BSM 458-AĞ PROGRAMLAMA

Hafta5: Soket Kavramı ve Soket Programlama

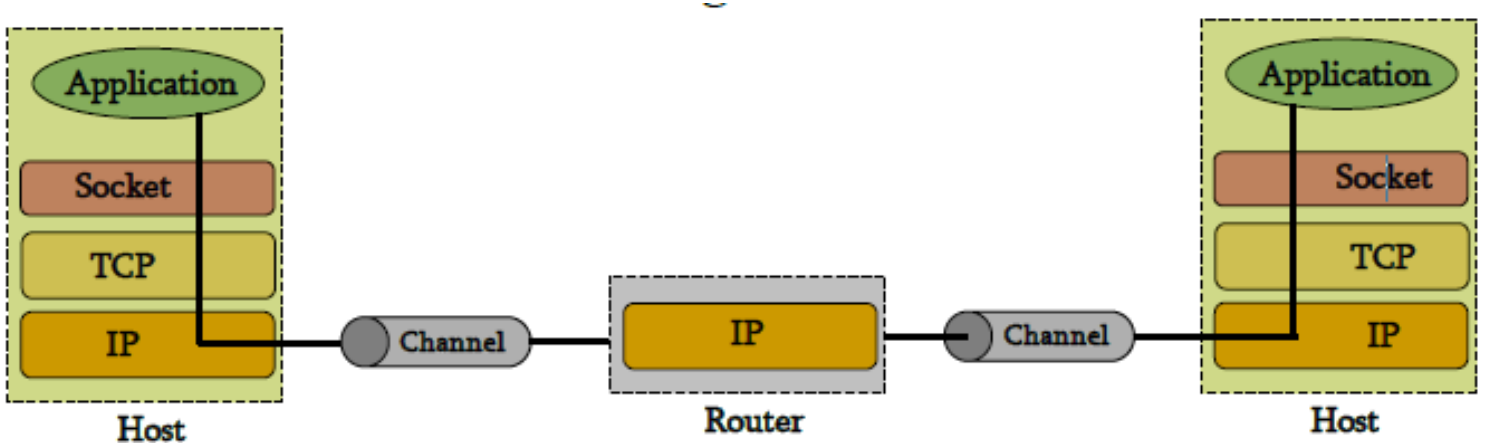
Dr. Öğr. Üyesi Musa BALTA
Bilgisayar Mühendisliği Bölümü
Bilgisayar ve Bilişim Bilimleri Fakültesi

Haftalık İçerik

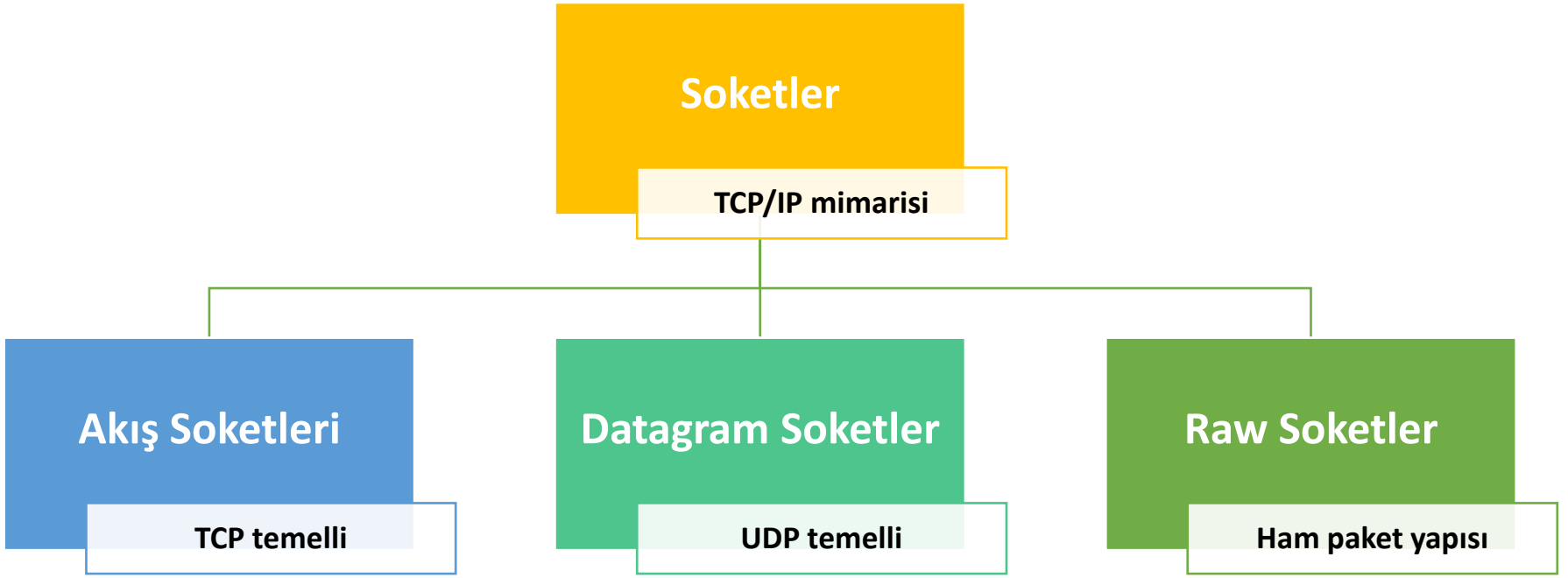
- Soket Kavramı ve Türleri
- İstemci-Sunucu Mimarisi
- Soket Prosedürleri
- İstemci-Sunucu İletişimi
- TCP ve UDP için Örnek Kod Yapıları
- Mesaj Oluşturma

Soket Kavramı

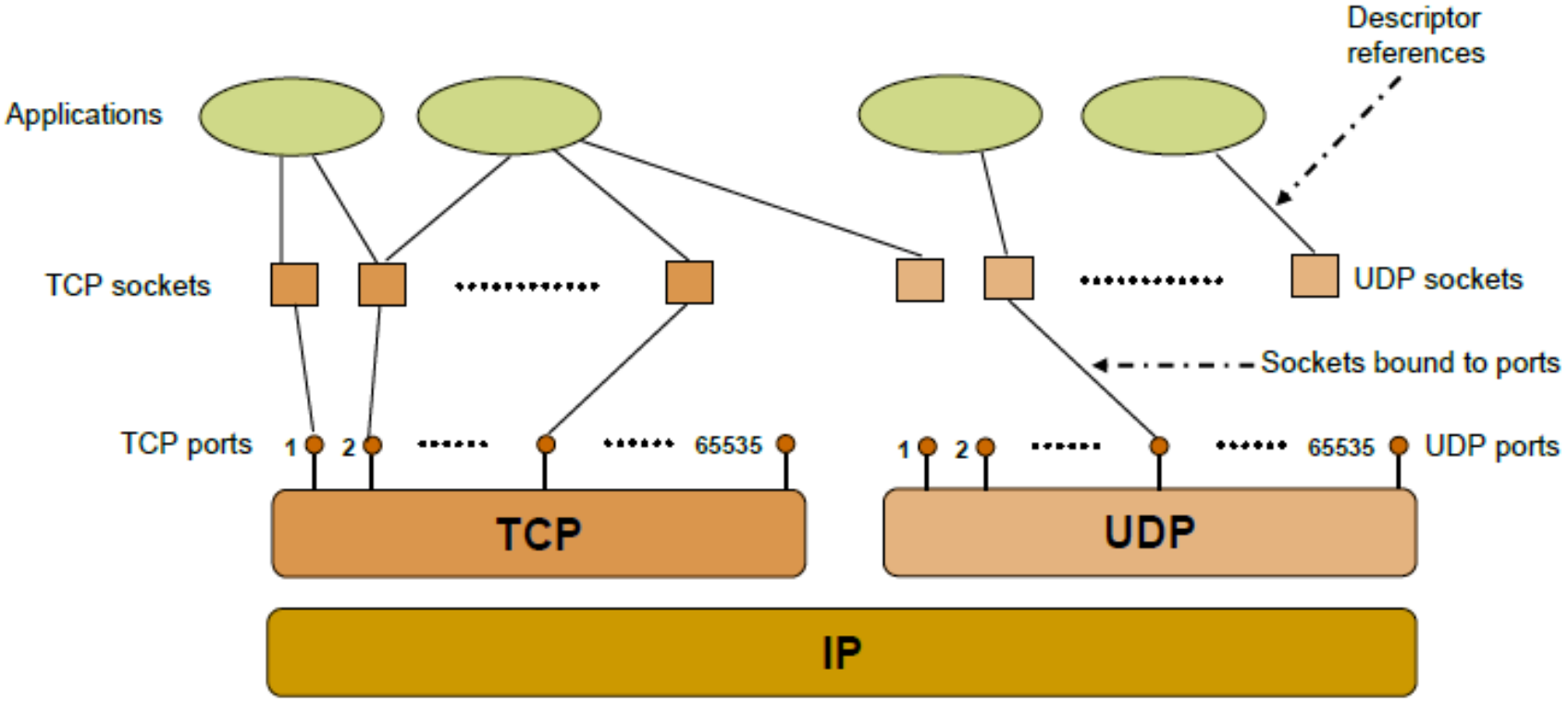
- Soketler, TCP ve UDP kullanan iki bilgisayar arasındaki **genel iletişim mekanizmasını** sağlar.
- Ağ programlama için Standart API'lerdir.
- **Soket kavramı;**
  - *IP adresi*
  - *Taşıma katman protokolü*
  - *Port Numarası*

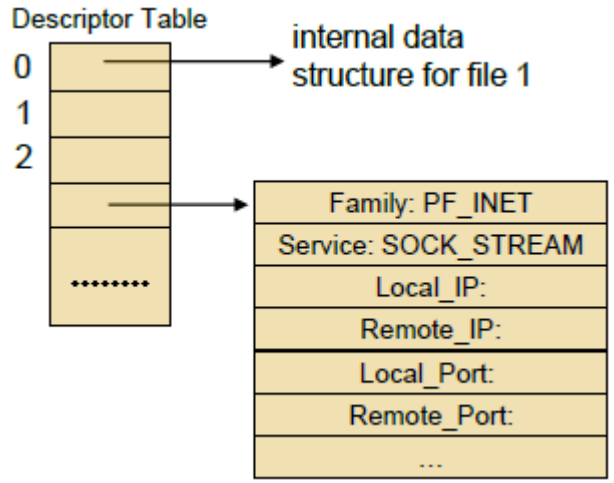


Soket Türleri



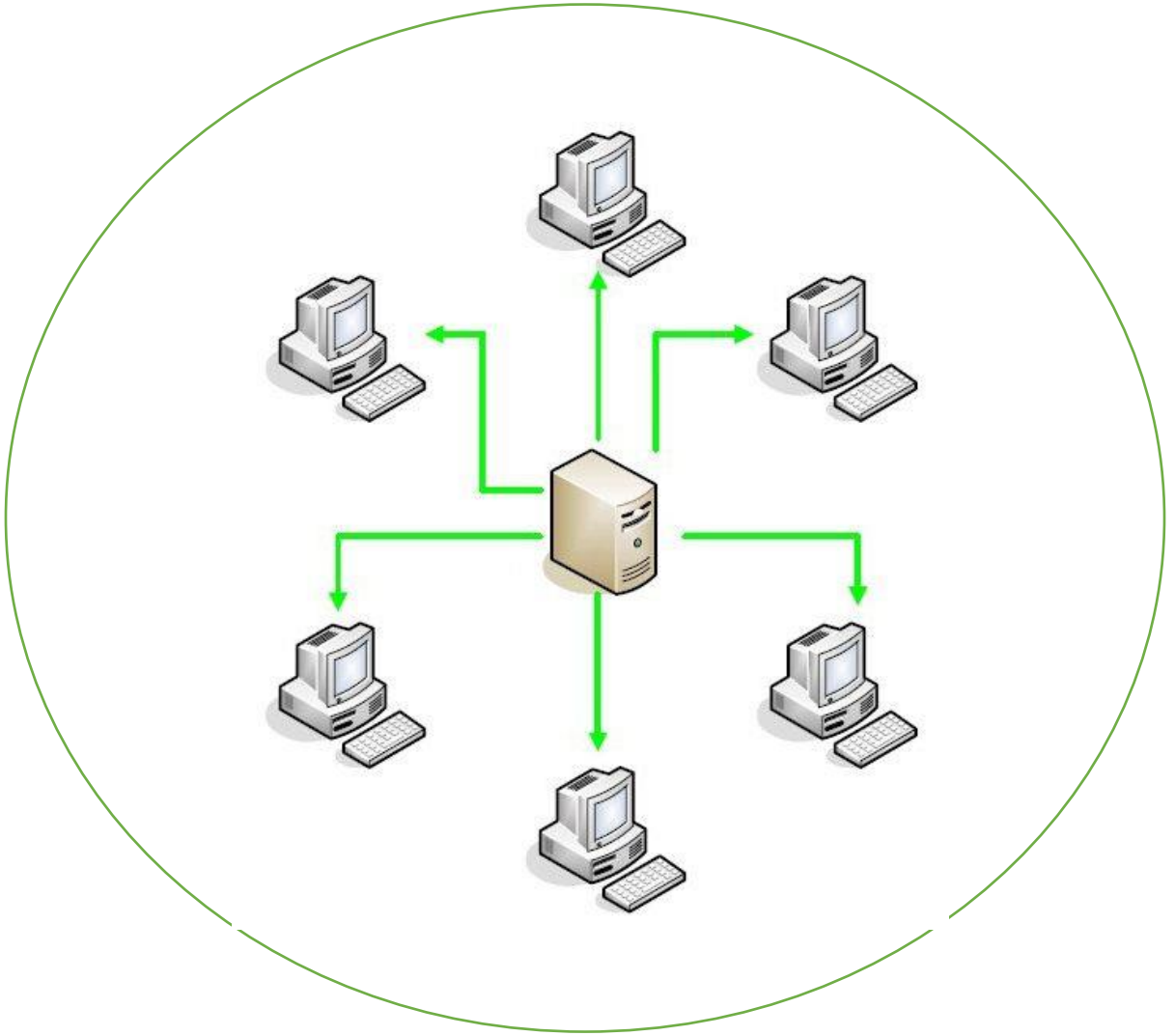
Soket Kavramı-Katmanlı Gösterim





İstemci-Sunucu Mimarisi

- ***Sunucu;***
  - Pasif bir şekilde bağlantı isteklerini bekler.
  - İstemci/istemcilere cevap verir.
  - Pasif soket yapısı
- ***İstemci;***
  - Bağlantıyı başlatır.
  - Sunucunun port ve IP adresini bilmesi gerekir.
  - Aktif soket yapısı



Soket Prosedürleri

Method	Tanımlama
1) socket()	Uç noktalarda bir bağlantı (soket) oluşturur.
2) bind()	Bir lokal IP ve portu bir sokete bağlar/ilişkilendirir.
3) listen()	Bağlantıları pasif olarak dinler.
4) connect()	Bir başka sokete bağlantıyı başlatır.
5) accept()	Yeni bir bağlantıyı kabul eder.
6) write()	Sokete veri yazar.
7) read()	Soketten veri okur.
8) sendto()	Bir başka UDP soketine datagram gönderir.
9) recvfrom()	Bir başka UDP soketinden datagram okur.
10) close()	Bağlantıyı sonlandırır. (bağlantıyı koparır)

Socket API Adres Tanımlamaları

- Socket API, adresler için **genel** bir veri türü tanımlar:

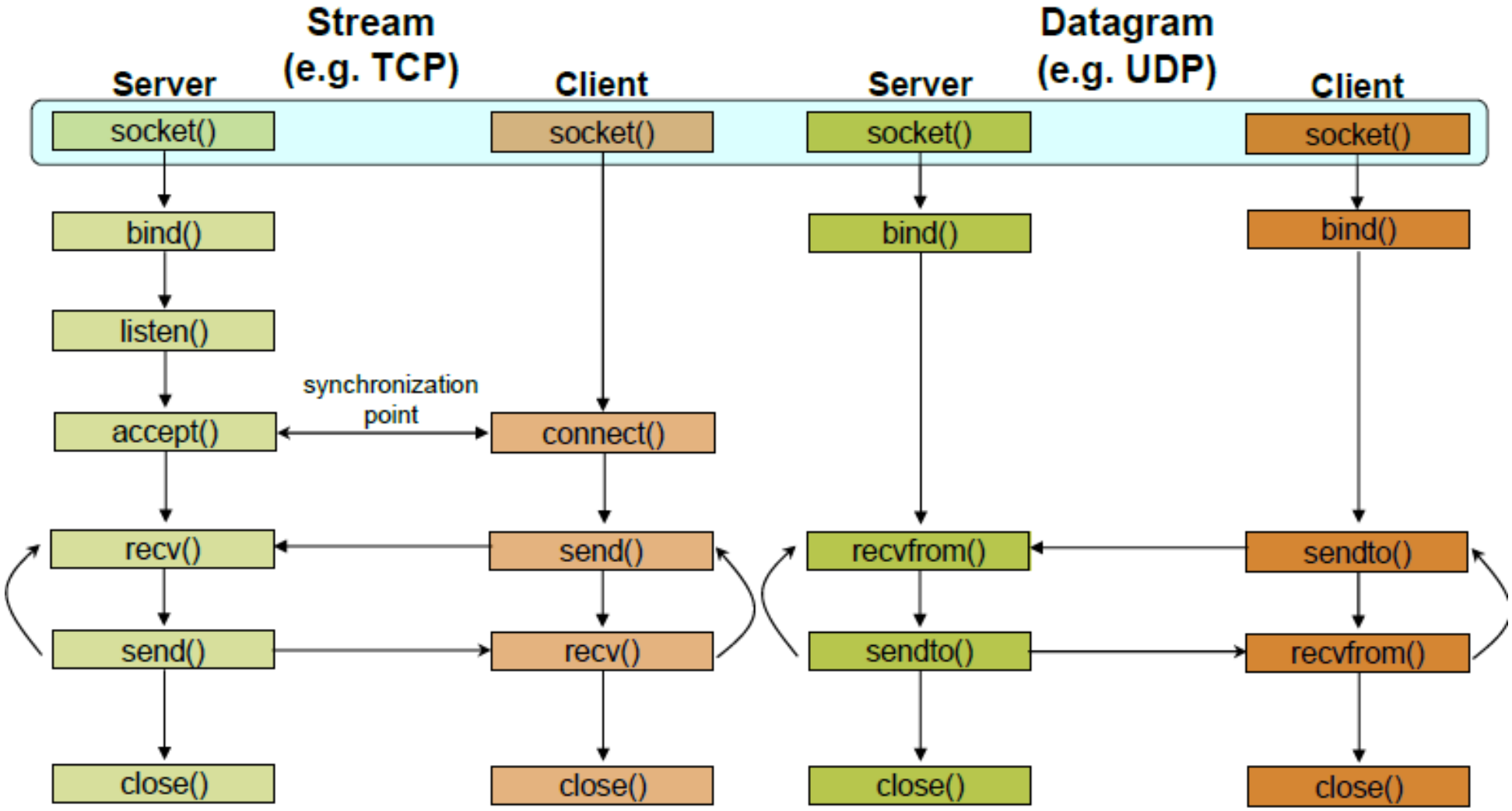
```
struct sockaddr {
   unsigned short sa_family;  /* Address family (e.g. AF_INET) */
   char sa_data[14];          /* Family-specific address information */
}
```

- **TCP/IP** adresleri için kullanılan sockaddr'ın özel biçimi:

```
struct in_addr {
   unsigned long s_addr;          /* Internet address (32 bits) */
}
struct sockaddr_in {
   unsigned short sin_family;     /* Internet protocol (AF_INET) */
   unsigned short sin_port;       /* Address port (16 bits) */
   struct in_addr sin_addr;     /* Internet address (32 bits) */
   char sin_zero[8];              /* Not used */
}
```

- **Not:** sockaddr_in bir sockaddr'a dönüştürülebilir.

İstemci-Sunucu İletişimi-Soket Oluşturma

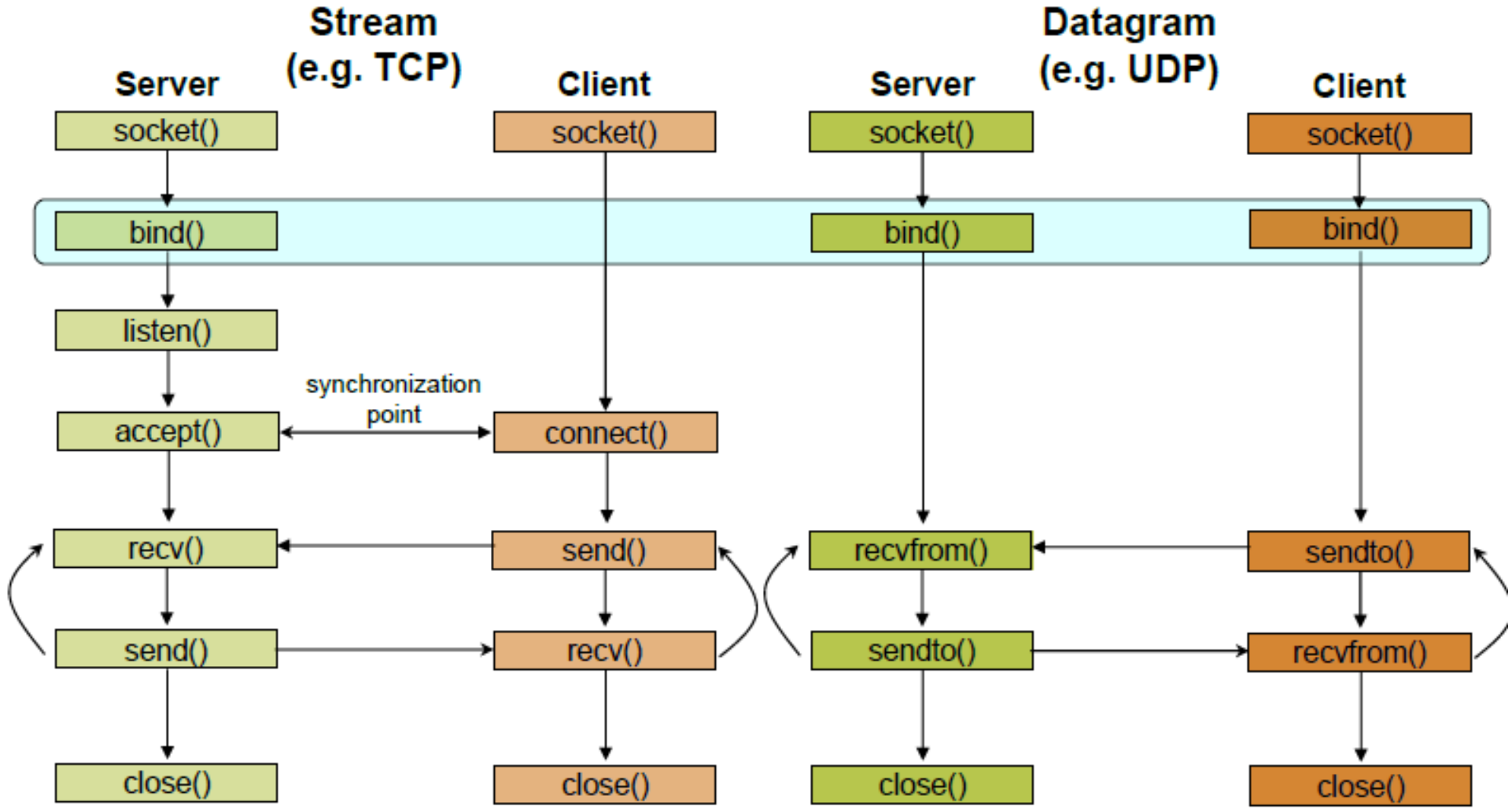


İstemci-Sunucu İletişimi-Soket Oluşturma

```
int sockid = socket(family, type, protocol);
```

- **sockid;** Soket tanımlayıcısı
- **family;** integer, iletişim domaini
  - PF_INET, IPv4 protocols, Internet addresses (typically used)
  - PF_UNIX, Local communication, File addresses
- **type;** iletişim tipi
  - SOCK_STREAM -reliable, 2-way, connection-based service
  - SOCK_DGRAM -unreliable, connectionless, messages of maximum length
- **protocol;** protokolü belirleme
  - IPPROTO_TCP IPPROTO_UDP
  - usually set to 0 (i.e., use default protocol)

İstemci-Sunucu İletişimi-Sokete İlişkilendirme



İstemci-Sunucu İletişimi-Sokete İlişkilendirme

```
int status = bind(sockid, &addrport, size);
```

- **sockid;** Soket tanımlayıcısı
- **addrport;** makinenin IP ve port adresleri, soket yapısı
    - TCP/IP sunucusu için, internet adresi genellikle INADDR_ANY olarak ayarlanır, yani herhangi bir gelen arayüzü seçer
- **size;** addrport yapısının boyutu (byte mk.)

TCP ile bind örneği

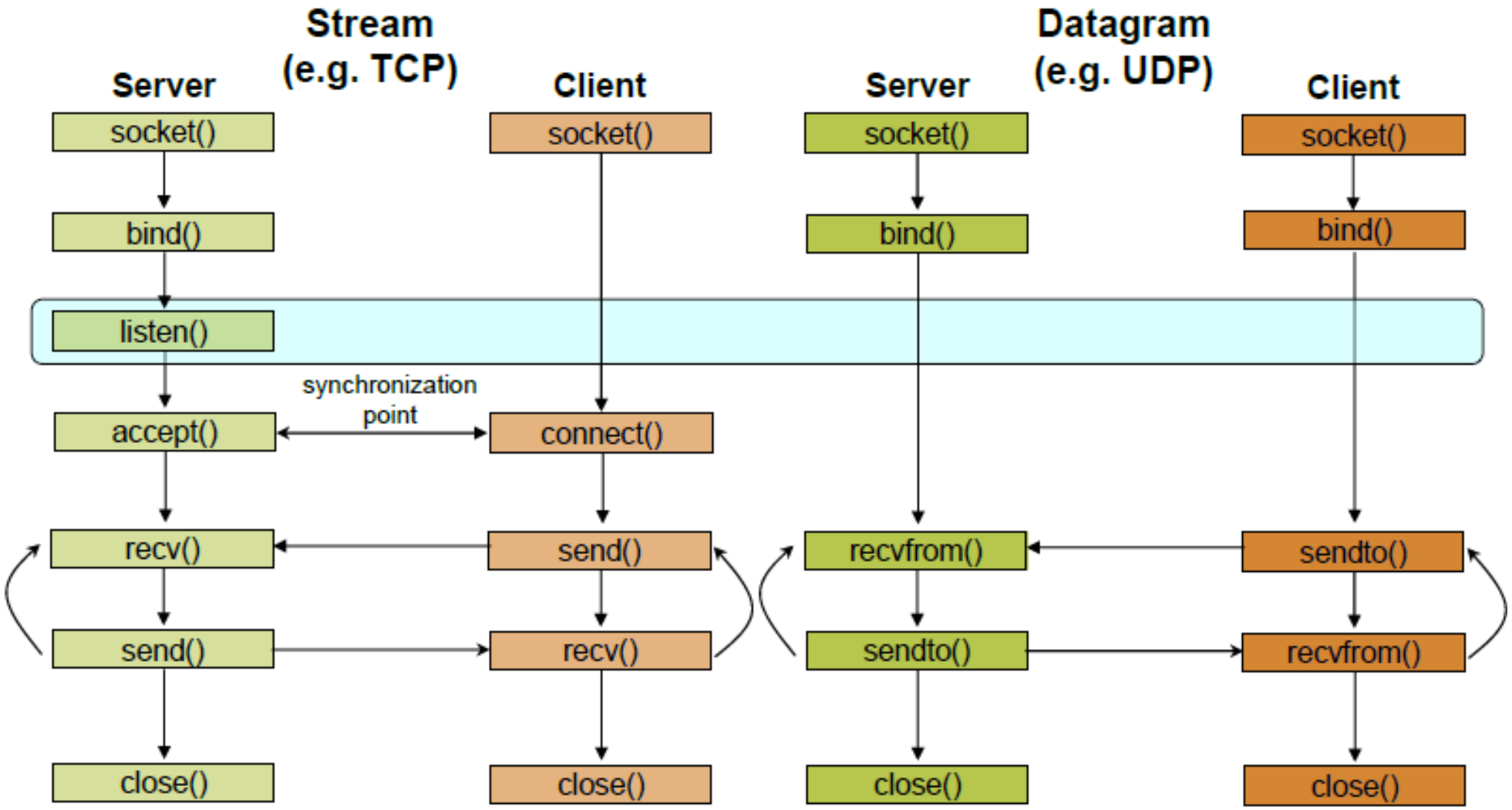
```
int sockid;
struct sockaddr_in addrport;
sockid = socket(PF_INET, SOCK_STREAM, 0);

addrport.sin_family = AF_INET;
addrport.sin_port = htons(5100);
addrport.sin_addr.s_addr = htonl(INADDR_ANY);
if(bind(sockid, (struct sockaddr *) &addrport, sizeof(addrport))!= -1) {
   …}
```

İstemci-Sunucu İletişimi-Sokete İlişkilendirme (devam)

- Her iki soket türü için **bind()** işlemi atlanabilir;
- **Datagram Socket:**
  - Eğer sadece gönderiyorsa, bağlamaya gerek yok. İşletim sistemi, soket her paket gönderdiğinde bir bağlantı noktası bulur.
  - Eğer alıyorsa, bağlamaya gerek var.
- **Stream Socket:**
  - Bağlantı kurulum aşamasında hedef belirlenir.
  - Gönderen portunun bilinmesine gerek yoktur (bağlantı kurulumu sırasında alıcı uç bağlantı noktasından haberdar edilir)

İstemci-Sunucu İletişimi-Dinleme

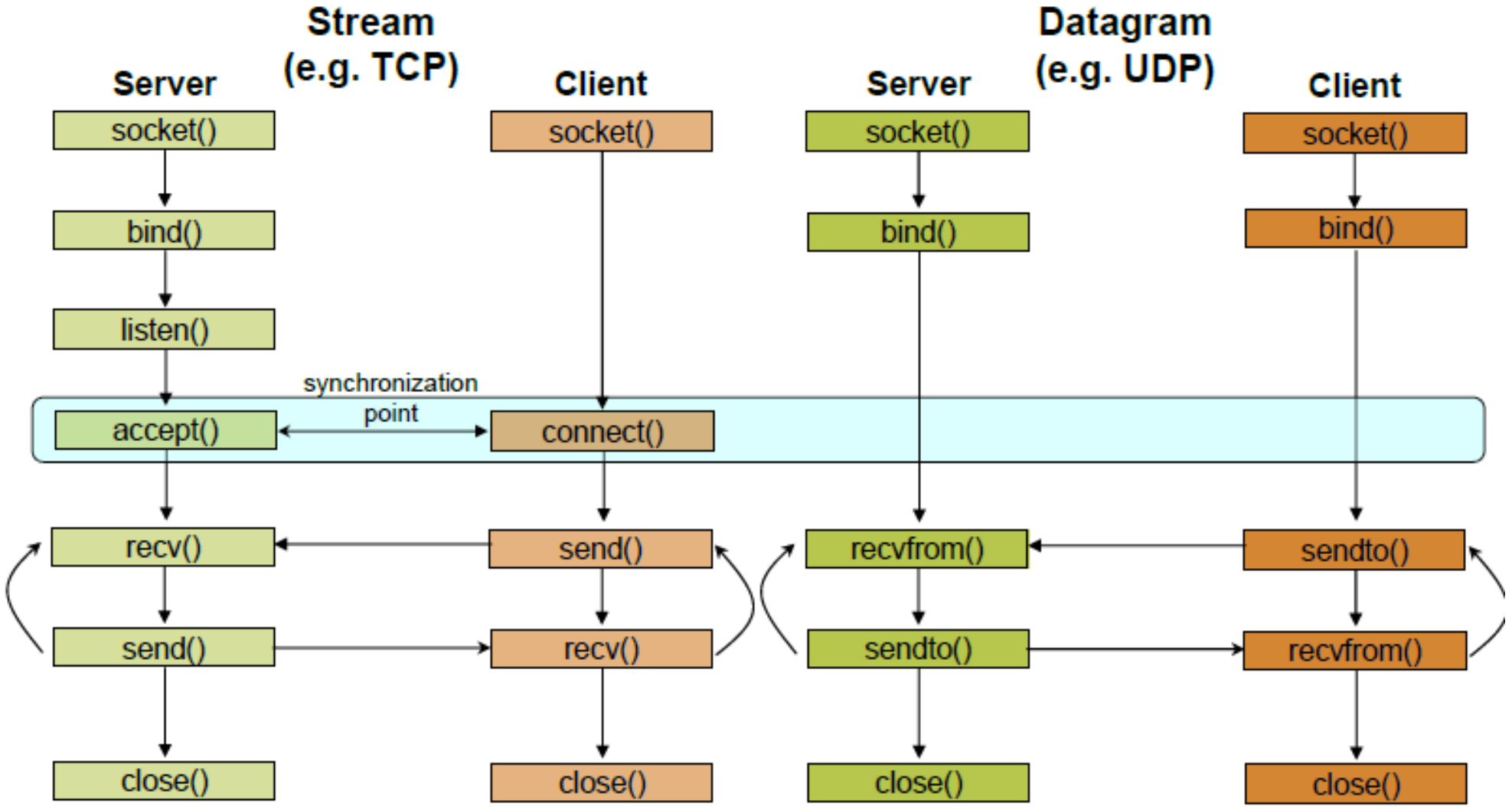


İstemci-Sunucu İletişimi-Dinleme (TCP protokolü için)

```
int status = listen(sockid, queueLimit);
```

- **sockid;** Soket tanımlayıcısı
- **queuelen;** integer, bir bağlantı için bekleyen aktif katılımcı/istemci sayısı
- **status;** eğer dinleme var ise 1, yoksa yani hata var ise 0
- Listening socket (sockid)
  - Asla gönderme ve alma işlemleri için kullanılmaz!
  - Sadece sunucu tarafından yeni soket bağlantıları almak için kullanılır.

İstemci-Sunucu İletişimi-Bağlantı Kabulü



İstemci-Sunucu İletişimi-Bağlantı Kurma (connect) - (accept)

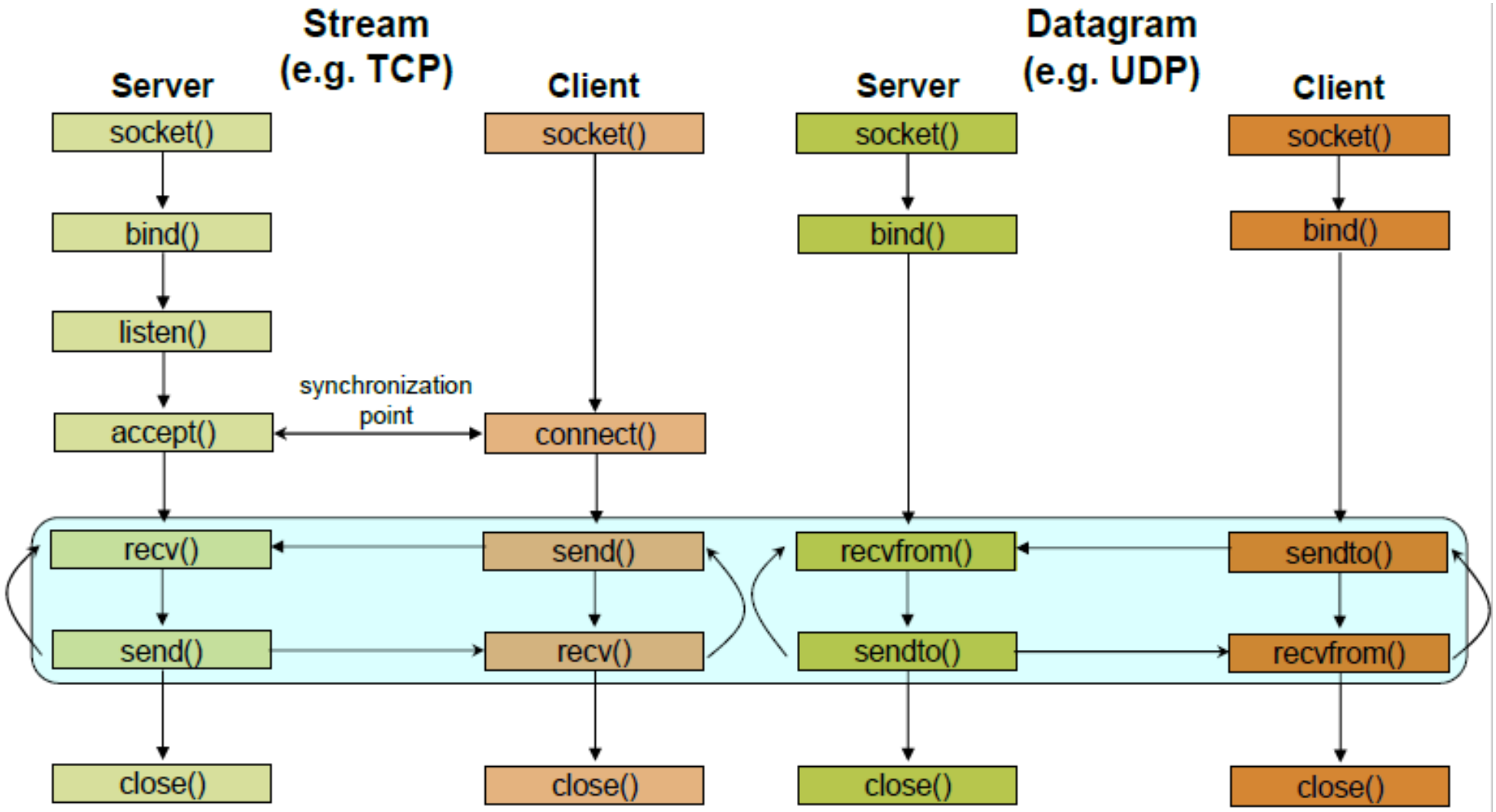
```
int status = connect(sockid, &foreignAddr, addrlen);
```

- **sockid;** Soket tanımlayıcısı
- **foreignAddr;** Pasif katılımcı/istemci adresleri, sockaddr yapısı
- **addrlen;** integer, adres boyutu
- **status;** Eğer bağlantı başarılı ise 0, başarısızsa -1

```
int s = accept(sockid, &clientAddr, &addrLen);
```

- **clientAddr;** Aktif katılımcı/istemci adresleri, sockaddr yapısı
- **addrlen;** istemcinin adres boyutu

İstemci-Sunucu İletişimi-Veri Okuma/Yazma



İstemci-Sunucu İletişimi-Veri Okuma/Yazma (Akış Soketleri)

```
int count = send(sockid, msg, msgLen, flags);
```

- **msg;** const void[], iletilecek mesaj
- **msgLen;** integer, iletilecek mesaj boyutu
- **flags;** integer, özel seçenekler, genellikle sadece 0
- **count;** iletilen byte sayısı, eğer hata varsa -1

```
int count = recv(sockid, recvBuf, bufLen, flags);
```

- **recvBuf;** void[], alınan byteları depolar.
- **bufLen;** alınan bytlerın boyutu
- **flags;** integer, özel seçenekler, genellikle sadece 0
- **count;** iletilen byte sayısı, eğer hata varsa -1

İstemci-Sunucu İletişimi-Veri Okuma/Yazma (Datagram Soketleri)

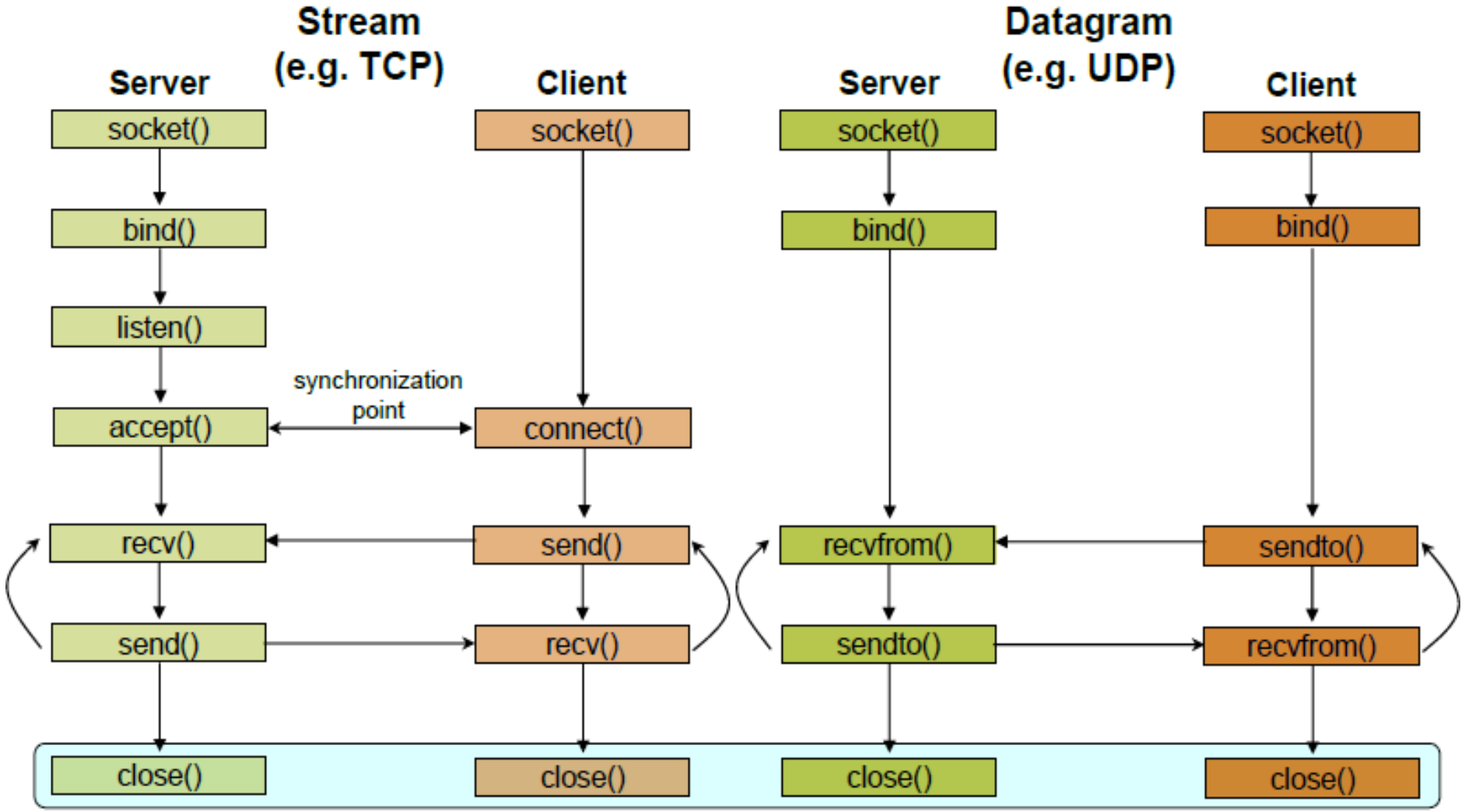
```
int count = sendto(sockid, msg, msgLen, flags,
&foreignAddr, addrlen);
```

- **msg**, **msgLen**, **flags**, **count**; aynı
- **foreignAddr**: sockaddr yapısı, hedef adres
- **addrLen**: foreignAddr'in boyutu

```
int count = recvfrom(sockid, recvBuf, bufLen,
flags, &clientAddr, addrlen);
```

- **recvBuf**, **bufLen**, **flags**, **count**; aynı
- **clientAddr**: sockaddr yapısı, istemcinin adresi
- **addrLen**: clientAddr'nin boyutu

İstemci-Sunucu İletişimi-Soketi Kapatma



İstemci-Sunucu İletişimi-Soketi Kapatma

```
status = close(sockid);
```

- **sockid:** Soket kapatma tanımlayıcısı.
- **status:** Eğer işlem başarılı ise 0, hata oluşursa -1 döndür.
- Soket kapatılırken;
  - Akış soketleri için bağlantı sonlandırılır.
  - Soketin kullandığı Port numarası serbest bırakılır.

İstemci-Sunucu İletişimi-Akış Temelli (Stream)Örnek

```
/* Create socket for incoming connections */
if ((servSock = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP)) < 0)
     DieWithError("socket() failed");
```

İstemci

1. Bir TCP Soketi oluşturma
2. Bağlantı kurma
3. İletişim
4. Bağlantıyı kapatma

Sunucu

1. Bir TCP Soketi oluşturma
2. Sokete Port atama
3. Soketi dinleme
4. Tekrar:
   a. Yeni bağlantı kabul etme
   b. İletişim
   c. Bağlantıyı kapatma

İstemci-Sunucu İletişimi-Akış Temelli (Stream)Örnek

```
echoServAddr.sin_family = AF_INET;                    /* Internet address family */
echoServAddr.sin_addr.s_addr = htonl(INADDR_ANY);     /* Any incoming interface */
echoServAddr.sin_port = htons(echoServPort);          /* Local port */

if (bind(servSock, (struct sockaddr *) &echoServAddr, sizeof(echoServAddr)) < 0)
    DieWithError("bind() failed");
```

İstemci

1. Bir TCP Soketi oluşturma
2. Bağlantı kurma
3. İletişim
4. Bağlantıyı kapatma

Sunucu

1. Bir TCP Soketi oluşturma
2. Sokete Port atama
3. Soketi dinleme
4. Tekrar:
   a. Yeni bağlantı kabul etme
   b. İletişim
   c. Bağlantıyı kapatma

İstemci-Sunucu İletişimi-Akış Temelli (Stream)Örnek

```
/* Mark the socket so it will listen for incoming connections */
if (listen(servSock, MAXPENDING) < 0)
        DieWithError("listen() failed");
```

İstemci

1. Bir TCP Soketi oluşturma
2. Bağlantı kurma
3. İletişim
4. Bağlantıyı kapatma

Sunucu

1. Bir TCP Soketi oluşturma
2. Sokete Port atama
3. Soketi dinleme
4. Tekrar:
   a. Yeni bağlantı kabul etme
   b. İletişim
   c. Bağlantıyı kapatma

İstemci-Sunucu İletişimi-Akış Temelli (Stream)Örnek

```
for (;;) /* Run forever */
{
  clntLen = sizeof(echoClntAddr);

  if ((clientSock=accept(servSock,(struct sockaddr *)&echoClntAddr,&clntLen))<0)
      DieWithError("accept() failed");
  ...
```

İstemci

1. Bir TCP Soketi oluşturma
2. Bağlantı kurma
3. İletişim
4. Bağlantıyı kapatma

Sunucu

1. Bir TCP Soketi oluşturma
2. Sokete Port atama
3. Soketi dinleme
4. Tekrar:
   a. Yeni bağlantı kabul etme
   b. İletişim
   c. Bağlantıyı kapatma

İstemci-Sunucu İletişimi-Akış Temelli (Stream)Örnek

```
/* Create a reliable, stream socket using TCP */
if ((clientSock = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP)) < 0)
    DieWithError("socket() failed");
```

İstemci

1. Bir TCP Soketi oluşturma
2. Bağlantı kurma
3. İletişim
4. Bağlantıyı kapatma

Sunucu

1. Bir TCP Soketi oluşturma
2. Sokete Port atama
3. Soketi dinleme
4. Tekrar:
   a. Yeni bağlantı kabul etme
   b. İletişim
   c. Bağlantıyı kapatma

İstemci-Sunucu İletişimi-Akış Temelli (Stream)Örnek

```
echoServAddr.sin_family = AF_INET;                    /* Internet address family */
echoServAddr.sin_addr.s_addr = inet_addr(echoservIP); /* Server IP address*/
echoServAddr.sin_port = htons(echoServPort);          /* Server port */

if (connect(clientSock, (struct sockaddr *) &echoServAddr,
                        sizeof(echoServAddr)) < 0)
    DieWithError("connect() failed");
```

İstemci

1. Bir TCP Soketi oluşturma
2. Bağlantı kurma
3. İletişim
4. Bağlantıyı kapatma

Sunucu

1. Bir TCP Soketi oluşturma
2. Sokete Port atama
3. Soketi dinleme
4. Tekrar:
   a. Yeni bağlantı kabul etme
   b. İletişim
   c. Bağlantıyı kapatma

İstemci-Sunucu İletişimi-Akış Temelli (Stream)Örnek

Sunucunun kabul prosedürü artık engellenmemiş olur ve istemcinin soketi adresi geri döndürülür.

```
for (;;) /* Run forever */
{
  clntLen = sizeof(echoClntAddr);

  if ((clientSock=accept(servSock,(struct sockaddr *)&echoClntAddr,&clntLen))<0)
     DieWithError("accept() failed");
  ...
```

İstemci

1. Bir TCP Soketi oluşturma
2. Bağlantı kurma
3. İletişim
4. Bağlantıyı kapatma

Sunucu

1. Bir TCP Soketi oluşturma
2. Sokete Port atama
3. Soketi dinleme
4. Tekrar:
   a. Yeni bağlantı kabul etme
   b. İletişim
   c. Bağlantıyı kapatma

İstemci-Sunucu İletişimi-Akış Temelli (Stream)Örnek

```
echoStringLen = strlen(echoString);     /* Determine input length */

/* Send the string to the server */
if (send(clientSock, echoString, echoStringLen, 0) != echoStringLen)
     DieWithError("send() sent a different number of bytes than expected");
```

İstemci

1. Bir TCP Soketi oluşturma
2. Bağlantı kurma
3. İletişim
4. Bağlantıyı kapatma

Sunucu

1. Bir TCP Soketi oluşturma
2. Sokete Port atama
3. Soketi dinleme
4. Tekrar:
   a. Yeni bağlantı kabul etme
   b. İletişim
   c. Bağlantıyı kapatma

İstemci-Sunucu İletişimi-Akış Temelli (Stream)Örnek

```
/* Receive message from client */
if ((recvMsgSize = recv(clntSocket, echoBuffer, RCVBUFSIZE, 0)) < 0)
    DieWithError("recv() failed");
/* Send received string and receive again until end of transmission */
while (recvMsgSize > 0) { /* zero indicates end of transmission */
   if (send(clientSocket, echobuffer, recvMsgSize, 0) != recvMsgSize)
      DieWithError("send() failed");
   if ((recvMsgSize = recv(clientSocket, echoBuffer, RECVBUFSIZE, 0)) < 0)
      DieWithError("recv() failed");
}
```

İstemci

1. Bir TCP Soketi oluşturma
2. Bağlantı kurma
3. İletişim
4. Bağlantıyı kapatma

Sunucu

1. Bir TCP Soketi oluşturma
2. Sokete Port atama
3. Soketi dinleme
4. Tekrar:
   a. Yeni bağlantı kabul etme
   b. İletişim
   c. Bağlantıyı kapatma

İstemci-Sunucu İletişimi-Akış Temelli (Stream)Örnek

```
close(clientSock);
```

```
close(clientSock);
```

İstemci

1. Bir TCP Soketi oluşturma
2. Bağlantı kurma
3. İletişim
4. Bağlantıyı kapatma

Sunucu

1. Bir TCP Soketi oluşturma
2. Sokete Port atama
3. Soketi dinleme
4. Tekrar:
   a. Yeni bağlantı kabul etme
   b. İletişim
   c. Bağlantıyı kapatma

İstemci-Sunucu İletişimi-Datagram Soket Örnek

```
/* Create socket for sending/receiving datagrams */
if ((servSock = socket(PF_INET, SOCK_DGRAM, IPPROTO_UDP)) < 0)
    DieWithError("socket() failed");
```

```
/* Create a datagram/UDP socket */
if ((clientSock = socket(PF_INET, SOCK_DGRAM, IPPROTO_UDP)) < 0)
    DieWithError("socket() failed");
```

İstemci

1. Bir UDP Soketi oluşturma
2. Sokete Port atama
3. İletişim
4. Bağlantıyı kapatma

Sunucu

1. Bir UDP Soketi oluşturma
2. Sokete Port atama
3. Tekrar:
    a. İletişim

İstemci-Sunucu İletişimi-Datagram Soket Örnek

```
echoServAddr.sin_family = AF_INET;                  /* Internet address family */
echoServAddr.sin_addr.s_addr = htonl(INADDR_ANY);   /* Any incoming interface */
echoServAddr.sin_port = htons(echoServPort);        /* Local port */

if (bind(servSock, (struct sockaddr *) &echoServAddr, sizeof(echoServAddr)) < 0)
    DieWithError("bind() failed");
```

```
echoClientAddr.sin_family = AF_INET;                  /* Internet address family */
echoClientAddr.sin_addr.s_addr = htonl(INADDR_ANY);   /* Any incoming interface */
echoClientAddr.sin_port = htons(echoClientPort);      /* Local port */

if(bind(clientSock,(struct sockaddr *)&echoClientAddr,sizeof(echoClientAddr))<0)
    DieWithError("connect() failed");
```

İstemci

1. Bir UDP Soketi oluşturma
2. Sokete Port atama
3. İletişim
4. Bağlantıyı kapatma

Sunucu

1. Bir UDP Soketi oluşturma
2. Sokete Port atama
3. Tekrar:
   a. İletişim

İstemci-Sunucu İletişimi-Datagram Soket Örnek

```
echoServAddr.sin_family = AF_INET;                    /* Internet address family */
echoServAddr.sin_addr.s_addr = inet_addr(echoservIP); /* Server IP address*/
echoServAddr.sin_port = htons(echoServPort);          /* Server port */

echoStringLen = strlen(echoString);     /* Determine input length */

/* Send the string to the server */
if (sendto( clientSock, echoString, echoStringLen, 0,
            (struct sockaddr *) &echoServAddr, sizeof(echoServAddr))
          != echoStringLen)
     DieWithError("send() sent a different number of bytes than expected");
```

İstemci

1. Bir UDP Soketi oluşturma
2. Sokete Port atama
3. İletişim
4. Bağlantıyı kapatma

Sunucu

1. Bir UDP Soketi oluşturma
2. Sokete Port atama
3. Tekrar:
   a. İletişim

İstemci-Sunucu İletişimi-Datagram Soket Örnek

```
for (;;) /* Run forever */
{
    clientAddrLen = sizeof(echoClientAddr)    /* Set the size of the in-out parameter */
    /*Block until receive message from client*/
     if ((recvMsgSize = recvfrom(servSock, echoBuffer, ECHOMAX, 0),
          (struct sockaddr *) &echoClientAddr, sizeof(echoClientAddr))) < 0)
          DieWithError("recvfrom() failed");

    if (sendto(servSock, echobuffer, recvMsgSize, 0,
               (struct sockaddr *) &echoClientAddr, sizeof(echoClientAddr))
            != recvMsgSize)
          DieWithError("send() failed");
}
```

İstemci

1. Bir UDP Soketi oluşturma
2. Sokete Port atama
3. İletişim
4. Bağlantıyı kapatma

Sunucu

1. Bir UDP Soketi oluşturma
2. Sokete Port atama
3. Tekrar:
   a. İletişim

İstemci-Sunucu İletişimi-Datagram Soket Örnek

```
close(clientSock);
```

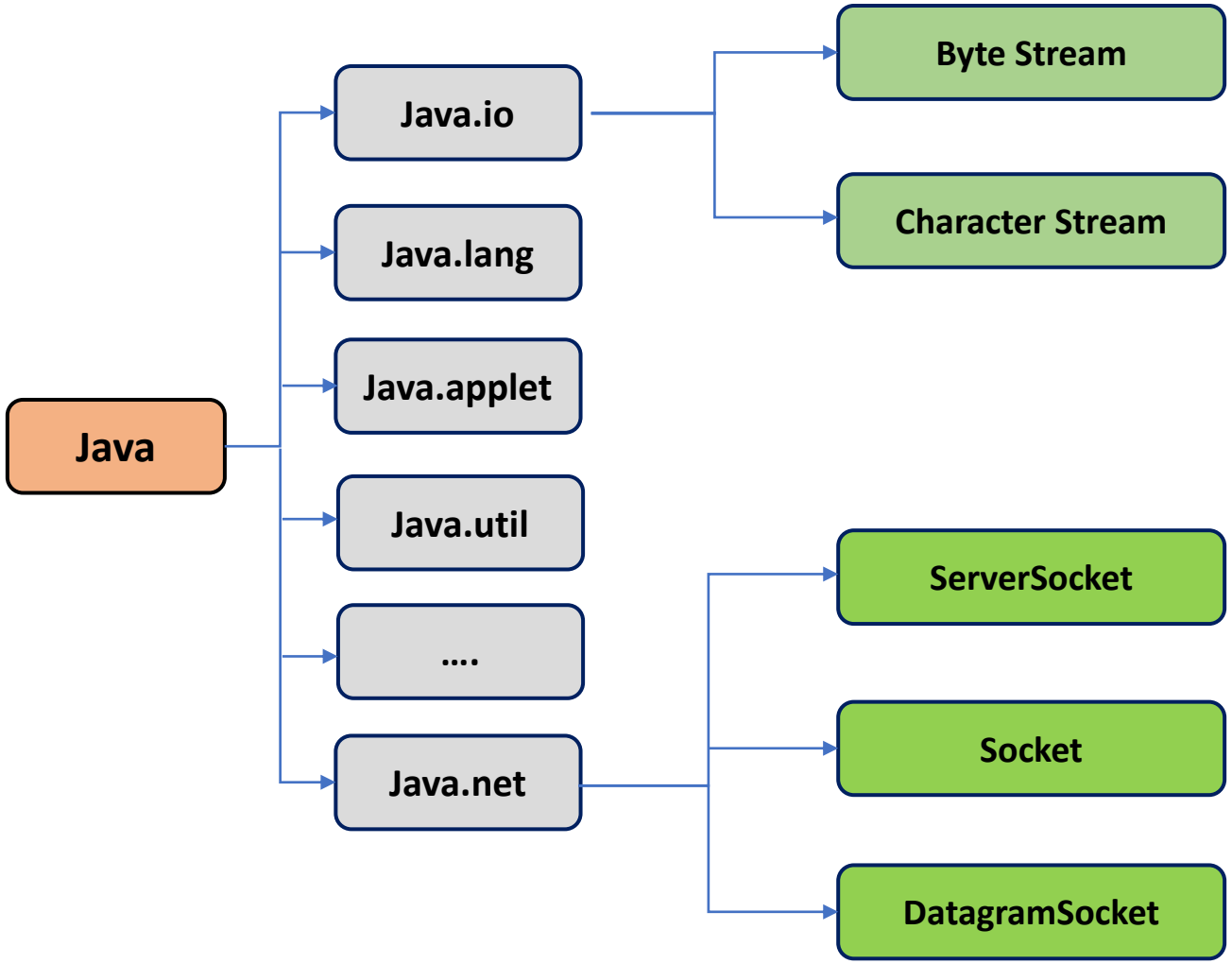
İstemci

1. Bir UDP Soketi oluşturma
2. Sokete Port atama
3. İletişim
4. Bağlantıyı kapatma

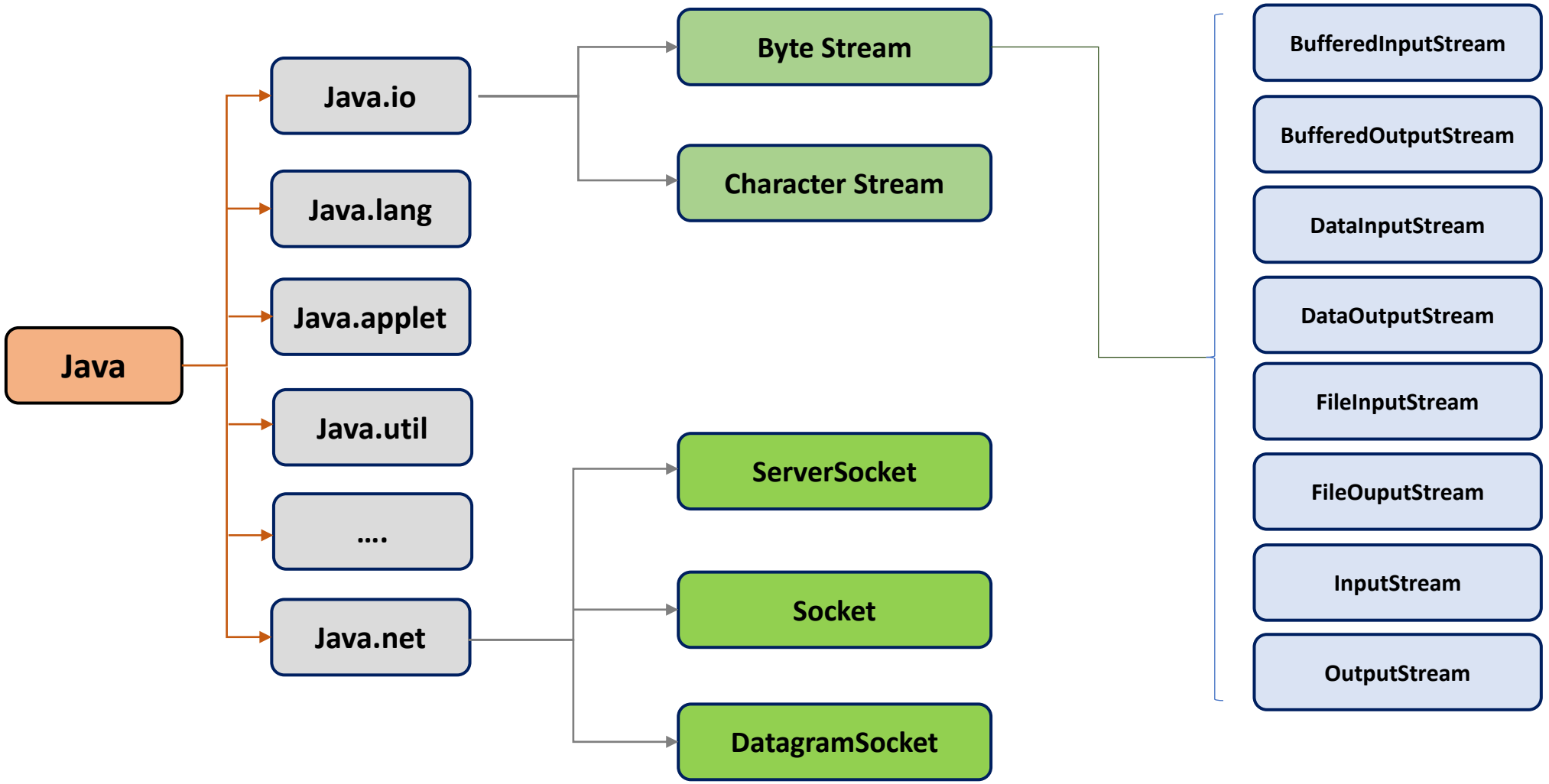
Sunucu

1. Bir UDP Soketi oluşturma
2. Sokete Port atama
3. Tekrar:
   a. İletişim

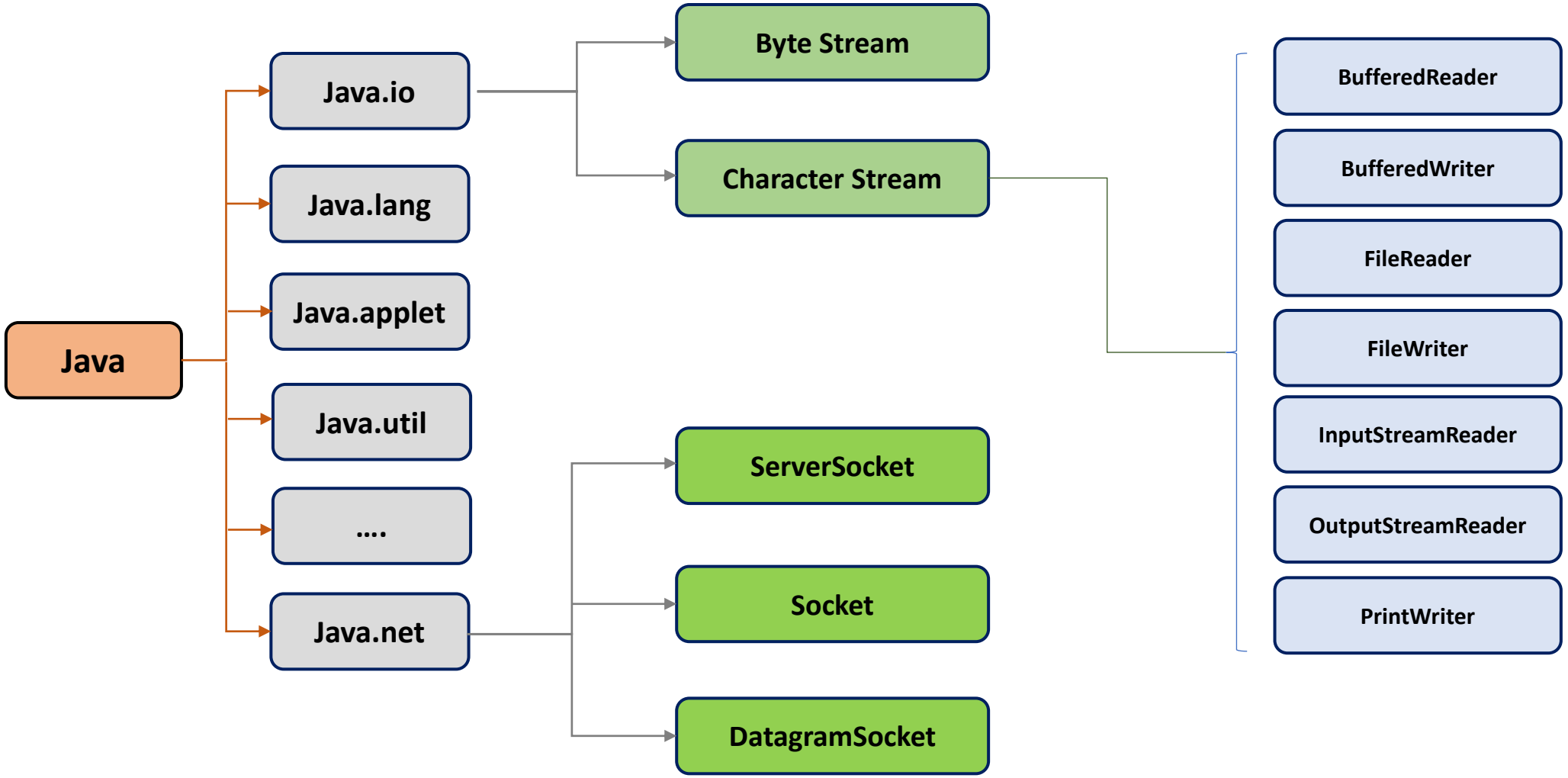
İstemci-Sunucu İletişimi-Java Programlama Dili



İstemci-Sunucu İletişimi-Java Programlama Dili



İstemci-Sunucu İletişimi-Java Programlama Dili



ServerSocket Sınıfı - Java

- Java.net.ServerSocket sınıfı, sunucu uygulamaları tarafından bir **bağlantı noktası** almak ve **istemci isteklerini** dinlemek için kullanılır. ServerSocket sınıfının dört kurucusu vardır;

Method	Tanımlama
1) public ServerSocket(int port) throws IOException	Belirtilen bağlantı noktasına bağlı bir sunucu soketi oluşturmaya çalışır. Bağlantı noktası başka bir uygulama tarafından zaten bağlıysa bir istisna oluşur.
2) public ServerSocket(int port, int backlog) throws IOException	Önceki yapıcıya benzer şekilde, backlog parametresi bir bekleme sırasında kaç gelen istemcinin saklanacağını belirtir.
3) public ServerSocket(int port, int backlog, InetAddress address) throws IOException	Önceki kurucuya benzer şekilde, InetAddress parametresi bağlanacak yerel IP adresini belirtir. InetAddress, birden fazla IP adresi olabilen sunucular için kullanılır ve sunucunun IP adreslerinden hangisinin istemci isteklerini kabul edeceğini belirlemesini sağlar.
4) public ServerSocket() throws IOException	Bağlı olmayan bir sunucu soketi oluşturur. Bu yapıcıyı kullanırken, sunucu soketini bağlamaya hazır olduğunuzda bind () yöntemini kullanın.

Server Socket Sınıfı-devam

Method	Tanımlama
1) public int getLocalPort()	Sunucu soketinin dinlediği bağlantı noktasını döndürür. Bu yöntem, bir yapıcıda bağlantı noktası numarası olarak 0'ı geçtiyseniz ve sunucunun sizin için bir bağlantı noktası bulmasına izin verirseniz yararlıdır.
2) public Socket accept() throws IOException	Gelen bir istemciyi bekler. Zaman aşımı değerinin setSoTimeout () yöntemi kullanılarak ayarlandığı varsayılarak, istemci belirtilen bağlantı noktasındaki sunucuya bağlanana veya soket zaman aşımına uğrayana kadar bu yöntem engellenir. Aksi takdirde, bu yöntem süresiz olarak engellenir.
3) public void setSoTimeout(int timeout)	Accept () sırasında sunucu soketinin bir istemci için ne kadar bekleyeceği zaman aşımı değerini ayarlar.
4) public void bind(SocketAddress host, int backlog)	Soketi SocketAddress nesnesindeki belirtilen sunucuya ve bağlantı noktasına bağlar.

Socket Sınıfı

- Java.net.Socket sınıfı, hem istemcinin hem de sunucunun birbirleriyle iletişim kurmak için kullandığı soketi temsil eder. İstemci bir Socket nesnesini örnek oluşturarak alırken, sunucu bir Socket nesnesini accept () yönteminin dönüş değerinden alır.

Method	Tanımlama
1) public Socket(String host, int port) throws UnknownHostException, IOException	Bu yöntem, belirtilen bağlantı noktasındaki belirtilen sunucuya bağlanmaya çalışır. Bu kurucu bir istisna atmazsa, bağlantı başarılı olur ve istemci sunucuya bağlanır.
2) public Socket(InetAddress host, int port) throws IOException	Bu yöntem, ana bilgisayarın bir InetAddress nesnesi tarafından gösterilmesi dışında, önceki yapıcı ile aynıdır.

Socket Sınıfı-devam

Method	Tanımlama
3) public Socket(String host, int port, InetAddress localAddress, int localPort) throws IOException	Belirtilen adrese ve bağlantı noktasına yerel ana bilgisayarda bir yuva oluşturarak belirtilen ana bilgisayara ve bağlantı noktasına bağlanır.
4) public Socket(InetAddress host, int port, InetAddress localAddress, int localPort) throws IOException	Bu yöntem, önceki yapıcı ile aynıdır, ancak ana bilgisayar bir String yerine InetAddress nesnesi tarafından gösterilir.
5) public Socket()	Bağlı olmayan bir soket oluşturur. Bu soketi bir sunucuya bağlamak için connect () yöntemini kullanın.

Socket Sınıfı-devam

- Socket sınıfına bazı ilgi yöntemleri burada listelenmiştir. Hem istemci hem de sunucuda bir Socket nesnesi bulunduğuna dikkat edin, bu nedenle bu yöntemler hem istemci hem de sunucu tarafından çağrılabilir.

Method	Tanımlama
1) public void connect(SocketAddress host, int timeout) throws IOException	Bu yöntem soketi belirtilen ana bilgisayara bağlar. Bu yöntem yalnızca Soketi bağımsız değişken yapıcısını kullanarak başlattığınızda gereklidir.
2) public InetAddress getInetAddress()	Bu yöntem, bu soketin bağlı olduğu diğer bilgisayarın adresini döndürür.
3) public int getPort()	Soketin uzak makinede bağlı olduğu bağlantı noktasını döndürür.

Socket Sınıfı-devam

Method	Tanımlama
4) public int getLocalPort()	Soketin yerel makinede bağlı olduğu bağlantı noktasını döndürür.
5) public SocketAddress getRemoteSocketAddress()	Bu yöntem, bu soketUzak soketin adresini döndürür. Soketin çıkış akışını döndürür. Çıkış akışı uzak soketin giriş akışına bağlıdır.
6) public InputStream getInputStream() throws IOException	Soketin giriş akışını döndürür. Giriş akışı uzak soketin çıkış akışına bağlıdır.
7) **public OutputStream getOutputStream() throws IOException**	Soketin çıkış akışını döndürür. Çıkış akışı uzak soketin giriş akışına bağlıdır.

Mesaj Oluşturma – Veri Kodlama

- İstemci sunucuya 2 integer değişken **(x ve y)** göndermek istediğinde;
- **1^{st} Çözüm**: **Karakter Kodlama** (örn. ASCII)
  - Ekran yazdırmak veya görüntülemek için aynı gösterim kullanılır.
  - Random olarak daha büyük sayıların gönderilmesine izin verir.

x = 17,998,720 y = 47,034,615

49	55	57	57	56	55	50	48	32	52	55	48	51	52	54	49	53	32
1	7	9	9	8	7	2	0	_	4	7	0	3	4	6	1	5	_

```
sprintf(msgBuffer, "%d %d ", x, y);
send(clientSocket, strlen(msgBuffer), 0);
```

Mesaj Oluşturma – Veri Kodlama (devam)

- **Handikaplar;**
- İkinci sınırlayıcı gerekli
  - Aksi takdirde sunucu bir diğer veri katarından ayrım yapamaz.
- msgBuffer yeteri kadar büyük olmalı
- strlen yalnızca mesajın bytelarını sayar.

- **Sonuc;**
- Bu çözüm tam anlamıyla yeterli olamamaktadır;
  - Her rakam bir bayt yerine 4 bit kullanılarak temsil edilebilir
  - Sayıları manipüle etmek sakıncalıdır.

Mesaj Oluşturma – Veri Kodlama

- **2^{st} Çözüm**: **Değerleri Gönderme**
  - Bir protokol kullanılır;
    - Her bir integer için kaç bit kullanılır.
    - Ne tip bir kodlama (**2'ye tümleyen, işaretli sayılar, işaretsiz sayılar** gibi) kullanılır.

1st Implementation

```
typedef struct {
  int x,y;
} msgStruct;
...
msgStruct.x = x;  msgStruct.y = y;
send(clientSock, &msgStruct, sizeof(msgStruct), 0);
```

2nd Implementation

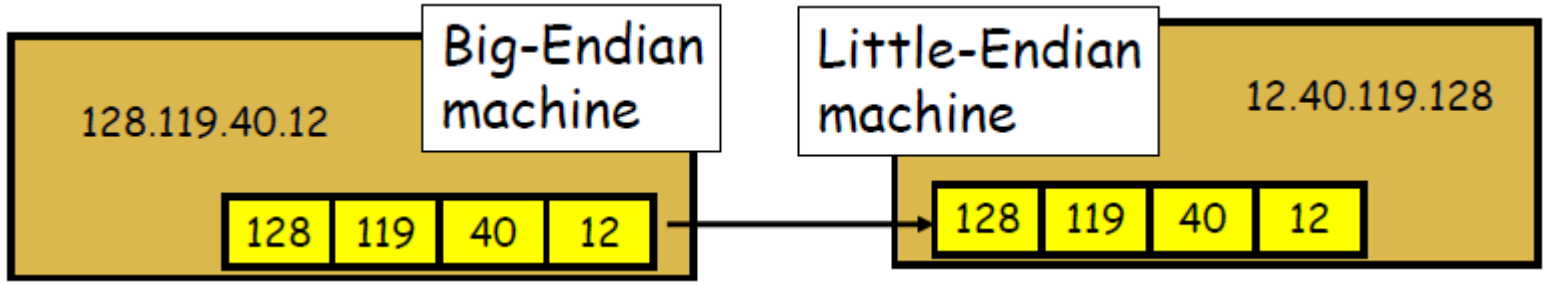
```
send(clientSock, &x, sizeof(x)), 0);
send(clientSock, &y, sizeof(y)), 0);
```

2nd implementation works in any case?

Mesaj Oluşturma – Byte Dizilimi

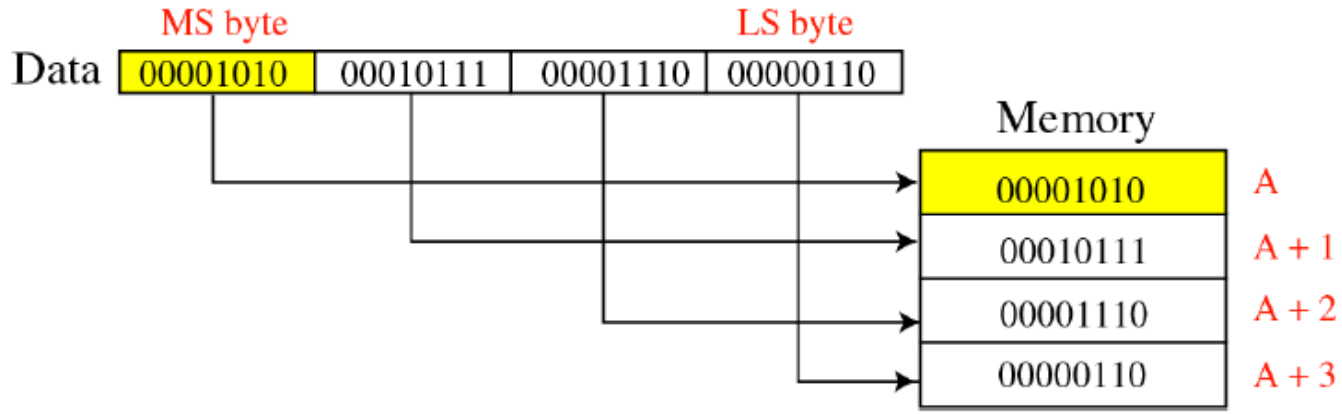
- Adres ve portlar integerlar gibi depolanırlar;
  - u_short sin_port; (16 bit)
  - in_addr sin_addr; (32 bit)

- Farklı makineler ve işletim sistemleri farklı sözcük/byte dizilimlerine sahiptirler.
  - **little-endian:** düşük bytelar önce
  - **big-endian**: yüksek bytelar önce

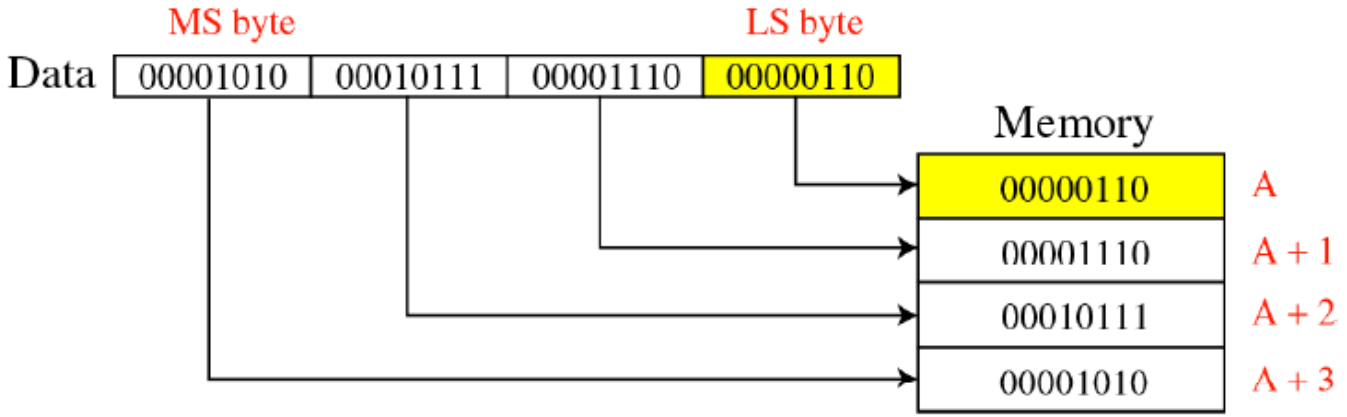


Mesaj Oluşturma – Byte Dizilimi (devam)

- Big-Endian:



- Little-Endian:



Mesaj Oluşturma – Byte Dizilimi (devam)

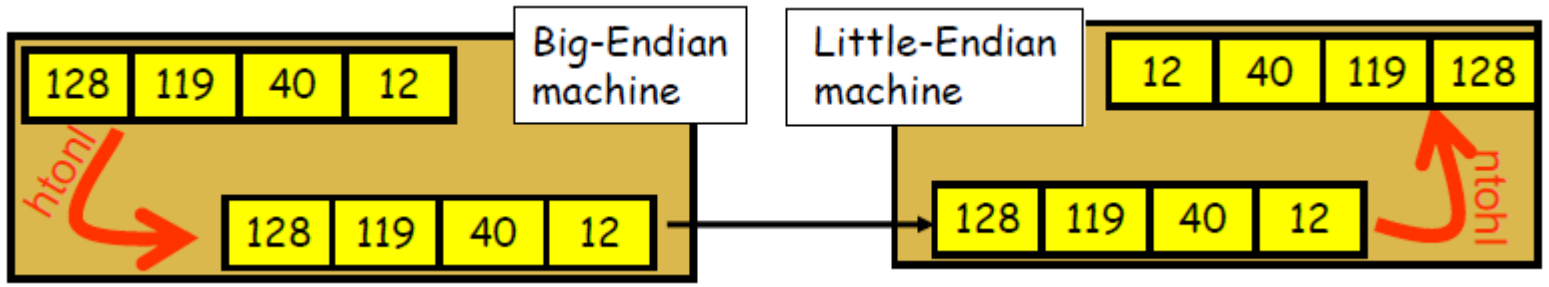
- **Host Byte-Ordering:** bir ana bilgisayar tarafından kullanılan bayt sıralaması (büyük veya küçük)

- **Network Byte-Ordering :** ağ tarafından kullanılan bayt sıralaması – her zaman big endian

```
u_long htonl(u_long x);
u_short htons(u_short x);
```

```
u_long ntohl(u_long x);
u_short ntohs(u_short x);
```

- Big-endian kullanılan makinelerde, bu rutinler bir şey yapmaz.
- Little-endian kullanılan makinelerde, byte dizilimi tersine çevrilir.



Mesaj Oluşturma – Byte Dizilimi (Örnek Kod)

- **İstemci**

```
unsigned short clientPort, message;   unsigned int messageLenth;

servPort = 1111;
message = htons(clientPort);
messageLength = sizeof(message);

if (sendto( clientSock, message, messageLength, 0,
            (struct sockaddr *) &echoServAddr, sizeof(echoServAddr))
          != messageLength)
     DieWithError("send() sent a different number of bytes than expected");
```

- **Sunucu**

```
unsigned short clientPort, rcvBuffer;
unsigned int recvMsgSize ;

if ( recvfrom(servSock, &rcvBuffer, sizeof(unsigned int), 0),
     (struct sockaddr *) &echoClientAddr, sizeof(echoClientAddr)) < 0)
     DieWithError("recvfrom() failed");

clientPort = ntohs(rcvBuffer);
printf ("Client's port: %d", clientPort);
```