



Budapesti Műszaki- és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Távközlési és Médiainformatikai Tanszék

Médiakommunikáció labor

Kísérleti fájlmegosztó tervezése
(BMEVITT5321)

Készítette:

Darázsi László XZ1J94

Lada22@gmail.com

Muráti Ákos JNC2FC

akos@murati.hu

Szeles József VC4DV7

szjoe5@freemail.hu

2009. december 9.

1. A kitűzött feladat

1.1 Alkalmazási szintű hálózati protokoll tervezése

- Kísérleti fájlcsere rendszer protokolljának meghatározása
- Központosított átfedő létrehozása
 - Központi kiszolgálón tartják nyilván az egyes ügyfeleknél található fájlok névsorát
 - A fájlcsere közvetlenül történik az ügyfelek között
 - Ez a Napster-jellegű fájlcsere rendszer

1.2 Egyenrangú (P2P) hálózat létrehozása

- A fájlcsere rendszer központi kiszolgálójának és ügyfélprogramjának elkészítése
- A központi kiszolgáló egy webes felületű MySQL alapú linuxos szoftver
- Ügyfélszoftvernek Windows XP-n kell tudni működni

1.3 Alkalmazási szintű többesadás

- Alkalmazási szintű többesadás feladatkör beépítése a rendszerbe

2. A feladat értelmezése

A munka során egy *Napster* jellegű p2p alapú fájlcsere rendszert kellett létrehozunk. A feladat megoldása több részre bontható. Első lépésében a fájlcsere által használt protokollt kellett definiálnunk, jól átgondolva az alkalmazás működésének minden részletét. Ez a része a megoldásnak magába foglalja az alkalmazás által nyújtott funkciók lefixálását, a felhasználandó technológiák által nyújtott lehetőségek megvitatását, illetve a megvalósítás részleteinek és a kommunikáció folyamatának átgondolását.

A feladat második részét a kidolgozott és megtervezett fájlcsere rendszer központi kiszolgálójának és kliensprogramjának megvalósítása implementálása képezte.

A feladat harmadik része pedig a már helyesen működő fájlcsere rendszer kibővítése alkalmazási szintű többesadás lehetőségével.

Miután megtaláltuk a felhasználandó technológiákat, átgondoltuk, hogy mely funkciókat hogyan szeretnénk megvalósítani, hogyan fognak kommunikálni egymással a hálózat elemei (peer-peer, peer-server kommunikáció). Ezután megvitatottuk, milyen adatbázis táblákat kell létrehozunk és mit kell tárolnunk az adatbázisban.

3. Megvalósítás

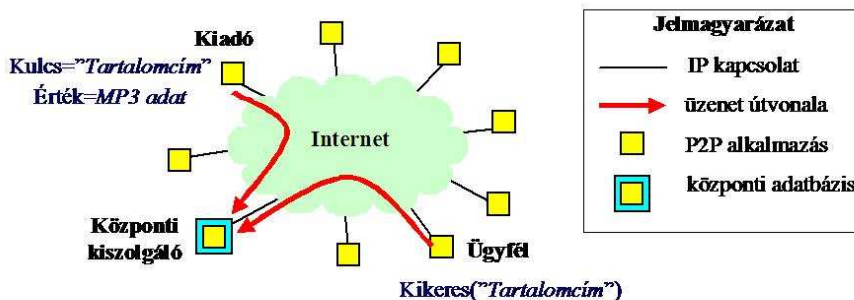
3.1 A megvalósítással kapcsolatos általános információk

3.1.1 Napster alapú fájlcsere rendszer, elméleti háttér

A *Napster* az első központosított építményű p2p átfedő volt. Fontos tulajdonsága, hogy az összes peer megosztásai egy helyen, egy központi szerveren voltak indexelve. Amikor egy kliens kapcsolatba lépett a szerverrel, elküldte az IP címét és a megosztani kívánt fájl adatait, amit a szerver eltárolt (fájl tárolási helye, fájlnev, létrehozás ideje, stb....).

A kliensek, ha le szerettek volna tölteni egy fájlt, nem volt más dolguk, mint egy keresési kérést indítani a szerverhez, majd a válasz alapján kapcsolódni a visszakapott IP címen elhelyezkedő klienshez és letölteni a fájlt.

A *Napster* hátránya az egyetlen szerver, ami az egész rendszer lelke. A szerver egyrészt a rendszer szűk keresztmetszetét képezi, ugyanakkor a meghibásodása következtében a hálózat nem képes tovább működni.



Cél egy *Napster*hez hasonló elven működő p2pfájlcsere rendszer létrehozása.

3.1.2 Informatikai háttér, rendelkezésre álló technológiák

A feladat megoldásához adott volt egy *Debian Linux* operációs rendszert futtató számítógép, mely a *turul.eet.bme.hu* domain néven érhető el. A szerver *MySQL* adatbáziskezelőt és *Apache* webszerveret futtat, így a feladat megoldása során rendelkezésünkre állt egy *MySQL* adatbázis, illetve egy *PHP* kiszolgálót futtató webszerver. A klienseknek a 316. laborban lévő *Windows XP* operációs rendszeren kell tudni futniuk, a feladat bemutatására itt kerül sor.

A szervert *PHP*-vel valósítottuk meg, amely közvetlenül hozzáfér az adatbázishoz, míg a kliensszoftvert *.NET* technológiát használva *C#* nyelven implementáltuk.

A kliens-szerver kommunikációhoz, (HTTP feletti) XML alapú RPC hívásokat használtunk. Ehhez egy külső XML-RPC PHP modul¹ használtunk a szerver oldalon, az XML-ek kezelésének megkönnyítése érdekében. A peer-peer kommunikáció a két peer között egy Async socketen keresztül valósul meg.

3.1.3 Tervezett architektúra, működés

A szerver egy adatbázishoz kapcsolódik, az adatbázis tárolja a klienseket, a megosztott fájlokat és az épp folyamatban lévő letöltéseket. Amikor egy kliens bejelentkezik (online állapotba kerül), akkor ezt jelzi a szervernek, ami ezt az információt (a kliens egyéb adataival együtt) rögzíti az adatbázisban. Ha a kliens korábban még nem jelentkezett be ehhez a szerverhez (vagyis új kliensről van szó), akkor a szerver a klienst felveszi az adatbázisba, ha már járt ennél a szervernél korábban (régibb, ismert kliens), akkor frissíti a kliens adatait és online-ra állítja az adatbázisban a klienshez tartozó bejegyzést.

Ha a kliens új fájlokat oszt meg, arról a szervert tájékoztatja, ami felveszi a megosztott fájlok adatait az adatbázisba. A megosztott fájlok között a keresés úgy zajlik, hogy a kliens elküldi a szervernek a keresési információkat, a szerver pedig elvégzi a keresést az adatbázisban, majd visszaküldi az eredményt a kliensnek. Így tájékoztatja a klienst, hogy az adott keresésre milyen fájlokat talált, illetve, hogy azok mely peereknél vannak megosztva. Ha egy peer úgy dönt, hogy letölti valamelyik fájlt, akkor a keresésre kapott válaszból már tulajdonában van azoknak az információknak, amelyek segítségével elérheti a letöltendő fájlt. A letöltés így a peerek között zajlik a szervertől függetlenül. A peernek, aki a letöltést kezdeményezi, tájékoztatnia kell a szervert a letöltés elindulásáról, illetve befejezéséről. Ennek megfelelően a szerver nyilván tudja tartani az épp folyamatban lévő letöltéseket, melyeket a <http://turul.eet.bme.hu/~mk091/index.php> oldalon lehet megtekinteni.

3.1.4 Kliens-szerver kommunikáció, RPC hívások

Amikor egy kliens kapcsolatba szeretne lépni a szerverrel, akkor a <http://turul.eet.bme.hu/~mk091/server.php> URL-en futó alkalmazás 32100-as portjához kell kapcsolódnia. A kliens és szerver közti kommunikáció RPC hívásokon keresztül valósul meg.

Egy bekezdéssel lentebb láthatók a megvalósított RPC hívások, melyeket a kliens küldhet a szervernek. Egy ilyen hívás fogadásakor a szerver értelmezi a parancsot, majd elvégzi a különböző műveleteket (az adatbázis műveleteket is beleértve).

RPC hívások:

- p2p.heartbeat

¹ Keith Devens XML-RPC Client/Server modul, <http://www.keithdevens.com/software/xmlrpc/>

Ez a hívás több célt szolgál, egyrészt ez a hívás jelzi a kliens bejelentkezési szándékát a szervernek. Másrészt a rendszer ezt a parancsot használja a kliensről tárolt információk frissen tartására, ugyanis ez a hívás a szerverrel közli a kliens fontosabb adatait, ennek megfelelően a szerver egy ilyen hívás fogadásakor frissíti a kliens állapotát és adatait az adatbázisban (a kliens adatai, pl. az IP cím és a többi adat is változhat, ilyen lehet mozgó kliens wifivel). Ezt a hívást a kliens előre beállított időközönként (10 – 60 sec) küldi a szervernek, tájékoztatva arról, hogy még itt van a hálózaton. Fogalmazhatnánk úgy is, hogy a kliens tulajdonképp a beállított periódusidő letelte után újra és újra bejelentkezik.

- `p2p.by`

A kliens kijelentkezéskor küldi, ilyenkor offline állapotba kerül, lekapcsolódik a hálózatról. Az offline állapot következtében nem lehet letöltést indítani. A kliens megosztásai nem törlődnek, azonban nem listázhatóak. (A kliensről tárolt mező nem törlődik az adatbázisból, csak az online flag-je változik.)

- `p2p.list_share`

Ezzel az RPC hívással tud a kliens keresni a megosztások között. (Valójában az adatbázisban a szerver keres, a kliens csak kéri a keresés elvégzésére. A keresés eredményeit a szerver visszaküldi.) Csak online peerek megosztásait listázza ki.

- `p2p.register_share`

Ezzel a hívással tud a kliens megosztani új fájlokat. Ez a hívás tulajdonképp kéri a szervert, hogy vegye fel a fájlt az adatbázisba.

- `p2p.unregister_share`

Ezzel a hívással tudja a kliens törölni a megosztott fájljait. A hívás kéri a szervert, hogy a kívánt fájlt törölje az adatbázisból. A kérés következtében a fájl valóban törlődik az adatbázisból.

- `p2p.register_download`

Ezt a hívást a kliens akkor küldi a szervernek, amikor egy fájlt kezd letölteni egy másik kliensről. Ennek hatására a szerver felveszi a letöltést az adatbázisba.

- `p2p.unregister_download`

A letöltés befejezésekor a kliens ezzel a hívással értesíti a szervert a letöltés befejezéséről. Ekkor a szerver az adott letöltést törli az adatbázisból.

- XMLRPC_method_not_found

Nem létező RPC hívás meghívásakor a szerver ezt adja vissza, elvileg ilyen nem fordulhat elő helyes működés esetén.

3.1.5 Kliens-kliens kommunikáció

A kliensek közti kommunikáció a két peer között egy Async socketen keresztül valósul meg. Minden kliensen fut egy kiszolgáló szál, amely a megadott porton aszinkron várja a socket hívásokat `get|N|file` formában. A kliensek a nagyobb fájlokat darabokra bontják és a fájlt, a letöltő kliens tulajdonképp fájlszeleteket kér. Az N változó azt jelzi, hogy hányadik fájldarabot kéri a kliens, a „file” pedig a megosztási mappán belüli relatív elérési útvonalat jelöli fájlnevével együtt. A feltöltő kliens válaszul egy bájtstort küld, aminek az első értéken kódolja azt, hogy van-e további fájldarab vagy sem. Az első bit értéke 0-ha nincs további fájlrészlet vagy 1, ha még van további chunk.

Letöltésnél a kliens a `register_download` RPC hívással regisztrálja letöltését a szerveren, a letöltés befejezését pedig `unregister_download` hívással jelzi a szervernek.

3.1.6 Tervezett funkciók, illetve azok részletei

- *Új kliens felvétele az adatbázisba, vagy ha már a kliensről van bejegyzés az adatbázisban, akkor a kliens adatainak frissítése*

A kliens bejelentkezik a szerverre egy heartbeat RPC hívással, a szerver felveszi őt az adatbázisba, beállítja a szükséges paramétereit, illetve, ha már a kliens fel van véve az adatbázisba, csak offline állapotban van, akkor a klienst online-ra állítja és frissíti az adatait.

(Eredeti terv szerint a szerver egy idő után a nem heartbeat-olt klienseket offline-ra állítja, ám ez a funkció még nincs teljes körűen megvalósítva.)

- *Kliens törlése az adatbázisból*

Valójában klienst nem törölünk, csak offline-ra állítjuk az állapotát. Ilyenkor a megosztásai fizikailag bent maradnak az adatbázisban, de nem listázhatóak.

- *Új megosztás felvétele az adatbázisba*

A megosztás megadásakor egy mappát adhatunk meg, ilyenkor a kliens a mappában található összes fájlt megosztja (az almappákban lévő fájlokat is beleértve, azonban mappákat nem oszt meg).

- *Megosztás törlése az adatbázisból*

A megosztások ilyenkor fizikailag is törlődnek az adatbázisból.

- *Keresés a megosztások között*

A megosztások között tudunk keresni, majd a keresés eredményéből kiválaszthatjuk, hogy mely fájlokat szeretnénk letölteni.

- *Peerek közti fájlletöltés*

A fájlok letöltése a kliensek közt a szervertől függetlenül valósul meg.

- *Aktív letöltések nyilvántartása*

Az épp folyamatban lévő letöltéseket a szerver tárolja, így lehet tudni, hogy az adott időpillanatban hol, mely peerek közt milyen letöltések vannak folyamatban. A <http://turul.eet.bme.hu/~mk091/index.php> oldalon az épp folyamatban lévő letöltések megtekinthetők.

- *Beragadt kliensek automatikus offline-ra állítása*

Eredeti terv szerint egy időzítő figyelte volna a szerveren, hogy a régen szervert látogatott (régen heartbeatet küldött) kliensek állapota valóban offline-e és az ilyen „hibásan online” kliensek állapotát automatikusan (bizonyos idő letelte után) offline-ra állította volna. Ez az állapot (hibásan online) egy kliens esetében ritkán állhat elő, ezért ez a funkció még nem került megvalósításra.

- *Régi és új kliensek megkülönböztetése bejelentkezéskor*

A régi kliens kérheti régi client_ID-je használatát bejelentkezéskor, amennyiben már járt korábban az adott szerveren. Ekkor a szerver, ha valóban talál ilyen bejegyzést a kliensről az adatbázisban, akkor a kliens a bejelentkezést követően a régi client_ID-ját kapja meg.

- *Hitelesítés, secret használatával, secret kezelés*

A rendszer megvalósítja a kliensek egyfajta hitelesítését. A szerver minden egyes heartbeat RPC hívás fogadásakor, generál egy secret-et, amit a heartbeat-re küldött válaszban visszaküld a kliensnek. Ezt a szerver és az adott kliens eltárolja, majd a szervernek küldött következő heartbeat

küldésekor visszaküldi a szervernek. Így a szerver tudja, hogy valóban azzal a klienssel beszélget, akivel szeretne, hiszen azt a secret-et, csak az a kliens ismerheti, akinek korábban elküldte.

3.1.7 Adatbázis (adatbázis táblák, stb.)

Clients tábla

Minden klienshez tárolunk egy azonosítót (`client_ID`), amit a szerver generál. A peerek közti sikeres kommunikációhoz szükség van a kliens IP címére és a használt portra (`host`, `port` mezők az adatbázisban). Tárolunk egy változót, ami azt mutatja, hogy a kliens online-e vagy sem, illetve tároljuk, hogy legutóbb mikor küldött heartbeatet (`lastCheck` mező). A `Clients` táblában tárolásra kerül még az aktuális secret, amit a szerver és a kliens közötti kommunikációhitelesítésre használunk.

Field	Type	Null	Default
<u>client_ID</u>	int(20)	No	
host	varchar(255)	No	
secret	varchar(255)	Yes	NULL
port	int(11)	No	0
lastCheck	datetime	Yes	NULL
online	(1)	No	f

Files tábla

A `Files` táblában tároljuk a megosztott fájlokat, illetve azok adatait. Minden fájl kap egy egyedi azonosítót (`file_ID`). A `hash` mezőt két különböző helyen lévő azonos fájl azonosítására használtuk volna eredeti terv szerint, azonban ez a funkció még nem került megvalósításra. Minden fájlhoz tartozik egy külső kulcs (`client_ID`), amely azt mondja meg, hogy az adott fájl melyik kliensnél található. Az adott megosztásról tároljuk még a relatív elérési útját (`folder` mező) (a relatív elérési úton a megosztási mappától számított relatív útvonalat értjük), a fájl nevét (`name`), kiterjesztését (`type`), a teljes elérési utat az adott peer gépén fájlnevével együtt (`contents`), a fájl méretét byte-ban (`size`), illetve egy hibaszámlálót (`errorCount`), ami a fájl sikertelen eléréseinek számát számolja. (Ha a fájl sikertelen elérésének száma egy korlátot meghalad, akkor a szerver törölheti a fájlt.)

Field	Type	Null	Default
<u>file_ID</u>	int(20)	No	
hash	varchar(255)	Yes	NULL
client_ID	int(20)	No	0
folder	tinytext	No	
name	tinytext	Yes	NULL

type	varchar(20)	Yes	NULL
contents	tinytext	Yes	NULL
size	int(11)	Yes	0
errorCount	int(11)	No	0

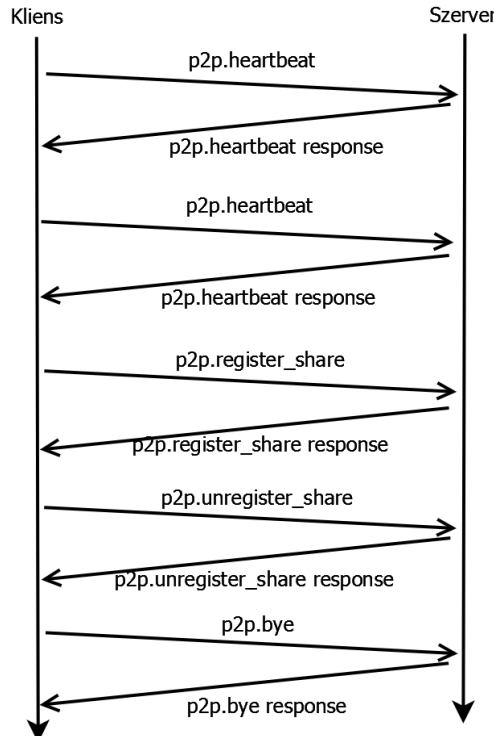
Downloads tábla

A Downloads tábla az épp aktív letöltéseket tárolja. A tábla egy-egy külső kulcsot tárol, a fájlról. Egyrészt egy file_ID-t, ami a letöltés alatt álló fájl azonosítja, és az épp letöltést végző kliens client_ID-jét. (Azt hogy honnan töltünk le, annak a kliensnek az azonosítóját a file_ID-n keresztül meg tudjuk határozni.)

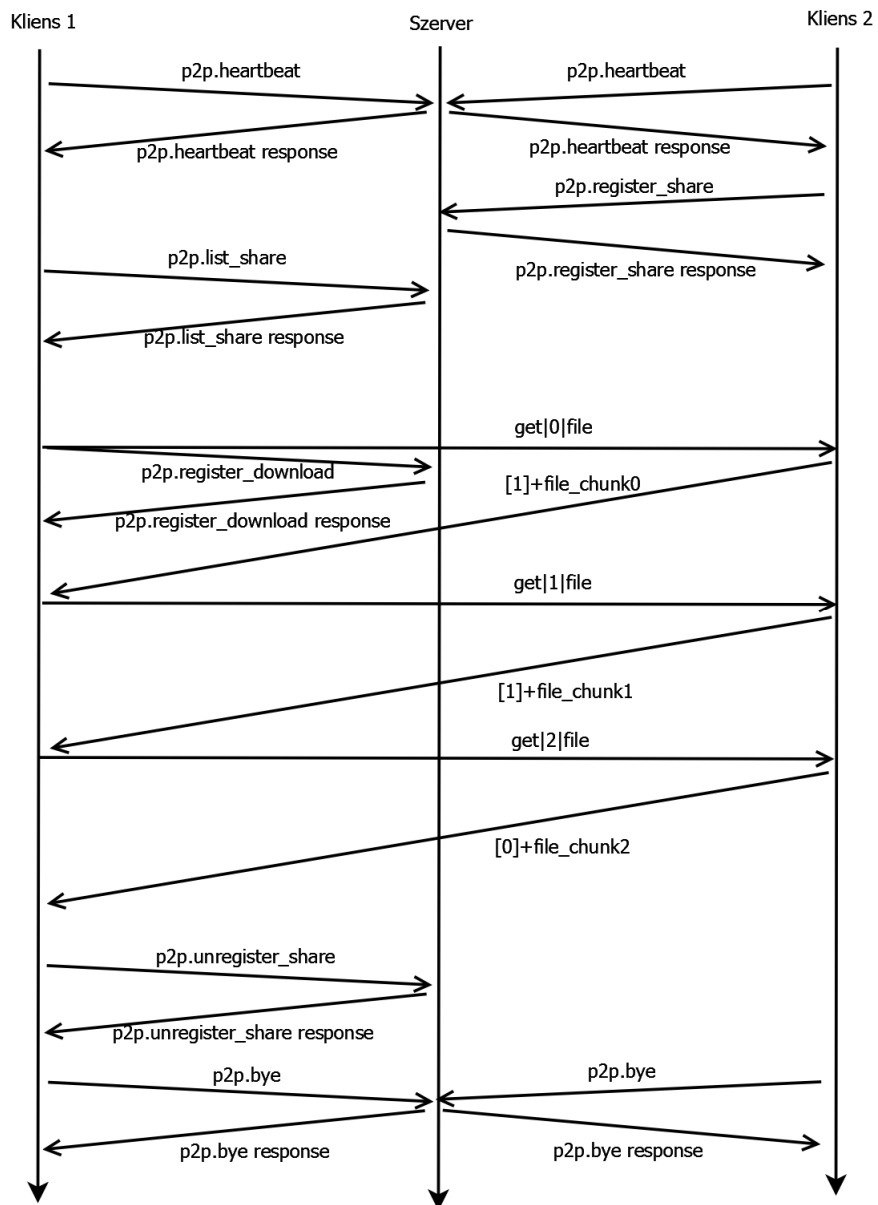
Field	Type	Null	Default
client_ID	int(20)	No	0
file_ID	int(20)	No	0

3.1.8 A kommunikáció folyamata (folyamatábra)

Bejelentkezés, fájlmegosztás, fájlmegosztás törlése, kijelentkezés folyamata.

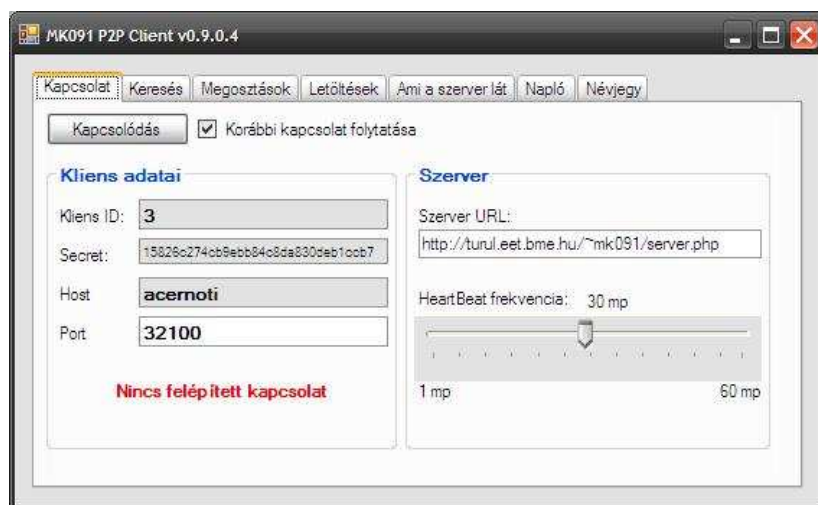


Kliens bejelentkezés, fájlmegosztás, fájlkeresés, letöltés, kijelentkezés.

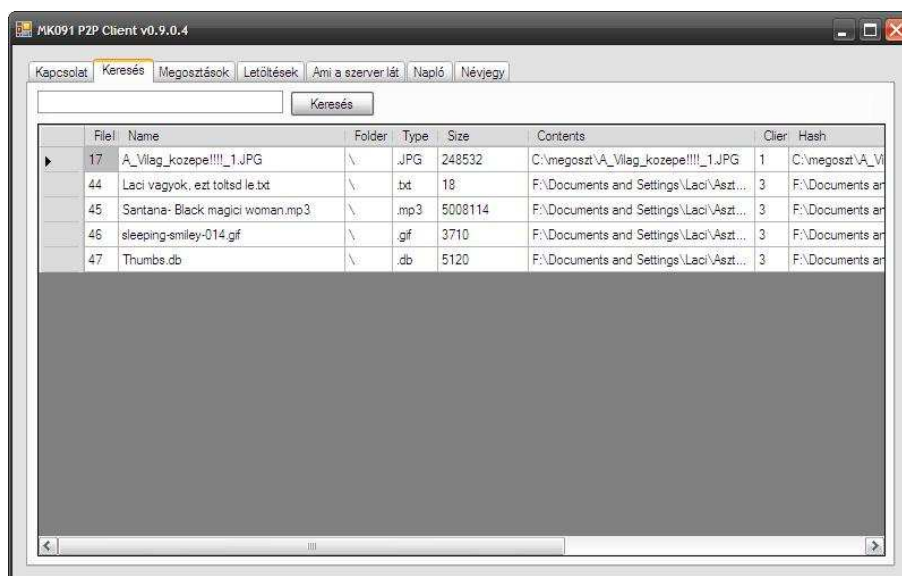


3.1.9 A kliensalkalmazás és az általa nyújtott funkciók

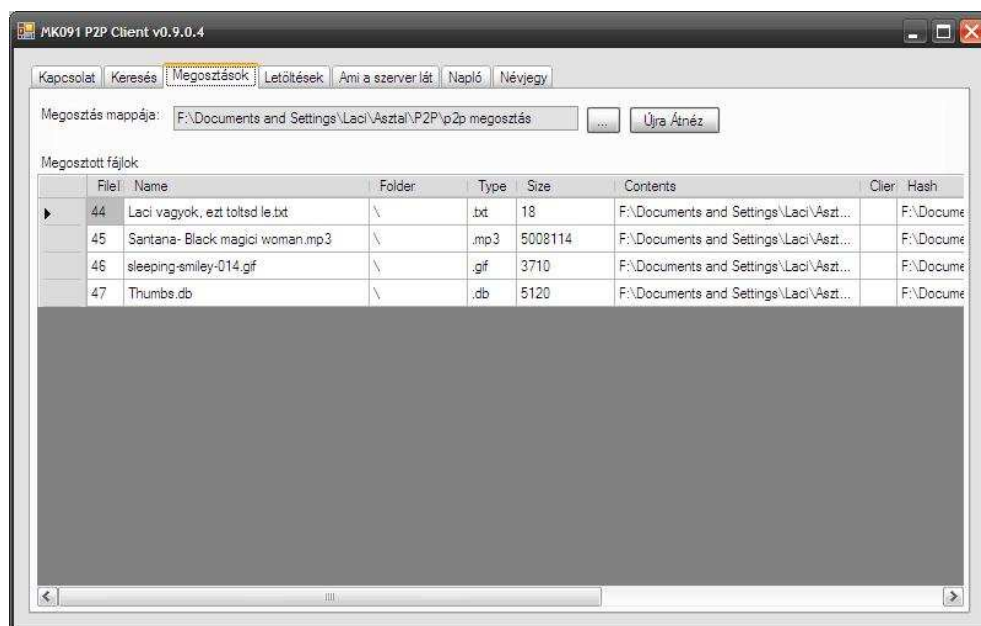
A kliensalkalmazás Windows operációs rendszeren futó telepítője a <http://turul.eet.bme.hu/~mk091/setup/> URL-en keresztül letölthető. Az alkalmazás telepítése után a következő ablak tárul elénk:



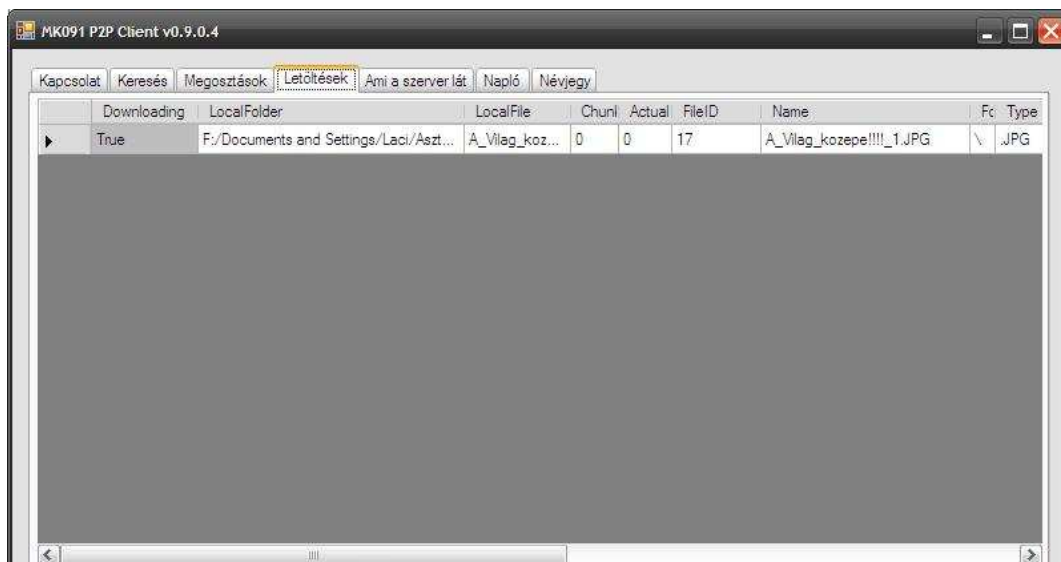
A kapcsolat fülön lehetőségünk van a kliens és a szerver közti kapcsolat menedzselésére. Állíthatjuk, hogy új kliens ID-val szeretnénk bejelentkezni, vagy a legutóbb használt ID-val. Beállítható a szerver URL és port, ahová kapcsolódnunk szeretnénk, valamint a szervernek küldött heartbeat-ek küldési frekvenciája.



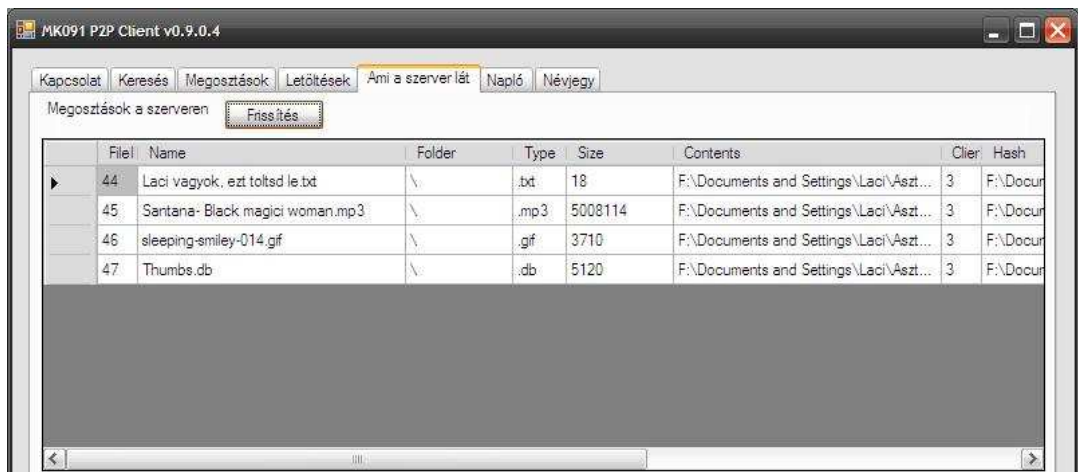
A keresés fülön kereshetünk az online felhasználók megosztásai között, a keresési eredményekre duplán kattintva megkezdhetjük a kívánt fájl letöltését a megosztást végző kliensektől.



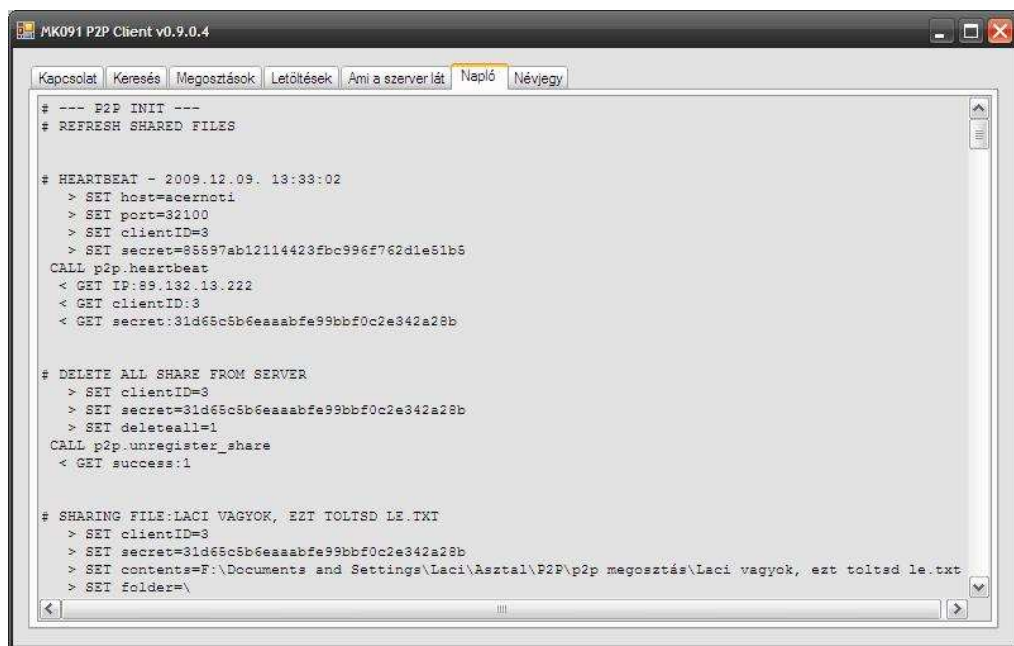
A megosztások fül szolgál az általunk megosztani kívánt fájlok elérési útjának megadására. Egy mappa beállítására van lehetőségünk, melyben található fájlok (az almappákban található fájlokkal együtt) megosztásra kerülnek. Ha a megosztási mappában lévő fájlok megváltoznak, a megosztásokat az „Újra Átnéz” gombbal frissíthetjük.



A letöltések fülön a letöltött fájlok, illetve a letöltés alatt álló fájlok listáját láthatjuk. Arról, hogy a letöltés befejeződött-e a „Downloading” oszlop ad információkat.



Az „Amit a szerver lát” nevű fül, azokat a megosztásainkat listázza ki, amelyeket a szerver lát.



A napló fül, a küldött/fogadott üzeneteket és az azokkal együtt küldött/fogadott információkat jelzi számunkra.

3.1.10 Alkalmazási szintű többesadás

Sajnos önálló labor feladataink miatt implementálni már nem volt időnk multicast hálózatunkat, de az alábbiak szerint gondoltuk annak megvalósítását.

A multicast fát a szerver építi, tartja nyilván, illetve frissíti a kliensek heartbeat üzeneteikor. A kliensek IP-jük alapján egy bináris keresőfában lesznek eltárolva úgy, hogy IP-jük alapján egy szám-kulcsot rendelünk hozzájuk, melynek értékével szűrjük be a fába. Az üzenetek válaszára a szerver az aktuális titok és ID információk mellett visszaküldi a kliensnek a fában lévő szomszédjainak elérését.

Egy node leválásával a teljes részfa leszakad a multicast rendszerből. A kieső node-ok problémáját úgy orvosolnánk, hogy a kiesés felismerés után a legközelebbi heartbeat üzenettel ezek az elszeparált csomópontok a fa frissítése miatt újra felfűzésre kerülnek, így legrosszabb esetben a 3. heartbeat ütemben helyreáll az összekapcsolás. (Megj: 2 heartbeat után detektálható egy node kiesése)

Ha egy kliens egy szabályozó üzenetet szeretne multicast küldeni, akkor közvetlenül egy heartbeat után elküldi azt minden szomszédjának egy TTL (Time to Live) számlálóval szomszéd socketlistenerének.

Miután a másik kliens fogadta az üzenetet, a TTL értékét eggyel csökkentve továbbküldi azt saját szomszédjainak, kivéve annak a peer-nek, akitől érkezett.

Az üzenetek továbbítása addig történik, míg $TTL > 0$, ezzel kiküszöbölve az üzenetek végtelen felesleges pattogását.