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

```
# Create a TCP socket using IPv4
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as client_s:
    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. get_user_name(): ask user about his name and send it to the server

```
def get_user_name(cs):
    print("Hi!")
    while True:
        user_name = input("\nEnter your name (Only letters are allowed): ").strip()
        if re.match("^[a-zA-Z]+(?: [a-zA-Z]+)*$", user_name):
            break

        print("Invalid name. Only letters are allowed. Please try again.")

    cs.sendall(user_name.encode("utf-8"))
    print(f"Welcome {user_name}!\n")
```

3. main_menu(): display the main menu

```python
def main_menu():
    main_menu = {"1": "headlines", "2": "sources", "3": "Quit"}
    while True:
        print("\nMain menu:")
        for key, value in main_menu.items():
            print(f"{key}. {value}")

        main_selection = input("Select an option: ").strip()

        if main_selection in main_menu:
            main_desc = main_menu[main_selection]
            break
        else:
            print("Invalid selected number! Try again.")

    return main_selection, main_desc
```

4. Headlines_menu() + Sources_menu(): display either headlines menu or sources menu depending on the user choice.

```
def Headlines_menu():
    Headlines = {
        "1": "keywords",
        "2": "category",
        "3": "country",
        "4": "all",
        "5": "main",
    }
    print("\n---- Headlines menu ----")
    for id, option in Headlines.items():
        print(f"{id} - {option}")

    Headlines_selection = input("\nSelect your option: ").strip()

    if Headlines_selection in Headlines:
        Headlines_desc = Headlines[Headlines_selection]
        return Headlines_selection, Headlines_desc
    else:
        print("Invalid selected number! Returning to main menu...")
        return -1, -1

"""
```

```
def Sources_menu():
    sources = {
        "1": "category",
        "2": "country",
        "3": "language",
        "4": "all",
        "5": "main",
    }

    print("\n---- Sources menu ----")
    for id, option in sources.items():
        print(f"{id} - {option}")

    sources_selection = input("Select your option: ").strip()

    if sources_selection in sources:
        source_desc = sources[sources_selection]
        return sources_selection, source_desc
    else:
        print("Invalid selected number! Returning to main menu...")
        return -1, -1
```

5. response(cs, main_selection): display response of the user request

```python
def response(cs, main_selection):
    try:
        # Check if the response is JSON which means a valid category, or country, or language
        response = receive_complete_data(cs)
        dict = json.loads(response)
        n = len(dict) + 1 # Allow 'back' to the main menu option to the user
        counter = 1

        for article in dict:
            if main_selection == "1":  # Headlines menu
                print(f"No. ({counter})")
                print(f"Source name: {article['name']} ")
                print(f"author: {article['author']} ")
                print(f"title: {article['title']} \n")
            elif main_selection == "2":  # sources menu
                print(f"No. ({counter})")
                print(f"Source name: {article['name']} \n")
            counter += 1
        print(f"{n}. Back to the main menu\n")

        return n

    # Otherwise: it is not a JSON response which means invalid category, or country, or language.
    # Print an appropriate message.
    except json.JSONDecodeError:
        print(f"{response}")
        return -1
```

6. detailed_response(cs, main_selection): display of user detailed request if required

```
    """
def detailed_response(cs, main_selection):
    try:
        detailed_response = receive_complete_data(cs)
        detailed_response_dict = json.loads(detailed_response)
        source_name = detailed_response_dict.get("source", {}).get(
            "name", "Unknown Source"
        )

        url = detailed_response_dict.get("url", "No URL")
        description = detailed_response_dict.get("description", "No Description")
        print("Source name: ", source_name)
        print("URL: ", url)
        print("Description: ", description)

        if main_selection == "1":
            author = detailed_response_dict.get("author", "Unknown Author")
            title = detailed_response_dict.get("title", "No Title")
            published_at = detailed_response_dict.get("publishedAt", "Unknown Date")

            # Handle date and time extraction
            if "T" in published_at:
                date, time = published_at.split("T")
                print("Publish date: ", date)
                print("Publish time: ", time)
            else:
                print("Published at: ", published_at)
            print("Author: ", author)
            print("Title: ", title)

        else:
            country = detailed_response_dict.get("country", "Unknown Country")
            category = detailed_response_dict.get("category", "Unknown Category")
            language = detailed_response_dict.get("language", "Unknown Language")
            print("Country: ", country)
            print("Category: ", category)
            print("Language: ", language)

    except json.JSONDecodeError:
        print(detailed_response)
        return -1
```

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
def get_headlines(option, value):
    """
    This function is for getting headlined from the endpoint
    /top-headlines with specific criteria chosed by the client
    with the using of params
    """       You, 6 hours ago • improve structure of server and commenting

    params = {"apiKey": API_KEY, "pageSize": 15}
    if option == "keywords":
        params["q"] = value
    elif option == "category":
        params["category"] = value
    elif option == "country":
        params["country"] = value

    try:
        response = requests.get(f"{BASE_URL}/top-headlines", params=params)
        if response.status_code != 200:
            """
            if code is not 200, then the request was not accepted
            """
            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 get_all_headlines():
    """
    All headlines have different endpoint which is /everything
    """
    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()
```

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

```python
def get_sources(option, value):
    """
    getting the sources from endpoint /sources with the use of
    params to pass the API key and filtering criteria
    """
    params = {"apiKey": API_KEY, "pageSize": 15}
    if option == "category":
        params["category"] = value
    elif option == "country":
        params["country"] = value
    elif option == "language":
        params["language"] = value

    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"}
```

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
    and save the full API response into a json file
    """
    safe_client_name = re.sub(r"[^\w]", "_", client_name)
    file_name = (
        f"{safe_client_name}_{list.replace(' ', '_')}"
        f"-{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
def prepare_list(response, list):
    """
    Taking the list of articles/sources and
    prepare a list with only brief data for
    only 15 elements
    """
    response = response[:15]  # only 15 results
    prepared_list = []

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

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

    return prepared_list
```

5. handle request After receiving a request, this fuction will handle it and return the full response list with source/headlines list

```python
def handle_request(list, option, client_name, value=""):
    """
    Checking wether the client is looking for healdlines or
    sources list and communicate with the approprate function
    """
    if list == "headlines":
        if option == "all":
            print(f"client {client_name} requested to search for all {list}")
            full_response = get_all_headlines()
        else:
            print(
                f"client {client_name} requested to search for {list} by {option} ({value})"
            )
            full_response = get_headlines(option, value)
        response = full_response.get("articles", [])

    elif list == "sources":
        if option == "all":
            print(f"client {client_name} requested to search for all {list}")
        else:
            print(
                f"client {client_name} requested to search for {list} by {option} ({value})"
            )
        full_response = get_sources(option, value)
        response = full_response.get("sources", [])

    return response, full_response
```

6. 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():

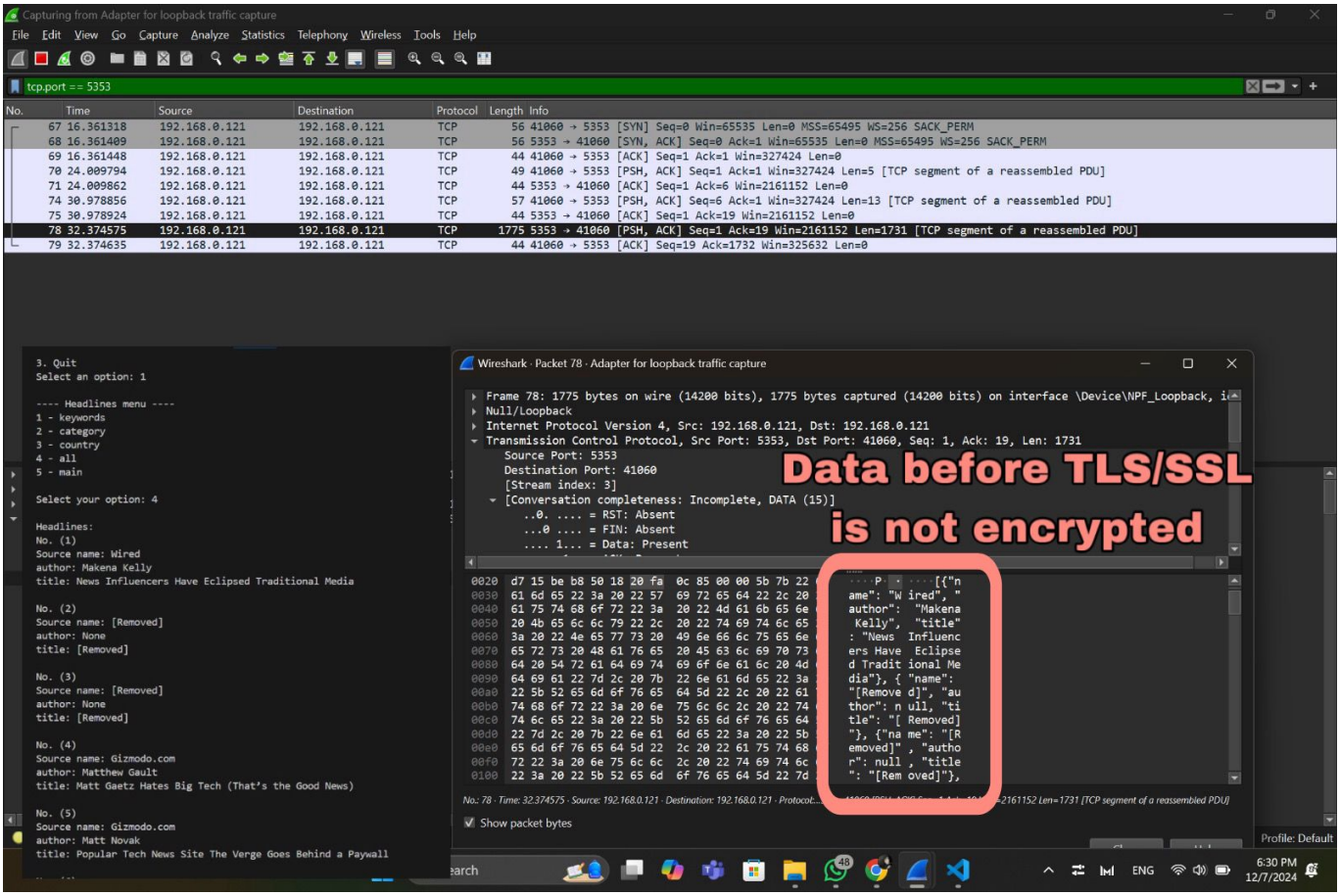
A function for retrieving the local IP of the device.
The IP address retrieved by this function is assigned for internal use(not public). It works by contacting 8.8.8.8 which is Google's public DNS server with a UDP socket where there is no need for sending data, this step is just to determine which 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

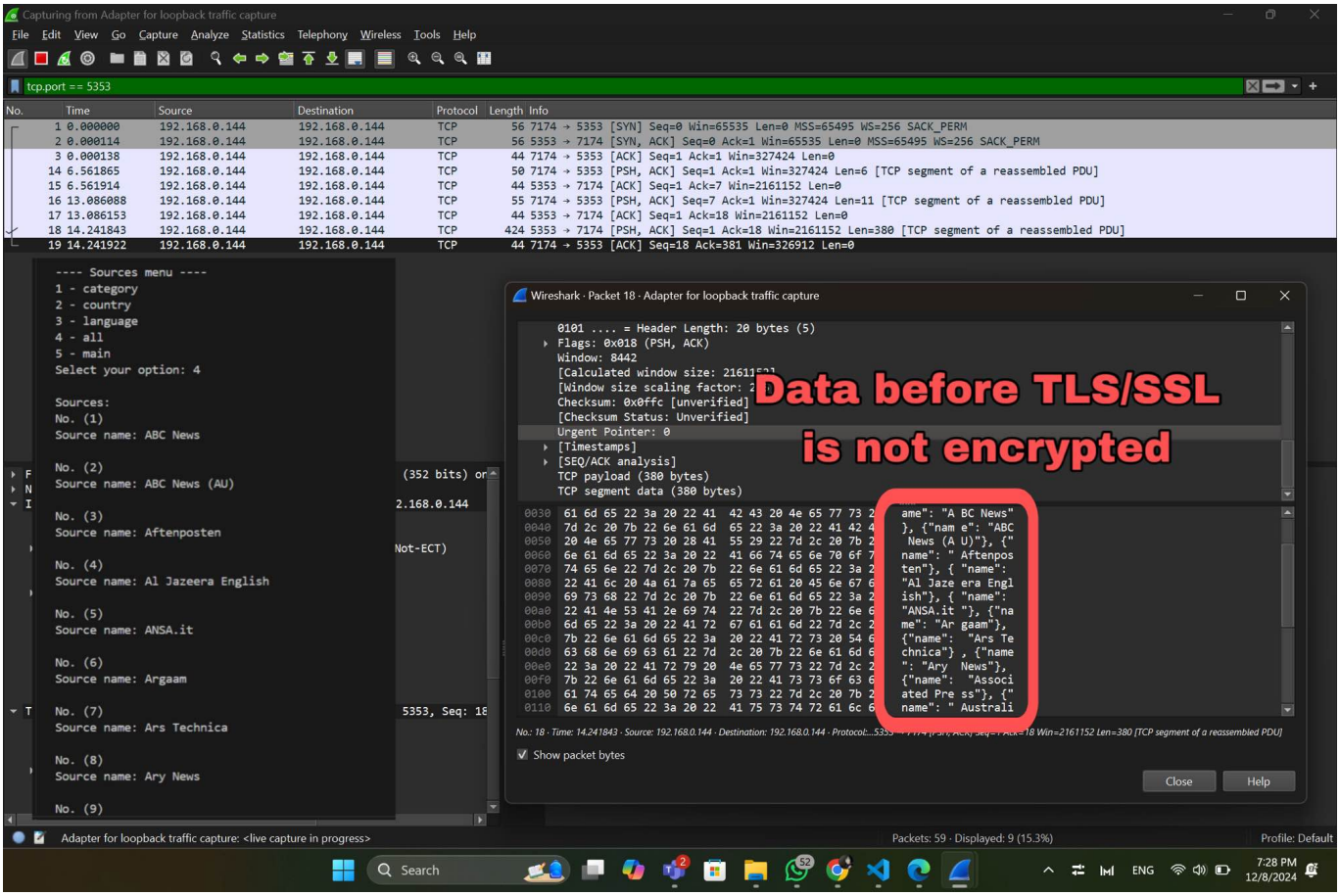
TLS\SSL (security) TLS/SSL is a transport security protocol that provides a secure way for communication by providing some services. One of those services is confedintiality which depends on the encryption. On the screenshots below you can notice the data before applying TLS/SSL and after.

- Before TLS/SSL:

(1.)

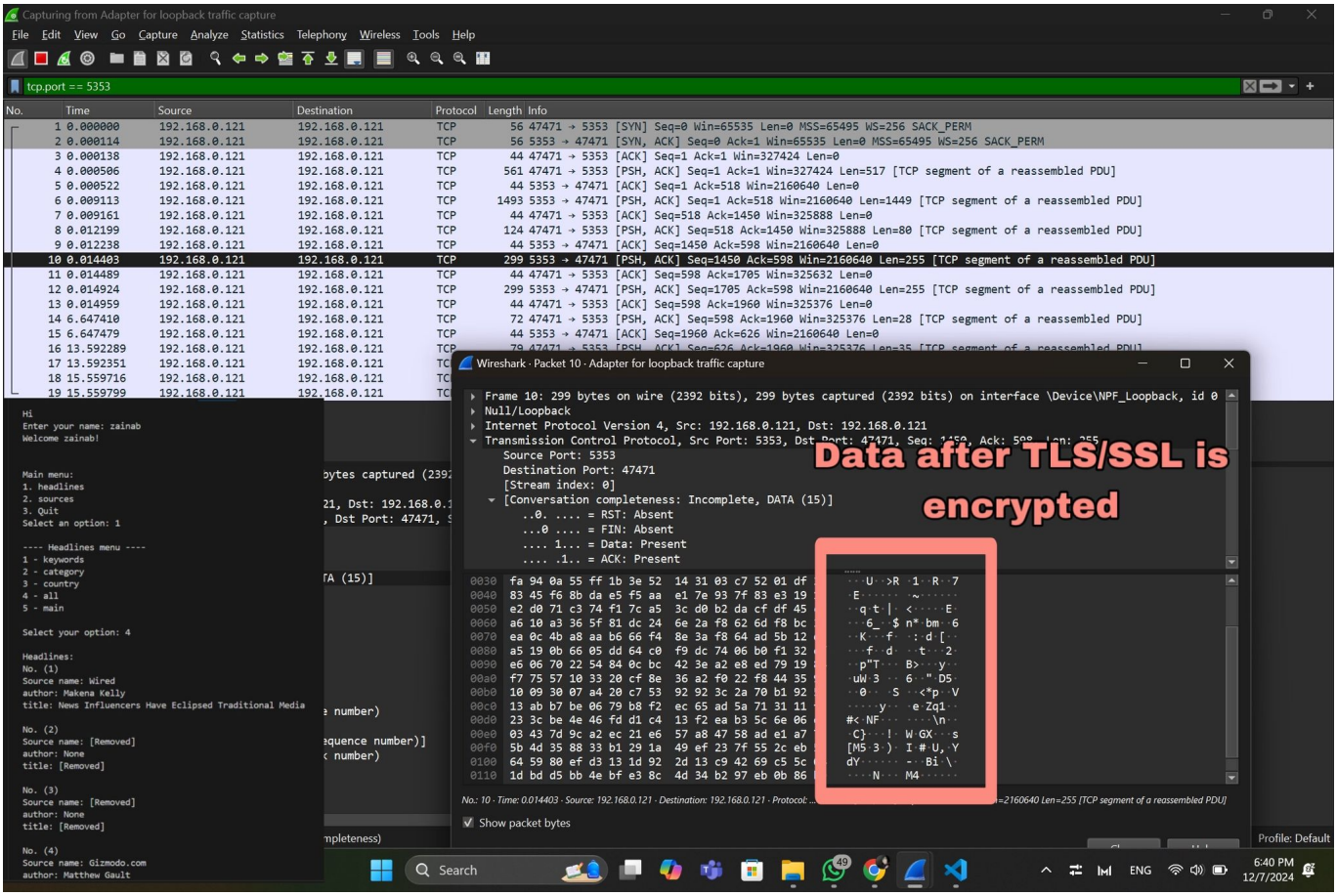


(2.)

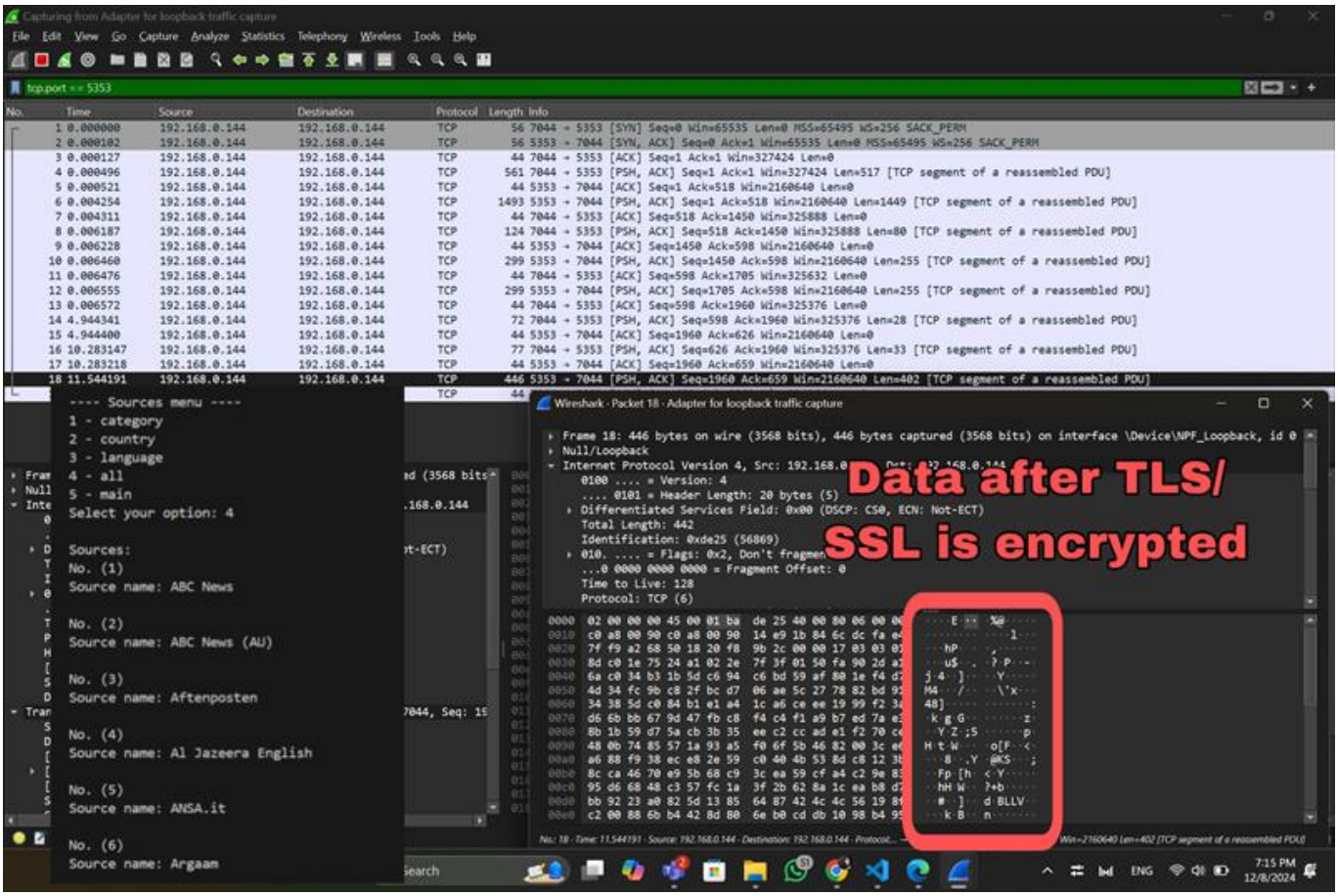


- After TLS/SSL:

(1.)



(2.)



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