REPORT 3: IPV4, IPV6, UNICAST, MULTICAST, BROADCAST, ANYCAST

CONTENT

I. INTRODUCTION

Practical Report on Exploring Fundamental Knowledge and Experimentation with IPv4, IPv6 and Data Transmission Methods in Computer Networks: Unicast, Multicast, Broadcast, Anycast

Objective: To help the team understand the differences, configuration approaches, and testing procedures in real-world network environments.

1. IPv4

- IPv4 (Internet Protocol version 4): it is the most widely used version of the IP protocol, designed for identifying and routing data packets across the network, uses 32-bit addresses supporting approximately 4.3 billion unique IDs
- It supports unicast, broadcast, and multicast. Broadcast allows delivery to all hosts on a subnet, e.g: ARP requests
- However, the address space is exhausted, and IPv4 lacks built-in security and automated addressing features

2. IPv6

- IPv6 uses 128-bit addresses (e.g., 2001:0db8::1), offering a practically limitless address space for future expansion
- IPv6 eliminates broadcast; group communications now rely on multicast or anycast
- It includes enhancements like stateless autoconfiguration (SLAAC), improved security(IPSec), and support for unicast, multicast, and anycast

3. Unicast

- Unicast is a one-to-one communication model: a packet is sent from one source to a single destination.
- It is the standard model for most internet services – web browsing, SSH, streaming – and is supported by both IPv4 and IPv6

4. Multicast

- Multicast enables one-to-many communication; one sender transmits a single stream is received by group members only

- In IPv4, group management uses IGMP; in IPv6, MLD performs the equivalent function
- Ideal for applications like IPTV, video coferencing, and software distribution, multicast is efficient in bandwitdth usage

5. Broadcast

- In IPv4, broadcast allows sending a packet to all devices on the same subnet – e.g., ARP or DHCP broadcasts.
- This method can lead to network congestion and inefficiency
- IPv6 removes broadcast entirely replacing it with more selective multicast mechanisms

6. Anycast

- Anycast is one-to-nearest-of-many model supported explicitly in IPv6
- Multiple servers share the same anycast IP; traffic is routed to the nearest one based on routing metrics
- Widely used for CDN services and DNS root servers, anycast offers natural load-balancing and low-latency path selection

II. PREREQUISITES

1. Required Knowledge & Theory

- Networking fundamentals: familiar with TCP/IP stack, subnetting/CIDR, routing, and the address types – Unicast, Broadcast, Multicast, Anycast
- Group membership protocols: use IGMP for IPv4 and MLD for IPv6 to join multicast groups
- Python socket programming: create UDP sockets, configure broadcast and multicast options (SO_BROADCAST, IP_ADD_MEMBERSHIP, etc)
- Linux networking tools: ability to configure network interfaces (ip addr, route, sysctl), inspect multicast group membership (ifconfig | grep MULTICAST), and debug using ping, ping6, ip –f inet maddr, ip –6 maddr, and packet capture tools (Wireshark, tcpdump)

2. Hardware

- 3 Raspberry Pi (wuth WebOS) connected via Ethernet

- 1 Switch
- Ethernet cables (pre-crimped) to build an isolated network

3. Software Tools

- Python 3 with socket module for crafting UDP scripts. Multicast send/receive
- Utilities: ifconfig, ip, route, sysctl, ping, ping6, ip maddr
- For Anycast demo: FRR (Free Range Routing) installed on each Pi to advertise the anycast IPv6 via BGP or OSPF

4. Python Program Requirements

IPv4 programs:

- **Unicast**: standard UDP/TCP socket connect/send
- **Broadcast**: enable broadcast with setsockopt(SOL_SOCKET, SO_BROADCAST)
- **Multicast**: join group using IGMP via setsocketopt(IPPROTO_IP, IP_ADD_MEMBERSHIP)

IPv6 programs:

- **Unicast**: UDP/TCP using IPv6 family sockets
- **Multicast**: join IPv6 multicast groups using MLD via setsocketopt(IPPROTO_IPV6, IPV6_JOIN_GROUP)
- **Anycast**: sending to the anycast IPv6; routing ensures selection of nearest node

III. CONFIGURATIONS

1. Setup for IPV4:

- Command for device 1:

```
ip addr add 10.10.16.48/24 dev eth0 #set IP
systemctl restart NetworkManager
```

- Command for device 2:

```
sysctl net.ipv4.icmp_echo_ignore_broadcasts=0 #enable echo reply
```

```
ip addr add 10.10.16.108/24 dev eth0 #set IP sudo systemctl restart
NetworkManage
```

- Command for device 3:

```
sysctl net.ipv4.icmp_echo_ignore_broadcasts=0 #enable echo reply

ip addr add 10.10.16.92/24 dev eth0 #set IP

systemctl restart NetworkManage
```

2. Setup for IPV6:

Configure IPv6 for Anycast:

- Sender (Raspberry Pi 1): Assign an IPv6 address to Raspberry Pi 1.

```
ip -6 addr add fd00::1/64 dev eth0
```

- Receiver (Raspberry Pi 2 and 3): Assign the same Anycast IPv6 address to Raspberry Pi 2 and 3.

```
ip -6 addr add fd00::100/128 dev eth0 nodad
```

IV. TESTING

After successfully configuring IPv6 and IPv4, we will use Python program to send and receive packets over network communications.

1. IPV4

1.1 Multicast

- **Sender**:

```
import socket

MCAST_GRP = '224.1.1.1'
MCAST_PORT = 5007
MULTICAST_TTL = 2

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP)
sock.setsockopt(socket.IPPROTO_IP, socket.IP_MULTICAST_TTL, MULTICAST_TTL)

message = b"robot"

sock.sendto(message, (MCAST_GRP, MCAST_PORT))

print("Đã gửi thông điệp.")
```

-

In this program, we will send the message `b"robot"` to the multicast group address **IPv4 multicast range: 224.1.1.1**. This is the IPv4 multicast address where the packet will be delivered. When a packet is sent to this address, all devices that have “joined” the multicast group will receive the packet.

-After successful execution, the screen will display:” Đã gửi thông điệp.”

- **Receiver**

Device 2 is configured to join the IPv4 multicast group at 224.1.1.1, enabling it to receive packets sent to this address

```
# Cấu hình multicast group và cổng
MCAST_GRP = '224.1.1.1'
MCAST_PORT = 5007
IS_ALL_GROUPS = True  # Nếu True, nhận tất cả các nhóm multicast trên cổng đó
```

Device 2 runs the Python program receive1.py. After execution, the result will be displayed on the screen**.**

1.2 Unicast

- **Sender**

Configure direct sending to the recipient's IP address:

```
# Cấu hình
HOST = '10.10.16.48'  # Địa chỉ IP của máy nhận
PORT = 5007           # Cổng giao tiếp
```

Next, execute the Python script to initiate direct packet transmission to Device 2's IP address, carrying the message 'test unicastv4

```
root@raspberrypi4-64:/var/rootdirs/home/root# vi sendunicastv4.py
root@raspberrypi4-64:/var/rootdirs/home/root# vi sendunicastv4.py
root@raspberrypi4-64:/var/rootdirs/home/root# vi sendunicastv4.py
root@raspberrypi4-64:/var/rootdirs/home/root# python3 sendunicastv4.py
Enter your message (or 'exit' to quit): test unicastv4
Sent: test unicastv4
Enter your message (or 'exit' to quit): ^CTraceback (most recent call last):
  File "/var/rootdirs/home/root/sendunicastv4.py", line 13, in <module>
    message = input("Enter your message (or 'exit' to quit): ")
KeyboardInterrupt

root@raspberrypi4-64:/var/rootdirs/home/root# vi sendunicastv4.py
root@raspberrypi4-64:/var/rootdirs/home/root# python3 sendunicastv4.py
Enter your message (or 'exit' to quit): testing for unicastv4
Sent: testing for unicastv4
Enter your message (or 'exit' to quit): testing for unicastv4
Sent: testing for unicastv4
Enter your message (or 'exit' to quit): ^CTraceback (most recent call last):
  File "/var/rootdirs/home/root/sendunicastv4.py", line 13, in <module>
    message = input("Enter your message (or 'exit' to quit): ")
KeyboardInterrupt

root@raspberrypi4-64:/var/rootdirs/home/root# vi sendunicastv4.py
root@raspberrypi4-64:/var/rootdirs/home/root# python3 sendunicastv4.py
Enter your message (or 'exit' to quit): test unicastv4
Sent: test unicastv4
Enter your message (or 'exit' to quit):
```

- **Receiver**

The device is configured to listen for incoming packets from any network interface via UDP port 5007.

```
# Cấu hình
HOST = '0.0.0.0'  # Lắng nghe trên tất cả các giao diện mạng
PORT = 5007
```

Once Device 2 is successfully configured, execute the Python script to begin listening for incoming packets.

```
PS C:\Users\kieth> ssh root@10.10.16.48
root@raspberrypi4-64:/var/rootdirs/home/root# vi receiveunicastv4.py
root@raspberrypi4-64:/var/rootdirs/home/root# python3 receive
receive.py           receive_5.py         receive_uni_v6.py
receive1.py          receive_6.py         receivebroadcast.py
receive2.py          receive_broad.py     receiver_sniffer.py
receive_10.py        receive_multi_v4.py  receiveunicastv4.py
receive_3.py         receive_multi_v6.py
receive_4.py         receive_uni_v4.py
root@raspberrypi4-64:/var/rootdirs/home/root# python3 receiveunicastv4.py
Listening for unicast messages...
^CTraceback (most recent call last):
  File "/var/rootdirs/home/root/receiveunicastv4.py", line 14, in <module>
    data, addr = s.recvfrom(1024)
KeyboardInterrupt

root@raspberrypi4-64:/var/rootdirs/home/root# vi receiveunicastv4.py
root@raspberrypi4-64:/var/rootdirs/home/root# python3 receiveunicastv4.py
Listening for unicast messages...
Received: test unicastv4 from ('10.10.16.92', 38980)
```

After receive successful , screen is display message address IP sender’s

1.3 Broadcast

- **Sender :**

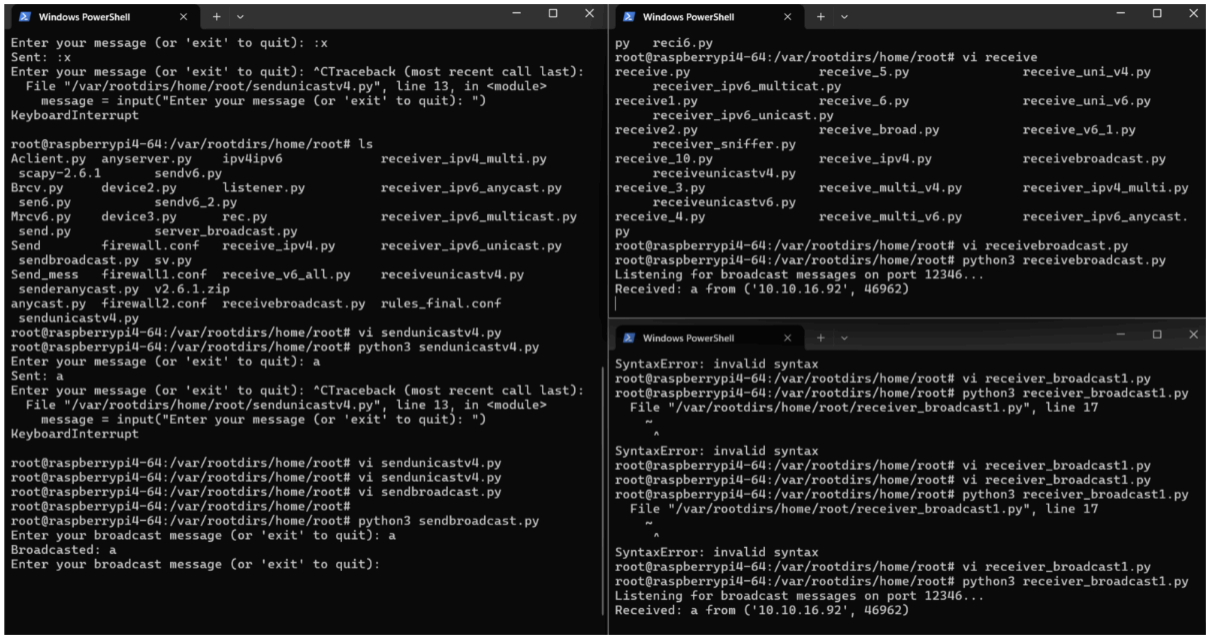
Configure address sender’s

```
# Cấu hình
BROADCAST_IP = '10.10.16.255'  # Địa chỉ broadcast của subnet
PORT = 12346                   # Cổng giao tiếp
```

- **Receiver**

Configure the listening address and communication port:

```
# Cấu hình
HOST = '0.0.0.0'  # Lắng nghe trên tất cả các giao diện mạng
PORT = 12346       # Cổng giao tiếp
```



2. IPV6

2.1 Multicast

- **Sender**

Configure the sender device to transmit data to the multicast group address **`ff04::111`** via UDP port **1000.**

```
# Cấu hình IPv6 Multicast
MULTICAST_GROUP = "ff04::111"  # All nodes link-local multicast
PORT = 10000
HOP_LIMIT = 1  # Tương đương TTL trong IPv4
```

- **Receiver**

Receiver1 is configured to listen for IPv6 multicast traffic on group address **`ff04::123`** using the designated UDP port

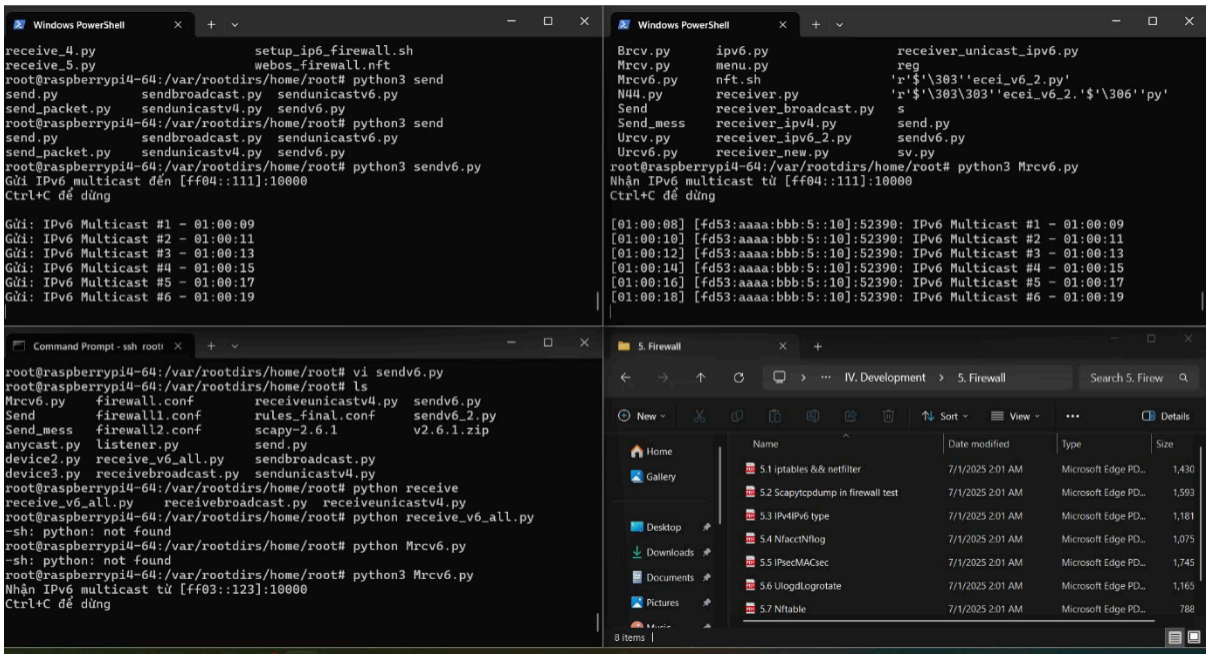
```
# Cấu hình IPv6 Multicast (phải giống v♦.i sender)
MULTICAST_GROUP = "ff04::123" # All nodes link-local multicast
PORT = 10000
INTERFACE_INDEX = 0  # 0 = interface mặc định
```

Receiver2 is configured to receive IPv6 multicast traffic on group address **`ff04::111`** via the designated UDP port.

```
# Cấu hình IPv6 Multicast (phải giống v♦.i sender)
MULTICAST_GROUP = "ff04::111"  # All nodes link-local multicast
PORT = 10000
INTERFACE_INDEX = 0  # 0 = interface mặc định
```

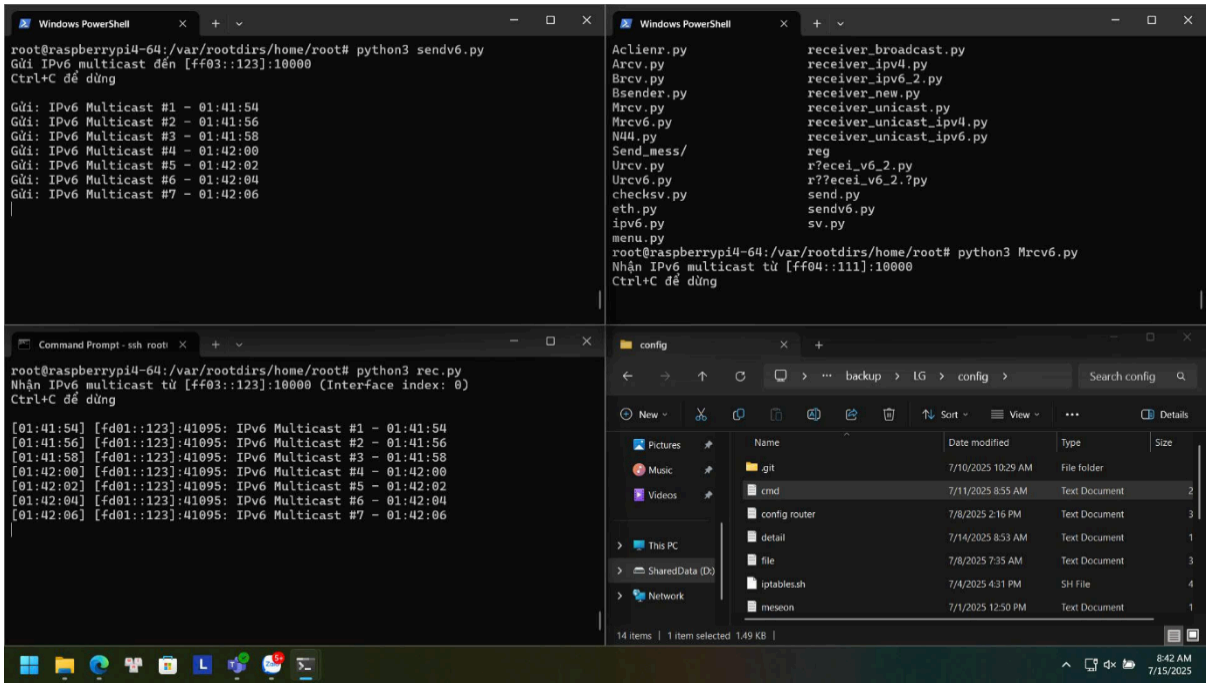
Sender Configuration: The sender device is set to transmit the message "IPv6 Multicast" to the multicast group address `ff04::111` via UDP port 10000.

- **Receiver1**: Configured with multicast group address `ff04::123.` → Will **not receive** the message due to address mismatch.
- **Receiver2**: Configured with multicast group address `ff04::111.` → Will **successfully receive** the message, confirming proper group membership and multicast delivery.



Sender Configuration: The sender device transmits the message "IPv6 Multicast" to the multicast group address **`ff04::123`** using UDP port 10000.

- **Receiver1**: Configured to listen on multicast group address **`ff04::123`**. → Will **successfully receive** the message, confirming proper group membership and multicast delivery.
- **Receiver2**: Configured to listen on multicast group address **`ff04::111`**. → Will **not receive** the message due to address mismatch.



2.2 Unicast

- **Sender**

The sender is set up to transmit IPv6 packets directly to the destination address **`fd53:1234:5678:5::39`** using UDP port **`10000`**

```
# Cấu hình
DEST_IP = "fd53:1234:5678:5::39"
PORT = 10000
```

- **Receiver**

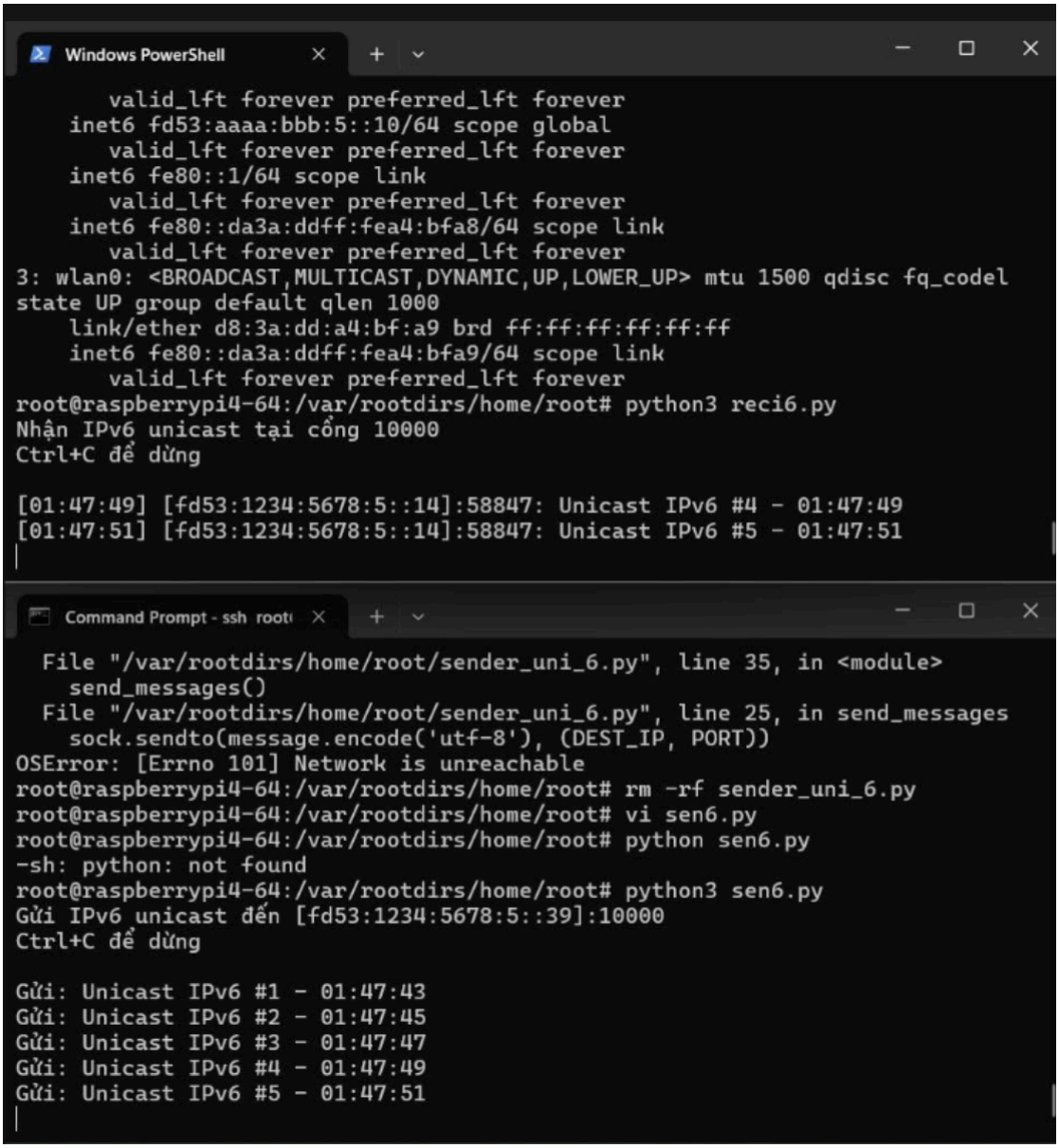
The receiving socket is bound to UDP port 10000 and configured to listen on all available network interfaces by specifying an empty string ('') as the host address

```
# Cấu hình
PORT = 10000

def create_receiver():
    """Tạo socket nhận IPv6 unicast"""
    sock = socket.socket(socket.AF_INET6, socket.SOCK_DGRAM)
    sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

    # Bind socket đến cổng
    sock.bind(('', PORT))
    return sock
```

Sender: The sender device uses the destination IPv6 address **`fd53:1234:5678:5::39`** to transmit data to the receiver over the **UDP protocol**, using **port 10000**.

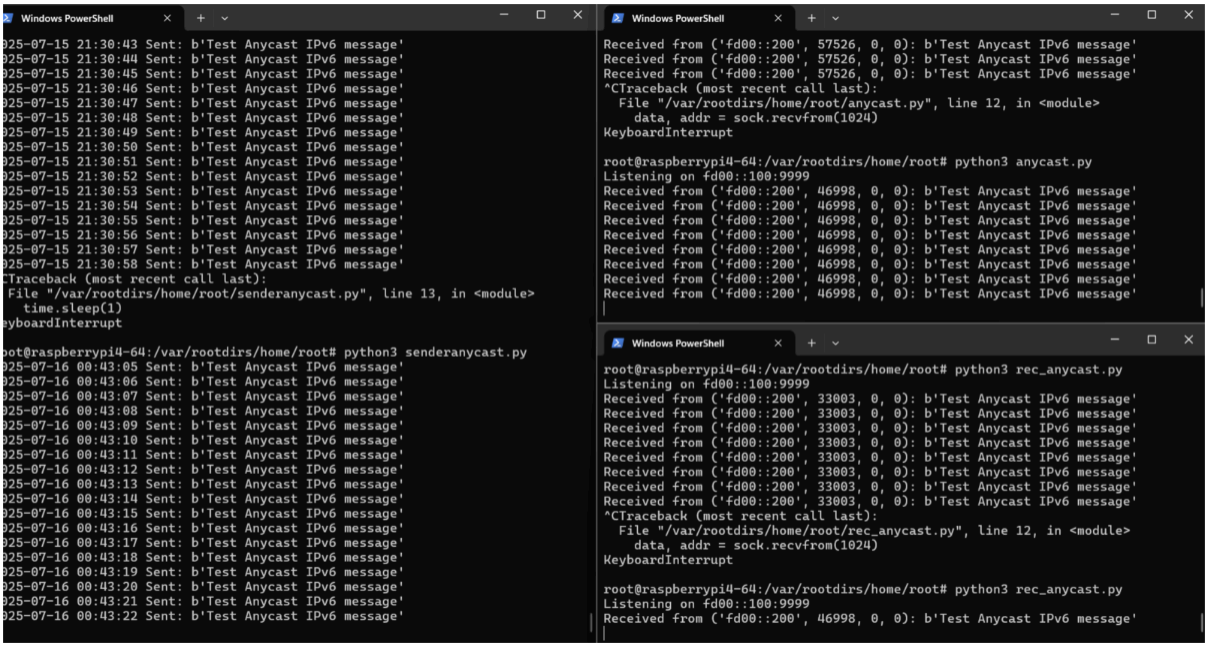
The transmitted message follows the format: "`Unicast IPv6 #<sequence number> - <timestamp>`", e.g., "`Unicast IPv6 #4 - 14:21:36`"

Receiver: The receiver is configured to **listen on UDP port 10000** across all network interfaces.
→ Successfully received messages "`Unicast IPv6 #4`" and "`Unicast IPv6 #5`".

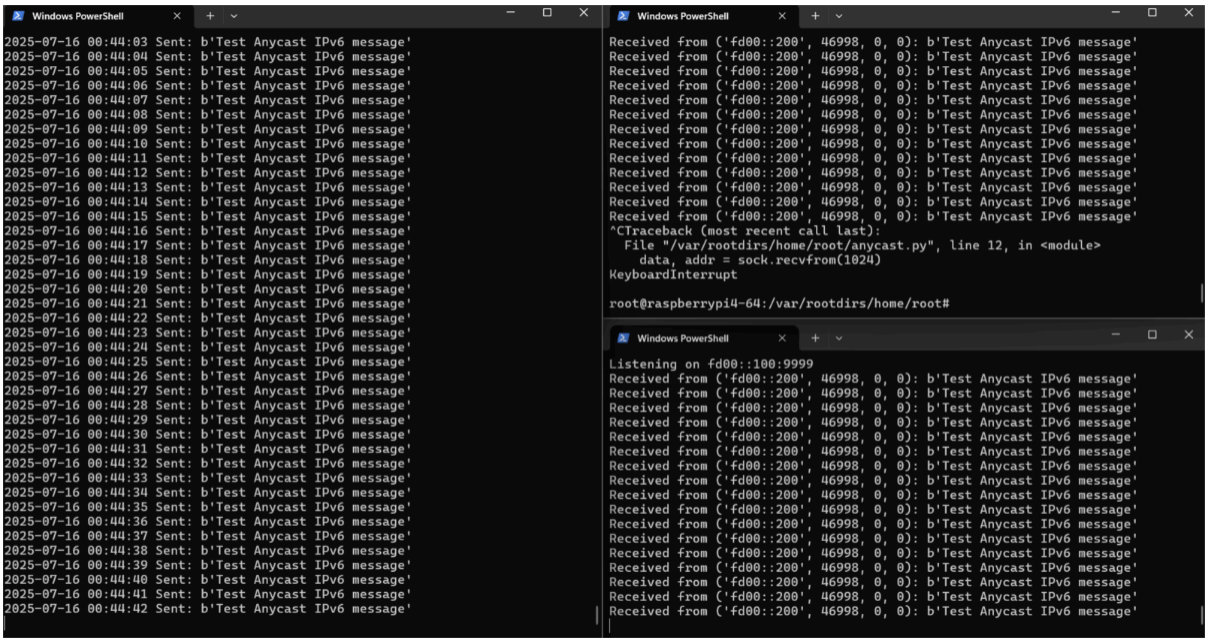


2.3 Anycast

With anycast, we assign the same IP address to two Raspberry Pis and configure them to ignore duplicate address detection (nodad), allowing both devices to share a single IP. When code is executed to send data, it will be delivered to whichever Raspberry Pi receives the packet first.



When one receiver is turned off on one side, the remaining Raspberry Pi becomes the nearest node according to the anycast mechanism.



V. PERFORMANCE AND SECURITY EVALUATION

Transmission Type	Transmission Mode	Advantages	Disadvantages	Performance
Unicast	1 source → 1 destination	Easy to control, high security	Not efficient for multiple destinations	High for one-to-one; degrades with many sessions
Broadcast	1 source → all devices in subnet	Fast, simple within LAN	Consumes bandwidth, poor security	Low if many nodes respond
Multicast	1 source → group of subscribed hosts	Efficient for group transmission, bandwidth-saving	Complex to configure, vulnerable to sniffing	High when supported by multiple recipients
Anycast	1 source → nearest destination in group	Low latency, fault-tolerant routing	Requires BGP, difficult to debug and simulate	Very high due to optimized routing