Hálózati alkalmazások

Számítógép-hálózatok

Dr. Lencse Gábor
egyetemi tanár
Széchenyi István Egyetem, Távközlési Tanszék
lencse@sze.hu

Tartalom

- Bevezető: emlékeztető a portszámokról
- Domain Name System (DNS)
- Távoli elérési protokollok: TELNET, SSH, SCP
- Levelező protokollok: SMTP, POP3, IMAP4, POP3S, IMAP4S
- Fájl átviteli protokoll: FTP
- Rövid HTML ismertető
- Web hozzáférési protokollok: HTTP, HTTPS

BEVEZETÉS

Emlékeztető: portszámok kiosztása

- **System Ports** (Well Known Ports) rendszer portok („jól ismert” portok): 0-1023
  - Alapvető, általánosan használt hálózati szolgáltatások
  - Privilegizált szerver programok (root jog) szolgáltathatnak
- **User Ports** (Registered Ports) felhasználói portok (regisztrált portok): 1024-49151
  - Egyéb szolgáltatások számára (bárki igényelhet az IANA-tól)
  - Nem kell privilegizált módban futtatni a szerver programot
- **Dynamic/Private/Ephemeral Ports** dinamikusan kiosztott portok: 49152-65535
  - Ebből a tartományból az operációs rendszer oszt ki (forrás) portokat a kommunikációhoz a programoknak.

Kiosztás: http://www.iana.org/assignments/port-numbers

Nem mind arany, ami fénylik

- Nem minden operációs rendszer követi az RFC 6335 rendelkezéseit
  - Lehet, hogy pl. már 32768-tól osztanak ki dinamikus portokat
- NAT esetén már 1024-től osztanak ki forrás portszámokat
- De a TCP kapcsolat felépítésének iránya alapján azonosítható, hogy melyik a cél port!
- Viszont: tűzfalat nem alapozhatunk csupán portszámra!

Néhány alapvető alkalmazás

port	alk. röv.	hálózati alkalmazás neve
20	FTP	File Transfer Protocol, data
21	FTP	File Transfer Protocol, control
22	SSH	Secure SHell
23	Telnet	Telnet
25	SMTP	Simple Mail Transfer Protocol
53	DNS	Domain Name System
80	HTTP	HyperText Transfer Protocol
110	POP3	Post Office Protocol – Version 3
143	IMAP4	Internet Message Access Protocol – Version 4
443	HTTPS	HTTP over TLS/SSL
993	IMAPS	IMAP4 over TLS/SSL
995	POP3S	POP3 over TLS/SSL

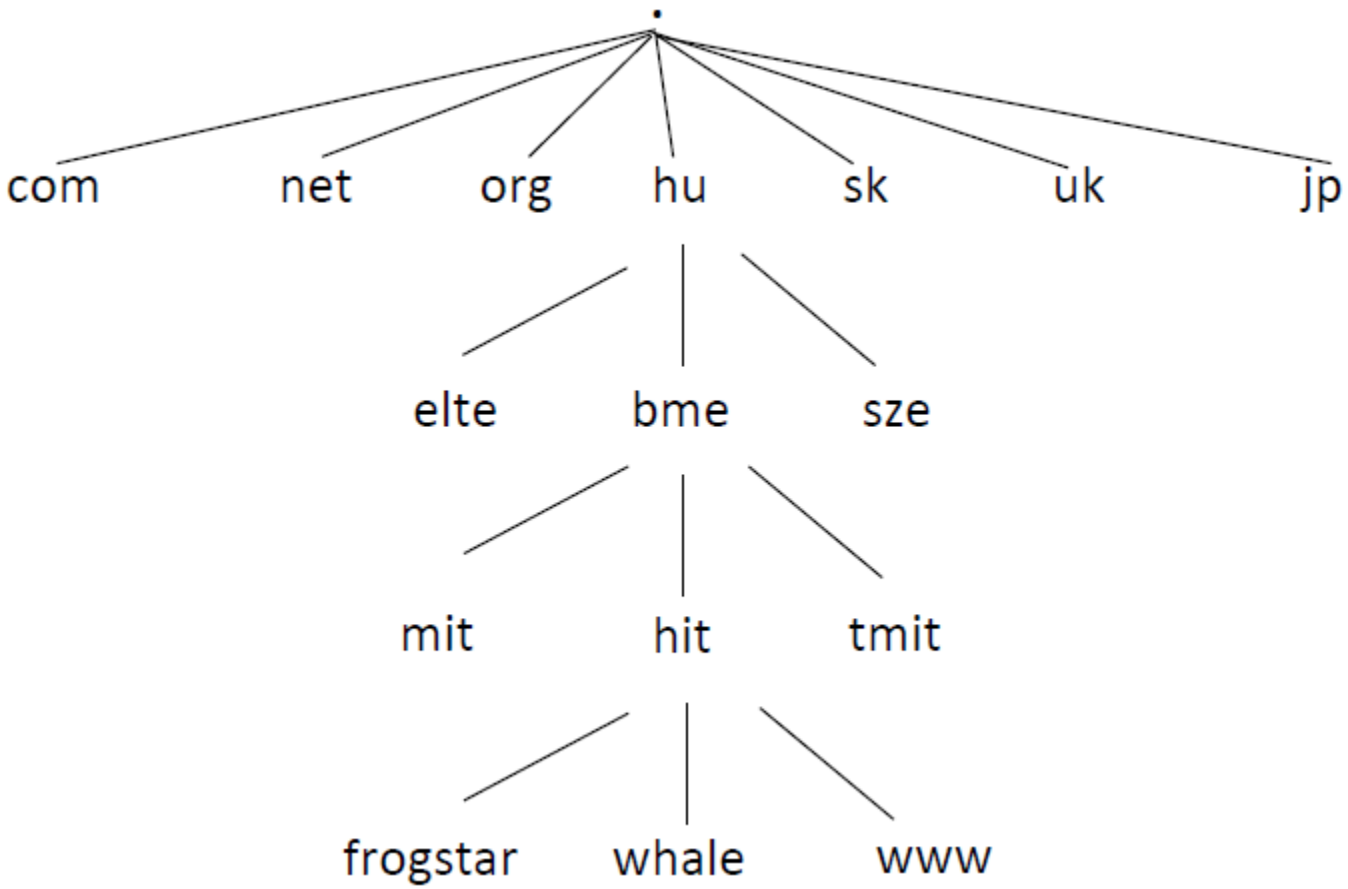
DOMAIN NAME SYSTEM

IP-cím helyett szimbolikus név

- IP-címeket nehéz megjegyezni (már IPv4-nél is)
- A szervezetre utaló szimbolikus neveket könnyebb
- Kell egy leképzés: szimbolikus név --> IP-cím
- Ezt valósítja meg a DNS: Domain Name System

Szimbolikus nevek

- Példa: **`whale.hit.bme.hu.`**



Szimbolikus nevek

- Legfelül egy „.” (pont) van, amit a nevek írásakor általában elhagyunk
- Alatta vannak a legfelső szintű tartományok (TLD: Top Level Domain).
- Ezek további tartományokra bomlanak, ami tetszőleges mélységig folytatható.
- Az ábrán:
  - hu: Magyarország
  - bme: Budapesti Műszaki és Gazdaságtudományi Egyetem
  - hit: pedig a Hálózati Rendszerek és Szolgáltatások Tanszék
  - whale: számítógép neve (*relatív név*)
- FQDN: Fully Qualified Domain Name, *abszolút név*:

  `whale.hit.bme.hu.` (a pont jelzi, hogy FQDN)

TDL-k fajtái

- gTLD: generic TLD (szervezetek fajtái szerint)
  - Például:

gTLD	szervezet fajtája
com	üzleti vállalkozás
org	non-profit szervezet
net	hálózattal kapcsolatos
edu	oktatási intézmény
gov	USA közigazgatás

- ccTLD: country code TLD (országkód alapján)
  - Az ITU (International Telecommunication Union) illetve az ISO 3166 szabvány szerinti kétbetűs ország megjelölést követik, pl. hu, sk, ua ≠ uk (gb helyett!), ro, rs, hr, si ≠ sl, at, de, ru, jp, cn, in, stb.

Subdomainek létrehozása

- Egy FQDN középső része utal a szervezetre
- Egy szervezet szabadon definiálhat *altartomány*okat (subdomain) akár több szinten is
- Például: bme.hu-n belül:
  - **`hit.bme.hu`**
  - **`tmit.bme.hu`**
  - **`iit.bme.hu`**
- A **`bme.hu`** zónát leíró zónafájlban bejegyzések
  - Az aldomain tartalmának leírását delegálják
  - A **`whale`** és a **`frogstar`** bejegyzése a **`hit.bme.hu`** zónát leíró zónafájlban van

Domain és zóna közötti különbség

- Domain: aminek az a végződése
- Zóna: ami az adott zónafájlban nyer feloldást
- Példák
  - a **`www.bme.hu`** és a **`www.hit.bme.hu`** egyaránt a **`bme.hu`** domainben vannak, de
  - a **`www.bme.hu`** a **`bme.hu`** zónában van,
  - a **`www.hit.bme.hu`** a **`hit.bme.hu`** zónában van.

Root DNS szerverek

- A *root zóna* feloldása a feladatuk
- Történeti okokból logikailag 13 db van
  - Mert 512 bájtba ennyi fért bele
- Van szimbolikus nevük is:
  - **`a.root-servers.net`**
  - **`b.root-servers.net`**
  - …
  - **`m.root-servers.net`**
- Mindegyik elérhető IPv4 és IPv6 címen is
- Mindegyik anycast címen van
- Összesen több száz példány van
- Térkép: http://www.root-servers.org/

Névfeloldás menete: kezdet

- Mi történik, amikor egy hálózati alkalmazás találkozik egy szimbolikus névvel?
  1. Megkéri a *névfeloldó*t, hogy adja meg az IP-címet
     - A *névfeloldó* (name resolver) egy könyvtári függvény
     - Linux alatt régebben **`gethosbyname()`** – obsolete, helyette:
     - **`getaddrinfo()`** – IPv4-hez és IPv6-hoz is használható
     - Ezt a függvényt hívja meg az alkalmazás
  2. A névfeloldó a gép beállításainak függvényében jár el
     - Linux alatt **`/etc/nsswitch.conf`** fájlban például:

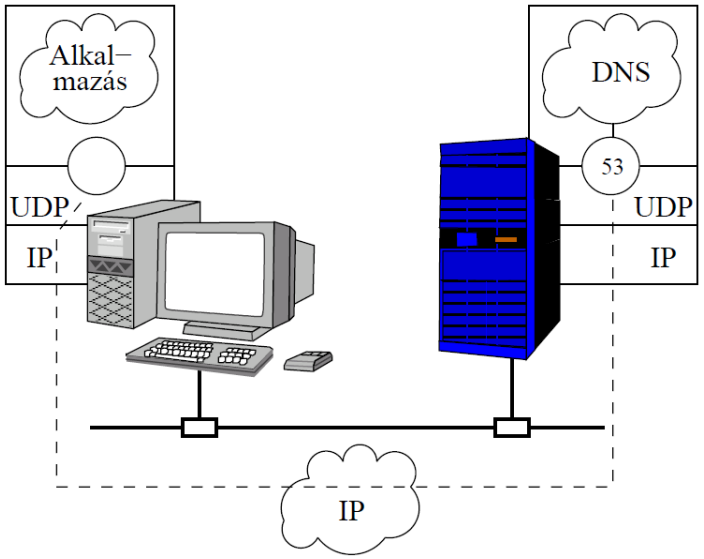
       `hosts: files, dns` bejegyzés esetén

       először a **`/etc/hosts`** fájlban keres,

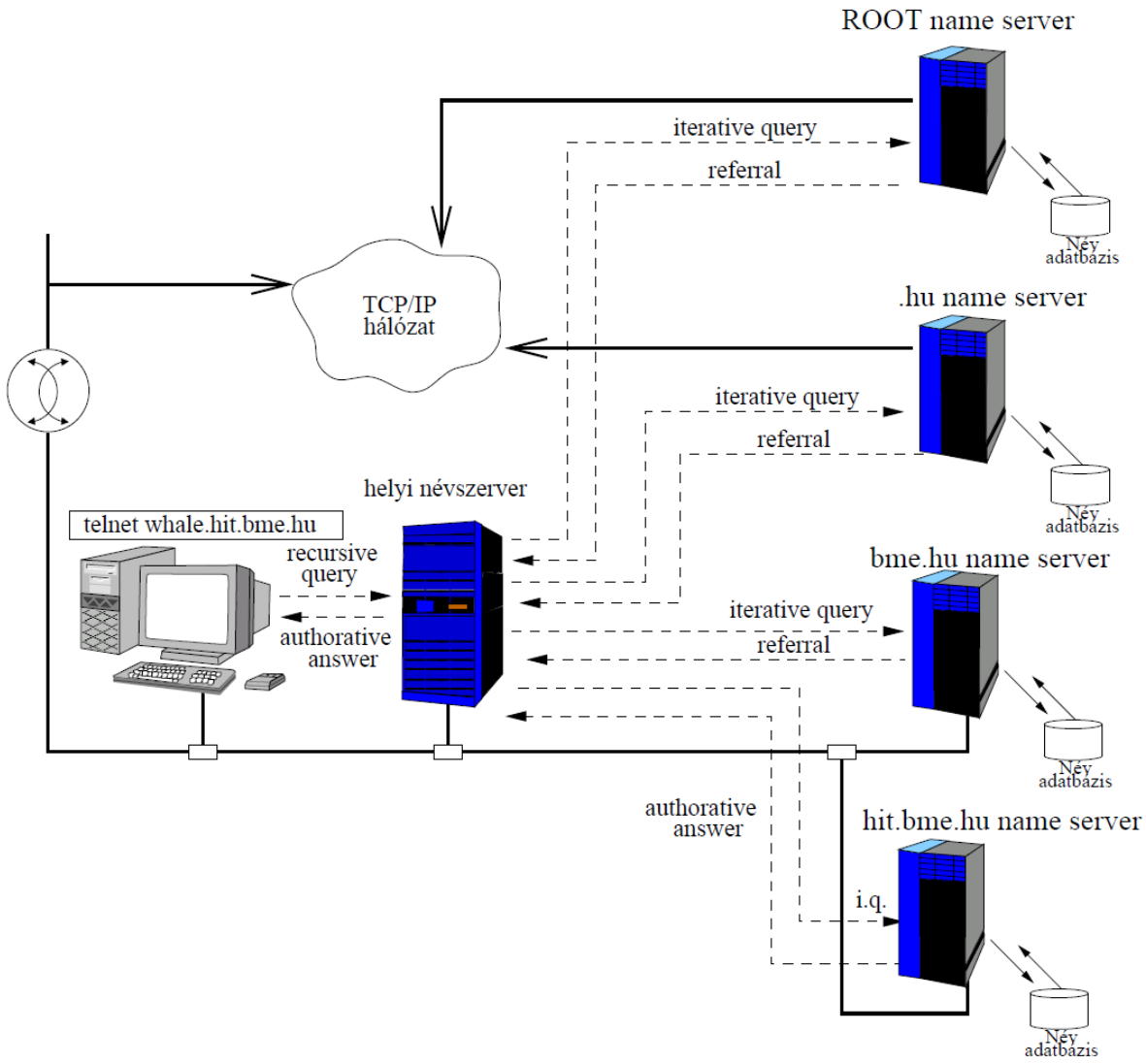
       majd a **`/etc/resolv.conf`** fájlból kiolvassa a névkiszolgáló (name server) IP-címét, és ahhoz fordul

Kommunikáció a helyi névkiszolgálóval

- Alapesetben UDP fölött, mert:
  - Egy kérés + egy válasz: 2 üzenet
  - Ha TCP-t használna, akkor: 3 + 2 + 4 = 9 üzenet lenne
- De ha a válasz nem fér bele egy UDP datagramba, akkor megkérdi TCP fölött



Példa névfeloldásra

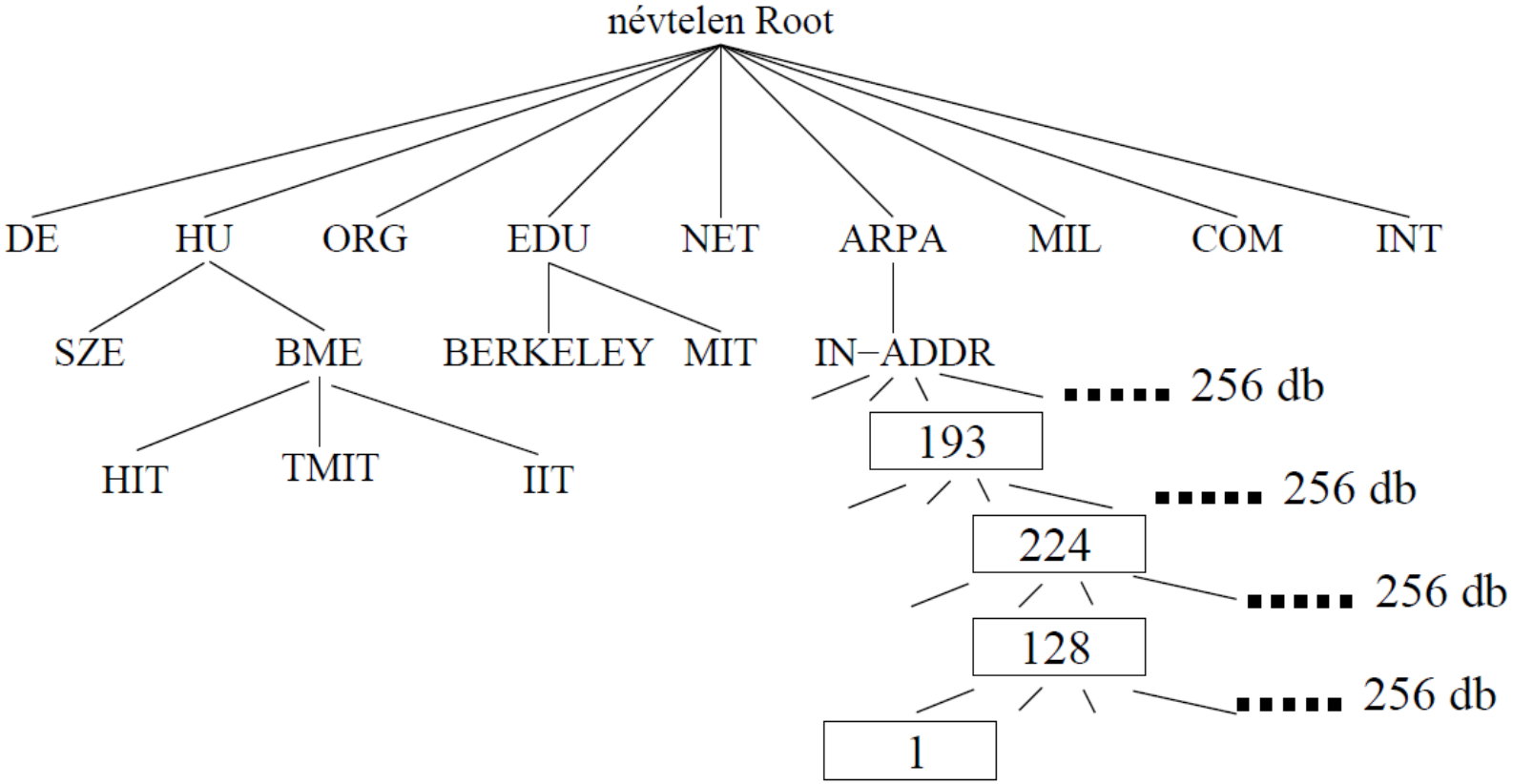


Névfeloldás lépései a példa alapján

- Kliens megszólítja a helyi névkiszolgálót
  - *recursive query*: végső választ vár rá
- Helyi névkiszolgáló TLD-től kezdve „nyomoz”
  - *iterative query*k sorozata: 1-1 lépéssel jut közelebb
  - *referral*nak hívják ezeket a válaszokat
  - végül a zónáért felelős DNS szerver: *authoritative answer*
- Helyi DNS szerver válasza a kliensnek
  - *authoritative answer*

Reverse mapping

- A másik irány: IP-cím alapján szimbolikus név
  - A 193.224.128.1 IP-címhez milyen szimbolikus név tartozik?
  - Válasz az **`1.128.224.193.in-addr.arpa.`** ágon



Fontosabb DNS rekordok

- A – IPv4 cím (Address)
- AAAA – IPv6 cím (4x olyan hosszú, mint az A)
- PTR – szimbolikus név (PoinTeR)
- CNAME – alias (Canonical NAME)
- MX – levelező kiszolgáló (Mail eXchanger)

Caching

- A névszerverek tárolják és felhasználják a megszerzett információt
  - A tárolás legfeljebb az információ érvényességi idejének lejártáig történhet, amit az információ gazdája az információval együtt szolgáltat.
- Caching only name server
  - Nincs olyan zóna, amiért felelős lenne, csak recursive queryk megválaszolása a feladata

Domain név regisztráció fogalmai

- **registry** Egy TLD adminisztrálásának felelőse, delegálja az adott TLD subdomainjeit, de az ügyfelek közvetlenül nem fordulhatnak hozzá.
- **registrar** Magyarul regisztrátor, általában egy profit orientált cég (de akár egyesület is lehet), akihez az ügyfelek fordulhatnak domain név regisztrációért.
- **registration** A domain név regisztráció folyamata.
- A **hu** domain adminisztrálásának felelőse az Internet Szolgáltatók Tanácsa (ISZT).

  https://www.domain.hu/domainregisztracios-szabalyzat/

Domain nevek helyesírása – eredeti

- Az eredeti, RFC 1035 szerinti szabályok
  - A szimbolikus nevekben (más kifejezéssel domain nevekben) nem különböztetjük meg a kis- illetve a nagybetűket.
  - A pontok közötti szakaszokat angolul *label*nek hívják.
  - Egy label hossza 1-63 karakter lehet,
    - betűvel („a”-„z”) kell kezdődnie
    - betűket, számjegyeket („0”-„9”) és kötőjelet („-”) tartalmazhat, de az utóbbi csak a belsejében fordulhat elő, a határán nem.
  - Az FQDN teljes hossza nem haladhatja meg a 255 karaktert.
- Megjegyzés: egyes Microsoft szoftverek a fent felsorolt megengedett karaktereken kívül még az aláhúzás jelet („_”) is használják.

Nemzetköziesített domain nevek

- IDN: Internationalized Domain Name
  - Lehetővé teszi például a magyar ékezetes karaktereket is
  - De „fából vaskarika”
    - A unicode karaktereket tartalmazó stringet egy megfelelő algoritmussal (Punycode, RFC 3492) átkódolják ASCII karakterekből álló stringgé, majd elé teszik az ún. ACE (ASCII Compatible Encoding) prefixet („**xn--**") és a DNS rendszerben így tárolják.
    - Természetesen az átkódolás után nyert ASCII stringre továbbra is érvényesek a korábbi szabályok (például egy label hossza legfeljebb 63 karakter lehet).
  - További szempontok:

    https://www.domain.hu/ekezetes-hu-domain-nevek/

DNS szerver implementációk

- BIND: Berkeley Internet Domain Name Server
  - Legszélesebb körben használt megoldás
  - „de facto” szabvány (pl. zónafájljait mások is értik)
- Továbbiak:
  - NSD
  - Knot DNS (csak authoritative)
  - Unbound (csak recursive)
  - PowerDNS
  - YADIFA

TÁVOLI BEJELENTKEZÉS: TELNET, SSH, SCP

Röviden a telnetről

- Korábban távoli bejelentkezésre használták
  - Ma erre nem használjuk, mert titkosítás nélkül viszi át a jelszavakat és a tartalmat is
  - Helyette ssh-t használunk
- Más protokollok vizsgálatára használjuk
  - Például egy SMTP szerverre így lépünk be vele:
  - **`telnet localhost 25`**

SSH: biztonságos távoli bejelentkezés

- Kliens – szerver architektúra
  - Az SSH szerver (sshd) a 22-es TCP porton várja a kliens csatlakozását
- Mielőtt használnánk, szükség van:
  - gépenként nyilvános és titkos kulcs
  - felhasználónként nyilvános és titkos kulcs (ha kulcs alapú authentikációt szeretnénk használni)

SSH kapcsolat fő lépései

1. titkos csatorna létrehozása a két gép között
2. a felhasználó azonosítása
3. titkosított kommunikáció
   - ha az előző kettő sikeres, akkor ez biztonságos

(az első kettőt részletesen megnézzük)

Titkos csatorna létrehozása

- **Előfeltétel**: A kliens programot futtató gépnek ismernie kell a szerver programot futtató gép nyilvános kulcsát.
  - Egyébként lásd: sebezhetőség
- **A lényeg**: kapcsolatkulcs létrehozása, amit a kapcsolat lezárásáig a két fél minden további kommunikációja során titkos kulcsú titkosítás kulcsaként használ. Ezzel a kulccsal valamilyen titkos kulcsú algoritmussal (3DES, blowfish, CAST128, Arcfour) titkosítják az adatfolyamot.
  - Érdeklődőknek: Diffie-Hellman kulcscsere protokoll (Valójában nem kulcsok cseréjéről, hanem kulcsmegegyezésről van szó.)
- **Sebezhetőség**: Ha a kliens gépen nincs meg a szerver gép nyilvános kulcsa, és a felhasználó úgy dönt, hogy elfogadja a – remélhetőleg a szerver által felajánlott – kulcsot, akkor azt kockáztatja, hogy amennyiben a kulcs a támadótól származik, akkor nem a szerverrel, hanem a támadóval hozott létre közös kapcsolatkulcsot.

A felhasználó azonosítása

- Az authentikáció a következő három, erejében egyre gyengülő megoldás valamelyikével történik:
  1. Erős azonosítás nyilvános kulcsú módszerrel
     - A gyakorlatban is megismerjük
  2. Jelszavas azonosítás
     - A már létrehozott titkos csatornán keresztül történik, így biztonságos, ha a titkos csatorna létrehozása sikeres volt.
     - Ha a kapcsolatkulcs a támadóval közös, akkor a felhasználó jelszava a támadó birtokába jut!
  3. Berkeley r* (szerű megoldás) használata
     - Annyit kell róla tudni, hogy tiltsuk le. ☺

SSH a gyakorlatban

- Távoli gép nyilvános kulcsát itt tároljuk:
  `~/.ssh/known-hosts`
- Ha erős azonosítást szeretnénk használni
  - Létrehozunk egy kulcspárt az **`ssh-keygen`** paranccsal
    - Lehet RSA vagy DSA
    - Pl. DSA esetén a kulcsok: **`id-dsa`** és **`id-dsa.pub`**
  - Publikus kulcsunkat elhelyezzük a cél gépen
    - pl. a **`~/.ssh/authorized_keys`** fájlba
      `ssh-copy-id -i ~/.ssh/id_dsa.pub user@gep`
- Így már jelszó nélkül beléphetünk ☺

Verziókról, fájlnevekről: https://www.openssh.com/txt/release-3.0

Az scp használata

- Az SSH csomag része az **`scp`** is.
  - Logikája hasonló mint a Linux **`cp`** parancsáé
  - Példák:
    - Tegyük fel, hogy a **`pc6`** gép előtt ülünk, és a **`pc2`** gépen **`joska`** a felhasználói nevünk. Másoljuk át a helyi gépen az aktuális könyvtárban található **`cica.jpg`** fájlt a **`pc2`** gépre a **`/tmp`** könyvtárba **`macska.jpg`** névre:

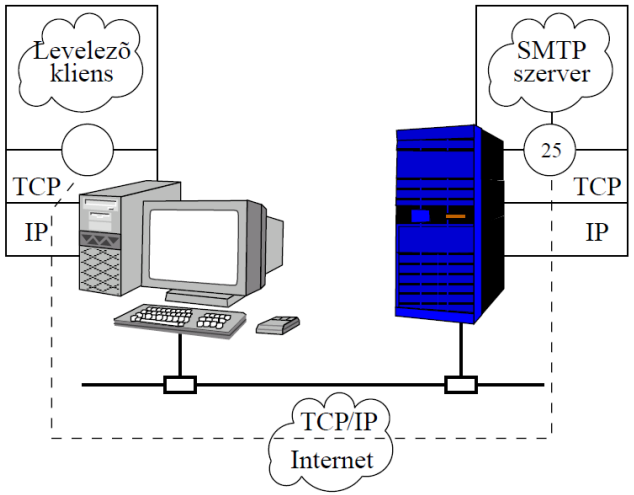
      `scp cica.jpg joska@pc2:/tmp/macska.jpg`
    - Amennyiben a **`pc2`** gépen a home könyvtárunkban van egy **`kutya.jpg`** nevű fáj, amit a helyi gép aktuális könyvtárába szeretnénk másolni, akkor azt megtehetjük például így:

      `scp joska@pc2:kutya.jpg .`

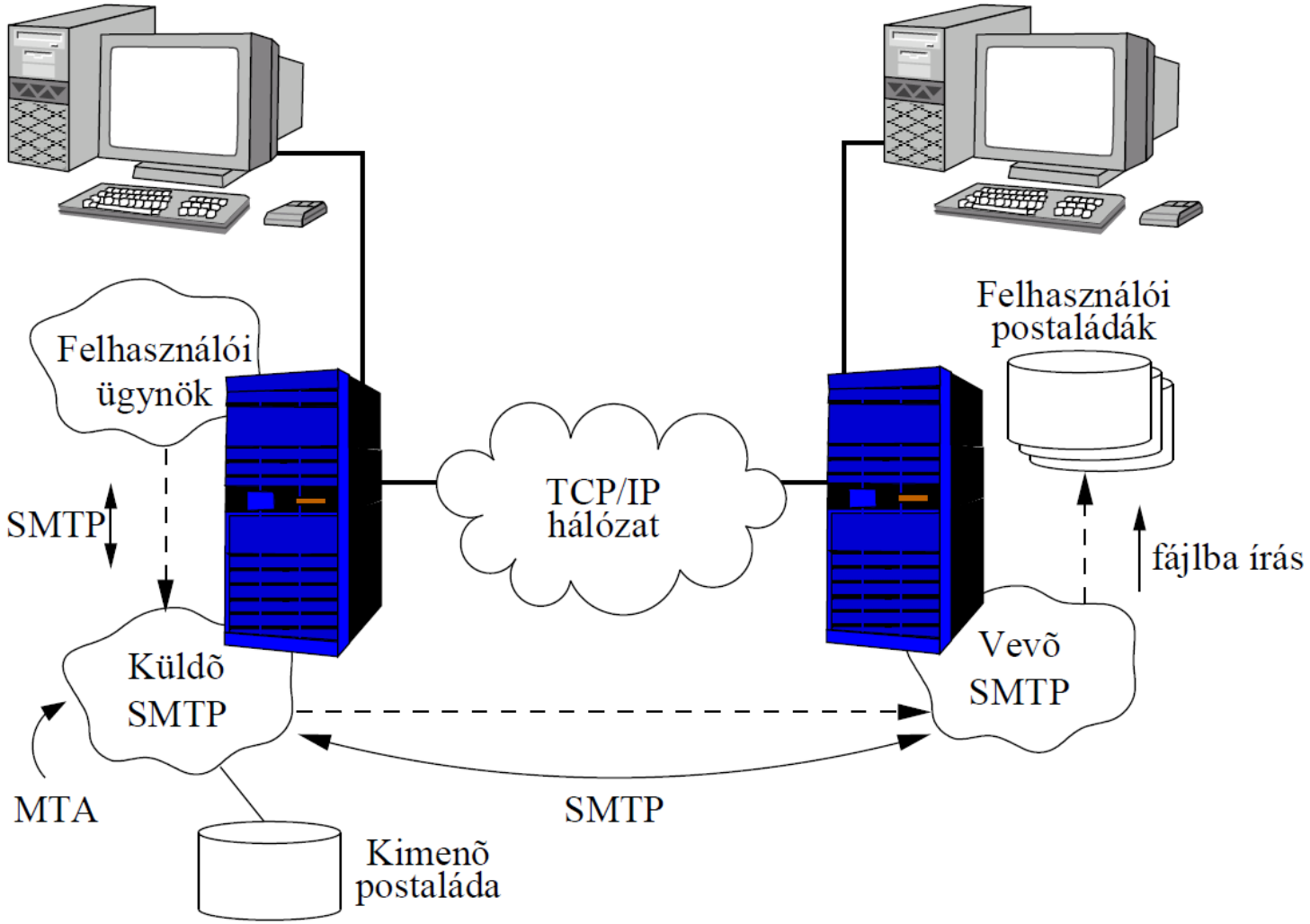
SMTP

Simple Mail Transfer Protocol

- ASCII kódú szöveges üzenetek továbbítására képes TCP/IP protokollt használó hostok között
- Fogalmak (eredeti koncepció)
  - MUA/UA: (Mail) User Agent – *felhasználói ügynök*
    - Levelező kliens program, pl. Thunderbird
  - MTA: Message Transfer Agent – *üzenetátviteli ügynök*
    - Levelező szerver (2 funkció)
      - kimenő
      - címzetté



Levélküldés SMTP-vel (régen)



Message Submission Protocol

- Spam (kéretlen levelek) küldése miatt
- UA és kimenő SMTP szerver között használjuk
- SMTP-hez hasonló, de:
  - Eltérő portszám: 25 helyett 587
  - Kliens azonosítása (authentication)
- MSA: Message Submission Agent
  - Feladata a UA-től a levelet átvenni
- További információ: RFC 6409

SMTP üzenetformátum

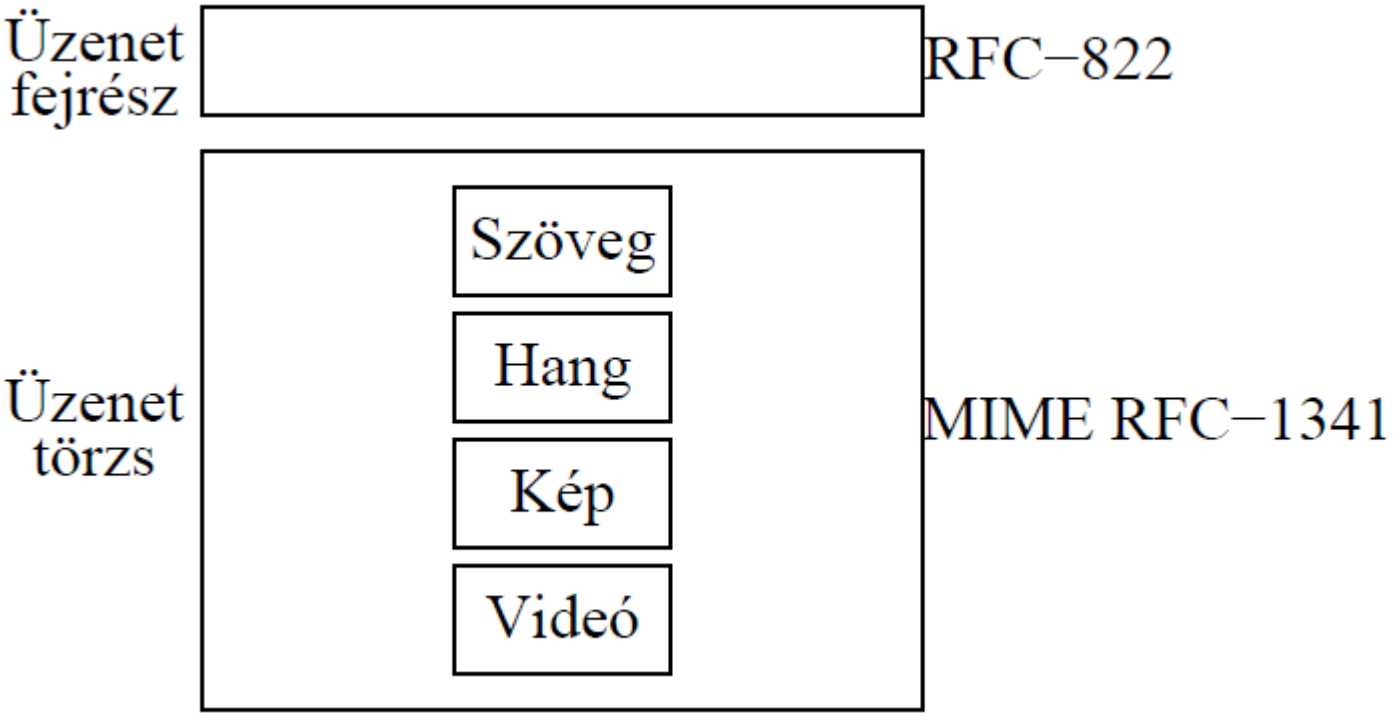
- Eredeti definíció miatt RFC 822 fejrésznek is hívják
- Aktuális definíció: RFC 5322
- E-mail címek
  - 1. példa: **`lencse@rs1.sze.hu`**
    - A @ előtt: postafiók neve
    - A @ után a host FQDN neve (nem szükségszerű!)
  - 2. példa: **`lencse@hit.bme.hu`**
    - Kérdezzük le az MX rekordot!

```
lencse@dev:~$ host -t mx hit.bme.hu
hit.bme.hu mail is handled by 5 frogstar.hit.bme.hu.
```

Nem ASCII kódú információ átvitele

- Át kell kódolni ASCII-re
- Korábban uuencode/uudecode
- Ma MIME: Multipurpose Internet Mail Extensions
  - egyszerű szöveg, gazdagabban formázott szöveg (rich text), kép, hang, videó, HTML, stb. kódolására
  - Megadja
    - tartalom típusa (pl. text/plain, application/pdf)
    - kódolás (pl. quoted-printable, base64)
  - Üzenet törzsében részekre osztott tartalom
    - Egy terminál az jelenít meg belőle, amit tud
  - Eredeti definíció: RFC 1341, ma: RFC 2045-2049

MIME üzenet (csak illusztráció)



SMTP protokoll parancsai

Parancs	Jelentés
HELO küldő	A küldő gép, amelyen a MUA fut.
MAIL FROM: feladó_címe	Ez a parancs adja meg a feladó postaláda címét.
RCPT TO: célcím	Ez a parancs adja meg a cél postaláda nevét. Több címzett esetén többször kell alkalmazni a parancsot.
DATA	A parancs után lehet megadni a levél tartalmát. Az üzenet végét egy olyan sor jelzi, amelyben csak egy pont van.
QUIT	A parancs hatására a vevő OK választ küld és bezárja a kapcsolatot.
RSET	Ez a parancs törli az épp aktuális folyamatot, utána újra lehet kezdeni.
NOOP	Ez a parancs nem hajt végre műveletet. A vevő egy OK választ ad. A parancs akkor hasznos, ha meg akarunk győződni róla, hogy a kapcsolat rendben van-e, a szerver működik-e.

SMTP szerver válaszai

- 3 számjegyes üzenetkód + szöveges üzenet (angol):

  `nnn Szöveges üzenet`

  - Az 1. számjegy az eredmény (hiba) jellegét adja meg, például: 2xx: pozitív eredmény, 4xx: tranziens hiba, 5xx: permanens hiba, stb.
  - Néhány példa (szövege magyarra fordítva)

Kód	Jelentés
250	Kért levél művelet OK, sikeresen befejeződött.
450	Kért levél művelet nem történt meg, a postaláda nem elérhető. Például a postaláda foglalt.
550	Kért levél művelet nem történt meg, a postaláda nem elérhető.
354	Kezdje a levelet. Befejezés: <CR><LF>.<CR><LF>. (Azaz egy sorban csak egy pont áll.)

Néhány fogalom

- Envelope sender, envelope recipient
  - A levél fejrészében megjelenő küldő/címzett
  - Mint amit egy postai levél borítékjára írunk
  - MAIL FROM: és RCPT TO: parancsokkal adjuk meg
- Látszólagos feladó, címzett
  - Ami a levelező kliens programban látszik
  - Amint egy hivatalos levélen szerepel a feladó és a címzett
  - A DATA parancson belül adjuk meg „From: ” és „To: ” után
  - Hasonlóan adható meg a levél tárgya „Subject: ” után

Példa SMTP-vel való levélküldésre

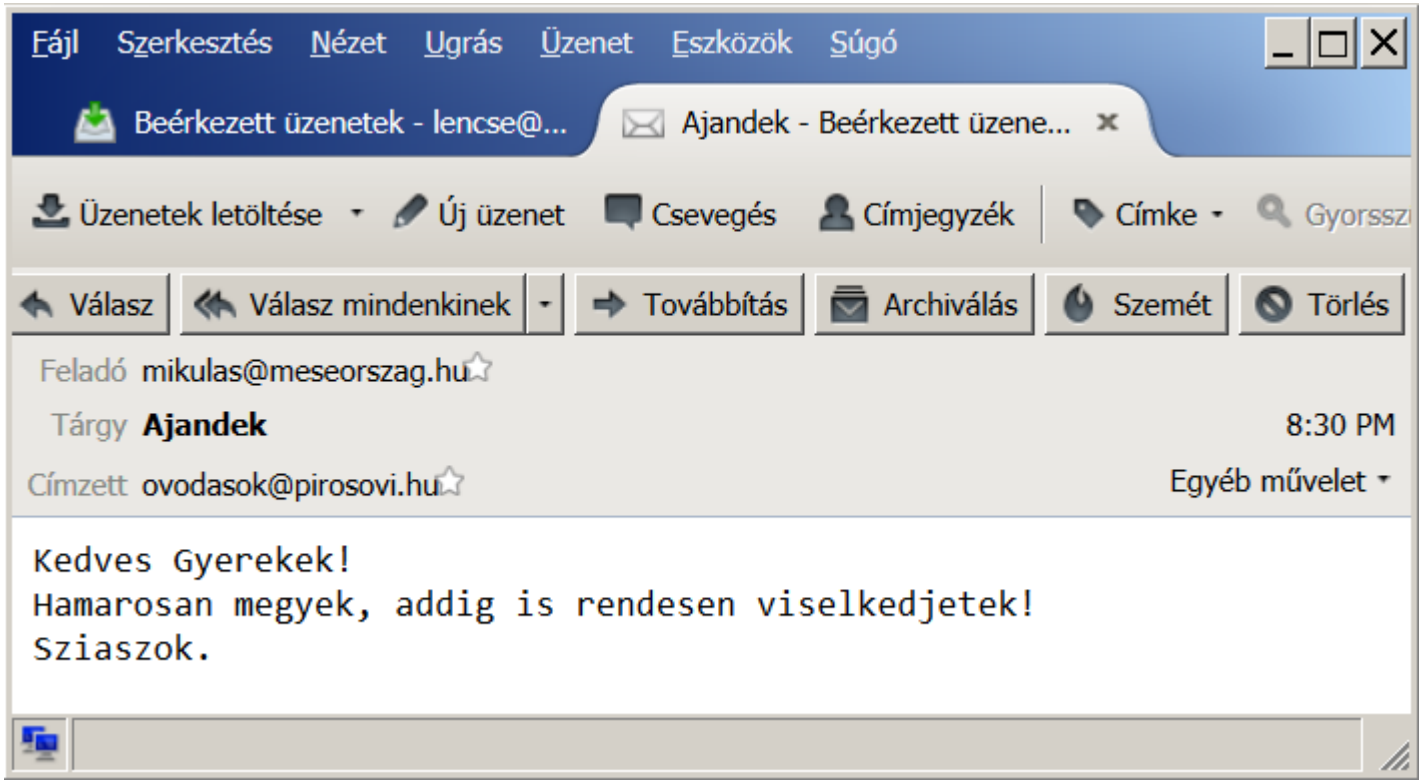
```
lencse@server:~$ telnet localhost 25   ← kapcsolódunk az MTA-hoz
Trying ::1...                         ← a telnet írja ki
Connected to localhost.               ← a telnet írja ki
Escape character is '^]'.             ← a telnet írja ki
220 server.test ESMTP Postfix (Debian/GNU)  ← a szerver köszönt bennünket
helo localhost                        ← a MUA a helyi gépen fut
250 server.test                       ← az MTA válasza, ez az FQDN
mail from: lencse@sze.hu              ← envelope sender megadása
250 2.1.0 Ok                          ← az MTA válasza
rcpt to: lencse@hit.bme.hu            ← envelope recipient megadása
250 2.1.5 Ok                          ← az MTA válasza
```

(folytatjuk)

Példa SMTP-vel való levélküldésre

```
data                                          ← minden mást ezen belül adunk meg
354 End data with <CR><LF>.<CR><LF>           ← MTA: így kell befejezni a data részt
From: mikulas@meseorszag.hu                   ← a levélben ez jelenik meg feladóként
To: ovodasok@pirosovi.hu                      ← a levélben ez jelenik meg címzettként
Subject: Ajandek                              ← a levélben ez jelenik tárgyként
                                              ← üres sor a levél szövegének elválasztására
Kedves Gyerekek!                              ← a levél szövegének eleje
Hamarosan megyek, addig is rendesen viselkedjetek!
Sziaszok.                                     ← a levél szövegének vége
.                                             ← ebben a sorban csak egy pont van
250 2.0.0 Ok: queued as 636EE60062            ← az MTA válasza
```

Így néz ki a levél…



Megjegyzések

- Tesztelésen kívül mások nevében nem küldünk levelet!
- Egy MTA nem továbbíthat akárhonnan akárhova levelet authentikáció nélkül, mert akkor: Open relay
  - Feketelistára szokták tenni, mert SPAM küldésre használható!
  - Érdeklődőknek: https://en.wikipedia.org/wiki/Open_mail_relay

Postafiók implementációk

- A „mailbox” formátum
  - Egyetlen egy nagy fájl
  - Az új levelek kezdete: sor elején „**`From `**” (5 karakter)
    - Ha levél törzsében fordul elő, akkor „**`>From `**”-ra cserélik, és megjelenítéskor a „**`>`**” karaktert kihagyják belőle.
  - Utólag az RFC 4155-ben dokumentálták
  - Elhelyezés Linux alatt:
    - Gyakran **`/var/spool/mail/username`** fájlban
    - De lehet a user home könyvtárában is

Postafiók implementációk

- A „maildir” formátum
  - Minden levél külön fájlban van
  - Elhelyezés Linux alatt:
    - Gyakran a felhasználó home könyvtárában egy **`Maildir`** nevű könyvtárban, de lehet máshol is
  - Ezen belül három további könyvtár:
    - **`tmp`** nevű könyvtárba érkeznek meg a levelek
    - **`new`** nevű könyvtárba teszi át az MTA, ha megérkeztek
    - **`cur`** nevű könyvtárba helyezi át a MUA az új leveleket megnyitás előtt (a **`new`** nevű könyvtárból).

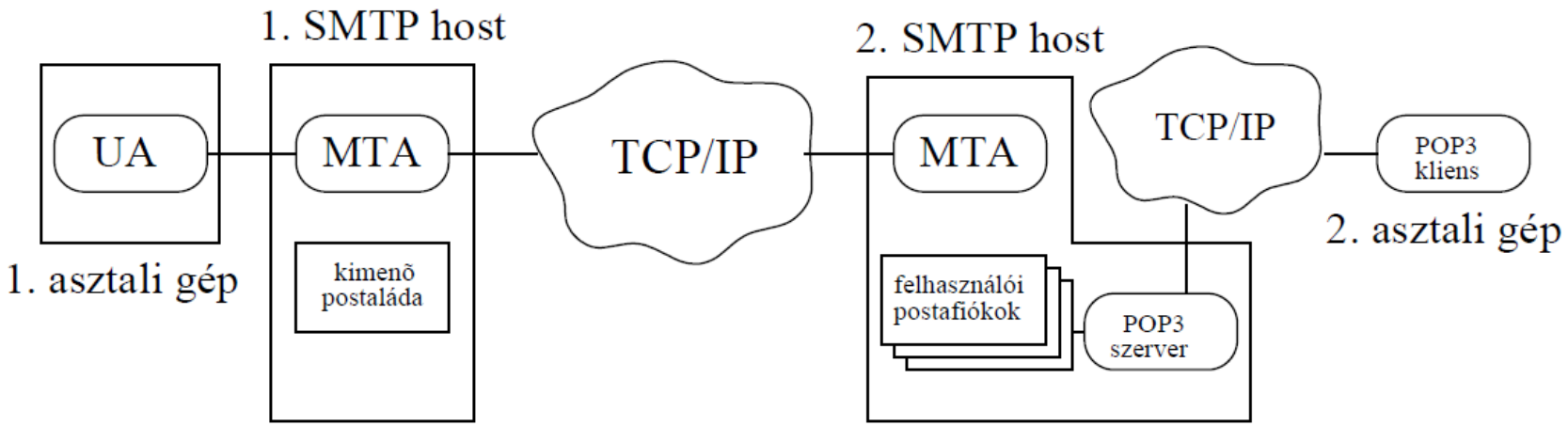
POP3

Post Office Protocol Version 3

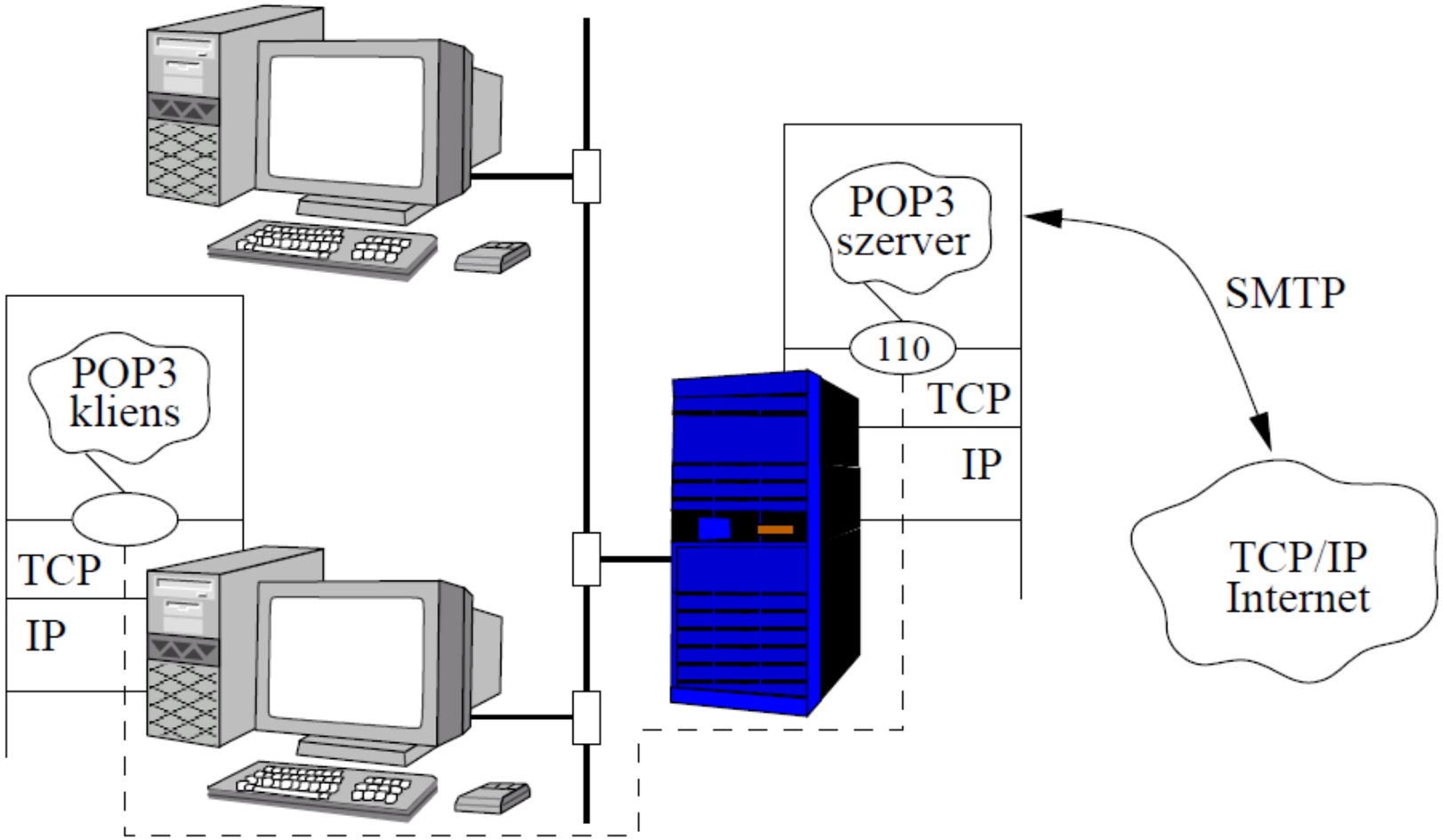
- Miért van szükség az SMTP-n kívül másra is?
  - SMTP-hez követelmény, hogy a címzett számítógépe állandóan legyen:
    - Bekapcsolt állapotban
    - Interneten elérhető
  - Ez sok esetben nem teljesül
    - pl. otthoni asztali gépek, notebookok
- Megoldás
  - Fogadó SMTP szerver egy állandóan elérhető gép
  - A postafiókot távolról érjük el: POP3 protokollal

Post Office Protocol Version 3

- A címzett leveleit kezelő SMTP szerver állandóan elérhető
- A postafiókhoz a felhasználó a POP3 protokoll segítségével fér hozzá



POP3 kliens-szerver kapcsolat



POP3 protokoll parancsai

- Az SMTP-hez hasonlóan 4 betűs parancsok
- A válaszokban viszont nem szám, hanem +OK vagy -ERR után szöveges rész, a parancstól függően
- Vannak
  - Kötelező parancsok
  - Opcionális parancsok

Kötelező POP3 parancsok –1

Parancs	Jelentés
STAT	Erre a parancsra olyan választ kapunk, amely egy +OK stringgel kezdődik, majd egy szóköz után az üzenetek száma és még egy szóköz után a levelek összmérete látható oktettekben megadva.
LIST [üzenet_sorszám]	Ha megadunk üzenet sorszámot, akkor a parancs hatására a szerver válaszképpen egy +OK string után, szóközzel elválasztva kiírja a kért üzenetazonosító mellé a méretét oktettekben. Ha nem adunk meg sorszámot, akkor egy többsoros válasz kapunk, melynek első sora egy +OK stringgel kezdődik, majd egy szöveges üzenettel folytatódik. A következő sorok elején az üzenet sorszáma és szóköz után a hozzá tartozó üzenet mérete látható.
RETR üzenet_sorszám	Ez a parancs arra szolgál, hogy letöltsük a POP3 szerveren lévő üzenetünket. Válaszként ad egy +OK string-et, majd egy szóköz után a letöltött üzenet méretét oktettekben. Ezután következik maga az üzenet (az, amelyiknek a sorszámát megadtuk). Ha olyan azonosítót adunk meg, amely nem létezik, egy -ERR üzenetet küld a szerver.

Kötelező POP3 parancsok – 2

Parancs	Jelentés
DELE üzenet_sorszám	A parancs az adott üzenetet törölésre kijelöli.
NOOP	Ez a parancs nem hajt végre műveletet. A szerver egy +OK választ ad. A parancs akkor hasznos, ha meg akarunk győződni róla, hogy a kapcsolat rendben van-e, és a POP3 szerver működik-e.
RSET	Ez a parancs törli az összes törlésre való kijelölést. A szerver visszaad egy +OK stringet.
QUIT	A parancs hatására a szerver törli az összes törlésre kijelölt levelet, és az elvégzett művelettől függően +OK vagy -ERR stringet ad vissza, majd lezárja a kizárólagos elérést a postafiókon, és lebontja a TCP kapcsolatot.

Opcionális POP3 parancsok

Parancs	Jelentés
USER név	Ezzel a paranccsal adható meg a kezelni kívánt postaláda.
PASS jelszó	Ez a parancs adja meg a kiválasztott postaládához tartozó jelszót.
TOP üzenet_sorsz n	A parancs hatására a szerver válasza egy +OK string, majd kiírja az üzenet fejrészét és egy üres sor után az üzenet törzsből n sort. Ha ez a szám nagyobb, mint ahány sor van az üzenetben, akkor a szerver elküldi az összes sort.
UIDL [üzenet_sorszám]	Minden üzenet kap egy egyedi azonosítót (UID), amely alapján bárhol azonosítható lesz a levél. Ez a parancs az UID azonosító alapján listázza ki az üzeneteket.
APOP név digest	A név azonosítja a felhasználót, a digest pedig egy MD5-ös (Message Digest version 5) digest string. Ezt a parancsot az egyszerű nyílt szöveg stringeket használó USER/PASS azonosítási metódus helyett szokták használni. A legfontosabb tulajdonsága, hogy a jelszót nem nyílt szövegként küldi el.

Példa POP3 használatára

SZ:	(kapcsolatra vár a 110-es TCP porton)	
K:	(megnyitja a kapcsolatot)	*Kapcsolódási állapot*
SZ:	+OK POP3 users.tilb.sze.hu v2000.70 server ready	
K:	USER <*felhasználó_név*>	
SZ:	+OK User name accepted, password please	*Hitelesítési állapot*
K:	PASS <*jelszó*>	
SZ:	+OK Mailbox open, 8 messages	
K:	STAT	
SZ:	+OK 8 153681	
K:	LIST	
SZ:	+OK Mailbox scan listing follows	
SZ:	1 29486	
SZ:	2 512	
	...	*Tranzakciós állapot*
SZ:	8 65677	
SZ:	<CR><LF>.<CR><LF>	
K:	RETR 1	
SZ:	+OK 29486 octets	
	itt van a levél tartalma	
SZ:	<CR><LF>.<CR><LF>	
K:	QUIT	
SZ:	+OK Sayonara	*Frissítési állapot*
K:	(bezárja a kapcsolatot)	
SZ:	(vár a következő kapcsolatra)	

K=Kliens SZ=Szerver

IMAP4

Internet Message Access Protocol v. 4

- Miért nem elég a POP3?
- Mert nem teszi lehetővé a levelek letöltése nélkül a levelek részeinek áttekintését, azokkal közvetlenül a szerveren való műveletvégzést.
  - Például hány db csatolt fájlom van, azoknak mi a neve, mérete, csak az egyiket szeretném letölteni…
- Megoldás: POP3 helyett IMAP4

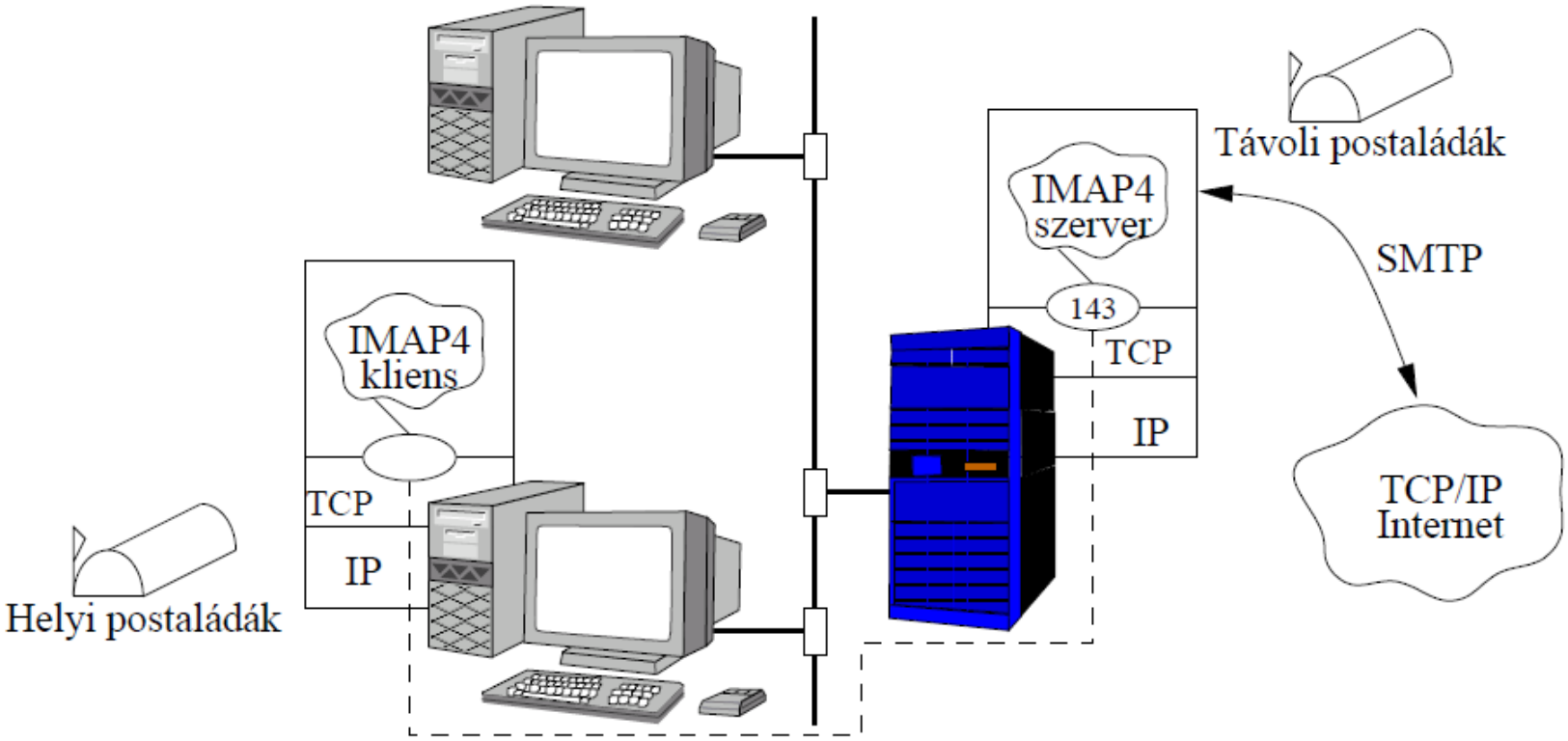
IMAP4 lehetővé teszi

- levelek elérését és szerkesztését a szerveren anélkül, hogy letöltenénk őket
- levelek és csatolt fájlok áttekintését anélkül, hogy letöltenénk őket
- levelek letöltését kapcsolat nélküli munkához
- szinkronizálást a helyi és a szerveren lévő postaládák között

Az IMAP4 műveletei között szerepel:

- postaládák létrehozása, átnevezése, törlése
- új levelek érkezésének ellenőrzése
- levelek eltávolítása a postaládából
- levelek állapotjelzőinek beállítása, és törlése
- RFC-822 fejrész felismerése és MIME kódolású levelek elemzése („érti” a MIME kódolást)

IMAP4 kliens-szerver kapcsolat

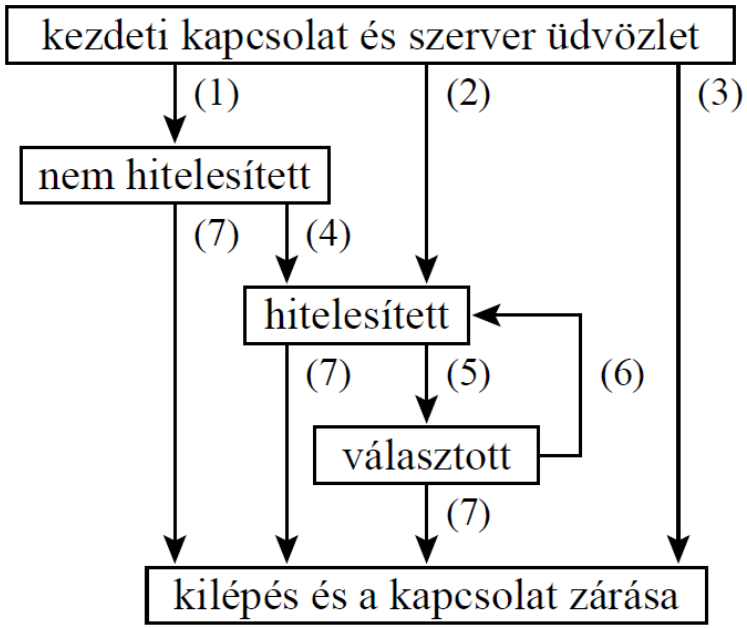


Az IMAP4 állapotai

- azonosítás előtti állapot
  - a felhasználónak azonosítania kell magát (hitelesítés)
- azonosított (más szóval: hitelesített) állapot
  - felhasználó már hitelesített, de mielőtt kiadná a parancsokat, ki kell választania egy postaládát
- választott állapot
  - a felhasználó kezelheti a kiválasztott postaládát
  - és kiválaszthat helyette másikat is
- kijelentkezett állapot
  - (a kliens kérése vagy a szerver döntése alapján kerül ide)
  - a szerver bezárja a kapcsolatot

Az IMAP4 állapotátmenetei:

Állapotátmenetek:

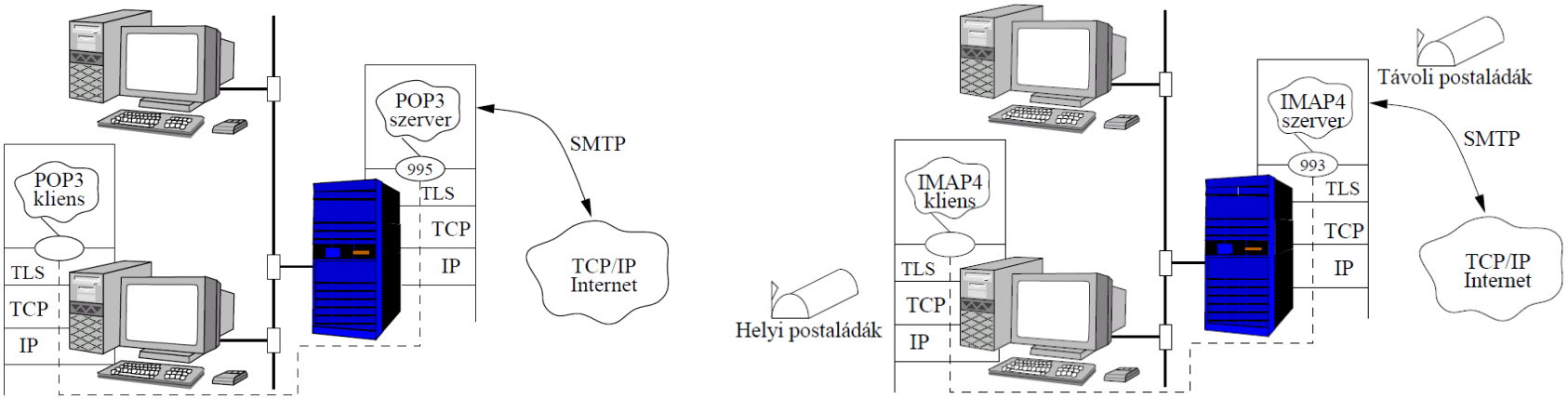


1. kapcsolat előzetes azonosítás nélkül (OK üdvözlés)
2. előzetesen azonosított kapcsolat (PREAUTH üdvözlés)
3. visszautasított kapcsolat (BYE üdvözlés)
4. sikeres LOGIN vagy AUTHENTICATE (azonosítás) parancs
5. sikeres SELECT (választás) vagy EXAMINE (vizsgálat) parancs
6. CLOSE (bezárás) parancs, vagy sikertelen SELECT, illetve EXAMINE parancs
7. LOGOUT parancs, a szerverről való kilépés és a kapcsolat bontása.

POP3S/IMAP4S

POP3S/IMAP4S

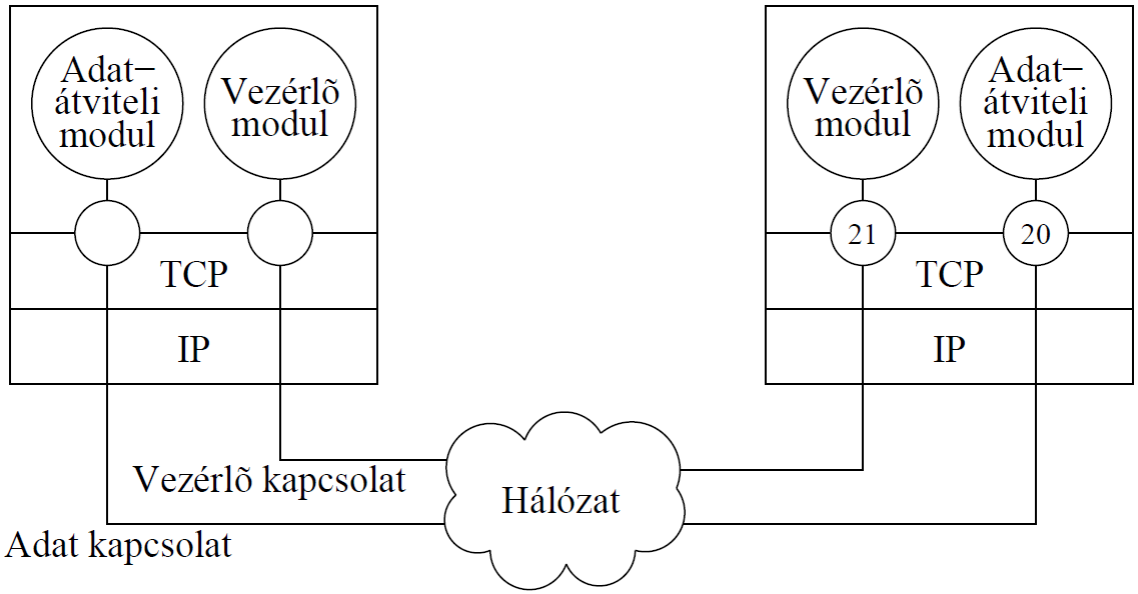
- A POP3/IMAP4 protokollok biztonságos verziói
  - Plusz egy réteg: SSL vagy TLS
  - A parancsok ugyanazok, de a kommunikáció így titkosított csatornán folyik, nem lehallgatható.



FILE TRANSFER PROTOCOL

File Transfer Protocol

- FTP kliens feladata
  - Fájlok letöltése FTP szerverről
  - Fájlok feltöltése FTP szerverre, ha engedélyezve van.



Kliens és szerver között: két kapcsolat

- Vezérlő kapcsolat
  - A kliens a szerver 21-es TCP portjára csatlakozik
  - A kapcsolat folytonosan fennáll, a kliens parancsait és a szerver válaszait továbbítják rajta
- Adat kapcsolat
  - Akár a kliens, akár a szerver kezdeményezheti a felépítését
  - Csak az adatátvitel (fájl vagy könyvtárlista) ideje alatt áll fenn, utána lebomlik.

FTP protokoll parancsai, és válaszaik

- Legfeljebb 4 betűs parancsok, paraméter lehet
  - Opcionális parancsok a táblázatban dőlt betűvel
- Az FTP szerver válaszai
  - három számjegyű kód + szöveges üzenet:

    `nnn Szöveges üzenet`
  - Például:

    `200 Command okay.`

FTP protokoll parancsai – 1

Parancs	Jelentés
USER <felhasználói_név>	Megadja az FTP kapcsolatot kezdő felhasználó nevét.
PASS <jelszó>	Megadja a USER parancsban megadott felhasználóhoz tartozó jelszót.
CWD <könyvtár>	Megváltoztatja az aktuális könyvtárat a szerveren.
CDUP	Átvált az aktuális könyvtárról annak szülő könyvtárára.
PORT h1,h2,h3,h4,p1,p2	*Aktív mód* esetén megadja a TCP végpont információkat. Segítségével megadhatjuk az FTP kliens portszámát az FTP szervernek. A h1-h4-ig az IP-cím, p1 és a p2 pedig a portszám oktettjei decimálisan, vesszőkkel elválasztva. Értelmezésük a hálózati bájtsorrend (MSB) szerinti.
PASV	Megadja, hogy az FTP szerver várakozzon az adatkapcsolatra (*passzív mód*), amit a kliens fog létrehozni. Erre a parancsra adott válasz tartalmazza a TCP végpontot, ahol az FTP szerver várja a kapcsolódást.

Aktív és passzív mód

- Aktív módban az adatkapcsolatot a szerver építi fel a kliens felé
  - A kliens a PORT parancsban küldi el az IP-címet és a portszámot
  - NAT esetén *protocol helper*re van szükség
  - Tűzfal „nem engedi be” az adatkapcsolatot
- Passzív módban a kliens építi fel az adatkapcsolatot a szerver felé
  - A kliens a PASV parancsban kéri a paramétereket
  - A szerver a válaszban megadja
  - NAT és tűzfal esetén is működhet. ☺

FTP protokoll parancsai – 2

Parancs	Jelentés
TYPE <kód>	Megadja az adatmegjelenítés típusát.
RETR <elérési_út>	Megadott fájl letöltése.
STOR <elérési_út>	Megadott fájl tárolása a szerveren (feltöltés).
RNFR <elérési_út>	Megadja a régi elérési utat ahhoz a fájlhoz, amely át lesz nevezve. Az RNTO parancsnak kell követnie.
RNTO <elérési_út>	Megadja az új elérést a fájlhoz. Az RNFR parancs után kell használni.
DELE <elérési_út>	Megadott fájl törlése.
LIST [könyvtár]	Kiírja a megadott könyvtárban lévő fájlok neveit és attribútumait. Ha nincs megadva könyvtár, akkor az aktuális könyvtárat listázza.
NLST [könyvtár]	Kiírja a megadott könyvtárban lévő fájlok neveit attribútumok nélkül. (Könyvtár neve nélkül az aktuálisat. Az mget parancs használja.)

ASCII és bináris átviteli mód

- Szöveges fájlokban a sorok végét mi jelzi?
  - Linux/Unix: 10-as kódú karakter („LF”)
  - Windows: 13-as és 10-es kódú karakterek együtt („CR-LF”)
- ASCII módban konverzió történik
  - Linux alól Windows alá:
    - LF --> CR-LF (10-es kódú karakter elé beszúr 13-as kódút)
  - Windows aló Linux alá:
    - CR-LF --> LF (10-es kódú karakter előtti 13-as kódút törli)
- Bináris módban nincs konverzió
  - Kép, program, adatfájl, stb. tartalma ne változzon!

FTP protokoll parancsai – 3

Parancs	Jelentés
ABOR	A szerver megszakítja az aktuális parancs végrehajtását.
RMD <könyvtár>	Megadott könyvtár törlése.
MKD <könyvtár>	Megadott könyvtár létrehozása.
PWD	Kiírja az aktuális könyvtár elérési útját a szerveren.
SYST	Megkapható a paranccsal, hogy milyen operációs rendszer fut a távoli gépen.
HELP [parancs]	Segítséget nyújt a parancs használatához. Ha nem adunk meg parancsot, kiírja a szerveren használható összes parancsot.
NOOP	Nincs művelet, a szerver küld egy visszaigazolást.
QUIT	Bezárja az FTP kapcsolatot.

FTP a felhasználó szemszögéből

- Az FTP lehetővé teszi a felhasználó számára a távoli gépen történő interaktív fájl- és könyvtárhozzáférést és a következő műveletek végrehajtását:
  - A helyi vagy távoli gépen lévő könyvtárak listázása
  - Fájlok átnevezése és törlése (jogosultság szükséges)
  - Fájlok átvitele a távoli gépről a helyi gépre (letöltés)
  - Fájlok átvitele a helyi gépről a távoli gépre (feltöltés, általában csak adott könyvtárba engedélyezik)
- Felhasználói interface
  - GUI (Graphical User Interface)
  - Parancssor (parancsait könnyebb leírni)

FTP a felhasználó szemszögéből

- Bejelentkezés parancssoros klienssel:

  `ftp [host_név]`

- Azonosítás felhasználói névvel és jelszóval
  - Biztonsági probléma: jelszót nyíltan viszi át
- Anonymous FTP
  - Felhasználó: anonymous
  - Jelszóként e-mail címet kér
    - ezt ma már nem szoktuk megadni, általában elég a „@” karakter léte
  - /pub könyvtárból lehet általában letölteni
  - /pub/incoming könyvtárba esetleg feltölteni

Példa FTP-zés – 1

1. Kiadjuk az FTP parancsot és mellé a host nevét.

```
ftp ftp.hu.debian.org
```

2. Ha a host elérhető, a következőhöz hasonló üzenetet kapunk. A bejelentkezés részletei eltérhetnek.

```
Connected to ftp.freepark.org.
220 ftp.freepark.org FTP server (Version 6.00LS) ready.
User (ftp.freepark.org:(none)):
```

3. Megadjuk a felhasználónevet és a jelszót.

```
User (ftp.freepark.org:(none)): anonymous
331 Guest login ok, send your email address as password.
Password:
230 Guest login ok, access restrictions apply.
ftp>
```

Példa FTP-zés – 2

4. A bejelentkezés után használhatjuk a **?** vagy a **help** parancsot:

```
ftp> help
Commands may be abbreviated.  Commands are:

!               delete          literal         prompt          send
?               debug           ls              put             status
append          dir             mdelete         pwd             trace
ascii           disconnect      mdir            quit            type
bell            get             mget            quote           user
binary          glob            mkdir           recv            verbose
bye             hash            mls             remotehelp
cd              help            mput            rename
close           lcd             open            rmdir
ftp>
```

Példa FTP-zés – 3

5. A **pwd** paranccsal megkaphatjuk, hogy a távoli hoston mi az aktuális könyvtár.

```
ftp> pwd
257 "/" is current directory.
ftp>
```

6. Könyvtár váltásra a **cd** parancsot használjuk

```
ftp> cd /pub/linux/distributions/debian/dists/wheezy/
250 CWD command successful.
ftp>
```

Példa FTP-zés – 3

7. Az aktuális könyvtár fájljainak listáját az **`ls`** vagy **`dir`** parancsokkal kaphatjuk meg.

```
ftp> ls
200 PORT command successful.
150 Opening ASCII mode data connection for 'file list'.
Contents-kfreebsd-i386.gz
Contents-kfreebsd-amd64.gz
...
226 Transfer complete.
ftp: 263 bytes received in 0,00Seconds
263000,00Kbytes/sec.
ftp>
```

Példa FTP-zés – 4

- Fájlok letöltése a **`get fájlnév`** paranccsal
- Több fájl: **`mget`** paranccsal
- Fájlok feltöltése a **`put fájlnév`** paranccsal
- Átviteli mód váltás **`bin`** és **`ascii`** parancsokkal
- Kilépés a **`quit`** paranccsal.

HTML DIÓHÉJBAN

HyperText Markup Language – 1

- A HTML az SGML (Standard Generalized Markup Language) egy implementációja.
- Az SGML szabvány megad egy általános módszert, hogy miképpen készítsünk olyan dokumentumokat, amelyek hyperlinkeket tartalmaznak.
- A *hyperlink* egy kiemelve látható kifejezés a szövegben, amelyet kiválasztva egy újabb dokumentumot kapunk.
  - Egy hyperlink kiválasztása többféle cselekményt is kiválthat, mint például egy másik weboldal megjelenítése, levél küldése, felhasználótól adatok kérése *form*on keresztül, távoli bejelentkezés vagy fájl átvitel kezdeményezése, adatbázis lekérdezése, program futtatása és így tovább.

HyperText Markup Language – 2

- Az olyan dokumentumot, amely hyperlinket tartalmaz, *hyper-dokumentum*nak is hívjuk.
- A hyperlinkek mutathatnak azonos, illetve különböző web szerveren lévő dokumentumokra egyaránt.
- A HTML dokumentumokat a web kliensek, azaz web böngészők segítségével érheti el a felhasználó, melyet saját gépén futtat.
- A böngésző a HTTP protokoll segítségével tölti le a HTML dokumentumot, megjeleníti azt a képernyőn grafikusan, esetleg szöveges formátumban.
  - A HTML leíró nyelv alkalmas a HTML dokumentum különböző elemeinek leírására, melyeket felhasználva a böngésző grafikusan megjelenítheti a weblapot. A linkek általában aláhúzott szövegként jelennek meg. (Természetesen nemcsak szöveg, hanem kép is lehet link.) A linkekre kattintva letölti azt a dokumentumot, amire a link mutat.

A HTML nyelv elemei – 1

- Minden HTML dokumentum *tag*eket tartalmaz (magyarul leginkább *címkének* lehetne nevezni), a következő struktúrában:

```
<tag>
</tag>
```

- A *tag* egy hivatkozás egy speciális kulcsszóra, amelyet a HTML dokumentum különböző komponenseinek megadására használunk. A lezáró tag **</tag>** megadja a komponens végét. Majdnem minden HTML tagnek megvan a hozzá tartozó lezáró tagje. Például minden HTML dokumentum eleje és vége követi a következő megadást:

```
<HTML>
A HTML nyelvű dokumentum különböző elemeit e két
tag közé kell elhelyezni.
</HTML>
```

A HTML nyelv elemei – 3

- A **<HTML>** és **</HTML>** tagek között meg kell adni a HTML dokumentum fejrészét és törzsét.
  - A fejrészt a **<HEAD>** és **</HEAD>** tagek között,
  - a törzset pedig a **<BODY>** és **</BODY>** tagek között kell megadni:

```
<HTML>
<HEAD> Ez itt a fejrész helye. </HEAD>
<BODY> Ez itt a törzs helye. </BODY>
</HTML>
```

- Az üres sorokat és a sortörés karaktereket a tagek között nem vesszük figyelembe. Pl. az alábbi kód ekvivalens a fentivel:

```
<HTML><HEAD> Ez itt a fejrész helye. </HEAD>

<BODY> Ez itt a törzs helye. </BODY></HTML>
```

A HTML nyelv elemei – 4

- Az üres sorok és a sortörések általában javítják a HTML dokumentumok forrásának olvashatóságát, ezért ajánlatos alkalmazni őket.
- A tagek nem érzékenyek a kis- illetve nagybetűkre és az egyéb szövegformázásra. Ha azt szeretnénk, hogy a böngésző által megjelenített weblapon sortörést, üres sort, egynél több szóközt vagy egyéb formázást hajtsunk vége, akkor speciális HTML tageket kell a forrásban elhelyeznünk.

A HTML nyelv elemei – 5

- A weblap címét a **<TITLE>** és **</TITLE>** tagek között kell megadni. A következő szöveg a böngésző címrészében (az ablak címsorában) fog megjeleni:

```
<HTML>
<HEAD>
<TITLE>Ez az első egyszerű weblapom!</TITLE>
</HEAD>
<BODY> Ez itt a törzs helye. </BODY>
</HTML>
```

- Az oldal részeként megjelenő főcímet a **<H1>** taggel adjuk meg!
  - További szinteken használhatók a **<H2>** - **<H6>** tagek
  - A **<B>** és **</B>** tag-ek közé írt szöveg félkövér, míg a **<I>** és **</I>** tagek közé írt szöveg dőltbetűs lesz.

A HTML nyelv elemei – 6

- Hyperlink megadásához használjuk a következő taget:

  <A HREF="URL" egyéb_paramtérek>hyperlink szöveg</A>

  – A tagnek számos paramétert lehet megadni az *egyéb paraméterek* résznél, melyek módosítják a tag viselkedését.

  – Általában a HREF paramétert használjuk az *URL* cím megadására.

  – A *Uniform Resource Locator* feladata, hogy meghatározza egy erőforrás helyét az Interneten.

  – Az URL általános szintakszisa:

  protokoll://gépnév/elérési_út

  - A *protokoll* többféle lehet, például: http, ftp, file.
  - A *gépnév* egy IP cím vagy egy DNS szimbolikus név, amely az erőforrást birtokló hostot adja meg.
  - Az *elérési_út* a könyvtárat és a fájl nevét adja meg, ahol az erőforrás elérhető. Az *elérési_út* érzékeny lehet a kis- és nagybetűkre, ha a web szerver Unixot használó hoston fut. Az *elérési_út* opcionális. Ha nem adjuk meg, akkor a HTML dokumentum alapértelmezett neve Unix alapú szervereknél általában **index.html**.

A HTML nyelv elemei – 7

- Például URL-ekre:

```
http://www.hit.bme.hu/~lencse
ftp://ftp.bme.hu
telnet://users.tilb.sze.hu
file://~lencse/public_html/index.html
mailto: micimacko@szazholdaspagony.hu
```

- Megjegyzés: az URL-nél általánosabb az URI, ami a Uniform Resource Identifier rövidítése. Érdeklődőknek ajánljuk: http://en.wikipedia.org/wiki/Uniform_Resource_Identifier.

A HTML nyelv elemei – 8

- Nézzük meg a következő HTML forrást, amelyben bemutatunk néhány hasznos tag-et:

```
<HTML><HEAD><TITLE>Hasznos tag-ek</TITLE></HEAD>
<BODY> <!-- Ez itt egy megjegyzés. -->
<H1>Bemutatunk néhány tag-et</H1>
<P>Ezt a bekezdést egy bekezdés taggel kezdtük. A
bekezdés tag automatikusan formázza a szöveget
megjelenítéskor.</P>
<BR><!-- Az előző tag sortörést idéz elő a
szövegben.>
<HR><!-- Az előző tag egy vízszintes vonalat húz.>
</BODY></HTML>
```

A HTML nyelv elemei – 9

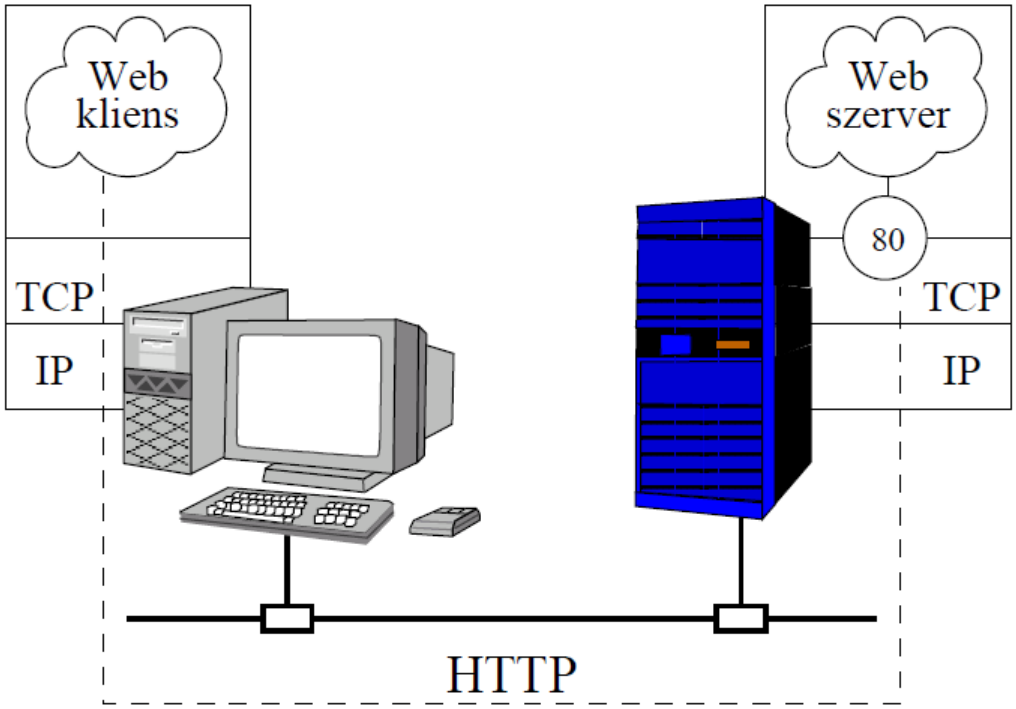
- A HTML lehetőséget ad számozott és számozatlan listák létrehozására, melynek tagjai külön-külön sorban lesznek:

```
<OL> <!-- Számozott lista>
<LI> Lista elem 1
<LI> Lista elem 2

<LI> Lista elem N
</OL>

<UL> <!-- Számozatlan lista>
<LI> Lista elem 1
<LI> Lista elem 2

<LI> Lista elem N
</UL>
```

HTTP/HTTPS

HyperText Transfer Protocol

- A web kliens (böngésző) a HTTP protokoll segítségével kommunikál a web kiszolgálóval.
  - A TCP kapcsolat alapértelmezésben a web szerver 80-as portjára épül ki.
    - Az URL-ben megadható más is, például: **`http://www.szerverem.hu:8080`**



HTTP verziók, parancsok

- Számunka a legfontosabb parancs: GET
  - 1.0 verzió (RFC 1945)

    `GET /index.html HTTP/1.0`
    - A kérésben nincs benne a kiszolgáló
      - Már felépült a TCP kapcsolat, minek is kellene?
  - 1.1 verzió (~~RFC 2068~~, ~~RFC 2616~~, RFC 7230-7235)

    `GET /index.html HTTP/1.1`

    `HOST: ipv6.tilb.sze.hu`
    - A kérésben szerepel a kiszolgáló neve
      - A virtuális webszervereknek szüksége van rá!
  - „2.0” helyett „2” verzió (RFC 7540)
    - Érdeklődőknek: https://http2.github.io/faq/

HTTP szerver válasza (1.1)

- A szerver a HTML dokumentum letöltésére irányuló kérés megérkezése után küld egy HTTP választ.
- A HTTP válasz három részből áll:
  - státusz (response status)
  - fejrész (response header)
  - adat (response data)

HTTP szerver válaszának részei

- A *válasz státusz* egyetlen sor, tartalmazza:
  - a szerver HTTP verziójának számát
  - a státusz kódot
    - megadja a kérés eredményét (ezt könnyű a web böngészőnek értelmeznie),
  - majd az utána következő szöveg emberek számára értelmezi a státusz kód jelentését.
- Példa:

```
HTTP/1.1 200 OK
```

HTTP szerver válaszának részei

- A *válasz fejrész* tartalmazza a következő információkat:
  - a szerver típusa,
  - MIME verziószám (ha használatos)
  - a tartalom típusa
  - további információk
- Egy üres sor választja el a 3. résztől
  - Ami lehet egy HTML fájl, kép, stb.

Példa válasz fejrészre

```
Date: Tue, 01 Sep 2015 11:59:34 GMT
Server: Apache/2.2.16 (Debian)
Last-Modified: Mon, 05 Mar 2012 07:10:59 GMT
ETag: "19a09c-d9-4ba79a0d586c0"
Accept-Ranges: bytes
Content-Length: 217
Vary: Accept-Encoding
Content-Type: text/html
```

TCP kapcsolat sorsa

- HTTP/1.0 esetén bezáródott
  - Minden kérés/válasz párhoz külön TCP kapcsolat
- HTTP/1.1 esetén beállítás kérdése
  - Egy TCP kapcsolaton át több objektumot szoktak letölteni

Egy kis demonstráció

- Egy telnet kliens segítségével jelentkezzünk be egy web szerver 80-as portjára:

```
telnet vip.tilb.sze.hu 80
```

- A kapcsolat létrehozása közben a következőhöz hasonló üzeneteket láthatunk:

```
Trying 193.224.130.171...
```
← ezeket a sorokat a telnet írja ki

```
Connected to vip.tilb.sze.hu
Escape character is
```

A következő lépésben úgy teszünk, mintha mi lennénk a web böngésző (kliens) és kiadjuk a GET parancsot a következő módon:

```
GET / HTTP/1.1
Host: ipv6.tilb.sze.hu
```

Egy kis demonstráció

- Ebben a példában egyben a virtuális webszerverek használatát is bemutatjuk. A DNS rendszerben az **`ipv6.tilb.sze.hu`** szimbolikus név egy CNAME (alias) a **`vip.tilb.sze.hu`** szimbolikus névre. A névfeloldás után az azonos IP-cím miatt a kiszolgáló program csak a Host mező tartalma alapján képes eldönteni, hogy mely oldalt kell küldenie.
- Miután lenyomtuk az Enter billentyűt, nem történik semmi, mivel a HTTP kérést két <CR><LF> karakternek kell követnie.
- Miután másodszor is lenyomjuk az enter billentyűt, megjelenik a HTTP válasz.

Egy kis demonstráció

```
HTTP/1.1 200 OK                                    ← válasz státusz
Date: Tue, 01 Sep 2015 12:12:17 GMT                ← válasz fejrész eleje
Server: Apache/2.2.9 (Debian)
Last-Modified: Tue, 01 Sep 2015 12:01:15 GMT
ETag: "dc48a-117-51eae4e93b8c0"
Accept-Ranges: bytes
Content-Length: 279
Vary: Accept-Encoding
Content-Type: text/html
X-Pad: avoid browser bug                           ← válasz fejrész vége
                                                   ← üres sor (elválasztás)
<HTML>                                             ← válasz adatrész eleje
<HEAD>
   <META charset="utf-8">
   <TITLE>ipv6.tilb.sze.hu</TITLE>
</HEAD>
<BODY>
<H1>IPv6 kutatócsoport</H1>
<P>Az oldal szerkesztés alatt áll. Kérjük, addig tekintse meg
publikációinkat <A
HREF="http://www.hit.bme.hu/~lencse/publications/">itt</A>.</P>
</BODY>
</HTML>                                            ← válasz adatrész vége
Connection closed by foreign host.                 ← ezt a telnet írta ki
```

Virtuális dokumentum fogalma

- Olyan dokumentum, ami nem található meg a webszerveren, hanem a szerver hozza létre
  - Könyvtárlisták (szépen megformázva)
  - Hibaüzenetek

Virtuális dokumentum hibaüzenet

```
GET ezt elrontottuk

HTTP/1.1 400 Bad Request
Date: Tue, 01 Sep 2015 12:35:29 GMT
Server: Apache/2.2.9 (Debian)
Vary: Accept-Encoding
Content-Length: 306
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>400 Bad Request</title>
</head><body>
<h1>Bad Request</h1>
<p>Your browser sent a request that this server could not
understand.<br />
</p>
<hr>
<address>Apache/2.2.9 (Debian) Server at vip.tilb.sze.hu Port
80</address>
</body></html>
Connection closed by foreign host.
```

HTTPS

- A HTTPS a HTTP biztonságos változata
  - TLS (Transport Layer Security) vagy SSL (Secure Socket Layer) fölött működő HTTP.
  - Jelentősége igen nagy, hiszen a mindennapi élet során egyre több tranzakciót hajtunk végre webes felületen keresztül.
    - Például: banki műveletek, rendszerek távoli adminisztrációja.
- Biztonsági kockázatot jelent, hogy az SSL összes verziójában, továbbá a TLS egyre újabb verzióiban is sebezhetőségeket találtak.
  - Érdeklődőknek:
    - https://www.acunetix.com/blog/articles/tls-vulnerabilities-attacks-final-part/
    - https://www.packetlabs.net/tls-1-1-no-longer-secure/
  - Ezért legalább TLS 1.2-t, de ha lehet, TLS 1.3-at használjunk!

HTTPS

- Az ssh/scp-hez hasonlóan a https is nyilvános kulcsú kriptográfián alapul, itt egy (potenciálisan hamis) tanúsítvány elfogadása jelenti a legnagyobb veszélyt!
- További veszélyforrás, ha a böngészőnk nem ellenőrzi a CRL-eket (Certificate Revocation List - a visszavont tanúsítványok listája).

Összefoglalás

- Domain Name System (DNS)
- Távoli elérési protokollok: TELNET, SSH, SCP
- Levelező protokollok: SMTP, POP3, IMAP4, POP3S, IMAP4S
- Fájl átviteli protokoll: FTP
- Rövid HTML ismertető
- Web hozzáférési protokollok: HTTP, HTTPS

Kérdések?

KÖSZÖNÖM A FIGYELMET!

Dr. Lencse Gábor
egyetemi tanár
Széchenyi István Egyetem, Távközlési Tanszék
lencse@sze.hu