [qecirkmk] PORT: 41505
10 [dyjqrdft] PORT: 42444
11 [ahsdceoc] PORT: 47627
12 [mvabcjvf] PORT: 45453
13 [ickbywyy] PORT: 48793
14 [alxelwhd] PORT: 49152
15 [jaqyawrh] PORT: 43901
16 [alaswayb] PORT: 43121
17 [bdpzgthp] PORT: 43094
18 [gayurzba] PORT: 45093
19 [xgavkvxt] PORT: 45868
20 [wmxssjws] PORT: 43389
21 [tyabyxru] PORT: 40630
22 [avjctclu] PORT: 49449
23 [ctitxnpm] PORT: 47662
24 [rbkgkwni] PORT: 43671
25 [mtdtsinx] PORT: 40940
26 [soacaaqq] PORT: 41168
27 [onfwbzix] PORT: 40496
28 [osszaddq] PORT: 49494
29 [mltxkcgw] PORT: 45935
30 [yfutvjvt] PORT: 46379
31 [bmvjakrn] PORT: 48948
```

CODE

```
'''
PROGRAMMER: Jakob K. West
USERNAME: jwest21
PROGRAM: network.py

DESCRIPTION: UDP Socket Programming -
Given protocol.py and manager.py complete
the python script for network.py
'''

import socket
import random
import threading
import protocol
import time


ipv4_addr = protocol.get_myIP()

setup_lock = threading.Lock()
print_lock = threading.Lock()


def tprint(s):

    print_lock.acquire()
    print(s)
    print_lock.release()


def find_manager():

    # Establish initial contact with BEACON to get port number for manager
    beacon_port = protocol.beacon_port

    setup_lock.acquire() # Needed to share beacon port on local_host

    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    sock.bind((ipv4_addr, beacon_port))

    beacon_found = False
    while not beacon_found:
        data, addr = sock.recvfrom(4096)
        (recipient, sender, message) = protocol.depacketize(data)
        if "BEACON" == sender:
            if "MANAGER" in message:
                manager_port = int(message["MANAGER"])
                manager_addr = (addr[0], manager_port)
                beacon_found = True
    sock.close()
    setup_lock.release()
    return manager_addr


def join_network(manager_address):

    # Use a random port number until the manager assigns a permanent one
    port = random.randrange(23000, 24000)
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    sock.bind((ipv4_addr, port))

    # Request a node name and port number from the manager
    node_port = -1
    while node_port < 0:
        datagram = protocol.packetize("MANAGER", "UNKNOWN", {"HELLO":''})
        sock.sendto(datagram, manager_address)
        datagram, addr = sock.recvfrom(4096)
        (recipient, sender, message) = protocol.depacketize(datagram)
        if ("USENAME" in message) and ("USEPORT" in message):
            node_name = message["USENAME"]
            node_port = int(message["USEPORT"])
    sock.close()

    return (node_name, node_port)


def router(router_number):

    forwarding_table = {}

    # Find the manager and join the network
    manager_address = find_manager()
    (node_name, node_port) = join_network(manager_address)

    tprint("%02d [%s] PORT: %5d" % (router_number, node_name, node_port))

    # Create the final socket for the router on the assigned port
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    sock.bind((ipv4_addr, node_port))

    while True:

        # Wait for a message
        datagram, addr = sock.recvfrom(4096)
        (recipient, sender, message) = protocol.depacketize(datagram)

        if sender == "MANAGER":

            if "NEIGHBOR" in message:
                neighbor_port = message["NEIGHBOR"]
                forwarding_table[neighbor_port] = {"distance": 1} # Direct connection
                # Prepare and send LINK message
                link_msg = protocol.packetize(node_name, "UNKNOWN", {"LINK":
f"{node_port}"})
                sock.sendto(link_msg, (ipv4_addr, int(neighbor_port)))
            elif "DUMP" in message:
                # Prepare and send TABLE message
                table_msg = {node: data for node, data in forwarding_table.items()}
                sock.sendto(protocol.packetize(node_name, "MANAGER", table_msg),
manager_address)

            else: # Message from a neighbor

                if "LINK" in message:
                    neighbor_name = sender # Assuming sender name is valid

                if neighbor_name not in forwarding_table or
forwarding_table[neighbor_name]['distance'] > 2:
                    forwarding_table[neighbor_name] = {"distance": 2} # Indirect
connection via one hop
                    # Respond with a LINK message if the recipient was "UNKNOWN"
                    if recipient == "UNKNOWN":
                        response_link_msg = protocol.packetize(node_name,
neighbor_name, {"LINK": f"{node_port}"})
                        sock.sendto(response_link_msg, (ipv4_addr, addr[1]))

                elif "EXCHANGE" in message:
                    # Processing EXCHANGE message
                    modified = False
                    for entry in message:
                        node, dist = entry.split(',')
                        dist = int(dist)
                        if node not in forwarding_table or
forwarding_table[node]['distance'] > dist + 1:
                            forwarding_table[node] = {"distance": dist + 1}
                            modified = True

                    # If the forwarding table was modified, send a table summary to
each neighbor
                    if modified:
                        for neighbor in forwarding_table:
                            summary_msg = {node: data for node, data in
forwarding_table.items()}
                            sock.sendto(protocol.packetize(node_name, "UNKNOWN",
summary_msg), (ipv4_addr, int(neighbor)))


if __name__ == '__main__':

    nodes_in_network = 32
    threads = []
    for i in range(nodes_in_network):
        threads.append(threading.Thread(target = router, args = [i]))
    for thread in threads:
        thread.start()

# Jakob West
```