MESSAGE BROKER

تانیا جواهرپور ۹۷۳۳۰۱۷

شرح پروژه:

همانطور که در صورت پروژه گفته شده بود، در این پروژه باید از طریق سرور بین کلاینت‌ها پیام جا به جا کرد، به این صورت که کلاینتی درخواست publish کردن می‌دهد در موضوع مورد نظر پیام را منتشر میکند و کلاینت‌هایی که درخواست subscribe داده‌اند که به صورت مستمر در حال گوش کردن هستند، در صورتی که پیامی در موضوعی که آن‌ها عضو آن هستند منتشر شد، آن را دریافت و نمایش میدهند.

این دلال پیام از ping و pong جهت اطمینان از برقراری ارتباط هم پشتیبانی میکند.

ارتباط از طریق TCP protocol است و امکان اتصال هم‌زمان چند کلاینت برقرار است.

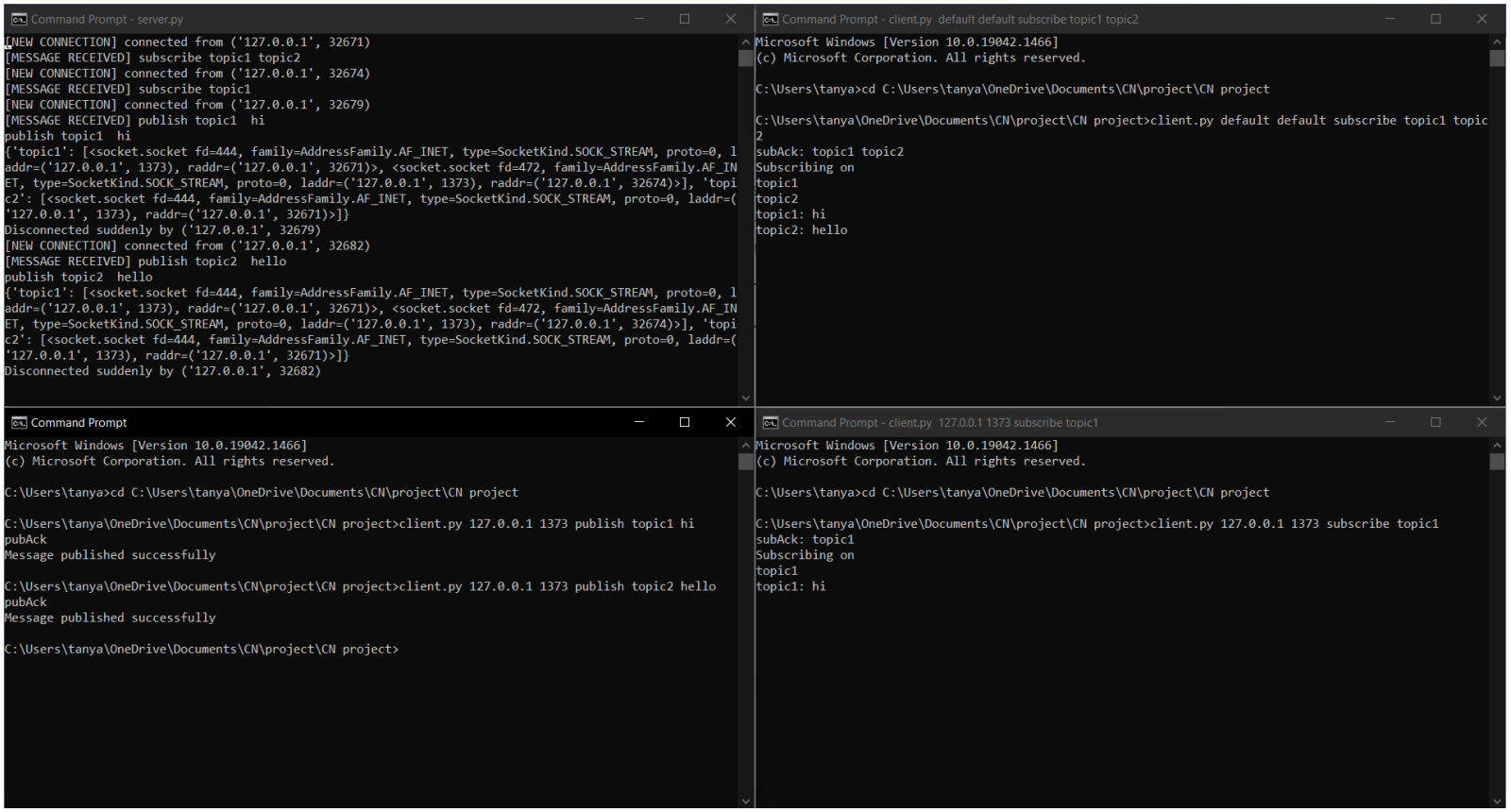
طراحی پروتکل:

همانطور که گفته شد ارتباط از طریق پروتکل TCP است، در نتیجه نیاز به IP و PORT مبدا و مقصد است و سرور نیاز دارد بعد از قبول کردن هر ارتباط این موارد را کامل بداند.

نحوه‌ی ورودی گرفتن در این پروتکل به این صورت است که آرگمان اول آدرس و آرگمان دوم پورت مورد نظر را مشخص میکند؛ کلاینت برای اتصال به سرور ما میتواند تنها عبارت default را برای این دو مورد تایپ کند. آرگمان سوم نوع دستور را مشخص میکند که می‌تواند subscribe، publish و یا ping باشد. اگر subscribe بود آرگمان چهارم به بعد موضوعات مد نظر را مشخص میکند، توجه داشته باشید موضوعات با فاصله از هم تفکیک میشوند و اگر موضوعی شامل چند کلمه باشد کلمات آن نباید با فاصله جدا شوند و می‌توان از underline یا موارد مشابه برای چسباندن کلمات استفاده کرد. در صورتی که آرگمان سوم publish بود آرگمان چهارم باید موضوع مورد نظر، که حتما باید موضوعی موجود در موضوعات باشد، هست و پس از آن یعنی آرگمان پنجم به بعد شامل پیام مورد نظر است که می‌خواهد منتشر کند و می‌تواند شامل فاصله هم باشد.

تعداد آرگمان‌های ورودی در کد بررسی می‌شود و اگر تعدادی کمتر از مقادیر مجاز داشت پیام مناسب چاپ می‌شود. در ادامه نحوه‌ی ورودی گرفتن و نمایش هم نشان داده خواهد شد.

نمونه ورودی و خروجی




```
C:\Users\tanya\OneDrive\Documents\CN\project\CN project>client.py 127.0.0.1 1373 publish topip3 test
INVALID TOPIC!
INVALID TOPIC!

C:\Users\tanya\OneDrive\Documents\CN\project\CN project>client.py 127.0.0.1 1373 subscribe
NO TOPIC DETECTED!
Please try again.

C:\Users\tanya\OneDrive\Documents\CN\project\CN project>client.py 127.0.0.1 1373 ping
pong

C:\Users\tanya\OneDrive\Documents\CN\project\CN project>client.py 127.0.0.1 1373
INVALID INPUT!
```

```
C:\Users\tanya\OneDrive\Documents\CN\project\CN project>client.py 127.0.0.1 1373
INVALID INPUT!

C:\Users\tanya\OneDrive\Documents\CN\project\CN project>client.py 127.0.0.1 1373 publish
NO TOPIC OR MESSAGE DETECTED!

C:\Users\tanya\OneDrive\Documents\CN\project\CN project>client.py 127.0.0.1 1373 publish topip1
NO MESSAGE DETECTED!

C:\Users\tanya\OneDrive\Documents\CN\project\CN project>client.py 127.0.0.1 1373 subscribe
NO TOPIC DETECTED!
Please try again.

C:\Users\tanya\OneDrive\Documents\CN\project\CN project>client.py 127.0.0.1 1373 subscrib
INVALID INPUT!
```

نحوه برقراری ارتباط:

سمت سرور

در سمت سرور پس از این که آدرس و پورت اختصاص داده شد، سرور شروع به گوش دادن می‌کند و در صورتی که کلاینتی درخواست ارتباط داد درخواست پذیرفته شده و ترد ساخته میشود تا کلاینت‌ها عضو شوند.

```python
def main():
    # HOST = socket.gethostbyname(socket.gethostname())
    HOST_INFORMATION = (HOST, PORT)
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.bind(HOST_INFORMATION)
    print("[SERVER START] Server is listening ...")
    start(s)


def start(server):
    server.listen()
    while True:
        conn, address = server.accept()
        t = threading.Thread(target=handle_client, args=(conn, address))
        t.start()
```

پیام کلاینت هم ابتدا طولش دریافت شده دیکود میشود و سپس با تابع execute_message پیام آن استخراج میشود.

```python
def handle_client(conn, address):
    print("[NEW CONNECTION] connected from {}".format(address))
    Connected = True
    while Connected:
        try:
            message_length = int(conn.recv(MESSAGE_LENGTH_SIZE).decode(ENCODING))
            msg = conn.recv(message_length)
            if not msg:
                continue
            msg = msg.decode(ENCODING)
            print("[MESSAGE RECEIVED] {}".format(msg))
            if msg == "DISCONNECT":
                Connected = False
            else:
                execute_message(msg, conn)
        except:
            remove_client(conn)
            print('Disconnected suddenly by', address)
            break
    conn.close()
```

سپس با توجه به اولین کلمه‌ی موجود در رشته که نشان‌دهنده‌ی درخواست کلاینت است تابع مربوطه صدا زده می‌شود.

```
def execute_message(msg, conn):
    split_msg = msg.split()
    if split_msg[0] == "subscribe":
        subscribe_handler(conn, split_msg[1:])
        # return True
    elif split_msg[0] == "publish":
        print(msg)
        publish_handler(conn, msg)
    elif split_msg[0] == "ping":
        pong(conn)
    elif split_msg[0] == "pong":
        ping(conn)
```

در صورتی که درخواست عضویت بوده باشد و عضویت موفقیت‌آمیز باشد subAck به همراه موضوعاتی که کلاینت عضو آن‌ها شده است ارسال می‌شود.

```
def subscribe_handler(conn: socket.socket, message):
    for msg in message:
        if msg in topics_members.keys():
            if conn not in topics_members[msg]:
                topics_members[msg].append(conn)
        else:
            topics_members[msg] = [conn]

    msg = "subAck:"
    for topic in topics_members.keys():
        if conn in topics_members[topic]:
            msg += " " + topic
    send_msg(conn, msg)
```

اگر درخواست انتشار بود هم در صورت انتشار موفقیت آمیز Ack ارسال می‌گردد.

```
def publish_handler(conn: socket.socket, message):
    message = message.partition(' ')[2]
    topic = message.partition(' ')[0]
    msg = topic + ":"
    msg += message.partition(' ')[2]
    if topic not in topics_members.keys():
        send_msg(conn, "INVALID TOPIC!")
    else:
        send_msg(conn, "pubAck")
        for client in topics_members[topic]:
            try:
                send_msg(client, msg)
            except:
                remove_client(client)
```

برای دو درخواست ping و pong هم پیام به شکل زیر ارسال می‌شود.

```
def ping(conn: socket.socket):
    send_msg(conn, "ping")


def pong(conn: socket.socket):
    send_msg(conn, "pong")
```

سرور هم برای ارسال پیام به کلاینت ابتدا پیام را انکود کرده و سپس طول آن و در نهایت پیام انکود شده را میفرستد.

```
def send_msg(server, msg):
    message = msg.encode(ENCODING)
    msg_length = len(message)
    msg_length = str(msg_length).encode(ENCODING)
    msg_length += b' ' * (MESSAGE_LENGTH_SIZE - len(msg_length))

    server.send(msg_length)
    server.send(message)
```

این دیکشنری هم به منظور نگهداری موضوعات و کلاینت‌های عضو هر موضوع استفاده می‌شود.

```
topics_members = {}
```

سمت کلاینت

در این سمت هم ابتدا پورت و آدرس استخراج شده و سپس ارتباط برقرار می‌شود.

```
def main():
    if len(sys.argv) <= 3:
        print("INVALID INPUT!")
        sys.exit()
    if sys.argv[1] == "default":
        # HOST = socket.gethostbyname(socket.gethostname())
        HOST = '127.0.0.1'
    else:
        HOST = sys.argv[1]
    if sys.argv[2] == "default":
        PORT = 1373
    else:
        PORT = sys.argv[2]

    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    SERVER_INFORMATION = (HOST, int(PORT))
    s.connect(SERVER_INFORMATION)
    client_msg(s)
```

پیام‌های دریافتی از سمت سرور دقیقا مانند سمت سرور دیکود شده و چاپ میشود. در این بخش این موضوع که اگر پس از ۱۰ ثانیه پیامی دریافت نشد هم پشتیبانی می‌شود تا مورد خواسته شده در دستور کار مبنی بر این عدم دریافت pubAck پیاده شود.

در صورتی که پیامی در موضوعی که کلاینت عضو است منتشر گردد در چاپ شدن پیام ارسالی از سمت سرور نمایش داده می‌شود. همچنین اگر در موضوعی عضو شده باشد یا برای بررسی ارتباط درخواست پینگ ارسال شده باشد در این تابع هندل می‌شود.

```python
def server_msg(conn: socket.socket):
    conn.settimeout(10.0)
    while True:
        message_length = int(conn.recv(MESSAGE_LENGTH_SIZE).decode(ENCODING))
        msg = conn.recv(message_length)
        # msg = client.recv(1024)
        if not msg:
            continue
        msg = msg.decode(ENCODING)
        print(msg)
        message = msg
        conn.settimeout(None)
        split_msg = message.split()
        if split_msg[0] == "subAck:":
            print("Subscribing on ")
            for m in split_msg[1:]:
                print(m)
        elif msg == "pubAck":
            print("Message published successfully")
            sys.exit()
        elif msg == "INVALID TOPIC!":
            print(msg)
            sys.exit()
        elif split_msg[0] == 'pong':
            # print("PONG!")
            sys.exit()
        elif split_msg[0] == 'ping':
            pong(conn)
```

پیام ارسالی کلاینت در این بخش با توجه به درخواستش به حالت استاندارد در این پروتکل تبدیل میشود.

```python
def client_msg(conn: socket.socket):
    while True:
        if sys.argv[3] == "subscribe":
            subscribe(conn, sys.argv[4:])
        elif sys.argv[3] == "publish":
            publish(conn, sys.argv[4:])
        elif sys.argv[3] == "ping":
            ping(conn)
        # elif sys.argv[3] == "pong":
        #     pong(conn)
        else:
            print("INVALID INPUT!")
            sys.exit()
        try:
            server_msg(conn)
        except socket.error:
            print("TIMEOUT: No response from server")
```

در صورتی که درخواست عضویت داده باشد در ابتدای رشته کلمه‌ی subscribe و بعد با فاصله به ترتیب موضوعات مورد نظر ارسال میشود.

```
def subscribe(conn: socket.socket, message):
    # split_msg = message.split()
    if len(message) < 1:
        print("NO TOPIC DETECTED!")
        print("Please try again.")
        sys.exit()
    msg = "subscribe"
    for m in message:
        msg += " " + m
    send_msg(conn, msg)
```

اگر درخواست انتشار داده باشد در ابتدا کلمه‌ی publish سپس به ترتیب موضوع و رشته‌ی مورد نظر ارسال میشود.

```
def publish(conn: socket.socket, message):
    msg = "publish "
    if len(message) == 0:
        print("NO TOPIC OR MESSAGE DETECTED!")
        sys.exit()
    if len(message) == 1:
        print("NO MESSAGE DETECTED!")
        sys.exit()
    msg += message[0] + " "
    message = message[1:]
    for m in message:
        msg += " " + m
    send_msg(conn, msg)
```

برای دو درخواست ping و pong هم پیام به شکل زیر ارسال می‌شود.

```python
def ping(client: socket.socket):
    send_msg(client, "ping")


def pong(client: socket.socket):
    send_msg(client, "pong")
```