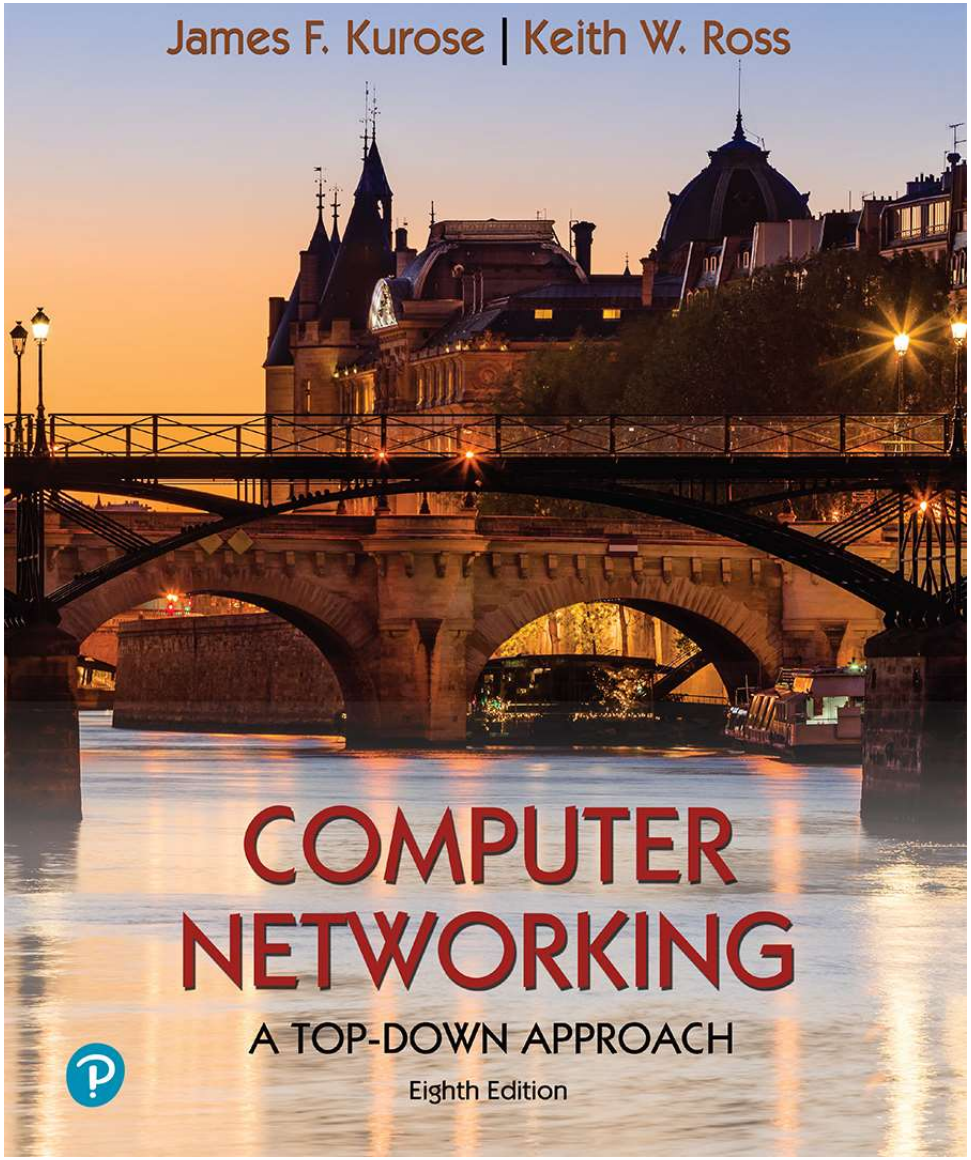


Bölüm 2
Uygulama Katmanı

Dr. Öğretim Üyesi A. Erhan AKKAYA

Uygulama katmanı: Genel bakış/yol haritası

- Ağ uygulamalarının prensipleri
- Web ve HTTP
- E-mail, SMTP, IMAP
- Domain Name System DNS

- P2P uygulamalar
- video akışı ve içerik dağıtım ağları
- UDP ve TCP ile soket programlama

Uygulama katmanı: genel bakış

Hedeflerimiz :

- Uygulama katmanı protokollerinin kavramsal ve uygulama yönleri
  - taşıma katmanı servis modelleri
  - client-server paradigması
  - peer-to-peer paradigması
- popüler uygulama katmanı protokollerini inceleyerek protokoller hakkında bilgi edineceğiz
  - HTTP
  - SMTP, IMAP
  - DNS
- ağ uygulamalarının programlanması
  - socket API

Bazı ağ uygulamaları

- Sosyal ağ
- Web
- Metin mesajlaşma
- E-posta
- çok kullanıcılı ağ oyunları
- depolanmış video akışı (YouTube, Hulu, Netflix)
- P2P dosya paylaşımı
- voice over IP (e.g., Skype)
- gerçek zamanlı video konferans
- Internet arama
- uzak bağlantı
- …

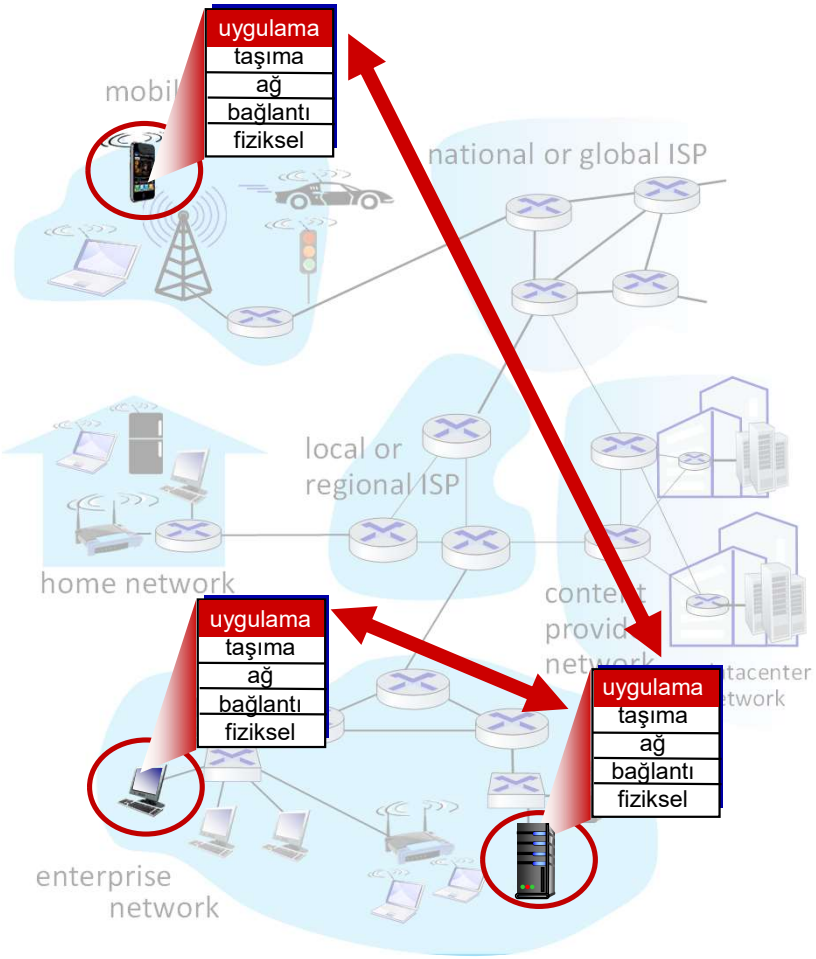
Bir ağ uygulaması oluşturma

programlar yazılmalı:

- (farklı) uç sistemler üzerinde çalıştırma
- ağ üzerinden iletişim
- örneğin, web sunucusu yazılımı tarayıcı yazılımı ile iletişim kurar

ağ çekirdekli cihazlar için yazılım yazmaya gerek yok

- ağ çekirdekli cihazlar kullanıcı uygulamalarını çalıştırmaz
- uç sistemlerdeki uygulamalar hızlı uygulama geliştirme, yayma



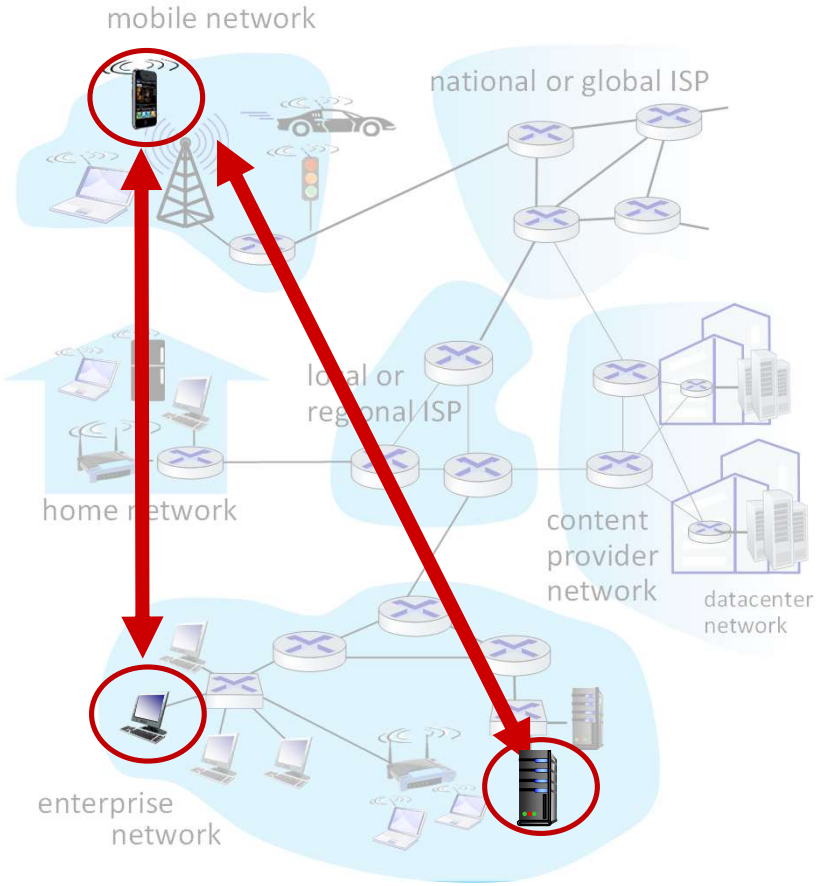
İstemci-sunucu paradigması

Server/sunucu:

- her zaman açık ana bilgisayar
- kalıcı IP adresi
- genellikle veri merkezlerinde, ölçeklendirme mevcut

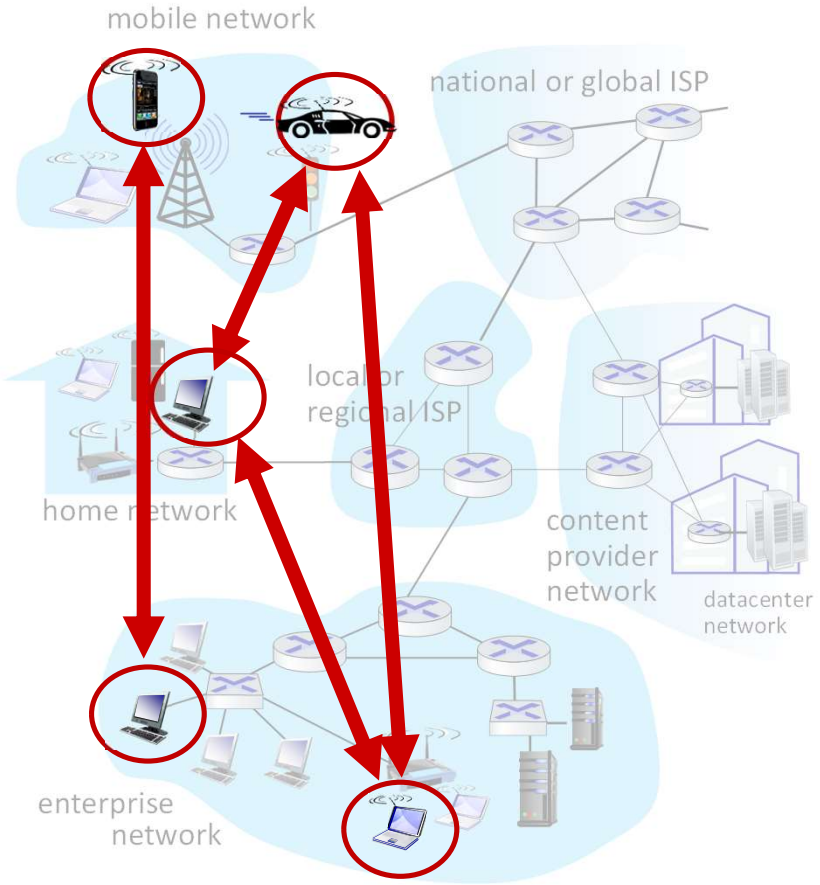
Clients/istemci :

- iletişim, sunucu ile iletişim
- aralıklı olarak bağlanabilir
- dinamik IP adreslerine sahip olabilir
- birbirleriyle doğrudan iletişim *kurmazlar*
- örnekler: HTTP, IMAP, FTP



Peer-peer mimari

- *Her zaman açık sunucu yok*
- Rastgele uç sistemler doğrudan iletişim kurar
- Eşler diğer eşlerden servis talep eder, karşılığında diğer eşlere servis sağlar
  - *Kendi kendine ölçeklenebilirlik – yeni eşler, yeni servis taleplerinin yanı sıra yeni servis kapasitesi de getirir*
- eşler zaman zaman bağlanıyor ve IP adreslerini değiştiriyor
  - Kompleks Yönetimi
- Örnek: P2P dosya paylaşımı



Proses / İşlem iletişimi

İşlem/process: Bir ana bilgisayar içinde çalışan program

- aynı ana bilgisayar içinde, iki işlem, işlemler arası iletişimi (işletim sistemi tarafından tanımlanır) kullanarak iletişim kurar
- Farklı ana bilgisayarlardaki işlemler *mesaj* alışverişi yaparak iletişim kurar

İstemciler, sunucular

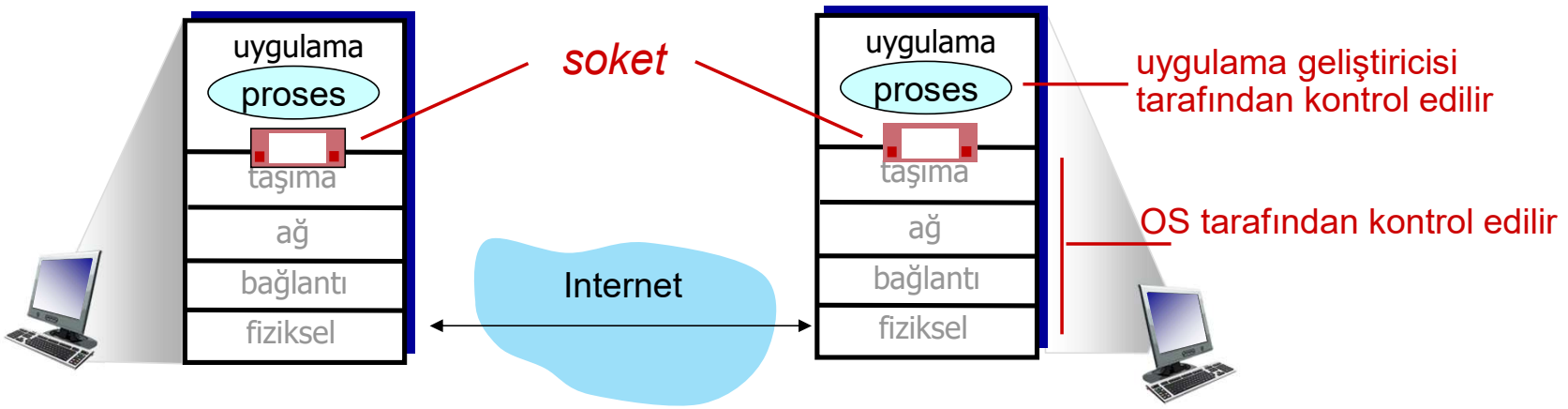
İstemci işlemi: İletişimi başlatan işlem

Sunucu işlemi: Kendisiyle iletişime geçilmesini bekleyen işlem

- Not: P2P mimarilerine sahip uygulamaların istemci işlemleri ve sunucu işlemleri vardır

Soket

- Süreç:: kendi soketine mesaj gönderir/kendi soketinden mesaj alır
- Soket kapıya benzetilebilir
  - Gönderme süreci mesajı kapıdan dışarı iter
  - Gönderen süreç, mesajı alıcı süreçteki sokete iletmek için kapının diğer tarafındaki taşıma altyapısına güvenir
  - Her iki tarafta birer tane olmak üzere iki soket söz konusudur



Adresleme İşlemleri

- mesajları almak için, süreç tanımlayıcıya sahip olmalıdır
- ana cihazın benzersiz 32 bit IP adresi vardır
- *S:* sürecin üzerinde çalıştığı ana bilgisayarın IP adresi süreci tanımlamak için yeterli midir?
  - *C:* Hayır, aynı ana bilgisayarda birçok işlem çalışıyor olabilir

- Tanımlayıcı, ana bilgisayardaki işlemle ilişkili hem IP adresini hem de port numarasını içerir.
- port numarası örnekleri:
  - HTTP sunucusu: 80
  - mail sunucusu: 25
- HTTP mesajlarını gaia.cs.umass.edu web sunucusuna göndermek için:
  - IP adres: 128.119.245.12
  - port numarası: 80

Uygulama katmanı protokolü aşağıdakileri tanımlar:

- Değişimi yapılan mesaj tipi,
  - örnek request, response
- Mesaja ait syntax:
  - mesajlardaki hangi alanlar ve alanların nasıl tanımlandığı
- Mesaja ait semantic:
  - alanlardaki bilginin anlamı
- süreçlerin mesajları ne zaman ve nasıl gönderip yanıtlayacağına ilişkin kurallar

Açık protokoller:

- RFC ile tanımlanırlar, herkesin protokol tanımına ve yapısına erişimi vardır
- birlikte çalışabilirliğe olanak sağlar
- örnek, HTTP, SMTP

tescilli protokoller:

- örneğin, Skype

Bir uygulama hangi taşıma servisine ihtiyaç duyar?

Veri bütünlüğü

- bazı uygulamalar (örn. dosya aktarımı, web işlemleri) %100 güvenilir veri aktarımı gerektirir
- diğer uygulamalar (örn. ses) bir miktar kaybı tolere edebilir

Aktarım Hızı

- bazı uygulamalar (örneğin multimedya) "etkili" olmak için minimum miktarda iş hacmi gerektirir
- diğer uygulamalar ("elastik uygulamalar") elde ettikleri her türlü verimi kullanırlar

Bir uygulama hangi taşıma servisine ihtiyaç duyar?

Veri bütünlüğü

- bazı uygulamalar (örn. dosya aktarımı, web işlemleri) %100 güvenilir veri aktarımı gerektirir
- diğer uygulamalar (örn. ses) bir miktar kaybı tolere edebilir

Aktarım Hızı

- bazı uygulamalar (örneğin multimedya) "etkili" olmak için minimum miktarda iş hacmi gerektirir
- diğer uygulamalar ("elastik uygulamalar") elde ettikleri her türlü verimi kullanırlar

Bir uygulama hangi taşıma servisine ihtiyaç duyar?

Zamanlama

- bazı uygulamalar (örn. internet telefonu, interaktif oyunlar) "etkili" olmak için düşük gecikme gerektirir

Güvenlik

- şifreleme, veri bütünlüğü, …

Taşıma servisi gereksinimleri : ortak uygulamalar

application	Veri kaybı	Aktarım hızı	Zaman hassasiyeti
Dosya aktarımı/download	kayıpsız	esnek	yok
e-posta	kayıpsız	esnek	yok
Web belgeleri	kayıpsız	esnek	yok
Gerçek zamanlı ses ve video	kayıp-toleranslı	ses: 5Kbps-1Mbps video:10Kbps-5Mbps	var, 10 ms
Kayıtlı ses/video	kayıp-toleranslı	Yukarı ile aynı	evet, birkaç sn.
Etkileşimli oyunlar	kayıp-toleranslı	birkaç kbps-10kbps	evet, 100 ms
Telefon mesajlaşması	kayıpsız	esnek	var ve yok

İnternet taşıma protokol servisleri

TCP servisi:

- *güvenilir veri taşıması:* gönderme ve alma işlemleri arasında
- *akış kontrolü:* gönderici alıcıyı bunaltamaz
- *tıkanıklık kontrolü:* ağ aşırı yüklendiğinde göndericiyi yavaşlatma
- *sağlamaz:* zamanlama, minimum iş gücü garantisi, güvenlik
- *bağlantı odaklı:* istemci ve sunucu işlemleri arasında kurulum gerekli

UDP servisi:

- *güvenilmez veri taşıması:* gönderme ve alma işlemleri arasında
- *sağlamaz:* güvenilirlik, akış kontrolü, tıkanıklık kontrolü, zamanlama, iş gücü garantisi, güvenlik veya bağlantı kurulumu.

S: UDP neden var?

İnternet taşıma protokol servisleri

TCP servisi:

- *güvenilir veri taşıması:* gönderme ve alma işlemleri arasında
- *akış kontrolü:* gönderici alıcıyı bunaltamaz
- *tıkanıklık kontrolü:* ağ aşırı yüklendiğinde göndericiyi yavaşlatma
- *sağlamaz:* zamanlama, minimum iş gücü garantisi, güvenlik
- *bağlantı odaklı:* istemci ve sunucu işlemleri arasında kurulum gerekli

UDP servisi:

- *güvenilmez veri taşıması:* gönderme ve alma işlemleri arasında
- *sağlamaz:* güvenilirlik, akış kontrolü, tıkanıklık kontrolü, zamanlama, iş gücü garantisi, güvenlik veya bağlantı kurulumu.

S: UDP neden var?

İnternet taşıma protokol servisleri

TCP servisi:

- *güvenilir veri taşıması:* gönderme ve alma işlemleri arasında
- *akış kontrolü:* gönderici alıcıyı bunaltamaz
- *tıkanıklık kontrolü:* ağ aşırı yüklendiğinde göndericiyi yavaşlatma
- *sağlamaz:* zamanlama, minimum iş gücü garantisi, güvenlik
- *bağlantı odaklı:* istemci ve sunucu işlemleri arasında kurulum gerekli

UDP servisi:

- *güvenilmez veri taşıması:* gönderme ve alma işlemleri arasında
- *sağlamaz:* güvenilirlik, akış kontrolü, tıkanıklık kontrolü, zamanlama, iş gücü garantisi, güvenlik veya bağlantı kurulumu.

S: UDP neden var?

İnternet taşıma protokol servisleri

TCP servisi:

- *güvenilir veri taşıması:* gönderme ve alma işlemleri arasında
- *akış kontrolü:* gönderici alıcıyı bunaltamaz
- *tıkanıklık kontrolü:* ağ aşırı yüklendiğinde göndericiyi yavaşlatma
- *sağlamaz:* zamanlama, minimum iş gücü garantisi, güvenlik
- *bağlantı odaklı:* istemci ve sunucu işlemleri arasında kurulum gerekli

UDP servisi:

- *güvenilmez veri taşıması:* gönderme ve alma işlemleri arasında
- *sağlamaz:* güvenilirlik, akış kontrolü, tıkanıklık kontrolü, zamanlama, iş gücü garantisi, güvenlik veya bağlantı kurulumu.

S: UDP neden var?

Internet uygulamaları, uygulama katmanı protokolleri ve temel taşıma protokolleri

Uygulama	Uygulama katmanı protokolü	Taşıma protokolü
Dosya aktarımı/download	FTP [RFC 959]	TCP
Elektronik posta	SMTP [RFC 5321]	TCP
Web dökümanlar	HTTP 1.1 [RFC 7230]	TCP
Internet telefon	SIP [RFC 3261], RTP [RFC 3550], veya özel (Skype)	TCP veya UDP
Multimedya akışı	HTTP (YouTube), DASH	TCP
İnteraktif oyunlar	WOW, FPS (tescilli)	UDP veya TCP

TCP'nin Güvenliğini Sağlama

Vanilya TCP ve UDP soketleri:

- Şifreleme yok
- Sokete gönderilen açık metin parolaları açık metin olarak İnternet'te dolanır(!)

Transport Layer Security (TLS)

- şifreli TCP bağlantıları sağlar
- Veri bütünlüğü
- Uç nokta kimlik doğrulaması

Uygulama katmanında uygulanan TLS

- uygulamalar TLS kitaplıklarını kullanır ve bu kitaplıklar da sırayla TCP kullanır

TLS soket API

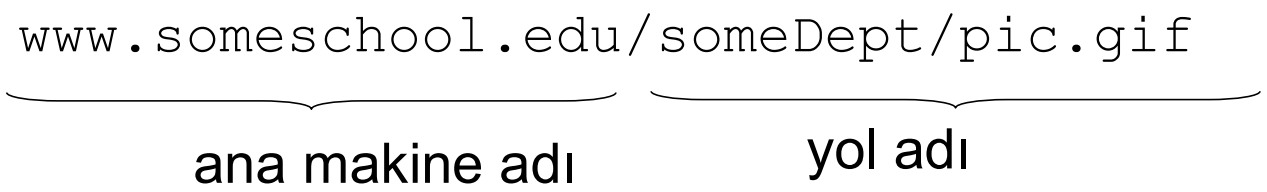
- sokete gönderilen açık metin interneti şifreli olarak geçer

Uygulama katmanı: Genel bakış/yol haritası

- Ağ uygulamalarının prensipleri
- Web ve HTTP
- E-mail, SMTP, IMAP
- Domain Name System DNS
- P2P uygulamalar
- Video akışı ve içerik dağıtım ağları
- UDP ve TCP ile soket programlama

Web ve HTTP

- web sayfası, her biri farklı Web sunucularında depolanabilen ***nesnelerden*** oluşur
- nesne HTML dosyası, JPEG görüntüsü, Java uygulaması, ses dosyası olabilir…
- web sayfası, *her biri bir URL* ile *adreslenebilen birkaç referans nesnesi* içeren *temel HTML dosyası*ndan oluşur



HTTP genel bakış

HTTP: hypertext transfer protocol

- Web'in uygulama katmanı protokolü
- İstemci/sunucu (client/server) modeli:
  - *istemci:* Web nesnelerini isteyen, alan (HTTP protokolünü kullanarak) ve "görüntüleyen" tarayıcı
  - *sunucu:* Web sunucusu isteklere yanıt olarak nesneler gönderir (HTTP protokolünü kullanarak)

HTTP genel bakış (devam)

HTTP, TCP kullanır:

- istemci sunucuya TCP bağlantısı başlatır (soket oluşturur), port 80
- sunucu istemciden gelen TCP bağlantısını kabul eder
- Tarayıcı (HTTP istemcisi) ve Web sunucusu (HTTP sunucusu) arasında HTTP mesajları değiş tokuş edilir (uygulama katmanı protokol mesajları)
- TCP bağlantısı kapatılır

HTTP "durumsuz"dur

- sunucu geçmiş *istemci* istekleri hakkında hiçbir bilgi tutmaz

Ayrıca

"durumu" koruyan protokoller karmaşıktır!

- geçmiş tarih (durum) muhafaza edilmelidir
- sunucu/istemci çökerse, "durum" hakkındaki bilgi tutarsız olabilir

HTTP bağlantıları: iki türdür

Kalıcı olmayan HTTP

1. TCP bağlantısı açılır
2. TCP bağlantısı üzerinden en fazla bir nesne gönderilir
3. TCP bağlantısı kapatılır

Birden fazla nesnenin indirilmesi için birden fazla bağlantı gerekir

Kalıcı HTTP

- Sunucuya TCP bağlantısı açılır
- istemci ile sunucu arasındaki tek bir TCP bağlantısı üzerinden birden fazla nesne gönderilebilir
- TCP bağlantısı kapatılır

Kalıcı olmayan HTTP örneği

Kullanıcı URL'yi girer : **`www.someSchool.edu/someDepartment/home.index`**
(metin, 10 jpeg resme referanslar içerir)

1a. HTTP istemcisi, 80 numaralı bağlantı noktasında www.someSchool.edu adresindeki HTTP sunucusuna (proses) TCP bağlantısı başlatır

1b. www.someSchool.edu ana bilgisayarındaki HTTP sunucusu 80 numaralı bağlantı noktasında TCP bağlantısını bekler, bağlantıyı "kabul eder" ve istemciyi bilgilendirir

2. HTTP istemcisi TCP bağlantı soketine HTTP *istek mesajı* (URL içeren) gönderir. Mesaj, istemcinin someDepartment/home.index nesnesini istediğini belirtir

3. HTTP sunucusu istek mesajını alır, istenen nesneyi içeren *yanıt mesajını* oluşturur ve mesajı soketine gönderir

zaman

Kalıcı olmayan HTTP örneği

Kullanıcı URL'yi girer : **`www.someSchool.edu/someDepartment/home.index`**
(metin, 10 jpeg resme referanslar içerir)

4. HTTP sunucusu TCP bağlantısını kapatır.

5. HTTP istemcisi html dosyası içeren yanıt mesajı alır, html görüntüler. Html dosyası ayrıştırılır, başvurulan 10 jpeg nesnesi bulunur.

6. 1-5 arasındaki adımlar 10 jpeg nesnesinin her biri için tekrarlanır

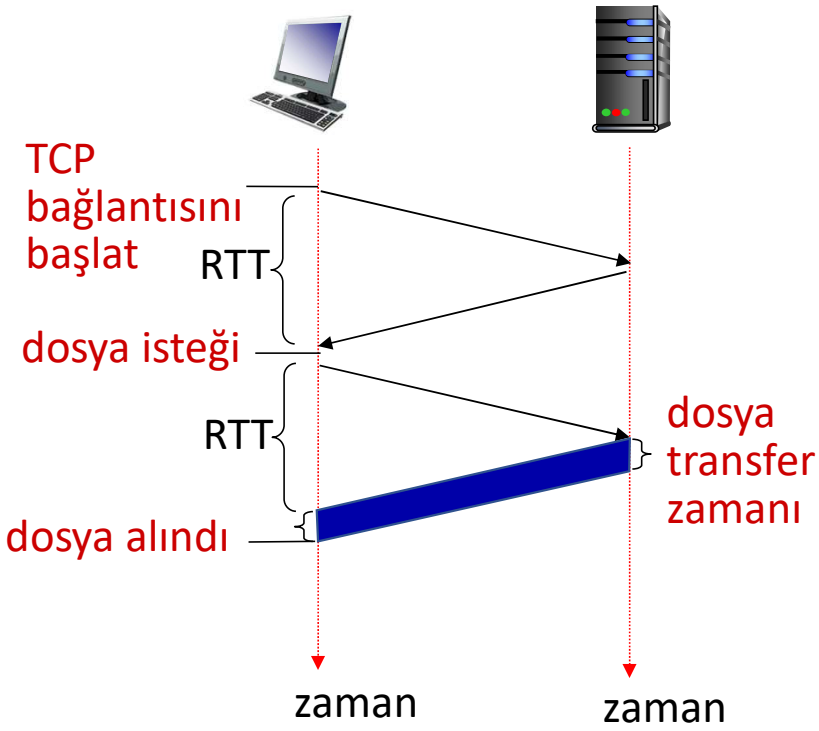
zaman

Kalıcı olmayan HTTP: yanıt süresi

RTT: Küçük bir paketin istemciden sunucuya gidip gelmesi için geçen süre

HTTP yanıt süresi (her bir nesne için):

- TCP bağlantısını başlatmak için bir RTT
- HTTP isteği için bir RTT ve HTTP yanıtının ilk birkaç baytının döndürülmesi
- nesne/dosya iletim süresi



Kalıcı olmayan HTTP yanıt süresi = 2RTT+ dosya transfer zamanı

Kalıcı HTTP (HTTP 1.1)

Kalıcı olmayan HTTP sorunları:

- nesne başına 2 RTT gerektirir
- Her TCP bağlantısı için işletim sistemine ek yükü vardır
- tarayıcılar genellikle referans verilen nesneleri paralel olarak almak için birden fazla paralel TCP bağlantısı açar

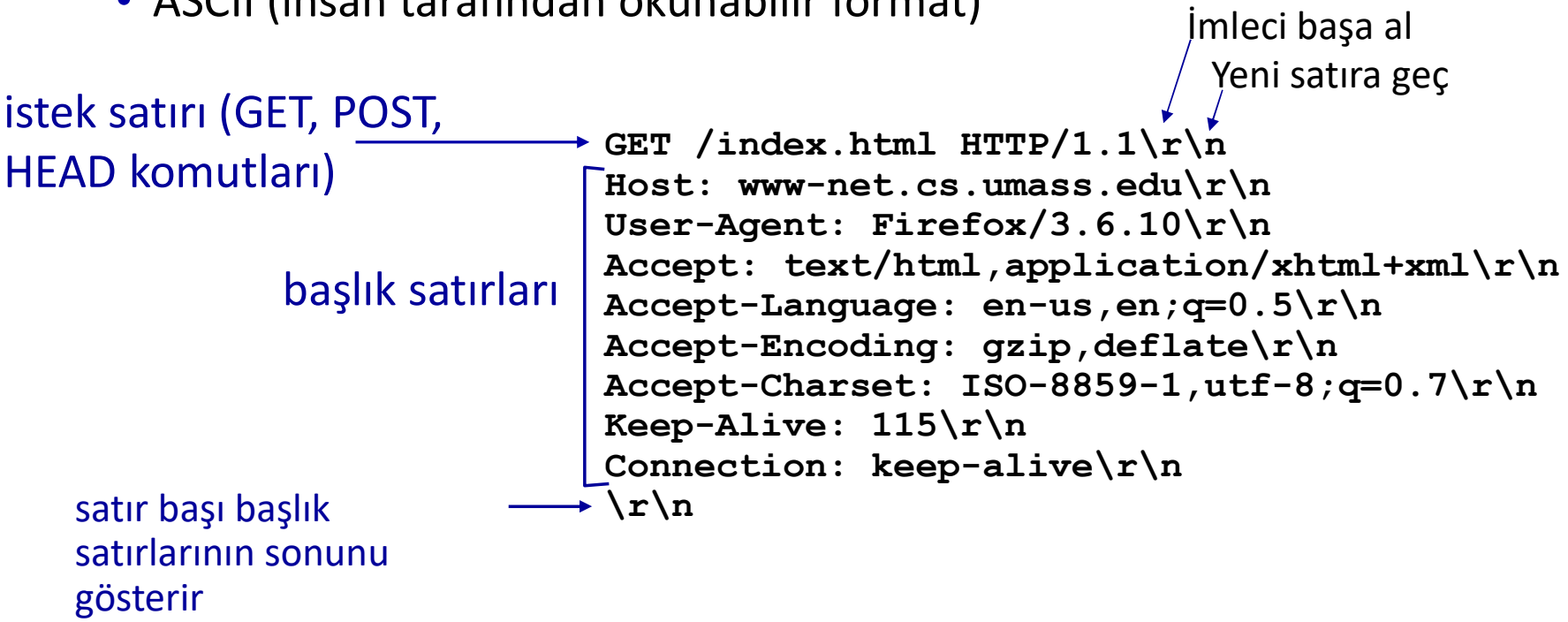
Kalıcı HTTP (HTTP1.1):

- sunucu yanıt gönderdikten sonra bağlantıyı açık bırakır
- aynı istemci/sunucu arasında açık bağlantı üzerinden gönderilen sonraki HTTP mesajları
- istemci, başvurulan bir nesneyle karşılaşır karşılaşmaz istek gönderir
- başvurulan tüm nesneler için bir RTT kadar az (yanıt süresini yarıya indirir)

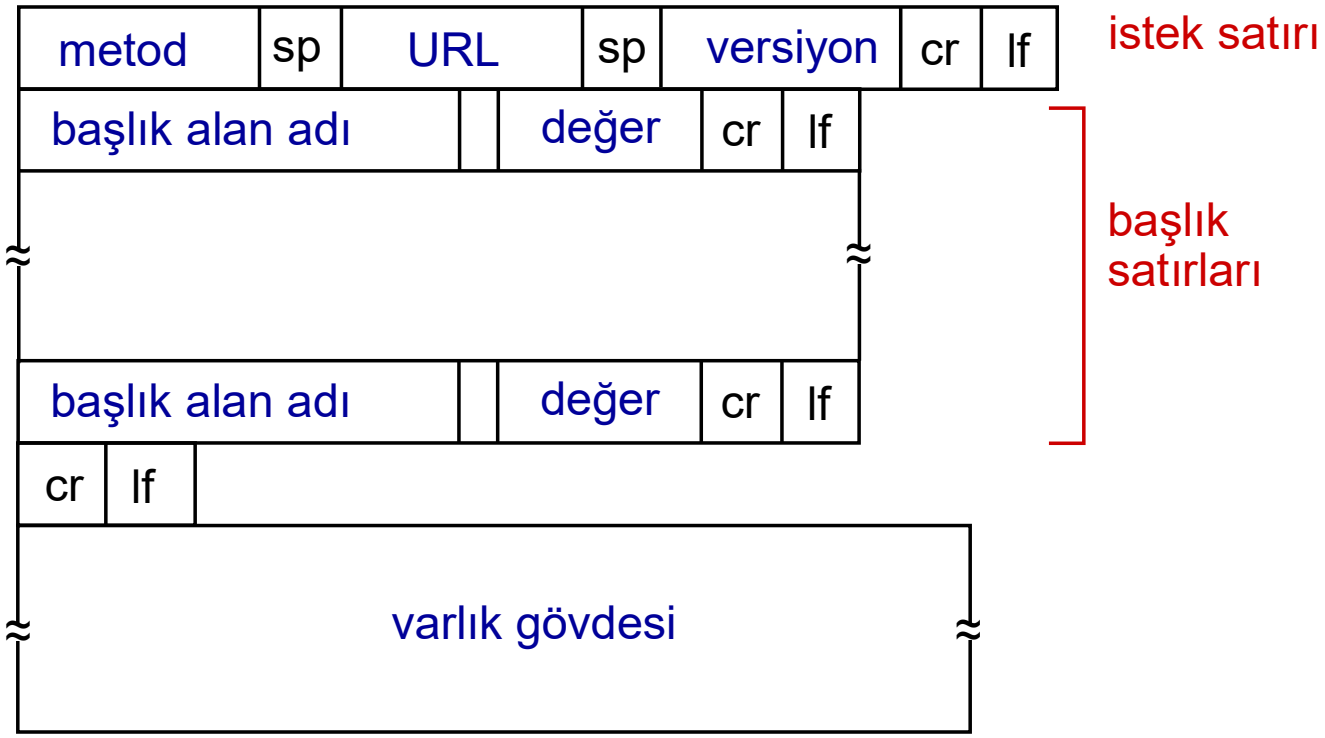
HTTP istek (request) mesajı

İmleci başa al + Yeni satıra geç = Satır başı yap

- İki tür HTTP mesajı vardır: *istek*, *yanıt*
- HTTP istek mesajı:
  - ASCII (insan tarafından okunabilir format)



HTTP istek mesajının genel formatı



Diğer HTTP istek metotları

POST metodu:

- web sayfası genellikle form girişi içerir
- HTTP POST istek mesajının varlık gövdesinde istemciden sunucuya gönderilen kullanıcı girdisi

GET metodu: (sunucuya veri göndermek için):

- HTTP GET istek mesajının URL alanına kullanıcı verilerini eklenir ('?' işaretinden sonra):

```
www.somesite.com/animalsearch?monkeys&bananas
```

HEAD metodu:

- belirtilen URL bir HTTP GET yöntemiyle istendiğinde döndürülecek üstbilgileri (yalnızca) ister.

PUT metodu:

- sunucuya yeni dosya (nesne) yükler
- POST HTTP istek iletisinin varlık gövdesindeki içerikle belirtilen URL'de var olan dosyayı tamamen değiştirir

HTTP yanıt (response) mesajı

durum satırı (protocol durum kodu durum ifadesi)

başlık satırları

veri, örneğin, istenen HTML dosyası

```
HTTP/1.1 200 OK\r\n
Date: Sun, 26 Sep 2010 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007 17:00:02
   GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=ISO-8859-
   1\r\n
\r\n
data data data data data ...
```

HTTP yanıt durum kodları

- Durum kodu sunucudan istemciye yanıt mesajında 1. satırda görünür.
- Bazı örnek kodlar:

200 OK
- istek başarılı oldu, istenen nesne bu mesajın ilerleyen bölümlerinde

301 Moved Permanently
- istenen nesne kalıcı olarak başka bir yere taşındı

400 Bad Request
- istek mesajı sunucu tarafından anlaşılmadı

404 Not Found
- istenen belge bu sunucuda bulunamadı

505 HTTP Version Not Supported
- Sunucu istenen HTTP sürümünü desteklemiyor.