Raspberry Pi wireless communication

Initial objective

The goal was to establish wireless communication between a computer(running `ROS`) and a `Raspberry Pi` using `ROS` topics. The computer would act as the **publisher** and the `Raspberry Pi` as the **subscriber**, both communicating over Wi-Fi.

Issue encountered

While attempting to implement `ROS` communication, a key issue arose:

- The computer uses `Ubuntu` on `WSL2`, which creates a separate virtual network (subnet) from the physical network.
- As a result, the `Raspberry Pi` and the computer could not directly communicate over `ROS`.

Alternative solution adopted

To overcome this network limitation, a functional alternative was implemented using `Python` `scripts` with `TCP sockets`. This approach maintains the core objective of enabling wireless communication and simulates the publisher/subscriber model.

Steps taken

1. Network setup

- Both the computer and the `Raspberry Pi` were connected to the **same Wi-Fi network**.

> For the connection between both devices to be successful, it is necessary to disable the firewall on the computer.

- To get the IP on a Windows system check the IPv4 row executing the `ipconfig` command in the command prompt (`cmd`):

```
ipconfig
[...]
Wireless LAN adapter WiFi:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::bbfe:6a34:a2c0:d8fe%15
   IPv4 Address. . . . . . . . . . . : 192.168.1.138
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.1.1
```

  In modern Linux-based OS, it is possible to use `ip a` in a similar fashion.

- Network connectivity was confirmed using `ping` from the computer to the `Raspberry Pi`.

```
pc to raspberry  ping 192.168.1.139

Haciendo ping a 192.168.1.139 con 32 bytes de datos:
Respuesta desde 192.168.1.139: bytes=32 tiempo=8ms TTL=64
Respuesta desde 192.168.1.139: bytes=32 tiempo=9ms TTL=64
Respuesta desde 192.168.1.139: bytes=32 tiempo=9ms TTL=64
```

```
raspberry to pc  ping 192.168.1.135
PING 192.168.1.135 (192.168.1.135) 56(84) bytes of data.
64 bytes from 192.168.1.135: icmp_seq=1 ttl=128 time=5.84 ms
64 bytes from 192.168.1.135: icmp_seq=2 ttl=128 time=9.38 ms
64 bytes from 192.168.1.135: icmp_seq=3 ttl=128 time=4.99 ms
```

```
^C
--- 192.168.1.135 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 5ms
rtt min/avg/max/mdev = 4.988/6.736/9.379/1.902 ms
```

Mind that the local IP addresses most likely will differ from different execution/connections to the network.

---

2. Code Implementation

Publisher `script` (`Windows` computer)

The following `Python` script was used to run the **publisher** on the `Windows` machine. It opens a server `socket`, accepts incoming connections, and allows the user to type messages that are sent to the connected client:

```python
import socket
import threading

HOST = '0.0.0.0'
PORT = 65432

clients = []
clients_lock = threading.Lock()

def handle_client(conn, addr):
    print(f"[Connected by {addr}]")
    try:
        while True:
            data = conn.recv(1024)
            if not data:
                break
            print(f"Message received from {addr}: {data.decode()}")
    except ConnectionResetError:
        print(f"Connection closed by {addr}")
    finally:
        with clients_lock:
            clients.remove(conn)
        conn.close()
        print(f"Connection with {addr} closed")

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.bind((HOST, PORT))
    s.listen()
    print(f"[Server listening on {HOST}:{PORT}]")

    def accept_clients():
        while True:
            conn, addr = s.accept()
            with clients_lock:
                clients.append(conn)
            threading.Thread(target=handle_client, args=(conn, addr), daemon=True).start()

    threading.Thread(target=accept_clients, daemon=True).start()

    print("You can send messages to all connected clients.")

    while True:
        msg = input("Message to send: ")
```

```
        with clients_lock:
            for client in clients:
                try:
                    client.sendall(msg.encode())
                except Exception as e:
                    print(f"Error sending to client: {e}")
```

Subscriber script (`Raspberry Pi`)

The following `Python` script was used to run the **subscriber** on the `Raspberry Pi`. It connects to the publisher's `IP` and prints any messages received:

```
import socket

WINDOWS_IP = '192.168.1.135'
PORT = 65432
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.connect((WINDOWS_IP, PORT))
    print(f"Connected to the server at {WINDOWS_IP}:{PORT}")
    while True:
        data = s.recv(1024)
        if not data:
            print("Connection closed by the server.")
            break
        print("Message received:", data.decode())
```

---

3. Functionality test

- The publisher script was executed on the `Windows` computer.

```
py publisher_server_windows.py
```

- The subscriber script was ran on the `Raspberry Pi`.

```
python3 subscriber_raspberry.py
```

- Messages typed in the publisher were displayed in real time on the `Raspberry Pi` terminal.

Output on `Raspberry Pi`:

```
Connected to the server at 192.168.1.135:65432
Message received: hello
Message received: bye
```