TCP/IP

TCP/IP

- Kullanım olarak İki katmanlı bir haberleşme protokolüdür
- Üst katman TCP (Transfer Control Protocol) verinin iletimden önce paketlere ayrılmasını ve alıcıda bu paketlerin yeniden düzgün bir şekilde birleştirilmesini sağlar.
- Alt katman IP (Internet Protocol) ise, iletilen paketlerin istenilen ağ adresine yönlendirilmesini kontrol eder.

- İlk olarak 80'li yıllarda Amerikan Savunma Bakanlığı (DoD) tarafından OSI tabanlı sistemlere alternatif olarak geliştirilmiştir.
- DoD'un Amerikan piyasasındaki ana belirleyici olması, bu protokolün Amerikan yazılımlarında standart kabul edilmesine neden oldu.
- Internet'in atası sayılabilecek ARPANet bu nedenle TCP/IP ile doğdu. Internet kullanımının büyük bir hızla artması ile birlikte, TCP/IP OSI üzerinde bir üstünlük kurmuş oldu.

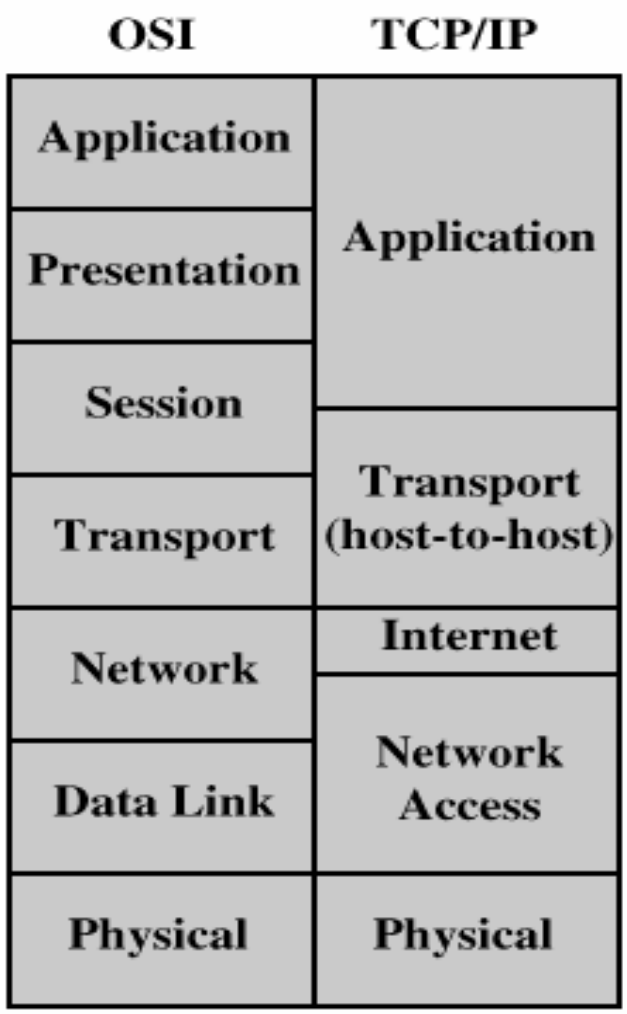
TCP/IP Protokol Yapısı

- Uygulama Katmanı (Application Layer): farklı sunucular üzerindeki süreç ve uygulamalar arasında iletişimi sağlar.
- Taşıma katmanı (Host to host or Transport Layer): Noktadan noktaya veri akışını sağlar.

- Internet Katmanı: Router lar ile birbirine bağlanmış ağlar boyunca verinin kaynaktan hedefe yönlendirilmesini sağlar.
- Ağ Erişim Katmanı: Uç sistem ile alt ağ arasındaki lojik arabirime ilişkin katmandır.
- Fiziksel Katman: İletişim ortamının karakteristik özelliklerini, sinyalleşme hızını ve kodlama şemasını belirler.

TCP/IP ile OSI arasındaki farklar

- TCP/IP haberleşme görevini karmaşık bir iş olarak niteleyerek daha basit alt görevlere böler. Her bir alt görev diğer alt görevler için belirli servisler sunar ve diğer alt görevlerin servislerini kullanır. OSI modeli de aynı kavramı kullanır, ancak OSI modelinde her bir katmandaki protokollerin özellikleri ve birbirleri ile ilişkileri kesin bir dille tanımlanmıştır. Bu özellik OSI modeli ile çalışmayı daha verimli kılar.
- OSI modelinde katmanların görevlerinin kesin bir şekilde belirlenmiş olması yeni bir protokol geliştirmeyi kimi zaman güçleştirebilir. TCP/IP ise böyle bir kısıtlama getirmediğinden, gerektiğinde yeni bir protokol mevcut katmanlar arasına rahatlıkla yerleştirilebilir.
- OSI modelinde gerekmeyen bir katmanın kullanılmaması gibi esnek bir yapıya izin verilmemektedir. TCP/IP ise katı kurallarla tanımlı olmadığından gereksinim duyulmayan katmanların kullanılmamasına izin verir. Örneğin uygulama katmanında olmasına rağmen doğrudan IP üzerinden kullanılabilen protokoller mevcuttur.

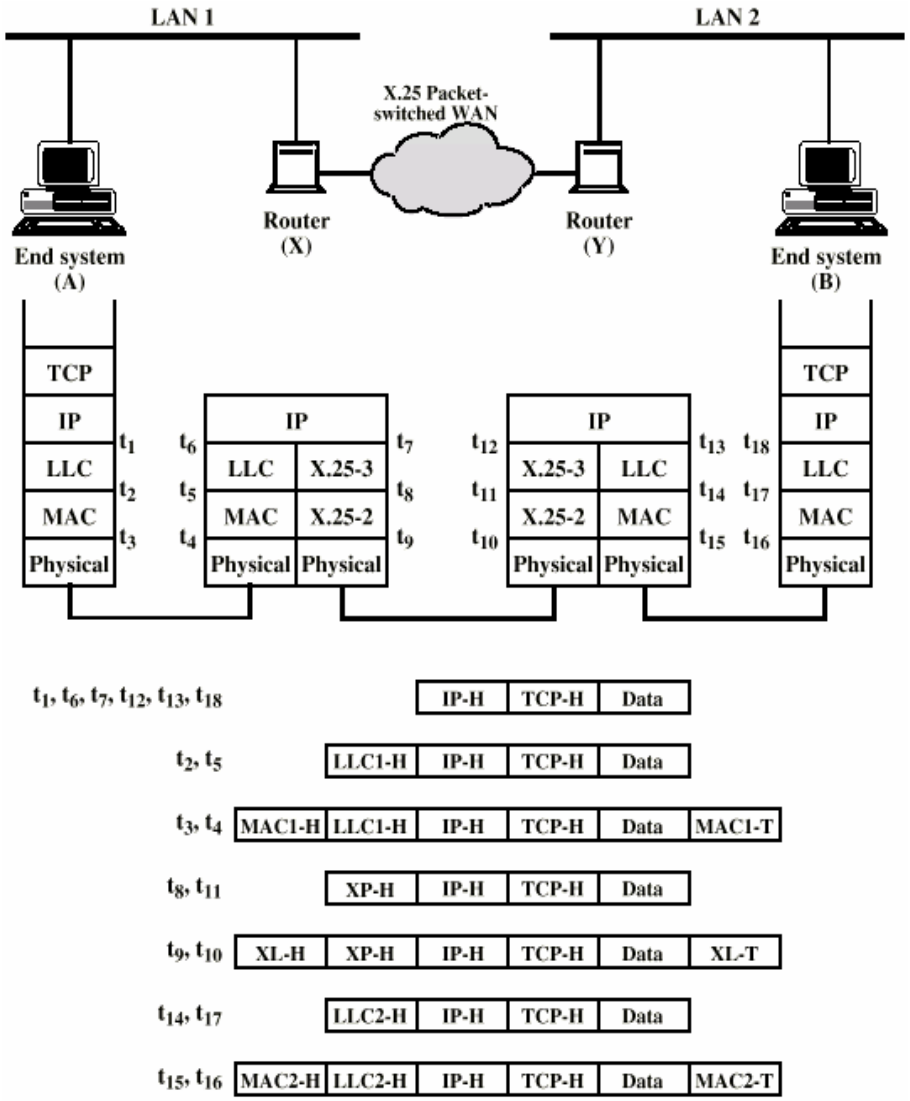

OSI
TCP/IP
Application
Presentation
Session
Transport
Network
Data Link
Physical
Application
Transport (host-to-host)
Internet
Network Access
Physical

IP – Internet Protokolü
Bağlantısız Haberleşme

- IP, uç sistemler arasında bağlantısız bir haberleşme servisi (datagram) sunar

Avantajları

- Bağlantısız bir internet uygulaması esneklik sağlar.
- Bağlantısız bir internet servisinin dayanıklılığı yüksektir.
- Bağlantısız bir internet servisi bağlantısız iletim protokolleri için en iyi seçimdir.



Tipik Ağlar Arası TCP/IP haberleşmesi

IP Hizmetleri

- **Kaynak adresi**: IP paketinin kaynağının internetwork adresi
- **Hedef adresi**: IP paketinin hedefinin internetwork adresi
- **Protokol**: Alıcının protokolü
- **Servis tipi göstergesi**: Ağ üzerinde dolaşan veri biriminin iletim boyunca ne şekilde değerlendirileceğini belirler
- **Tanımlayıcı**: Kaynak ve hedef adresleri ile kullanıcı protokolünden yararlanarak veri birimini diğer veri birimlerinden ayırt edebilmek için kullanılır. Yeniden düzenleme ve hata raporlama için gereklidir.
- **Parçalama-tanımlayıcısı**: IP'nin veriyi parçalayıp parçalamayacağını belirler.
- **Yaşam Süresi (TTL):** Ağ düğümleri cinsinden, verinin ne kadar süre boyunca ağ üzerinde iletileceğini belirler.
- **Veri uzunluğu**: Aktarılan verinin uzunluğu
- **Seçenek verisi**: IP kullanıcısı tarafından istenen seçenekler.
- **Veri**: Aktarılacak olan kullanıcı verisi

Gönder(
    Kaynak Adresi
    Protokol
    Servis Tipi
    Tanımlayıcı
    Parçalama
    TTL
    Veri Boyu
    Seçenek
    Veri
)

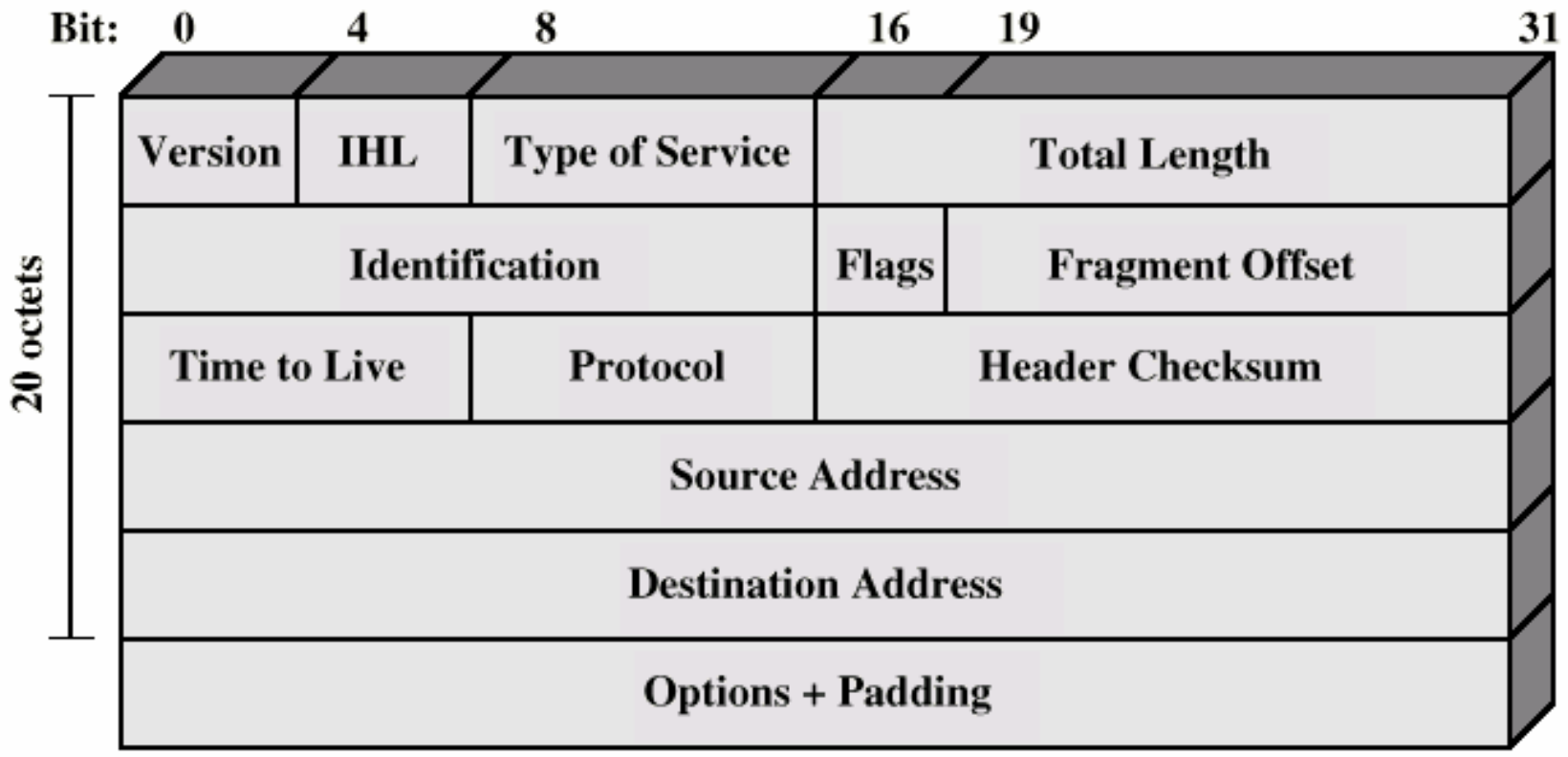
Al(
    Kaynak Adresi
    Protokol
    Servis Tipi

    Veri Boyu
    Seçenek
    Veri
)

IP hizmet temelleri ve parametreleri

- Servis tipi parametresi servis kalitesi (QoS) sağlamak için kullanılabileceği gibi, yönlendirme kararları verilirken de kullanılabilir. Örneğin bir yönlendirici iletilen veri için birden fazla yol çizebiliyorsa, bir sonraki noktada veri akışının en hızlı olduğu alternatif yolu seçebilir. Bu parametre günümüzde DiffServices olarak değiştirilmiştir.
- Seçenekler parametresi, ileride ihtiyaç duyulabilecek genişlemelere olanak sağlamaktadır. Halen kullanılmakta olanlar ise şunlardır:
  - Güvenlik etiketi,
  - kaynak yönlendirme,
  - yönlendirme kaydı,
  - akış tanımlama,
  - zaman damgası.

IP Protokolü Paket Yapısı IPV4 RFC791, 1981



IP Protokolü Paket Yapısı

- Versiyon (4bit)
- Internet başlık uzunluğu(IHL) (4bit): 32 bit cinsinden başlığın uzunluğu, min. Değeri 5(20 oktet)
- Servis tipi(8 bit): Güvenilirlik (reliability), öncelik, gecikme ve throughput parametreleri
  - bits 0–2: Precedence (111 - Network Control, 110 - Internetwork Control, 101 - CRITIC/ECP, 100 - Flash Override, 011 - Flash, 010 - Immediate, 001 - Priority, 000 - Routine)
  - bit 3: 0 = Normal Delay, 1 = Low Delay
  - bit 4: 0 = Normal Throughput, 1 = High Throughput
  - bit 5: 0 = Normal Reliability, 1 = High Reliability
  - bit 6: 0 = Normal Cost, 1 = Minimize Monetary Cost
  - bit 7: TANIMSIZ

  Bu alan 1998 yılından sonra Differentiated services alanı olarak anlam taşımaktadır ve içeriği değişmiştir.

- Toplam uzunluk(16bit):Oktet cinsinden toplam datagram boyu
  - Minimum 20 byte dır, 65535 olabilir

- Tanımlayıcı(16 bit): Datagramı diğer datagramlardan ayıran kaynak adresi, hedef adresi ve kullanıcı protokolü ile belirlenen parametre. Datagram internet üzerinde olduğu sürece eşsiz olmalı

- Bayraklar(3 bit): İkisi kullanılmaktadır.
  - More biti parçalama ve birleştirme için kullanılır.
  - Parçalamayı engelle biti ise hedefin birleştirme özelliği yoksa kullanılır. Ancak bu bit set edildiğinde datagramın boyutu yol boyunca bir noktada alt ağın kapasitesini aşıyorsa, datagramın kaybetilmesine neden olur.

- Parça ofseti(13bit): Mevcut parçanın özgün datagramın hangi parçası olduğunu gösterir. 64 bitlik üniteler cinsinden hesaplanır.

IP Protokolü Paket Yapısı

- **TTL (8bit):** yönlendirici adımı cinsinden hesaplanır.
- **Protokol(8bit):** Bir üst düzeydeki protokolü gösterir. Bunlar 140 civarındadır. En popülerleri;
  - 1: Internet Control Message Protocol (ICMP)
  - 2: Internet Group Management Protocol (IGMP)
  - 6: Transmission Control Protocol (TCP)
  - 17: User Datagram Protocol (UDP)
  - 89: Open Shortest Path First (OSPF)
  - 132: Stream Control Transmission Protocol (SCTP)
- **Başlık checksum(16bit):** Başlık için hata kontrol parametresi. Her yönlendiricide yeniden hesaplanır. Başlıktaki 16 bitlik değerlerin toplamının bire tümleyenidir.
- **Kaynak adresi(32 bit):** Kaynağın IP adresi
- **Hedef adresi(32 bit)**
- **Seçenekler(değişken):** Gönderen kullanıcı tarafından belirlenen ek seçenekler
- **Opsiyonlar ve Ekleme(değişken):** Veri alanı 8 bitin tam katları olmak zorundadır. Gerek duyulduğunda eksik bitleri tamamlamak için kullanılır.
- **Veri(değişken):** 8 bitin tam katları olmak zorunda. En fazla 65535-Baslık oktet olabilir.

IP Adresleri , IPV4

- IP adresleri 32 bittir. IP adresleri kaynak yetersizliği, özel şebekeler yaratma gibi amaçlarla sınıflandırılmıştır. 5 farklı türü vardır.
  - A sınıfı
  - B sınıfı
  - C sınıfı
  - D sınıfı
  - E sınıfı

AĞLARIN ADRESLERE GÖRE SINIFLANDIRILMASI-1

Sınıf	Öndeki Bitler	Ağ Boyutu *Bit alanı boyu*	Geri kalan bit sayısı
Class A	0	7	24
Class B	10	14	16
Class C	110	21	8
Class D (multicast)	1110 (224-239)		
Class E (reserved)	1111		

AĞLARIN ADRESLERE GÖRE SINIFLANDIRILMASI-2

Sınıf	Öndeki Bitler	Ağ sayısı	Ağ başına adres sayısı
Class A	0	126	16,277,214
Class B	10	16,384	65,534
Class C	110	2,097,152	254

Sınıf	Öndeki Bitler	Başlangıç	Bitiş	CIDR	Default alt ağ maskesi
Class A	0	0.0.0.0	127.255.255.255	/8	255.0.0.0
Class B	10	128.0.0.0	191.255.255.255	/16	255.255.0.0
Class C	110	192.0.0.0	223.255.255.255	/24	255.255.255.0
Class D (multicast)	1110	224.0.0.0	239.255.255.255	/32	
Class E (reserved)	1111	240.0.0.0	255.255.255.255	/32	

IP Adresleri - Class A

- 32 bit küresel internet adresi
- Network ve host parçaları
- Class A
  - binary 0 ile başlar
  - tüm 0 rezerve
  - 01111111 (127) loopback için rezerve
  - 1.x.x.x den 126.x.x.x ya
  - Tümü kullanımda

IP Adresleri - Class B

- Başlangıç 10
- 128.x.x.x - 191.x.x.x
- İkinci oktet network adresinin parçası
- 2^{14} = 16,384 class B adres
- Tümü kullanımda

IP Adresleri - Class C

- Başlangıç 110
- 192.x.x.x - 223.x.x.x
- İkinci ve üçüncü oktet network adresinin parçası
- 2^{21} = 2,097,152 adres

Özel adresler

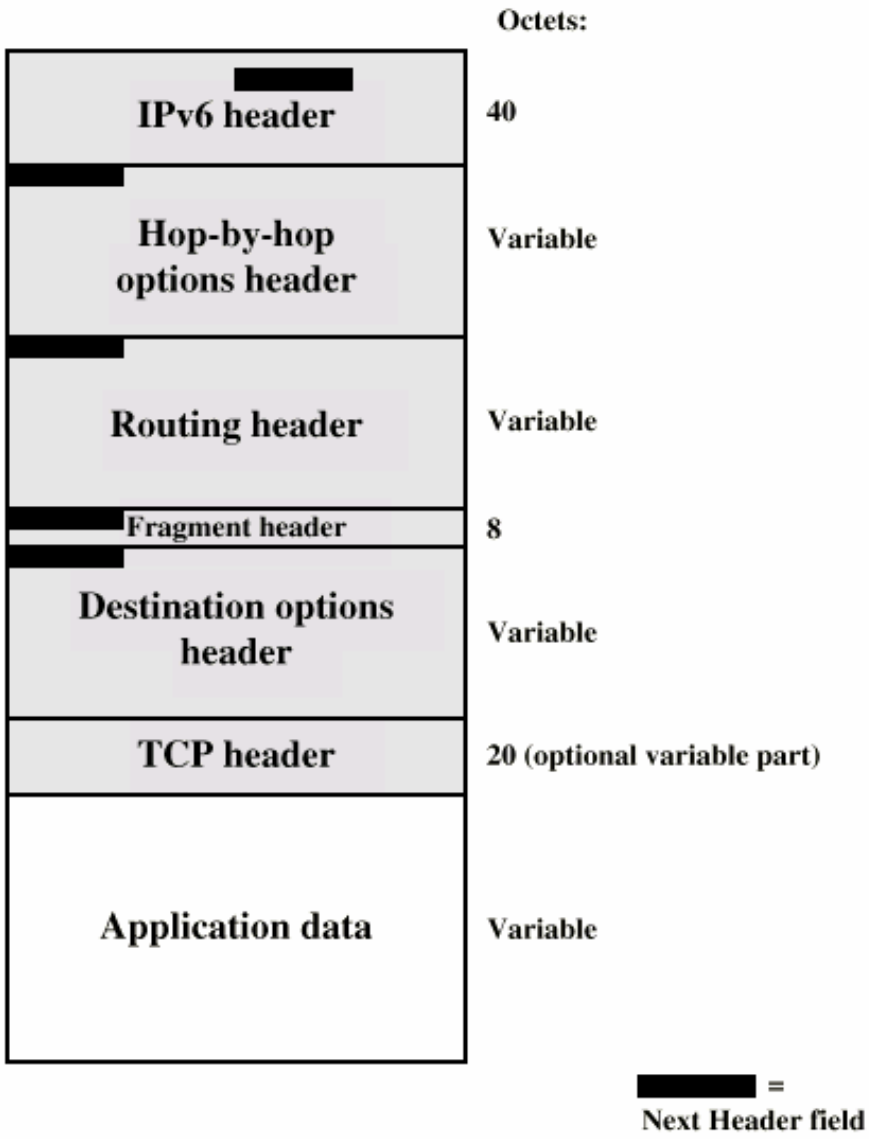
İsim	IP adres Bölgesi	Ip adres sayısı	*Sınıf*	Tanımalama
24-bit block	10.0.0.0 – 10.255.255.255	16,777,216	single class A	RFC 1597
20-bit block	172.16.0.0 – 172.31.255.255	1,048,576	16 aralıksız class B	(obsolete), RFC 1918
16-bit block	192.168.0.0 – 192.168.255.255	65,536	256 aralıksız class C	

IPV4 adres tıkanıklığı

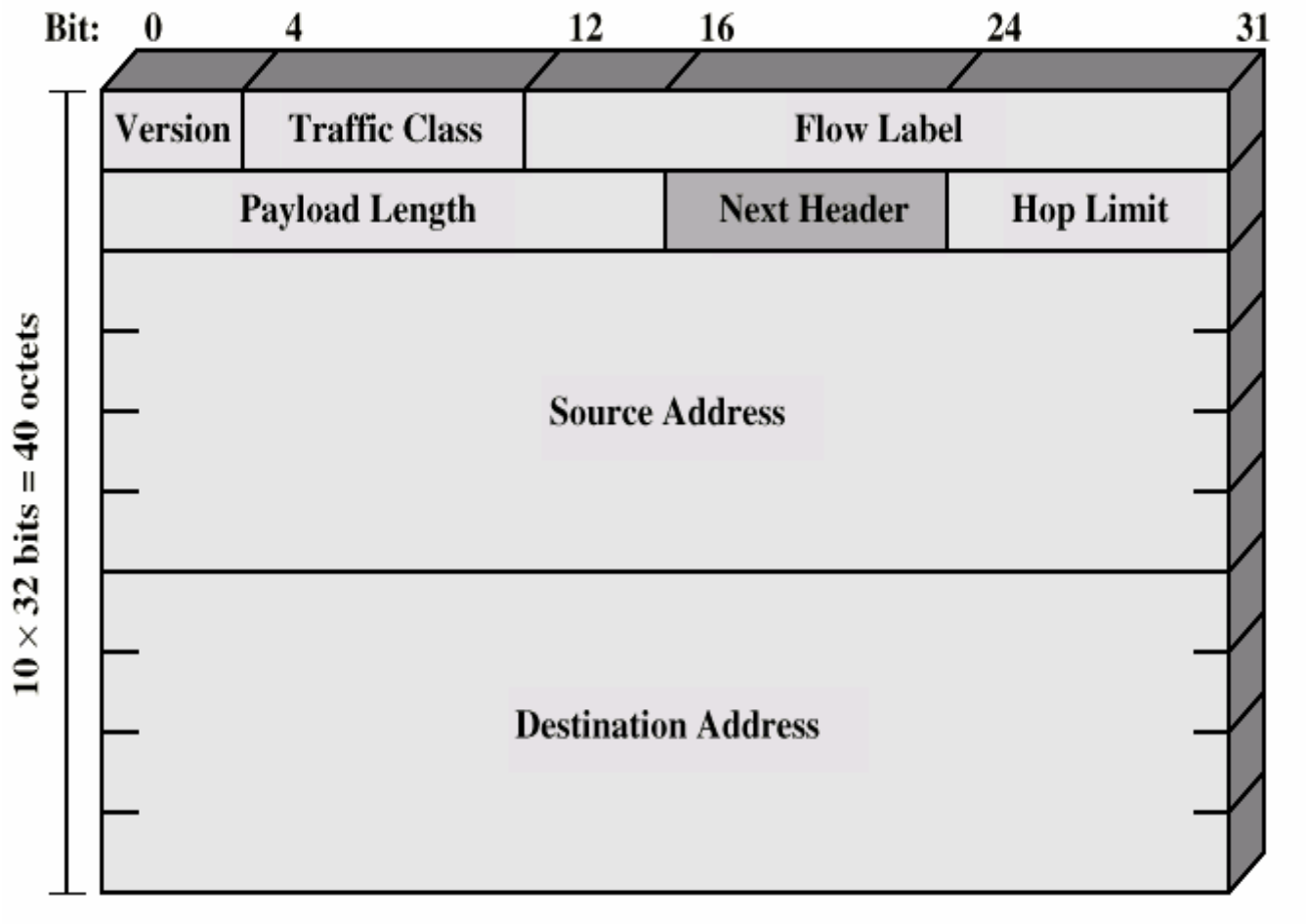
- NAT (Network Address Translation)
- Özel Adreslerin Kullanımı
- DHCP (Dinamik Host yapılandırma protokolü)

IPv6 (IPNG) RFC2460, 1998

- **Uzatılmış adres alanı**: IPv6 adresleme için 128 bit kullanır.
- **Geliştirilmiş Seçenek Mekanizması**: IPv6 başlığı ile iletim katmanı başlığı arasına yeni isteğe bağlı kullanılabilen başlıklar eklenmiştir. Yönlendirme işleminin daha hızlı yapılmasına olanak sağlar.
- **Kendiliğinden Adres Düzenleme**: ICMPv6 yı kullanarak IPv6 adreslerinin dinamik olarak atanmasını sağlar.
- **Arttırılmış Adresleme Esnekliği**:
- **Kaynak Ayırma İçin Destek**: Gerçek zamanlı video aktarımı gibi özelleştirilmiş trafik akışını düzenleyebilmek için geliştirilmiştir.
- **Güvenlik**: IPv6 güvenlik ve onaylama seçenekleri sunar.


Octets:
IPv6 header
40
Hop-by-hop options header
Variable
Routing header
Variable
Fragment header
8
Destination options header
Variable
TCP header
20 (optional variable part)
Application data
Variable
=
Next Header field

IPv6 Başlığı



TCP

- Bağlantıya dayalı bir protokoldür.
- Layer 4 ISO Transport'a denktir.
- Akış (Stream) tabanlıdır

TCP Hizmet Modeli

- TCP servisi soketlerle sağlanır
- Soketler IP adresi ve port numarasına sahiptir.
- Port numarası verinin makinada nereye gideceğini belirler.
- Tüm TCP bağlantıları noktadan noktaya ve full duplex

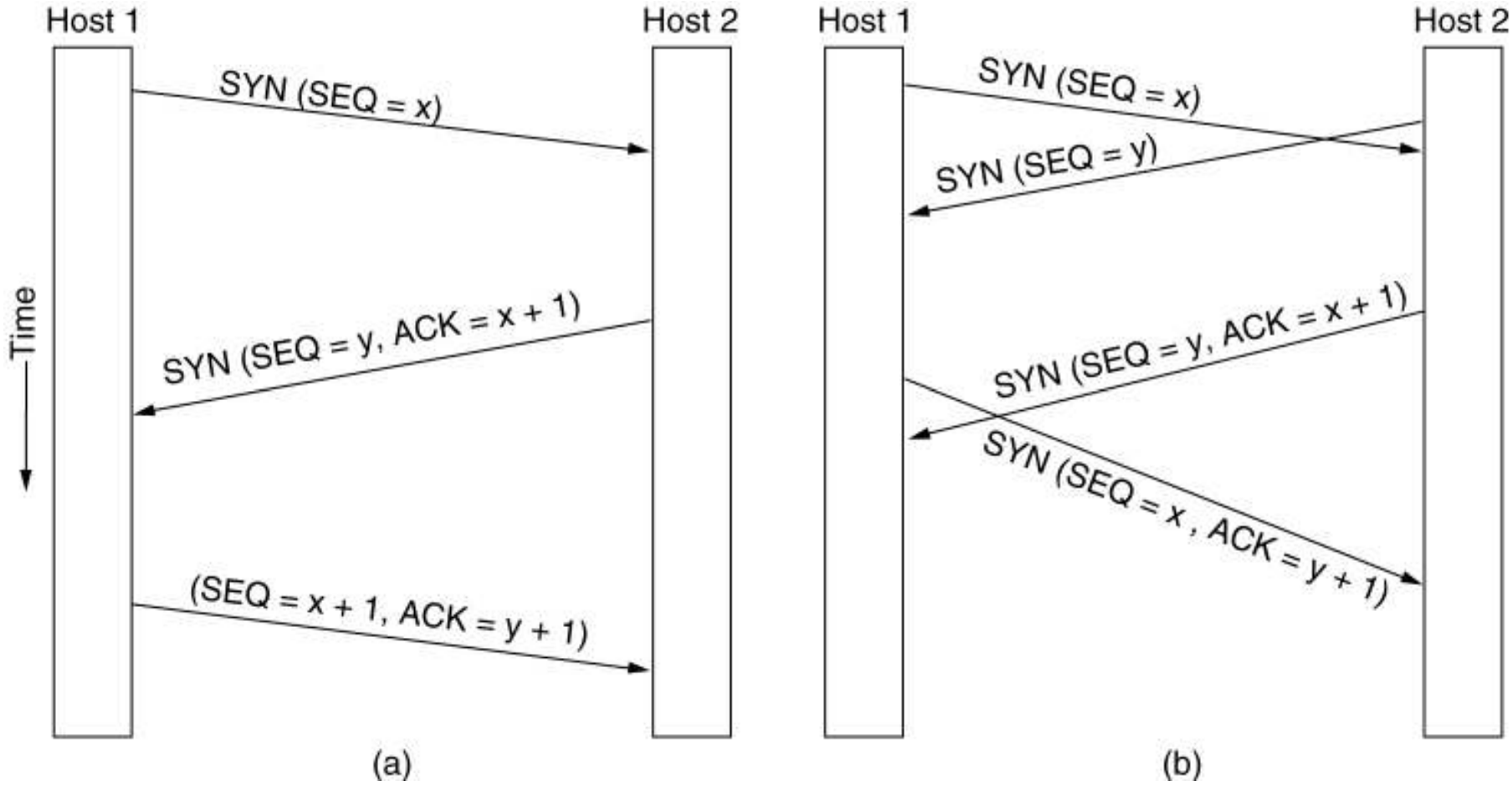
TCP Servis Modeli

- TCP veriyi tamponlayabilir.
- PUSH bayrağı ile veri doğrudan gönderilebilir.
- URGENT DATA bayrağı özel veri göndermek için kullanılır. Ör. DEL ya da CTRL-C.

TCP Bağlantı Kuruluşu

- Üç aşamalı el sıkışma kullanılır.
- Sunucu gelen istekleri pasif konumda bekler.
- İstemci, gerekli parametrelerle sunucudan bağlantı isteği yapar.
- Sunucu onay koduyla birlikte bağlantıyı sağlar.

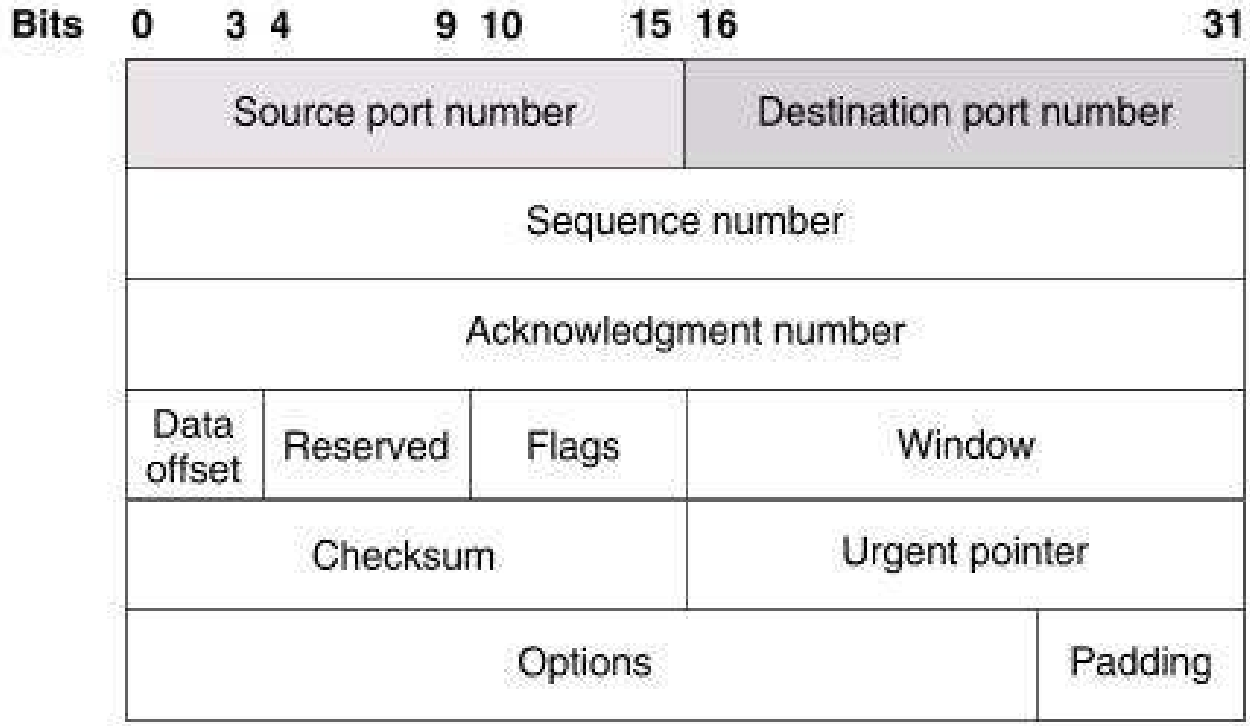
TCP Bağlantı Kurma



TCP Bağlantı Sonlandırma

- Her hangi bir tarafı FIN biti ile bağlantıyı sonlandırabilir.
- FIN kabul edildiğinde veri akışı kesilir.
- Bağlantının diğer tarafında da aynı işlem gerçekleştirilir.

TCP Başlığı



00	01	02	03	04	05
U	A	P	R	S	F

Bayrak alanı

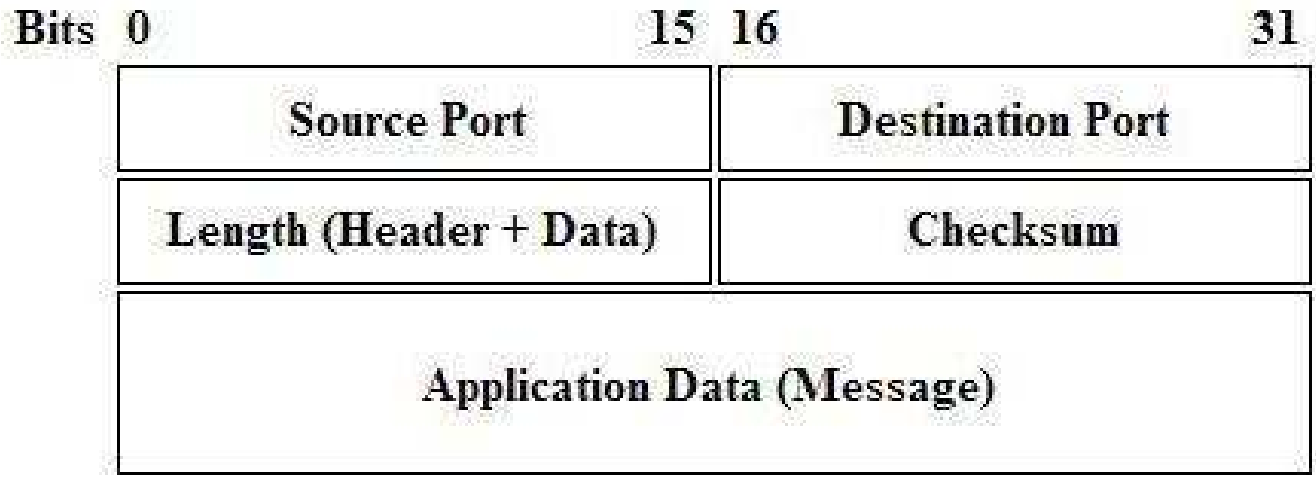
TCP Başlığı

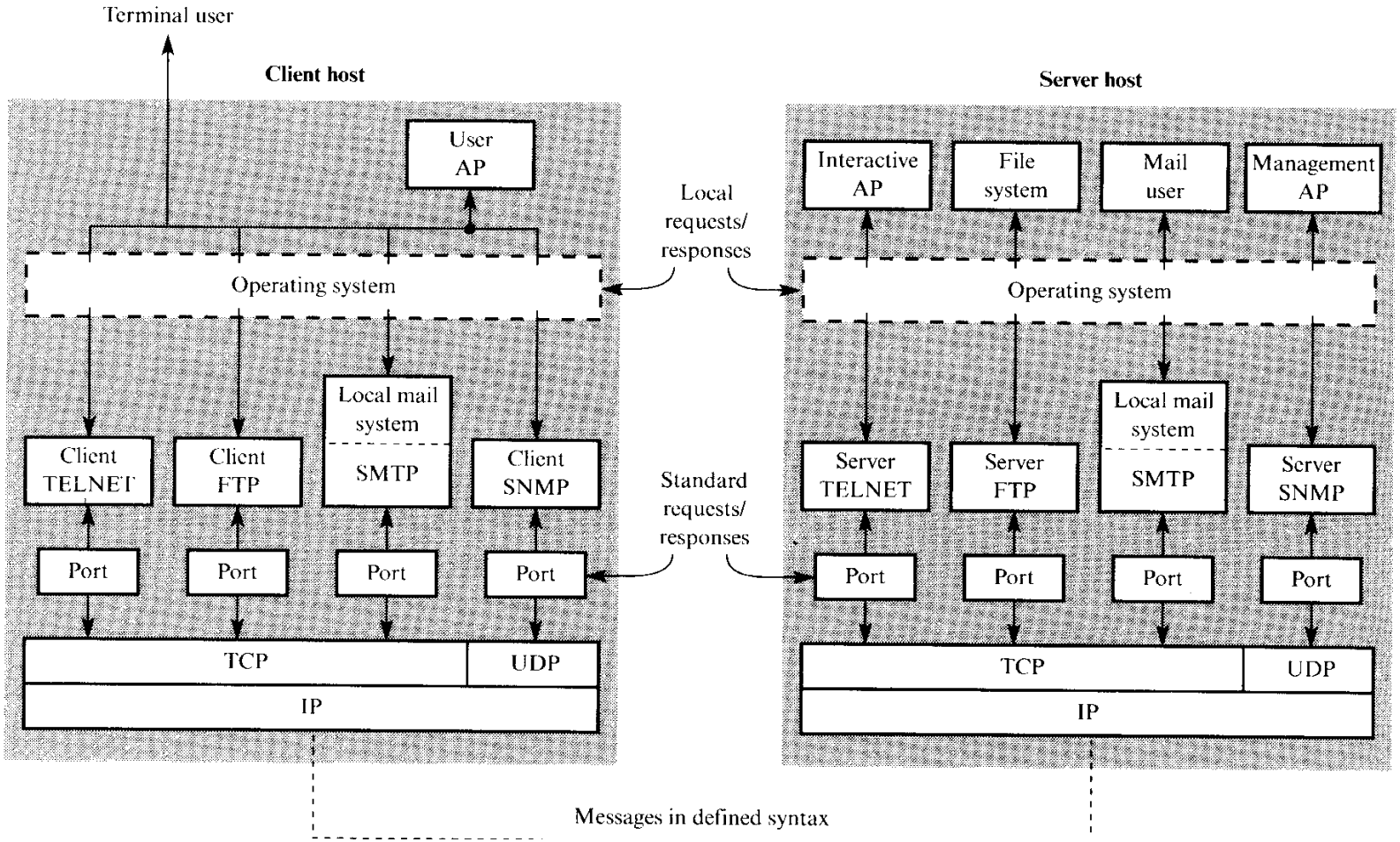
- Kaynak Portu (16 bit): kaynak servis erişim noktası
- Hedef Portu(16 bit): hedef servis erişim noktası
- Sıra Numarası(32 bit): Segment içindeki ilk veri oktetinin sıra numarası. SYN varsa, başlangıç sıra numarası
- Kabul numarası(32 bit): Sonraki oktetin sıra numarası.
- Veri Ofset(4 bit): Başlıktaki 32 bitlik kelimelerin sayısı
- Ayrılmış(6 bit): Gelecekte kullanılmak üzere ayrılmış 6 bit
- Bayraklar(6 bit): URG, ACK, PSH, RST, SYN, FIN
- Pencere(16 bit): oktet cinsinden akış kontrol bölgesi
- Checksum(16 bit)
- Acil Göstergesi(16 bit): acil veri takip eden okteti gösterir
- Seçenekler(değişken): geçerli tek seçenek, maksimum segment boyutudur.

UDP

- IP’ye port adresleme özelliği kazandırır
- Checksum seçimliktir
- Uzunluk alanı başlık ve veriyi kapsar

UDP BAŞLIĞI


Bits 0
15 16
31
Source Port
Destination Port
Length (Header + Data)
Checksum
Application Data (Message)

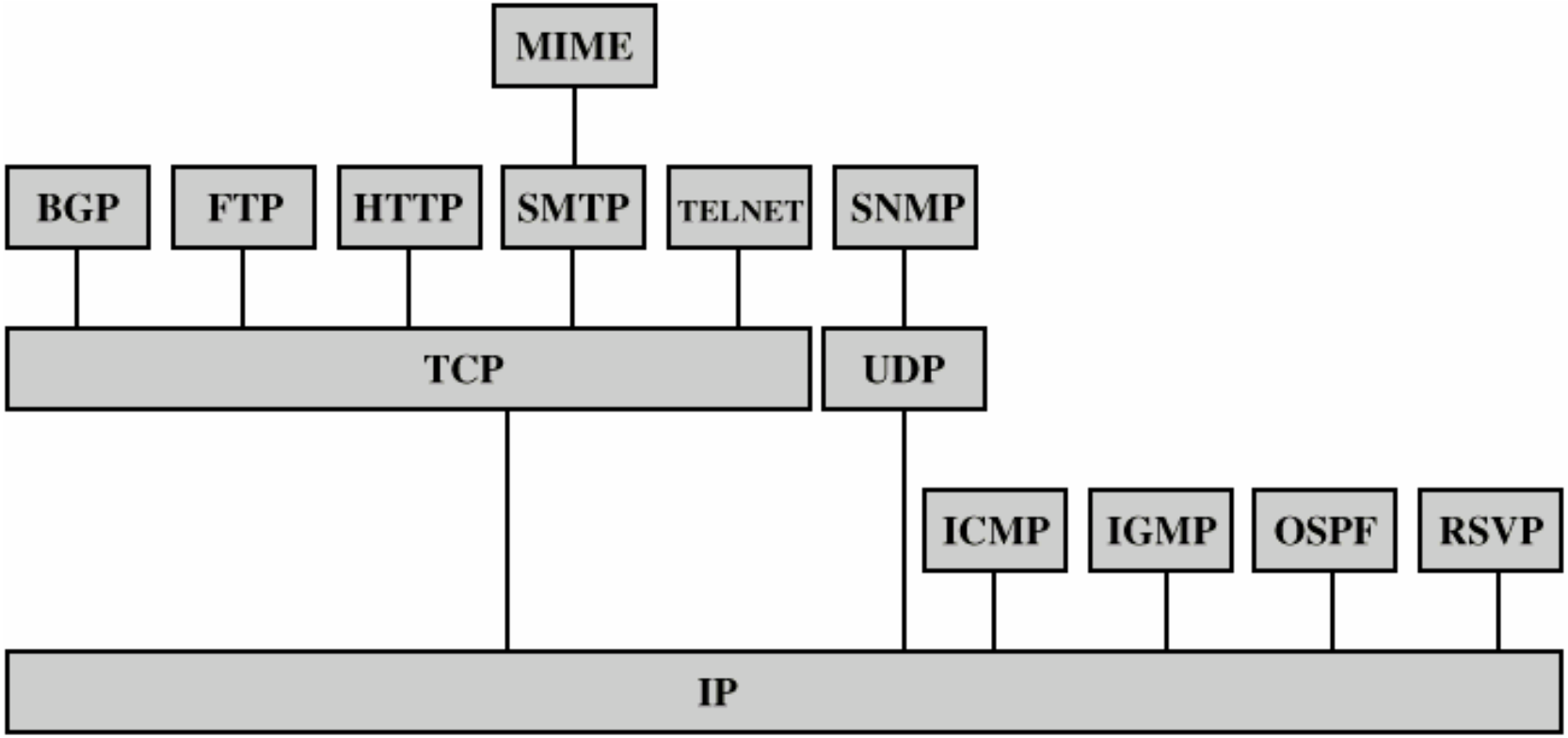


TCP/IP Uygulama Protokolleri

İnternet Uygulamaları: Uygulama ve İletim Protokolleri

Uygulama	Uygulama katmanı protokolü	İletim protokolü
e-mail	smtp [RFC 821]	TCP
Uzak terminal erişimi	telnet [RFC 854]	TCP
Web	http [RFC 2068]	TCP
Dosya transferi	ftp [RFC 959]	TCP
Multimedya bit akışı	proprietary	TCP or UDP

TCP/IP de bazı protokoller



BGP = Border Gateway Protocol
FTP = File Transfer Protocol
HTTP = Hypertext Transfer Protocol
ICMP = Internet Control Message Protocol
IGMP = Internet Group Management Protocol
IP = Internet Protocol
MIME = Multi-Purpose Internet Mail Extension
OSPF = Open Shortest Path First
RSVP = Resource ReSerVation Protocol
SMTP = Simple Mail Transfer Protocol
SNMP = Simple Network Management Protocol
TCP = Transmission Control Protocol
UDP = User Datagram Protocol

SNMP

- Dağıtık sistemlerde yönetim amaçlı bir protokol
- OSI karşılığı CMIP (Common Management Information Protocol)
- Önemli yetenekleri
  - Get: yönetim istasyonunun , istasyondaki nesne değerini alabilme
  - Set: İstasyondaki nesne değerini değiştirebilme
  - Notify: istasyondaki oluşan özel durumları yönetim istasyonununa iletebilme

SNMP

- Yönetim bilgi üssü MIB
  - Skaler değişkenler ve tablolar
- Yöneticinin istasyonlara giden ve gelen MIB değişkenlerine erişebilmesi için TRAP lar

SNMP - simple network management protocol
MIB - management information base

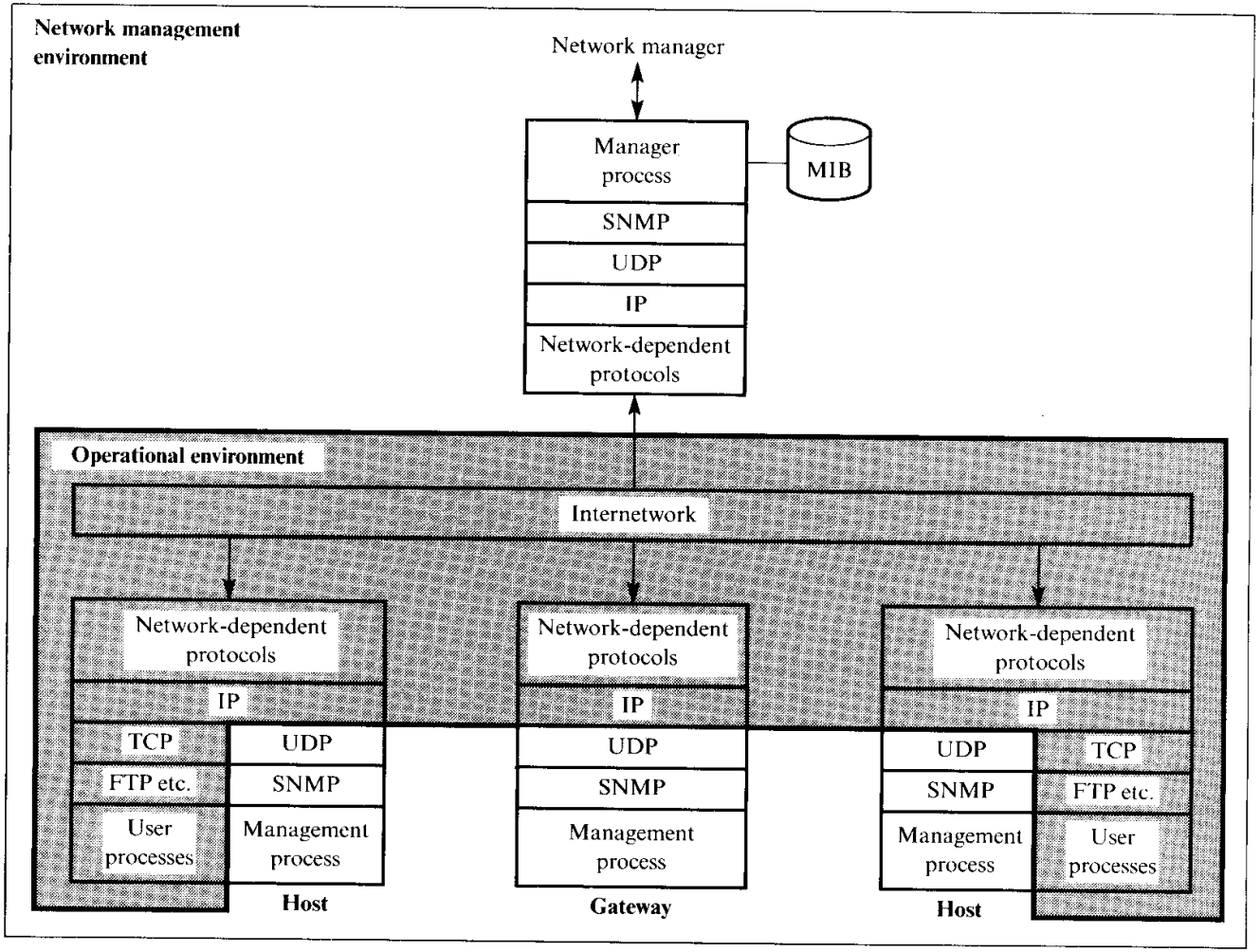


Fig. 13.7 SNMP network management

SMTP-MIME

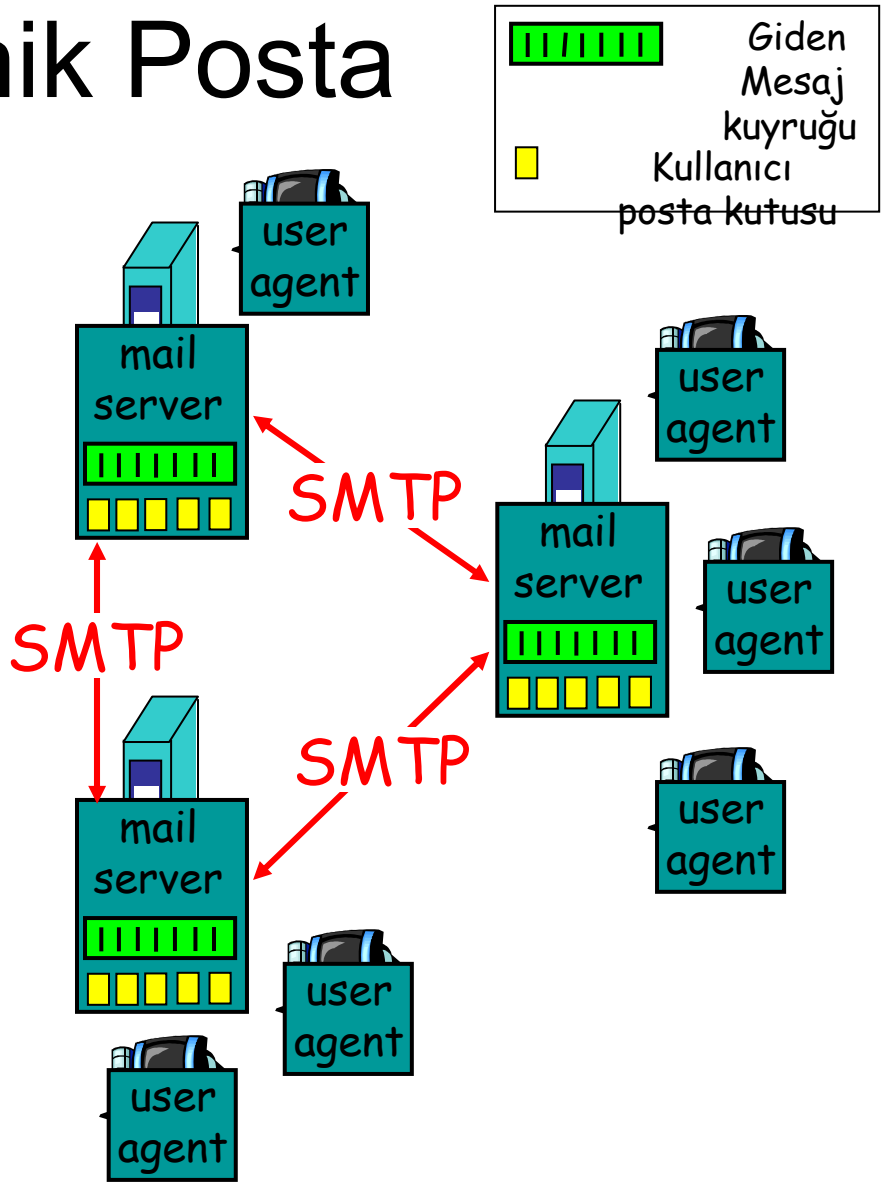
Elektronik Posta

Üç ana bileşen

- Kullanıcı arayüzü
- Posta sunucusu
- smtp

Kullanıcı arayüzü

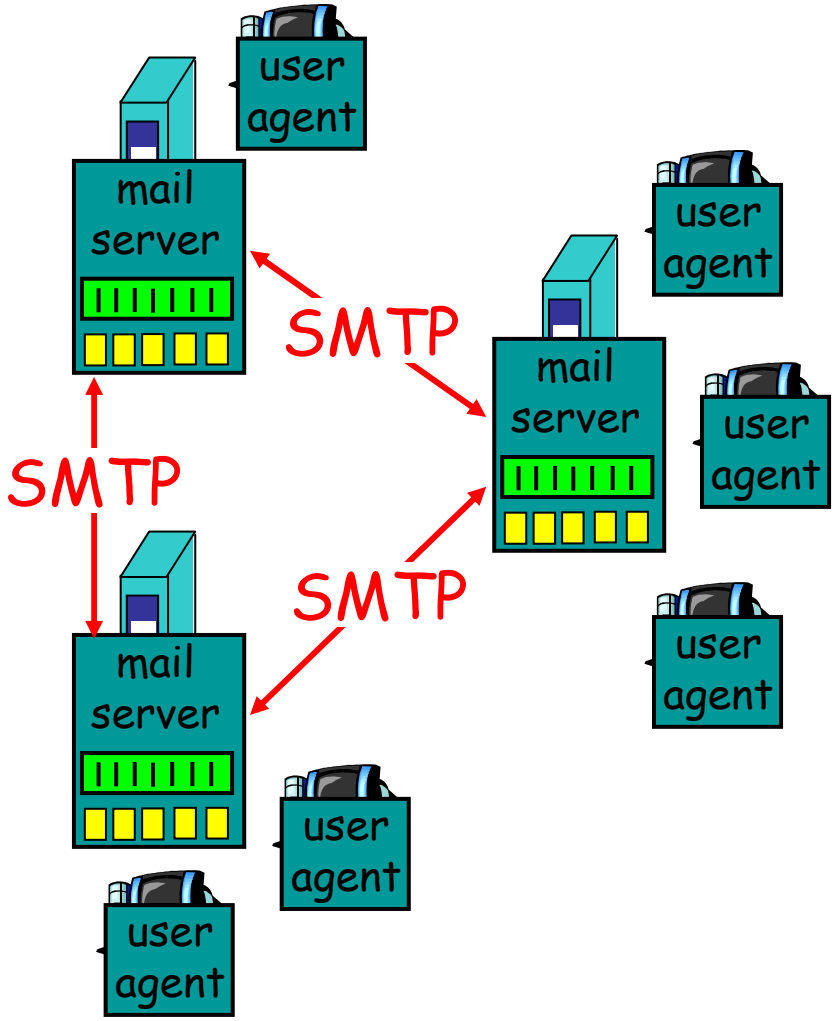
- “mail görüntüleyici”
- Posta mesajlarını okumaki yazmak, değiştirmek için
- Ör. Eudora, Outlook, elm, Netscape Messenger
- Giden ve gelen mesajlar sunucuda saklanır



Posta Sunucusu

Posta sunucusu

- mailbox kullanıcının okunmamış gelen postalarını tutar
- message gönderilen postalar için kuyruk
- smtp protocol sunucular arası posta göndermek için
  - client: gönderen posta sunucusu
  - "server": alan posta sunucusu



smtp [RFC 821]

- Eposta mesajını istemciden sunucuya taşımak için tcp ve port 25 kullanır.
- Doğrudan iletim: gönderen sunucudan alan sunucuya
- İletim üç aşamalıdır:
  - El sıkışma
  - Mesaj iletimi
  - sonlandırma
- Komut/yanıt
  - Komutlar ASCII metin
  - yanıtlar: durum kodu ve deyimler
- Mesajlar 7-bit ASCII olmak zorunda

telnet ile smtp

- **`telnet servername 25`**
- Sunucudan 220 yanıtını bekle
- HELO, MAIL FROM, RCPT TO, DATA, QUIT komutlarını gir

Yukarıdaki yönerge ile e-posta aracı kullanmadan e-posta atılabilir.

Örnek smtp

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250  Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C:   How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

smtp

- smtp sürekli bağlantı kullanır
- smtp mesajları (başlık & gövde) 7-bit ascii olmalı
- Bazı karakter dizilerine izin verilmez (e.g., `CRLF.CRLF`). Bu tür mesajlar kodlanmak zorundadır. (genellikle base-64 ya da işaretli yazılabilir olarak)
- smtp sunucu mesajın sonunu `CRLF.CRLF` koduyla anlar

http ile karşılaştırma

- http: pull
- email: push
- Her ikisi de ASCII komut/yanıt ve durum kodları kullanır
- http: her nesne kendi yanıt mesajına sahiptir.
- smtp: birden fazla nesne tek bir çok-parçalı mesajla gönderilir.

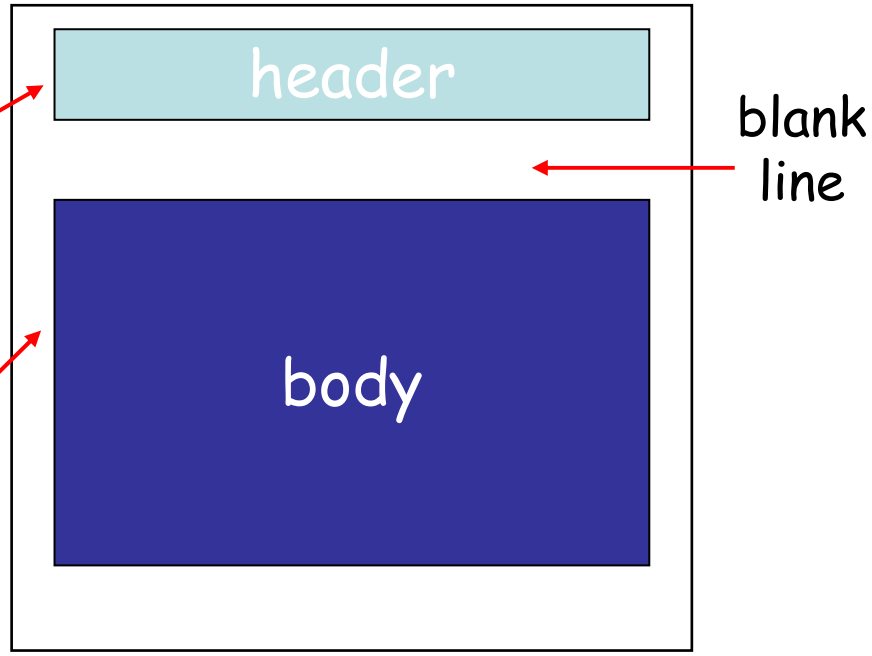
Mail mesaj formatı

smtp: eposta mesajlarını iletebilmek için protokol

RFC 822: metin mesaj formatı:

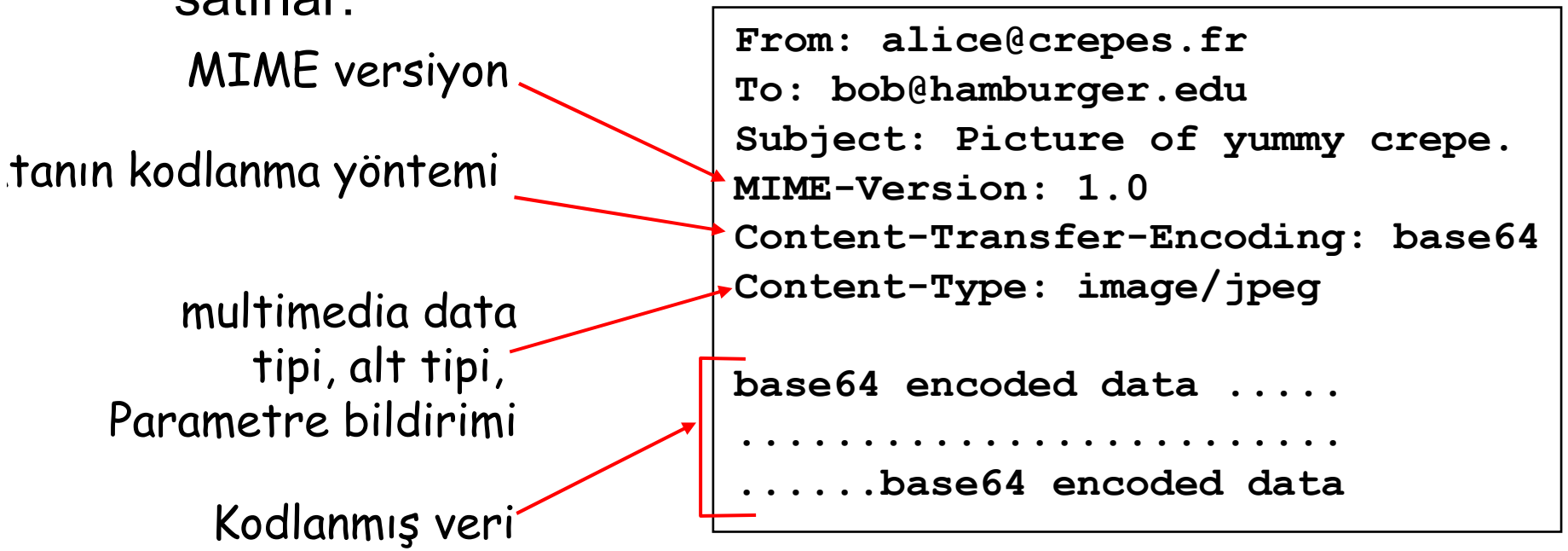
- Başlık mesajları:
  - To:
  - From:
  - Subject:

  smtp komutlarından farklı!!!
- gövde
  - "mesaj", yalnızca ASCII karakterler



Mesaj formatı: multimedia eklentileri

- MIME: multimedia mail extension, RFC 2045, 2056
- Başlıkta MIME içeriği tipini belirlemek için ek satırlar:



MIME tipleri

`Content-Type: type/subtype; parameters`

Metin

- örnek alt tipler: **`plain`**, **`html`**

Resim

- örnek alt tipler : **`jpeg`**, **`gif`**

Ses

- örnek alt tipler : **`basic`** (8-bit mu-law encoded), **`32kadpcm`** **(32 kbps coding)**

Video

- örnek alt tipler : **`mpeg`**, **`quicktime`**

Uygulamalar

- Görüntülenmeden önce bir okuyucuya gereksinim duyan diğer veriler
- örnek alt tipler : pdf, **`octet-stream`**

Multipart Type

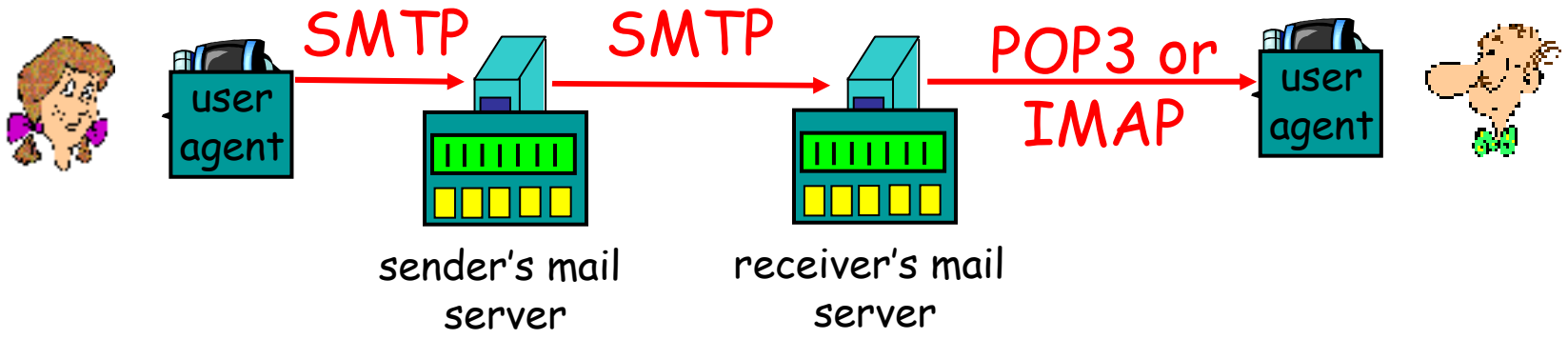
```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=98766789

--98766789
Content-Transfer-Encoding: quoted-printable
Content-Type: text/plain

Dear Bob,
Please find a picture of a crepe.
--98766789
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

base64 encoded data .....
.........................
......base64 encoded data
--98766789--
```

Mail erişim protokolleri



- SMTP: alıcının sunucusunda teslimat/saklama için
- Mail erişim protokolü: sunucudan indirmek için
  - POP: Post Office Protocol [RFC 1939]
    - yetkilendirme (agent <-->server) ve indirme
  - IMAP: Internet Mail Access Protocol [RFC 1730]
    - Daha fazla özellik (daha karmaşık)
    - Sunucuda saklanan mesajları değiştirmek için
  - HTTP: Hotmail , Yahoo! Mail, vs.

POP3 protokolü

Yetkilendirme fazı

- İstemci komutları:
  - **`user:`** kullanıcı adı bildir
  - **`pass:`** parola
- Sunucu yanıtı
  - **`+OK`**
  - **`-ERR`**

İletişim fazı,

istemci:

- **`list:`** mesaj sayısını göster
- **`retr:`** sırasına göre mesajı indir
- **`dele:`** sil
- **`quit : çık`**

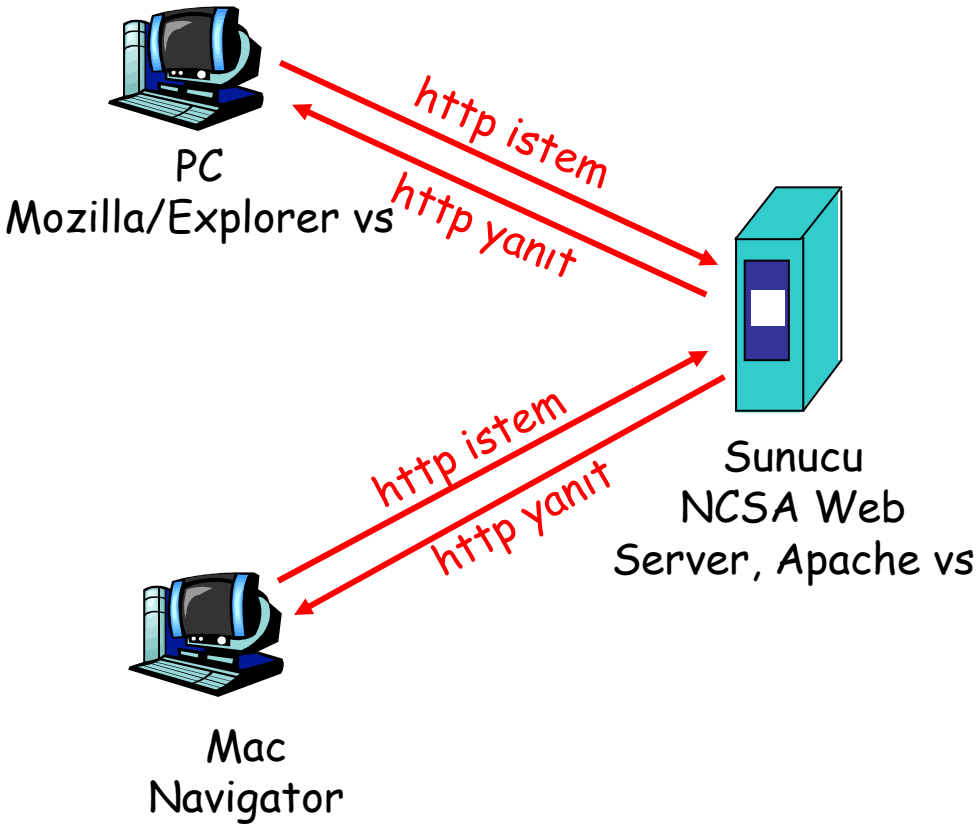
```
S: +OK POP3 server ready
C: user alice
S: +OK
C: pass hungry
S: +OK user successfully logged on
```

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

Web: http protokolü

http: hypertext transfer protocol

- Webin uygulama katmanı protokolü
- İstemci/sunucu modeli
  - *istemci:* istemde bulunan, web nesnelerini görüntüleyen tarayıcı
  - *sunucu:* Web sunucusu istenen nesneleri yollar
- http1.0: RFC 1945
- http1.1: RFC 2068



http protokolü

http: TCP iletim servisi:

- İstemci TCP bağlantısını başlatır, port 80
- Sunucu istemcinin isteğini kabul eder.
- tarayıcı (http client) ve Web sunucu (http server) arasında http mesajlaşma (uygulama katmanı mesajlar)
- TCP bağlantısı sonlandırılır

http "durumsuz"dur

- Sunucu istemcinin geçmiş istemlerini hatırlamaz.

"durum" yönetimi yapan protokoller karmaşıktır.

- geçmiş (durum) yönetilmeli
- İstemci/sunucu çökerse "durum" değişebilir, yeniden değerlendirilmeli

TCP/IP SERVER ve CLIENT uygulamaları (Windows)

Server

- Winsock ilklendirilir.
- Soket oluşturulur
- Sokete bağlanılır
- Bir client bağlantısı beklenir
- Client ten bağlantı kabul edilir.
- Veri alınır ve gönderilir.
- Bağlantı sonlandırılır.

Client

- Winsock ilklendirilir.
- Soket oluşturulur.
- Server'a bağlanılır.
- Veri alınır ve gönderilir.
- Bağlantı sonlandırılır.

Örnek bir C source kodu @VS6.0
library WS2_32.lib kullanıldı.

```
#include <stdafx.h>

#include <stdio.h>
#include "winsock2.h"

SOCKET ClientSocket = INVALID_SOCKET;
char recvbuf[512];
    int iResult, iSendResult;
    int recvbuflen = 512;

void main() {
  //----------------------
  // Initialize Winsock
  WSADATA wsaData;
  int iResult = WSAStartup(MAKEWORD(2,2), &wsaData);
  if (iResult != NO_ERROR)
    printf("Error at WSAStartup()\n");
```

```
//---------------------
  // Create a SOCKET for listening for
  // incoming connection requests
  SOCKET ListenSocket;
  ListenSocket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
  if (ListenSocket == INVALID_SOCKET) {
    printf("Error at socket(): %ld\n", WSAGetLastError());
    WSACleanup();
    return;
  }
  //---------------------
  // The sockaddr_in structure specifies the address family,
  // IP address, and port for the socket that is being bound.
  sockaddr_in service;
  service.sin_family = AF_INET;
  service.sin_addr.s_addr = inet_addr("127.0.0.1");
  service.sin_port = htons(27015);
```

```
//----------------------
  // Bind the socket.
  if (bind( ListenSocket,
    (SOCKADDR*) &service,
    sizeof(service)) == SOCKET_ERROR) {
    printf("bind() failed.\n");
    closesocket(ListenSocket);
    return;
  }

  iResult = listen(ListenSocket, SOMAXCONN);
    if (iResult == SOCKET_ERROR) {
        printf("listen failed: %d\n", WSAGetLastError());
        closesocket(ListenSocket);
        WSACleanup();
        return ;
    }

    // Accept a client socket
    ClientSocket = accept(ListenSocket, NULL, NULL);
    if (ClientSocket == INVALID_SOCKET) {
        printf("accept failed: %d\n", WSAGetLastError());
        closesocket(ListenSocket);
        WSACleanup();
        return ;
    }

    // No longer need server socket
    closesocket(ListenSocket);
```

```
// Receive until the peer shuts down the connection
   do {
      iResult = recv(ClientSocket, recvbuf, recvbuflen, 0);
      if (iResult > 0) {
         printf("Bytes received: %d\n", iResult);

      // Echo the buffer back to the sender
         iSendResult = send( ClientSocket, recvbuf, iResult, 0 );
         if (iSendResult == SOCKET_ERROR) {
            printf("send failed: %d\n", WSAGetLastError());
            closesocket(ClientSocket);
            WSACleanup();
            return ;
         }
         printf("Bytes sent: %d\n", iSendResult);
      }
      else if (iResult == 0)
         printf("Connection closing...\n");
      else  {
         printf("recv failed: %d\n", WSAGetLastError());
         closesocket(ClientSocket);
         WSACleanup();
         return ;
      }

   } while (iResult > 0);

   // shutdown the connection since we're done
   iResult = shutdown(ClientSocket, SD_SEND);
   if (iResult == SOCKET_ERROR) {
      printf("shutdown failed: %d\n", WSAGetLastError());
      closesocket(ClientSocket);
      WSACleanup();
      return ;
   }
```