BLM3022 Ağ Teknolojileri

Proje Raporu

Proje İsmi: Soket Programlama ile Grup Sohbet Uygulaması

Kişilerin Çalışma Yüzdesi:

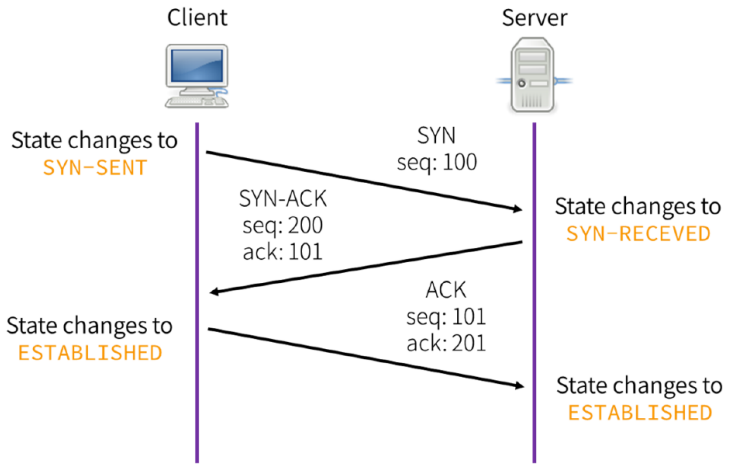
Grup Sorumlusu:	**Melih Ocakcı**	**25**	**Yazılım**
	Onur Demir	25	Yazılım, Rapor
	Oğuzhan Hayvacı	25	Rapor
	Ömer Talha Baysan	25	Rapor

1 Uygulamada Kullanılan Teknolojiler

Uygulamamızın amacı istemci-sunucu (client-server) mimarisi kullanarak istemciler arasında iletişim sağlamaktır. İşlemler arasında iletişim kurmak amacı ile soketler ile tcp bağlantıları kullanılmıştır.

1.1 TCP Tanım ve Çalışma Prensibi

TCP, Uygulama programlarının ve bilgi işlem cihazlarının bir ağ üzerinden mesaj alışverişinde bulunmasını sağlayan bir iletişim standardı olan İletim Kontrol Protokolü anlamına gelir. İnternet üzerinden paket göndermek,veri ve mesajların ağlar üzerinden başarılı bir şekilde teslim edilmesini sağlamak için tasarlanmıştır.TCP, internetin kurallarını tanımlayan temel standartlardan biridir.



Şekil 1: TCP 3-Way Handshake

TCP, verileri bir sunucu ve bir istemci arasında iletilebilecek şekilde düzenler. Bir ağ üzerinden iletilen verilerin bütünlüğünü garanti eder. TCP verileri iletmeden önce, bir kaynak ile hedef arasında bir bağlantı kurar ve iletişim başlayana kadar canlı kalmasını sağlar. Daha sonra büyük miktarda veriyi daha küçük paketlere bölerek süreç boyunca veri bütünlüğünün korunmasını sağlar.Sonuç olarak, veri iletmesi gereken üst düzey protokoller TCP Protokolü kullanır.

1.2 Ethernet

Ethernet, kablolu bir yerel alan ağında (LAN) veya geniş alan ağında (WAN) cihazları bağlamak için kullanılan geleneksel teknolojidir. Cihazların bir dizi kural veya ortak ağ dili olan bir protokol aracılığıyla birbirleriyle iletişim kurmasını sağlar.

Ethernet, aynı LAN veya kampüs ağındaki diğer cihazların bilgileri tanıması, alması ve işleyebilmesi için ağdaki cihazların verilerinin nasıl biçimlendirildiğini ve iletildiğini açıklar.

1.3 Network Layer (Ağ Katmanı)

Ağ katmanı,verilerin farklı ağlarda bulunan bir ana bilgisayardan diğerine iletilmesi için çalışır. Aynı zamanda paket yönlendirme (mevcut yol sayısından paketi iletmek için en kısa yolun seçilmesi) ile ilgilenir. Gönderici ve alıcının IP adresleri, ağ katmanı tarafından başlığa yerleştirilir. Ağ katmanının işlevleri şunlardır;

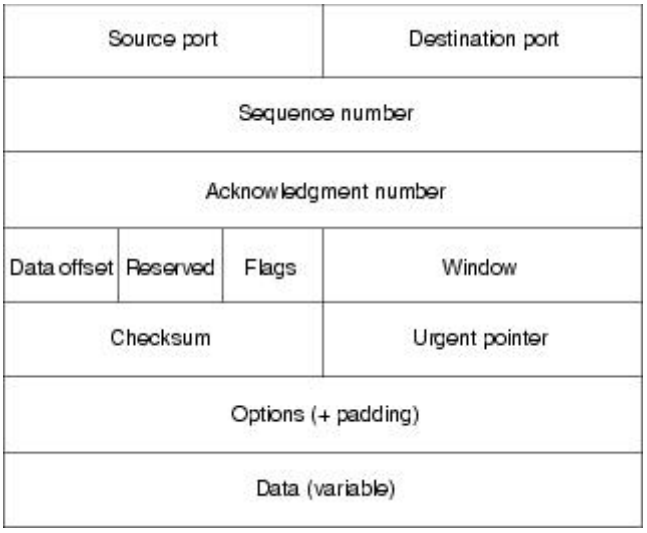
-Yönlendirme: Ağ katmanı protokolleri, kaynaktan hedefe hangi yolun uygun olduğunu belirler. Ağ katmanının bu işlevi yönlendirme olarak bilinir.

-Mantıksal Adresleme: Ağlar arası her cihazı benzersiz bir şekilde tanımlamak için ağ katmanı bir adresleme şeması tanımlar. Gönderici ve alıcının IP adresleri, ağ katmanı tarafından başlığa yerleştirilir. Böyle bir adres, her cihazı benzersiz ve evrensel olarak ayırt eder.

2 Uygulamanın Tanımı

2.1 Uygulama Katmanı

Biz uygulama katmanında soket programlama kullanıyoruz. Web ve HTTP gibi protokoller kullanmıyoruz çünkü herhangi bir şifreleme kullanmıyoruz. Sokete direkt olarak erişerek paketleri gönderiyoruz. Uygulamamızda message broker da kullanılan fan-out mekanizmasını broadcast fonksiyonu olarak basit soketler ile yazdık böylece message broker ın karmaşıklığından kurtulmuş olduk.



Şekil 2 : TCP frame WireShark ile sonraki sayfada görülmektedir.

```
package packet;
public Packet(String username, String message) {
        this.username = username;
        this.message = message;
}
```

Şekil 3 : Payload

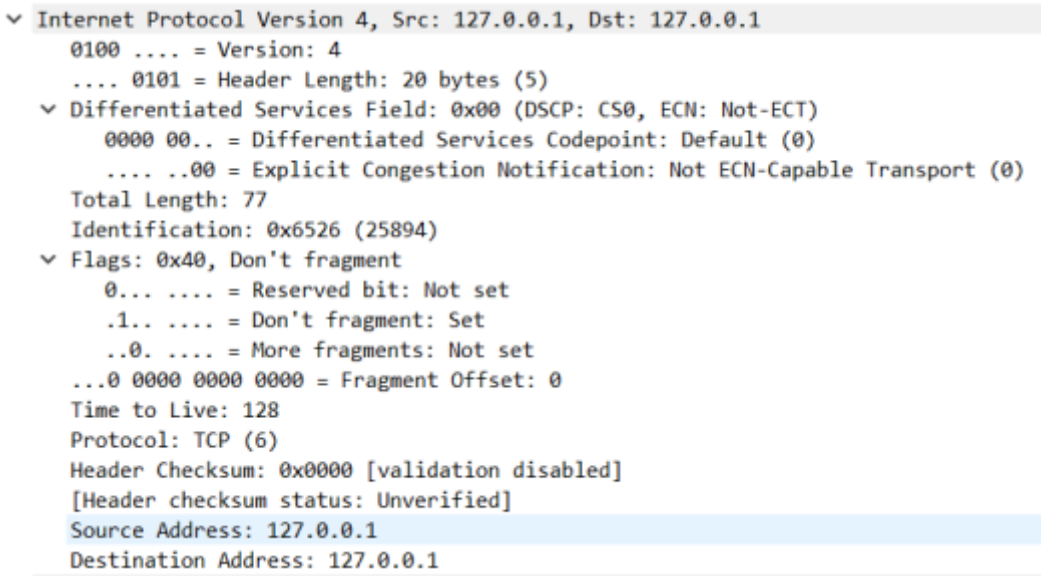
Kullandığımız payload Şekil 9' daki gibidir. Yeni oluşan client'a kullanıcı adı verilir ve mesajlarını burdan gönderirler.

```
0000  02 00 00 00 45 00 00 4d  65 26 40 00 80 06 00 00   ····E··M e&@·····
0010  7f 00 00 01 7f 00 00 01  26 11 1f 90 cc 08 d6 0e   ········ &·······
0020  56 00 4b 59 50 18 ff d3  c6 e1 00 00 73 71 00 7e   V·KYP··· ····sq·~
0030  00 00 74 00 17 73 61 64  66 61 73 64 66 61 73 64   ··t··sad fasdfasd
0040  66 73 61 64 66 73 61 64  66 73 64 66 71 00 7e 00   fsadfsad fsdfq·~·
0050  04                                                 ·
```

Şekil 4 : Paketin İçeriği

```
Transmission Control Protocol, Src Port: 9745, Dst Port: 8080, Seq: 1, Ack: 1, Len: 37
    Source Port: 9745
    Destination Port: 8080
    [Stream index: 1]
    [Conversation completeness: Incomplete (12)]
    [TCP Segment Len: 37]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 3423131150
    [Next Sequence Number: 38    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 1442859865
    0101 .... = Header Length: 20 bytes (5)
  Flags: 0x018 (PSH, ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgment: Set
      .... .... 1... = Push: Set
      .... .... .0.. = Reset: Not set
      .... .... ..0. = Syn: Not set
      .... .... ...0 = Fin: Not set
      [TCP Flags: ·······AP···]
    Window: 65491
    [Calculated window size: 65491]
    [Window size scaling factor: -1 (unknown)]
    Checksum: 0xc6e1 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  [Timestamps]
      [Time since first frame in this TCP stream: 0.000000000 seconds]
      [Time since previous frame in this TCP stream: 0.000000000 seconds]
  [SEQ/ACK analysis]
      [Bytes in flight: 37]
      [Bytes sent since last PSH flag: 37]
    TCP payload (37 bytes)
    TCP segment data (37 bytes)
```

Şekil 5 : TCP Header İçeriği



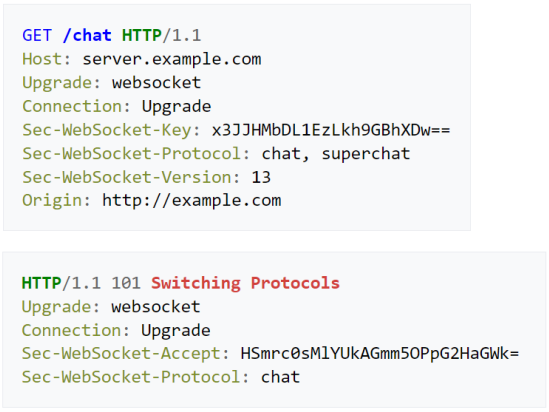
Şekil 6 : IP Protokol

3 Ek Uygulama

3.1 Araştırdığımız Geliştirilmiş TCP Bağlantı ve Protokoller

3.1.1 Websocket Nedir?

"Websocket"leri full duplex yapıda iletişimi sağlayan iletişim protokolüdür. "HTTP" de olduğu gibi "OSI" referans modelinin 7 katmanı olan uygulama katmanında çalışmaktadır ve "Websocket"leri "IETF" tarafından "RFC6455" dökümantasyonu ile 2011 de "Google" tarafından öne sürülmüştür. "Websocket"ler "HTTP" den farkı çift yönlü iletişim sağlamasıdır. Bu da grup sohbet uygulamasında kullanılmak için "HTTP" kullanımına göre daha uygun olduğunu göstermektedir. Çift yönlü browser-servis iletişimi "Comet" ve "Adobe Flash Player" multimedia aracının içerisinde kullanılmıştır fakat "TCP handshake" ve HTTP header(request header, response header, representation header, payload header gibi) yükünden dolayı küçük ölçekli mesajlarda örneğim end-to-end metin mesajlaşmasında yavaş kalmaktadır. WebSocket ws ve wss şeklinde iki şekildedir. Örnek bir websocket sitesi şu şekilde olabilir. "wss://game.examle.com/ws/updates"ve HTTP websocket desteklemektedir. Bununla ilgili örnek bir HTTP requesti aşağıdaki gibidir.



Şekil 7: WebSocket http request(isteği) ve response(cevabı).

Ayrıca websocket uygulamamızdaki "http://localhost:8080", HTTP uzantılıdır. HTTP'nin upgrade header'ına websocket yazılarak http websocket desteği, bu çağrı sadece başlangıç bağlantısında yapılmaktadır, sağlamaktadır. Böylelikle websocket, http bazında implementasyonu yapılmış aracılardan (proxies, firewalls, caches ve diğer aracılar (intermediaries)) faydalanmaktadır.

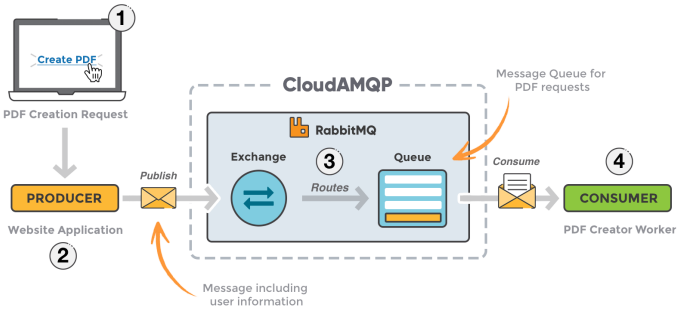
3.1.2 Neden WebSocketleri HTTP Yerine Tercih Edilebilir?

HTTP ve Websocket arasındaki kullanım açısından en büyük farkı stateful ve stateless olma durumudur. WebSocket stateful iken http REST yaklaşımı state-less'dır. HTTP ile gönderilen verinin durumu olmadığı için birden fazla server açılarak verinin işlenmesi sağlanabilir (bu avantaj horizontally scalable diye adlandırılır. Yan yana sunucular kurularak masraf azaltılır.) fakat bu avantaj tüm serverlerin yönetilmesini gerektirir bunun yerine stateful websocketler ile kuyruk mekanızmasından yararlanarak belli verileri depolayıp göndermek serveri daha verimli kılabilir fakat server sadece dikey olarak ölçeklenebilir yani tek bir serverin ram ve cpu kapasitesi artırılarak server kapasitesinde genişlik sağlanır. Ayrıca state(durum bilgisi) sayesinde erken gelen mesajlar tek bir servere yönlendirilebilir ve WebSocketler tek bir seferde büyük miktarda veri göndermek ve almak için kullanılabilir(high-load prensibi) bu yüzden grup sohbet odası için websocket kullanmak faydalıdır.

3.1.3 Message Broker Nedir?

Message broker uygulamaların birbiri ile iletişime geçmesini sağlayan bileşendir. Bir uygulamadan alınan mesajlar başka uygulamalara broadcast yapılabilir ya da mesajlar farklı uygulamalara filtrelenerek gönderilebilir.

Message Broker uygulamaları çoğunlukla kuyruk veri yapısı üzerinden , Şekil 2 de görüldüğü gibi çalışır.



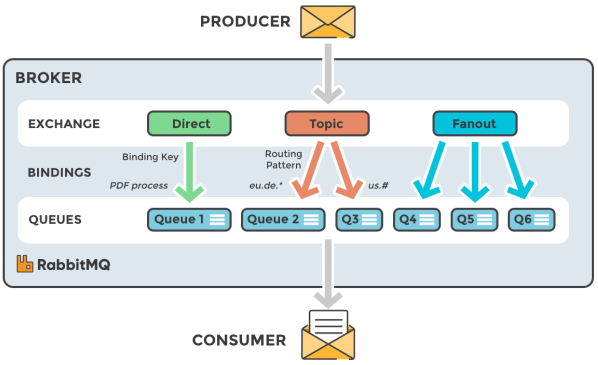
Şekil 8: Message Broker Arayüzü

Şekil 2 de görülen üç bileşene göre üretici mesajı üretir ve bunu message broker'a iletir. Üretici(Producer) message broker'a mesaj yollayan basit bir arayüzdür. Message Broker verilen mesajları tüketiciye gönderir tüketici veriyi yayma işlevi gören servis bileşenidir.

En iyi bilinen üç örnek message broker şunlardır; Kafka,RabbitMQ ve ActiveMQ dir. Bu kuyruklar mesajın depolanıp gönderilmesi ya da farklı uygulamalara yayılması rolünü üstlenir.

3.1.4 MessageBroker Nasıl Çalışır?

Üreticilerden verilen mesajlar message broker içerisinde yer alan exchange denilen yapılara iletilir. Exchange noktaları mesajı ilgili kuyruğa iletmekle görevlidir bunun için binding ve routing key kullanır. Kulandığı yaklaşıma göre exchange tipleri direk(direct), bölünme(fanout) , konu(topic) ve header olarak aşağıdaki şekilde görüldüğü gibi ayrılmaktadır.



Şekil 9: RabbitMQ takas(exchange) tipleri

Böylece queue ları yönetmek ve uygulamalara high load veriler iletmek daha kolaylaştırılır.

Bu da statefull soketlerle(websocket gibi.) daha verimli hale getirilir.

3.1.6 STOMP ve AMQP Protokolü Nedir?

STOMP (Simple (Streaming) Text Oriented Messaging Protocol) , bu protocol message brokerlar ile uyumlu text verisinin iletilmesi için kullanılan protokoldür. Server tarafında daha fazla işlem yükü gerektirir. Bu protocol aynı http de olduğu gibi bazı komutları fakat daha http gibi kapsamlı değildir. Bu komutlar şunlardır;

- Connect
- Send
- Subscribe
- Unsubscribe
- Begin
- Commit
- Abort
- ACK
- NACK
- DISCONNECT

aşağıda websocket aracılığı ile gönderilen CONNECT mesajına karşılık gelen ack frame görülmektedir.

```
var frame = "CONNECT\n"
            + "login: websockets\n";
            + "passcode: rabbitmq\n";
            + "nickname: anonymous\n";
            + "\n\n\0";
ws.send(frame);
```

Şekil 10 : STOMP null-terminated frame örneği

```
CONNECTED
session: <session-id>

^@
```

Şekil 11 : STOMP frame cevabı

AMQP(Advanced Message Queue Protocol) ise mesaj bazlı iletişimi sağlamak için kullanılan middleware dır. İçerisinde Şekil 3 de görüldüğü gibi routing , queuing, security algoritmaları bulunur. AMQP bir mesajın nasıl verileceği ve queue larda oluşacak flow ve congestion control lerini de içermektedir. Bu flow kontrollerini sağlamak için flow frame leri kullanılmaktadır. Diğer link protokolü frameleri şunlardır;

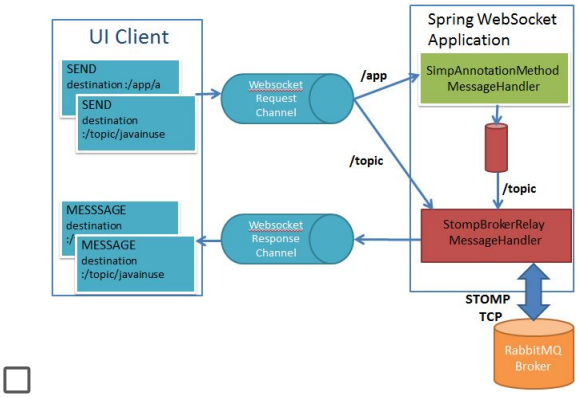
- open(connection)
- begin(session)
- attach(link)
- transfer
- flow
- disposition
- detach(link)
- end(session)
- close(connection)

Oluşturulan kuyruklara belli özellikler atanmaktadır. Bu özellikler ad, çalışma süresi, alacağı mesajın ttl süresi, kuyruğun uzunluğu, connection kapatıldığında queue nun kapatılıp kapatılmayacağı gibi.

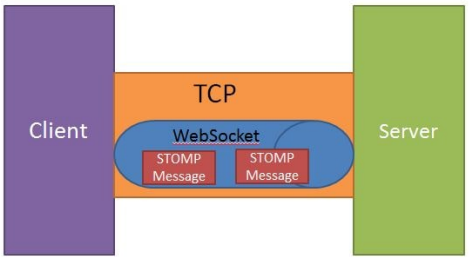
Bu protokolde consumer kişiler push ve pull yapabilme hakkına sahiptir. push hakkı ile kendilerine yönlendirilecek kuyrukları seçebilirler. Ayrıca exchange sırasında yönlendirilmeyen paketler silinebilir ya da depolanabilir bu framelere dead-letter denilmektedir.

3.1.7 Uygulama 2 :Spring Boot, WebSocket ve RabbitMq ile Grup Sohbet Uygulaması

Client spring boot kullanarak aldığı paket bilgisini rabbitMQ ya bir handler vasıtası ile stomp kullanan tcp bağlantısı ile iletir. UI kısmını springe bağlamak için "/websocketApp" "/app/chat.sendMessage" "/app/chat.newUser"gibi rpc ler kullandık. Bu RPC lerden ilki sockJS ile stomp clienti açmayı sağlar. Üçüncü prc benzer şekilde gönderim yapılmasını ve rabbitMQ ya subscribe yapılmasını sağlar böylece bağlantı ile ilgili kuyruk oluşturulur ve websocketin gönderimi sağlanır. Aşağıda İşlemler ve WebSocket Çağrısı figürleri görülmektedir



Şekil 12 : Uygulamanın Temel Modüllerinin İşlemsel Görünüşü



Şekil 13 : WebSocket TCP bağlantısı