Szakdolgozat

Torony Csaba

Debrecen

2008

Debreceni Egyetem

Informatikai Kar

Peer-to-Peer hálózatok napjainkban

Témavezető:
Dr. Sztrik János
Tanszékvezető egyetemi tanár

Készítette:
Torony Csaba
mérnök informatikus hallgató

Debrecen

2008

Tartalomjegyzék:

Köszönetnyilvánítás:

Szeretném megköszönni Dr. Sztrik János Tanár Úrnak, hogy elvállalta szakdolgozatom témavezetői szerepét és hasznos észrevételeivel segítette megírását.

1. Bevezetés

Ha a statisztikákat vizsgáljuk, ma több mint 1,4 milliárd ember használja korunk legnagyobb számítógépes hálózatát, az Internetet. Ez a szám napról, napra rendkívül gyorsan emelkedik is. Ebből következtethetünk arra, hogy a Földön átlagosan minden hatodik ember és velük együtt rendkívül sok számítógép és mobil eszköz is kapcsolódik az Internethez. Az a tény, hogy a számítógépek kapacitása nagyrészt kihasználatlan, számomra egyértelművé tette, hogy az ezen erőforrásokat is kiaknázó Peer-to-Peer (a későbbiekben: P2P) rendszerek jelenleg is fontos szerepet játszanak már, de a jövőben kiemelt jelentőségűek lesznek.

Tekintettel a bennük rejlő lehetőségekre kezdtem el én is ezt a témát kutatni. Miután megismerkedtem a P2P hálózatok legújabb csoportját jelentő, az elosztott hash táblán (a továbbiakban: DHT) alapuló alkalmazásokkal is, és azok felhasználási területeivel, figyelmemet arra fordítottam, hogy a rendszerek hatékonyságát megismerjem a kliens - szerver architektúrájú hálózatokhoz képest.

Napjaink talán egyik legégetőbb problémája a felhasználók növekvő száma és a sávszélességigényes szolgáltatások terjedése miatt a sávszélesség ésszerű és hatékony kihasználása. Emiatt is érdeklődöm e téma iránt.

A P2P hálózatok ma leggyakrabban használt területe a fájlcserélés. Az emberek többsége mikor e tényt – egyáltalán – megtudja, rögtön a kalózoldalak és a nagy szerzői jogi károkozás jut eszébe, holott a fájlcserélést nem csak illegális tartalmak közvetítésére használhatjuk, hanem ahogy a későbbiekben látni fogjuk sok minden másra is.

Diplomamunkámban arra törekszem, hogy részletesen bemutassam a P2P rendszerek működést, de a teljesség és lehető legnagyobb részletesség igénye nélkül, hiszen e terjedelem messzemenően kevés lenne hozzá. Egy – egy rendszer akár egy külön dolgozat alapját képezhetné.

2. Kliens-szerver architektúra

A kliens-szerver kifejezést először az 1980-as években használták olyan számítógépekre (PC-kre) amelyek hálózatban működtek. A ma ismert modell a 80-as évek végén vált elfogadottá. A kliens-szerver szoftver architektúra egy sokoldalú, üzenetalapú és moduláris infrastruktúra, amely azért alakult ki, hogy a használhatóságot, rugalmasságot, együttműködési lehetőségeket és bővíthetőséget megnövelje a centralizált, nagygépes, időosztásos rendszerekhez képest.

Kliens

A kliens (angolul client) olyan számítógép, amely hozzáfér egy (távoli) szolgáltatáshoz, amelyet egy számítógéphálózathoz tartozó másik gép nyújt. A kifejezést először önálló programmal nem rendelkező végkészülékekre illetve terminálokra alkalmazták, amelyek legfontosabb szerepe az volt, hogy a hálózaton keresztül kapcsolatba lépjenek az időosztással működő nagygépekkel és elérhetővé tegyék azok szolgáltatásait.

Jellemzői

- Kéréseket, lekérdezéseket küld a szervernek
- A választ a szervertől fogadja.
- Egyszerre általában csak kisszámú szerverhez kapcsolódik
- Közvetlenül kommunikál a felhasználóval, általában egy GUI-n (Graphical User Interface = Grafikus felhasználói felület) keresztül

Kiszolgáló

A kiszolgáló vagy szerver (angolul server) olyan (általában nagyteljesítményű) számítógépet, illetve szoftvert jelent, ami más gépek számára a rajta tárolt vagy előállított adatok felhasználását, a kiszolgáló hardver erőforrásainak (például nyomtató, háttértárolók, processzor) kihasználását, illetve más szolgáltatások elérését teszi lehetővé.

Jellemzői

- Passzív, a kliensektől várja a kéréseket
- A kéréseket, lekérdezéseket feldolgozza, majd visszaküldi a választ
- Általában nagyszámú klienshez kapcsolódik egyszerre
- Általában nem áll közvetlen kapcsolatban a felhasználóval

A kiszolgálókat többféleképpen csoportosíthatjuk, például:

- a funkciójuk szerint, például webkiszolgálók, FTP-kiszolgálók, adatbázis-kiszolgálók;
- a kiszolgált kör alapján, például internetes kiszolgálók, intranetes kiszolgálók;
- a teljesítményük alapján.

A kliens-szerver olyan architektúra, amely elválasztja egymástól a klienst és a szervert, és az esetek nagy többségében egy számítógép hálózaton alakítják ki. A hálózat klienseit és szervereit más néven csomópontnak (angolul node) is nevezhetjük. A kliens-szerver architektúra legalapvetőbb formájában mindössze kétfajta csomópont van, a kliens és a szerver. Ezt az egyszerű architektúrát két szintűnek (angolul two-tier) hívják.

Bonyolultabb architektúrák is léteznek, amelyek 3 különböző típusú csomópontból állnak: kliensből, alkalmazás szerverből (application server) valamint adatbázis szerverből (database server). Ezt három szintű (three-tier) architektúrának hívják, és a leggyakrabban alkalmazott a kliens-szerver megoldások közül. Amelyek kettőnél több szintet tartalmaznak többszintű (multi-tiered) és n-szintű (n-tiered) architektúrának is nevezzük.

A három szintű kiépítésben az alkalmazásszerverek azok, amelyek kiszolgálják a kliensek kéréseit, és az adatbázisszerverek az alkalmazásszervereket szolgálják ki adatokkal. Ennek a rendszernek nagy előnye a bővíthetőség.

A több szintű kiépítés előnye, hogy egyensúlyozza és elosztja a feldolgozásra váró adatmennyiséget és munkát a több és gyakran redundáns, specializált csomópont között. Ez javítja a rendszer teljesítményét, és a megbízhatóságát is, hiszen a feladatok párhuzamosan több szerveren is elvégezhetőek. Hátránya, hogy nagyobb az adatátviteli forgalom a hálózaton és, hogy nehezebben programozható illetve tesztelhető egy kétszintű architektúránál, mert

több eszközt kell összehangolni a kliensek kéréseinek kiszolgálásához.

A kliens-szerver architektúra előnyei

A legtöbb esetben a kliens-szerver architektúra lehetővé teszi, hogy a feladatokat elosszuk olyan számítógépek között, amelyek csak a hálózaton keresztül érintkeznek egymással, ami megkönnyíti a karbantartás elvégzését. Megoldható például, hogy javítsunk, frissítsünk, áthelyezzünk vagy akár kicseréljünk egy szervert anélkül, hogy klienseire ez bármilyen hatással lenne. Ezt a változtatásoktól való függetlenséget információ elrejtésnek vagy angolul encapsulation-nek nevezik.

Az összes adat a szerver(ek)en tárolódik, amelyek általában sokkal erőteljesebb biztonsági ellenőrzéssel rendelkeznek, és jobban tudják szabályozni az erőforrásokhoz és adatokhoz való hozzáférést.

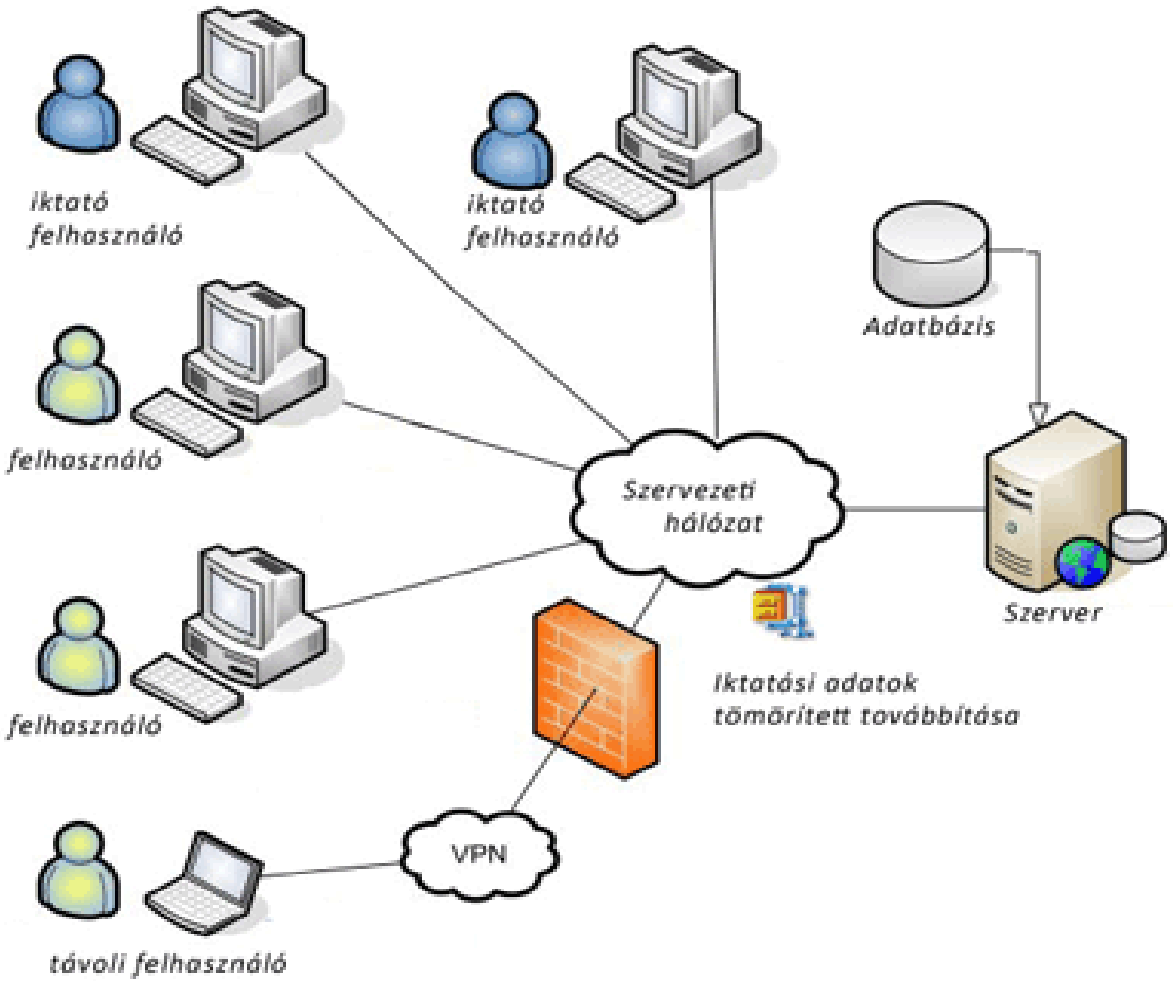
Mivel az adattárolás centralizált, könnyebb frissíteni az adatokat, mint amennyire ez egy P2P rendszerben lehetséges lenne. Utóbbi architektúrában több száz, vagy ezer résztvevő gépen kell megoldani az adatok megváltoztatását, ami időigényessége mellett a hibák előfordulásának lehetőségét is megnöveli.

Az architektúra hátrányai

A forgalomtorlódás a hálózaton a kliens-szerver paradigma kezdete óta egy nagy kérdés. Ahogyan az egyidejű klienskérések száma - egy adott szerver felé - növekszik, a szerver egyszerűen túlterheltté válhat. Jó kontraszt ez a P2P hálózatokkal szemben, ahol a sávszélesség annál jobban nő, minél több csomópont kapcsolódik a hálózathoz, hiszen a P2P hálózat teljes sávszélességét - durván - a kapcsolódott csomópontok sávszélességének összegeként kapjuk.

A kliens-szerver paradigmából hiányzik a jó P2P hálózatra jellemző erőteljesség. Ha egy központi, kritikus szereppel bíró szerver elromlik az ügyfél-kiszolgáló architektúra esetén, az ügyfelek kéréseit nincs, ami teljesítse. Ezzel szemben a P2P hálózatokban az erőforrások a sok csomópont között megoszlanak. Még akkor is, ha egy vagy több csomópont

elhagy egy letöltést, a megmaradó csomópontoknál még meg kellene lenniük azoknak adatoknak (adatszeleteknek), amelyek hiányoznak és a letöltés befejezhető.



1. ábra: Kliens-szerver modell

3. P2P hálózatok és alkalmazások

Mielőtt a rendszer hatékonyságát vizsgálnánk és különböző példákon keresztül bemutatnánk a benne rejlő lehetőségeket, fontos látnunk, hogy mit is takarnak a „P2P hálózatok”, a „DHT alapú hálózatok” mint fogalmak, valamint hogy ezeknek milyen alkalmazási lehetőségei és hiányosságai vannak.

Napjainkban az elosztott rendszerek jelentősége megkérdőjelezhetetlen. Az elosztott rendszereken, mint általános csoporton belül is a közismert többrétegű architektúrák mellett egyre inkább tért nyernek az egyenrangú résztvevők együttműködésén alapuló, úgynevezett Peer-to-Peer (P2P) rendszerek.

A P2P hálózatok az ügyfél-kiszolgáló kapcsolathoz képest jelentősen eltérő módon működnek: a szerepek nincsenek előre meghatározva; többnyire követelmény is, hogy az összes résztvevő képes legyen valamilyen erőforrást a rendszer egésze számára elérhetővé tenni viszonzásképp az általa igénybevett szolgáltatásokért. Az így megosztható erőforrások általában a következő három kategóriába sorolhatók: fájlok, számítási kapacitás, felhasználói jelenlét (legegyszerűbb esetben csevegés).

3.1 Korai P2P hálózatok

Amikor a mai Internetre gondolunk, akkor egy, az egész világot körülölelő információs hálózat jut az eszünkbe, melyen egymástól földrajzilag elkülönülő számítógépek illetve mobil eszközök, számos különböző céllal kommunikálhatnak egymással. A legtöbb feladatra egy-egy elterjedt protokollt vesznek igénybe. Mivel ezek elsősorban feladat-specifikus eszközök, így a megvalósításban jelentős eltéréseket mutatnak. A köztük fennálló rokonságot azonban lehetetlen letagadni, ugyanis koncepcionálisan azonos alapokon nyugszanak. A ma legelterjedtebb protokollok, mint például a HTTP, az FTP, vagy az SMTP, mindannyian a mára klasszikussá vált kliens-szerver architektúrára épülnek. Ez annyit jelent,

hogy léteznek kiemelt, szervereknek nevezett számítógépek a hálózatban, melyek a kliensektől származó kérésekre várakoznak, és amikor ilyen megkeresések érkeznek, akkor megpróbálják azokat kiszolgálni. Ezzel a szemlélettel ellentétben, a P2P architektúránál megszűnik minden ilyen megkülönböztetés, így a hálózatba bekapcsolódott valamennyi számítógép egyenrangú, és azonos feladatokat lát el.

A P2P rendszerek az aktuális feladattól függetlenül egy úgynevezett overlay („fölöttes") network létrehozásával működnek. Erre azért van szükség, mert ezek a rendszerek alapvetően szomszédossági alapon működnek: hogy ne legyen szükség, kitüntetett szerepű résztvevőkre, ugyanakkor elegendő legyen, hogy minden node (csomópont) csak korlátos számú másikkal tartson kapcsolatot, az üzenetek továbbítása logikai pont-pont kapcsolatokon keresztül történik.

Amennyiben a Peer-to-Peer hálózatok történetét vizsgáljuk, egészen a mai Internet keletkezéséig vissza kell nyúlnunk. Talán már sokan hallottak az ARPANET nevű hálózatról, mely az Internet, tágabb értelemben pedig minden mai csomagkapcsolt hálózat ősének tekinthető. A legtöbb, a világháló történetével foglalkozó írás, könyv, tanulmány megemlíti, elmondja róla, hogy 1969-ben, az amerikai védelmi minisztérium megbízásából hozták létre. Esetleg arra is kitérnek, hogy a létrehozásával egy olyan hálózat megalkotása volt a cél, mely működőképes marad akár egy, az Amerikai Egyesült Államokat célzó, összehangolt atomcsapást követően is. Arról azonban kevesen beszélnek, hogy már ez a kezdeti hálózat is P2P alapokon nyugodott. Kezdetben ugyanis a két, majd a négy csomópontból álló hálózatban nem voltak kiemelt résztvevők, mindegyik egyenrangúnak számított.

Az Internet fejlődése és növekedése során - elsősorban gazdasági és technológiai megfontolásokból - egyre inkább eltért a kezdeti szemlélettől és elmozdult a szerver-kliens architektúra irányába. Ez elsősorban arra vezethető vissza, hogy a 70-es, 80-as években a nagyobb terhelést is elviselő, szervernek alkalmas eszközök ára olyan magas volt, hogy azt nem engedhette meg magának más, csupán a hadsereg, az egyetemek, kutatóintézetek, később pedig azon vállalkozások, melyek az Internetet befektetésre érdemes területként kezelték.

A 90-es évek végére, a technológia fejlődése és az alkatrészek árának esése megteremtette a lehetőséget arra, hogy egyes alkalmazások visszatérjenek a kezdeti koncepcióhoz, a P2P architektúrához. Ezzel próbálták meg az addig perifériára szorult számítógépekben rejlő lehetőségeket, s ezzel együtt a parlagon heverő erőforrásokat is kiaknázni. Erőforrások alatt több mindent értünk, mint például hálózati, vagy számítási kapacitást, tárhelyet, és így tovább. Ha a hálózatot még mindig kizárólag kutatók és amatőr programozók használták volna, akkor talán ennyi motiváció is elég lett volna, hogy a P2P új erőre kapjon. A hosszú évek azonban teljesen átalakították az Internetet használók összetételét. Az átlagos felhasználót nem nagyon érdekelte a lehetőség, hogy akár egy kicsit is több számítási kapacitás fölött rendelkezhessen, ennél sokkal többre értékelte az Interneten fellelhető információt, valamint szórakoztató funkcióit. Az 1995-ben megjelent MP3 formátum, mely lehetővé tette a zenei fájlok tömörítését és ezzel együtt a hálózaton keresztül történő terjesztését, megadta azt, amire a puszta technikai előny nem volt képes. Az átlagos felhasználó számára is megmutatta a P2P hálózatok értelmét, és elindította a P2P fájlcserélés máig meg nem szűnő hullámát.

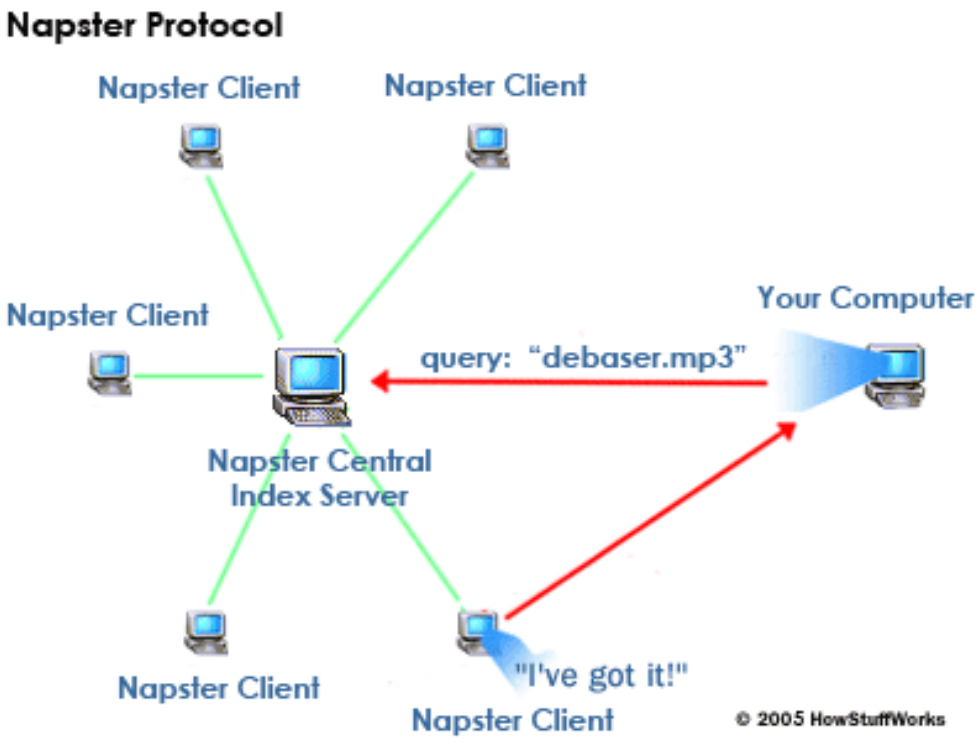
A P2P rendszerekben az útvonalválasztás, ill. a protokoll nem választható el az alkalmazástól, hiszen egy fájlmegosztó alkalmazás esetében elegendő, ha legalább egy példányt megtalálunk egy adott állományból, míg egy csevegés-jellegű (chat, whiteboard, vagy bármilyen applikáció-megosztás) együttműködés esetében értelemszerűen szükséges, hogy az összes résztvevőhöz eljussanak az események. Ezen okok miatt általános protokollszintű optimalizációról nem beszélhetünk.

3.1.1 Napster

Az „újgenerációs” P2P alkalmazások egyik úttörő, de egyértelműen az első igazán elterjedt képviselője Napster névre hallgatott. Az 1999-ben induló rendszer elsődleges célja a felhasználók számítógépein elhelyezett zenei fájlok megoszthatóvá tétele volt. Ennek köszönhetően nagyon gyorsan kiváltotta a zeneipar képviselőinek ellenszenvét, amelyet számos, főként jogi támadás követett. A kialakult háborúnak köszönhető az is, hogy a mai napig ördöginek állítják be a P2P hálózatokat, annak ellenére, hogy nem minden ilyen alkalmazás irányul kizárólag a fájlcserére.

A rendszer technológiailag hibridnek volt nevezhető. Igaz ugyan, hogy a felhasználók egymás között tudtak fájlokat mozgatni, azonban a keresés még centralizáltan, egy szerverfarmon belül történt. Ebből a központosítottságból adódóan a rendszer számos problémával küzdött. A felhasználók számával együtt növekedő leterheltséget nagyon nehezen tudták fejlesztésekkel kompenzálni, ugyanakkor a szerverek a hálózat többi részétől történő elvágása pedig képes volt teljesen meg is bénítani a rendszert. A perek következményeit már nem tudta kiheverni az egyébként még ma is működő hálózat. A bíróság kötelezte a közvetített tartalmak szűrésére, így a felhasználók lassan elpártoltak az immáron félkarúvá vált óriástól.

A későbbi rendszerek már tanultak a Napster hibáiból és megpróbáltak elkerülni minden olyan megoldást, mely technológiailag ilyen labilis volt, vagy amely jogilag ennyire könnyen támadhatóvá tette volna működésüket.



2. ábra: A Napster protokoll működési elve

3.1.2 Gnutella

Nehéz megállapítani, hogy mely hálózat lett a Napster sikerének igazi örököse. Az viszont biztos, hogy a Gnutella névre hallgató alkalmazás kitöltötte az elődje visszaszorítása után támadt űrt. Okult is annak hibáiból, így az új rendszer már teljesen elosztottan, központi irányítás nélkül működött. Míg a Napster a már említett, és a gyenge láncszemnek bizonyuló szerverfarmján keresztül végezte a kereséseket, addig az új hálózat mindezt egy elárasztásos algoritmussal valósította meg. Igaz, hogy ez a megoldás jogilag támadhatatlanná tette, de technikailag nem bizonyult a legkörültekintőbb megoldásnak. A rendszer méretének növekedése során ugyanis jelentősen megnőtt a felhasználók által generált kérések száma, amely néha jelentős mértékben leterhelte, s ezzel együtt le is lassította a hálózatot. A megoldás keresése során kialakítottak egy többszintű hierarchikus hálózatot, amely a keresés problémáját már többé-kevésbé képes volt áthidalni. Ennek értelmében felosztották a hálózatot kiemelt csomópontok (supernode) között, majd ezekre a területekre korlátozták az elárasztást. Az így kapott zónák közötti kommunikáció a supernode-ok közreműködésével zajlott és így korlátozható maradt.

A Gnutella a kevéssé számításigényes protokollok közé tartozik, a P2P hálózaton belül broadcast-ot valósít meg. Az egyes kérések által generált forgalmat az úgynevezett TTL (time to live, hátralevő élettartam) paraméterrel korlátozza, ami általában 7-ről indul, és minden továbbításkor csökken eggyel (mikor eléri a 0-t, a csomag nem kerül továbbításra). Az egyszer már feldolgozott csomagok újraprocesszálásának (és ami még lényegesebb: újratovábbításának) elkerülése érdekében általános módszer az utoljára látott n csomag azonosítójának megőrzése. A Gnutella ezt az információt a forrás összeköttetéssel együtt tárolja, és a válaszok back-route-olására is felhasználja.

A Gnutella kérések ugyanis nem tartalmaznak IP címeket, csak a válaszok. Ez egyfajta névtelenséget biztosít a résztvevők számára, bár megkérdőjelezhető hasznosságú ez a névtelenség.

A mai Internet közönsége érthető okokból bizalmatlan. Ennek az egyik leggyakoribb megnyilvánulása a tűzfalak használata, amelyek igen erőteljesen elhatárolják a kiszolgálókat

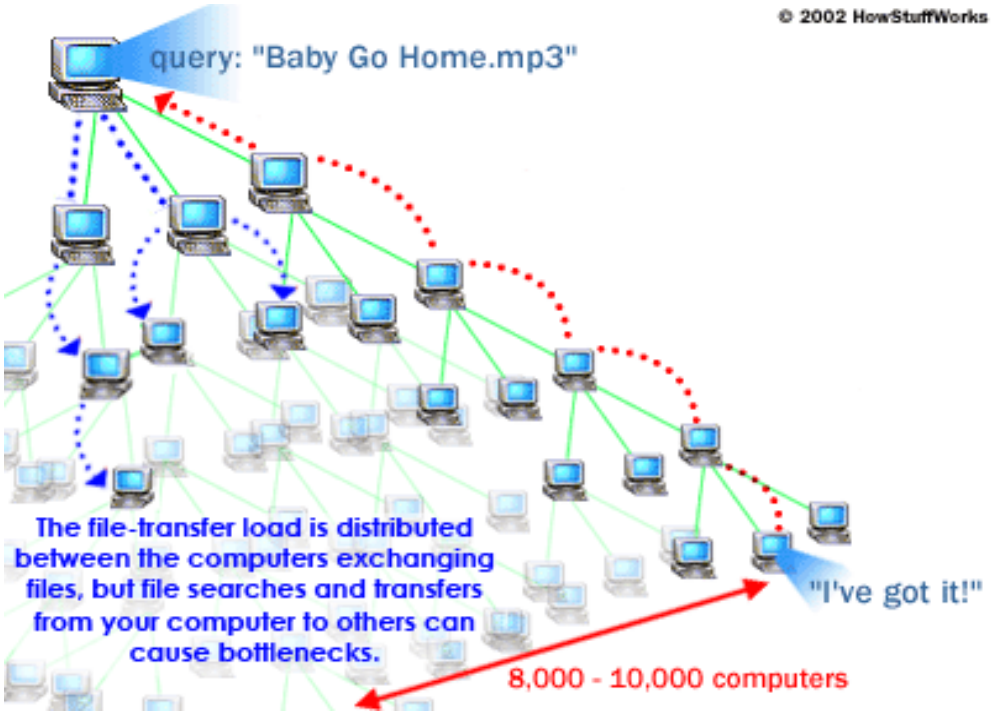
és az ügyfeleket. Ez egy olyan alapvető probléma, amellyel valamilyen módon minden P2P rendszernek számolnia kell: kapcsolatlétrehozás szempontjából a tűzfalak egyirányúak. Az overlay network szempontjából tehát szükséges olyan node-ok jelenléte, amelyek képesek kapcsolatokat fogadni, nincsenek tűzfal mögött. A nagyobb probléma a fájlátvitel támogatása jelenti; ezt a Gnutella akkor bírja megoldani, ha a két node közül legfeljebb az egyiket védi tűzfal. Ugyanis – ha a szerver szerepet betöltő csomópont van tűzfal mögött – lehetőség van az overlay network-ön belül egy úgynevezett Push Request továbbítására, melynek hatására a fájlátvitelt a szerver fogja megindítani, így a tűzfal – rendszeradminisztrátorok által tipikusan nem támogatott – átkonfigurálására sincs szükség, és mégis létrejöhet az áttöltés.

Topológiák:

A Gnutella specifikáció a legteljesebb mértékben nyitva hagyja a hálózatépítés és egyáltalán, a logikai hálózatban megvalósítandó célszerű topológia kérdését. Négy alap topológiát mutatnék meg röviden.

- Random Mesh: teljesen véletlenszerű gráf, tehát megengedjük izolált node-ok jelenlétét is
- Connected Mesh: található a gráfban olyan részfa, amely minden csomópontot tartalmaz
- Semi-Random Mesh: véletlenszerű gráf, de minden csomópont rendelkezik legalább egy kapcsolattal
- Connected Stars: szoftver szempontból továbbra is él a homogenitás, de az erőteljesebb hardverrel rendelkező résztvevők felismerése segítségével létrehozható egy alhálózat, amelyen a többi csomópont csupán 1-1 kapcsolaton át, levélként függ

Ezt a megoldást is számos másik követte, azonban a fájlcserélő alkalmazások egyik legnehezebben megoldható problémája továbbra is a keresés maradt. Míg egyes próbálkozások a Gnutella kereséséhez hasonló algoritmusokkal hidalták át a problémát, akadtak olyanok is, amelyek visszakanyarodtak a már bevált módszerekhez és a hálózattól elkülönítve végezték el az állományok katalogizálását, kereshetővé tételét.



3. ábra: Gnutella letöltési folyamat

3.1.3 DirectConnect

A DirectConnect egy peer-to-peer (P2P) fájlcserélő alkalmazás, melyet a NeoModus munkatársa, Jon Hess írt és 2001 második felében jelent meg. Ma ezzel a névvel rendszerint az általa használt hálózati protokollt illetik.

A Direct Connect hálózat nem annyira decentralizált, mint a Gnutella vagy a FastTrack, mivel hubokat használ különböző felhasználói csoportok összekapcsolására. Egy hubra általában a hasonló érdeklődésű userek kapcsolódnak. A hub csak akkor enged be, ha megfelelünk bizonyos feltételeknek. Régebben ez csak a megosztott adatok mennyiségét jelentette, tehát egyes hubokra csak akkor léphettek be a felhasználók, ha megosztottak pár gigabyte adatot a gépükről. Ma már azonban lehetőség van másfajta feltételek megszabására és ellenőrzésére is, például már az is beállítható, hogy mennyi legyen a hubon való kereséshez szükséges megosztás. Tetszőlegesen sok hub létezik elszórva a világban, ezek egymástól függetlenek. A kliens egyszerre több hub-hoz is csatlakozhat, egy hub-on a kliensek száma a hub kapacitásának megfelelően korlátozott. Jelentős eltérés a Napster központjához képest, hogy a hub-ok nem tartják nyilván a csatlakozott felhasználók megosztásait, nem végeznek

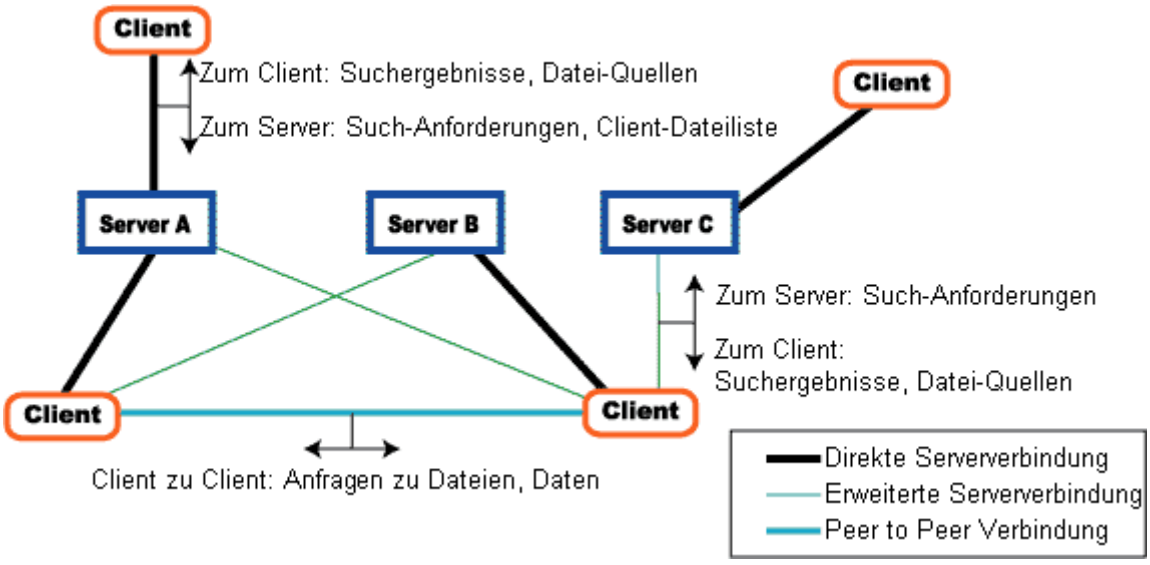
keresést, csak továbbítják a kérést a kliensek fele, így kevésbé terheltek. A korai DirectConnect rendszerekben az azonos állományok azonosítása még nem megoldott, a kliens alkalmazások a felhasználóra bízták az azonosság megállapítását, egyetlen feltétel az azonosnak ítélt állományok megegyező mérete volt. A chat kezdettől fogva integrálva van a protokollba és így a hubok kis közösségeket alkotnak, tehát többet jelentenek a névtelen fájlcserénél. Ez jóval "barátságosabbá" teszi a DC használatát.

A hubokon futó szoftverek lehetővé teszik úgynevezett operátor felhasználók létrehozását. Ezek a felhasználók felügyeleti joggal rendelkeznek a hub felett, tehát módosíthatják a beállításokat, és figyelmeztethetik, illetve legvégső esetben kirúghatják a hubról a szabálytalanul viselkedő felhasználókat. Továbbá a legtöbb hubszoftverben lehetőség van VIP felhasználók létrehozására, akikre általában nem érvényesek a hub korlátozásai (például nem kell adatokat megosztaniuk a belépéshez).

Ahhoz, hogy a DC hubok szolgáltatásait használhassuk, szükség van egy DC kliensre. A legelső DC kliens az eredeti NeoModus-féle DC kliens volt, de ma ezt már egyáltalán nem használják. A ma használatos kliensek alapját a DC++ képezi, aminek a felhasználói felülete jelentősen különbözik az eredeti NeoModus kliensétől. A DC++ egyszálas kliens, vagyis egy fájlt egyszerre csak egy felhasználótól tud tölteni. A kliens alkalmazások lehetővé tették a megszakadt letöltések folytatását más azonosnak ítélt forrásból, ellenőrizve, hogy a már letöltött rész vége megegyezik az új forrásállományban. A DC kliensek egy fejlettebb csoportját képezik a többszálas kliensek, amik egy fájlt egyidőben több felhasználótól is képesek tölteni, ezáltal a sebesség jelentősen megnövekedhet. Ezek a többszálas kliensek nem sorban töltik le a fájlokat, hanem kis részekre bontják azokat, így két különböző részt két különböző felhasználótól tudnak letölteni. A legelterjedtebb többszálas kliens a StrongDC++, és akárcsak a DC++ esetében, a StrongDC-ből fejlesztették ki a többi többszálas klienst. Ezeket a klienseket folyamatosan fejlesztik, nem lehet megmondani, hogy melyik a legjobb közülük. Vannak továbbá kifejezetten operátorok számára készített kliensek is. A választék nagyon nagy, és mindenkinek saját magának kell eldöntenie, hogy melyik kliens felel meg legjobban az igényeinek. Az összes DC++ és StrongDC-alapú kliens nyílt forráskódú, és ingyenesen letölthetők, általában a fejlesztők weboldaláról. Van már Linuxon működő DC kliens is, sőt a Debian Linux disztribúcióval alapértelmezésben már DC kliens is jár.

3.1.4 eDonkey, eMule

Az eDonkey hálózat a korábbi megoldásokhoz hasonlóan több kisebb szervert használ, amelyekhez csatlakoznak a kliensek. A szerverek indexelik a megosztott állományok neveit, így gyors keresés valósítanak meg, valamint a szerverek egymással is kommunikálnak. Jelentős eltérés a korábban ismertetett rendszerektől, hogy az állományok letöltését hatékonyabban oldja meg. A rendszer az állományok azonosítására determinisztikus algoritmussal generált hash értékeket használ, amely csak az állomány tartalmától függ, ezzel automatizálható az azonos források keresése. A letöltés során az állomány kb. 9MB-os részekre tagolódik, és a különböző részeket más-más felhasználótól is be lehet szerezni, így egy állományt egyszerre több felhasználótól lehet tölteni, ezzel jelentősen növelve a letöltési sebességet. A letöltött részek hibátlanságát egy, a részre jellemző hash érték segítségével lehet ellenőrizni. A kész részek automatikusan megosztódnak a rendszerben, még az egész állomány befejezése előtt. A kliensek igényelik a még hiányzó részeket az állománnyal rendelkező többi klienstől, így várakozási sorok alakulnak ki.



4. ábra: eDonkey működése

5. ábra: A jól ismert eMule logo

Az **eMule** a számítástechnikában az egyik legnépszerűbb peer-to-peer fájlcserélő alkalmazás. Az eDonkey hálózatot használja, de többre képes, mint a hivatalos eDonkey kliens.

Megkülönböztető sajátossága az eMule-nek, a kliens csomópontok közvetlenül is tudnak forrásokat cserélni egymással, képes gyorsan helyreállítani a hibásan letöltött fájlokat, egy pontrendszer segítségével jutalmazza a gyakori feltöltőket, valamint Zlib tömörítéssel továbbítja a fájlokat, hogy takarékoskodjon a sávszélességgel. Ezenkívül képes böngészőből fogadni az „ed2k" linkeket, és letölteni az állományokat, amelyre ezek a linkek hivatkoznak. Ezek a linkek biztosítják, hogy az állomány valóban az, amire a neve utal, nem pedig egy azonos nevű hamisítvány („fake"). Az utóbbi időben sok ed2k-linkeket közzétevő oldalt leállíttattak a hatóságok, mivel hogy az általuk kínált linkek sértik a szerzői jogokat.

Az eMule nyílt forrású, a GNU (Genereal Public License) alá eső szoftver. A Windows operációs rendszerre készült, de van Linuxos (xMule) és platformfüggetlen (aMule) leágazása is.

Az újabb változatok a Kademlia protokoll révén központi szerver nélkül is képesek működni.

3.1.5 BitTorrent

A BitTorrent protokoll, melyre Európa talán legelterjedtebb P2P fájlcserélő alkalmazásai is épülnek, szintén ez utóbbi kategóriába tartozik. Ez a hálózat kezdetben

kizárólag a fájlok mozgatását oldotta meg, a kívánt tartalmakat más úton kellett megkeresni. Erre a célra talán a legegyszerűbb megoldást alkalmazták, miszerint a publikálni kívánt állományokról készült leíró, úgynevezett torrent fájlokat, weboldalakon terjesztették. Az alkalmazás sikere elsősorban a protokollba beépített letöltési hatékonyságot fokozó szabályoknak köszönhető.

Fájl megosztása:

Fájl megosztásához létre kell hozni egy *torrent* típusú (kiterjesztésű) fájlt, amely az alábbiakat tartalmazza:

- a letöltendő fájl(ok) neve, mérete, és minden egyes fájldarabka ellenőrzőösszege (ezzel lehet ellenőrizni, hogy nem sérült-e a fájldarab),
- a tracker-szerver címe,
- néhány egyéb adat.

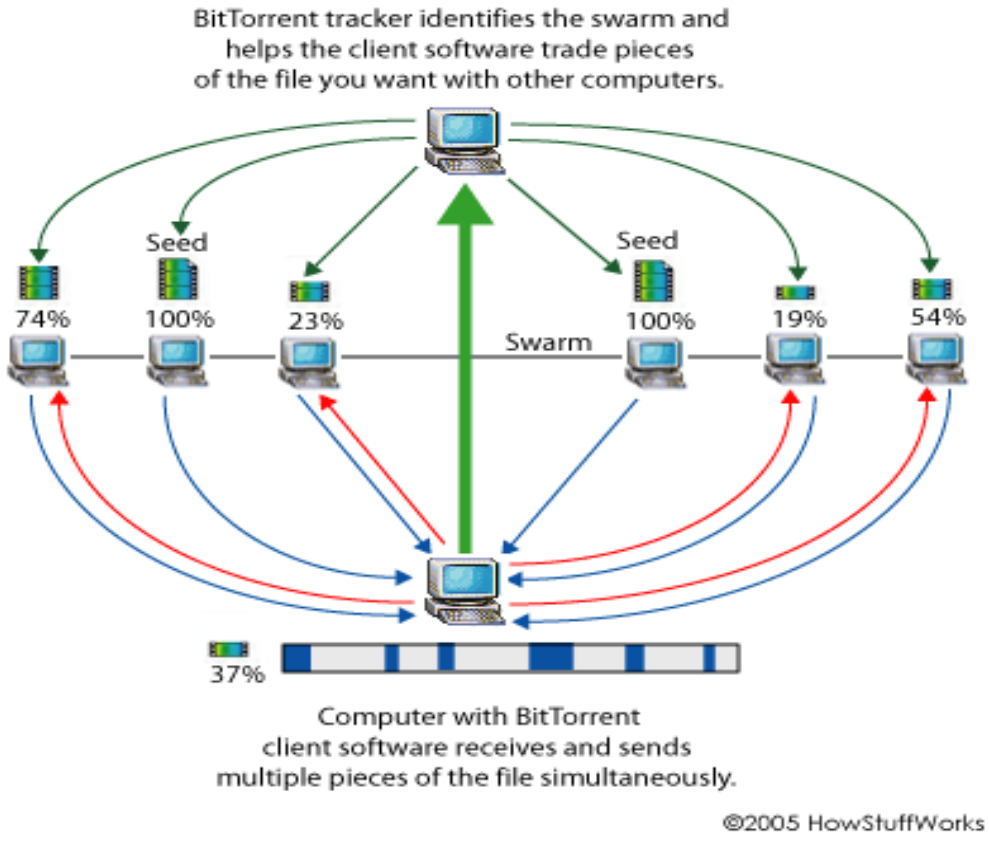
A .torrent fájl nagy előnye, hogy könnyű másokkal megosztani, mert kicsi a mérete, ezért könnyedén felhelyezhető egy honlapra, vagy elküldhető e-mail mellékleteként.

Miután a .torrent fájl generálása megtörtént, a seed fogadja a kéréseket, és kiszolgál más klienseket, amelyek a .torrent fájl alapján felkeresik, többnyire a tracker segítségével. Ez pontos információval rendelkezik arról, hogy mely fájldarabok kinél találhatók meg. A kliens ez alapján felméri, hogy mely darabka a legritkább, és megpróbálja azt letölteni, - ezzel az eljárással biztosítva azt, hogy az állomány, a forrás kilépése esetén is, a lehető legnagyobb valószínűséggel maradjon elérhető - majd ha az megvan, a következő darabot keresi meg. Így a csomópontok rá vannak kényszerítve a megszerzett darabok cserélgetésére, ezzel tehermentesítik az eredeti forrást. Megoldották azt is, hogy minden megszerzett rész letöltése után azonnal feltölthetővé is vált, ezzel törekedve a globális értelemben vett sebesség maximalizálásra. Miután a kliens megszerzett egy darabot, az ellenőrzőösszeg alapján ellenőrzi integritását, ezzel elkerülhető, hogy hibás fájldarabok keringjenek a hálózaton. Ez bekövetkezhet véletlen hibával, de történtek szándékos mérgezések bizonyos zene- és filmkiadók, illetve szerzői jogvédő hivatalok részéről. Ha a kliens teljesen letölti a fájl(oka)t, seeddé válik.

Hátránya a módszernek, hogy ha a seedek nem szolgáltatják tovább a torrentet, akkor az esetleg már nem elérhető a publikált .torrent fájlok ellenére. Esetleg csak fájldarabok keringenek a hálózaton, anélkül, hogy egésszé lehetne azokat összeállítani.

A letöltések optimalizációja érdekében azoknak kellene elsőként letöltési lehetőséget kapni, akik megelőzőleg sokat töltöttek fel. Ehhez minden felhasználóhoz számon kell tartani a letöltés/feltöltés arányt. A protokoll jelenleg ilyen szempontból még nem tökéletes, hiszen a tracker csak a klienstől tudja, hogy mennyit töltött fel, illetve le – a kliens pedig elvileg akár hamis adatokat is küldhet.

Az alkalmazás újabb verziói azonban már ennél is továbbléptek. A keresést egy merőben új megoldás, az úgynevezett DHT alapú hálózatok segítségével, teljesen elosztottan valósították meg. Ennek értelmében a hálózatba bekapcsolódott számítógépeken is el lehetett érni az azokra korábban elmentett torrent fájlokat, ezzel kínálva egy alternatív lehetőséget az eddig kizárólag weben alapuló megoldás mellett.



6. ábra: Bittorrent P2P letöltési folyamat

3.2 DHT alapú hálózatok

A korábban bemutatott, úgynevezett strukturálatlan P2P hálózatok mindegyike rendelkezett valamilyen hiányossággal, amely elsősorban a keresésekben nyilvánult meg. A BitTorrent esetében, mint láthattuk, ez a funkció egyszerűen hiányzik a protokollból. A Napster-nél mindez egy központosított, könnyen támadható módon, a Gnutella esetében pedig egy túlzott hálózati leterheltséget okozó, teljesen elosztottan működő megoldással valósult meg. Elmondható, hogy ezek a hálózatok nem tudtak megnyugtató választ adni a felmerült problémára. Jogosan merült fel tehát az igény, egy, a csomópontokon való strukturált adattárolást lehetővé tevő, teljesen elosztottan működő, skálázható és robosztus, új hálózati megoldásra. Ezeket az elvárt igényeket szem előtt tartva születtek meg az elosztott hash táblán alapuló P2P hálózatok.

A különböző megvalósítások közös alapelve, hogy a résztvevő csomópontok, valamilyen kriptográfiai hash algoritmus segítségével, egy új, a hálózaton belüli azonosításra szolgáló címet kapnak. Tartalma alapján ugyanebbe a címtérbe képzik le az elhelyezett adatokat is, majd valamilyen - hálózatonként különböző - eljárás szerint az egyes csomópontokhoz rendelik azokat. Ha a későbbiekben valamilyen kérés érkezne az egyik résztvevőhöz, annak már nincs más feladata, mint továbbítani azt az információért felelős csomópont felé, illetve amennyiben ő az, úgy válaszolni rá.

A kriptográfiai hash függvények alkalmazása több szempontból is jó választásnak bizonyult. Egyrészt különböző méretű tartalmakból képes egyenlő hosszúságú lenyomatokat képezni, amely ezen hálózatok esetében lehetővé teszi az állományok csomópontokhoz való rendelését, másrészt rendelkeznek egy rendkívül pozitív tulajdonsággal, amely értelmében egymáshoz nagyon hasonló bemenetek hatására nagyon különböző kimenettel szolgálnak. Ezt a lehetőséget kihasználva biztosítható, hogy a rendszerben elhelyezett állományok azonosítóit szétszórjuk a címtérben, amely nagyfokú terhelésmegosztást biztosít a csomópontok között.

Tekintettel arra, hogy ezek a hálózatok teljesen elosztottan, központi vezérlés nélkül működnek, megbízható módon kell biztosítani a résztvevők menedzselését is. Lehetővé kell tenni új csomópontok felvételét a hálózatba, a tagok működőképességének ellenőrzését,

valamint a hálózatot szándékosan elhagyó, vagy abból meghibásodásból fakadóan kieső résztvevők kezelését.

Biztosítani kell továbbá a rendszeren belüli üzenetek útvonal-irányítását, amihez elengedhetetlen feltétel az, hogy az egyes csomópontok ismerjenek további tagokat is, amelyek adatbázisát rendszeres frissítésekkel naprakészen kell tartani. Ugyanakkor szem előtt tartva a rendszerrel szemben támasztott elvárásokat, nem feledkezhetünk meg egy kritikus követelményről, a skálázhatóságról sem. Ebből fakadóan senki sem rendelkezhet információval a hálózat teljes egészéről. Különösen igazzá válik ez egy ilyen dinamikusan változó környezetben, ahol előre meg nem jósolható módon jelennek meg új tagok és távoznak - akár előzetes figyelmeztetés nélkül - korábban a hálózatban szereplő csomópontok.

Az egyes DHT alapú hálózatok között mutatkozó alapvető különbségek a hash tér értelmezésében, felosztásában, valamint a csomópontok által kezelt útvonal-irányítási táblák szerkezetében jelentkeznek.

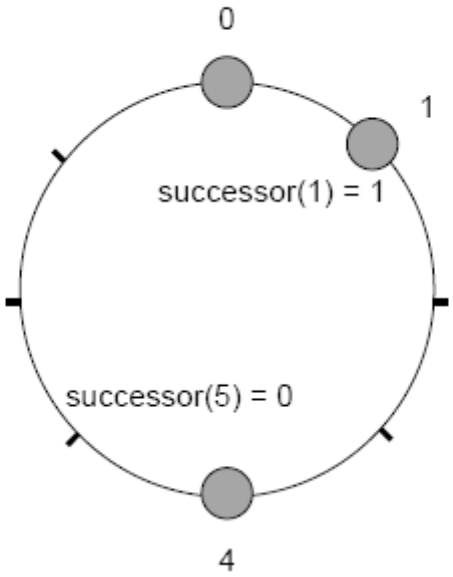
Az alábbiakban bemutatom a legfontosabb DHT alapú P2P hálózatok sajátosságait.

3.2.1 Chord

A Chord esetében mind a csomópontok, mind az elhelyezett információk egy m bites azonosító kulcsot kapnak, mely [0, 2^m-1] intervallumból vehet fel értéket. Ez az előbbi esetben a résztvevő IP címének, vagy egyéb egyedi azonosítójának, utóbbi esetben az elhelyezett állomány tartalmának egy kriptográfiai hash függvénnyel képzett lenyomatát jelenti. A rendszer kidolgozói erre az SHA-1 algoritmust javasolják, mely esetben ez az azonosító 160 bites. A csomópontok, az így kapott azonosítók alapján növekvő sorrendben, gyűrű topológiába rendeződnek, vagyis ebben az értelemben a 1-es azonosító nagyobb, mint a 0-ás, ami pedig nagyobb, mint 2^m-1.

Amennyiben adatot helyezünk el a hálózatban, úgy azt csupán el kell juttatnunk a későbbiekben a tárolásért felelős csomóponthoz, amely a későbbiekben érkező, a tárolt információra vonatkozó kéréseket hivatott kiszolgálni. Minden tárolt adatért az a csomópont lesz a felelős, amelyik azonosítója a legkisebb az adaténál nagyobb azonosítók közül. Ezt a függvényt, amely a felelős csomópontot meghatározza, successor-nak nevezzük.

Egy egyszerű Chord gyűrű látható az 7. ábrán, m=3 érték mellett. A kép azt az esetet mutatja be, amikor a 8 elemű címtérben 3 csomópont található, mégpedig a 0-ás, az 1-es, valamint a 4-es azonosítókkal. Ekkor azon információ, amelyre a lenyomatképző függvényt alkalmazva 1-es értéket kapunk az 1-es, míg például 5-ös érték esetében a 0-ás azonosítóval rendelkező csomópont lesz a felelős.



7. ábra: Chord gyűrű *m* paraméter *m=3* értéke esetén

Az alkalmazott lenyomatképző hash függvény előnyeiből adódóan, mind a csomópontok, mind a tárolt információk nagy valószínűséggel egyenletesen oszlanak el a címtérben, ami biztosítja, hogy a lehető legegyenletesebben oszoljon el a terhelés a rendszerben.

A hálózaton belüli üzenetek útvonal-irányítása érdekében minden résztvevő csomópont ismeri az őt követő szomszédját, valamint emellett karban tart egy finger table-nek nevezett táblázatot, amely néhány, a címtér távolabbi részén elhelyezkedő tagról is tartalmaz információkat. Erre azért van szükség, hogy a hálózat növekedésével se nőjön meg jelentős mértékben az üzenetek kézbesítéséhez szükséges idő. Mindazonáltal a finger table sem növekszik jelentősen, mérete csupán O (log N) nagyságrendű, ahol N a hálózat tagjainak száma.

3.2.2 CAN

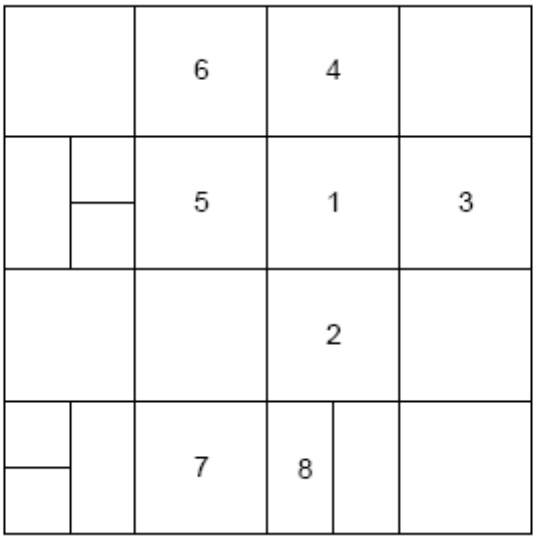
Egy másik DHT implementáció a CAN. A CAN egy teljesen elosztott, Internet szinten is skálázható megoldás, mely különböző szolgáltatásokat (beillesztés, keresés, törlés) képes nyújtani. Az előzőekben megismert Chord-dal ellentétben ez a megoldás nem egy gyűrűt használ a kulcstér értelmezésére, hanem egy D-dimenziós ortogonális koordináta rendszerben kialakított D tóruszt, azaz egy D-dimenziós ciklikus teret (ez a D érték a rendszer paramétere, annak beállításánál megadható). Minden csomópont ennek egy-egy szegmenséért felel, az ebben lévő állományokat tárolja, és az ezekre irányuló kérésekre válaszol. Ezen Ortogonális tér „szeleteket" zónának nevezzük.

A belépés során az új tag választ egy véletlenszerű pontot a D-dimenziós térben, majd kapcsolatba lép az adott pontért felelős csomóponttal. Miután ez sikeresen megtörtént, a szükséges információ- és adatcserét követően a kérdéses koordináta zónát megfelezik és a továbbiakban az új tag azért a részért lesz felelős, amelybe a korábban kiválasztott pont esik.

Az útvonal-irányítás sikerének érdekében minden résztvevő tárolja a közvetlen szomszédjához tartozó IP címet, valamint az általa kezelt zóna leíróit. A CAN esetében két csomópontot akkor nevezhetünk szomszédosnak, ha zónáik egymást D-1 dimenzió mentén átfedik és 1 dimenzió mentén szomszédosak. Például a 8. ábrán látható esetben az 1-es számmal jelölt csomópontnak szomszédja a 2-es, 3-as, 4-es, 5-ös zónák, ugyanakkor a 6-os, és 7-es nem. A tagok által tárolt információkat felhasználva, a rendszer a forrás, valamit a célcím ismeretében képes egyenes vonalban továbbítani a rendszer üzeneteit. Ebből a tulajdonságból fakadóan egy-egy keresés csupán $O\left(D * n^{\left(\frac{1}{D}\right)}\right)$ lépést követően eljuttatható az arra válaszolni hivatott csomóponthoz.

A 8. ábrán látható esetben, amennyiben az 1-essel jelölt zónáért felelős csomópont üzenetet küld egy, a 7-es zónába eső koordinátára, úgy az a következőképpen jutna el a címzettjéhez. Először a feladó megvizsgálná, hogy a szomszédjai közül valamelyik zónája tartalmazza-e a keresett címet. Pozitív eredmény esetén eljuttatná hozzá azt, ellenkező esetben megvizsgálná, hogy melyikük esik közelebb a ciklikus térben a célcímhez, és átadná neki azt továbbítás céljából. A következő csomópont szintén az előbb említett vizsgálatokat elvégezve

vagy közvetlenül kézbesítené a címzettnek, vagy egy, a céltól kisebb távolságra lévő szomszédjának adná át. Ennek megfelelően, példánkban egy ilyen üzenet az 1-es csomópontot követően a 2-es, majd a 8-as zóna érintésével jutna el a címzetthez.



8. ábra: CAN zóna-felosztás kétdimenziós tér esetén

Az útvonal-irányítás terén léteznek más, az optimálisabb üzenettovábbítást célzó megoldások is. Ezek törekedhetnek egy egyenletesebb zóna felosztásra, a fizikai topológia szerint kisebb távolságra lévő csomópontok CAN-en belül is „közelebb” való elhelyezésére, vagy a továbbítási útvonalak lerövidítésére. Ez utóbbi célt szolgálja a különböző CAN valóságok létrehozása és menedzselése. Ennek értelmében a csomópontok a teret K különböző változatban osztják fel egymás között, ahol az egyes változatokat valóságoknak nevezzük. Ezt követően minden egyes útvonal-irányítási döntés során megvizsgálják valamennyit, és ez alapján választják ki azt a valóságot, amelyben az üzenetet továbbítani fogják, mégpedig azt, amelyikben a cél elérése érdekében a legkevesebb további csomópontot szükséges érinteni.

Az üzenettovábbítás hatékonysága függ a koordináták számától, azaz a tér dimenziójától is, a már említett $O\left(D * n^{\left(\frac{1}{D}\right)}\right)$ módon. Ebből látható, hogy az útvonalak hossza a koordináták számának növelésével lerövidíthető.

A rendszer a korábbiak mellett képes a meghibásodott, illetve kilépő csomópontok kezelésére is. Ebben az esetben az általuk felügyelt zónához tartozó valamennyi feladatot az egyik, a távozást észlelő szomszéd veszi át. Amennyiben lehetséges, úgy a két zónát egyesíti, egyéb esetben a kérdéses számítógép a továbbiakban egyszerre két CAN node-ot futtat párhuzamosan.

3.2.3 Pastry

Az utolsóként említésre kerülő rendszer a Pastry névre hallgat. Az előzőekhez hasonlóan a rendszer bemutatását most is a kulcstér, illetve annak felosztásának ismertetésével kezdem. Ebben a hálózatban mind a csomópontok, mind az elhelyezett tartalmak egy-egy 128 bites azonosítót kapnak. A tagok belépésükkor egy egyenletes véletlenszám-generátor segítségével határozzák meg azonosítójukat, melyet később a hálózatban is hirdetni fognak. Ennek a megoldásnak a legnagyobb előnye az, hogy az így kiosztott értékek nagy valószínűséggel egyenletesen oszlanak el a térben, ugyanakkor sem ez nem garantálható, sem az, hogy két csomópont nem kaphat véletlenül teljesen megegyező azonosítót. Elsősorban az utóbbi lehetőségre való tekintettel, a rendszer kidolgozói az előbbi megoldás alternatívájaként említik a csomópontok egy nyilvános (mindenki által megismerhető) kulcsa, vagy IP címe, hash függvény segítségével képzett lenyomatának használatát.

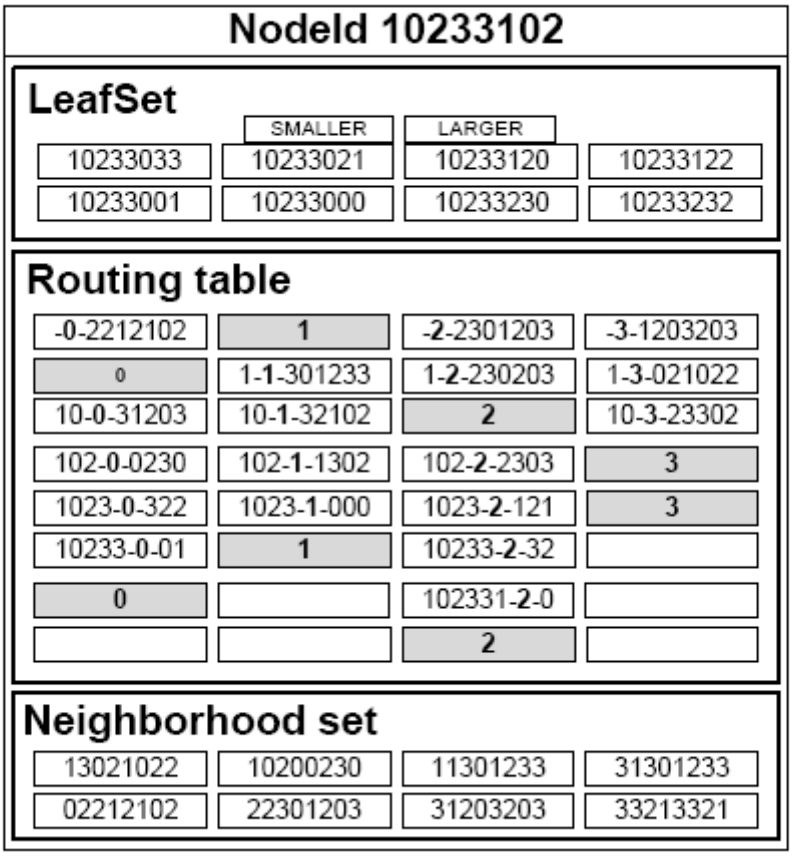
Értelmezett továbbá egy távolságfüggvény is, melyben két azonosító távolságának a számok különbségének abszolút értéke felel meg. Az egyes állományok csomóponthoz rendelése is az említett metrika szerint történik. Minden elhelyezett adatért a hálózat azon résztvevője lesz a felelős, amely azonosítója a legközelebb áll az állomány tartalmából, nevéből, illetve tulajdonosának nevéből kriptográfiai hash függvény segítségével képzett értékhez. Az, hogy mit veszünk figyelembe az azonosító meghatározásakor, kizárólag azon

felsőbb rétegbeli alkalmazás függvénye, amely a Pastry által nyújtott szolgáltatásokat igénybe veszi.

A hálózat kidolgozói más, a Pastry által nyújtott szolgáltatásokra épülő rendszereket is kidolgoztak. Ilyen a PAST, amely állományok tartós elhelyezését teszi lehetővé, illetve egy alkalmazásszintű multicast-ot lehetővé tevő, SCRIBE nevű alkalmazás is. Az előbbi esetben például az elhelyezett fájlok azonosítóinak meghatározása azok nevei és tulajdonosai, utóbbi esetben kizárólag az egyes csoportok témái alapján történik.

A hálózatot alkotó csomópontok elsődleges feladata a belépő, kilépő, eltűnő tagok kezelésén túl, hogy a rendszer működéséhez elengedhetetlen, illetve annak valódi hasznát nyújtó üzeneteket továbbítsák. Ahhoz, hogy egy-egy üzenetet eljuttassanak annak felelőséhez, minden csomópont egy lokális értelemben optimálisnak tekinthető döntést hoz. Megvizsgálja az üzenet címzettjét, majd az általa ismert csomópontok közül eljuttatja ahhoz, amely azonosítója a sajátjánál hosszabb perfixében egyezik meg a keresett értékkel. Ha ilyet nem talál, akkor egy olyat keres, amelynek a kulccsal közös perfixe ugyanolyan hosszú, de a rendszer metrikája alapján, numerikusan közelebb áll a célhoz. Amennyiben nem talált sem az első, sem a második feltételnek megfelelő csomópontot, úgy biztos lehet benne, hogy ő a felelős a kért információért és nincs más teendője, minthogy válaszoljon a kapott kérésre.

Ahhoz, hogy az egyes lokális döntések a rendszer egésze szempontjából azonos, optimális választáshoz vezessenek, az egyik legfontosabb kérdés, hogy a csomópontok milyen, az ismert szomszédokat tartalmazó adatbázisokat tároljanak és tartsanak karban. Minden egyes résztvevő rendelkezik egy útvonal-irányítási táblával (routing table), egy, a szomszédokat tartalmazó gyűjteménnyel (neighborhood set), valamint egy közeli azonosítókkal rendelkező csomópontgyűjteménnyel (leaf set). Ahhoz, hogy megvizsgáljuk, milyen információk alapján tud egy-egy csomópont megfelelő döntést hozni, mely végül valóban a kérés felelőséhez, illetve a címzetthez juttatja el az egyes üzeneteket, legjobb, ha megvizsgálunk egy példát.



9. ábra: Példa a Pastry csomópontok által tárolt információra

A 9. ábrán látható, hogy egy olyan hálózatban, ahol az azonosítók 8 számjegyből állhatnak, egy számjegy pedig 2^2 különböző értéket vehet fel, hogyan is épülhetnek fel az 10233102 azonosítóval rendelkező csomópontnál tárolt információkat tartalmazó adatbázisok. (Az ábrán nincsenek feltüntetve az azonosítókhoz tartozó IP címek.)

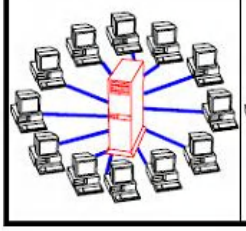
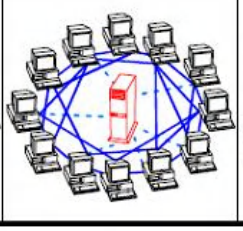
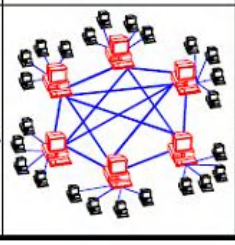
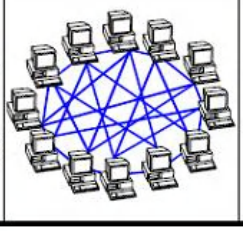
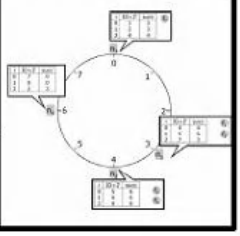
Az egyes csomópontok által feladott, vagy kézbesítésre átvett üzeneteket először az útvonal-irányítási tábla segítségével kísérli meg továbbítani. Ennek felépítése egészen egyszerű. A tábla minden egyes sora tartalmaz egy-egy olyan csomópont-azonosítót, amelynek biztosan legalább egyel, hosszabb perfixe egyezik meg a keresett címmel. Vagyis az ábrán látható esetben, ha a címzett azonosítója 0-ással, 2-essel, vagy 3-assal kezdődne, úgy biztosan ismerünk egy-egy olyan csomópontot, amelyre ez szintén teljesül. Ha 1-essel kezdődik, úgy mi már csak olyan csomópontot kereshetünk, amelynek címe legalább két számjegyben megegyezik a keresettel. Akkor megvizsgáljuk a táblázat második sorát, ahol

kizárólag ilyen azonosítók szerepelnek. Ha még itt is megegyezik a saját azonosító perfixe a címzettel, úgy ezt folytatjuk a következő soron és így tovább. Ezzel a megoldással garantálható, hogy az üzenet előbb-utóbb eljut a címzetthez. Az útvonal-irányítás hatékonyságának fokozására szolgál a szomszédokat, valamint a legközelebbi azonosítókat tartalmazó két gyűjtemény. A leaf set névre keresztelt L halmaz tartalmazza a csomópont azonosítójánál nagyobb azonosítók közül a legközelebbi $\frac{|L|}{2}$ darab nagyobb és az ugyanennyi kisebb értéket. Ennek felhasználásával egy, a keresett címhez közeli azonosítóval rendelkező csomópont már biztosan, lokálisan el tudja dönteni, hogy L mely eleme lehet a felelős a kérdéses információért, és egyenesen hozzá tudja továbbítani. Ugyanezen halmaz felhasználásával tudja megállapítani azt is, hogy az adott üzenetnek ő lesz-e a címzettje, vagy pedig valamelyik közvetlen szomszédja. Az algoritmust szem előtt tartva láthatjuk, hogy az útvonal-irányítás, bármely két csomópont között legfeljebb $\left|\log_{2^b} N\right|$ lépésből állhat, míg a táblák csupán (körülbelül) $\left|\log_{2^b} N\right|$ * (2^b – 1) bejegyzést tartalmaznak, ahol N a hálózatban résztvevő csomópontok számát jelöli, b pedig a rendszer egy paramétere, amelyben az egyes azonosítók 2^b különböző számjegyből állhatnak. A szomszédokat tartalmazó halmaz segítségével megkísérelhető az útvonalak egy másik mértékkel vett, elsősorban földrajzi értelemben vett optimalizálása is. Ennek érdekében a döntés meghozatalakor a csomópontok ezt az információt is figyelembe veszik.

A rendszer teljesen önszerveződő, valamint képes a dinamikusan változó környezethez is alkalmazkodni. Ennek megfelelően képes kezelni az új csomópontok belépését, a meglévők távozását, illetve figyelmeztetés nélkül való kiesését is. Ezen menedzsment funkciók ellátásához szükséges üzenetek száma, valamint a keletkezett overhead, összhangban a már ismertetett funkciókkal képes megőrizni a hálózat skálázhatóságát és robosztusságát. Figyelembe véve azt is, hogy a rendszer teljesen elosztottan működik, elmondható, hogy kielégít minden, a DHT hálózatokkal szemben támasztott kívánalmat illetve követelményt.

4. Fájlmegosztás megvalósított Peer-to-Peer rendszerekben

Azokat a P2P architektúrákat veszem górcső alá, amelyek célja alapvetően a fájl megosztás. Az ilyen rendszerek a legelterjedtebbek, és többféle megoldás alakult ki a felhasználók közötti dokumentum megosztására.

Client-Server	**Peer-to-Peer**			
1. Server is the central entity and only provider of service and content. → Network managed by the Server 2. Server as the higher performance system. 3. Clients as the lower performance system Example: WWW	1. Resources are shared between the peers 2. Resources can be accessed directly from other peers 3. Peer is provider and requestor (Servent concept)			
	Unstructured P2P			**Structured P2P**
	Centralized P2P	*Hybrid P2P*	*Pure P2P*	*DHT-Based*
	1. All features of Peer-to-Peer included 2. Central entity is necessary to provide the service 3. Central entity is some kind of index/group database Example: Napster	1. All features of Peer-to-Peer included 2. Any terminal entity can be removed without loss of functionality 3. → dynamic central entities Example: Gnutella 0.6, JXTA	1. All features of Peer-to-Peer included 2. Any terminal entity can be removed without loss of functionality 3. → No central entities Examples: Gnutella 0.4, Freenet	1. All features of Peer-to-Peer included 2. Any terminal entity can be removed without loss of functionality 3. → No central entities 4. Connections in the overlay are "fixed" Examples: Chord, CAN
				

10. ábra: P2P rendszerek csoportosítása

4.1 Centralizált, hibrid

Egy hibrid P2P rendszer valójában nem teljesen felel meg a P2P rendszer definíciójának. Bár minden csomópont egyenrangú, a felhasználók dokumentumainak attribútumai (fájlnév, milyen IP-vel rendelkező gépen található, minőség, szerző, cím, stb.) egy központi szerveren vagy szerver csoporton vannak tárolva. Az egyik első P2P rendszer, a

Napster is ilyen architektúrájú, az ilyen rendszerek a P2P rendszerek legelső generációjához tartozik.

Ezen architektúra előnyének mondható, hogy a szerver kiszolgáló-képességének határáig gyors, a keresés az összes dokumentumban történik, viszonylag kis hálózati forgalmat igényel. E sok előnyből azt gondolhatnánk, hogy mindenki ezt az architektúrát választja. De egy ilyen, félig centralizált rendszer hátránya, hogy a dokumentumok adatait tároló szerver miatt rendelkezik a centralizált rendszerek minden hátrányával, vagyis az egész rendszer működése megszűnik a szerver hibája, vagy megszűntetése miatt (mint például a Napster esetében, jogi okok miatt).

4.2 Teljesen elosztott, homogén

A P2P rendszerek következő generációja a teljesen elosztott, homogén, „igazi peer-to-peer" rendszer. Leggyakoribb példája a Gnutella hálózat. Az ilyen rendszerben valóban minden résztvevő (csomópont) egyforma: a csomópontok teljesen azonos szerepűek, ugyanazt a feladatot látják el. Egy ilyen rendszerben a dokumentumok adatai nem egy központi szerveren találhatók, hanem ténylegesen egy adott résztvevő tudja csak, hogy milyen attribútumokkal rendelkeznek az általa tárolt dokumentumok.

A keresés tehát megnehezedik: egy adott fájl keresése a hálózatban már nem csupán egy központi szerverhez való adatküldésre korlátozódik. Az egyik csomópont megkap egy kérést (keresést), megvizsgálja, hogy nála megtalálható-e az adott dokumentum, majd ha nem, tovább küldi az általa ismert résztvevőknek, akik szintén ezeket a műveleteket hajtják végre a kérésen.

Az előnyök és hátrányok már láthatók: ebben a rendszerben egy résztvevő hibából, vagy más okokból történő leállása, leállítása nem okoz problémát. Azonban a keresés lassabb lesz, és ami a legfontosabb, nagy hálózati forgalmat generál egy keresés. Ha egy adott kérés bizonyos mélységű továbbküldözgetésével szeretnénk elérni egy adott dokumentum megtalálásának valószínűségét, akkor figyelembe kell vennünk, hogy az általunk használt résztvevő megengedheti-e magának azt a hálózati forgalmat, ami ehhez szükséges.

Sok rendszer született, ami ezt az architektúrát választja, és még több, ami alapvetően nem ezen az architektúrán alapul, de támogatja az ilyen protokollt. A legtipikusabb példák: LimeWire, Shareaza, Morpheus.

4.3 Félig elosztott, hierarchikus

Manapság a gyakorlatban (fájlmegosztásra) használt rendszerek jelentős része hierarchikus szerkezetű. Ilyen rendszereknél bár nem minden csomópont azonos, de azonos viselkedési típusok, szerepek közül választhatnak. Egy nagyobb hálózati forgalmat, vagy nagyobb számítási kapacitást kiszolgálni képes csomópont kiválaszthatja a neki megfelelő szerepet, amivel a többlet erőforrásai kihasználhatóvá válnak, míg a kisebb teljesítményű résztvevők sem kényszerülnek olyan feladatok ellátására, amire nem képesek.

Az ilyen rendszerek jelentős része mindössze kétszintű: a kisebb képességű résztvevők közvetlenül nem vesznek részt a keresésben, hanem dokumentumaik információit odaadják egy nagyobb képességű résztvevőnek, akivel közvetlen kapcsolatban állnak. Egy adott attribútumokkal rendelkező dokumentum iránti kérést szintén ehhez a nagyobb képességű csomóponthoz intézi. A nagyobb képességű résztvevő (super-node) a többi társához továbbítja a kérést, akik mind információval rendelkeznek a hozzájuk kapcsolódó kisebb képességű résztvevők dokumentumairól is.

Ez az architektúra tehát az előző kéttípusú rendszere előnyeit próbálja ötvözni. Talán ennek köszönhető, hogy a legtöbb mai rendszer kétszintű: a talán legtöbbek által ismert Kazaa, és Grokster társa, illetve korábban a Morpheus rendszer is (mindannyian a FastTrack nevű rendszert használják), valamint a szintén közkedvelt WinMX is, de a Gnutella 0.6 vagy 2 protokollt használó számos rendszer (Shareaza, LimeWire) is ebbe a kategóriába tartozik.

4.4 Elosztott indexelés

A legelső, centralizált változat azért volt gyorsabb (és kisebb erőforrás-igényű) a második, decentralizált változatnál, mert nem kellett az összes résztvevőt végigkérdezni, rendelkezik-e az adott dokumentummal, hanem egy központi szerver tudott erre a kérdésre válaszolni. A központi szerver ilyen szerepét azonban rábízhatjuk a többi résztvevőre is. Egy

megfelelően szervezett struktúra majdnem hasonlóan gyors lehet, míg hibákra és egyéb tényezőkre érzéketlen marad az elosztott tárolás következményeként.

Az indexek legtöbbször bináris formába alakítás után egy fa, vagy egy gyűrű struktúrában tárolhatók, ahol az index utáni keresés logaritmikus mértékű a résztvevők számához képest, ami hatékonynak mondható. Az ilyen rendszerek azonban többnyire egyszerre csak egy kritérium (például fájlév) indexelését támogatják, ami nem elég flexibilis.

Ezen a problémán például az előző architektúrákkal való kombinálással lehet segíteni (például Gridella). Szintén ezen a problémán próbálnak megoldást javasolni olyan struktúrák, ahol több, hierarchikus felépítésű index-szerkezet létezik (például „Content-based aggregation network"). Szintén jellemző ezen kívül a keresések gyorsítása olyan módon, hogy a P2P struktúra figyelembe veszi az egyes résztvevők földrajzi elhelyezkedését, vagy a hálózati késleltetést (például Grapes).

Ezeknek a rendszereknek tehát rendkívüli előnyük a teljes elosztott működés melletti gyors, kis hálózati forgalmat generáló keresés a dokumentumok között, míg hátrányul sorolhatjuk fel a csökkent mértékű flexibilitáson kívül a meglehetősen bonyolult felépítést is. Egy index fa, vagy gyűrű, esetleg egyéb, elhelyezkedésen alapuló struktúra felépítése és karbantartása legtöbbször nem egyszerű algoritmusokat igényel. Külön probléma lehet – a résztvevők folyamatos változása (kilépés, belépés) miatt – a rendszer nem stabil állapotban lévő teljesítményére vonatkozó állítások megfogalmazása is.

Az ilyen rendszerek sok előnye, ám sok problémája miatt többféle megoldás is született, ami elosztott indexelést alkalmaz. Példaként sorolható fel: Gridella (P-Grid fa), Chord, CAN, Pastry, Tapestry, Freenet, Grapes, stb.

5. P2P alkalmazási területei és hatékonyságuk

A fájlmegosztáson kívül a P2P rendszereknek sok más alkalmazási területe is van, amire esetleg nem is gondolnak az egyszerű felhasználók. Mint a következőkben látni fogjuk, használják a kép-, és hangátvitel során, tudományos kutatások esetén, de még akár az oktatási intézményekben is.

5.1 Skype

A világhálón barangolók szinte biztos, hogy találkoztak már olyan programok nevével, mint a Live Messenger, a Skype stb. A legtöbben ezeket a programokat még mindig a szöveges kommunikációra használják. Pedig nagyon sok egyéb lehetőség is benne rejlik a programban. A Skype használható Internen keresztüli telefonálásra, videótelefonálásra, sőt konferenciabeszélgetések tartására is.

A Skype egyike az Internetet, mint telefonhálózatot használó kliens programoknak. A telefonbeszélgetéseknél a beszélgetéseket feldarabolva, az egyéb adatokhoz hasonlóan csomagonként szállítja a világháló. Az Internet protokolljainak segítségével a beszéden kívül olyan információkat is továbbítanak a hálózaton, amelyek az egyes csomagoknak a célba juttatásához szükségesek. Az Internet jelenleg leginkább elterjedt protokollját, az IP-t telefonálásra használó rendszert Voice over IP-nek, röviden a VoIP-nak hívják.

Ez tehát azt is jelenti, hogy a VoIP rendszerű telefonálás kiépítése minden további átalakítás nélkül lehetséges egy-egy az Internetre csatlakozó számítógépen, vagy számítógépes hálózaton. A VoIP technológia használatával végzett telefonálásnál a résztvevők ugyanazt az eljárást követik, mint a hagyományos telefonálás esetében. A hagyományos telefonbeszélgetésekhez hasonlóan itt is három része van a kommunikációnak, először fel kell építeni az összeköttetést, amit a beszélgetés közvetítése, átvitele követ, és végül jön az összeköttetés leépítése. A két rendszer között különbség, mint már említettem, abban áll, hogy míg a hagyományos telefonkapcsolatnál a beszéd átvitele folyamatos, itt a beszélgetés adatait kis csomagok szállítják a kommunikációban résztvevő partnerek között. A

hagyományos telefonrendszerek egy beszélgetéshez egy teljes vonalat foglalnak le, a VoIP rendszerek működésekor a telefonrendszerek egy vonalon keresztül több beszélgetést is lebonyolítanak. Ezen a különálló tevékenységterületen alapulva különböző protokollok végzik a kapcsolat fel és leépítését, a csomagok továbbítását.

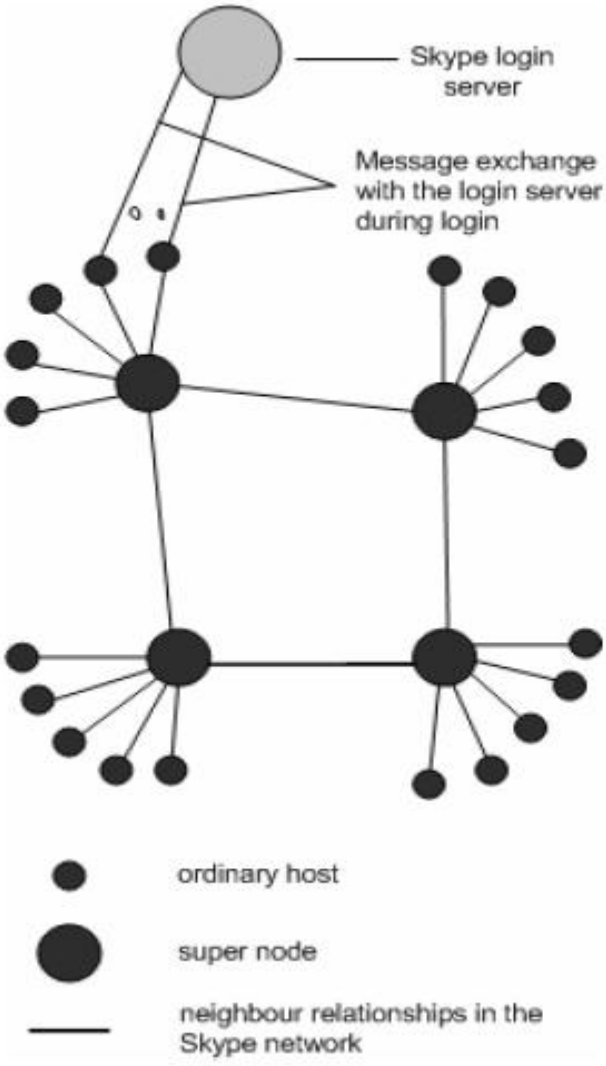
Egy IP alapú hálózatban ahhoz, hogy a beszélgető partnerek közötti kapcsolatot be lehessen állítani, a hívott fél IP címének ismertnek kell lennie a hálózaton belül, de nem feltétlenül a hívó fél részéről. Földrajzilag stabil csatlakozások (a felhasználóhoz rendelt statikus IP címek) igen ritkán vannak az IP alapú címzési rendszeren nyugvó hálózatokban. A telefonszámok és az IP címek állandó összekapcsolását olyan dolgok is lehetetlenné teszik, mint a kommunikáció résztvevőinek helyváltoztatása, a felhasználók váltakozása ugyanannál a számítógépnél. Különböző megoldások segítségével biztosítják a résztvevők számára az IP címektől független telefonszámokat. A Skype azon túl, hogy a legtöbben egy, a számítógépen futó azonnali üzenetküldő és telefon kliens programként ismerik, egyben egy ilyen Interneten keresztüli telefonálást is lehetővé tévő rendszer.

5.1.1 Skype protokoll

Skype egy saját Internet telefonálásra (VoIP) alkalmas hálózatot használ. A Skype által készített protokoll nyilvánosan nem érhető el és a hivatalos alkalmazások, amik használják a protokollt, nem nyílt forrásúak. A Skype és VoIP ügyfelek közti fő különbség az, hogy a Skype a hagyományos szerver-kliens architektúra helyett, a P2P modellt használja. A felhasználói jegyzéket teljesen decentralizálják és szétosztják a csomópontok között a hálózatban, ami azt jelenti, hogy a rendszer nagyon könnyen óriási méretű lehet, (jelenleg több mint 300 millió regisztrált felhasználó) összetett és költséges központosított infrastruktúra nélkül. A Skype hálózat nem vagy nehezen képes együttműködésre másik VoIP hálózattal.

A Skype olyan egyenrangú hálózat, amely három fő entitással bír: supernode-ok, szokásos csomópontok és a bejelentkezés szerver. Ez egy overlay hálózat: minden egyes ügyfél felépíti és frissíti az elérhető csomópontok listáját, ami host cache-ként ismert. A host cache tartalmazza a supernode-ok IP-címeit és portszámait. A kommunikációt titkosítására az RC4 eljárást alkalmazza, mely módszer használta nem biztosít titkosságot, helyette csupán

elhomályosítja a forgalmat. Supernode-k kommunikációt továbbítanak más ügyfeleknek tűzfal mögött. Bármelyik Skype ügyfél supernode-dá tud válni, amennyiben jó sávszélessége, nincsen tűzfala és megfelelő feldolgozási kapacitással rendelkezik. Supernode-k slotokba csoportosulnak, 9-10 supernode alkot egy slotot. A slotok blokkokba csoportosulnak, 8 slotból áll egy blokk. A Skype a többi peeren keresztül irányítja a hívásokat a hálózaton, hogy megkönnyítse a csatlakozást a szimmetrikus NAT és tűzfalak által elfedett felhasználókhoz. Ez mindazonáltal egy plusz terhet tesz azokra, akik NAT nélkül csatlakoznak az Internethez, hiszen a számítógépeiket és hálózati sávszélességüket arra használhatják, hogy irányítsák a más felhasználók hívásait. A Skype zárt forráskódú, és a protokollt nem szabványosították.



11.ábra: A Skype hálózat

Pozitívumok:

- Széleskörű kommunikációs lehetőségek, az azonnali üzenetküldésen keresztül, a beszélgetéseken át a videókonferenciáig.
- Költségtakarékos, hiszen nincsen szükség óriási beruházásra, hogy teljesen új hálózatot építsenek ki.
- A Skype rendszere az Interneten keresztül is képes jó minőségű hangátvitel biztosítása.
- Könnyen átlátható és értelmezhető felület a fizetős, extra szolgáltatások esetében is.

Negatívumok:

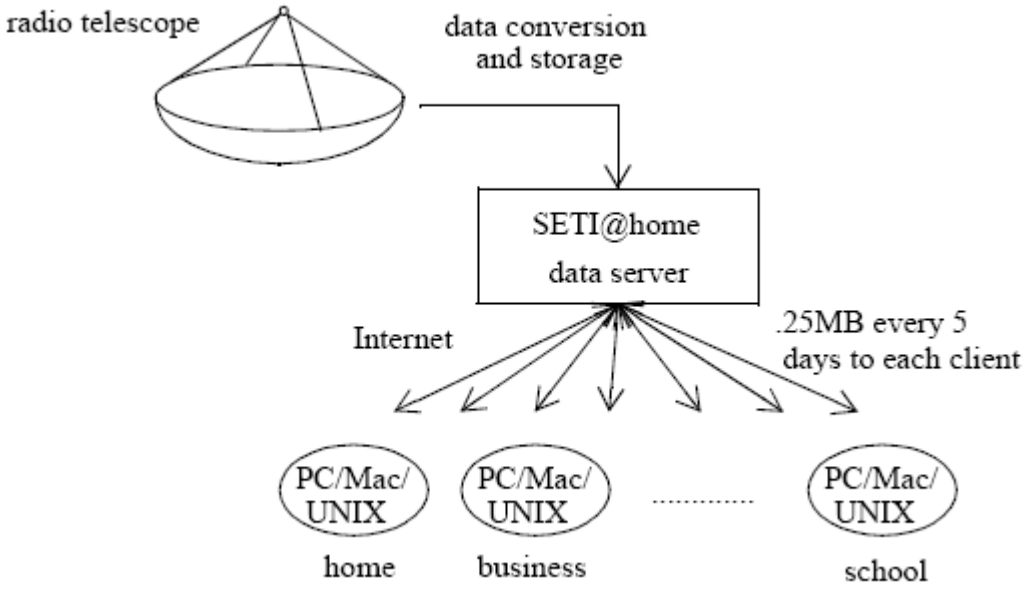
- A Skype használatával kapcsolatban a legsúlyosabb problémákat a telefon és üzenetküldő rendszer saját protokollrendszere okozza. Ez a protokollrendszer nem tud együttműködni, vagy csak körülményes és hibákkal teli együttműködésre képes olyan, az Internetes telefonálásban megszokott egyéb rendszerekkel, mint az Internetes beszélgetésben résztvevők azonosítását lehetővé tévő SIP, vagy a kommunikációt biztosító H.323 protokoll.
- A Skype használatának további hátránya, hogy a szoftver által használt kommunikációs protokollokon keresztül a kommunikációban résztvevő számítógépek támadhatók. Egy ilyen támadás kockázata a magán felhasználásban csekély, de céges használat esetén komoly biztonsági gondokat jelenthet.
- Ha a fő célra, azaz telefonálásra akarja valaki használni a programot, ahhoz ISDN gyorsaságú adatátvitel szükséges.

5.2 SETI @ home

A SETI @ home projekt talán az egyik legjobb példája annak, hogy a P2P nem csak fájlmegosztásra használható, hanem elosztott számításokra is. SETI (Search Extraterrestrial Intelligence) a kutatóprojektnek az a célja, hogy idegen civilizációkat fedezzen fel a Földön kívül. A hatalmas mexikói Arecibo távcső által az űrből fogott rádiójeleket veszi és továbbítja darabokban az Internetre kötött kihasználatlan PC-k millióinak.

A rendszer alapja, hogy az Internetre kapcsolt számítógépek nagy részének számítási kapacitása többnyire kihasználatlan. Tudományos kísérletek esetén – főképp az űrből fogott

rádiójelek értelmezése esetén - óriási számítási kapacitásra van szükség. Ilyen mértékű teljesítménnyel csak szuperszámítógépek rendelkeznek, amik építése hatalmas költséggel jár. Viszont ha az egyszerre kapott adatmennyiséget lebontjuk kis darabokra és elküldjük a hálózatba kötött sok-sok kihasználatlan számítógépeknek, akkor hatalmas teljesítményt érhetünk el. Ez az alapja a SETI projektnek is. A P2P hálózatot használja az intenzív számítási igény terjesztésére.



12. ábra: A SETI@home architektúra

A SETI projekt már eléggé régi ahhoz, hogy nagyon magas minőségi szintet elérjenek a szolgáltatás terén. Két főbb összetevője van: az adatbázis-szerver és az ügyfelek. Ahogy látszik bármilyen platformon használható a kliens. A sok év alatt kiderült, hogy az adatbázis jól méretezhető és biztonságos, hiszen több millió lelkes felhasználója van a projektnek.

Érdekes, a tartalomipartól a gyökerekhez visszavezető fejlesztés a BOINC, amely a University of California Berkeley falain belül ad lehetőséget klasszikus, a P2P hagyományos céljait célul tűző projekteknek, amelyeknek megosztott számítási kapacitásra van szüksége egyfajta szuperszámítógép gyanánt. A BOINC gyakorlatilag egy szoftver platform, amely a GRID computing-ot teszi elérhetővé bárki számára. Néhány éve például az űrkutatásra szakosodott SETI is a BOINC részévé vált, de bárki definiálhat saját projektet a BOINC-on

belül, amely független a többitől és saját adatbázist, valamint szervereket is használhat. A becsatlakozó önkéntesek a szabad forráskódú szoftver installálását követően maguk dönthetik el, hogy számítógépük holt idejében mely projekteket, és milyen arányban támogatják számítási kapacitásuk átadásával.

5.3 P2P az oktatásban

A Bittorrent állományterjesztési technológia számlájára új sikert jegyezhetünk fel: a holland INHOLLAND Egyetem tíz helyszínen 6500 számítógépet tart karban, frissít folyamatosan, nem ritkán 3,5 gigabájtnyi letöltendő adatot eljuttatva a gépekre. Könnyű belátni, ennyi számítógépet ellátni komoly infrastruktúrát, komplett szerverparkot igényel.

A holland egyetem eddig kéttucatnyi szervert használt erre a célra, melyek szépen lassan minden számítógéphez eljuttatták az adatokat, jelentős erőforrást és nem utolsósorban időt felemésztve. A klasszikus szerver-kliens felállásnál azonban létezik egy sokkal hatékonyabb módszer, a közismert állománycserélő technológia, a Bittorrent.

A Bittorrent peer-to-peer (P2P) módszere lényegében minden kliensből disztribúciós szervert generál: maguk az egyes kliensek is az elosztó rendszer részei lesznek, az egyes gépek feltöltési sávszélességét is igénybe veszik az anyagok további számítógépekre juttatásához. Szerverekre szinte nincs is szükség, csupán az első néhány teljes letöltés erejéig használják őket: az egyetem kéttucatnyiból 20 szervert le is kapcsolhatott.

Az új, Bittorrentet használó frissítő rendszer kevesebb, mint 4 óra alatt 22,2 terabájtnyi adatot szórt szét a 6500 számítógép között, sokkal gyorsabban és kevesebb sávszélességet felemésztve, mint korábban: ehhez az eredményhez ezt megelőzően 4 teljes napra volt szükség. Először az egyetem vezetősége elutasította a P2P technológia alkalmazását, egy demót követően azonban gyorsan változtatott álláspontján.

Hazánkban - legjobb tudomásom szerint - ezt a módszert nem használják. Itthon a jól bevált módszert alkalmazva, többnyire egy előre elkészített lemezképet használnak, amivel nem csak az operációs rendszer legújabb frissítései, hanem az alkalmazások is frissítésre kerülnek.

6. Összefoglalás és kitekintés

Dolgozatomban igyekeztem minél jobban összefoglalni a P2P rendszerek alkalmazási lehetőségeit és ezek működési elvét, természetesen a lehetőségek adta mértékig. Ahogy egyértelműen látható az online világ számára rendkívüli potenciál rejlik a technológiában, de nyilvánvaló az is, hogy újabb és újabb kérdéseket is vet fel, főképp jogi téren, valamint sok esetben biztonságtechnikai téren is. Ezen kérdések megoldása nagyon fontos feladata a szakmának.

Ismertettem a kliens-szerver architektúra jellemzőit, majd betekintést adtam a P2P rendszerek működési elvébe. Részletesen megnéztem az egyes alkalmazási technikákat a fejlődési szakaszoknak megfelelően. Így a korai DirectConnecttől és a Napstertől elindulva a Gnutellán keresztül eljutottunk a manapság rendkívüli népszerűségnek örvendő BitTorrentig illetve az elosztott hash táblán alapuló rendszerekig. Csak a legjelentősebb rendszereket vizsgálva.

Igyekeztem besorolni és rendszerezni az adott hálózatokat a különböző szempontok alapján. Megvizsgáltam az alkalmazásukat a fájlcseréléstől különböző területen, mint az oktatásban, kommunikációban vagy akár a tudományban, hiszen ma az emberek döntő többsége azonnal az illegális, warez tevékenységgel azonosítja a P2P rendszerek többségét.

Napjainkban a növekvő internetfelhasználók kiszolgálása, a növekvő adatforgalom egyre inkább új technológiák bevezetését feltételezi. A peer-to-peer rendszerek mindezeknek az elvárásoknak eleget tesznek, és fontosságuk ma már megkérdőjelezhetetlen. A mai számítástechnikában nagyon nehéz előre jósolni, viszont egyes kutatók szerint ez a technológia jelentheti a világháló új korszakát. Rendkívül érdekes téma a GRID hálózatok (elosztott számítási rendszerek, melyek a szabad feldolgozási és tárkapacitást használják fel szuperszámítógépek létrehozására, a SETI projekthez hasonlóan), ami dolgozatomnak nem képezi témáját. Jómagam is egyet értek azzal, hogy ez a technológia a jövő korszak meghatározója.

7. Szójegyzék

- **availability (elérhetőség):** A *torrent* teljes másolatainak száma a kliens számára. Minden *seed* 1-et ad ehhez a számhoz. Egy csatlakozott *peer*, amelynek csak töredékek állnak a rendelkezésére, csak egy tört számot ad az *elérhetőséghez.*

- **torrent:** Jelentheti a meta-fájlt az adatokkal, vagy jelentheti azt a fájlt is, amire hivatkozik, a környezettől függően.

- **peer (csomópont):** A *peer* egy másik számítógépen futó kliens, főképp azokat a klienseket értjük ezen, amelyek még nem az egész fájlt, csak részeit birtokolják.

- **seed (mag, megosztó):** A *seed* egy olyan *peer*, amely már rendelkezik az összes darabkával, de továbbra is megosztja a fájlt. Minél több *seed* van, annál nagyobb az esélye a sikeres letöltésnek.

- **leech (leszívó, letöltő):** A kifejezést azokra a *peer*ekre használjuk, amelyeknek kifejezetten rossz a feltöltés/letöltés arányuk, vagy elhagyják a *boly*t rögtön azután, miután befejezték a letöltést. Ez az általános BitTorrent etikettel ellenkezik. A téves értelmezés szerint a *leech* egy olyan *peer*, amely még nem rendelkezik minden fájldarabkával.

- **swarm (boly):** Együttesen az összes *peer*t, ami megosztja a .torrent fájlt nevezzük *boly*nak. Négy peer és két *seed* hattagú *boly*t jelent.

- **tracker (**nyomonkövető**):** A *tracker* egyfajta bróker feladatot lát el: közvetít a *peer*ek között. A trackeren általában nincs meg a fájl, a *tracker* nem vesz részt az adatcserében.

8. Irodalomjegyzék

[1.] World Internet Usage Statistics. *Internet World Stats.* Miniwatts Marketing Group http://www.internetworldstats.com/stats.htm.

[2.] **Andrew S. Tanenbaum**: Számítógép-hálózatok. Budapest: Panem, 2004.

[3.] Computer History Museum. *Computer History Museum - Exhibits - Internet History.* http://www.computerhistory.org/exhibits/internet_history/.

[4.] The Napster Controversy. *A Brief History of Napster.* http://iml.jou.ufl.edu/projects/Spring01/Burkhalter/Napster history.html

[5.] **Vida R., Simon Cs.**: Peer-to-Peer (P2P) hálózatok. BME. http://qosip.tmit.bme.hu/~vida/eloadas/.

[6.] Gnutella.com. http://www.gnutella.com/.

[7.] **Andy Oram**: Peer-to-Peer – Harnessing the Power of Disruptive Technologies. O'Reilly, 2001.

[8.] Edonkey2000, http://en.wikipedia.org/wiki/EDonkey2000

[9.] Emule, http://en.wikipedia.org/wiki/EMule

[10.] **Jörg Eberspächer, Rüdiger Schollmeier, Stefan Zöls, Gerald Kunzmann**: Structured P2P Network sin Mobile and Fix Environments, In *proceedings of HET-NETs '04 International Working Conference on Performance Modelling and Evaluation of Heterogeneous Networks,* July 2004 Ilkley, U.K

[11.] BitTorrent.com. *What is BitTorrent? - BitTorrent.* http://www.bittorrent.com/what-is-bittorrent.

[12.] **Dejan S. Milojicic, Vana Kalogeraki, Rajan Lukose, Kiran Nagaraja1, Jim Pruyne, Bruno Richard, Sami Rollins, Zhichen Xu**: Peer-to-Peer Computing**.** 2002.

[13.] *Chord: A scalable peer-to-peer lookup service for Internet applications.* **Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., Balakrishnan, H.** San Diego : ACM SIGCOMM, 2001.

[14.] *A scalable content-addressable network.* **Ratnasamy, S., Francis, P., Handley, M., Karp, R., and Shenker S.** San Diego : Proc. ACM SIGCOMM, 2001.

[15.] *Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems.* **Rowstron, A., Druschel, P.** Heidelberg, Germany : 18th IFIP/ACM International Conference on Distributed Systems Platforms, 2001.

[16.] **Salman A. Baset, Henning Schulzrinne**: *An Analysis of the Skype Peer-to-Peer Internet Telephony Protocoll***,** Columbia University New York, 2004.

[17.] **David P. Anderson, Jeff Cobb, Eric Korpela, Matt Lebofsky, Dan Werthimer:** SETI@home: an experiment in public-resource computing, 2002.

[18.] Frissítések terjesztése BitTorrent segítségével a hollandiai Inholland Egyetemen. http://www.inholland.nl/Content/News/Nieuws2008/200803/INHOLLAND+aan+de+BitTorrent.htm

[19.] BOINC, http://boinc.berkeley.edu

[20.] **Bodó Balázs, Halácsy Péter, Korsós Milán, Prekopcsák Zoltán, Szalai András:** P2P hálózatok vizsgálata, Kitchen Budapest Medialab 2007. http://milan.kitchenbudapest.hu/~milan/p2p/p2p_report_0926.pdf