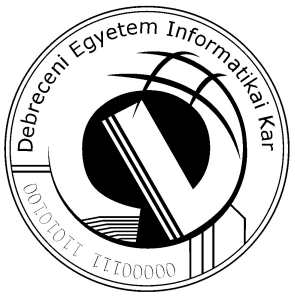


Proxy Cache szerverek hatékonyságának vizsgálata

Performance Modeling of Proxy Cache Servers

Doktori (PhD) értekezés

Bérczes Tamás

Témavezető:
Prof. Dr. Sztrik János

Debreceni Egyetem
Természettudományi Doktori Tanács
Informatikai Tudományok Doktori Iskolája
Debrecen, 2011

Köszönetnyilvánítás

Ezúton is szeretnék köszönetet mondani mindazoknak akik közvetve vagy közvetlenül hozzájárultak a disszertációm elkészítéséhez.

Szüleimnek, akik felneveltek, támogattak és elindítottak az életben.

Feleségemnek, aki mindenben támogatott.

Bátyámnak, aki végig segített tanulmányaim alatt.

Témavezetőmnek, Dr. Sztrik Jánosnak, a rengeteg hasznos tanácsért, útmutatásért és bátorításért, amely nélkül ez a dolgozat nem készülhetett volna el.

Tartalomjegyzék

1 Bevezetés **1**

2 Sorbanállási rendszerek és hálózatok **5**
2.1. Sorbanállási rendszerek vizsgálata: 5
2.2. Sorbanállási hálózatok vizsgálata 8

3 Web szerver modell **13**

4 Proxy Cache szerver modell **19**
4.1. A javasolt modell 19
4.2. Numerikus eredmények 24
4.2.1. Az érkezési intenzitás hatása a válaszidőre 24
4.2.2. A külső érkezési intenzitás hatása a válaszidőre . . . 25
4.2.3. A fájl méretének hatása a válaszidőre 28
4.2.4. A találati valószínűség hatása a válaszidőre 30

5 A Web szerver és a Proxy Cache szerver meghibásodásának hatása **35**
5.1. A javasolt modell 35
5.2. Numerikus eredmények 40
5.2.1. Eredmények 45

6 A Proxy Cache szerver GI/G/1 approximációs modellje **49**
6.1. A GI/G/1 approximáció 49
6.2. A Proxy Cache szerver GI/G/1 modellje 53
6.3. Numerikus eredmények 60
6.4. Konklúzió 62

7 A heterogén forgalom hatása a Proxy Cache szerverek hatékonyságára **65**

7.1. Több igény-osztályt tartalmazó sorbanállási hálózatok . . . 65
7.2. Módosított Proxy Cache szerver modell 68
7.3. Numerikus eredmények 74
7.3.1. A belső igények érkezési intenzitásának hatása a válaszidőre 74
7.3.2. A külső igények érkezési intenzitásának hatása a válaszidőre 75
7.3.3. A fájlméret hatása a válaszidőre 79

8 Összefoglaló **85**

9 Conclusion **91**

Irodalomjegyzék **95**

Ábrajegyzék

1.1. Proxy Szerver konfiguráció: (a) egyedülálló, (b) transzparens 2

2.1. Egyszerű sor 6
2.2. Folyamatok egyesítése és osztása 9
2.3. Direkt visszacsatolás 11

3.1. Web szerver modell 15
3.2. Fájl méret hatása a Web szerver válaszidejére 16
3.3. Fájl méret hatása a Web szerver válaszidejére 2 17
3.4. A Web szerver sebességének hatása a válaszidőre 17

4.1. Proxy Cache szerver modellje 20
4.2. A belső igények érkezési intenzitásának hatása 1. 26
4.3. A belső igények érkezési intenzitásának hatása 2. 26
4.4. A belső igények érkezési intenzitásának hatása 3. 27
4.5. A belső igények érkezési intenzitásának hatása 4. 27
4.6. A külső igények érkezési intenzitásának hatása 1. 28
4.7. A külső igények érkezési intenzitásának hatása 2. 29
4.8. A külső igények érkezési intenzitásának hatása 3. 29
4.9. A külső igények érkezési intenzitásának hatása 4. 30
4.10. A találati valószínűség hatása a válaszidőre 32
4.11. Az érkezési intenzitás hatása a válaszidőre, módosított paraméter szerver paraméter értékekkel 32

5.1. Proxy Cache szerver modellje 36
5.2. A belső igények érkezési intenzitásának hatása a válaszidőre blokkolt és nem blokkolt esetben 40
5.3. A foglalt Proxy Cache szerver meghibásodásának hatása a válaszidőre 41

5.4. A nem foglalt Proxy Cache szerver meghibásodásának hatása a válaszidőre 41
5.5. A Proxy Cache szerver javítási intenzitásának hatása a válaszidőre 42
5.6. A foglalt Web szerver meghibásodási intenzitásának hatása a válaszidőre 42
5.7. A nem foglalt Web szerver meghibásodási intenzitásának hatása a válaszidőre 43
5.8. A Web szerver javítási intenzitásának hatása a válaszidőre . 43
5.9. Proxy Cache szerver puffer méretének hatása a válaszidőre 1 44
5.10. Proxy Cache szerver puffer méretének hatása a válaszidőre 2 44

6.1. Proxy Cache szerver módosított modellje 53

7.1. Proxy Cache szerver heterogén forgalmi modellje 69
7.2. Heterogén forgalom; belső igények érkezési intenzitásának hatása a válaszidőre 1 75
7.3. Heterogén forgalom; belső igények érkezési intenzitásának hatása a válaszidőre 2 76
7.4. Heterogén forgalom; belső igények érkezési intenzitásának hatása a válaszidőre 3 76
7.5. Heterogén forgalom; belső igények érkezési intenzitásának hatása a válaszidőre 4 77
7.6. Heterogén forgalom; külső igények érkezési intenzitásának hatása a válaszidőre 1 78
7.7. Heterogén forgalom; külső igények érkezési intenzitásának hatása a válaszidőre 2 78
7.8. Heterogén forgalom; Az a osztályú fájlok méretének hatása a válaszidőre 1 82
7.9. Heterogén forgalom; Az a osztályú fájlok méretének hatása a válaszidőre 2 82
7.10. Heterogén forgalom; A b osztályú fájlok méretének hatása a válaszidőre 83

Táblázatok jegyzéke

3.1. **Web szerver modell paraméterei** 18

4.1. **A fájl méretének hatása a válaszidőre** 31
4.2. **A fájl méretének hatása a válaszidőre, módosított paraméterek mellett** 33
4.3. **A Proxy Cache szerver modell jelölései** 34

6.1. **Szimuláció validálása** 62
6.2. **Approximációs eredmények** 63

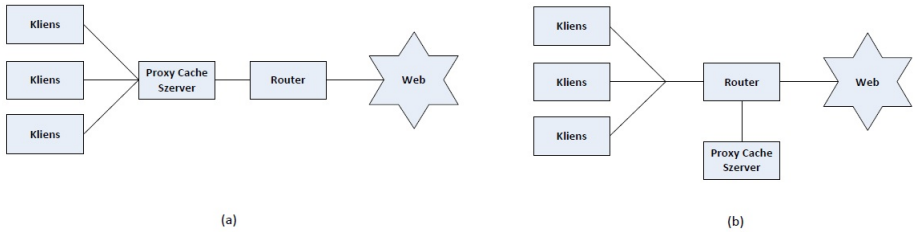
7.1. **Fájlméret hatása a válaszidőre, az a osztály aránya 20%** 80
7.2. **Fájlméret hatása a válaszidőre, az a osztály aránya 40%** 80
7.3. **Az a osztály arányának hatása a válaszidőre** 81
7.4. **Heterogén forgalomi modell paraméterei** 83

1 Bevezetés

Napjainkban az egyik leginkább közkedvelt információszerzési lehetőség az internet használata. Az internet gyors és egyszerű lehetőséget biztosít több ezer Webszerver adatainak a megismerésére, letöltésére. Az internet használata az elmúlt években rohamosan növekedett. A felhasználók száma a 2001-es 474 millióról 2002-re 590 millióra növekedett. 2006-ra az internetet használók száma elérte a 948 milliót. Figyelembe véve, hogy 1996-ban mindösszesen 40 millióan használták az internetet, a növekedés üteme igen jelentős. A felhasználók számának növekedésével párhuzamosan növekedett az internet forgalma is. Ennek hatására egyre nagyobb igény mutatkozik a színvonalas és gyors internet elérésre és kiszolgálásra. Az információ-keresés és letöltés közben a válasz a távoli Web szervertől a kliens gépéig gyakran igen sok időt vesz igénybe. Egy tartalom többszöri és egyidejű elérése miatt a válaszidők növekednek, ezért indokolt a tartalmak felhasználók környékén való tárolása, amely által a késleltetés csökkenthető. Ennek egyik megoldási lehetősége a böngésző szoftverbe való implementálás ([1]). Ebben az esetben a tárolt adatokhoz azonban csak egy személy férhet hozzá. Egy másik lehetőség Proxy Cache szerver használata.

Telepítési helyük alapján a Proxy Cache szerverek két típusát különböztetjük meg: Kliens oldali Proxy Cache szerverről beszélünk, amikor a Proxy Cache szerver a kliens hálózatának része. Megfordított Proxy Cache szerverről beszélünk, amikor a Proxy szerver a távoli Web szerver közelébe van telepítve. Ebben az esetben a Web szerver nagyobb igényszámot tud kiszolgálni, valamint javul a szolgáltatás minősége is. Jelen disszertáció keretében a kliens oldalra telepített Proxy Cache szervereket fogjuk vizsgálni.

A kliens oldali Proxy Cache szervereket a konfigurációjuk alapján 2 csoportba soroljuk:



1.1. ábra. Proxy Szerver konfiguráció: (a) egyedülálló, (b) transzparens

- **Egyedülálló Proxy Cache szerver:**

  Ez a konfiguráció látható az 1.1a. ábrán. A konfiguráció egyik hátránya, hogy amennyiben a Proxy Cache szerver elromlik, a kliensek számára az egész internet kapcsolat elérhetetlenné válik. Ilyen esetben a hálózat újrakonfigurálása szükséges a Proxy Cache szerver javításáig.

- **Transzparens Proxy Cache szerver:**

  Ez a konfiguráció látható az 1.1b. ábrán. Transzparens Proxy Cache szervert használva megszüntethetjük a Proxy Cache szerverek egyik legnagyobb hátrányát, nevezetesen a kliens Web browserek konfigurációjának szükségességét.

A felhasználó szemszögéből nézve lényegtelen, hogy az általa keresett fájl fizikailag hol található: egy Proxy Cache szerveren valahol a munkahelyének belső hálózatán vagy a világ túlsó felén egy távoli Web szerveren. A keresett dokumentum érkezhet a távoli Web szervertől vagy a Proxy Cache szervertől. Kliens oldalról nézve a Proxy Cache szerver funkciója ugyanaz mint egy Web szervernek, valamint a Web szerver felől nézve a Proxy Cache szerver ugyanúgy viselkedik, mint egy kliens.

Feltételezhető, hogy egy Proxy Cache szerver beüzemelése egy cég belső hálózata és az internet közé, kisebb sávszélesség igényt valamint kisebb

válszidőket eredményezhet [14]. Így a vállalatok több felhasználót kapcsolhatnak ugyanakkora sávszélességre, mivel a Proxy Cache szerver redundánsan tárolja az adatokat, több felhasználó számára.

Korábbi kutatások elsősorban a különböző "Cachelési" algoritmusok vizsgálatával illetve fejlesztésével foglalkoztak [1], [4], [26], [31].

Jelen disszertáció keretében a Proxy Cache szerverek hatékonyságát vizsgáljuk meg. Kutatásunk nem a különböző algoritmusok közötti különbségekre irányúl, hanem kifejezetten azt vizsgálja, hogy milyen környezeti feltételek mellett éri meg egy Proxy Cache szerver üzemeltetése [11], [9], [7], [10].

A disszertáció további része a következőképpen van struktúrálva: A 2. fejezetben egy rövid, lényegretörő ismertetőt adok a sorbanállási rendszerekről és hálózatokról, majd a 3. fejezetben bemutatom a [32] által felállított Web szerver modell matematikai alapjait, valamint néhány numerikus eredménnyel szemléltetem a Web szerver modell működését. További Web szerver modellek találhatóak a [22], [21], [19] cikkekben.

A 4. fejezetben bemutatom az általunk általánosított Proxy Cache szerver modellt, majd megvizsgáltam, hogy milyen paraméterértékek mellett érheti meg egy Proxy Cache szerver üzemeltetése, azaz milyen esetekben lesz egy igény teljes válaszideje alacsonyabb Proxy Cache szerver használatával, mint nélküle.

Az 5. fejezetben további általánosításként feltételeztem, hogy mind a Web szerver mind pedig a Proxy Cache szerver elromolhat, azaz nem megbízhatóak. Az így kapott modell bonyolultsága miatt a válaszidők kiszámításához a MOSEL (Modeling, Specification and Evaluation Language) [6] programcsomagot használtam. A MOSEL egy leíró nyelv, mely segítségével különböző programcsomagokat használhatunk, mint például az SPNP-t (Stochastik Petri Net Package) vagy a TimeNet programcsomagot. A MOSEL által szolgáltatott eredményeket grafikusan is ábrázolni tudjuk az IGL (Intermediate Graphical Language) segítségével, mely része a MOSEL-nek.

A 6. fejezetben az érkezési folyamat egy úgynevezett "GI - General interarrival" folyamat, melyet az érkezési időközök várható értékével és a relatív szórásnégyzetével (c^2) jellemzünk, valamint a kiszolgálási idő bár-

milyen általános eloszlású lehet. Az így kapott modellben a rendszerparaméterek kiszámításához a QNA GI/G/1 approximációt használtam [12]. Az aproximáció validálásához egy szimulációs programot készítettem.

A 7. fejezetben megvizsgáltam, milyen hatással van a heterogén forgalom a Proxy Cache szerver hatékonyságára. Ebben az esetben a keresett fájlokat a méretük alapján két osztályba soroljuk. Ebben az esetben a válaszidők kiszámításához külön kell vizsgálni a két csoportba tartozó kérések válaszidejeit, majd ezek segítségével kaphatjuk meg egy tetszőleges igény átlagos válaszidejét.

Végül a 8. fejezetben a disszertációban szereplő eredmények összefoglalója található.

2 Sorbanállási rendszerek és hálózatok

Mint általában a számítógépes rendszereknél, a Web szerverek is több kérést kell kiszolgáljanak, melyek mindegyike verseng a szükséges erőforrásokért. Mivel egy időben egy kérés egyszerre csak egy erőforrást érhet el, így a többi kérésnek várakoznia kell egy pufferben. Amint egy erőforrás felszabadul, egy kérés a pufferből automatikusan átkerül a kiszolgálócsatornába, valamint minden újonnan érkező kérés a pufferbe kerül amennyiben a kiszolgáló csatorna nem szabad. A sorbanállás elmélet segítségével számíthatóak ki a rendszer jellemzői, mint a sorhossz (a rendszerben lévő igények száma), a válaszidő (egy igény rendszerben eltöltött ideje), stb., valamint a sorbanállási rendszerek és hálózatok segítségével vizsgálhatóak a bonyolultabb alkalmazási rendszerek is [21], [18], [20]. Ebben a fejezetben igyekszünk egy rövid, lényegretörő bemutatást adni a sorbanállási rendszerekről és hálózatokról.

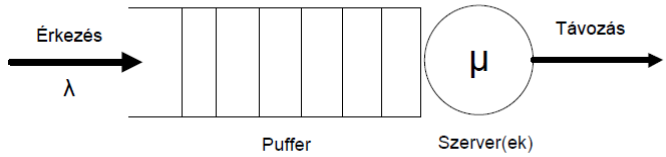
2.1. Sorbanállási rendszerek vizsgálata:

Egy sorbanállási modellt az alábbi szerkezetű kóddal jellemezhetünk:

$$A/B/m/k/n/P,$$

ahol

- ***A*** - az egymást követő igények beérkezése között eltelt idő eloszlása:
  - D - determinisztikus
  - M - exponenciális



2.1. ábra. Egyszerű sor

  - G - általános

- ***B*** - a kiszolgási idő eloszlása:
  - D - determinisztikus
  - M - exponenciális
  - G - általános
- ***m*** - a kiszolgáló egységek száma.
- ***k*** - a rendszer kapacitása. Abban az esetben ha nincs megadva akkor a kapacitást végtelennek tekintjük.
- ***k*** - az igények forrás kapacitása. Abban az esetben ha nincs megadva akkor a kapacitást végtelennek tekintjük.
- ***P*** - a kiszolgálási protokoll. Értékei lehetnek:
  - FIFO - First In Last Out
  - LIFO - Last In First Out
  - Prioritásos kiszolgálás.

  Abban az esetben ha nincs megadva az alapértelmezett a FIFO.

A 2.1-es ábrán egy egyszerű sorbanállási modell látható, ahol az érkezési intenzitás (az egységnyi idő alatt beérkezett igények átlagos száma) λ, valamint az átlagos kiszolgálási idő $\frac{1}{\mu}$. Amennyiben az érkezési intenzitás kisebb mint a kiszolgálás intenzitása ($\lambda < \mu$) azt mondjuk, hogy

a rendszer stabil, azaz minden kérés ki lesz szolgálva, valamint a rendszerben tartózkodó igények átlagos száma (N) véges. Egy kérés átlagos tartózkodási ideje (T) a Little formula (lásd [27]) alapján:

$$T = \frac{N}{\lambda}. \tag{2.1}$$

Az $M/M/1$-es sorbanállási rendszerről beszélünk, amennyiben az érkezések közötti időintervallumok független λ paraméterű exponenciális eloszlású valószínűségi változók valamint a kiszolgálási idő eloszlása is exponenciális μ paraméterrel. Ebben az esetben a rendszer paramétereket a következő képletek adják (stabil esetben):

- P_0 annak a stacionárius valószínűsége, amikor a rendszer üres

$$P_0 = \frac{1}{1 + \sum_{k=1}^{\infty} \left(\frac{\lambda}{\mu}\right)^k}, \tag{2.2}$$

$$P_0 = 1 - \frac{\lambda}{\mu}.$$

- P_k az a stacionárius valószínűség, amikor a rendszer a k állapotban van, ahol

$$P_k = P_0 \left(\frac{\lambda}{\mu}\right)^k. \tag{2.3}$$

- A rendszerben tartózkodó igények átlagos száma:

$$N = \frac{\rho}{1 - \rho}, \tag{2.4}$$

  ahol $\rho = \frac{\lambda}{\mu}$.

- A pufferben tartózkodó igények átlagos száma:

$$N_q = \frac{\rho^2}{1 - \rho}. \tag{2.5}$$

- A szerver kihasználtsága

$$U_s = 1 - P_0 = \rho. \tag{2.6}$$

- Az átlagos várakozási idő

$$W = N\frac{1}{\mu} = \frac{\rho}{\mu(1-\rho)}. \tag{2.7}$$

- A rendszerben töltött átlagos idő

$$T = \frac{1}{\mu - \lambda}. \tag{2.8}$$

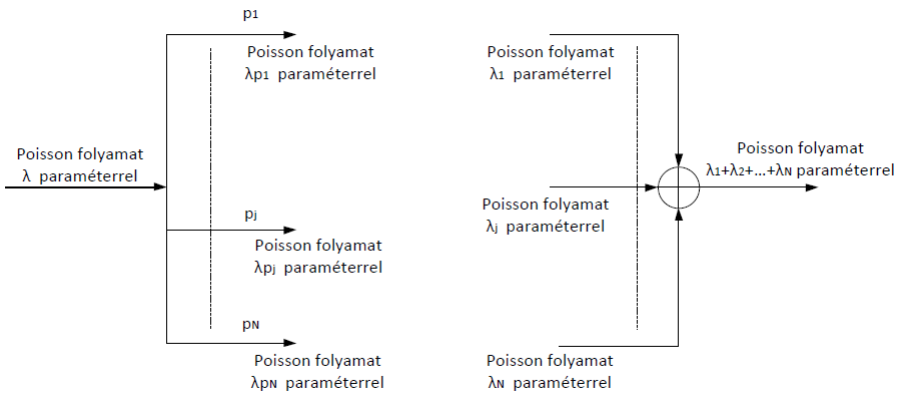
2.2. Sorbanállási hálózatok vizsgálata

Rendszerint egy egyedüli sor nem alkalmas egy bonyolultabb rendszer modellezéséhez, mint például egy Web szervert tartalmazó rendszer. Ilyen rendszereket általában hálózati modellekkel ábrázolhatunk, ahol a rendszerben lévő sorokra úgy tekintünk, mint különálló csomópontokra. Az ilyen sorbanállási hálózatokat nyitottnak nevezzük, ha az új igények a rendszeren kívülről érkeznek, és a későbbiekben el is fogják hagyni a rendszert. Egy nyitott sorbanállási hálózatot nyitott Jackson hálózatnak nevezünk, ha teljesülnek a következő feltételek:

- Az i-edik sor kiszolgálási ideje μ_i paraméterű exponenciális eloszlású valószínűségi változó
- Az i-edik sorhoz a hálózaton kívülről érkező kérések Λ_i paraméterű Poisson-folyamat szerint érkeznek
- A kiszolgálási protokol FCFS

Burke és Jackson tétele alapján (lásd [12]) a következő megállapításokat tehetjük:

- Poisson-folyamatok véletlen elágaztatásával Poisson-folyamatokat kapunk. (Lásd 2.2 grafikont.)



2.2. ábra. Folyamatok egyesítése és osztása

- Független Poisson-folyamatok egyesítése Poisson-folyamat. (Lásd 2.2 grafikont.)
- $M/M/1$-es sorbanállási rendszer esetében a távozási folyamat Poisson-folyamat, intenzitása pedig megegyezik az érkezési intenzitással.
- A rendszerben lévő különálló sorok úgy viselkednek mintha függetlenek lennének egymástól.
- A sorokhoz érkező folyamatok Poisson-folyamatként viselkednek abban az esetben is ha valójában nem azok. Ilyen eset például a direkt visszacsatolás (Lásd 2.3 ábrát), mely esetben a sorhoz érkező folyamat nem Poisson-folyamat, de a távozási folyamat Poisson.

Tekintsünk egy nyitott Jackson hálózatot, melyben K darab sor található. Az i-edik sorhoz a rendszeren kívülről érkező folyamat legyen Poisson-folyamat Λ_i paraméterrel. A $\Lambda_i = 0$ intenzitás azt jelenti, hogy az i-edik sorhoz nem érkezik kívülről igény. Mivel egy nyitott rendszerről beszélünk, feltételezzük, hogy legalább egy $\Lambda_j > 0$. A i-edik csomópont által kiszolgált igény a j-edik csomóponthoz továbbítódik $p_{i,j}$ valószínűséggel, vagy elhagyja a rendszert $\left(1 - \sum_{j=1}^{K} p_{ij}\right)$ valószínűséggel. A j-edik sor

kiszolgálási ideje μ_j paraméterű exponenciális eloszlás, valamint a sorhoz érkező folyamat teljes érkezési intenzitása λ_j, ahol

$$\lambda_j = \Lambda_j + \sum_{i=1}^{K} \lambda_i p_{ij} \tag{2.9}$$

minden $j = 1, 2, \cdots, K$ esetén.

Rendszerparaméterek:

- *Teljes áteresztőképesség:*

$$\lambda = \sum_{i=1}^{K} \Lambda_i. \tag{2.10}$$

- *Átlagos forgalmi intenzitás a j-edik csomópontnál:*

$$\rho_j = \frac{\lambda_j}{\mu_j}. \tag{2.11}$$

- *Egy igény látogatásainak száma a j-edik csomópontnál:*

$$V_j = \frac{\lambda_j}{\lambda} = \frac{\Lambda_j}{\lambda} + \sum_{i=1}^{K} V_i p_{ij}. \tag{2.12}$$

- *Az átlagos igényszám a j-edik csomópontnál:*

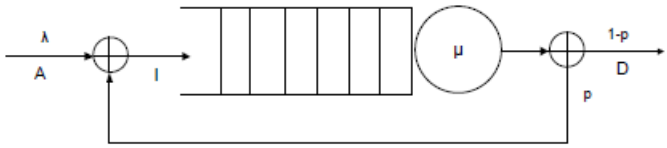
$$N_j = \frac{\rho_j}{1 - \rho_j}. \tag{2.13}$$

- *Az átlagos válaszidő a j-edik csomópontnál:*

$$T_j = V_j * \frac{N_j}{\lambda_j}, \tag{2.14}$$

azaz

$$T_j = \frac{N_j}{\lambda} = \frac{1}{\lambda} \frac{\rho_j}{1 - \rho_j}. \tag{2.15}$$



2.3. ábra. Direkt visszacsatolás

- *Az átlagos igényszám a rendszerben:*

$$N = \sum_{j=1}^{K} N_j = \sum_{j=1}^{K} \frac{\rho_j}{1-\rho_j}. \tag{2.16}$$

- *Az átlagos válaszidő a rendszerben:*

$$T = \frac{N}{\lambda} = \frac{1}{\lambda} \sum_{j=1}^{K} \frac{\rho_j}{1-\rho_j} = \sum_{j=1}^{K} \frac{B_j}{1-\lambda B_j}, \tag{2.17}$$

ahol $B_j = \frac{V_j}{\mu_j}$ egy igény teljes átlagos feldolgozási ideje a j-edik csomópontnál.

3 Web szerver modell

Ebben a fejezetben bemutatjuk, hogyan lehet modellezni egy egyszerű Web szervert nyitott sorbanállási hálózatok segítségével. (Lásd [32].) Egy Web szerver működési modellje látható a 3.1 ábrán. A modell részletesebb ismertetése előtt sorbavesszük a válaszidőt meghatározó főbb paramétereket:

- *Érkezési intenzitás* (λ): A Web szerverhez érkező HTTP fájl kérések átlagos száma másodpercenként.
- Az *átlagos fájl méret* (F): A Web szerver által kiszolgált fájlok átlagos mérete byteban megadva. Ez az érték természetesen nagyban függ a Web szervertől is.
- *Puffer méret* (B): A Web szerverhez tartozó puffer méret, mely a szerver diszk blokk méretével egyezik meg. [32] alapján az átlagos puffer méretet 2000 byte.
- Inicializációs idő (I), Statikus szerver idő (Y) valamint a Dinamikus szerver arány (R) együttesen a Web szerver kiszolgálási intenzitását határozzák meg. Az I ábrázolja a Web szerverhez való kapcsolódáshoz szükséges összes egyszeri inicializációs időt, mint például a TCP kapcsolat kiépítése, vagy a "suffix mapping". Ennek a csomópontnak a kiszolgálási intenzitása $\frac{1}{I}$. Y reprezentálja a puffer feldolgozásával töltött időt, mely független a puffer méretétől. R pedig a byte/másodpercben megadott arányt jelenti, mellyel a Web szerver a puffer tartalmát továbbítja. A Web szerver átlagos kiszolgálási intenzitása ezek alapján:

$$\mu_{Web} = \frac{1}{Y + \frac{B}{R}}. \tag{3.1}$$

A Web szerver kiszolgálási intenzitását meghatározó paraméterek részletes leírása megtalálható a [32]-ben.

- Kliens hálózati sávszélesség: N_c, Szerver hálózati sávszélesség: N_s.

Amint látható a 3. ábrán a modell 4 csomópontot tartalmaz. Két csomópont ábrázolja a Web szerver architektúráját, valamint 2 csomópont ábrázolja a kliens és szerver oldali hálózati kommunikációs időt (File transfer time). Gyakran az egyszerű hálózati kommunikációt is sorbanállási modellel ábrázolják, de jelen esetben megelégszünk egyszerűen konstans transzfer idők használatával ([32], [8]). Feltételezzük, hogy az igények λ paraméterű-Poisson-folyamat alapján érkeznek a kliensek felől. Mielőtt az igények megérkeznének a Web szerverhez, először át kell essenek egy egyszeri inicializációs eljáráson, amit az első csomópont szemléltet. Az inicializációs időt a

$$T_{Init} = \frac{1}{\frac{1}{I} - \lambda} \tag{3.2}$$

képlet szolgáltatja. Ha a felhasznló által kért fájl mérete nagyobb, mint a szerver kimenő puffere, akkor egy visszacsatolási ciklus kezdődik, mely addig tart, míg az igény kiszolgálása be nem fejeződik. Legyen q annak a valószínűsége, hogy a fájlt első próbálkozásra sikerül feldolgozni, ahol

$$q = \min\left(1, \frac{B}{F}\right). \tag{3.3}$$

Ezek alapján a Web szerverhez érkező igények intenzitása, figyelembe véve az esetleges visszacsatolást:

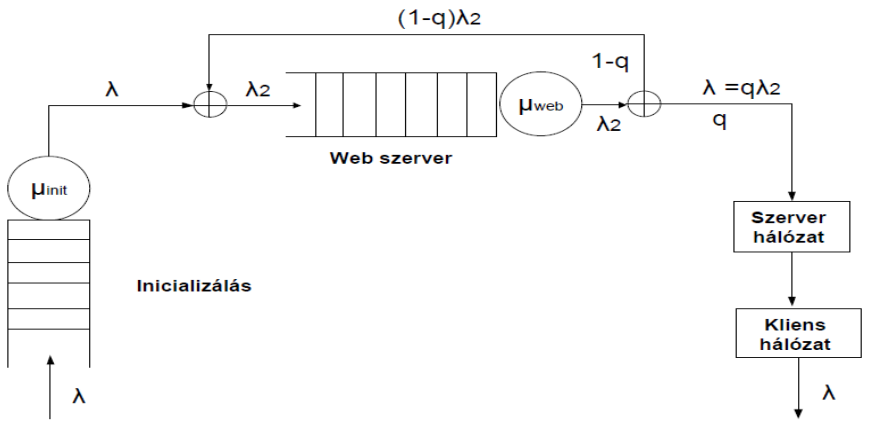
$$\lambda_2 = \frac{\lambda}{q}. \tag{3.4}$$

A korábbiak alapján a Web szerver kiszolgálási intenzitása:

$$\mu_{Web} = \frac{1}{Y + \frac{B}{R}}, \tag{3.5}$$

így a Web szerver kiszolgálási ideje (2.15) alapján:

$$T_{Web} = \frac{1}{\lambda} \cdot \frac{\lambda_2}{\mu_{Web} - \lambda_2}. \tag{3.6}$$



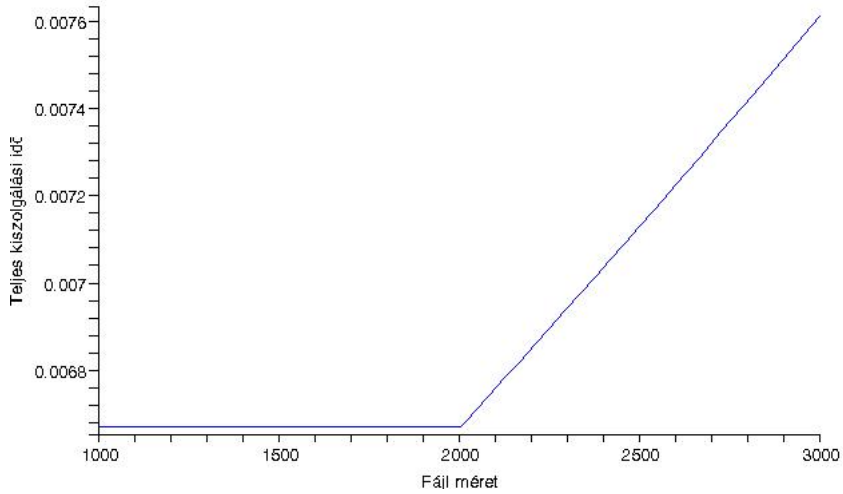
3.1. ábra. Web szerver modell

A teljes válszidőhöz (T) a modell alapján még hozzá kell vegyük azt a transzfer időt amíg az igény átjut a szerver és a kliens hálózatán. Ezek alapján válaszidőt az alábbi képlet adja:

$$\begin{aligned} T &= \frac{1}{\frac{1}{I} - \lambda} + \frac{1}{\lambda} * \frac{\lambda_2}{\mu_{Web} - \lambda_2} + \frac{F}{N_s} + \frac{F}{N_c} \\ &= \frac{I}{1 - \lambda * I} + \frac{YR + B}{qR - \lambda(YR + B)} + \frac{F}{N_s} + \frac{F}{N_c}. \end{aligned} \tag{3.7}$$

A következő néhány grafikonon megvizsgáljuk, hogyan változik a Web szerver válaszideje a szerver sebességének valamint a fájl méretének változtatásával. Mivel jelen esetben minket csak a Web szerverrel kapcsolatos válaszidők érdekelnek a 3.7 képletet annyiban módosítjuk, hogy az áthaladási időt a kliens és a szerver hálózatokon nem vesszük figyelembe. Így a

$$T_{Init+Web} = \frac{I}{1 - \lambda * I} + \frac{YR + B}{qR - \lambda(YR + B)} \tag{3.8}$$

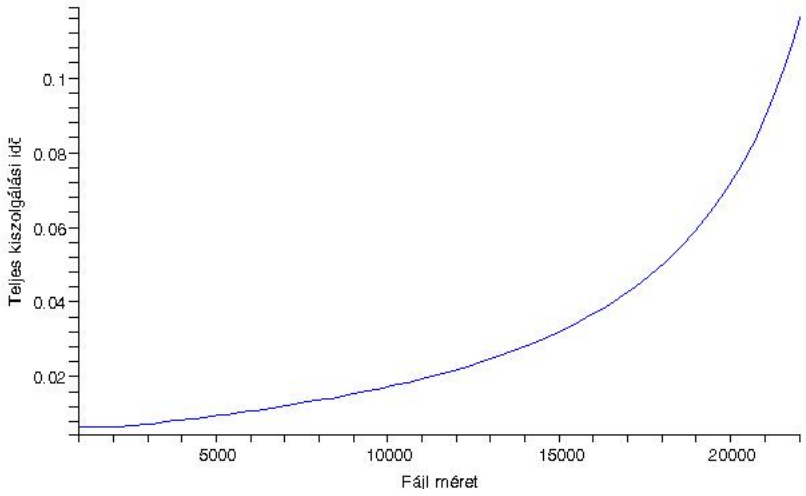


3.2. ábra. Fájl méret hatása a Web szerver válaszidejére

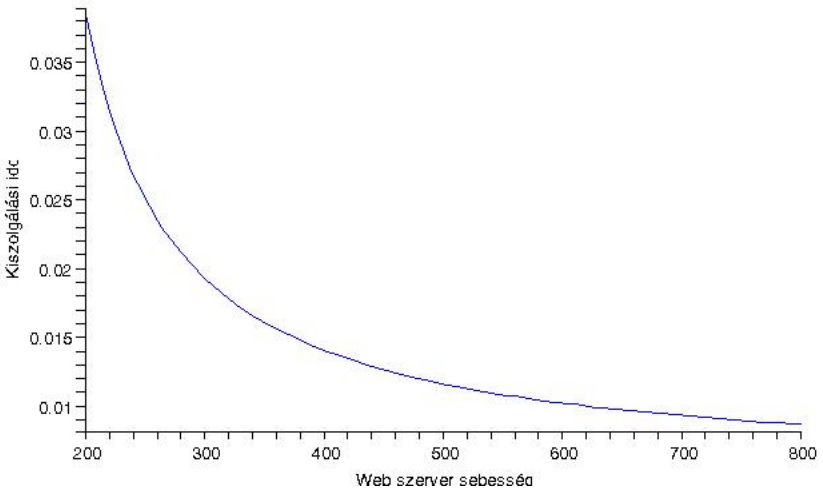
képlettel vizsgálni tudjuk a hálózati transzfer időktől megtisztított válaszidőt.

A 3.2 grafikonon megfigyelhetjük, milyen hatással van a fájl méretének növelése a válaszidőre. A számításban használt paraméterek értékei a 3.1. táblázatban láthatóak.

A (3.3) képlet alapján láthatjuk, hogy nem történik visszacsatolás a Web szerveren, amíg a fájl mérete el nem éri a puffer méretét. Tehát a válaszidő konstans egészen addig, míg el nem éri a fájl a puffer méretet. Ezt figyelhetjük meg a 3.2 grafikonon is. Amint látható, ahogy a fájl mérete meghaladja a puffer méretét, elkezdődik a Web szervernél a visszacsatolás, azaz a tartózkodási idő növekedésnek indul. A 3.3 grafikonon a válaszidőt szintén a fájl méret függvényeként ábrázoltuk. Láthatjuk, hogy a válaszidő a fájl méretének növelésével szintén növekszik. A 3.4 grafikonon a Web szerver sebességének hatását figyelhetjük meg a válaszidőre. Amint azt várni lehetett, a szerver sebességének növelésével a válaszidő csökken.



3.3. ábra. Fájl méret hatása a Web szerver válaszidejére 2



3.4. ábra. A Web szerver sebességének hatása a válaszidőre

3.1. táblázat. **Web szerver modell paraméterei**

λ:	Az érkezési intenzitás	50 kérés/másodperc
F:	Az átlagos fájl méret (byte-ban)	változó
B:	A Web szerver kimenő puffere (byte-ban)	2000 byte
I:	Inicializálási idő (másodperc)	0.004 másodperc
Y:	Statikus szerver idő (másodperc)	0.000016 másodperc
R:	Dinamikus server arány (byte/másodperc)	1310720 byte/másodperc

4 Proxy Cache szerver modell

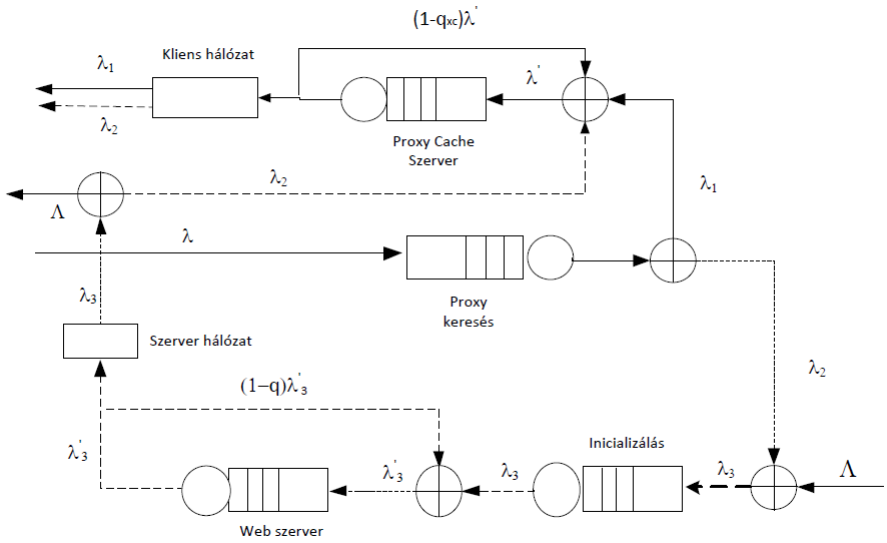
Ebben a fejezetben részletesen bemutatjuk egy Proxy Cache szerver matematikai modelljét és annak működését.

A fejezetben szereplő eredmények a [9] cikkünkben szereplő modell általánosítása.

4.1. A javasolt modell

Proxy Cache szervert használva, ha egy fájlt le akarunk tölteni egy távoli Web szerverről, először meg kell vizsgálni, hogy a keresett fájl megtalálható-e a Proxy Cache szerveren. Ennek a valószínűségét p-vel jelöljük. Amennyiben a keresett dokumentum megtalálható a Proxy Cache szerveren, egy másolat a fájlról azonnal továbbítódik a felhasználónak. Amennyiben a dokumentum nem található meg a Proxy Cache szerveren, az igény továbbítódik a távoli Web szerverhez. A dokumentum a Web szerverről először a Proxy Cache szerverre érkezik vissza, ahonnan egy másolat a fájlról azonnal a felhasználóhoz kerül. Az eredeti példány tárolódik a Proxy Cache szerveren, így a későbbiekben elérhető lesz a felhasználók számára. A Proxy Cache szerver hatékonysága a következő tényezőktől függ:

- találati arány (a kért dokumentum milyen valószínűséggel található meg a Proxy Cache szerveren)
- Proxy Cache szerver karakterisztikája - sebessége
- kliens oldali sávszélesség
- szerver oldali sávszélesség



4.1. ábra. Proxy Cache szerver modellje

- a belső és külső igények érkezési intenzitása
- Web szerver karakterisztikája - sebessége

A Proxy Cache szerver matematikai modelljének megalkotásához a [32] és [11] modelljeit vettük alapul. A [11]-ben szereplő Proxy Cache szerver modellben a Web szerver és a Proxy Cache szerver ugyanannak a zárt részhálózatnak a csomópontjaiként voltak ábrázolva, mely nem tekinthető megfelelőnek mivel ebben az esetben a Web szerver leterheltségét csak a Proxy Cache szerveren nem található igények határozzák meg, valamint a Proxy Cache szerver ábrázolása is hiányos, mivel a keresési időt és a Proxy Cache szervert egy csomópontként jelölte. Ezen kívül a Proxy Cache szervert ábrázoló csomópontot a Web szerverekre jellemző visszacsatolás nélkül ábrázolta, mely szintén torzítja a kapott eredményeket. A [8] cikkünkben részletesen bemutattuk a [11]-ben szereplő modell hiányosságait.

Az általunk általánosított modell a 4.1 ábrán látható. A 4.1 ábra egy igény lehetséges útját mutatja a felhasználótól kiindulva egészen a visszaérkezésig.

A modellben szereplő paraméterek listája a 4.3 táblázatban látható. A modell alapján a távoli Web szerverhez két irányból érkezhetnek igények. Egyfelelől a Proxy Cache szerver irányából, valamint a Proxy Cache szervertől teljesen függetlenül érkezhetnek az úgynevezett külső igények, melyek természetesen nem mennek keresztül a Proxy Cache szerveren. Feltételezzük, hogy az igények a Proxy Cache szerverhez λ paraméterű Poisson-folyamat szerint érkeznek, valamint a Web szerverhez kívülről érkező igények Λ paraméterű Poisson-folyamat alapján érkeznek.

Az egyenes vonal (λ_1) reprezentálja azt az esetet, amikor a keresett fájl megtalálható a Proxy Cache szerveren. Szaggatott vonallal rajzolva jelöltük (λ_2) azon igények útját, melyek nem találhatóak meg a Proxy Cache szerveren, így ezek az igények továbbítódnak a távoli Web szerverhez. λ_1 és λ_2 ezen igények intenzitásai:

$$\lambda_1 = p * \lambda, \tag{4.1}$$

$$\lambda_2 = (1 - p) * \lambda, \tag{4.2}$$

ahol p a találati valószínűség.

Ahogy már korábban jeleztük, a Web szerver nem csak a Proxy Cache szerver felől érkező igényeket szolgálja ki, hanem az úgynevezett külső igényeket is, melyek érkezési intenzitása Λ. λ_3 jelöli a Web szerverhez érkező összes kérés érkezési intenzitását, mely

$$\lambda_3 = \lambda_2 + \Lambda. \tag{4.3}$$

Ahogy a 3. fejezetben ismertettük, a Web szerverhez érkező igényeknek először át kell esniük egy egyszeri inicializáción. Jelen modell alapján ez az inicializálás a λ_3 intenzitású folyamaton megy végbe, és az inicializálási idő:

$$\frac{1}{\frac{1}{I_s} - \lambda_3}. \tag{4.4}$$

A Web szerver és a Proxy Cache szerver hatékonyságát három - három paraméterrel írhatjuk le (lásd a 3. fejezetet). Ezek rendre a B_s, Y_s, R_s illetve B_{xc}, Y_{xc}, R_{xc}. Ha a kért fájl mérete nagyobb mint a Web szerver kimenő pufferének mérete, akkor egy visszacsatolási ciklus kezdődik, mely addig tart míg a fájl feldolgozása teljes egészében be nem fejeződik. Legyen

$$q = \min\left(1, \frac{B_s}{F}\right) \tag{4.5}$$

annak a valószínűsége, hogy a fájl továbbítása elsőre sikerült. Legyen $\lambda_3^{'}$ a Web szerverhez érkező igények intenzitása figyelembe véve a visszacsatolási ciklust. Így

$$\lambda_3^{'} = \frac{\lambda_3}{q}. \tag{4.6}$$

Hasonlóan a Proxy Cache szerver esetén legyen

$$q_{xc} = \min\left(1, \frac{B_{xc}}{F}\right) \tag{4.7}$$

annak a valószínűsége, hogy a keresett fájl továbbítása a Proxy Cache szervernél elsőre sikerült, valamint legyen $\lambda^{'}$ a Proxy Cache szerverhez érkező igények intenzitása, ahol

$$\lambda^{'} = \lambda_1 + \lambda_2 + (1 - q_{xc}) * \lambda^{'} \tag{4.8}$$

azaz

$$\lambda^{'} = \frac{\lambda}{q_{xc}}. \tag{4.9}$$

A fenti eredményeket felhasználva kapjuk T_{xc}-t, egy belső igény teljes válaszidejét Proxy Cache szerver használatával, ahol

$$
\begin{aligned}
T_{xc} \;=\; & \frac{1}{\frac{1}{I_{xc}}-\lambda} + p\left\{ \frac{\frac{F}{B_{xc}}}{\frac{1}{\left(Y_{xc}+\frac{B_{xc}}{R_{xc}}\right)}-\frac{\lambda}{q_{xc}}} + \frac{F}{N_c} \right\} \\
& + (1-p)\left\{ \frac{1}{\frac{1}{I_s}-\lambda_3} + \frac{\frac{F}{B_s}}{\frac{1}{\left(Y_s+\frac{B_s}{R_s}\right)}-\frac{\lambda_3}{q}} \right. \\
& \left. + \frac{F}{N_s} + \frac{\frac{F}{B_{xc}}}{\frac{1}{\left(Y_{xc}+\frac{B_{xc}}{R_{xc}}\right)}-\frac{\lambda}{q_{xc}}} + \frac{F}{N_c} \right\}
\end{aligned}
\tag{4.10}
$$

valamint 3.7 alapján legyen T egy igény válaszideje Proxy Cache szerver használata nélkül, ahol

$$
T = \frac{1}{\frac{1}{I_s}-(\lambda+\Lambda)} + \frac{\frac{F}{B_s}}{\frac{1}{\left(Y_s+\frac{B_s}{R_s}\right)}-(\lambda+\Lambda)/q} + \frac{F}{N_s} + \frac{F}{N_c}. \tag{4.11}
$$

A T_{xc} válaszidő három részből tevődik össze: az első annak az időtartama, míg eldől, hogy az igényelt fájl megtalálható-e a Proxy Cache szerveren vagy nem. Ez a (2.8) képletből adódik, ahol I_{xc} az átlagos kiszolgálási idő. A képlet második tagja annak a válaszideje, amikor az igény megtalálható a Proxy Cache szerveren, melynek kiszolgálási intenzitása (3.5) alapján $\frac{1}{Y_{xc}+\frac{B_{xc}}{R_{xc}}}$ valamint $\frac{F}{N_c}$ az "utazási" idő amíg a keresett fájl keresztüljut a kliens hálózaton. A képlet harmadik tagja reprezentálja annak az igénynek a válaszidejét, mely nem található meg a Proxy Cache szerveren. Ez további öt részre tagolódik. Az első a 3.2 alapján az egyszeri inicilaizálási idő, a második a 2.15 alapján a Web szerver kiszolgálóegységénél töltött idő. A harmadik a Web szerver hálózatán való áthaladáshoz szükséges Web szerver oldali transzfer idő. A negyedik rész a fentiekhez hasonlóan a Proxy Cache szerverhez visszaérkező igények kliens felé való továbbításának időtartamát ábrázolja. Az utolsó ötödik rész pedig a kliens hálózat transzfer idejét ábrázolja.

A rendszer akkor lesz stabil, ha a rendszerben minden sorbanállási csomópont stabil. A 4.1 modellben 4 sorbanállási csomópont található, melyek mindegyikére teljesülnie kell a következő feltételeknek:

- Keresési csomópont:
  $\rho_{Lookup} < 1$, ahol $\rho_{Lookup} = \lambda * I_{xc}$, azaz $\lambda < \frac{1}{I_{xc}}$.
- Proxy Cache szerver:
  $\rho_{PCS} < 1$, ahol $\rho_{PCS} = \frac{\lambda^{'}}{\mu_{PCS}}$, azaz $\lambda < \frac{R_{xc}q_{xc}}{Y_{xc}R_{xc}+B_{xc}}$.
- Egyszeri inicializálási csomópont:
  $\rho_{Init} < 1$, ahol $\rho_{Init} = \lambda_3 * I_s$, azaz $\lambda < \frac{1}{(1-p)I_s} - \frac{\Lambda}{1-p}$.
- Web szerver:
  $\rho_{Web} < 1$, ahol $\rho_{Web} = \frac{\lambda_3^{'}}{\mu_{Web}}$, azaz $\lambda < \frac{R_s q}{(1-p)(Y_s R_s+B_s)} - \frac{\Lambda}{(1-p)}$.

Tehát a rendszer stabil, ha

$$\lambda < \min\left(\frac{1}{I_{xc}}, \frac{R_{xc}q_{xc}}{Y_{xc}R_{xc}+B_{xc}}, \frac{1}{(1-p)I_s} - \frac{\Lambda}{1-p}, \frac{R_s q}{(1-p)(Y_s R_s+B_s)} - \frac{\Lambda}{(1-p)}\right). \tag{4.12}$$

4.2. Numerikus eredmények

A numerikus eredmények a MAPLE 9 program segítségével lettek kiszámolva. A számításokhoz a Web és Proxy Cache szerver paraméterek értékeit [28] alapján határoztuk meg. Ezek az értékek: $I_s = I_{xc} = 0.004$ másodperc, $B_s = B_{xc} = 2000$ byte, $Y_s = Y_{xc} = 0.000016$ másodperc, $R_s = R_{xc} = 1.25$ Mbyte/másodperc, $N_s = 1544$ Kbit/másodperc valamint $N_c = 128$ Kbit/másodperc. A fejezetben található grafikonokban pontozott vonallal ábrázoltuk a Proxy Cache szervert tartalmazó esetet, valamint egyenes vonallal a Proxy Cache szervert nem tartalmazó esetet.

4.2.1. Az érkezési intenzitás hatása a válaszidőre

A 4.2 ábra a teljes válaszidőt ábrázolja a Proxy Cache szerver felől érkező igények intenzitás függvényeként. Ezen a grafikonon a külső igények érkezési intenzitása $\Lambda = 100$ igény/másodperc, valamint a találati valószínűség $p = 0.1$. Mint ahogyan látható, Proxy Cache szerver installálása csak igen magas belső érkezési intenzitás ($\lambda > 100$ igény/másodperc) esetén éri meg.

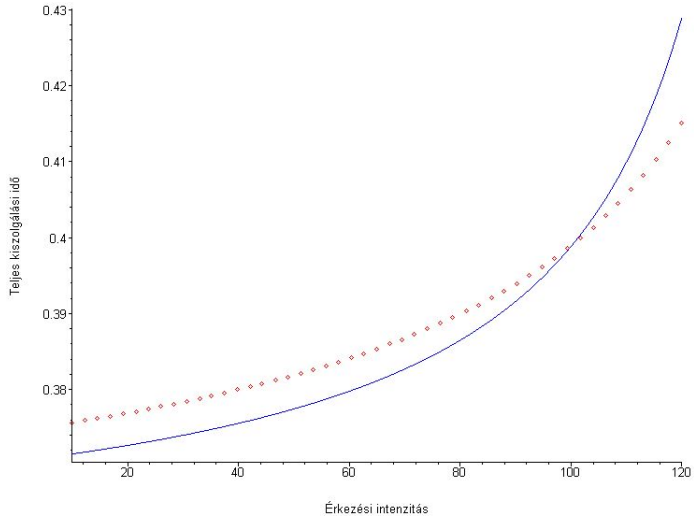
A 4.3 grafikon esetében ugyanazokat a paramétereket használtuk, egyedül a találati valószínűséget növeltük $p = 0.15$-re. Megfigyelhetjük, hogy ilyen paraméterek esetében a Proxy Cache szerver használatával a válaszidők már $\lambda > 80$ igény/másodperc esetén alacsonyabbak, mint Proxy Cache szerver használata nélkül.

A 4.4 grafikonon azt az esetet vizsgáltuk meg, amikor alacsony találati valószínűség mellett $p = 0.1$, egy relatíve leterhelt Web szervertől kérünk le adatot. (A külső kérések intenzitása $\Lambda = 150$ kérés/másodperc). Mint ahogyan várható volt, terhelt Web szerver esetén a Proxy Cache szerver üzemeltetése már kis belső igény intenzitás esetén is megtérül. ($\lambda > 50$).

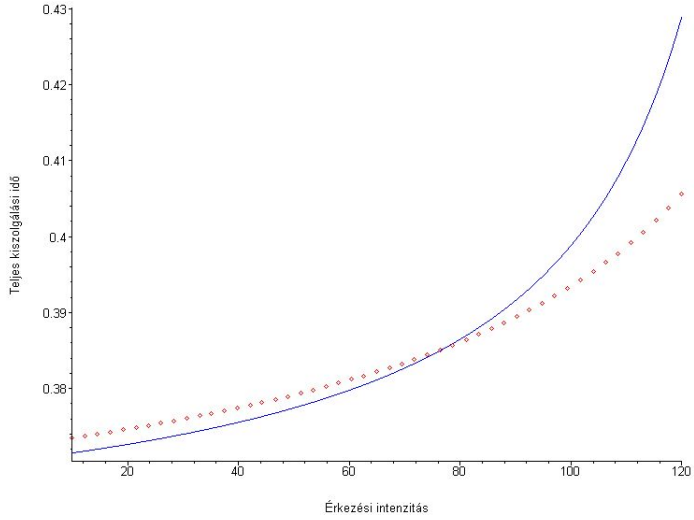
Amennyiben pedig a belső keresési szokások hasonlóak, azaz a találati valószínűség magasabb és a Web szerver is leterhelt ($p = 0.15$ és $\Lambda = 150$), a Proxy Cache szerver használata már $\lambda > 20$ intenzitás esetén is alacsonyabb válaszidőket eredményez, mint ahogyan a 4.5 grafikonon látható. A 4.2, 4.3, 4.4 valamint 4.5 ábrák tanulmányozása alapján megállapíthatjuk, hogy a Proxy Cache szerver használata alacsonyabb válaszidőket eredményez, amennyiben magas $p > 0.15$ találati valószínűséget vagy leterhelt Web szervert használunk.

4.2.2. A külső érkezési intenzitás hatása a válaszidőre

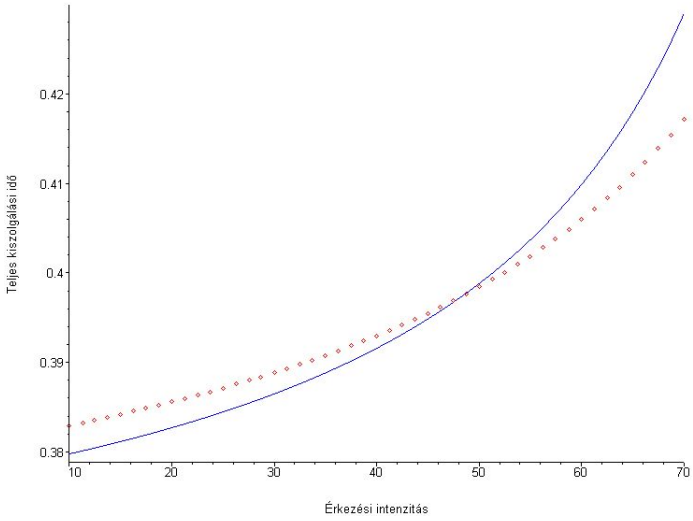
A következő részben megvizsgáljuk, hogyan hat a Web szerverhez érkező külső igények intenzitása a belső igények teljes válaszidejére. A 4.6 grafikonon a Proxy Cache szerver felől érkező "belső" igények intenzitása viszonylag alacsony $\lambda = 20$ kérés/másodperc, valamint a találati valószínűség is alacsony $p = 0.1$. Mint ahogyan látjuk a válaszidő még $\Lambda = 170$ kérés/másodperc külső érkezési intenzitás mellett is nagyobb Proxy Cache szerver használatával mint anélkül. A 4.7 grafikonon az előző esethez képest csak a belső érkezési intenzitást növeltük $\lambda = 70$ kérés/másodpercre, a találati valószínűség maradt változatlan ($p = 0.1$). Mint ahogy láthatjuk, amennyiben a Web szerver leterheltsége alacsonyabb ($\Lambda < 130$) a válaszidők Proxy Cache szerver használatával nagyobbak mint nélküle. Viszont ha a külső igények érkezési intenzitása magas ($\Lambda > 130$) akkor a válaszidő alacsonyabb Proxy Cache szerver használatával. A 4.8 grafikonon a találati valószínűség megduplázása ($p = 0.2$) mellett vizsgáltuk a



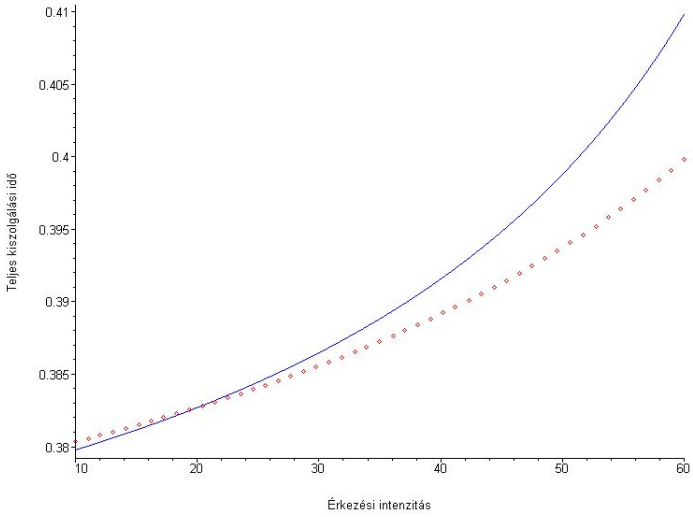
4.2. ábra. $p = 0.1$, $F = 5275$ byte, $\Lambda = 100$



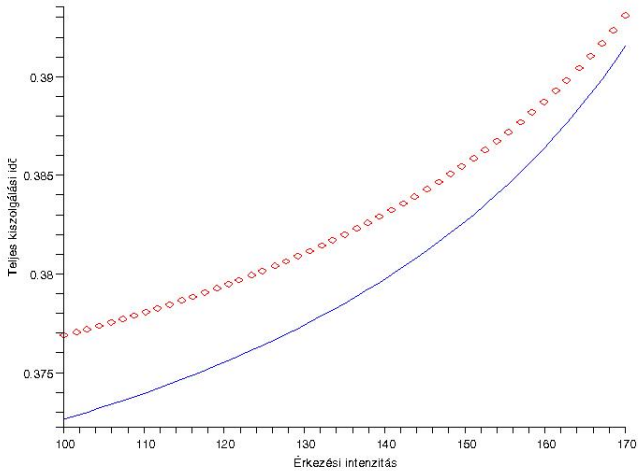
4.3. ábra. $p = 0.15$, $F = 5275$ byte, $\Lambda = 100$



4.4. ábra. $p = 0.1$, $F = 5275$ byte, $\Lambda = 150$



4.5. ábra. $p = 0.15$, $F = 5275$ byte, $\Lambda = 150$

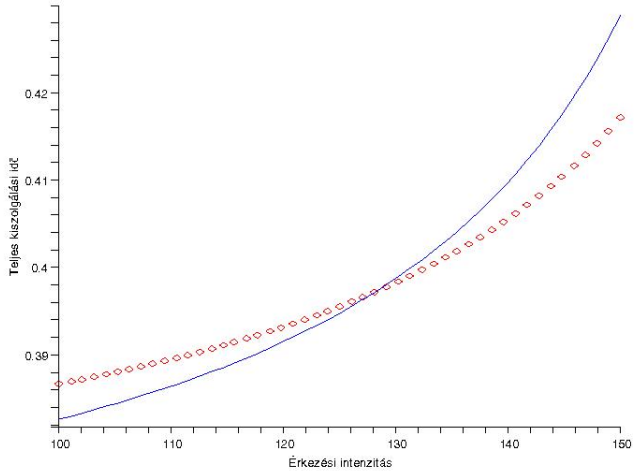


4.6. ábra. $p = 0.1$, $F = 5275$ byte, $\lambda = 20$

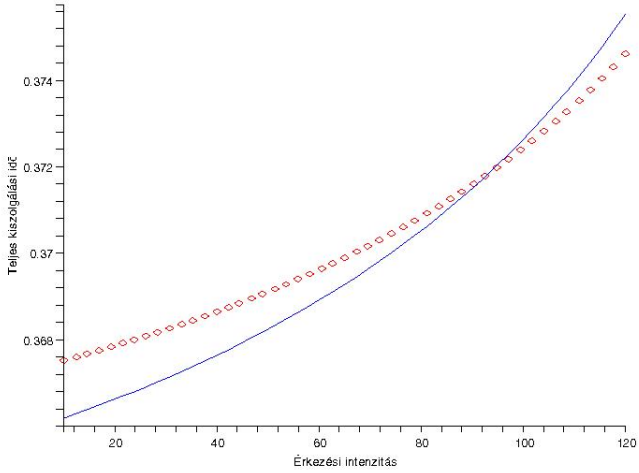
válaszidőket alacsony belső érkezési intenzitás mellett ($\lambda = 20$). Ebben az esetben azt láthatjuk, hogy a válaszidők alacsonyabbak Proxy Cache szerver használatával, ha $\Lambda > 100$. A 4.9 garfikonon azt láthatjuk, hogy magas találati valószínűség használata mellett ($p = 0.3$) a válaszidők már minden egyéb paramétertől függetlenül alacsonyabbak Proxy Cache szerver használatával.

4.2.3. A fájl méretének hatása a válaszidőre

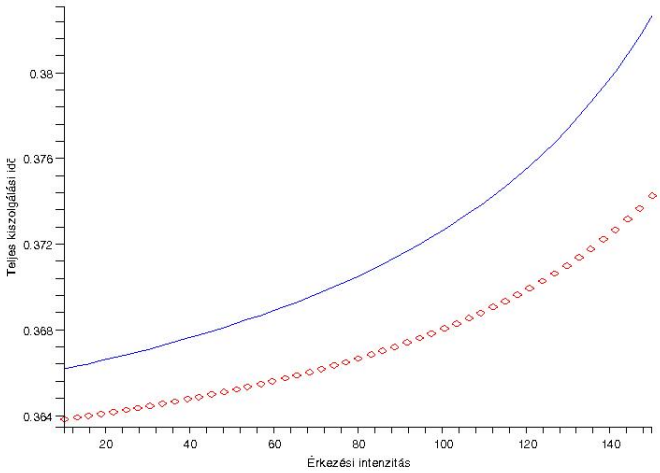
Ebben a fejezetben arra kerestük a választ, hogyan befolyásolja a fájl mérete a Proxy Cache szerver hatékonyságát. Mint ahogyan a 4.10 képletből látszik a fájl méret az úgynevezett "transzfer" időn kívűl a visszacsatolási valószínűség mértékét befolyásolja, mely mind a Web szervert mind a Proxy Cache szervert érinti. A Proxy Cache szerver nélküli eset 4.11 képletben szintén szerepel mind a kliens mind a szerver oldali "transzfer" idő, valamint a Web szerver esetében a visszacsatolási valószínűség. Proxy Cache szerver használatakor, ha a keresett fájl megtalálható a Proxy



4.7. ábra. $p = 0.1$, $F = 5275$ byte, $\lambda = 70$



4.8. ábra. $p = 0.2$, $F = 5275$ byte, $\lambda = 20$



4.9. ábra. $p = 0.3$, $F = 5275$ byte, $\lambda = 20$

szerveren az igény nem kell keresztülhaladjon a szerver hálózatán, valamint a Web szerveren, csak a kliens hálózaton és a Proxy Cache szerveren. Ezek alapján azt várjuk, hogy a fájl méretének növelésével a Proxy Cache szerver használata egyre kifizetődőbb lesz. A 4.1 táblázatban a válaszidőket figyelhetjük meg a fájl méretének függvényében ahol a használt paraméterek: $\lambda = 70$, $\Lambda = 50$, $p = 0.2$. Amint láthatjuk kis fájlméret esetén Proxy Cache szerverrel nagyobb válaszidőket kapunk, mely a fájl méretének növelésével egyre jobban közelíti a Proxy Cache szerver nélküli időket, mígnem 7000 byte feletti fájlméret esetén a Proxy Cache szerver használatával már alacsonyabb válaszidőket kapunk.

4.2.4. A találati valószínűség hatása a válaszidőre

A 4.10 grafikonon láthatjuk a találati valószínűség hatását. A teljes válaszidő Proxy Cache szerver használata nélkül független a találati valószínűségtől. Ezt a grafikonon a konstans $T = .3726542708$ vonal ábrázolja. A grafikont vizsgálva láthatjuk, hogy a találati valószínűség növelésével a

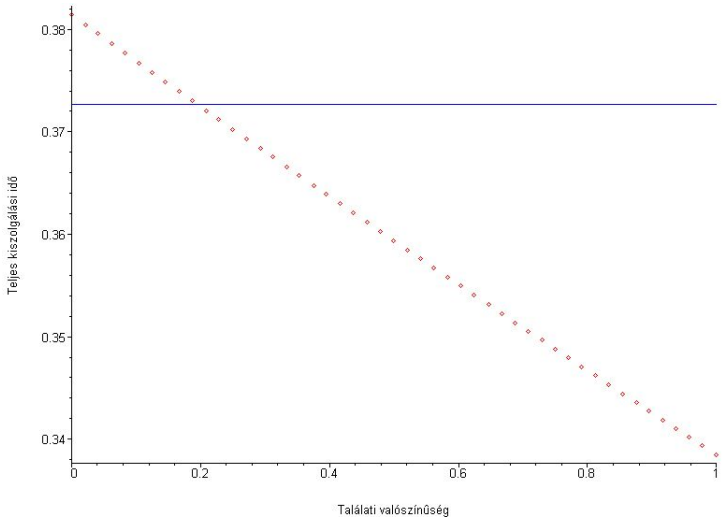
Fájl méret	Válaszidő Proxyval	Proxy nélkül	Különbség
1000	0.0793576509	0.0763196223	0.0030380285
2000	0.1476041906	0.1449469371	0.0026572535
3000	0.2162571163	0.2139376511	0.0023194652
4000	0.2852891224	0.2833129458	0.0019761766
5000	0.3548299856	0.3532714229	0.0015585627
6000	0.4250825771	0.4241768649	0.0009057122
7000	0.4963874129	0.4967750462	-0.0003876333
8000	0.5693749309	0.5728735971	-0.0034986662
9000	0.6453916529	0.6582764074	-0.0128847545
10000	0.7280473497	0.7874725350	-0.0594251853

4.1. táblázat. **A fájl méretének hatása a válaszidőre**

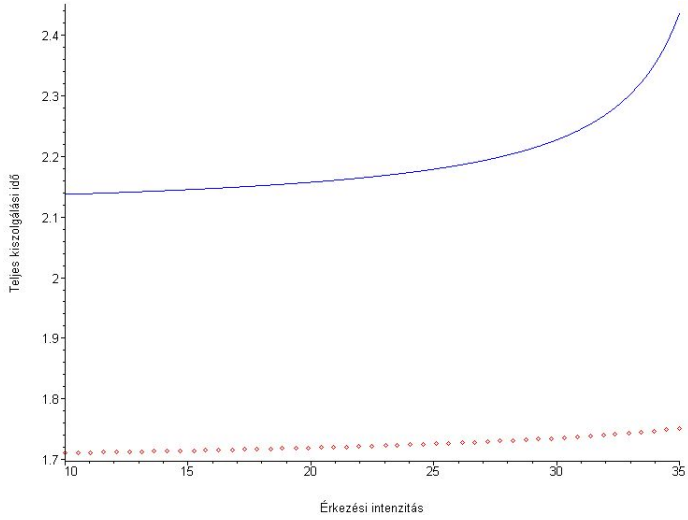
válaszidő Proxy Cache szerver használatával egyre jobban csökken. $p \approx 0.2$ értéknél a két válaszidő megegyezik, míg $p > 0.2$ esetén Proxy Cache szerver használatával alacsonyabb válaszidőket kapunk.

A bemutatott numerikus eredményekhez használt konstans adatok a [28] alapján lettek kiválasztva. Az említett konstansokkal rendelkező rendszer a mai hálózatokat tekintve nem tekinthető megfelelőnek, ezért a paramétereket megváltoztattuk. A módosított konstans értékek: $I_s = I_{xc} = 0.004$ másodperc, $B_s = B_{xc} = 4000$ byte, $Y_s = Y_{xc} = 0.0000016$ másodperc, $R_s = R_{xc} = 50$ Mbyte/másodperc, $N_s = 5$ Mbit/másodperc valamint $N_c = 20$ Mbit/másodperc.

A 4.11 grafikonon láthatjuk, hogy a módosított paraméterek mellett a belső érkezési intenzitásának hatását a válaszidőre. Ha a fájl méretet 1 Mbyte-ra állítjuk, akkor a válaszidők Proxy Cache szervert használva lényegesen alacsonyabbak, mint Proxy Cache szerver használata nélkül. A nagyobb eltérés abból adódik, hogy a kliens oldali sávszélesség jóval nagyobb mint a szerver oldali, így a Proxy Cache szerver használata lényegesen jobb eredményt szolgáltat. A 4.2 táblázatban a válaszidőket figyelhetjük meg a fájl méretének függvényében ahol a használt paraméterek: $\lambda = 20$, $\Lambda = 10$, $p = 0.2$. Amint láthatjuk kis fájlméret esetén Proxy Cache szerverrel nagyobb válaszidőket kapunk. Amikor a fájlméret eléri a 10 Kbyte-ot alacsonyabb válaszidőt kapunk Proxy Cache szerver használatával,



4.10. ábra. $\lambda = 20$, $\Lambda = 100$, $F = 5275$ byte



4.11. ábra. $p = 0.25$, $F = 1$ Mbyte, $\Lambda = 10$

Fájl méret	Válaszidő Proxyval	Proxy nélkül	Különbség
1000 byte	0.009296574550	0.006564973644	0.002731600906
5000 byte	0.01575827084	0.01464310722	0.00111516362
10 Kbyte	0.02422346863	0.02522606314	-0.00100259451
100 Kbyte	0.1731753428	0.2114664212	-0.0382910784
0.5 Mbyte	0.8572366484	1.067839041	-0.2106023926
1 Mbyte	1.717521088	2.154404036	-0.436882948
1.5 Mbyte	2.623580309	3.527825467	-0.904245158

4.2. táblázat. **A fájl méretének hatása a válaszidőre, módosított paraméterek mellett**

de különbség lényegében elenyésző. Amikor a fájl méret eléri a 100 Kbyte-ot a különbség még mindig csak 0.038 másodperc. Viszont amikor a fájl méretetet 1.5 Mbyte-ra emeltük Proxy Cache szerver használatával a válaszidők közti különbség már megközelíti az 1 másodpercet.

4.3. táblázat. **A Proxy Cache szerver modell jelölései**

λ:	A belső igények érkezési intenzitása
Λ:	A külső igények érkezési intenzitása
F:	Az átlagos fájl méret (byte-ban)
p:	A találati valószínűség
B_{xc}:	A Proxy Cache szerver kimenő puffere (byte-ban)
I_{xc}:	A Proxy Cache szerver keresési ideje (másodpercben)
Y_{xc}:	A Proxy Cache szerver statikus szerver ideje (másodpercben)
R_{xc}:	A dinamikus szerver arány a Proxy szerveren (byte/másodperc)
N_c:	Kliens hálózati sávszélesség (bit/másodperc)
B_s:	Web szerver kimenő puffere (byte-ban)
I_s:	Inicializálási idő (másodpercben)
Y_s:	A Web szerver statikus szerver ideje (másodperc)
R_s:	A Web szerver dinamikus szerver aránya (byte/másodperc)
N_s:	Szerver hálózati sávszélesség (bit/másodperc)

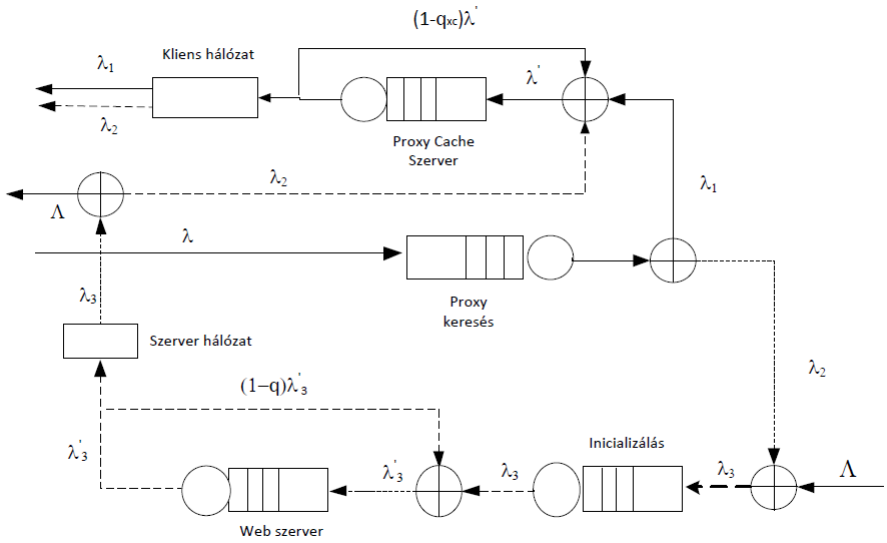
5 A Web szerver és a Proxy Cache szerver meghibásodásának hatása

Jelen esetben az előző fejezetben ismertetett Proxy Cache szerver modellt általánosítjuk úgy, hogy egy méginkább valósághű modellt kapjunk. A 4. fejezetben mind a Proxy Cache szerver, mind pedig a távoli Web szerver megbízhatóak voltak, most pedig feltesszük, hogy egyik sem megbízható, azaz bármelyikük elromolhat. Az általánosítással az a célunk, hogy megvizsgáljuk a szerverek meghibásodásának hatását a rendszerparaméterekre. A továbbiakban vizsgálni fogjuk a teljesítménybeli különbségeket, amennyiben "blokkolt" valamint "intelligens" forrásokat feltételezünk.

A fejezetben szereplő eredmények közlésre be vannak nyújtva (J6).

5.1. A javasolt modell

Mivel a vizsgált Markov-lánc állapottere, mely a módosított modellt leírja túl nagy, az egyensúlyi egyenlet felírása és ezek megoldása túl bonyolult lenne, ezért a MOSEL programcsomagot használjuk a modell leírására valamint a rendszerjellemzők kiszámítására. A MOSEL (Modeling, Specification and Evaluation Language) [6] egy leíró nyelv, mely segítségével különböző programcsomagokat használhatunk, mint például az SPNP-t (Stochastik Petri Net Package) vagy a TimeNet programcsomagot. A MOSEL által szolgáltatott eredményeket grafikusan is ábrázolni tudjuk az IGL (Intermediate Graphical Language) segítségével, mely része a MOSEL-nek. A modellünk grafikus ábrázolása nem változott a módosításokkal, így az 5.1 ábrán láthatjuk egy igény lehetséges útját a felhasználótól kiindulva egészen a visszaérkezésig. A modellben szereplő paraméterek listája a 4.3 táblázatban látható. Mint ahogyan a 4. fejezetben, most is feltételezzük,



5.1. ábra. Proxy Cache szerver modellje

hogy az igények a Proxy Cache szerverhez λ paraméterű Poisson-folyamat szerint érkeznek, valamint a Web szerverhez kívülről érkező igények Λ paraméterű Poisson-folyamat alapján érkeznek. Az 5.1 (hasonlóan a 4.1.-hez)ábrán egyenes vonal (λ_1) reprezentálja azt az esetet, amikor a keresett fájl megtalálható a Proxy Cache szerveren valamint szaggatott vonallal rajzolva jelöltük (λ_2) azon igények útját, melyek nem találhatóak meg a Proxy Cache szerveren, így ezek az igények továbbítódnak a távoli Web szerverhez. A Web szerver valamint a Proxy Cache szerver karakterisztikáját meghatározó paraméterek B_s, Y_s, R_s valamint B_{xc}, Y_{xc}, R_{xc} rendre a szerver kimenő puffere, a statikus szerver idő valamint a dinamikus szerver arány. A (3.5) képlet alapján a Web szerver kiszolgálási intenzitása:

$$\mu_{Web} = \frac{1}{Y_S + \frac{B_S}{R_S}}, \tag{5.1}$$

valamint a Proxy Cache szerver kiszolgálási intenzitása:

$$\mu_{PCS} = \frac{1}{Y_{xc} + \frac{B_{xc}}{R_{xc}}}. \tag{5.2}$$

Ha a keresett fájl nagyobb mint a Web szerver kimenő puffere akkor egy visszacsatolási ciklus kezdődik, mely addig tart míg a teljes fájl kiszolgálása be nem fejeződik. Legyen q annak a valószínűsége, hogy a fájlt egyből sikerül továbbítani, ahol

$$q = \min\left(1, \frac{B_s}{F}\right). \tag{5.3}$$

Teljesen hasonlóan modellezhető a Proxy Cache szerver is, ahol a távozó folyamat visszacsatolásának a valószínűsége $1 - q_{xc}$ ahol,

$$q_{xc} = \min\left(1, \frac{B_{xc}}{F}\right). \tag{5.4}$$

A Proxy Cache szerver és a Web szerver meghibásodhat a $(t, t + dt)$ intervallumban $\delta_{pcs}dt + o(dt)$ valamint $\delta_{web}dt + o(dt)$ valószínűséggel ha szabadok, valamint $\gamma_{pcs}dt + o(dt)$ és $\gamma_{web}dt + o(dt)$ valószínűséggel ha foglaltak. Ha a Proxy Cache szerver vagy a Web szerver foglalt állapotban romlanak el, akkor a megszakadt igény feldolgozása a javítás befejezése után folytatódik. A javítási idő exponenciális eloszlású $1/\nu_{pcs}$ és $1/\nu_{web}$ átlaggal. Ha a szerverek közül valamelyik elromlik két különböző eset lehetséges.

- **Blokkolt eset**: a szerver meghibásodása alatt nem érkezik új igény a szerverhez.
- **Nem blokkolt eset**: a szerver meghibásodása alatt is érkezhetnek újabb igények a szerverhez.

Amint látható a modellezett rendszer figyelembe vesz kétféle meghibásodási esetet: foglalt vagy szabad szerver állapotot, valamint két kiszolgálási esetet: blokkolt és nem blokkolt esetet, mely a rendszer nagymértékű bonyolultságát eredményezi.

A rendszer állapotát a t időpillanatban a

$$X_{PCS}(t) = (Y_{PCS}(t), C_{PCS}(t), Q_{PCS}(t)), \tag{5.5}$$

valamint a

$$X_{Web}(t) = (Y_{Web}(t), C_{Web}(t), Q_{Web}(t)) \tag{5.6}$$

folyamat írja le, ahol $Y_{PCS}(t) = Y_{Web}(t) = 0$ ha a szerver működik, és $Y_{PCS}(t) = Y_{Web}(t) = 1$ ha a szerver hibás, valamint $C_{PCS}(t) = C_{Web}(t) = 0$ ha a szerver nem foglalt, valamint $C_{PCS}(t) = C_{Web}(t) = 1$ ha a szerver foglalt. Legyen $Q_{PCS}(t)$ és $Q_{Web}(t)$ a pufferben lévő igények átlagos száma a Proxy Cache szerver valamint a Web szerver esetén.

Mivel a használt valószínűségi változók független exponenciális eloszlásúak a rendszer állapotát leíró folyamatok véges állapotterű Markov-láncok.

Legyenek a stacionárius valószínűségek:

$$P_{PCS}(q, r, j) = \lim_{t\to\infty} P(Y_{PCS}(t), C_{PCS}(t), Q_{PCS}(t)), \tag{5.7}$$

$$q = 0, 1, r = 0, 1, j = 0, \cdots, K_{PCS},$$

és

$$P_{Web}(q, r, j) = \lim_{t\to\infty} P(Y_{Web}(t), C_{Web}(t), Q_{Web}(t)), \tag{5.8}$$

$$q = 0, 1, r = 0, 1, j = 0, \cdots, K_{Web},$$

ahol K_{PCS} és K_{Web} a szerverek puffer mérete.

Amint sikerült megkapnunk a fenti valószínűségeket, az egyensúlyi állapothoz tartozó rendszerjellemzőket a következő képletek alapján kapjuk meg:

- *A szerverek kihasználtsága*

$$U_{S,PCS} = \sum_{j=0}^{K_{PCS}} P_{PCS}(0, 1, j), \tag{5.9}$$

$$U_{S,Web} = \sum_{j=0}^{K_{Web}} P_{Web}(0,1,j). \tag{5.10}$$

- *A szerverek elérhetősége*

$$A_{PCS} = \sum_{r=0}^{1} \sum_{j=0}^{K_{Pcs}} P_{PCS}(0,r,j), \tag{5.11}$$

$$A_{Web} = \sum_{r=0}^{1} \sum_{j=0}^{K_{Web}} P_{Web}(0,r,j). \tag{5.12}$$

- *Az átlagos igényszámok*

$$M_{PCS} = \sum_{q=0}^{1} \sum_{r=0}^{1} \sum_{j=0}^{K_{Pcs}} jP_{PCS}(q,r,j), \tag{5.13}$$

$$M_{Web} = \sum_{q=0}^{1} \sum_{r=0}^{1} \sum_{j=0}^{K_{Web}} jP_{Web}(q,r,j). \tag{5.14}$$

- *Az átlagos válaszidők*
  A Little formulák felhasználásával [27] az átlagos válaszidők a következő összefüggésekkel kaphatóak meg:

$$T_{PCS} = M_{PCS}/\lambda_{PCS}, \tag{5.15}$$

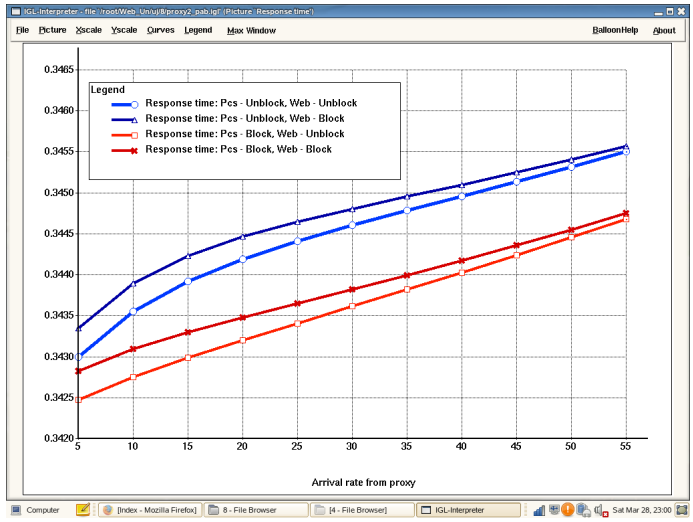
  ahol λ_{PCS} a Proxy Cache szerverhez érkező igények átlagos intenzitása, valamint

$$T_{Web} = M_{Web}/\lambda_{Web} \tag{5.16}$$

  ahol λ_{Web} a Web szerverhez érkező igények átlagos intenzitása

- *Az igények teljes válaszideje*

$$\begin{aligned} T = {} & T_{Lookup} + p * \left(T_{PCS} + \frac{F}{N_c} \right) \\ & + (1-p) * \left(T_{Init} + T_{Web} + \frac{F}{N_s} + T_{PCS} + \frac{F}{N_c} \right), \end{aligned} \tag{5.17}$$

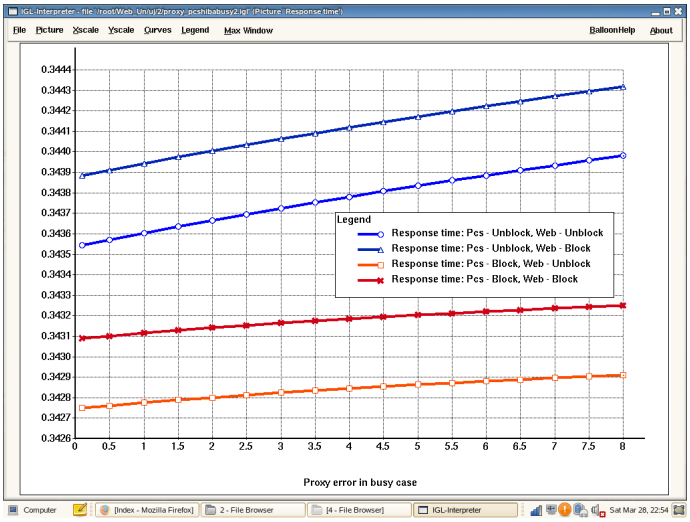


5.2. ábra. $p = 0.25$, $\Lambda = 10$, $\nu_{pcs} = \nu_{web} = 10$, and $\delta_{pcs} = \delta_{web} = \gamma_{pcs} = \gamma_{web} = 0.2$

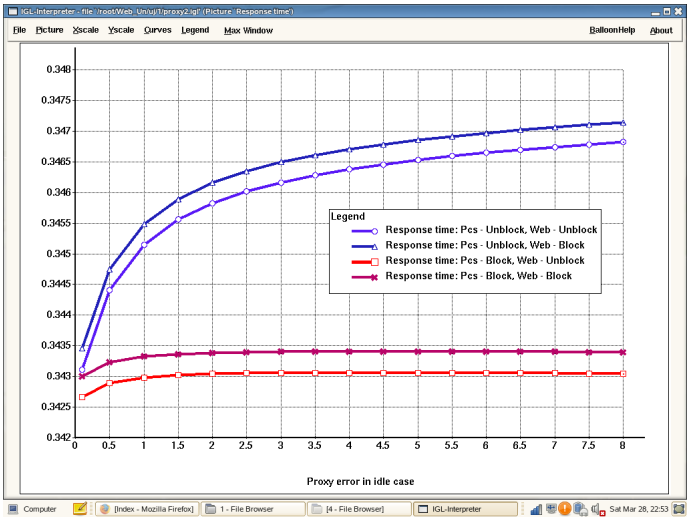
ahol $T_{Lookup} = \frac{1}{\frac{1}{I_{xc}} - \lambda}$ a keresési idő, amíg eldől, hogy az igényelt fájl megtalálható-e a Proxy Cache szerveren vagy nem, valamint $T_{Init} = \frac{1}{\frac{1}{I_s} - \lambda_3}$ az egyszeri inicializálási idő.

5.2. Numerikus eredmények

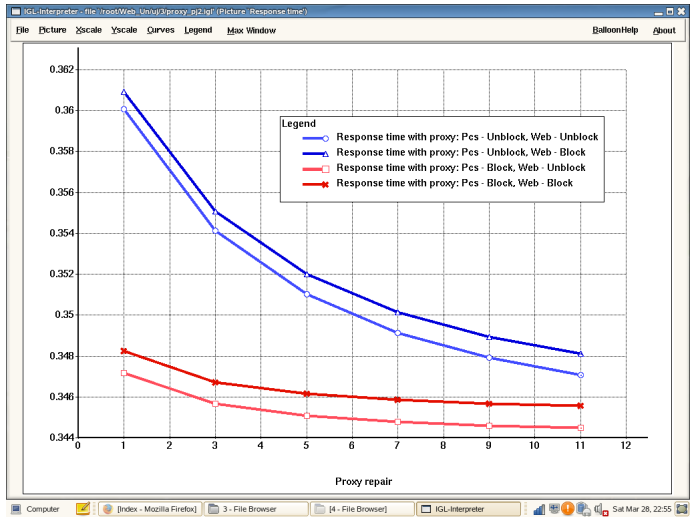
A következőekben a numerikus eredményeket grafikusan ábrázoljuk, hogy bemutassuk a meghibásodási és a javítási intenzitások hatását a teljes válaszidőkre. A számításokhoz a Web és Proxy Cache szerver paraméterek értékeit [28] alapján határoztuk meg mint ahogyan a 4. fejezetben is. Ezek az értékek: $I_s = I_{xc} = 0.004$ másodperc, $B_s = B_{xc} = 2000$ byte, $Y_s = Y_{xc} = 0.000016$ másodperc, $R_s = R_{xc} = 1.25$ Mbyte/másodperc, $N_s = 1544$ Kbit/másodperc valamint $N_c = 128$ Kbit/másodperc.



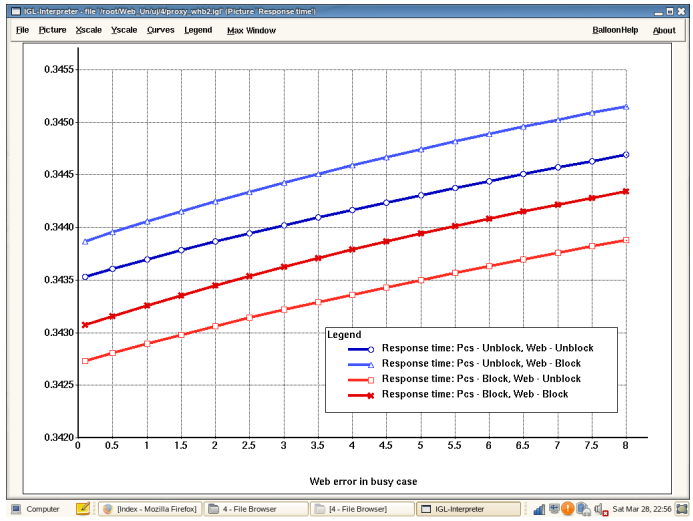
5.3. ábra. $p = 0.25$,$\lambda = \Lambda = 10$, $\nu_{pcs} = \nu_{web} = 10$, and $\delta_{pcs} = \delta_{web} = \gamma_{web} = 0.2$



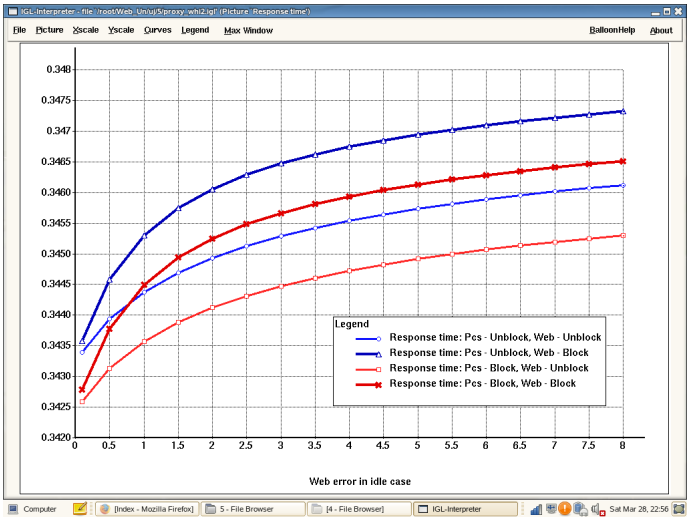
5.4. ábra. $p = 0.25$,$\lambda = 30, \Lambda = 10$, $\nu_{pcs} = \nu_{web} = 10$, and $\delta_{web} = \gamma_{pcs} = \gamma_{web} = 0.2$



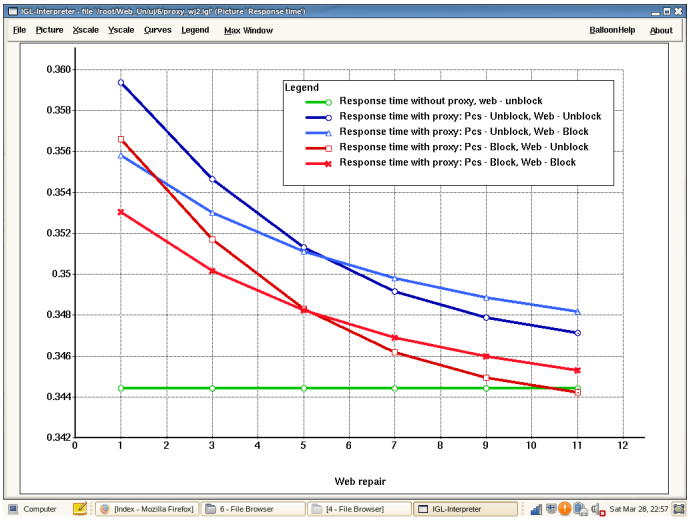
5.5. ábra. $p = 0.25$,$\lambda = 40, \Lambda = 10$, $\nu_{web} = 10$, and $\delta_{pcs} = \delta_{web} = \gamma_{pcs} = \gamma_{web} = 2$



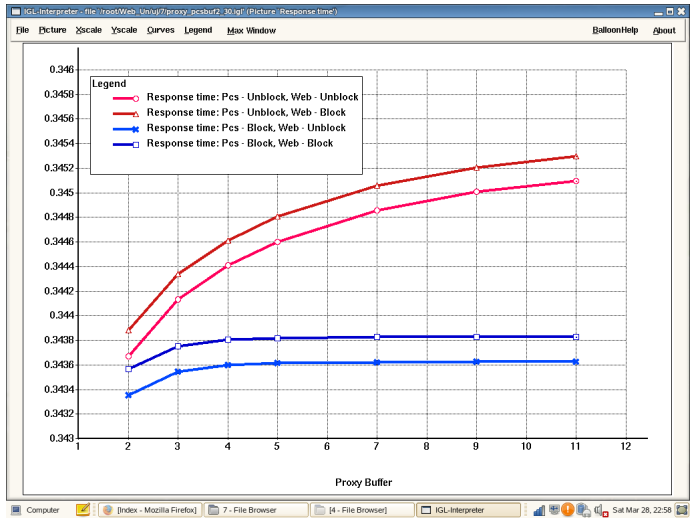
5.6. ábra. $\lambda = \Lambda = 10$, $\nu_{pcs} = \nu_{web} = 10$, and $\delta_{pcs} = \delta_{web} = \gamma_{pcs} = 0.2$



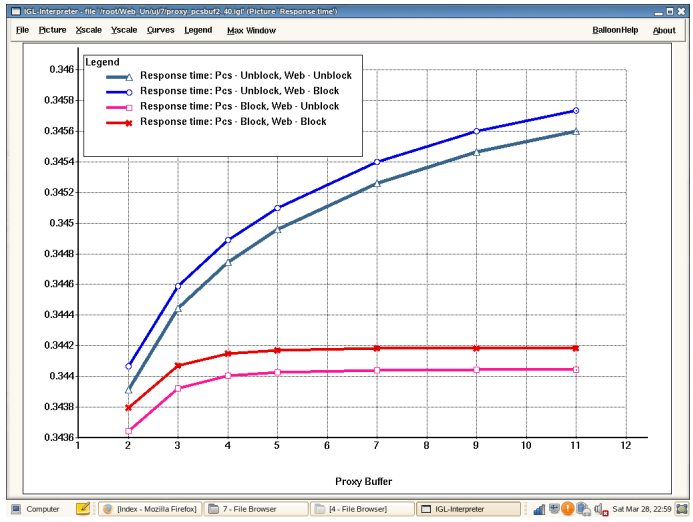
5.7. ábra. $\lambda = \Lambda = 10$, $\nu_{pcs} = \nu_{web} = 10$, and $\delta_{pcs} = \gamma_{pcs} = \gamma_{web} = 0.2$



5.8. ábra. $\lambda = \Lambda = 10$, $\nu_{pcs} = 10$, and $\delta_{pcs} = \delta_{web} = \gamma_{pcs} = \gamma_{web} = 2$



5.9. ábra. $\lambda = 30, \Lambda = 10$, $\nu_{pcs} = \nu_{web} = 10$, and $\delta_{pcs} = \delta_{web} = \gamma_{pcs} = \gamma_{web} = 0.2$



5.10. ábra. $\lambda = 40, \Lambda = 10$, $\nu_{pcs} = \nu_{web} = 10$, and $\delta_{pcs} = \delta_{web} = \gamma_{pcs} = \gamma_{web} = 0.2$

5.2.1. Eredmények

- Az 5.2 grafikonon a használt paraméterek értékei: $p = 0.25$ a találati valószínűség, $\Lambda = 10$ igény/másodperc a külső igények érkezési intenzitása, $\nu_{pcs} = \nu_{web} = 10$ a javítási intenzitás, azaz az átlagos javítási idők $\frac{1}{\nu_{pcs}} = \frac{1}{\nu_{web}} = 0.1$, és $\delta_{pcs} = \delta_{web} = \gamma_{pcs} = \gamma_{web} = 0.2$ a Proxy Cache szerver és a Web szerver meghibásodásának intenzitása üresjárati és foglalt állapotokban. A grafikonon az igények teljes válaszidejét láthatjuk a belső érkezési intenzitás függvényében blokkolt és nem blokkolt esetekben. Amint látható az átlagos válszidő a belső érkezési intenzitás növekedésével szintén növekszik, amint azt a 4. fejezetben láthattuk. Megfigyelhetjük, hogy amennyiben egy szerver blokkolt és nem blokkolt állapotában vizsgáljuk a válaszidőket, az érkezési intenzitás növekedésével a két allapot közötti különbség csökken.

- Az 5.3 valamint az 5.4 grafikonokon a Proxy Cache szerver meghibásodásának hatását vizsgálhatjuk meg a teljes válaszidőre, foglalt és üresjárati szerver állapotban. A használt paraméterek: $p = 0.25$,$\lambda = \Lambda = 10$, $\nu_{pcs} = \nu_{web} = 10$, és $\delta_{pcs} = \delta_{web} = \gamma_{web} = 0.2$ valamint a 5.4 grafikonon használt paraméter értékek: $p = 0.25$,$\lambda = 30, \Lambda = 10$, $\nu_{pcs} = \nu_{web} = 10$, és $\delta_{web} = \gamma_{pcs} = \gamma_{web} = 0.2$. Amint megfigyelhetjük a válaszidők alacsonyabbak a blokkolt Proxy Cache szerverek esetében, mint a nem blokkolt esetekben. Megvizsgálva a 5.4 (az átlagos válszidő az üresjárati Proxy szerver meghibásodásának függvénye) grafikont azt tapasztaljuk, hogy abban az esetben amikor a Proxy Cache szerver ugyanabban az állapotban van és csak a Web szerver állapota változik, pl. Proxy Cache szerver blokkolt és a Web szerver egyik esetben blokkolt a másikban nem (kék grafikonok), valamint a Proxy szerver nem blokkolt és a Web szerver egyik esetben blokkolt a másikban pedig nem (piros grafikonok) a válaszidőt ábrázoló görbék párhuzamosak. Ez abból adódik, hogy a Web szerver meghibásodási és javítási intenzitásai azonosak, csak a Web szerver blokkolási tulajdonsága változik. Tovább vizsgálva a 5.4 grafikont azt tapasztaljuk, hogy amennyiben a Proxy Cache szerver blokkolt az igények teljes válaszideje magasabb meghibásodási együttható esetén konstans lesz, azaz független lesz az üresjárati

meghibásodási intenzitásra.

- Az 5.5 grafikonon megvizsgálhatjuk a Proxy Cache szerver javítási intenzitásának hatását a válaszidőkre. A használt paraméterek: $p = 0.25$,$\lambda = 40$, $\Lambda = 10$, $\nu_{web} = 10$, és $\delta_{pcs} = \delta_{web} = \gamma_{pcs} = \gamma_{web} = 2$. Amint látható a javítási intenzitás növelésével a válaszidők csökkennek. Ebben az esetben a Web szerver meghibásodási és javítási intenzitásai változatlanok, ebből adódik, hogy amikor a Proxy Cache szerver blokkolási algoritmusa megegyezik (blokkolt és nem blokkolt esetek) és csak a Web szerver blokkolási metódusa különböző a válaszidőket ábrázoló görbék párhuzamosak. (piros illetve kék görbék).

- Az 5.6 és 5.7 grafikonokon a Web szerver meghibásodási intenzitásának hatását figyelhetjük meg a válaszidőkre foglalt valamint üresjárati esetekben. A grafikonon használt paraméterek értékei: $\lambda = \Lambda = 10$,$\nu_{pcs} = \nu_{web} = 10$, és $\delta_{pcs} = \delta_{web} = \gamma_{pcs} = 0.2$ az 5.6 grafikonon, valamint $\lambda = \Lambda = 10$,$\nu_{pcs} = \nu_{web} = 10$, és $\delta_{pcs} = \gamma_{pcs} = \gamma_{web} = 0.2$ az 5.7 grafikonon. Amint látható a teljes válaszidők nagyobbak lesznek minden esetben (blokkolt vagy nem blokkolt esetek) amennyiben nagyobb meghibásodási arányt használunk. Az előző esetekhez hasonlóan amikor a Web szerver blokkolási metódusai megegyeznek és csak a Proxy Cache szerver blokkolási módszerei változnak a válaszidőket ábrázoló görbék párhuzamosak. (Web szerver blokkolt és a Proxy szerver blokkolási módszere változik valamint a Web szerver nem blokkolt és a Proxy szerver blokkolási módszere változik).

- Az 5.8 ábra azt mutatja meg, hogyan változik a teljes válaszidő a Web szerver javítási intenzitásának növekedésével. A grafikonon használt paraméterek: $\lambda = \Lambda = 10$, $\nu_{pcs} = 10$, and $\delta_{pcs} = \delta_{web} = \gamma_{pcs} = \gamma_{web} = 2$. Amint látható, a válaszidők az 5.5 grafikonhoz hasonlóan csökkennek a javitási intenzitás növelésével. Az 5.6 és 5.7 grafikonok esetében megmutatott párhuzamosság itt is megfigyelhető ugyanolyan blokkolási módszer esetén. (A Web szerver blokkolási módszere megegyezik, csak a Proxy Cache szerver blokkolási módszerét változtatjuk.)

- Az 5.9 és az 5.10 grafikonok a válaszidőt a Proxy Cache szerver puffer méretének függvényeként ábrázoltuk. A használt paraméterek: $\lambda = 30, \Lambda = 10$,$\nu_{pcs} = \nu_{web} = 10$, és $\delta_{pcs} = \delta_{web} = \gamma_{pcs} = \gamma_{web} = 0.2$ az 5.9 grafikonon, valamint $\lambda = 40, \Lambda = 10$,$\nu_{pcs} = \nu_{web} = 10$, és $\delta_{pcs} = \delta_{web} = \gamma_{pcs} = \gamma_{web} = 0.2$ az 5.10 grafikonon. Mindkét grafikonon megfigyelhetjük, hogy amennyiben a Proxy Cache szerver blokkolt, úgy a válaszidők egy bizonyos puffer méret után már nem változnak tovább. Viszont amennyiben a Proxy Cache szerver nem blokkolt, akkor a válaszidők a puffer méretének növelésével természetesen növekszik. A korábban megállapított párhuzamosság itt is megfigyelhető.

6 A Proxy Cache szerver GI/G/1 approximációs modellje

A jelen fejezetben bemutatjuk a Proxy Cache szerver GI/G/1 approximációs modelljét, mely egy példa a Paraméter dekompozíciós eljárás használatára (lásd [12]), ahol az egyes csomópontokat egymástól elkülönülten vizsgáljuk.

A fejezetben szereplő eredmények a [7] cikkben lettek publikálva.

Ebben a modellben az érkezési folyamat egy úgynevezett "GI - General inter-arrival" folyamat, melyet az érkezési időközök várható értékével és a relatív szórásnégyzetével (c^2) jellemzünk, valamint a kiszolgálási idő bármilyen általános eloszlású lehet.

6.1. A GI/G/1 approximáció

Az approximáció használatához a következő feltételeknek kell teljesülniük:

- Az érkezési folyamat úgynevezett "felújítási" folyamat kell legyen, azaz az érkezési időközök független, azonos eloszlású valószínűségi változók.
- A kiszolgálási idők valószínűségi változója bármilyen általános eloszlású lehet.
- Adott az érkezési folyamat inetenzitása λ_A, valamint az érkezási folyamat relatív szórásnégyzete (c_A^2).
- Adott a kiszolgálási idő várható értéke τ_S, valamint a kiszolgálási idő relatív szórásnégyzete (c_S^2).

- Az azonnali visszacsatolást, amikor egy sor távozó folyamata vissza van irányítva egyből ugyanahhoz a sorhoz, külön kell vizsgálni.

Ez az aproximáció olyan algoritmusokat szolgáltat, melyekkel modellezhetjük az általános hálózati folyamatokat, mint például a forgalom egyesítést, a sortól való távozást, valamint a forgalom szétválását.

Minden esetben, a részletes számítások előtt a modellt módosítanunk kell, hogy eliminálhassuk az azonnali visszacsatolásokat. Ezt a módosítást az érintett sor kiszolgálási idejének megváltoztatásával végezzük, melyet a későbbiekben részletezünk.

A szükséges kalkulációk eredményekép az aproximáció segítségével megkapjuk a szükséges rendszerjellemzőket (átlagos sorhossz, átlagos várakozási idő, stb.), mind a sorokra, mind pedig az egész hálózatra vonatkozóan.

A következőekben bemutatjuk azon approximációs egyenleteket, melyek segítségével leírhatóak a korábban említett hálózati forgalomhoz tartozó folyamatok, mint például az egyesítés, szétválás, távozás, valamint bemutajuk az azonnali visszacsatolás törléséhez szükséges paraméter változtatásokat. (lásd [12], [5]).

1) *GI folyamatok egyesítése:* n darab független GI folyamat (mindegyiket rendre jellemez a λ_j és c_j^2, j $= 1, \ldots, n$) egyesítése amint egy következő csomópontba lépnek, egy GI folyamattal közelítheünk, mely paraméterei: λ_A és c_A^2, az átalgos érkezési intenzitás és az érkezési időközök relatív szórásnégyzete. Az átlagos intenzitást és a relatív szórásnégyzetet az alábbi egyenletek szolgáltatják (lásd [12]):

$$\lambda_A = \sum_{j=1}^{n} \lambda_j, \tag{6.1}$$

és

$$c_A^2 = \varpi \sum_{j=1}^{n} \frac{\lambda_j}{\lambda_A} c_j^2 + 1 - \varpi, \tag{6.2}$$

ahol

$$\varpi = \frac{1}{1 + 4(1-\rho)^2(\nu - 1)}, \tag{6.3}$$

$$\nu = \frac{1}{\sum_{j=1}^{n} \left(\frac{\lambda_j}{\lambda_A}\right)^2} \tag{6.4}$$

és ρ a csomópont kihasználtsága, azaz $\rho = \lambda_A \tau_S$ és τ_S a ksizolgálási idő várható értéke.

2) *Egy sortól távozó folyamat:* Egy csomóponttól távozó folyamatot közelíthetünk egy GI folyamattal, ahol λ_D jelenti az átlagos távozási intenzitást, valamint c_D^2 jelenti a távozó igények közötti időközök relatív szórásnégyzetét. Az egyensúlyi állapot következményeként tudjuk, hogy az átlagos érkezési intenzitás megegyezik az átlagos távozási intenzitással, azaz

$$\lambda_D = \lambda_A. \tag{6.5}$$

A távozási folyamat relatív szórásnégyzete pedig megkapható a következő egyenlettel:

$$c_D^2 = \rho^2 c_S^2 + \left(1 - \rho^2\right) c_A^2. \tag{6.6}$$

3) *Egy GI folyamat véletlen osztása:* Egy GI folyamatot, melyet a λ és c^2 paraméterek jellemeznek n darab független folyamatra osztunk, rendre p_i valószínűséggel ($\sum_{i=1}^{n} p_i = 1$). Ekkor az i-edik folyamat paramétereit a következőképpen számíthatjuk ki:

$$\lambda_i = p_i \lambda, \tag{6.7}$$

$$c_i^2 = p_i c^2 + (1 - p_i). \tag{6.8}$$

4) *Az azonnali visszacsatolás törlése:* Ha egy csomóponttól távozó folyamatot azonnal visszairányítunk ugyanahhoz a sorhoz (Q_i), akkor a Q_i sorhoz érkező folyamat valójában a csomóponton kívülről érkező igények intenzitásának (λ_i), valamint a visszacsatolt folyamat intenzitásának ($p_{ii}\lambda_i$) az összege.

Az azonnali visszacsatolás megszüntetése a sor átlagos kiszolgálási idejének valamint a relatív szórásnégyzetnek a megfelelő módosításával történik. Feltételezve, hogy a kiszolgálási idő eloszlásának az eredeti paraméterei: $\tau_{S,U}$ - az átlagos kiszolgálási idő, és $c^2_{S,U}$ - a kiszolgálási idő relatív szórásnégyzete, az azonnali visszacsatolás eliminálásával a kiszolgálási idő eloszlásának módosult paraméterei a következők:

$$\tau_{S,M} = \frac{\tau_{S,U}}{1 - p_{ii}}, \tag{6.9}$$

$$c^2_{S,M} = p_{ii} + (1 - p_{ii})c^2_{S,U}, \tag{6.10}$$

és

$$W_{q,M} = \frac{W_{q,M}}{1 - p_{ii}}. \tag{6.11}$$

5) *Átlagos várakozási idő:*

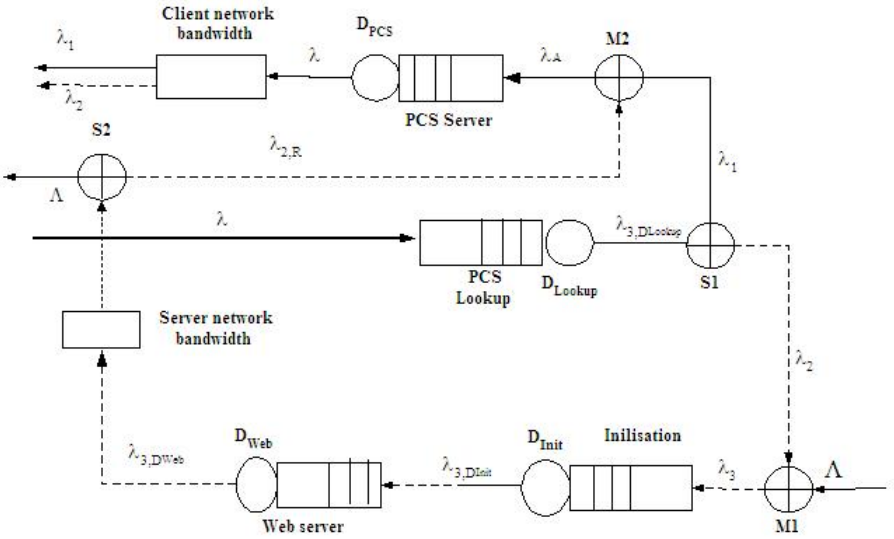
Ha a vizsgált csomópont egy GI/G/1 -es sor akkor alkalmazhatjuk a Kramer és Langenbach-Belz approximációt (lásd [37]):

$$W_q = \frac{\tau_S \cdot \rho(c^2_A + c^2_D)\beta}{2(1 - \rho)}, \tag{6.12}$$

ahol

$$\beta_{Web} = \begin{cases} exp\left(\frac{2(1-\rho)(1-c^2_A)^2}{3\rho(c^2_A+c^2_D)}\right), & c^2_A < 1 \\ 1, & c^2_A \geq 1 \end{cases}$$

.



6.1. ábra. Proxy Cache szerver módosított modellje

6.2. A Proxy Cache szerver GI/G/1 modellje

Az eddig tanulmányozott Proxy Cache szerver modellt (lásd a 4. fejezetet) úgy szeretnénk általánosítani, hogy a modellben szereplő M/M/1-es sorok helyett GI/G/1-es sorokat használunk. A kívánt rendszerparamétereket a fentebb tárgyalt GI/G/1 -es approximáció segítségével fogjuk megkapni.

Feltételezzük, hogy a belső igények a Proxy Cache szerverhez GI folyamat szerint érkeznek λ érkezési intenzitással, és c_λ^2 relatív szórásnégyzettel. A Web szerverhez kívűlről érkező igények szintén GI folyamat alapján érkeznek Λ és c_Λ^2 paraméterekkel.

A 6.1 ábra a módosított modell alapján mutatja egy igény lehetséges útját a felhasználótól kiindulva egészen a visszaérkezésig. A modellben szereplő paraméterek listája a 4.3 táblázatban látható.

A "PCS LOOKUP" szemlélteti azt a sort melynél eldől, hogy a keresett

fájl megtalálható-e a Proxy Cache szerveren vagy nem. A "PCS LOOKUP" sorhoz tartozó kiszolgálási idő bármilyen tetszőleges eloszlás lehet μ_{Lookup}, c^2_{Lookup} paraméterekkel. Az "Initialisation" sor kiszolgálási ideje az előzőekhez hasonlóan szintén általános eloszlású μ_{Init} és c^2_{Init} paraméterekkel. A Web szerver valamint a Proxy Cache Szerver kiszolgálási idejének eloszlását a μ_{Web}, c^2_{Web} és μ_{PCS}, c^2_{PCS} paraméterek jellemzik ahol:

$$\mu_{Lookup} = \frac{1}{I_{xc}}, \tag{6.13}$$

$$\mu_{Init} = \frac{1}{I_S}, \tag{6.14}$$

és

$$\mu_{Web} = \frac{1}{Y_S + \frac{B_S}{R_S}}, \tag{6.15}$$

$$\mu_{PCS} = \frac{1}{Y_{xc} + \frac{B_{xc}}{R_{xc}}}, \tag{6.16}$$

ahol I_{xc} a Proxy Cache szerverhez tartozó úgynevezett "keresési" idő (lásd a 4. fejezetet) valamint I_s a Web szerverhez tartozó úgynevezett egyszeri inicializálási idő (lásd a 3. fejezetet).

A B_s ás B_{xc} paraméterek a Web szerver illetve a Proxy Cache szerver kimenő puffer kapacitása, az Y_s és Y_{xc} a statikus szerver idők valamint az R_s és R_{xc} a dinamikus szerver arány a Web szerver és Proxy Cache szerver esetén.

Az eredeti modell módosításának az első lépése a szervereknél esetlegesen előforduló azonnali visszacsatolások törlése a sorok paramétereinek módosításával. A 6.1-es ábra az eredeti 4.1-es ábra módosítása, ahol a Proxy Cache szerver és a Web szerver visszacsatolásai törölve vannak.

A visszacsatolások törlése után a két szerver módosított paraméterei a (6.9), (6.10) alapján:

$$\mu_{Web,M} = \mu_{Web} * q, \tag{6.17}$$

$$c^2_{Web,M} = (1 - q) + q * c^2_{Web}, \tag{6.18}$$

és

$$\mu_{PCS,M} = \mu_{PCS} * q_{xc}, \tag{6.19}$$

$$c^2_{PCS,M} = (1 - q_{xc}) + q_{xc} * c^2_{PCS}. \tag{6.20}$$

Az új modellben 2 forgalom szétágazási pont (S1,S2), 2 darab forgalom egyesülési pont (M1,M2), valamint 4 darab különálló sor található (PCS Lookup, PCS Server, Initialization, Web server). Ezekben a pontokban a folyamatok paramétereit újra kell kalkulálnunk.

Az S1 pontban az igények két irányba ágaznak szét p és $1-p$ valószínűséggel.

A "PCS Lookup" sort elhagyó igények távozási folyamatának (D_{Lookup}) az újrakalkulált paraméterei (távozási intenzitás, távozóigények időközeinek relatív szórásnégyzete) a (6.5) és a (6.6) képletek alapján a következőek:

$$\lambda_D = \lambda, \tag{6.21}$$

$$c^2_{D_{Lookup}} = \rho^2 * c^2_{Lookup} + \left(1 - \rho^2\right) * c^2_{\lambda}, \tag{6.22}$$

ahol

$$\rho = \frac{\lambda}{\mu_{I_{xc}}}. \tag{6.23}$$

A 6.1 ábrán az egyenes vonallal (λ_1) azokat az igényeket jelöltük melyek megtalálhatóak a Proxy Cache szerveren, azaz egyből továbbíthatóak a kliensnek. Szaggatott vonallal (λ_2) ábrázoltuk azokat az igényeket,

melyek nem szolgálhatók ki a Proxy Cache szerverrel, azaz az igények továbbítódnak a távoli Web szerver felé.

A két folyamathoz tartozó paraméterek értékei a (6.7) és a (6.8) alapján:

$$\lambda_1 = p * \lambda_D, \tag{6.24}$$

$$c_1^2 = p * c_{D_{Lookup}}^2 + (1 - p), \tag{6.25}$$

és

$$\lambda_2 = (1 - p) * \lambda_D, \tag{6.26}$$

$$c_2^2 = (1 - p) * c_{D_{Lookup}}^2 + p. \tag{6.27}$$

Az M1 pontban a belső igények egy része (λ_2) egyesülnek a külső igényekkel. Az így kapott folyamat (λ_3) paraméterei felhasználva a (6.1) valamint a (6.2) képleteket:

$$\lambda_3 = \lambda_2 + \Lambda, \tag{6.28}$$

$$c_3^2 = w * \left(\frac{\lambda_2}{\lambda_3} * c_2^2 + \frac{\Lambda}{\lambda_3} * c_\Lambda^2 \right) + (1 - w), \tag{6.29}$$

ahol

$$w = \frac{1}{4 * (1 - \rho)^2 * (\nu - 1)}, \tag{6.30}$$

$$\nu = \frac{1}{\left(\frac{\lambda_2}{\lambda_3}\right)^2 + \left(\frac{\Lambda}{\lambda_3}\right)^2}, \tag{6.31}$$

és

$$\rho = \frac{\lambda_3}{\mu_{Init}}. \tag{6.32}$$

Az egyszeri inicializálási sortól távozó igények ($\lambda_{3,D_{Init}}$) paraméterei:

$$\lambda_{3,D_{Init}} = \lambda_3, \tag{6.33}$$

$$c^2_{3,D_{Init}} = \rho^2 * c^2_{Init} + \left(1 - \rho^2\right) * c^2_3, \tag{6.34}$$

ahol

$$\rho = \frac{\lambda_{3,D_{Init}}}{\mu_{Init}}. \tag{6.35}$$

Az egyszeri inicializálás után az igények megérkeznek a Web szerverhez, ahonnan már korábban töröltük a visszacsatolást. A Web szervert elhagyó folyamat ($\lambda_{3,D_{Web}}$) paraméterei:

$$\lambda_{3,D_{Web}} = \lambda_3, \tag{6.36}$$

$$c^2_{3,D_{Web}} = \rho^2 * c^2_{Web,M} + \left(1 - \rho^2\right) * c^2_{3,D_{Init}}, \tag{6.37}$$

ahol

$$\rho = \frac{\lambda_3}{\mu_{Web,M}}. \tag{6.38}$$

Az S2 pontban a Web szervert elhagyó $\lambda_{3,D_{Web}}$ folyamat két irányba ágazik el. Az igények egyik része a külső igények, melyek $\frac{\Lambda}{\lambda_2+\Lambda}$ valószínűséggel távoznak, valamint az igények másik része a belső igények ($\lambda_{2,R}$) melyek továbbítódnak a Proxy Cache szerver felé.

A $\lambda_{2,R}$ folyamat paraméterei:

$$\lambda_{2,R} = \lambda_2, \tag{6.39}$$

és

$$c^2_{2,R} = \frac{\lambda_2}{\lambda_2 + \Lambda} * c^2_{3,D_{Web}} + \left(1 - \frac{\lambda_2}{\lambda_2 + \Lambda}\right). \tag{6.40}$$

Az M2 pontban a λ_1 folyamat és a Web szervertől visszatérő $\lambda_{2,R}$ folyamat egyesül. Az így kapott λ_A folyamat paraméterei:

$$\lambda_A = \lambda_1 + \lambda_{2,R} = \lambda_1 + \lambda_2 = \lambda, \tag{6.41}$$

$$c_A^2 = w * \left(\frac{\lambda_1}{\lambda} * c_1^2 + \frac{\lambda_2}{\lambda} * c_{2,R}^2 \right) + (1 - w), \tag{6.42}$$

ahol

$$w = \frac{1}{4 * (1 - \rho)^2 * (\nu - 1)}, \tag{6.43}$$

$$\nu = \frac{1}{\left(\frac{\lambda_1}{\lambda}\right)^2 + \left(\frac{\lambda_2}{\lambda}\right)^2}, \tag{6.44}$$

és

$$\rho = \frac{\lambda}{\mu_{PCS,M}}. \tag{6.45}$$

Így a teljes válaszidőt a (4.10) alapján a következő egyenletek adják.

$$\begin{aligned} T_{xc} = T_{Lookup} + p * \left\{ T_{PCS} + \frac{F}{N_c} \right\} \\ + (1 - p) * \{ T_{Init} + T_{Web} + \frac{F}{N_s} + T_{PCS} + \frac{F}{N_c} \right\}, \end{aligned} \tag{6.46}$$

ahol a (6.12) alapján kapjuk:

$$\begin{aligned} T_{Lookup} &= W_{Lookup} + \frac{1}{\mu_{Lookup}} = \\ &= \frac{\frac{1}{\mu_{Lookup}} * \rho_{Lookup} * \left(c_\lambda^2 + c_{Lookup}^2 \right) * \beta}{2 * (1 - \rho_{lookup})} + \frac{1}{\mu_{Lookup}}, \end{aligned} \tag{6.47}$$

$$\beta = \begin{cases} \exp\left(-\frac{2*(1-\rho_{Lookup})*\left(1-c_\lambda^2\right)^2}{3*\rho_{Lookup}*\left(c_\lambda^2 + c_{Lookup}^2\right)} \right), & c_\lambda^2 < 1 \\ 1, & c_\lambda^2 \geq 1 \end{cases},$$

$$\rho_{Lookup} = \frac{\lambda}{\mu_{Lookup}},$$

valamint

$$\begin{aligned} T_{PCS} &= W_{PCS} + \frac{1}{\mu_{PCS,M}} = \\ &= \frac{\frac{1}{\mu_{PCS,M}} * \rho_{pcs} * \left(c_A^2 + c_{pcs,M}^2\right) * \beta}{2 * (1 - \rho_{pcs})} + \frac{1}{\mu_{pcs,M}}, \end{aligned} \tag{6.48}$$

ahol

$$\beta = \begin{cases} exp\left(-\frac{2*(1-\rho_{pcs})*\left(1-c_A^2\right)^2}{3*\rho_{pcs}*\left(c_A^2+c_{pcs,M}^2\right)}\right), & c_A^2 < 1 \\ 1, & c_A^2 \geq 1 \end{cases},$$

$$\rho_{pcs} = \frac{\lambda_A}{\mu_{pcs,M}}.$$

Ismét felhasználva a (6.12) képletet kapjuk:

$$\begin{aligned} T_{Init} &= W_{Init} + \frac{1}{\mu_{Init}} = \\ &= \frac{\frac{1}{\mu_{Init}} * \rho_{Init} * \left(c_3^2 + c_{Init}^2\right) * \beta_{Init}}{2 * (1 - \rho_{Init})} + \frac{1}{\mu_{Init}}, \end{aligned} \tag{6.49}$$

ahol

$$\beta_{Init} = \begin{cases} exp\left(-\frac{2*(1-\rho_{Init})*\left(1-c_3^2\right)^2}{3*\rho_{Init}*\left(c_3^2+c_{Init}^2\right)}\right), & c_3^2 < 1 \\ 1, & c_3^2 \geq 1 \end{cases},$$

$$\rho_{Init} = \frac{\lambda_3}{\mu_{Init}},$$

és

$$\begin{aligned} T_{Web} &= W_{Web} + \frac{1}{\mu_{Web,M}} = \\ &= \frac{\frac{1}{\mu_{Web,M}} * \rho_{web} * \left(c^2_{D_{Init}} + c^2_{web,M}\right) * \beta_{web}}{2 * (1 - \rho_{web})} + \frac{1}{\mu_{web,M}}, \end{aligned} \tag{6.50}$$

$$\beta_{Web} = \begin{cases} exp\left(-\frac{2*(1-\rho_{web})*\left(1-c^2_{D_{Init}}\right)^2}{3*\rho_{web}*\left(c^2_{D_{Init}}+c^2_{web,M}\right)}\right), & c^2_{D_{Init}} < 1 \\ 1, & c^2_{D_{Init}} \geq 1 \end{cases},$$

$$\rho_{web} = \frac{\lambda_3}{\mu_{web,M}}. \tag{6.51}$$

6.3. Numerikus eredmények

A Numerikus eredményekhez az előző fejezetekben használt paramétereket vesszük alapul. Azaz a használt szerver paraméterek a következőek: $I_s = I_{xc} = 0.004$ másodperc, $B_s = B_{xc} = 2000$ bytes, $Y_s = Y_{xc} = 0.000016$ másodperc, $R_s = R_{xc} = 1.25$ Mbyte/s, $N_s = 1544$ Kbit/s, és $N_c = 128$ Kbit/s. Mint ahogyan a korábbi fejezetekben leírtuk a használt paraméterek a [28] eredmények alapján lettek kiválasztva.

Az approximációs eljárás validálása érdekében egy szimulációs programot készítettünk. A program Microsoft Visual Basic 2005 -ben íródott, .NET framework 2.0 alatt. A szimulációt egy T2300 Intel processzort (1.66 GHz) és 2 GB RAM memóriát tartalmazó PC-n futtattuk. Először az elkészített szimulációs program validálását kellett elvégezzük.

A validálási folyamathoz a szimulációs program segítségével a válaszidőket exponenciális eloszlású modell esetén számítottuk ki, melyet már össze tudtunk hasonlítani az analitikus eredményekkel, melyeket a (4.10) képlet alapján számolhatunk. A szimulációs és analitikus eredmények exponenciális eloszlás estén a 6.1 táblázatban találhatóak. Amint látható a válaszidők nagyon közel vannak egymáshoz, értékük az első 4 tizedesjegyig megegyeznek.

Az approximációs eljárás validálásához a következő eloszlásokat használtuk: (Lásd [34])

- Abban az esetben ha a relatív szórásnégyzet $0 < c_X^2 < 1$ akkor egy $E_{k-1,k}$ általánosított Erlang-eloszlást használunk ami egy E_{k-1} és egy E_k Erlang-eloszlás keveréke, melynek a sűrűségfüggvénye:

$$f(t) = p * \mu^{k-1} * \frac{t^{k-2}}{(k-2)!} * e^{-\mu t} + (1-p) * \mu^k * \frac{t^{k-1}}{(k-1)!} * e^{-\mu t} \quad (6.52)$$

  ahol $0 \leq p \leq 1$, és $t \geq 0$.

  Ebben az esetben az $E_{k-1,k}$ eloszlás p (ill. $1-p$) valószínűséggel $k-1$ (ill. k) független μ paraméterű exponenciális eloszlás összege. Ha p illetve μ értékeit az alábbi mennyiségek alapján választjuk

$$p = \frac{1}{1+c_X^2}\left(kc_X^2 - \left(k\left(1+c_X^2\right) - k^2c_X^2\right)^{1/2}\right) \text{ és } \mu = \frac{k-p}{E(X)}$$

  akkor az $E_{k-1,k}$ eloszlás várható értéke $E(X)$ és relatív szórása c_X, valamint $\frac{1}{k} \leq c_X^2 < \frac{1}{k-1}$.

- Amennyiben $c_X > 1$ egy $H2(p_1; p_2; \mu_1; \mu_2)$ hyper-exponenciális eloszlást használunk, melynek sűrűségfüggvénye:

$$f(t) = p_1 * \mu_1 * e^{-\mu_1 t} + p_2 * \mu_2 * e^{-\mu_2 t} \quad (6.53)$$

  ahol $0 \leq p_1, p_2 \leq 1$, és $t \geq 0$.

Paraméterek	Analitikus eredm.	Szimuláció	Approx.	Eltérés
$\lambda = 20, \Lambda = 100$	0,425793	0,425706	0,425793	0,000087
$\lambda = 80, \Lambda = 100$	0,430135	0,430136	0,430135	0,000001

6.1. táblázat. **Szimuláció validálása**

Mivel a hyper-exponenciális eloszlást az első két momentuma nem határozza meg egyértelműen, a

$$\frac{p_1}{\mu_1} = \frac{p_2}{\mu_2}$$

kiegyensúlyozott várhatóérték feltételt használjuk. (lásd [34])

Amennyiben a H_2 eloszlás paramétereit az alábbi értékek alapján választjuk

$$p_1 = \frac{1}{2}\left(1 + \sqrt{\frac{c_X^2 - 1}{c_X^2 + 1}}\right), \quad p_2 = 1 - p_1,$$

és

$$\mu_1 = \frac{2p_1}{E(X)}, \quad \mu_2 = \frac{2p_2}{E(X)}.$$

akkor a H_2 hyper-exponenciális eloszlás várható értéke $E(X)$ és relatív szórása c_X.

A numerikus eredmények egyszerűbb átláthatósága végett a modellben szereplő minden sor esetében ugyanazt a relatív szórásnégyzet értéket használtuk.

A 6.2 táblázatban láthatóak a különböző paraméterekkel számított szimulációs és approximációs eredmények.

6.4. Konklúzió

A disszertáció jelen részében módosítottuk a 4. fejezetben bemutatott Proxy Cache szerver modellt úgy, hogy az exponenciális eloszlású érkezési

Érkezési intenzitás	c^2	Szimuláció	Approximáció	Eltérés
$\lambda = 20$ $\Lambda = 100$	0,1	0,423084	0,423129	0,000045
	0,8	0,425127	0,425189	0,000062
	1,2	0,423042	0,426328	0,003286
	1,8	0,421744	0,427979	0,006235
$\lambda = 80$ $\Lambda = 100$	0,1	0,423591	0,423974	0,000383
	0,8	0,428624	0,428725	0,000101
	1,2	0,425821	0,431507	0,005686
	1,8	0,424049	0,435869	0,01182

6.2. táblázat. **Approximációs eredmények**

folyamatok helyett GI folyamatot használunk, valamint az exponenciális kiszolgálási idők helyett tetszőleges G eloszlást használunk. Hogy megkapjuk a teljes válaszidőt a QNA approximációs eljárást használtuk. A szimulációs és approximációs eredmények a 6.2 táblázatban láthatóak.

Amennyiben $c^2 < 1$ az approximációs eljárással és a szimulációs programmal kapott válaszidők nagyon közeliek; legalább az első 3-4 tizedesjegyig megegyeznek. Abban az esetben, amikor $c^2 > 1$ a szimulációs és approximációs eljárással kapott válaszidők csak az első 2-3 tizedes jegyig egyeznek. Látható, hogy amikor a relatív szórásnégyzet $1,2$, a két eljárás közöti különbség $0,005686$, valamint ha a relatív szórásnégyzet $1,8$, a válaszidők közötti különbség nagyobb $(0,01182)$. Észrevehetjük, hogy nagyobb relatív szórásnégyzetet használva az approximáció pontossága csökken.

7 A heterogén forgalom hatása a Proxy Cache szerverek hatékonyságára

Jelen fejezetben a 4. fejezetben megismertetett modellt szeretnénk általánosítani úgy, hogy a felhasználóktól érkező igényeket két csoportba soroljuk a lekért információk, fájlok mérete alapján.

7.1. Több igény-osztályt tartalmazó sorbanállási hálózatok

Tekintsünk egy nyitott Jackson-hálózatot, melyben K darab sor található. Tételezzük fel, hogy a rendszerben lévő és oda érkező igények C különböző osztályba sorolhatóak. Minden c osztály nyitott λ_c össz érkezési intenzitással. Így a rendszerhez tartozó érkezési intenzitást a

$$\overline{\lambda} = (\lambda_1, \lambda_2, \cdots, \lambda_C)$$

vektor írja le. A k-adik sorhoz a rendszeren kívülről érkező, bármely osztályhoz tartozó folyamat legyen Poisson-folyamat $\Lambda_{c,k}$ paraméterrel. A $\Lambda_{c,k} = 0$ intenzitás azt jelenti, hogy a k-adik sorhoz nem érkezik kívülről c osztályhoz tartozó igény. Mivel egy nyitott rendszerről beszélünk, feltételezzük, hogy legalább egy $\Lambda_{c,j} > 0$.

Az i-edik csomópont által kiszolgált c osztályhoz tartozó igény a j-edik csomóponthoz továbbítódik $p_{c,ij}$ valószínűséggel, vagy elhagyja a rendszert

$$\left(1 - \sum_{j=1}^{K} p_{c,ij}\right)$$

valószínűséggel.

A j-edik sor kiszolgálási ideje μ_j paraméterű exponenciális eloszlás, valamint a sorhoz érkező c osztályhoz tartozó folyamat teljes érkezési intenzitása:

$$\lambda_{c,j} = \Lambda_{c,j} + \sum_{i=1}^{K} \lambda_{c,i} p_{c,ij} \tag{7.1}$$

minden $j = 1, 2, \cdots, K$ esetén.

Rendszerparaméterek: [27]

- *A c-edik osztály áteresztőképessége a k-adik csomópontnál:*

$$X_{c,k} = \lambda_c V_{c,k}, \tag{7.2}$$

  ahol $V_{c,k}$ a c osztályhoz és k-adik sorhoz tartozó látogatások száma, azaz

$$V_{c,k} = \frac{\lambda_{c,k}}{\lambda_c}. \tag{7.3}$$

- *A c-edik osztály teljes áteresztőképessége:*

$$X_c = \lambda_c. \tag{7.4}$$

- *A c osztály kihasználtsága a k-adik csomópontnál:*

$$U_{c,k} = X_{c,k} S_{c,k} = \lambda_c D_{c,k}, \tag{7.5}$$

  ahol $S_{c,k}$ a c osztályhoz tartozó átlagos kiszolgálási idő a k-adik sornál, valamint

$$D_{c,k} = V_{c,k} S_{c,k}. \tag{7.6}$$

- *A kihasználtság a k-adik csomópontnál:*

$$U_k = \sum_{c=1}^{C} U_{c,k}. \tag{7.7}$$

- *A c osztályhoz tartozó átlagos igényszám a k-adik csomópontnál:*

$$N_{c,k} = \frac{U_{c,k}}{1 - \sum_{j=1}^{C} U_{j,k}}. \tag{7.8}$$

- *A c osztályhoz tartozó átlagos válaszidő a k-adik csomópontnál:*

$$T_{c,k} = \frac{D_{c,k}}{1 - \sum_{j=1}^{C} U_{j,k}}. \tag{7.9}$$

- *A c osztályhoz tartozó átlagos igény szám a rendszerben:*

$$N_c = \sum_{j=1}^{K} N_{c,j}. \tag{7.10}$$

- *A c osztályhoz tartozó átlagos válaszidő a rendszerben:*

$$T_c = \sum_{j=1}^{K} T_{c,j}. \tag{7.11}$$

- *Egy tetszőleges igény átlagos válaszideje a rendszerben:*

$$T = \sum_{c=1}^{C} \frac{T_c X_c}{X}, \tag{7.12}$$

ahol

$$X = \sum_{c=1}^{C} X_c. \tag{7.13}$$

7.2. Módosított Proxy Cache szerver modell

A következő részben részletesen bemutatjuk a 4. fejezetben ismertetett modellben végrehajtott módosításokat, melyek segítségével vizsgálni tudjuk a heterogén fájlok hatását.

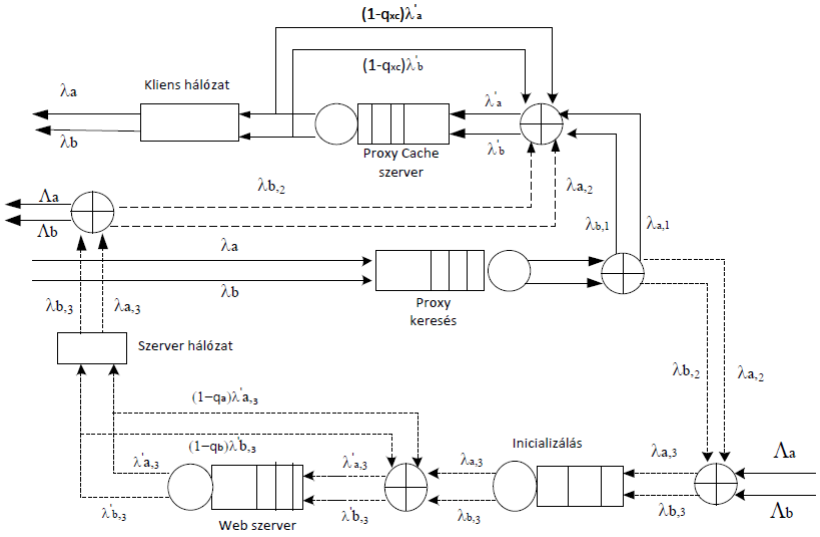
A kliensek által keresett fájlokat két osztályba soroljuk a méretük alapján. Amennyiben a fájl mérete az átlagosnál nagyobb az a osztályba soroljuk, míg ellenkező esetben amikor a fájl mérete kicsi "normál" fájlról beszélünk és a b osztályba soroljuk. Mindkét osztályba tartozó igény esetén először megvizsgáljuk, hogy a fájl megtalálható-e a Proxy Cache szerveren vagy sem. Ezt a találati valószínűséget p_a illetve p_b -vel jelöljük az a illetve a b osztályhoz tartozó fájlok esetén. Amennyiben a keresett fájl megtalálható a Proxy Cache szerveren, akkor mindkét osztály esetén a fájl egy másolata azonnal továbbítódik a klienshez. Ellenkező esetben, amikor is a fájl nem található meg a Proxy Cache szerveren az igény továbbítódik a távoli Web szerverhez függetlenül az osztályától. Miután az igényelt fájl visszaérkezik a Proxy Cache szerverhez egy másolat továbbítódik a klienshez.

Az 7.1 ábra mutatja egy belső fájl lehetséges útját az igény indulásától egészen a fájl klienshez való megérkezéséig. Az ábrán az a illetve b index jelöli, hogy a keresett fájl az a vagy a b osztályhoz tartozik. Az ábrán és a fejezetben használt jelölések megtalálhatóak a 7.4 táblázatban.

Feltételezzük, hogy a belső a osztályhoz tartozó igények a Proxy Cache szerverhez λ_a, míg a b osztályhoz tartozó igények λ_b paraméterű Poisson-folyamat szerint érkeznek, valamint a Web szerverhez kívülről érkező igények Λ_a illetve Λ_b paraméterű Poisson-folyamat alapján érkeznek az a illetve b osztályhoz tartozó igények esetén.

Az egyenes vonal ($\lambda_{a,1}$ ill. $\lambda_{b,1}$) reprezentálja azt az esetet amikor a keresett fájl megtalálható a Proxy Cache szerveren. Szaggatott vonallal rajzolva jelöltük ($\lambda_{a,2}$ ill. $\lambda_{b,2}$) azon igények útját, melyek nem találhatóak meg a Proxy Cache szerveren, így ezek az igények továbbítódnak a távoli Web szerverhez. A $\lambda_{a,1}$ és $\lambda_{a,2}$ valamint $\lambda_{b,1}$ és $\lambda_{b,2}$ intenzitások értékei:

$$\lambda_{a,1} = p_a * \lambda_a \text{ és } \lambda_{b,1} = p_b * \lambda_b, \tag{7.14}$$



7.1. ábra. Proxy Cache szerver heterogén forgalmi modellje

$$\lambda_{a,2} = (1 - p_a) * \lambda_a \text{ és } \lambda_{b,2} = (1 - p_b) * \lambda_b. \tag{7.15}$$

A Web szerverhez érkező igények teljes intenzitása a Web szerver felé továbbított belső igények illetve a külső igények intenzitásának az összege, azaz

$$\lambda_{a,3} = \Lambda_a + \lambda_{a,2} \text{ és } \lambda_{b,3} = \Lambda_b + \lambda_{b,2}. \tag{7.16}$$

A 4. fejezetben tárgyaltak alapján a Web szerverhez érkező igényeknek át kell esniük egy egyszeri inicializálási folyamaton, melyet a 7.1 ábrán az "Inicializálás" csomópont szemléltet. Az egyszeri inicializáláshoz szükséges idő mindkét osztályhoz tartozó fájlok esetén:

$$\frac{1}{\frac{1}{I_s} - (\lambda_{a,3} + \lambda_{b,3})}. \tag{7.17}$$

A Web szerver valamint a Proxy Cache szerver karakterisztikáját meghatározó paraméterek B_s, Y_s, R_s valamint B_{xc}, Y_{xc}, R_{xc} rendre a szerver kimenő puffere, a statikus szerver idő valamint a dinamikus szerver arány (lásd a 3. fejezetet), alapján a Web szerver és a Proxy Cache szerver kiszolgálási intenzitása

$$\mu_{Web} = \frac{1}{Y_S + \frac{B_S}{R_S}}, \tag{7.18}$$

$$\mu_{PCS} = \frac{1}{Y_{xc} + \frac{B_{xc}}{R_{xc}}}. \tag{7.19}$$

Ha a keresett fájl nagyobb mint a Web szerver kimenő puffere, akkor egy visszacsatolási ciklus kezdődik, mely addig tart míg a teljes fájl kiszolgálása be nem fejeződik. Legyen q_a illetve q_b annak a valószínűsége, hogy keresett a illetve b osztályhoz tartozó fájlt egyből sikerül továbbítani, ahol

$$q_a = \min\left(1, \frac{B_s}{F_a}\right) \tag{7.20}$$

illetve

$$q_b = \min\left(1, \frac{B_s}{F_b}\right). \tag{7.21}$$

Teljesen hasonlóan modellezhető a Proxy Cache szerver is, ahol a távozó folyamat visszacsatolásának a valószínűsége $1-q_{a,xc}$ illetve $1-q_{b,xc}$, ahol

$$q_{a,xc} = \min\left(1, \frac{B_{xc}}{F_a}\right) \tag{7.22}$$

az a osztályhoz tartozó fájl esetén, illetve

$$q_{b,xc} = \min\left(1, \frac{B_{xc}}{F_b}\right) \tag{7.23}$$

a b osztályhoz tartozó fájl esetén.

A belső a osztályhoz tartozó igények válaszidejét T_a^{xc} -vel valamint a belső b osztályhoz tartozó igények válaszidejét T_b^{xc}-vel jelöljük, melyeket a következő képletek határoznak meg:

$$\begin{aligned} T_a^{xc} = & \frac{1}{\frac{1}{I_{xc}} - (\lambda_a + \lambda_b)} \\ & + p_a * \left\{ \frac{\frac{1}{q_{a,xc}} * (Y_{xc} + \frac{B_{xc}}{R_{xc}})}{1 - \sum_{j=a}^{b} \frac{\lambda_j}{q_j}(Y_{xc} + \frac{B_{xc}}{R_{xc}})} + \frac{F_a}{N_c} \right\} \\ & + (1 - p_a) * \left\{ \frac{1}{\frac{1}{I_s} - (\lambda_{a,3} + \lambda_{b,3})} + \frac{\frac{1}{q_a} * (Y_s + \frac{B_s}{R_s})}{1 - \sum_{j=a}^{b} \frac{\lambda_{j,3}}{q_j}(Y_s + \frac{B_s}{R_s})} + \frac{F_a}{N_s} \right. \\ & \left. + \frac{\frac{1}{q_{a,xc}} * (Y_{xc} + \frac{B_{xc}}{R_{xc}})}{1 - \sum_{j=a}^{b} \frac{\lambda_j}{q_{j,xc}}(Y_{xc} + \frac{B_{xc}}{R_{xc}})} + \frac{F_a}{N_c} \right\}, \end{aligned} \tag{7.24}$$

és

$$\begin{aligned} T_b^{xc} = & \frac{1}{\frac{1}{I_{xc}} - (\lambda_a + \lambda_b)} \\ & + p_b * \left\{ \frac{\frac{1}{q_{b,xc}} * (Y_{xc} + \frac{B_{xc}}{R_{xc}})}{1 - \sum_{j=a}^{b} \frac{\lambda_b}{q_{b,xc}}(Y_{xc} + \frac{B_{xc}}{R_{xc}})} + \frac{F_b}{N_c} \right\} \\ & + (1 - p_b) * \left\{ \frac{1}{\frac{1}{I_s} - (\lambda_{a,3} + \lambda_{b,3})} + \frac{\frac{1}{q_b} * (Y_s + \frac{B_s}{R_s})}{1 - \sum_{j=a}^{b} \frac{\lambda_{j,3}}{q_j}(Y_s + \frac{B_s}{R_s})} + \frac{F_b}{N_s} \right. \\ & \left. + \frac{\frac{1}{q_{b,xc}} * (Y_{xc} + \frac{B_{xc}}{R_{xc}})}{1 - \sum_{j=a}^{b} \frac{\lambda_b}{q_{b,xc}}(Y_{xc} + \frac{B_{xc}}{R_{xc}})} + \frac{F_b}{N_c} \right\}. \end{aligned} \tag{7.25}$$

Így a teljes válaszidő:

$$T_{xc} = \frac{\lambda_a}{\lambda_a + \lambda_b} * T_a^{xc} + \frac{\lambda_b}{\lambda_a + \lambda_b} * T_b^{xc}. \tag{7.26}$$

A T_a^{xc} az a osztályhoz tartozó fájlok esetében egy belső igény átlagos válaszideje. Ennek a kiszámításához a hálózati modellünket három részhálózatra bontjuk. Ennek megfelelően a T_a^{xc} válaszidő három részből tevődik össze. Az első rész annak az időtartama, míg eldől, hogy a keresett a osztályú fájl megtalálható-e a Proxy Cache szerveren vagy sem. Ez a (7.9) képletből adódik, ahol I_{xc} az átlagos kiszolgálási idő. (Lásd 4. fejezetet.) A képlet második tagja annak a válaszideje, amikor a keresett fájl megtalálható a Proxy Cache szerveren, melynek sebessége, azaz kiszolgálási intenzitása a (3.5) alapján $Y_{xc} + \frac{B_{xc}}{R_{xc}}$. Ennek az esetnek a valószínűsége p_a. A második tag szintén két részből tevődik össze. Az első a (7.9) képlet alapján a Proxy Cache szervernél eltöltött idő, ahol a szerverhez érkező a osztályú fájlok érkezési intenzitása $\lambda_a^{'} = \frac{\lambda_a}{q_{a,xc}}$. A képletrész második tagja, pedig a kliens hálózaton való áthaladási idő, mely a 3. fejezetben leírtak alapján $\frac{F_a}{N_c}$. A képlet harmadik tagja azt az esetet írja le, amikor a fájl nem található meg a Proxy Cache szerveren, ezért az igény továbbítódik a távoli Web szerverhez. Ennek az esetnek a valószínűsége $1 - p_a$. A képlet ezen része további öt tagból áll. Az első a (3.2) és (7.9) alapján az úgynevezett egyszeri inicializálási idő. A második tag az igény Web szervernél eltöltött ideje, ahol a Web szerverhez érkező igények intenzitása $\lambda_{a,3}^{'} = \frac{\lambda_{a,3}}{q_a}$. A harmadik és az ötödik tag a fájlnak a szerver illetve kliens hálózaton való átjutáshoz szükséges "utazási" idő. A negyedik tag a Proxy Cache szerverhez visszaérkező igény kliens felé továbbításának az ideje.

A (7.25) képlet egy b osztályhoz tartozó belső igény válaszidejét jelöli. Amenynyiben nem használunk Proxy Cache szervert, akkor a keresett válaszidők a következőképpen alakulnak:

$$T_a = \frac{1}{\frac{1}{I_s} - ((\lambda_a + \Lambda_a) + (\lambda_b + \Lambda_b))} + \frac{\frac{1}{q_a} * (Y_s + \frac{B_s}{R_s})}{1 - \sum_{j=a}^{b} \frac{(\lambda_j + \Lambda_j)}{q_j}(Y_s + \frac{B_s}{R_s})} + \frac{F_a}{N_s} + \frac{F_a}{N_c}, \tag{7.27}$$

és

$$T_b = \frac{1}{\frac{1}{I_s} - ((\lambda_a + \Lambda_a) + (\lambda_b + \Lambda_b))} + \frac{\frac{1}{q_b} * (Y_s + \frac{B_s}{R_s})}{1 - \sum_{j=a}^{b} \frac{(\lambda_j + \Lambda_j)}{q_j}(Y_s + \frac{B_s}{R_s})} + \frac{F_b}{N_s} + \frac{F_b}{N_c}. \tag{7.28}$$

Így egy belső igény válaszideje Proxy Cache szerver nélkül:

$$T = \frac{\lambda_a}{\lambda_a + \lambda_b} * T_a + \frac{\lambda_b}{\lambda_a + \lambda_b} * T_b. \tag{7.29}$$

Megvizsgálva a (7.24)-(7.29) képleteket látható, hogy azokban az esetekben amikor valamelyik nevező nullához közelít a válaszidő a végtelenhez tart. Legyen $\lambda_b/\lambda_a = \Lambda_b/\Lambda_a = m$ a b illetve a osztályhoz tartozó igények érkezési intenzitásának a hányadosa. Így a válaszidő a végtelenhez közelít, amennyiben a lenti egyenletek közül az egyik teljesül.

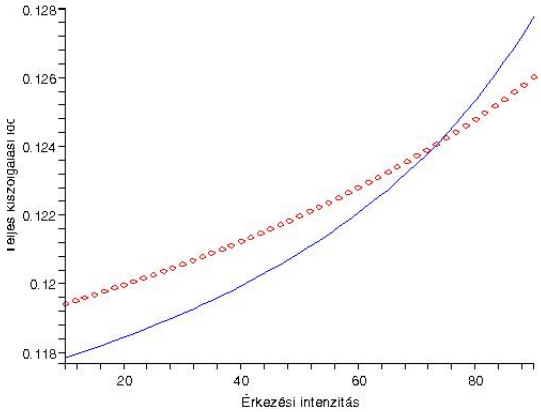
$$\begin{aligned}
\lambda &= \tfrac{1}{I_{xc}}, \\
\lambda_a &= \tfrac{q_{a,xc} q_{b,xc} R_{xc}}{(q_{b,xc} + m q_{a,xc})(Y_{xc} R_{xc} + B_{xc})}, \\
\lambda_b &= \tfrac{m q_{a,xc} q_{b,xc} R_{xc}}{(q_{b,xc} + m q_{a,xc})(Y_{xc} R_{xc} + B_{xc})}, \\
\lambda_{a,3} + \lambda_{b,3} &= \tfrac{1}{I_s}, \\
\lambda_{a,3} &= \tfrac{q_a q_b R_s}{(q_b + m q_a)(Y_s R_s + B_s)}, \\
\lambda_{b,3} &= \tfrac{m q_a q_b R_s}{(q_b + m q_a)(Y_s R_s + B_s)}, \\
\lambda + \Lambda &= \tfrac{1}{I_s}, \\
\lambda_a + \Lambda_a &= \tfrac{q_a q_b R_s}{(q_b + m q_a)(Y_s R_s + B_s)}, \\
\lambda_b + \Lambda_b &= \tfrac{m q_a q_b R_s}{(q_b + m q_a)(Y_s R_s + B_s)}
\end{aligned}$$

7.3. Numerikus eredmények

A következőekben vizsgált numerikus eredményekhez a használt szerver paraméterek a korábbi fejezetekben használtakkal megegyeznek. A számításokhoz a Web és Proxy Cache szerver paraméterek értékei [28] alapján: $I_s = I_{xc} = 0.004$ másodperc, $B_s = B_{xc} = 2000$ byte, $Y_s = Y_{xc} = 0.000016$ másodperc, $R_s = R_{xc} = 1.25$ Mbyte/másodperc, $N_s = 1544$ Kbit/másodperc, valamint $N_c = 128$ Kbit/másodperc. Az a illetve b osztályhoz tartozó fájlok méretét [28] alapján választottuk: $F_a = 7000$ byte, valamint $F_b = 1000$ byte. A fejezetben található grafikonokon pontozott vonallal ábrázoltuk a Proxy Cache szervert tartalmazó esetet, valamint egyenes vonallal a Proxy Cache szervert nem tartalmazó esetet.

7.3.1. A belső igények érkezési intenzitásának hatása a válaszidőre

A 7.2 grafikonon a válaszidőt a belső igények érkezési intenzitásának függvényeként ábrázoltuk. Ebben az esetben az a osztályú fájlok aránya az összes igény között 10 %, a külső igények érkezési intenzitása 100 kérés/másodperc, a Proxy Cache szerveren a találati valószínűségek rendre $p_a = p_b = 0.25$. A két osztályhoz tartozó fájl méretek pedig $F_a = 7000$ byte valamint $F_b = 1000$ byte. Amikor λ kisebb 75 kérés/másodpercnél, a válaszidő Proxy Cache szerver használatával nagyobb mint Proxy használata nélkül. Azaz ebben az esetben igen magas $\lambda > 75$ kell legyen a belső igények érkezési intenzitása, hogy megérje a Proxy Cache szerver üzemeltetése. A 7.3 grafikonon ugyanazokat a rendszer paramétereket használtuk, csak az a osztályhoz tartozó igények arányát növeltük meg 20%-ra. Mint ahogyan látható, ebben az esetben a válaszidők már $\lambda > 65$ igény/másodperc esetén alacsonyabbak Proxy Cache szerver használatával. A 7.4 grafikonon azt az esetet látjuk, amikor az a osztályú fájlok esetében a találati valószínűséget növeljük $p_a = 0.4$-re. A grafikonon használt többi paraméter értékei: az a osztály aránya = 20%, a külső érkezési intenzitás $\Lambda = 100$ igény/másodperc, a használt fájl méretek $F_a = 7000$ byte, $F_b = 1000$ byte, valamint a belső igények esetén a találati valószínűség $p_a = 0.4$ és $p_b = 0.25$. A grafikonon látható, hogy ilyen magas találati valószínűség esetén a válaszidők minden belső igény érkezési intenzitás

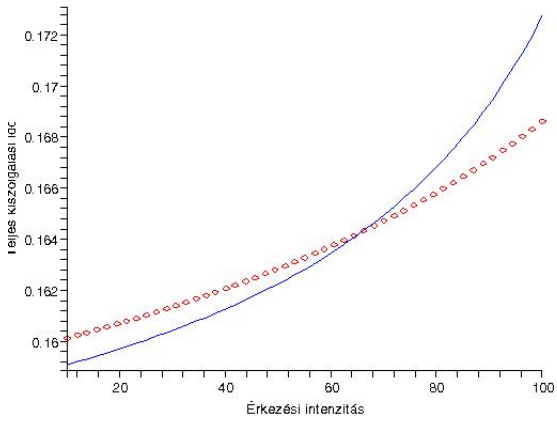


7.2. ábra. 10% a osztály, $\Lambda = 100$, $p_a = p_b = 0.25$, $F_a = 7000$ bytes, $F_b = 1000$ bytes

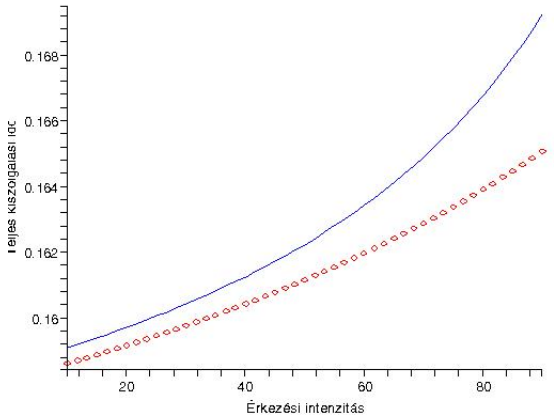
mellett alacsonyabbak Proxy Cache szerver használatával. Amennyiben csak az a osztályhoz tartozó fájlok találati valószínűségét csökkentjük, természetesen a Proxy Cache szerver hatékonysága drasztikusan romlik. Ezt láthatjuk a 7.5 grafikonon ahol a használt paraméterek megegyeznek a 7.4 grafikonon használt értékekkel, kivéve az a osztályú fájlok találati valószínűségét, ami $p_a = 0.15$. A grafikonok elemzésével látható, hogy a Proxy Cache szerver hatékonysága alacsonyabb találati valószínűség esetén, csak magas belső érkezési intenzitás mellett jár alacsonyabb válaszidőkkel. Viszont extrém magas találati valószínűség használatával ($p_a = 0.4$) a Proxy Cache szerver használata minden esetben alacsonyabb válaszidőket eredményez.

7.3.2. A külső igények érkezési intenzitásának hatása a válaszidőre

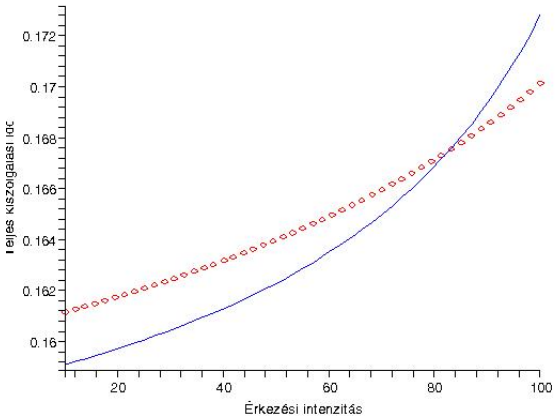
A következő grafikonok segítségével a külső igények hatását fogjuk megvizsgálni. A 7.6 grafikonon használt paraméterek: az a osztály aránya = 30%, a belső igények érkezési intenzitása $\lambda = 10$ igény/másodperc, a



7.3. ábra. 20% a osztály, $\Lambda = 100$, $p_a = p_b = 0.25$, $F_a = 7000$ bytes, $F_b = 1000$ bytes

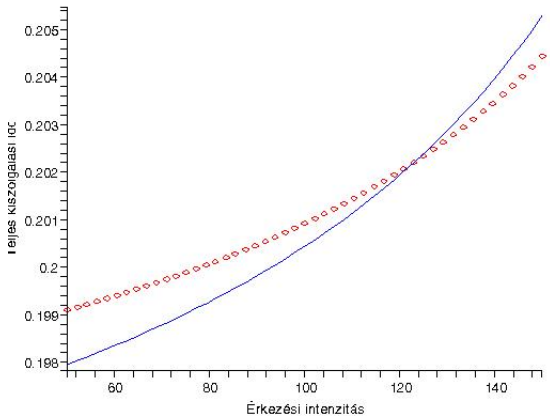


7.4. ábra. 20% a osztály, $\Lambda = 100$, $p_a = 0.4, p_b = 0.25$, $F_a = 7000$ bytes, $F_b = 1000$ bytes

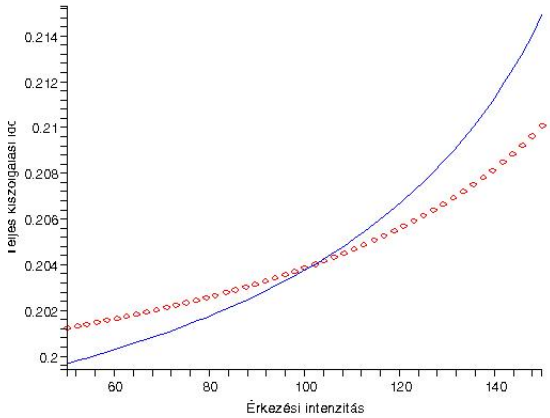


7.5. ábra. 20% a osztály, $\Lambda = 100$, $p_a = 0.15, p_b = 0.25$, $F_a = 7000$ bytes, $F_b = 1000$ bytes

használt fájl méretek $F_a = 7000$ byte, $F_b = 1000$ byte, valamint a Proxy Cache szerveren a találati valószínűségek rendre $p_a = p_b = 0.25$. Amint látható, ha a külső igények érkezési intenzitása $\Lambda > 125$ igény/másodperc alacsony belső érkezési intenzitás ($\lambda = 10$) és viszonylag alacsony találati valószínűség ($p_a = p_b = 0.25$) mellett is alacsonyabb válaszidőket kapunk Proxy Cache szerver használatával. A 7.7 grafikonon a használt paraméterek megegyeznek a 7.6 grafikon paramétereivel, csak a belső igények érkezési intenzitását növeltük $\lambda = 50$-re. Amint várható volt ebben az esetben már alacsonyabb külső érkezési intenzitás mellett is alacsonyabb válaszidőket kapunk Proxy Cache szerver használatával ($\Lambda > 105$). Megvizsgálva a 7.6 - 7.7 grafikonokat általánosságban elmondhatjuk, hogy a külső igények érkezési intenzitásának növelésével a válaszidők nőnek függetlenül a Proxy Cache szerver jelenlététől. Amennyiben a külső igények intenzitása elég nagy, viszonylag kis belső érkezési intenzitás és találati valószínűség esetén is alacsonyabb válaszidőket kaphatunk Proxy Cache szerver használatával.



7.6. ábra. 30% a osztály, $\lambda = 10, p_a = p_b = 0.25$, $F_a = 7000$ bytes, $F_b = 1000$ bytes



7.7. ábra. 30% a osztály, $\lambda = 50, p_a = p_b = 0.25$, $F_a = 7000$ bytes, $F_b = 1000$ bytes

7.3.3. A fájlméret hatása a válaszidőre

A 7.8-7.9 grafikonokon a teljes válaszidőt az a osztályhoz tartozó fájl méretének függvényeként, míg a 7.10 grafikonon a b osztályhoz tartozó fájl méretének függvényeként ábrázoljuk. A 7.8 grafikonon a használt paraméterek értékei: az a osztály aránya = 40%, a belső igények érkezési intenzitása $\lambda = 50$ igény/másodperc, a külső igények érkezési intenzitása $\Lambda = 100$ igény/másodperc, a használt b osztályhoz tartozó fájlok mérete $F_b = 1000$ byte, valamint a Proxy Cache szerveren a találati valószínűségek rendre $p_a = p_b = 0.25$. Amint a grafikonon látható, az a osztályú fájlméret növelésével a válaszidők mind Proxy Cache szerver használatával, mind nélküle növekednek. Az ábrázolt két görbe csak $F_a > 15000$ byte esetén távolodik el egymástól. A részletesebb vizsgálat érdekében a kapott pontos válaszidőket a 7.1 illetve a 7.2 táblázatokban láthatjuk, ahol az a osztály aránya rendre 20 illetve 40 százalék. A 7.1 táblázatban megfigyelhetjük, hogy kisebb a osztályú fájlméret esetén a Proxy Cache szerver használata nagyobb válaszidőket eredményez. De amint a fájl mérete eléri a 12000 byte-ot a válaszidők alacsonyabbak lesznek Proxy Cache szerver használatával. A 7.2 táblázatban a teljes válaszidőket láthatjuk amikor az a osztály aránya 40%. Megfigyelhetjük, hogy magasabb a osztály arány mellett a válaszidők szintén magasabbak, viszont a Proxy Cache szerver használatának az előnye már kisebb fájl méretnél megmutatkozik ($F_a = 6000$ byte).

A 7.9 grafikonon az alap paraméterek változatlanok, egyedül az a osztály arányát növeltük meg 70%-ra. Megfigyelhetjük, hogy a grafikonon szereplő két görbe közötti eltérés számottevően nő a fájl méret növelésével, azaz magas a osztályú arány és nagy fájl méret használatával a Proxy Cache szerver használata kifizetődő. A 7.10 grafikonon a teljes válaszidőt a b osztályhoz tartozó fájl méretének függvényeként ábrázoljuk. A használt paraméterek: az a osztály aránya = 40%, a belső igények érkezési intenzitása $\lambda = 50$ igény/másodperc, a külső igények érkezési intenzitása $\Lambda = 100$ igény/másodperc, a használt a osztályú fájl mérete $F_a = 7000$ byte valamint a Proxy Cache szerveren a találati valószínűségek rendre $p_a = 0.25$ illetve $p_b = 0.35$.

Amint látható, Proxy Cache szerver használatával a használt paraméterek

Fájl méret(a osztály)	Txc	T	Eltérés
$F_a = 2000$	0.09446809706	0.09317433644	0.00129376062
$F_a = 4000$	0.1217843586	0.1207717535	0.0010126051
$F_a = 6000$	0.1491608228	0.1484324701	0.0007283527
$F_a = 8000$	0.1766073005	0.1761686716	0.0004386289
$F_a = 10000$	0.2041361569	0.2039958857	0.0001402712
$F_a = 12000$	0.2317632395	0.2319342123	-0.0001709728
$F_a = 14000$	0.2595092442	0.2600101405	-0.0005008963
$F_a = 16000$	0.2874017905	0.2882593062	-0.0008575157
$F_a = 18000$	0.3154786670	0.3167308131	-0.0012521461

7.1. táblázat. **Fájlméret hatása a válaszidőre, az a osztály aránya 20%**

Fájl méret(a osztály)	Txc	T	Eltérés
$F_a = 2000$	0.1077452991	0.1067106059	0.0010346932
$F_a = 4000$	0.1624380248	0.1619687395	0.0004692853
$F_a = 6000$	0.2174133590	0.2175321551	-0.0001187961
$F_a = 8000$	0.2727864463	0.2735464100	-0.0007599637
$F_a = 10000$	0.3287558693	0.3302670825	-0.0015112132
$F_a = 12000$	0.3856999610	0.3881879423	-0.0024879813
$F_a = 14000$	0.4444483664	0.4484027240	-0.0039543576
$F_a = 16000$	0.5072639578	0.5139139582	-0.0066500004
$F_a = 18000$	0.5833834923	0.5970200201	-0.0136365278

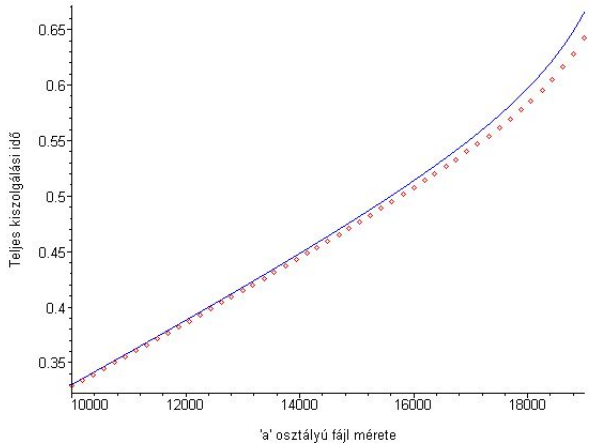
7.2. táblázat. **Fájlméret hatása a válaszidőre, az a osztály aránya 40%**

Az a osztály aránya	Txc	T	Eltérés
10	0.1219838832	0.1209124357	0.0010714475
20	0.1628746062	0.1622902553	0.0005843509
30	0.2038840246	0.2037976778	0.0000863468
40	0.2450404415	0.2454704817	-0.0004300402
50	0.2863831749	0.2873589491	-0.0009757742
60	0.3279687079	0.3295360446	-0.0015673367
70	0.3698815197	0.3721118312	-0.0022303115
80	0.4122543652	0.4152604812	-0.0030061160
90	0.4553092755	0.4592749103	-0.0039656348

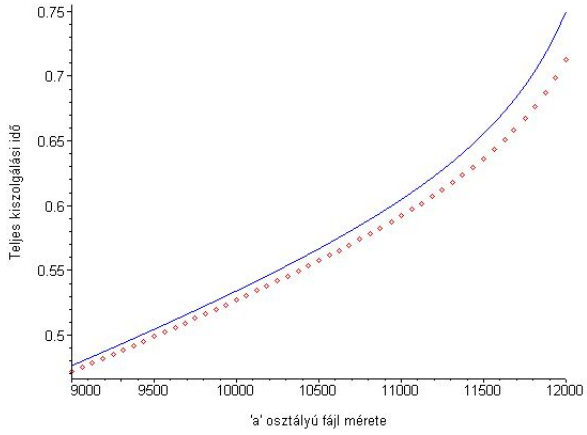
7.3. táblázat. **Az a osztály arányának hatása a válaszidőre**

mellett a válaszidők végig kisebbek mint Proxy Cache szerver használata nélkül. Megfigyelhetjük, hogy a "normál" - azaz b osztályú fájl mérete 1000-2000 byte-os intervallumban lényegesen nem befolyásolja a $T_{xc} - T$ különbséget.

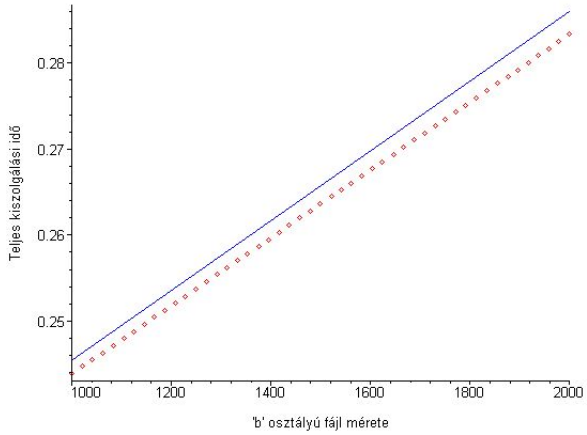
A 7.3 táblázatban az a osztályhoz tartozó fájlok arányának hatását láthatjuk a válaszidőre. A használt paraméterek: a belső igények érkezési intenzitása $\lambda = 50$ igény/másodperc, a külső igények érkezési intenzitása $\Lambda = 100$ igény/másodperc, a használt a illetve b osztályú fájl méretek $F_a = 7000$ byte illetve $F_b = 1000$ byte valamint a Proxy Cache szerveren a találati valószínűségek rendre $p_a = 0.25$ illetve $p_b = 0.25$. Láthatjuk, hogy az a osztályhoz tartozó tartalom arányának növelésével a válaszidők nőnek függetlenül attól, hogy installáltunk-e Proxy Cache szervert vagy sem. Valamint megfigyelhetjük, hogy az a osztályhoz tartozó fájlok arányának növelésével a különbség a két válaszidő között $(T_{xc} - T)$ egyre kisebb és 40% fölötti a osztályú tartalom esetén a használt paraméter értékek mellett, Proxy Cache szerver használatával már alacsonyabb válaszidőket kapunk, mint Proxy Cache szerver használata nélkül.



7.8. ábra. 40% a osztály, $\lambda = 50$, $\Lambda = 100, p_a = 0.25, p_b = 0.25$, $F_b = 1000$ bytes



7.9. ábra. 70% a osztály, $\lambda = 50$, $\Lambda = 100, p_a = 0.25, p_b = 0.25$, $F_b = 1000$ bytes



7.10. ábra. 40% a osztály, $\lambda = 50$, $\Lambda = 100, p_a = 0.25, p_b = 0.35$, $F_a =$ 7000 bytes

7.4. táblázat. **Heterogén forgalomi modell paraméterei**

λ_a:	belső a osztályú igények érkezési intenzitása
λ_b:	belső b osztályú igények érkezési intenzitása
Λ_a:	külső a osztályú igények érkezési intenzitása
Λ_b:	külső b osztályú igények érkezési intenzitása
F_a:	az a osztályhoz tartozó fájl mérete (byte-ban)
F_b:	a b osztályhoz tartozó fájl mérete (byte-ban)
p_a:	találati valószínűség az a osztályhoz tartozó fájlok esetén
p_b:	találati valószínűség a b osztályhoz tartozó fájlok esetén
B_{xc}:	a Proxy cache szerver kimenő puffere (byte-ban)
I_{xc}:	a Proxy Cache szerver keresési ideje (másodpercben)
Y_{xc}:	a PCS Statikus szerver ideje (másodpercben)
R_{xc}:	a dinamikus szerver arány a Proxy Cache szerveren (byte/másodperc)
N_c:	kliens hálózati sávszélesség (bit/másodperc)
B_s:	Web szerver kimenő puffere (byte-ban)
I_s:	Inicializálási idő (másodpercben)
Y_s:	a Web szerver statikus szerver ideje (másodperc)
R_s:	a Web szerver dinamikus szerver aránya (byte/másodperc)
N_s:	kliens hálózati sávszélesség (bit/másodperc)

8 Összefoglaló

A disszertációban a Proxy Cache szerverek hatékonyságát vizsgáltam meg részletesen. A 2. fejezetben rövid, lényegretörő ismertetőt adtam a sorbanállási rendszerek és hálózatok elméleti hátteréről, majd a 3. fejezetben ismertettem a Slouthber által felállított [32] Web szerver modellt.

Proxy Cache szervert használva, ha egy fájlt le akarunk tölteni egy távoli Web szerverről, először meg kell vizsgálni, hogy a keresett fájl megtalálható-e a Proxy Cache szerveren. Ennek a valószínűségét p-vel jelöljük. Amennyiben a keresett dokumentum megtalálható a Proxy Cache szerveren, egy másolat a fájlról azonnal továbbítódik a felhasználónak. Amennyiben a dokumentum nem található meg a Proxy Cache szerveren, az igény továbbítódik a távoli Web szerverhez. A dokumentum a Web szerverről először a Proxy Cache szerverre érkezik vissza, ahonnan egy másolat a fájlról azonnal a felhasználóhoz kerül. Az eredeti példány tárolódik a Proxy Cache szerveren, így a későbbiekben elérhető lesz a felhasználók számára.

A 4. fejezetben részletesen bemutattam az általunk általánosított Proxy Cache szerver modellt, mely egy tetszőleges igény útját ábrázolja a felhasználótól kiindulva egészen a visszaérkezésig. Feltételeztem, hogy az igények a Proxy Cache szerverhez λ paraméterű Poisson-folyamat szerint érkeznek, és a Web szerverhez kívülről érkező igények Λ paraméterű Poisson-folyamat alapján érkeznek, valamint mind a Proxy Cache szerver mind pedig a Web szerver kiszolgálási ideje exponenciális eloszlású valószínűségi változó. Kiszámítottam egy tetszőleges belső igény válaszidejét Proxy Cache szerver használata esetén, valamint Proxy Cache szerver használata nélkül. Így viszgálni tudtam a Proxy Cache szerver hatékonyságát különböző paraméterértékek mellett.

Tanulmányoztam a Proxy Cache szerver hatékonyságát a belső valamint külső érkezési intenzitások függvényében, valamint vizsgáltam a keresett fájl méretének és a Proxy Cache szerver találati valószínűségének hatását a válaszidőre. Megállapítottam, hogy a fájlméret növelésével a Proxy Cache szerver hatékonysága javul, azaz Proxy Cache szerver használatával kisebb válaszidőket kapunk mint Proxy Cache szerver használata nélkül. Szintén növelhető a Proxy Cache szerver hatékonysága a találati valószínűség valamint a külső és belső érkezési intenzitás növelésével.

Az 5. fejezetben a korábban ismertetett Proxy Cache szerver modellt általánosítottam úgy, hogy egy méginkább valósághű modellt kapjunk. A korábbiakban mind a Proxy Cache szerver, mind pedig a távoli Web szerver megbízhatóak voltak, most pedig feltételeztem, hogy egyik sem megbízható, azaz bármelyikük elromolhat. Az általánosítással az volt a célom, hogy megvizsgáljam a szerverek meghibásodásának hatását a rendszerparaméterekre.

A vizsgált Markov lánc állapottere, mely a módosított modellt leírja túl nagy, az egyensúlyi egyenlet felírása és ezek megoldása túl bonyolult lenne. Ezért a MOSEL (Modeling, Specification and Evaluation Language) [6] programcsomagot használtam a modell leírására valamint a rendszerjellemzők kiszámítására. Feltételeztem, hogy a Proxy Cache szerver és a Web szerver meghibásodhat a $(t, t+dt)$ intervallumban $\delta_{pcs}dt + o(dt)$ valamint $\delta_{web}dt + o(dt)$ valószínűséggel ha szabadok, valamint $\gamma_{pcs}dt + o(dt)$ és $\gamma_{web}dt + o(dt)$ valószínűséggel ha foglaltak. Ha a Proxy Cache szerver vagy a Web szerver foglalt állapotban romlanak el, akkor a megszakadt igény feldolgozása a javítás befejezése után folytatódik. A javítási idő exponenciális eloszlású $1/\nu_{pcs}$ és $1/\nu_{web}$ átlaggal. Ha a szerverek közül valamelyik elromlik két különböző esetet különböztettem meg:

- **Blokkolt eset**: a szerver meghibásodása alatt nem érkezik új igény a szerverhez.
- **Nem blokkolt eset**: a szerver meghibásodása alatt is érkezhetnek újabb igények a szerverhez.

A MOSEL programcsomagot használva meghatároztam a válaszidőket. Megvizsgáltam a Proxy Cache szerver és a Web szerver különböző meg-

hibásodási és javítási paramétereinek hatását a válszidőre mind foglalt mind pedig szabad szerverek esetében.

A 6. fejezetben az érkezési folyamat egy úgynevezett "GI - General interarrival" folyamat, melyet az érkezési időközök várható értékével és a relatív szórásnégyzetével (c^2) jellemzünk, valamint a kiszolgálási idő bármilyen általános eloszlású lehet. Az approximáció [12] használatához a következő feltételeknek kell tejesülniük:

- Az érkezési folyamat úgynevezett "felújítási" folyamat kell legyen, azaz az érkezési időközök független, azonos eloszlású valószínűségi változók.
- A kiszolgálási idők valószínűségi változója bármilyen általános eloszlású lehet.
- Adott az érkezési folyamat inetnzitása λ_A, valamint az érkezési folyamat relatív szórásnégyzete (c_A^2).
- Adott a kiszolgálási idő várható értéke τ_S, valamint a kiszolgálási idő relatív szórásnégyzete (c_S^2).
- Az azonnali visszacsatolást amikor egy sor távozó folyamata vissza van irányítva egyből ugyanahhoz a sorhoz, külön kell vizsgálni.

Ez az aproximáció olyan algoritmusokat szolgáltat, melyekkel modellezhetjük az általános hálózati folyamatokat, mint például a forgalom egyesítést, a sortól való távozást, valamint a forgalom szétválását.

Minden esetben, a részletes számítások előtt a modellt módosítanunk kell, hogy eliminálhassuk az azonnali visszacsatolásokat. Ezt a módosítást az érintett sor kiszolgálási idejének megváltoztatásával végezzük.

A szükséges kalkulációk eredményeképp az aproximáció segítségével megkapjuk a szükséges rendszerjellemzőket (átlagos sorhossz, átlagos várakozási idő, stb.), mind a sorokra, mind pedig az egész hálózatra vonatkozóan.

Módosítottam a Proxy Cache szerver modellt, hogy alkalmazni lehessen rá a GI/G/1 approximációt. Újrakalkuláltam a szükséges rendszerjellemzőket, hogy megkapjam a keresett válaszidőket. Az így kapott eredmények validálására egy szimulációs programot készítettem, mely segítségével ellenőrizhető az approximáció helyessége. Megvizsgáltam két különböző esetet. Abban az esetben amikor a használt relatív szórásnégyzetek egynél kisebbek illetve azt az esetet amikor a használt relatív szórásnégyzetek egynél nagyobbak.

Megállapítottam, hogy amennyiben a használt relatív szórásnégyzet egynél kisebb az approximációval számított válaszidők az első 3-4 tizedesjegyig megegyeznek a szimulációs eredményekkel. Amikor a relatív szórásnégyzet egynél nagyobb az approximációval kapott válaszidők csak az első 2-3 tizedesjegyig egyeznek.

A 7. fejezetben módosítottam az eredeti Proxy cache szerver modellt, hoggy vizsgálni lehessen a heterogén forgalom hatását a Proxy Cache szerverek hatékonyságára. A kliensek által keresett fájlokat méretük alapján két osztolybá soroltam. Az átlagosnál nagyobb méretű fájlok az a, míg a kis méretű, úgynevezett "normál" fájlok a b osztályba kerülnek. Mindkét osztályba tartozó igény esetén először megvizsgáljuk, hogy a fájl megtalálható-e a Proxy Cache szerveren vagy sem. Ezt a találati valószínűséget p_a illetve p_b -vel jelöljük az a illetve b osztályba tartozó fájlok esetén. Amennyiben a keresett fájl megtalálható a Proxy Cache szerveren, akkor mindkét osztály esetén a fájl egy másolata azonnal továbbítódik a klienshez. Ellenkező esetben, amikor is a fájl nem található meg a Proxy Cache szerveren az igény továbbítódik a távoli Web szerverhez függetlenül az osztályától. Miután az igényelt fájl visszaérkezik a Proxy Cache szerverhez egy másolat továbbítódik a klienshez.

Feltételeztem, hogy mindkét osztályhoz tartozó igények a Proxy Cache szerverhez Poisson-folyamat szerint érkeznek, és a Web szerverhez kívülről érkező igények szintén Poisson-folyamat alapján érkeznek, valamint mind a Proxy Cache szerver mind pedig a Web szerver kiszolgálási ideje független exponenciális eloszlású valószínűségi változó. Kiszámítottam egy tetszőleges belső igény válaszidejét Proxy Cache szerver használata esetén, valamint Proxy Cache szerver használata nélkül. Így viszgálni tudtam a Proxy Cache szerver használatát különböző paraméterértékek mellett.

Megvizsgáltam a Proxy Cache szerver hatékonyságát a belső valamint külső érkezési intenzitások függvényében, valamint vizsgáltam az a illetve b osztályhoz tartozó fájlok méretének és az a illetve b osztály arányának hatását a válaszidőre. Megmutattam, hogy mind a belső mind pedig a külső érkezési intenzitás növelésével a válaszidők nőnek, függetlenül a Proxy Cache szerver jelenlététől. Amennyiben az a osztályos kérések arányát növeljük a válaszidők szintén nőnek, valamint magas a osztály arányt használva már alacsonyabb érkezési intenzitás esetén is megéri a Proxy Cache szerver használata. Alacsony a osztály arány, alacsony érkezési intenzitás és alacsony találati valószínűség használatával Proxy Cache szerverrel magasabb válaszidőket kapunk mint nélküle.

9 Conclusion

The first part of the dissertation (see Chapter 2) was devoted to give a small overview on the queueing and queueing network theory. In Chapter 3. I described a Web server model created by Slouthber [32].

Using Proxy Cache server, if any information or file is requested to be downloaded, first it is checked whether the document exists on the Proxy Cache server or not. (We denote the probability of this existence by p). If the document can be found on the Proxy Cache server then its copy is immediately transferred to the user. In the opposite case the request will be sent to the remote Web server. After the requested document arrived back to the Proxy Cache server then a copy of it is delivered to the user.

In Chapter 4. I described a Proxy Cache server model, which illustrates the path of a request starting from the user and ending with the return of the answer to the user. We assume that the requests of the Proxy Cache server users arrive according to a Poisson process with rate λ, and the external requests arrive to the Web server according to a Poisson process with rate Λ, respectively, and the Proxy Cache server and Web server have an exponentially distributed service time distribution. I have calculated the overall response time with and without Proxy Cache server. I analyzed how various factors affect the performance of a Proxy Cache server. These factors include the arrival rates of requests, the "cache hit rate" probability and the external arrival rates. I also examine the effect of the file size. I noticed that, when the arrival rate of requests increases, then the response times increase as well, regardless of the existence of a Proxy Cache server. When we used a high visit rate with a high cache hit rate probability, then the response time gap was more significant between the cases with and without a Proxy Cache server, so in this case we get smaller response time using Proxy Cache server. Increasing the visit rate for the external users, the difference between response time with and without a Proxy

Cache server was smaller and smaller until this difference vanished and the existence of a Proxy Cache server resulted lower response times. We can increase the performance of Proxy Cache server using higher file size or higher cache hit rate probability.

In chapter 5. I generalize the performance model of the Proxy Cache server using a more realistic case when the Proxy Cache server and the remote Web server are unreliable. Our aim is to illustrate graphically the effect of the non-reliability of both Proxy Cache servers and Web servers on the steady state system measures.

Since the state space of the describing Markov chain is very large, it is dificult to calculate the system measures in the traditional way of writing down and solving the underlying steady-state equations. To simplify this procedure we used the software tool MOSEL (Modeling, Specification and Evaluation Language), see [6], to formulate the model and to obtain the performance measures. The Proxy Cache server and the Web server can fail during the interval $(t, t+dt)$ with probability $\delta_{pcs}dt+o(dt)$ and $\delta_{web}dt+o(dt)$ if they are idle, and with probability $\gamma_{pcs}dt+o(dt)$ and $\gamma_{web}dt+o(dt)$ if they are busy, respectively. If the Proxy Cache server or the Web server fails in busy state, it continues servicing the interrupted request after it has been repaired. The repair time is exponentially distributed with a finite mean $1/\nu_{pcs}$ and $1/\nu_{web}$. If one of the servers fails two different cases can be treated:

- **Blocked case**: during the CPU is down, no new requests may come to the server buffer.
- **Unblocked case**: the new requests can fill the server buffer during the breakdown, until it is full.

All the times involved in the model are assumed to be mutually independent of each other. Using Mosel I had to calculate the overall response time. I examined the effect of the non-reliability of both Proxy Cache servers and Web servers on the steady state system measures and the difference in the performance using blocked and intelligent sources.

In Chapter 6. we assume that the arrival process is a general (GI) arrival process characterised by a mean arrival rate and a squared coefficient of variation (SQV) of the inter-arrival time, and the service time may have

any general distribution. In order to apply the GI/G/1 approximation we assume the following:

- The arrival process to a network node is renewal, so the arrival intervals are independent, identically distributed random variables.
- The service time may have any general distribution.
- We know the parameters of the arrival process: λ_A - the mean arrival rate and c_A^2 - the SQV of the inter-arrival time.
- We know the parameters of the service time τ_S - the mean service time, and c_S^2 - the SQV of the service time.
- Immediate feedback, where a fraction of the output of a particular queue enters the queue once again, needs special treatment.

This approximation contains procedures required for modeling of the basic network operations of merging, departure and splitting, arising due to the common sharing of the resources and routing decisions in the network. Before the detailed analysis of the queueing network is done, the method first removes immediate feedback in a queue by suitably modifying its service time. The approximation provide performance measures (i.e. mean queue lengths, mean waiting times, etc.) for both per-queue and per-network.

I modified the performance model of Proxy Cache servers to get a more powerful variant when the inter-arrival times and the service times are generally distributed. In this case we can use the GI/G/1 approximation to obtain the overall response time. I recalculated the basic performance parameters of the modified performance model using the approximation method. The accuracy of the new model is validated by means of a simulation study over an extended range of test cases. I studied two different cases. When the SQV < 1 the overall response time obtained by approximation is very close to the response time obtained by simulation; they are the same at least up to the 3rd-4th decimal digit. In case when the SQV > 1 the response times are the same only to 2nd-3rd decimal digit. So, using greater SQV the approximation error is greater.

The focus of the Chapter 7. is to examine the performance behavior of Proxy Cache servers when we use heterogeneous traffic. In this thesis we describe the multi-class queuing network model of the Proxy Cache server, where we separate the requests in two classes by virtue of their size. If the size of the requested document is greater than average we put it into class a. In the opposite case, when the size of the requested file is small we put it into class b. In booth cases first it is checked whether the document (class a or class b) exists on the Proxy Cache server or not. We denote the probability of this existence by p_a in case of class a and by p_b in case of class b. If the document can be found on the Proxy Cache server then its copy is immediately transferred to the user. In the opposite case the request will be sent to the remote Web server. After the requested document arrived back to the Proxy Cache server then a copy of it is delivered to the user.

We assume that the requests of the Proxy Cache server users for both classes arrive according to a Poisson process with rate λ_a and λ_b, and the external requests for booth classes arrive to the Web server according to a Poisson process with rate Λ_a and Λ_b, respectively, and the Proxy Cache server and Web server have an exponentially distributed service time distribution. I have calculated the overall response time with and without a Proxy Cache server. I analyzed how various factors affect the performance of a Proxy Cache server when we use heterogeneous traffic. In general when the arrival rate of requests increases, then the response time increases as well regardless of the existence of a Proxy Cache server. Increasing the percentage of the class a the response time increases too. When we use a higher percentage of the class a and we use a high arrival rate, then the response time gap is more significant between the cases with and without a Proxy Cache server. Using a low percentage of class a files, a low arrival rate and low cache hit rate probability we get higher response time in presence of a Proxy Cache server.

Irodalomjegyzék

[1] C. Aggarwal, J.L. Wolf, and P.S. Yu. Caching on the world wide web. *IEEE Transactions on Knowledge and Data Engineering*, 11:94–107, 1999.

[2] V.A.F. Almeida, J.M. de Almeida, and C.S. Murta. Performance analysis of a www server. *Proceedings of the 22nd International Conference for the Resource Management and Performance Evaluation of Enterprise Computing Systems*, 13, 1996.

[3] Allen A.O. *Probability, statistics, and queueing theory : with computer science applications.* Academic Press, New York, 1997.

[4] M.A. Arlitt and C.L. Williamson. Internet web servers: workload characterization and performance implications. *IEEErACM Transactions on Networking*, 5:631–645, 1997.

[5] I. Atov. Qna inverse model for capacity provisioning in delay constrained ip networks. Centre for Advanced Internet Architectures. Technical Report, 2004.

[6] K. Begain, G. Bolch, and H. Herold. *Practical performance modeling, application of the MOSEL language.* Kluwer Academic Publisher, Boston, 2001.

[7] T. Berczes. Approximation approach to performance evaluation of proxy cache server systems. *Annales Mathematicae et Informaticae*, 36:15–28, 2009.

[8] T. Berczes, G. Guta, G. Kusper, W. Schreiner, and J. Sztrik. Analyzing web server performance models with the probabilistic model checker prism. Technical report no. 08-17 in RISC Report Series, 2008.

[9] T. Berczes and J. Sztrik. Performance modeling of proxy cache servers. *Journal of Universal Computer Science*, 12:1139–1153, 2006.

[10] T. Berczes, J. Sztrik, and C.S. Kim. The impact of multimedia traffic on the performance of a proxy cache server. *Annales Univ. Sci. Budapest, Sect. Comp.*, 25:153–169, 2005.

[11] I. Bose and H.K. Cheng. Performance models of a firms proxy cache server. *Decision Support Systems and Electronic Commerce*, 29:45–57, 2000.

[12] S.K. Bose. *An introduction to queueing systems.* Kluwer Academic/Plenum Publishers, New York, 2002.

[13] L. Breuer and D. Baum. *An Introduction to Queueing Theory and Matrix-Analytic Methods.* Springer Verlag, Netherlands, 2005.

[14] S.J. Caughey, D.B. Ingham, and M.C. Little. Flexible open caching for the web. *Computer Networks and ISDN Systems*, 29:1007–1017, 1997.

[15] X. Chao, M. Miyazawa, and M. Pinedo. *Queueing Networks : Customers, Signals and Product Form Solutions.* John Wiley & Sons, New York, 1999.

[16] H. Chen and D. Yao. *Fundamentals of Queuing Networks: Performance, Asymptotics, and Optimization.* Springer, New York, 2001.

[17] R. Disney and P. Kiessler. *Traffic processes in queueing networks : a Markov renewal approach.* Johns Hopkins University Press, 1987.

[18] T. V. Do and R. Chakka. Simulation and analytical approaches for estimating the performability of a multicast address dynamic allocation mechanism. *Simulation Modelling Practice and Theory*, 18(7):971–983, 2010.

[19] T. V. Do, U. R. Krieger, and R. Chakka. Performance modeling of an apache web server with a dynamic pool of service processes. *Telecommunication Systems*, 39(2):117–129, 2008.

[20] Tien Van Do. A new solution for a queueing model of a manufacturing cell with negative customers under a rotation rule. *Performance Evaluation*, 68(4):330 – 337, 2011. G-Networks and their Applications - G-Networks.

[21] V.T Do, R. Chakka, T. Le Nhat, and O. Gemikonakli. A new performance model for web servers. *Federation of European Simulation Societies*, pages 19–21, 2004.

[22] V.T Do, R. Chakka, T. Le Nhat, and U. Krieger. Performance modelling of a web server with a dynamic pool of service processes. *Center for Mathematics and Computer Science*, pages 19–21, 2006.

[23] G. Fayez. *Analysis of Computer and Communication Networks*. Springer, 2008.

[24] Bolch G., Greiner S., H. de Meer, and Trivedi K.S. *Queueing Networks and Markov Chains*. Wiley-Interscience, 2006.

[25] E. Gelenbe and G. Pujolle. *Introduction to Queueing Networks*. John Wiley & Sons, 1998.

[26] M. Kurcewicz, W. Sylwestrzak, and A. Wierzbicki. A filtering algorithm for web caches. *Computer Networks and ISDN Systems*, pages 2203–2209, 1998.

[27] E.D. Lazowska, J. Zahorjan, G.S. Graham, and K.C. Sevcik. *Quantitative System Performance*. Prentice Hall, 1984.

[28] D.A. Menasce and V.A.F. Almeida. *Capacity Planning for Web Performance: Metric, Models, and Methods*. Prentice Hall, 1998.

[29] Balsamo S. and Onvural R. Person V. de Nitto. Analysis of queueing networks with blocking. *International Series in Operations Research and Management Science*, 31.

[30] R. Sahner, K. Trivedi, and A. Puliafito. *An Example-Based Approach Using the SHARPE Software Package*. Kluwer Academic Publisher, 2002.

[31] P. Scheuermann, J. Shim, and R. Vingralek. A case for delayconscious caching of web documents. *Computer Networks and ISDN Systems*, 29:997–1005, 1997.

[32] L.P. Slothouber. A model of web server performance. *5th International World Wide Web Conference*, 1996.

[33] H. Takagi. *Stochastic analysis of computer and communication systems*. Elsevier Science Inc., New York, 1990.

[34] H.C. Tijms. *Stochastic Modelling and Analysis. A computational approuch*. John Wiley & Sons, New York, 1986.

[35] N.M. Van Dijk. *Queueing Networks and Product Forms: A Systems Approach*. John Wiley & Sons, 1993.

[36] W. Whitt. Approximations for the gi/g/m queue. *Productions and Operations Management*, 2:114–161, 1933.

[37] W. Whitt. Performance of the queueing network analyzer. *Computer Networks and ISDN Systems*, 62:2817–2843, 1983.

Publikációim:

- **[J] Folyóiratcikkek**:

J1 T. Berczes, J. Sztrik, Kim, C.S., The impact of multimedia traffic on the performance of a proxy cache server. *Annales Univ. Sci. Budapest, Sect. Comp.*, **25** (2005), 153-169

J2 T. Berczes and J. Sztrik, Performance Modeling of Proxy Cache Servers. *Journal of Universal Computer Science.*, **12** (2006), 1139–1153

J3 T. Berczes and J. Sztrik, Performance evaluation of proxy cache servers. *Híradástechnika.*, Selected Papers **LXI** (2006/1), 2-5

J4 T. Berczes, Approximation approach to performance evaluation of Proxy Cache Server systems. *Annales Mathematicae et Informaticae.*, **36** (2009), 15-28

J5 T. Berczes, G. Guta, G. Kusper, W. Schreiner and J. Sztrik, Comparing the Performance Modeling Environment MOSEL and the Probabilistic Model Checker PRISM for Modeling and Analysing Retrial Queueing Systemszes, *Annales Mathematicae et Informaticae.*, **37** (2010), 51-75

J6 T. Berczes, A. Hazy, and J. Sztrik, The impact of servers breakdown on the performance of proxy cache servers Mathematical and Computer Modelling (Submitted)

- **[C] Konferencia kiakdványok:**

C1 T. Berczes and J. Sztrik, A queueing network model to study Proxy Cache Servers. *Proceedings of 7th International Conference on Applied Informatics.*, Eger, Hungary Vol. **1** (2007), 203-211.

C2 T. Berczes, G. Guta, G. Kusper, W. Schreiner and J. Sztrik, Analyzing a Proxy Cache Server Performance Model

with the Probabilistic Model Checker PRISM. *WWV 2009 Automated Specification and Verification of Web Systems 5th International Workshop.*, Linz (2009).

- Angel Vassilev Nikolov, Effects of the coherency on the Performance of the Web Cache Proxy Server. *International Journal of Computer Science and Network Security.*, **9** (2009), 158-162.
- LIU Xu-jun, MA Yue and YU Dong Analysis and Evaluation of Real-time Performance of Publish/Subscribe Communication Mode. *Computer Engineering.*, Vol. **9**, No. **20** (2010), 229-231.

- **[O] Egyéb**:

O1 T. Berczes, G. Guta, G. Kusper, W. Schreiner and J. Sztrik, Analyzing Web Server Performance Models with the Probabilistic Model Checker PRISM. *Technical report no. 08-17 in RISC Report Series* , University of Linz, Austria. November 2008

O2 T. Berczes, G. Guta, G. Kusper, W. Schreiner and J. Sztrik, Comparing the Performance Modeling Environment MOSEL and the Probabilistic Model Checker PRISM for Modeling and Analysing Retrial Queueing Systems. *Technical report no. 07-17 in RISC Report Series* , University of Linz, Austria. November 2007