**Mở rộng class Server để cho Node vừa có chức năng “nghe” gói tin UDP (của Kademlia) vừa “nghe” gói tin từ Websocket nếu node đó sau NAT.**

Để mở rộng class Server của Kademlia lắng nghe đồng thời cả gói tin từ UDP và WebSocket relay (với relay có thể từ 1 đến nhiều endpoint để fallback), ta sẽ:

**Mục tiêu:**

* Khi khởi tạo Server, chạy song song *(mỗi node đều chạy song song lắng nghe ở 2 cổng khác nhau, 1 cổng là lắng nghe UDP mặc định của Kademlie, 1 cổng là lắng nghe websocket nếu Node là sau NAT)*:
  + UDP listener (mặc định của Kademlia)
  + WebSocket relay listener (từ client relay gửi đến)
* Gói tin nhận từ WebSocket sẽ được parse như gói tin UDP,
* Ghi đè các hàm \_\_getattr\_\_ (sửa hàm func() để tùy node sau NAT hay không mà dùng sendto() hoặc send\_via\_relay()
* Ghi đè hàm \_accept\_request () cũng kiểm tra node có sau NAT hay không để dùng sendto() hoặc send\_via\_relay()

**Giải pháp:**

1. **Extend Server class để thêm hàm start\_relay\_listener() (*để lắng nghe từ server websocket)***
2. **Relay client gửi JSON có dạng { "from": [ip, port], "data": ... }**
3. **Gói tin WebSocket chuyển vào \_solve\_datagram() như là nhận gói tin UDP thật**
4. *Phân tích dữ liệu Meta từ data để xem node có NAT không và tìm URL relay để gửi gói tin qua relay.*

Ghi chú: Khái niệm “ghi đè” các hàm tức là mở rộng Class và def lại hàm đó.

Phải thêm 1 hàm sendto\_via\_relay () để gửi message qua relay.

Ghi đè toàn bộ các hàm rpc\_\* và call\_\* để thêm thuộc tính meta vào.

**Triển khai**

# extended\_server.py

import asyncio

import json

from kademlia.network import Server

from kademlia.protocol import KademliaProtocol

import websockets

class ExtendedProtocol(KademliaProtocol):

def \_\_init\_\_(self, \*args, relay\_clients=None, \*\*kwargs):

super().\_\_init\_\_(\*args, \*\*kwargs)

self.relay\_clients = relay\_clients or {} # key = (ip, port), value = websocket

# Hàm này để xử lý nhận tin nhắn từ Relay về và gọi hàm xử lý trực tiếp

def **receive\_relay\_message**(self, data: bytes, addr: Tuple[str, int]):

"""Xử lý dữ liệu từ WebSocket relay giống như UDP."""

#phân tích dữ liệu data để lấy đúng data như khi gửi UDP

# ví dụ **data\_udp** = data … (phân tích data ở đây để trả vể data\_udp)

asyncio.ensure\_future(**self.\_solve\_datagram**(**data\_udp**, addr))

async def send\_relay\_response(self, addr, response\_bytes):

relay\_ws = self.relay\_clients.get(addr)

if relay\_ws:

msg = json.dumps({

"to": list(addr),

"data": response\_bytes.hex()

})

await relay\_ws.send(msg)

class ExtendedServer(Server):

def \_\_init\_\_(self, \*args, relay\_port=8765, \*\*kwargs):

super().\_\_init\_\_(\*args, \*\*kwargs)

self.relay\_port = relay\_port

self.relay\_clients = {}

async def **start\_relay\_listener**(self):

async def handler(ws):

try:

init = await ws.recv()

info = json.loads(init)

node\_id = info.get("node\_id")

print(f"[RelayWS] Node {node\_id} connected")

async for msg in ws:

try:

pkt = json.loads(msg)

addr = tuple(pkt.get("from"))

data = pkt.get("data")

if addr and data:

relay\_data = {

"from": addr,

"data": data,

"relay\_ws": ws # để phản hồi

}

self.protocol.**receive\_relay\_message**(message, addr\_of\_node\_relay)

**# Lệnh trên gọi sang hàm xử lý tin nhắn được gửi qua relay**

except Exception as e:

print("[RelayWS] Error:", e)

finally:

print("[RelayWS] Closed")

print(f"[RelayWS] Listening on port {self.relay\_port}")

await websockets.serve(handler, "0.0.0.0", self.relay\_port)

**# Đoạn sau (bôi xanh) cần xem lại xem có cần không**

async def listen(self, port, interface="0.0.0.0"):

await super().listen(port, interface)

# Override protocol để inject relay\_clients

self.protocol = ExtendedProtocol(

source\_node=self.node,

storage=self.storage,

ksize=self.ksize,

relay\_clients=self.relay\_clients

)

self.protocol.router = self.protocol.router

self.protocol.set\_server(self)

self.transport, \_ = await self.loop.create\_datagram\_endpoint(

lambda: self.protocol, local\_addr=(interface, port)

)

# Start WebSocket relay listener song song

asyncio.create\_task(self.start\_relay\_listener())