

BİLGİSAYAR AĞLARI

LABORATUVAR

1. Önkoşullar

- 1.1. Protokol kavramını bilmeli
- 1.2. Client-Server terimlerinin anlamlarını bilmeli.
- 1.3. Katmanlı mimariyi bilmeli
- 1.4. Java diline hakim olmalı
- 1.5. Veri yapılarını bilmeli
- 1.6. Thread ve Thread senkronizasyonu konularını bilmeli

2. TCP Nedir?

TCP (Transmission Control Protocol) Transfer Kontrol Protokolü demektir.

TCP/IP modelinde taşıma (transport) katmanında bulunan bir veri iletim protokolüdür. Bu katmanda veri gerekli büyüklükte paketlere (segment) bölünerek kapsülленir. Bu protokolde veriler parçalar halinde paketlenir ve hata denetimli olarak gönderilir. Bir veri paketi karşı tarafa hatalı ya da eksik iletilirse bunun kontrolü sağlanır ve paket yeniden gönderilir. Böylece hatasız veri akışı sağlanmış olur.

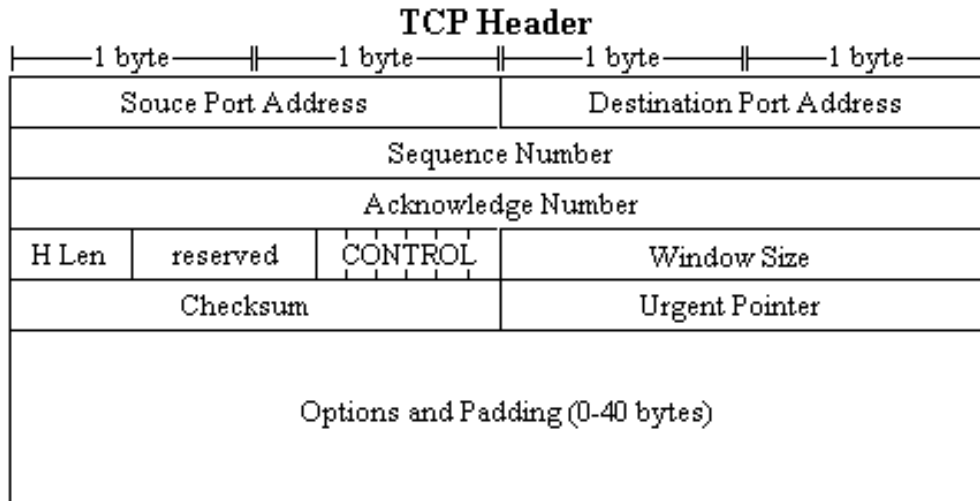
Hata denetimi TCP protokolüne ekstra yük getirir ve veri iletimini yavaşlatabilir. Bu yavaşlamaya rağmen eğer gönderilen verinin her bir ayrıntısı önemliyse bu transfer protokolü seçilir.

TCP:

- TCP “connection-oriented” çalışan bir taşıma (transport) katmanı protokolüdür.
- Verilere sıra numarası ekler. (Sequence Number)
- Veriler parçalara (segment) bölünerek yollanır.
- Güvenilirdir. Kaybolan veri tekrar yollanır. Verinin hedefe gidip gitmediğini kontrol eder.
- Akış kontrolü vardır. (Flow Control)

KC
FATİH
SULTAN
MEHMET
VAKIF ÜNİVERSİTESİ
2010

3. TCP Paket Başlıkları



Resim 1: TCP

- Source Port: Bir paket giderken geri gelecek cevapları dinleyen portun adresini de götürmelidir.
- Destination Port: Karşı bilgisayarda dinlenen port numarası bilgisi burada tutulur. Gönderilen port karşı tarafta dinlenmiyorsa paket iletilmez.
- Sequence Number: Paket sıra numarası
- Acknowledge Number: Onay numarası
- H len-Data Ofset: TCP başlığının uzunluğunu gösterir.
- Reserved: Daha sonradan kullanılmak üzere rezervedir.
- Control: 6 bit kontrol bilgisidir. Segment ile ilgili kontrol bilgilerini taşır.
 - URG: Urgent Pointer field significant
 - ACK: Acknowledgment field significant
 - PSH: Push Function
 - RST: Reset the connection
 - SYN: Synchronize sequence numbers
 - FIN: No more data from sender
- Window size: Alıcının ne kadarlık bir data alabileceğini gösterir.
- Checksum: Segmentin hatalı ulaşp ulaşmadığını kontrol etmek için kullanılır.
- Urgent Pointer: Bir verinin acil olarak iletilmek istendiği durumlarda kullanılır.
- Option and Padding: TCP verisinin bittiğini ve data kısmının geldiği bilgisini taşır.

4. Server-Client İletişimi

TCP client-server bağlantısında üçlü el sıkışma protokolü kullanılır. (Three way handshaking).

İletişim client tarafından başlatılır. Client SYN bitini 1 yapar ve rasgele bir sıra numarasını (-ISN- Initial Sequence number, ISN her işletim sisteminde farklı bir algoritma ile üretilir) sıra numarası yerine yazar, herhangi bir onay dönmediğinden onay kısmı 0'dır. Bu paket ilgili servera gönderilir.

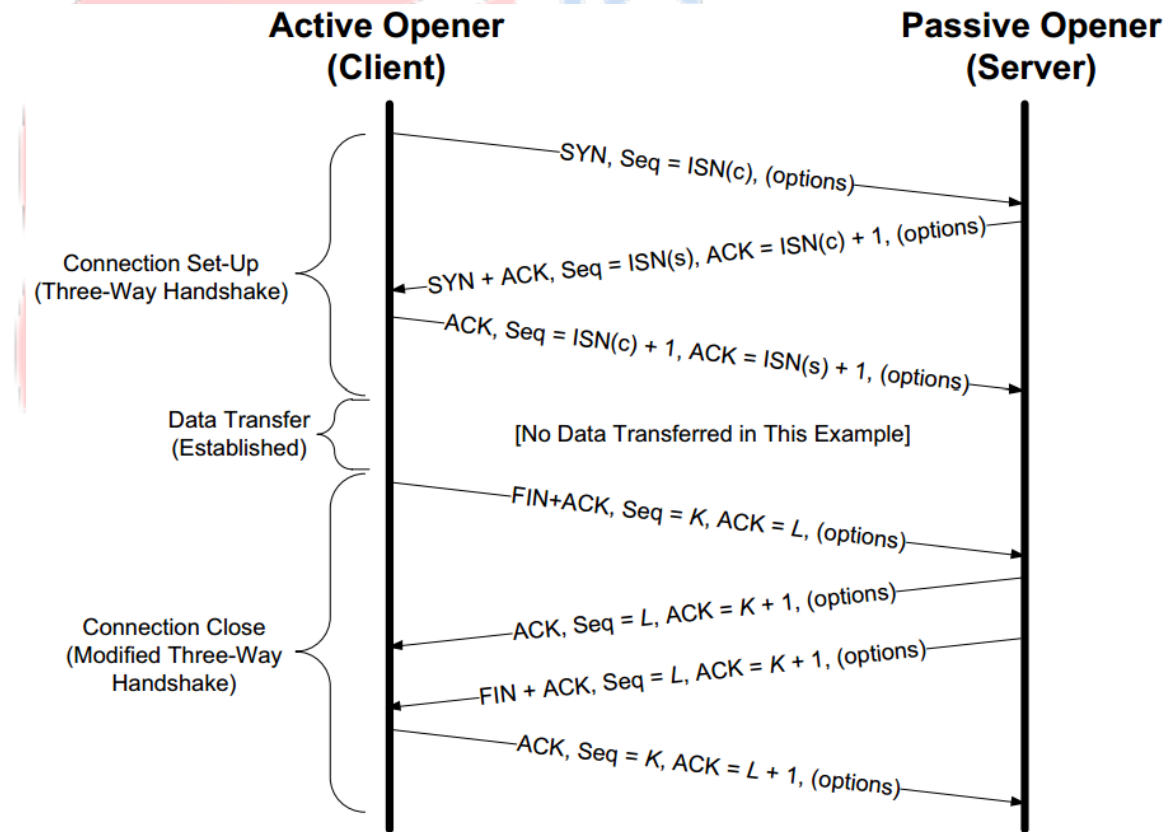
Server paketi aldığı anda SYN ve ACK biti 1 olan ve kendi ürettiği ISN sıra numarası taşıyan ayrıca client'tan aldığı paketin onay numarasını taşıyan ($ACK=ISN(c)+1$) başka bir TCP paket gönderir.

Bundan sonra client servera bir paket daha gönderir. Bu pakette ACK bayrağı 1'dir ve serverdan gelen paketin kabul numarası ($ACK=ISN(s)+1$), client'ın paket sıra numarası ise (önceki pakete verilen + 1).

Üçlü el sıkışma gerçekleşmiş, yani bağlantı kurulmuş olur. Artık veri transferi ve iletişim başlayabilir.

Haberleşme aşamasında kontrol, sıra ve kontrol verileri üzerinden olur.

İletişimi bitirmek için FIN biti 1 yapılarak gönderilir ve karşı taraftan FIN biti 1 olan bir mesaj gelir ve bağlantı koparılır.



Resim 2: TCP Bağlantı Kurma

5. TCP Soket Programlama

İnternet ağlarında soket programlama oldukça yaygın kullanılır. Soket ulaşılmak istenilen bilgisayarın IP değeri ve port numarasından meydana gelir. Bu değerleri kullanarak iletişime geçmek istediğimiz bilgisayara erişmiş oluruz.

Bilgisayarımızda port numarası olarak bir tam sayı (16-bit unsigned integer, ranging from 0 to 65535) atarız. Bu tam sayı 1024'ten küçük olamaz çünkü bu portlar özel işlemler için ayrılmıştır. Bunların dışında 65535 dolaylarına kadar port numaralarını kullanabiliriz. Yine de eğer bir port numarası kullanılmak isteniyorsa internet üzerinden kullanım durumunun araştırılması iyi olur. ([Referans: port numaraları listesi](#))

İşlemleri doğru yaptıysanız Resim 1' de görüldüğü gibi iki adet paket yakalamış olması gerekir. Bağlantı isteği client olan bilgisayarımızdan gitmektedir. Bu "GET" isteği olarak isimlendirilir. Bir başka deyişle bilgisayarımız belirli bir servera bir web sayfası için "GET" isteğinde bulunur. Buna karşılık karşı server HTML dosyasını client bilgisayarımıza indirir ve tarayıcı bu sayfayı görselleştirerek son kullanıcıya sunar.

Soket programlamada server-client terimleri de oldukça sık kullanılır. Server kendine gelen istekleri cevaplamak için sürekli dinlemede kalan bilgisayardır, dinlemede kalmayı sağlayan programa da server programı denir. Client ise server bilgisayarına istekte bulunan yani hizmet isteyen bilgisayar veya türevleridir.

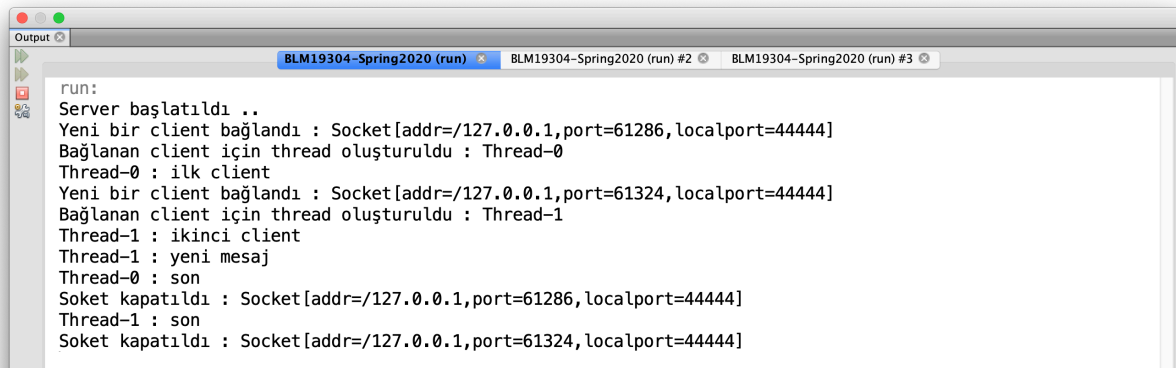
5.1. Java ile TCP Server Yazımı

Soket programlama tüm programlama dilleriyle yapılabilir. Fakat bunun kolaylığı zorluğu dilden dile değişir. Bu laboratuvarıda Java Programlama dili ile Server yazımını gerçekleştireceğiz.

Her Server yazılımı bilgisayarda bir port dinler ve bu porta bir mesaj geldiğinde server yazılımı gerekli işlemleri yapar. Java dilinde TCP server yazılımı için kullanılacak kütüphaneler:

- java.net.ServerSocket
- java.lang.Thread
- java.io

ServerSocket sınıfı yazacak olduğumuz server'ın soketini oluşturacak. io kütüphanesinden input ve output stream sınıflarını kullanacağız. Thread ise sürekli arka planda dinlemede kalmamızı sağlayacak olan sınıftır.



```
run:
Server başlatıldı ..
Yeni bir client bağlandı : Socket[addr=/127.0.0.1,port=61286,localport=44444]
Bağlanan client için thread oluşturuldu : Thread-0
Thread-0 : ilk client
Yeni bir client bağlandı : Socket[addr=/127.0.0.1,port=61324,localport=44444]
Bağlanan client için thread oluşturuldu : Thread-1
Thread-1 : ikinci client
Thread-1 : yeni mesaj
Thread-0 : son
Soket kapatıldı : Socket[addr=/127.0.0.1,port=61286,localport=44444]
Thread-1 : son
Soket kapatıldı : Socket[addr=/127.0.0.1,port=61324,localport=44444]
```

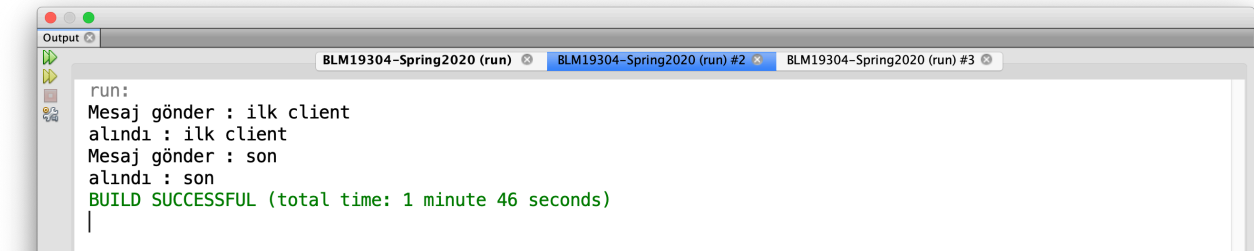
Resim 3: Server Konsol Çıktısı

5.2. Java ile TCP Client Yazımı

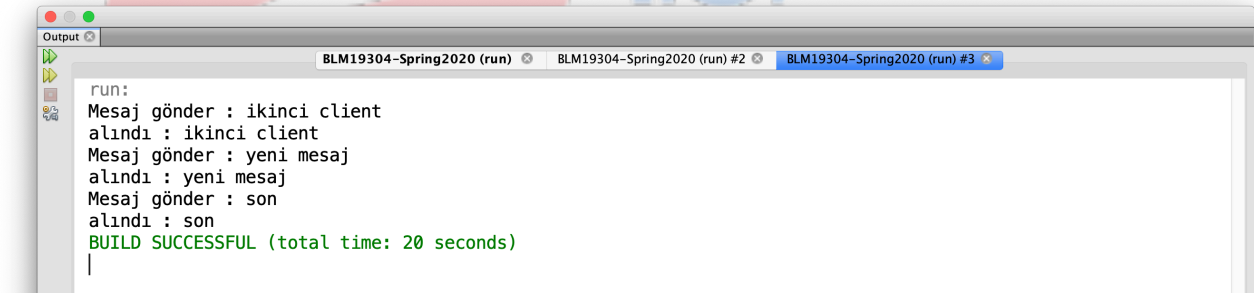
Serverda olduğu gibi TCP client da bir soket nesnesine sahip olmalıdır. Client serverdan farklı olarak “server ip” ve “server port” numaralarını bilmelidir. Bu bilgiler mesajın yollanacağı bilgisayar ve port numarası olacaktır. Client açık olduğu sürece kendi soketini sürekli dinler.

Java dilinde TCP server yazılımı için kullanılacak kütüphaneler:

- java.net.Socket
- java.io



Resim 4: Client-1 Konsol Çıktısı



Resim 5: Client-2 Konsol Çıktısı

Resim 3’de verilen server konsol çıktılarının elde edilebilmesi için işlemler şu sırayla yapılmalıdır:

1. Server java kodu çalıştırılır (Server)
2. Client java kodu çalıştırılarak ilk client’ın bağlanması sağlanır (Client-1)
3. Bağlanan ilk client’tan konsol kullanılarak “ilk client” mesajı gönderilir
4. Client java kodu çalıştırılarak ikinci bir client’ın bağlanması sağlanır (Client-2)
5. Bağlanan ikinci client’tan konsol kullanılarak “ikinci client” mesajı gönderilir
6. Bağlanan ikinci client’tan konsol kullanılarak “yeni mesaj” mesajı gönderilir
7. Bağlanan ilk client’tan konsol kullanılarak “son” mesajı gönderilir ve bağlantı koparılır
8. Bağlanan ikinci client’tan konsol kullanılarak “son” mesajı gönderilir ve bağlantı koparılır

Bu senaryo yukarıdaki resimlerde verilen çıktıların gerçekleşmesini sağlayan işlemleri içerir. Farklı şekillerde denemeler yaparak, birden fazla client’ın server bağlantısını test ediniz. GitHub’a yüklenen kodları anlamaya çalışarak küçük değişiklikler yapmaya çalışınız. Kullanımın daha kolay olması için arayüz giydirmesi yapabilirsiniz.