پروژه دلال پیام

محدثه سادات اطیابی ۹۷۲۶۰۰۴

خلاصه ای از پروژه

در این پروژه، سرور نقش دلال پیام را دارد. درواقع روند پروژه به این صورت است که کلاینت هایی که می خواهند روی موضوع خاصی گوش دهند، به سرور درخواست subscribe میدهند و بصورت مستمر روی ارتباط گوش می دهند تا اطلاعاتی که روی آن موضوع ارسال می شود را دریافت کنند. بعضی از کلاینت ها هم هستند که می خواهند روی موضوع های خاصی اطلاعاتی را به همه کلاینت هایی که به آن گوش میدهند برسانند. در نتیجه، این کلاینت ها اطلاعات مورد نظرشان را بر روی سرور publish می کنند و سرور هم به عنوان دلال، این اطلاعات را به سایر کلاینت های مرتبط می رساند. دو دستور دیگر هم وجود دارد که کلاینت و سزوز می توانند به وسیله آنها، از برقرار بودن ارتباط اطمینان حاصل کنند. با ارسال ping و دریافت pong از طرف مقابل، مطمئن می شویم که ارتباط برقرار است.

پروتکل:

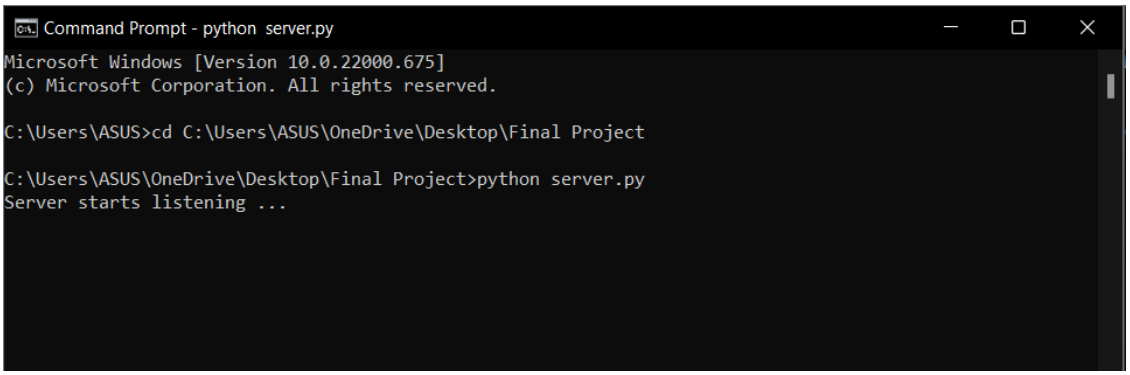
در این ارتباط از TCP استفاده می شود و به همین دلیل به IP address و port number برای برقراری ارتباط نیاز داریم. در ابتدا باید درخواست برقراری ارتباط ارسال شود و درصورت توافق طرفین، این ارتباط برقرار می شود.

برقراری ارتباط در این پروژه از طریق command line است و ورودی را هم در همین محیط می گیریم. تعداد آرگومان های ورودی در این پروژه بسته به دستوری که وارد می کنیم، متفاوت است اما همگی حداقل سه آرگومان ورودی دارند. آرگومان اول IP address سرور و آرگومان دوم port number آن است. در این پروژه کلاینت می تواند این دو آرگومان را بصورت دستی وارد کند و هم می تواند مقدار default را وارد کند که در اینصورت مقادیر 127.0.0.1 و 1370 تنظیم می شود. توجه کنید که در صورت پروژه از ما خواسته شده که روی پورت ۱۳۷۳ سرور را اجرا کنیم اما در سیستم من، این پورت توسط برنامه دیگری استفاده می شود و به همین دلیل از پورت ۱۳۷۰ استفاده می کنم. آرگومان سوم نوع دستوری است که می خواهیم سرور اجرا کند که می تواند یکی از سه مورد subscribe، publish و ping باشد.

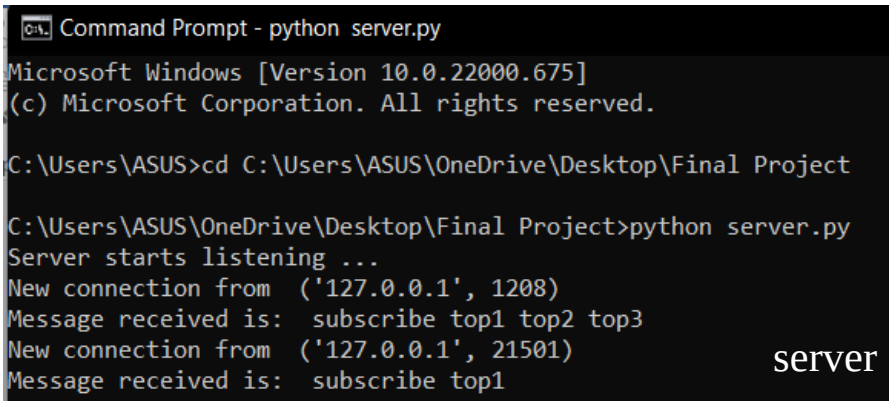
اگر دستور ما subscribe باشد، آرگومان چهارم به بعد موضوعاتی هستند که کلاینت می خواهد به آنها گوش دهد. اگر دستور publish باشد، آرگومان چهارم موضوعی است که می خواهیم درباره آن اطلاعاتی بفرستیم و آرگومان پنجم به بعد، اطلاعاتی هستند که باید ارسال شوند. در این حالت، موضوعی که انتخاب می کنیم حتما

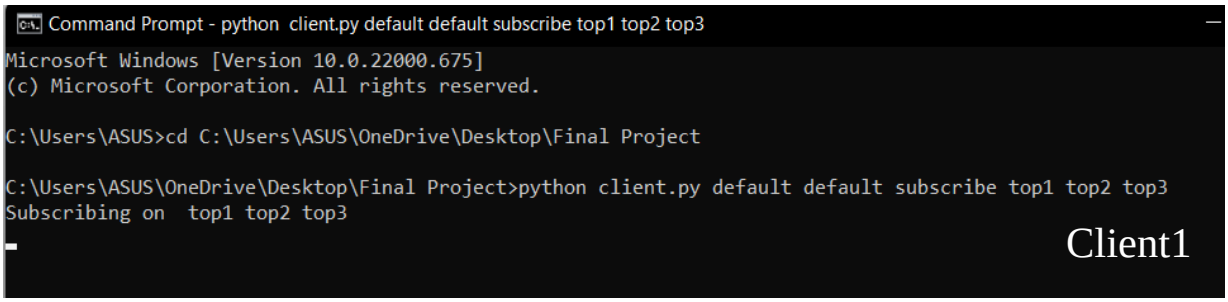
باید جزو موضاعاتی باشد که قبلا کسی آنها را subscribe کرده است و درحال گوش دادن به آن هستند. توجه کنید که همه آرگومان ها با white space از هم جدا می شوند. در ادامه سناریوهای مختلفی که ممکن است رخ دهد را بررسی می کنیم و خروجی های آنها را می بینیم.

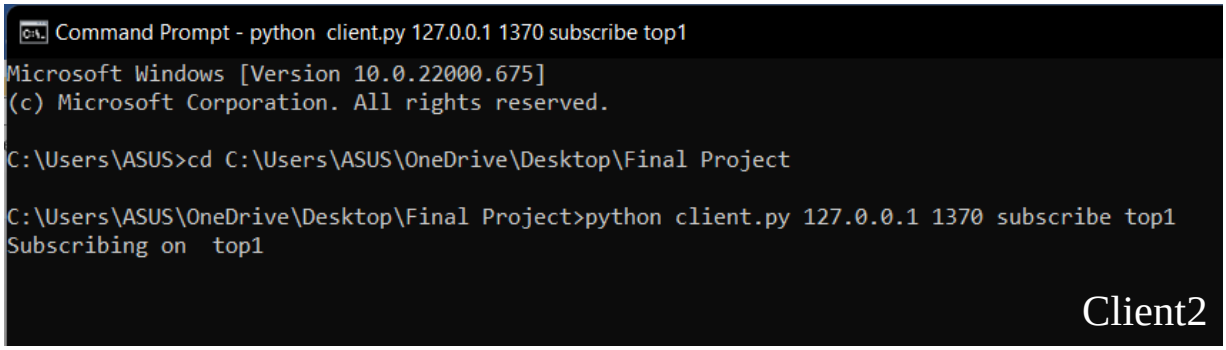
در ابتدا باید سرور را اجرا کنیم:



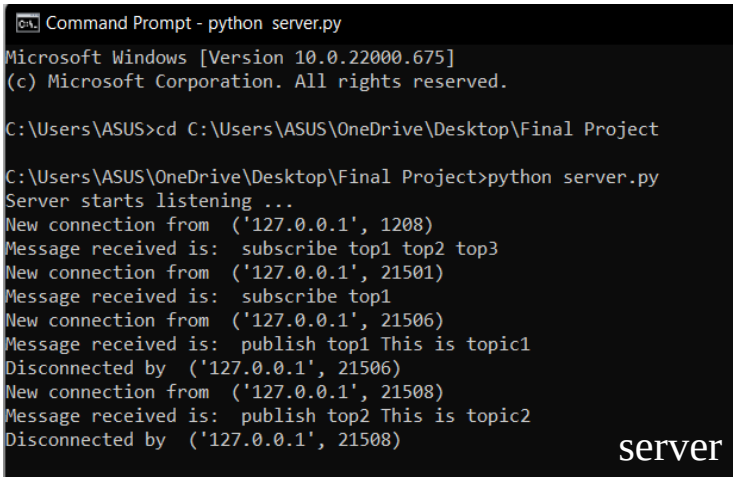
حال دو کلانت را با دستور subscribe به سرور متصل می کنیم. اولین کلاینت با دستور default متصل می شود و روی موضوعات top1، top2 و top3 گوش می دهد. کلاینت دوم با دادن آدرس 127.0.01 و شماره پورت 1370 به سرور متصل می شود و روی موضوع top1 گوش می دهد.

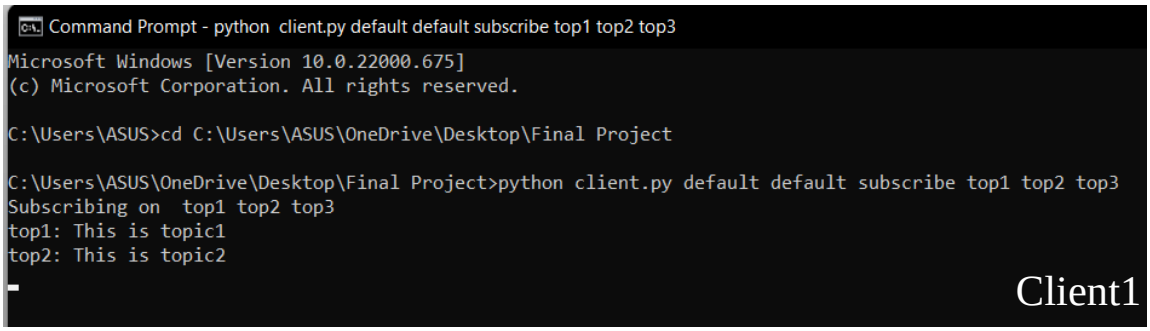


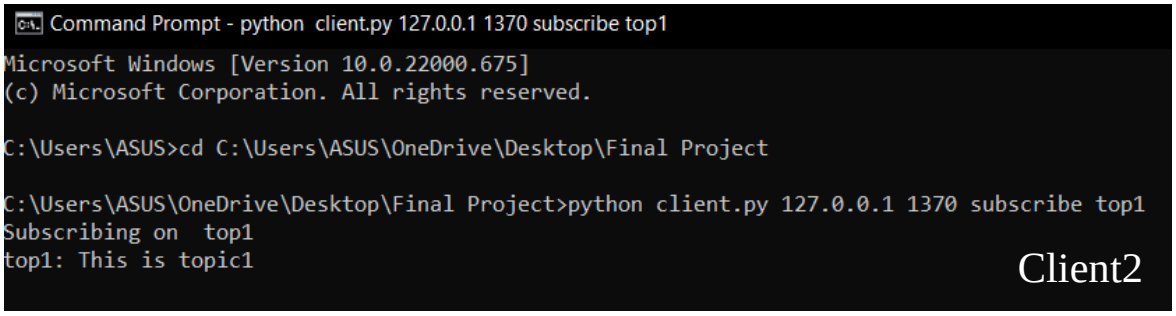


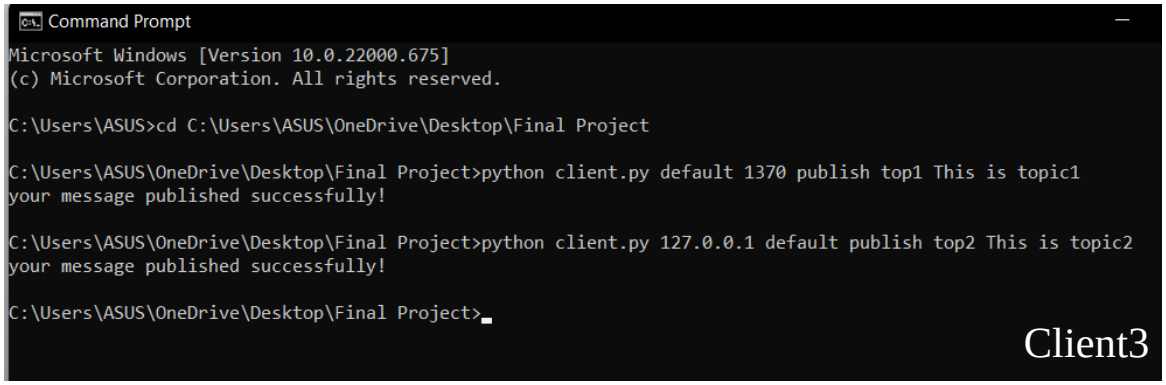


حال کلاینت سوم را فعال می کنیم به گونه ای که برای موضوع top1 اطلاعات This is topic1 را ارسال می کند و برای موضوع top2 اطلاعات This is topic2 را می فرستد. می بینیم که اطلاعات مربوط به top1 به هر دو کلاینت ۱ و ۲ ارسال می شود و اطلاعات مربوط به top2 تنها به کلاینت ۱ ارسال می شود. علاوه بر این می بنیم که هرکدام از آرگومان های اول و دوم را می توانیم بصورت عددی یا default وارد کنیم و الزاما این دو شبه بهم نیستند.

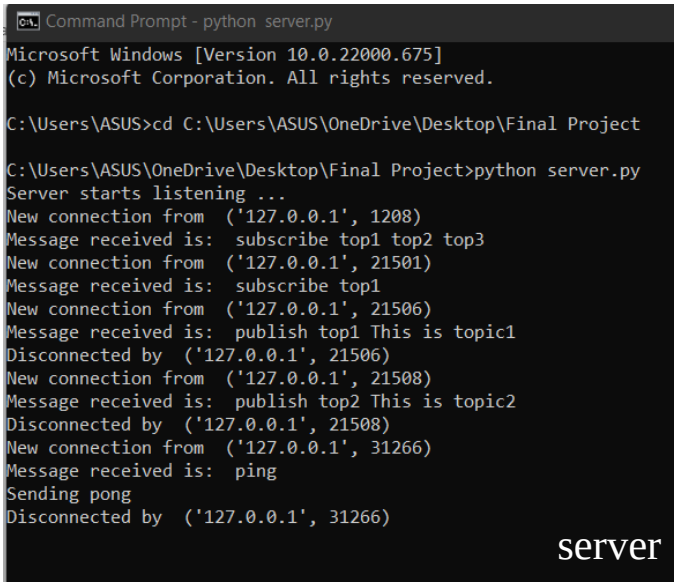


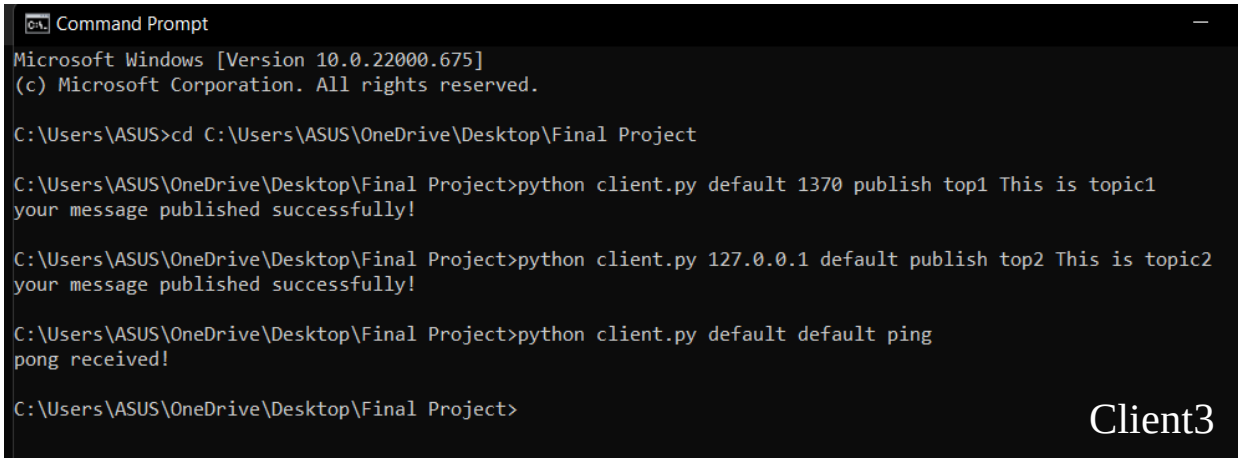




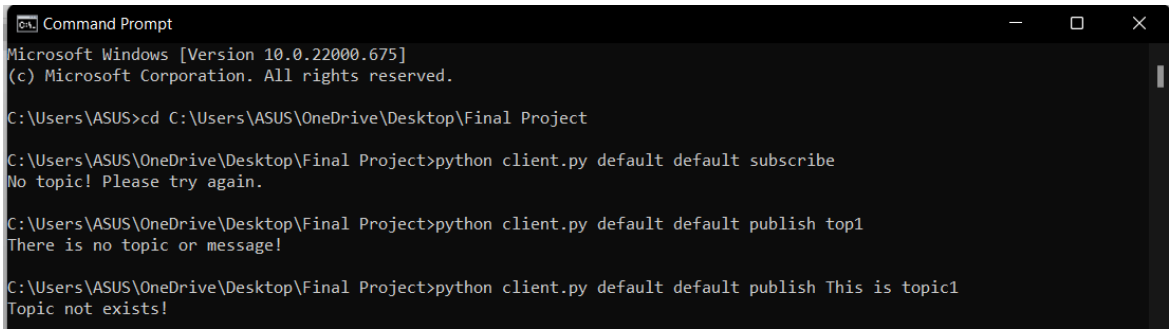


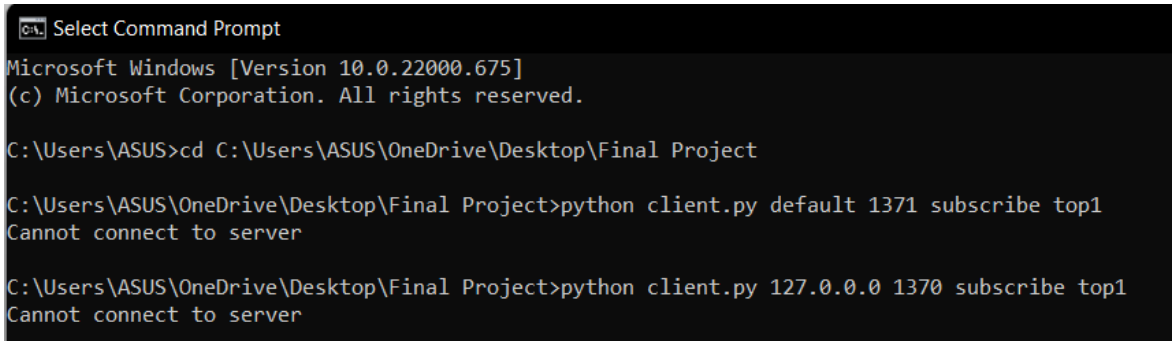
در آخر دستور ping را بررسی می کنیم و می بینیم که سزوزی که در حال اجراست، پاسخ pong را برمی گرداند.

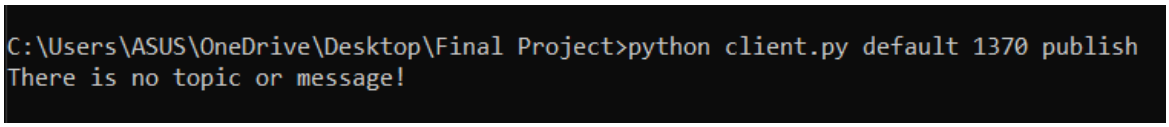




و در پایان دستورهایی که با خطا مواجه می شوند را بررسی می کنیم.







پیاده سازی:

سمت سرور

آدرس host را 127.0.0.1 و شکاره پورت را 1370 تنظیم می کنیم. داده ها بصورت کد اسکی تبادل می شوند و برای encode و decode کردن از asci استفاده می کنیم. یک دیکشنری بنام client_topic داریم که به ازای هر موضوعی که برای آن درخواست شنود داده می شود را به همراه اطلاعات کلاینت ذخیره می کند. در ابتدا تابع start_server فراخوانی می شود و یک سوکت TCP باز می کند. سپس روی آدرس IP و پورت داده شده، این سوکت bind می شود و شروع به گوش دادن به آن می کند. حال به ازای هر درخواستی که داده می شود، یک thread می سازد و برای آن thread، تابع handler را صدا می زند و آن را شروع می کند. توجه کنید که اطلاعات آدرس و پورت برای thread بر اساس کلاینت نظیرش تنظیم می شود و به همین دلیل با تابع accept این اطلاعات را استخراج می کنیم.

```python
import socket
import threading

HOST = '127.0.0.1'
PORT = 1370
ENCODING = 'ascii'

client_topic = {}  # It's like {"topic": "client"}


def start_server():
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.bind((HOST, PORT))
    s.listen()
    print("Server starts listening ...")
    while True:
        conn, address = s.accept()
        t = threading.Thread(target=handler, args=(conn, address))
        t.start()
```

حال تابع handler را بررسی می کنیم. در ابتدا اطلاعات آدرس IP و پورت کلاینتی که متصل شده است را چاپ می کنیم. سپس تا زمانیکه کلاینت داده ای براس ارسال داشته باشد، به آن گوش می دهیم. درهربار دریافت پیام، ابتدا طول آن را می گیریم و سپس به اندازه طول دریافتی، داده را دریافت می کنیم و در بافر میریزیم. باید طول دریافتی و خود پیام را decode کنیم چراکه بصورت کد اسکی ارسال می شوند. سپس پیام دریافتی را split می

کنیم تا آرگومان های مختلف آن را بدست آوریم. حال بر اساس آرگومان اول که نوع درخواست را مشخص کی کند، تصمیم می گیریم که چه کاری باید انجام شود. اگر ping را ببینیم، عبارت pong را بر می گردانیم و برعکس.

```
def handler(conn, address):
    print("New connection from ", address)
    while True:
        try:
            msg_len = int(conn.recv(1024).decode(ENCODING))
            msg = conn.recv(msg_len)
            if not msg:
                continue
            msg = msg.decode(ENCODING)
            print("Message received is: ", msg)
            split_msg = msg.split()
            if split_msg[0] == 'subscribe':
                subscribe(conn, split_msg)
            elif split_msg[0] == 'publish':
                publish(conn, split_msg)
            elif split_msg[0] == 'ping':
                print("Sending pong")
                send_message(conn, "pong")
            elif split_msg[0] == 'pong':
                send_message(conn, "ping")
        except:
            disconnect_client(conn)
            print("Disconnected by ", address)
            break
    conn.close()
```

یکی از عملیات ها subscribe است. در ابتدا بررسی می کنیم که آیا درخواستی با این عنوان داشتیم یا نه. اگر نداشته باشیم، key جدیدی ایجاد می کنیم و کلاینت را هم اضافه می کنیم. اما اگر این عنوان را قبلا داشته باشیم و این ملاینت قبلا برای آن درخواست نداده باشد، آن را به لیست این key اضافه می کنیم. سپس پیامی را با عنوان subACK تولید می کنیم و همه موضوعاتی که پذیرفته ایم را در آن قرار می دهیم و برای کلاینت به عنوان پاسخ درخواستش می فرستیم.

```
def subscribe(conn, split_message):
    for message in split_message[1:]:
        if message in client_topic.keys():
            if conn not in client_topic[message]:
                client_topic[message].append(conn)
        else:
            client_topic[message] = [conn]

    message = "subACK"
    for topic in client_topic.keys():
        if conn in client_topic[topic]:
            message += " " + topic
    send_message(conn, message)
```

برای درخواست publish، موضوع مورد نظر و اطلاعات مربوطه را بهم می چسبانیم و برای همه کلاینت هایی که به آن موضوع گوش میدهند، می فرستیم. برای کلاینتی که این اطلاعات را فرستاده هم پیام pubACK را می فرستیم تا بداند که عملیات publish کردن موفقیت آمیز بوده است. اگر ارسال برای سایر کلاینت ها موفقیت آمیز نباشد، آن کلاینت را حذف می کنیم از لیست. در صورتیکه موضوعی که برایش اطلاعات ارسال شده است در لیست نباشد و کسی به آن گوش ندهد، پیام Topic not exists! برای کلانت ارسال کننده اطلاعات ارسال می شود.

```
def publish(conn, split_message):
    #print(client_topic)
    message = split_message[1] + ":"
    for msg in split_message[2:]:
        message += " " + msg
    if split_message[1] in client_topic.keys():
        send_message(conn, "pubACK")
        for client in client_topic[split_message[1]]:
            try:
                send_message(client, message)
            except:
                disconnect_client(client)
    else:
        send_message(conn, "Topic not exists!")
```

برای حذف کلاینت از لیست و قطع ارتباط با آن لازم است تا اطلاعات آن را از لیست همه موضوعاتی که درحال گوش دادن به آنهاست، حذف کنیم. سپس ارتباط با آن را close می کنیم. حال چون در لیست کوضوعات تغییات ایجاد شده است، در پایان چک می کنیم که اگر براثر حذف یک کلاینت، یک لیست خالی شده است آن لیست را حذف کنیم تا کسی نتواند روی آن اطلاعاتی را ارسال کند.

برای ارسال پیام هم یک تابع داریم که در آن ابتدا طول پیامس که میخواهیم ارسال کنیم را به کد اسکی تبدیل می کنیم. از آنجا که بافر ارسال دارای اندازه ۱۰۲۴ است، فضای باقی مانده را با white space پر می کنیم تا اطلاعات اضافه ای ارسال نشود. درنهایت ایتدا طول پیام و سپس متن آن را ارسال می کنیم.

```
def disconnect_client(conn):
    changed = False
    temp = False
    for topic in client_topic:
        if conn in client_topic[topic]:
            temp = topic
            changed = True
            client_topic[topic].remove(conn)
    conn.close()
    if changed and len(client_topic[temp]) == 0:
        client_topic.pop(temp, None)


def send_message(conn, message):
    msg = message.encode(ENCODING)
    msg_len = str(len(msg)).encode(ENCODING)
    msg_len += b' ' * (1024 - len(msg_len))
    conn.send(msg_len)
    conn.send(msg)
```

سمت کلاینت

در این سمت، آدرس IP و پورت ثابت نیست و این اطلاعات را از کاربر می گیریم. در ابتدا بررسی می کنیم که اگر تعداد ورودی ها کمتر از ۴ تاست، ورودی معتبر نیست و نمی توانیم درخواستی را اجرا کنیم. بسته به اینکه کاربر آدرس IP و پورت را خودش وارد کرده یا از آدرس پیش فرض استفاده کرده است، مقادیر HOST و PORT را مقداردهی می کنیم. سپس روی آدرس و پورت مورد نظر یک سوکت TCP باز می کنیم. حال باید درخواست کاربر را بررسی کنیم. گفتیم که درخواست آرگومان سوم است و با بررسی آن، می توانیم انتخاب کنیم که چه کاری باید انجام دهیم. اگر دخواست یکی از موارد subscribe، publish و ping نباشد یعنی نامعتبر است و باید ارتباط را ببندیم. سپس تابع handler را صدا می زنیم که امکان timeout شدن داردو به همین دلیل این exception را هندل می کنیم.

```python
def start_client():
    # Check whether the inputs are enough to start the client or not
    if len(sys.argv) <= 3:
        print("Invalid input!")
        sys.exit()

    # Set the host address
    HOST = '127.0.0.1' if (sys.argv[1] == 'default') else sys.argv[1]
    # Set the port number
    PORT = 1370 if (sys.argv[2] == "default") else int(sys.argv[2])

    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect((HOST, PORT))
    while True:
        if sys.argv[3] == "subscribe":
            subscribe(s, sys.argv)
        elif sys.argv[3] == "publish":
            publish(s, sys.argv)
        elif sys.argv[3] == "ping":
            send_message(s, "ping")
        else:
            print("Invalid input!")
            sys.exit()
        try:
            handler(s)
        except socket.error:
            print("Timeout. No response from server")
            sys.exit()
```

در تابع handler در ابتدا یک تایمر داریم چراکه همه پیام هایی که می فرستیم دارای بازه زمانی ۱۰ ثانیه ای برای دریافت پاسخ هستند. سپس پیام ارسالی از سمت سرور را دریافت می کنیم که ابتدا طول آن را می فرستد و سپس خود پیام را ارسال می کند. هر دو عبارت دریافتی را decode می کنیم و تایمر را خاموش می کنیم. سپس پیام دریافتی را تحلیل می کنیم. اگر پیام Topic not exists! دریافت کرده باشیم، یعنی پیامی که میخواستیم publish کنیم موضوعی داشته که معتبر نیست و ارتباط را می بندیم. اگر subACK بگیریم یعنی درخواست subscribe پذیرفته شده. پس عبارت Subscribing on را برای موضوعات پذیرفته شده، چاپ می کنیم. اگر pubACK را بگیریم یعنی پیامی که می خواستیم publish کنیم با موفقیت انجام شده است. اگر ping بگیریم باید پاسخ pong را بدهیم و اگر pong را بگیریم یعنی ارتباط با سرور برقرار است و ارتباط را قطع می کنیم. اگر هیچکدم از این موارد نباشند، کافیست پیام را چاپ کنیم.

```python
def handler(conn):
    # Time limit of 10 seconds
    conn.settimeout(10.0)
    while True:
        msg_len = int(conn.recv(1024).decode(ENCODING))
        msg = conn.recv(msg_len)
        if not msg:
            continue
        msg = msg.decode(ENCODING)
        conn.settimeout(None)
        split_msg = msg.split()
        if msg == "Topic not exists!":
            print(msg)
            sys.exit()
        elif split_msg[0] == "subACK":
            text = "Subscribing on "
            for msg in split_msg[1:]:
                text += " " + msg
            print(text)
        elif split_msg[0] == "pubACK":
            print("your message published successfully!")
            sys.exit()
        elif split_msg[0] == "ping":
            send_message(conn, "pong")
        elif split_msg[0] == "pong":
            print("pong received!")
            sys.exit()
        else:
            print(msg)
```

در تابع subscribe ابتدا بررسی می کنیم که اگر طول درخواست+موضوع کمتر از ۱ است یعنی موضوع را نداریم. پس با خطا مواجه می شویم. اگر خطا نداشتیم، عبارت subscribe را به همرا موضوع و اطلاعات داده شده به عنوان یک پیام برای سرور ارسال میکنیم.

```
def subscribe(conn, message):
    if len(message[3:]) <= 1:
        print("No topic! Please try again.")
        sys.exit()
    msg = "subscribe"
    for message in message[4:]:
        msg += " " + message
    send_message(conn, msg)
```

در تابع publish ابتدا باید بررسی کنیم که طول موضوع+اطلاعات کمتر از ۲ نباشد. چراکه در اینصورت یعنی یا موضوع را نداریم و یا اطلاعات را نداریم که در هر دو صورت دارای خطا هستیم. اگر خطا نداشتیم، عبارت publish را به همراه اطلاعاتی که دریافت کرده این را در قالب پیام درمی آوریم و برای سرور ارسال می کنیم.

```
def publish(conn, split_message):
    if len(split_message[4:]) <= 1:
        print("There is no topic or message!")
        sys.exit()
    message = "publish "
    for msg in split_message[4:]:
        message += msg + " "
    send_message(conn, message)
```

برای ارسال پیام مانند سمت سرور، ابتدا طول پیام به کد اسکی را بدست می آوریم و سپس به اندازه اختلاف طول پیام و اندازه بافر (که ۱۰۲۴ است) از white space استفاده می کنیم تا مطمئن باشیم که اطلاعات دیگری همراه پیام ارسال نمی شود. در آخر هم طول پیام و سپس خود پیام را برای سرور ارسال می کنیم.

```
def send_message(conn, message):
    msg = message.encode(ENCODING)
    msg_len = str(len(msg)).encode(ENCODING)
    msg_len += b' ' * (1024 - len(msg_len))
    conn.send(msg_len)
    conn.send(msg)
```

چون کلاینت می تواند آدرس IP و ورت را وارد کند و ممکن است که مقادیر معتبری وارد نکند، تابع start_client را با exception ... try فراخوانی می کنیم تا اگر خطایی رخ داد، آن را هندل کنیم و سرور معتبر و فعال را فراخوانی کنیم.

```
if __name__ == '__main__':
    try:
        start_client()
    except socket.error:
        print("Cannot connect to server")
```