UNIVERZITET U BEOGRADU
MATEMATIČKI FAKULTET

Master Rad

Blumov filtar za p2p mreže

mentor:
prof. dr Miodrag Živković

student:
Saša Pršić
br. indeksa: 1011/2011

komisija:
dr Miodrag Živković red. prof.
dr Predrag Janičić red. prof.
dr Saša Malkov docent

2015.

Sadržaj

Poglavlje 1

Uvod

Blumov filtar je jednostavna, prostorno efikasna struktura podataka za predstavljanje skupa podataka koja omogućava upite o pripadnosti elemenata skupu. Blumov filtar prilikom upita o pripadnosti podatka skupu dozvoljava *lažan potvrdan* (eng. false positive) odgovor odnosno grešku. Prostorna efikasnost kompenzuje mogućnost pojave ovakvih grešaka; pri tome se može postići da verovatnoća pojave lažnog potvrdnog odgovora o pripadnosti elementa skupu bude dovoljno mala. Barton Blum (eng. Burton Bloom) je uveo Blumov filtar sedamdesetih godina prošlog veka i od tada se ova struktura podataka često koristi u aplikacijama za rad sa bazama podataka, a u skorije vreme i u mrežnim aplikacijama.

Nadmreža je pojam koji se koristi za mreže povezanih čvorova čija međusobna povezanost ne zavisi od topologije fizičke mreže koja ih povezuje kao ni od protokola kojim pojedinačni čvorovi komuniciraju. Nekada je i Internet bio sagrađen kao nadmreža koja je bazirana na telefonskoj mreži. P2P mreža je vrsta decentralizovane i distribuirane mrežne arhitekture u kojoj se pojedini čvorovi ponašaju i kao potrošači i kao snabdevači resursa. Za razliku od toga klijent – server arhitekturu karakteriše odnos saradnje programa u aplikacijama. Server komponenta pruža funkciju ili uslugu jednom ili više klijenata koji iniciraju takve zahteve.

Postoje različiti tipovi mrežnih aplikacija u kojima se primenjuje Blumov filtar: u saradnji *nadmreže* (eng. overlay) i *p2p* (eng. peer-to-peer) mreža; kod usmeravanja paketa Blumov filtar omogućava ubrzavanje i pojednostavljenje protokola za usmeravanje paketa; za sumiranje podataka o ruterima ili o nekim drugim mrežnim uređajima.

Zajedničko za ove mrežne aplikacije je to što je Blumov filtar sažet način reprezentovanja skupa ili liste. Odnosno, korišćenjem Blumovog filtra može se značajno smanjiti memorijski prostor potreban za čuvanje podataka, o čemu će biti reči u ovom radu. U ovom radu opisan je princip rada Blumovog filtra, njegove prednosti i nedostaci. U drugom poglavlju rada opisani su osnovni koncepti i arhitektura p2p mreža. U trećem i četvrtom poglavlju predstavljeni su princip rada i primena Blumovog filtra. U petom poglavlju je opisana primena Blumovog filtra u socijalnim mrežama, a u poglavlju šest dati su primeri kako

povećati sigurnost u mrežama. U sedmom poglavlju data je programska realizacija algoritama u programskom jeziku Java i diskusija dobijenih rezultata, na osnovu kojih su na kraju izvedeni zaključci.

Poglavlje 2

P2P Mreže

U poslednjoj deceniji P2p (skraćenica od eng. peer-to-peer) sistemi su zauzeli veliki udeo u Internet protoku. P2p mreže se mogu shvatiti kao velike distribuirane baze podataka. Na primer, problem pretraživanja sadržaja baziranog na različitim kriterijumima u visoko decentralizovanom sistemu nije bio dobro rešen. Početno rešenje je bilo bazirano na kompleksnim algoritmima, kakav je bio protokol Gnutella[1]. Istraživanjem distribuiranih sistema, mreža i upravljanja podacima tražilo se efikasnije rešenje pristupa i upravljanja podacima u p2p sistemima. Kao rezultat, dobijena je nova arhitektura sistema i novi skup algoritama. Najznačajnije prednosti koje su motivisale ova istraživanja su: skalabilnost, minimalna cena komunikacije i decentralizovana kontrola. Ideja korišćenja Blumovog filtra je smanjenje protoka informacija prilikom komunikacije između čvorova kao i smanjenje memorijskog prostora potrebnog za čuvanje informafija na samim čvorovima. Princip p2p sistema se koristi u širokom kontekstu i implicira različite tehničke koncepte i prilaze [1].

U današnje vreme je popularna decentralizovana razmena velike količine digitalnog sadržaja (razmena fajlova bez posrednika, odnosno p2p razmena). P2p mreža ima značajne prednosti u poređenju sa standardnim klijent-server sistemima (p2p sistem se automatski skalira sa brojem korisnika za razliku od klijent-server sistema). Primeri primene p2p mreža su mreža *Skajp* (eng. Skype) za telefoniranje i distribuiranje različitog digitalnog sadržaja kroz *Bittorent* (eng. Bittorrent). Razlika između p2p mreža i centralizovanog sistema je u tome što su p2p mrežni sadržaj i informacije distribuirani u različitim elementima širom mreže. Kako je popularnost p2p mreža rasla, tako je i Blumov filtar dobijao sve veći značaj u p2p aplikacijama pre svega zbog efikasnog korišćenja *mrežnog protoka* (eng. bandwith) i prilikom komunikacije čvorova i slanja skupa podataka.

2.1 Koncepti

Uopšteno govoreći, p2p sistem je sistem koji sadrži skup P čvorova (eng. peers) koji izvršavaju zajednički zadatak i povezani su sa susedima u logičkoj mreži koja se zove p2p nadmreža ili mreža. Čvorovi komuniciraju sa svojim susedima

[1] Gnutella - prva decentralizovana p2p mreža.

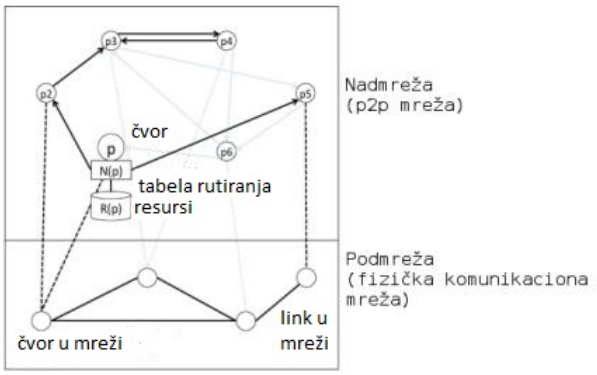
preko komunikacione mreže, koja se zove *osnovna mreža* (eng. underlay network). Komunikaciona mreža je obično Internet, ali se mogu koristiti i druga sredstva komunikacije kao što su *bežične* (eng. wireless [2]) mreže. U idealnom slučaju, pretpostavlja se da različiti čvorovi imaju ekvivalentne uloge, stanje je distribuirano između čvorova i kontrola je decentralizovana. Drugim rečima, ne postoji entitet koji ima pristup globalnom pogledu na sistem, ili ima globalnu kontrolu. Kao rezultat, odluke su bazirane na lokalnim informacijama koje imaju čvorovi. P2p sistemi se često smatraju samoorganizujućim sistemima. Prva osnovna funkcija p2p sistema je održavanje nadmreže. Svaki čvor $p \in P$ održava skup suseda $N(p) \subseteq P$ za koji zna njihovu fizičku adresu u osnovnoj mreži. Struktura podataka koja se koristi za čuvanje informacija o susedima se naziva tabela usmeravanja. Različite arhitekture osnovne mreže se razlikuju po strukturi i po sadržaju svojih tabela usmeravanja. Karakteristika p2p mreža je mali skup suseda. Zavisnost skupa suseda od ukupnog skupa čvorova $n =\mid P \mid$ je konstantna ili logaritamska. Čvorovi mogu samostalno da odluče da li će se priključiti p2p sistemu ili će ga napustiti. Proces priključivanja i napuštanja se naziva promena statusa (eng. churn). Prilikom priključivanja čvora $p \in P$ p2p mreži, čvor p mora da kontaktira čvor $q \in P$ koji je trenutno deo mreže. Zato ova operacija ima potpis $join(p, q)$. Čvorovi u svakom trenutku mogu da napuste mrežu tako da ova operacija ima potpis $leave(p)$. Protokol održavanja nadmreže obezbeđuje da prilikom priključivanja i napuštanja čvorova struktura nadmreže ostane očuvana. Na primer, čvorovi imaju tabele rutiranja koje zadovoljavaju ograničenja nasleđena iz arhitekture podmreže. Protokol održavanja nadmreže je distribuiran, što ima za rezultat decentralizovanu kontrolu p2p nadmreže.

P2p sistemi se najčešće koriste za obezbeđivanje pristupa ili razmene skupa resursa R (dokumenti, medijski sadržaji, podaci, fizički resursi...) U fizičke resurse spadaju: prostori za skladištenje, računarski kapacitet ili mrežni protok. Takvi sistemi se nazivaju p2p sistemi za deljenje resursa i oni podržavaju osnovne funkcije za upravljanje resursima. Resursi u p2p sistemima se identifikuju preko jedinstvenog resursnog identifikatora koji se nalazi u prostoru identifikatora resursa I(R).

- *insert(p, r)*: dodaje resurs $r \in R$ čvoru $p \in P$.
- *lookup(p, id(r))*: p_r zahtev za čvor $p \in P$ da vrati jednu adresu čvora $p_r \in P$ koji drži resurs sa identifikatorom $id(r) \in id(I(R))$.
- *query(p,q)*: p_q izdat upit q čvoru $p \in P$ da vrati fizičku adresu skupa čvorova $p_q \subseteq P$ koji drži resurse koji zadovoljavaju određene uslove upita.

U cilju implementacije ovih funkcija, p2p mreže koriste protokole rutiranja za lociranje čvorova koji drže određene resurse. Protokoli rutiranja su distribuirani algoritmi koji koriste strukturu nadmreža za razmenu poruka između čvorova. Nadmreže i protokol rutiranja su od velike važnosti u p2p arhitekturi sistema. Čvorovi su specifični tip resursa. Oni se identifikuju pomoću identifikatora iz skupa identifikatora $I(P)$. Isti skup operacija koji važi za resurse važi i za čvorove. Na primer, operacija *pronalaženja* (eng. lookup) bi imala formu $lookup(p, id(p_s))$: p vrati fizičku adresu datog čvora koji ima identifikator $id(p_s)$ [1]. Koncept pristupa resursu je prikazan na slici 1.

[2] Wireless - mreže koje koriste radio talase za prenos podataka.



Slika 2.1: Koncept pristupa resursu.

2.2 Arhitektura sistema

P2p nadmreže su veoma važan deo p2p sistema. Svaka funkcija p2p sistema je implementirana kao distribuirani algoritam koji omogućava korišćenje nadmreže, pomoću koje se može pristupiti svakoj informaciji o stanju koje je prisutno u p2p sistemu. Postoje tri glavne vrste nadmrežnih arhitektura:

- Nestrukturirane nadmreže
- Strukturirane nadmreže
- Hijerarhijske nadmreže

2.3 Nestrukturirane nadmreže

U nestrukturiranim nadmrežama, izbor suseda čvora nije ograničen stanjem njihovih suseda, odnosno susedni čvorovi se slobodno mogu spojiti nakon dobijanja fizičke adrese. Slično tome, i izbor resursa koje čvor može da kontroliše nije ograničen stanjem čvorova. U čistim nestrukturiranim nadmrežama, čvor ne sadrži nikakvu informaciju o stanju suseda u svojoj tablici usmeravanja. Međutim, ta pretpostavka često utiče na optimizaciju performansi nestrukturiranih nadmreža.

Nestrukturirane nadmreže obično izazvaju veliki promet poruka za obavljanje pretraživanja. Za njihov rad nema posebnih ograničenja, osim izbegavanja prekida mreže, koje mora biti zadovoljeno[1].

2.4 Strukturirane nadmreže

Strukturirane nadmreže se zasnivaju na organizaciji čvorova i resursa s nekim zajedničkim, specifičnim prostorom identifikatora I, odnosno, $I = I(P) = I(R)$. Identifikatori resursa i čvorova se preslikavaju na identifikator prostora korišćenjem heš funkcije $h_r : R \to I$ i $h_p : P \to I$.

Izbor suseda čvora i odgovornost za upravljanje odgovarajućim resursima, zavise od identifikatora čvora. Tablice rutiranja obično sadrže linkove *kratkog*

dometa (eng. short-range links) koji upravljaju susedima čvora u prostoru identifikatora, koji osiguravaju osnovnu povezanost u mreži i linkove dalekog dometa (eng. long-range links), koji ubrzavaju pretraživanja resursa po identifikatoru. Značajna razlika u odnosu na nestrukturirane nadmreže je da se te pretrage obavljaju pohlepnim usmeravanjem. Kriterijum usmeravanja pretrage jeste prosleđivanje poruke na čvor koji najviše umanjuje udaljenost do cilja pretrage. Poruke pretrage se ne prosleđuju na sve čvorove već samo na one koji zadovoljavaju ovaj kriterijum. [1].

Za strukturirane nadmreže sprovedeno je istraživanje mrežnih performansi i stabilnosti u realnim uslovima u mreži.

- *Održavanje.* Važna karakteristika performansi je ponašanje strukturirane nadmreže prilikom priključivanja i izlaženja čvorova iz mreže (*churn*) [2]. Pošto se u strukturiranoj nadmreži nameću stroga ograničenja, održavanje tih ograničenja može postati teško u slučaju da učestalost promena statusa čvorova (*churn*) prelazi određeni procenat. Tada troškovi održavanja mogu postati preveliki, ili čak može biti nemoguće održavati konzistentnu mrežnu strukturu.

- *Uravnoteženje opterećenja.* Čak i kada se koriste jedinstvene heš funkcije za distribuciju resursa među čvorovima, dešava se da opterećenje čvorova nije ravnomerno [3]. Postoje više tehnika da se to popravi: uključujući virtualne čvorove, izbor dve mogućnosti [4] kao i korišćenje višestrukih heš funkcija.

- *Replikacije.* Imajući u vidu to da su čvorovi često nestabilni, resursi mogu postati nedostupni ako nisu replicirani unutar p2p sistema. Tipična šema replikacije ne čuva resurs samo u čvoru odgovornom za resurs, već i na nekim susednim čvorovima kako bi se povećala dostupnost.

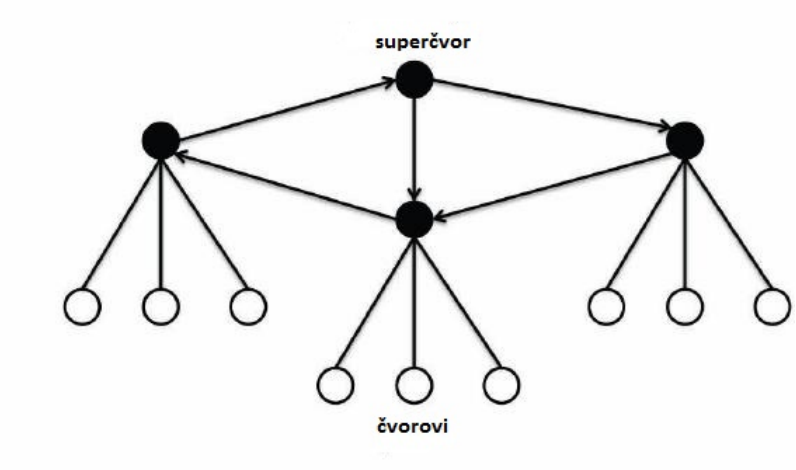
- *Usmeravanje bazirano na udaljenosti čvorova.* Logički linkovi u nadmreži mogu povezati čvorove koji su daleki u mreži, tj. povezani su dugim putanjama u fizičkoj mreži. Ako je usmeravanje bazirano na udaljenosti čvorova, tada je struktura nadmreže usklađena sa podmrežom, tako da susedni linkovi u nadmreži imaju i kratke veze na fizičkoj mreži.[5].

2.5 Hijerarhijske nadmreže

U praksi, čvorovi imaju različite performanse: jedni čvorovi su nestabilni i ostaju u p2p sistemu veoma kratko vreme, drugi čvorovi pokazuju više serverskih karakteristika i ostataju u sistemu duže vreme. Na osnovu tih karakterictika, većina široko korišćenih p2p sistema, kao što su: Gnutella, eMule[3] i Skype ima hijerarhijski pristup [6]. Jaki čvorovi, tzv. *superčvorovi* (eng. superpeers), preuzmaju veći deo opterećenja u odnosu na ostale čvorove. U osnovi superčvor arhitekture je podskup $S \subset P$ svih čvorova koji preuzimaju ulogu superčvorova. Superčvorovi se povezuju u nestrukturiranu ili strukturiranu nadmrežu kako bi rutirali globalne zahteve za pretraživanje. Primer organizacije superčvor

[3]eMule je p2p aplikacija za razmenu podataka za Microsoft Windows operativni sistem

arhitekture je prikazan na slici 2. Obični čvorovi povezuju se sa jednim ili više superčvorova i prosleđuju zahteve za pretraživanje i ažuriranje superčvorovima sa kojima su povezani. Superčvorovi obično održavaju globalni indeks svih resursa dostupnih na običnim čvorovima koji su povezani sa njima. Osim toga, obični čvorovi koji su povezani sa superčvorovima mogu formirati lokalnu nadmrežu za obradu zahteva za pretraživanje. Samo ako lokalni zahtevi ne mogu biti uspešno rešeni, superčvor ih prosleđuje na globalnu superčvor nadmrežu [1].



Slika 2.2: Primer organizacije superčvor mreže. Superčvorovi su povezani pomoću nestrukturirane ili strukturirane mreže.

2.6 Upravljenje podacima u P2P mrežama

P2p upravljanje podacima je specifičan oblik distribuiranog upravljanja podacima, koje se smatra i distribucijom podataka velikih razmera sa decentralizovanim pristupom. U p2p upravljanju podacima, barem jedan deo sistema je implementiran korišćenjem p2p arhitekture. Istorijski posmatrano p2p upravljanje podacima ima nekoliko korena. Prvi koren se zasniva na proširenju osnove p2p sistema bogatijim pretraživačkim sposobnostima, kao što je opisano u prethodnom delu. Budući da je lokacija resursa ključni problem p2p sistema, oduvek je postojao problem pretrage. Poboljšanje mogućnosti pretraživanja iz jednostavnog pretraživanja na osnovu ključa do strukturiranog i pretraživanja na osnovu sličnosti započeto je u distribuiranim sistemima. Drugi koren se može pripisati usvajanju p2p idejnog rešenja za distribuirane sisteme, Veb baze podataka. Treći koren se može naći u području distribuiranog prikupljanja informacija. S obzirom da Veb prikupljanje podataka zahteva vrlo skalabilna rešenja, razmotreni su p2p arhitekturski koncepti za ovaj problem.

P2p sistemi su razvijeni za izgradnju velikih distribuiranih sistema oslanjajući se na decentralizovano upravljanje i samoorganizaciju. [1]. Neke od najjednostavnijih pretpostavki, koje su temelj za razvoj novih tehnika, glase:

- *Sistem velikih razmera:* distribuirane baze podataka i informacioni sistemi obično su male ili srednje veličine i uključuju i do nekoliko stotina čvorova. P2p arhitektura sistema ima i do milion čvorova [1].

- *Decentralizacija:* tradicionalno distribuirano upravljanje podacima obično uključuje neke oblike centralizovanog nadzora, što može podrazumevati zastoj za sisteme velikih razmera, što predstavlja nedostatak ukoliko dođe do otkazivanja ove centralne tačke [1].

- *Cena komunikacije:* u distribuiranom upravljanju podacima, komunikacija između čvorova ima manje važnu ulogu u odnosu na p2p sistem, gde se prenos liste smatra dominantnim troškom. Ovaj je važno kod p2p preuzimanja, kada treba preneti potencijalno duge liste [1].

- *Stabilnost:* u distribuiranim sistemima upravljanja podacima pretpostavlja se da su neuspesi retka pojava, dok se p2p sistemima često dešavaju[1].

2.7 Arhitekture

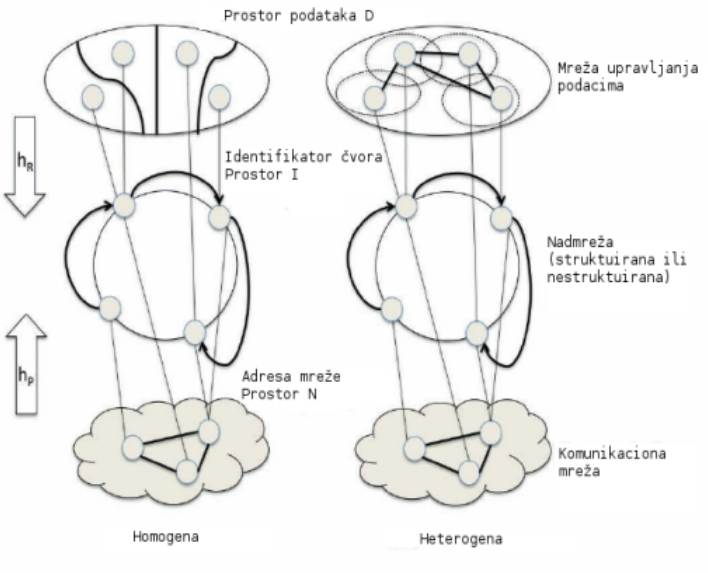
Arhitektura p2p sistema za upravljanje podacima uvodi novi sloj upravljanja podacima iznad p2p sistema. P2p sistem pruža usluge sloju upravljanja podacima, čime se ostvaruju različite vrste nezavisnosti podataka. Fizička nezavisnost podataka znači da operacije pristupa podacima, kao što je pretraga po vrednosti, zavise od same implementacije u aplikaciji a ne od p2p sistema. P2p nadmreže podržavaju nezavisnost podataka i razlikuju se dva tipa:

- *Efikasan pristup podacima.* P2p nadmreže pružaju usluge koje odgovaraju primitivama za pristup podacima koje su korišćene u tradicionalnim sistemima za upravljanje podacima radi ostvarivanja fizičke nezavisnosti podataka. To se odnosi na indeksiranje, kao što su heš tabele ili B+ stabla. Te primitive podržavaju efikasan pristup podacima.

- *Izolacija od mrežne dinamike.* P2p sistem obrađuje mrežnu dinamiku, kao što je priključivanje čvorova mreži ili napuštanje iste. Promenom topologije mreže menjaju se i performanse sistema, međutim ti fenomeni postaju nezavisni od sloja upravljanja podacima.

U zavisnosti od toga kako je organizovan sloj upravljanja, razlikuju se dva glavna skupa p2p sistema za upravljanje podacima. U oba slučaja, pretpostavka je da podaci odgovaraju nekom modelu podataka, što može biti strukturirani ili višedimenzionalni model podataka. Podaci takvog modela se nazivaju *prostor podataka (D)*. U *homogenim* p2p sistemima za upravljanje podacima, čvorovi zajednički upravljaju jednom distribuiranom logičkom bazom podataka $R_D \subseteq D$. Različiti čvorovi imaju jednako logičko pravo pristupa fizički distribuiranom podatku R_D. Objekti podatka u R_D se smatraju resursima upravljanja p2p

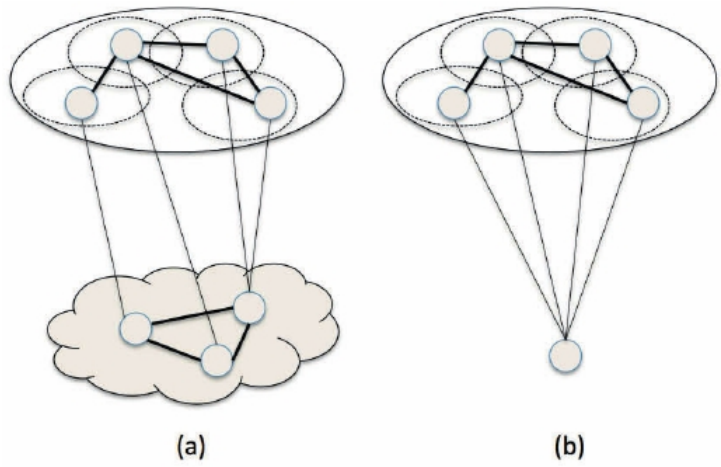
sistema. Podaci se preslikavaju iz prostora podataka D na prostor identifikatora čvorova I pomoću preslikavanja $h : D \to I$. P2p nadmreža p2p sistema omogućuje distribuirane pristupne putanje. Ovaj primer je opisan na slici 2.3 (levo). Arhitektura p2p sistema se sastoji od sloja nadmreže na vrhu komunikacione mreže a čvorovi su preslikani u identifikator prostora I pomoću heš funkcije $h_p : P \to I$. Ova vrsta sistema je analogna klasičnom sistemu distribuiranih baza podataka [1].



Slika 2.3: Generička struktura p2p sistema za upravljanje podacima.

U *heterogenom* p2p sistemu za upravljanja podacima, čvorovi iz sloja upravljanja podacima upravljaju različitim logičkim bazama podataka, odnosno, svaki čvor p ima drugačiji pogled $R_D(p)$ na prostor podataka. Kao posledica, ista operacija izvršena nad različitim čvorovima može proizvesti različite rezultate, što je suprotno u odnosu na homogene p2p sisteme za upravljanje podacima. Ova arhitektura je prikazanana slici 2.3 (desno). Okruženju svakog čvora, odgovara podprostor prostora podataka kojim se može pristupiti i označen je elipsom oko čvorova. Osim toga, čvorovi su povezani sa drugim čvorovima u p2p mreži u sloju upravljanja podacima.

U heterogenom p2p sistemu za upravljanja podacima, odnos između mreže upravljanja podacima i nadmreže je analogan odnosu između nadmreže i fizičke mreže p2p sistema. Na slici 2.4 (a) prikazan je slučaj podudaranja strukture mreže za upravljanje podacima i nadmreže. Ovo se obično dešava kada se identifikator prostora mrežnog sloja preslikava na prostor podataka aplikacije. Slika 2.4 (b) prikazuje primer u kojem se p2p interakcija odvija samo na logičkom nivou, dok je fizički sistem implementiran u jednom, centralizovanom mrežnom čvoru [1].



Slika 2.4: Specijalni slučaj generičke arhitekture heterogenog sistema za upravljanja podacima (a) podudaranja strukture mreže za upravljanje podacima i nadmreže, (b) odvijanje p2p interakcije na logičkom nivou.

Poglavlje 3

Blumov filtar

Blumov filtar je struktura podataka koja može da skladišti elemente skupa na prostorno efikasan način, ako je prihvatljiva mala greška prilikom testiranja pripadnosti nekog elementa u Blumovom filtru. Pored osnovne verzije u ovom poglavlju su predstavljene i izvedene verzije filtra.

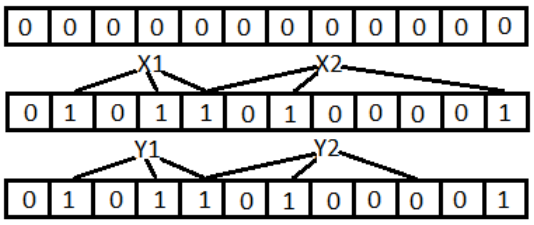
3.1 Osnovna verzija Blumovog filtra

Blumov filtar je prostorno efikasna struktura podataka koja se može koristiti za predstavljanje skupa elemenata, koja rešava problem brze provere pripadnosti elementa u velikom skupu podataka.

Problem. *Za dati skup $S = \{x_1, x_2, \ldots, x_n\} \subseteq U, |U| = N$, predstaviti skup S korišćenjem malog memorijskog prostora, tako da se na efikasan način može odgovoriti da li proizvoljni element x pripada skupu S .*

Blumov filtar je nazvan po Bartonu H. Blumu, koji je utvrdio da je za novi tip heš tabela, koji je sada poznat kao Blumov filtar, potrebno manje vremena da se ustanovi da li elementi pripadaju tabeli i manje prostora za skladištenje ovih elemenata. Međutim, nedostatak Blumovog filtra je taj što je moguća greška kada se testira prisutnost elemenata u filtru, odnosno pojava lažnog pozitivnog odgovora. Za testiranje pripadnosti elemenata skupu može se koristiti obična heš funkcija, ali to daje nekontrolisanu verovatnocu laznog odgovora.

Blumov filtar se sastoji od niza bitova veličine m, koji su prvobitno inicijalizovani na 0. Dodavanje elemenata iz skupa $S = \{x_1, x_2, \ldots, x_n\}$ se vrši na sledeći način. Za svaki element x_i koji je dodat, koristi se k različitih heš funkcija $h_1, \ldots, h_k$ sa opsegom $\{1, \ldots, m\}$ za računanje k različitih heš vrednosti $h_1(x_i), \ldots, h_k(x_i)$. Pretpostavlja se da ove heš funkcije preslikavaju svaki element u slučajni broj koji je uniformno raspodeljen na tom opsegu. Zatim se bit $h_j(x_i)$ postavlja na 1, za $j = 1, \ldots, k$. Bitovi Blumovog filtra mogu se postavljati na 1 više puta, ali samo prvo postavljanje ima efekta. Provera da li se element y nalazi u skupu obavlja se na sličan način. Istih k heš funkcija se računaju za ključ elementa y, pa se proverava da li su svi bitovi $h_i(y)$, $i = 1, 2, \ldots, k$, jednaki 1. Ako su svi bitovi 1, element je u skupu, iako postoji mala verovatnoća da taj element nije u skupu zbog dodavanja drugih elemenata. U tom



Slika 3.1: Šematski prikaz rada Blumovog filtra.

slučaju dobija se lažan pozitivni odgovor, odnosno odgovor da se element nalazi u skupu, a on se u stvari ne nalazi [7]. Ako je jedan ili više tih bitova 0, element y svakako nije u skupu.

Primer rada filtra ($m = 12, k = 3$) može se videti na slici 3.1. filtar je na početku prazan i sve vrednosti su inicijalizovane na 0. Za svaki element x_i izračunava se k heš vrednosti ključa tog elementa, pa se na dobijene lokacije u tabeli upisuju jedinice. Da bi se proverilo da li se element y nalazi u skupu, proverava se da li su na svim k mestima (heš vrednosti ključa elementa) upisane jedinice. Smatra se da element y_1 pripada skupu, jer je u tabeli na k odgovarajućih mesta jedinica. Element y_2 nije u skupu zato što se 0 nalazi na mestu jednog bita.

Konstrukcija Blumovog filtra omogućuje kompromis između verovatnoće f lažnog pozitivnog odgovora i veličine m Blumovog filtra. Verovatnoća pojave lažnog pozitivnog odgovora se računa na sledeći način. Verovatnoća da jedan fiksirani bit od m bitova ostane 0 je:

$$1 - \frac{1}{m}$$

korišćenjem k heš funkcija i nakon dodavanja n elemenata ta verovatnoća postaje:

$$p = \left(1 - \frac{1}{m}\right)^{kn} \approx e^{\frac{-kn}{m}}$$

Verovatnoća lažnog pozitivnog odgovora f jednaka je verovatnoći da svi k bitovi koji se testiraju budu 1, odnosno

$$f = (1-p)^k = (1 - (1 - \tfrac{1}{m})^{kn})^k \approx (1 - e^{\frac{-kn}{m}})^k$$

Koristi se činjenica da je $e^{\frac{-kn}{m}}$ dobra aproksimacija za $(1 - \frac{1}{m})^{kn}$. Verovatnoća lažnog pozitivnog odgovora opada sa povećanjem dužine Blumovog filtra m. Verovatnoća pojave greške se povećava sa dodavanjem novih elemenata n. Minimizacija verovatnoće greške lažnog pozitivnog može se postići smanjenjem $(1-e^{\frac{-kn}{m}})^k$ u odnosu na k. Za optimalnu vrednost za k izvod ovog izraza je 0. Za dato m i n, vrednost k koja minimalizuje verovatnoću f lažnog pozitivnog odgovora je

$$k = \frac{m}{n} \ln 2 \approx \frac{9m}{13n}, \tag{1.1}$$

Kao rezultat se dobija da verovatnoća lažnog pozitivnog odgovora iznosi:

$$(\tfrac{1}{2})^k \approx 0.6185^{\frac{m}{n}},$$

Verovatnoća p da neki bit bude 0 iznosi

$$p \approx e^{\frac{-kn}{m}} = \tfrac{1}{2} \tag{1.2}.$$

Optimalna vrednost za k izračunata jednačinom (1.1) nije uvek prirodni broj, tako da je najbolja vrednost jedan od dva najbliža cela broja. Najpraktičnije je uzeti za k prirodni broj manji od optimalne vrednosti, jer je tada potrebno manje računanja za dodavanje i testiranje elemenata u Blumovom filtru [8].

U tabeli 1 su navedene optimalne vrednosti k i odgovarajuće verovatnoće lažnog pozitivnog odgovora.

Tabela 1. Verovatnoća lažnog pozitivnog odgovora za Blumove filtre različitih veličina i optimalnog broja heš funkcija.

odnos veličina filtra i skupa	m/n	2	8	16	24
optimalan broj heš funkcija	k_{opt}	1.39	5.55	11.1	16.6
broj heš funkcija	k	1	5	11	16
verovatnoća pojave lažnog pozitivnog odgovora	f	0.393	0.0216	$4.59 \cdot 10^{-4}$	$9.84 \cdot 10^{-6}$

3.2 Brojački Blumov filtar

U osnovnoj verziji Blumovog filtra opisano je kako je moguće skup elemenata S predstaviti Blumovim filtrom $F(S)$. Jedan po jedan, elementi iz S se dodaju u $F(S)$, postavljanjem niza bitova na 1. Međutim, brisanje elemenata iz $F(S)$ nije moguće. Ako element y hoćemo da obrišemo, bitovi na lokaciji $h_i(y)$ za $i = 1, \ldots, k$ se mogu proveriti, ali se ti bitovi ne mogu postaviti na 0, jer su oni mogli da se postave na 1 dodavanjem drugih elemenata različitih od y. Ovo dovodi do nemogućnosti brisanja iz standardnog filtra. Problem se može rešiti brojačkim Blumovim filtrom koji se pokazao kao osnova za efikasan protokol za proksi servere prilikom razmene keš memorije. Da bi ovo bilo moguće, proksi[1] serveri dele svoje podatke međusobno. Pregled sadržaja keš[2] memorije je u osnovi Blumov filtar koji opisuje skup URL-ova koje je proksi server keširao u datom trenutku. URL-ovi moraju da budu obrisani iz Blumovog filtra (na primer, kada su keširani URL-ovi zastareli). Za tu svrhu se koristi brojački Blumov filtar.

U brojačkom Blumovom filtru, umetanje elemenata predstavlja uvećavanje brojača. Niz bitova dužine m se zamenjuje nizom od m brojača, a svaki brojač

[1] Proksi server je server koji drugim računarima na posredan način dozvoljava pristup sadržajima na Internetu.

[2] Keš memorija je brza memorija malog kapaciteta u kojoj se čuvaju podaci koji se često koriste.

C_i za $i = 1, \ldots, k$ ima b bitova. Dodavanjem ili brisanjem elementa y, vrednost k brojača se ili uvećavaju ili umanjuju za 1. Određivanje broja elemenata $x \in S$ podrazumeva izračunavanje C_i brojača i uzimanje najmanje vrednosti. Brisanje elemenata, odnosno umanjenje brojača može dovesti do pojave lažnog negativnog odgovora, jer postavljanjem bita na 0 koji je bio deo k heš funkcija nekog elementa y koji smo obrisali, neće prijavljivati da element x pripada skupu S, jer će heš funkcija $h_i(x)$ na tom mestu biti 0.

Problem nastaje kada dođe do prekoračenja brojača, odnosno kada vrednost brojača dostigne vrednost $2^b - 1$ i ne može da se više uvećava jer dolazi do prekoračenja. Uobičajeno rešenje prilikom prekoračenja je da se brojač ostavi na maksimalnoj vrednosti, što dovodi do male verovatnoće pojave lažnog negativnog odgovora kada se brojač umanji do 0. Veličina b brojača treba da bude dovoljno velika da garantuje malu verovatnoću prekoračenja. Ovaj i naredni deo teksta može se naći u referenci [8]

3.3 Komprimovani Blumov filtar

Ukoliko je potrebno filtar preneti kao poruku onda se on treba komprimovati kako bi se njegova veličina smanjila. P2p razmena podataka koristi komprimovane Blumove filtre za distribuciju tabela rutiranja. Ukoliko se za smanjenje lažnog pozitivnog odgovora koristi optimalna vrednost broja heš funkcija k tada je verovatnoća da je bit postavljen na 1 jednaka $\frac{1}{2}$. Ovo znači da je niz bitova nasumičan i ne može dobro da se komprimuje.

Osnova komprimovanog Blumovog filtra je u tome da promenom rasporeda bitova u filtru, on se može komprimovati i na taj način podaci se mogu lakše preneti. Ovo se postiže tako što se broj heš funkcija k bira na taj način da bitovi vektora imaju manju verovatnoću od $\frac{1}{2}$. Posle prenosa podataka, filtar se dekomprimuje. Veličina k nije optimalna vrednost za nekomprimovani Blumov filtar, ali kao rezultat njenim korišćenjem može se dobiti manji komprimovani filtar.

3.4 Blumier filtar

Blumov filtar dozvoljava samo upite o članstvu elemenata koji se nalaze u skupu $S = x_1, \ldots, x_k$. Za razliku od njega, Blumier filtar može da kodira funkciju $f(x)$ dodeljivanjem vrednosti svakom elementu $x \in S$.

Na primer, skup objekata S se može sačuvati kao ključevi u Blumier filtru i svaki objekat $x \in S$ može imati jednu od boja {crvena, zelena, plava, bela}. Slanjem upita o objektu kao odgovor se dobija njegova boja. U ovom slučaju skup vrednosti koje mogu da se dobiju upitom za element x je $R = \{\perp, crvena, zelena, plava, bela\}$, tako da je $f(x) = \perp$ za sve x izvan svih fiksiranih podskupova $S \subseteq D$ veličine n.

Mogućnost lažnog pozitivnog odgovora u regularnom Blumovom filtru ima sledeće posledice na Blumier filtar i kodiranu funkciju f: odgovori na upite za $f(x)$ su uvek tačni kada je $x \in S$ (nema lažnog negativnog odgovora) i skoro uvek tačni kada je $x \in D \backslash S$ (lažni pozitivni odgovor). Blumier filtri se mogu implementirati kao protocni niz (eng. "pipeline") Blumovih filtra.

Neka je $R = \{\perp, 1, 2\}$. Neka je A (odnosno B) podskup od S koji se preslikava u 1 (odnosno 2). Ključ koji je upitan je prošao kroz niz Blumovih filtra i to parova $(F(A_i), F(B_i))$, za $i = 0, 1, \ldots, \alpha$. Prvi par u nizu odgovara $A_0 = A$ i $B_0 = B$. Dakle ključ y koji prolazi test da li je član $F(A_0)$ a ne $F(B_0)$ je verovatno u A i obrnuto. Ključevi koji padaju na oba testa sigurno nisu ni u A ni u B. Problem nastaje kad god prođe test za $F(A_0)$ i $F(B_0)$. U ovom slučaju, u jednom od tih filtra je došlo do lažnog pozitivnog odgovora. Ovaj problem se otklanja tako što se sledeći par u filtru popuni kao lažni pozitivni odgovor prethodne faze. Neka je A_i skup ključeva A_{i-1} koji prolazi test u $F(B_{i-1})$; isto tako, B_i je skup ključeva u B_{i-1}, koji je prošao test u $F(A_{i-1})$. Broj parova α se mora izabrati tako da skupovi A_α i B_α budu prazni.

3.5 Distribuirano heširanje

Blumov filtar je moguće koristiti za deljenje Veb keš memorije. Kao primer može se navesti saradnja proksi servera. Ukoliko se u keš memoriji ne nalazi odgovarajuća stranica (keš promašaj), proksi pokušava da utvrdi da li drugi proksi sadrži željenu Veb stranicu. Ako sadrži, zahtev se pre upućuje tom proksi serveru nego što se bira učitavanje sa Veba.

Da bi ovakvi sistemi bili efikasni, proksi serveri treba da znaju sadržaj drugih proksija. Da bi se smanjila komunikacija i promet poruka, proksi serveri ne prenose *URL-liste* koje odgovaraju tačnom sadržaju svojih keš memorija, već povremeno šalju Blumove filtre koji predstavljaju sadržaj tih keš memorija. Ako proksi želi da utvrdi da li drugi proksi ima stranicu u svom kešu, to proverava u odgovarajućem Blumovom filtru. U slučaju lažnog pozitivnog odgovora, proksi može tražiti stranicu od drugog proksija. Ukoliko taj proksi nema traženu stranicu u svom kešu, dolazi do dodatnog kašnjenja. Lažni pozitivni i lažni negativni odgovor se mogu pojaviti i ako se ne koristi Blumov filtar, jer se sadržaj keš memorije često menja. Ova tehnika se koristi u Veb proksiju otvorenog koda *Squid*[3], gde se Blumovi filtri nazivaju prikaz keša (eng. Cache Digest [10]). Budući da se sadržaj keš memorije često menja, preporučuje se da keš memorije koriste brojački Blumov filtar kako bi bolje pratile sadržaj keš memorije, i slale odgovarajuće standardne {0, 1} odgovore drugim proksi serverima. U suprotnom, bi trebalo formirati novi Blumov filtar kad god dođe do promene na proksi serveru [7].

[3]Squid je proksi server koji kešira Veb sadržaj. On ima široke namene (ubrzanje Veb servera keširanjem zahteva koji se ponavljaju, keširanje Veba i povećanje sigurnosti filtriranjem prometa.)

Poglavlje 4

Primena Blumovog filtra: P2P/Nadmreže

Zbog svoje jednostavnosti i efikasnosti, Blumov filtar je na početku primenjivan za skladištenje skupa reči nekog jezika odnosno, za proveru ispravnosti napisanih reči. Takođe, Blumov filtar je korišćen u mrežama u okviru desktop Veb asistent pregledača *Vistabar* u cilju smanjenja zahteva za opis Veb stranica koje nisu ocenjene i nemaju kritike od strane korisnika. Razvojem p2p mreža Blumovi filtri su postali sve zastupljeniji.

Sa povećanjem popularnosti p2p mreža, sve je zastupljenija i primena Blumovog filtra u p2p mrežama za razmenu digitalnog sadržaja, keširanje sadržaja kao i za povećanje sigurnosti.

4.1 P2P mreže umerene veličine

P2p mreže koriste heš tabele kako bi locirale objekte u mreži. Distribuirane heš tabele (DHT) omogućavaju veliku skalabilnost p2p sistema prilikom kopiranja, odnosno prenosa digitalnog sadržaja sa jednog čvora na drugi. DHT je klasa decentralizovanih distribuiranih sistema koja obezbeđuje pretraživanje slično heš tabelama. Parovi (ključ, vrednost) se čuvaju u DHT, i čvor koji je deo mreže može na efikasan način da preuzme vrednost koja je povezana sa datim ključem.

Za p2p sisteme koji imaju više stotina čvorova, Blumovi filtri mogu osigurati efikasniji način lociranja objekata, u odnosu na distribuirane heš tabele. Čuvanje liste objekata u svakom drugom čvoru u p2p sistemu može biti ograničavajući faktor, ali pogodno je imati Blumov filtar za svaki drugi čvor. Na primer, umesto 64 - bitnog identifikatora za svaki objekat, Blumov filtar može koristiti mnogo manje bitova po objektu. Lažni pozitivni odgovor u ovoj situaciji donosi uzaludne zahteve za objekte na čvorovima koji ih ne sadrže. Još jedan problem koji treba rešiti prilikom implementacije jeste i taj koliko često filtre treba ažurirati.

Postoji sličan pristup koji koristi dodatno grupisanje i hijerarhiju. On se sastoji u tome da se uvede hijerarhija, tako da grupe čvorova vodi čvor-lider. Lideri su stabilniji i dugotrajniji čvorovi koji formiraju p2p mrežu korišćenjem

Blumovog filtra i pokrivaju objekte koji su u grupi. Lider grupe kontroliše rutiranje kao i probleme unutar grupe [7].

4.2 Približni skup za prenos sadržaja

Blumovi filtri mogu rešiti različite probleme u p2p aplikacijama (problem približnih skupova za prenos podataka odnosno, izračunavanje razlike dva skupa). Pretpostavimo da čvor A ima skup elemenata S_A, i čvor B ima skup elemenata S_B. Čvor B želi poslati čvoru A skup podataka koje ima, tako da A može započeti slanje podataka koje B nema, odnosno podatke $S_A \setminus S_B = \{x : x \in S_A, x \notin S_B\}$. Čvor B šalje Blumov filtar koji predstavlja skup njegovih elemenata, A zatim prolazi kroz njegove elemente, proveravajući svaki u svom Blumovom filtru, i šalje elemente koji se ne nalaze u S_B. Zbog lažnog pozitivnog odgovora, neće biti poslati svi elementi $S_A \setminus S_B$, ali većina elemenata $S_A \setminus S_B$ hoće [7].

4.3 Lociranje podataka u p2p mrežama

Blumov filtar se koristi za pronalaženje podataka u *OceanStore*[1] p2p mreži. Proces pronalaženja podatka zasniva se na traženju njegovog jedinstvenog identifikatora. Budući da su dokumenti replicirani na više čvorova, treba pronaći najbližu repliku.

Predloženo je rešenje hibridnog algoritma koji kombinuje prednosti determinističkog (definisano) i probabilističkog (nije definisano već se sastoji od više pokušaja u zavisnosti od prethodnog rezultata) rutiranja (usmeravanja). Ako je neka replika daleko od izvora upita, deterministička lokacija je gotovo optimalna, ali u slučaju replike koja je u blizini izvora upita, bolje je izabrati probabilističku varijantu rutiranja. Hibridni lokacioni mehanizam se pokreće probabilističkom pretragom replika u opsegu manjem od d skokova (eng. hops). Ukoliko probabilistički algoritam ne pronađe replike, stupa deterministički algoritam.

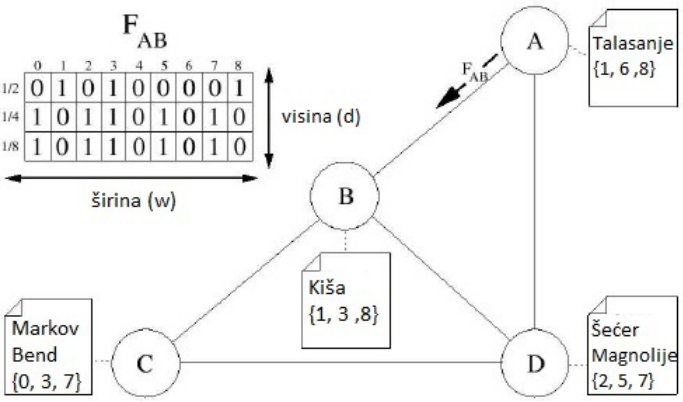
Probabilistički algoritam koji koristi Blumove filtre čine dva algoritma: algoritam za upite, koji rutira upite od čvora do čvora u potrazi za replikom, i algoritam ažuriranja koji prosleđuje informacije o lokaciji ako se skup sadržaja u čvoru promenio. Ove informacije o lokaciji na svakom čvoru se čuvaju u Blumovim filtrima (Attenuated Bloom filters - *ABF*[2]).

ABF predstavlja niz od d standardnih Blumovih filtra $F_1, ..., F_d$. U svakom Blumovom filtru F_i, identifikatori dokumenta se mogu pronaći prolaskom kroz najvise i cvorova mreze. Svi dokumenti koji se mogu pronaći u d skokova (eng. hops) su smešteni u jednom ABF filtru. U nadmreži p2p sistema, svaki čvor ima određen broj svojih suseda. Za svakog suseda čuva se različit ABF filtar i on sadrži dokumente koji se mogu naći pomoću tog čvora. Na primer, neki čvor ima sused n_1 i čuva odgovorajući ABF A_1. U ovom filtru *(A_1)*, F_1 čuva dokumente koji se nalaze na n_1, dokumenti F_2 su svi dokumenti koji se čuvaju

[1] OceanStore je globalna mreža za čuvanje podataka, koja pruža konzistentan, dostupan i izdržljiv alat za čuvanje podataka. Infrastruktura se sastoji od nepouzdanih servera.

[2] Attenuated Bloom filters može se posmatrati kao niz dužine D standardnih Blumovih filtra.

na susedima čvora n_1, itd. Primer jednostavne mreže sa ABF filtrom prikazan je na slici 6.



Slika 4.1: Primer mreže sa *ABF* filtrom za link *AB*. U Blumovom filtru u vrsti i *ABF* filtra dokumenti su smešteni tako da postoji i čvorova do njih. ABF je matrica F_{AB} sa visina d (rastojanj od čvora A) i širinom w (dokumenti) i predstavlja niz od d normalnih Blumovih filtra. Za probabilističku pretragu suseda, svaki sused je povezan sa jednim ABF. Potencijalna vrednost dokumenta "Šećer mangolije" je suma svih nivoa u kojima se pretraga poklapa $\frac{1}{4} + \frac{1}{8} = \frac{3}{8}$.

Ukoliko postoji skup ABF filtra, sa informacijom o lokaciji, može se proslediti upit do suseda koji ima najveću verovatnoću posedovanja traženog dokumenta. Ovo je princip rada algoritma upita. Lažni pozitivni odgovori u bilo kom Blumovom filtru mogu voditi do čvorova koji ne sadrže dokument. Ako se ovo dogodi, pretraga se vrši deterministčkim algoritmom. U svakom upitu, čuva se lista od najviše d čvorova koji su već prosledili upit, kako ne bi došlo do beskonačne petlje.

Algoritam ažuriranja ima funkciju ažuriranja svih ABF filtra kada se čvoru doda novi dokument. Ovo se postiže ukoliko svaki čvora čuva ne samo svaki ABF filtar za svaki odlazeći link, već i za kopiju putanje njegovih suseda u suprotnom smeru. Kada se novi dokument doda u čvor, on računa promenjene bitove u svom filtru kao i u svakom od filtra svojih suseda koji predstavlja putanju do njega. Ovi bitovi se šalju svim susedima kako bi putanje koje upućuju do njega ostale ažurirane. Na ovaj način dolazi do promena u celoj mreži. Veličina ABF filtra zavisi od broja suseda svakog čvora, kao i prosečnog broja dokumenata koje čvorovi imaju. Ovaj deo teksta kao i poglavlje 4.4 može se haći u referenci [8].

4.4 Efikasna p2p pretraga ključeva

Blumov filtar može da se koristi u distribuiranoj pretrazi ključeva p2p sistema. Osnova ovakve pretrage je distribuirana heš tabela (DHT), koja predstavlja rešenje pretraživanja ključeva u distribuiranom prostoru.

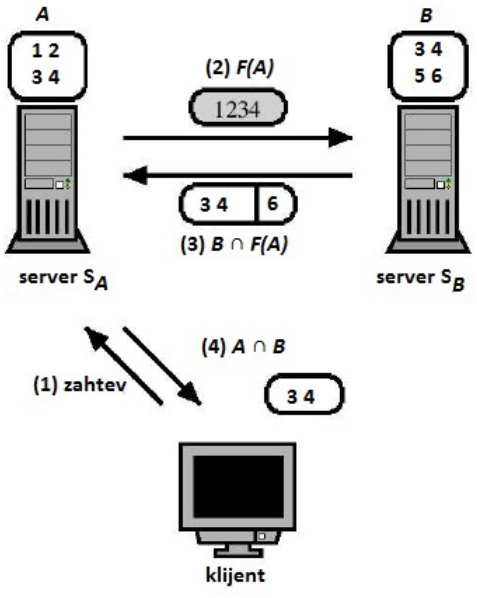
Ovaj proces pretrage se obavlja u tri koraka. U prvom, string pretrage se deli na različite ključne reči $k_1, \ldots, k_n$, koje se proveravaju u DHT. Pretraga

pomoću ključne reči k_i kao rezultat daje adresu čvora p_i, koji čuva identifikator tog dokumenta koji se odnosi na ključnu reč k_i. Ukoliko se dobije skup adresa čvorova koji čuvaju identifikator tog dokumenta tada se uzima adresa najbližeg čvora.

U drugom, čvorovi p_i za $i = 1, \ldots, n$ bivaju upitani za njihov skup identifikatora D_i.

Oni se kombinuju uzimanjem preseka $D = D_1 \cap \ldots \cap D_n$, kako bi rezultat pretrage sadržao sve ključne reči. Na kraju, dokumenti iz D koji zadovoljavaju kriterijume pretrage se učitavaju korišćenjem druge DHT tabele pomoću koje se pronalaze čvorovi koji sadrže željene dokumente.

Blumovi filtri se koriste u drugom koraku kako bi se efikasno izračunao presek skupa na različitim čvorovima bez slanja cele liste identifikatora. Svaki skup D_i ima mnogo više elemenata nego krajnji skup D. Zato, slanjem Blumovih filtra koji predstavljaju skup D_i može da se smanji protok informacija. Primer pronalaženja preseka skupova dokumenata je predstavljen na slici 4.2.



Slika 4.2: Blumov filtar pomaže smanjenju protoka informacija prilikom pronalaženja preseka. Sive kutije predstavljaju Blumov filtar *F(A)* skupa *A*. Klijent šalje serveru S_A zahtev za pretragu dokumenata. Server S_A prenosi serveru S_B pomoću Blumovog filtra $F(A)$ reprezentaciju svojuh dokumenata. Server S_B računa presek $B \cap F(A)$ i vraća rezultat serveru S_A a server S_A taj presek prosledjuje do klijenta, što predstavlja rezultat (3, 4).

Što se više ključnih reči koristi, veća je verovatnoća da skup D sadrži neke dokumente koji nisu traženi zbog lažnog pozitivnog odgovora. Ovaj problem se može rešiti tako što će čvor p_n poslati rezultat preseka D nazad do p_1, kako bi se izračunao $D \cap D_1$ i uklonio lažni pozitivni odgovor, a rezultat prosledio do svih ostalih čvorova još jednom. Pošto je skup D relativno mali, ova komunikacija se može obaviti i bez korišćenja Blumovog filtra.

Optimalna veličina Blumovog filtra može da se izračuna za slučaj dve ključne reči. Neka je skup identifikatora dokumenata povezan sa ključnim rečima i neka one budu A i B. Ako je sav protok korišćen za presek minimalizovan, optimalna veličina Blumovog filtra može da se izračuna na osnovu sledeće jednačine:

$$m = |A| \log_{0.6185}(2.08 \frac{|A|}{|B|j}),$$

gde je j veličina identifikatora.

Poglavlje 5

Pronalaženje sličnosti u p2p mrežama

U p2p mrežama mogu se pojaviti klasteri - čvorovi sa sličnim sadržajem, što podrazumeva da čvorovi imaju istu grupu prijatelja ili iste interese. Klasterovanje je efikasno, zato što čvorovi sa istim interesima često pretražuju čvorove sa istim tipovima podataka. Ove pretrage mogu biti ograničene u okviru samog klastera, pa je zbog toga za čvorove u interesu da pronađu druge čvorove sa sličnim karakteristikama.

Komunikacija između dva čvora može se veoma efikasno ostvariti ako se koriste Blumovi filtri u oba smera kako bi predstavili skupove podataka i ako je skup podataka koji učestvuje u komunikaciji podskup skupa koje čvorovi već imaju.

5.1 Sličnost sadržaja

Ovaj pristup organizuje čvorove bazirajući se na sličnosti njihovog sadržaja odnosno, na sličnosti njihovih lokalnih dokumenata. Ovaj pristup pokušava da grupiše relevantne čvorove u cilju smanjenja broja irelevantnih čvorova prilikom upita.

Umesto proveravanja sličnosti dokumenata, proverava se sličnost njihovih filtra, što je mnogo povoljnije pošto je filtar za skup dokumenata mnogo manji od njih samih. Takođe upoređivanje filtra je mnogo efikasnije od upoređivanja dva skupa dokumenata.

Sledeći primer objašnjava ovaj princip rada. Čvor n_i kada se priključi sistemu šalje svim učesnicima mreže svoj lokalni filtar $F(D_i)$. Kada čvor n_j primi ovu poruku šalje odgovor na kojoj je udaljenosti njegov lokalni filtar od n_i-tog filtra *rastojanje(F(D$_i$), F(D$_j$))*. Čvor n_i odlučuje da će se priključiti čvoru koji je na najmanjem rastojanju (na primer najsličnijem sebi).

Na ovaj način čvorovi sa relevantnim sadržajem bi trebalo da se grupišu i tako da formiraju klastere koji se baziraju na sličnom sadržaju. Cilj ove organizacije je taj da kada se jednom pošalje upit relevantnom klasteru, kao odgovor dobiju čvorovi sa sličnim dokumentima.

5.2 Otkrivanje zajedničkih prijatelja

Umesto informacija o sadržaju, čvorovi mogu takođe da razmenjuju i Blumove filtre koji čuvaju informacije o identifikatorima čvorova koji su njihovi prijatelji. Na ovaj način, čvorovi mogu da odrede u kojoj meri imaju slične društvene susede. Ove mere mogu se uzeti kao deo sigurnosne funkcije: ako čvor deli mnogo prijatelja sa mnom, onda on mora da bude ugledan član moje zajednice.

Informacija o zajedničkim prijateljima može se koristiti za nalaženje prethodnih *IP* adresa čvorova, pošto mnogi čvorovi imaju nove *IP* adrese ili novi broj porta svaki put kada postanu dostupni (eng. online). Da bi se ovo rešilo svaki čvor šalje *pozdravnu poruku* svim čvorovima koji su mu prijatelji svaki put kada postane dostupan. Pretpostavimo da čvor p_1 postaje dostupan i pošalje svojim prijateljima pozdravnu poruku. Do nekih njegovih prijatelja će poruka stići, međutim većina tih čvorova nije dostupna ili je dostupna sa novom adresom. Kada čvor p_1 zna sve informacije o zajedničkim prijateljima, on može da koristi te informacije kako bi pronašao prijatelje sa kojima može da se poveže. Naravno, potrebno je uvek imati mali broj prijatelja sa kojima može da se poveže, jer u suprotnom čvor se potpuno isključuje iz mreže, i tada kao jedina mogućnost preostaje da se upita superčvor za informacije o adresi prijatelja. Superčvorovi su uvek dostupni i imaju statičku adresu.

Proces pronalaženja informacija o zajedničkim prijateljima je sličan procesu pronalaženja sličnosti sadržaja (poglavlje 5.1). Čvor šalje listu svojih prijatelja sačuvanu u Blumovom filtru svim svojim prijateljima, a oni odgovaraju podskupom liste sa zajedničkim prijateljima jednostavnim presekom dva skupa podataka. Kada dobiju informaciju o zajedničkim prijateljima, čvorovi mogu da se povežu i da traže adresu određenog prijateljskog čvora. Čvor može da pošalje identfikatore sačuvane u Blumovom filtru svih čvorova za koje ne zna trenutne statuse. Čvor koji prima informacije može da odgovori listom adresa čvorova koje on zna. Ovaj princip pronalaženja čvorova se naziva pronalaženje dostupnih čvorova (eng. online peer discovery).

Poglavlje 6

Sigurnost u p2p mrežama

Blumove filtre karakteriše velika efikasnost u praktučnoj primeni, budući da je moguće razmeniti ove filtre korišćenjem malog protoka.

U ovom delu prikazana su rešenja kako da se dobje kompaktan Blumov filtar, a da se osigura njegova verodostojnost, odnosno da filtar stvarno odgovara nekom skupu.

6.1 Zloupotreba Blumovog filtra

U p2p aplikacijama čvor šalje Blumov filtar kako bi dokazao da ima odgovarajući skup informacija, odnosno njegov sadržaj ili informacije o drugim čvorovima. Čvor-primalac ima poverenja u ove informacije i koristi ih kako bi odlučio da li je interesantno razmenjivanje informacija sa pošiljaocem. Nedostatak ovog sistema je to što je moguće da pošiljalac kreira Blumov filtar i promeni ga tako da primalac misli da je skup podataka mnogo veći nego što zapravo jeste. U ovakvoj situaciji, pošiljalac može da stekne neku prednost u odnosu na primaoca.

U aplikacijama u kojima se izračunavaju sličnosti sadržaja, Blumov filtar je popunjen skupom podataka (fajlovima) pošiljaoca. Ako primalac zaključi da pošiljalac sadrži fajlove slične primaocu on će razmeniti informacije o sadržaju primaoca. Pošiljalac je u prednosti ako primalac misli da on ima mnogo fajlova sličnih njegovim. Ova prednost se može lako postići postavljanjem svih bitova u filtru na 1, posle čega primalac može da zaključi da on ima sve fajlove. Mnogi čvorovi će zaključiti da je čvor interesantan, pa će dalje rutiranje obavljati preko ovog čvora, dajući mu veliki značaj. Samo onda kada čvorovi zatraže stvarni fajl od malicioznog čvora, saznaće da ti fajlovi ne postoje ili nisu fajlovi koje očekuju. Bez primene nekih mera bezbednosti veoma je lako da maliciozan čvor zloupotrebi Blumov filtar za komunikaciju i dobije veliki uticaj u celoj p2p mreži.

Slična alternativa Blumovom filtru se koristi za komunikaciju u drugim aplikacijama koje daju čvoru mogućnost da se pretvara da zna identifikatore svih čvorova ili da dele sve prijatelje sa nekim. Ova zloupotreba omogućava čvorovima da uvećaju prestiž u mreži i tako smanje kvalitet same mreže. Ovakve jednostavne zloupotrebe treba onemogućiti. Zato čvor koji prima informacije treba da bude u mogućnosti da proveri da li čvor stvarno poseduje taj skup podataka koji je smešten u njegovom Blumovom filtru. Primaocu su posebno interesantni

elementi za koje testira pripadnost u filtru i na osnovu kojih donosi odluke. Najčešće je to podskup skupa podataka smeštenih u Blumovom filtru. Naredni deo teksta može se naći u referenci [8].

6.2 Blumovi filtri sa statičkom gustinom bitova

U cilju sprečavanja Blumovog filtra da vrši neke neželjene promene, treba koristiti filtar sa nekim standardnim svojstvima na svim čvorovima. Na primer, može se fiksirati statička veličina m svakog filtra. Ovo rešenje daje verovatnoću pojavljivanja lažnog pozitivnog odgovora zavisnu od elemenata koji su u filtru. Na ovaj način, što se više elemenata dodaje u filtar, veća je verovatnoća pojave grešaka i teža je provera da li te elemente stvarno u poseduje taj pošiljalac. Maliciozan čvor može poslati niz bitova popunjen jedinicama i garantovati da sadrži sve elemente koji postoje.

Bolje rešenje je fiksirati statičku verovatnoću pojave lažnog pozitivnog odgovora i to se postiže fiksiranjem broja bitova po elementu. Blumov filtar tada linearno raste sa brojem elemenata u njemu. Prednost ovog rešenja je u tome što će filtar imati optimalnu veličinu. Kod ovakvih filtra verovatnoća da je jedan od bitova 1 je približno 0.5 (jednačina 1.2). Maksimalni broj bitova postavljenih na 1 posle dodavanja n elemenata, korišćenjem k heš funkcija je $n \cdot k$. Korišćenjem optimalnog broja k dobija se najviše $\ln 2 \cdot m$ jedinica u nizu bitova. Ovakav način korišćenja Blumovog filtra onemogućava maliciozne čvorove da postavljaju nasumične bitove niza na 1. Čvor koji prima podatke može jednostavno da proveri da li broj jedinica u nizu prelazi granicu od $m \cdot \ln 2$ i ako je tako može da ignoriše taj filtar. Jedini način da se maliciozni čvor pretvara da ima više elemenata je da pošalje duži filtar postavljanjem nasumičnih vrednosti na 1. Međutim, dolazi do povećanog protoka u malicioznom čvoru i on se pretvara da poseduje ograničen broj nasumičnih elemenata.

6.3 Komunikacija na bazi dva različita Blumova filtra

Ukoliko je potrebna velika sigurnost, može se zahtevati da svi čvorovi komuniciraju slanjem dva različita Blumova filtra. Oba filtra će imati jednake i statičke verovatnoće lažnih pozitivnih odgovora, ali će koristiti različite heš finkcije. Kada čvor koji prima informacije zaključi da je element u prvom filtru, on onda proverava da li je i u drugom. Ukoliko utvrdi da jeste, onda on može sa velikom verovatnoćom da tvrdi da pošiljalac poseduje taj element. U protivnom, dešava se lažni pozitivni odgovor ili pošiljalac laže. S obzirom da je verovatnoća lažnog pozitivnog odgovora oba filtra poznata primaocu, on može da odredi da li je to stvarno lažni pozitivni odgovor ili treba da označi pošiljaoca kao malicioznog.

Pošiljalac ne može da kreira Blumov filtar na takav način da čuva elemente slične onima koje sadrže primalac kada su oni kreirani od nasumično postavljenih bitova. Pretpostavimo da pošiljalac ima 100 elemenata, ali želi da primalac veruje da ih ima više. On može da pokuša da kreira dva Blumova filtra 1000 puta veća od statičke gustine bitova po elementu. Posle dodavanja realnih elemenata,

on postavlja nasumično bitove na 1 tako da su na kraju u oba filtra otprilike polovina bitova jedinice. Ono što je on uradio je kreiranje Blumovog filtra sa 100 zajedničkih elemenata i u svakom filtru po 900 nasumičnih elemenata. Verovatnoća da je neki od elemenata koji pripadaju primaocu jednak jednom od nasumičnih elemenata u primarnom Blumovom filtru jednaka je verovatnoći lažnog pozitivnog odgovora f_1. Verovatnoća da je taj element u drugom filtru je jednaka verovatnoći drugog lažnog pozitivnog odgovora f_2. Ovo vodi do ukupne verovatnoće da pošiljalac može da se pretvara da ima element primaoca $f_1 \cdot f_2$. Ovo je ujedno i verovatnoća lažnog pozitivnog odgovora dva Blumova filtra.

Umesto slanja jednog Blumovog filtra sa m/n bitova po elementu, bolje je poslati dva Blumova filtra veličine $m/2n$ bita po elementu. Ukupna količina podataka na osnovu ovog rešenja jednaka je veličini slanja jednog Blumovog filtra. Veoma je teško kreirati dva Blumova filtra koji predstavljaju iste elemente postavljanjem nasumičnih bitova na 1.

6.4 Izazov-odgovor

Primenom gore navedenih rešenja, čvor se teško može pretvarati da ima određene elemente. U praksi, ovi elementi mogu biti identifikatori datoteka ili oznake korisnika. Ako primalac, čvor B ne želi samo da bude siguran da pošiljalac, čvor A poseduje identifikator datoteke, već i samu datoteku, on mora da zahteva od pošiljaoca da dokaže da stvarno ima taj dokument. Ova komunikacija, gde primalac, čvor B, zahteva od pošiljaoca, čvora A, neke privatne informacije, se naziva *izazov-odgovor*.

Izazov-odgovor u kome pošiljalac mora da dokaže da poseduje neke fajlove, može da se izvrši na sledeći način:

- Izazivač čvor B pita za određene nasumične izabrane delove fajla, a čvor A mu ih šalje kao odgovor.
- Izazivač šalje nasumični, slučajni broj, tzv. *salt* [1] čvoru B, a on pretražuje fajlove sa tim salt-om i šalje nazad odgovor.
- Izazivač šalje listu salt-ova čvoru B, a on koristi ove podatke da kreira novi Blumov filtar svih fajlova i šalje nazad odgovor cvoru A.

Salt je inicijalizacioni vektor koji se koristi kao ulaz heš funkcije, koji utiče na heš vrednosti svih objekata tako da je moguć pronalazak određene heš vrednosti ako se zna i salt i sam objekat. Pošto onaj koji odgovara ne zna unapred sa kojim salt-om je potrebno heširati fajl, on to ne može da uradi unapred.

Ovaj metod je veoma zahtevan zbog heširanje velikih fajlova. Problem je takođe taj što izazivač mora da testira odgovor koji dobija. Izazivač mora da ima kopiju tog fajla ili da zna nekog kome veruje i ima taj fajl. U suprotnom ovaj metod nema efekta.

[1] Salt je dodatni podatak koji se koristi prilikom heširanja kako bi se povećala sigurnost.

Poglavlje 7

Programska realizacija

U ovom poglavlju dat je opis programske realizacije Blumovog filtra u programskom jeziku *Java* i simulacije rada filtra kada se u njega ubacuju nasumične niske karaktera. Drugi deo ovog poglavlja sadrži rezultate dobijene testiranjem Blumovog filtra da se proceni verovatnoća pojave lažnog pozitivnog odgovora.

7.1 Programska realizacija Blumovog filtra

Programska realizacija Blumovog filtra i merenje lažnog pozitivnog odgovora u odnosu na različit broj heš funkcija i broj bitova po elementu dodatih u filtar je implementirana u Java/Spring MVC (eng. Model-View-Controller) okruženju.

Odlučeno je da ova aplikacija bude implementirana kao veb aplikacija zbog veoma intuitivnog načina unosa podataka i prikaza istih.

Aplikacija pruža grafički prikaz pojave lažnog pozitivnog odgovora u odnosu na ulazne parametre (broj bitova po elementu i broj heš funkcija) prilikom dodavanja nasumičnih niski karaktera u filtar. Za unos ulaznih podataka koristi se *jQuery UI* [11] javaskript biblioteka koja pruža mogućnost prikaza opsega i broja bitova i broja heš funkcija u vidu klizača. Za odabir različitih heš funkcija koristi se *Select2* [12] javaskript biblioteka koja pruža mogućnost odabira opcije u vidu padajućeg menija. Za prikaz grafika funkcije pojave lažnog pozitivnog odgovora koristi se *Highcharts*[13] javaskript biblioteka koja pruža bogat skup opcija prikaza grafika funkcije.

HomeController java klasa se koristi za prosleđivanje zahteva od strane korisnika servisu na dalju obradu podataka. Metode klase *BloomFilterService* se koriste za sva izračunavanja u aplikaciji (pravljenje Blumovog filtra i računanje verovatnoće lažnog pozitivnog odgovora, generisanje nasimičnih stringova). *RezultatModel* klasa se koristi za čuvanje podataka dobijenih testiranjem Blumovog filtra kako bi se prikazali korisniku na stranici.

Realizacija Blumovog filtra iskorišćena je za eksperimentalnu procenu zavisnosti verovatnoće lažnog pozitivnog odgovora od veličine filtra. Osnovni element Blumovog filtra je generisanje proizvoljnog broja kvalitetnih heš kodova za dati string.

Pored kvalitetnih heš funkcija u programskoj realizaciji koristi se i nekvalitetna heš funkcija kako bi se pokazalo ponašanje Blumovog filtra i u ovom slučaju.

Heš funkcije koje su korišćene [14] :

- Aditivna heš funkcija - jednostavna heš funkcija čija se vrednost izračunava tako što se svi karakteri stringa saberu i broj koji se dobije predstavlja heš vrednost datog stringa. Ovaj algoritam se smatra veoma lošim i treba ga izbegavati za heširanje stringova. Na primer, niske karaktera "asd" i "dsa" imaju iste heš funkcije, odnostno, permutacijom karaktera dobija se uvek ista heš vrednost, zbog čega se ova heš funkcija smatra veoma lošom.

```
long hash = 0L;
for (int i = 0; i < characters.length; i++) {
     hash += characters [i];
}
```

- Bernstajnova (eng. Bernstein) heš funkcija - dobra heš funkcija i koristi se u mnogim praktičnim implementacijama. Iako se ne ponaša najbolje prilikom permutacije bitova, dobro se pokazala kao heš funkcija kratkih niski karaktera. Treba je koristiti sa rezervom jer nije u potpunosti objašnjeno zbog čega se najbolje ponaša sa konstantom 33. Ova heš funkcija predstavlja vrednost polinoma čiji su koeficijenti karakteri stringa u tacki 33.

```
long hash = 0L;
for (int i = 0; i < characters.length; i++) {
     hash = 33 * hash + characters[i];
}
```

- FNV(Fowler/Noll/Vo) heš funkcija - dobra heš funkcija koja nije namenjena heširanju stringova, ali daje veoma dobre rezultate. Jedna od prednosti heš funkcije FNV je da se veoma jednostavno implementira. Kao i kod Bernstajnova heš funkcije ova heš funkcija predstavlja vredsnot polinoma u tački 16777619 pri čemu je sabiranje zamenjeno XOR funkcijom u toku računanja polinoma Hornerovom šemom.

```
long hash = 2166136261L;
for (int i = 0; i < characters.length; i++) {
     hash = (hash * 16777619) ^ characters[i];
}
```

- SAX (eng. Shift Add Xor) heš funkcija - namenjena je heširanju stringova, ali je moguće koristiti je i u druge svrhe jer daje odlične rezultate.

```
long hash = 0L;
for (int i = 0; i < characters.length; i++) {
     hash ^= (hash << 5) + (hash >>> 2) + characters[i];
}
```

Prilikom heširanja stringa *characters* heš funkcija obradjuje u petlji jedan po jedan karakter stringa (*characters[i]*). Za implementaciju h_i - te heš funkcije, za $i > 1$, koriste se prethodno inicijalizovane različite početne vrednosti heš funkcije. Početne vrednosti koje se koriste su: 33, 37, 1549, 3767, 7687, 9337, 9739.

Za generisanje heš kodova koristi se metod sa prototipom:

```
public long hashCode(String s, int hkBr, HashFunction hFunc)
```

Ulazni parametri funkcije su:

- s - niska karaktera za koju se računa heš funkcija
- hkBr - redni broj heš funkcije odnosno, indeks u nizu početnih vrednosti heš funkcija
- hFunc - tip heš funkcije koju treba primeniti prilikom heširanja

Za generisanje nasumičnih stringova koristi se metod iz paketa *org.apache.commons.lang (RandomStringUtils.random(int length, string chars))*. Dužina nasumičnih stringova je između 1 i 10 tačnije, dužine su 1, 3, 5, 7 i 9 i ti stringovi predstavljaju nasumične reči nekog jezika dok dužine koje ne zadovoljavaju uslov odbacamo i ponavljamo preces sve dok se ne dobije dužina koja zadovoljava uslov. Metod *nextGaussian()* instance klase *Random* daje promenljive sa Gausovom (0,1) raspodelom.

Verovatnoća dobijanja reči dužine 5 jednaka je verovatnoći da *r.nextGaussian() + 0.5d* bude 0 odnosno, da promenljiva *r.nextGaussian()* bude u intervalu od -1.5 do 0.5 što daje verovatnoću od 0.6247. Verovatnoća dobijanja reči dužine 7 jednaka je verovatnoći da slučajan broj *r.nextGaussian()* bude u intervalu od 0.5 do 1.5 sto daje verovatnoću 0.2417.

```
int length = 0;
while(length < 1 || length > 10) {
      length = 5 + 2 * (int)(r.nextGaussian() + 0.5d);
}
String s = RandomStringUtils.random(length, LETTERS);
```

Blumov filtar se sastoji od BitSet-a. BitSet je *Java* klasa koja implementira niz bitova proizvoljne dužine. Promenljiva *hashMask* se koristi prilikom računanja indeksa bita koji je potrebno postaviti na 1 ili prilikom provere vrednost na traženom indeksu.

Instanca filtra se inicijalizuje brojem bitova i brojem heš funkcija. Radi bolje efikasnosti zahteva se da broj bitova bude stepen broja 2. Na primer, konstruktor Blumovog filtra *BloomFilter(int log2noBits, int noHash)* za prosleđen *log2noBits* 16 pravi Blumov filtar veličine 2^{16}.

```
private final int noHash;
```

```
private final BitSet data;
private final int hashMask;

public BloomFilter(int log2noBits, int noHash) {
    this.data = new BitSet(1 << log2noBits);
    this.noHash = noHash;
    this.hashMask = (1 << log2noBits) - 1;
}
```

Nakon izračunavanja heš koda, korišćenjem jedne od gore pomenutih heš funkcija, indeks bita se izračunava pomoću dobijenog heš koda i maske (promenljiva *hashMask*) korišćenjem funkcije bitsko I (eng. AND). Za upis i traženje u filtru koriste se funkcije *add(String s, HashFunction hFunc)* i *boolean contains(String s, HashFunction hFunc)*;

Ulazni parametri funkcija su:

- s - niska karaktera za koju se računa heš funkcija
- hFunc - tip heš funkcije koju treba primeniti prilikom heširanja

Funkcija *add* se koristi za dodavanje stringova u Blumov filtar tako što se svaki put prilikom dodavanja elementa na osnovu prosleđenog tipa heš funkcije izračuna heš kod datog stringa i dobijena vrednost upiše u filtar (onoliko puta koliko ima heš funkcija u filtru).

```
public void add(String s, HashFunction hFunc) {
      for (int n = 1; n <= noHash; n++) {
           long hc = hashCode(s, n, hFunc);
           int bits = (int) (hc) & this.hashMask;
           data.set(bits);
      }
}
```

Za proveru da li je string sadržan u filtru koristi se funkcija *contains*:

```
public boolean contains(String s, HashFunction hFunc) {
       for (int n = 1; n <= noHash; n++) {
            long hc = hashCode(s, n, hFunc);
            int bits = (int) (hc) & this.hashMask;
            if (!data.get(bits))
                return false;
        }
        return true;
```

Za generisanje grafika zavisnosti verovatnoće pojave lažnog pozitivnog odgovora od broja heš funkcija koristi se metod sa prototipom:

```
RezultModel genStatFnHashNo
(
    int hashStart, int hashEnd,
```

```
    int bitsStart, int bitsEnd,
    HashFunction hFuncType
)
```

Ulazni parametri funkcije su:

- hashStart - početni broj heš funkcija
- hashEnd - krajnji broj heš funkcija
- bitsStart - početni broj bitova po elementu
- bitsEnd - krajnji broj bitova po elementu
- hFuncType - tip heš funkcije koja se koristi prilikom heširanja stringova

Metod *genStatFnHashNo* vraća objekat klase *ResultModel* koji sadrži verovatnoće pojave lažnog pozitivnog odgovora u zavisnosti od broja heš funkcija.

Objekat klase *RezultModel* ima sledeće promenljive:

- List<BitModel> list - promenljiva predstavlja listu objekata tipa BitModel
- List<String> xAxis - promenljiva predstavlja listu vrednosti x-ose grafika

Objekat klase *BitModel* ima sledeće promenljive:

- String name - promenljiva predstavlja ime grafika (u ovom sličaju broj heš funkcija korišćene u simulaciji)
- List<Double> data - promenljiva predstavlja listu vrednosti na y-koordinati

Za generisanje grafika zavisnosti verovatnoće pojave lažnog pozitivnog odgovora od broja bitova po elementu koristi se metod sa prototipom:

```
RezultModel genStatFnBitsPerItems
(
    int hashStart, int hashEnd,
    int bitsStart, int bitsEnd,
    HashFunction hFunc
)
```

Metod prihvataju pet argumenata tipa i to:

- hashStart - početni broj heš funkcija
- hashEnd - krajnji broj heš funkcija
- bitsStart - početni broj bitova po elementu
- bitsEnd - krajnji broj bitova po elementu
- hFuncType - tip heš funkcije koja se koristi prilikom heširanja stringova

Metod *genStatFnBitsPerItems* vraća objekat klase *ResultModel* koji sadrži verovatnoće pojave lažnog pozitivnog odgovora u funkciji od broja bitova po elementu.

U ovom slučaju promenljive objekta klase *RezultModel* čuvaju sledeće informacije:

- List<BitModel> list - promenljiva predstavlja listu objekata tipa BitModel
- List<String> xAxis - promenljiva predstavlja listu vrednosti x-ose grafika

Objekat klase *BitModel* ima sledeće promenljive:

- String name - promenljiva predstavlja ime grafika (u ovom sličaju broj broj bitova po elementu u filtru korišćen u simulaciji)
- List<Double> data - promenljiva predstavlja listu vrednosti na y-koordinati

Obe metode, *genStatFnBitsPerItems, genStatFnHashNo* dodaju elemente u prethodno napravljen Blumov filtar na osnovu ulaznih parametara, a zatim testiraju pripadnost elemenata i beleže pojave lažnog pozitivnog odgovora.

7.2 Simulacije

U nastavku su navedeni rezultati simulacije Blumovog filtra sa različitim brojem heš funkcija i brojem bitova po elementu.

Dva ulazna podatka koja se koriste za simulaciju rada Blumovog filtra su:

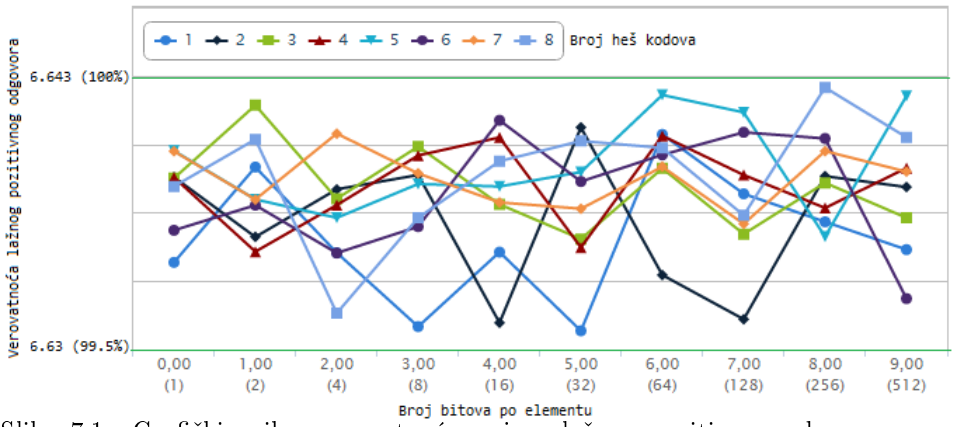
- Broj bitova po elementu, koji pokazuje koliko puta je više bitova u filtru od maksimalnog broja elemenata u skupu.
- Broj heš funkcija, a samim tim i broj bitova koji se postavljaju na 1 za svaki dodati element.

Prethodnom analizom Blumovog filtra možemo predvideti da važi:

- Što je puniji skup bitova (što je manje bitova po elementu), to je veća šansa da se dobije lažni pozitivni odgovor.
- Kada je skup relativno prazan (što je više bitova po elementu), odnosno ako se koristi više heš funkcija po elementu, manja je verovatnoća lažnog pozitivnog odgovora. Za dati element koji nije u skupu, manja je verovatnoća da nasumična kombinacija bitova drugih elemenata lažno označi taj element kao prisutan.
- Za datu veličinu filtra postoji tačka maksimuma za koju dalje povećanje broja heš funkcija prosto znači da se skup bitova popuni veoma brzo, tako da se dobija više lažnih pozitivnih odgovora nego sa manje heš funkcija.

- Korišćenjem dobrih heš funkcija za heširanje stringova verovatnoće pojave lažnih pozitivnih odgovora u filtrima ne bi trebalo da se drastično razlikuju.

Prethodni zaključci proverni su serijom eksperimenata. Prilikom pokretanja simulacije dodaje se 2^{14} nasumičnih stringova u Blumove filtre čije veličine variraju od 2^{14} do 2^{23} bitova, gde je broj heš funkcija menjan u intervalu od 1 do 8 (tačan broj heš funkcija je prikazan na slikama). Za svaku od ovih kombinacija heš funkcija i veličina filtra nastaje odgovarajući filtar i dodaje 2^{14} stringova. Pored toga formira se i jedan običan skup elemenata (java HashSet) kao kontrolni skup u koji se uporedo dodaju isti elementi kako bi se kasnije odredilo da li proizvoljno izabran element stvarno pripada skupu ili je došlo do pojave lažnog pozitivnog odgovora. U poslednjem koraku se generiše određen broj nasumičnih stringova (u ovom slučaju milion kako bi se povećala tačnost eksperimenta) za koje se vrše testovi pripadnosti elemenata skupu. Za svaki od ovih stringova koji nisu u skupu, a za koje filtar prijavi da jesu, beleži se pojava lažnog pozitivnog odgovora; za proveru da li je element zaista u skupu koristi se kontrolni skup elemenata.

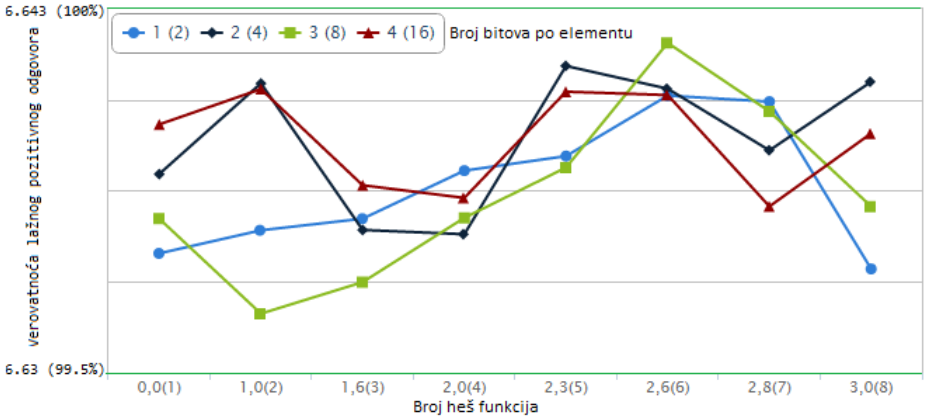


Slika 7.1: Grafički prikaz verovatnoće pojave lažnog pozitivnog odgovora u odnosu na broja bitova po elementu u $\log 2$ razmeri korišćenjem aditivne heš funkcije.

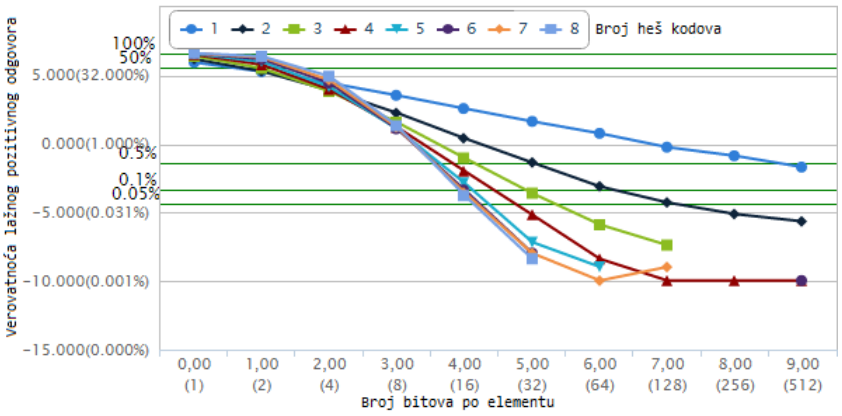
Na slikama 7.1 i 7.2 prikazano je kako izbor loše heš funkcije može da utiče na veoma veliku verovatnoću pojave lažnog pozitivnog odgovora. Na svim graficima verovatnoća je velika bez obzira na broj bitova po elementu i broj heš funkcija. Ovakve heš funkcije, kao što je aditivna heš funkcija, treba izbegavati za heširanje stringova jer čine filtar neupotrebljivim.

Na slikama 7.3, 7.4, 7.5 je uočena slična zavisnost funkcije od broja bitova po elementu. Odnosno, ako su heš funkcije dobre, onda se slično ponašaju kada se menja broj bitova po elemntu. Prva vrednost na x-osi predstavlja Blumov filtar sa jednim bitom po elementu, druga vrednost sa dva bita, treća sa četiri bita po elementu itd.

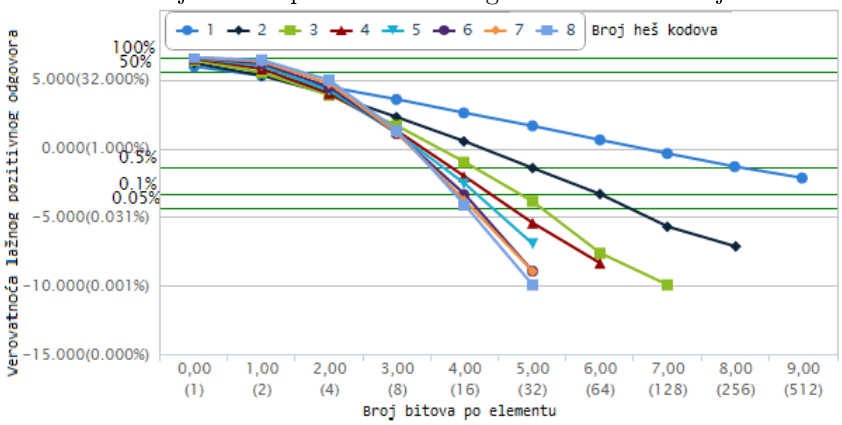
Dodeljivanjem malog broja bitova po elementu kao rezultat se dobija velika verovatnoća lažnog pozitivnog odgovora. Sa jednim ili dva bita po elementu



Slika 7.2: Grafički prikaz verovatnoće pojave lažnog pozitivnog odgovora u zavisnosti od broja heš funkcija u log 2 razmeri korišćenjem aditivne heš funkcije.



Slika 7.3: Grafički prikaz verovatnoće pojave lažnog pozitivnog odgovora u zavisnosti od broja bitova po elementu u log 2 razmeri korišćenjem Bernstajnove



Slika 7.4: Grafički prikaz verovatnoće pojave lažnog pozitivnog odgovora u zavisnosti od broja bitova po elementu u log 2 razmeri korišćenjem FNV heš funkcije.

zapaža se da je bolje imati manje heš funkcija, jer veliki broj heš funkcija popuni filtar suviše brzo. Kako se broj bitova po elementu povećava, tako se i verovatnoća lažnog pozitivnog odgovora smanjuje.

Sa četiri bita po elementu, dodavanjem više heš funkcija poboljšava se verovatnoća lažnog pozitivnog odgovora do optimalne vrednosti od tri heš funkcije i verovatnoća lažnog pozitivnog odgovora je manja od 14.6%.

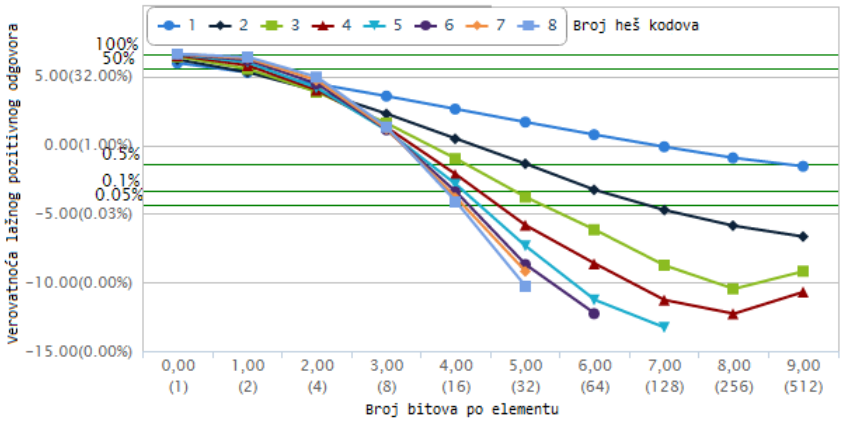
Sa 8 bitova po elementu optimalna verovatnoća lažnog pozitivnog odgovora je između 2.44 i 2.5% i to sa četiri heš funkcije, u zavisnosti od načina heširanja slike 7.3, 7.4, 7.5.

Na osnovu dobijenih rezultata može se zaključiti da je 8 alociranih bitova po elementu dovoljno za malu verovatnoću pojave lažnog pozitivnog odgovora bez obzira na korišćeni broj heš funkcija. Takođe, izbor dobre heš funkcije ne utiče drastično na pojavu lažnog pozitivnog odgovora, dok izbor loše heš funkcije čini filtar neupotrebljivim.

Na slikama 7.6, 7.7 i 7.8 prikazano je kako se verovatnoća lažnog pozitivnog odgovora menja u funkciji od broja heš funkcija za Blumov filtar sa $2, 4, 8$ i 16 bitova po elementu. Na ovim graficima uočava se slično ponašanje verovatnoće lažnog pozitivnog odgovora. Postoji granični broj heš funkcija u odnosu na veličinu filtra, iznad kojeg se prostor u filtru popunjava suviše brzo, pa se verovatnoća pojave lažnog pozitivnog odgovora naglo povećava. Nakon dve, odnosno tri heš funkcije (slike 7.6, 7.7, 7.8) kriva sporo opada. Za 1 i 2 bita po elementu povećanje broja heš kodova povećava verovatnoću jer se filtar popuni suviše brzo (slike 7.6, 7.7, 7.8).

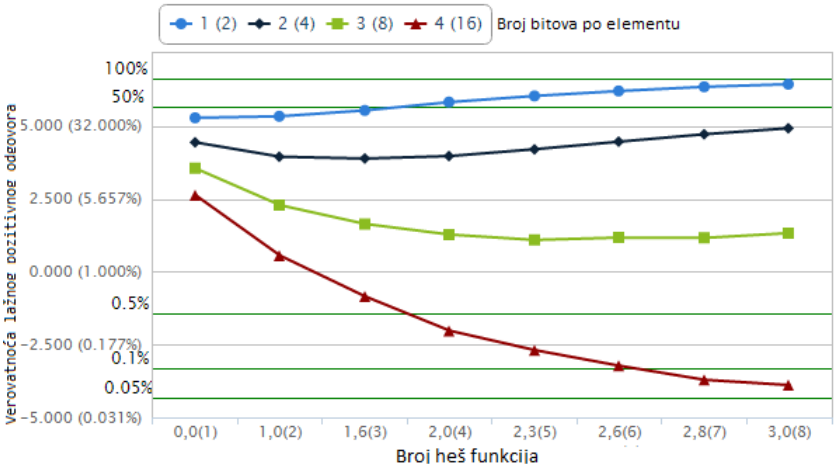
Sa 8 bitova po elementu primećuje se da posle tri heš funkcije kriva brzo dostiže plato i optimalna vrednost heš kodova je 5. Sa 16 bitova po elementu kriva sporije opada nakon 7 heš kodova.

Sa povećanjem broja heš funkcija dolazi do smanjenja verovatnoće lažnog pozitivnog odgovora, ali računanje velikog broja heš kodova zahteva više vremena. Zaključak je da za dva i četiri bita po elementu sa povećanjem broja heš kodova dolazi da povećanja verovatnoće lažnog pozitivnog odgovora, dok

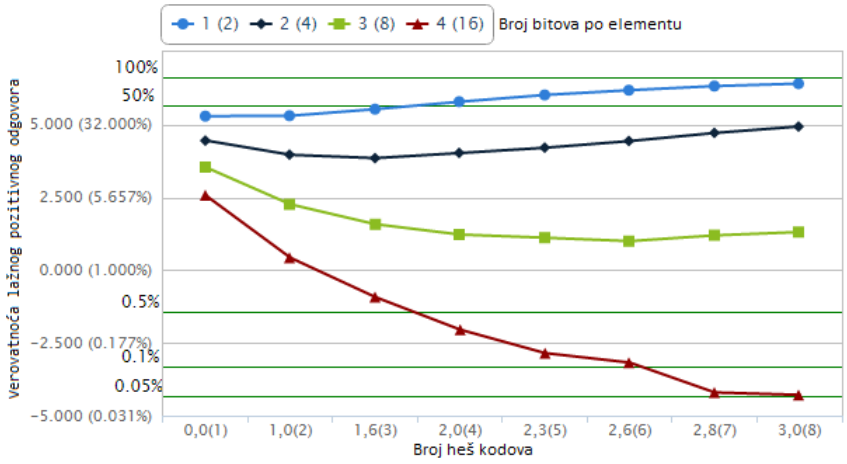


Slika 7.5: Grafički prikaz verovatnoće pojave lažnog pozitivnog odgovora u zavisnosti od broja bitova po elementu u $\log 2$ razmeri korišćenjem SAX heš funkcije.

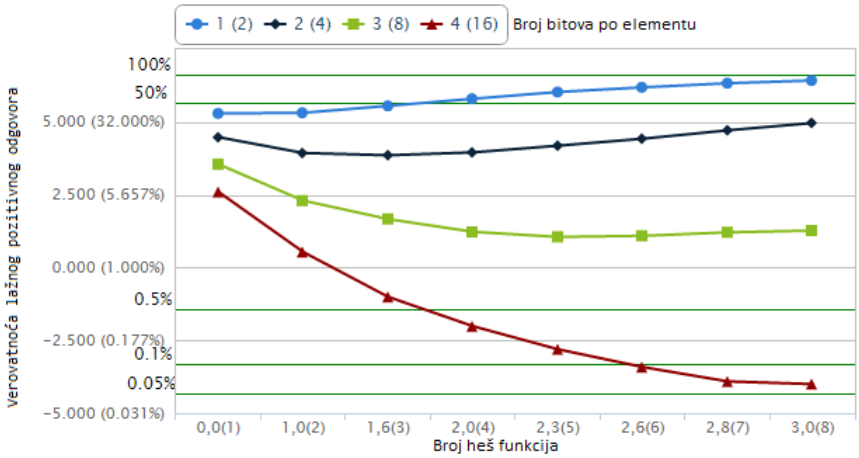
za 8 i 16 bitova po elementu dovoljno je četiri odnosno 8 heš funkcija malu verovatnoću pojave lažnog pozitivnog odgovora.



Slika 7.6: Grafički prikaz verovatnoće pojave lažnog pozitivnog odgovora u zavisnosti od broja heš funkcija u $\log 2$ razmeri korišćenjem Bernstajnove heš funkcije.



Slika 7.7: Grafički prikaz verovatnoće pojave lažnog pozitivnog odgovora u zavisnosti od broja heš funkcija u $\log 2$ razmeri korišćenjem FNV funkcije.



Slika 7.8: Grafički prikaz verovatnoće pojave lažnog pozitivnog odgovora u zavisnosti od broja heš funkcija u $\log 2$ razmeri korišćenjem SAX heš funkcije.

Poglavlje 8

Zaključak

U ovom radu dat je pregled Blumovih filtra kao i njihove primene u p2p mrežama. Blumov filtar je efikasna struktura podataka koja može da sačuva informacije o pripadnosti skupa elemenata smanjujući količinu potrebne memorije i vreme provere pripadnosti elemenata skupu. Blumovi filtri nude fleksibi-lnost u pravljenju izbora između veličine filtra i verovatnoće lažnog pozitivnog alarma. Efikasnost koja se dobija manjim filtrom obično je dovoljna kompenzacija za probleme zbog češće pojave lažnog alarma. Blumov filtar može da se koristi i kao osnova za pravljenje kompleksnijih struktura podataka kao što su Blumier filtar ili brojački Blumov filtar, kao i za prenos podataka o pripadnosti elemenata skupu.
Važno svojstvo filtra je da su podaci u njemu dostupni samo onim korisnicima koji u svom filtru imaju isti element. Zahvaljujući jednosmernosti heš funkcje, pripadnost elementa skupu može da proveri samo korisnik koji može da izračuna heš vrednost, odnosno onaj ko poseduje element. To je značajno u slučaju prenosa važnih skupova podataka preko p2p mreže.
Blumov filtar omogućuje razmenu, odnosno prenos velike količine podataka. Zbog svoje jednostavnosti, ekonomičnosti, brzine i sigurnosti Blumov filtar dobija sve veći značaj i primenu u različitim algoritmima, modernim mrežnim sistemima kao i aplikacijama.

Literatura

[1] K. Aberer, Peer-to-Peer Data Management, Morgan publishers, 2011.

[2] P. K. Gummadi, R. Gummadi, S. D. Gribble, S. Ratnasamy, S. Shenker, I. Stoica, The impact of dht routing geometry on resilience and proximity. In Proc. ACM Int. Conf. on Data Communication, 381–394, New York, NY, USA, 2003. DOI: 10.1145/863955.863998 9.

[3] A. Rao, K. Lakshminarayanan, S. Surana, R. Karp, I. Stoica, Load balancing in structured p2p systems, InProc. 2nd Int. Workshop Peer-to-Peer Systems, 68–79, 2003. DOI: 10.1007/978-3-540-45172-3-6 9.

[4] J. Byers, J. Considine, M. Mitzenmacher, Simple load balancing for distributed hash tables, In Proc. 2nd Int. Workshop Peer-to-Peer Systems, 80–87, 2003. DOI: 10.1007/978-3-540-45172-3-7 9.

[5] A. I. T. Rowstron, P. Druschel, Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems, In Proc. IFIP/ACM Int. Conf. on Distributed Systems Platforms, 329–350, 2001. DOI: 10.1007/3-540-45518-3-18 7, 10.

[6] A. H. Rasti, D. Stutzbach, R. Rejaie, On the long-term evolution of the two-tier gnutella overlay, In Proc. 25th Annual Joint Conf. of the IEEE Computer and Communication Societies, 1–6, 2006. DOI: 10.1109/INFOCOM.2006.44 10.

[7] A. Broder, M. Mitzenmacher, Network Applications of Bloom Filters: Survey, Internet Mathematics, 1:4, 485-509 2003.

[8] J. Roozenburg, A Literature Survey on Bloom Filters. Research Assigment in Computer Science 2005.

[9] L. Fan, P. Cao, J. Almeida, and A. Broder. Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol. IEEE/ACM Trans. Netw., 8(3):281-293, 2000.

[10] A. Rousskov, D. Wessels, Cache Digests. Computer Networks and ISDN Systems 30:22-23, 2155-2168 1998.

[11] https://jqueryui.com/

[12] https://select2.github.io/

[13] http://www.highcharts.com/

[14] http://www.eternallyconfuzzled.com/tuts/algorithms/jsw_tut_hashing.aspx