Lencse Gábor

HÁLÓZATI ALKALMAZÁSOK

1. kiadás
(1.12 – 2008. 09. 02.)

Széchenyi István Egyetem
Távközlési Tanszék
Győr, 2008.

A SZE Távközlési Tanszék támogatásával készült.

Ez az egyetemi jegyzet Karanjit S. Siyan: "Inside TCP/IP" Third Edition, New Riders Publishing, Idiana, 1997. (Chapter 13.) magyar fordításának felhasználásával készült. A fordítást eredetileg Kismődi Tamás végezte. Dolgozott még az anyag javításán Kapitány Zsolt és Jáger Attila is.

Írta és szerkesztette: **Dr. Lencse Gábor**
egyetemi docens, PhD

Lektorálta: **Varga György**
okleveles villamosmérnök, hálózati rendszergazda

ISBN nincs

Kiadja a SZE Távközlési Tanszék. Felelős kiadó dr. Lencse Gábor.
Műszaki szerkesztő dr. Lencse Gábor.

Tartalomjegyzék

Előszó

Ez a jegyzet a Széchenyi István Egyetem harmadéves *távközlés-informatika szakirány*os villamosmérnök BSc hallgatóinak oktatott *Protokollok és szoftverek* tárgyhoz készült. A tárgy anyagának körülbelül a felét tartalmazza: főleg az egyes hálózati szolgáltatások kliens-szerver közötti kommunikációjával foglalkozik. Az egyes szolgáltatások nyújtásával kapcsolatos ismereteket a tárggyal párhuzamosan oktatott *Hálózati operációs rendszerek* tárgy anyaga tartalmazza. A jegyzet erősen épít az előző félévben oktatott *Számítógép-hálózatok* tárgy anyagára.

Ha valaki hibát fedez fel ebben a jegyzetben, akkor azt a távközlés-informatika szakirány honlapján [4] a tárgynál megtalálható módon tudja bejelenteni.[1] Ugyanott megtalálható az aktuális hibajegyzék elérhetősége is.

A jegyzet tárgykörébe tartozó kérdésekben végső tekintélyt az RFC-k [8] képviselnek, azaz ha a jegyzetben valaki hibát vél felfedezni, akkor ellenőrizze az adott kérdést a vonatkozó RFC-ben. De az RFC-k is folyamatosan változnak, használatuk előtt az RFC indexben [9] érdemes ellenőrizni, hogy nincs-e újabb.

A jegyzetet jelenleg elektronikusan adjuk ki hallgatóinknak, a nyomdai formában való kiadás feltételei pillanatnyilag nem állnak rendelkezésünkre.

[1] A tárgy hallgatói az elsőként bejelentett hibákért valamilyen jutalmazásban részesülnek, ennek módját és mértékét félévenként állapítjuk meg.

A jegyzetet évente tervezzük megújítani, időközben hibajegyzéket adunk ki, de előfordulhat, hogy korábban jelenik meg újabb változat. A tárgy honlapján mindig a legfrissebb, verziószámmal és időbélyeggel ellátott változat található.

Eredményes tanulást kívánok!

Lencse Gábor

Bevezetés

A TCP/IP felett működő hálózati alkalmazások magukban foglalják az OSI 5., 6., és 7. rétegének funkcionalitását. Nem minden alkalmazás igényli az 5. és 6. réteg szolgáltatásait. A hálózati alkalmazások foglalkoznak az alkalmazási réteg adatainak formázásával, küldésével és fogadásával. Azonban nem tartalmazzák a felhasználó felé nyújtott kezelői felületet.

Ez a jegyzet a következő főbb hálózati alkalmazásokat tárgyalja:

- Domain Name System (DNS)
- Levelező protokollok: SMTP, POP3, IMAP4, POP3S, IMAP4S
- Távoli elérési protokollok: telnet, Berkeley r*, ssh, scp
- Fájl átviteli protokollok: FTP, TFTP
- Fájl hozzáférési protokollok: NFS, Web NFS, SMB
- Web hozzáférési protokollok: HTTP, HTTPS

Bár nem hálózati protokoll, de a tárgyban oktatjuk a HTML alapjait is, erről is van egy rövid bevezető. További protokollokról és haladó web programozásról a tárgy honlapján további segédanyagok találhatók.

1. fejezet

Domain Name System

A DNS leírása megtalálható az RFC 1034 (STD 13) "Domain Names – Concepts and Facilities"-ben.[1] Egy jó szakkönyv a témában: [1].

1.1. A DNS feladata

A hálózatba kötött számítógépeket egyedi azonosítóval (IP cím) kell ellátni. A felhasználók – lévén emberek – inkább valamilyen számukra könnyebben megjegyezhető, esetleg többlet információt hordozó elnevezéseket tudnak jól megjegyezni, így bevezették a szimbolikus neveket (*domain name*). Például a www.tilb.sze.hu szimbolikus név a Magyarországon (hu) működő Széchenyi István Egyetem (sze) Távközlés-Informatika Laborjának (tilb) webszerverére (www) utal.

Az Internet fejlődésének kezdeti szakaszában az IP cím –

[1] Ezenkívül számos más RFC is létezik DNS témában, egy részük bizonyos részleteket definiál (például RFC 1035 "Domain Names – implementation and specification), mások pedig kiterjesztések (például RFC 2137 "Domain Name System Dynamic Update" és RFC 2136 "Dynamic Updates in the Domain Name System (DNS UPDATE)").

szimbolikus név párokat egyetlen fájlban tárolták, amit aztán megfelelő rendszerességgel le kellett tölteni. Kis gépszám esetén még működött is ez a megoldás, de a rohamos fejlődés eredményeként a folyamatosan változó fájlt már nem lehetett állandóan töltögetni. Ezért egy elosztott adatbázist hoztak létre, amelynek egyes részeit ott tárolják, ahol az információ keletkezik.

A modern Internetben a számítógépek, más néven *host*ok, a Domain Name System (DNS) mechanizmusa alapján kapnak nevet. A DNS az IP cím mellé rendel tehát egy szimbolikus nevet, melyen az elérhető. A DNS-t közvetlenül használja majdnem minden hálózati alkalmazás, mert a felhasználók tipikusan úgy hivatkoznak a host nevekre, mint a DNS nevekre. Például, ha egy felhasználó egy telnet kapcsolatot szeretne létrehozni, a következő parancsot adja ki:

```
$ telnet whale.hit.bme.hu
```

A telnet kliens válasza a következő üzenet:

```
Trying 152.66.248.88...
Connected to whale.hit.bme.hu.
Escape character is '^]'.
```

A telnet kliens lefordította a host nevet `whale.hit.bme.hu` egy 32-bites IP címre (152.66.248.88). Ezt a fordítást hajtotta végre a DNS.

A DNS elsődleges feladata tehát a szimbolikus névhez tartozó IP cím megadása. Bizonyos esetekben szükség van a fordított irányú leképzésre is, hogy az IP címhez találjuk meg a szimbolikus nevet (reverse mapping). Ezenkívül a DNS fontos szerepet játszik még a levelező protokolloknál is, amikor egy e-mail cím megfelelő részéből meg kell találnia az illetékes levelező kiszolgálót.

Felső szintű Domain	Megnevezés
COM	Gazdálkodó/üzleti szervezet
EDU	Oktatási intézmények
MIL	Katonaság
GOV	U.S. közigazgatás
NET	Hálózat ellátó
ORG	Non-profit szervezet
ARPA	ARPANET
INT	Nemzetközi szervezet
HU	Ország: Magyarország
US	Ország: Egyesült Államok
UK	Ország: Egyesült Királyság
DE	Ország: Németország

1.1. táblázat. Néhány példa felső szintű domain-re (TLD)

1.2. A DNS nevek hirearchikus rendszere

1.2.1. A domain nevek felépítése

Amint láttuk, egy szimbolikus név több, egymástól ponttal elválasztott részből (*label*) áll. Az ilyen névhasználat egy egyezményes megállapodásból született. A hierarchikus névrendszerben, amelyet a DNS használ, a név egy hierarchikus fában helyezkedik el. A fa legtetején áll a root domain, melynek a neve a pont szimbólum („."). Minden név ezen közös root alá tartozik, de ezt a pontot általában elrejtjük, ha hierarchikus nevet adunk meg a hálózati alkalmazásokban. A felső szintű domainekre példák láthatók az 1.1. táblázatban. Az ISO 3166 szabvány szerint az országra utaló kétbetűs megnevezést használjuk államok esetében, kivétel a United Kingdom, ahol az ISO szabvány szerinti GB helyett DNS-nél a `.uk`-t használják. Ezeket

ccTLD-knek (coutry code Top Level Domain) nevezzük, szemben a gTLD-kkel (generic Top Level Domain), amelyek közül néhányat csak az USA-ban (például: gov), másokat az egész világon használnak (például: com). Egy TLD-n belül találhatók a közép szintű domain-ek, melyek azonos felső szintű domain-nel rendelkeznek. Példa:

```
rs1.sze.hu
```

Az `rs1.sze.hu` névben a host (lokális) neve az `rs1`, amely a `sze.hu` domain-ben van. Ha egy másik host neve `neptun` és ugyancsak a `sze.hu` a domain-ben helyezkedik el, akkor a *Fully Qualified Domain Name* (FQDN) a következő:

```
neptun.sze.hu.
```

A fenti címben a `neptun` a relatív név, míg a `neptun.sze.hu.` az abszolút név, amit a végén található pont („.”) jelez.[2]

1.2.2. A domain nevek helyesírása

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, betűket („a”-„z”), számjegyeket („0”-„9”) és kötőjelet („-”) tartalmazhat, de az utóbbi csak a belsejében fordulhat elő, a határán nem.[3] Az FQDN maximális hossza nem haladhatja meg a 255 karaktert.

[2]Bizonyos alkalmazások esetén (a köznapi internet használtban nagyon gyakran) a záró pontot elhagyhatjuk. Vannak azonban olyan esetek, amikor a pont elhagyása a helyi domainnek a névhez való automatikus hozzáírását eredményezi!

[3]Lektori megjegyzés: Egyes Microsoft szoftverek a fent felsorolt megengedett karaktereken kívül még az aláhúzásjelet („_”) is használják.

1.2.3. Subdomain-ek létrehozása

Az FQDN középső része utal a szervezetre (és utalhat a szervezeten belüli egyéb szerkezetre, rendszerre is). Egy szervezet szabadon definiálhat *subdomain*-eket a szervezeten belül, de ha megteszi, akkor fel kell töltenie a hozzá tartozó domain név szerver adatbázisát, hogy az végre tudja hajtani a névfeloldásokat. Példaként tekintsük a BME-t, melynek a domain neve a következő:

`bme.hu`

Ha ennek az egyetemnek különálló hálózatai vannak, például a *Híradástechnikai*, a *Távközlési és Médiainformatikai* valamint az *Irányítástechnikai és Informatika* tanszékeken, akkor definiálhat subdomain-eket mint a `hit`, `tmit`, `iit`, és ezt az információt továbbítja a BME DNS szerverének, hogy el tudják végezni a névfeloldást, akkor a következő neveken lehet elérni az egyetem tanszékeit:

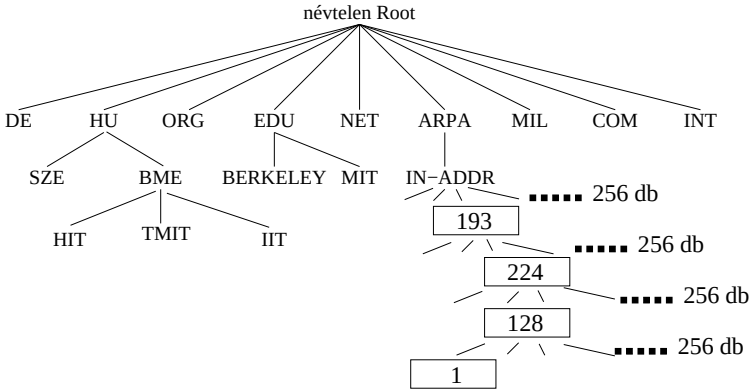
```
hit.bme.hu
tmit.bme.hu
iit.bme.hu
```

Nem szükséges minden domain-hez egy DNS szervert alkalmazni, egy közös szerver elláthat több domain-t is, de természetesen megtehető az is, hogy például BME példájában a tanszékek külön névkiszolgálót üzemeltetnek.

A 1.1. ábra a névhierarchiát mutatja. Az `in-addr.arpa.` alatti részfa a visszafele irányuló leképzéshez (reverse mapping) szükséges, erre később visszatérünk.

1.2.4. Root DNS szerverek

Számos névkiszolgáló kezeli a domain neveket a root domainen. Korábban logikailag 13 szerverre volt lehetőség, mert a



1.1. ábra. Hierarchikus nevek a DNS-ben

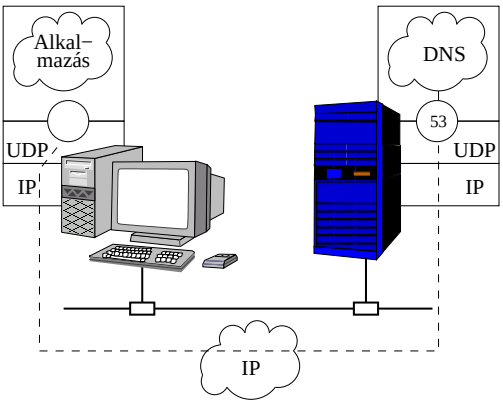
`root.hints` fájlnak bele kellett férnie egy minden host által kötelezően támogatott méretű UDP csomagba (a DNS üzenet számára 512 oktet állt rendelkezésre), valójában azonban némelyik szerver IP címe anycast cím, így fizikailag ennél több szerver használta volt lehetséges. A csomagméret problémát az RFC 2671 megoldotta. Ez azért is fontos, mert 2008. február óta némelyik szerver már IPv6 címen is elérhető, ehhez mindenképpen szükség volt a nagyobb csomagméretre. A root domain névkiszolgálit ma A-tól M-ig terjedő betűkkel jelöljük, és a nevük úgy néz ki, hogy: `a.root-servers-net`, `b.root-servers.net`, ..., `m.root-servers.net`. Korábban az üzemeltetőjük névtartományából (domain) származó nevük volt. Az 1.2. táblázatban látható néhány a *root domain name server*ek közül.

1.3. A DNS működése

A TCP/IP alkalmazások beállíthatók oly módon, hogy használják a DNS névfeloldást. Amennyiben egy ilyen TCP/IP alkal-

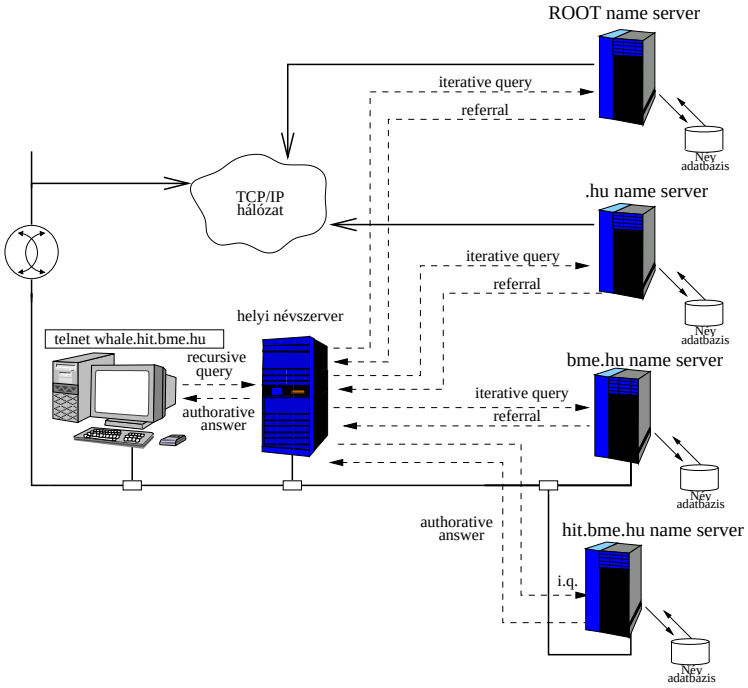
Új név	Régi név	IP cím
a.root-servers.net	ns.internic.net	198.41.0.4
b.root-servers.net	ns1.isi.edu	192.228.79.201
c.root-servers.net	c.psi.net	192.33.4.12
d.root-servers.net	terp.umd.edu	128.8.10.90
e.root-servers.net	ns.nasa.gov	192.203.230.10

1.2. táblázat. Root DNS szerverek



1.2. ábra. DNS kliens-szerver kommunikáció

mazás találkozik egy host névvel, akkor megkéri a *névfeloldó*t (name resolver), hogy adja meg a szimbolikus névhez tartozó IP címet. A névfeloldó egy könyvtári függvény, ami az alkalmazás része (Unix alatt C nyelvben a `gethostbyname()` függvény). Ezt a függvényt hívja meg az adott alkalmazás. A gépen lévő beállítások figyelembevételével megtörténik a névfeloldás. Legyen például Linux alatt a `/etc/nsswitch.conf` fájlban az alábbi bejegyzés:



1.3. ábra. DNS névfeloldási példa

`hosts: files, dns`

Ebben az esetben először a `/etc/hosts` fájlban keres, ahol össze vannak rendelve az IP címek és szimbolikus nevek. (Tipikusan csak néhány darab, leginkább csak a gép saját interfészének a címei.) Először ebben keresi az adott névhez az IP címet. Ha nem találja, akkor az `/etc/resolv.conf` fájlból kiolvassa a névkiszolgáló (name server) IP címét.[4] A meghívott névfeloldó helyi függvény ilyenkor a névkiszolgálóhoz fordul, hogy adja meg

[4]Ilyen több is lehet. Ha az első (primary) valamiért nem válaszol, akkor a másodikhoz (secondary) fordul, és így tovább.

például a kérdéses szimbolikus névhez tartozó IP címet. Ez mutatja be az 1.2. ábra. Kövessük nyomon az 1.3. ábra példáján, hogy mi is történik ezután! (A `whale.hit.bme.hu` gép IP címét derítjük ki.) Azt a kérdés-felelet párost, amivel a name resolver megszólítja a helyi névkiszolgálót *recursive query*nek hívjuk. A recursive query azt jelenti, hogy a megkérdezett (helyi) névkiszolgáló köteles végső választ adni a kliens kérdésére.[5] Közben esetleg más névkiszolgálókat is meg kell szólítania. Ehhez a helyi névkiszolgálón el kell helyezni egy tipp fájlt (`root.hints`). Ebben a fájlban root name server IP címek találhatók, de nem feltétlenül mindegyik up-to-date, hanem csak tippek. A névkiszolgáló ezek közül az első elérhető szervertől lekéri az aktuális root server listát. Ebből a listából aztán kideríthető próbálkozással, hogy melyiknek a legkisebb a válaszideje (RTT – Round Trip Time: a teljes oda-vissza út ideje és a válasz előállításának együttes ideje). Ez a root névkiszolgáló van a „legközelebb" időben, ezért utána hozzá fog fordulni. Az 1.3. ábrán látható, hogy a megkérdezett helyi name server *iterative query*-vel fordul a kiválasztott root name serverhez, ami egy *referral*lal válaszol, hogy a példánkban a `whale.hit.bme.hu` névből a `hu` végződésért melyik szerver felelős. Ezután újabb kérdés most már ehhez, hogy a `bme.hu`-ért ki a felelős, és így tovább, míg végül a helyi névkiszolgáló a `hit.bme.hu` zónáért felelős szervertől megtudja egy *authoritative answer*rel a `whale.hit.bme.hu` IP címét, és visszaküldi ezt az őt megkérdező kliensnek.

A DNS kérdés/válasz típusú kommunikációt követ és UDP-t használ, mint szállítási protokollt. Az UDP sokkal célszerűbb a rövid kérdés/válaszon alapuló alkalmazásoknak, mert nincs kapcsolatfenntartási vezérlés az adatátvitelben (egy TCP kapcsolat felépítése 3, lebontása 4 üzenet lenne), és válasz hiányában leg-

[5]Bizonyos esetben a recursive query visszautasítható, a lényeg azon van, hogy ha válaszol, akkor teljes választ kell adnia, szemben a később említendő *iterative query*re adandó *referral*lal.

feljebb újra kérdez. Két névszerver közötti zónafájl csere (mivel az nem fér bele egy UDP csomagba), TCP protokoll fölött történik.

A DNS legszélesebb körben használt megvalósítása a Berkeley Internet Domain Name Server (BIND), amely eredetileg a BSD Unix-ban volt elérhető. Most elérhető a legfontosabb Unix platformokon. A Unix rendszerekben a BIND-ot megvalósító program neve: *named* (name daemon).

1.4. Reverse mapping

Míg a DNS szimbolikus névből képez le IP címet, addig a *reverse mapping* ennek a fordítottját végzi, IP címhez rendel szimbolikus nevet. Ehhez a leképzés tervezői akár létrehozhattak volna egy másik rendszert, de nyilvánvalóan célszerűbb volt a meglevő DNS-t felhasználni. Sőt, még új névfát sem kellett létrehozniuk! A leképzés alapelve a következő: az 1.1. ábrán látható fa egy pontját kinevezzük egy másik fa gyökerének. Ez a pont az `in-addr.arpa.` ág.[6] Az egyszerűség kedvéért induljon innen 256 ág és azok mindegyikén szintén 256 ág. Ez történjen így összesen 4 szinten, mivel egy IP cím négy oktetből áll. Tekintsük a 193.224.128.1 IP címet. Az `in-addr` csomóponban válasszuk ki a 193-at, ott a 224-et, ott a 128, ott pedig az 1-et! Itt tároljuk el a hozzá tartozó szimbolikus nevet (`rs1.sze.hu.`). Vegyük észre, hogy amit így a névfában bejártunk, azt szimbolikus névként a levéltől a gyökér felé kiolvasva a következőt kapjuk: `1.128.224.193.in-addr.arpa.`, amiben az IP cím oktetjei éppen fordított sorrendben leírva szerepelnek!

Persze a valóságban az alhálózatokra bontás nem 8 bites határokon történik, így a részfa nem lesz ennyire szabályos.

[6] IPv6 esetén pedig az `ip6.arpa.` ág.

1.5. Caching

A névszerverek tárolják a már lekérdezett IP címeket, és újabb kéréseknél felhasználják. Ez a közbenső eredményekre (adott zónáért felelős névkiszolgáló IP címe) és a végeredményre (a kérdésés szimbolikus névhez tartozó IP cím) egyaránt vonatkozik, sőt, a negatív eredményeket (adott szimbolikus névhez nem tartozik IP cím) is tárolják. A tárolás legfeljebb egy lejárati időig (TTL: Time To Live) történhet, amit az információ gazdája (az adott zónáért felelős névkiszolgáló) az információval együtt szolgáltat.

1.6. Domain név regisztráció

A domain nevek megvásárolhatók. A különböző TLD-kre *registry*k felügyelnek, ami hatósági jogkört gyakorlását jelenti. Ezek által feljogosított *registrar*ok foglalkoznak a névbejegyzésekkel anyagi ellenszolgáltatás fejében. Magyarországon a beadott domain név igények 14 napig várólistára kerülnek, és ha ezen idő alatt nem jelentkezik nagyobb prioritású felhasználó, regisztrálható a név. További információ: [6].

2. fejezet

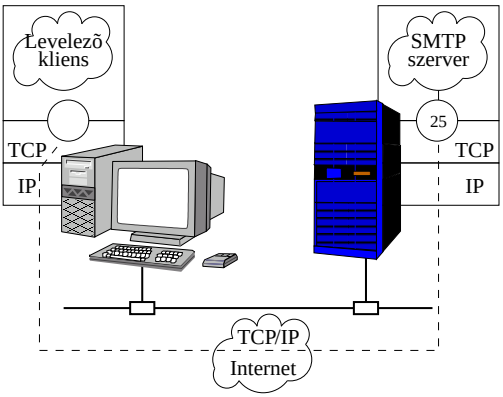
Levelező protokollok

Az elektronikus levelezés egyik legelterjedtebb hálózati alkalmazás. Igen sokféle protokoll létezik az elektronikus levelezés megvalósítására, azonban a Simple Mail Transfer Protocol (SMTP) az egyik legelterjedtebb. Kifejezetten levél küldésre szolgál. A mobil és személyi számítógépen dolgozó felhasználók nagy száma miatt kifejlesztettek további protokollokat is, úgymint a POP3 (Post Office Protocol Version 3), ami csak a levelek letöltését teszi lehetővé. Az IMAP4 (Internet Message Access Protocol Version 4) szintén a levelek letöltésére szolgál. Ezeken kívül még kifejlesztették az utolsó kettő biztonságos verzióját a POP3S-t és az IMAP4S-t. A következőkben ezekkel foglalkozunk.

2.1. Simple Mail Transfer Protocol

Az SMTP ASCII kódú szöveges üzenetek továbbítására képes TCP/IP protokollt használó hostok között, ha azok levelezésre is konfigurálva vannak.

2.1.1. A levél átvitelének folyamata



2.1. ábra. Az SMTP kliens-szerver architektúra.

Amikor a felhasználó kezdeményezi egy levél küldését, a *User Agent* (felhasználói ügynök, a felhasználó által használt „levelező program", például Thunderbird) kapcsolatba lép a kimenő SMTP szerverrel (Unix esetén például postfix) és átadja neki a levelet. A 2.1. ábrán a kliens szerver kapcsolat látható. A szerver oldalon a kapcsolat a 25-ös porton épül ki, míg kliens oldalon ez nem meghatározott előre (az operációs rendszer dinamikusan oszt ki egy szabad protot). Kövessük nyomon a 2.2. ábrán, hogy mi történik ezután! A kimenő SMTP szerver a levelet egy kimenő postafiókban tárolja, majd amint rákerül a sor, megkísérli annak átvitelét, azaz az e-mail cím alapján a DNS által szolgáltatott információt felhasználva (MX record) TCP kapcsolatot épít ki a címzett leveleit kezelő SMTP szerverrel és megtörténik az átvitel. Ha ez nem sikerül, akkor egy kimenő postafiókban tárolja a levelet, és bizonyos időkö-

zönként újra megkísérli az átvitelt.[1] A vevő oldalon az SMTP process fogadja a kapcsolódást és veszi az üzenetet, így bekerül a levél a címzett bejövő postaládájába. Ha a cél hoston nem létezik az e-mail címben megadott postaláda, akkor a feladó erről a saját kimenő SMTP szerverétől levélben értesítést kap. Mind a kimenő, mint a fogadó SMTP szervert, amelyek a levelek átviteléért felelnek, *Message Transfer Agent*nek (üzenetátviteli ügynök) nevezik. Vegyük észre a 2.2. ábrán, hogy ugyanazt az SMTP protokollt használják a user agent és a kimenő SMTP szerver (küldő MTA) valamint a küldő és a fogadó MTA-k között! Ma viszont az elharapózó *spam* (kéretlen levélküldés) miatt a UA és az küldő MTA között az úgynevezett *Message Submission Protocol* (RFC 2476) szerinti üzenetváltás zajlik, ami az SMTP-hez nagyon hasonlít, de a 25-ös helyett az 587-es porton kommunikál és megköveteli a kliens azonosítását.

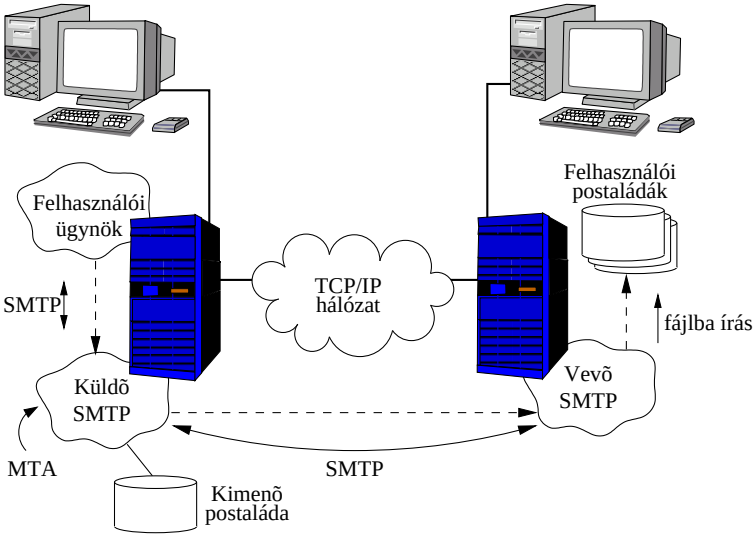
Az SMTP üzenetformátumát eredetileg az RFC 822-ben definiálták, ma a releváns a RFC a 2822-es. A levelek fejrészét gyakran hívják 822-es (vagy 2822-es) fejrésznek. Egy példa a levélcímre:

```
lencse@rs1.sze.hu
```

A @ jel előtti szöveg megadja a postaláda nevét (`lencse`), a @ jel utáni szöveg pedig egyszerű esetben a host FQDN nevét (`rs1.sze.hu`). Az utóbbi azonban nem szükségszerű. A @ jel utáni rész alapján az SMTP a DNS segítségével megkeresi az adott zónához tartozó levelező kiszolgálót.[2]

[1] További sikertelenség esetén bizonyos idő (néhány óra) után a feladónak figyelmeztető üzenetet (e-mail) küld, de még tovább próbálkozik. Ha néhány nap múlva végül feladni kényszerül, arról is e-mailt küld a feladónak.

[2] Konkrétan az *MX record* adja meg @ jel utáni rész, mint zóna *mail exchanger*ét.



2.2. ábra. SMTP levelezés

2.1.2. Nem ASCII-ben kódolt információ átvitele

Ha nem ASCII kódú szöveges üzenetet akarunk küldeni SMTP-n keresztül, hanem bináris fájlt, hangot, akkor megfelelő kódolási eljárást kell alkalmaznunk, amely az átviendő információt ASCII kódokká alakítja.

A klasszikus eljárás Unix alatt a `uuencode`. Az így kódolt üzenet visszaalakítására a `uudecode` parancs használható.

Ha például a `virag.jpg`-t szeretnénk levélben elküldeni, akkor először elkódoljuk.

```
uuencode viragocska.jpg < virag.jpg > virag.jpg.uu
```

Majd az elkódolt fájlt elküldjük:

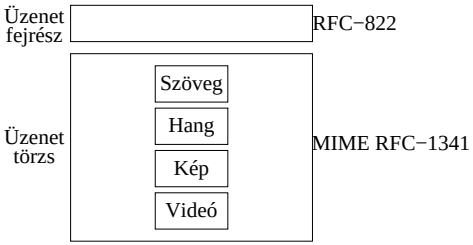
```
mail lencse@rs1.sze.hu < virag.jpg.uu
```

Ha a címzett a levél megfelelő részét `virag.jpg.uu` néven menti el, akkor a dekódolás:

```
uudecode virag.jpg.uu
```

Ennek eredményeként megkapja a `viragocska.jpg` nevű fájlt, aminek a tartalma megegyezik az eredeti `virag.jpg` nevű fájl tartalmával.

Egy másik megoldás (ilyenkor is szöveges üzenetet küldünk SMTP-n keresztül) a MIME (Multipurpose Internet Mail Extensions) protokoll. A MIME megtalálható az RFC 1896, RFC 2045, RFC 2046 és az RFC 2049-ben. A MIME képes különböző típusú adatokat is kódolni, mint egyszerű szöveg, gazdagabban formázott szöveg (rich text), kép, hang, videó, HTML dokumentum és így tovább. A MIME megadja a tartalom típusát (text/plain, application/pdf) is és az átvitelnél alkalmazott kódolást (quoted-printable, base64) is.



2.3. ábra. MIME üzenet

A MIME üzenet törzse részekre osztott tartalommal rendelkezik, és a MIME felhasználói ügynök válogathat a megjelenítendő adatok közül. Például egy „dumb” (buta) terminál, amely nem képes sem hang lejátszásra, sem kép, sem videó megjelenítésre, az üzenetnek csak a szöveges részét írja ki a képernyőre. A MIME egy másik hasznos tulajdonsága, hogy használhat mu-

tatót olyan adatra, amely valahol máshol került tárolásra. Például ez a mutató megadhat egy FTP helyet, ahonnan letölthető az adott dokumentum. Ez a megoldás megszünteti annak a szükségességét, hogy egy körlevélben mindenkinek elküldjük az adott dokumentumot, csak annak kell megvárnia a letöltést, akit érdekel az információ.

2.1.3. Az SMTP protokoll

A 2.1. táblázatban láthatók az SMTP parancsok, amelyekkel üzenetet küldhetünk. Minden SMTP parancs (vagy annak első tagja) négy karakter hosszú. Az SMTP szerver az SMTP parancsok fogadása után egy három számjegyes állapot kóddal és egy szöveges üzenettel válaszol a következő formátumban:

```
nnn Szöveges üzenet
```

Ahol az *nnn* a három számjegyből álló állapotkód, ami az SMTP programnak szól. Az első számjegy az eredmény (hiba) jellegét adja meg, például: 2xx: pozitív eredmény, 4xx: tranziens hiba, 5xx: permanens hiba, stb. A számjegyeket követő szöveges üzenet az emberi értelmezést szolgálja. A lehetséges válaszok megtalálhatók a 2.2. táblázatban.

A 2.3. táblázatban bemutatunk egy példát arra, hogy SMTP parancsokkal hogyan lehet levelet küldeni. (K jelöli a küldőt, V pedig a vevő felet.)

Ezt a levelet a `lencse@rs1.sze.hu` postaládacímre a szerver kézbesítésre elfogadta, mivel a levél szerver válasza `OK` volt.

Azonban a helyi `kazmer@users.tilb.sze.hu` címre nem fogadta el, mert hibaüzenet volt a válasz (550-es állapotkód) azaz a postaláda nem elérhető.

A levél fejlécében megjelenő *From:*, *To:*, *Subject:* mezőket és a levél törzsét a DATA fázisban küldi át az SMTP kliens. (A fejrész és a törzs között egy üres sor az elválasztó jel.) A *MAIL*

Parancs	Jelentés
HELO *küldő*	A küldő gép, amelyen a UA 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 végén a következőnek kell állnia `<CRLF>.<CRLF>`.
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.

2.1. táblázat. SMTP parancsok (kliens, vagy küldő MTA)

FROM: és az *RCPT TO:* az úgynevezett *envelope* (boríték), amit az SMTP protokoll használ a kézbesítéshez.

Az SMTP-ről szóló dokumentumok felsorolása a 2.4. táblázatban látható.

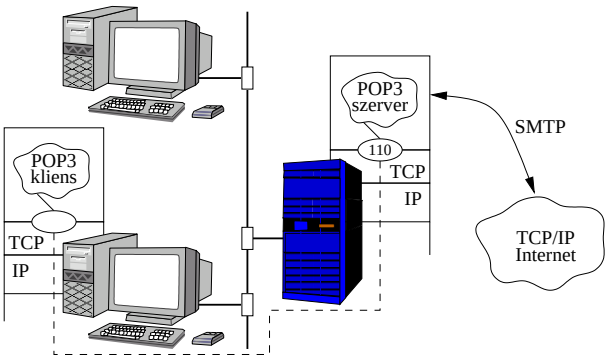
2.2. Post Office Protocol Version 3

Az SMTP protokoll elvárja, hogy a levelet fogadó mail szerver on-line (bekapcsolt és a hálózaton elérhető) legyen, különben a TCP kapcsolat nem hozható létre. Éppen ezért nem praktikus asztali számítógépeknél kizárólag SMTP alapokra bízni a levelezést, mivel az asztali gépeket munka után ki szokták kapcsolni.

Sok hálózati környezetben az SMTP levelek fogadását egy

Kód	Jelentés
250	Kért levél művelet `OK`, sikeresen befejeződött.
251	A felhasználó nem helyi, továbbításra kerül a megadott útvonalra. <forward-path>
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.
451	Hiba a kért műveletben.
551	A felhasználó nem helyi, kérlek add meg az útvonalat. <forward-path>
452	Kért levél művelet nem történt meg. Nincs elég szabad hely.
553	Kért levél művelet nem történt meg. A postaláda neve nem elérhető. Például szintaktikailag hibás.
354	Kezdje a levelet. Befejezés a `<CRLF>.<CRLF>`.
554	Művelet nem sikerült

2.2. táblázat. SMTP válaszok (szerver vagy fogadó MTA)



2.4. ábra. POP3 kliens/szerver architektúra

olyan SMTP host végzi, amely mindig be van kapcsolva. Ez az SMTP host látja el a postafiók (mail-drop) szolgálatot. A

```
K:  HELO users.tilb.sze.hu
V:  HELO users.tilb.sze.hu, Pleased to meet you

K: MAIL FROM: jampy@users.tilb.sze.hu
V: 250 OK

K: RCPT To: lencse@rs1.sze.hu
V: 250 OK

K: RCPT To: kazmer@users.tilb.sze.hu
V: 550 No such user here

K: DATA
V: 354 Start mail input; end with <CRLF>.<CRLF>
K: From: telapo@example.com
K: To: ovodasok@example.net
K: Subject: ajandek
K:
K: üzenet szövege
K: <CRLF>.<CRLF>
V: 250 OK
```

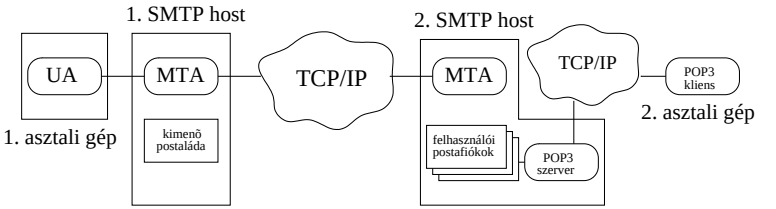
2.3. táblázat. Levél küldése SMTP parancsokkal

Protokoll	Név	RFC#
SMTP	Simple Mail Transfer Protocol	2821
SMTP-SIZE	SMTP Service Extension for Message Size Declaration	1870
SMTP-EXT	SMTP Service Extensions (ESMTP)	1869
MAIL	Internet Message Format	2822

2.4. táblázat. SMTP dokumentumok

munkaállomások kapcsolatba lépnek az SMTP hosttal, majd letöltik az üzeneteket egy kliens/szerver levelező protokoll-lal,

mint például a POP3 (Post Office Protocol version 3), melynek leírása megtalálható az RFC 1939-ben. A POP3 a TCP szállítási protokollt használja, és a POP3 szerver a szabványos 110-es TCP porton érhető el. A POP3 protokoll kliens-szerver architektúrája a a 2.4. ábrán látható.



2.5. ábra. Asztali gépen dolgozó felhasználók közti levelezés

Bár az üzenetek letöltéséhez a POP3-at használjuk, a munkaállomás felhasználója továbbra is az SMTP levél szervert használja az üzenetek elküldéséhez. A 2.5. ábrán látható, hogy hol van az egyes protokollok helye az asztali gépen dolgozó felhasználók közti levelezésben. Természetesen a felhasználói levelező program mind a User Agent, mint a POP3 kliens funkciót ellátja.

A 2.5. táblázatban a POP3 protokoll kötelező parancsai láthatók, melyeknek minden implementációban szerepelniük kell. A 2.6. táblázatban látható parancsok opcionálisak. Bár a USER és a PASS parancsok opcionális parancsok az RFC 1939-ben, de általában támogatottak. A USER/PASS parancsok azért opcionálisak, mert van egy másik lehetőség: az APOP parancs, amely az MD5 (Message Digest version 5) hitelesítő eljárást használja.

A 2.7. táblázatban látható egy példa a POP3 szerver és a POP3 kliens közötti kapcsolatra. (K jelöli a klienst, SZ pedig a szervert.) A példában látható, hogy a POP3 kapcsolat kezdeti részében belépünk egy *kapcsolódási állapotba*. A kap-

Parancs	Jelentés
STAT	Erre a parancsra kapunk egy választ, amely egy +OK stringgel kezdődik, majd egy szóköz után az üzenetek száma és még egy szóköz után a levelek összmérete látható oktetekben megadva.
LIST [üzenet_sorszám]	Ha megadunk üzenet sorszámot, akkor a parancs hatására a szerver válaszképpen kiírja a kért üzenet azonosító mellé a méretét oktetekben. Ha nem adunk meg sorszámot, akkor egy többsoros válasz kapunk, melynek első sora egy +OK stringgel kezdődik, majd egy szóköz után az üzenetek száma és még egy szóköz után a levelek mérete látható oktetekben. A következő sorok első karaktere az üzenet sorszám és szóköz után 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 oktetekben. Ezután következik maga az üzenet (az, amelyiknek a sorszámát megadtuk). Ha olyan azonosítót adunk meg amely nem létezik egy -ERR üzenetet küld a szerver.
DELE <üzenet_sorszám>	A parancs az adott üzenetet törölésre kijelöli.
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 és a POP3 szerver működik-e.
RSET	Ez a parancs törli az összes törlésre való kijelölést. A szerver visszaad egy +OK string-et.
QUIT	A parancs hatására a szerver törli az összes törlésre kijelölt levelet, és az elvégzett művelettől függően +OK vagy -ERR string-et ad vissza, majd lezárja a kizárólagos elérést a mail-drop-on és lebontja a TCP kapcsolatot.

2.5. táblázat. A kötelező POP3 parancsok

Parancs	Jelentés
USER <név>	Ezzel a paranccsal adható meg a kezelni kívánt postaláda.
PASS <string>	Ez a parancs adja meg a szerver/postaláda-specifikus jelszót a felhasználóhoz.
TOP <üzenet_sorszám> <n>	A parancs hatására a szerver válaszol egy +OK string-et, 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, 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.

2.6. táblázat. Opcionális POP3 parancsok

csolódási állapotban létrehozzuk a TCP kapcsolatot a POP3 szerverrel. A következő lépésben a POP3 kapcsolat belép a *hitelesítési állapotba*. Ebben az állapotban a felhasználónak meg kell adnia a felhasználói nevét illetve a jelszavát, hogy azt hitelesíthesse a szerver. A korábbi POP3 implementációkban a felhasználói név és jelszó információkat nyílt szövegként továbbították, ami azt jelenti, hogy ha valaki megszerezte a csomagokat, kiolvashatta belőlük a felhasználói név/jelszó kombinációkat. Biztonságosabb alternatívát nyújt az RFC 1939-ben leírt MD5-ös hitelesítési folyamat. Miután a felhasználót hitelesítette a szerver, a POP3 kapcsolat átlép a *tranzakciós állapotba*. Ebben az állapotban számos parancs kiadására van lehetőségünk - mint például a STAT, LIST, RETR, DELE, RSET és

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

2.7. táblázat. POP3 szerver és a POP3 kliens közötti kapcsolat

így tovább. A 2.7. táblázatban a POP3 kliens kiad egy STAT parancsot, amire válaszul megkapta, hogy 8 üzenete érkezett, és azok együttes mérete 153681 oktet. A POP3 kliens ezután a LIST parancsot használta, hogy lekérdezze az üzenetek listáját. A szerver válaszképpen megadta az üzenet azonosító száma mellé az adott üzenet méretét is. Ezután a kliens a RETR parancsot használta az üzenet azonosítóval, hogy letöltse a levelét. A POP3 kliens beállításaitól függően a kliens letörölheti a letöltött üzenetet a szerverről. Ha a POP3 kliens kiadja a QUIT parancsot, a POP3 kapcsolat átlép a *frissítési állapot*ba. Ebben az állapotban mind a POP3 kliens, mind a POP3 szerver frissíti a belső állapotait, hogy rögzítse, mennyi üzenet van a postaládáikban. Végül a TCP kapcsolat bezárul.

2.3. Internet Message Access Protocol V. 4

A POP3 egy jó kliens/szerver protokoll a leveleink letöltéséhez, azonban néhány esetben ez kevés lehet. Például POP3-on keresztül nem vizsgálhatjuk meg a leveleinket, mielőtt letöltöttük volna azokat, és a POP3 nem teszi lehetővé a levelekkel (azok részeivel) való közvetlen műveletvégzést a szerveren. Tehát, ha ilyen feladatokat szeretnénk megoldani, akkor az IMAP4-et célszerű használnunk a POP3 helyett.

Az Internet Message Access Protocol Version 4 (revision 1) egy kliens/szerver protokoll, mellyel elérhetjük és szerkeszthetjük elektronikus leveleinket a szerveren. A protokoll lehetővé teszi, hogy a távoli postafiókon elvégezhessük ugyanazokat a műveleteket, amelyeket a helyi postaládánkon el tudunk végezni. Az IMAP4 továbbá támogatja a kapcsolat nélküli munkát, és lehetőséget biztosít az újraszinkronizálásra a szerverrel kapcsolatfelvétel esetén.

Egy IMAP4 kliens szolgáltatásai lehetővé teszik:

- Az e-mail-ek elérése és szerkesztési lehetősége a szerveren

anélkül, hogy letöltenénk őket

- Levelek és csatolt fájlok áttekintése anélkül, hogy letöltenénk őket
- Levelek letöltése kapcsolat nélküli munkához
- Szinkronizálás a helyi és a szerveren lévő postaládák között

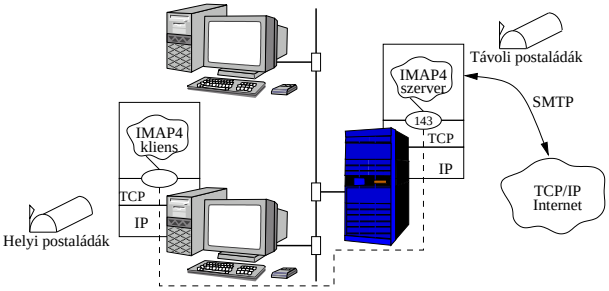
Az IMAP4 műveletei között szerepelnek:

- 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)
- Keresés és szelektív letöltés az üzenet tulajdonságaiból, szövegéből, illetve annak részeiből

Az üzenetek eléréshez az IMAP4-ben számokat használunk. Ezek a számok vagy az üzenetek sorszámai vagy az egyedi azonosítói. Az IMAP4 csak egy szerver elérését támogatja. A több IMAP4 szerver elérését lehetővé tevő konfigurációt az IETF-ben fontolgatják.

Ugyanúgy mint a POP3, az IMAP4 sem képes a levelek feladására. Ezt a funkciót általában egy levél átviteli protokoll látja el, ilyen például az SMTP. A 2.6. ábra egy IMAP4-es kliens szerver kapcsolatot mutat.

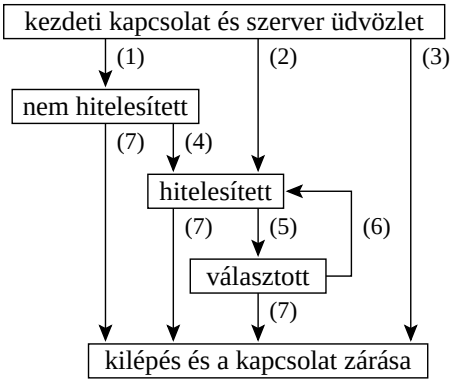
Az IMAP4 viselkedését írja le a 2.7. ábra egy állapot diagrammban. Az ábrán számozással jelölt állapotátmenetek a következők:



2.6. ábra. IMAP4 kliens/szerver architektúra

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, szerver kikapcsolása vagy kapcsolat bontása.

Az IMAP4 a 2.7. ábrán látható állapotok valamelyikében található. A legtöbb IMAP4 parancs csak bizonyos állapotban adható ki. Ha a kliens nem megfelelő állapotban adja ki a parancsot, akkor protokoll hiba generálódik.



2.7. ábra. IMAP4 állapotdiagram

A következő IMAP4 állapotokat definiáljuk ebben a részben:

- Azonosítás előtti állapot
- Azonosított állapot
- Választott állapot
- Kijelentkezett állapot

Az *azonosítás előtti állapot*ban az IMAP4 kliens azonosítási „okmányokat" nyújt be. A legtöbb parancs nem használható, míg a felhasználó nem azonosította magát. A kapcsolat kezdetén ebbe az állapotba kerülünk, hacsak nem történt előzetes azonosítás (preauthentication).

Az *azonosított állapot*ban a felhasználó már hitelesített, de mielőtt kiadná a parancsokat, ki kell választania egy postaládát. Ebbe az állapotba lépünk, ha az előzetesen azonosított kapcsolat kezdődik vagy ha a kliens elfogadható azonosítási okmányokat nyújtott be. Amennyiben hiba történik a postaláda

kiválasztásánál, újra bekerülünk ebbe az állapotba, hogy egy másik postaládát választhassunk ki.

A *választott állapot*ban vagyunk, ha sikeresen kiválasztottuk a kezelni kívánt postaládát.

*Kijelentkezett állapot*ba a kliens kérése, vagy a szerver döntése nyomán kerülhetünk. Ekkor az IMAP4 szerver bezárja a kapcsolatot.

A 2.8. táblázatban láthatók az IMAP4-gyel kapcsolatos RFC-k.

RFC#	Állapot	RFC cím
2095	A	IMAP/POP AUTHorize Extension for Simple Challange/Response
2088	A	IMAP4 nonsynchronizing literals
2087	A	IMAP4 QUOTA extension
2086	A	IMAP4 ACL extension
2061	I	IMAP4 COMPTIBILITY WITH IMAP2BIS
2060	A	INTERNET MESSAGE ACCESS PROTOCOL VERSION 4rev1
1733	I	DISTRIBUTED ELECTRONIC MAIL MODELS IN IMAP4
1732	I	IMAP4 COMPATIBILITY WITH IMAP2 AND IMAP2BIS
1731	A	IMAP4 authentication mechanisms

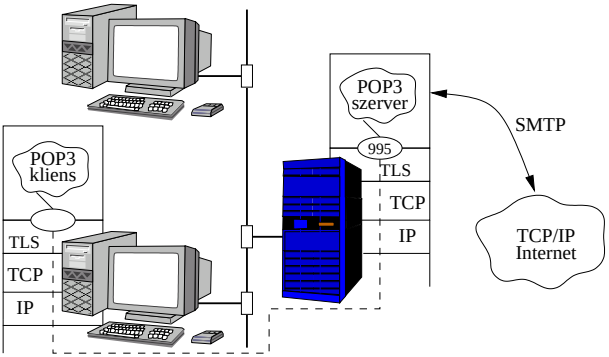
I=Információ A=Ajánlás (Proposed Standard)

2.8. táblázat. IMAP4 RFC-k

2.4. POP3S

A POP3 protokoll biztonságos verziója. Gyakorlatilag kiegészül egy újabb réteggel: SSL 3.0 (Secure Socket Layer 3-as verzió), vagy TLS 1.0 (Transport Layer Security 1.0). Lásd 2.8. ábra. A szerver 995-ös porton kommunikál. A protokoll parancsai meg-

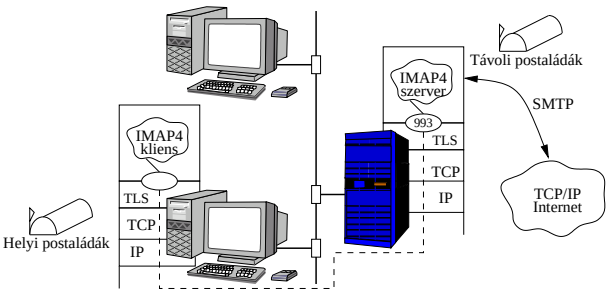
egyeznek a POP3 parancsaival, csak az a különbség, hogy itt a TLS réteg miatt a kliens-szerver kommunikáció egy titkosított csatornán folyik, így nem lehallgatható.



2.8. ábra. POP3S kliens/szerver architektúra

2.5. IMAP4S

Az IMAP4 protokoll biztonságos verziója. Gyakorlatilag kiegészül egy újabb réteggel: SSL 3.0 (Secure Socket Layer 3-as verzió) vagy TLS 1.0 (Transport Layer Security 1.0). Lásd 2.9. ábra. A szerver a 993-as porton kommunikál. A protokoll parancsai megegyeznek az IMAP4 parancsaival, csak az a különbség, hogy itt a TLS réteg miatt a kliens-szerver kommunikáció egy titkosított csatornán folyik, így nem lehallgatható.



2.9. ábra. IMAP4S kliens/szerver architektúra

3. fejezet

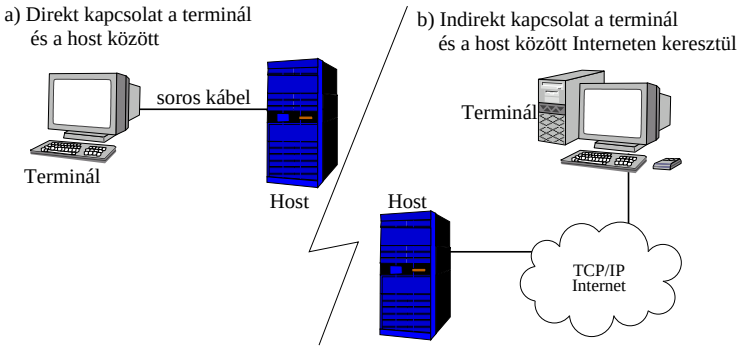
Távoli elérési protokollok

A nagy számítógépekre a felhasználók terminálról szoktak bejelentkezni. A terminál jellemzői annak hardverétől és a hoston futó operációs rendszer által definiált terminálmodelltől függnek. Miután a kapcsolat létrejött a terminál és a host között, a felhasználó karaktersorozatokban parancsokat küldhet a host felé. A hoston futó terminállillesztő veszi és puffereli a karaktersorozatokat, összeállítja belőlük a felhasználói parancsokat. Ezeket átadja a hostnak, amely végrehajtja a user által kért műveleteket és visszaküldi az eredményt.

Ha a terminál hálózaton keresztül kapcsolódik a hosthoz, akkor szükség van egy távoli elérési protokollra, amely biztosítja azokat a szolgáltatásokat melyek közvetlen kapcsolat esetén elérhetőek (3.1. ábra). Az elsődleges távoli elérési protokollok a TCP/IP protokoll családban a következők:

- Telnet
- Berkeley r* segédprogramok

A biztonságos kommunikációra való igény miatt létrejött az ssh és az scp.



3.1. ábra. Terminál/Host architektúra

Távoli gépre munkavégzés céljából `telnet`tel ma már nem szoktunk bejelentkezni, mivel mind a jelszót, mind a teljes kommunikációt nyílt szövegként viszi át. Mégis érdemes vele foglalkozunk, mert így előjönnek a távoli bejelentkezéskor felmerülő problémák, illetve a tárgyból is használni fogjuk az egyes protokollok vizsgálatánál.

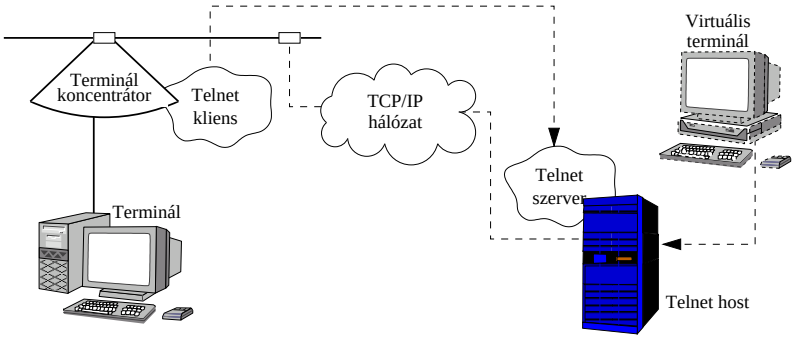
Szintén a titkosítás hiánya miatt a Berkeley r* parancsok használata is gyakorlatilag megszűnt, néha clusterekben alkalmazzák még őket. Tárgyalásuknak szintén didaktikai értelme van: logikájukat megértve egyben az őket leváltó ssh/scp parancsok logikájához is közelebb kerülünk.

3.1. Telnet

A *telnet* protokollt arra használjuk, hogy emulálja egy terminál kapcsolódását a hosthoz. TCP protokollt használ az információátvitelre a terminál billentyűzete és a host között, válaszirányban a terminál kijelzőjén jelennek meg a host üzenetei.

A 3.2. ábra egy telnet kapcsolatot mutat. Ahhoz, hogy mű-

ködhessen egy telnet kapcsolat, a felhasználó munkaállomásán egy telnet kliensnek, a távoli hoston egy telnet szervernek kell futnia. A telnet kliens és a szerver között TCP kapcsolat van. A telnet szerver a 23-as TCP porton várja a kapcsolat felépülését. Unix rendszerekben a telnet szervert `telnetd`-nek hívják (telnet daemon, ejtsd: telnet-dí).



3.2. ábra. Példa egy telnet kapcsolatra

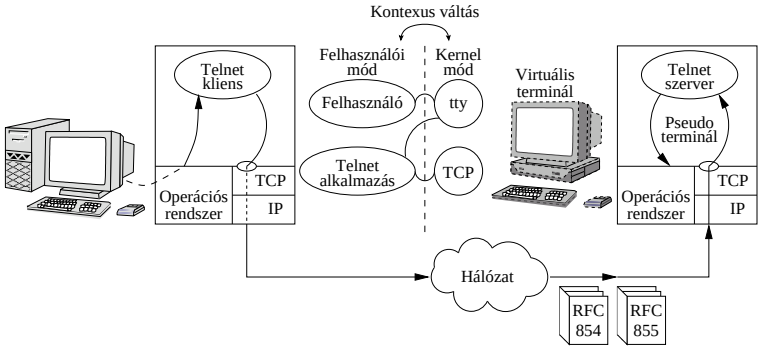
A felhasználó által begépelt parancsokat veszi a telnet szerver, és elküldi a szerveren futó operációs rendszer felé, ami úgy hajtja azokat végre, mintha egy helyi terminálon adták volna ki. A végrehajtott parancs eredményét a telnet szerver visszaküldi a telnet kliensnek. A telnet kliens a szervertől kapott eredményeket kiírja a felhasználó munkaállomásának képernyőjére.

Csak bejegyzett felhasználók jelentkezhetnek be a szerverre. A bejelentkezés után az, hogy milyen parancsokat adhatunk ki, kizárólag a szervergépen futó operációs rendszertől függ.

3.1.1. Telnet architektúra

A 3.3. ábra egy telnet kliens/szerver modellt mutat. Amikor a felhasználó elindít egy telnet alkalmazást és megadja a célállo-

mást, egy TCP kapcsolat épül fel a célállomás 23-as portján keresztül. Az adatok a telnet kliens és a telnet szerver között TCP kapcsolaton keresztül áramolnak. Rendes körülmények között a TCP kapcsolat akkor bomlik le, amikor a felhasználó kiadja a kijelentkezés parancsot, hogy bezárja a telnet kapcsolatot.



3.3. ábra. Telnet kliens/szerver modell

Miután a TCP kapcsolat létrejött, a telnet kliens és a telnet szerver megállapodnak a paraméterekben, melyek meghatározzák a terminál típusát (Telnet Option Negotiation) és a működés módját. Például egy terminálnál beállítható, hogy csak akkor küldjön el egy begépelt sort, ha azt befejeztük, pedig az alapbeállításban a karakterenkénti küldés van megadva.

Ha a felhasználó gépelni kezd, akkor azt átveszi a terminál meghajtóprogramja, és átadja a telnet kliensnek. A telnet kliens általában minden billentyűleütést külön TCP szegmensben küld el. Ez a hálózat gazdaságos kihasználása szempontjából elég nagy pazarlás (például Ethernet hálózat esetén 1/64-ed a kihasználtság), de egy gyors hálózat számára ez nem okoz problémát, ha csak néhány telnet felhasználó van. A telnet szerver válaszai több karakterből álló üzenetek lehetnek, így sokkal job-

ban kihasználhatja a hálózatot, mint a telnet kliens.

A telnet szervert sok implementációban az Internet daemon (szuper daemon) indítja el, amely egy konfigurációs fájlban megadott listán szereplő portokat figyel. Ha kapcsolat-felépítési szándékot észlel a 23-as TCP porton (amely általában a telnet szerveré), akkor elindítja a telnet szervert. Gyakran szokott egymás mellett több telnet szerver is futni, hogy nagy számú telnet klienst tudjon kezelni egy host. Egy másik megoldás, hogy egy telnet szerver külön szálat használ az egyes telnet kapcsolatokhoz.

A 3.3. ábrán egy felhasználói módban futó telnet kliens és szerver látható. A modern operációs rendszerek két módban képesek programokat futtatni: *felhasználói* és *kernel* módban. A legtöbb alkalmazás felhasználói módban fut, amely korlátozott elérést nyújt a számítógép hardvere felé. Ez a mód korlátozza az olyan alkalmazások használatát, melyek a rendszer összeomlását okozhatják. Az operációs rendszer szolgáltatásai, a protokollok, az eszközök kezelői általában csak kernel módban futnak. Továbbá kernel módban a programok korlátlan hozzáférést kapnak a számítógép hardveréhez.

Ha a telnet kliens és a szerver is felhasználói módban fut, és el szeretnénk érni valamilyen operációs rendszer szolgáltatást, mint például a TCP/IP protokollt, egy *kontextus váltás*t kell végrehajtanunk, hogy átkapcsoljunk a kernel módba, ahol elérhető a TCP/IP protokoll.

Kliens oldalon amikor egy karakter megérkezett a terminál-tól, kontextus váltás történik felhasználói módból kernel módba, hogy elérje az operációs rendszer teletype (tty) meghajtóprogramját, amely kernel módban fut. A tty meghajtóprogram átadja ezt a karaktert a telnet kliensnek, amely felhasználói módban fut. Így tehát a kontextus váltás történik kernel módból a felhasználói módba. Amikor a telnet kliens elküld egy karakter adatot a TCP-nek, egy másik kontextus váltás történik, mivel

a TCP kernel módban fut.

A szerver oldalon is kontextus váltás történik, amikor a TCP átadja a beérkezett karaktert a telnet szerver programnak. A telnet szerver elküldi az összes karakter adatot az operációs rendszernek egy pseudoteletype-on (ptty) keresztül, ami több kontexus váltást is tartalmazhat. Amennyiben nagy számú karakter érkezik, sok kontextus váltásra van szükség ami a gép lelassulását okozhatja kis teljesítményű gépek esetén. Ha más alkalmazások futnak a telnet szerveren, a telnet által okozott terhelés azok futására is hatással van.

A fent leírtak természetesen minden más távoli bejelentkezési eljárásnál is jelen vannak (például a később tárgyalt ssh-nál is).

A telnet szabvány (más néven STD 8) megtalálható a következő RFC-kben:

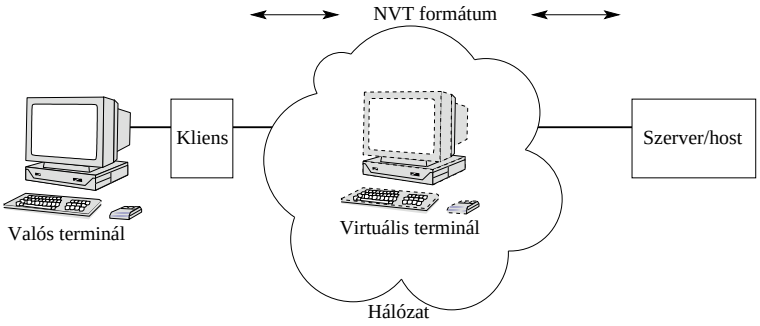
- RFC 854 – „Telnet Protocol Specification"
- RFC 855 – „Telnet Option Specification"

3.1.2. Az NVT Terminál és formátuma

Az NVT a terminál hardverek és a telnet szerver képességek széles skálájához való alkalmazkodás érdekében lehetőséget nyújt a paraméterekről való megállapodásra. Ezeket a paramétereket definiálja a hálózati virtuális terminál (Network Virtual Terminal, NVT) koncepció, amely a 3.4. ábrán látható.

Az NVT eszköz és az NVT protokoll meghatároz egy eljárást a vezérlő és adatinformációk átviteléhez. A terminál hardverek, a vezérléseik és az adat információk közötti különbségeket az NVT eszköz modell és a NVT protokoll kezeli.

A telnet kliens lefordítja a felhasználó billentyűleütéseit egy NVT formátumú karaktersorozattá. Az NVT formátum kezeli a felhasználói adat és a vezérlő adat kódolását. A *felhasználói*



3.4. ábra. Hálózati virtuális terminál

adat tartalmazhat betűket és számjegyeket melyeket a felhasználó begépel. A *vezérlő adat* tartalmazhat olyan karaktereket, amelyek megállítanak egy futó processzt (Ctrl-C sok rendszerben), törlés karaktert (backspace), sor törlést és így tovább. A telnet szerver elfogadja a NVT adat formátumot és lefordítja a helyi operációs rendszer karakterformátumára. A választ a telnet szerver NVT adatsorozatként küldi el. A választ fogadva a telnet kliens olyan formátumba konvertálja, amilyent használ. Összegzésképpen:

- A telnet kliens fordít az NVT formátum és a kliens gép formátuma között.
- A telnet szerver fordít az NVT formátum és a szerver gép formátuma között.

Az NVT formátum hét bites US ASCII kódot használ és fenntartja a legmagasabb helyértékű bitet a karakterben a parancs szekvenciának. A US ASCII karakterkészlet 95 megjeleníthető karaktert tartalmaz: betűket, számokat, központozási jeleket és 33 vezérlőkódot. A 3.1. táblázatban látható néhány szabványos vezérlő karakter.

Név	Kód	Jelentés
NUL	0	nincs művelet
BEL	7	hang/kép jelzés
BS	8	mozgás egy pozícióval balra
HT	9	mozgás a következő vízszintes tabulátorra
LF	10	mozgás (függőlegesen) a következő sorra
VT	11	mozgás a következő függőleges tabulátorra
FF	12	mozgás a következő oldal tetejére
CR	13	mozgás az adott sor elejére

3.1. táblázat. Szabványos NVT vezérlőkarakterek

A 3.5. ábrán példát mutatunk be az NVT protokoll működésére. A felhasználónak a parancshoz a következő karakter szekvenciát kell begépelnie:

```
cp inpo<bs>ut.doc x
```

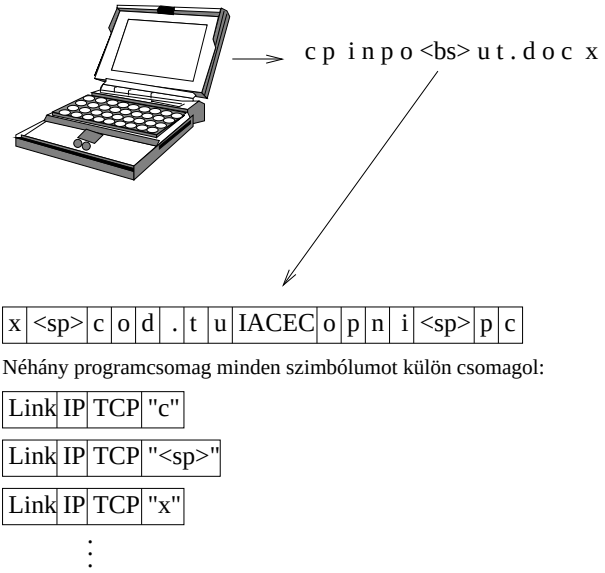
A <*bs*> jelenti a backspace karaktert ami ki fogja törölni az előzőleg hibásan begépelt *o* betűt. A telnet kliens elküldte a NVT karaktereket US ASCII megjelenítést használva, minden karakter külön TCP szegmensben. Ha megfigyeljük a <*bs*> karakter a következőképpen nézett ki a 3.5. ábrán: `IACEC`. Az IAC jelentése az interpret-as-command (értelmezd mint parancsot) karakter, aminek a kódértéke 255. Az EC karakter a törlés, aminek a kódértéke 247.

3.1.3. Telnet beállítások meghatározása

Miután a telnet kapcsolat létrejött TCP protokollon keresztül, mindkét oldalon – telnet kliens és szerver – meghatározzák azokat a beállításokat, melyeket használni fognak. Ezek az opciók például a következők lehetnek: terminál típus, fél-duplex vagy duplex mód, sor mód, karakter mód, stb.

Kód	Jelentés
IAC(255)	a következő oktet parancsot tartalamaz
IAC(255) DON'T(254)	a küldő kéri, hogy a vevő törölje a beállítást
IAC(255) DO(253)	a küldő kéri, hogy a vevő engedélyezze a beállítást
IAC(255) WON'T(252)	a küldő törölni akarja a beállítást
IAC(255) WILL(251)	a küldő engedélyezni akarja a beállítást
IAC(255) GA(249)	„kezdje el/folytassa adását” jelzés ("Go Ahead")
IAC(255) EL(248)	„sor törlés” jelzés ("Erase Line")
IAC(255) EC(247)	„karakter törlés” jelzés ("Erase Character")
IAC(255) AYT(246)	„ott vagy?” jelzés ("Are You There?")
IAC(255) AO(245)	„kimenet megszakítása” jelzés ("Abort Output")
IAC(255) IP(244)	„process megszakítása” jelzés ("Interrupt Process")
IAC(255) BRK(243)	„Break” jelzés
IAC(255) NOP(241)	„nincs művelet” jelzés ("No Operation")
IAC(255) SB(250)	Adott opciókban való megegyezés kezdete
IAC(255) SE(241)	Adott opciókban való megegyezés vége

3.2. táblázat. Telnet parancsok hivatkozáslistája



3.5. ábra. Példa az NVT protokoll működésére

A telnet kliens speciális telnet kódokat küld a beállítás megadására, mint DO, WILL, DON'T, WON'T. A szerver szintén hasonló kódokkal válaszol, mint WON'T, DON'T, WILL és a DO. A kódokat karakterekként küldik el, és mindegyik előtt ott kell legyen a 255 (IAC) parancsot jelző kód. Például a 253-as kód le van foglalva a DO parancs számára, és 252-es pedig WON'T parancs számára, így tehát:

255 253 = IAC DO = DO beállítás indikátor

255 252 = IAC WON'T = WON'T beállítás indikátor

A 3.2. táblázatban látható a telnet parancsok hivatkozáslistája, és a 3.3. táblázatban kódértékek vannak néhány telnet beállításhoz.

Név	Kód	RFC#	Jelentés
Echo	1	857	A vevő engedélyezi a visszhang adatot
SGA	3	858	Megszünteti a „kezdje el/folytassa adását” jelzés küldését az adat végén
Status	5	859	TELNET beállítás állapotának lekérdezése a távoli oldalról
Timing Mark	6	860	Időzítő jelzés elhelyezésének kérése a visszatérő adatfolyamba (szinkronizálás céljából)
Terminal Type	24	884	Információcsere a terminál típusairól
NAWS	31	1073	Ablakméret beállítása
Tspeed	32	1079	Információ elküldése a Terminál sebességéről
TFC	33	1080	Terminál (távoli) Flow Control (adatáramlás vezérlése)
Linemode	34	1116	Teljes sorok küldése karakterek helyett
Xdisploc	35	1096	X képernyő helye

3.3. táblázat. Kódértékek néhány telnet beállításhoz

3.2. Berkeley r* segédprogramok

A BSD Unix disztribúció elérhetővé tett számos parancsot, melyeket r parancsoknak vagy r* segédprogramoknak hívnak. Ezek a parancsok végrehajthatnak egy távoli gépen bejelentkezést, futtatást, stb. A parancsokat arra tervezték, hogy olyan hálózatokban, ahol a felhasználókban megbízhatunk, a távoli gépekhez átlátszó (transzparens) hozzáférést nyújtsanak. Ez alatt azt értjük, hogy a felhasználónak nem kell magát felhasználói névvel és jelszóval azonosítani, a távoli gépen enélkül is ugyanúgy dolgozhat, mint a helyi gépen. Ez a szituáció különlegesen előnyös az egyetemek területén, ahol a diákoknak és a tanároknak több számítógépre is van bejelentkezési jogosultságuk, és transzparens hozzáférésre van szükségük.

Sok más platform - mint a Unix implementációk, VMS, DOS, NT és így tovább - átvette az r* segédprogramokat az operációs rendszerbe, így ezek a programok széles körben elérhetők. A 3.4. táblázatban látható néhány általánosan elterjedt r* parancs.

3.2.1. Az átlátszó hozzáférés beállítása

Az r* parancsok átlátszó hozzáférésre a *bizalom* segítségével valósul meg. Ahhoz, hogy a felhasználónak átlátszó hozzáférése legyen a hosthoz, a felhasználó számítógépének a neve meg kell legyen adva a távoli gép következő fájljai valamelyikében:

- **hosts.equiv** Unix rendszerekben ez a fájl a `/etc` könyvtárban található. Ez a fájl tartalmazza azokat a gépeket, amelyekben *megbízunk* (trusted host), ami azt jelenti, hogy a hostnak átlátszó hozzáférése lehet ahhoz a távoli géphez, amelyikben a rendszergazda bejegyezte őket a `hosts.equiv` fájlba.

Parancs	Jelentés
rlogin	Bejelentkezés egy távoli gépre.
rexec	Parancs futtatása egy távoli gépen.
rsh	Egy shell (parancsértelmező) futtatása a távoli számítógépen úgy, hogy az végrehajtja az általunk megadott parancsot (amit a parancssorba beírunk).
rcp	Fájlok másolása távoli rendszerek között, illetve a távoli és a helyi számítógép között.
rwho	Kiírja azoknak a felhasználóknak a listáját akik bejelentkeztek a hálózatba.
rwall	Üzenetet küld az összes felhasználónak a meghatározott gépen.
ruptime	Kiírja a hálózatban lévő gépek terhelését, a felhasználók számát és hogy mióta vannak bekapcsolva a gépek.

3.4. táblázat. Néhány r* parancs

- **.rhosts** A felhasználó alkalmazhatja ezt a fájlt arra, hogy engedélyezze más gépek felhasználóinak, hogy az adott gépen (amelyiken az engedélyt adja) az ő jogaival (a login neve alatt) dolgozhatnak. Ezt a fájlt a felhasználó saját home könyvtárában kell elhelyezni. Csak saját magának és csak olvasási joga lehet a fájlra (Unix alatt).

Amellett, hogy megadjuk a host nevet a `hosts.equiv` vagy a `.rhosts` fájlokban, azon felhasználókat is felsorol*hat*juk akik átlátszó hozzáférést kaphatnak a gépen. Ha a felhasználókat nem soroljuk fel, akkor az összes felhasználó átlátszó hozzáférést kap az előzőekben megadott gépről. Ezen fájlok használatának és tartalmának részletei különbözőek lehetnek más-más rendszernél, ezért ajánlott megnézni a host dokumentációját, hogy az adott gépen pontosan hogyan kell ezeket a parancsokat használni.

Unix rendszerekben az `/etc/passwd` fájlt (amely a regisztrált felhasználók listáját tartalmazza) is felhasználják annak el-

lenőrzésére, hogy a felhasználóknak tényleg van-e felhasználói neve az adott gépre.

3.2.2. Az rlogin parancs

Az `rlogin` parancs nem használ olyan szintű beállítás-meghatározást, mint a telnet, így a távoli gépnek vagy ismernie kell a terminál típusát vagy úgy kell bekonfigurálni, hogy elfogadja a felhasználó bejelentkezésekor kért terminéltípust. Néhány rlogin implementáció továbbítja a helyi felhasználó környezeti beállításait a távoli gépre a bejelentkezéskor.

Általában amikor a felhasználó kiad egy parancsot, mint az rlogin, a következőképpen adja meg a távoli gép nevét:

```
rlogin <host_név>
```

Például:

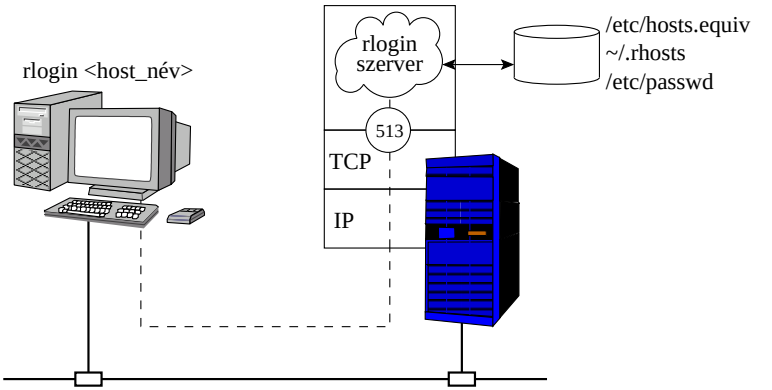
```
$rlogin users.tilb.sze.hu
```

Itt nincs szükség a felhasználó névre és a jelszóra, mert a felhasználót hitelesíti az rlogin szerver (aminek a neve `rlogind` a Unix rendszerekben). Az rlogin szerver a távoli gépen fut, és ellenőrzi a `host.equiv`, a `.rhosts` és (Unix rendszereknél) a `/etc/passwd` fájlokat (látható az 3.6. ábrán).

3.2.3. Az rsh parancs

Az `rsh` parancs segítségével egy távoli gépen lefuttathatunk egy parancsot, melynek eredménye visszajut hozzánk. Például, ha a `users.tilb.sze.hu` távoli gépen akarjuk a `ps -a` parancsot futtatni, és az eredményt visszakapni, a következő parancsot kell kiadnunk:

```
rsh users.tilb.sze.hu ps -a
```



3.6. ábra. Az rlogin használatának illusztrációja

Az r* segédprogramok képesek kezelni mindkét gép környezetét, és értik a fájlrendszer fogalmakat, mint a *standard input* (alapértelmezett bemenet), *standard output* (alapértelmezett kimenet) és a *standard error* (alapértelmezett hiba), melyeket sok operációs rendszer támogat a Unix-on kívül is. Amennyiben az előbbi `rsh` parancs kimenetét egy helyi fájlban szeretnénk eltárolni, a következő parancsot kell kiadnunk:

```
rsh users.tilb.sze.hu ps -a  > helyi_file
```

3.2.4. Az rcp parancs

Az `rcp` segédprogram lehetővé teszi a másolást távoli rendszerek között, illetve a távoli és a helyi számítógép között. A távoli fájlnevekre a következőképen hivatkozhatunk:

```
hostname:pathname
```

Ahol a *hostname* adja meg a host nevét (vagy IP címét), a *pathname* pedig a fájl elérési útját.

A következő parancs fájlt másol a távoli `users.tilb.sze.hu` gépről a helyi gépre:

```
rcp users.tilb.sze.hu:/etc/passwd passwd.users-rol
```

A következő parancs fájlt másol a távoli `pc2.tilb.sze.hu` gépről a távoli `pc3.tilb.sze.hu` gépre:

```
rcp pc2.tilb.sze.hu:/tmp/a pc3.tilb.sze.hu:/tmp/a2
```

Természetesen a parancs sikeres végrehajtásához korábban mindkét távoli gépen meg kellet tenni a szükséges bejegyzéseket a helyi gépre (ahol a parancsot kiadjuk) vonatkozólag!

3.2.5. A Berkeley r* segédprogramok a biztonság szempontjából

Az a probléma az átlátszó hozzáféréssel, hogy ha egy behatoló hozzáférést nyer egy kritikus fájlhoz, mint például ahhoz, amely tartalmazza a megbízható gépek (trusted hosts) listáját, annak megváltoztatásával hozzáférést nyerhet a rendszerhez.

Néhány r* implementáció nem működik IP címek megadásával, kizárólag DNS szimbolikus neveket fogad el. Ez a biztonsági megoldás szükségessé teszi, hogy minden gép regisztrálva legyen a DNS-ben. A behatoló ilyenkor csak úgy tud hozzáférést szerezni, ha regisztrálja magát a DNS-ben, ami viszont lehetővé teszi a behatoló azonosítását.

Sok nem Unix r* implementációban az r parancs használata előtt a felhasználónak meg kell adnia egy helyi jelszót. Ezeket a jelszavakat azonban egy helyi fájlban tárolják, és ha a behatoló ellopja a fájlt, kinyerheti belőle a távoli gép hozzáférésének jogát.

Ami a transzparens elérésben részt vevő gépek biztonsági szintjét illeti, fontos látnunk, hogy ha egy A gépen transzparens

hozzáférést engedélyezünk valamely B gépről, akkor az A gép biztonsági szintje semmiképpen sem lehet magasabb a B gép biztonsági szintjénél, hiszen ha a B gépre betörtek, akkor nyitva áll az út az A gép felé.

3.3. Távoli elérés biztonságosan

A téma mélyebb megértéséhez szükséges kriptográfiai alapokkal a *Hálózatok biztonsága* tárgy foglalkozik. A tárgy jegyzete jelenleg bárki számára elérhető a szakirány honlapján: [4].

3.3.1. SSH alapok

Az SSH csomag arra szolgál, hogy egy hálózatban lévő gépről egy másik gépre biztonságosan tudjunk információt továbbítani.

Az SSH is kliens-szerver architektúrában működik, az ssh szerver (`sshd`) a 22-es TCP porton várja az ssh kliens csatlakozását. Egy ssh kapcsolat létrejöttéhez azonban szükség van néhány előzetes feltételre, konkrétan bizonyos kulcsok rendelkezésre állására, amelyek a következők:

- gépenként nyilvános és titkos kulcs
- felhasználónként nyilvános és titkos kulcs

Egy ssh kapcsolat létrejöttének és működésének a menete a következő

1. titkos csatorna létrehozása a kép gép között
2. a felhasználó azonosítása
3. titkosított kommunikáció

Titkos csatorna létrehozása a kép gép között

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 [10]. (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
2. Jelszavas azonosítás
3. Berkeley r* (szerű megoldás) használata

Ezek közül a nyilvános kulcsú módszert a gyakorlatban is megismerjük.

A jelszavas azonosítás azért lehetséges, mert a jelszó is az első lépésben létrehozott titkos csatornán megy át. Éppen ebből

származik a sebezhetősége is, ha a felhasználó a támadó által felajánlott nyilvános kulcsot fogadott el!

A Berkeley r* további fájlokkal bővül, a `/etc/hosts.equiv` és a `$HOME/.rhosts` fájlokon kívül: `/etc/ssh/shosts.equiv` és `$HOME/.shosts` – ezzel bővebben nem foglalkozunk.

Titkosított kommunikáció

Amennyiben az előző két lépés hibátlan volt, akkor az említett hagyományos titkosítók valamelyikével titkosítják az ssh kliens és szerver közötti adatfolyamot.

3.3.2. SSH a gyakorlatban

Az egyes gépeken `~/.ssh/known-hosts` fájlban lehet tárolni bizonyos távoli gépeknek a nyilvános kulcsát. Ha az SSH kliens kapcsolatot vesz fel a távoli géppel és a nyilvános kulcsa itt megtalálható, akkor létrejöhet egy biztonságos kommunikáció. Ha egy velem kommunikáló távoli gép kulcsát elfogadtam, de valójában nem azzal a géppel kommunikálok, amellyel szándékoztam, akkor jelszavas azonosítás esetén a támadó kezébe kerül a jelszavam.

Ha távoli gépre való belépéskor erős azonosítást szeretnék használni, akkor előbb létre kell hoznom az `ssh-keygen` programmal egy kulcspárt. Ekkor létrejön két fájl: az `id-dsa` tartalmazza a titkos és az `id-dsa.pub` tartalmazza a publikus kulcsomat. Az utóbbit el kell helyeznem azon a gépen, ahova erős azonosítással szeretnék belépni.

Az ssh-t nem csupán távoli gépre történő karakteres terminállal való bejelentkezésre lehet használni, hanem port forwardolásra is, amivel más adatfolyamokat is titkosítottan vihetünk át. A gyakorlatban például X elérésre is szokták használni.

3.3.3. Az rcp, scp parancsok gyakorlása

Feltételezzük hogy:

- A `pc6.tilb.sze.hu` gép előtt ülünk.
- A `pc2.tilb.sze.hu` gépről szeretnénk a `cica.jpg`-t (a home könyvtárunkból) `macska.jpg` néven átmásolni a `pc3.tilb.sze.hu`-ra.
- A `pc6.tilb.sze.hu` gépen a `/etc/resolv.conf` fájlban szerepel a `search tilb.sze.hu`, melynek hatására nem kell kiirni a teljes nevet, elég a `pc2` illetve a `pc3`.
- A `pc2` és a `pc3` gépeken a `hosts.equiv` fájlban szerepel a `pc6.tilb.sze.hu` gép.

A másolás `rcp`-vel a következő módon történik:

```
[user@pc6]$:rcp pc2:cica.jpg pc3:macska.jpg
```

A másolás `scp`-vel csak két lépésben megy:

```
[jozsi@pc6]$ scp joska@pc2:cica.jpg
[jozsi@pc6]$ scp cica.jpg jozso@pc3:macska.jpg
```

Feltéve, hogy a `pc2` gépen `joska`, a `pc3`-on `jozso`, míg a helyi gépen `jozsi` a felhasználónevünk.

4. fejezet

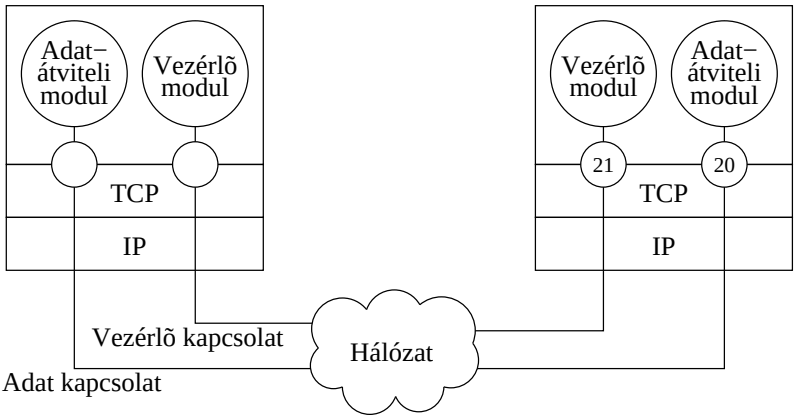
Fájl átviteli protokollok

Az adatokat a főbb rendszerekben fájlnak (állomány) nevezett egységekben tárolják. A fájlnak egy rendszeren a következő jellemzői lehetnek: fájlnév, a létrehozás, az utolsó módosítás, az utolsó hozzáférés időbélyege, a fájl mérete és még további részletek az operációs rendszertől függően. Az adatátvitel két rendszer között általában a fájlok vagy azok részeinek átvitelét jelenti.

A TCP/IP feletti alkalmazások között a következő két protokoll szolgálja számítógépek között teljes fájlok átvitelét: a File Transfer Protocol (FTP) és a Trivial File Transfer Protocol (TFTP). (Ezt a feladatot végzi az `scp` is, csak azt a Berkeley r* parancsokkal való rokonság miatt ott tárgyaltuk.)

4.1. File Transfer Protocol

Az FTP két rendszer között TCP-t használ a fájlok átvitelére IP hálózaton keresztül. A korábbiakhoz hasonlóan az FTP is kliens-szerver architektúrában működik, amint azt a 4.1. ábra mutatja. Az FTP kliens a felhasználó gépén fut és kapcsolatot kezdeményez a szerverrel. Az FTP szerver a szabványos 21-es



4.1. ábra. Az FTP működési modellje

TCP porton várja a kapcsolat felépítését. Amikor a kapcsolat kérés megérkezett, lejátszódik a háromutas kézfogás a TCP kapcsolat felépítéséhez. Ezt a TCP összeköttetést *vezérlő kapcsolatnak* hívják; az FTP parancsok és válaszok küldésére használják. Amikor az FTP kliens kiad egy parancsot az adatátvitel megkezdésére, a kliens elindít egy adatátviteli folyamatot (process) egy helyi TCP porton. Ezt a helyi TCP port számot elküldi a szervernek a vezérlőkapcsolaton keresztül. Mikor az FTP szerver megkapja az FTP kliens adatátviteli processének port számát a szerver elindít egy adatátviteli process-t a szabványos 20-as TCP porton, amely kiad egy TCP kapcsolat felépítési kérést, hogy létrejöhessen az összekötetés az FTP kliens adatátviteli process-ével a kliens által korábban elküldött sorszámú porton keresztül. Ezt az összeköttetést *adat kapcsolatnak* hívják, és kizárólag a kért fájl (könyvtárlista) adatainak átvitelére használják. Amennyiben az adatkapcsolat felépítése az imént leírt módon a szerver irányából a kliens felé történik

"aktív módról" beszélünk. A tűzfalak miatt ma nagyon gyakran úgynevezett *"passzív módú"* FTP-t használunk, ami azt jelenti, hogy a szerver küldi ki a portszámot a kliensnek, és a kliens kezdeményezi az adatkapcsolatot a szerver gép megadott portja felé.

Az FTP lehetővé teszi, hogy le és feltölthessenek fájlokat. Ezek a fájlok mind az adatkapcsolaton keresztül továbbítódnak, csakúgy mint például a könyvtárak listázása. Összegzésül a következő két összeköttetés jön létre egy FTP kapcsolat alkalmával:

- Vezérlő kapcsolat (TCP, FTP_KLIENS_IP, KLIENS_PORT1, FTP_SZERVER_IP, 21)
- Adat kapcsolat (TCP, FTP_KLIENS_IP, KLIENS_PORT2, FTP_SZERVER_IP, 20)

Az adat kapcsolat csak a fájl (vagy könyvtárlista) átvitelének idejére jön létre és bezáródik, amint az átvitel befejeződött. Új kapcsolat jön létre minden alkalommal, amikor egy fájlt átviszünk. A vezérlő kapcsolat fennmarad, amíg azt a kliens vagy a szerver be nem zárja. Általában a kliens szakítja meg a kapcsolatot, azonban túlterhelés vagy más probléma esetén a szerver is bezárhatja. Például ha adott ideig az FTP kliens nem fordul hozzá, akkor be szokta zárni.

Az FTP működési modellben látható, hogy két külön process fut egyszerre. Bizonyos esetekben nem jöhet létre két különálló process. Például, ha egy olyan korlátozott operációs rendszert használunk, amely nem képes több process-t egymás mellett futtatni, vagy az FTP egy beágyazott rendszer, amely operációs rendszer nélkül fut. Ezekben az esetekben egy egyszerű, egy darabból álló program vagy process kezeli mindkét FTP komponenst. Tekintet nélkül arra, hogy az FTP kliens vagy szerver milyen megoldást használ, az FTP protokoll mindenképp két kapcsolatot igényel.

Megjegyzés: a fentiekkel ellentétben az elterjedt rendszerekben az FTP protokollban meghatározott két funkcionális komponenst a valóságban egyetlen program valósítja meg!

Az FTP szabvány megtalálható az RFC 959 (STD 9) „File Transfer Protocol" című részében. A következőkben az FTP kliens-szerver kommunikációs parancsaival ismerkedünk meg.

4.1.1. FTP parancsok

A parancsokat az FTP vezérlő kapcsolatán keresztül továbbítják a telnet által használt adatformátumban. (Emlékeztetőül: a telnet NVT formátumot használ adattovábbításra.) Az FTP parancsok nem használják az NVT formátum minden lehetőségét, csak annak egy részhalmazát. (Például nem használja az option negotiation-re használt NVT parancsokat.)

Az FTP parancsok maximum négy karakteres string-ek, opcionálisan paraméterekkel kiegészítve. A parancsok és jelentésük a 4.1, a 4.2 és a 4.3. táblázatban láthatók.

Az FTP szerver válasza egy három számjegyű kódból, illetve szóköz után opcionálisan szöveges üzenetből áll:

```
nnn Szöveges üzenet
```

Az *nnn* a három számjegyű kódot jelenti.

Például az FTP szerver válaszolhatja a következőt:

```
200 Command okay.
```

A 4.4. és a 4.5. táblázatokban található az FTP szerver néhány lehetséges válasza és azok jelentése.

4.1.2. FTP a felhasználó szemszögéből

Az FTP lehetővé teszi a felhasználó számára a távoli gépen történő interaktív fájl- és könyvtárhozzáférést és a következő műveletek végrehajtását:

Vezérlő parancs	Jelentés
USER <felhasználói_név>	Megadja az FTP kapcsolatot kezdő felhasználó nevét.
PASS <jelszó>	Megadja a USER parancsban megadott felhasználóhoz tartozó jelszót.
ACCT <account>	Megadja a PASS parancs után a további account információkat. (A parancs opcionális.)
CWD <könyvtár>	Megváltoztatja az aktuális könyvtárat a szerveren.
CDUP	Átvált az aktuális könyvtárról annak szülő könyvtárára.
SMNT <elérési_út>	Mount-ol egy külső fájlrendszer adatstruktúrájából anélkül, hogy megváltoztatná a felhasználói nevet vagy jelszót.
REIN	Visszaállítja a kezdeti értékeket a felhasználó azonosítása előtti állapotba. A USER parancsnak kell követnie.
QUIT	Bezárja az FTP kapcsolatot.
PORT h1,h2,h3,h4,p1,p2	Megadja a TCP végpont információkat. Segítségével megadhatjuk az adat-process FTP kliens portjának számát az FTP szervernek. A h1-h4-ig az IP cím oktetjei decimálisan pontokkal elválasztva, p1 és a p2 az adat-process portjának száma decimálisan pontokkal elválasztva.
PASV	Megadja, hogy az FTP szerver várakozzon az adatkapcsolatra, amit a kliens fog létrehozni. Erre a parancsra adott válasz tartalmazza a TCP végpontot, ahol az FTP szerver figyel.
TYPE <kód>	Megadja az adatmegjelenítés típusát.
STRU <kód>	Megadja a fájl struktúráját. (Például: fájl, rekord, oldal.)
MODE <kód>	Megadja az átvitel módját. (Például: byte folyam (stream), blokk (block), tömörített (compressed).)

4.1. táblázat. FTP parancsok – 1. rész

Vezérlő parancs	Jelentés
RETR <elérési_út>	Megadott fájl letöltése.
STOR <elérési_út>	Megadott fájl tárolása a szerveren (feltöltés).
STOU <elérési_út>	Hasonló, mint a STOR, azonban az eredményfájlnak, melyet létrehoz, egyedi neve kell, hogy legyen a könyvtárban.
APPE <elérési_út>	Hasonló, mint a STOR, azonban ha a megadott fájlnév már létezik, az új adatot folyamatosan hozzáírja a régihez.
ALLO <argumentumok>	Helyet foglal a szerveren a fájlnak.
REST <argumentumok>	Meghatároz a szerveren egy jelző pontot, ahonnan a fájl átvitele folytatódhat.
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.
ABOR	A szerver megszakítja az aktuális parancs végrehajtását.
DELE <elérési_út>	Megadott fájl törlése.
RMD <könyvtár>	Megadott könyvtár törlése.
MKD <könyvtár>	Megadott könyvtár létrehozása.
PWD	Kiírja az aktuális könyvtár elérési útját a szerveren.
LIST [könyvtár]	Kiírja a megadott könyvtárban lévő fájlok neveit és attribútumait. Ha nincs megadva könyvtár, akkor az aktuális könyvtár fájljainak neveit írja ki.
NLST [könyvtár]	Kiírja a megadott könyvtárban lévő fájlok neveit attribútumok nélkül. Ha nincs megadva könyvtár, akkor az aktuális könyvtár fájljainak neveit írja ki hasonlóképpen. (Az `mget` parancs használja.)
SITE <paraméterek>	Megadja a szerver specifikus parancsait.

4.2. táblázat. FTP parancsok – 2. rész

Vezérlő parancs	Jelentés
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.

4.3. táblázat. FTP parancsok – 3. rész

- 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)

Ahhoz, hogy FTP-t használjunk, szükséges egy kliens alkalmazás. Ilyen kliens alkalmazások sok platformra elérhetők. Bizonyos platformok rendelkeznek grafikus felhasználói interfésszel (Graphical User Interface, GUI), míg mások parancssort használnak. Az oktatás szempontjából érdemesebb a parancssoros interfészt tárgyalni, mert könnyebb az egyes parancsok azonosítása. Feladat: keressük meg az összefüggést az FTP felhasználói parancsai (4.6. táblázat) és az FTP kliens-szerver kommunikáció parancsai között.

Ahhoz, hogy megnyissunk egy FTP kapcsolatot, a következő parancsot kell kiadnunk:

```
ftp [host_név]
```

A *host_név* annak az FTP szervernek a szimbolikus neve vagy IP címe, amelyre be szeretnénk jelentkezni. Ha a host

Kód	Jelentés
200	Parancs rendben
500	Szintaktikus hiba, nem értelmezett parancs
501	Szintaktikus hiba a paraméterben
202	Parancsot nem implementálták
502	Parancsot nem implementálták
503	Rossz sorrendben megadott parancsok
504	Parancsot nem implementálták ezzel a paraméterrel
110	Restart marker reply
211	Rendszer állapot
212	Könyvtár állapot
213	Fájl állapot
214	Segítség üzenet
215	NAME rendszer típus
120	Üzemkész nnn perc múlva
220	Üzemkész egy új felhasználónak
221	Szolgáltatás bezárja a vezérlő kapcsolatot
421	Szolgáltatás nem elérhető, vezérlő kapcsolat zárása
125	Adat kapcsolat már nyitva, átvitel kezdődik
225	Adat kapcsolat megnyitva, nincs átvitel
425	Adat kapcsolatot nem tudja megnyitni
226	Adat kapcsolat bezárása
426	Kapcsolat bezárva, átvitel megszakadt
227	Passzív módba lépés (h1,h2,h3,h4,p1,p2)
230	Felhasználó bejelentkezve
530	Nincs bejelentkezve
331	Felhasználó neve rendben, jelszóra vár
332	Felhasználói név szükséges a bejelentkezéshez
532	Felhasználói név szükséges a fájl mentéséhez
150	Fájl állapot rendben, adatkapcsolat megnyitása következik
250	Kért fájl művelet rendben lezajlott
257	„PATHNAME” elérési út létrehozva
350	Kért fájl művelet további információra vár
450	Kért fájl művelet nem történt meg

4.4. táblázat. Visszaigazolási kódok – 1. rész

Kód	Jelentés
451	Kért fájl művelet megszakítva, helyi hiba
452	Kért fájl művelet nem történt meg
550	Kért fájl művelet nem történt meg
551	Kért fájl művelet megszakítva, ismeretlen
552	Kért fájl művelet megszakítva
553	Kért fájl művelet nem történt meg

4.5. táblázat. Visszaigazolási kódok – 2. rész

nevét nem adjuk meg, akkor csak néhány FTP parancs adható ki.

Ha kiadjuk a `help` parancsot vagy parancsként a kérdőjelet (`?`), a kliens alkalmazás kiírja az alkalmazható parancsok listáját. A 4.6. táblázatban látható egy ftp kliens parancslista.

!	debug	mdir	sendport	site
$	dir	mget	put	size
account	disconnect	mkdir	pwd	status
append	exit	mls	quit	struct
ascii	form	mode	quote	system
bell	get	modtime	recv	sunique
binary	glob	mput	reget	tenex
bye	hash	newer	rstatus	tick
case	help	nmap	rhelp	trace
cd	idle	nlist	rename	type
cdup	image	ntrans	reset	user
chmod	lcd	open	restart	umask
close	ls	prompt	rmdir	verbose
cr	macdef	passive	runique	?
delete	mdelete	proxy	send	

4.6. táblázat. Karakteres felületű FTP kliens parancsai

A bejelentkezéshez például a következő parancsot kell kiadnunk:

```
ftp users.tilb.sze.hu
```

A parancs kiadása után a következő lépés az azonosítás, melyben meg kell adni a felhasználói nevet és a jelszót. Az FTP szerver az FTP host hitelesítő mechanizmusát veszi alapul a felhasználó jogainak meghatározásához. Tehát ha van egy `jampy` felhasználónév bejegyzésünk a távoli hoston, akkor bejelentkezhetünk az FTP szerverre mint `jampy` és használhatjuk a hoston megadott jelszavunkat.

Számos FTP host támogatja az *Anonymous* (névtelen) bejelentkezést. Amennyiben Anonymous-t adunk meg felhasználói névként, a jelszavunk „guest" (vendég, de sok rendszerben az e-mail címünket kell megadni – néhány esetben csak a „@" karakter meglétét figyelik) és be tudunk lépni a hostra a helyi adminisztrátor által korlátozott jogokkal.

A következő példa egy rövid FTP kapcsolatot kísér végig.

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

   ```
   $ ftp ftp.tilb.sze.hu
   ```

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

   ```
   Connected to ftp.tilb.sze.hu.
   220 ProFTPD 1.2.5rc1 Server (Debian) [ftp.tilb.sze.hu]
   Name(ftp.tilb.sze.hu:root):
   ```

   A kapott üzenet után bejelentkezhetünk mint *Anonymous* felhasználó.

3. Megadjuk a felhasználónevet és jelszót.

   ```
   Name (ftp.tilb.sze.hu:root): Anonymous
   331 Anonymous  login ok, send your complete email
   address as your password.
   ```

```
Password:
230-Udvozlet! Ez itt a Tavkozles informatika
laboratorium anonymous ftp szerver szolgaltatasa...
230 Anonymous access granted, restrictions apply.
Remote system type is UNIX. Using binary mode to
transfer files.
ftp>
```

4. A bejelenkezés után használhatjuk a `?` vagy a `help` parancsot:

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

   A parancsra kapott listát az előzőekben már láthattuk.

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

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

   Minden FTP parancs állapotát egy számkóddal jelzi vissza a host, amelyet egy szöveges üzenet egészít ki.

6. 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.
drwxr-xr-x 2 0 root 16 Nov 5 2003 protokollok
drwxr-xr-x 4 0 root 96 Sep 17 2003 pub
-rw-r--r-- 1 0 root 92 Aug 28 2003 welcome.msg
226-Transfer complete.
226 Quotas off
```

   Az információ megjelenítése Unix stílusú, mivel egy Unix rendszerre jelentkeztünk be.

7. Ha tudjuk, hogy egy Unix rendszer FTP szerverére csatlakoztunk, akkor a következő hasznos parancsot használhatjuk. Ha `ls` parancsot használunk, a Unix `ls` parancsa fog lefutni. Kiegészíthető az `ls` parancs bármely Unix opcióval, mint például a `-lR` opció, amely egy rekurzív hosszú formátumú listát ad a fájlokról és alkönyvtárakról. A `-lR` opció nem szabványos FTP parancs, de alkalmazható Unix szervereken, illetve ott, ahol emulálják azt. A parancs segítségével egy gyors áttekintét kaphatunk az FTP hoston elérhető fájlokról.

8. Könyvtár váltásra a `cd` parancsot használjuk. Az alábbi példában a `/pub/debian/packages` alkönyvtárba lépünk be.

```
ftp> cd pub
250 CWD command successful.
ftp> cd debian
250 CWD command successful.
ftp> cd packages
250 CWD command successful.
```

9. Ahhoz, hogy egy fájlt másoljunk a távoli gépről a helyi hostra a következő FTP parancsot kell kiadnunk:

```
get <távoli_fájl> [helyi_fájl]
```

   Ha a `helyi_fájl`-t nem adjuk meg akkor ugyanaz lesz a fájl neve a helyi gépen mint a távolin. Szövegfájloknál az FTP támogatja a sorvégjel konverziót a kölönböző operációs rendszereknél, ha az átviteli mód ASCII. A bináris fájlok továbbításához használjuk az `image` vagy a `bin` parancsot hogy kikapcsoljuk ezt a konverziót.

   UNIX alatt a sorok végét a 10-es kódú [LF] karakter jelzi. DOS és Windows alatt a sorok végét a 13-as és 10-es kódú

karakterek [CR-LF] együttesen jelzik. ASCII módú átvitel esetén DOS -> UNIX irányban minden 13-10 karakterpárt 10-es kódú karakter helyettesít. UNIX -> DOS átvitel esetén minden 10-es kódú karakter helyett 13-10-es karakterpár kerül a szövegbe.

10. Ha egy olyan fájlt szeretnénk letölteni, amely nem elérhető, mint például az `rfc1365.txt`, az FTP a következőt fogja válaszolni:

```
ftp> get rfc1365.txt
local: rfc1365.txt remote: rfc1365.txt
200 PORT command successful
550 rfc1365.txt: No such file or directory
```

Az alábbiakban látható egy sikeres FTP `get` parancs:

```
ftp> get pico-4.2.tgz
local: pico-4.2.tgz remote: pico-4.2.tgz
200 PORT command successful.
150 Opening BINARY mode data connection for
pico-4.2.tgz (508305 bytes).
226 Transfer complete. 508305 bytes received in
0.0626 secs (7.9e+03 Kbytes/sec)
```

11. Az FTP kapcsolat bezárásához a `close` parancsot kell kiadnunk:

```
ftp> close
221 Goodbye.
```

12. Ahhoz, hogy teljesen kilépjünk az FTP-ből a `bye` parancsot használhatjuk. Ez a parancs bezárja az esetleg nyitva hagyott FTP kapcsolatot, mielőtt kilép az FTP-ből:

```
ftp> bye
$
```

4.2. Trivial File Transfer Protocol

A TFTP az UDP protokollt használja átviteli protokollként. Mivel az UDP nem garantálja az adat megérkezését, a TFTP-nek kell megoldania a hibajavítást. A TFTP egyszerű PAR (Positiv Acknowledgment Retransmission) sémát használ a hibajavításra. A TFTP definiál egy nyugtázó üzenetet, amellyel visszaigazolást kap a megérkezett adatról. Amennyiben egy meghatározott időn belül nem érkezik meg a nyugta, újra elküldi az adatot.

A TFTP-t úgy tervezték, hogy egyszerű és kis kódmérettel alkalmazható legyen akár önbetöltő ROM-ban egy munkaállomáson. A TFTP-t gyakran használják BOOTP-vel együtt. A BOOTP-t a konfigurációs információk kinyerésére, a TFTP-t pedig az operációs rendszer image-ének letöltésére használják.

A TFTP szabványának leírása megtalálható az RFC 1350 „The TFTP Protocol (Revision 2)"-ban és kiegészítések az RFC 1782, 1783, 1784 és 1785-ben.

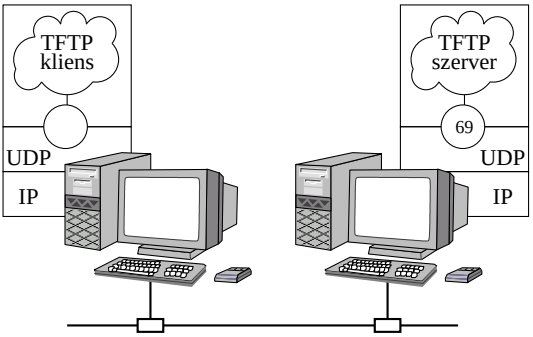
4.2.1. TFTP üzenetformátum

A TFTP öt féle típusú üzenetet definiál. Ezen üzenettípusok a 4.7. táblázatban láthatók.

Művelet kód	Leírás
1	Olvasás kérés (Read request, RRQ)
2	Írás kérés (Write request, WRQ)
3	Adat (DATA)
4	Nyugta (Acknowledgment, ACK)
5	Hiba (ERROR)

4.7. táblázat. TFTP üzenettípusok

Az TFTP kliens küld egy kezdő olvasás (RRQ) vagy írás (WRQ) kérés üzenetet (4.2. ábra) a TFTP szerver 69-es UDP portjára. A TFTP szerver ezen a porton várja az olvasás/írás kérést.

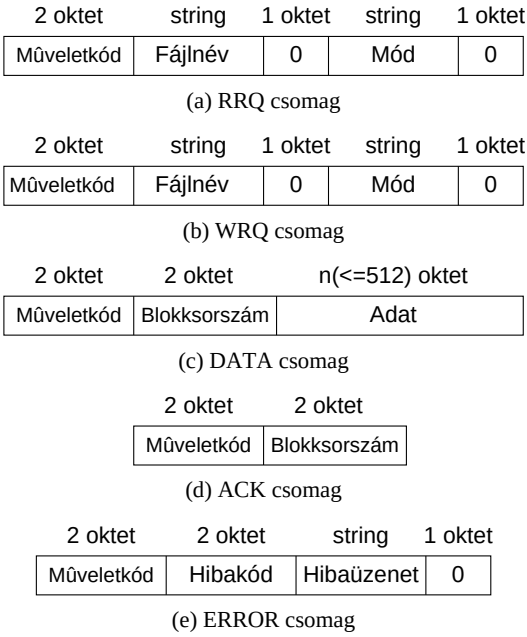


4.2. ábra. TFTP kliens/szerver kapcsolat

Az írás és olvasás kérés formátuma a 4.3. ábrán látható. Minden TFTP üzenetnek van egy *műveleti kód* mezője, melynek lehetséges értékei és jelentései a 4.7. táblázatban megtalálhatók. Az írás és olvasás kérés üzenetekben megtalálható egy fájlnév és egy mód mező. A fájlnév egy karakterekből álló string, amely egy nulla oktetre végződik. A mód mező meghatározza az átviendő fájl típusát. Az értéke lehet *netascii*, *octet* vagy *mail* és egy nulla oktetre végződik. Mivel a fájlnév és a mód is nulla oktetre végződik, változó méretűek lehetnek, mert a null oktet fogja jelezni a végüket.

4.2.2. TFTP művelet

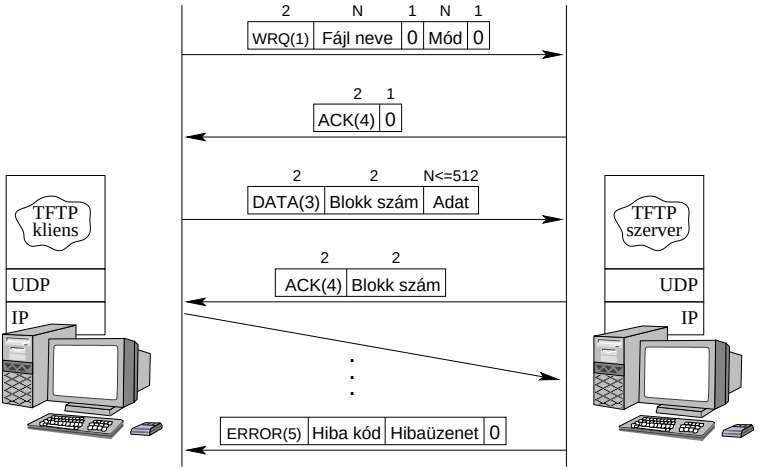
Az írás/olvasás kérés megérkezése után a TFTP szerver rögzíti a kliens IP címét és portszámát. A későbbiekben a válaszokat a TFTP szerver ezen IP címre és portszámra továbbítja.



4.3. ábra. TFTP üzenet formátum

Az írás/olvasás kérés nyugtázása után az adatátviteli folyamatban adat tartalmú üzenetek és azok nyugtázásai következnek. A TFTP 512 oktet méretű üzenteket küld. Ha az adat tartalmú üzenet mérete kisebb mint 512, az az átvitel végét jelenti. Amennyiben az áviendő fájl mérete (oktetben) az 512 egész számú többszöröse, akkor a TFTP szerver küld egy nulla adat tartalmú üzenetet, amely jelzi az adatátvitel végét. Az adatblokkok sorszáma egyesével nő. A 4.4. ábrán látható egy adatátvitel, melyben a TFTP kliens adta ki az írás kérés üzenetet.

A TFTP felhasználó azonosítás nélkül képes az adatátvitel lebonyolítására. A fájlok átvitelekor csak a fájl nevét kell



4.4. ábra. Példa egy TFTP adatátvitelre

megadni. Mivel felhasználói account és jelszó nélkül használható, a rendszergazdák általában letiltják ezt a funkciót, vagy korlátozzák az átvihető fájlok körét.

Hálózati eszközök (például routerek) konfigurációját gyakran TFTP-vel töltik át, ez azonban nem okoz gondot, mivel kívülről nem (vagy erősen korlátozottan) elérhető hálózaton használják.

5. fejezet

Fájl hozzáférési protokollok

A fájlokhoz való közvetlen hozzáférésre – ahol egy távoli fájlrendszer a helyi fájlrendszer részeként látszik – használható a Network File System (NFS) és a Server Message Block (SMB).

5.1. Network File System

A Network File System (NFS) egy fájlkezelő protokoll, melyet eredetileg a SUN Microsystems fejlesztett, majd sok gyártó megvásárolta a licenszét. Az NFS lehetővé teszi, hogy a számítógép, amelyen az NFS szerver program fut, *exportálja* a fájlrendszerét egy kliensnek. A fájlrendszer *exportálása* azt jelenti, hogy a kliens számára elérhető lesz az, még ha a szerver géptől eltérő operációs rendszert használ is, feltéve ha fut az NFS kliens program.

A `/etc/exports` fájlba kell bejegyezni, hogy mely könyvtárakat mely gépek számára osztunk meg. Néhány példa a lehetséges bejegyzések szintaktikájára:

```
/nyilvanos pc6.tilb.sze.hu
/home *.tilb.sze.hu(ro)
/usr *.tilb.sze.hu pc6.tilb.sze.hu(rw)
```

A megosztások érvénybe lépéséhez nem kell újraindítani a gépet, küldhetünk a szerver programnak egy szignált, hogy olvassa újra a konfigurációs fájlt:

```
kill -HUP <pid>    # ahol a pid a process id
```

A Debian Linux rendszerben az NFS alrendszer vezérléséhez a `/etc/init.d/nfs-kernel-server` nevű scriptet kell használni. A script az alábbi módon paraméterezhető:

```
/etc/init.d/nfs-kernel-server start
/etc/init.d/nfs-kernel-server stop
/etc/init.d/nfs-kernel-server restart
```

Az 5.1. ábrán látható, amint az NFS szerver exportálja a `/home` könyvtárat. Ez az exportált könyvtár egyszerre több kliens által is elérhető, akár különböző operációs rendszert futtató gépekről. Minden NFS kliens úgy látja az exportált fájlrendszert, mint a saját fájlrendszerének egy részét. Például egy PC-n DOS alatt futó NFS kliens egy hálózati meghajtóként fogja elérni a fájlrendszert, egy Unix NFS kliens saját fájlrendszerébe linkelve érheti el.

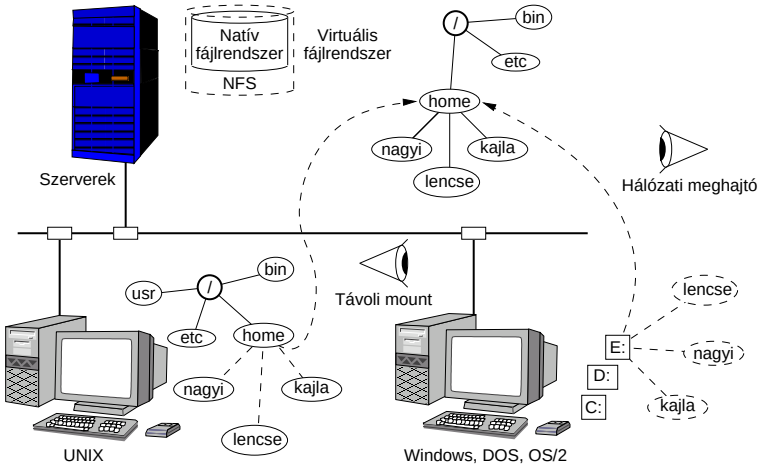
Lássunk két példát könyvtár felkapcsolására a helyi fájlrendszerbe:

```
# mount -t nfs nfs.tilb.sze.hu:/nyilvanos /mnt
```

A parancs szintaktisa: *honnan:mit* (gépnév és könyvtárnév kettősponttal elválasztva, szóköz nélkül) és *hová* (könyvtárnév) akarunk felkapcsolni, a `-t nfs` a partíció típusát adja meg.

A `pc5`-ös gépen a `pc6` gép számára szeretnénk megosztani a `/home/jozsi` könyvtárat és a `pc6`-on felkapcsolni a `/home/joco` könyvtárba.

```
root@pc5# echo "/home/jozsi pc6.tilb.sze.hu" >> /etc/exports
root@pc5# /etc/init.d/nfs-kernel-server restart
root@pc6# mount -t nfs pc5.tilb.sze.hu:/home/jozsi /home/joco
```



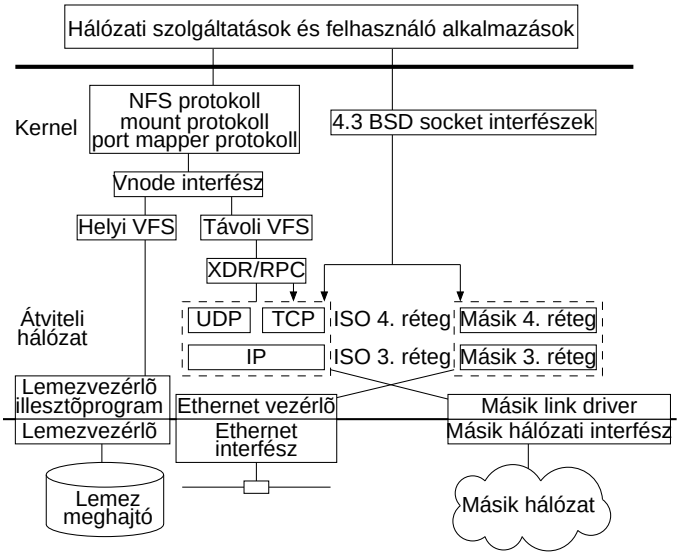
5.1. ábra. Az NFS használata

5.1.1. NFS protokollok

Az NFS hálózati alkalmazást egy magas szintű NFS protokollkészlet segítségével valósították meg. Az 5.2. ábrán különböző NFS protokollok láthatók. Az OSI adatkapcsolati és fizikai szintjein számos technológiát támogat az NFS. A hálózati rétegben az NFS IP-t használ. A szállítási rétegben az NFS UDP-t vagy TCP-t használ. Ha az NFS-t UDP felett használjuk, akkor nagyon ajánlott az opcionális UDP ellenőrző összeget használni.

Távoli gépek között értelmetlen a klasszikus eljáráshívásról beszélnünk, ahol a veremben tárolódnak szekvenciálisan az eljáráshívás paraméterei. Erre a problémára nyújt megoldást (a vi-

szony rétegben) a Remote Procedure Call (RPC, távoli eljáráshívás) protokoll. A hívásokat két szám azonosítja: a programszám és az eljárásszám. Több egymástól teljesen eltérő RPC protokoll létezik, ezért megkülönböztetésképpen az NFS RPC-jét gyakran hívják Sun-RPC-nek. Az RPC lehetővé teszi, hogy az NFS szolgáltatásokat a szerveren az eljáráshívás módszerével érjék el, amit a programozók jól ismernek. Az RPC magas szintű hozzáférést biztosít az NFS szerverhez anélkül, hogy a kommunikációs protokollok bonyolult részleteibe merülne. A programozóknak sem kell elmélyedniük a kommunikáció rejtelmeiben, hogy RPC-vel elérhető hálózati szolgáltatásokat használjanak. A Sun-RPC protokoll dokumentációja megtalálható az RFC 1057-ben "RPC: Remote Procedure Call Protocol Specification version 2" cím alatt.



5.2. ábra. Az NFS protokollok

Az OSI megjelenítési rétegében az NFS az External Data Representation (XDR) protokollt használja. Az XDR biztosítja az egységes adatmegjelenítést.[1] Mindkét fél XDR formátumra konvertálja az adatokat. Az XDR dokumentációja megtalálható az RFC 1832-ben "XDR: External Data Representation Standard" cím alatt.

Az OSI alkalmazási rétegében az NFS a Network File System (NFS) protokollt használja. Az NFS olyan műveletek végrehajtását teszi lehetővé, mint például fájl írása, létrehozása, olvasása az NFS szerveren. Az alkalmazási réteg néhány protokollal támogatja az NFS-t, úgy mint például a *Mount* és a *Portmapper* protokollok. A Mount protokoll végzi az NFS mount-olási folyamatát, és a Portmapper protokoll adja meg a szolgáltatás eléréshez a port számát a kliensek. A Portmapper protokoll a szabványos 111-es UDP portot használja. Ha egy NFS kliens hozzáférést szeretne valamilyen NFS szolgáltatáshoz, egy kérést küld a 111-es UDP portra. A szerver programok, melyek valamilyen szolgáltatást szeretnének nyújtani, beregisztrálják magukat a Portmapper-nél. Ha beérkezik egy kérés egy regisztrált szolgáltatásra, a Portmapper választ küld, amely tartalmazza a port számát, melyen keresztül elérhető a szolgáltatás.

5.2. Server Message Block

A *Server Message Block* protokoll is könyvtárak megosztását teszi lehetővé. A `/etc/samba/smb.conf` fájlban lehet konfigurálni. Például egy megosztás az alábbi módon hozható vele létre:

```
[Megosztas]
```

[1] Például az egész számok ábrázolásánál az egyik számítógép a legmagasabb, a másik a legalacsonyabb helyiértékű bájtot tárolja az alacsonyabb memóriacímen. Ezt az eltérést át kell hidalni.

```
Path=/home/pista
```

Bővebben foglalkozunk vele a *Hálózati operációs rendszerek* tárgyban.

6. fejezet

Internet elérési protokollok

Több protokollt hoztak létre azzal a céllal, hogy dokumentumokat tegyenek elérhetővé az Interneten. A két kiemelkedő protokoll a HTTP (HyperText Transfer Protocol) és a Gopher (volt). Ezen protokollok közül a legszélesebb körben a HTTP-t használják, melyet népszerűbb nevén World Wide Web (WWW) protokollnak hívnak. A Gopher mára már majdnem teljesen kihalt, így csak nagyon röviden tárgyaljuk.

6.1. World Wide Web

A World Wide Web-et gyakran csak web-nek (pókháló) hívják. A web sok *web szerver*ből áll. A web szerverek dokumentumokat tárolnak, amelyek tartalmazhatnak például szöveget, képet, hangot és videót, melyeket *weblapokba* (web pages) szerveztek. Minden weblap tartalmazhat *link*eket (hyperlink), amelyek mutatók web dokumentumokra. A hyperlink-ek mutathatnak azonos, illetve különböző web szerveren lévő dokumentumokra egyaránt. A linkekkel keresztül-kasul behálózott dokumentumok

szövevénye átnyúlik ország- és kontinenshatárokon, ezt fejezi ki a World Wide Web (világ méretű pókháló) egyik kedves magyarítása: világszőttes.

A web dokumentumokat a web kliensek, azaz web böngészők segítségével érheti el a felhasználó, melyet saját gépén futtat. A weblapokat, mint dokumentumot egy speciális nyelven kódolják, amelyet HTML-nek (HyperText Markup Language) nevezünk. Miután a böngésző letöltötte a HTML dokumentumot, megjeleníti azt a képernyőn grafikusan, esetleg szöveges formátumban. A HTML tartalmazza a HTML dokumentum különböző elemeinek leírását, melyeket felhasználva a böngésző grafikusan megjelenítheti a weblapot. A linkek általában aláhúzott szövegként jelennek meg. (Figyelem: kép is lehet link!) A linkekre kattintva letölti azt a dokumentumot, amire a link mutat. A következőkben a HTML nyelvről és a HTTP protokollról lesz szó.

6.1.1. HyperText Markup Language

A HTML a Standard Generalized Markup Language (SGML) egy implementációja. Az SGML szabvány megad egy általános módszert, hogy miképpen készítsünk olyan dokumentumokat, amelyek hyperlink-eket 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 kép megjelenítése, levél küldése, felhasználótól adatok kérése form-on keresztül, távoli bejelentkezés vagy fájl átvitel kezdeményezése, adatbázis lekérdezése, program futtatása és így tovább. Az olyan dokumentumot, amely hyperlinket tartalmaz, *hyperdokumentum*nak is hívjuk.

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

A <HTML> és </HTML> tag-ek között meg kell adni a HTML dokumentum fejrészét és törzsét. A fejrészt a <HEAD> és </HEAD> tagek között, és 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. Például a következő HTML kód jelentésben teljesen megegyezik az előzővel:

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

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

Az üres sorok és a sortörések általában javítják a HTML dokumentumok forrásának olvashatóságát, ezért ajánlatos alkalmazni. A tagek nem érzékenyek a kis- illetve nagybetűkre és az

egyéb szövegformázásra. Ha azt szeretnénk, hogy a böngésző által megjelenített weblapon sortörést, újsort, vagy egyéb formázást hajtsunk vége, akkor speciális HTML tageket kell a forrásban elhelyeznük. 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>
```

Hyperlink megadásához használjuk a következő taget:

```
<A HREF="URL" egyéb_paraméterek > hyperlink
szöveg </A>
```

A tagnek számos paramétert lehet megadni az *egyéb paraméterek* résznél, melyek modósítják 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* bármelyik lehet az Interneten használt protokollok közül, mint például: `http`, `ftp`, `telnet`, `gopher`, `file`. Ha megadunk egy HTML dokumentumot egy másik gépen, akkor a `http` (HyperText Transfer Protocol) protokollt szoktuk használni a letöltésére. Ha egy dokumentumot fájl transzferrel szeretnénk letölteni, az `ftp` protokollt kell használnunk, és így tovább. Ha egy helyi fájlt szeretnénk megnyitni a saját gépünkről a web böngészőben, akkor a `file` protokollt kell megadnunk.

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`. Itt látható néhány példa az URL címekre:

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

Megjegyzés: az URL-nél általánosabb az URI, ami a Uniform Resorce Identifier rövidítése. Érdeklődőknek ajánljuk: [11]

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 tag-gel 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 lehetőséget ad számozott és számozatlan listák létrehozására, melynek tagjai ugyancsak külön-külön sorban lesznek. A számozatlan lista minden tagja elé egy jelet tesz. A lista szintaktikája a következő:

```
<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>
```

Érdemes még megjegyezni, hogy a <B> és </B> tag-ek közé írt szöveg félkövér, míg a <I> és </I> tagek közé írt szöveg dőltbetűs lesz. Továbbá használhatunk fejrészeket, amelyek <H1>-től <H6>-ig különböző megjelenítésűek.

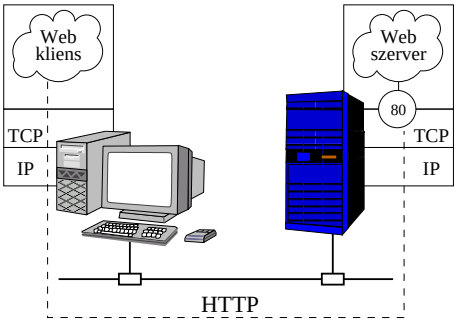
A HTML nyelvvel sokkal mélyebben foglalkozunk a tárgy anyagának másik felében.

6.1.2. HyperText Transfer Protocol

A web kliens (böngésző) a HTTP protokoll segítségével kommunikál a web kiszolgálóval. (Hívják web szervernek, HTTP szervernek is. Az egyik legelterjedtebb megvalósítása az Apache `httpd`.)

Mielőtt letöltünk egy HTML dokumentumot, létrejön egy kapcsolat a böngésző és a web szerver között. A TCP kapcsolat alapértelmezetten a 80-as porton jön létre (6.1. ábra). Megadhatunk más portokat is, mint például a 8080-as port, de alapértelmezetten a 80-as portot foglalták le a HTTP számára.

Ha azt szeretnénk, hogy a web böngészőnk ne az alapértelmezett portot használja, akkor az URL címben meg kell ad-



6.1. ábra. HTTP kliens-szerver kapcsolat

nunk a kívánt portot. Az alábbi példában egy URL-t láthatunk, amellyel a 8080-as portra kapcsolódunk:

```
http://www.tilb.sze.hu:8080
```

A kapcsolat létrejötte után a web böngésző küld egy kérést a HTTP protokollt használva. A kérés tartalmazza a letölteni kívánt objektum nevét és az általunk használt HTTP protokoll verziószámát. A HTTP protokoll szöveg string-eket használ, így az emberek számára is könnyen érhetőek a kiadott parancsok. Például a következő HTTP kérést küldi el a web böngésző a web szervernek egy dokumentum letöltéséhez:

```
GET /index.html HTTP/1.0
```

Itt a `GET` HTTP parancsot használjuk az `/index.html` letöltéséhez. A `HTTP/1.0` argumentummal pedig megadtuk, hogy milyen verziószámú HTTP protokollt szeretnénk használni. Vegyük észre, hogy a kérésből teljesen hiányzik a web kiszogáló címe! Ez alapesetben azért nem okoz(ott) gondot, mert a TCP kapcsolat már úgyis a megfelelő kiszolgálóval jött létre. Virtuális webszerverek esetén azonban egyetlen kiszolgáló több név

alatt is működik, és szüksége van a host információra, ezért az 1.1-es HTTP verzióban el is küldik.[1] Például:

```
GET /index.html HTTP/1.1
HOST: www.tilb.sze.hu
```

A szerver a HTML dokumentum letöltésére irányuló kérés megérkezése után küld egy HTTP választ. A HTTP válasz három részből áll:

- státusz
- fejléc
- adat

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

```
HTTP/1.1 200 OK
```

A válasz státusz után következik a válasz fejléc. A válasz fejléc tartalmazza a következő információkat: a szerver típusa, MIME verziószám, a tartalom típusa[2] (megadja, hogy az adat részben milyen típusú tartalom van), és végül egy üres sor. A fejlécet és az adatot mindig egy üres sor választja el. Az alábbiakban látható egy példa a válasz fejlécre:

[1] A protokoll leírása megtalálható az RFC 2068-ban "Hypertext Transfer Protocol HTTP/1.1" cím alatt.

[2] A levelezésnél már láttunk példát a MIME által támogatott tartalomtípusokra és kódolásokra.

```
Date: Tue, 20 Jul 2004 14:24:29 GMT
Server: Apache/1.3.26 (Unix) Debian GNU/Linux
Last-Modified: Tue, 07 Oct 2003 07:33:45 GMT
Tag: "1025-b2-3f826c59"
Accept-Ranges: bytes
Content-Length: 178
Connection: close
Content-Type: text/html; charset=iso-8859-2
```

Ha a dokumentum, amit lekérdezünk egy HTML dokumentum, akkor a válaszadat egyszerű szöveg lesz, amely a HTML dokumentumot tartalmazza.

Miután a HTTP kérés/válasz lebonyolódott a TCP/IP kapcsolat bezáródik. Külön TCP kapcsolat jön létre minden egyes HTTP kérés/válasz folyamathoz.[3]

A 6.2. ábrán látható egy tipikus HTTP kapcsolat. Az első lépés a TCP kapcsolat megnyitása a 80-as porton. A következő lépésben a kliens küld egy GET HTTP/1.0 parancsot. A dokumentum átvitele után a web szerver bezárja a kapcsolatot. Egy másik dokumentum letöltéséhez egy új kapcsolatot kell létrehozni.

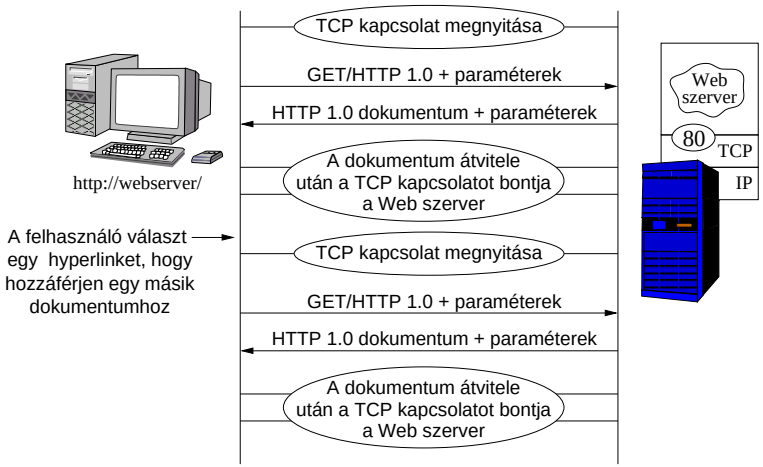
Ismerjük meg jobban a HTTP működését! Egy telnet kliens segítségével jelentkezzünk be egy web szerver 80-as portjára a következő módon:

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

A kapcsolat létrehozása közben a következőhöz hasonló üzeneteket láthatunk:

```
Trying 193.224.130.171...
Connected to vip.tilb.sze.hu.
Escape character is '^]'.
```

[3]Ezt HTTP 1.1 esetén másképpen is be lehet állítani, majd *Hálózati operációs rendszerek* tárgyból tanulni fogjuk!



6.2. ábra. Egy tipikus HTTP kommunikáció

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

```
GET /index.html HTTP/1.0
```

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:

```
HTTP/1.1 200 OK
Date: Tue, 19 Aug 2008 08:57:27 GMT
Server: Apache/1.3.33 (Debian GNU/Linux)
Last-Modified: Wed, 15 Mar 2006 12:56:09 GMT
ETag: "417c4-13d-44180ee9"
Accept-Ranges: bytes
Content-Length: 317
Connection: close
```

```
Content-Type: text/html
X-Pad: avoid browser bug

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>
<head>
<title>vip.tilb.sze.hu</title>
</head>
<body>
<table align="center" width="100%" height="100%"
 cellspacing="0" cellpadding="0" border="0">
<tr>
<td align="center"><h1>VIP.TILB.SZE.HU</h1></td>
</tr>
</table>
</body>
</html>

Connection closed by foreign host.
```

A HTTP válaszban megtalálható a státusz, a fejléc, illetve egy üres sor után az adat rész is. A HTML dokumentum átvitele után a kapcsolat automatikusan megszakad. Ha egy másik dokumentumot szeretnénk letölteni a HTTP/1.0-val, akkor egy új kapcsolatot kell nyitnunk.

Az előbb látott példában sikeresen letöltöttünk egy HTML dokumentumot. Próbáljuk ki, hogy mi történik, ha egy olyan dokumentumot szeretnénk letölteni, ami nem létezik. Hozzunk létre egy újabb telnet kapcsolatot a web szerverrel és adjuk ki például a következő HTTP parancsot:

```
GET /asdf.html HTTP/1.0
```

Nyomjuk le kétszer az enter-t és a következő választ kapjuk:

```
HTTP/1.1 404 Not Found
Date: Tue, 19 Aug 2008 09:05:38 GMT
```

```
Server: Apache/1.3.33 (Debian GNU/Linux)
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HTML><HEAD>
<TITLE>404 Not Found</TITLE>
</HEAD><BODY>
<H1>Not Found</H1>
The requested URL /asdf.html was not found on this server.<P>
<HR>
<ADDRESS>Apache/1.3.33 Server at localhost Port
80</ADDRESS>
</BODY></HTML>
Connection closed by foreign host.
```

Várakozásunknak megfelelően a szerver nem találta a kért dokumetumot. Ezt jelzi a visszaadott a 404-es hibakód és a mellette levő szöveges üzenet is.

A HTTP válasz ilyenkor is tartalmaz egy HTML dokumentumot, amelynek tartalma elárulja, hogy a kért dokumentumot nem találta. Az a HTML dokumentum, amelyet a web szerver visszaadott, egy *virtuális dokumentum*. Ilyen dokumentumok nem találhatók meg a web szerveren, ezeket dinamikusan generálja a szerver hasonló természetű hibák esetén.

Házi feladat: próbáljuk ki, milyen válasz küld a web szerver egy olyan hibásan kiadott HTTP parancsra, mint például a következő:

```
GET ezt egy hibásan kiadott parancs
```

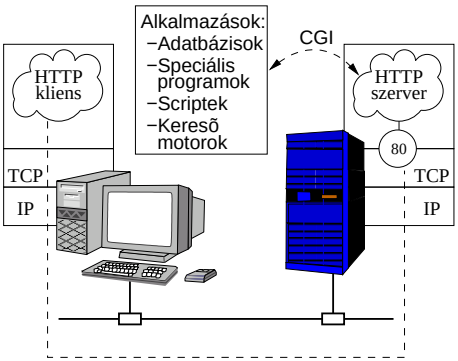
Megemlítjük, hogy a GET-en kívül más eljárások is vannak, például nagyobb mennyiségű adat felküldésére használható a POST, ahol a kérés body-jában van a feltöltendő adat.[4]

[4]GET-nél csak az URL-ben lehet, ami kis mennyiségű adatnál nem okoz gondot.

6.1.3. Web indexelés és CGI átjárók

Ahhoz, hogy megtaláljunk egy adott kulcsszót a web-en, számos *keresőmotor*t (search engine) használhatunk. A keresőmotorok web dokumentumokat indexelnek; követnek minden linket a web dokumentumban, és indexelnek minden megtalált dokumentumot. Sok web böngésző tartalmaz elérési módozatokat ezekhez a kereső motorokhoz. Keresőmotor található a `www.google.com`-on, a `www.hudir.hu`-n és még számos helyen.

Számos keresőmotor használ Common Gateway Interface-t (CGI), amely lehetővé teszi, hogy más alkalmazásokat is futtassunk (lásd 6.3. ábra). A CGI-t arra használják, hogy kéréseket továbbítson más alkalmazásoknak, amelyek ugyanazon vagy akár egy másik szerveren futnak. A futtatás eredményét visszaküldi az alkalmazás a CGI-n keresztül a HTTP szervernek, majd az továbbküldi a web kliensnek.

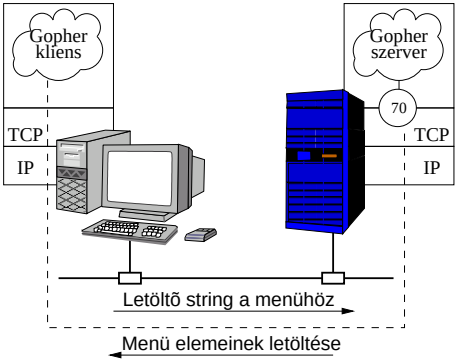


6.3. ábra. Common Gateway Interface (CGI)

6.2. Gopher

A Gopher protokollt és eszközöket Minnesotai Egyetem fejlesztette ki. A Gopher-ben a menü címek a hyperlink-ek. Ha kiválasztunk egy menüpontot, egy újabb menüt vagy egy dokumentumot nyithatunk meg ugyanazon vagy akár egy másik szerveren. A menük készletét – melyek egymásba vannak linkelve – *gopher space*-nek hívjuk.

A 6.4. ábrán láthatunk egy kapcsolatot a Gopher kliens és a Gopher szerver között. A Gopher szerver a 70-es TCP porton várja a letöltő stringet, melyet a Gopher kliens küld. A Gopher szerver visszaküld egy menüt, amely a letöltő stringen alapul.



6.4. ábra. Gopher kliens/szerver

A Veronica (Very Easy Rodent-Oriented Networkwide Index to Computerized Archives) egy speciális eszköz, amely a Gopher menükben képes keresni. A Veronica általában megtalálható a Gopher menük legfelső szintjeiben.

A Gopher protokoll (melynek részleteivel nem foglalkozunk) megtalálható az RFC 1436-ban. Érdeklődőknek ajánljuk a következő magyar nyelvű forrást: [7].

Irodalomjegyzék

Könyvek

[1] Albitz, Paul and Cricket Liu: *DNS and BIND*, 4th ed., O'Reilly, Sebastopol, CA, 2001.

[2] Lencse Gábor: *Számítógép-hálózatok*, 2. kiadás, Universitas-Győr Nonprofit Kft., Győr, 2008.

[3] Siyan, Karanjit S.: *Inside TCP/IP*, Third Edition, New Riders Publishing, Indiana, 1997.

Webes források

[4] *A távközlés-informatika szakirány honlapja*
`http://www.tilb.sze.hu`

[5] IANA: *Port Numbers*
`http://www.iana.org/assignments/port-numbers`

[6] *Internet Szolgáltatók Tanácsa*
`http://www.domain.hu`

[7] HUP Wiki: *Gopher*
`http://wiki.hup.hu/index.php/Gopher`

[8] *Request for Comments*
`http://www.ietf.org/rfc.html`

[9] *RFC Index*
`http://www.ietf.org/iesg/1rfc_index.txt`

[10] Wikipedia: *Diffie-Hellman_ key_ exchange*
`http://en.wikipedia.org/wiki/Diffie-Hellman_key_exchange`

[11] Wikipedia: *Uniform_ Resource_ Identifier*
`http://en.wikipedia.org/wiki/Uniform_Resource_Identifier`