Debreceni Egyetem
Informatika Kar

Informatikai rendszerek sztochasztikus modellezése

Témavezető:

Dr. Sztrik János

egyetemi tanár

Készítette:

Máté Balázs

Programtervező Informatikus BSc

Debrecen

2009

Tartalomjegyzék

1. Bevezetés **4**

2. Google PageRank **6**

2.1. Bevezetés 6

2.2. A web Markov modellje 7

2.2.1. A PageRank vektor kiszámítása 10

2.2.2. A modell közelebbről 11

2.2.3. A módszer konvergenciája 15

2.3. A PageRank vektor frissítése 17

2.4. Numerikus számítás 18

3. Réselt Aloha **20**

3.1. Bevezetés 20

3.2. A Markov-lánc 21

3.3. A réselt Aloha hatékonysága 24

3.3.1. A réselt Aloha egy speciális esete 25

4. Unicast és Multicast kommunikáció **27**

4.1. Bevezetés 27

4.1.1. Multicast vezetéknélküli LAN felett 29

4.1.2. Unicast és multicast idődiagramm 30

4.2. A sztochasztikus modell 31

4.2.1. Unicast kommunikáció 31

4.2.2. Multicast kommunikáció 32

4.3. Hatékonysági vizsgálat 35

4.3.1. Egy streamelt média, $M = 1$ 35
4.4. Összegzés 36

5. Összegzés **38**

Köszönetnyilvánítás

Köszönettel tartozom Dr. Sztrik János Tanár Úrnak, hogy észrevételeivel és ötleteivel segítette szakdolgozatom megírását.

1. fejezet

Bevezetés

Szakdolgozatomban három igen különböző informatikai rendszer sztochasztikus modelljét és azok vizsgálatát fogom ismertetni. Látni fogjuk, hogy ezen a különböző problémák modellezése és hatékonysági vizsgálata során egy óriási, matematikusok által megalapozott és kifejlesztett eszköztárat kapunk, amely segítségével sok korábbi eredmény egyszerűen felhasználható.

A legjobb példa erre a Google két alapítójának, Sergey Brinnek és Lawrence Pagenek a kilencvenes évek végén született ötlete arra a kérdésre, hogy hogyan kellene a weblapokat valamilyen jól meghatározott szisztéma szerint fontossági sorrendbe rakni. Számos korábbi mennyiségbeli mérték után kitalálták a Google PageRank algoritmusát, amely a weblapok egy minőségbeli sorrendjét adja meg. A módszer lényege, mint az dolgozatom első fejezetében látható, a Web mögött meghúzódó linkstruktúrán való véletlen szörfölés megfeleltetése egy Markov-láncnak. Ezen megfeleltetés után már könnyen meghatározhatóak a egyensúlyi eloszlás létezésének a feltételei, majd ezt biztosítva számos ismert módszer áll rendelkezésre a stacionárius eloszlás kiszámolására, ami éppen a Google sorrendalkotásában felhasznált PageRank értékekkel egyezik meg.

A második fejezetben a modell bevezetése után látni fogjuk és megérthetjük annak az okát, hogy az Aloha protokoll esetén a kihasználtság miért is 18%, míg a réselt Aloha protokollnál 36%, valamint, hogy miért válik instabillá a réselt Aloha, amint a csomópontok száma egyre nő.

Az utolsó fejezetben az unicast és a multicast kommunikáció modelljeit ismertetem, valamint a gyakorlati életből hozott adatok segítségével egy kézzel fogható összehasonlítást kapunk a hatékonyságbeli különbségre. Érdemes megjegyezni, hogy az unicast esetében felhasználható volt a már ismert $M/M/1/K$ véges befogadóképességű rendszernél meghatározott stacionárius eloszlás.

Témaválasztásom annak volt köszönhető, hogy igazán érdekelnek a valódi gyakorlati problémák mögött meghúzodó matematikai modellek, valamint azonfajta kíváncsiság, hogy milyen formában alkalmazhatóak az alkalmazott esetleg a tiszta matematika eszközei a való életben felbukkanó kérdések során. Dolgozatom megírásakor fontos szerepet játszott, hogy minden egyes fejezetben egy rövid bevezetés után betekintést nyerjünk a "kulisszák mögé", megismerkedve ezzel a mélyen meghúzodó modellekkel és azok használával.

Nem célom a Markov-láncok általános bevezetésének, az élmélet megalapozásának tárgyalása, így az elkövetkezendőekben feltételezem, hogy az olvasó tisztában van a mind a diszkrét mind a folytonos idejű Markov-láncok fogalmával, valamint ismer olyan fogalmakat, mint átmenetvalószínűség, átmenetvalószínűség mátrix, sztochasztikus mátrix, stacionárius eloszlás, Poisson-folyamat.

2. fejezet

Google PageRank

2.1. Bevezetés

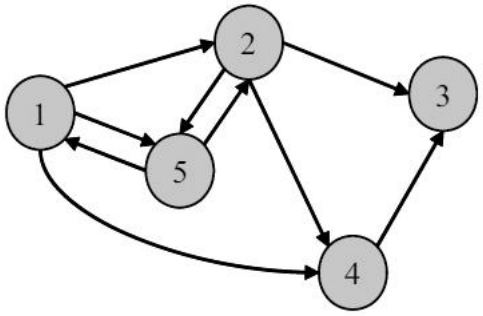
A fejezetben a Google PageRank algoritmusáról, valamint a Web PageRank által is használt modelljéről lesz szó. A felhasználó kérésének kiszolgálására a keresőmotorok elsőnek végrehajtanak egy keresést, hogy megkapjanak minden olyan oldalt, amely tartalmazza a felhasználó által megadott kifejezést. A Word Wide Web hatalmas méretének köszönhetően, ez az első lépés óriási mennyiségű weblapot szolgáltathat eredményül, olyannyira, hogy egy kifejezésre sokezer találat is gyakori. A lapok listájának csökkentése céljából a keresőmotorok valamilyen rendezést alkalmaznak.

A következőkben a Google két alapítója, Sergey Brin és Larry Page által kifejlesztett, link-alapú rangsoroló rendszerrel, a Google PageRankkel fogok foglalkozni. A PageRank 1998-as kitalálása előtt már nagy irodalma volt a tudományos publikációk analízisének, amelyekben valamilyen fontosság kiszámítása volt a cél, azonban a weblapok egy sokkal heterogénebb, változatosabb közeget alkotnak mint a tudományos publikációk. A lelkiismeretesen felülvizsgált publikációkhoz képest a weblapok burjánzanak mindenféle minőségellenőrzés és publikálási költség nélkül. Egy egyszerű programmal, gyorsan óriási számú weblap készíthető, amelyek egyes weblapok értékelését merőben átalakíthatják, így webes környezetben olyan rendszerre van szükség amely valamilyen értelemben robosztus és nem felel könnyedén ilyen manipulálások hatására.

Ha a weblapoknál, hasonlóan mint a publikációknál, az oldalra mutató linkek számát számolnánk, az eredmény számos esetben eltérne attól amit általában várnánk. A PageRank nem csak azt veszi számba, hogy hány link mutat az egyes oldalra, hanem azt is, hogy azokat a linkeket tartalmazó oldalak mennyire "fontosak".
A PageRank és a mögötte lévő sztochasztikus modell a Web link struktúrát használja fel arra, hogy a lapokat "fontossági" sorrendbe állítsa.

2.2. A web Markov modellje

A World Wide Web hiperlink struktúrája tekinthető úgy, mint egy irányított gráf N csúccsal. A webgráfon minden csúcs egy bizonyos weblapot, míg az ezeket összekötő élek a lapok közötti linkeket reprezentálják. Vegyünk egy pár weblapot, amelynek gráf és mátrix reprezentációja az alábbi ábrákon található, hogy látható legyen a PageRank mögött meghúzódó ötlet.



$$P = \begin{bmatrix} 0 & P_{12} & 0 & P_{14} & P_{15} \\ 0 & 0 & P_{23} & P_{24} & P_{25} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & P_{43} & 0 & 0 \\ P_{51} & P_{52} & 0 & 0 & 0 \end{bmatrix}$$

Bármely gráf topológiája meg van határozva a csúcsmátrixa által. Egy értelmes kritérium a weblapok fontosságnak meghatározásához a weblap meglátogatásinak száma, amely pontosan ahhoz fog vezetni, hogy problémánkat egy diszkrét idejű Markov-lánccal modellezzük, ahol az átmenetvalószínűség mátrix összhangban van a webgráf csúcsmátrixával.

Az átmenetvalószínűségmátrix P_{ij} eleme azzal a valószínűséggel egyezik meg, hogy az i-edik lapról (i-edik állapotból) a j-edik lapra (j-edik állapotba) lépünk. A $s_k[n]$ annak a valószínűségével fog megegyezni, hogy az n-edik időpillanatban a k-adik oldal van meglátogatva. A stacionárius eloszlásban a π_k értéke a látogatások számának időhányadával fog megegyezni, azaz, hogy átlagoson milyen valószínűséggel lesz egy adott weblap megláto-

gatva. Pontosan ez a π_k érték fog megfelelni a Google által használt fontosság mértékének. Maga az ötlet egyszerű, de még meg kell mutatni, hogy hogyan számíthatóak ki a P_{ij} értékek, valamint hogy létezik a π_i stacionárius eloszlás. Biztosítva, hogy létezik a stacionárius eloszlás, és hogy ez hogyan számítható ki , π_i-k kiszámítása több milliárd weblap esetén számos gyakorlati problémát vet fel.

A következőkben fel lesz téve, hogy ha egy adott i-edik weblapon állunk akkor minden különböző linkre kattintás valószínűsége megyegyezik, azaz egyenlő valószínűséggel lépünk tovább bármilyen linkelt weblapra. Az előbbi azt jelenti, hogy $P_{ij} = \frac{1}{d_i}$ ahol d_i jelöli az i-edik lapon lévő linkek számát, azaz az i-edik oldal közvetlen szomszédjainak a számát. Az előbbi mátrix a következőképpen módosul.

$$P = \begin{bmatrix} 0 & \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \end{bmatrix}$$

Az egyenletesség feltévése a legtöbb esetben a legjobb módszer ami használható ha nincsen addicionális információ. Ha például elérhető olyan információ, hogy a véletlen szörfölő kétszer akkora valószínűséggel megy a negyedik lapra a kettesről, mint a többire akkor a második sor a következőképpen alakul: $\begin{pmatrix} 0 & 0 & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{pmatrix}$. Ha szimplán a csúcsmátrixot használjuk, mint a P átmenetvalószínűség mátrix alatt meghúzódó struktúrát, akkor nem biztosítható, hogy P sztochasztikus mátrix. Vegyük például a példánkban látható harmadik csúcsot, amely nem tartalmaz egyetlen kimenő linket sem. Az ilyen csúcsokat lógó (dangling) csúcsoknak nevezzük. Példának okáért számos weblap mutathat egy olyan fontos dokumentumra az interneten, amely önmaga nem tartalmaz egyetlen kimenő linket sem.

Ha az oldalhoz tartozó sor P-ben csupa nulla elemeket tartalmaz akkor az megsérti a sztochasztikus mátrixok alaptulajdonságát, mégpedig azt, hogy a sorösszegnek 1-nek kell lennie azaz, hogy $\sum_{j=1}^{n} P_{ij} = 1$. Hogy helyrehozzuk a sztochasztikus mátrixtól való eltérését, egy olyan v nemnulla vektorral kell kicserülnünk amelyre $||v||_1 = 1$. A legegyszerűbb megoldást most is, ha egyenlően osztjuk el az értékeket. Ezeket után a mátrix a

következő alakot ölti:

$$\bar{P} = \begin{bmatrix} 0 & \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ 0 & 0 & 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \end{bmatrix}$$

A probláma az, hogy még ezen módosítások sem biztosítják a stacionárius eloszlás létezését. A következő tétel biztosítja, hogy milyen féltételek mellett létezik egy Markov-láncnak stacionárius eloszlása:

1. Tétel. *Ha egy irreducibilis Markov-lánc $A_1, A_2, \ldots$ állapotai visszatérő, gyakori aperiodikus állapotok, akkor létezik a*

$$\lim_{n\to\infty} P_{jk}^{(n)} = \pi_k$$

határérték j és k minden értékére és nem függ j-től; π_k az A_k állapot visszatérési időtartamának várható értékének reciprokával egyenlő. A P_k számok pozitívak és eleget tesznek a

$$\pi_k = \sum_{j=1}^{\infty} P_{jk}\pi_j$$

egyenletrendszernek és a

$$\sum_{j=1}^{\infty} \pi_k = 1$$

feltételnek. Az előbbi egyenletrendszernek nincs más az utóbbi feltételnek megfelelő megoldása.

Mint tudjuk, egy irreducibilis Markov-lánc esetén bármely állapotból elérhető bármely másik állapot. Különleges természetének köszönhetően a World Wide Web szinte biztosan reducubilis Markov-lánchoz vezet. Irreducibilis mátrix előállításahoz Brin és Page a következővel változtatással állt elő:

$$\bar{\bar{P}} = \alpha\bar{P} + (1-\alpha)\frac{u.u^T}{N} \tag{2.1}$$

ahol $0 < \alpha < 1$ az úgynevezett csillapítási (damping) faktor, $u.u^T$ egy $N \times N$ mátrix amely minden eleme 1, $\bar{P}$ pedig az előbb megadott nulla sor nélküli mátrix. A $\bar{P}$ sztochasztikus mátrix és a $u.u^T$ sztochasztikus perturbációs mátrix lineáris kombinációja biztosítja, hogy $\bar{\bar{P}}$ egy irreducubilis sztochasztikus mátrix. Minden egyes csúcs közvetlenül csatlakozik minden másik csúcshoz, amely a Markov-láncot olyan irredecibilis Markov-lánccá teszi,

amelynek állapotai visszetérőek, gyakori, aperiodikus állapotok.

Egy kis általánosításhoz vezet, ha a $\frac{u.u^T}{N}$ mátrixot lecseréljük egy uv^T mátrixra, ahol v egy valószínűségi vektor, ahol még meg kell azt követelnünk, hogy v minden egyes komponense nemnulla, hogy biztosítva legyen az elérhetőség. Brin és Page a v^T vektort perszonalizációs (personalization) vektornak nevezte, amely lehetőséget ad az egyenletességtől való eltérésre. Ezen lépések után megérkeztünk a Brin és Page Markov átmenetvalószínűgmátrixhoz, amely a következő:

$$\bar{\bar{P}} = \alpha\bar{P} + (1-\alpha)uv^T. \tag{2.2}$$

$v^T = \begin{bmatrix} \frac{1}{16} & \frac{4}{16} & \frac{6}{16} & \frac{4}{16} & \frac{1}{16} \end{bmatrix}$ és $\alpha = \frac{4}{5}$ esetén

$$\bar{\bar{P}} = \begin{bmatrix} \frac{1}{80} & \frac{19}{60} & \frac{3}{40} & \frac{19}{60} & \frac{67}{240} \\ \frac{1}{80} & \frac{1}{20} & \frac{41}{120} & \frac{19}{60} & \frac{67}{240} \\ \frac{1}{16} & \frac{1}{4} & \frac{3}{8} & \frac{1}{4} & \frac{1}{16} \\ \frac{1}{80} & \frac{1}{20} & \frac{7}{8} & \frac{1}{20} & \frac{1}{80} \\ \frac{33}{80} & \frac{9}{20} & \frac{3}{40} & \frac{1}{20} & \frac{1}{80} \end{bmatrix}$$

Ha a most tárgyalt megoldási menet lenne implementálva, akkor a kiindulási, nagyon ritka P mátrix le lenne cserélve a sűrű $\bar{\bar{P}}$ mátrixra, amely a tárgyhelyigényt drasztikusan megemelné. Ennél hatékonyabb egy speciális r vektort definiálása, amelynél $r_j = 1$ ha a j-edik sor nulla sor, egyébként nulla. Ekkor $\bar{P} = P + rv^T$ a rank-one változtatása P-nek és így

$$\bar{\bar{P}} = \alpha(P + rv^T) + (1-\alpha)u.v^T = \alpha P + (\alpha r + (1-\alpha)u)v^T.$$

2.2.1. A PageRank vektor kiszámítása

A π stacionárius eloszlás kielégíti a $\pi = \pi\bar{\bar{P}}$ egyenletrendszert, de ahelyett, hogy megoldanánk az egyenletrendszert Brin és Page azt javasolta, hogy π a $\pi = \lim_{k\to\infty} s[k]$ formula segítségével legyen kiszámolva. Egy $s[0]$ kezdeti vektorból kiindulva (legtöbbször $s[0] = \frac{u^T}{N}$) az $s[k+1] = s[k]\bar{\bar{P}}$ iterációt ismételve m-szer kapjuk a megoldást, ahol m elég nagy ahhoz, hogy $||s[m] - \pi|| \leq \varepsilon$, ahol ε egy előírt tolerancia. Ha valaki jobban szemügyre veszi az előbbi egyenletet, akkor látható, hogy ez a módszer a $\pi = \pi\bar{\bar{P}}$ sajátvektor feladat

megoldása hatványmódszer segítségével. Mielőtt a módszer iterációjának konvergenciáját vizsgálnánk, először vizsgáljuk meg az $s[k+1] = s[k]\bar{\bar{P}}$ iterációt.

$$s[k+1] = s[k]\bar{\bar{P}} = s[k](\alpha P + (\alpha r + (1-\alpha)u)v^T)$$

Mivel $s[k]u = 1$,

$$s[k+1] = \alpha s[k]P + ((\alpha s[k]r + (1-\alpha))v^T.$$

Az előbbi formulaból az következik, hogy csak $s[k]$ és a rendkívül ritka P mátrix szorzatát kell kiszámolni, így a $\bar{P}$,$\bar{\bar{P}}$ mátrixok sehol sincsenek kiszámolva vagy letárolva. Ez azért fontos mert a weblapok óriási száma (2004-ben 4.3 milliárd weblap volt beindexelve a Google által) rendkívüli tárhelybeli és sebességbeli problémákat vetne fel. Mivel a P ritka mátrix-szal történik a számolás, ez azt jelenti, hogy minden vektor-mátrix szorzás kiszámítható $nnz(P)$ flop alatt, ahol $nnz(P)$ a nemnulla elemek száma P-ben. Mivel a linkek átlagos száma oldalanként 3-10, így $O(nnz(P)) \approx O(n)$. Megjegyzendő még, hogy a hatvány módszer esetén minden egyes iterációnál elegendő egyetlen vektort letárolni (az aktuálisat), míg más módszereknél, több vektor tárolása szükséges.

2.2.2. A modell közelebbről

Az α csillapítási faktor

Brin és Page a számolásoknál $\alpha = 0.85$ értéket használ, amely után felmerülhet az emberben, hogy miért pont annyit, más érték esetleg sokat változtatna a sorrenden vagy gyorsabb konvergenciát biztosítani-e. A következő részekben látható, hogy megvan a maga oka a 0.85-nek, mégpedig a konvergencia sebessége. Ezen értékkel a 2004-es adatokra támaszkodva 114 iteráció elégséges $\tau = 10^{-8}$ tolerancia szint mellett. Összehasonlításképp, 0.99 esetén ugyanekkora tolerancia szinttel 1833 iterációra lenne szükség. Egy 4.3 millárdszor 4.3 milliárdos ritka mátrix esetén minden iteráció számít és a Googlenél néhány száz iterációnál több nem megengedett.

Azon feltételezés, hogy a 0.85 jól modellezi a webező viselkedését, is közrejátszhatott a megfelelő érték kiválasztásában, mivel a 0.85 érték azt is feltételezi, hogy egy véletlen szörfölő az esetek 5/6 részében valamely oldalon elhelyezett linkre kattint követve ezzel a Web struktúráját (ezt fejezi ki az $\alpha\bar{P}$ rész), míg a fentmaradó 1/6 részben új címet ad meg a

böngészősávban, úgymond "teleportál" (ezt fejezi ki az $(1-\alpha)uv^T$ rész). 0.99 esetén nemcsak a konvergencia gyorsasága sokkal alacsonyabb, de sokkal nagyobb hangsúlyt is kap a Web struktúrája a "teleportáláshoz" képest. A 0.99-re kapott PageRank vektor merőben más lehet mint a 0.85-re kapott, bár lehet, hogy valósabb értéket ad. Mint kísérletek kimutatták, különböző α értékekre igen különböző érdemények alakulhatnak ki. Ezen különbségek nem a lista elején, hanem egyre inkább lefelé haladva a listán kerülnek elő.

A v^T perszonalizációs vektor

Mint láttuk a 2.1 egyenlet után be lett vezetve a v^T perszonalizációs vagy "teleportáló" vektor. Mivel v^T valószínűségi vektor pozitív elemekkel, így minden oldal elérhető lesz minden oldalról így megmarad a Markov-lánc irreducibilitása. Az $\frac{1}{n}u^T \rightarrow v^T$ változtatással azt értük el, hogy a teleportációs valószínűségek nem egyenletesen vannak elosztva. Így ezen vektorral az befolyásolható, hogy egy véletlen szörfölő milyen valószínűséggel fog egy adott oldalra teleportálni. Fontos megjegyezni, hogy ezen változtatás nem befolyásolja a hatvány módszer jó tulajdonságait. Tehát ahhoz, hogy a PageRank vektor személyre szabott legyen, csupán a v^T vektort kell módosítani.

A Googlenek szándékában állt, hogy különféle perszonalizációs vektorokat használjon különböző osztályba sorolt felhasználók számára. Például az egyik osztály tagjai ha nem linket követnek hanem teleportálnak, akkor valószínűbb, hogy sport oldalra ugranak míg egy másik csoport tagjai pedig ilyenkor inkább hírekről szóló oldalakra ugranak. Ezen kétféle mód két különböző vektorral megoldható, habár az eddigi kulcsszó független és felhasználó független PageRanket felhasználó függővé és számolásigényesebbé teszi. Mindazonáltal ezen eszköz segítségével a Google eredményeket ért el a spammelő linkfarmokkal szemben. Ezen link farmok azt a célt szolgálják, hogy becsapva a keresőcégeket a klienseket a rangsorokban előrébb juttatják.

Vegyük a következő példát: egy cég úgy dönt, hogy üzleti tevékenyégének egy részét az interneten bonyolítja és ehhez csináltat magának egy saját weboldalt, majd felkér egy keresőmotor optimalizálással foglalkozó céget, hogy javítson az oldala besorolásán. Ezen javításhoz az egyik út a link farm használáta. Tudván, hogy egy oldal PageRankje nő, ha magas PageRankű oldal mutat rá, a link farm az általa birtokolt magas PageRankű

oldaláról az ügyfél oldalára fog linkelni. Ezen optimalizálással foglalkozó cégek számos összekapcsolt oldalt tartanak fent különböző fontos és közkedvelt témákkal kapcsolatban, amelyek magas PageRankkel bírnak. A Googlenek sikerült a v^T vektor megváltoztatásával ezen egyértelműen káros farmok működését meggátolni.

Tárolási kérdések

Ha a Web kis részeinek megfelelő P mátrix befér a memóriába, akkor a *PageRank* kiszámítása a megszokott módon történhet, viszont számos esetben ezen mátrix nem fér el a memóriában, így a kutatóknak ilyenkor valamilyen új ötlettel kell előrukkolni a tárolással és implementálással kapcsolatban. Ilyen esetekben vagy összetömörítik a szükséges adatokat annyira, hogy a tömörített változat már belefér a memóriába és utána valamilyen trükkös megoldással implementálják a PageRank egy módosítását az összetömörített adatokon, vagy az adatokat tömörítetlenül hagyják és a számítások egy I/O hatékony implementációjával állnak elő.

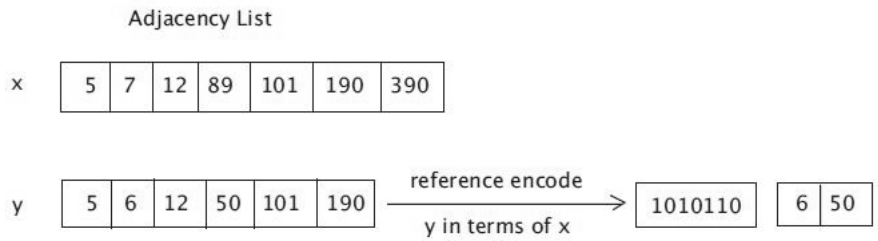
Azon P mátrixok esetén, amelyek befférnek a memóriába, is ajánlott néhány módszert bevetni, hogy csökkenjen a számolási költség az egyes iterációkban. Erre egy példa, amikor a P mátrix $D^{-1}G$ alakban van előállítva, ahol a D diagonális mátrixban a főátlóbeli elemek az egyes sorokban található nemnulla értékek száma, ami tulajdonképpen az oldalon lévő linkek számával egyezik meg, míg a G mátrix a nullákat és egyeseket tartalmazó szomszédsági mátrix. Ezen átalakítás segítségével a hatványmódszerbeli iterációkban lévő mátrix-vektor szorzás költsége csökkenthető $nnz(P) - n$-nel, ahol $nnz(P)$ a P mátrixban található nemnulla elemek számát jelöli.

A teljes Web esetén ahelyett, hogy a mátrix vagy annak tömörített változata lenne letárolva, inkább a P vagy G mátrix oszlopai szerinti szomszédsági lista van. A mátrix egyes oszlopaiban találhatóak meg az oldalra mutató linkek, ami a PageRank esetében fontosabb, mint az sorok, amelyekben az oldalról induló linkekről van információ. Mivel minden iteráció során az $s[k]P$ vektor-mátrix szorzásnál a P mátrix oszlopaira van szükség, így az oszlopok gyors elérése alapvetően fontos az algoritmus hatékonyságához.

A kezdeti példa reprezentációja:

Csomópont	Bejövő linkek
1	5
2	1, 5
3	2, 4
4	1, 2
5	1, 2

A lista méretének csökkentésére is vannak különböző eljárások. A rés módszer (gap technique) azt használja ki, hogy a linket tartalmazó és a linkelt oldal általában lexikografikusan igen közel helyezkedik el egymáshoz, így legtöbbször elég kis értékeket letárolni, amelyek az oldalak indexei közötti különbségeket reprezentálják. A hivatkozási módszer (reference encoding technique) esetén pedig az oldalokon lévő linkek közötti hasonlóság felhasználásával csökkenthető a tárigény. Ez azt jelenti, hogy ha az a és b weblapokon elhelyezkedő linkek között vannak egyezők, akkor b-nél elég az a listáját hivatkozni, valamint a nem megyező linkeket még külön feltüntetni. A hivatkozás a következők szerint történik: a b oldalnál sorba kell venni a listáját és el kell készíteni egy bitvektort, amiben egyes érték szerepel ha az a listában lévő oldal linkelve van b-ről is, míg ellenkező esetben nulla szerepel. Ezen átalakítás az alábbi ábrával szemléltethető:



2.1. ábra. Hivatkozási módszer, Ábra: [1]

2.2.3. A módszer konvergenciája

1. Lemma. *Ha a P sztochasztikus mátrix sajátértékei $\{1, \lambda_2, \lambda_3, \ldots, \lambda_n\}$, akkor a $\bar{P} = \alpha P + (1-\alpha)uv^T$ (v^T tetszőleges valószínűségi vektor) mátrix sajátértékei $\{1, \alpha\lambda_2, \alpha\lambda_3, \ldots, \alpha\lambda_n\}$.*

Bizonyítás: Kiindulva az ismert sajátérték egyenletből kapjuk, hogy

$$\begin{aligned}\det(\bar{P} - \lambda I) &= \det(\alpha P - \lambda I + (1-\alpha)uv^T) \\ &= \det((\alpha P - \lambda I)(I + (\alpha P - \lambda I)^{-1}(1-\alpha)uv^T) \\ &= \det(\alpha P - \lambda I)\det(I + (1-\alpha)(\alpha P - \lambda I)^{-1}uv^T)\end{aligned}$$

valamint felhasználva, hogy

$$\det(I + cd^T) = 1 + d^T c,$$

amely a következő mátrix azonosságból következik

$$\begin{pmatrix} I & 0 \\ d^T & 1 \end{pmatrix}\begin{pmatrix} I + cd^T & c \\ 0 & 1 \end{pmatrix}\begin{pmatrix} I & 0 \\ -d^T & 1 \end{pmatrix} = \begin{pmatrix} I & c \\ 0 & 1 + d^T c \end{pmatrix}$$

következik, hogy

$$\det(\bar{P} - \lambda I) = \det(\alpha P - \lambda I)(1 + v^T(1-\alpha)(\alpha P - \lambda I)^{-1}u)$$

Mivel a P mátrix sztochasztikus mátrix és így a sorösszegeket egyenlőek eggyel kapjuk, hogy $Pu = u$, így $(\alpha P - \lambda I)u = (\alpha - \lambda)u$, amiből következik, hogy $(\alpha P - \lambda I)^{-1}u = (\alpha - \lambda)^{-1}u$. Ezt felhasználva

$$1 + v^T(1-\alpha)(\alpha P - \lambda I)^{-1}u = 1 + \frac{1-\alpha}{\alpha - \lambda}v^T u = 1 + \frac{1-\alpha}{\alpha - \lambda} = \frac{1-\lambda}{\alpha - \lambda}$$

mivel $v^T u = 1$, hiszen v valószínűségi vektor, így

$$\det(\bar{P} - \lambda I) = \det(\alpha P - \lambda I)\frac{1-\lambda}{\alpha - \lambda}$$

Felhasználva, hogy bármely $g(x)$ polinom esetén $g(A)$ sajátértékei: $g(\lambda_1), \ldots, g(\lambda_n)$, valamint a karakterisztikus polinom

$$\det(g(A) - \lambda I) = \prod_{k=1}^{n}(g(\lambda_k) - \lambda)$$

kapjuk, hogy

$$\det(\bar{P} - \lambda I) = \prod_{k=1}^{n}(\alpha\lambda_k - \lambda)\frac{1-\lambda}{\alpha-\lambda} = (1-\lambda)\prod_{k=2}^{n}(\alpha\lambda_k - \lambda),$$

amelyből látható, hogy $\bar{P}$ sajátértékei $\{1, \alpha\lambda_2, \alpha\lambda_3, \ldots, \alpha\lambda_n\}$.

■

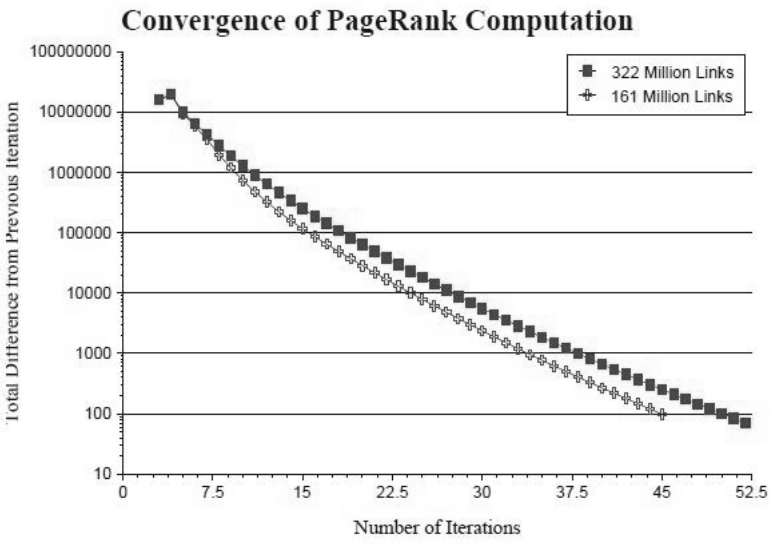
Felhasználva, hogy egy Markov-lánc konvergenciájának sebességét a stacionárius eloszláshoz az abszolút értékben második legnagyobb sajátérték határozza meg, valamint az előző lemmát, kapjuk, hogy α megfelelő kiválasztásával gyorsítható a konvergencia. Érdekes megjegyeznünk, hogy a konvergencia gyorsításához számos módszert kitaláltak a kutatók, úgy mint az egyes iterációkban való számolás idejének csökkentésére, vagy az össziterációszám csökkentésére.

Iteráción belüli munka csökkentése

Az egyik ilyen irányú módszert Kamvar fejlesztette ki és a neve alkalmazkodó PageRank (adaptive PageRank). A módszer csökkenti az iterációkban befektett munkát azzal, hogy jobban "szemügyre veszi" az elemeket a számolt vektorban. Ez annyit tesz, hogy egyes oldalak gyorsabban konvergálnak a PageRank értükükhöz mint mások, így azokat az alkalmazkodó módszer "lezárja", és nem használja a későbbi számításokban. Ezen módszer 17%-os gyorsítást eredményezett, viszont van egy szépséghibája mégpedig, az hogy a módszer konvergenciája nincs bebizonyítva, csak kis adathalmazokon lett megmutatva.

A másik módszert Chris Lee és kollégái fejlesztették ki a Stanford egyetemen, amelynek az ötlete az, hogy a csomópontokat lógó és nemlógó csoportba osztja, és az utóbbin végez számolásokat. Mivel tudjuk, hogy a lógó csomópontokból csupa azonos sorok lesznek képezve, így a felhasznált mátrix mérete erősen csökkenthető.

A PageRank egy 322 millió linket tartalmazó adatbázisból körülbelül 52 iteráció volt szükséges a konvergenciához, míg fél adatbázis esetén 45 iterációra. Mint láthattuk, α megfelelő kiválasztásával gyorsítható az eljárás, de azt is el kell fogadni, hogy a 2.2 egyenlet megváltoztatja a webgráf igazi sajátosságait. Brin és Page módszere arra, hogy meglegyen a kívánt irreducibilitás módosítja a webgráf igazi természetét még akkor is, ha a "kapcsolat erőssége" rendkívül alacsony: $\frac{1}{N}$ ha $v^T = \frac{u^T}{N}$. Ahelyett, hogy összekötnénk minden egyes csúcsot, létezik egy másik megoldás is az irreducibilitás eléréséhez, Langvilletől



2.2. ábra. Konvergencia teljes és fél adatbázis esetén, Ábra: [2]

és Meyertől, amely kevésbé változtatja meg a webgráfot. Módszerük abból áll, hogy készítenek egy álcsúcspontot (dummy node) amely minden csúcsponttal össze van kötve és amelyhez minden csúcspont csatlakozik.

2.3. A PageRank vektor frissítése

Lássunk néhány adatot azzal kapcsolatban, hogy hogyan változnak a weblapok. Egyes kutatók kimutták 2000-ben, hogy a saját adatbázisukban lévő honlapok 40%-a változott egy héten belül, míg a .com végződésűek 23%-a naponta. Vegyünk például egy hírekkel foglalkozó oldalt, ahol a főoldalon található linkek óráról órára változnak. Ugyan a PageRank ilyen gyakorisággal nem frissíthető, törekedni kell a naprakészségre.

Ismert, hogy egy iteráció akár több napig is eltarthat, valamint, hogy havonta frissítik az értéket és a nulláról számolják újra. Számos kutató foglalkozik a PageRank frissítésével, valamint azzal a kérdéssel, hogy a régebbi számolási eredmények felhasználhatóek-e valamilyen módon az új számítás esetében. A Google egy szóvivője egy 2002-es konferencián elmondta, hogy az adott havi vektor kiszámítását az előző havival kezdve semmilyen gyorsulást nem hozott.

A probléma a következő: adott a régei P Markov-mátrix és a stacionárius π vektor,

valamint az új $\bar{P}$ mátrix és találjuk meg az új $\bar{\pi}$ vektort. Egy intuitív probálkozás, hogy kezdővektortként π, választjuk remélve, hogy ha $\bar{P}$ közel van P-hez akkor a stacionárius vektorok is közel lesznek. Hacsak tényleg nincs a két vektor igen közel egymáshoz, akkor ezen kezdővektorbeli trükk semmilyen előrelépést nem nyújt.

2.4. Numerikus számítás

A következő matlab forráskód a [3] oldalon érhető el különböző példaadatbázisokkal együtt.

```
function [pi,time,numiter]=pagerank(pi0,H,v,n,alpha,epsilon);

% PAGERANK  computes the PageRank vector for an n-by-n Markov
%           matrix H with starting vector pi0 (a row vector),
%           scaling parameter alpha (scalar), and teleportation
%           vector v (a row vector).  Uses power method.
%
% EXAMPLE:[pi,time,numiter]=pagerank(pi0,H,v,900,.9,1e-8);
%
% INPUT:            pi0 = starting vector at iteration 0 (a row vector)
%         H = row-normalized hyperlink matrix (n-by-n sparse matrix)
%         v = teleportation vector (1-by-n row vector)
%         n = size of P matrix (scalar)
%         alpha = scaling parameter in PageRank model (scalar)
%         epsilon = convergence tolerance (scalar, e.g. 1e-8)
%
% OUTPUT:   pi = PageRank vector
%           time = time required to compute PageRank vector
%           numiter = number of iterations until convergence
%
% The starting vector is usually set to the uniform vector,
% pi0=1/n*ones(1,n).
% NOTE: Matlab stores sparse matrices by columns, so it is faster
%       to do some operations on H', the transpose of H.
```

```
% get "a" vector, where a(i)=1, if row i is dangling node
%   and 0, o.w.

rowsumvector=ones(1,n)*H';
nonzerorows=find(rowsumvector);
zerorows=setdiff(1:n,nonzerorows); l=length(zerorows);
a=sparse(zerorows,ones(l,1),ones(l,1),n,1);



k=0;
residual=1;
pi=pi0;
tic;

while (residual ≥ epsilon)
  prevpi=pi;
  k=k+1;
  pi=alpha*pi*H + (alpha*(pi*a)+1-alpha)*v;
  residual=norm(pi-prevpi,1);
end
numiter=k;
time=toc;
```

3. fejezet

Réselt Aloha

3.1. Bevezetés

Az Aloha protokoll egy egyszerű példája a többes hozzáférésű kommunikációs sémának, amelynek direkt leszármazottjaként van számon tartva az Ethernet. Az Alohát, amely egyébként hellót jelent hawaii nyelven, a hetvenes évek elején találták ki, hogy csomag kapcsolt rádiós kommunikációt nyújtson egy központi számítógép és a különböző campusokon lévő adatterminálok között a Hawaii egyetemen. A réselt Aloha egy diszkrét idejű verziója a sima Aloha protokollnak, ahol minden egyes továbbított csomagnak ugyanakkora a hossza és ahol minden egyes csomagnak egy időrésre van szüksége a továbbításhoz. A következőkben a [4]-ben megtalálható modellt és hatékonysági elemzést fogom bemutatni.

Vegyünk egy N csomópontból álló hálózatot, amelyben a csomópontok egy közös kommunikációs csatornán keresztül kommunikálni a réselt Aloha protokollt használva. Legyen az A beérkezési folyamat minden egyes csomópontnál a Poisson-folyamat. Tegyük fel, hogy ezen Poisson-folyamatok függetlenek egymástól és minden egyes csomópontnál $\frac{\lambda}{N}$ az intenzitás, tehát az összintenzitás az N csomópontból álló rendszernél λ.

A réselt Aloha protokollnál a csomópont az újonnan érkezett csomagot a következő időrésben továbbítja. Ha két csomópont ugyanabban az időrésben továbbít csomagot, akkor ütközés következik be, és ezt azt eredményezi, hogy a csomagokat újra kell küldeni. Azt a csomópontot, amelynek a csomagját újra kell küldeni, backlogged csomópontnak

nevezzük. Még ha egy új csomag érkezik is a backlogged csomóponthoz akkor is az újraküldendő lesz az első továbbítandó, valamint az egyszerűség kedvéért feltesszük, hogy az újonnan érkezőket eldobjuk, hogy figyelmen kívül hagyhassuk a csomópontoknál lévő várakozási sort. Ha a backlogged csomópontok újra továbbítanák a következő időrésben a csomagokat, akkor biztos, hogy megint ütközés lépne fel, ezért a backlogged csomópontok véletlen számú időrést várnak, hogy továbbítsák a csomagokat. Az egyszerűség kedvéért még feltesszük azt, hogy p_r annak a valószínűsége, hogy sikeres továbbítás történik a következő időrésben. Ezen p_r érték minden egyes backlogged csomópontnál és minden egyes időrésnél megegyezik. Az ütközés és a sikeres továbbítás közötti időrések száma geometriai eloszlást követ p_r paraméterrel, tehát

$$P(T_r = k) = p_r(1 - p_r)^{k-1},\ k = 1, 2, \ldots \tag{3.1}$$

3.2. A Markov-lánc

A réselt Aloha egy diszkrét idejű $X_k \in \{1, 2, \ldots, N\}$ Markov-láncot alkot, ahol a j állapot azt jelzi, hogy az N csomópontból hány darab backlogged csomópont, k pedig a k-adik időrést jelöli. A j darab backlogged csomópontból minden egyes csomópont továbbítani fog egy csomagot a következő időrésben p_r valószínűséggel, valamint a többi $N - j$ darab csomópont is biztosan továbbítani fog újonnan érkező csomagot, feltéve, hogy az aktuális időrésben érkezett új csomag. Az utóbbi esemény $p_a = P(A > 0) = 1 - P(A = 0)$ valószínűséggel következik be. Ha feltesszük, hogy a beérkezési folyomat Poisson-folyamat, akkor

$$p_a = 1 - \exp\left(-\frac{\lambda}{N}\right), \tag{3.2}$$

azonban a következő számítások általános esetben is igazak.

Annak a valószínűsége, hogy a következő időrésben a j állapotban n darab backlogged csomópont továbbít újra csomagot, binomiális eloszlású, mégpedig

$$b_n(j) = \binom{j}{n} p_r^n (1 - p_r)^{j-n}. \tag{3.3}$$

Hasonlóan, annak a valószínűsége, hogy n darab nem backlogged csomópont továbbít a j állapotban

$$u_n(j) = \binom{N-j}{n} p_a^n (1-p_a)^{N-j-n}. \tag{3.4}$$

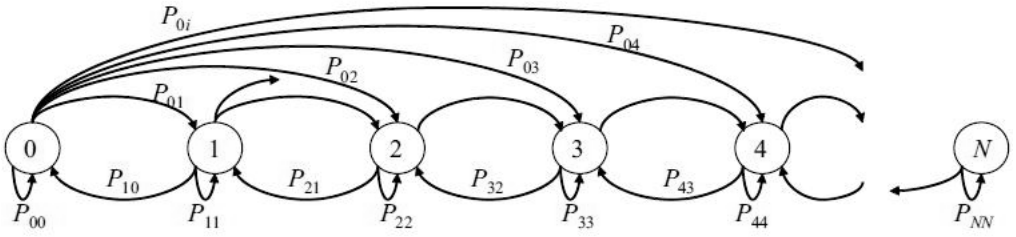
Egy csomag akkor és csak akkor lesz sikeresen továbbítva ha pontosan egy újonnan érkezett és nulla backlogged vagy pontosan egy backlogged és nulla újonnan érkezett továbbít. A sikeres továbbításnak a valószínűsége a j állapotban időrésenként:

$$p_s(j) = u_1(j)b_0(j) + u_0(j)b_1(j) \tag{3.5}$$

A $P_{j,j+m}$ átmenetvalószínűség

$$P_{j,j+m} = \begin{cases} u_m(j), & 2 \le m \le N-j \\ u_1(j)(1-b_0(j)) & m = 1 \\ u_1(j)b_0(j) + u_0(j)(1-b_1(j)) & m = 0 \\ u_0(j)b_1(j) & m = -1 \end{cases} \tag{3.6}$$

A j állapot, ami j darab backlogged csomópontot jelöl, átugrik a $j-1$-edik állapotba, ha nem lett újonnan érkező csomag küldve és pontosan egy sikeres újratovábbítás volt. A j állapot marad a j állapotban, ha egy újonnan érkezett csomag küldése történt és backlogged csomópont nem továbbított újra, valamint akkor ha nincs új csomag továbbítása, és vagy nulla vagy legalább egy újratovábbítás történt. A j állapotból a $j+1$ állapotba kerülünk, ha pontosan egy új csomag továbbítása és legalább egy újratovábbítás történt. Végül, a j-ből $j+m$-be lépünk, ha n darab nem backlogged csomópont továbbít egyszerre függetlenül attól, hogy backlogged csomópont újratovábbít-e, vagy nem.
A megfelelő Markov-lánc a következő ábrával szemléltethető:



Az P átmenetvalószínűség mátrix a kövekező struktúrájú:

$$P = \begin{bmatrix} P_{00} & P_{01} & P_{02} & \dots & \dots & P_{0N} \\ P_{10} & P_{11} & P_{12} & \dots & \dots & P_{1N} \\ 0 & P_{21} & P_{22} & \dots & \dots & P_{2N} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & P_{N_1,N-2} & P_{N-1,N-1} & P_{N-1,N} \\ 0 & 0 & \dots & 0 & P_{N,N-1} & P_{N,N} \end{bmatrix}$$

Amikor $N \to \infty$, a réselt Alohának egy különös jellemzője van, mégpedig, hogy nem létezik a π stacionárius eloszlás. Habár kis N-ekre a stacionárius eloszlás kiszámolható, amint N nő, a réselt Aloha instabillá válik. A várható különbség a backlogged csomópontok számában időegységenként

$$\mathbb{E}(X_{k+1} - X_k | X_k = j) = (N-j)p_a - p_s(j), \tag{3.7}$$

amely megegyezik az új érkezők átlagos számának és a várható sikeres továbbítások számának különbségével. Az előbbi eredmény bizonyítása a következő:

$$\mathbb{E}(X_{k+1} - X_k | X_k = j) = \sum_{l=-1}^{N-j} lP(X_{k+1} - X_k = l | X_k = j) = \sum_{l=-1}^{N-j} l\frac{P(X_{k+1} - X_k = l, X_k = j)}{P(X_k = j)}$$

$$= \sum_{l=-1}^{N-j} l\frac{P(X_{k+1} = j + l, X_k = j)}{P(X_k = j)} = \sum_{l=-1}^{N-j} lP_{j,j+l} = -u_0(j)b_1(j) + u_1(j)(1 - b_0(j)) + \sum_{l=2}^{N-j} lu_l(j))$$

$$= -\underbrace{(u_0(j)b_1(j) + u_1(j)b_0(j))}_{p_s(j)} + \underbrace{\sum_{l=1}^{N-j} lu_l(j))}_{\mathbb{E}Bi(N-j,p_a)=(N-j)p_a} = (N-j)p_a - p_s(j)$$

Ezen $\mathbb{E}(X_{k+1} - X_k | X_k = j)$ értéket gyakran driftnek nevezik. Ha a drift pozitív minden egyes k időrésre, akkor a Markov-lánc átlagosan magasabb állapotok felé lép, amely a fenti ábrán valamilyen jobbra lépéssel egyezik meg. Mivel $p_s(j) \leq 1$ és $p_a = 1 - \exp(-\frac{\lambda}{N})$,

$$\lim_{N\to\infty} \mathbb{E}(X_{k+1} - X_k | X_k = j) = \infty \tag{3.8}$$

Mivel a drift végtelenbe tart, ez at jelenti, hogy átlagosan a backlogged csomópontok száma korlátlanul nő, ami azt sugallja, hogy a Markov-lánc tranziens, ha $N \to \infty$.

3.3. A réselt Aloha hatékonysága

A következőkben a j állapotban lévő sikeres továbbítás valószínűségét fogjuk tanulmányozni.

$$\begin{aligned} p_s(j) &= u_1(j)b_0(j) + u_0(j)b_1(j) \\ &= (N-j)p_a(1-p_a)^{N-j-1}(1-p_r)^j + jp_r(1-p_r)^{j-1}(1-p_a)^{N-j} \\ &= \left[\frac{(N-j)p_a}{1-p_a} + \frac{jp_r}{1-p_r}\right](1-p_a)^{N-j}(1-p_r)^j \end{aligned}$$

Kis p_r és p_a értékekre, a j állapotban lévő sikeres továbbítás valószínűsége közelíthető Taylor-sorba fejtés segítségével. Mivel $(1-x)^\alpha = e^{\alpha \ln(1-x)} = e^{-\alpha x}(1+o(1))$ és $\frac{x}{1-x} = x + o(x^2)$

$$\begin{aligned} p_s(j) &= [(N-j)p_a + jp_r + o(p_a^2 + p_r^2)]e^{-[(N-j)p_a + jp_r]}(1+o(1)) \\ &= [(N-j)p_a + jp_r]e^{-[(N-j)p_a + jp_r]}(1+o(1)) \end{aligned}$$

Hasonlóan annak a valószínűsége, hogy nem volt egyetlen továbbítás sem

$$\begin{aligned} p_{no}(j) &= u_0(j)b_0(j) = (1-p_r)^j(1-p_a)^{N-j} \\ &= e^{-[(N-j)p_a + jp_r]}(1+o(1)) \end{aligned}$$

Így alacsony p_a és p_r esetén a sikeres továbbítás, és a nulla darab csomag továbbításának valószínűsége a j állapotban jól közelíthető a következőkkel:

$$\begin{aligned} p_s(j) &\simeq t(j)e^{-t(j)} \\ p_{no}(j) &\simeq e^{-t(j)} \end{aligned}$$

ahol $t(j) = (N-j)p_a + jp_r$ az újonnan érkező csomagok és újratovábbítások várható száma, azaz a továbbítási kísérleteknek teljes intenzitása a j állapotban. A $t(j)$-t nevezik nyújtott forgalomnak és G-vel jelölik. Ahogy a fenti analízis mutatja kis p_a és p_r értékekre $p_s(j)$ és $p_{no}(j)$ jól közelíthető $t(j)$ paraméterű Poisson-eloszlással. Továbbá $p_s(j)$ interpretálható a j állapotból való távozás intenzitásának vagy teljesítménynek. $S_{RAloha} = Ge^{-G}$, amelynek $G = t(j) = 1$ esetén van maximuma. A réselt Aloha legalább kettő csomópont esetén vett η_{RAloha} hatékonysága az a maximális időhányad, amely során a csomagok továbbítása sikeres, amely $\max p_s(j) = e^{-1}$, így $\eta_{RAloha} = 36\%$.

A sima Aloha esetén, ahol a csomópontok tetszőleges időpontban kezdhetik meg a továbbítást és nem csak az időrés elején, csak fele olyan hatékonyan teljesít mint a réselt

Aloha, így $\eta_{PAloha} = 18\%$. Emlékezzünk arra, hogy feltettük, hogy minden csomag ugyanolyan hosszú és mindegyiknek egy időrés szükséges a továbbításához. Az Alohánál egy t időpontban küldött csomag akkor lesz sikeresen továbbítva, ha nincs másik küldött csomag a $(t-1, t+1)$ időintervallumban. Ez pontosan két időrésnek felel meg, ami magyarázatot ada arra, hogy $\eta_{PAloha} = \frac{1}{2}\eta_{RAloha}$ miért igaz. Ezen megfigyelés azt is megmutatja, hogy $p_{no}(j) \simeq e^{-2t(j)}$ mivel a sikeres időintervallum alatt átlagosan kétszer annyi új csomag és kétszer annyi újratovábbítás történik, mint a réselt Aloha esetén. Az S teljesítmény körülbelül megegyezik a továbbítások teljes intenzitásának (G) $p_{no}(j)$-vel vett szorzatával, így $S_{PAloha} = Ge^{-2G}$.

3.3.1. A réselt Aloha egy speciális esete

A következőkben azon változtatással élünk a réselt Aloha protokollhoz képest, hogy ebben az esetben ha n darab csomag ütközik, abból egy sikeresen megérkezik. Feltételezzük azt is, hogy minden csomagnak egyenlő valószínűsége $(1/n)$ van arra, hogy az érkezzen meg sikeresen. Ezen módosított változat (*Aloha with ideal capture effect*) ismertetése és a többi Alohával való összehasonlítása [5] alapján történik. Az érkezés ebben az esetben is λ paraméterű Poisson-folyamat alapján történik. Hozzá kell tennünk, hogy az intenzitás, mint ahogyan azt az előző részben láttuk, nem csak az új beérkezésekből jön, hanem az újraküldésekből is. Ezért λ helyett Λ intenzitást használunk a következőkben, így t idő alatt a G nyújtott forgalom Λt.

η_{CAloha} értéket az ütközött csomagok számának a reciprokának Poisson súlyokkal vett összegeként fogjuk kiszámítani, a következőek szerint

$$\eta_{CAloha} = \frac{S_{CAloha}}{G} = \sum_{n=0}^{\infty} \frac{1}{n+1}\frac{(\Lambda t)^n}{n!}e^{-\Lambda t} = \frac{e^{-\Lambda t}}{\lambda t}\sum_{i=1}^{\infty}\frac{(\Lambda t)^i}{t!} = \frac{e^{-\Lambda t}}{\Lambda t}\sum_{i=0}^{\infty}\left(\frac{(\Lambda t)^i}{t!} - 1\right)$$
$$= \frac{e^{-\Lambda t}}{\Lambda t}(e^{-\Lambda t} - 1) = \frac{1 - e^{-\Lambda t}}{\Lambda t} = \frac{1 - e^{-G}}{G}$$

Így

$$S_{CAloha} = 1 - e^{-G}$$

Amint az a fenti képletből látható, ha G nő, S tart 1-hez, valamint ha G közel van nullához $(G \to 0)$, akkor S is közel van hozzá. Az is észrevehető, hogy nincsenek stabil-

itási problémák, ami annak köszönhető, hogy ideális esetet tételeztünk fel. Gyakorlatban előfordulhat, hogy a legerősebb jellel rendelkező csomag megérkezik, de ez csak nagyon speciális szituációkban igaz.

Összehasonlítás

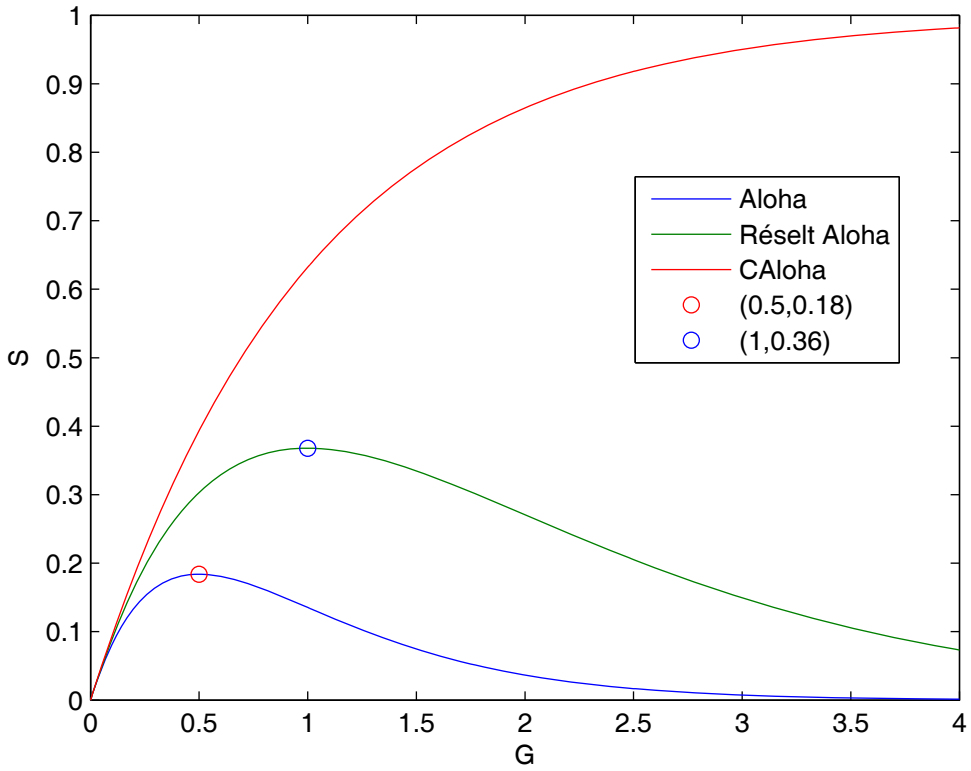
Ahogyan az előbbi részekből kiderült:

$$S_{Aloha} = Ge^{-2G} \tag{3.9}$$

$$S_{RAloha} = Ge^{-G} \tag{3.10}$$

$$S_{CAloha} = 1 - e^{-G} \tag{3.11}$$

Ezek a következő ábrán szemléltethetőek:



3.1. ábra. Aloha protokollok kihasználtsági összehasonlítása

4. fejezet

Unicast és Multicast kommunikáció

4.1. Bevezetés

Az **unicast** azon kommunikáció leírására szolgál, ahol az információ egyik pontból egy másik pontba van küldve. Ebben az esetben pontosan egy küldő és pontosan egy fogadó van. Az unicast továbbítás, ahol egy csomag egyetlen forrásból egy meghatározott címre van küldve, még mindig a domináns formája LAN esetén a továbbításnak. Minden LAN és az IP hálózatok is támogatják a unicast módot, valamint a legtöbb felhasználó ismer standard unicast alkalmazásokat, úgy mint például a http, smtp, ftp, telnet, amelyek a TCPt használják szállítási protokollként.

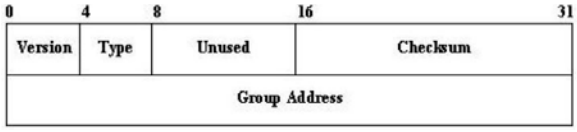
A **multicast** azon kommunikáció leírására szolgál, ahol az információ egy vagy több pontból, pontok egy halmazába kerül küldésre ezzel egy több-többes kapcsolatot kiépítve. Ebben az esetben lehet egy vagy több küldő, valamint az információ el van osztva fogadók egy halmazának. Példának hozható multicast alkalmazásárara egy videó szerver, amely TV adást nyújt a felhasználók számára. Magas minőségű videók szimultán nyújtása sok felhasználó esetén kimeríti a kapacitást, még akkor is, ha nagy a használt sávszélesség és erős videó klipp szerver van használatban. A multicast kommunikáció használata szignifikánsan csökkentheti ezen terheket rengeteg felhasználó esetén.

Az IP multicast csomagformátum megegyezik az unicast formátummal annyi eltéréssel, hogy a célcímek egy speciális osztálya van használva (IPv4 D osztály), amely egy speciális multicast csoportot címez. Mivel a TCP csak az unicastot támogatja, a multicastot használó alkalmazásoknak UDPt kell alkalmazniuk.

A multicast címzés implementálható mind WAN mind LAN környezetben. WAN esetén, mint például az Internetnél minden routernek az úton támogatnia kell a multicast címzést azzal, hogy implementál valamilyen multicast protokollt. Általában a multicast routing protokoll esetében fel van tételezve, hogy az út során minden router támogatja a multicast címzést, amely nem egy realisztikus feltételezés. A Multicast Backbone javasolt egy alternatív implementációt, amelynél csak arra van szükség, hogy a perem (edge) routereknek kell támogatnia a multicastot, míg a többinek nem szükséges.

LAN esetén a multicast közvetlenül telepíthető a hálózati kártyák driverének frissítésével, hogy megértsék a multicast keretet. Problémák léphetnek fel multicast vezetéknélküli LAN esetén való telepítése során, köszönhetően a vezetéknélküliség sajátosságainak, úgy mint sávszélességi korlát, csatorna interferencia.

Bárki aki multicast információt szeretne kapni, tudnia kell a multicast címet. Ezután küldeni fog egy kérést a lokális multicast routere felé, hogy csatlakozni kíván a multicast csoportba. Ha a csatlakozási folyamat sikeres volt, a csoport minden tagja képes lesz megkapni ugyanazt az információt a küldötől a multicast csoportcímen keresztül. A multicast routerek ki fogják választani az utat a használt routing algoritmusuk alapján, majd eldöntik, hogy replikálják-e az adatokat egy vagy több interfész felé.



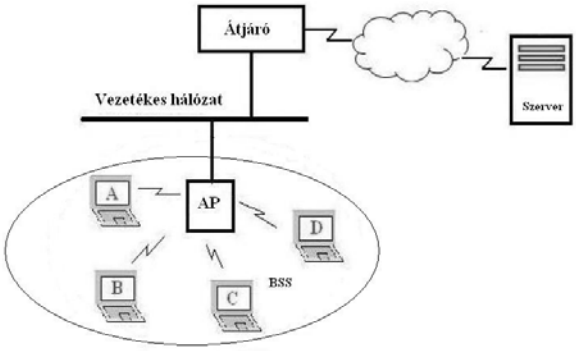
4.1. ábra. IGMP üzenet formátum

Az Internet Group Management Protocol (IGMP) az Internet Protocol (IP) támogatására van definiálva, hasonlóan mint az ICMP unicast kommunikáció esetében. Az IGMP a multicast routerek egymásközötti és a hosztokkal való információcserére szolgál. A csoport IP címe D osztályú, 224.0.0.0-tól 239.255.255.255-ig. Két fajta IGMP üzenet létezik: Report és Query. A Reportot a hoszt küldi, hogy csatlakozni szeretne a multicast csoporthoz, míg a Queryt a lokális router küldi rendszeresen, hogy felfedezze és fenttartsa a multicast csoport tagságot.

4.1.1. Multicast vezetéknélküli LAN felett

A vezetéknélküli LAN manapság igen elterjedt, a vézetéknélküli végfelhasználók száma drasztikus megnőtt köszönhetően a kivitelezés egyszerűségének valamint a kölcséghatékonyságának. A vezetéknélküli felhasználók szokásai megegyeznek a vezetékes felhasználókéval, így ők is ugyanúgy használnak multicastos alkalmazásokat, amelynél a tiszta siker nem garantálható vezetéknélküli LAN esetén.

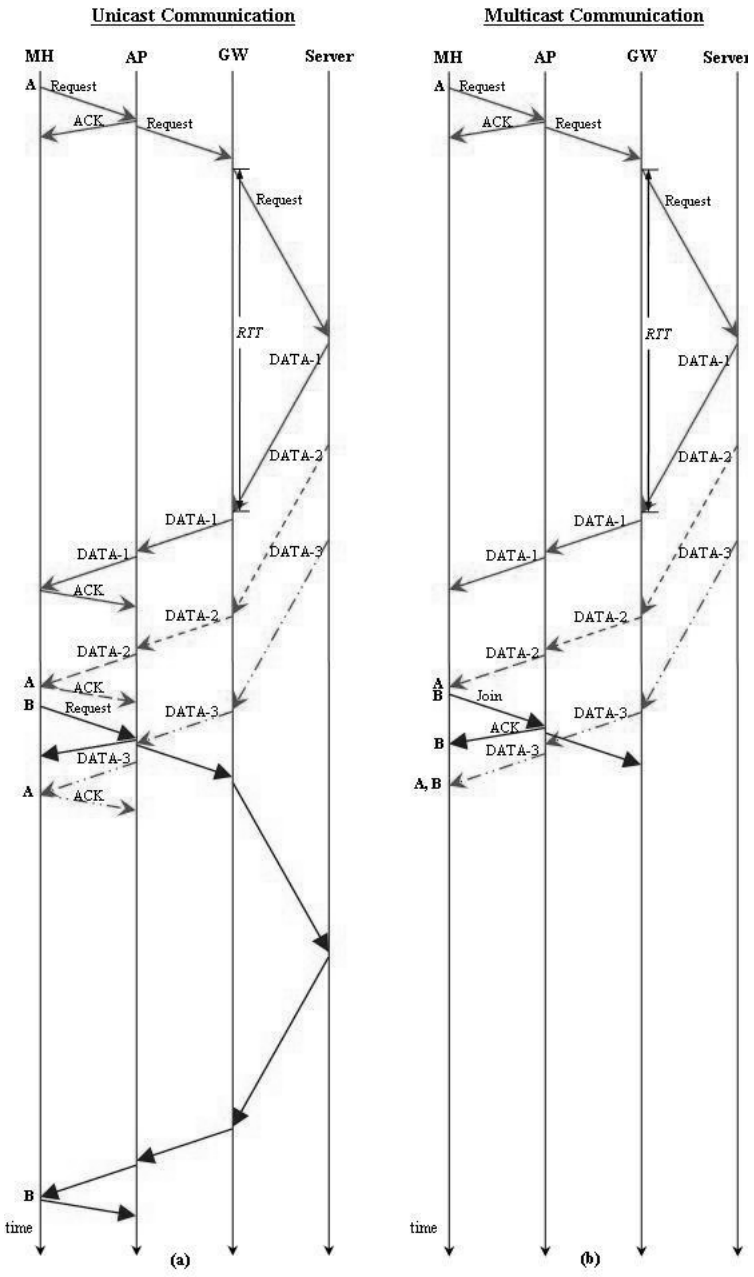
Az IEEE 802.11 a defacto szabvány a vezetéknélküli LANhoz. A mobile hostnak (MH) az alap szolgáltatás halmazban (basic service set, BSS) össze kell lennie kapcsolva egy hozzáférési ponttal (access point), hogy a hálózati szolgáltatásokhoz hozzáférjen. Az MH más hosztokkal csak a hozzáférési ponton keresztül tud kommunikálni, amely így híd is a vezetékes hálózathoz, mint a 4.1.1 ábrán is látható.



Vezetéknélküli LAN esetén való multicast használatnál a multicast szerver helye jelentős. Ha a multicast szerver a BSS-en belül helyezkedik el, akkor a szerver közvetlenül küld multicast kereteket az APnak unicast címzést használva, majd az AP továbbküldi a kapott keretet a csoport minden tagjának a csoport multicast címét használva. Az unicast keretek a küldőtől nyugtázva kell, hogy legyenek, hogy elkerüljék az újraküldést, azonban a multicast keretek az APtől a tagoknak nincsenek nyugtázva. Ezért nem történik újraküldést, amely megbízhatatlanná teszi a multicast kommunikációt. Ha a multicast szerver a BSS-en kívül helyezkedik el akkor minden keret a küldőtől az APn keresztül multicast keretformátummal lesz küldve, így az előbb említett megbízhatatlanság megmarad. Ezen probléma orvosolására számos megbízható multicast protokollt találtak ki.

4.1.2. Unicast és multicast idődiagramm

Az A és B mobile hosztok ugyanazon adat streamet kérik egy BSSen kívül helyezkedő szervertől. Unicast kommunikáció esetén B-nek tekintélyes időt kell várnia míg az első keret megérkezik, míg multicast esetén B-nek kevesebbet kell várnia, amely egy nagyon fontos tulajdonság és jól kivehető a mellékelt ábrán is.



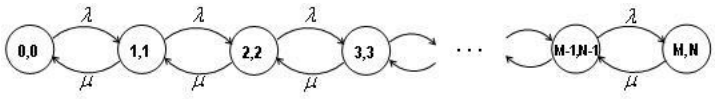
4.2. A sztochasztikus modell

Az alábbiakban a [6] cikkben találató modellt és eredményeket ismertetem. Legyen M a megengedett élő stream médiák száma (például TV csatorna), míg N a szimultán kért sessionök száma. $M(t)$ sztochasztikus folyamat jelölje a t időpillanatban megengedett élő stream médiák számát, az $N(t)$ sztochasztikus folyamat pedig jelölje a t időpillanatban szimultán kért sessionök számát. A későbbiekben feltesszük a következőket:

- A élő stream médiák kéréseinek érkezése Poisson-folyamat szerinti
- $M(t)$ és $N(t)$ független
- nincsen interferencia vagy rejtett állomás jelenség

4.2.1. Unicast kommunikáció

Legyen M és N a Markov-lánc két állapotváltozója, az érkezési intenzitás legyen λ és minden session számára a távozási intenzitás legyen μ. Az előzőekből következik, hogy minden session átlagos $\frac{1}{\mu}$ ideig lesz kiszolgálva.



Legyenek $P_{i,j} = \lim_{t\to\infty} P(M(t) = i, N(t) = j)$ a stacionárius állapotban lévő valószínűségek, és legyen λ,μ az érkezési és távozási intenzitás az i, j állapotban. Ha a rendszer tétlen, akkor a legbaloldalibb állapotban van, amelyre $M, N = 0, 0$. Ha egy új kérés érkezik be és $0 \leq i = j < N$, akkor eggyel jobbra lépünk, azaz az (i, j) állapotból az $(i + 1, j + 1)$ állapotba lépünk. Ha a session úgy dönt, hogy távozik vagy abbahagyja a vételt és $i = j > 0$ akkor a rendszer az (i, j) állapotból az $(i - 1, j - 1)$ állapotba kerül. A lényeg abban rejlik az unicast esetében, hogy ha új kérés érkezik függetlenül attól, hogy egy már sugárzott vagy nem sugárzott streamet kér, új adat sessiont kell létrehozni. Az előbbiből következik, hogy minden állapotra igaz, hogy $M = N$. Amint a megengedett streamelt médiák vagy konkurens sessionök elérik a maximumot, a további session kérelmek elu-

tasításra kerülnek. Ezekből következik, hogy

$$\lambda_{m,n} = \begin{cases} \lambda, & \text{ha } m = n = 0, 1, 2 \ldots, N-1 \\ 0, & \text{egyébként} \end{cases}$$

$$\mu_{m,n} = \mu, \text{ ha } m = n = 1, 2, \ldots, N$$

Állapotegyenletek

$$(\lambda + \mu)P_{i,j} - \lambda P_{i-1,j-1} - \mu P_{i+1,j+1} = 0$$

ha $0 < i = j < N$

$$\lambda P_{0,0} - \mu P_{1,1} = 0$$

ha $i = j = 0$

$$\mu P_{M,N} - \lambda P_{M-1,N-1} = 0$$

ha $i = M$,$j = N$

Valamint a normalizáló feltétel:

$$\sum_{i=1}^{M} \sum_{j=1}^{n} P_{i,j} = 1.$$

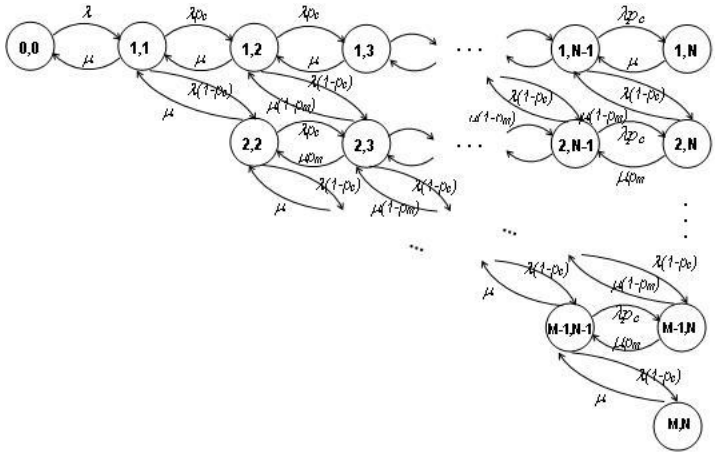
Ezen Markov-lánc ismerős lehet sorbanállás elméletből hiszen ez éppen a $M/M/1/K$ véges befogadóképességű rendszerrel egyezik meg. Így használjuk fel esetünkben a ott tanultakat, mégpedig, hogy

$$P_{k,k} = \begin{cases} \frac{1-\lambda/\mu}{1-(\lambda/\mu)^{N+1}} \left(\frac{\lambda}{\mu}\right)^k, & \text{ha } k \leq N \\ 0, & \text{egyébként} \end{cases}$$

4.2.2. Multicast kommunikáció

Multicast esetében a modell bonyolultabbá válik, ahogy az a következőkből ki fog derülni. Hasonlóan az unicast esethez, a rendszer tétlenségét a $(0, 0)$ állapot jelzi. Az első kérés beérkezésekor a rendszer az $(1, 1)$ állapotba lép az unicast esettel megegyezően, azonban ha a következő kérés az aktív streamhez való csatlakozást kéri (amelynek p_c a valószínűsége), az $M, N = 1, 2$ állapotba lépünk, tehát $(i, j) \rightarrow (i, j + 1)$. Ha a kérés egy új streamért érkezik (ennek valószínűsége $1-p_c$), akkor az $M, N = 2, 2$ állapotba lépünk, tehát $(i, j) \rightarrow$

$(i+1, j+1)$ alakú a változás. A közbenső átmenetek hasonlóan zajlanak, mindaddig amíg a megengedett elő streamek vagy konkurens sessionök el nem érik maximumukat. Ha valamely session távozik vagy megállítja a fogadást, akkor meg kell különböztetni, hogy az az utolsó session az adott streamhez vagy nem. Legyen p_m annak a valószínűsége, hogy az adott session nem az utolsó. Ebben az esetben $(i, j) \rightarrow (i, j-1)$ lépés történik. Ha az adott session az utolsó volt a streamnél (aminek $1 - p_m$ a valószínűsége) akkor az (i, j) állapotból a $(i-1, j-1)$ állapotba ugrunk. Ha elérnénk a megengedett streamek vagy kokurens sessionök maximumát, akkor az új sessionök blokkolva lesznek.



A következő táblázatban láthatóak a Markov-lánc nemnulla átmenetvalószínűségei:

$$
\begin{aligned}
P(i+1, j+1|i, j) &= \lambda & i &= j = 0 \\
P(i, j+1|i, j) &= \lambda p_c & i &\in [1, M-1], j \in [1, N-1] \\
P(i+1, j+1|i, j) &= \lambda(1-p_c) & i &\in [1, M-1], j \in [1, N-1] \\
P(i-1, j-1|i, j) &= \mu & i &= j \\
P(i, j-1|i, j) &= \mu & i &= 1, j \in [1, N] \\
P(i, j-1|i, j) &= \mu p_m & i &\in [2, M], j \in [2, N] \\
P(i-1, j-1|i, j) &= \mu(1-p_m) & i &\in [2, M-1], j \in [2, N]
\end{aligned}
$$

ahol, p_c annak a valószínűsége, hogy egy MH csatlakozik egy multicast csoporthoz, míg p_m annak a valószínűsége, hogy egy MH kilép a multicast csoportból, de nem az utolsó session a csoportban.

Állapotegyenletek

$$(\lambda+\mu)P_{i,j} - \lambda(1-p_c)P_{i-1,j-1} - \lambda p_c P_{i,j-1} - \mu p_m P_{i,j+1} - \mu(1-p_m)P_{i+1,j+1} = 0$$

ha $1 < i < j < N$

$$\lambda P_{0,0} - \mu P_{1,1} = 0$$

ha $i = j = 0$

$$(\lambda+\mu)P_{1,1} - \lambda P_{0,0} - \mu P_{1,2} - \mu P_{2,2} = 0$$

ha $i = j = 1$

$$\mu P_{1,N} - \lambda p_c P_{1,N-1} = 0$$

ha $i = 1$,$j = N$

$$(\lambda+\mu)P_{i,j} - \lambda p_c P_{i,j-1} - \mu P_{i,j+1} - \mu(1-p_m)P_{i+1,j+1}$$

ha $i = 1$,$1 < j < N$

$$\mu P_{i,j} - \lambda(1-p_c)P_{i-1,j-1} - \lambda p_c P_{i,j-1} = 0$$

ha $1 < i < M$,$j = N$

$$(\lambda+\mu)P_{i,j} - \lambda(1-p_c)P_{i-1,j-1} - \mu p_m P_{i,j+1} - \mu P_{i+1,j+1} = 0$$

ha $1 < i = j < M$

$$(\lambda+\mu)P_{M,j} - \lambda(1-p_c)P_{M-1,j-1} - \lambda p_c P_{M,j-1} - \mu p_m P_{M,j+1} = 0$$

ha $i = M$,$M < j < N$

$$\mu P_{N,N} - \lambda(1-p_c)P_{N-1,N-1} = 0$$

ha $i = j = N$

Normalizáló feltétel:

$$\sum_{i=1}^{M}\sum_{j=1}^{n} P_{i,j} = 1.$$

4.3. Hatékonysági vizsgálat

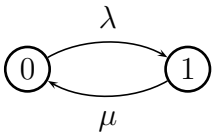
Az előző részben feállított modellt vizsgálata fog történni $M = 1$ esetén. Az egyszerűség kedvéért minden egyes streamelt mediának 1 Mbps szükséges, valamint, hogy a maximális konkurens sessionök száma 5.

4.3.1. Egy streamelt média, $M = 1$

Unicast

Ebben az esetben egyetlen session van engedélyezve és minden más session blokkolva lesz. Az átmenetdiagramm a következő ábrán látható.
A stacionárius megoldás:

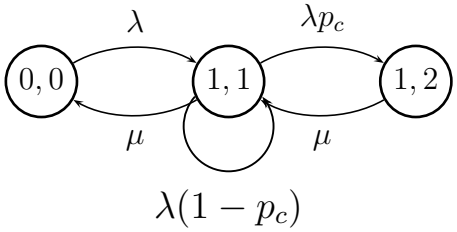
$$P_0 = \frac{\mu}{\lambda + \mu},\ P_1 = \frac{\lambda}{\mu + \lambda}.$$



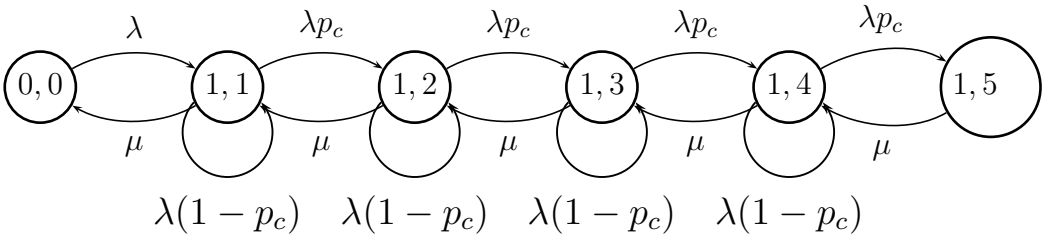
4.2. ábra. Unicast, $M = 1$

Multicast

Habár ebben az esetben is csak egyetlen stream médiánk van, itt egészen addig nem kerülnek blokkolásra a kérelmek amíg a maximális konkurens sessionök számát el nem érjük, ami esetünkben 2 és 5.



4.3. ábra. Multicast, $M = 1$,$N = 2$

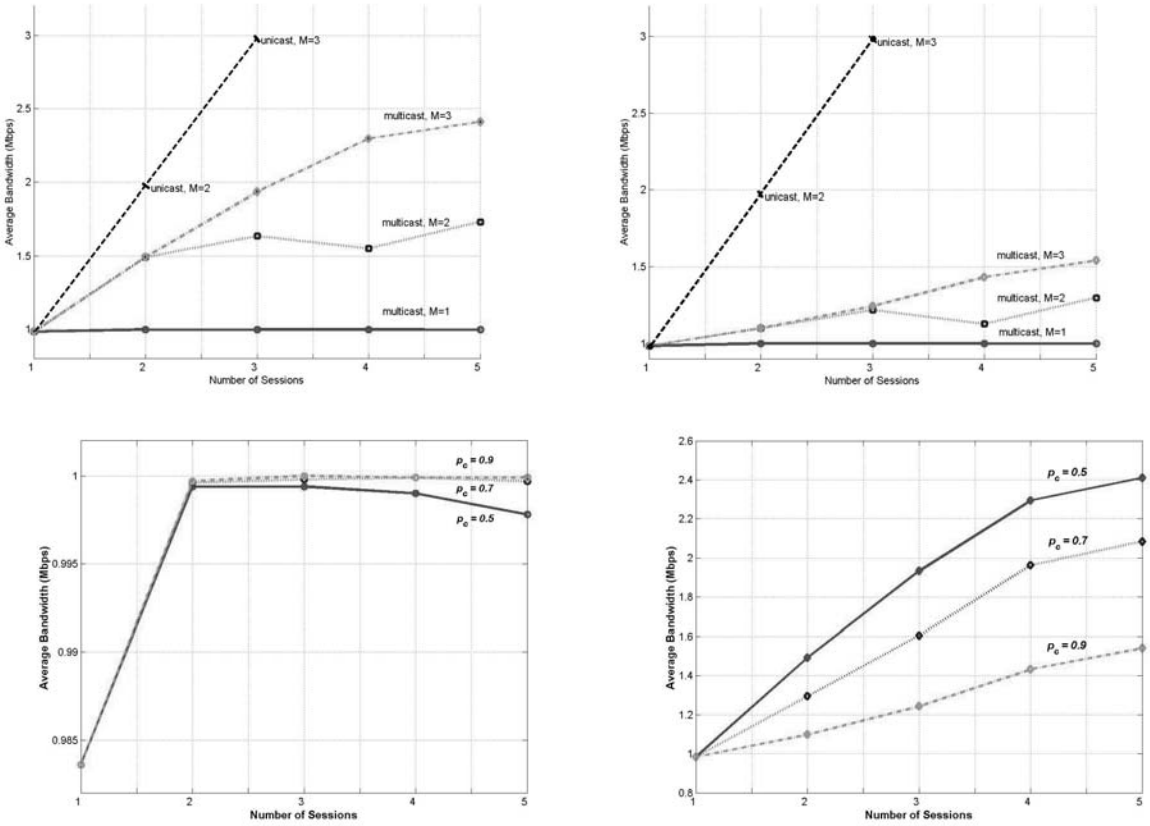


4.4. ábra. Multicast, $M = 1$,$N = 5$

$M = 1$ és $N = 2$ esetén a stacionárius állapotbeli valószínűségek

$$P_{0,0} = \frac{\mu^2}{\lambda^2 p_c + \lambda\mu + \mu^2},\ P_{1,1} = \frac{\lambda\mu}{\lambda^2 p_c + \lambda\mu + \mu^2},\ P_{0,0} = \frac{\lambda^2 p_c}{\lambda^2 p_c + \lambda\mu + \mu^2}.$$

4.4. Összegzés



Ábrák: [6]

A következőkben néhány numerikus példát fogunk bemutatni, ezzel szemléltetve az unicast és multicast közötti hatékonysági különbséget. A session távozási intenzitéás $\mu = 1/30$ perc, amely 30 perces átlagos session időt jelent, valamint legyen az beérkezési intenzitás

$\lambda = 2$ kérés percenként. A fenti ábrákon látható az átlagos sávszélesség sessionönként. Unicast esetén az átlagos sávszélesség $M = 1, 2, 3$ esetében egyszerűen $1, 2, 3$. Míg multicast esetén külön látható, hogy mi változik ha annak a valószínűségét, hogy a kérés egy már meglévő streamhez akkor csatlakozni $p_c = 0.5$-re és $p_c = 0.9$-re. $p_c = 0.5$ esetén, amint növeljük a streamek számát, multicast esetében úgy csökken az átlagos sávszélesség az unicasthoz képest, köszönhetően annak, hogy több session is ugyanazt kapja anélkül, hogy több sessionnek lenne elküldve külön. $p_c = 0.9$ esetén ez a különbség méginkább kiéleződik. Ezen adatokból kivehető, hogy minél nagyobb annak a valószínűsége, hogy az új kérés csatlakozik egy régihez, annál kevesebb lesz a szükséges sávszélesség sessionönként, valamint, hogy a mobile hosztok viselkedése nagyban befolyásolja a rendszer hatékonyságát, hiszen nem mindegy, hogy egy már használt streamhez csatlakozik az új session vagy nem. Egy másik fontos aspektus az összehasonlításban, hogy multicast esetén mivel átlagosan kevesebb sávszélesség kell egy sessionnek, így többen tudnak csatlakozni a megengedett streamekhez, így kevesebb kérés is lesz visszautasítva.

5. fejezet

Összegzés

Dolgozatom során láthattuk, hogy milyen jelentős eszköztárat kínál a Markov-láncok elmélete, amely felhasználásának talán legékesebb példája a Google "lelke" a PageRank algoritmus és mögötte álló modell. Remélem, hogy dolgozatom tanulmányozása után az olvasó is és arra a megállapításra jut, hogy akülönböző rendszerek milyen jól modellezhetőek Markov-láncok segítségével, valamint, hogy milyen kézzel fogható eredmények érhetőek el azok hatékonysági vizsgálatával (gondoljunk csak a két Aloha kihasználtságára).

Érdekesnek találtam, hogy milyen régi, alapvető problémák kerülnek más megvilágításba a gyakorlati problémák során, amelyeknek fő okozója a rendszerek óriási mérete. Gondoljunk csak a PageRank vektor meghatározásához elméletben egy Markov-lánc létező stacionárius eloszlásának a kiszámítása a feladat, de egy több milliárdszor több milliárdos mátrix esetén a megszokott módszerek sokszor nem működnek, így a kutatóknak új, adott problémára szabott megoldásokkal kell előállniuk.

Végezetül fontosnak tartom megjegyezni és kihangsúlyozni, hogy egzakt eredmények a matematika és óriási eszköztára nélkül nem szerezhetőek bármilyen informatikai rendszerekkel kapcsolatos problémáról is legyen szó. Úgy gondolom utóbbi állításomat sikerült szakdolgozatommal alátámasztani.

Irodalomjegyzék

[1] C. Langville, A. Meyer. Deeper inside pagerank. 2006.

[2] S. Page, L. Brin. The pagerank citation ranking: Bringing order to the web. 1998.

[3] A. Langville. http://www.cofc.edu/ langvillea/prdatacode/lang-perspagerank.m.

[4] P. Van Mieghem. *Performance Analysis of Communications Networks and Systems.* Cambridge University Press, 2006.

[5] G. Giambene. *Queueing Theoery and Telecommunications.* Springer, 2005.

[6] S. Phonphoem, A. Li-On. Performance analysis and comparison between multicast and unicast over infrastructure wireless lan. 2006.

[7] J. Sztrik. *Bevezetés a Sorbanállási elméletbe és alkalmazásaiba.* Kossuth Egyetemi Kiadó, 2000.

[8] A. Rényi. *Valószínűségszámítás.* Tankönyvkiadó, 1981.

[9] K.S. Trivedi. *Probability and Statistics with Reliability, Queueing and Computer Science Applications.* Prentice-Hall, Englewood Cliffs, 1982.

[10] G. Bolch, S. Greiner, H. de Meer, and K.S. Trivedi. *Queueing Networks and Markov Chains.* John Wiley and Sons, 1998.

[11] http://www.erg.abdn.ac.uk/users/gorry/course/intro-pages/uni-b-mcast.html.

[12] http://en.wikipedia.org/wiki/alohanet.