

INF443 QUIZ 2

Social Media Analyser

Onur Alp Gündüz - 21401958
Kerim Ayberk Çıtak - 20401870
Onur Çobanoğlu - 20401879

5 décembre 2024

1 Giriş

Bu rapor, dağıtık bir sohbet uygulamasının geliştirilmesi için hazırlanan Python tabanlı bir istemci-sunucu (client-server) mimarisinin teknik detaylarını açıklamaktadır. Uygulama, istemcilerin bir sunucu aracılığıyla birbirleriyle haberleşmesini sağlar. Bu iletişimde sunucu, istemci bağlantılarını yöneten ve mesajları hedeflerine ulaştıran bir aracı rolü üstlenir.

1.1 Amaç

Uygulamanın amacı, TCP/IP protokolü üzerinden çalışan, dağıtık ve çoklu istemci desteği sunan bir sohbet uygulaması geliştirmektir. Uygulamada aşağıdaki özellikler sağlanmıştır :

- İstemciler, kendilerini sunucuya özel bir tanıtım protokolüyle tanıtır.
- İstemciler arasında mesaj iletimi özel bir iletişim protokolüyle gerçekleştirilir.
- Sunucu, bağlantıları ve mesaj trafiğini yöneten merkezi bir kayıt tablosu tutar.
- Mesajlar ikili (binary) formatta iletilir.

1.2 Kullanılan Teknolojiler

- **Python 3.8+** : Kodlama için kullanılan programlama dili.
- **Socket Modülü** : TCP/IP bağlantılarını yönetmek için kullanıldı.
- **Threading Modülü** : Çoklu istemci desteği için çoklu iş parçacığı (thread) mimarisi kullanıldı.
- **Struct Modülü** : Mesajların ikili formatta işlenmesini sağlamak için kullanıldı.

2 Sistem Mimarisi

2.1 Genel Bakış

Sistem iki ana bileşenden oluşmaktadır :

1. **Sunucu (Server)** : İstemciler arasındaki mesaj trafiğini yönetir. Her istemcinin adını ve bağlantısını bir kayıt tablosunda tutar.
2. **İstemci (Client)** : Diğer istemcilere mesaj göndermek ve mesaj almak için sunucuyla iletişim kurar.

2.2 Protokoller

2.2.1 Tanıtım Protokolü

Bir istemci ilk kez bağlandığında, kendini sunucuya tanıtmak için özel bir mesaj gönderir. Bu mesaj şu formatta olmalıdır :

[name] [name]

Bu iki alan aynı olmalıdır, aksi takdirde sunucu bağlantıyı reddeder.

2.2.2 İletişim Protokolü

Mesaj gönderimi sırasında şu format kullanılır :

[receiver_name] [message_content]

receiver_name, mesajın hedefini belirtir ve message_content alanı mesajın içeriğini taşır.

3 Sunucu Kodu

```
def FindTargetAndSend(self, data):
    try:
        targetName, messageContent = struct.unpack("50s1024s", data)
        targetName = targetName.strip(b'\x00').decode('utf-8')
        messageContent = messageContent.strip(b'\x00').decode('utf-8')

        with lock:
            if targetName not in connectedNames:
                return f"{targetName} has not connected to server yet! Connected users: {connectedNames[:]}"

            targetIndex = connectedNames.index(targetName)
            selectedConnection = connectedCons[targetIndex]
            sentMessage = f"{messageContent} from {self.connectionName}"

            if self.connectionName == targetName:
                return "Can't send a message to yourself"

            selectedConnection.send(sentMessage.encode('utf-8'))
            return f"Message sent successfully from {self.connectionName} to {targetName}"
    except Exception as e:
        return f"Error: {str(e)}"
```

```

def run(self):
    global connectedNames, connectedCons

    while True:
        try:
            data = self.conn.recv(1074)
            if not data:
                break

            if not self.isNameSet:
                # Introduction Protocol
                try:
                    name1, name2 = struct.unpack("50s50s", data)
                    name1 = name1.strip(b'\x00').decode('utf-8')
                    name2 = name2.strip(b'\x00').decode('utf-8')

                    if name1 != name2:
                        self.conn.send(b"Error: Introduction names do not match")
                        continue

                    self.isNameSet = True
                    self.connectionName = name1

                    with lock:
                        connectedNames.append(self.connectionName)
                        connectedCons.append(self.conn)

                    print(f"{self.connectionName} added to table")
                    self.conn.send(f"{self.connectionName}, this is server. Welcome!".encode('utf-8'))
                except:
                    self.conn.send(b"Error: Invalid introduction format")
                    continue

            # Handle normal messages
            message = self.FindTargetAndSend(data)
            self.conn.send(message.encode('utf-8'))

        except Exception as e:
            print(f"Error: {e}")
            break

    # Cleanup
    with lock:
        if self.connectionName in connectedNames:
            index = connectedNames.index(self.connectionName)
            connectedNames.pop(index)
            connectedCons.pop(index)

    print(f"{self.connectionName} has been disconnected!")
    self.conn.close()

```

3.1 Sunucu İşlevselliği

Sunucu aşağıdaki işlevleri yerine getirir :

- Yeni istemcileri kabul eder ve kayıt tablosuna ekler.
- Mesajları protokol formatına göre işler ve hedef istemciye iletir.
- Bağlantısı kesilen istemcileri kayıt tablosundan kaldırır.

3.2 Hata Yönetimi

Sunucu, aşağıdaki durumlar için hata yönetimi uygular :

- Yanlış formatta gelen mesajlar reddedilir.
- Bağlantısı kesilen istemciler temizlenir.
- Hedef istemcinin mevcut olmaması durumunda hata mesajı döndürülür.

4 İstemci Kodu

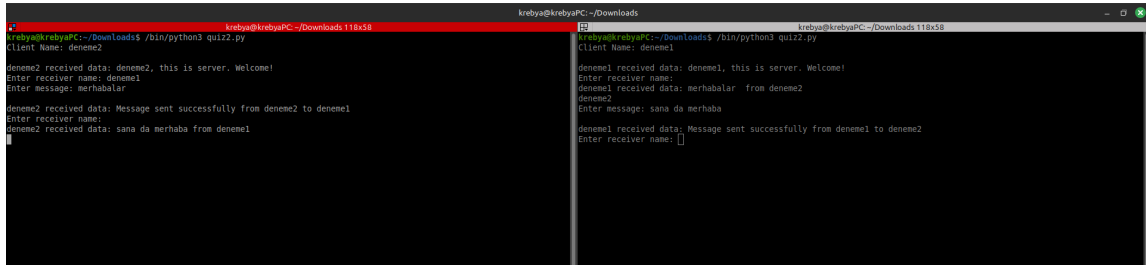
4.1 İstemci İşlevselliği

İstemci aşağıdaki işlevleri yerine getirir :

- İlk bağlantı sırasında kendini sunucuya tanıtır.
- Kullanıcıdan alınan girdiye göre diğer istemcilere mesaj gönderir.
- Sunucudan gelen mesajları dinler ve ekrana yazdırır.

4.2 Girdi ve Çıktı Formatı

- Kullanıcı, mesaj göndermek için şu formatı kullanmalıdır :
[receiver_name] [message]
- İstemci, sunucudan gelen mesajları konsola yazdırır.



```
krebys@krebysPC:~/Downloads$ ./bin/python3 quiz2.py
Client Name: deneme2
deneme2 received data: deneme2, this is server. Welcome!
Enter receiver name: deneme1
Enter message: merhabalar
deneme2 received data: Message sent successfully from deneme2 to deneme1
Enter receiver name:
deneme2 received data: sana da merhaba from deneme1
Enter receiver name:

krebys@krebysPC:~/Downloads$ ./bin/python3 quiz2.py
Client Name: deneme1
deneme1 received data: deneme1, this is server. Welcome!
Enter receiver name:
deneme1 received data: merhabalar from deneme2
deneme1
Enter message: sana da merhaba
deneme1 received data: Message sent successfully from deneme1 to deneme2
Enter receiver name:
```

5 Test ve Değerlendirme

5.1 Test Senaryoları

Aşağıdaki senaryolar test edilmiştir :

1. İstemcinin doğru tanıtım yapması.
2. Hedefi olmayan mesajların reddedilmesi.
3. Bir istemciden diğerine mesaj iletimi.
4. Çoklu istemcinin aynı anda bağlanması.

5.2 Sonuçlar

Tüm test senaryoları başarılı bir şekilde tamamlanmıştır. Sistem, beklenen işlevsellikleri yerine getirmiştir.

6 Sonuç ve Öneriler

Bu proje, dağıtık sistemler üzerinde çalışan basit bir sohbet uygulamasının nasıl geliştirileceğini göstermektedir. Gelecekte yapılabilecek iyileştirmeler şunlardır :

- Mesaj şifrelemesi için SSL/TLS entegrasyonu.
- Daha kullanıcı dostu bir arayüz (GUI) geliştirilmesi.
- Mesaj teslim garantisi için bir kuyruk sistemi entegrasyonu.