

**HTTP – Hypertext Transfer Protocol**  
**és**

**HTTPS - Hypertext Transfer  
Protocol Secure**

# A HTTP protokoll

A **HTTP** (Hypertext Transfer Protocol) kifejezetten a Web számára megtervezett **hálózati protokoll**.

Vagyis a Webnek saját kommunikációs szabályai vannak, amelyeknek együtt kell működniük az Internet protokollokkal, hogy továbbíthatók legyenek a Web dokumentumok.

# HTTP – HyperText Transfer Protocol

- RFC 2068
- Web
  - Tim Berners-Lee, CERN
  - a második „killer application”
- Parancsorientált állapotkódokkal
- Speciális fejlécek
- Portok TCP 80, 8080
- Proxy
  - Kliens névében jár el
  - Főként a hatékony gyorsítótárazás miatt



# HTTP hypertext szállítási protokoll

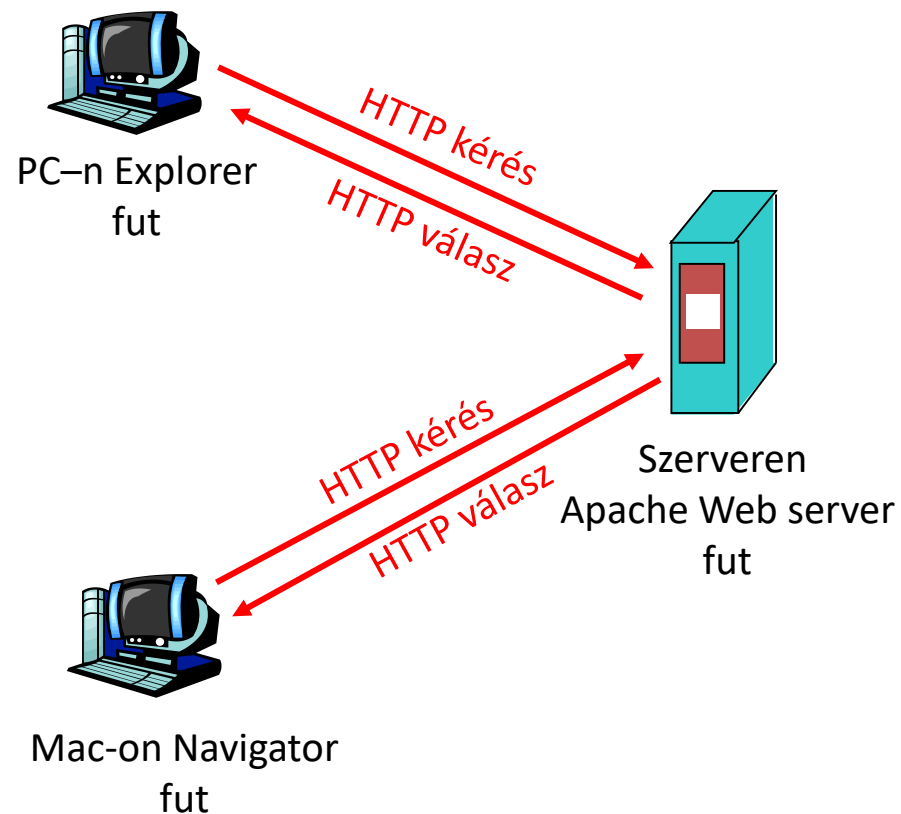
**HTTP:** Web alkalmazási réteg protokoll  
kliens/szerver modell

*kliens:* böngésző, mely kér, fogad,  
„megmutat” webes objektumokat

*szerver:* webszerver a kérésekre  
válaszol, objektumokat küld

HTTP 1.0: RFC 1945

HTTP 1.1: RFC 2068



# A HTTP kliens szerver kapcsolat

A HTTP vezérli a Web kiszolgáló és a kliens böngésző közötti együttműködést.

1. A kliens böngészővel Web-helyhez kapcsolódunk = a böngésző a Web-hely kiszolgálójától **kéri** a Web-oldal (HTML kódjeleket és ASCII szöveget tartalmazó fájl) megjelenítését.
2. A kliens inicializálja a TCP kapcsolatot (létrehoz egy socket-et) a szerverrel, a 80-as porton
3. Válaszul a Web kiszolgáló **elküldi** a kért fájl másolatát a böngészőnek. A böngésző a fájl megjelenítésekor a HTML kódjeleket olvasva formázza meg a szöveget.
4. Amikor grafikának megfelelő kódjelet (URL hivatkozást) olvas, kéri a kiszolgálót hogy küldje el a grafikát tartalmazó fájl másolatát.

# HTTP kapcsolatok

## Nem perszisztens HTTP

Legfeljebb egy objektumot küld egy TCP kapcsolaton.

### **HTTP/1.0**

Nem perszisztens HTTP-t használ

## Perszisztens HTTP

Összetett objektumok küldhetőek egyetlen TCP kapcsolaton keresztül a kliens és szerver között.

### **HTTP/1.1**

Perszisztens kapcsolatot használ alapértelmezésben

# Web és HTTP

## Definíciók

A weboldal **alap HTML-file**, amely **beágyazott (hivatkozott) objektumokat** tartalmaz

Az objektum lehet HTML file, JPEG kép, Java, applet, audio file,...

**Minden objektum egy URL-vel címezhető**

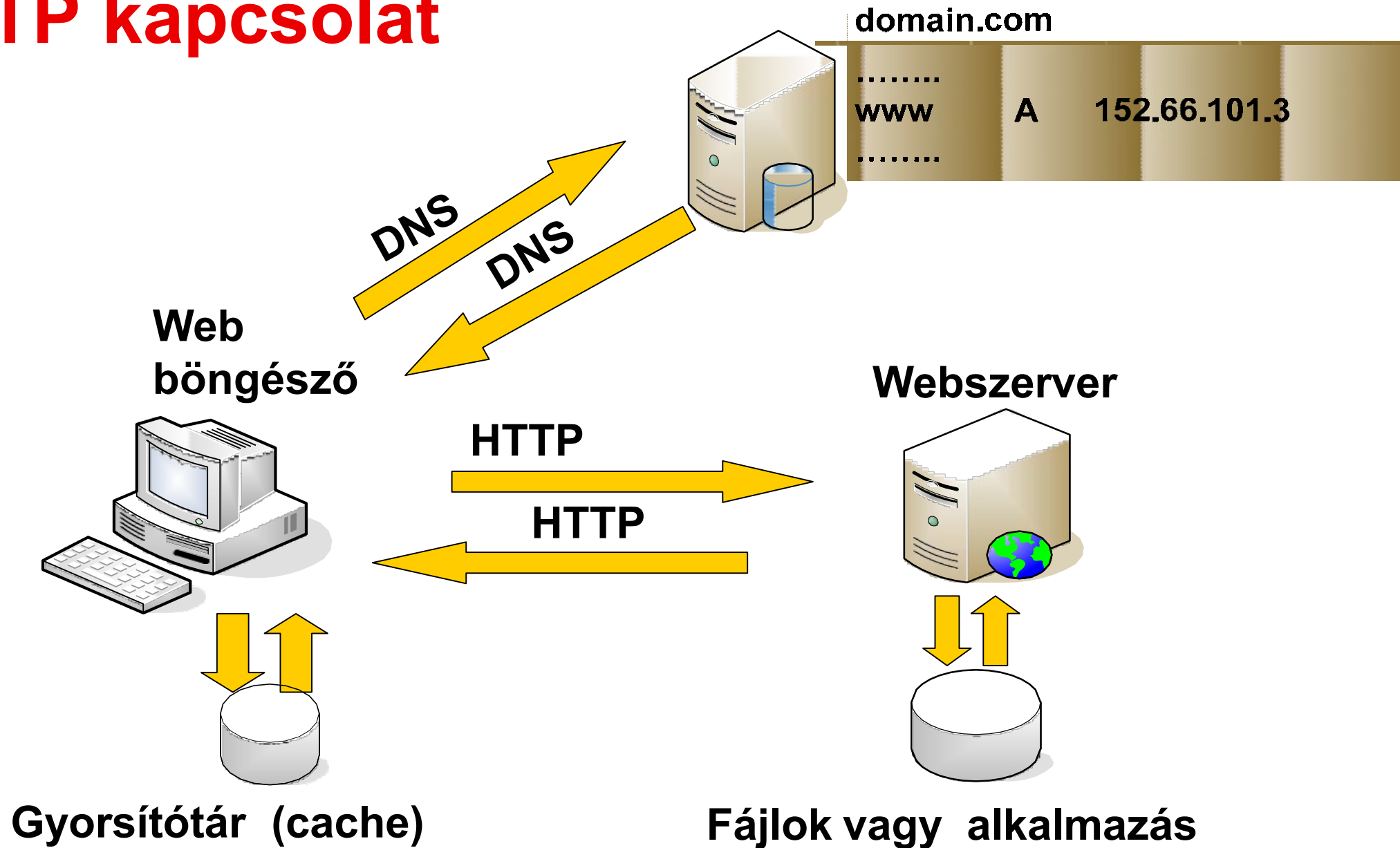
Példa URL-re:

`www.someschool.edu/someDept/pic.gif`

host név

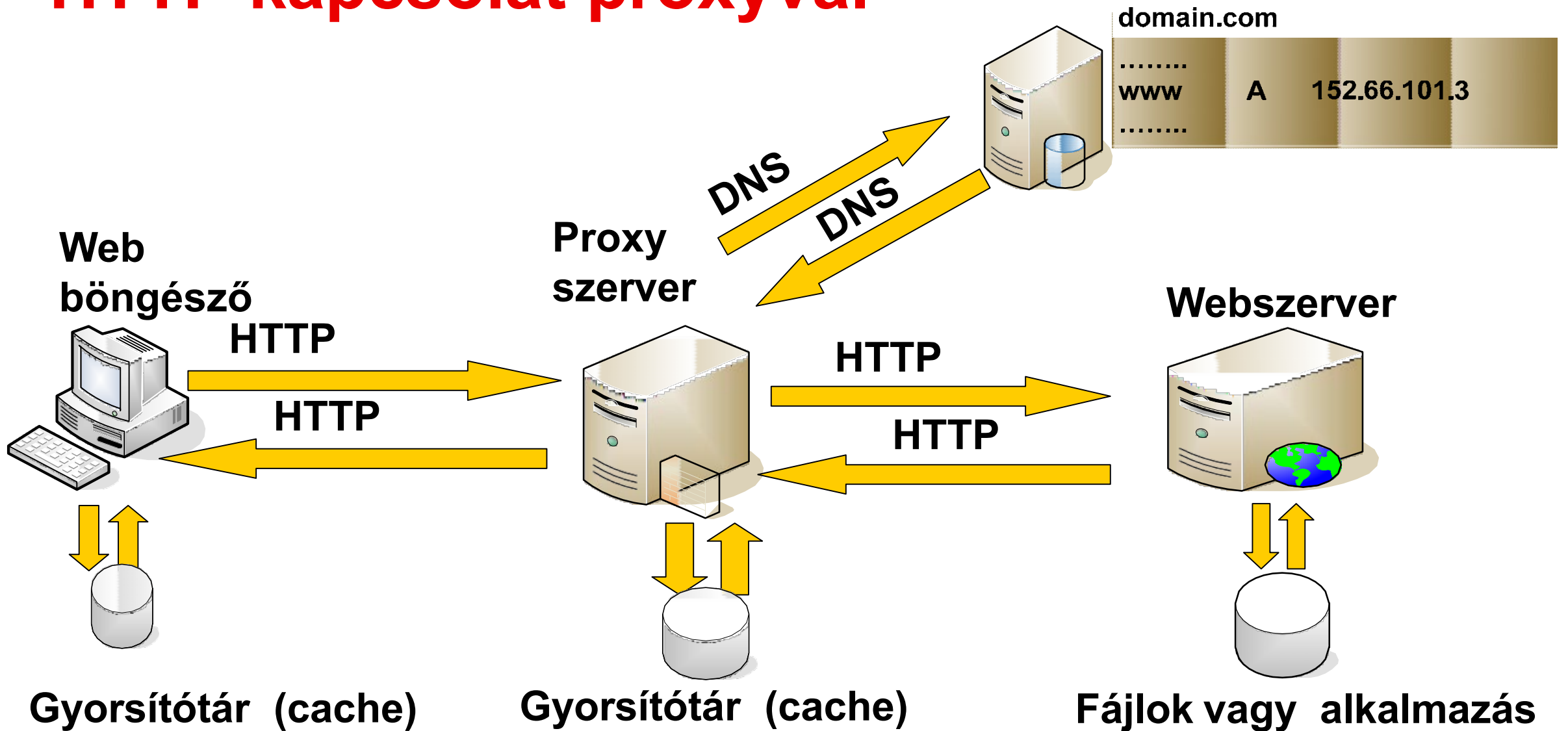
elérési útvonal

# HTTP kapcsolat



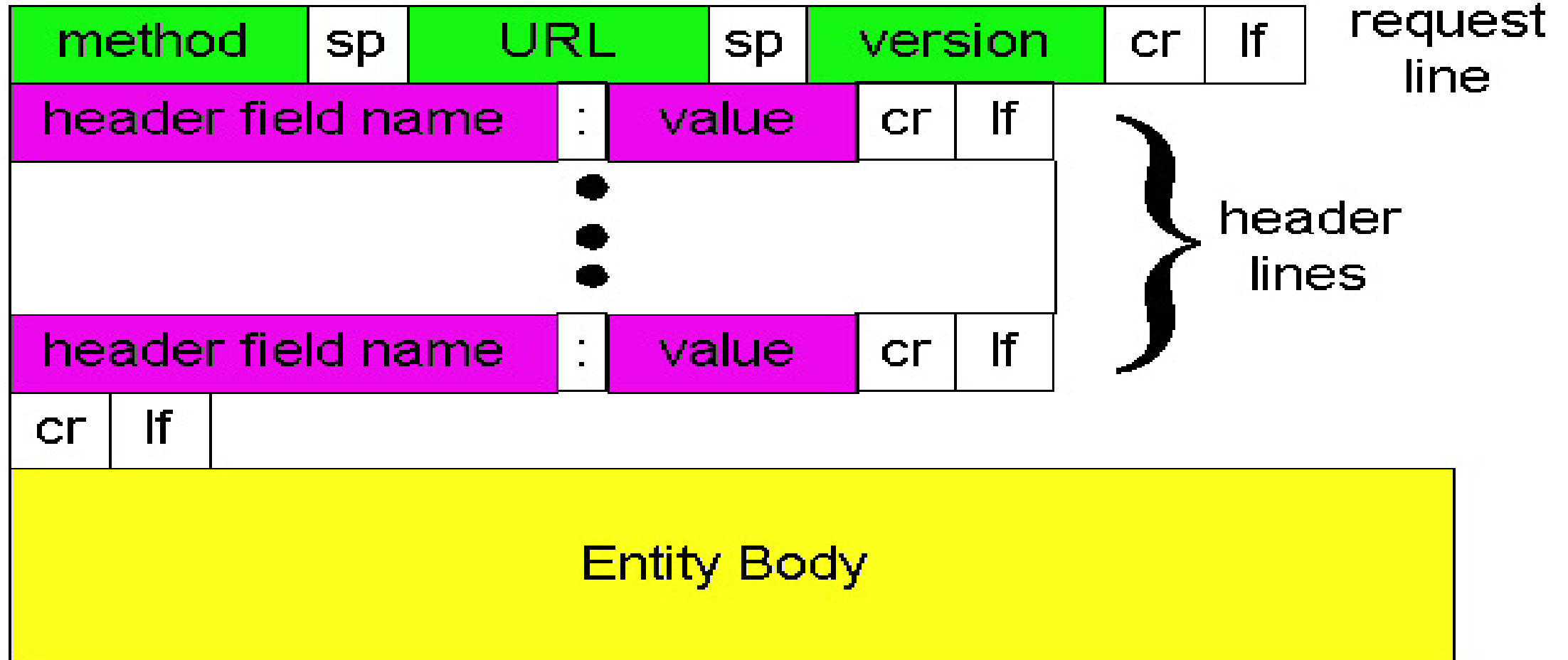


# HTTP kapcsolat proxyval



# Általános HTTP üzenet formátum

Kérés üzenet



Entity Body: POST-nál (Form küldés) van értelme!

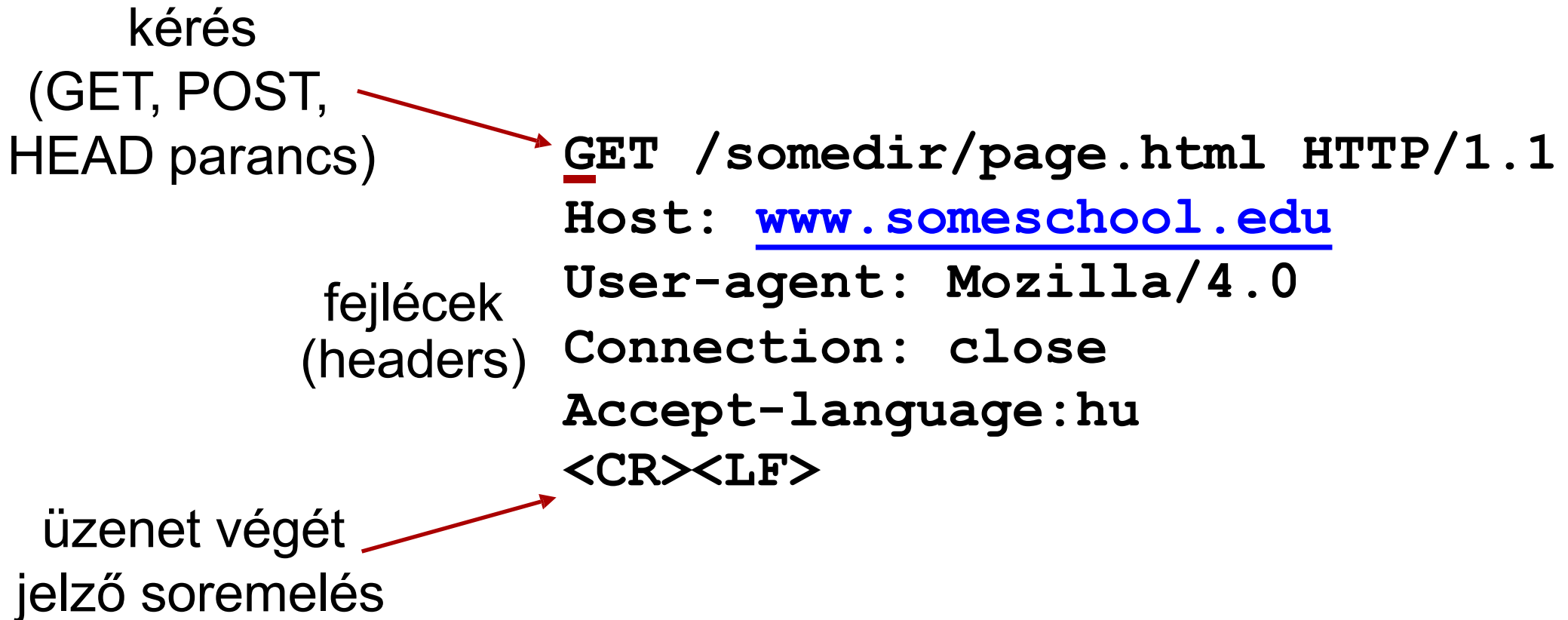
# HTTP kérés

---

kérés  
(GET, POST,  
HEAD parancs)

fejlécek  
(headers)

üzenet végét  
jelző soremelés



GET /somedir/page.html HTTP/1.1  
Host: [www.someschool.edu](http://www.someschool.edu)  
User-agent: Mozilla/4.0  
Connection: close  
Accept-language: hu  
<CR><LF>

The diagram illustrates the structure of an HTTP request. On the left, three labels are positioned: 'kérés (GET, POST, HEAD parancs)' with a red arrow pointing to the 'GET' method; 'fejlécek (headers)' with a red arrow pointing to the 'Host', 'User-agent', 'Connection', and 'Accept-language' lines; and 'üzenet végét jelző soremelés' with a red arrow pointing to the '<CR><LF>' line. The request itself is displayed on the right, with the 'GET' method underlined and the 'Host' value as a blue hyperlink.

# Gyakori HTTP metódus (parancsok)

## ■ GET <URL> HTTP/1.1

- adott URL tartalmának lekérése

## ■ HEAD

- mint a GET, de csak a metaadatokat adja vissza

## ■ POST

- a kliens ezzel tud adatokat küldeni a szervernek

## ■ PUT

- a POST-hoz hasonló, fájlfeltöltésre alkalmas

## ■ DELETE

- adott URL tartalmának törlése

Ezen kívül még 3 egyéb metódus létezik: TRACE, OPTIONS, CONNECT

# Kérés fejlécek

## Header lines

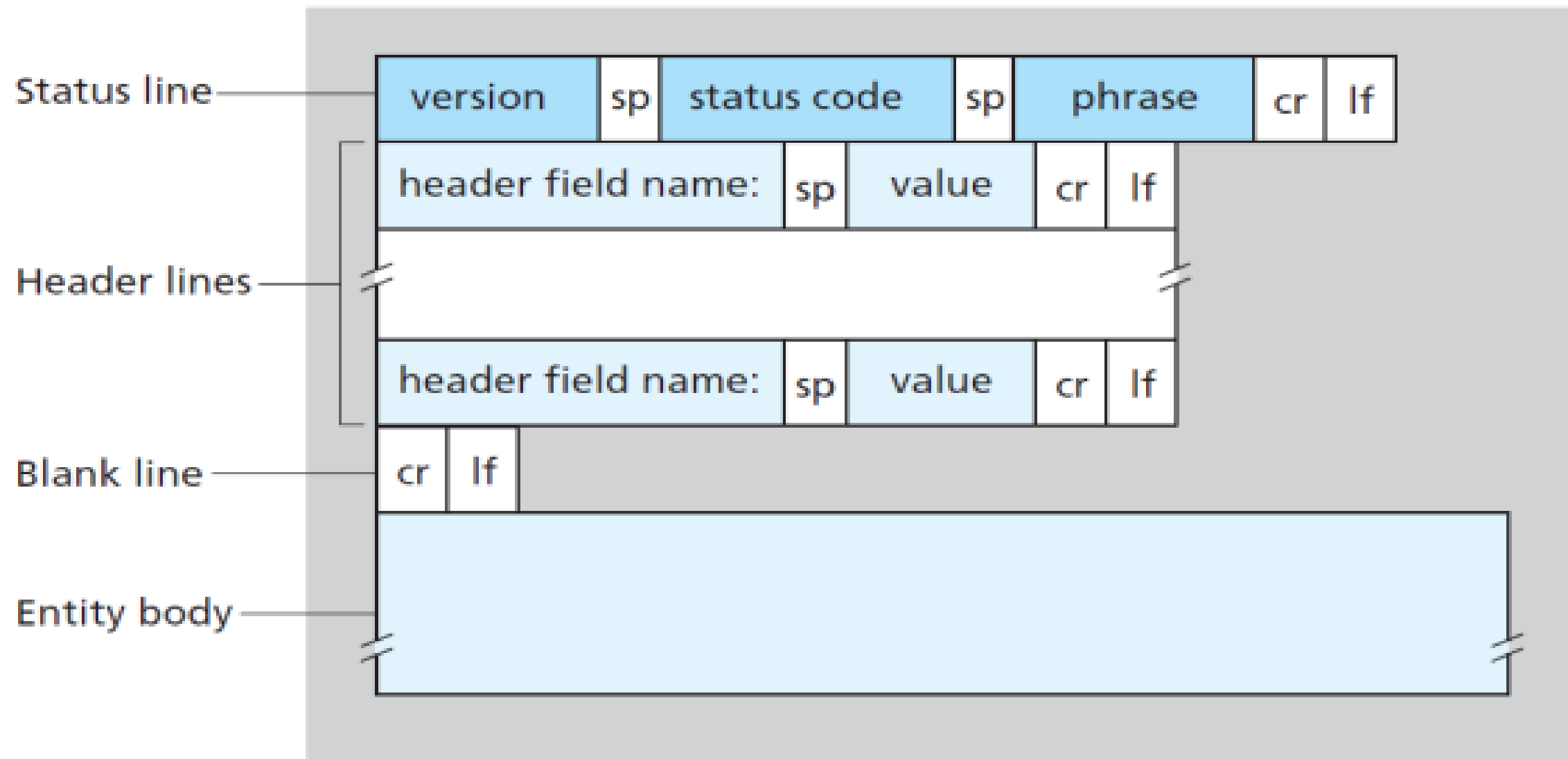
- **Host:** a host, amelyen a kért objektum elérhető
- **Connection:** a válasz megérkezését követően mi történjen a megnyitott kapcsolattal
- **User-agent:** ami indította a kérést (pl.: böngésző)
- **Accept-language:** milyen típusú objektumot kér a felhasználó

## Entity body

- HTTP GET kérés esetében ez a mező nem kerül kitöltésre, ugyanis csak egy objektum iránti kérést indítottunk, és a szerver minden szükséges információt megkap a fejlécből. Viszont egy HTTP **POST** kérés esetén adatokat szeretnénk továbbítani a szerver felé (pl.: egy űrlap kitöltését követően az űrlap mezőibe írt adatokat). Ekkor kap értelmet a törzs rész.

# HTTP válasz

Válasz üzenet



# HTTP válasz üzenet

állapotkód

HTTP/1.1 200 OK

fejlécek

Connection close

Date: Thu, 06 Aug 1998 12:00:15 GMT

Server: Apache/1.3.0 (Unix)

Last-Modified: Mon, 22 Jun 1998 .....

Content-Length: 6821

Content-Type: text/html

adat  
(pl. a kért  
HTML fájl)

data data data data data ...

# HTTP állapotkód csoportok

A HTTP-állapotkódokat a kliens kérésére adott válasz első sora tartalmazza. Az állapotkódok 3 számjegyből állnak, az első számjegy utal a tartalmukra, ez a számjegy 1-től 5-ig terjedhet.

Ez alapján a következő csoportjai vannak az állapotkódoknak:

- 1xx** - tájékoztató információk: A kérést megkapta a szerver, feldolgozás következik
- 2xx** - sikeres kérés: A kérést sikeresen megkapta, elfogadta, megértette a szerver.
- 3xx** - átirányítás: További tevékenységekre van szükség a kérés befejezéséhez.
- 4xx** - klienshiba: A kérés rossz szintaxisú vagy nem teljesíthető.
- 5xx** - szerverhiba: A szervernek nem sikerült egy helyes kérést végrehajtania.



# Gyakori HTTP állapotkódok

Kód	Jelentés	Leírás
200	OK	
201	Created	POST sikeres
202	Accepted	Kérés elfogadva
204	No content	Nincs semmi a kliensnek
400	Bad request	Hibás kérés
401	Unauthorized	Hitelesítés szükséges
403	Forbidden	Hozzáférés megtagadva
404	Not found	Nem található
500	Internal Server Error	Belső szerver hiba
503	Service Unavailable	Pillanatnyilag nem szolgálható ki

# Válasz fejléc

## Header lines

**Connection:** a válasz elküldését követően mi történjen a megnyitott kapcsolattal

**Date:** a válasz elküldésének időpontja

**Server:** milyen Web-szerver generálta a választ

**Last-Modified:** a fájl utolsó módosításának időpontja

**Content-Length:** hány bájtos a válasz üzenet

**Content-Type:** a válasz törzse (Entity body) milyen típusú (pl.: szöveg)

## Entity body

A kért objektum tényleges tartalma. <adatok>

# Sütik - Cookies



# Süтик

Bizonyára mindenki találkozott már olyan érdekes jelenséggel, hogy egy tetsző-leges web-oldal (nem első alkalommal történő) látogatása közben az oldalon korábban meglátogatott részei vagy az ahhoz kapcsolódó részei közvetlenül elénk vannak rakva.

*Például, egy webshop-ban szétnéztem az akciós laptopok között, és némelyiket részletesebben is megvizsgáltam (azaz rákattintottam a termékre). Másnap pedig visszalépve az oldalra egyből néhány laptop fogad....*

A web-oldalaknak a legtöbb esetben érdekük, hogy nyomon kövessék a felhasználót, milyen oldalakat látogatott. Ezekből az információkból viselkedési szokásokat lehet felállítani, és ezek alapján a felhasználó passzív igényeit tudjuk kielégíteni.

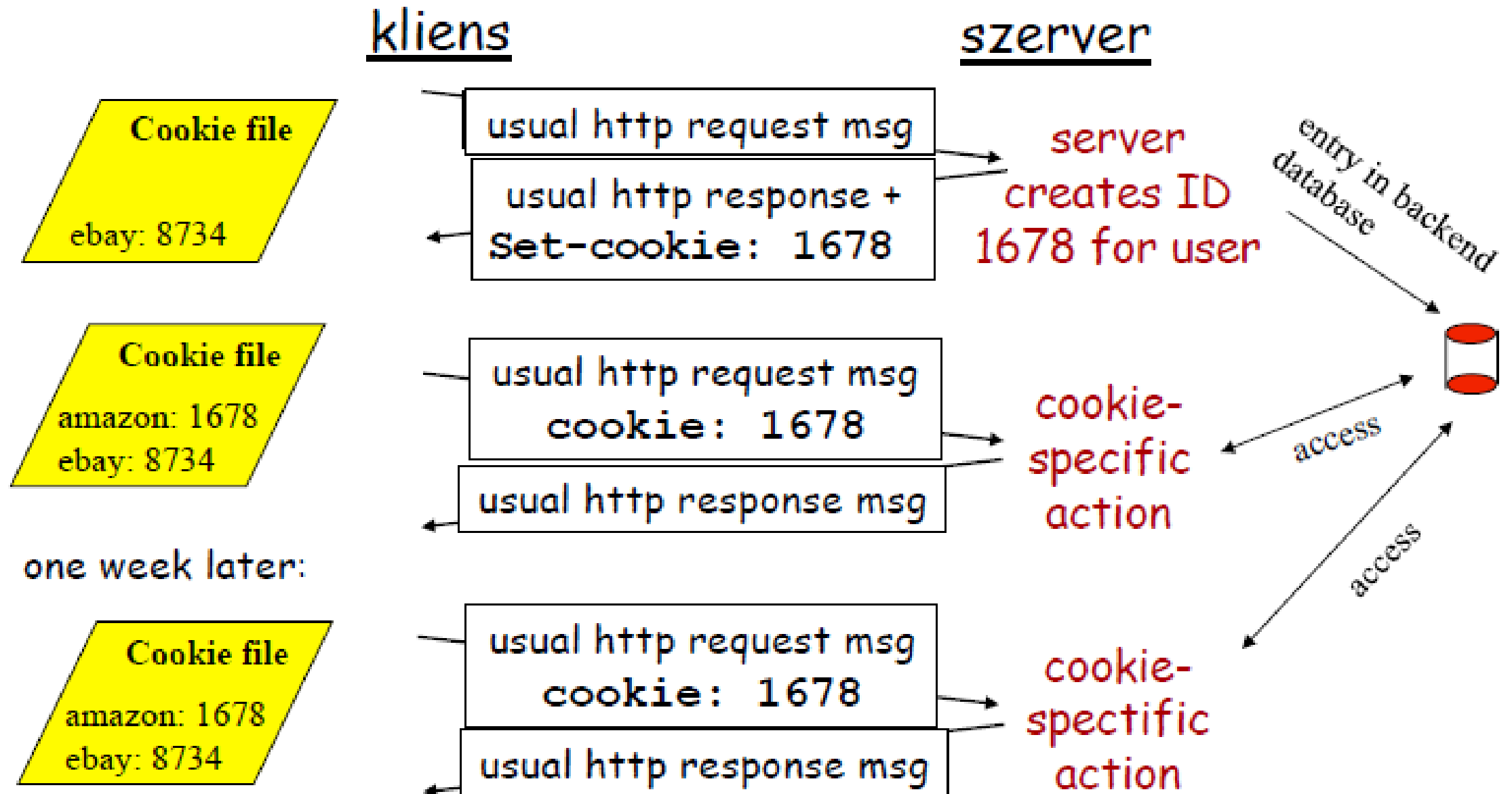
# Süti működése

**A süti technológia négy komponensből áll:**

1. egy süti-fejléc sor a HTTP válasz üzenetben
2. egy süti-fejléc sor a HTTP kérés üzenetben
3. egy süti-fájl a felhasználó rendszerében, amelyet a böngésző menedzsel
4. a back-end adatbázis a Web-oldalnál

1. A folyamat egy hagyományos HTTP kéréssel indul.
2. A kérést fogadja a web-szerver, majd készít egy egyedi azonosítót, és egy, az azonosítóval ellátott belépési pontot back-end adatbázisában.
3. A web-szerver reagál a kérésre egy válasszal, azonban ez kiegészül a korábbiakban látott válaszhoz képest egy Set-cookie: fejléccel.

# Sütik működése folytatás:



# Süтик működése folytatás:

4. A böngésző fogadja a választ, és látja a Set-cookie: fejléctet. **Ekkor a rendszerünkben található süti-fájlt kiegészíti a egy új bejegyzéssel,** mely tartalmazza a szerver címét, és a fejlécben található azonosítót.
5. Ezt követően, ha ugyanezen szerver felé indítunk kérést, a HTTP kérés **üzenetben megjelenik egy Cookie: fejléc az azonosítóval ellátva.** Ez a folyamat idézi elő, hogy a szerver kövesse a felhasználó aktivitását egy web-oldalon.

# Süti fejlécek

Sütik formátuma: egy egyszerű süti név értékpárból áll

Egy egyszerű süti így van beállítva:

**Set-Cookie: <cookie-name>=<cookie-value>**

A HTTP kérés már létező sütivel

```
GET /sample_page.html HTTP/2.0
Host: www.example.org
Cookie: yummy_cookie=choco; tasty_cookie=strawberry
```

HTTP válasz üzenet süti bejegyzéssel - Set-Cookie

```
HTTP/2.0 200 OK
Content-Type: text/html
Set-Cookie: yummy_cookie=choco
Set-Cookie: tasty_cookie=strawberry

[page content]
```



# HTTPS, SHTTP

## Titkosítás, azonosítás, adatbiztonság

HTTP protokollban nincs titkosítás ezért szükséges bevezetni a titkosított módosítását

### **HTTPS:**

Új réteg a HTTP alatt TLS (Transport Layer Security)

### **SHTTP:**

A HTTP kibővítése új utasításokkal melyek segítségével becsomagoljuk az eredeti HTTP csomagot.

# HTTPS

A **https** egy séma, amely biztonságos [http](#) kapcsolatot jelöl.

Szintaktikailag megegyezik a http sémával, amelyet a [HTTP](#) protokollnál használnak.

A HTTPS azt jelzi, hogy a HTTP és a TCP szintek közé titkosító/autentikáló [SSL](#) vagy [TLS](#) használó réteg került beiktatásra.

A HTTPS protokoll alapértelmezetten 443 portot használja (A titkosítatlan HTTP rendszerint a 80-as TCP portot használja.)

**Ahhoz, hogy egy webserver https kéréseket tudjon fogadni, az adminisztrátornak tanúsítványt (angolul public key certificate) kell készítenie.**

# HTTPS

## Kétkulcsos titkosítás

A https alapja a kétkulcsos titkosítás. A kulcspár két összetartozó, nagyon nagy (több száz jegyű) szám. A titkosítandó szöveget az egyik szám segítségével, egy nyilvános eljárással rejtjelezzük. Az így kapott üzenet ugyanazzal az eljárással, de a kulcspár másik tagjának segítségével fejthető csak vissza.

Ha a kulcspár egyik tagját titokban tartjuk, a másik kulcsot nyilvánosságra hozzuk, és valaki egy üzenetet a nyilvános kulcsunkkal rejtjelez, akkor azt csak mi fogjuk tudni elolvasni. **A titkosítás mindig a nyilvános kulccsal történik.**

# HTTPS

## Tanúsítóhely

**Ha egy webszerver nyilvános kulcsát le tudjuk kérdezni, és azzal titkosítunk, akkor az üzenetünket csak a webszerver fogja tudni elolvasni.** Az üzenet titkosságával tehát már nincs baj, abban viszont még nem lehetünk biztosak, hogy a titkosított üzenetünk ahhoz jutott, akinek szántuk, hiszen titkosító kulcspárt bárki tud csinálni.

A hitelesség biztosításának egyik módja a tanúsítvány. Ez egy igazolás arról, hogy a webszerver nyilvános kulcsa hiteles. Az igazolás kiadója egy tanúsítóhely (Certificate Authority), az igazolás hitelességét az ő digitális aláírása biztosítja, valamint az a tény, hogy a tanúsítóhelyet a böngésző írói felvették a böngészőben a hiteles tanúsítóhelyek listájába.

A tanúsítóhelynek tehát hasonló szerepe van a digitális hitelesítésben, mint a közjegyzőnek a papír alapúban.