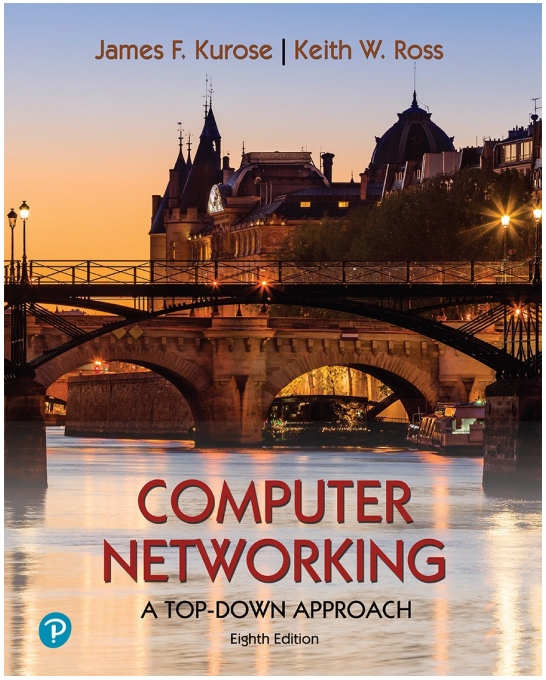
Bölüm 2
Uygulama Katmanı

Doç. Dr. Mehmet Dinçer Erbaş
Bolu Abant İzzet Baysal Üniversitesi
Mühendislik Fakültesi
Bilgisayar Mühendisliği Bölümü



Computer Networking: A Top-Down Approach
8th edition
Jim Kurose, Keith Ross
Pearson, 2020

Bölüm 2: konular

2.1 Ağ uygulamalarının prensipleri

2.2 Web and HTTP

2.3 FTP

2.4 Electronik posta

- SMTP, POP3, IMAP

2.5 DNS

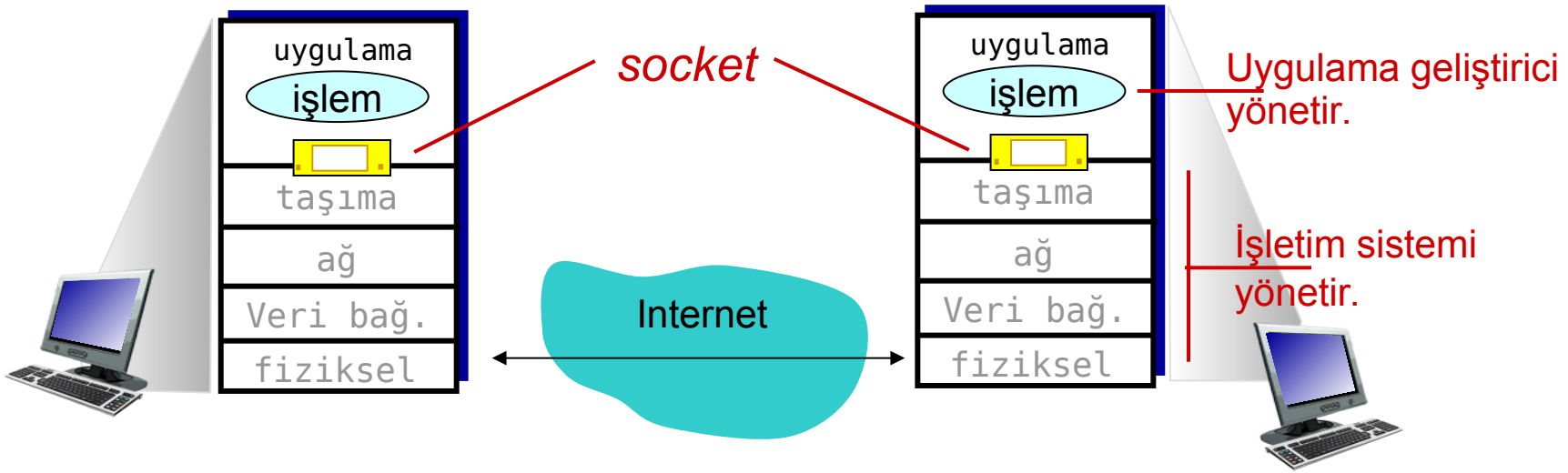
2.6 P2P uygulamaları

2.7 UDP ve TCP ile soket programlama

Soket programlama

hedef: Sunucu/istemci mimarisi ile soket kullanarak uygulama yazmayı öğrenelim.

soket: uygulama işlemleri ve uçtan uca taşıma protokolü arası kapı.



Soket programlama

Taşıma hizmeti için iki ayrı soket tipi:

- *UDP:* güvenilir olmayan datagram
- *TCP:* güvenilir, byte yayın odaklı

Uygulama örneği:

1. İstemci klavyeden bir karakter satırı okur (veri) ve veriyi sunucuya gönderir.
2. Sunucu veriyi alır ve karakterleri büyük harfe çevirir.
3. Sunucu değiştirilmiş veriyi istemciye gönderir.
4. İstemci değiştirilmiş veriyi alır ve ekranında gösterir.

UDP ile soket programlama

UDP: sunucu ile istemci arasında "bağlantı" yok.

Veri göndermeden önce elsıkışma yapılmaz.

Gönderici IP heden adresi ve port numarasını her pakete iliştirir.

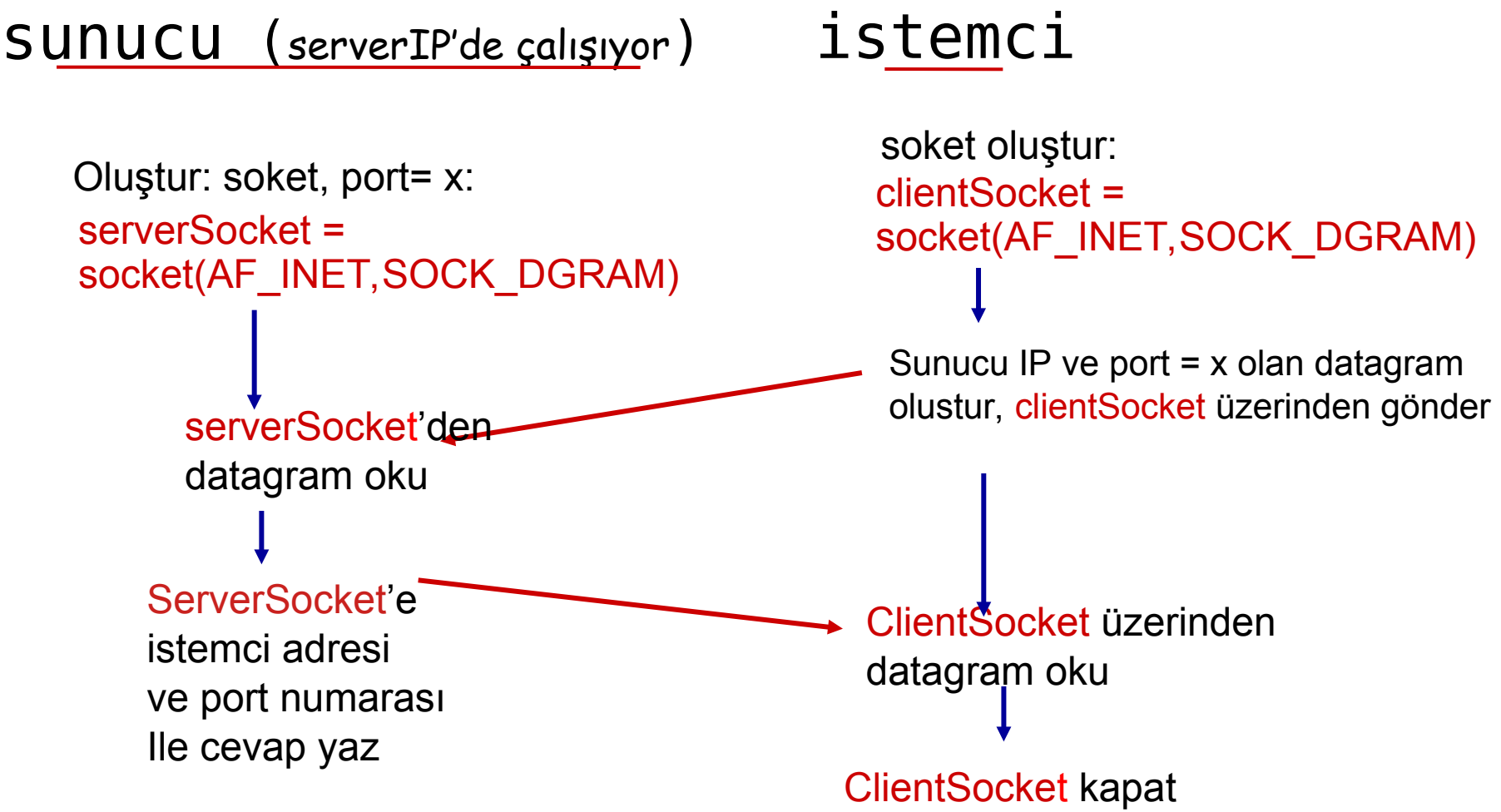
Alıcı alınan paketten IP adresi ve port numarası

UDP: gönderilen veri kaybolabilir veya sırası bozulmuş şekilde alınabilir.

Uygulama tarafından bakıldığında:

UDP güvenilir olmayan şekilde byte gruplarının ("datagtamlar") istemci ile sunucu arasında taşınmasını sağlar.

İstemci/sunucu soket etkileşimi: UDP



Örnek uygulama: UDP istemci

Python UDPClient

Python soket kütüphanesi

Sunucu için UDP soketi oluştur

Kullanıcıdan klavye girdisi al

Sunucu adresini ve port numarasını mesaja ekle; sokete gönder

Soketten okuduğun karakterleri string'e çevir

Alınan mesajı yazdır ve soketi kapat

```
from socket import *
serverName = 'hostname'
serverPort = 12000
clientSocket = socket(socket.AF_INET,
                      socket.SOCK_DGRAM)
message = raw_input('Input lowercase sentence:')
clientSocket.sendto(message,(serverName, serverPort))
modifiedMessage, serverAddress =
                      clientSocket.recvfrom(2048)
print modifiedMessage
clientSocket.close()
```

Örnek uygulama: UDP sunucu

Python UDPServer

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('', serverPort))
print "The server is ready to receive"
while 1:
    message, clientAddress = serverSocket.recvfrom(2048)
    modifiedMessage = message.upper()
    serverSocket.sendto(modifiedMessage, clientAddress)
```

UDP soket oluştur

Soketi yerel port numarası 12000 ile eşleştir.

Sonsuz döngü

UDP soketten mesajı oku, istemcinin adresi al (istemci ip ve port)

Istemciye büyük harfli mesajı geri gönder

TCP ile soket programlama

Istemci sunucuya erişmeli

Sunucu işlemi çalışıyor olmalı.

Sunucu, istemcinin erişimini karşılayan soketi (kapıyı) önceden oluşturmalı.

Istemci sunucu ile şu şekilde iletişime geçer:

Sunucu işleminin IP adresi ve port numarasını belirterek TCP soketi oluşturur.

Istemci soket oluşturduğunda:
istemci TCP tarafı sunucu TCP tarafına bağlantıyı oluşturur.

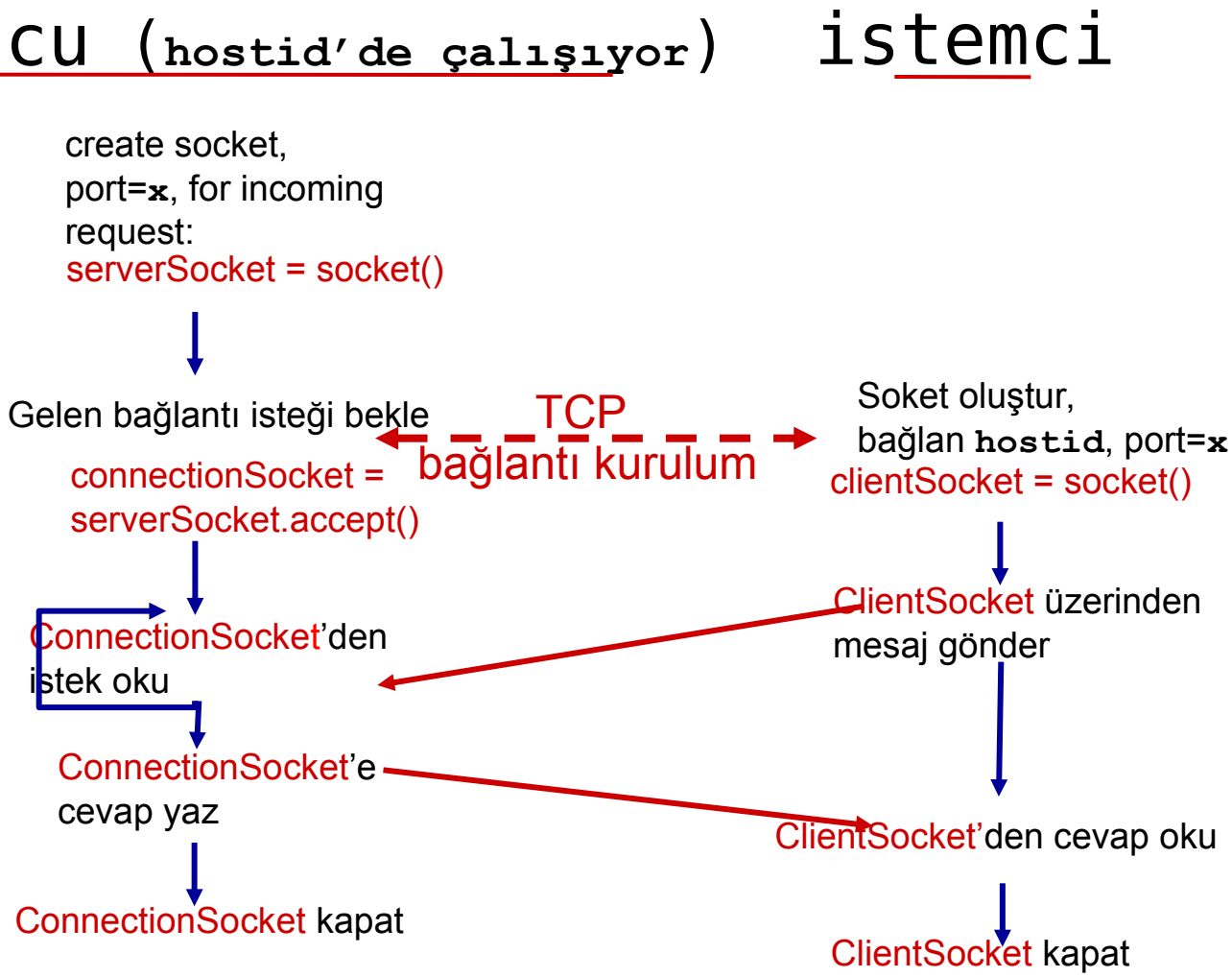
Istemci kendisi ile iletişime geçtiğinde, sunucu TCP tarafı sunucu işleminin iletişime geçen istemci ile haberleşebilmesi için yeni soket oluşturur.

- Bu sayede sunucu birden fazla istemci ile konuşabilir.
- Kaynak port numarası ile farklı istemciler belirlenebilir.

Uygulama tarafından bakıldığında

TCP istemci ve sunucu arasında güvenilir, doğru sıralı byte-akışı şeklinde taşıma (“pipe”) sağlar.

İstemci/sunucu soket etkileşimi: TCP



Örnek uygulama: TCP istemci

Python TCPClient

```
from socket import *
serverName = 'servername'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
sentence = raw_input('Input lowercase sentence:')
clientSocket.send(sentence)
modifiedSentence = clientSocket.recv(1024)
print 'From Server:', modifiedSentence
clientSocket.close()
```

Sunucu için TCP soketi oluştur, uzak port 12000

Sunucu ismi ve port numarası eklemeye gerek yok.

Örnek uygulama: TCP sunucu

Python TCPServer

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind(('',serverPort))
serverSocket.listen(1)
print 'The server is ready to receive'
while 1:
    connectionSocket, addr = serverSocket.accept()

    sentence = connectionSocket.recv(1024)
    capitalizedSentence = sentence.upper()
    connectionSocket.send(capitalizedSentence)
    connectionSocket.close()
```

TCP karşılama soketi oluştur

Sunucu gelen TCP istekleri için dinlemeye başlar.

Sonsuz döngü

Sunucu gelen istekler için accept() fonksiyonunda bekler, dönüş yaparsa yeni soket oluşturulur.

Soketten mesaj okunur (ancak adres UDP benzeri okunmaz)

Bu istemciye soket kapanır (ancak karşılama soketi halen açık)

Bölüm 2: özet

Ağ uygulaması üzerine konumuz tamamlandı.

- Uygulama mimarileri
  - istemci-sunucu
  - P2P
- Uygulama hizmeti ihtiyaçları
  - güvenilirlik, bant genişliği, gecikme
- Internet taşıma hizmeti modeli
  - Bağlantı-odaklı, güvenilir: TCP
  - Güvenilir olmayan, datagram: UDP

- Belli protokoller:
  - HTTP
  - FTP
  - SMTP, POP, IMAP
  - DNS
  - P2P: BitTorrent
- soket programlama: TCP, UDP soketleri

Bölüm 2: özet

En önemli konu: protokoller hakkında bilgi aldık.

- Tipik istek/cevap mesaj değişimi:
  - Istemci bilgi veya hizmet ister.
  - Sunucu veri, durum kodu ile karşılık verir.
- mesaj formatı:
  - başlık: veri hakkında bilgi veren alanlar.

veri: iletilen bilgi.

Önemli konular:

- kontrol vs. veri mesajları
  - bant-içi, bant-dışı
- merkesi vs. dağınık
- Durum bilgisi tutmayan vs. durum bilgisi tutan
- güvenilir vs. güvenilir olmayan msj transferi
- “karmaşıklık ağ kenarına”