SERVER-CLİENT SOCKET PROGRAMLAMA

Dr.Öğr. Üyesi Ahmet KARADOĞAN

Sunucu(Server) - İstemci(Client)

- **Sunucu**: Ağ üzerindeki diğer cihazlara (istemcilere) hizmet/veri sağlayan sistem

- **İstemci**: Sunucudan hizmet/veri alan sistem

Sunucu- İstemci Örnekleri

Kullanım Alanı	**Sunucu Örnekleri**	**İstemci Örnekleri**
İnternet ve Web Hizmetleri	Apache, Nginx, IIS, Tomcat	Google Chrome, Firefox, Edge, Safari
E-posta Sistemleri	Gmail, Outlook, Exchange Server	Microsoft Outlook, Apple Mail, Thunderbird, Gmail Uygulaması
Dosya Paylaşım ve Bulut Depolama	Google Drive, Dropbox, FTP Sunucuları	Google Drive Uygulaması, FileZilla
Online Oyunlar ve Multiplayer Sistemler	Minecraft, GTA V, PUBG, Call of Duty Sunucuları	Oyuncuların Bilgisayarları, PlayStation, Xbox

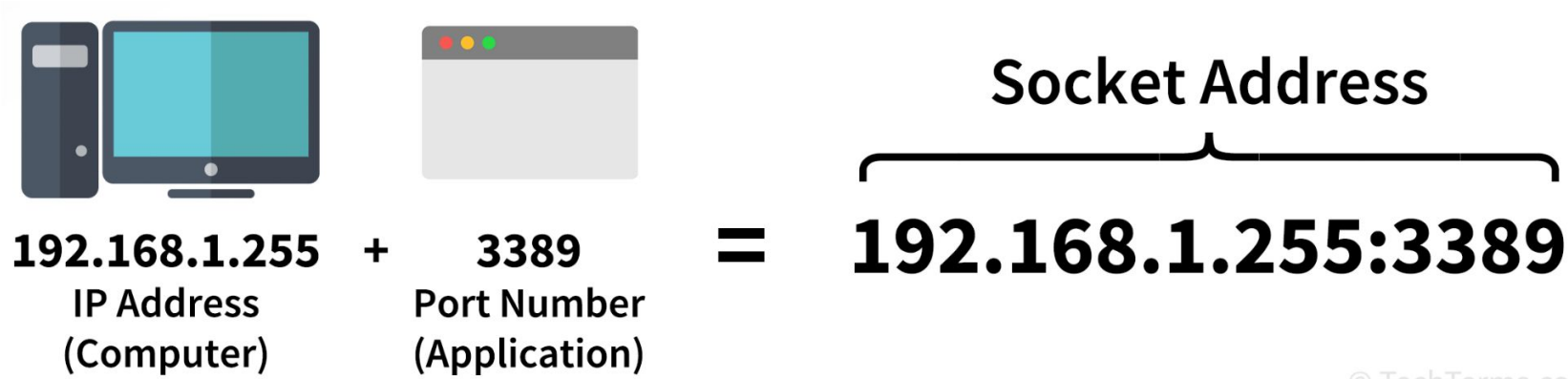
Sunucu- İstemci Örnekleri

Uzaktan Erişim ve Kontrol	Windows RDP, SSH Sunucusu, TeamViewer, Anydesk	Uzak Masaüstü Yazılımı, SSH İstemcisi (Terminal)
Canlı Yayın ve Streaming Servisleri	YouTube, Spotify, Twitch, Zoom, Meet	Mobil Uygulamalar, Tarayıcılar, Akıllı TV'ler
Bankacılık ve Finans Sistemleri	Banka Sunucuları	Mobil Uygulamalar, ATM'ler, POS Cihazları
Mesajlaşma ve Sosyal Medya	WhatsApp, Telegram, Twitter	Mobil Uygulamalar, Web Tarayıcıları

Socket

- Ağ üzerinde çalışan iki program/bilgisayar arasında çift yönlü iletişimi sağlayan bir uç nokta(end point)
- Bir sunucudaki uygulamaya bağlanmak için kullanılan bir kapı
- Socket, bir IP adresi ve bir port numarası ile tanımlanır.
- İletişim kurulacak uygulama, bir socket oluşturarak belirli bir IP adresi ve port numarasına bağlanabilir veya diğer uygulamaların bağlantı taleplerini kabul edebilir.

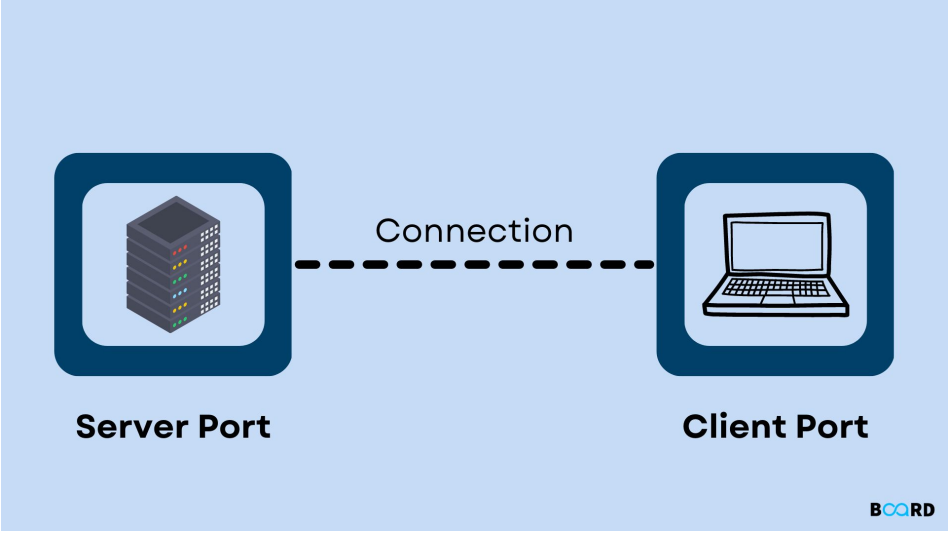
Socket


Socket Address
192.168.1.255
IP Address
(Computer)
+
3389
Port Number
(Application)
=
192.168.1.255:3389
© TechTerms.com

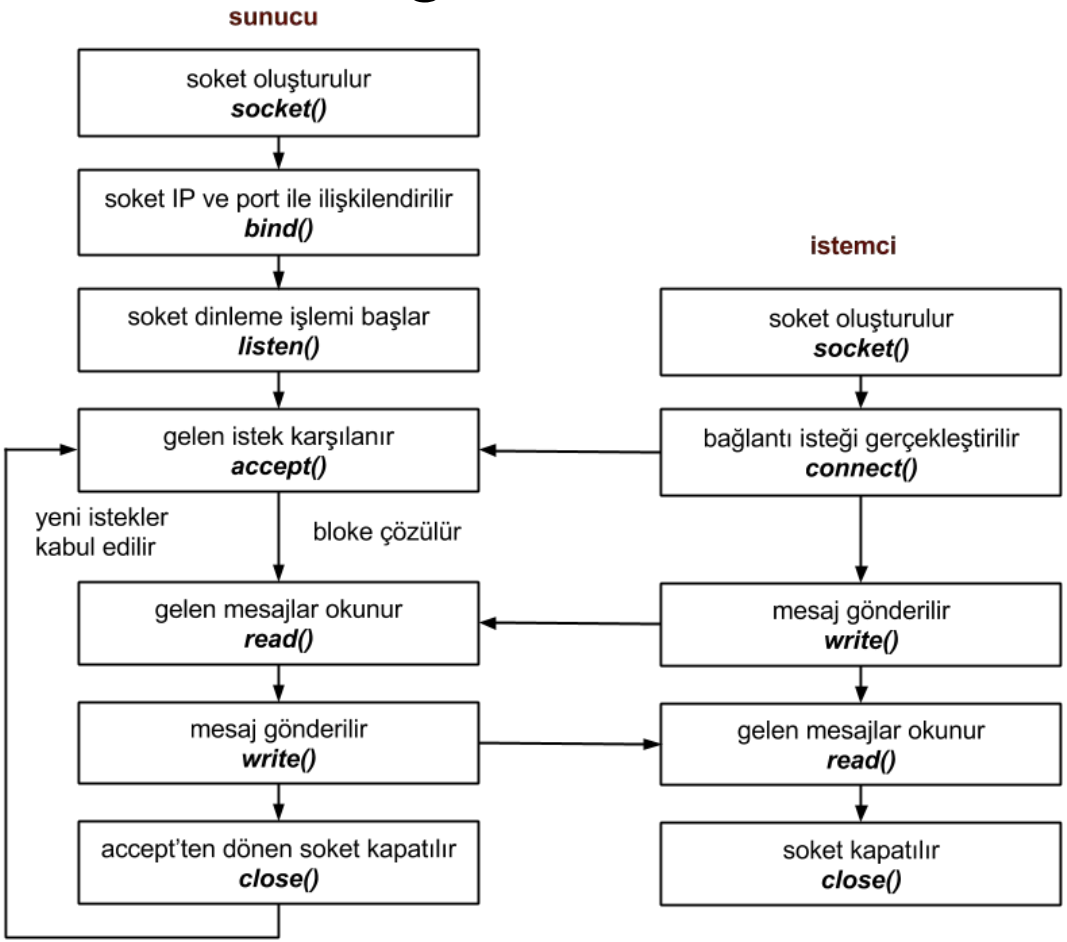
Socket

- **Sunucu(Server) socketi** : Gelen bağlantıları kabul etmek için kullanılır
- **İstemci(Client) socketi** :

  Bir sunucuya bağlanmak için kullanılır.



Sunucu-İstemci Bağlantısı



JAVA İLE
SOCKET PROGRAMLAMA

Java IP Adres Sorgulama

Local ip:

```
InetAddress adres=InetAddress.getLocalHost();
```

URL adı ile sorgulama:

```
InetAddress adres=InetAddress.getByName("inonu.edu.tr");
```

Adres bilgisi yazdırma:

```
System.out.println("adres:"+adres.toString());

System.out.println("host:"+adres.getHostName());

System.out.println("ip:"+adres.getHostAddress());
```

Java TCP Socket Sınıfları (Sunucu)

- Sunucu socketi açma:

```
ServerSocket serverSocket = new ServerSocket(PORT);
```

- İstemci bağlantısını kabul etme:

```
Socket socket = serverSocket.accept();
```

- İstemciye mesaj gönderme:

```
PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
out.println(msj);
```

- İstemciden mesaj alma (BufferedReader ile):

```
BufferedReader in=new BufferedReader(new InputStreamReader(socket.getInputStream()));
String gelenVeri = in.readLine();
```

- İstemciden mesaj alma (Scanner ile):

```
Scanner input = new Scanner(socket .getInputStream());
String gelenVeri = input.nextLine();
```

Java TCP Socket Sınıfları (İstemci)

- İstemci socketi oluşturma:

```
Socket socket = new Socket("localhost", PORT);
```

- Sunucuya mesaj gönderme:

```
PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
out.println("Merhaba, ben istemci!");
```

- Sunucudan mesaj alma(BufferedReader ile):

```
BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
String cevap= in.readLine();
```

- Sunucudan mesaj alma (Scanner ile):

```
Scanner input = new Scanner(socket .getInputStream());
String cevap= input.nextLine();
```

UDP (User Datagram Protocol)

- Verilerin iletimi için bağlantısız hizmet sunan protokol
- Hızlı veri iletimi gerektiren gerçek zamanlı uygulamalarda tercih edilir
- Hata kontrolü yapılmaz
- Bazı veri kayıpları olabilir, bunlar önemsenmez
- Veri iletiminde Datagram Paket denilen bağımsız veri birim nesneleri kullanılır

TCP (Transmission Control Protocol)	UDP (User Datagram Protocol)
Bağlantılıdır (Connection-Oriented) Sunucu ve istemci arasında sürekli bir bağlantı gerektirir.	Bağlantısızdır (Connectionless) Her veri paketi bağımsız olarak iletilir.
Güvenilirdir. Paketlerin eksiksiz ve sıralı ulaşmasını garanti eder.	Güvenilir değildir. Paket kaybı olabilir ve veriler sıralı gelmeyebilir.
Hata kontrolü yapar ve kayıp paketleri yeniden gönderir.	Hata kontrolü yapmaz ve kayıp paketleri yeniden göndermez.
Daha yavaş (Hata kontrolü ve doğrulama nedeniyle ek yük içerir).	Daha hızlı (Paketleri olduğu gibi gönderir, ek yükü azdır).
Dosya transferi (FTP), web tarayıcıları (HTTP/HTTPS), e-posta (SMTP, IMAP, POP3), veritabanı bağlantıları	Gerçek zamanlı ses/video akışı, online oyunlar, VoIP, DNS, DHCP
Güvenlik Duvarı geçişi daha kolaydır - Çoğu güvenlik duvarı TCP trafiğine izin verir.	Güvenlik Duvarı geçişi daha zordur - Güvenlik duvarları UDP trafiğini engelleyebilir.

UDP Socket Oluşturma (Sunucu)

- Sunucu socketi açma:
  ```
  DatagramSocket serverSocket = new DatagramSocket(PORT);
  ```
- Gelen datagram paketini tutmak için tampon bellek oluştur :
  ```
  byte[] buffer = new byte[256];
  ```
- Gelen Datagram paket nesnesi oluştur:
  ```
  DatagramPacket inPacket = new DatagramPacket(buffer, buffer.length);
  ```
- İstemci bağlantısını kabul etme:
  ```
  serverSocket.receive(inPacket);
  ```
- İstemciden mesaj alma:
  ```
  String gelenMesaj = new String(inPacket.getData(), 0, inPacket.getLength());
  ```
- İstemciye gönderilecek paketi oluşturma ve gönderme:
  ```
  DatagramPacket outPacket = new DatagramPacket(gidenMesaj.getBytes(),
gidenMesaj.length(), clientAddress, clientPort);
  serverSocket.send(outPacket);
  ```

UDP Socket Oluşturma (İstemci)

- Sunucu socketi açma (port numarası verilmez):

```
DatagramSocket socket = new DatagramSocket();
```

- Sunucu adresi tanımla

```
InetAddress host = InetAddress.getLocalHost();
InetAddress host = InetAddress.getByName("localhost");
```

- Giden Datagram paket nesnesi oluştur:

```
String mesaj = giris.nextLine();
DatagramPacket outPacket = new DatagramPacket(mesaj.getBytes(), mesaj.length(), host,
1234);
```

- Paketi gönder:

```
socket.send(outPacket);
```

- Sunucudan yanıt paketi alma ve stringe dönüştürme:

```
byte[] buffer = new byte[256];
DatagramPacket inPacket = new DatagramPacket(buffer, buffer.length);
socket.receive(inPacket);
String gelenMesaj = new String(inPacket.getData(), 0, inPacket.getLength());
```

UDP Socket Oluşturma (İstemci)-Ek Ayarlar

- Sunucudan gelecek yanıt için bekleme süresi belirlenebilir:

```
socket.setSoTimeout(3000);
```

- Gönderim işlemi ayrı bir döngü içerisine alınıp yanıt gelene kadar veya belli bir gönderim sayısı kadar gönderme işlemi denenmeye devam edilebilir. (Bu durumu test etmek için client çalışırken sunucu durdurulabilir)

```
do {
      socket.send(outPacket);
      ...
} while (cevapGeldi == false);
```

- Sunucu veya istemci tarafında gelen paketin adresini almak için

```
InetAddress gelenAddress = inPacket.getAddress();
int gelenPort = inPacket.getPort();
```