Lab 3: NATS Protocol Testing

Lab 3: NATS Protocol Testing

Objective

Set up NATS server and create Python publisher/subscriber for messaging.

Overview

NATS is a messaging protocol similar to MQTT but communicates directly server-client without an intermediate broker.

Installation

```
# Arch Linux
paru -S nats-server-bin

# Python library
pip install nats-py
```

Implementation

subscriber.py

```
import asyncio
import nats

async def main():
    nc = await nats.connect("nats://localhost:4222")
```

```
    async def message_handler(msg):
        print(f"Received: {msg.data.decode()} on {msg.subject}")

    sub = await nc.subscribe("iot.sensor", cb=message_handler)
    print("Subscriber listening on 'iot.sensor'...")

    try:
        while True:
            await asyncio.sleep(1)
    except KeyboardInterrupt:
        pass
    finally:
        await sub.unsubscribe()
        await nc.drain()

if __name__ == "__main__":
    asyncio.run(main())
```

publisher.py

```
import asyncio
import nats

async def main():
    nc = await nats.connect("nats://localhost:4222")

    messages = [
        "Temperature: 25.5C",
        "Humidity: 60%",
        "Pressure: 1013 hPa"
    ]

    for msg in messages:
        await nc.publish("iot.sensor", msg.encode())
        print(f"Published: {msg}")
        await asyncio.sleep(0.5)

    await nc.drain()
    print("Done publishing")

if __name__ == "__main__":
    asyncio.run(main())
```

Running

1. Start NATS server:

```
nats-server &
```

2. Start subscriber:

```
python subscriber.py &
```

3. Run publisher:

```
python publisher.py
```

Output

```
Subscriber listening on 'iot.sensor'...
Press Ctrl+C to stop
Published: Temperature: 25.5C
Received: Temperature: 25.5C on iot.sensor
Published: Humidity: 60%
Received: Humidity: 60% on iot.sensor
Published: Pressure: 1013 hPa
Received: Pressure: 1013 hPa on iot.sensor
Done publishing
```

Conclusion

NATS provides a simple and fast pub/sub messaging system. The Python nats-py library makes it easy to create publishers and subscribers using async/await patterns.