Multithreaded News Client/Server Information System

Project Description

The Multithreaded News Client/Server is a simple Information System that aims to enable a secure exchange of information about recent news between clients and the server. The server retrieves the news from https://newsapi.org/ depending on the client's request, where it can manage the connection with multiple clients in the same time. However, the system provides the user with a menu so it can choose between headlines and sources easily and provide a detailed response if required.

Semester

S1 2024-2025

Group

Group name: B4 **Course Code:** ITNE352
Section: 02
Students: Israa Isa Ahmed Altaitoon (ID: 202206492)
Zainab Hasan Isa Alobed (ID: 202206986)

Table of Contents

Requirements

Follow these steps to set up the project locally:

Any disruption in the internet connection will prevent the system from functioning properly
Ensure you are connected to the internet so API can work successfully

1. Clone the repository:

```
git clone https://github.com/Zainab-Alobed/ITNE352-Project-Group-B4
```

2. Install required libraries:

```
pip install -r required.txt
```

3. Run the server.py

```
python server.py
```

4. Run the client.py

```
python client.py
```

How to run the system:

Run the server:

1. Navigate to the server directory:

```
cd server
```

2. Start the server:

```
python server.py
```

Run the client:

1. Navigate to the client directory:

```
cd client
```

2. Start the client:

```
python client.py
```

Interacting with the server:

1. The user will be asked about his name, and send it to the server
2. The main menu will be displayed in client side that contains three options (the user must input a valid number 1-3):

1. Headlines
2. Sources
3. Quit

3. A submenu of either headlines or sources will be displayed depending on the user choice

4. Later on, a maximum of 15 article will be displayed to provide the user the ability to request detailed information or go back to the main menu (Each request will be directly send to the server and print the response back in the client side)

5. The user can select (3) Quit to terminate the program

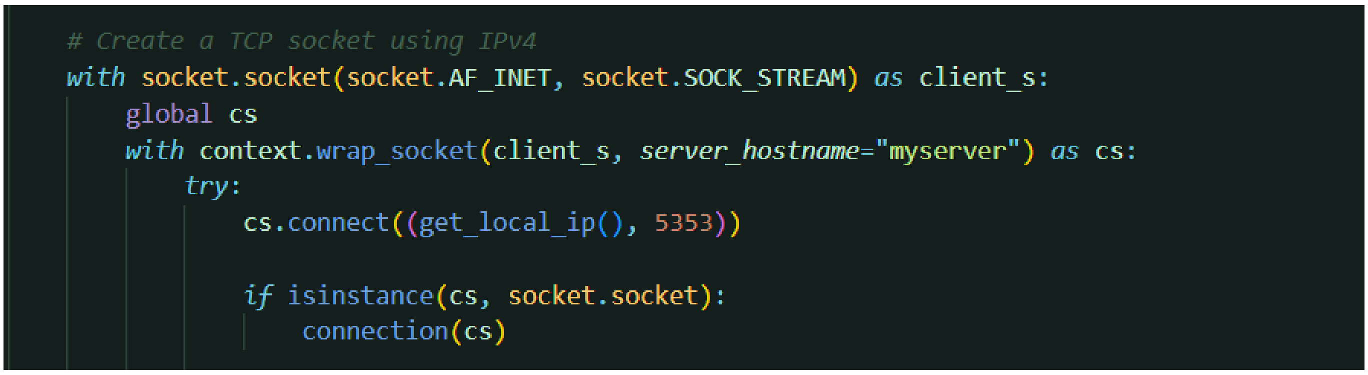
The scripts

Client script

- Purpose:
  interaction with server (Sends the user requests to the server and displays response)

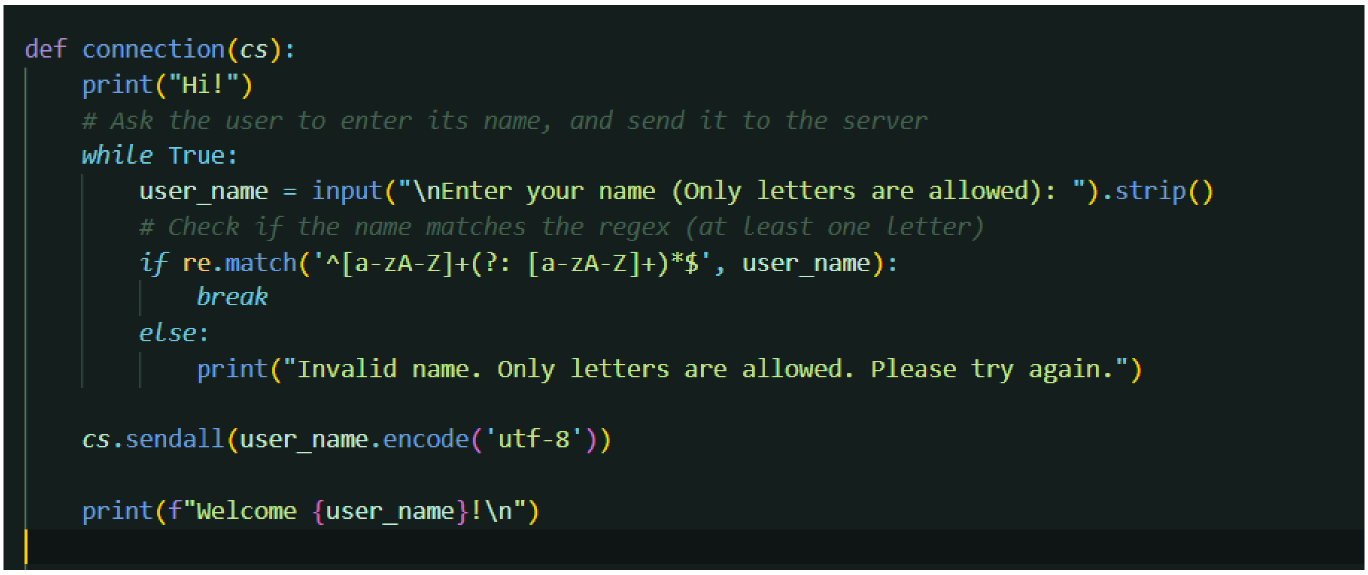
- Functions:

1. Create a TCP socket using IPv4 to connect to the server



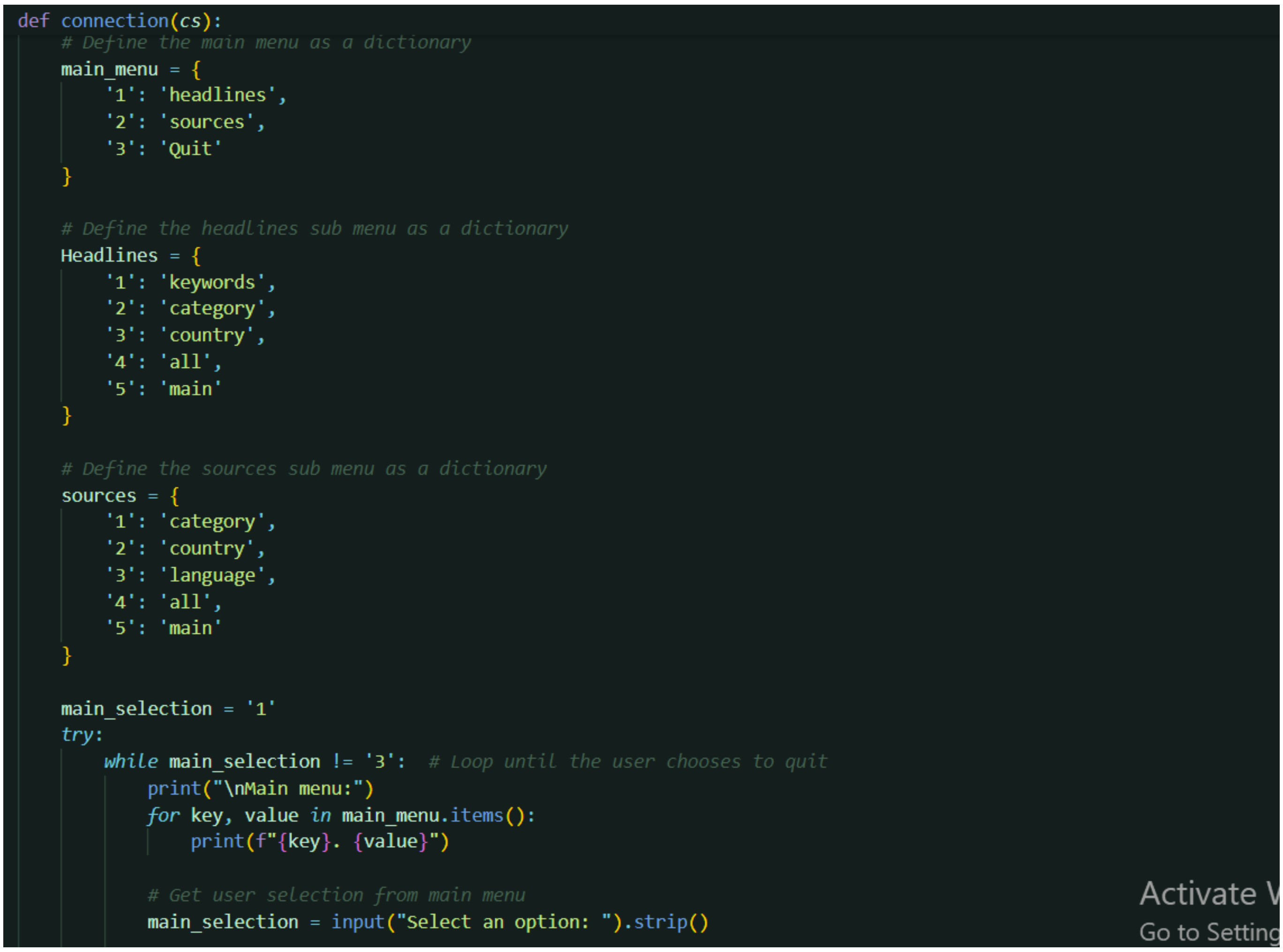
```python
# Create a TCP socket using IPv4
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as client_s:
    global cs
    with context.wrap_socket(client_s, server_hostname="myserver") as cs:
        try:
            cs.connect((get_local_ip(), 5353))

            if isinstance(cs, socket.socket):
                connection(cs)
```

2. Ask user about his name and send it to the server



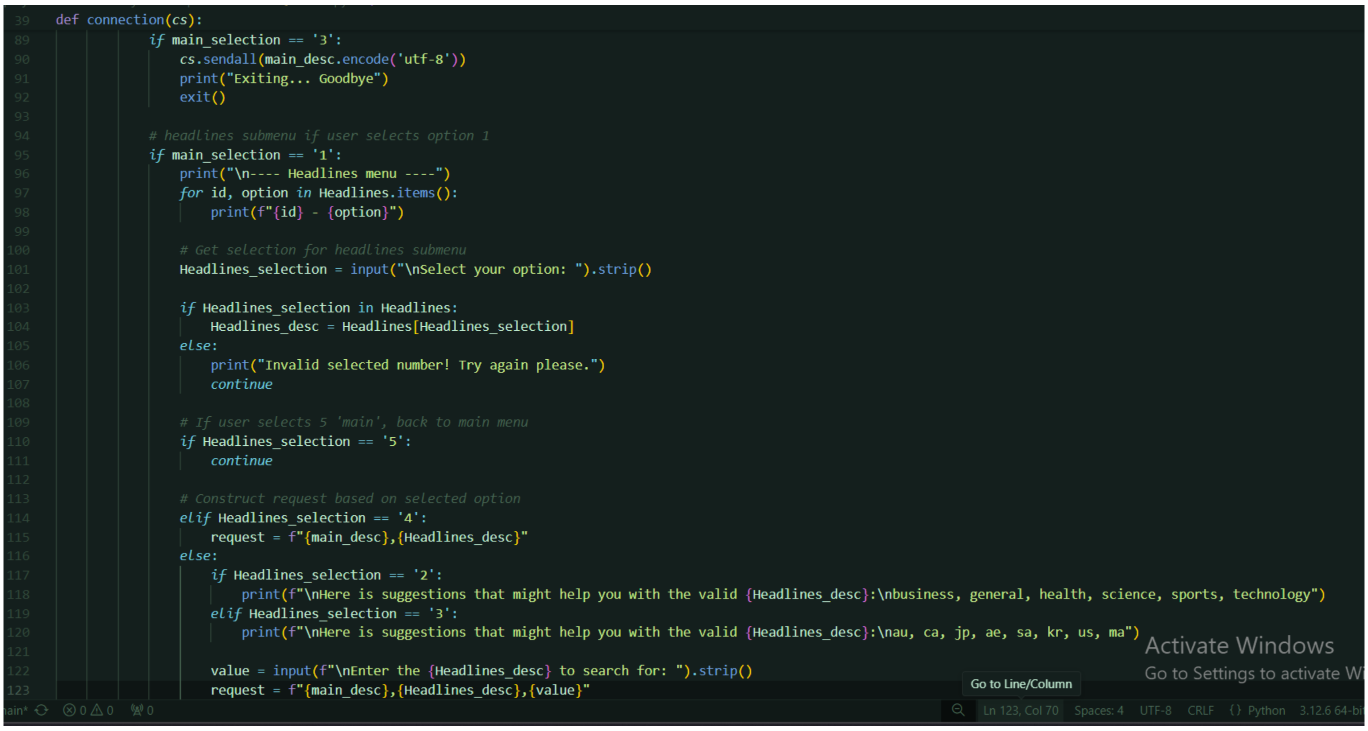
```python
def connection(cs):
    print("Hi!")
    # Ask the user to enter its name, and send it to the server
    while True:
        user_name = input("\nEnter your name (Only letters are allowed): ").strip()
        # Check if the name matches the regex (at least one letter)
        if re.match('^[a-zA-Z]+(?: [a-zA-Z]+)*$', user_name):
            break
        else:
            print("Invalid name. Only letters are allowed. Please try again.")

    cs.sendall(user_name.encode('utf-8'))

    print(f"Welcome {user_name}!\n")
```

3. Display the main menu

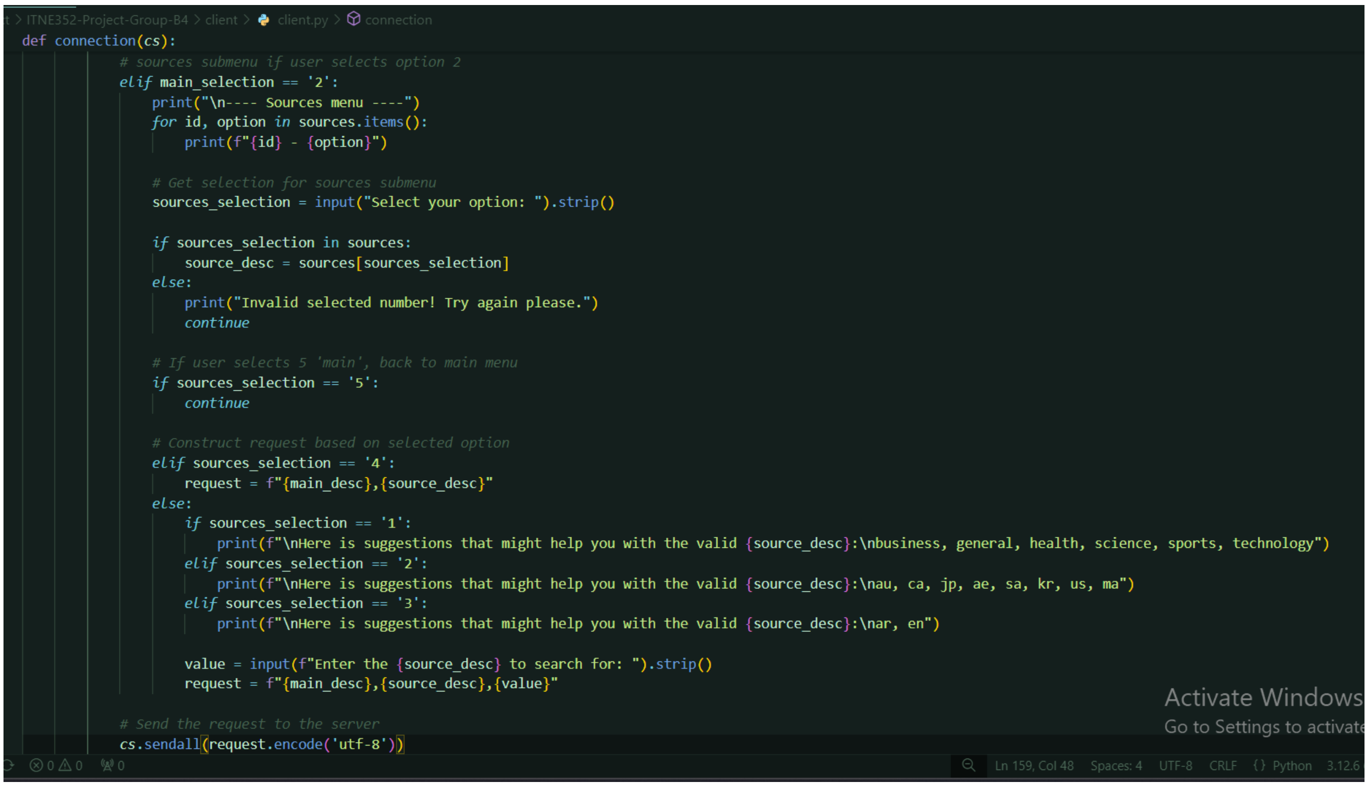


```python
def connection(cs):
    # Define the main menu as a dictionary
    main_menu = {
        '1': 'headlines',
        '2': 'sources',
        '3': 'Quit'
    }

    # Define the headlines sub menu as a dictionary
    Headlines = {
        '1': 'keywords',
        '2': 'category',
        '3': 'country',
        '4': 'all',
        '5': 'main'
    }

    # Define the sources sub menu as a dictionary
    sources = {
        '1': 'category',
        '2': 'country',
        '3': 'language',
        '4': 'all',
        '5': 'main'
    }

    main_selection = '1'
    try:
        while main_selection != '3':  # Loop until the user chooses to quit
            print("\nMain menu:")
            for key, value in main_menu.items():
                print(f"{key}. {value}")

            # Get user selection from main menu
            main_selection = input("Select an option: ").strip()
```

4. Display either (1) Headlines menu or (2) Sources menu depending on the user choice. Then, Send the request and recieve/display the response from the server (allow the request of detailed response)



```python
def connection(cs):
        if main_selection == '3':
            cs.sendall(main_desc.encode('utf-8'))
            print("Exiting... Goodbye")
            exit()

        # headlines submenu if user selects option 1
        if main_selection == '1':
            print("\n---- Headlines menu ----")
            for id, option in Headlines.items():
                print(f"{id} - {option}")

            # Get selection for headlines submenu
            Headlines_selection = input("\nSelect your option: ").strip()

            if Headlines_selection in Headlines:
                Headlines_desc = Headlines[Headlines_selection]
            else:
                print("Invalid selected number! Try again please.")
                continue

            # If user selects 5 'main', back to main menu
            if Headlines_selection == '5':
                continue

            # Construct request based on selected option
            elif Headlines_selection == '4':
                request = f"{main_desc},{Headlines_desc}"
            else:
                if Headlines_selection == '2':
                    print(f"\nHere is suggestions that might help you with the valid {Headlines_desc}:\nbusiness, general, health, science, sports, technology")
                elif Headlines_selection == '3':
                    print(f"\nHere is suggestions that might help you with the valid {Headlines_desc}:\nau, ca, jp, ae, sa, kr, us, ma")

                value = input(f"\nEnter the {Headlines_desc} to search for: ").strip()
                request = f"{main_desc},{Headlines_desc},{value}"
```



```python
def connection(cs):
        # sources submenu if user selects option 2
        elif main_selection == '2':
            print("\n---- Sources menu ----")
            for id, option in sources.items():
                print(f"{id} - {option}")

            # Get selection for sources submenu
            sources_selection = input("Select your option: ").strip()

            if sources_selection in sources:
                source_desc = sources[sources_selection]
            else:
                print("Invalid selected number! Try again please.")
                continue

            # If user selects 5 'main', back to main menu
            if sources_selection == '5':
                continue

            # Construct request based on selected option
            elif sources_selection == '4':
                request = f"{main_desc},{source_desc}"
            else:
                if sources_selection == '1':
                    print(f"\nHere is suggestions that might help you with the valid {source_desc}:\nbusiness, general, health, science, sports, technology")
                elif sources_selection == '2':
                    print(f"\nHere is suggestions that might help you with the valid {source_desc}:\nau, ca, jp, ae, sa, kr, us, ma")
                elif sources_selection == '3':
                    print(f"\nHere is suggestions that might help you with the valid {source_desc}:\nar, en")

                value = input(f"Enter the {source_desc} to search for: ").strip()
                request = f"{main_desc},{source_desc},{value}"

        # Send the request to the server
        cs.sendall(request.encode('utf-8'))
```

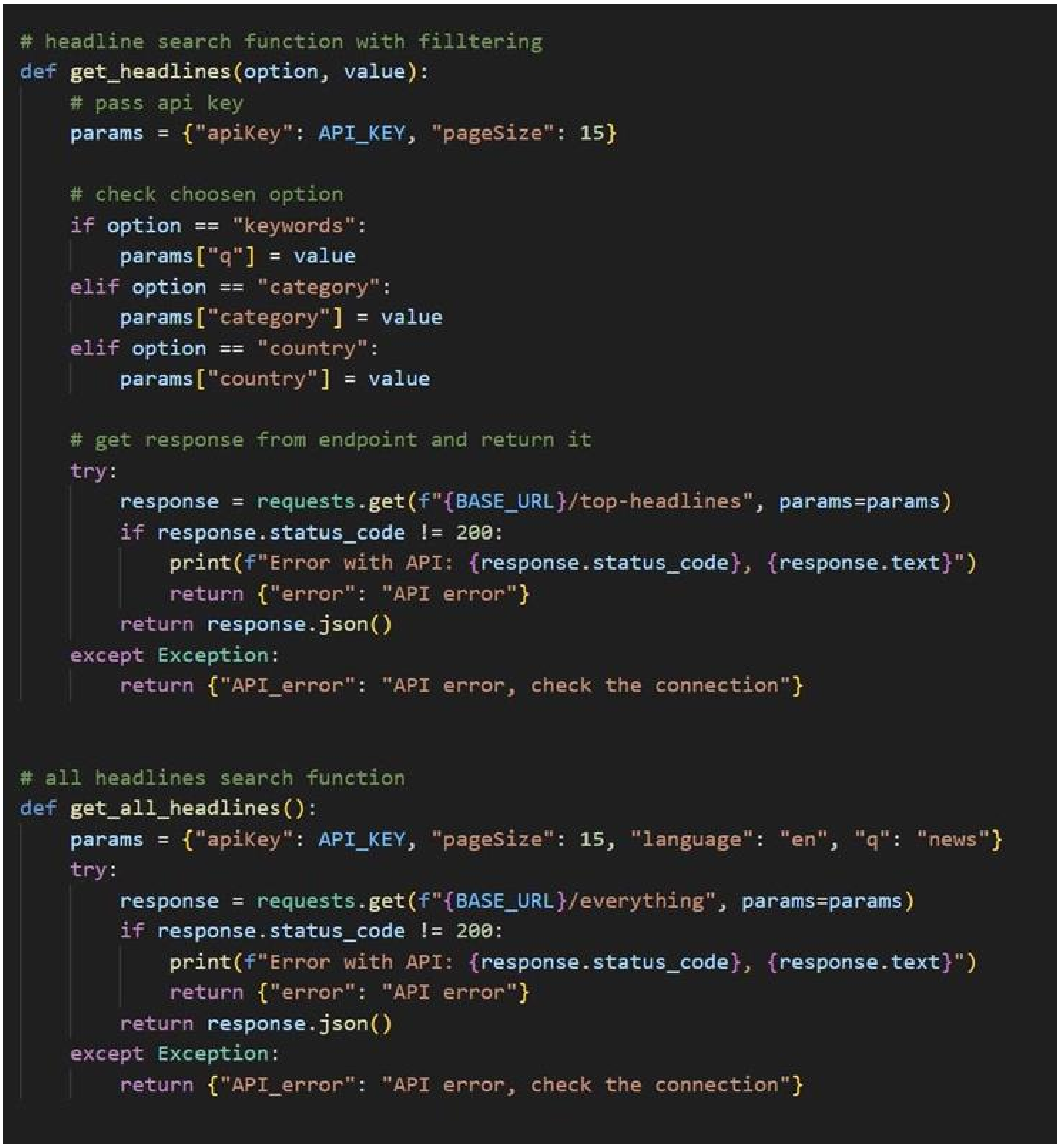
Server script

- Purpose:
  The server receives requests from the client for headlines/sources from newsapi.
  The server then will retrieve the requested data from an appropriate API endpoint. After getting the response, the server will save the response into a file named client name and the requested option then prepare a list containing brief information about a maximum of 15 headlines/sources and send it to the client.
  The client can choose a specific headline/source from the list to get more information about it, which will also prepared by the server. The server will keep getting requests from a maximum of 3 clients.

- packages: re, socket, json, threading, requests, ssl, os

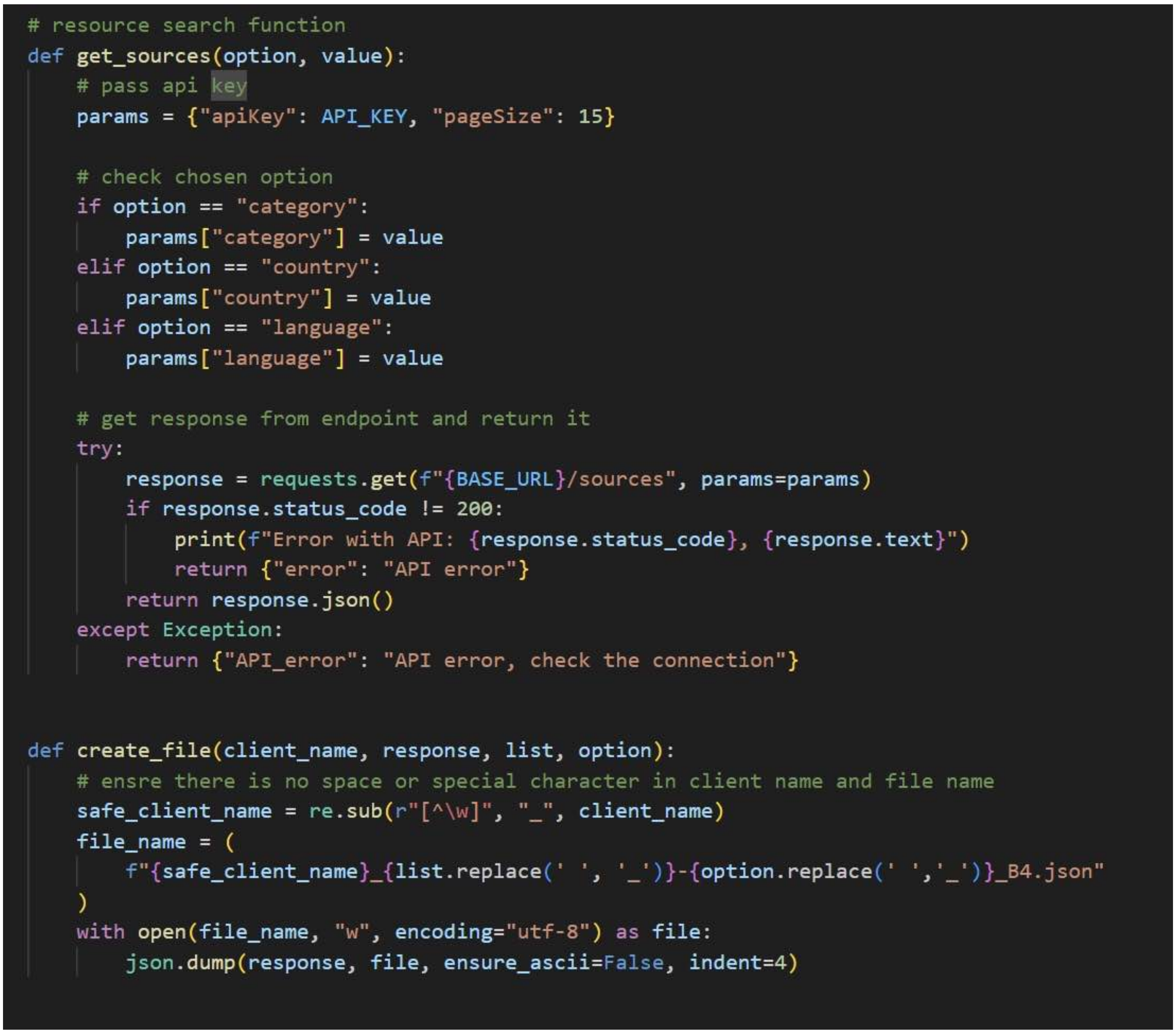
- Functions:

1. get_headlines + get_all_headlines
   Those two functions are responsible for retrieving headlines from API endpoints /everything and //top-headlines



```python
# headline search function with filltering
def get_headlines(option, value):
    # pass api key
    params = {"apiKey": API_KEY, "pageSize": 15}

    # check choosen option
    if option == "keywords":
        params["q"] = value
    elif option == "category":
        params["category"] = value
    elif option == "country":
        params["country"] = value

    # get response from endpoint and return it
    try:
        response = requests.get(f"{BASE_URL}/top-headlines", params=params)
        if response.status_code != 200:
            print(f"Error with API: {response.status_code}, {response.text}")
            return {"error": "API error"}
        return response.json()
    except Exception:
        return {"API_error": "API error, check the connection"}


# all headlines search function
def get_all_headlines():
    params = {"apiKey": API_KEY, "pageSize": 15, "language": "en", "q": "news"}
    try:
        response = requests.get(f"{BASE_URL}/everything", params=params)
        if response.status_code != 200:
            print(f"Error with API: {response.status_code}, {response.text}")
            return {"error": "API error"}
        return response.json()
    except Exception:
        return {"API_error": "API error, check the connection"}
```

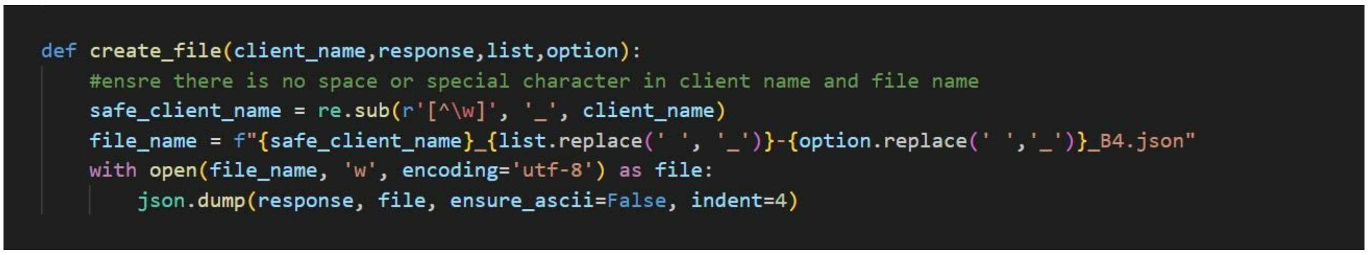
2. get_sources
   To retrieve sources from the API endpoint /sources



```python
# resource search function
def get_sources(option, value):
    # pass api key
    params = {"apiKey": API_KEY, "pageSize": 15}

    # check chosen option
    if option == "category":
        params["category"] = value
    elif option == "country":
        params["country"] = value
    elif option == "language":
        params["language"] = value

    # get response from endpoint and return it
    try:
        response = requests.get(f"{BASE_URL}/sources", params=params)
        if response.status_code != 200:
            print(f"Error with API: {response.status_code}, {response.text}")
            return {"error": "API error"}
        return response.json()
    except Exception:
        return {"API_error": "API error, check the connection"}


def create_file(client_name, response, list, option):
    # ensre there is no space or special character in client name and file name
    safe_client_name = re.sub(r"[^\w]", "_", client_name)
    file_name = (
        f"{safe_client_name}_{list.replace(' ', '_')}-{option.replace(' ','_')}_B4.json"
    )
    with open(file_name, "w", encoding="utf-8") as file:
        json.dump(response, file, ensure_ascii=False, indent=4)
```

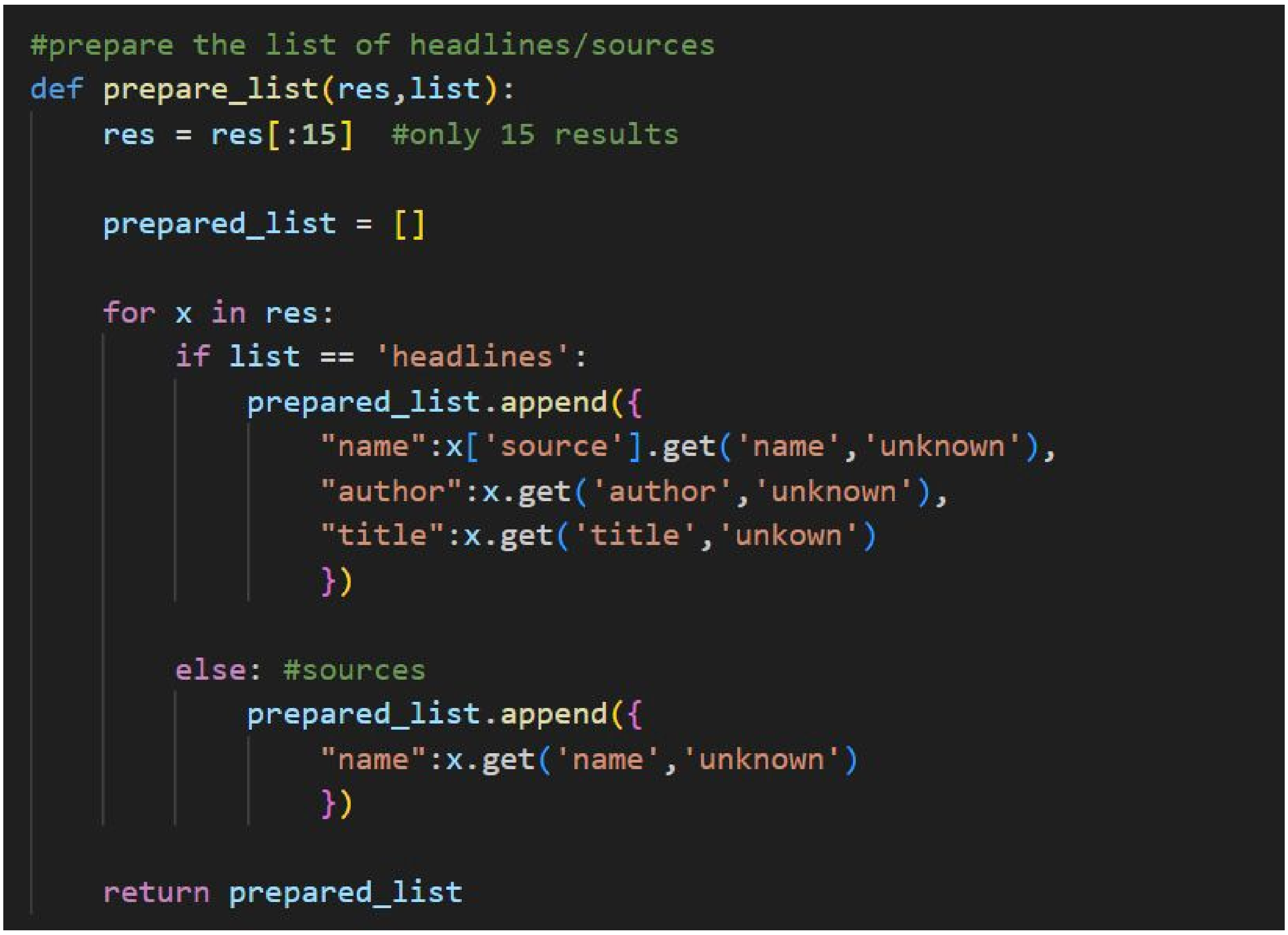
3. create_file
   This function will save the API responses into a file named with "<client_name>

   <group_ID>.json"



```python
def create_file(client_name,response,list,option):
    #ensre there is no space or special character in client name and file name
    safe_client_name = re.sub(r'[^\w]', '_', client_name)
    file_name = f"{safe_client_name}_{list.replace(' ', '_')}-{option.replace(' ','_')}_B4.json"
    with open(file_name, 'w', encoding='utf-8') as file:
        json.dump(response, file, ensure_ascii=False, indent=4)
```

4. prepare_list
   It will create a list containing only the information that the client will display to the user from the API response.



```python
#prepare the list of headlines/sources
def prepare_list(res,list):
    res = res[:15]  #only 15 results

    prepared_list = []

    for x in res:
        if list == 'headlines':
            prepared_list.append({
                "name":x['source'].get('name','unknown'),
                "author":x.get('author','unknown'),
                "title":x.get('title','unkown')
                })

        else: #sources
            prepared_list.append({
                "name":x.get('name','unknown')
                })

    return prepared_list
```

5. search This function is responsible for receiving clients' requests and responding to them.

Additional functions that used in both client and server side

- get_local_ip():

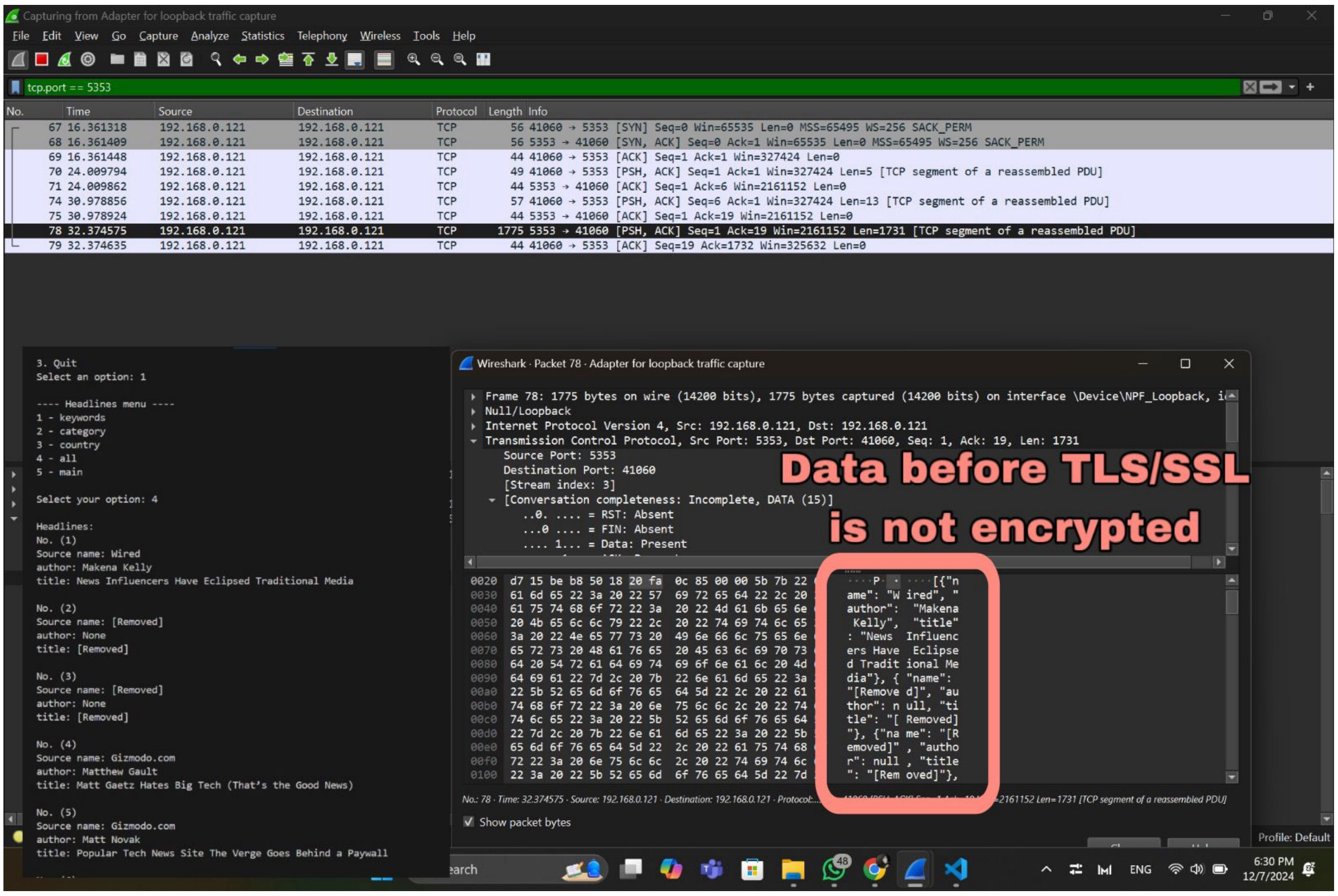
A function for retrieving the local IP of the device.
The IP address retrieved by this function is assigned for internal use(not public). It works by contacting 8.8.8.8 which is Google’s public DNS server with a UDP socket where there is no need for sending data, this step is just to determine which network interface is being used.
After creating the socket it connects it to the DNS server. We can get the IP address using s.getsockname() which returns a tuple of the IP address and port number. s.getsockname()[0] will be the local IP address we need.
Finally, the method returns the IP address.

Additional Concepts

TLS\SSL (security) TLS/SSL is a transport security protocol that provides a secure way for communication by providing some services. One of those services is confedintiality which depends on the encryption. On the screenshots below you can notice the data before applying TLS/SSL and after.

- Before TLS/SSL:

(1.)


Capturing from Adapter for loopback traffic capture
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp.port == 5353
Data before TLS/SSL
is not encrypted
Show packet bytes
Profile: Default

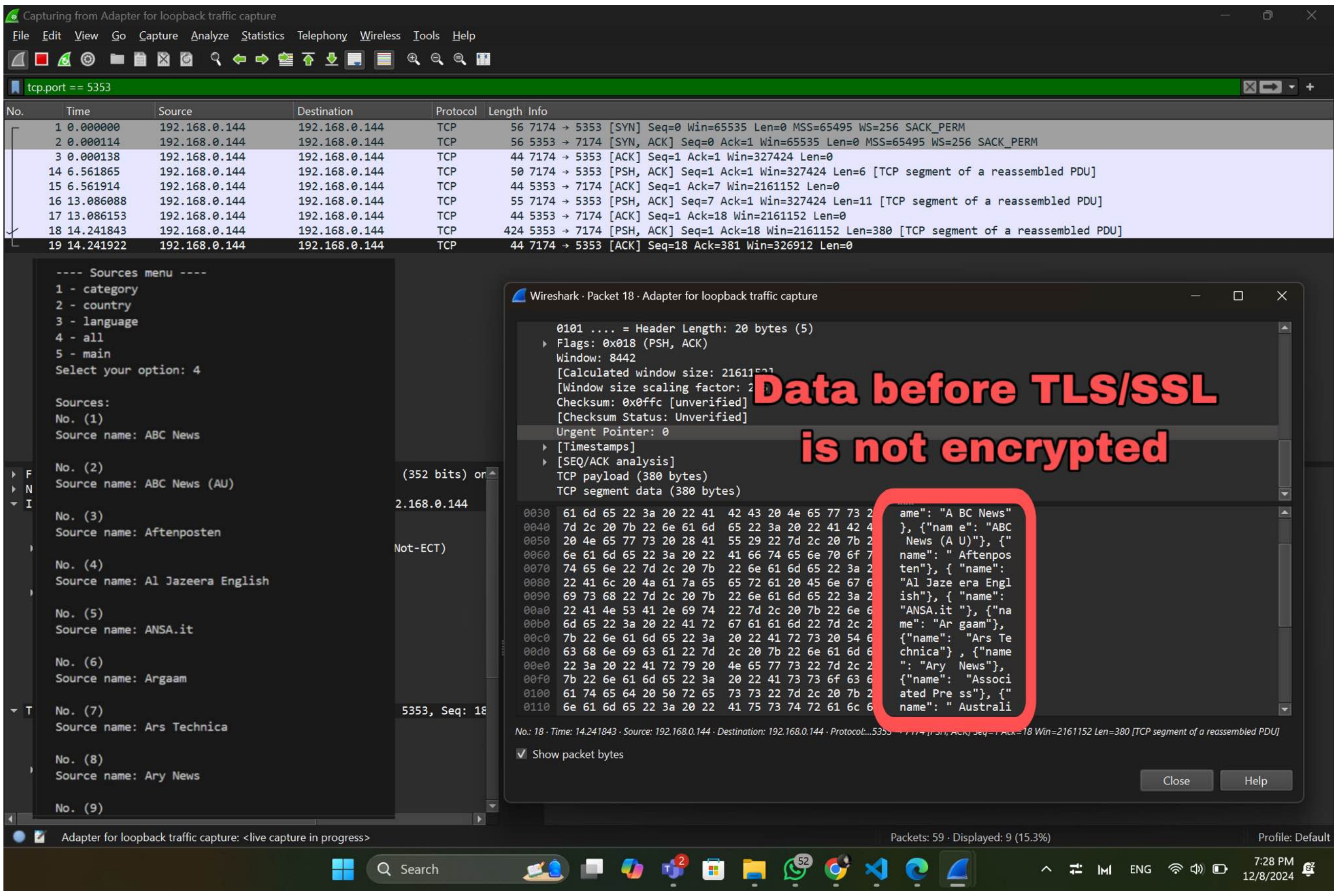
```
No.  Time       Source         Destination    Protocol Length Info
  67 16.361318  192.168.0.121  192.168.0.121  TCP        56 41060 → 5353 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
  68 16.361409  192.168.0.121  192.168.0.121  TCP        56 5353 → 41060 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
  69 16.361448  192.168.0.121  192.168.0.121  TCP        44 41060 → 5353 [ACK] Seq=1 Ack=1 Win=327424 Len=0
  70 24.009794  192.168.0.121  192.168.0.121  TCP        49 41060 → 5353 [PSH, ACK] Seq=1 Ack=1 Win=327424 Len=5 [TCP segment of a reassembled PDU]
  71 24.009862  192.168.0.121  192.168.0.121  TCP        44 5353 → 41060 [ACK] Seq=1 Ack=6 Win=2161152 Len=0
  74 30.978856  192.168.0.121  192.168.0.121  TCP        57 41060 → 5353 [PSH, ACK] Seq=6 Ack=1 Win=327424 Len=13 [TCP segment of a reassembled PDU]
  75 30.978924  192.168.0.121  192.168.0.121  TCP        44 5353 → 41060 [ACK] Seq=1 Ack=19 Win=2161152 Len=0
  78 32.374575  192.168.0.121  192.168.0.121  TCP      1775 5353 → 41060 [PSH, ACK] Seq=1 Ack=19 Win=2161152 Len=1731 [TCP segment of a reassembled PDU]
  79 32.374635  192.168.0.121  192.168.0.121  TCP        44 41060 → 5353 [ACK] Seq=19 Ack=1732 Win=325632 Len=0
```

```
3. Quit
Select an option: 1

---- Headlines menu ----
1 - keywords
2 - category
3 - country
4 - all
5 - main

Select your option: 4

Headlines:
No. (1)
Source name: Wired
author: Makena Kelly
title: News Influencers Have Eclipsed Traditional Media

No. (2)
Source name: [Removed]
author: None
title: [Removed]

No. (3)
Source name: [Removed]
author: None
title: [Removed]

No. (4)
Source name: Gizmodo.com
author: Matthew Gault
title: Matt Gaetz Hates Big Tech (That’s the Good News)

No. (5)
Source name: Gizmodo.com
author: Matt Novak
title: Popular Tech News Site The Verge Goes Behind a Paywall
```

Wireshark · Packet 78 · Adapter for loopback traffic capture

```
› Frame 78: 1775 bytes on wire (14200 bits), 1775 bytes captured (14200 bits) on interface \Device\NPF_Loopback, i
› Null/Loopback
› Internet Protocol Version 4, Src: 192.168.0.121, Dst: 192.168.0.121
⌄ Transmission Control Protocol, Src Port: 5353, Dst Port: 41060, Seq: 1, Ack: 19, Len: 1731
    Source Port: 5353
    Destination Port: 41060
    [Stream index: 3]
  ⌄ [Conversation completeness: Incomplete, DATA (15)]
      ..0. .... = RST: Absent
      ...0 .... = FIN: Absent
      .... 1... = Data: Present

0020  d7 15 be b8 50 18 20 fa  0c 85 00 00 5b 7b 22    ····P· · ····[{"n
0030  61 6d 65 22 3a 20 22 57  69 72 65 64 22 2c 20    ame": "W ired", "
0040  61 75 74 68 6f 72 22 3a  20 22 4d 61 6b 65 6e    author":  "Makena
0050  20 4b 65 6c 6c 79 22 2c  20 22 74 69 74 6c 65     Kelly",  "title"
0060  3a 20 22 4e 65 77 73 20  49 6e 66 6c 75 65 6e    : "News  Influenc
0070  65 72 73 20 48 61 76 65  20 45 63 6c 69 70 73    ers Have  Eclipse
0080  64 20 54 72 61 64 69 74  69 6f 6e 61 6c 20 4d    d Tradit ional Me
0090  64 69 61 22 7d 2c 20 7b  22 6e 61 6d 65 22 3a    dia"}, { "name":
00a0  22 5b 52 65 6d 6f 76 65  64 5d 22 2c 20 22 61    "[Remove d]", "au
00b0  74 68 6f 72 22 3a 20 6e  75 6c 6c 2c 20 22 74    thor": n ull, "ti
00c0  74 6c 65 22 3a 20 22 5b  52 65 6d 6f 76 65 64    tle": "[ Removed]
00d0  22 7d 2c 20 7b 22 6e 61  6d 65 22 3a 20 22 5b    "}, {"na me": "[R
00e0  65 6d 6f 76 65 64 5d 22  2c 20 22 61 75 74 68    emoved]" , "autho
00f0  72 22 3a 20 6e 75 6c 6c  2c 20 22 74 69 74 6c    r": null , "title
0100  22 3a 20 22 5b 52 65 6d  6f 76 65 64 5d 22 7d    ": "[Rem oved]"},

No.: 78 · Time: 32.374575 · Source: 192.168.0.121 · Destination: 192.168.0.121 · Protocol:... [illegible] =2161152 Len=1731 [TCP segment of a reassembled PDU]
```

(2.)


Capturing from Adapter for loopback traffic capture
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp.port == 5353
Data before TLS/SSL
is not encrypted
Show packet bytes
Close
Help
Profile: Default

```
No.  Time       Source         Destination    Protocol Length Info
   1 0.000000   192.168.0.144  192.168.0.144  TCP        56 7174 → 5353 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
   2 0.000114   192.168.0.144  192.168.0.144  TCP        56 5353 → 7174 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
   3 0.000138   192.168.0.144  192.168.0.144  TCP        44 7174 → 5353 [ACK] Seq=1 Ack=1 Win=327424 Len=0
  14 6.561865   192.168.0.144  192.168.0.144  TCP        50 7174 → 5353 [PSH, ACK] Seq=1 Ack=1 Win=327424 Len=6 [TCP segment of a reassembled PDU]
  15 6.561914   192.168.0.144  192.168.0.144  TCP        44 5353 → 7174 [ACK] Seq=1 Ack=7 Win=2161152 Len=0
  16 13.086088  192.168.0.144  192.168.0.144  TCP        55 7174 → 5353 [PSH, ACK] Seq=7 Ack=1 Win=327424 Len=11 [TCP segment of a reassembled PDU]
  17 13.086153  192.168.0.144  192.168.0.144  TCP        44 5353 → 7174 [ACK] Seq=1 Ack=18 Win=2161152 Len=0
  18 14.241843  192.168.0.144  192.168.0.144  TCP       424 5353 → 7174 [PSH, ACK] Seq=1 Ack=18 Win=2161152 Len=380 [TCP segment of a reassembled PDU]
  19 14.241922  192.168.0.144  192.168.0.144  TCP        44 7174 → 5353 [ACK] Seq=18 Ack=381 Win=326912 Len=0
```

```
---- Sources menu ----
1 - category
2 - country
3 - language
4 - all
5 - main
Select your option: 4

Sources:
No. (1)
Source name: ABC News

No. (2)
Source name: ABC News (AU)

No. (3)
Source name: Aftenposten

No. (4)
Source name: Al Jazeera English

No. (5)
Source name: ANSA.it

No. (6)
Source name: Argaam

No. (7)
Source name: Ars Technica

No. (8)
Source name: Ary News

No. (9)
```

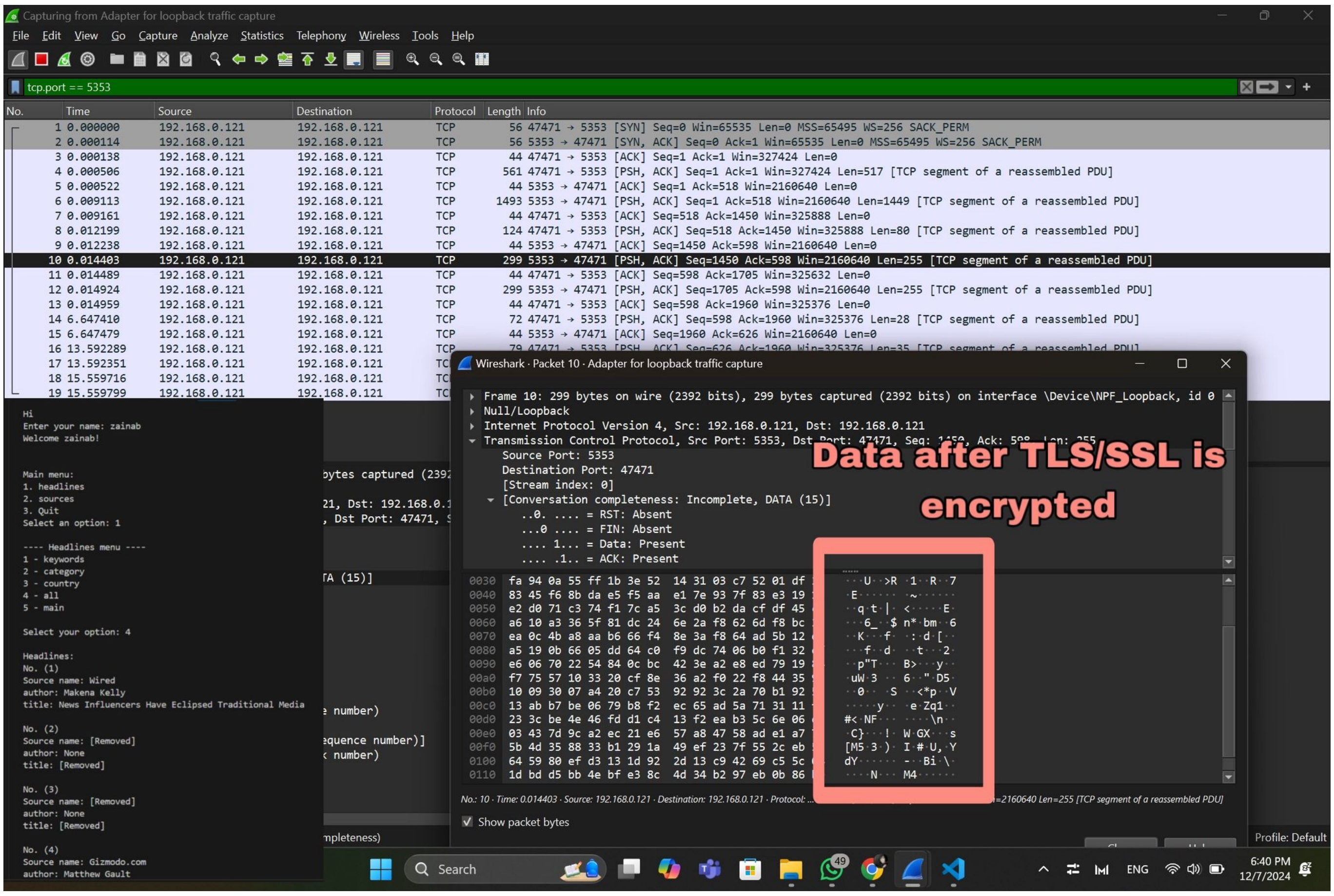
Wireshark · Packet 18 · Adapter for loopback traffic capture

```
    0101 .... = Header Length: 20 bytes (5)
  › Flags: 0x018 (PSH, ACK)
    Window: 8442
    [Calculated window size: 21611
    [Window size scaling factor: 2
    Checksum: 0x0ffc [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  › [Timestamps]
  › [SEQ/ACK analysis]
    TCP payload (380 bytes)
    TCP segment data (380 bytes)

0030  61 6d 65 22 3a 20 22 41  42 43 20 4e 65 77 73 2    ame": "A BC News"
0040  7d 2c 20 7b 22 6e 61 6d  65 22 3a 20 22 41 42 4    }, {"nam e": "ABC
0050  20 4e 65 77 73 20 28 41  55 29 22 7d 2c 20 7b 2     News (A U)"}, {"
0060  6e 61 6d 65 22 3a 20 22  41 66 74 65 6e 70 6f 7    name": " Aftenpos
0070  74 65 6e 22 7d 2c 20 7b  22 6e 61 6d 65 22 3a 2    ten"}, { "name":
0080  22 41 6c 20 4a 61 7a 65  65 72 61 20 45 6e 67 6    "Al Jaze era Engl
0090  69 73 68 22 7d 2c 20 7b  22 6e 61 6d 65 22 3a 2    ish"}, { "name":
00a0  22 41 4e 53 41 2e 69 74  22 7d 2c 20 7b 22 6e 6    "ANSA.it "}, {"na
00b0  6d 65 22 3a 20 22 41 72  67 61 61 6d 22 7d 2c 2    me": "Ar gaam"},
00c0  7b 22 6e 61 6d 65 22 3a  20 22 41 72 73 20 54 6    {"name":  "Ars Te
00d0  63 68 6e 69 63 61 22 7d  2c 20 7b 22 6e 61 6d 6    chnica"} , {"name
00e0  22 3a 20 22 41 72 79 20  4e 65 77 73 22 7d 2c 2    ": "Ary  News"},
00f0  7b 22 6e 61 6d 65 22 3a  20 22 41 73 73 6f 63 6    {"name":  "Associ
0100  61 74 65 64 20 50 72 65  73 73 22 7d 2c 20 7b 2    ated Pre ss"}, {"
0110  6e 61 6d 65 22 3a 20 22  41 75 73 74 72 61 6c 6    name": " Australi

No.: 18 · Time: 14.241843 · Source: 192.168.0.144 · Destination: 192.168.0.144 · Protocol:...5353 [illegible] 18 Win=2161152 Len=380 [TCP segment of a reassembled PDU]
```

Adapter for loopback traffic capture: <live capture in progress>   Packets: 59 · Displayed: 9 (15.3%)

- After TLS/SSL:

(1.)

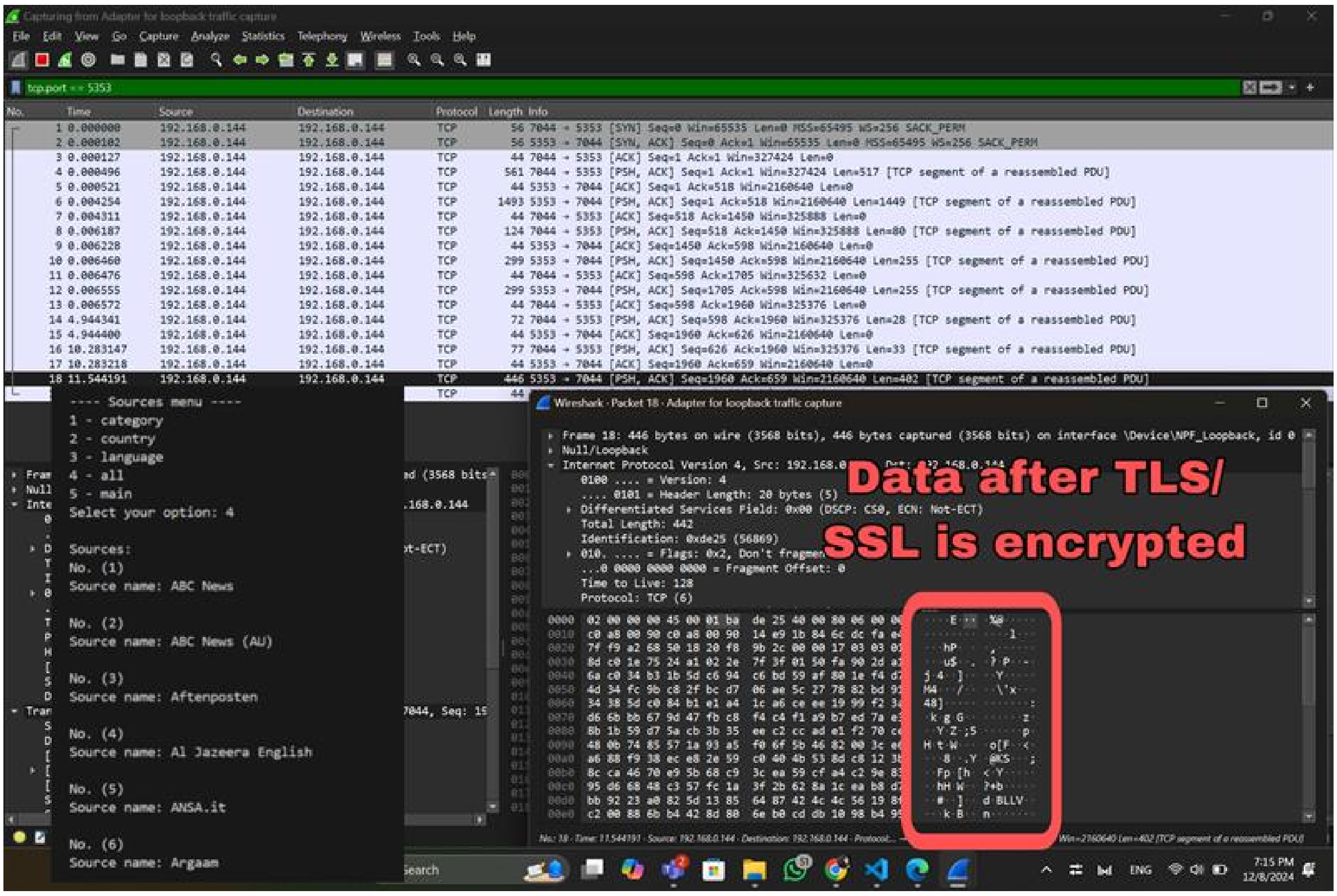

Capturing from Adapter for loopback traffic capture
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Wireshark · Packet 10 · Adapter for loopback traffic capture
Data after TLS/SSL is encrypted
Show packet bytes
Profile: Default
Search
6:40 PM 12/7/2024

```
tcp.port == 5353

No. Time         Source          Destination     Protocol Length Info
  1 0.000000     192.168.0.121   192.168.0.121   TCP      56 47471 → 5353 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
  2 0.000114     192.168.0.121   192.168.0.121   TCP      56 5353 → 47471 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
  3 0.000138     192.168.0.121   192.168.0.121   TCP      44 47471 → 5353 [ACK] Seq=1 Ack=1 Win=327424 Len=0
  4 0.000506     192.168.0.121   192.168.0.121   TCP     561 47471 → 5353 [PSH, ACK] Seq=1 Ack=1 Win=327424 Len=517 [TCP segment of a reassembled PDU]
  5 0.000522     192.168.0.121   192.168.0.121   TCP      44 5353 → 47471 [ACK] Seq=1 Ack=518 Win=2160640 Len=0
  6 0.009113     192.168.0.121   192.168.0.121   TCP    1493 5353 → 47471 [PSH, ACK] Seq=1 Ack=518 Win=2160640 Len=1449 [TCP segment of a reassembled PDU]
  7 0.009161     192.168.0.121   192.168.0.121   TCP      44 47471 → 5353 [ACK] Seq=518 Ack=1450 Win=325888 Len=0
  8 0.012199     192.168.0.121   192.168.0.121   TCP     124 47471 → 5353 [PSH, ACK] Seq=518 Ack=1450 Win=325888 Len=80 [TCP segment of a reassembled PDU]
  9 0.012238     192.168.0.121   192.168.0.121   TCP      44 5353 → 47471 [ACK] Seq=1450 Ack=598 Win=2160640 Len=0
 10 0.014403     192.168.0.121   192.168.0.121   TCP     299 5353 → 47471 [PSH, ACK] Seq=1450 Ack=598 Win=2160640 Len=255 [TCP segment of a reassembled PDU]
 11 0.014489     192.168.0.121   192.168.0.121   TCP      44 47471 → 5353 [ACK] Seq=598 Ack=1705 Win=325632 Len=0
 12 0.014924     192.168.0.121   192.168.0.121   TCP     299 5353 → 47471 [PSH, ACK] Seq=1705 Ack=598 Win=2160640 Len=255 [TCP segment of a reassembled PDU]
 13 0.014959     192.168.0.121   192.168.0.121   TCP      44 47471 → 5353 [ACK] Seq=598 Ack=1960 Win=325376 Len=0
 14 6.647410     192.168.0.121   192.168.0.121   TCP      72 47471 → 5353 [PSH, ACK] Seq=598 Ack=1960 Win=325376 Len=28 [TCP segment of a reassembled PDU]
 15 6.647479     192.168.0.121   192.168.0.121   TCP      44 5353 → 47471 [ACK] Seq=1960 Ack=626 Win=2160640 Len=0
 16 13.592289    192.168.0.121   192.168.0.121   TCP      79 47471 → 5353 [PSH, ACK] Seq=626 Ack=1960 Win=325376 Len=35 [TCP segment of a reassembled PDU]
 17 13.592351    192.168.0.121   192.168.0.121   TC
 18 15.559716    192.168.0.121   192.168.0.121   TC
 19 15.559799    192.168.0.121   192.168.0.121   TC
```

```
Hi
Enter your name: zainab
Welcome zainab!


Main menu:
1. headlines
2. sources
3. Quit
Select an option: 1

---- Headlines menu ----
1 - keywords
2 - category
3 - country
4 - all
5 - main

Select your option: 4

Headlines:
No. (1)
Source name: Wired
author: Makena Kelly
title: News Influencers Have Eclipsed Traditional Media

No. (2)
Source name: [Removed]
author: None
title: [Removed]

No. (3)
Source name: [Removed]
author: None
title: [Removed]

No. (4)
Source name: Gizmodo.com
author: Matthew Gault
```

```
Frame 10: 299 bytes on wire (2392 bits), 299 bytes captured (2392 bits) on interface \Device\NPF_Loopback, id 0
Null/Loopback
Internet Protocol Version 4, Src: 192.168.0.121, Dst: 192.168.0.121
Transmission Control Protocol, Src Port: 5353, Dst Port: 47471, Seq: 1450, Ack: 598, Len: 255
    Source Port: 5353
    Destination Port: 47471
    [Stream index: 0]
  [Conversation completeness: Incomplete, DATA (15)]
      ..0. .... = RST: Absent
      ...0 .... = FIN: Absent
      .... 1... = Data: Present
      .... .1.. = ACK: Present

0030  fa 94 0a 55 ff 1b 3e 52  14 31 03 c7 52 01 df   ···U··>R ·1··R··7
0040  83 45 f6 8b da e5 f5 aa  e1 7e 93 7f 83 e3 19   ·E······ ·~······
0050  e2 d0 71 c3 74 f1 7c a5  3c d0 b2 da cf df 45   ··q·t·|· <·····E·
0060  a6 10 a3 36 5f 81 dc 24  6e 2a f8 62 6d f8 bc   ···6_··$ n*·bm·6
0070  ea 0c 4b a8 aa b6 66 f4  8e 3a f8 64 ad 5b 12   ··K···f· ·:·d·[··
0080  a5 19 0b 66 05 dd 64 c0  f9 dc 74 06 b0 f1 32   ···f··d· ··t···2·
0090  e6 06 70 22 54 84 0c bc  42 3e a2 e8 ed 79 19   ··p"T··· B>···y··
00a0  f7 75 57 10 33 20 cf 8e  36 a2 f0 22 f8 44 35   ·uW·3 ·· 6··"·D5·
00b0  10 09 30 07 a4 20 c7 53  92 92 3c 2a 70 b1 92   ··0·· ·S ··<*p··V
00c0  13 ab b7 be 06 79 b8 f2  ec 65 ad 5a 71 31 11   ·····y·· ·e·Zq1··
00d0  23 3c be 4e 46 fd d1 c4  13 f2 ea b3 5c 6e 06   #<·NF··· ····\n··
00e0  03 43 7d 9c a2 ec 21 e6  57 a8 47 58 ad e1 a7   ·C}···!· W·GX···s
00f0  5b 4d 35 88 33 b1 29 1a  49 ef 23 7f 55 2c eb   [M5·3·)· I·#·U,·Y
0100  64 59 80 ef d3 13 1d 92  2d 13 c9 42 69 c5 5c   dY······ -··Bi·\·
0110  1d bd d5 bb 4e bf e3 8c  4d 34 b2 97 eb 0b 86   ····N··· M4······

No.: 10 · Time: 0.014403 · Source: 192.168.0.121 · Destination: 192.168.0.121 · Protocol: ... =2160640 Len=255 [TCP segment of a reassembled PDU]
```

(2.)


Capturing from Adapter for loopback traffic capture
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Wireshark · Packet 18 · Adapter for loopback traffic capture
Data after TLS/SSL is encrypted
Search
7:15 PM 12/8/2024

```
tcp.port == 5353

No. Time         Source          Destination     Protocol Length Info
  1 0.000000     192.168.0.144   192.168.0.144   TCP      56 7044 → 5353 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
  2 0.000102     192.168.0.144   192.168.0.144   TCP      56 5353 → 7044 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
  3 0.000127     192.168.0.144   192.168.0.144   TCP      44 7044 → 5353 [ACK] Seq=1 Ack=1 Win=327424 Len=0
  4 0.000496     192.168.0.144   192.168.0.144   TCP     561 7044 → 5353 [PSH, ACK] Seq=1 Ack=1 Win=327424 Len=517 [TCP segment of a reassembled PDU]
  5 0.000521     192.168.0.144   192.168.0.144   TCP      44 5353 → 7044 [ACK] Seq=1 Ack=518 Win=2160640 Len=0
  6 0.004254     192.168.0.144   192.168.0.144   TCP    1493 5353 → 7044 [PSH, ACK] Seq=1 Ack=518 Win=2160640 Len=1449 [TCP segment of a reassembled PDU]
  7 0.004311     192.168.0.144   192.168.0.144   TCP      44 7044 → 5353 [ACK] Seq=518 Ack=1450 Win=325888 Len=0
  8 0.006187     192.168.0.144   192.168.0.144   TCP     124 7044 → 5353 [PSH, ACK] Seq=518 Ack=1450 Win=325888 Len=80 [TCP segment of a reassembled PDU]
  9 0.006228     192.168.0.144   192.168.0.144   TCP      44 5353 → 7044 [ACK] Seq=1450 Ack=598 Win=2160640 Len=0
 10 0.006460     192.168.0.144   192.168.0.144   TCP     299 5353 → 7044 [PSH, ACK] Seq=1450 Ack=598 Win=2160640 Len=255 [TCP segment of a reassembled PDU]
 11 0.006476     192.168.0.144   192.168.0.144   TCP      44 7044 → 5353 [ACK] Seq=598 Ack=1705 Win=325632 Len=0
 12 0.006555     192.168.0.144   192.168.0.144   TCP     299 5353 → 7044 [PSH, ACK] Seq=1705 Ack=598 Win=2160640 Len=255 [TCP segment of a reassembled PDU]
 13 0.006572     192.168.0.144   192.168.0.144   TCP      44 7044 → 5353 [ACK] Seq=598 Ack=1960 Win=325376 Len=0
 14 4.944341     192.168.0.144   192.168.0.144   TCP      72 7044 → 5353 [PSH, ACK] Seq=598 Ack=1960 Win=325376 Len=28 [TCP segment of a reassembled PDU]
 15 4.944400     192.168.0.144   192.168.0.144   TCP      44 5353 → 7044 [ACK] Seq=1960 Ack=626 Win=2160640 Len=0
 16 10.283147    192.168.0.144   192.168.0.144   TCP      77 7044 → 5353 [PSH, ACK] Seq=626 Ack=1960 Win=325376 Len=33 [TCP segment of a reassembled PDU]
 17 10.283218    192.168.0.144   192.168.0.144   TCP      44 5353 → 7044 [ACK] Seq=1960 Ack=659 Win=2160640 Len=0
 18 11.544191    192.168.0.144   192.168.0.144   TCP     446 5353 → 7044 [PSH, ACK] Seq=1960 Ack=659 Win=2160640 Len=402 [TCP segment of a reassembled PDU]
                                                 TCP      44
```

```
---- Sources menu ----
1 - category
2 - country
3 - language
4 - all
5 - main
Select your option: 4

Sources:
No. (1)
Source name: ABC News

No. (2)
Source name: ABC News (AU)

No. (3)
Source name: Aftenposten

No. (4)
Source name: Al Jazeera English

No. (5)
Source name: ANSA.it

No. (6)
Source name: Argaam
```

```
Frame 18: 446 bytes on wire (3568 bits), 446 bytes captured (3568 bits) on interface \Device\NPF_Loopback, id 0
Null/Loopback
Internet Protocol Version 4, Src: 192.168.0.144, Dst: 192.168.0.144
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 442
    Identification: 0xde25 (56869)
  010. .... = Flags: 0x2, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 128
    Protocol: TCP (6)

0000  02 00 00 00 45 00 01 ba  de 25 40 00 80 06 00 00
0010  c0 a8 00 90 c0 a8 00 90  14 e9 1b 84 6c dc fa e4
0020  7f f9 a2 68 50 18 20 f8  9b 2c 00 00 17 03 03 01
0030  8d c0 1e 75 24 a1 02 2e  7f 3f 01 50 fa 90 2d a1
0040  6a c0 34 b3 1b 5d c6 94  c6 bd 59 af 80 1e f4 d7
0050  4d 34 fc 9b c8 2f bc d7  06 ae 5c 27 78 82 bd 91
0060  34 38 5d c0 84 b1 e1 a4  1c a6 ce ee 19 99 f2 3a
0070  d6 6b bb 67 9d 47 fb c8  f4 c4 f1 a9 b7 ed 7a e1
0080  8b 1b 59 d7 5a cb 3b 35  ee c2 cc ad e1 f2 70 ce
0090  48 0b 74 85 57 1a 93 a5  f0 6f 5b 46 82 00 3c e6
00a0  a6 88 f9 38 ec e8 2e 59  c0 40 4b 53 8d c8 12 3b
00b0  8c ca 46 70 e9 5b 68 c9  3c ea 59 cf a4 c2 9e 83
00c0  95 d6 68 48 c3 57 fc 1a  3f 2b 62 8a 1c ea b8 d7
00d0  bb 92 23 a0 82 5d 13 85  64 87 42 4c 4c 56 19 8f
00e0  c2 00 88 6b b4 42 8d 80  6e b0 cd db 10 98 b4 95

No.: 18 · Time: 11.544191 · Source: 192.168.0.144 · Destination: 192.168.0.144 · Protocol: ... Win=2160640 Len=402 [TCP segment of a reassembled PDU]
```

Acknowledgments

We would like to thank our instructor for providing this project so we can learn how to implement a python network system. Moreover, a big thanks to NewsAPI for providing the news.

Conclusion

This project demonstrates how secure connections, API integration, Sockects, and client-server communication can be practically implemented using a Python network application.

The development of the Multithreaded News Client/Server Information System helped us learn concepts of:

- Network programming (Python)
- API integration
- Multi-threaded
- client\server framework

Resources

- API: https://newsapi.org/
- TLS/SSL: https://docs.python.org/3/library/ssl.html