

Szegedi Tudományegyetem
Természettudományi és Informatikai Kar
Számítógépes Optimalizálás Tanszék

**A BitTorrent alapú kreditrendszerek
fenntarthatóságáról**

TDK dolgozat

Készítette:
Najzer Helga
programtervező informatikus MSc
szakos hallgató

Témavezető:
Dr. Vinkó Tamás
egyetemi adjunktus

Tartalomjegyzék

Tartalmi összefoglaló	4
1. Alapfogalmak	5
1.1. Peer-to-peer hálózatok	5
1.1.1. A p2p hálózatok szerkezete	6
1.1.2. Alkalmazási területek	6
1.2. BitTorrent	6
1.2.1. Alapfogalmak	6
1.2.2. Ösztönző mechanizmus	7
1.2.3. A .torrent fájlokról	8
1.2.4. További részletek	8
1.2.5. BitTorrent kliensek	8
1.3. BitTorrent közösségek	9
1.3.1. Megosztási arány kikényszerítés	9
1.3.2. Kreditrendszer	9
1.3.3. A törekvés jutalmazása	10
1.3.4. Gyenge pontok	10
2. A rendszer modellje	11
2.1. A rendszer modellje	11
2.1.1. A Trackerről	11
2.1.2. Rendszermodell	12
2.1.3. Adatfolyam	12
2.1.4. Swarmok szintje	12
2.1.5. Tranzakciók	13
2.2. Gyakorlati megvalósítás	13
3. Szimulációk	17
3.1. A fájl méret és a rendszer összeomlásának kapcsolata	17
3.1.1. Homogén peerek, alapeset	18
3.1.2. Homogén peerek, korlátozott feltöltés	19
3.1.3. Heterogén peerek, alapeset	20
3.1.4. Heterogén peerek, korlátozott feltöltés	21
3.2. A rendszerek fenntarthatósága a kezdeti kreditmennyiség tükrében	22
3.2.1. Homogén peerek, alapeset	22
3.2.2. Homogén peerek, korlátozott feltöltés	25
3.2.3. Heterogén peerek, alapeset	26

3.2.4. Heterogén peerek, korlátozott feltöltés	26
4. Összefoglalás	29
Nyilatkozat	29
Köszönetnyilvánítás	30
Irodalomjegyzék	31

Tartalmi összefoglaló

A zárt BitTorrent közösségek olyan internetes tartalommegosztó rendszerek, amelyekben valamilyen extra mechanizmus gondoskodik arról, hogy a felhasználók az őket érdeklő fájlok letöltése után bent maradjanak feltöltőként. Ezt ugyanis a standard BitTorrent protokoll nem biztosítja. Ilyen mechanizmus lehet például a megosztási arány kényszerítés (feltöltött adatmennyiség és a letöltött adatmennyiség hányadosa valamilyen küszöbérték felett kell, hogy legyen), vagy, ami ezzel ekvivalens, kredit fizetési rendszer alkalmazása. Dolgozatunkban olyan BitTorrent közösségek vizsgálatával foglalkozunk, ahol feltesszük, hogy kredit rendszer működik. Ebben az esetben egy felhasználó csak abban az esetben tölthet le egy fájlt, ha ahhoz elegendő mennyiségű kredittel rendelkezik. Kreditet gyűjteni feltöltéssel lehet.

Korábbi cikkekben kimutatták, hogy hasonlóan a pénzügyi világválsághoz, a zárt BitTorrent közösségekben is megfigyelhető a kredit krízis. Szimulációk segítségével megmutatható, hogy egy rendszer három állapotot érhet el: fenntartható állapot, *crunch* (nincsenek letöltők), illetve *crash* (nincsenek feltöltők).

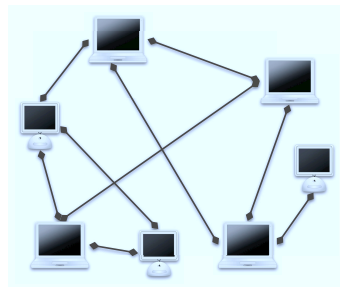
Dolgozatunkban a BitTorrent alapú kreditrendszerek fenntarthatóságához szükséges feltételeket határoztunk meg szimulációk segítségével. Tapasztalataink szerint a rendszer különösen érzékeny a felhasználók által birtokolt összes kredit mennyiségére, valamint az átlagos fájlméretre. Ezen két megfigyelés adta vizsgálataink alapját. Az első esetben elsősorban azt a kérdést feszegettük, hogy ha a rendszerben adott a gazdag felhasználók (azok, akiknek elegendő kredit áll rendelkezésre fájl(ok) letöltéséhez) aránya, akkor a fenntarthatóság mennyiben függ a kezdeti kredit mennyiségétől, illetve annak eloszlásától. Rámutatunk továbbá, hogy a gazdag és szegény felhasználók eloszlása akkor is problémát okozhat, ha a kreditmennyiség állandó, és csak az átlagos fájlméretben különböznek a vizsgált rendszerek.

1. fejezet

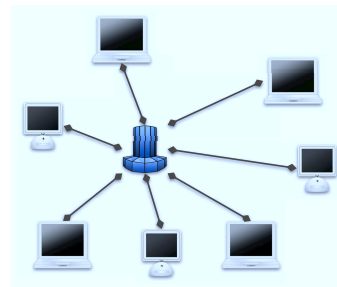
Alapfogalmak

1.1. Peer-to-peer hálózatok

Az elosztott rendszerek olyan hálózati alkalmazásarchitektúrák, melyek a felhasználók között osztják szét a feladatokat és információt. Két fő megvalósítási módjuk a szerver-kliens modell és a peer-to-peer (p2p) modell, amelyek illusztrációit az 1.1. ábrán mutatjuk be. A szerver-kliens architektúra a központi forrás segítségével nyújt szolgáltatásokat. Megvalósítási szempontból egyetlen központi szervert és sok klienst tartalmaz, melyeknek eltérő a feladata: a szerver felelőssége, hogy a kliensek felől érkező kéréseket kezelje, a kliensek pedig a tőle érkező adatokat dolgozzák fel. A peer-to-peer kifejezést ezzel szemben olyan rendszerekre alkalmazzuk, ahol a munka az összes komponens között van elosztva. Ezekben a hálózatokban tehát nincsen kitüntetett központi szerver, minden résztvevő fogad és küld is adatot, valamint mindannyian hozzájárulnak a közösség működéséhez a saját erőforrásaikkal.



(a) p2p



(b) szerver-kliens

1.1. ábra. A p2p valamint a szerver-kliens architektúrák illusztrációi

1.1.1. A p2p hálózatok szerkezete

A p2p modellen alapuló átfedő hálózatoknak két típusa van: strukturálatlan és strukturált. Előbbi esetben semmilyen megkötés sincs a hálózat topológiájára nézve, az egymáshoz kapcsolódó peerek alakítják ki véletlenszerűen az éleket. Ennek előnye, hogy a rendszerek könnyen felépíthetőek, a megoldás ugyanakkor hátránnyal is jár: amikor egy felhasználó (peer) szeretné megszerezni a kívánt fájl egy darabkáját, a kérésnek a teljes hálózaton végig kell futnia, hogy a lehető legtöbb alkalmas feltöltőt megtalálja, aminek nagy az erőforrásigénye. Ezt a gyengeséget küszöbölik ki a strukturált p2p hálózatok, melyekben a peerek előre meghatározott topológiába vannak rendezve. Ezen rendszerek leggyakrabban ún. DHT-t (elosztott hash táblát) használnak, mely (kulcs, érték) párokat tartalmaz a fájlok és tulajdonosaik azonosításához, így biztosítja a részt vevő peereknek a hatékony keresést. A DHT-t a hálózat résztvevői tartják karban, szintén elosztott módon. A legismertebb strukturált p2p hálózatok között találjuk a Chordot és a Pastry-t, míg a BitTorrent, a Napster és a Gnutella a strukturálatlan rendszerek táborát erősíti.

1.1.2. Alkalmazási területek

Mivel a szolgáltatás terhe a részt vevő peerek között van elosztva, a rendszer hatalmas hálózat esetén is jól skálázódik. Ez jelentős előnyt jelent például a tradicionális fájlmegosztókkal szemben, a felhasználók számának növekedése ugyanis a teljesítmény fokozódását vonja maga után. Napjainkban a p2p számos alkalmazási lehetőséget biztosít:

- fájlmegosztás, amely tekinthető a p2p tradicionális alkalmazásának,
- VOIP - itt elsősorban a Skype-ra kell gondolni, amely egy speciális szuperpeer hálózaton működik,
- video-on-demand, amely alkalmazás az utóbbi időben egyre nagyobb teret nyert,
- és a live streaming, amely az élő közvetítések megtekintését teszi lehetővé.

Ezen alkalmazások közül a fájlmegosztás a legnépszerűbb, érdemes tehát az ezt lehetővé tevő BitTorrenttel foglalkozni.

1.2. BitTorrent

1.2.1. Alapfogalmak

A BitTorrent széles körben használt p2p protokoll és fájlcsere rendszer. Alapötlete, hogy minden fájl darabkákra (vagy szeletekre) vágunk szét, majd a letöltő peerek (leecherek)

egyszerre akár több társukhoz is csatlakozva összegyűjtik a hiányzó részeket [4]. Azokat a peereket, akik rendelkeznek egy fájl minden részével seedereknek nevezzük. Az azonos fájlban érdekelt peerek tehát együttműködnek, ők alkotják a swarmokat. A le- és feltöltések megkezdésének könnyítésére a BitTorrent egy központi elemet alkalmaz, a tracker, amely egyben segíthet a keresett tartalmak megtalálásában is, bár ez utóbbi csak másodlagos jelentőségű. Minden letöltés előtt a leecher csatlakozik a trackerhez, mely egy véletlen listát ad vissza azon peerekről, melyek rendelkeznek a kiszemelt fájlal (ők tehát az online seederek), vagy annak bizonyos részeivel (az online leecherek). Ezután a leecher azon felhasználókhöz csatlakozik, akik rendelkeznek olyan darabkával, amelyre szüksége van. A letöltendő darabkák sorrendjét az határozza meg, hogy milyen gyakorisággal fordulnak elő a rendszerben - minél ritkábban, annál nagyobb valószínűséggel fogja azt letölteni a leecher (rarest first). Ez a stratégia segít a darabkák gyors szétterítésében és abban is, hogy a leecherek minél többet töltsenek fel és le egymás között, részben mentesítve ezalól a seedereket. A feltöltési sávszélesség kihasználásával nincs szükség nagy kapacitású központi szerverekre, így rendkívül hatékony rendszert kapunk.

1.2.2. Ösztönző mechanizmus

Természetesen szükséges, hogy a felhasználókat feltöltésre ösztönözzük: a BitTorrent erre egy kölcsönösségen alapuló megközelítést, a tit-for-tatet (TFT) alkalmaz, amely a játékelmélethez ismert iterált fogoly dilemma nevű probléma legrobosztusabb megoldási stratégiája [3]. A TFT biztosítja, hogy egy leecher csak akkor fog egy másik leechernek feltölteni, ha az viszonzásul megosztja vele az általa birtokolt darabokat, ily módon a kizárólag letöltő peereket büntetéssel sújtja. A probléma ezzel a megközelítéssel az, hogy arra nem ösztönzi a peereket, hogy letöltés után is a rendszerben maradjanak, és immár a teljes fájl birtokában megosszák azt a többi felhasználóval. Alapvetően minden letöltés után a leecherek ugyanazon swarmban maradnak és seederekké válnak, olyan felhasználókká, akik önzetlenül töltik fel a többieknek az általuk birtokolt szeleteket. A seederek jelenléte a rendszerben kulcsfontosságú, ugyanis új swarm létrehozásához minimum egy olyan felhasználó szükséges, akitől a többiek beszerezhetik a kívánt tartalmat. A sok seedert tartalmazó swarmok magas letöltési arányt biztosítanak a leechereknek, amely növeli a rendszer hatékonyságát.

Bár a seederek rendszerben tartásának előnyei nyilvánvalóak, a felhasználók többsége ezt nem veszi figyelembe egy-egy letöltéskor, s a kívánt fájl megszerzése után elhagyja a swarmot további feltöltés nélkül: ezt a helyzetet nevezzük hit and run-nak (H&R). Látható, hogy a H&R komoly nehézségeket okozhat egy BitTorrent rendszernek, melyek kiküszöbölésére napjainkban egyre több zárt BitTorrent közösség bukkan fel, amelyekben, mint azt a következő fejezetben részletesen tárgyaljuk is, különféle szabályozások

vannak érvényben az elérhető tartalmak letöltésére.

A kapcsolódó szakirodalomban is számos javaslatot találunk P2P rendszerek ösztönző mechanizmusának megoldására, elsősorban valamilyen virtuális fizetési eszköz, kredit bevonásával [15, 16, 17].

1.2.3. A `.torrent` fájlokról

A `.torrent` fájl a BitTorrent rendszerekre jellemző állomány, amely metaadatokat tartalmaz az elosztott fájlokról és mappákról, valamint általában magában foglalja a trackerek elérhetőségét is. Ezen fájlok tehát magát az adatot nem tartalmazzák, csak annak bizonyos jellemzőit: nevet, méretet, mappaszerkezetet, az integritásellenőrzéshez szükséges információkat. Megtévesztő lehet, hogy sokan torrent alatt nem feltétlenül magát a `.torrent` állományt, hanem a tényleges fájlt értik – ez mindig kontextusfüggő.

1.2.4. További részletek

A BitTorrent protokoll tartalmaz továbbá egy choke algoritmust, amely a leecherek között létrejövő kapcsolatokat befolyásolja. Minden peer fenntart állandó kapcsolatot a swarmban résztvevő többi peer egy részhalmazával, amely általában 40 és 80 közötti elemszámú. Ezt a halmazt a peer szomszédságának hívjuk. A leecher a szomszédai egy további részhalmazának tölt fel. A feltöltésre rendelkezésre álló kapacitását egyenlő részben osztja fel az ún. feltöltési résekbe (upload slot). Kétfajta feltöltési rést különböztetünk meg: általános rések (ebből általában 4 darab van) és az optimisztikus rést. Az előbbiben lévő peerekkel a TFT alapján megy a le és feltöltés, míg az utóbbi újabb (gyorsabb) peerek felderítésére használatos.

Bár a BitTorrent hatékonyságához nem férhet kétség, több próbálkozás is született annak növelésére. Ilyen például a rarest-first és a choke algoritmusok lecserélésére vonatkozó cikk [2], amelyben kimutatták, hogy ezen két stratégia elengedhetetlen része a rendszereknek, és kiváltásuk nem jár előnyökkel, mint azt korábban remélték.

1.2.5. BitTorrent kliensek

Az eredeti BitTorrent protokoll számos kiterjesztésének, teljesítménynövelő javításának technikai kivitelezése a különböző BitTorrent kliensek létrejöttét eredményezték. Ezek közül a legelterjedtebbek a

- uTorrent, amely a legnagyobb felhasználói táborral rendelkezik,
- Vuze, amelyet többek között a video-on-demand képességével is felvérteztek,

- Transmission, amely a Linux disztribúciók standard BitTorrent kliense.

Megemlítjük továbbá, hogy az Opera böngésző beépített BitTorrent klienst tartalmaz. A Tribler nevű kliens (amelyet a Delfti Műszaki Egyetemen fejlesztenek) részleteit gyakran akadémiai kutatási eredmények motiválták - ilyen például a tartalmak központi szerver nélküli keresésének lehetősége [7].

1.3. BitTorrent közösségek

A (privát) BitTorrent közösségek jellemzője, hogy ezekben a fel- és letöltés peerenként, központilag van nyilvántartva, mint ahogyan a fájlok is, melyek az adott rendszerben elérhetőek. Ezzel a módszerrel az adminisztrátorok könnyűszerrel nyomon követhetik, hogy ki, mikor, milyen swarmban vett részt, és mennyire támogatta a rendszer fenntarthatóságát. Ezekben a közösségekben a feljebb említett problémák hatására gyakori a különböző ösztönző politikák bevezetése a rendszer jó összfeltöltés/összletöltés hányadosának elérése érdekében. Ilyen módszer lehet például a H&R detektálása és büntetése, a megosztási arány vagy a kreditmennyiség ellenőrzése, sőt ezek tetszőleges kombinációja is sikerre vezethet. Kimutatott tény, hogy a zárt BitTorrent közösségekben az elérhető letöltési sebesség sokkal magasabb, a seedelési idő sokkal hosszabb, valamint a seederek/leecherek aránya is nagyobb, mint a nyílt rendszerekben [6].

1.3.1. Megosztási arány kikényszerítés

A zárt BitTorrent közösségek egyik legelterjedtebb ösztönző módszere a felhasználók megosztási arányának figyelemmel követése. Ennek keretein belül minden peer köteles a megosztási arányát (az összes feltöltésének és az összes letöltésének hányadosa) a küszöbértékkel egy szinten, vagy a felett tartani (a küszöbérték meghatározása a rendszeradminisztrátor feladata). Ennek elmulasztása miatt a tagok először figyelmeztetést kapnak, majd eltávolítják őket a letöltéstől; végső esetben a közösségből való kizárást is maga után vonhatja.

1.3.2. Kreditrendszer

A megosztási arány felügyeletével egyenértékű módszer a peerek által birtokolt kreditmennyiségek ellenőrzése [18]. Ebben az esetben minden peer köteles kreditjeinek számát (mely az összes feltöltésének és összes letöltésének különbsége) legalább a 0 szinten tartani. Mindkét technikánál jól látható, hogy az egyes felhasználók nem tudnak kibújni a központi ellenőrzés alól, és a seederek száma jelentősen növelhető a nyitott BitTorrent rendszerekhez képest.

1.3.3. A törekvés jutalmazása

Megemlítjük, hogy az utóbbi időben elterjedni látszik az a szabályozás, amely szerint egy adott zárt közösségben regisztrált felhasználók adott mennyiségű seedelési idővel is teljesíthetik a feltöltésre vonatkozó követelményeket. Ez a szabály tehát a rendelkezésre állás tényét jutalmazza, ilyenkor az a peer is teljesítheti a követelményeket, aki akar dolgozni (feltölteni), csak nem tud (mert például a fájl már nem népszerű, nem akarja senki sem letölteni). Ennek a szabálynak az alkalmazásával foglalkoznak a [9, 18] cikkek.

1.3.4. Gyenge pontok

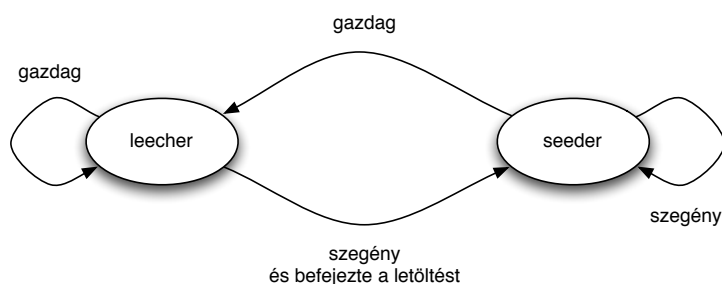
Mindezek ellenére azonban egy rendszer teljesítménye akkor is gyenge lehet, ha minden peerje önzetlenül tölt fel annyit, amennyit csak tud [14]. Előfordulhat ugyanis, hogy a seedelés következtében néhány peernél összpontosul a kreditmennyiség nagy része. Ilyen esetekben a többieknek nem jut elég kredit az egészséges áramlás fenntartásához, amely a rendszer teljesítményének romlásához, szélsőséges esetben pedig összeomláshoz vezet. Az ehhez hasonló helyzeteket, amelyekben a rendszer bedőlését a túl kevés kredit okozza, s így módon minden peer seedelni kénytelen, *crunch*nak nevezzük. Hasonlóan problémát okozhat az is, ha a felhasználók által birtokolt kreditek mennyisége túlságosan nagy - ilyen esetekben nincsen motiváció feltölteni, s újfent összeomlik a rendszer (*crash*).

2. fejezet

A rendszer modellje

2.1. A rendszer modellje

A felhasznált szimulátor absztrakciója a valós életbeli privát közösségeknek és magában foglalja azok meghatározó jellemzőit. A modell működése körökön alapul, melyek 1 időegységet jelentenek. Körönként minden peer aktiválódik, és végezhet valamilyen tevékenységet: folytathatja a megkezdett le- és feltöltését, vagy újakat inicializálhat. A modellezett közösségben kreditrendszer működik, a peerek a feltöltéssel szerezhetnek maguknak krediteket, melyeket letöltésekre költhetnek – ezt illusztrálja a 2.1 ábra.



2.1. ábra. A rendszer állapotdiagramja (a [14] cikk alapján)

2.1.1. A Trackerről

A szimulátorban megvalósított tracker a BitTorrent közösségekéhez hasonlóan központi komponens. Nyilvántartja azon swarmokat, melyek aktuálisan a közösségek számára elérhetőek, valamint tárolja a peerek által bejelentett fel- és letöltési mennyiségeket. Az alkalmazott tracker kredit alapú ösztönző politikát használ: ha egy peernek nincsen a fájl mérettel megegyező mennyiségű kreditje, seedelés útján kell elegendőt szereznie. A szimuláció indításához mindig meghatározunk egy gazdag-szegény arányt, majd a gazdag

peereknek annyi kreditet adunk, amennyivel elkezdhetik a letöltést, a szegények pedig 0-ról indulnak, ők lesznek kezdetben a seederek. Érdekes még megjegyezni, hogy zárt közösség lévén a rendszerben lévő kredit összmenyisége állandó.

2.1.2. Rendszermodell

A közösséget a benne lévő peerek reprezentálják: N db online peer van a rendszerben, egyenként U feltöltési kapacitással, mely a feltölthető szeletek számát adja meg az időegység (1 kör) függvényében. A letöltési sebesség alapesetben 8 darabka/kör, az ettől eltérő értékeket a szimulációknál jeleztük. Feltesszük továbbá, hogy a közösség zárt, azaz a szimulációk során nem érkezik új peer a rendszerbe. Ennek hiányában egy új felhasználó extra kreditet hozna a közösségbe, amely befolyással van annak fenntarthatóságára [18]. Fentebb láthattuk, hogy a valóságban is több, szinte teljesen zárt rendszer működik, így ez az egyszerűsítés nem sérti a modell valós mivoltát.

2.1.3. Adatfolyam

A letöltési modell a BitTorrent TFT mechanizmusát követi: a seederek szeleteket töltenek fel a leechereknek, akik egymás közt a már meglévő darabkáikat cserélgethetik. Minden felhasználó célja, hogy egyesével haladva az összes, a rendszerben lévő fájlt letöltse. Ha a letöltés után nincs elég kredit egy újabb fájl megszerzéséhez, a felhasználó a legutóbbi swarmban marad seederként, ameddig szükséges. Ezután a következő letöltendő fájl kiválasztása egyenletes eloszlás szerint történik. Bár utóbbi ugyan nem jellemző egy valós rendszerre, hiszen ott egyre bővül az elérhető fájlok halmaza, melyekből tetszés szerint válogathatnak a peerek, a szimulátorban lévő fájlok nincsenek megkülönböztetve, ezzel biztosítjuk az újabb fájlok bekerülésének lehetőségét. Amikor egy peer befejezte a letöltést, két eset lehetséges: marad elegendő kreditje egy újabb fájl letöltéséhez, és az előzőek szerint folytatja, vagy szegénnyé válik, és az aktuális swarmban marad seedelni.

2.1.4. Swarmok szintje

A közösség S db swarmot tartalmaz, melyekben tetszőleges számú peer tartózkodhat. A valóságban nincs megkötés, hogy egy peer pontosan hány swarmban vehet részt - egynél több esetén feltöltési sebessége megoszlik a swarmok között. Ugyanakkor a megosztási aránya aggregálódik, így feltehetjük, hogy minden pillanatban egy peer egyszerre csak egy swarmban van jelen, tehát egyértelműen meghatározható, hogy egy adott pillanatban a felhasználó leechernek vagy seedernek számít-e. Bár a valós rendszerek némileg bonyolultabbak, jelenleg nem foglalkozunk a hit'n'runokkal, és feltételezzük, hogy min-

den peer maximálisan kihasználja a feltöltési sebességét. Egy kör alatt minden swarm feltöltési egységeket (darabkákat) oszt ki a seederek és leecherek, illetve a leecherek és a leecherek között (attól függ, milyen típusú felhasználó a feltöltő az adott tranzakcióban). Az utóbbi variáció épp a BitTorrentben lévő tit-for-tat mechanizmus megvalósítása.

2.1.5. Tranzakciók

A tranzakciók a következőképp működnek: minden fájl meghatározott méretű darabkákra van szétosztva, és a BitTorrentben szokásos módon minden peer négy feltöltési szállal rendelkezik, mely 1–1 darabkának az átvitelére képes egységnyi idő alatt. Minden körben a swarmokban lévő peerek véletlen sorrendben aktiválódnak. Amikor egy peer sorra kerül, véletlenszerűen választ négy leechert, akiknek feltölt. Az átküldött darabok sorrendjét a BitTorrentből átvett rarest first szabály határozza meg, vagyis azon szeletek kerülnek először átadásra, melyeknek a legkevesebb replikájuk van jelen a swarmban. A gyakoriság szerint sorba rendezett daraboknak köszönhetően mind a négy feltöltési szálon azon szeletek kerülnek feltöltésre, melyekkel az adott leecher még nem rendelkezik, ezzel is növelve a rendszer hatékonyságát. Amennyiben a kiválasztott leecher már minden darabkával rendelkezik, melyeket a feltöltő kínálhat, akkor véletlenszerűen újabb peert választ ki. Ez a mechanizmus biztosítja, hogy minden körben maximálisan ki legyenek használva a feltöltési lehetőségek.

2.2. Gyakorlati megvalósítás

A szimulátorban tetszőleges számú független futtatás indítható egymás után (*runok*). Minden futtatás körökből épül fel, melyek száma szintén tetszőleges, így egyaránt lehetséges rövidebb és hosszabb időtartamokat lefedni a szimulációk során. A `Config.java` tartalmazza a futtatásokhoz szükséges paramétereket:

- a fentiek mellett megadható, hogy hány swarmot (S) és peert tartalmazzon a rendszer, utóbbiakból hány százalék kezdjen gazdagként (ők lesznek az első körben a leecherek),
- továbbá tetszőlegesen megadható a fájlok és a darabkáik mérete és a közösségben szétosztott kreditmennyiség is.

A szimulátor mindezekén túl arra is lehetőség nyújt, hogy a rendszerben lévő peerek közül véletlenszerűen kiválasszunk néhányat, akiknek a letöltési sebessége kisebb lesz az alapbeállításnál - ez a jelen dolgozatban használt szimulációk során a szabványosnak

megfelelő 4 darabka/kör feltöltési és 2 darabka/kör letöltési sebességet jelent. Ennek a funkciónak a finomhangolását is a konfigurációs osztályban tehetjük meg.

Minden futtatás két fő részből áll: először a swarmokat és peereket inicializáljuk a `populatePeersandSwarms` metódusban, majd a körök futtatásai következnek. Kezdetben létrehozuk a megadott számú swarmot, mindet azonos paraméterekkel - természetesen az ID adattagot kivéve. A rendszerben lévő kreditek mennyisége kétféleképp alakulhat:

- alapesetben a kezdéskor minden gazdag peer egy fájl letöltéséhez elegendőt kap, a szegények pedig 0 kredittel indulnak.
- A másik lehetőség, hogy megadjuk, összesen hány kreditet szeretnénk szétosztani a peerek között - ez a `Config` osztályban tehető meg, a szétosztást pedig a `Runner`-ben lévő `calcInitialCredits` metódus végzi. Ekkor először megpróbálunk minden gazdag peernek egy fájlhoz elegendő kreditmennyiséget adni, majd ha maradtak még szétosztható kreditek, sorra vesszük az összes peert, és további kreditekkel növeljük az általuk birtokolt mennyiséget.

Mindkét szétosztási körben először mindig a gazdag, majd a szegény peerek kerülnek sorra. Ha 5 kredit, vagy annál kevesebb maradt kiosztás közben, azt a következő alkalmas peer kapja, ha több, akkor a szétosztott kredit mennyisége egyenletesen véletlenül kerül kiválasztásra a $[0, \text{kreditmennyiség} \times \text{fájlméret}]$ intervallumból. Szegény peer esetén a kisorsolt kreditmennyiség csak akkor adható át a felhasználónak, ha az általa így birtokolt kreditmennyiség kisebb, mint a fájl mérete; ezzel biztosítjuk a kezdeti gazdag-szegény arány megtartását.

Ennek figyelembe vételével tehát adott mennyiségű és típusú peer inicializálódik a fenti stratégiák valamelyike szerint szétosztott kreditekkel. Miután egy peert létrehoztunk, sorsolunk neki egy swarmot, majd a kreditjeinek száma alapján egy seeding vagy egy leeching sessiont hozunk létre számára. A swarmok kiválasztása egyenletes valószínűséggel történik. Mivel a peerek véletlenszerűen kerülnek a swarmokba, előfordulhat, hogy akad olyan fájl, amely senkinek sincs meg, így a rendszerben lévő leecherek nem tudják letölteni. Emiatt az inicializálás utolsó lépéseként minden swarmot ellenőrzünk, és a seeder nélkülieket töröljük a rendszerből a hozzájuk tartozó esetleges letöltő sessionökkel együtt. Megjegyezzük, hogy a letöltő szálak törlése nem okoz problémát, mert a szimulátor a következő körben az ilyen módon swarm nélkül maradt leechereket egy másik fájlhoz irányítja át, a swarmok pedig életük során többféle fájlt is szimbolizálhatnak (egyszerre természetesen csak egyet).

Minden kör (*generation*) a tranzakciók lebonyolításával kezdődik (ennek megvalósítását a `transactSwarmsAndPeers` metódus tartalmazza). Elsőként a swarm összes

peerjére sorra vesszük, a fájl mely darabjai vannak meg neki, és minden darabkát a rarest-first szabálynak megfelelő sorrendben megpróbálunk feltölteni egy egyenletesen véletlenül választott peernek a swarmból. A szimulátorban lehetőség van a leecherek feltöltési módjának szabályozására is, a háromféle verzió:

- leechelés közben korlátlan mennyiségű darabkát oszthatnak meg a többi leecherrel,
- csak egyetlen fájlméretnyit tölthetnek fel (ekkor a folyamatban lévő letöltésük után rögtön egy másik swarmba mehetnek át újabb letöltést kezdeményezni),
- vagy egyáltalán nem tölthetnek fel letöltés közben (ekkor tehát kizárólag a seederektől lehet darabkákat szerezni).

Egészen addig folytatjuk a darabkák feltöltését, amíg azoknak a végére nem érünk, vagy az adott peer fel nem használta az összes rendelkezésre álló slotját.

A tranzakciók után következik a peerek állapotának frissítése: itt a felhasználó állapota alapján több lehetséges esetet különböztetünk meg:

- Ha a peer seedel, ellenőrizzük, elegendő kreditet gyűjtött-e össze a következő letöltéshez, és ha ez teljesül, akkor töröljük a jelenlegi feltöltő sessiont.
- Hasonló módon, ha leecherként vett részt egy swarmban, és ebben a körben befejezte a letöltést, akkor töröljük a sessiont, továbbá feljegyezzük a fájlt.

Ez utóbbi információra két esetben is szükségünk lehet: egyrészt felhasználható, amikor a peernek legközelebb seedelnie kell, másrészt újonnan letöltendő swarm keresésekor meggyorsítja a döntést. Ezután a következő körbeli viselkedését állítjuk be: ha nincsen még elegendő kreditje a letöltéshez, de jelenleg nem rendelkezik feltöltő sessionnel sem, a legutóbb letöltött fájlhoz tartozó swarmhoz adjuk hozzá seederként. Megjegyzendő, hogy mivel a modellünk nem címkézi a swarmokat, előfordulhat, hogy a következő körben már másik fájlt jelölünk ugyanazzal a sorszámmal. Ez nem probléma, ugyanis ilyen esetben a szóban forgó peerre tekinthetünk úgy, mint aki már rendelkezik az adott fájljal - akár egy korábbi, a rendszerben történt letöltés eredményeként, akár úgy, hogy ő maga adta hozzá az új fájlt a rendszerhez. Amennyiben a vizsgált peer megfelelő mennyiségű kredittel rendelkezik egy új letöltés elkezdéséhez, és nincsen folyamatban letöltése, egyenletes eloszlás alapján új, megfelelő swarmot kezdünk keresni számára. A kiválasztott swarmmal szemben támasztanunk kell bizonyos követelményeket: az adott peer nem leechelheti éppen a fájlt, valamint legalább 1 seedernek jelen kell lennie a swarmban, hogy legyen kitől letöltenie. A folyamatos modellünk miatt ekkor kétféle eset lehetséges: a swarm még nem szerepel a leecher letöltött fájlokat tartalmazó listájában, ekkor, ha az előző feltételek teljesülnek, elindulhat a letöltés (ilyenkor biztosak lehetünk benne, hogy bármilyen fájlt is

jelöljön éppen a swarm, a leecherünk még egyiket sem töltötte le, hiszen ekkor a swarm objektum a listájában lenne). A másik opció, hogy a swarm már szerepel a listában, ekkor feltételezzük, hogy időközben ugyanahhoz az ID-hez másik fájl tartozik.

Ebben az esetben, mivel nem tudjuk, mikor váltott identitást a swarm, illetve váltott-e egyáltalán, számolnunk kell annak a lehetőségével, hogy ugyanazt a fájlt jelöli, amelyet a leecher már megszerzett korábban. A rendszer egységének megtartásához ilyenkor kell feltételeznünk, hogy a swarm éppen ettől a pillanattól jelöl másik fájlt, azaz a leecher úgy fog belépni a swarmba, hogy egy új állományt kezd majd letölteni. Ha az utóbbi lehetőség áll fenn, csak akkor adhatjuk hozzá az új letöltőt, ha a swarm vagy csak seedereket tartalmaz, vagy ha a seederek mellett olyan leecherek vannak jelen, akik még egyetlen darabot sem töltöttek le a fájlból - ellenkező esetben ugyanis megszakítanánk a folyamatban lévő letöltéseiket, amelyet az idealizált modellünk nem enged meg, hiszen megbomlana a rendszer konzisztenciája.

Előfordulhat, hogy a fenti feltételek mellett nem sikerül megfelelő swarmot találni egy peernek, ilyenkor az értelemszerűen nem vesz részt semmilyen tranzakcióban a következő kör során. A rendszer minden egyes körben újra leellenőrzi, hogy a közösségekben jelen lévő swarmok közül megfelel-e valamelyik a peernek, hiszen a tranzakciók és az idő múlása miatt mind a swarmok által jelölt fájlok, mind a swarmokban lévő peerek összetétele változik.

A tranzakciók végrehajtása és a peerek állapotainak frissítése után előfordulhat, hogy egyes swarmok seederek nélkül maradnak. Törekedni kell az ilyen helyzetek megszüntetésére, ugyanis ebben az esetben a fájlt nem lehet majd letölteni, ezért csökken a rendszer teljesítménye. A `checkandPopulateDeadSwarms` metódus a következőképp működik: a körök végén minden egyes swarmot sorban ellenőriz, és ha olyat talál, melyben jelenleg egyetlen seeder sem tartózkodik, a többi swarm közül egyenletesen véletlenszerűen kisorsol egyet, és megnézi, jelenleg hány seeder tartózkodik benne. Ha a sorsolt swarmban nincsen legalább 2 feltöltő, megismétli a keresést, hiszen az ott lévő egyetlen seeder átirányítása újabb üres swarmot eredményezne. A keresést összesen legfeljebb $5S$ alkalommal ismételheti meg. Ha eközben sikerült megfelelő swarmot találnunk, végigmegyünk a sessionjein egészen addig, amíg egy seedelő session nem kerül sorra. A hozzá tartozó peer lesz a megfelelő seeder, akit átküldünk az üres swarmba. Ebben az esetben, miután a swarmokat nem követjük nyomon, ismét feltehetjük, hogy csak olyan fájlhoz küldünk seedert, amelyiket az már korábban letöltötte. A swarmok ellenőrzésével véget ér az adott kör, és a következővel folytatódik a szimuláció.

3. fejezet

Szimulációk

A [14] cikkben megmutatták, hogy ha a kezdeti kreditmennyiség az idő múlásával a rendszerben nem változik, akkor bizonyos kezdeti gazdag arányokra (pl. 10% és 90%) a rendszer nem lesz fenntartható, 80% esetén pedig nem lehet megjósolni, összeomlik-e vagy sem. A cikkben használt program szolgáltat alapjául a jelent fejezetben is használt szimulátornak.

Ebben a fejezetben ismertetjük a szimulációs vizsgálatainkat, valamint tárgyaljuk azok eredményeit. Megjegyezzük, hogy minden eredményünk 5 futtatás eredményének átlaga.

3.1. A fájl méret és a rendszer összeomlásának kapcsolata

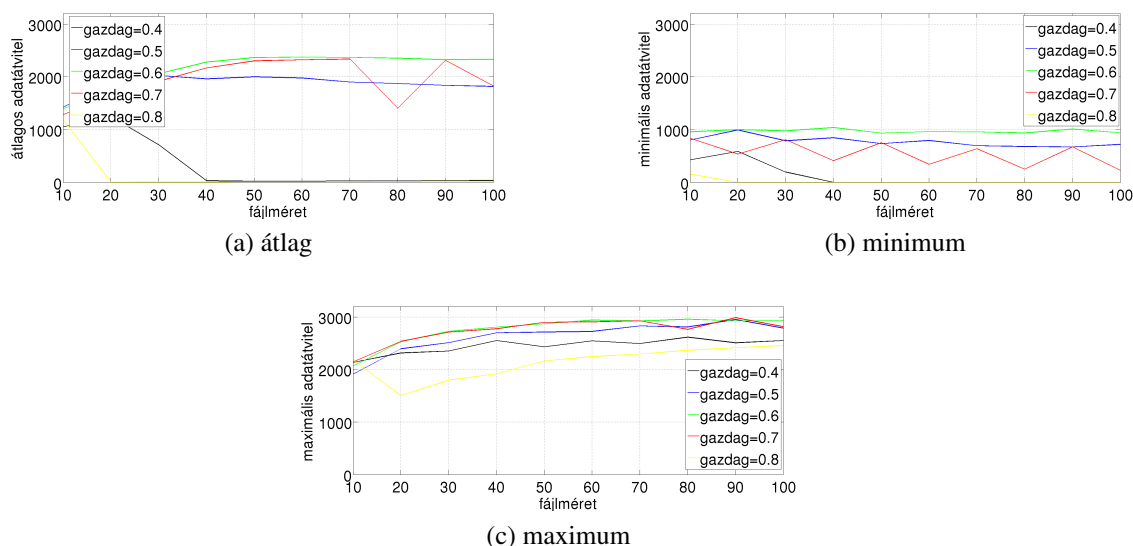
A [14] cikk eredményeit megismerve és reprodukálva látható volt, hogy a rendszer összeomlása nem kizárólag a szegények és gazdagok kezdeti arányától függ, hanem az átlagos fájl méret is jelentősen befolyásolhatja azt. A jelenség mélyebb tanulmányozásához különböző szimulációkat futtattunk a következő közös paraméterekkel: minden esetben $N = 1000$ peer és $S = 100$ swarm van jelen a rendszerben, a BitTorrent hagyományainak megfelelően minden felhasználó $U = 4$ db feltöltési szállal rendelkezik. A fájlok azonos méretűek (F), a darabkák mérete 1, ez tehát a feltöltési szálak szempontjából azt jelenti, hogy egy felhasználó egy kör alatt 4 darabkát tud átadni másoknak. A felhasználók célja, hogy minél több fájlt letöltsenek. Ha nem áll rendelkezésükre a megfelelő mennyiségű kredit, addig seedelnek, amíg egy fájl letöltéséhez elég kreditet össze nem szednek (mind a seederek, mind a leecherek tölthetnek tehát fel egymásnak). Minden szimuláció 2000 körből áll, és 5-ször ismételtük a futtatásokat. A kezdeti kredit mennyiségét a szokásos módon határoztuk meg: minden gazdag felhasználó 1 fájl letöltéséhez elegendő kreditet kap, a szegények pedig 0 kredittel indultak, az összkredit mennyisége tehát $N \cdot \text{gazdagok_aránya} \cdot F$. A szimulációk során a fájl méretet 10-től 100-ig növeltük, és minden egyes fájl méret esetén a gazdagok aránya 40%-80%-ig változott. A [14] cikkben

megmutatták, hogy az ilyen jellegű rendszerek 10%, 20% és 90%-os gazdag arány esetén nagyon hamar összeomlanak, így hasonló eredményekre jutva az ezen arányokra kapott értékeket mi sem jelenítettük meg statisztikáinkban.

3.1.1. Homogén peerek, alapeset

Az első szimulációban a fentiek felül minden peer azonos letöltési sebességgel rendelkezik, amely 8 darabka/kör. Ez a beállítás azt a megfigyelést mutatja a rendszerben, miszerint a BitTorrent közösségekben a feltöltési sebesség a szűk keresztmetszet. A 3.1a, 3.1b és 3.1c ábrák a rendszer átlagos, minimális és maximális adatátvitelét ábrázolják az átlagos fájl méret és a gazdagok arányának függvényében, 5 futtatás eredményeit átlagolva. Az adatátvitel szimulációinkban a tranzakciókban gazdát cserélő darabkák és kreditek mennyiségét is jelenti. Jól látható, hogy a fájl méret növekedésével a maximális adatátvitel is növekszik, amint az várható volt. A maximális értékeket vizsgálva ugyanakkor figyelembe kell vennünk, hogy a rendszer nem minden esetben fenntartható, amely erről az ábráról nem olvasható le - számos esetben ugyanis a rendszer nagy "lendülettel" indul el, azonban a nem optimális kezdeti konfiguráció miatt hamar lecsökken a teljesítmény. Ebből a szempontból több információt szolgáltat a minimális és az átlagos átvitel vizsgálata: ezeknél szembeötlő a 40% és 80%-os gazdag arány melletti hanyatló teljesítmény. Abban az esetben, amikor a gazdag peerek kezdetben 80%-ban voltak jelen, a rendszer 10-nél nagyobb fájl méret esetén biztosan összeomlott. Ennél árnyaltabb a helyzet a 40%-os gazdag arány esetén: 20-as fájl méret mellett a rendszer ugyan minden esetben fenntartható volt, de az átlagos teljesítmény a többi esettől elmaradó értéke már előre vetítette a problémát. 30-as fájl méretre ugyanez a rendszer már nem volt képes minden esetben működni, ez jól látható ha akár a minimális átmenetet tekintjük a maga 195-ös értékével, akár az átlagos átmenetet, amely csupán 705-ig emelkedett. Láthatjuk, hogy 40%-os gazdag arány mellett a 30-nál nagyobb fájl méretek esetén a rendszer hamar összeomlott.

Mindez azért is érdekes, mert a [14] cikkben tárgyaltak alapján a 40%-os kezdeti gazdag arány egy biztosan fenntartható rendszert jelentett. 80% esetén ugyanakkor a vizsgált rendszer ingadozott, és a fájl méret növelésével megmutatható, hogy ezekben az esetekben egyértelműen összeomlás vár rá, még ha a maximális adatátvitel értéke nagyobb fájlok esetén növekvő tendenciát is mutat. A többi konfiguráció esetén hasonló eredményeket kaptunk a [14]-ben említettekhez, ezek a kezdeti arányok tehát stabilan fenntartható rendszereket eredményeznek.

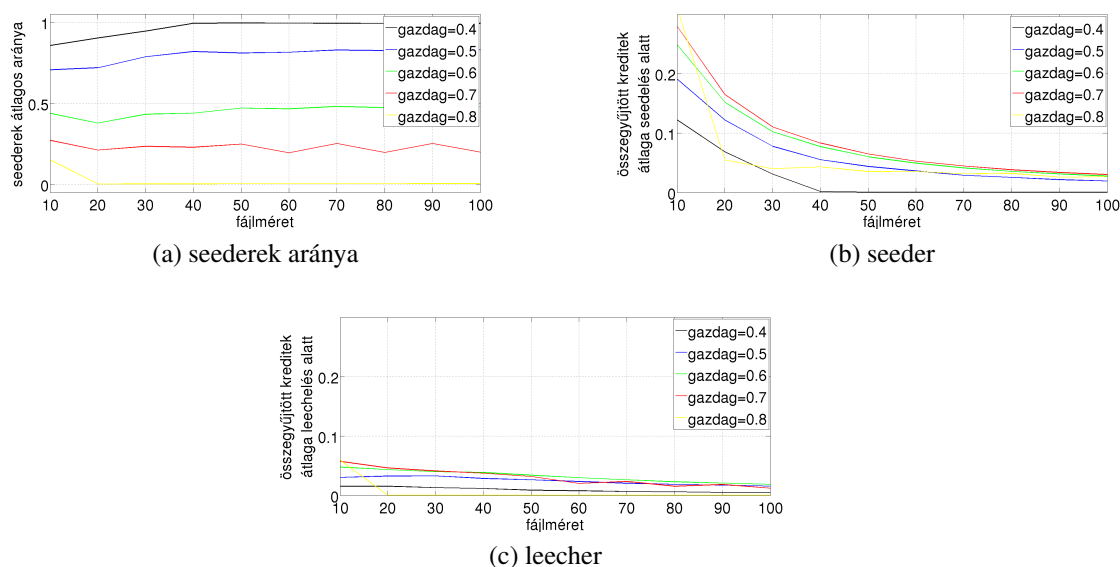


3.1. ábra. Teljesítménymutatók (homogén peerek, alapeset)

3.1.2. Homogén peerek, korlátozott feltöltés

A bedőlő rendszerek összeomlásának okait kutatva természetes módon merül fel a rendszerben áramló kredit nyomon követése. A bevezetésben említettük, hogy a rendszerösszeomlásoknak két fajtája van: crunch esetén nincsen elegendő kreditje a peereknek, hogy megkezdhessék a következő fájl letöltését, crash esetén pedig éppen ellenkezőleg, a túlzott kreditmennyiség miatt senki sincs motiválva, hogy feltöltsön. Látható tehát, hogy egyáltalán nem mindegy, a kreditek egy adott pillanatban milyen arányban oszlanak meg a közösség online felhasználói között. Vizsgálataink során tapasztaltuk, hogy az időnként elnyúló leechelési idő alatt előfordulhat, hogy egy peer kimagaslóan nagy mennyiségű kreditet szerez a feltöltéssel, akár annyit, hogy két fájl letöltése között nem kell visszatérnie seedelni. Erre az esetre akár úgy is tekinthetünk, hogy adott körben egy, ugyanabban a swarmban tartózkodó seeder elől vette el a krediteket. Az ilyen helyzetek miatt érdemes megvizsgálni, hogy a különböző kimenetelű rendszerekben a seederek és leecherek milyen arányban képviseltetik magukat, és a körök alatt milyen mennyiségű kreditet képesek szerezni a feltöltések során. Ahhoz, hogy felmérhessük, ténylegesen hatással van-e a leecherek által felvett kreditmennyiség a rendszer fenntarthatóságára, elkészítettük szimulációk egy olyan csoportját, melyben a leecherek letöltés alatt szerkezhető kreditmennyiségét az átlagos fájl mérettel korlátoztuk.

A 3.2a ábrán láthatjuk a fájl méret és az első körben gazdag peerek kezdeti arányának függvényében a seederek átlagos arányát az előző rendszerben a 2000 kör alatt. Érdekességgként megjegyezzük, hogy bár az intuíció azt sugallja, az átlagos 50–50%-os gazdag-szegény eloszlás fenntartható rendszert eredményezne, az optimális arány valószínűleg szorosan függ a kezdeti gazdag-szegény eloszlástól, ráadásul kis mértékben a fájl mé-



3.2. ábra. Kreditgyűjtés (homogén peerek, korlátozott feltöltés)

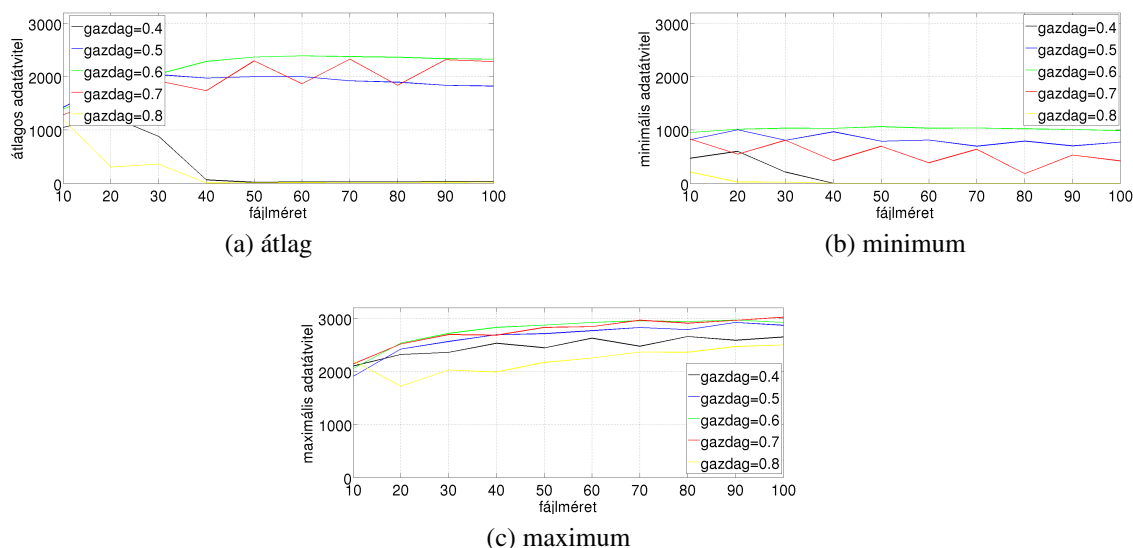
ret is befolyásolja. A 3.2b ábrán láthatjuk, hogy a fájl méret és a gazdagok kezdeti arányának függvényében egy seeder egy kör alatt átlagosan hány fájlra elegendő kreditet szerzett az alapesetként használt rendszerben. A 3.2c ábra hasonló paraméterekkel az egy leecher által a letöltés közben 1 kör alatt szerzett fájlok számát jeleníti meg. A seederek átlagos száma, valamint a seedelés és leechelés alatt szerzett fájlok tekintetében az összehasonlított rendszerek között nem találtunk különbséget. A leecherek feltöltésének korlátozása tehát nincs befolyással a peerek által különböző fázisokban szerzett kreditmennyiségekre. Ami a vizsgált rendszerek teljesítményét illeti, a minimális, maximális, és az átlagos adatátvitel értékeiben sem mutatkozott jelentős eltérés.

3.1.3. Heterogén peerek, alapeset

Napjainkban a felhasználók számos internetszolgáltató és csomag közül válogathatnak, így az elérhető le- és feltöltési sáv szélességek igen nagy tartományban mozognak. Így, bár a peerek egyforma és magas letöltési sebessége megengedhető relaxáció, a fenti szimulációkat olyan helyzetekre is lefuttattuk, amikor kétféle peer van jelen a rendszerben. Ebben az esetben a felhasználók feltöltési sebességei a megszokott 4 darabka/körös értéken maradtak, véletlenszerűen azonban olyan peereket inicializálunk, akiknek a letöltési sebessége negyede a többiekéhez képest - esetünkben tehát 2 darabka/kör vagy 8 darabka/kör lehet ez az érték.

Az előzőekben vizsgált kezdeti gazdag arányok nagy részénél nem látható jelentős változás, de a 40% és 80%-os értékeknél akadnak eltérések, lásd 3.3 ábra. Előbbinél megfigyelhető, hogy míg a minimális és maximális átvitt adatmennyiség egyik fájl méret

esetében sem változott számottevően, addig az átlagos érték 30 és 40 egység nagyságú fájlokra megnőtt. Ez egyenes következménye annak, hogy míg az előző esetben rendre bizonytalanul működő, majd összeomló rendszert kaptunk, addig a különböző letöltési sebességű peerek részvétele esetén 40%-os kezdeti gazdag arányt és 30-as fájl méretet tekintve fenntartható rendszer az eredmény, és még a 40-es fájl méret esetén is akadt olyan eset, amikor az előzőekhez képest jóval később álltak le a tranzakciók. Még inkább beszökő a változás 80%-nyi gazdag peer esetén: bár a szimulációk többségében a 20-as és 30-as fájl méretre is összeomlott a rendszer, az előző eredményeinkkel szemben előfordult, hogy fenntarthatóan viselkedett. Ezekben az esetekben ráadásul a rendszer átlagos átvitele alig 15%-kal maradt el a legjobb eredményt produkáló arányéhoz képest, ezt mutatja a 3.3c ábra.



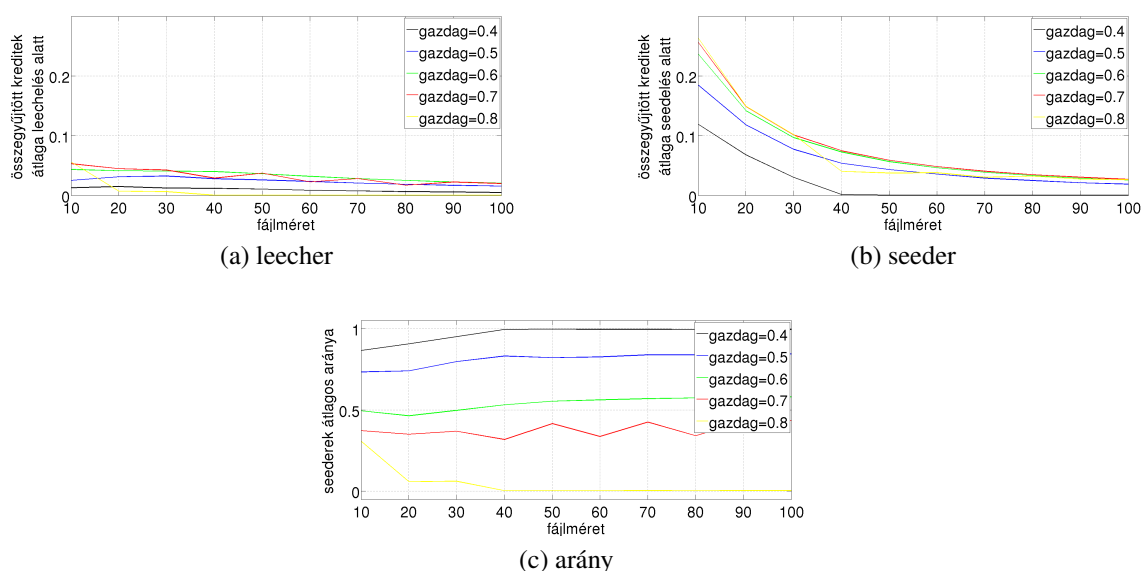
3.3. ábra. Teljesítménymutatók (heterogén peerek, alapeset)

Előző eredményeinket alátámasztják a 3.4a és 3.4b ábrák is, melyeken jól látható, hogy 80%-os gazdag-szegény arány, valamint 20-as és 30-as fájl méret esetén a homogén rendszerhez képest jelentősen megnőtt mind a leecherek, mind a seederek által szerzett átlagos kreditmennyiség.

A kapott eredmények biztatóak, látható ugyanis, hogy a valósághoz közelebb álló modellben a különböző peerek bevezetése a közösségek stabilabb viselkedését eredményezi. A kapott stabilabb rendszerben természetesen a seederek átlagos aránya is megnőtt (lásd 3.4c ábra).

3.1.4. Heterogén peerek, korlátozott feltöltés

Az 3.1.2 fejezethez hasonlóan ebben az esetben is megvizsgáltuk, befolyásolja-e a rendszer működését, ha limitáljuk a leecherek által egy letöltési session alatt feltölthető darab-



3.4. ábra. Kreditgyűjtés (heterogén peerek, alapeset)

kák mennyiségét. Homogén peerek esetén ez a korlátozás semmilyen jelentős hatást nem gyakorolt a rendszerre, a heterogén esetben azonban visszaesést tapasztalunk a limit nélküli feltöltést alkalmazó heterogén rendszerhez képest. Érdekes ugyanakkor észrevenni, hogy ezzel a jelenséggel a kiinduló állapothoz jutottunk, vagyis viselkedés szempontjából a homogén rendszerek hasonlóak a heterogén, limitált feltöltést megengedő közösségekhez.

3.2. A rendszerek fenntarthatósága a kezdeti kreditmennyiség tükrében

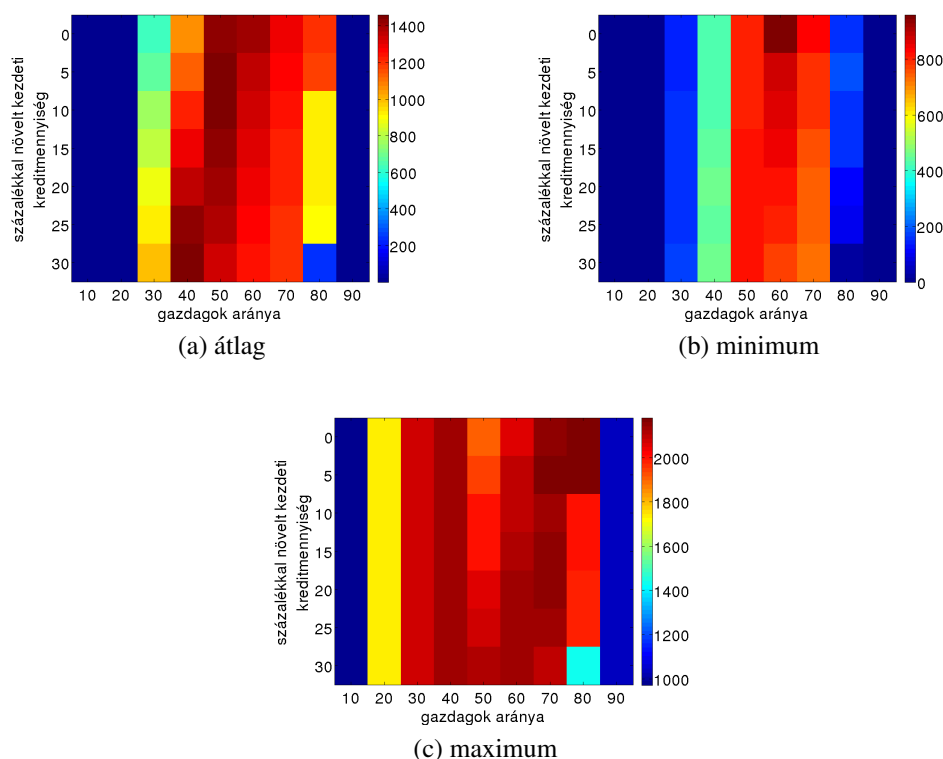
Az előző szakaszban láthattuk, hogy több tényező is befolyásolja a kredit alapú rendszerek fenntarthatóságát. A következőkben azt vizsgáljuk, hogy az alapértelmezett kezdeti kreditmennyiség növelésével milyen módon változik a közösségek viselkedése. A rendszer alap paraméterei megegyeznek az eddigiekkel.

3.2.1. Homogén peerek, alapeset

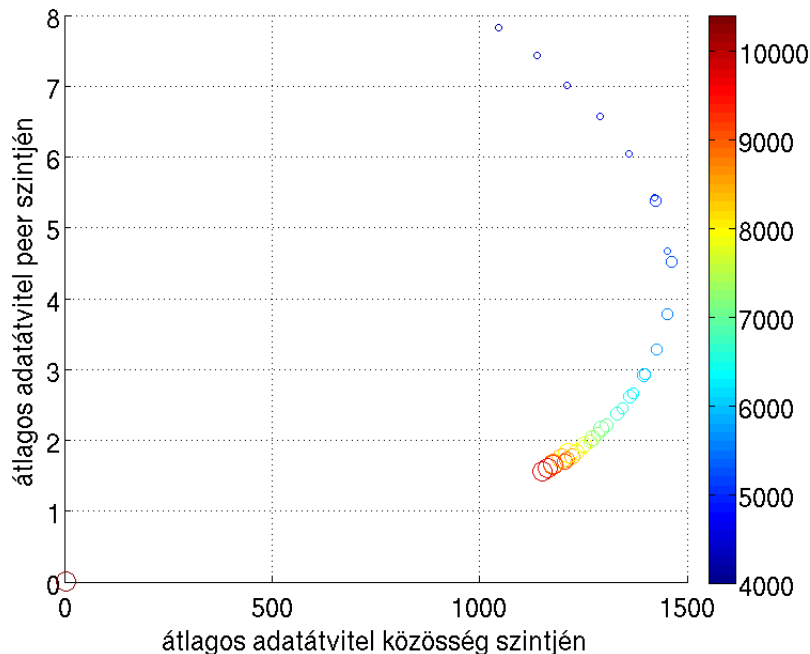
A korábbi szimulációk logikai sorrendjét megtartva most is azzal a rendszerrel kezdjük a vizsgálódást, melyben minden peer azonos letöltési sebességgel rendelkezik és a leecherek korlátozás nélkül tölthetnek fel a leechelés alatt. A rendszerbe injektált kredit mennyisége a kezdeti $N \cdot \text{gazdagok_aránya} \cdot F$ értékről indul, majd 5%-onként növelve annak 1,3-szereséig vizsgáljuk a kapott közösségeket. Ezek az esetek egyébként olyan

situációkat generálnak, amelyekben a gazdag peereknek általában nem pontosan egy fájl letöltésére elegendő kredit áll rendelkezésükre, valamint a szegény peereknek sem feltétlenül zérus a kreditmennyiségük. Ilyen situációkat egy valós rendszerben is kaphatunk, ahol a rendszer adminisztrátorai beavatkoznak a hatékonyság növelésének érdekében ún. freeleech vagy freeseed periódusok beiktatásával, vagyis kredit injektálással vagy éppen elvonással. Az eredmények illusztrálásaként hő térképeket készítettünk, amelyeken az x -tengelyen a rendszerben a futtatások kezdetekor található gazdagok arányát, míg az y -tengelyen a különböző kezdeti összkredit értékeket tüntettük fel. Az y -tengelyen tehát a 0 értékhez tartozó sor megegyezik az előző szakasz első szimulációjában tárgyalt futtatásokkal 10-es fájl méretre. A hő térképeken az átlagos (3.5a ábra), a minimum (3.5b ábra), illetve maximum (3.5c ábra) adatátviteli mennyiség értékeit tüntettük fel, 5 futtatásra átlagolva. A kék szín alacsonyabb, a piros pedig a magasabb értékeket mutatja. Megjegyezzük, hogy 10% és 20% gazdag arány esetén semmilyen vizsgált kezdeti kreditmennyiségre nem kaptunk fenntartható rendszert.

Amennyiben az átlagos adatátviteli mennyiségét vizsgáljuk, érdekes átlós mintát láthatunk a piros színnel kódolt (magas) értékekre. Ez egyértelműen megmutatja, hogy ha a gazdagok aránya alacsony, akkor a kiosztott kreditmennyiség növelésével, amennyiben pedig ez az arány magasabb, akkor a kreditmennyiség csökkentésével a rendszer teljesítménye növelhető.



3.5. ábra. Teljesítménymutatók (homogén peerek, alapeset)



3.6. ábra. Közösség versus egyéni teljesítmény

Ha a futtatás során a körönként mért adatátvitel mennyiségének minimumát vesszük teljesítménymutatónak (lásd a(z) 3.5b ábrát), akkor ugyanez az átlós minta megtalálható, itt azonban természetesen kisebb értékeket jeleznek a színek. Érdekes ugyanakkor észrevenni, hogy míg plusz kredit injektálása 30%, 40% és 50%-os gazdag arány esetén növelheti a rendszer átlagos adatátvitelét, addig sajnálatos módon a minimális átvitelre szinte egyáltalán nincs hatással. A legmagasabb minimális adatátviteli értéket a 60%-os gazdag arány és 0%-os kezdeti kreditmennyiség mellett kapjuk. Megjegyezzük, hogy ennek a teljesítménymértéknek abban az esetben van jelentősége, ha a vizsgált BitTorrent rendszer video-on-demand típusú, ahol azt kívánjuk biztosítani, hogy az adatátvitel minimuma legyen maximális.

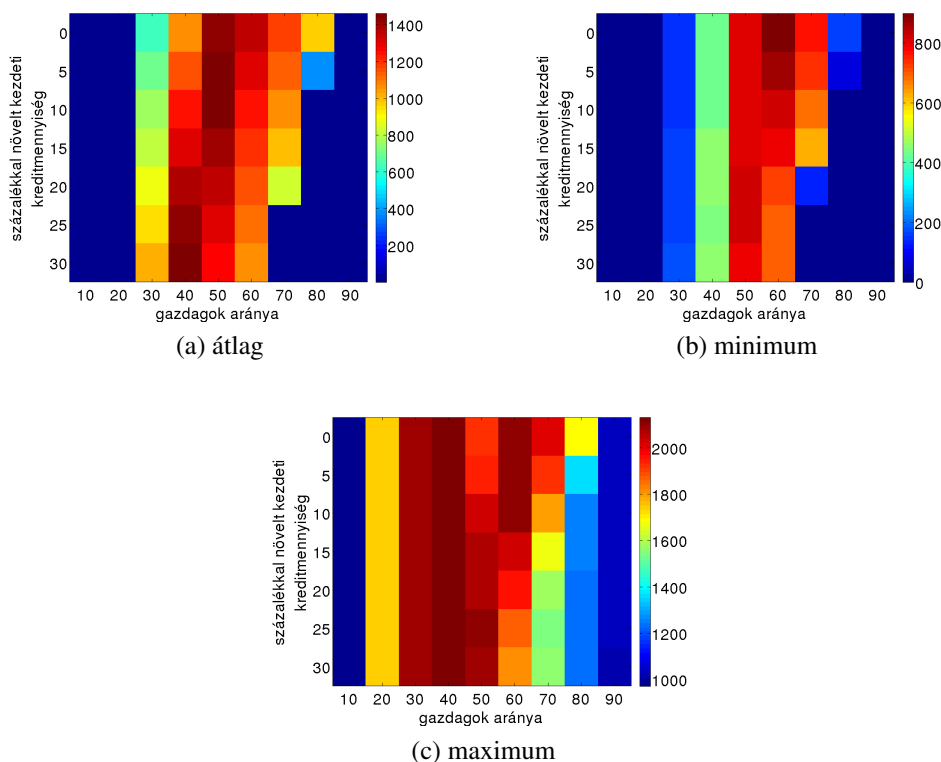
A maximális adatátviteli mennyiségeket a 3.5c ábra mutatja. Itt meg kell jegyeznünk, hogy az értékek önmagukban fenntartással kezelendők, elég csak azt az esetet megnézni, amikor 80%-os gazdag aránnyal indulunk. Ebben az esetben 10%-nál nagyobb kredit-növelés esetén viszonylag magas maximális átviteli értékeket kapunk, ugyanakkor a rendszer hajlamos az összeomlásra.

Eddigi vizsgálódásaink során csak azzal foglalkoztunk, hogy milyen hatással van a paraméterek változtatása a rendszer egészére. Érdekes megfigyelni emellett azt is, hogy a közösség tagjai szempontjából hogyan alakult pl. az átlagos adatátvitel a szimulációk során. A 3.6 ábra a következőket tartalmazza: az x tengelyen a közösség átlagos adatátviteli értékeit találjuk 10-es fájl méret esetén (ez megegyezik az 3.1.1 eset-

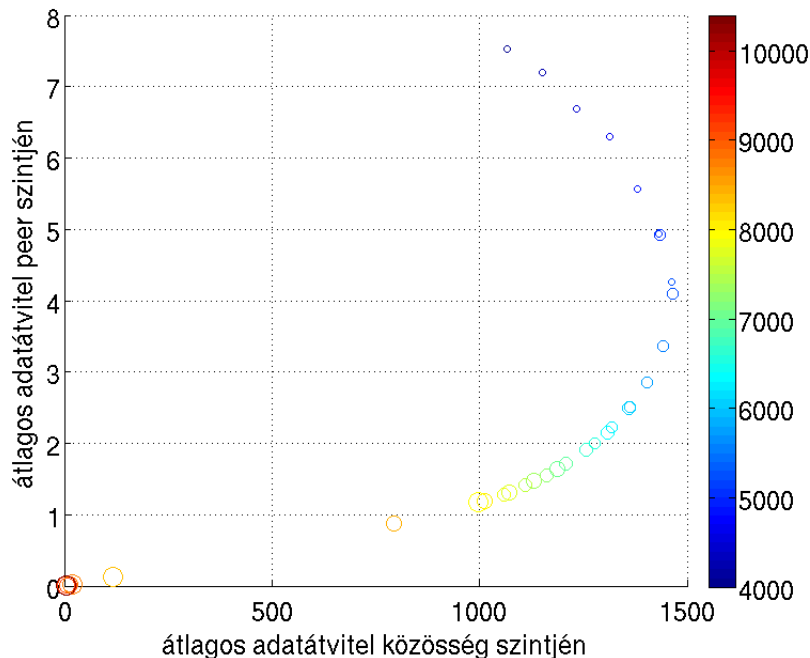
tel), az y tengelyen pedig a felhasználók által átlagosan szerzett fájlok számát. A karikák különböző nagysága a kezdeti gazdag-szegény arányra utal (minél nagyobb, annál több gazdag peerrel indultunk), a színek pedig a rendszer kezdeti kreditmennyiségét jelölik a hőtésképekhez hasonlóan. Jól látható, hogy a szimulált rendszerek esetében is a klasszikus dilemmával kerülünk szembe: az adott paraméterek mellett nem létezik olyan konfiguráció, amellyel a peerek és a rendszer átlagos adatátvittele egyaránt maximális lenne. Tisztán látszik ugyanakkor, hogy vannak kifejezetten kedvező kezdeti arányok és kreditmennyiségek, amelyekkel a teljesítmény mindkét fronton növelhető.

3.2.2. Homogén peerek, korlátozott feltöltés

Az előző szimulációk párhuzamosan megvizsgáltuk azt is, hogyan reagál a rendszer akkor, ha a leecherek feltöltésének mértékét a fájl mérettel korlátozzuk, minden mást pedig változatlanul hagyunk (lásd 3.7 ábra). Jól látható, hogy a kapott közösség sokkal érzékenyebb, ráadásul a biztosnak számító 70%-os aránnyal induló rendszer is összeomlik, ha túlságosan sok kredittel indítjuk. Ezt a megfigyelést az egyének és közösség átlagos átvitelének nagyságát együttesen ábrázoló 3.8 ábra is tükrözi, ahol az előző esethez képest jóval több zérus értékű kombináció található, ráadásul a gazdag-szegény arány és a kreditmennyiség együttes növekedéséhez tartozó körök is jobban tartanak a kisebb értékek felé.



3.7. ábra. Teljesítménymutatók (homogén peerek, korlátozott feltöltés)



3.8. ábra. Közösség versus egyéni teljesítmény

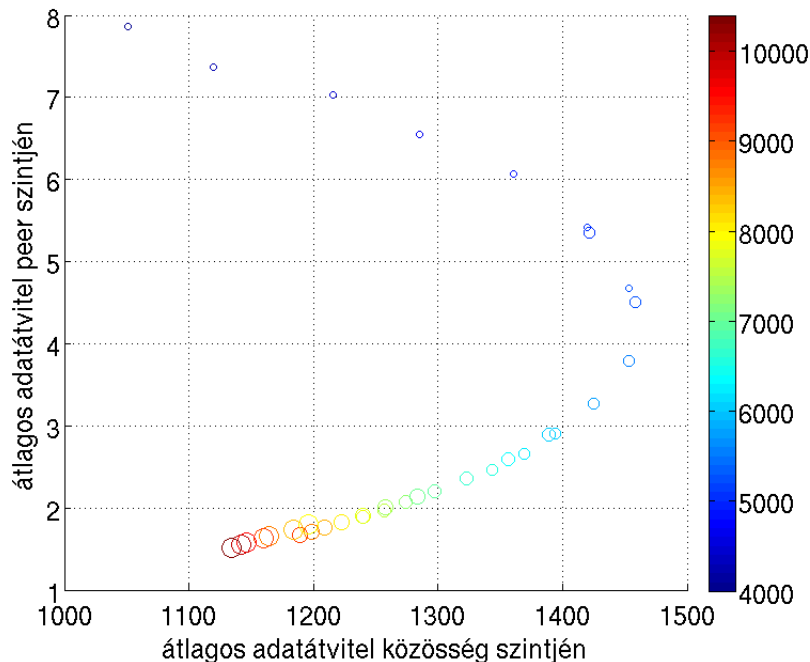
Mégis megfontolandó ennek a verzióknak az alkalmazása, ugyanis ha a minimális és maximális átviteli értékeket tekintjük, 50% és 60%-os kezdeti gazdag aránynál bizonyos esetekben már kis mennyiségű extra kredittel is növelhetjük a teljesítmény határértékeit.

3.2.3. Heterogén peerek, alapeset

A korábbi vizsgálatokkor láthattuk, hogy az eltérő letöltési sebességű peerek bevezetése képes javítani a rendszer teljesítményén, így a következő lépés ezen helyzetek elemzése. Az eddigiekhez hasonlóan először korlátlan feltöltést engedélyezünk a leechereknek, a többi paraméter az előzőekkel megegyező. Szembeötlő változást ekkor csupán a 80%-os kezdeti arány esetén találtunk, azonban a rendszernek nem csak az átlagos és maximális átvitele, hanem a stabilitása is növekedett. A 3.9 ábráról leolvasható, hogy a teljesítmény növekedése magában foglalja, hogy a rendszer szempontjából átlagosan 1000 alatti adatátviteli értékek egyáltalán nem fordulnak elő.

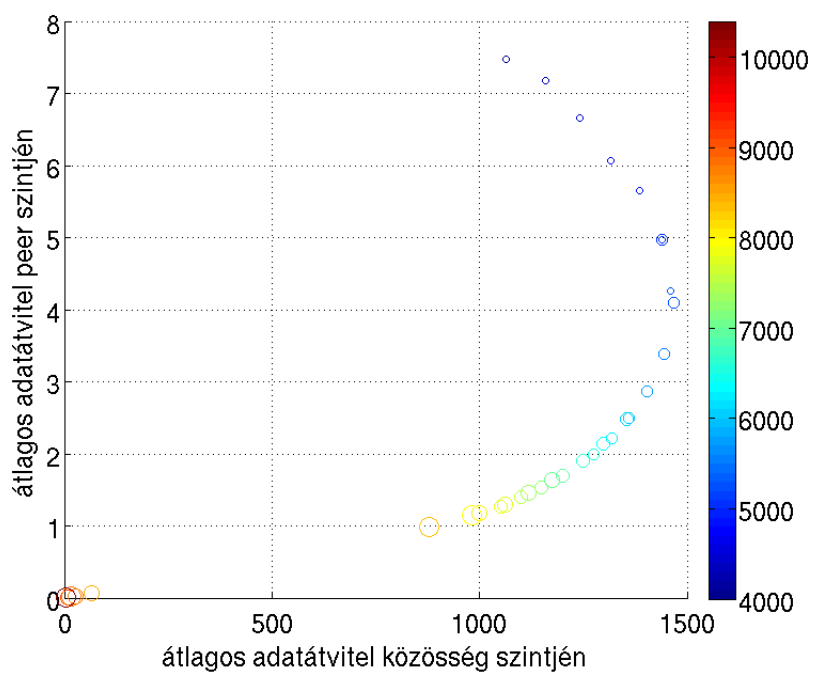
3.2.4. Heterogén peerek, korlátozott feltöltés

Korlátozva a letöltés alatt szerezhető kreditek mennyiségét az előző rendszerben, hasonló eredményekre jutottunk, mint azonos letöltési sebességű peereket tartalmazó közösségekben. Az átlagos átvitel csökkenéséért cserébe 20-60%-os aránynál az esetek többségében növelhető a rendszer maximális adatátvitele, 50%-nál pedig a minimális átvitel értéke



3.9. ábra. Közösség versus egyéni teljesítmény

is magasabb. A közösség és peer szinten elért átlagos átviteli értékeket mutató ábráról (3.10) az is leolvasható, hogy a rendszer nagyobb átlagos teljesítményének újfent ára van. Közel 0,5-es csökkenés látható ugyanis akármelyik vizsgált értéknél, ha a felhasználók által a futtatások alatt átlagosan gyűjtött fájlok számát vizsgáljuk; ráadásul sokkal több, főleg magas kezdeti gazdag arányhoz tartozó értéket találunk a 0 körül, ami azt sugallja, hogy alacsony vagy magas gazdag-szegény arány esetén sokkal érzékenyebb rendszert kaphatunk.



3.10. ábra. Közösség versus egyéni teljesítmény

4. fejezet

Összefoglalás

A dolgozatban a BitTorrent rendszerek fenntarthatóságát vizsgáltuk bizonyos paraméterek változásának tükrében. Az első részben az átlagos fájl méret változtatásával kapott rendszerek összeomlásának feltételeit kutattuk. Többféle típusú közösséggel foglalkoztunk: a letöltési sebesség szempontjából homogén és heterogén peereket tartalmazó rendszerek mellett azt is megfigyeltük, hatással van-e a fenntarthatóságra a leecherek letöltésük alatt az egymásnak történő feltöltéssel szereshető kreditjeik mennyisége. A második részben a kezdeti kreditmennyiséget változtattuk az előbb említett közösségekben, és azt vizsgáltuk, javítható-e a rendszer teljesítménye az alapesetnél több kredit injektálásával.

A fájl méretek vizsgálatainak eredményeképp kaptuk, hogy a különböző rendszerek fenntarthatósága valóban nem kizárólag a gazdag peerek kezdeti arányától, hanem a rendszerben lévő fájlok átlagos méretétől is jelentős mértékben függ. Ráadásul a heterogén peerek bevezetésével kapott közösségek, melyek közelebb állnak a valós rendszerekhez, kevésbé voltak hajlamosak összeomlani.

A másik fő irány, a kreditmennyiség változtatása mentén elindulva a következő eredményekre jutottunk: a kezdeti kreditmennyiség növelése bizonyos határokon belül, adott gazdag-szegény induló arányokra képes növelni a rendszer átlagos átviteli teljesítményét. Más kezdeti paraméterek esetén ugyanakkor a felhasználók átlagos adatátviteli növelhető, így az alkalmazott rendszer céljának függően célzottan tudunk javítani annak teljesítményén. Emellett, ha a leecherek által feltöltött adat mennyiségét korlátozzuk, a rendszer minimális és maximális teljesítményén is javíthatunk, így tehát a közösségek többféle teljesítménymutatójának növelésére is sikerült megoldást találni.

Köszönetnyilvánítás

Szeretnék köszönetet mondani témavezetőmnek, **Dr. Vinkó Tamás**nak a sok segítségért és türelméért, amelyet dolgozatom készítése során kaptam tőle.

Irodalomjegyzék

- [1] Andrade, N., Santos-Neto, E., Brasileiro, F., and Ripeanu, M., Resource demand and supply in bittorrent content-sharing communities, *Computer Networks* 53 (2009) 515–527.
- [2] Arnaud Legout, G. Urvoy-Keller, and P. Michiardi. 2006. Rarest first and choke algorithms are enough. In Proceedings of the 6th ACM SIGCOMM conference on Internet measurement (IMC '06). ACM, New York, NY, USA, 203-216.
- [3] Axelrod, R., The evolution of cooperation (Basic Books, 1984).
- [4] Cohen, B., Incentives Build Robustness in BitTorrent (2003).
- [5] Feldman, M., Papadimitriou, C., Chuang, J., and Stoica, I., Free-riding and white-washing in peer-to-peer systems, in Proceedings of the ACM SIGCOMM workshop on Practice and theory of incentives in networked systems, PINS'04 (ACM, New York, NY, USA, 2004), pp. 228–236.
- [6] Meulpolder, M., D'Acunto, L., Capotua, M., Wojciechowski, M., Pouwelse, J., Epema, D., and Sips, H., Public and private BitTorrent communities: A measurement study, in Proceedings of the 9th international conference on Peer-to-peer systems (USENIX Association, 2010), pp. 10-10,
- [7] Pouwelse, J., Garbacki, P., Wang, J., Bakker, A., Yang, J., Iosup, A., Epema, D., Rein- ders, M., Van Steen, M., and Sips, H., Tribler: a social-based peer-to-peer system, *Concurrency and Computation: Practice and Experience* 20 (2008) 127–138.
- [8] Zhang, C., Dhungel, P., Wu, D., Liu, Z., and Ross, K. W., Bittorrent darknets, in Proceedings of the IEEE INFOCOM (2010).
- [9] R. Rahman, M. Meulpolder, D. Hales, J.A. Pouwelse, D.H.J. Epema, and H.J. Sips. Improving efficiency and fairness in P2P systems with effort-based incentives. In Pro- ceeding of IEEE ICC, 2010.

- [10] I.A. Kash, E.J. Friedman, and J.Y. Halpern, Optimizing scrip systems: Efficiency, crashes, hoarders, and altruists, Proc. ACM Conference on Electronic Commerce, 2007.
- [11] I.A. Kash, J.K. Lai, H. Zhang, and A. Zohar, Economics of BitTorrent Communities, Proc. NetEcon, 2011.
- [12] X. Chen and X. Chu, Measurements, Analysis and Modeling of Private Trackers, Proc. IEEE P2P, 2010.
- [13] X. Chen, X.W. Chu, and Z. Li, Improving Sustainability of Private P2P Communities, Proc. IEEE ICCCN, 2011.
- [14] Rameez Rahman, David Hales, Tamás Vinkó, Johan Pouwelse, Henk Sips. No more crash or crunch: sustainable credit dynamics in a P2P community. In Proceedings of the International Conference on High Performance Computing and Simulation (HPCS 2010), pp. 332-340, 2010.
- [15] Turner, D.A. and Ross, K.W. A Lightweight Currency Paradigm for the P2P Resource Market. In Proc. 7th International Conference on Electronic Commerce Research, 2004.
- [16] Garbacki, P. and Epema, D.H.J. and van Steen, M. An Amortized Tit-For-Tat Protocol for Exchanging Bandwidth instead of Content in P2P Networks. In Proc. First International Conference on Self-Adaptive and Self-Organizing Systems, pp.119–128, 2007.
- [17] Ramachandran, A. and Sarma, A.D. and Feamster, N. BitStore: An Incentive-Compatible Solution for Blocked Downloads in BitTorrent. In Proc. 2nd Joint Workshop on Economics of Networked Systems and Incentive-Based Computing, 2007.
- [18] Jia, A.L.; Rahman, R.; Vinko, T.; Pouwelse, J.A.; Epema, D.H.J., "Systemic Risk and User-Level Performance in Private P2P Communities," IEEE Transactions on Parallel and Distributed Systems, 24(2013), pp.2503–2512.