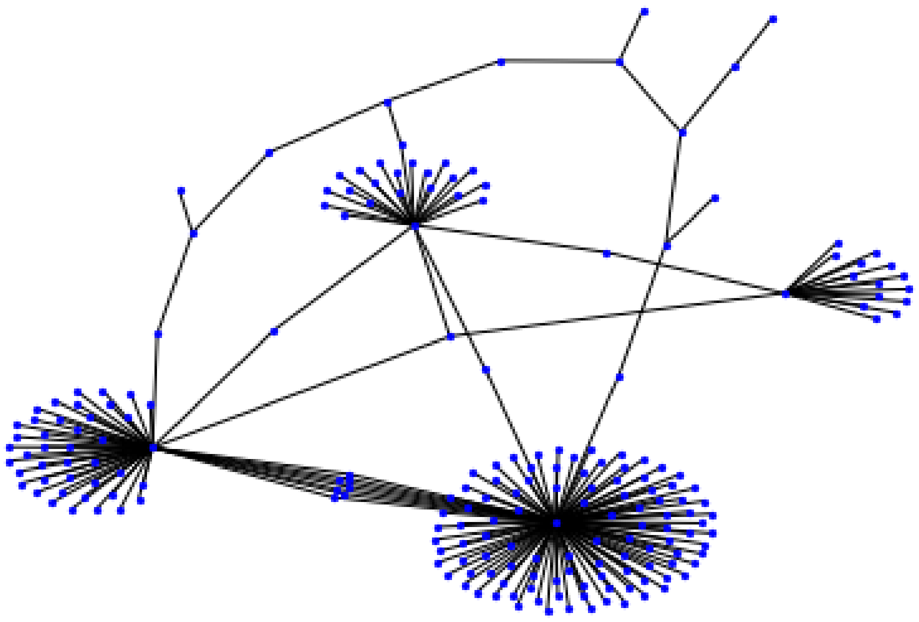
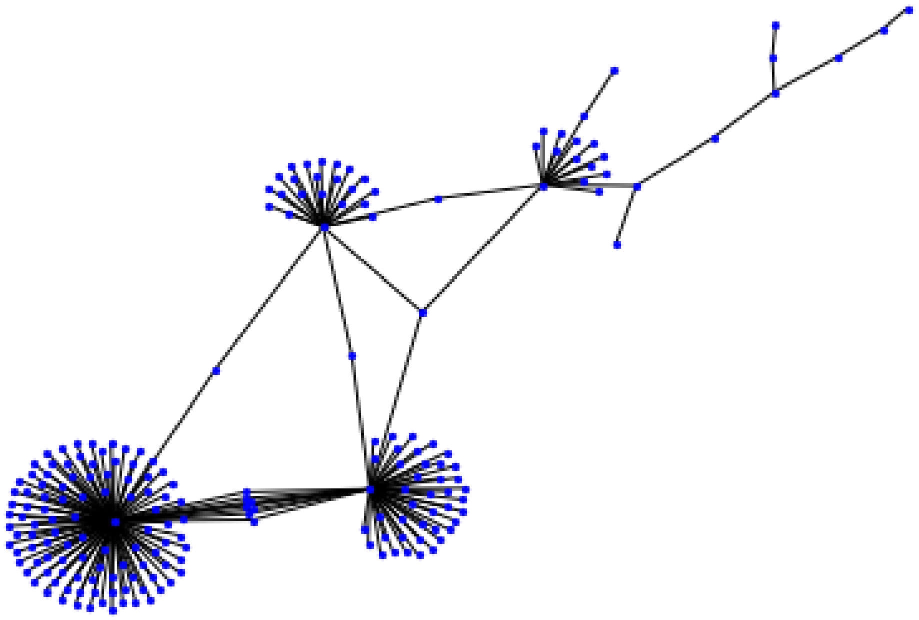
assignment5

November 3, 2021

[1]:
```python
import networkx as nx
import csv
import matplotlib.pyplot as plt
import community


total_number_packets = 0
total_size_packets = 0
source_ips = set([])
dest_ips = set([])
devices = set([])

G_MAC_IP = nx.Graph()
G_IP = nx.Graph()

with open("viber.csv", "r") as viber_csv_file:
    read_data = csv.reader(viber_csv_file, delimiter=",", quotechar='"')

    header = next(read_data)
    for i in range(0, len(header)):
        print(i, header[i])

    for line in read_data:
        ip_protocol = line[13]
        if not (ip_protocol == "6" or ip_protocol == "17"):
            continue
        total_number_packets += 1
        size_of_current_packet = int(line[2])
        total_size_packets += size_of_current_packet

        ip_src = line[11]
        ip_dst = line[12]
        mac_src = line[3]
        mac_dst = line[4]
        if ip_src not in source_ips:
            source_ips.add(ip_src)
        if ip_dst not in dest_ips:
            dest_ips.add(ip_dst)
        if mac_dst not in devices:
            devices.add(mac_dst)
        if mac_src not in devices:
            devices.add(mac_src)

        G_MAC_IP.add_edge(mac_src, ip_dst)
        G_IP.add_edge(ip_src, ip_dst)

print("total size of transmitted packets:", total_size_packets)
print("total amount of transmitted packets:", total_number_packets)
print("total amount of devices:", len(devices))
print("total amount of source IPs:", len(source_ips))
print("total amount of destination IPs:", len(dest_ips))

# 2. Visualize local communication network between IPs and MACs.
nx.draw(G_MAC_IP, node_color="blue", node_size=5)
plt.savefig("networks_IP_MAC.png")
plt.show()


# 3. Visualize local communication network between IPs.
nx.draw(G_IP, node_color="blue", node_size=5)
plt.savefig("networks_IP.png")
plt.show()


# 4. Identify Viber servers according to the protocol DNS
list_domain_ip = []

with open("v-dns.csv", "r") as dns_csv_file:
    data = csv.reader(dns_csv_file, delimiter=",", quotechar='"')

    header = next(data)
    for i in range(0, len(header)):
        print(i, header[i])

    viber_string = "viber.com"
    for line in data:
        info = line[6]
        source_ip = line[2]
        words_from_info = info.split(" ")
        contains_viber = False
        viber_domain = ""

        if "Standard query response" in info:
            continue
        for word in words_from_info:
            if viber_string in word:
                contains_viber = True
                viber_domain = word
        if contains_viber:
            domain_ip_pair = [viber_domain, source_ip]
            list_domain_ip.append(domain_ip_pair)
            print(domain_ip_pair)
```

```
0 frame.number
1 frame.time_relative
2 frame.len
3 eth.src
4 eth.dst
5 eth.type
6 arp.opcode
7 arp.src.hw_mac
8 arp.src.proto_ipv4
9 arp.dst.hw_mac
10 arp.dst.proto_ipv4
11 ip.src
12 ip.dst
13 ip.proto
14 tcp.srcport
15 tcp.dstport
16 udp.srcport
17 udp.dstport
total size of transmitted packets: 86696837
total amount of transmitted packets: 137878
total amount of devices: 12
total amount of source IPs: 180
total amount of destination IPs: 200
```





```
0 No.
1 Time
2 Source
3 Destination
4 Protocol
5 Length
6 Info
['content.cdn.viber.com', '192.168.150.2']
['secure.viber.com', '192.168.150.2']
['secure.viber.com', '192.168.150.2']
['aloha.viber.com', '192.168.150.2']
['www.cdn.viber.com', '192.168.150.2']
['aloha.viber.com', '192.168.150.2']
['market.viber.com', '192.168.150.2']
['share.viber.com', '192.168.150.2']
['content.cdn.viber.com', '192.168.150.2']
['share.viber.com', '192.168.150.2']
['aloha.viber.com', '192.168.150.2']
['content.cdn.viber.com', '192.168.150.2']
['content.cdn.viber.com', '192.168.150.2']
['share.viber.com', '192.168.73.13']
['www.cdn.viber.com', '192.168.73.13']
['share.viber.com', '192.168.150.2']
['share.viber.com', '192.168.150.2']
['share.viber.com', '192.168.73.13']
['share.viber.com', '192.168.150.2']
['share.viber.com', '192.168.150.2']
['aloha.viber.com', '192.168.73.13']
['content.cdn.viber.com', '192.168.73.13']
['aloha.viber.com', '192.168.73.13']
['content.cdn.viber.com', '192.168.73.13']
```

[2]:
```python
import networkx as nx
import csv
import matplotlib.pyplot as plt
import community
import seaborn as sns
# pip install community python-louvain
# pip3 install ...
```

[3]:
```python
G_IP = nx.Graph()

dest_ips_set = set([])

with open("viber.csv", "r") as viber_csv_file:
    read_data = csv.reader(viber_csv_file, delimiter=",", quotechar='"')
    header = next(read_data)

    for line in read_data:
        ip_protocol = line[13]
        if not (ip_protocol == "6" or ip_protocol == "17"):
            continue
        size_of_current_packet = int(line[2])
        ip_src = line[11]
        ip_dst = line[12]

        if ip_dst not in dest_ips_set:
            dest_ips_set.add(ip_dst)
        G_IP.add_edge(ip_src, ip_dst)
```

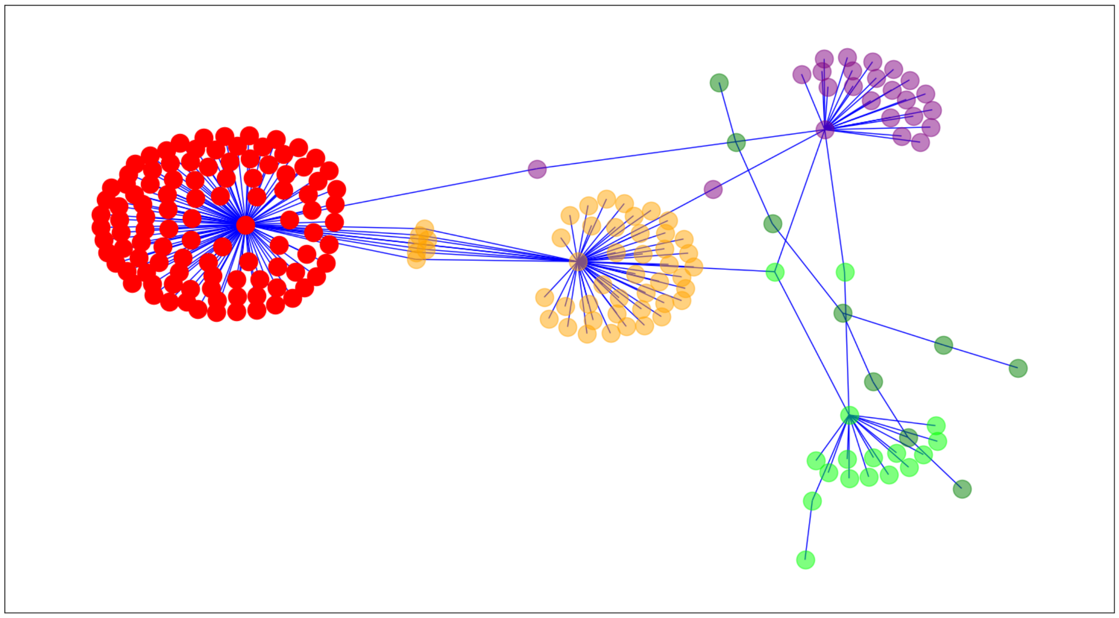
[5]:
```
# 1)(c)
# compute the length of dest.ips_set

# 1(d)
louvain_result = community.best_partition(G_IP)
# key: ip adress
# value: (arbitrary) community number

list_of_ips = []
list_of_comm = []
for key, value in louvain_result.items():
    list_of_ips.append(key)
    list_of_comm.append(value)
print("Number of different communities:", max(list_of_comm)+1)
```

```
Number of different communities: 5
```

[6]:
```
community0 = [list_of_ips[i] for i in range(len(list_of_ips))
              if list_of_comm[i] == 0]
```

[17]:
```
plt.figure(figsize=(19.2, 10.8), dpi=100)
pos = nx.spring_layout(G_IP)
nx.draw_networkx_edges(G_IP, pos=pos, node_size=1.5, edge_color="b")

nx.draw_networkx_nodes(community0, pos=pos, node_color="red")
cmm_colors = ["red", "green", "purple", "orange", "lime"]
for cmmnt in range(1, 5):
    community_tmp = [list_of_ips[i] for i in range(len(list_of_ips))
              if list_of_comm[i] == cmmnt]
    nx.draw_networkx_nodes(community_tmp, pos=pos,
                           node_color=cmm_colors[cmmnt],
                          alpha=0.5)
```

```
plt.show()
```



[ ]:
```
#range(5) = [0, 1, 2, 3, 4] == range(0, 5, 1)
#range(1, 5)
#range(1, 10, 3) = [1, 4, 7]
```

[18]:
```
# 2)
int_ips = []
for ip in list_of_ips:
    edges = G_IP.edges(ip)
    for edge in edges:
        node1 = edge[0]
        node2 = edge[1]
        if not louvain_result[node1] == louvain_result[node2]:
            int_ips.append(node1)
            int_ips.append(node2)
```

```
[('103.10.4.158', '192.168.73.12')]
```

[ ]:
```

```