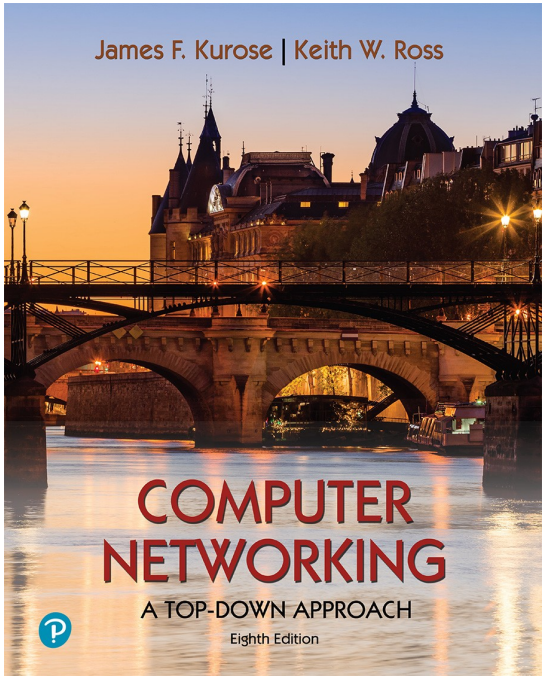
Bölüm 2
Uygulama Katmanı

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

Bölüm 2: Uygulama katmanı

hedeflerimiz:

- Ağ uygulama protokollerinin kavramsal ve oluşturulma prensipleri
  - Taşıma katmanı hizmet modelleri
  - Sunucu-istemci mimarisi
  - Eşler arası (İng: Peer-to-peer) mimari
- Populer uygulama katmanı protokellerini inceleyeceğiz.
  - HTTP
  - FTP
  - SMTP / POP3 / IMAP
  - DNS
- Ağ uygulaması geliştireceğiz.
  - soket API

Bilindik ağ uygulamaları

- e-mail
- web
- Metin iletisi
- Uzaktan erişim
- P2P dosya paylaşımı
- Çok-kullanıcılı ağ oyunları
- Saklanmış video yayınlama (YouTube, Hulu, Netflix)
- IP üzerinden sesli arama (e.g., Skype)
- Gerçek-zamanlı video konferans
- Sosyal ağlar
- Arama
- …

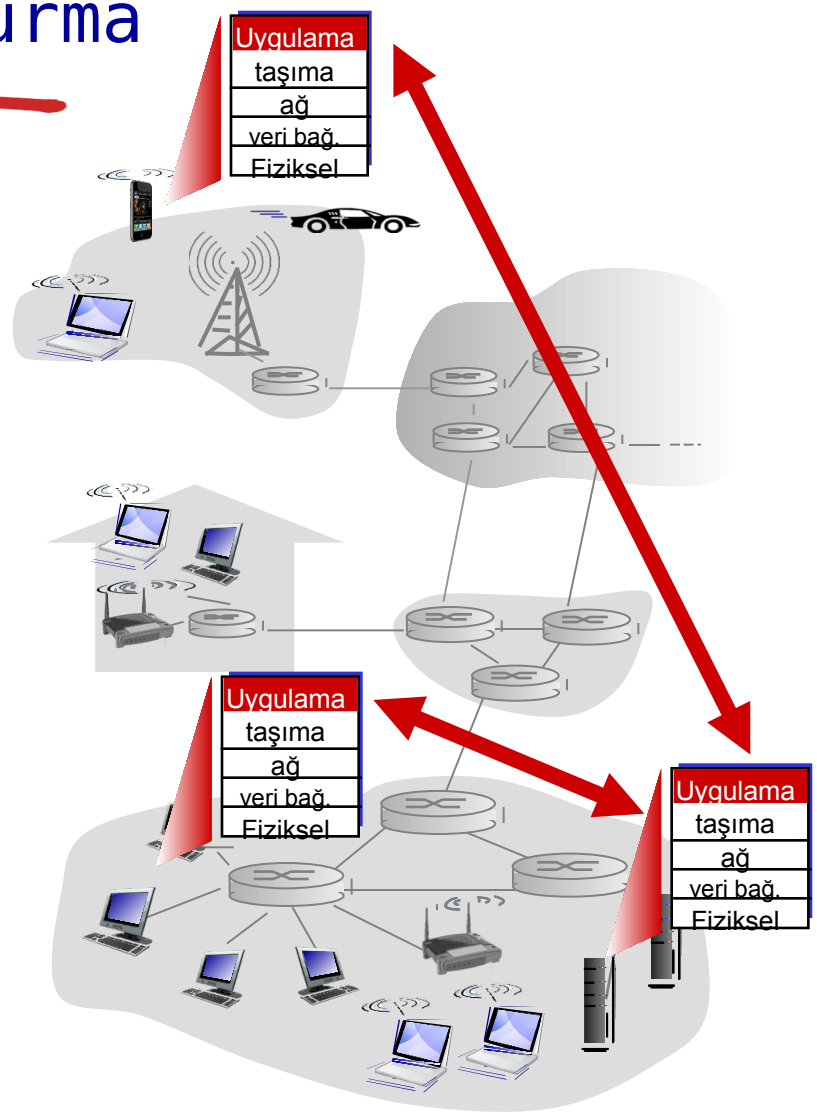
Bir ağ uygulaması oluşturma

Aağıdakileri yapan uygulamalar:

- (Farklı) Uç cihazlarda çalışır.
- Ağ üzerinden haberleşir.
- e.g., İstemci yazılımı ile haberleşen web sunucusu.

Ağ-çekirdeği cihazları için yazılım oluşturmaya gerek yoktur.

- Ağ-çekirdeği cihazları kullanıcı uygulamaları çalıştırmazlar.
- Uç cihazlar için uygulama yazılması hızlı uygulama oluşturulmasına ve uygulamanın hızlı şekilde yayılmasına olanak sağlar.

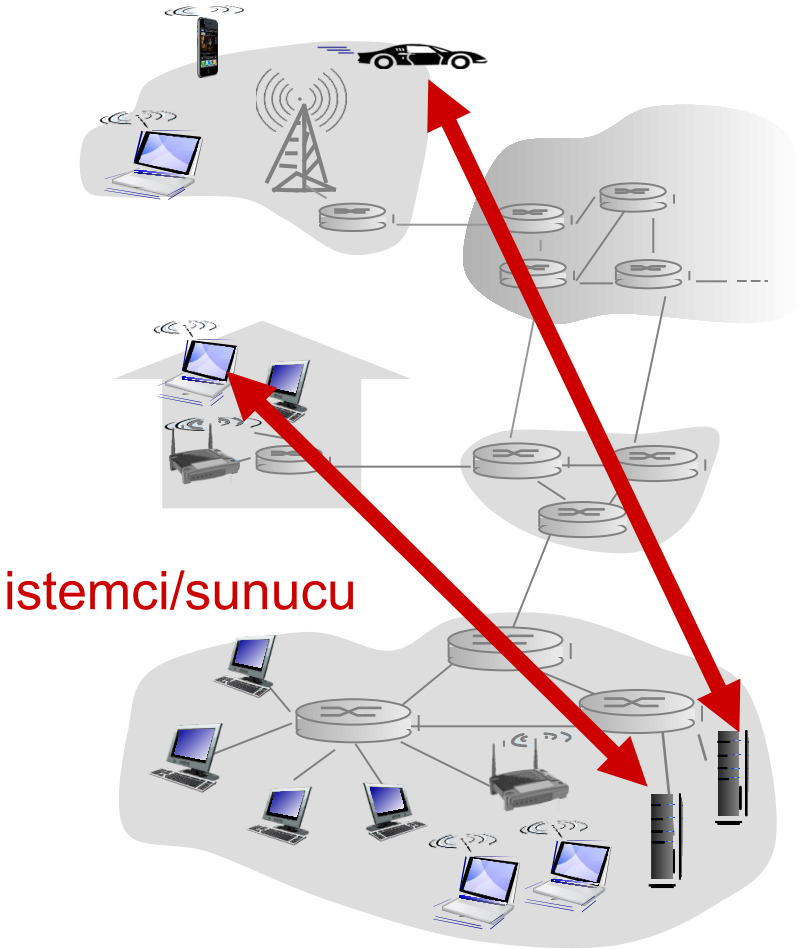


Uygulama Mimarileri

Uygulamanın olası yapıları:

- İstemci-sunucu
- Eşler arası: peer-to-peer (P2P)

İstemci-sunucu mimarisi



sunucu:

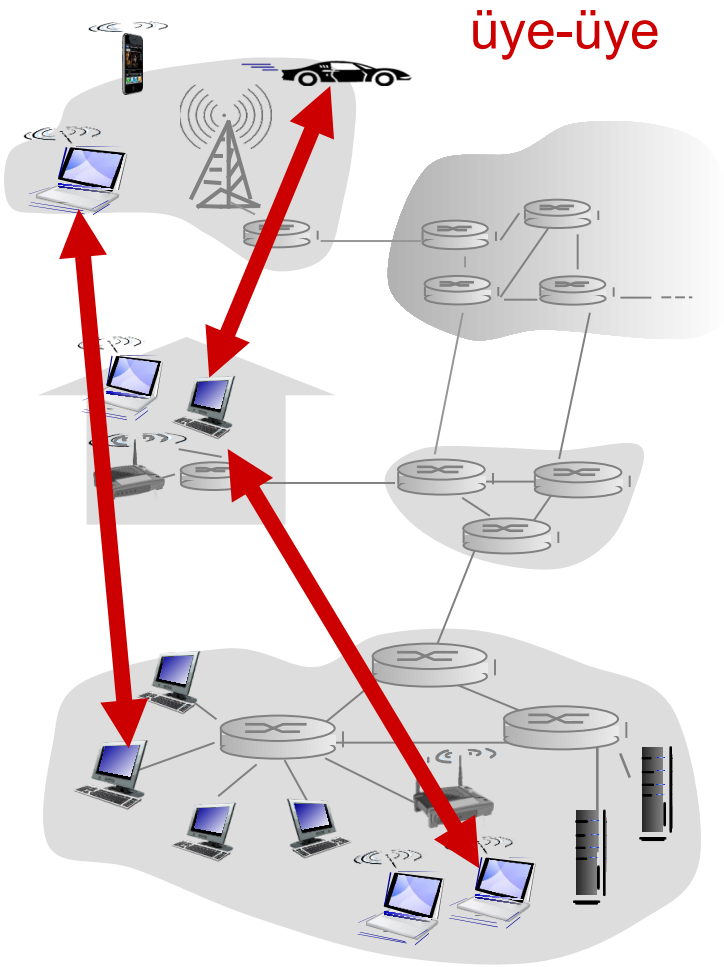
- Her zaman açık cihaz
- Sabit IP adresi
- Büyük trafik için veri merkezleri

İstemci:

- Sunucu ile iletişime geçer
- Aralıklı olarak bağlanabilir.
- Dinamik IP adresine sahip olabilir
- Birbirleri ile direk olarak iletişime geçmez.

P2P mimarisi

- Her zaman açık sunucu yoktur.
- Rastgele uç sistemler direk olarak haberleşir.
- Üyeler diğer üyelerden hizmet alır ve karşılığında diğer üyelere hizmet verir.
  - *Kendi ölçeklenebilirlik* – yeni üyeler geldikçe yeni hizmet isteği ve hizmet kapasitesi eklenir.
- Üyeler belli aralıklarla bağlanabilir veya IP adresini değiştirebilir.
  - Karmaşık yönetim mevcuttur.



İşlemler haberleşiyor

Işlem (İng: Process): *bir cihazda çalışan program*

- Aynı cihaz üzerinde iki farklı işlem, işlemler arası haberleşme ile haberleşirler.
  - İşletim sistemi sağlar.
- Farklı cihazlardaki işlemler mesaj alış-verişi ile haberleşirler.

istemciler, sunucular

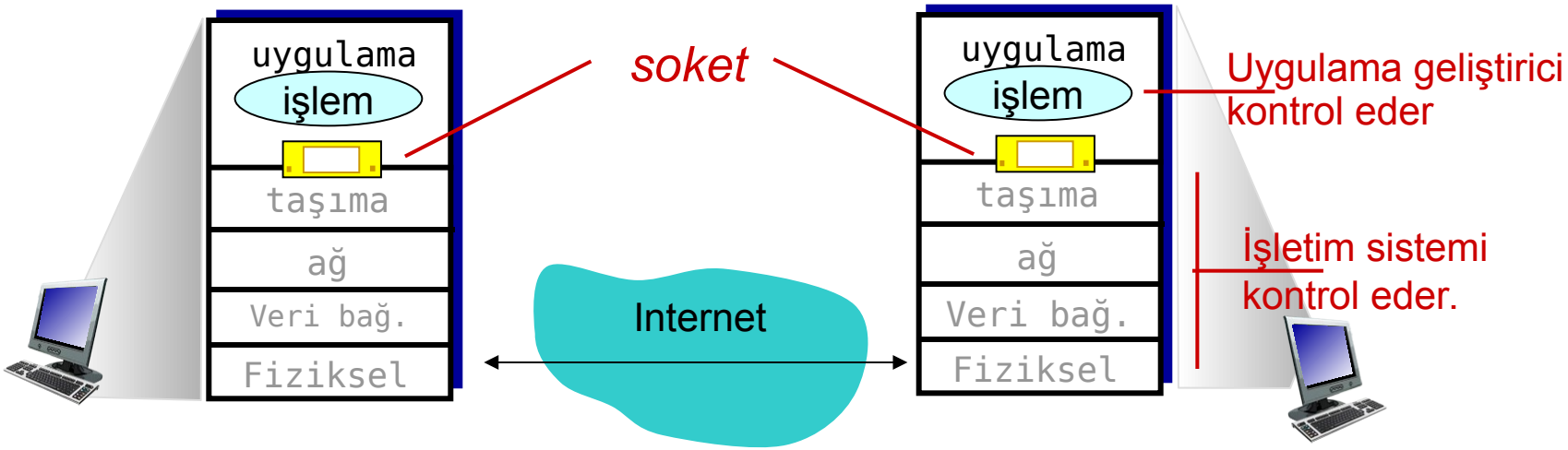
İstemci işlemi: Haberleşmeyi başlatan işlem

Sunucu işlemi: bağlantı kurulmasını bekleyen işlem

- Ayrıca: P2P mimarisi ile oluşturulmuş uygulamalar da istemci işlemi & sunucu işlemi bulundurur.

Soketler

- İşlemler mesajlarını soketlerinden gönderir/alır.
- Soketler kapıya benzer.
  - Gönderen işlem mesajını kapıya yollar.
  - Gönderen işlem, karşı taraftaki taşıma altyapısının gönderilen mesajı alıcı işleme ait sokete ulaştıracağını bilir.



Adresleme

- İşlem mesajları alabilmek için belirteç sahibi olmalıdır.
- Cihazlar 32-bit benzersiz IP adresine sahiptir.
- *S:* İşlemin çalıştığı cihazın IP adresi, işlemi belirlemek için yeterli midir?
  - *C:* hayır, birden fazla işlem aynı cihazda çalışabilir.
- *Belirteç* hem IP adresini hem de cihaz üzerindeki işlem ile alakalı port numarasını içerir.
- Örnek port numaraları:
  - HTTP sunucu: 80
  - Posta sunucu: 25
- gaia.cs.umass.edu web sunucusuna HTTP mesajı göndermek için:
  - IP adres: 128.119.245.12
  - port numarası: 80
- Detaylar daha sonra…

Uyg. katmanı protokolü şunları tanımlar

- Gönderilen mesaj tipleri,
  - e.g., istek, karşılık
- mesaj sözdizimi kuralları:
  - Mesajda hangi alanlar olmalı & alanlar nasıl ayrılmalı
- Mesaj anlamı
  - Alanlardaki bilgilerin anlamı
- Ne zamann ve nasıl mesaj işlemler mesaj gönderir & mesaja karşılık verir.

Açık protokoller:

- RFC ile tanımlanmıştır.
- Birlikte işlerliğe (İng: İnteroperability) izin verir
- e.g., HTTP, SMTP

özel protokoller:

- e.g., Skype

Bir uygulama taşıma hizmeti olarak neye ihtiyaç duyar?

Veri bütünlüğü

- bazı uygulamalar (e.g., dosya transfer, web işlemleri) %100 güvenilir veri transferi gerektirir.
- başka uygulamalar (e.g., audio) biraz kaybı tolere edebilir.

Zamanlama

- Bazı uygulamalar apps (e.g., Internet telefon, interaktif oyunlar) çok az gecikme olmasını bekler.

İş hacmi

- Bazı uygulamalar (e.g., multimedya) belli sürede en az iş hacmi garantisi bekler.
- diğerleri (“elastik uyg.”) elde edebildikleri iş hacmi ile idare eder

Güvenlik

- Şifreleme, veri bütünlüğü, …

Taşıma hizmeti ihtiyaçları: yaygın uygulamalar

uygulama	Veri kaybı	İş hacmi	Zaman hassas
dosya transfer	kayıp yok	elastik	Hayır
e-posta	kayıp yok	elastik	Hayır
Web dosyası	kayıp yok	elastik	Hayır
gerçek-zaman audio/video	kayıp-tolere	ses: 5kbps-1Mbps video:10kbps-5Mbps	evet, 100'ler ms
saklı ses/video	kayıp-tolere	Yukarıdaki ile aynı	evet, birkaç sn
interaktif oyunlar	kayıp-tolere	birkaç kbps gerekir	evet, 100'ler ms
metin mesajı	kayıp yok	elastik	evet ve hayır

İnternet taşıma protokoller hizmetleri

TCP hizmeti:

- Gönderici ve alıcı işlemler arası *güvenilir taşıma.*
- *Akış kontrolü:* gönderici aşırı trafik ile alıcıyı boğmaz.
- *Sıkışma kontrolü:* ağ aşırı yüklenildiğinde göndericiyi yavaşlatır.
- *Şunları sağlamaz:* zamanlama, minimum iş hacmi garantisi, güvenlik
- *bağlantı-odaklı:* istemci ve sunucu işlemleri arasında kurulum gerekli.

UDP hizmeti:

- Gönderici ve alıcı işlemler arasında *güvenilir olmayan* taşıma.
- *Şunları sağlamaz:* güvenilirlik, akış kontrolü, sıkışma kontrolü, zamanlama, minimum iş hacmi garantisi, güvenlik, veya bağlantı kurulumu.

S: Sizce neden UDP'ye ihtiyaç duyulmuş?

Internet uyg.: uygulama, taşıma protokolü

uygulama	uygulama katmanı protokolü	kullandığı taşıma protokolü
e-posta	SMTP [RFC 2821]	TCP
uzaktan erişim	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
dosya transfer	FTP [RFC 959]	TCP
çoklu-ortam yayımlama	HTTP (e.g., YouTube), RTP [RFC 1889]	TCP veya UDP
Internet telefon	SIP, RTP, özel (e.g., Skype)	TCP veya UDP

TCP'yi güvene alma

TCP & UDP

- Şifreleme yok
- Şifreler açık olarak sokete gönderilirse, internet üzerinde açık olarak ilerler.

SSL

- Şifrelenmiş TCP bağnatısı sağlar.
- Veri bütünlüğü
- Uç-nokta kimlik doğrulama

SSL uyg. katmanındadır.

- Uygulamalar SSL kütüphaneleri kullanarak TCP ile "konuşur".

SSL soket API

- Sokete açık olarak gönderilen şifreler Internet üzerinde şifrelenmiş olarak ilerler.
- Bölüm 7.

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

Web ve HTTP

Genel olarak:

- *web sayfası*, *nesnelerden* oluşur.
- Nesneler HTML dosyası, JPEG resim, Java aplet, ses dosyası … olabilir.
- Bir web sayfası *referans edilmiş birçok nesneyi* içeren *taban HTML-dosyası* içerir.
- Her nesne bir URL (İng: Uniform Resource Locater) ile erişilebilir.

www.someschool.edu/someDept/pic.gif

Cihaz adı

Yol adı

HTTP genel bakış

HTTP: hypertext transfer protocol

- Web'in uygulama katmanı protokolü
- istemci/sunucu modeli
  - *istemci:* (HTTP protokolü ile) Web nesnelerini isteyen, alan ve gösteren tarayıcı.
  - *sunucu:* Web sunucu istenilen nesneler (HTTP prokolü ile) gönderir.

HTTP genel bakış (devam)

TCP kullanır:

- Istemci sunucuya TCP bağlantısı başlatır (soket oluşturur), port 80.
- sunucu istemcinin TCP bağlantısını kabul eder.
- Tarayıcı (HTTP istemcisi) ile sunucu (HTTP sunucusu) arasında HTTP mesajı alışverişi (uygulama katmanı) yapılır.
- TCP bağlantısı kapatılır

HTTP "durumsuzdur"

- Sağlayıcı önceki istemci istekleri ile ilgili bilgi saklamaz.

sebep

Durum bilgisi saklayan prokoller karmaşıktır.

- geçmiş tarihçe (durum) tutulması gerekir
- Sağlayıcı veya istemci bir sorun yaaşrsa, ikisinin durum tarihçesi birbirine uymaz, iki farklı durumun birleştirilmesi gerekir.

HTTP bağlantıları

Kalıcı olmayan HTTP

- TCP bağlantısı üzerinden en fazla bir nesne gönderilebilir.
  - Daha sonra bağlantı sonlanır.
- Birden fazla nesne indirmek için birden fazla bağlantı gerekir.

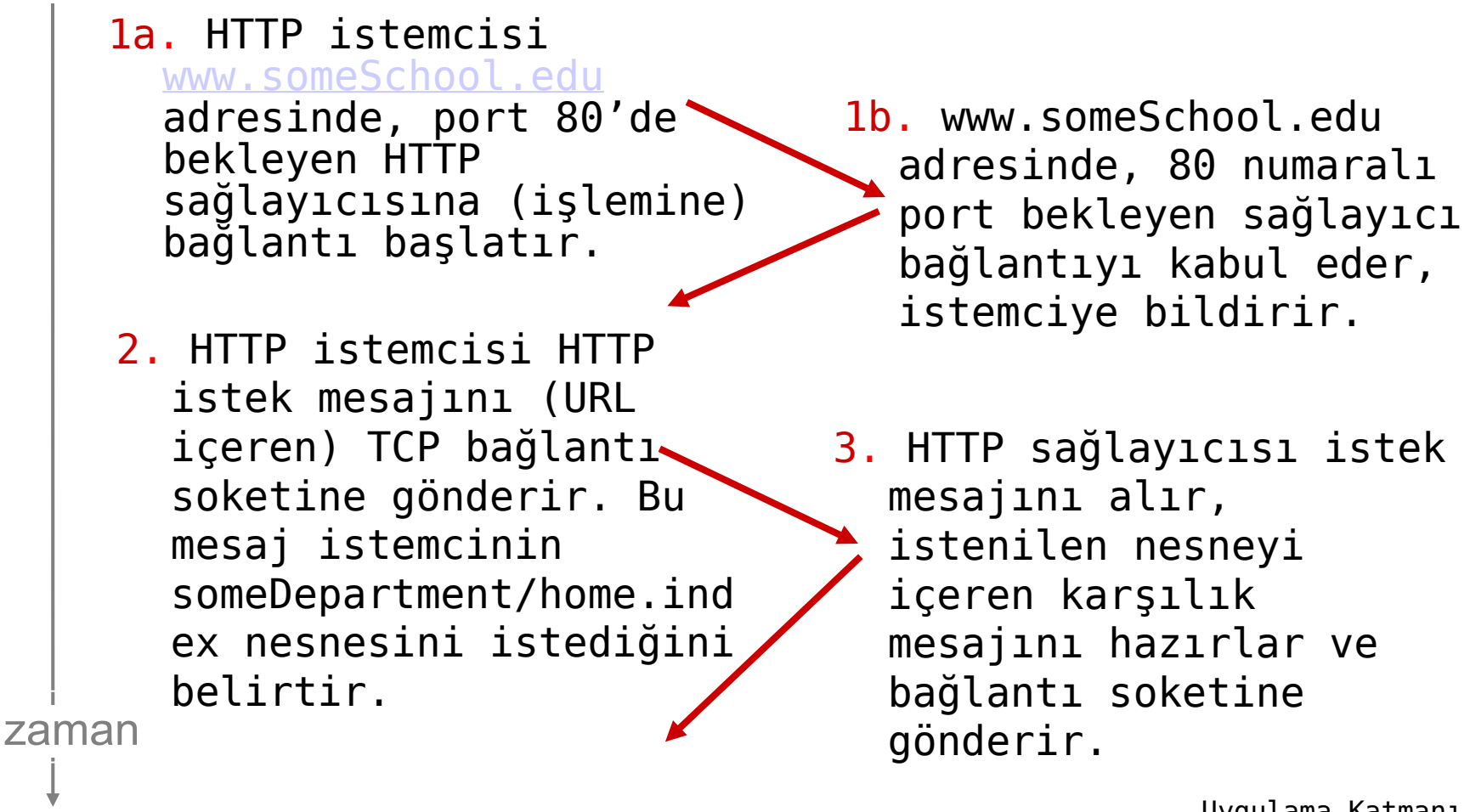
Kalıcı HTTP

- Aynı TCP bağlantısı üzerinden birden fazla nesne gönderilebilir.

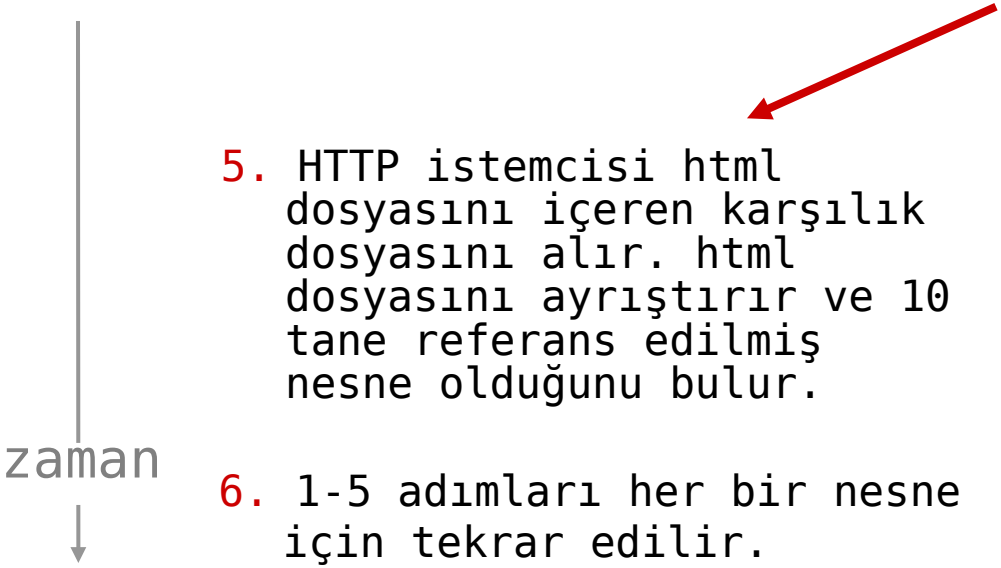
Kalıcı olmayan HTTP

Kullanıcı aşağıdaki URL girsin:
www.someSchool.edu/someDepartment/home.index

(metin ve, 10 jpeg resmine referans içersin)



Kalıcı olmayan HTTP (devam)



4. HTTP sağlayıcısı TCP bağlantısını kapatır.

5. HTTP istemcisi html dosyasını içeren karşılık dosyasını alır. html dosyasını ayrıştırır ve 10 tane referans edilmiş nesne olduğunu bulur.

6. 1-5 adımları her bir nesne için tekrar edilir.

Kalıcı olmayan HTTP: karşılık süresi

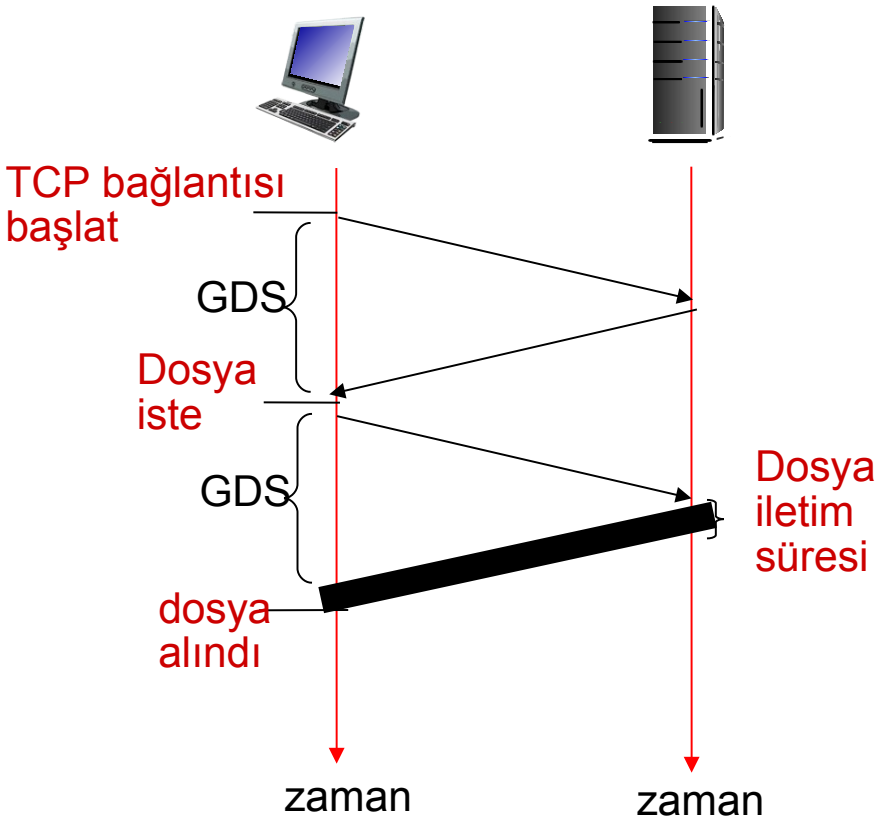
GDS - Gidiş-dönüş süresi (İng: RTT) küçük bir paketin istemciden sağlayıcı gidip gelme süresi

HTTP karşılık süresi:

- TCP bağlantısı için bir GDS.
- HTTP isteği ve HTTP karşılık mesajının ilk bölümlerinin ulaşması için bir GDS.
- Dosya iletim süresi

❖ Kalıcı olamayan HTTP karşılık süresi =

2GDS+ dosya iletim süresi



Kalıcı HTTP

Kalıcı olmayan HTTP sorunlar:

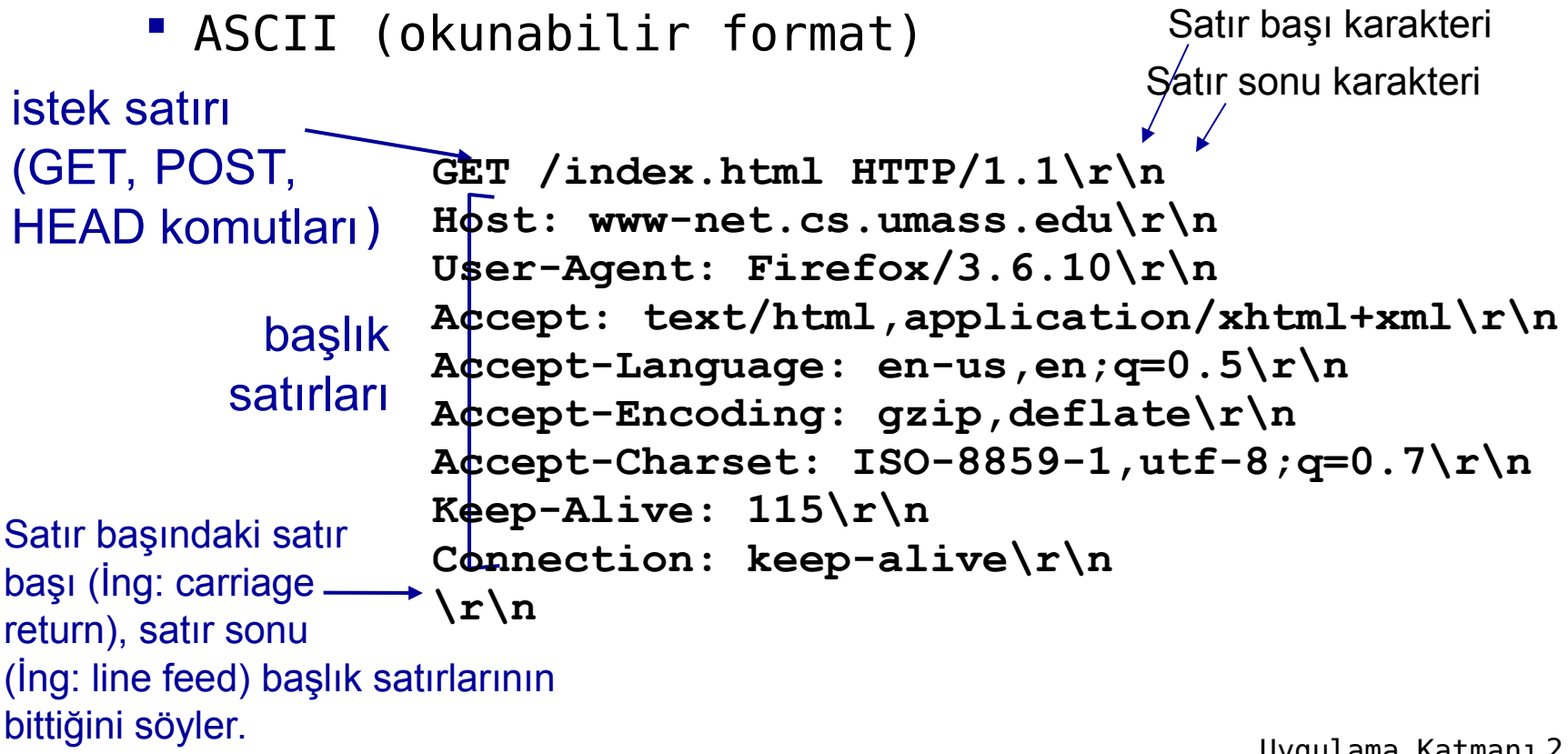
- Her bir nesne için 2 GDS gerekir.
- Her bir TCP bağlantısı için işletim sistemi zaman harcayacaktır.
- Tarayıcılar genellikle birden fazla paralel TCP bağlantısı açarak referans edilmiş nesnesleri hızlıca almaya çalışır.

kalıcı HTTP:

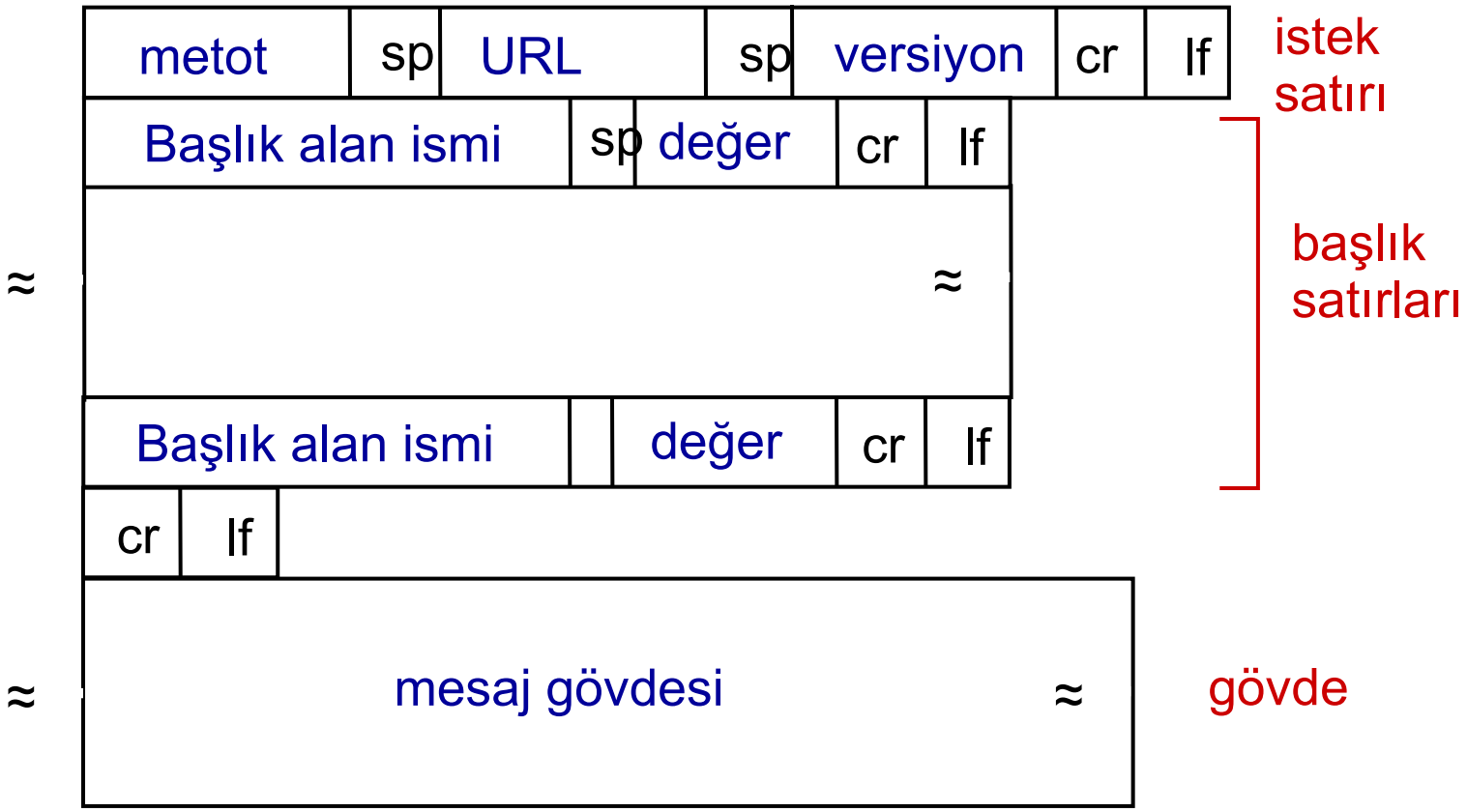
- Sağlayıcı karşılığı gönderdikten sonra bağlantıyı açık bırakır.
- İstemci ve sağlayıcı arasındaki takip eden mesajlar aynı bağlantı üzerinden gönderilir.
- İstemci referans edilmiş bir nesne ile karşılaşınca isteğini gönderir.
- Her referans edilmiş nesne için yaklaşık bir GDS süre geçer.

HTTP istek mesajı

- iki tip HTTP mesajı: *istek (İng: request), karşılık (İng: response).*
- HTTP istek mesajı:
  - ASCII (okunabilir format)



HTTP istek mesajı: genel format



Form girdisi yükleme

POST metodu:

- Web sayfaları genellikle form girdisi içerir.
- Girdi sağlayıcıya mesaj gövdesinde yüklenir.

URL metodu:

- GET metodu kullanır
- Girdi URL istek mesajının URL alanında yüklenir:

www.somesite.com/animalsearch?monkeys&banana

Metot tipleri

HTTP/1.0:

- GET
- POST
- HEAD
  - Sağlayıcı istenilen nesnenin cevabın içerisinde gönderilmemesini söyler.

HTTP/1.1:

- GET, POST, HEAD
- PUT
  - URL alanında belirtilen adrese mesaj gövdesinde bulunan dosyayı yükler.
- DELETE
  - URL alanında belirtilen dosyayı siler.

HTTP karşılık mesajı

durum satırı (protocol durum kodu durum ifadesi)

başlık satırları

veri, e.g., istenilen 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 karşılık durum kodları

- Durum kodu, sağlayıcıdan istemciye gönderilen karşılık mesajının ilk satırında bulunur:

200 OK

- Istek başarılı, istenilen nesne bu mesajın geri kalanında.

301 Moved Permanently

- Istenilen nesne taşınmış, dosyanın yeni yeri mesajın geri kalanında (Location:).

400 Bad Request

- Istek sağlayıcı tarafından anlaşılamadı.

404 Not Found

- İstenilen nesne sağlayıcıda yok.

505 HTTP Version Not Supported

HTTP denemesi (istemci)

1. Telnet ile bir Web sağlayıcısına bağlanın:

`telnet` gaia.cs.umass.edu 80

gaia.cs.umass.edu adresinde, port 80 (önceden tanımlı HTTP sağlayıcı port) TCP bağlantısı açılır. Gönderilen her metin gaia.cs.umass.edu 80 numaralı port gönderilir.

2. bir GET HTTP istek mesajı yaz:

```
GET /kurose_ross/interactive/index.php HTTP/1.1
Host: gaia.cs.umass.edu
```

bu mesajı yazarak (iki kez satır başı ile), bu minimal (ve bütün) GET isteğini HTTP sunucusuna gönderirsiniz.

3. HTTP sağlayıcısı tarafından gönderilen karşılık mesajını inceleyebilirsiniz.

Kullanıcı-sağlayıcı durumu: Çerezler (İng: Cookies)

Birçok Web sitesi çerez kullanır.

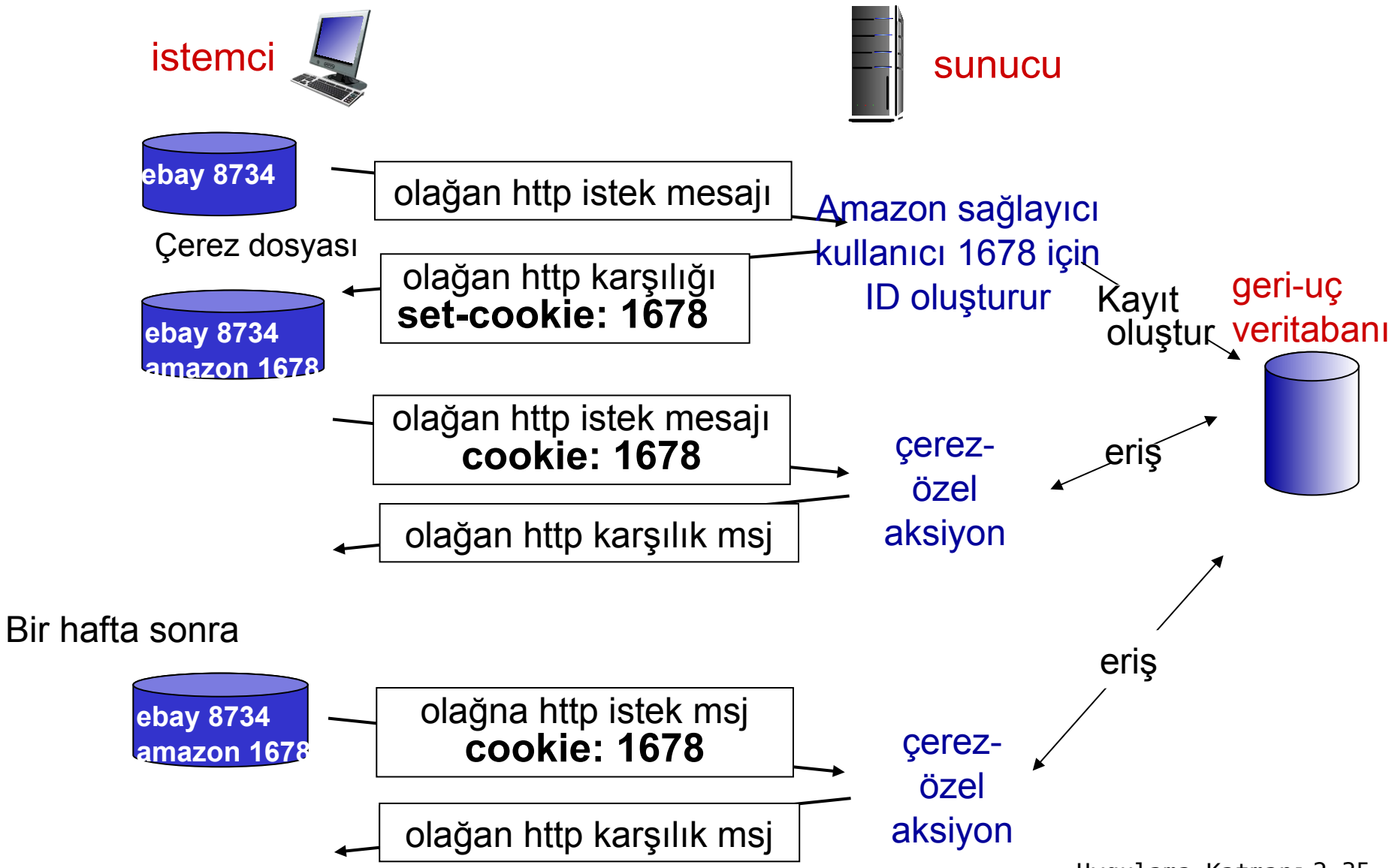
Çerezler dört bölümden oluşur:

1) HTTP karşılık mesajında çerez başlık satırı.
2) Bir sonraki HTTP istek mesajında çerez başlık satırı.
3) Kullanıcının cihazında kullanıcının tarayıcısı tarafından yönetilen çerez dosyası.
4) Web sitesinde geri-uç veritabanı.

örnek:

- Ayşe her zaman interete aynı PC üzerinden erişiyor.
- Bir e-ticaret sitesini ilk defa ziyaret ediyor.
- İlk HTTP isteği web sitesine ulaştığında, site aşağıdakileri oluşturur:
  - emsalsiz ID
  - Geri-uç veritabanında ID ile ilgili kayıt.

Çerezler: "durum" saklarlar



Çerezler (devam)

Çerezler ne için kullanılabilir:

- yetkilendirme
- Alışveriş sepetleri
- tavsiyeler
- Kullanıcı oturum durumu (Web e-posta)

Öte yandan

Çerezler ve mahremiyet:

- Çerezler sayesinde web siteleri sizin hakkınızda birçok bilgi toplayabilir.
- Sitelere isim ve e-posta gönderebilirsiniz.

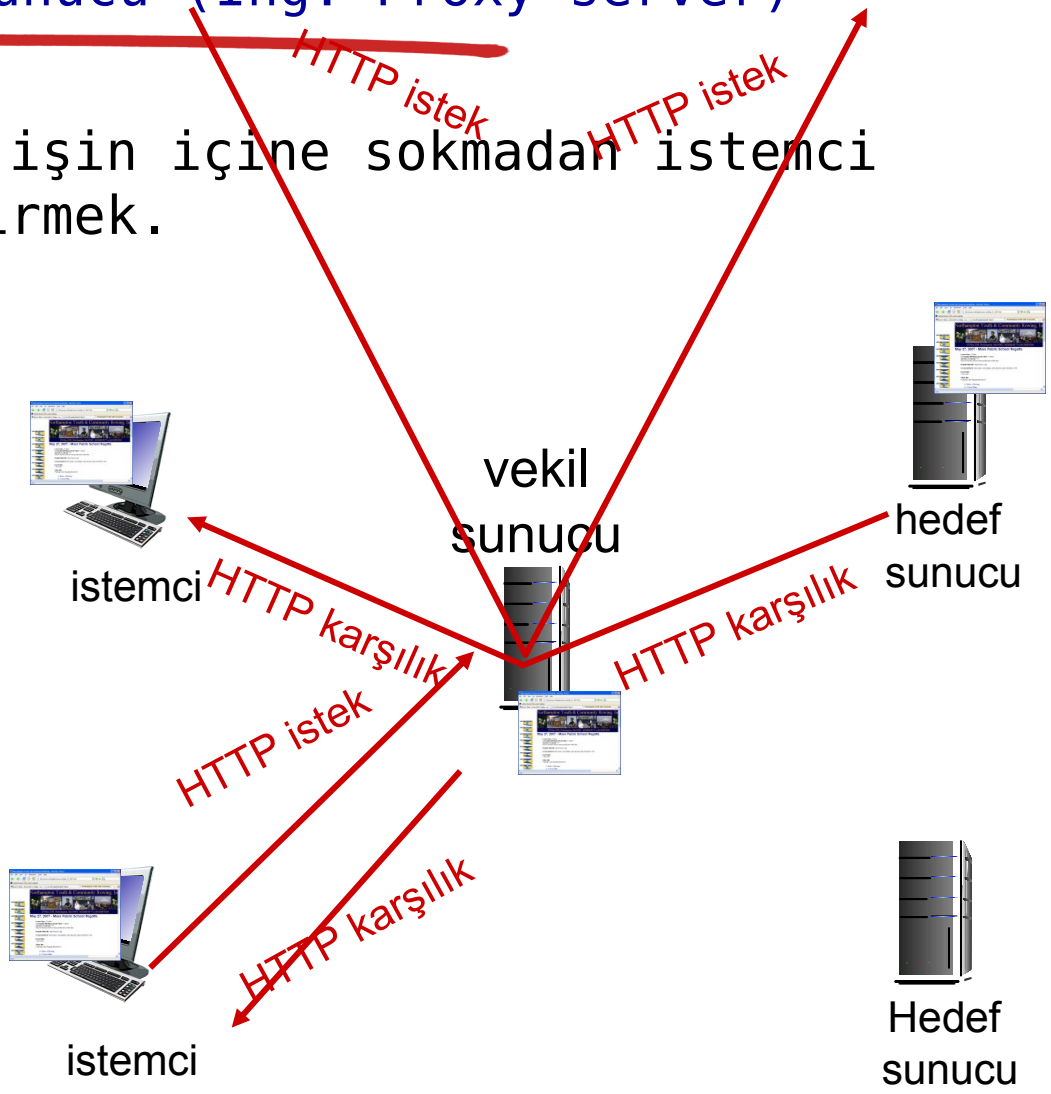
"durum" nasıl saklanır:

- Protokol uçnoktaları: çoklu işlem için gönderici/alıcı tarafında durum saklanır.
- Çerezler: http mesajı durum taşır.

Web önbelleği: vekil sunucu (İng: Proxy server)

hedef: Hedef sunucuyu işin içine sokmadan istemci isteğini gerçekleştirmek.

- Kullaıcı tarayıcıyı ayarlar: Önbellek üzerinden web erişimi
- Tarayıcı bütün HTTP isteklerini önbelleğe gönderir.
  - Nesne önbellekte ise: istemciye gönderilir.
  - Değilse, önbellek hedef sunucudan nesneyi alır, istemciye gönderir.



Web önbelleği

- Önbellek hem sunucu hem istemci olarak çalışır.
  - Başta istek yapan istemci için sunucu olarak.
  - Hedef sunucuya istemci olarak.
- Genellikle önbellek ISP tarafından oluşturulur. (universite, şirket, yerleşik ISP)

Önbelleklemenin yararları

- İstemci isteği için karşılık süresini azaltır.
- Kurumun erişim bağındaki trafiği azaltır.
- Internet üzerinde birçok önbellek bulunmaktadır. Bu sayede güçsüz içerik sağlayıcılarının efektif hizmet vermesi mümkün olur.

Önbellekleme örneği:

varsayımlar:

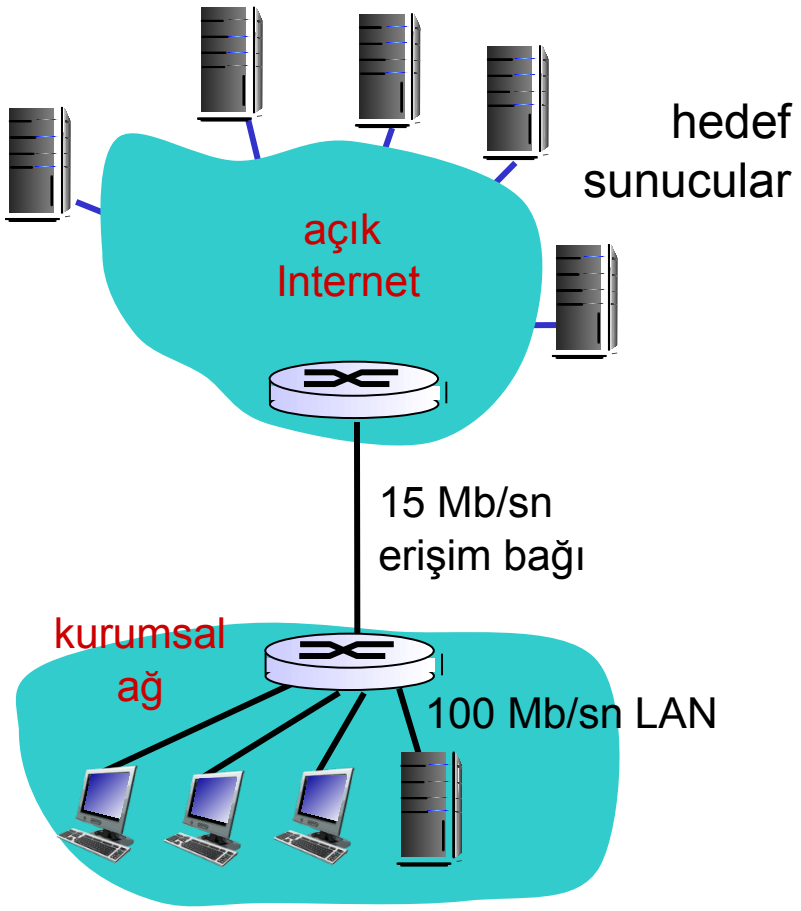
- Ort nesne büyüklüğü: 1 Mb.
- Tarayıcıdan hedef sunucuya ort istek sıklığı: 15/sn.
- Tarayıcılara ortalama veri hızı: 15 Mb/sn.
- Kurumsal yönlendiriden herhangi bir hedef sunucuya GDS: 2 sn.
- erişim bağ hızı: 15 Mb/sn.

sonuçlar:

problem!

- LAN kullanımı: 15%
- Erişim bağı kullanılı = 100%
- Toplam gecikme = Internet gecikmesi + erişim gecikmesi + LAN gecikmesi

  = 2 sn + dakikalar + milisaniyeler



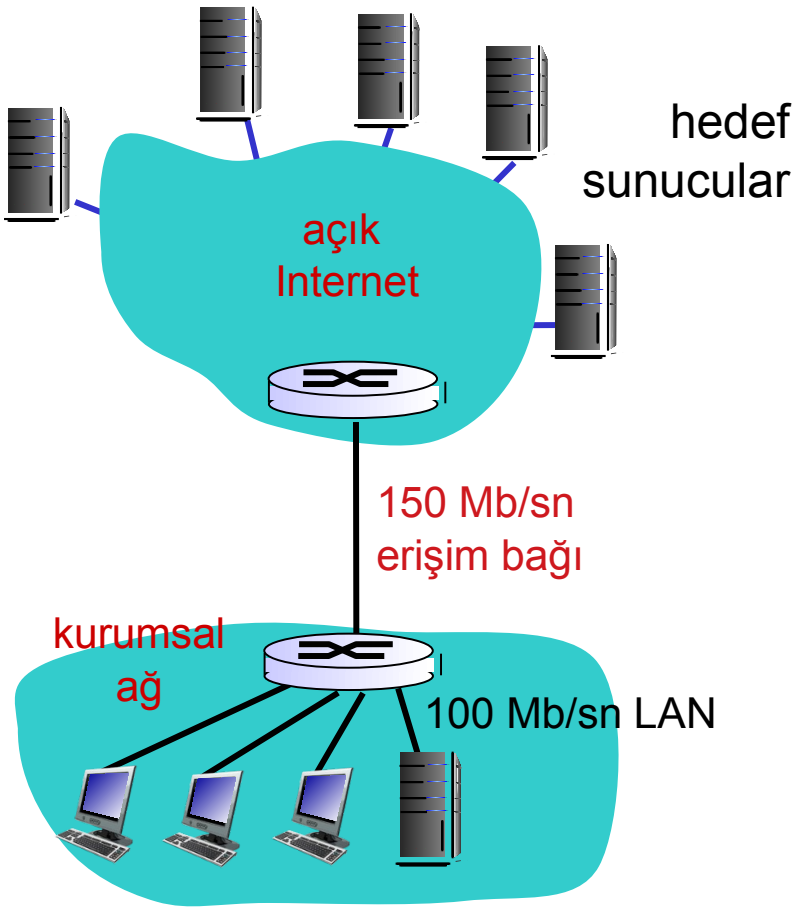
Önbellekleme örneği: Daha fazla erişim hızı ile

varsayımlar:

- Ort nesne büyüklüğü: 1 Mb.
- Tarayıcıdan hedef sunucuya ort istek sıklığı: 15/sn.
- Tarayıcılara ortalama veri hızı: 15 Mb/sn.
- Kurumsal yönlendiriden herhangi bir hedef sunucuya GDS: 2 sn.
- erişim bağ hızı: 150 Mb/sn.

sonuçlar:

- LAN kullanımı: 15%
- Erişim bağı kullanımı = %10
- Toplam gecikme = Internet gecikmesi + erişim gecikmesi + LAN gecikmesi
  = 2 sn + millisaniyeler + milisaniyeler



Oldukça pahalı bir çözüm!

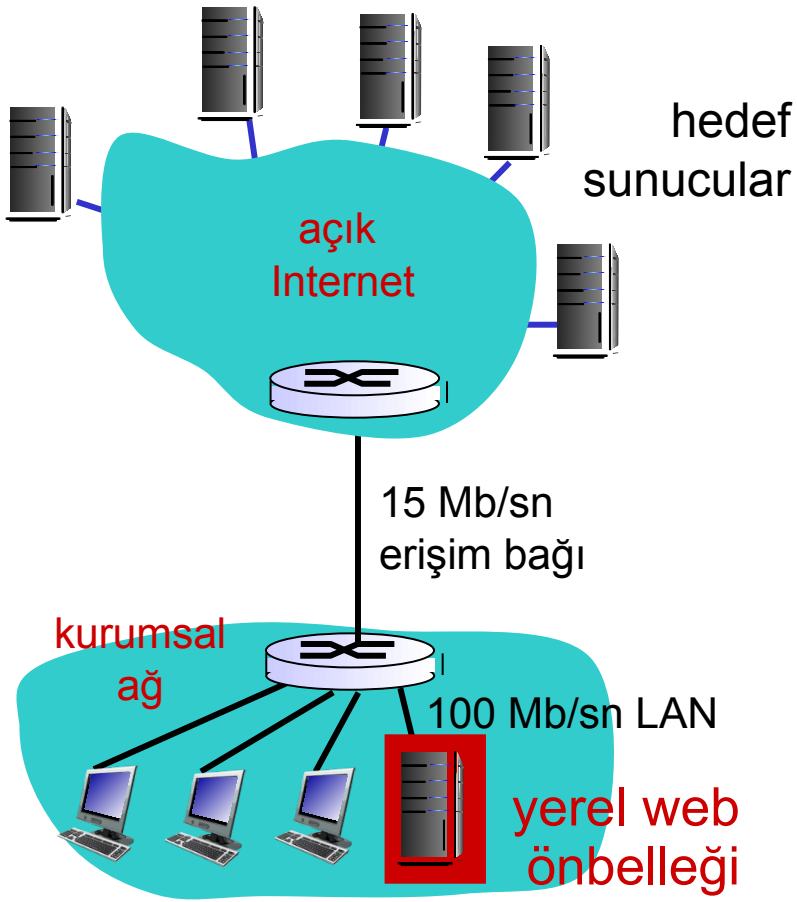
Önbellekleme örneği: Vekil sunucu kullanımı

varsayımlar:

- Ort nesne büyüklüğü: 1 Mb.
- Tarayıcıdan hedef sunucuya ort istek sıklığı: 15/sn.
- Tarayıcılara ortalama veri hızı: 15 Mb/sn.
- Kurumsal yönlendiriden herhangi bir hedef sunucuya GDS: 2 sn.
- erişim bağ hızı: 15 Mb/sn.

Vekil sunucu ile ilgili varsayımlar

- Vekil sunucu isabet oranı %40 olsun
  - İsteklerin %40'ı vekil sunucudan, geri kalan %60'ı hedef sunucudan gelsin.
- Erişim bağı kullanımı: 0.6*15 Mb/sn = 9 Mb/sn
  - Kullanım: 9/15 = 0.6 ==> %60.



Önbellekleme örneği: Vekil sunucu kullanımı

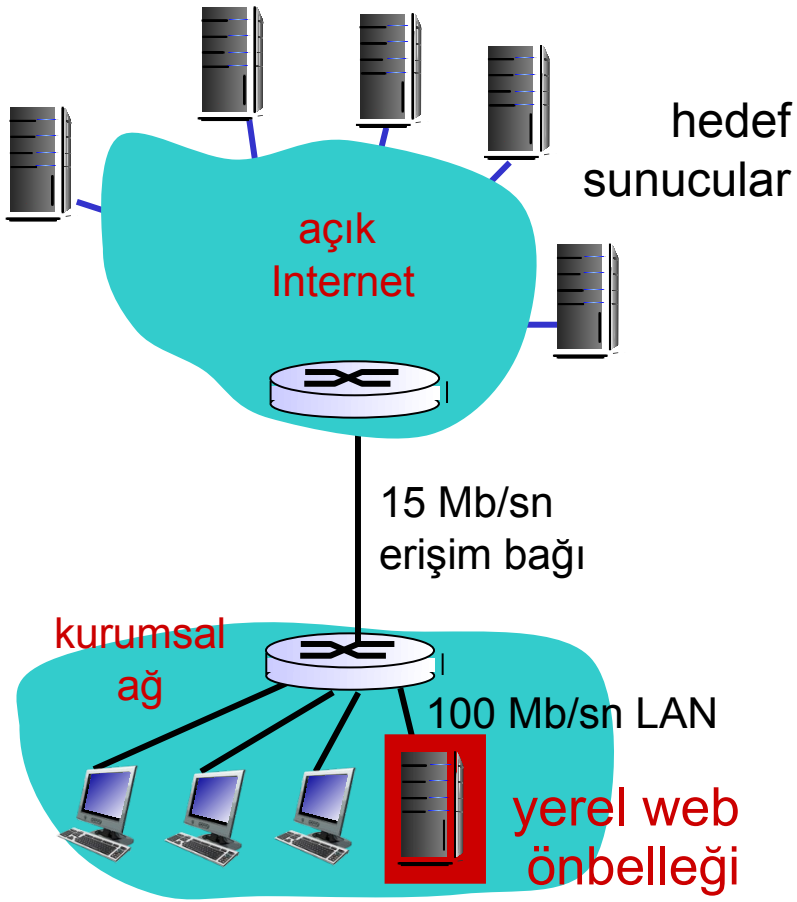
varsayımlar:

- Ort nesne büyüklüğü: 1 Mb.
- Tarayıcıdan hedef sunucuya ort istek sıklığı: 15/sn.
- Tarayıcılara ortalama veri hızı: 15 Mb/sn.
- Kurumsal yönlendiriden herhangi bir hedef sunucuya GDS: 2 sn.
- erişim bağ hızı: 15 Mb/sn.

- Vekil sunucu isabet oranı %40 olsun

Toplam gecikme:

- = 0.6 * (hedef sunucudan gecikme) + 0.4 * (önbellekten alım)
- = 0.6 * (2.01) + 0.4 * (milisaniyeler)
- = ~ 1.2 sn
- 10 kat hızlı internet almaktan hem daha hızlı hem de daha ucuz.



Koşullu GET

- *Hedef:* önbellekte güncel versiyon var ise nesneyi gönderme.
  - Nesne iletim gecikmesi yok.
  - Düşük bağ kullanımı.
- *Önbellek: HTTP isteği içinde önbelleklenmiş nesnenin tarihini belirt:*

  If-modified-since: <date>
- *sunucu:* önbelleklenmiş nesne güncel ise karşılık mesajında nesneyi gönderme:

  HTTP/1.0 304 Not Modified