SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 244

Raspodijeljeni sustav za analizu tijekova aktivnosti na Webu

Zvonimir Pavlinović

Zagreb, svibanj 2011.

DIPLOMSKI ZADATAK br. 244

Pristupnik: **Zvonimir Pavlinović**
Studij: **Računarstvo**
Profil: **Računarska znanost**

Zadatak: **Raspodijeljeni sustav za analizu tijekova aktivnosti na Webu**

Opis zadatka:
Proučiti i opisati načela i tehnologije za potporu rada Weba u stvarnom vremenu. Opisati najčešće korištene protokole za komunikaciju korisnika u društvenim mrežama u stvarnom vremenu. Predložiti arhitekturu i programski ostvariti proširiv raspodijeljeni sustav za prikupljanje, objedinjavanje i analizu korisničkih tijekova aktivnosti u društvenoj mreži u stvarnom vremenu. Navesti korištenu literaturu i primljenu pomoć.

Zahvaljujem se prof.dr.sc. Siniši Srbljiću na pruženoj prilici te mentorstvu tijekom čitavog studija. Želim se zahvaliti dipl.ing. Klemi Vladimiru na svoj pomoći tijekom izrade ovoga rada. Posebno se želim zahvaliti svojoj obitelji bez čijih savjeta, smjernica i potpore ovaj rad ne bih imao priliku ostvariti.

Sadržaj

1. Uvod

Razvijanjem računala, globalne mreže Internet [1] te World Wide Web [2] informacijskog sustava ostvarila se težnja čovjeka za jednostavnim i brzim oblikom širenja informacija i znanja. Korisnici, koji posjeduju računalo, na jednostavan način imaju mogućnost besplatnog pronalaska željenih informacija. Predajući samo nekoliko osnovnih podataka sustavu, korisnici određuju izvor informacija te dobivaju tražene informacije. Povećanjem broja korisnika povećava se i broj izvora informacija što otežava pojedincu pronalazak i grupiranje informacija vezanih za željenu tematiku. Razvijanjem naprednih web tražilica omogućen je jednostavan pronalazak informacija, no trenutni model raspodijeljene obrade sustava Web, klijent-poslužitelj [3], i dalje je imao znatna ograničenja.

Veliki broj pronađenih izvora podataka, iako grupiranih, zahtijevao je podosta radnje i utrošenog vremena od strane korisnika s ciljem pribavljanja najnovijih informacija. Korisnici bi morali svaki put predati zahtjev ciljanom poslužitelju za informacijama za koje ni sami ne znaju jesu li osvježene. U slučaju da informacije nisu osvježene, ovakav način rada dovodi do pribavljanja već pribavljenih informacija što predstavlja redundantan rad korisnika i poslužitelja te nepotrebno povećanje mrežnoga prometa. U vidu rješavanja spomenutih ograničenja, kao dodatni model rada Weba uveden je model objavi-pretplati[3] koji lišava korisnika nepotrebnog i beskorisnog rada te mu omogućava pregledavanje informacija na jednostavnijoj razini.

Prvobitna ostvarenja sustava temeljenih na modelu objavi-pretplati od korisnika zahtijevaju određivanje izvora informacija i tema čiji sadržaj korisnik želi pratiti, čime korisnik dobiva ulogu pretplatnika. Sav rad koji je korisnik morao obavljati pri modelu klijent-poslužitelj za njega pri ovom modelu obavlja primjenski program sakupljač, učestalim provjeravanjem dostupnosti novih sadržaja. Iako predstavlja napredak u odnosu na klijent-poslužitelj model, novi model objavi-pretplati imao je još uvijek nedostatak redundantnog rada sakupljača i objavljivača informacija. Razvoj PubSubHubbub [4] protokola omogućio

je optimalno smanjivanje rada objavljivača i programa sakupljača.

Sustavi koji se temelje na PubSubHubbub protokolu uvode novi entitet koji preuzima dio odgovornosti objavljivača informacija te smanjuje količinu njegovog rada. Također, smanjuje se i količina rada programa sakupljača. Iako se povećava složenost infrastrukture, osim minimiziranja rada sakupljača i poslužitelja, smanjuje se i mrežni promet.

Osim PubSubHubbub sustava, razvijene su mnoge tehnologije koje omogućuju korisnicima pribavljanje te korištenje podataka u stvarnom vremenu. Usluge, temeljene na potonjim tehnologijama, omogućile su pružanje komunikacijskih i informacijskih usluga koje predstavljaju novu generaciju World Wide Web sustava. Spomenute tehnologije i usluge, kao i PubSubHubbub protokol, nastale su pod utjecajem koncepta Weba u stvarnom vremenu [4].

U ovome radu biti će detaljno opisani modeli komunikacije u raspodijeljenim sustavima s naglaskom na model objavi-pretplati. Biti će opisana arhitektura i osnovni način World Wide Web sustava. Također, opisan je razvoj World Wide Weba te potreba za razvojem tehnologija i koncepata za komunikaciju i raspodijelu podataka u stvarnom vremenu. Opisane su tehnologije koje su danas široko korištene te koje su razvijene pod utjecajem koncepta Weba u stvarnom vremenu. Detaljno su opisane tehnološke karakteristike komunikacijskih protokola RSS i Atom, koji se najčešće koriste za objavljivanje novih informacija u stvarnom vremenu, te tehnički detalji PubSubHubbub protokola koji je nadogradnja spomenutih protokola te predstavlja okosnicu ovoga rada. Također, kao cilj ovoga rada, biti će predstavljeno i detaljno opisano programsko ostvarenje spomenutoga sustava s vlastitim nadogradnjama te opisanim prostorima za daljni napredak.

2. Raspodijeljeni sustavi

2.1. Uvod u raspodijeljene sustave

Sustav koji se sastoji od skupa neovisnih računala koji korisniku izgledaju kao cjeloviti sustav naziva se raspodijeljeni sustav [3]. Prema vrsti usluge koju obavljaju, raspodijeljeni sustavi dijele se na računalne, informacijske, sustave za pružanje informacijskih i komunikacijskih usluga te prožimajuće raspodijeljene sustave [3]. Računalni raspodijeljeni sustavi, kao što su grozd (engl. *cluster*), splet (engl. *grid*) te oblak (engl. *cloud*), koriste se za obavljenje složenih i računalno zahtjevnih zadataka. Informacijski sustavi pružaju usluge obrade transakcija te integriranja poslovnih aplikacija. Sustavi za pružanje informacijskih te komunikacijskih usluga predstavljaju sloj usluga te primjenskih komponenti koje sustav pruža. Prožimajući sustavi predstavljaju vrstu raspodijeljenih sustava koji intereagiraju sa svojom fizičkom okolinom te čiji su predstavnik, na primjer, senzorske mreže.

Razlog razvijanja raspodijeljenih sustava jest inherentna raspodijeljenost korisnika pa samim time informacija i ostalih resursa. Obilježja raspodijeljenih sustava su paralelne aktivnosti koje se koordiniraju razmjenom poruka između odgovarajućoh eniteta u sustavu. Entiteti dijele sredstva, ne poznaju stanja drugih entiteta te ne posjeduju isti vremenski takt. U slučaju ne poštivanja navedenih ograničenja, složenost raspodijeljenih sustava eksponencijalno se povećava, u vidu međusobne komunikacije. Kako bi se ispoštovala opisana ograničenja, u raspodijeljenim sustavima postoji posrednički sloj koji omogućava vremensko usklađivanje entiteta te koji prikriva činjenicu da su sredstva i procesi raspodijeljeni.

Kako bi se ostvarilo svojstvo fleksibilnosti na zahtjeve tržišta te neprekidno razvijanje, raspodijeljeni sustavi se ostvaruju uzimajuću o obzir otvorenost, transparentnost te sposobnost sustava na razmjeran rast [3]. Otvorenost nalaže da se sustav razvija te koristi tehnološke norme koje su vlasnički neovisne, javne te dobro određene. Ostvarujući transparentnost sustava iz određene perspektive,

korisnicima se prikrivaju značajke sustava što omogućuje lakše razvijanje sustava te smanjuje složenost sustava. Sposobnost razmjernog rasta predstavlja mogućnost pružanja usluga, po određenoj kvaliteti, i u slučaju da se povećava razina rasprostranjenosti, količine podataka, korisnika te ostalih resursa koji su vitalni za rad sustava.

Neovisno o namjeni raspodijeljenog sustava, komunikacija računala inherentno predstavlja osnovu rada te sukladno tomu određuje kvalitetu usluga koje sustav pruža te složenost izvođenja. Komunikacija između dva računala povezana mrežom posjeduje nekoliko obilježja koja određuju način na koji se podaci raspodijeljuju unutar sustava. Obilježja komunikacije opisana su u poglavlju 2.1. dok su u poglavlju 2.2. opisani modeli raspodijele podataka u raspodijeljenim sustavima.

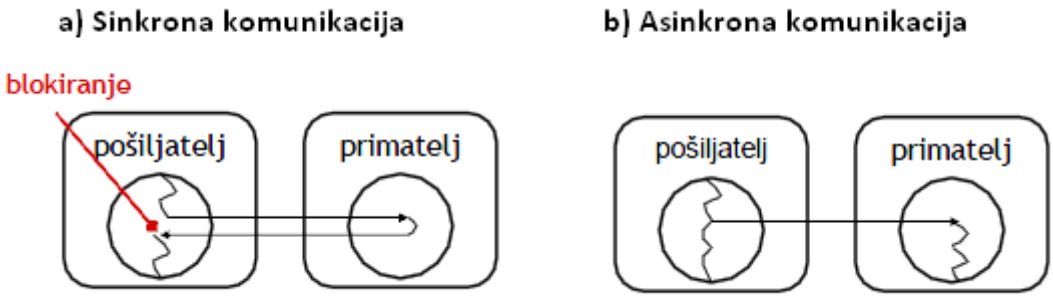
2.2. Obilježja komunikacije

Komunikacija između autonomnih procesa koji se izvode na različitim računalima u raspodijeljenim sustavima može se podijeliti na temelju obilježja ostvarivanja veze, uvjerenja isporuke poruke, sinkronosti, ovisnosti o referenci te preuzimanja podataka [3].

Prema obilježju ostvarivanja veze komunikacija između procesa može biti konekcijska i bezkonekcijska. Pri konekcijskoj komunikaciji između procesa stvara se eksplicitna veza, pomoću kontrolnih poruka, neposredno prije razmjene podataka. U slučaju bezkonekcijske komunikacije ne ostvaruje se veza prije slanje podataka već sve poruke prenose podatke.

Promatrajući obilježje uvjerenja isporuke poruke, komunikacija može biti perzistentna i tranzijentna. Perzistentna komunikacija osigurava isporuku poruke i u slučaju da neki od sudionika komunikacije nisu dostupni u trenutku izvođenja procesa komunikacije. Tranzijentna komunikacija osigurava isporuku poruke samo u slučaju da su svi sudionici komunikacije dostupni u trenutku slanja poruka.

Sinkrona komunikacija blokira pošiljatelja do primitka potvrde tj. podataka od strane primatelja. Asinkrona komunikacija, u suprotnosti sinkronoj komunikaciji, omogućuje pošiljatelju nastavak rada između slanja zahtjeva i primanja odgovora. Na slici 2.1. prikazan je model sinkrone (a) i asinkrone komunikacije (b).



Slika 2.1: Model sinkrone (a) i asinkrone (b) komunikacija

U slučaju da procesi u raspodijeljenom sustavu mogu komunicirati isključivo poznavanjem reference tj. jedinstvene oznake (adrese) drugih procesa tada se radi o komunikaciji ovisnoj o referenci. U slučaju da procesi ne moraju poznavati referencu drugih procesa, tada se radi o komunikaciji neovisnoj o referenci.

Bitna podjela komunikacije vrši se preko čimbenika preuzimanja podataka. U slučaju da se preuzimanje podataka vrši na načelu zahtjev-odgovor, gdje jedan proces zahtijeva te čeka primitak podataka od drugih procesa, tada se radi o povlačenju podataka. Kada se podaci preuzimaju na načelu potiskivanja, tada proces koji je poslao zahtjev nesmetano nastavlja svoj rad dok primitak podataka ostvaruje posebnim metodama.

Opisana obilježja komunikacije karakteristična su za pojedine modele raspodijeljene obrade opisane u poglavlju 2.2. Također, predstavljaju ograničenja istih u pružanju usluga krajnjem korisniku što predstavlja ključan čimbenik za razvijanje raspodijeljenih usluga.

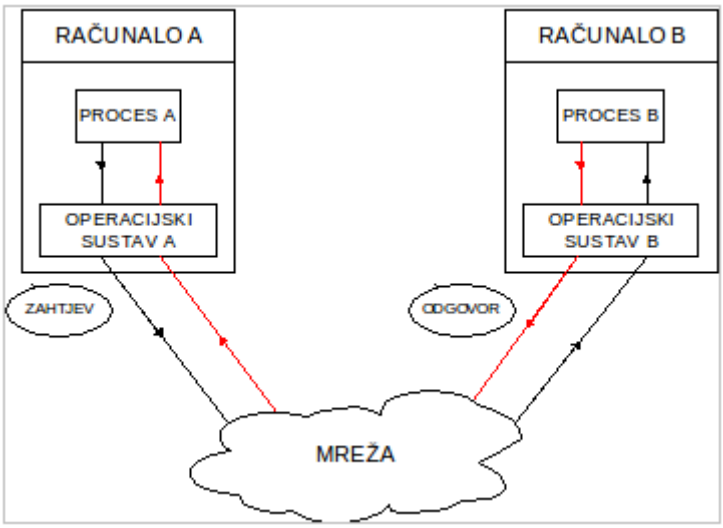
2.3. Modeli komunikacije

Kako je već spomenuto, obilježja komunikacije predstavljaju ključan čimbenik u razvoju raspodijeljenih sustava te su karakteristična za pojedini model komunikacije. U slijedećim poglavljima opisani su najkorišteniji modeli komunikacije koji se primjenjuju na globalnoj razini te su od posebne važnosti za razvoj programskih sustava temeljenih na raspodijeljenim sustavima.

2.3.1. Klijent-poslužitelj model

Model komunikacije (raspodijeljene obrade) klijent-poslužitelj zasnovan je na strogoj podjeli uloga između procesa koji ostvaruju komunikaciju. Klijentski pro-

ces zahtijeva uslugu dok poslužiteljski proces, sukladno nazivu, poslužuje klijenta ovisno o njegovom zahtjevu. Klijentski proces priprema i šalje zahtjev poslužitelju koji obrađuje zahtjev te vraća odgovor klijentu. Pojednostavljeni oblik komunikacije korištenjem modela klijent-poslužitelj prikazana je na slici 2.2.



Slika 2.2: Pojednostavljeni prikaz klijent-poslužitelj modela komunikacije

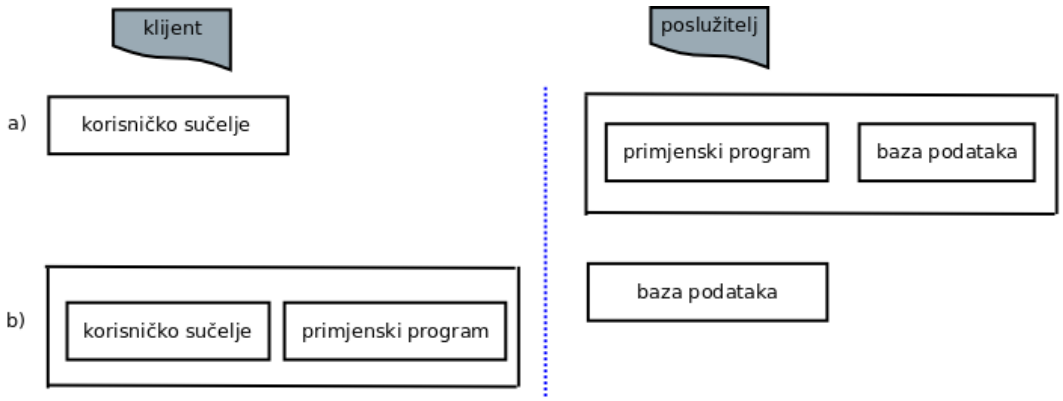
Komunikacija temeljena na modelu klijent-poslužitelj posjeduje slijedeća obilježja:

- ovisnost o referenci: klijent mora znati jedinstvenu oznaku poslužitelja kako bi započeo proces komunikacije te sukladno tomu poslužitelj mora znati oznaku klijenta kako bi vratio odgovor
- načelo povlačenja: komunikacija se vrši na principu zahtjev-odgovor gdje klijent šalje zahtjev, a poslužitelj vraća odgovor
- tranzijentnost: proces komunikacije završava uspješno jedino u slučaju da su oba sudionika dostupna tijekom spomenutog procesa

Sinkronost te (bez)konekcijsko obilježje ovisi o samome ostvarenju sustava. Najpoznatije ostvarenje sustava temeljenog na modelu klijent-poslužitelj jest *World Wide Web* koji je detaljnije opisan u poglavlju 3.

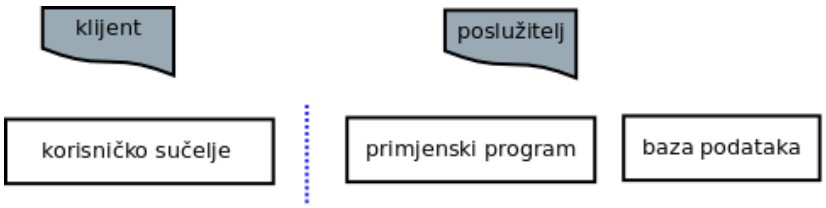
Izvedba klijenta i poslužitelja određuju arhitekturu sustava temeljenog na modelu klijent-poslužitelj. Dvoredna arhitekura klijent-poslužitelj (engl. *two-tier*) predstavlja pojednostavljenu arhitekturu gdje se na podsustavu klijenta nalazi programski sustav korisničkog sučelja u inačici *mršavog klijenta* te dodatno primjenski sustav u inačici *debelog klijenta* [3]. Na poslužiteljskoj strani nalaze se

primjenski program i baza podataka te samo baza podataka, respektivno. Na slici 2.3. prikazane su obje inačice dvoredne arhitekture klijent-poslužitelj modela komunikacije.



Slika 2.3: Arhitektura inačica *mršavog* (a) te *debelog* (b) klijenta

Za razliku od dvoreredne arhitekture, troredna arhitektura (engl. *three-tier*) programske sustave raspoređuje na tri podsustava [3]. Na strani klijenta se nalazi se korisničko sučelje dok poslužiteljski sustav čine dva podsustava koji izvode sustave primjenskog programa te baze podataka, kao što je prikazano na slici 2.4.

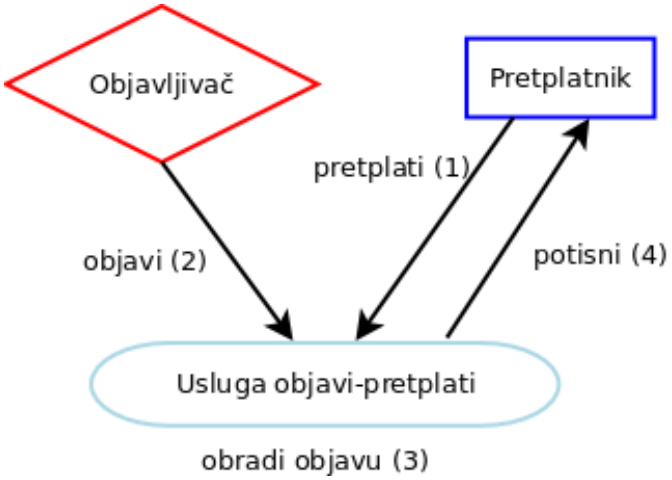


Slika 2.4: Troredna arhitektura modela klijent-poslužitelj

2.3.2. Objavi-pretplati model

Kao i model klijent-poslužitelj, model objavi-pretplati se temelji na strogoj podjeli uloga među sudionicima komunikacije. Uloge koje sudionici mogu poprimiti tijekom komunikacije jesu pretplatnik, objavljivač te usluga objavi-pretplati [3]. Za razliku od klijent-poslužitelj modela gdje klijent eksplicitno šalje zahtjev svaki put kada želi uslugu, objavi-pretplati model svoj rad temelji na pretplatama te potiskivanju podataka. Pretplatnici se pretplaćuju na određenu temu objavljivača te se podaci o pretplati spremaju pri usluzi objavi-pretplati. Svaka se objava informacija šalje ka usluzi koja na temelju preplata filtrira sadržaj informacija te

ga šalje ka odgovarajućim pretplatiteljima. Pojednostavljeni rad modela objavi-pretplati prikazan je na slici 2.5.



Slika 2.5: Pojednostavljeni oblik rada modela objavi-pretplati

Komunikacija temeljena na modelu objavi-pretplati posjeduje slijedeća obilježja:

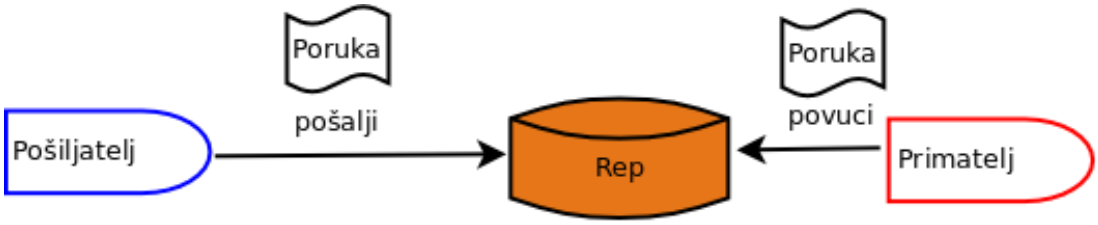
- bezkonekcijska komunikacija: između dva procesa koja komuniciraju ne postoji eksplicitno uspostavljanje veze pri komunikaciji, već se poruke šalju usluzi objavi-pretplati koja naknadno raspodijeljuje filtrirane objavljene informacije
- perzistentnost: pri komunikaciji dva procesa nije potrebno da oba procesa budu aktivna jer posrednik tj. usluga objavi-pretplati pohranjuje poruke
- anonimnost: objavljivač ne mora poznavati jedinstvenu oznaku pretplatnika jer se o tome brine usluga objavi-pretplati
- asinkronost: objavljivač šalje poruku te nastavlja sa svojim radom dok je za sprovođenje ostatka komunikacije zaslužna usluga objavi-pretplati
- načelo postiskivanja: usluga objavi-pretplati potiskuje objavljene podatke ka pretplatnicima bez njihova ranijeg eksplicitnog zahtjeva za istima

Usluga objavi-pretplati, osim kao centralizirana inačica, može biti ostvarena i kao zaseban raspodijeljeni posrednički sustav. Ovakvim ostvarenjem usluge sustav dobiva sposobnost razmjernog rasta pri značajnijim povećanjima broja pretplatnika i objavljivača. Svako računalo unutar sustava odgovorno je za raspodijelu sadržaja određenom broju pretplatnika te posjeduje tablicu usmjeravanja kako bi sadržaje koje primi usmjerilo ka odgovorajućim računalima. Ovakav način

rada predstavlja klasični oblik raspodijeljene usluge objavi-pretplati dok stvarna rješenja ovise o posjedničkim potrebama.

2.3.3. Model komunikacije porukama

Model komunikacije porukama, za razliku od prethodno opisanih modela, temelji se na blažoj podjeli uloga. Uloge koju računala mogu dobiti pri ovom modelu jesu pošiljatelj i primatelj poruke te rep [3]. Pošiljatelj šalje poruku primatelju koja se ne predaje dotičnome izravno već se prvotno sprema u pripadnu mu rep. Rep, slično usluzi objavi-pretplati, predstavlja posrednički sustav koji služi za spremanje poruka te za prosljeđivanje poruka primatelju na njegov zahtjev. Rep je organiziran kao FIFO (engl. *First In First Out*) stog gdje se poruke poslužuju primatelju po redu dospijeća te ga se u literaturi može pronaći pod nazivom Message Oriented Middleware (hrv. *porukama vođen posrednik*) [5]. Na slici 2.6. simbolično je prikazan rad modela komunikacije porukama.



Slika 2.6: Pojednostavljeni oblik rada modela komunikacije porukama

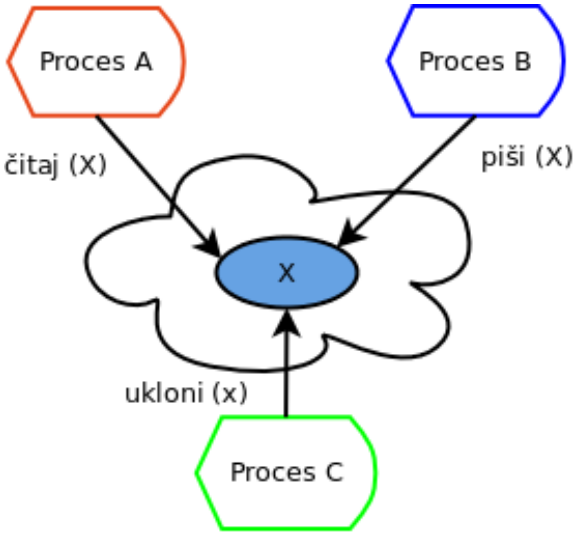
Model komunikacije porukama posjeduje slijedeća obilježja:

- bezkonekcijska komunikacija: između dva procesa koja komuniciraju ne postoji eksplicitno uspostavljanje veze pri uspostavljanju komunikacije već se poruke šalju u primateljev rep
- perzistentnost: pri komunikaciji dva procesa nije potrebno da oba procesa budu aktivna jer primateljev rep pohranjuje poruke
- ovisnost o referenci: objavljivač ne mora poznavati jedinstvenu oznaku primatelja, ali mora poznavati jedinstvenu oznaku primateljeva repa
- asinkronost: pošiljatelj šalje poruku te nastavlja sa svojim radom dok se ostatak komunikacije sprovodi na način da primatelj dohvaća podatke iz svog repa
- načelo povlačenja: primatelj dohvaća podatke iz odgovarajućeg reda na načelu povlačenja podataka

2.3.4. Dijeljeni podatkovni prostor

Pri modelu dijeljenog podatkovnog prostora pozornost se skreće sa sudionika u komunikaciji ka podacima koji se razmjenjuju. Arhitektura sustava se temelji na podacima koji su pohranjeni na dijeljenom podatkovnom prostoru. Svako računalo, tj. pripadni autonomni proces, može komunicirati s dijeljenim prostorom obavljajući operacije čitanja, pisanja te uklanjanja uređenih n-torki podataka [3].

Uređena n-torka je nedjeljiva zbirka imenovanih podataka koji mogu biti tipizirani. Čitanje te uklanjenje n-torki omogućeno je prosljeđivanjem obrasca podatkovnom prostoru na temelju kojeg se izvode tražene operacije nad podacima koji se podudaraju s obrazcem. Ilustrativni opis izvođenja komunikacije modela dijeljenog podatkovnog prostora prikazan je na slici 2.7.



Slika 2.7: Pojednostavljeni oblik rada modela komunikacije porukama

Komunikacija dijeljenim memorijskim prostorom posjeduje slijedeća obilježja:

- bezkonekcijska komunikacija: između dva procesa koja komuniciraju ne postoji eksplicitno uspostavljanje veze pri uspostavljanju komunikacije već se poruke pišu u dijeljeni prostor
- perzistentnost: pri komunikaciji dva procesa nije potrebno da oba procesa budu aktivna jer se poruke spremaju u dijeljeni prostor
- anonimnost: pošiljatelj ne mora poznavati jedinstvenu oznaku primatelja
- asinkronost: pošljatelj piše poruku te nastavlja sa svojim radom dok se ostatak komunikacije sprovodi na način da primatelj dohvaća podatke iz dijeljenog prostora

– načelo povlačenja: primatelj "povlači" podatke iz dijeljenog prostora

2.3.5. Modeli više razine

Spomenuti i opisani modeli komunikacije postigli su veliku primjenu u komercijalnim te posjedničkim rješenjima. Iako široko korišteni, u pojedinim primjenama spomenuti modeli imali su nedostatke koji su svladani razvijanjem modela na višoj razini apstrakcije. Takvi modeli pri internoj komunikaciji koriste neke od opisanih modela, ovisno o problemu kojeg pokušavaju svladati.

Model ravnopravnih sudionika

Nedostatci modela klijent-poslužitelj, kao što su smanjena sposobnost razmjernog rasta te jedna točka ispada, razlog su razvijanja modela ravnopravnih sudionika (engl. *peer to peer*) [6]. Svaki čvor u mreži (engl. *peer*) ima ulogu klijenta i poslužitelja. Čvorovi slobodno izlaze te ulaze u prekrivajuću mrežu (engl. *overlaying network*) koja je ostvarena na programskoj razini nad stvarnom mrežnom topologijom. Svaki čvor nudi dio svojih usluga te u svakom trenutku može poprimiti ulogu klijenta, poslužitelja te usmjernika.

Zbog decentralizacije podataka, sposobnosti razmjernog rasta te činjenice da nemaju jednu točku ispada, mreže ravnopravnih sudionika često su korištene, kako u komercijalne svrhe, tako i kao posjednička rješenja. Često korištena komercijalna rješenja su Napster, Gnutella, Kazaa te FreeNet [7].

Obilježja komunikacije procesa unutar sustava ostvarenih modelom ravnopravnih sudionika ovise o posjedničkim rješenjima.

Programski i pokretni agenti

Programski agenti model je komunikacije koji se temelji na računalnim programima koji obavljaju zadaće za korisnike tj. svog vlasnika [8]. Spomenuti programi raspolažu svojstvima kao što su umjetna inteligencija, samostalnost, reaktivnost te proaktivnost. Zadaće im variraju od prikupljanja podataka do izvršavanja složenih izračuna.

Pokretni agenti, za razliku od programskih agenata, imaju sposobnost kretanja kroz mrežu tj. premještanja s računala na računalo [8]. Posjeduju svojstva kao i programski agenti, stoga kretanje obavljaju samostalno, bez interakcije s korisnicima. Primjer korištenja opisanih modela jesu senzorske mreže.

Obilježja komunikacije procesa unutar sustava ostvarenih modelima programskih i pokretnih agenata ovise o posjedničkim rješenjima.

3. Arhitektura raspodijeljenog sustava World Wide Web

Razvoj Interneta, globalne mreže računala, postavio je temelj za razvoj informacijskih sustava koji omogućuju čovjeku jednostavnu i brzu razmjenu te pregled podataka. U tu svrhu razvijen je *World Wide Web*, informacijski raspodijeljeni sustav nad mrežom Internet koji je u kratkom vremenu stekao veliku primjenu. Razvoj *Weba*, kako će ga se nastavku rada povremeno oslovljavati, tijekom vremena omogućio je razvoj novih funkcionalnosti. Korisnicima je omogućeno korištenje usluga koje su se kretale od razine jednostavnog pregledavanja podataka do razine stvaranja podataka, međusobne komunikacije te korištenja virtualnih računalnih sustava.

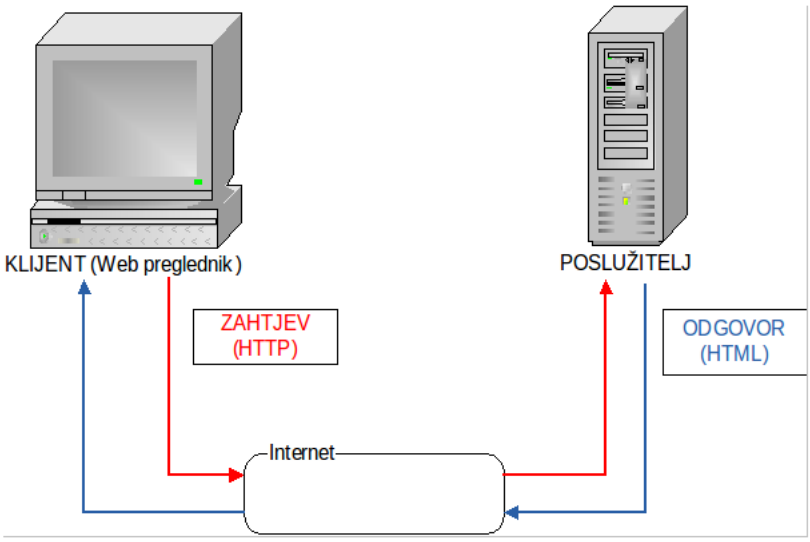
Kako je opisano u poglavlju 2., međusobna suradnja i interakcija računala u raspodijeljenim sustavima određena je modelom raspodijeljene obrade tj. komunikacije računala. Prvobitan model raspodijeljene obrade na Webu bio je model klijent-poslužitelj koji se trenutno najviše koristi te po kojemu je Web postao prepoznatljiv. Korisnici predaju zahtjev ciljanom poslužitelju te potonji, nakon obrade zahtjeva, korisniku vraća natrag tražene podatke što predstavlja komunikaciju na načelu povlačenja podataka. Tijekom vremena model klijent-poslužitelj je postao, za potrebe mnogih korisnika, nedovoljno koristan te se na Webu pojavljuju usluge koje svoj rad temelje na modelu raspodijeljene obrade objavi-pretplati čije je glavno obilježje komunikacija na načelu potiskivanja podataka.

3.1. Osnovni način rada

Model raspodijeljene obrade klijent-poslužitelj, kao što je već spomenuto, prvobitni je i osnovni model primijenjen na Webu. Osnovni način rada zahtijeva pripremu te slanje zahtjeva od strane pošiljatelja te obradu zahtjeva i slanje podataka(obavljanja usluge) natrag od strane primatelja. Osnovni komunikacijski

model podrazumijeva međuprocesnu komunikaciju koja zahtijeva vremensku usklađenost.

Kako bi pregledavali omiljene sadržaje te prikupljali potrebne informacije preko Weba, korisnici koriste, za takve primjene, specijalizirane alate kao što su web preglednici. Web preglednici u ime korisnika obavljaju dužnosti klijenta u modelu klijent-poslužitelj, kao što su priprema i slanje zahtjeva te obrada primljenih podataka. Korisnici web pregledniku predaju URI (engl. *Uniform Resource identifier*) [9] adresu izvora informacija što ujedno predstavlja njihovu jedinu obvezu i radnju u procesu pribavljanja željenih podataka. Web preglednik zatim automatski stvara zahtjev koji se HTTP (engl. *HyperText Transfer Protocol*) [10] protokolom prenosi do izvora informacija (poslužitelja) koji obrađuje zahtjev te vraća podatke u HTTP (engl. *HyperText Markup Language*) [11] zapisu. Završni korak obavlja web preglednik koji vraćene podatke korisniku prikazuje u njemu prihvatljivijem obliku. Opisani proces prikazan je na slici 3.1.



Slika 3.1: Prikaz osnovnog rada World Wide Web sustava

Slanje zahtjeva poslužitelju od strane klijenta te vraćanje podataka klijentu od strane poslužitelja obavljaju se primjenom TCP (engl. *Transfer Control Protocol*) [12] protokola na mrežnoj razini. Komunikacija ostvarena spomenutim protokolom jest konekcijska, tranzijentna i sinkrona što predstavlja značajno ograničenje pri korištenju usluga Weba.

3.2. Formalni opis

Web predstavlja ostvarenje programske arhitekture Representional State Transfer (REST) [13], koja se zasniva na stanju resursa. REST predstavlja teorijski model programske arhitekture raspodijeljenih sustava gdje su osnovni elementi podaci, sučelja te komponente. Podaci imaju obilježja sredstva, adrese te prikaza, što u ostvarenju Weba odgovara usluzi, URI adresi te HTML zapisu, respektivno. Komponente mogu biti korisnički agent (engl. *user agent*) kojemu u primjeru Weba odgovara web preglednik, izvorni poslužitelj (engl. *origin server*) te posrednik (engl. *proxy*). Osnovni tipovi sučelja su klijent te poslužitelj dok su pomoćni tipovi usmjernik te priručna memorija (engl. *cache*). Sučelja služe za razmjenu podataka te predstavljaju ulogu koju komponenta poprima.
REST uvodi nekoliko ograničenja na svoju arhitekturu čija su ostvarenja uvjetovana posjedničkim potrebama:

- klijent-poslužitelj podjela zadaća: omogućuje neovisan razvoj klijenta i poslužitelja
- poslužitelj bez pohrane stanja: smanjuje se složenost sustava, što omogućuje jednostavniji razvoj sustava
- priručna memorija na klijentu kao opcija: smanjuje se opterećenost poslužitelja
- standardizirana sučelja: omogućuje lakše i neovisno razvijanje te pojednostavljuje arhitekturu sustava
- slojevitost komponenti: klijent nije svjestan poslužuje li ga izvorni poslužitelj ili posrednik što sustavu daje sposobnost na razmjeran rast te ujednačavanje opterećenja
- izvršavanje koda na zahtjev: omogućuje se izvršavanje koda na klijentskoj strani što kao posljedicu ima smanjenje opterećenja poslužitelja

Navedena ograničenja omogućuju lakši razvoj REST sustava te pojednostavljuju arhitekturu sustava. Korištenjem HTTP protokola, Web predstavlja RESTFull Http inačicu REST arhitekture. Komunikacija između klijenta i poslužitelja izvodi se otvorenim HTTP protokolom što omogućuje standardizaciju sučelja te neovisan razvoj klijenta i poslužitelja. Poslužitelji ne pamte stanje na temelju zahtjeva što je izravna posljedica činjenice da je HTTP *stateless* (hrv. *bez stanja*) protokol. HTTP protokol podržava CRUD (Create, Read, Update, Delete) [10] metode kao i neke pomoćne metode koje su opisane u tablici 3.1.

Tablica 3.1: Prikaz i opis metoda HTTP protokola

IME	**OPIS**
GET	Idempotentna i sigurna[1] metoda koja dohvaća resurse
PUT	Idempotentna metoda koja osvježava stanje resursa
DELETE	Idempotentna metoda koja briše resurse
POST	Metoda koja stvara podatke
HEAD	Poslužitelj vraća odgovor bez tijela odgovora tj. samo metapodatke
OPTIONS	Poslužitelj vraća postavke te preduvjete koji su potrebni klijentu za ostvarenje komunikacije
TRACE	Klijentu se vraća poslani zahtjev s podacima o posrednicima koji su sudjelovali u prosljeđivanju zahtjeva
CONNECT	Koristi se za spajanje na posrednik koji predstavlja početnu točku u uspostavljanju sigurnog tunela komuniciranja

Svaki HTTP zahtjev veže se na određeni URI identifikator koji jednoznačno određuje resurs nad kojim se metoda, određena u zahtjevu, želi sprovesti. URI identifikator sastoji se od slijedećih komponenti:

- shema: označava pristupni protokol
- domaćin: simbolični naziv za adresu poslužitelja koji se preko usluga DNS sustava mapira u IP adresu
- put: put do resursa
- upit: upit koji pomaže pobližoj identifikaciji resursa
- odlomak: dio identifikatora koji se odnosi na određeni resurs unutar glavnog resursa, a čija namjena ovisi o posjedničkim potrebama (pri Webu web preglednici prikazaju izravno dio web stranice koji je određen odlomkom)

U tablici 3.2. prikazana je izmišljena URI adresa sa svim gore opisanim i navedenim komponentama.

Tablica 3.2: Primjer URI identifikatora

Shema	Domaćin	Put	Upit	Odlomak
http://	www.host.com	/p/u/t/	?upit1=x&upit2=y	#odlomak

[1]Sigurna metoda ne mijenja stanje resursa

3.3. Neprekidno razvijanje

Popularizacija Weba dovela je do njegova neprekidna razvoja koje je rezultiralo ostvarenjem novih funkcionalnosti. Razvoj skriptnog programskog jezika Javascript, koji se izvodi u web pregledniku te predstavlja REST ograničenje izvršenja koda na zahtjev, smanjio je opterećenje poslužitelja te omogućio evoluciju RIA sustava (engl. *Rich Internet Application*) [4]. Razvoj tehnologija poput AJAX-a (engl. *Asynchronous Javascript And XML*) [4] omogućio je novi korak u evoluciji web primjenskih sustava. Komunikacija između web preglednika i poslužitelja postala ja asinkrona što je oslobodilo prostor za ostvarenje dodatnih funkcionalnosti te povećalo razinu pozitivnog korisničkog doživljaja. Razvojem opisanih tehnologija korisniku su se pružile dodatne mogućnosti osim pregledavanja podataka. Korisnicima je omogućeno stvaranje podataka, vlastitih web-stranica te komunikacija s ostalim korisnicima što je krajnje rezultiralo razvojem velikih društvenih mreža kao što su Facebook, Twitter i MySpace.

Osim pružanja usluga korisnicima, razvoj Weba je omogućio i pružanje usluga računalima. Arhitektura zasnovana na uslugama (engl. *Service Oriented Architecture*) omogućuje računalima automatizirani pronalazak te korištenje udaljenih usluga temeljenih na otvorenim tehnologijama. Osim korištenja, računalima podređenim, REST usluga omogućen je pronalazak te korištenje usluga temeljenih na SOAP (engl. *Simple Object Acces Protocol*) protokolu [13]. Detalji i uvjeti korištenja usluga napisani su u XML [14] zasnovanom WSDL (engl. *Web Service Description Language*) jeziku. Opisani detalji nalaze se u zajedničkom XML zasnovanom repozitoriju UDDI (engl. *Universal Description Discovery and Integration*) [13] koji omogućuje automatizirani pronalazak pogodnih usluga. Poziv usluga se odvija spomenutim SOAP protokolom koji je temeljen na otvorenoj XML tehnologiji koja je detaljnije opisana u poglavlju 4.2.2.

Razvoj Web omogućio je i razvoj računarstva u oblacima (engl. *cloud computing*) [15, 16]. Računarstvo u oblacima predstavlja koncept pri kojem se virtualizirana računalna sredstva, platforme te programski sustavi izlažu kao usluge pod određenim uvjetima korištenja (engl. *Service Level Agreement*). Spomenute usluge se koriste na istom principu kao i npr. električna struja te se plaćaju po korištenju. Kao široko korišten sustav, Web je ubrzo postao bitan čimbenik u razvoju računarstva u oblacima kao sučelje prema prethodno opisanim uslugama. Iako bitni pojmovi i koncepti, uslužna arhitektura te računarstvo u oblacima nisu predmet ovog rada te neće biti detaljno opisani. Prikazani su kao logičan slijed

razvoja Weba te dobro ilustriraju način na koji se razina složenosti arhitekture Weba povećala njegovom popularizacijom.

4. Web u stvarnom vremenu

Razvijanjem Weba, opisanim u poglavlju 3., te povećanjem broja korisnika, proporcionalno se povećao i broj izvora informacija. Korisnicima Web postaje važan izvor informacija što rezultira njegovim sve učestalijim korištenjem. Kako bi se korisnicima omogućio jednostavan pronalazak te grupiranje željenih informacija razvijene su web tražilice od kojih su najpoznatija ostvarenja Google, Yahoo!, Bing, itd. Iako olakšano, opisano korištenje usluga Weba kao nedostatak je imalo povećan rad od strane korisnika zbog velikog broja izvora informacija tzv. web stranica. Korisnici bi se morali "premještati" sa jednog na drugi izvor informacija s ciljem provjeravanja dostupnosti novih sadržaja. Frekvencija pojavljivanja novih sadržaja kod mnogih web stranica nije određena i korisniku poznata zbog tematike samoga sadržaja, što mnoge korisnikove pokušaje čini neuspješnim.

Popularizacija Weba kao posljedicu, također, je imala razvijanje usluga na Webu koje korisnici imaju priliku koristiti u svakodnevnom životu nevezano za računala. Telefonija te emitiranje televizijskog programa nametnuli su se kao slijedeći korak u razvoju bogatih primjenskih sustava na Webu. Također, opisani rad Weba preko modela klijent-poslužitelj kao nedostatak ima redundantno opterećenje klijenta i poslužitelja te povećanje mrežnoga prometa. Kako bi se riješili spomenuti problemi, te ispunili funkcionalni zahtjevi nametnuti od strane korisnika, razvijeni su koncepti koji pokušavaju korisnicima pružiti usluge u stvarnom vremenu.

4.1. Komunikacija porukama

Od pojave računala te razvijanja posjedničkih mreža, komunikacija korisnika u stvarnom vremenu postaje jedna od najkorištenijih primjena raspodijeljenih sustava. Pojavom Interneta postavljeni su temelji za razvijanje arhitektura i protokola koji će omogućiti komunikaciju korisnika u stvarnom vremenu na globalnoj razini. Razvijeni su mnogi posjednički sustavi i protokoli koji su ostvarili postav-

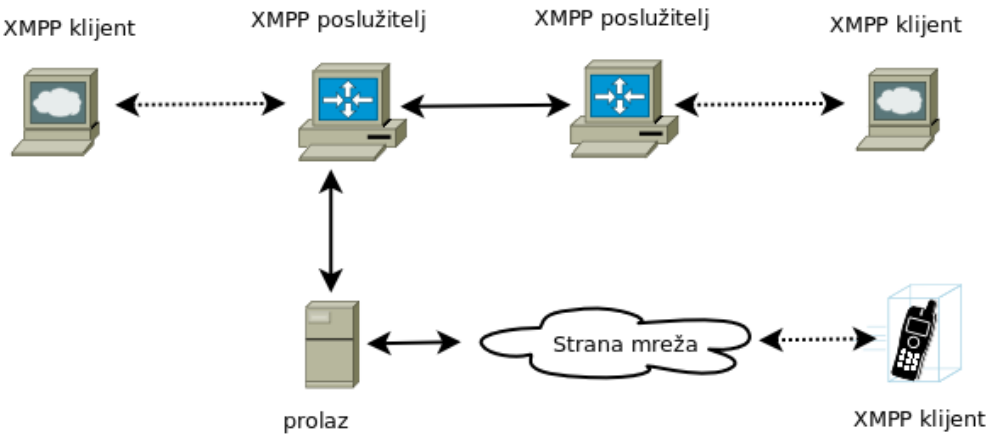
ljene ciljeve, no zakazali na sposobnosti međusobne integracije pošto posjednička rješenja najčešće znače i posjedničke tehnologije. Neovisno o tehnologijama koje koriste, opisani sustavi su korisnicima omogućili izravnu komunikaciju s drugim korisnikom (engl. *unicast*), s više korisnika od jednom (engl. *mulitcast*) te grupnu komunikaciju [17]. Iako su prvotni sustavi bili temeljeni isključivo na Internet topologiji, razvoj Weba omogućio je korištenje istog kao sučelja ka komunikacijskim kanalima. U poglavljima 4.1. te 4.2. opisane su otvorene tehnologije koje su postigle veliku primjenu u domeni komuniciranja u stvarnom vremenu, a pogotovo u IM (engl. *Instant Messaging*) programskim sustavima.

4.1.1. Protokol XMPP

Protokol XMPP (engl. *eXtensible Messaging and Presence Protocol*) predstavlja otvorenu tehnologiju, razvijenu o strane Jabber Software Foundation tvrtke, koja omogućava komunikaciju u stvarnom vremenu između dva ili više računala [17]. Zasnovan na XML tehnologiji, XMPP protokol omogućava proširljivost te razvoj usluga komuniciranja porukama, otkrivanja prisutnosti te zahtjev-odgovor usluga. Arhitektura sustava zasnovanih na XMPP protokolu sastoji se od tri elementa tj. uloge koje računala, koja sudjeluju u komunikaciji, mogu poprimiti:

- klijent: pokušava ostvariti komunikaciju
- poslužitelj: održava te nadzire komunikaciju i prosljeđivanje poruka
- prolaz (engl. *gateway*): služi kao most između različitih mreža npr. Interneta te mobilnih mreža

Na slici 4.1. simbolično je prikazana arhitekura sustava zasnovanih na XMPP protokolu.



Slika 4.1: Arhitektura sustava zasnovanog na XMPP protokolu

Prijenos poruka na mrežnoj razini odvija se TCP protokolom gdje se povezanost uspostavlja na početku sjednice. Jedna TCP veza može upravljati s više sjednica pošto svaka sjednica ima svoju jedinstvenu oznaku. Svaki entitet u sustavu ima vlasitu jedinstvenu oznaku koja predstavlja njegovu adresu te je oblika *korisnik@domena/resurs* [17]. Adrese daju poslužiteljima informacije koje su im dovoljne kako bi poruke usmjerili ka odgovarajućem entitetu. U XMPP sustavima ne postoji mogućnost dinamičkog otkrivanja podataka o usmjeravanju već se konačna povezanost među poslužiteljima ostvaruje pri uspostavljanju sjednice. Osnovni XML elementi kojima se omogućuje komunikacija između entiteta su slijedeći:

- <stream>, </stream>: služe za uspostavljanje sjednice te zatvaranje, respektivno
- <message>: označava poruku koja se prosljeđuje krajnjem entitetu preko niza poslužitelja po principu "spremi i potisni"
- <presence>: označava poruku koja se šalje svim odgovarajućim entitetima, a sadrži informacije o prisutnosti korisnika
- <iq>: predstavlja zahtjev pri modelu zahtjev-odgovor, a može poslužiti za, na primjer, razmjenu datoteka
- <error>: služi za dojavu pogreške u komunikaciji kao što je slanje poruka nakon što je sjednica zatvorena

Među osnovne elemente moguće je ugnjezditi proizvoljne elemente što odgovara spomenutoj činjenici da je protokol proširljiv. Prikaz takve poruke predstavljen je u tablici 4.1.

Tablica 4.1: Primjer XMPP poruke

```
<message from="korisnik1@domena1.com/laptop"
    to="korisnik2@domena2.com">
        <tijelo>
            Ovo je poruka
        </tijelo>
</message>
```

Specifikacija XMPP protokola uključuje i opis sigurnosnih mehanizama za sigurno izvođenje komunikacije. TLS (engl. *Transport Layer Procotol*) [18] protokolom enkriptiraju se XML poruke kako bi se ostvarila obilježja povjerljivosti

i integriteta podataka. SASL (engl. *Simple Authentication and Security Layer protocol* [19] omogućuje autentifikaciju entiteta u komunikaciji, kako pri odnosu klijent-poslužitelj tako i pri odnosu poslužitelj-poslužitelj.

Zbog otvorenosti, jednostavnosti te proširljivosti, XMPP protokol stekao je veliku primjenu kako u posjedničkim rješenjima, tako i u komercijalnim svrhama gdje je najpoznatije ostvarenje sustav *Gtalk* tvrtke Google Inc. Sve većim razvojem sustava u oblacima, pojavila se potreba za razvijanjem posredničkih sustava koji bi omogućili komunikaciju između *oblaka*. Prvotna rješenje, REST te SOA usluge, tijekom vremena postale su nedovoljno dobra rješenja zbog nedostatka komponente komunikacije u stvarnom vremenu. XMPP, sa svim već opisanim osobinama, postaje rješenje koje pronalazi primjenu i pri složenim sustavima, kao što su oblaci.

4.1.2. Protokol SIMPLE

SIMPLE (engl. *SIP for Instant Messaging and Presence Leveraging Extensions*) jest protokol za ostvarivanje komunikacije u stvarnom vremenu koji je razvijen kao nadopuna protokola SIP (engl. *Session Initiation Protocol*) [20]. Omogućuje izmjenu poruka u stvarnom vremenu te otkrivanje prisutnosti drugih entita, kao i protokol XMPP. Dio koji se nadograđuje na protokol SIP omogućuje ostvarivanje pretplata, obavijesti o događajima te objavljivanje sadržaja. Kao i XMPP, SIMPLE protokol jest otvorena tehnologija.

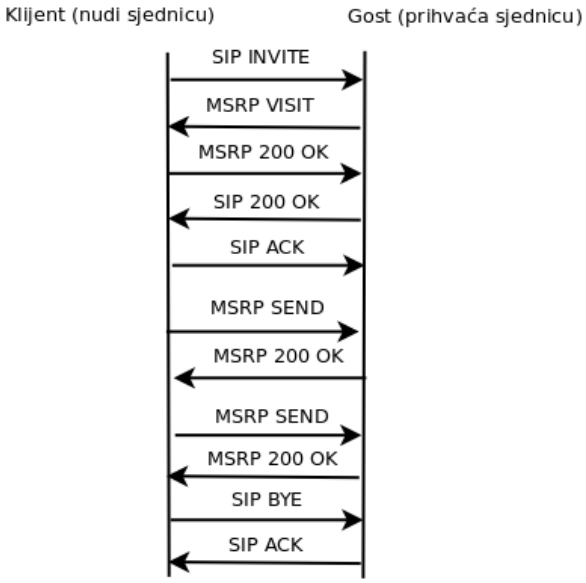
Osnovu rada SIMPLE sustava predstavlja SIP protokol koji je prvotno razvijen za uspostavljanje, nadziranje te zatvaranje multimedijskih sjednica u raspodijeljenim sustavima. Sintaksa protokola slična je široko korištenim protokolima HTTP te SMTP (engl. *Simple Mail Transfer Protocol*) [21] što ga čini proširljivim. Sustavi temeljeni na SIP protokolu sastoje se od entita koji mogu poprimiti pet uloga ostavljajući pojedinom entitetu mogućnost poprimanja više uloga istovremeno:

- korisnički agent klijent: krajni entitet koji postavlja SIP zahtjeve
- korisnički agent poslužitelj: obavlja klijentske zahtjeve te sukladno tome vraća odgovore
- posrednički poslužitelj: ima ulogu klijenta i poslužitelja te sukladno tome obavlja radnje u ime klijenta proslijeđujući zahtjeve drugim poslužiteljima
- preusmjeritelj: prihvaća SIP zahtjev na temelju kojeg vraća odgovarajuću adresu klijentu, ne prosljeđujući zahtjev drugim poslužiteljima

– registrar: poslužitelj koji služi kao baza podataka položaja SIP entiteta

Svaka SIP poruka može predstavljati zahtjev ili odgovor koji se mogu dinamički usmjeravati koristeći usluge preusmjeritelja te registrara. Kao nadogradnja na SIP, protokol SIMPLE omogućuje otkrivanje prisutnosti entiteta uvođenjem dodatnih metoda zahtjeva te uvodeći uslugu otkrivanja prisutnosti. Svaki entitet posjeduje vlastitu uslugu otkrivanja prisutnosti koja može biti ostvarena kao poslužitelj ili korisnički agent. Svaki entitet ima mogućnost pretplaćivanja na događaje drugog entiteta slanjem SUBSCRIBE zahtjeva pripadnoj usluzi otkrivanja prisutnosti. U zahtjevu se mogu specificirati parametri pretplate kako bi se dobila veća kontrola nad događajima entiteta kojeg se prati. Na ovaj način omogućuje se otkrivanje prisutnosti entita, kao pretpostavljena usluga, te dodatno funkcionalnost objave sadržaja, čime se povećava razina funkcionalosti cjelokupnog sustava.

Komunikacija porukama u stvarnom vremenu odvija se koristeći protokol MSRP (engl. *Message Session Relay Protocol*) [20] te uvođenjem novih metoda pri zahtjevima. Komunikacija se može odvijati kroz sjednice te neovisno o sjednici. Slanjem poruka pomoću zahtjeva s metodom SEND, odgovarajućem se entitetu šalje poruka bez uspostavljanja sjednice. Klijent upostavlja sjednicu koristeći zahtjev sa SIP metodom INVITE na što klijent, kojemu je poziv poslan, odgovara s VISIT zahtjevom. Na taj način uspostavlja se MSRP sjednica, koristeći protokol TCP na mrežnoj razini, gdje se poruke šalju koristeći spomenute SEND zahtjeve. Primjer komunikacije preko sjednica u SIMPLE sustavima prikazan je na slici 4.2.



Slika 4.2: Primjer[1] izmjene poruka u stvarnom vremenu pomoću SIMPLE protokola

Specifikacija SIMPLE protokola, za razliku od XMPP protokola, ne određuje mehanizme za izvođenje sigurne komunikacije već ostavlja mogućnost razvijanja posjedničkih rješenja. Složenost predstavlja nedostatak sustava zasnovanih na SIMPLE protokolu te upravo predstavlja prednost sustava XMPP protokola.

4.2. Protokoli tijeka

4.2.1. Medijske usluge

Razvojem Internet i World Wide Web tehnologija te povećanjem procesorske moći računala postavljeni su temelji za reprodukciju audio i video zapisa uživo. Korisnicima, koji nisu imali pristup standardnim radio-televizijskim sadržajima, postale su dostupne usluge koje su omogućile praćenje spomenutih sadržaja preko Weba. Iako su spomenute usluge zahtijevale veliku računalnu moć i propusnost mreže, njihova pojava proširila se i na poslovni svijet. Održavanje sastanaka preko video konferencija uživo pripomoglo je suradnji tvrtki iz različitih dijelova svijeta. Korisnici preko svojih primjenskih programa sudjeluju u komunikaciji na način da se pretplaćuju na podatke te primaju, na načelu potiskivanja, i reproduciraju iste dobivene od poslužitelja sadržaja. Razvoj protokola tijeka omogućio je razvoj opisanih usluga [22].

Jedan od široko korištenih protokola za ostvarivanje prijenosa podataka u stvarnom vremenu jest Real Time Streaming Protocol (RTSP) [23]. Protokol služi za uspostavu te upravljanje tijekovima podataka koji moraju biti isporučeni u stvarnom vremenu kao što su audio te video podaci. Za sami prijenos podataka zaslužan je Real Time Transport Protocol protokol (RTTP) [23] dok je zadaća RTSP protokola upravljanje sjednicama tijekova podataka. RTSP ostvaruje funkcionalnosti nad video i audio zapisima koje su korisnicima dostupne preko audio i video uređaja poput DVD reproducivača. Pružateljima usluga omogućuje se odabir između korištenja TCP i User Datagram Protocol (UDP) [24] protokola za prijenos podataka na nižim razinama. Iako najčešće radi u suradnji s RTTP protokolom, RTSP protokol je zaseban protokol te neki pružatelji usluga koriste vlastite protokole u zamjenu za RTTP.

Drugi često korišteni protokol jest HTTP Streaming protokol koji koristi HTTP protokol kao temelj prijenosa tijeka podataka [22]. Klijent započinje ko-

[1]SIP INVITE metoda služi za uspostavljanje sjednice, ACK za potvrdu prethodnog zahtjeva dok BYE metoda služi za zatvaranje sjednice

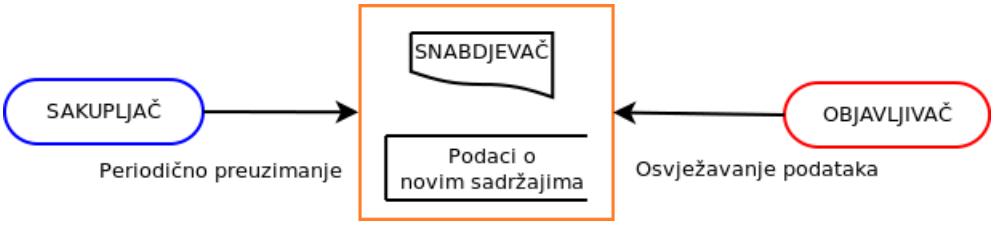
munikaciju sa poslužiteljem što predstavlja dio komunikacije na načelu povlačenja podataka. Za razliku od HTTP protokola, HTTP Streaming protokol održava stalnu vezu s klijentom šaljući mu podatke bez potrebe slanja zahtjeva. Ovaj dio suradnje klijenta i poslužitelja predstavlja komunikaciju na načelu potiskivanja podataka. Rad HTTP Streaming protokola predstavlja Comet model rada, koji je opisan detaljnije u poglavlju 4.5., gdje, kao što je već opisano, poslužitelj šalje klijentu podatke bez prethodnog slanja zahtjeva od strane klijenta.

Iako veliki broj sustava koristi opisana dva protokola za pružanje usluga u stvarnom vremenu, ostvareni su i drugi protokoli koji omogućuju izgradnju opisanih usluga. Neki od tih protokola su Microsoft Media Services (MMS) i Real Time Messaging Protocol (RTMP).

4.2.2. RSS

Jedna od prvih tehnologija koja je omogućila pružanje usluga slijedeći koncepte Real-time Weba jest tehnologija RSS (engl. *Really Simple Syndication*) [25]. RSS je protokol koji omogućuje ostvarenje objavi-pretplati sustava te je razvijen za olakšano praćenje informacija koje se učestalo mijenjaju tj. osvježavaju. Korisnik je lišen rada učestalog provjeravanja dostupnosti novih sadržaja koji se pak delegira programskom sustavu sakupljača informacija (engl. *aggregator*). Sustavi koji pružaju RSS usluge inačica su objavi-pretplati modela sustava gledajući iz perspektive korisnika. Korisnik se prijavljuje na praćenje određenog kanala sadržaja te mu se informacije potiskivaju od strane objavljivača informacija.

Gledajući unutrašnju strukturu sustava, RSS model nije inačica modela objavi-pretplati. Program sakupljač periodično provjerava dostupnost novih sadržaja određenog kanala što predstavlja oblik komunikacije na načelu povlačenja podataka. Također, sustav usluga RSS sustava ima tek podskup odgovornosti sustava usluga klasičnog modela objavi-pretplati [3]. Korisnici se pretplaćuju na određeni kanal sadržaja na način da programu sakupljaču predaju URL adresu spisa zvanog snabdjevač (engl. *feed*) [25]. Objavljivač pri objavi novih informacija osvježava snabdjevača s podacima o novim sadržajima. U RSS sustavima poslužitelj snabdjevača ima ulogu sustava usluga koji ne posjeduje odgovornosti filtriranja i slanja informacija već služi kao pasivni izvor podataka o objavi novih informacija. Ovakav rad poslužitelja snabdjevača predstavlja ograničenje RSS sustava pošto program sakupljač mora preuzeti sadržaj spisa te analizirati ga s ciljem provjere dostupnosti novih sadržaja. Model rada RSS sustava prikazan je na slici 4.3.



Slika 4.3: Prikaz rada RSS sustava

Preuzimanje sadržaja snabdjevača od strane sakupljača ostvaruje se HTTP protokolom. Način osvježavanja sadržaja snabdjevača od strane objavljivača informacija je posjednički određen dok su najpopularniji protokoli MetaWeblog i Blogger [26]. Stvaranje sadržaja dandanas je moguće ostvariti raznim alatima koji automatski stvaraju sadržaj na temelju HTML zapisa objavljenih sadržaja. *S*adržaj snabdjevača jest spis zapisan u XML obliku [14], koji je već nekoliko puta spomenut tijekom rada, a ovdje detaljnije objašnjen. XML je standardizirani jezik za označavanje podataka koji omogućuje određivanje strukture dokumenata te jednostavnu obradu istih od strane računala. Osnovna jedinica XML dokumenata jest element koji se sastoji od oznake te korisnog sadržaja. Imena elemenata te njihova, neobavezna, svojstva nalaze se između znakova „<“ i „>“ te čine spomenutu oznaku elementa. Između oznaka nalaze se vrijednosti elemenata tj. korisni sadržaj koje opet može biti skup XML elemenata.

XML zapis spisa snabdjevača sadrži podatke o kanalu sadržaja koji se prati te podatke o novim informacijama u kanalu. Osnovni te obavezni elementi opisa kanala, koji se nalaze u elementu *channel*, jesu ime (engl. *title*), poveznica na web stranicu kanala (engl. *link*) te opis (engl. *description*). Također, moguće je zapisati elemente kao što su jezik, vrijeme nastanka kanala, vrijeme zadnje objavljenje informacije, ime urednika i dr.

Svi elementi koji sadrže podatke o novo objavljenim sadržajima su neobavezni te se nalaze u elementu s oznakom „<item>“. Najkorišteniji elementi su ime, poveznica na web stranicu objavljenog sadržaja te opis. Drugi, manje korišteni, elementi su autor sadržaja, vrijeme objavljivanja, jedinstvena oznaka, komentari itd. Primjer mogućeg sadržaja spisa RSS snabdjevača prikazan je u tablici 4.2.

Tablica 4.2: Ilustrativni primjer RSS spisa snabdjevača

```
<?xml version="1.0" encoding="UTF-8" ?>
<rss version="2.0">
  <channel>
    <title>Primjer kanala</title>
    <description>Primjer snabdjevača</description>
    <link>http://www.fer.hr/diprad</link>
    <item>
      <title>Podaci o novom sadržaju</title>
      <description>Primjer podataka o novom sadržaju</description>
      <link>http://www.fer.hr/predmet/diprad?@=1zdji</link>
    </item>
  </channel>
</rss>
```

Iako nisu dosljedna inačica sustava modela objavi-pretplati, RSS sustavi su postigli veliku primjenu od strane korisnika zbog olakšanog korištenja. Korisnici bi se samo jednim pritiskom gumba miša na određenim web stranicama pretplatili na kanal sadržaja. Štoviše, napredna ostvarenja programa sakupljača pronalaze adrese snabdjevača iz sadržaja web stranice što lišava korisnika pretrage za adresom. Također, pojedini sustavi omogućuju korištenje RSS usluga za upravljanje elektroničkom poštom koja je jedna od prvih usluga temeljena na komunikaciji na načelu potiskivanja. Iako jednostavan za korištenje, ovaj model rada ima nedostatak redundantnog rada programa sakupljača i poslužitelja snabdjevača što rezultira povećanjem mrežnog prometa. Primjer RSS sakupljača koji je postigao veliku primjenu jest sustav *Google Reader* razvijen od strane tvrtke Google Inc.

4.2.3. Atom

Iako najkorišteniji, RSS protokol imao je nekolicinu nedostataka u strukturi XML zapisa snabdjevača. Kako bi se računalima omogućila lakša obrada sadržaja snabdjevača te povećao skup mogućnosti RSS sustava, razvijen je Atom (engl. *Atom Syndication Protocol*) protokol [27]. Model rada RSS i Atom sustava je istovjetan, no struktura Atom snabdjevača je detaljnija i stroža.

Prva velika razlika između RSS i Atom zapisa snabdjevača jest vrsta zapisa sadržaja koji se nalazi u elementima opisa. Dok RSS dopušta obični tekst te

preuređeni HTML (engl. *escaped HTML*) [11], bez informacija o kojoj se vrsti zapisa radi, Atom zapisi snabdjevača dopuštaju i korištenje XML zapisa radi mogućnosti ponovnog korištenja. Također, mora se odrediti o kojoj se vrsti zapisa sadržaja kako bi se olakšala obrada računalom. Sadržaj elemenata opisa može biti doslovno opis novo objavljenog sadržaja kao i potpuni sadržaj. RSS zapis ne koristi posebne oznake za označavanje opisanih obilježja dok su kod Atom zapisa one nužne. Svi ključni elementi poput imena, opisa te vremena osvježavanja snabdjevača su kod Atom zapisa nužni, dok su kod RSS zapisa, kao što je već spomenuto, na nekim mjestima neobavezni.

XML zapis RSS snabdjevača nije sadržan u određenom XML prostoru imena iako koristi neke od elemenata iz drugih prostora imena. Ovo obilježje otežava razvijateljima RSS sustava snalaženje u RSS zapisima snabdjevača dok to nije slučaj kod Atom zapisa. Zapisi Atom snabdjevača obavezno posjeduju informaciju o mjestu XML prostora imena dotičnog zapisa.

Opisane razlike predstavljaju prednosti Atom zapisa snabdjevača nad RSS zapisima. Također, postoje i temeljne razlike koje ne utječu na funkcionalnost protokola. Protokoli se razlikuju i po ključnim riječima elemenata kao i po načinu osvježavanja snabdjevača. Na primjer, ključnu riječ *channel*, koja kod RSS zapisa određuje kanal koji se prati, u Atom zapisima zamjenjuje riječ *feed*, itd. Popis ostalih razlika u ključnim riječima nije tema ovoga rada stoga u nastavku neće biti detaljno razrađen. Za osvježavanje sadržaja snabdjevača koristi se Atom Publishing Protocol protokol kao i slična posjedničke rješenja [27]. Primjer mogućeg sadržaja spisa Atom snabdjevača prikazan je u tablici 4.3.

Tablica 4.3: Ilustrativni primjer Atom spisa snabdjevača

```
<?xml version="1.0" encoding="UTF-8" ?>
<feed xmlns="http://www.fer.hr/2010/Atom">
   <title>Primjer kanala</title>
   <link href="http://www.fer.hr/"/>
   <updated>2010-11-8T16:30:02Z</updated>
   <entry>
      <title> Podaci o novom sadržaju </title>
      <link href="http://www.fer.hr/predmet/diprad?@=1zdji"/>
      <updated>2011-05-8T18:30:02Z </updated>
   </entry>
</feed>
```

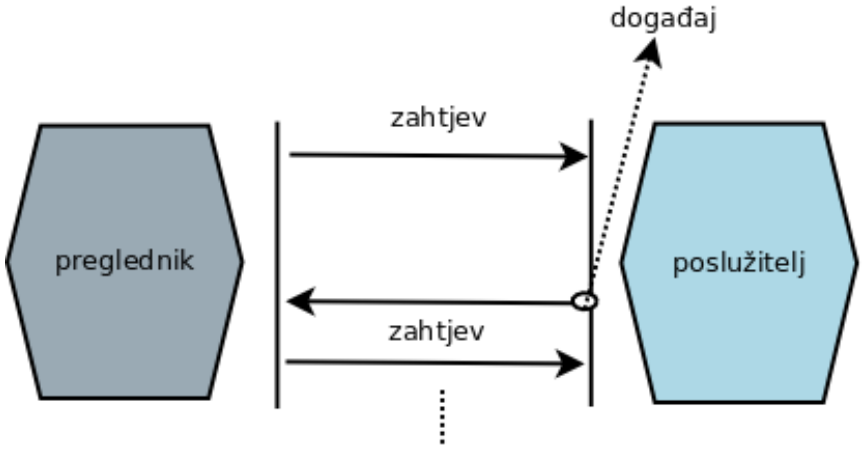
Uvođenje Atom protokola predstavilo je poboljšanje RSS protokola te omogućilo lakšu izgradnju sustava koji rade po RSS modelu. Unatoč činjenici da su protokoli često korišteni, obje vrste sustava imaju nedostatak redundantnog rada sakupljača i poslužitelja snabdjevača. Rješenje spomenutih nedostataka ostvareno je razvojem sustava PubSubHubbub te RSSCloud opisanim u poglavlju 5.

4.3. Comet tehnologija

Razvoj RSS i Atom usluga omogućio je korisnicima prikupljanje zanimljivih im informacija u prividno stvarnom vremenu. Svi korisnikovi podaci spremaju se u baze podataka na poslužiteljima pripadnih sustava prikupljača. Ovakav oblik komunikacije predstavlja odnos poslužitelj-poslužitelj. Razvojem Web tehnologija, naročito onih koji su utjecali na web preglednik, postavljeni su temelji za razvoj komunikacije u stvarnom vremenu pri odnosu preglednik-poslužitelj. Prepreka koja se pokušava svladati jest komunikacija preglednika i poslužitelja koja se odvija na principu zahtjev-odgovor koja onemogućuje periodično potiskivanje podaka od strane poslužitelja ka pregledniku. Zajednički naziv za tehnologije koje omogućuju izvođenje opisane komunikacije jest Comet [4]. Spomenute tehnologije opisane se u slijedećim poglavljima.

4.3.1. Tehnika dugog povlačenja

Tehnika dugog povlačenja (engl. *long polling* koristi AJAX [4] tehnologiju za ostvarenje Comet funkcionalnosti. Komunikacija se izvodi na istom principu kao i pri običnoj AJAX komunikaciji s poslužiteljom s iznimkom u vremenu odgovora poslužitelja. Poslužitelj odgovara tek nakon što ima potrebne podatke, na primjer nakon odgovarajućeg događaja. Nakon primitka potpunog odgovora klijent može poslati novi zahtjev. Primjer opisane komunikacije prikazan je na slici 4.4.



Slika 4.4: Prikaz izvođenja komunikacije preglednika i poslužitelja ostvarene tehnikom dugog povlačenja

Opisani pristup predstavlja zaobilaženje problema te na predstavlja doslovnu komunikaciju na načelu postiskivanja.

4.3.2. Tehnika skrivenog ugnježđenog HTML okvira

Tehnika skrivenog ugnježđenog HTML okvira ne koristi AJAX tehnologiju za ostvarivanje komunikacije u stvarnom vremenu već iskorištava pogodnosti HTTP protokola, točnije inačice 1.1. Spomenuta inačica, koja se trenutno koristi pri svim web preglednicima, omogućava vraćanje odgovara bez poznavanje njegove konačne duljine. Na taj način se pregledniku vraća odgovor u dijelovima.

Potiskivanje podataka ka pregledniku od strane poslužitelja omogućeno je kroz skriveni ugnježđeni okvir (engl. *hidden iframe*)koji se upisuje u korijenski HTML odgovor. Ugnježđeni okviri omogućuju ugradnju drugih HTML spisa u korijenski spis na način da se pregledniku preda URI adresa izvora HTML spisa. Pri prevođenju korijenskog HTML spisa dohvaća se HTML spis ugnježđenog okvira. Odgovor koji se vraća jest raskomadan u dijelove na način opisan

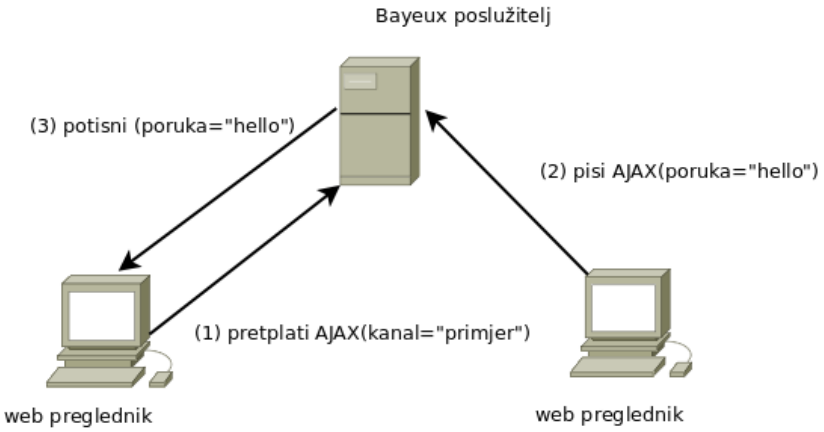
u prethodnom odlomku. Odgovor jest JavaScript [28] kod kojeg preglednik pri prevođenju odgovora odmah izvršava. Ovakav način komuniciranja predstavlja potiskivanje podataka ka pregledniku.

Spomenuta tehnika slična je HTTP Streaming pristupu opisanom u poglavlju 4.2. HTTP Streaming metodom zadržava se veza s klijentom te se pomoću posebnih tehnika pri odgovarajućim događajima šalju podatci paralelno ka više klijenata. Moderni web preglednici, poput sustava Firefox tvrtke Mozilla, omogućuju primanje rascjepkanih dijelova odgovora AJAX zahtjeva.

4.3.3. Protokol Bayeux

Održavanje maksimalno dvije istovremene HTTP veze od strane web preglednika [4] s bilo kojim poslužiteljem predstavlja ograničenje na prethodno opisane COMET tehnike u slučaju da se pokušava komunicirati s višestrukim izvorima podataka. Kako bi se omogućilo komuniciranje s višestrukim izvorima podataka, razvijen je Bayeux [29], protokol više razine apstrakcije koji omogućava dvosmjernu komunikaciju između web klijenta i poslužitelja te koji nije ograničen na samo web preglednike. Spomenuti protokol omogućuje komunikaciju pri odnosima klijent-poslužitelj, poslužitelj-klijent te klijent-klijent (preko poslužitelja).

Rad sustava Bayeux protokola zasniva se na objavi-pretplati modelu gdje se komunikacija objave i pretplate vrši preko proizvoljnih protokola modela klijent-poslužitelj. Pri sustavima s web preglednicima, prema Bayeux poslužitelju se otvaraju dvije (maksimalno dopuštene) TCP veze koje služe za dvosmjernu komunikaciju. Web preglednici se pretplaćuju na određeni kanal slanjem AJAX zahtjeva ka Bayeux poslužitelju u kojem određuju ime kanala. U slučaju događaja, koji je određen kao pisanje u kanal, poslužitelj potiskuje sadržaj pisanja ka svim pretplaćenim klijentima. Komunikacija preko imenovanih kanala omogućuje komuniciranje s više izvora informacije preko jedne HTTP veze. Potiskivanje podataka omogućeno je odabirom jedne od prethodno opisanih COMET tehnika. Simbolični rad sustava pri komunikaciji preglednik-preglednik prikazan je na slici 4.5.



Slika 4.5: Simbolični prikaz rada Bayeux protokola na primjeru komunikacije dva web preglednika

Poruke koje se izmjenjuju između entiteta u sustavu zapisane su u JSON obliku, koji je detaljno objašnjen u poglavlju 6.

4.3.4. Web priključnice

Nedostatak prethodno opisanih tehnika i tehnologija jest povećanje složenosti sustava što rezultira u smanjenju sposobnosti razmjernog rasta. Uvođenje dodatnih poslužitelja te održavanja dvije TCP veze predstavlja motivaciju za razvoj protokola WebSocket [30] (hrv. *web priključnice*). Protokol, razvijen pod inicijativom HTML5 [31] standarda, omogućuje dvosmjernu komunikaciju preko jedne TCP veze koristeći sposobnost HTTP protokola na nadogradnju pomoću *Upgrade* (hrv. nadogradnja) [10] zaglavlja.

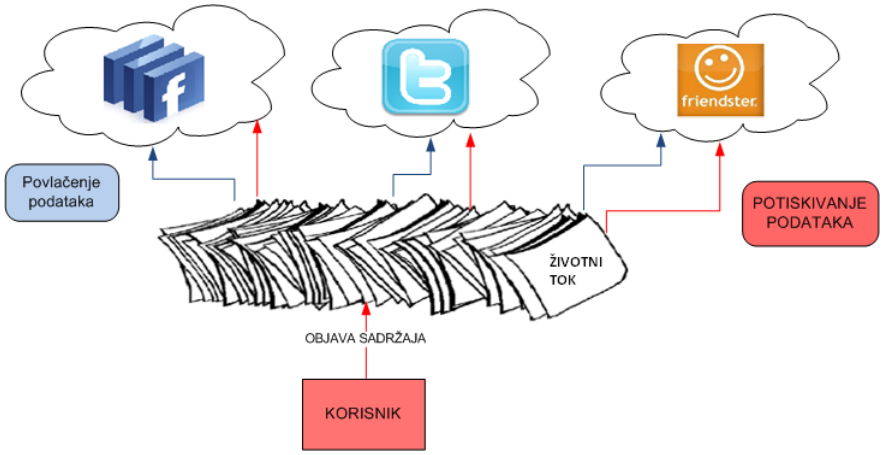
Klijent započinje komunikaciju šaljući HTTP GET zahtjev postavljajući *Upgrade* zaglavlje na *WebSocket*. Poslužitelj, u slučaju da prihvati zahtjev za nadogradnju, prelazi u WebSocket način rada gdje se između njega i klijenta poruke mogu slati u oba smjera te se iste obrađuju sukladno WebSocket protokolu. Prednost WebSocket protokola predstavlja smanjenje složenosti sustava preko jedne TCP veze te korištenje mrežnih vrata 80 i 443, koja služa za obradu HTTP i HTTPS protokola, respektivno. WebSocket protokol nije tema ovoga rada te sukladno tomu nije detaljnije objašnjen. Zainteresirani dodatne informacije mogu pronaći slijedeći popis literature.

4.4. Primjena na društvenim mrežama

Razvoj tehnologija kao što su RSS i Atom potaknulo je mnoge pružatelje Web usluga na razvoj sustava koji će omogućiti jednostavno povezivanje ljudi. Tehnologijama opisanim u prethodnim poglavljima korisnicima je omogućeno jednostavnije praćenje omiljenih sadržaja te izravno komuniciranje u stvarnom vremenu. Slijedeći korak u razvoju krajnjem korisniku usmjerenih i prilagođenih usluga predstavljaju društvene mreže.

Sustavi koji pružaju usluge društvenog umrežavanja omogućuju pojedincu komunikaciju sa drugim korisnicima te praćenje rada istih. Korisnici se pretplaćuju na praćenje rada drugih korisnika čime im je omogućeno dobivanje obavijesti sa sadržajima koje te isti korisnici objavljuju. Ovakav način rada opisanih sustava predstavlja model komunikacije na načelu potiskivanja podataka dok je model rada sustava, iz perspektive korisnika, istovjetan modelu rada RSS sustava. Neke od poznatih tvrtki koje pružaju opisanu vrstu usluge su Facebook, Twitter, Friendster i dr.

Kako bi se korisnicima olakšala pretraga i korištenje spremljenih događaja tj. spisa razvijena je programska arhitektura životnog tijeka (engl. *lifestream*) [32]. Životni tijek je programski sustav koji omogućuje korištenje niza aktivnosti s ciljem jednostavnog i brzog pronalaska informacija. Osnovnu radnju predstavlja skupljanje niza spisa poredanih u ovisnosti o vremenu u dnevnik. Životni tijek se stvara pomoću sakupljača tijeka koji skuplja podatke tj. spise s raznih društvenih mreža na načelu učestalog povlačenja podataka. Prednost korištenja sustava životnog tijeka jest mogućnost objavljivanja sadržaja u samom sustavu životnog tijeka te raspodjela sadržaja društvenim mrežama i krajnjem korisniku po principu potiskivanja podataka. Na slici 4.5. prikazan je simbolični model rada sustava životnog tijeka.



Slika 4.6: Simbolični rad sustava životnog tijeka

Poznatiji sustavi koji pružaju usluge životnog tijeka jesu Collectedin, Sweetcron, Lifestream [32] i dr. Društvene mreže, također, počinju slijediti trend potiskivanja podataka na Webu te pružaju programske usluge i sučelja koja omogućuju objavu sadržaja u stvarnom vremenu na načelu potiskivanja podataka. Društvene mreže Facebook i MySpace pružaju usluge slične principu PubSubHubbub sustava koji je detaljno objašnjen u poglavlju 5. Slijedeća generacija sustava životnog tijeka koristiti će spomenute usluge društvenih mreža u stvarnom vremenu kako bi rasteretile svoje poslužitelje od redundantog rada.

5. PubSubHubbub protokol

Kako je opisano u prethodnim poglavljima, sustavi RSS i Atom usluga posjeduju nedostatke redundantnog rada sakupljača i poslužitelja snabdjevač. Kako bi se uklonili spomenuti nedostatci, razvijen je protokol PubSubHubbub [4, 33] koji se zasniva na protokolu HTTP. Protokol je razvijen kao nadogradnja RSS i Atom sustava preuređenjem podjela uloga i odgovornosti entiteta te uvođenjem novog entiteta. Kako bi se sakupljača i objavljivača informacija lišilo nepotrebnog i beskorisnog rada, dio njihova rada delegiran je na novi entitet- središte (engl. *hub*) [33]. U nastavku rada na nekim mjestima će se umjesto naziva PubSubHubbub koristiti skraćenica PSHB.

Sustavi temeljeni na PSHB protokolu sastoje se od slijedećih entiteta: sakupljača (pretplatnika), objavljivača, snabdjevača i središta. Osnovna prednost PSHB sustava je potiskivanje podataka od strane objavljivača bez prethodnog slanja zahtjeva od strane sakupljača. Svaki objavljivač, koji pruža usluge temeljene na PSHB protokolu, logički je povezan sa određenim središtem kojeg obavještava o novo objavljenim sadržajima. Uloga središta jest zaprimanje pretplata od strane korisnika, preuzimanje sadržaja od snabdjevača te potiskivanje sadržaja prema korisnicima.

5.1. Ostvarivanje pretplate

Rad PSHB sustava započinje zahtjevom korisnika, preko programa sakupljača, za pretplaćivanjem na određenu temu objavljivača. Korisnik sakupljaču predaje URL adresu snabdjevača željene teme na kojemu se nalaze podaci o novo objavljenim sadržajima te podaci o središtu objavljivača. Sakupljač dohvaća spis snabdjevača te obradom pronalazi mjesto središta. Ovaj dio rada PSHB sustava nije dio specifikacije protokola, no mnogi ga sustavi ostvaruju kako bi povisili razinu usmjerenosti ka korisniku. Kako bi sakupljači otkrili mjesto središta, XML spis snabdjevača nadopunjuje se elementom koji to jednoznačno određuje. Dodaje se

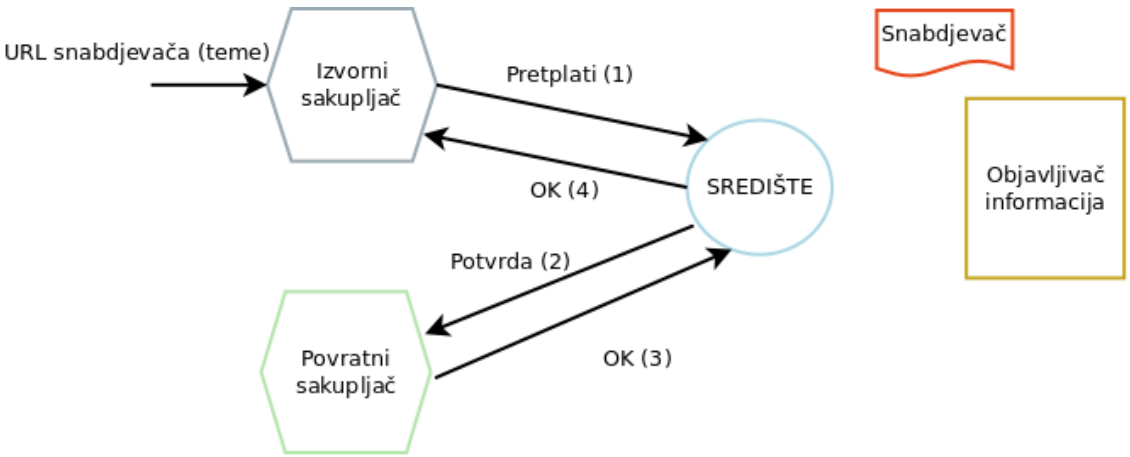
element s oznakom poveznice (engl. *link*) koji kao vrijednost atributa *rel* posjeduje niz znakova *hub* (hrv. središte). Primjer opisanog elementa prikazan je u tablici 5.1.

Tablica 5.1: Primjer elementa poveznice na javno dostupno središte

<link rel="hub" href="http://pubsubhubbub.appspot.com/subscribe"/>

Središtu se zatim šalje zahtjev za određenom temom. Veliku prednost PSHB sustava nad konkurentnim sustavima jest određivanje javne adrese povratnog sakupljača (engl. *callback*) [33] koji je zadužen za primanje sadržaja čime se ostvaruje fleksibilnost sustava. Potvrđivanje se obavlja na strani povratnog sakupljača te konačno na strani sakupljača koji je započeo proces pretplaćivanja čime isti završava.

Izvođenje potvrđivanja pretplate moguće je ostvariti sinkrono ili asinkrono. Pri sinkronom potvrđivanju, potvrđivanje od strane izvornog sakupljača omogućeno je nakon završetka potvrđivanja povratnog sakupljača. Sinkrono potvrđivanje koristi se u slučaju da postoji izvjesna mogućnost da potvrđivanje od strane povratnog sakupljača ne bude uspješno. Moguće je da se povratni sakupljač nalazi na računalu koje je udaljeno na sasvim drugom kontinentu ili ima nisku razinu dostupnosti. Na slici 5.1. prikazan je model sinkronog pretplaćivanja.



Slika 5.1: Prikaz toka izvođenja procesa sinkronog pretplaćivanja pri PSHB sustavu

U slučaju asinkronog potvrđivanja, izvorni i povratni sakupljač istovremeno izvode postupke potvrđivanja. Pri opisanom potvrđivanju postoji nedeterminizam u poretku izvršavanja potvrđivanja te se on koristi u slučajevima kada postoji

mala vjerojatnost neuspješnog potvrđivanja od strane oba sakupljača.

U tablici 5.2. navedeni su HTTP parametri zahtjeva pretplate, koji se središtu od strane sakupljača, šalju HTTP POST zahtjevom, te pripadni opisi. Obavezni parametri odgovaraju već opisanom načinu rada PSHB sustava gdje se određuju adresa povratnog sakupljača, vrsta pretplate, sinkronost provjere pretplate te adresa spisa snabdjevača koji odgovaraju recima tablice 5.1. *hub.callback*, *hub.mode*, *hub.verify* te *hub.topic*, respektivno [33].

Tablica 5.2: Parametri pretplate pri PSHB sustavu

PARAMETAR	**OPIS**	**OBAVEZAN**
hub.callback	URL adresa povratnog poslužitelja	+
hub.mode	vrijednost "subscribe" (hrv. *preptlati*) ili "unsubscribe" (hrv. *odjavi*)	+
hub.verify	vrijednost "sync" (hrv. *sinkrono*) ili "async" (hrv. *asinkrono*)	+
hub.topic	URL adresa spisa snabdjevača na koji se pretplaćuje te koji predstavlja temu	+
hub.lease_seconds	trajanje preplate u sekundama koji se pri izostavljanju smatra kao trajna pretplata	-
hub.verify_token	oznaka koja se koristi za obavljanje autentificirane pretplate	-
hub.secret	oznaka koja se koristi za obavljenje autentificirane raspodjele objavljenog sadržaja	-

Pri primitku zahtjeva za pretplatom, središte šalje HTTP GET zahtjev poslužitelju povratnog sakupljača gdje kao parametre šalje podatke prikazane u tablici 5.3.

Tablica 5.3: Parametri provjere pretplate pri PSHB sustavu

PARAMETAR	OPIS	OBAVEZAN
hub.challenge	sigurnosna oznaka koja omogućuje autentificiranje povratnog sakupljača	+
hub.mode	odgovara *hub.mode* parametru pri zahtjevu za pretplatom	+
hub.topic	odgovara *hub.topic* parametru pri zahtjevu za pretplatom	+
hub.lease_seconds	broj sekundi nakon čijeg će isteka središte poslati zahtjev za ponovno potvrđivanje	+
hub.verify_token	odgovara *hub.verify_token* parametru pri zahtjevu za pretplatom	-

Na temelju parametra *hub.verify_token* povratni sakupljač može ustvrditi o kojem se izvornom poslužitelju radi kako bi vratio odgovor sukladno tomu "poznaje" li izvornog poslužitelja. Osim parametra *hub.challenge* [33], koji mora biti vraćen središtu, ostale parametre povratni sakupljač koristi za osvježavanje baze podataka o pretplatama čiji mu se sadržaj potiskuje. U slučaju da povratni sakupljač nije dostupan, središte mora slati dodatne zahtjeve smanjujući frekvenciju slanja pri svakom negativnom odgovoru. Politika kraja provjere pretplate u slučaju negativnih odgovora nije određena pri specifikaciji protokola, već ovisi o posjedničkom ostvarenju. Izvornom poslužitelju se, pri prvom negativnom odgovoru šalje odgovor u kojem se, pomoću HTTP status kodova, određuje da je zahtjev u procesu provjere.

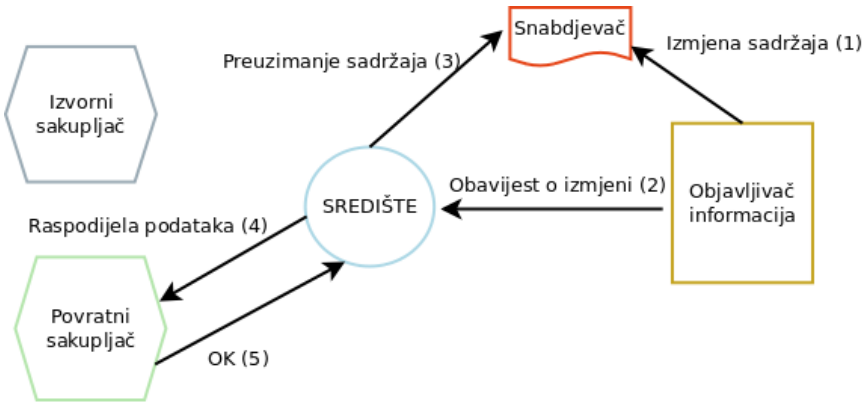
Središte prije isteka trajanja pretplate, ako je isto određeno, ponavlja postupak potvrđivanja iste teme kako bi potvrdilo pretplatnikovu (ne)zainteresiranost za temu. Ovaj postupak je posljedica procesa kojim središte pokušava lišiti korisnika ponovnog pretplaćivanja u slučaju da je korisnik i dalje zainteresiran za temu. Središte može slati zahtjeve za ponovnom pretplatom po vlastitoj politici kako bi oslobodilo resurse u slučaju da neki pretplatnici nisu zainteresirani za pretplatu iako im ista nije istekla. Podatak o vremenu takve pretplate daje se povratnom poslužitelju vrijednošću parametra *hub.lease_seconds*.

5.1.1. Ukidanje pretplate

Ukidanje pretplate ostvaruje se na identičan način kao i samo pretplaćivanje. Sakupljač šalje zahtjev za ukidanjem pretplate središtu koje potvrđuju ukidanje pretplate s povratnim poslužiteljem te izvornim poslužiteljem sinkrono ili asinkrono, ovisno o vrijednostima parametara zahtjeva. Jedina razlika je u parametru *hub.mode* koji se postavlja na vrijednost "unsubscribe".

5.2. Objava sadržaja

Objavljivanje sadržaja započinje objavljivač sadržaja pri objavi novih sadržaja. Nakon što osvježi spis snabdjevača, objavljivač dojavljuje središtu da je dostupan novi sadržaj predajući mu URL adresu snabdjevača čiji je sadržaj izmijenjen. Središte zatim preuzima sadržaj spisa snabdjevača te obradom pronalazi podatke o novim sadržajima. Slijedeći korak predstavlja višeodredišno potiskivanje spomenutih podataka. Odredište šalje odjednom podatke svim pretplatnicima na istu temu. Nakon što prime podatke, sakupljači odgovaraju središtu o uspješnosti primanja podataka. Model procesa objavljivanja sadržaja prikazan je na slici 5.2.



Slika 5.2: Prikaz toka izvođenja procesa raspodjele sadržaja pri PSHB sustavu

Obavijest o dostupnosti novih sadržaja ostvaruje se slanjem HTTP POST zahtjeva središtu, od strane objavljivača informacija, s parametrima navedenim u tablici 5.4.

Tablica 5.4: Parametri provjere pretplate pri PSHB sustavu

PARAMETAR	OPIS	OBAVEZAN
hub.topic	URL adresa snabdjevača čiji je sadržaj izmijenjen	+
hub.mode	niz znakova *publish*	+

Središte HTTP GET zahtjevom dohvaća sadržaj spisa snabdjevača. Središte može u listu zaglavlja zahtjeva staviti zaglavlje *User-Agent* te ga popuniti ukupnim brojem pretplatnika na pripadnu temu. Također, središte mora slijediti HTTP preusmjeravanja koja mogu biti nametnuta od strane poslužitelja snabdjevača, kako bi se došlo do sadržaja.

Raspodjela sadržaja povratnim sakupljačima obavlja se istovremenim slanjem HTTP POST zahtjeva koji kao tijelo zahtjeva ima dohvaćeni sadržaj. Ako je pri pretplati određen parametar *hub.secret*, kao u tablici 5.2., tada se obavlja autentificirano slanje sadržaja. U listu zaglavlja se dodaje zaglavlje *X-Hub-Signature* [33] koje kao vrijednost ima izlaz SHA1 [34] metode sažimanja koja kao ključ sažimanja prima vrijednost parametra *hub.secret*. Povratni sakupljač, koji, također, posjeduju vrijednost *hub.secret* obavlja istu metodu te uspoređuje izlaz s vrijednošću iz zahtjeva. U slučaju pozitivne usporedbe, sakupljač zna da je primio sadržaj od valjanog središta[1].

Također, PSHB protokol ostvaruje funkcionalnost optimalnog slanja podataka pri Atom [27] sustavima. Središta mogu sakupljati podatke o novim sadržajima za određenog sakupljača te ih odjednom sve poslati umjesto pojedinačnog slanja. Ostvarena funkcionalnost predstavlja sposobnost na razmjeran rast jer se smanjuje ukupna komunikacija između odredišta i sakupljača pošto središte šalje samo jedan zahtjev te sakupljač vraća samo jedan odgovor.

5.2.1. Dodatne mogućnosti

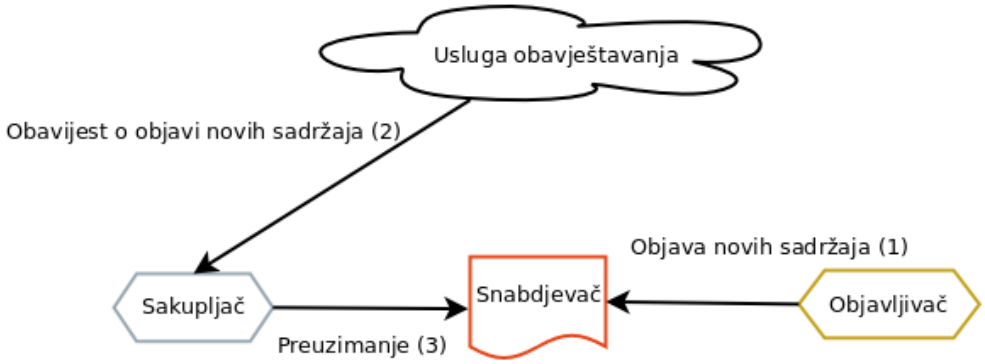
U slučaju da određeno vrijeme ne dobiva obavijesti o osvježavanjima određene teme, središte može poslati zahtjev objavljivaču za provjerom dostupnosti novih sadržaja. Iako je takav oblik komunikacije ostvaren na načelu povlačenja podataka, sustav dobiva novu funkcionalnost koja se može koristiti kao provjera

[1]Valjana središta su upravo ona koja su odabrana od strane izvornih sakupljača te samo ona imaju vrijednost parametra *hub.secret*

dostupnosti objavljivača sadržaja.

5.3. PubSubHubbub i RSSCloud

Usporedno s tehnologijom PSHB protokola razvila se i tehnologija RSSCloud [35]. Spomenuta tehnologija omogućuje unapređenje RSS sustava na sličan način kao što je ostvareno razvojem PSHB sustava. Program sakupljač pretplaćuje se na obavještavanje o objavi novih sadržaja pozivanjem funkcije objavljivača, koja se nalazi u oblaku. Podaci o funkciji nalaze se u spisu snabdjevača pod elementom *cloud*. Funkcija iz poziva otkriva mjesto sakupljača tj. njegovu IP [36] adresu. U trenutku kada objavljivač objavi nove sadržaje i osvježi spis snabdjevača, šalje se obavijest sustavu obavještavanja. Sustav obavještavanja obavješćuje sakupljača o dostupnosti novih sadržaja. Slijedeći korak predstavlja preuzimanje sadržaja spisa snabdjevača od strane sakupljača te obrada podataka. Na slici 5.3. prikazan je model rada RSSCloud sustava.



Slika 5.3: Prikaz rada RSSCloud sustava

Nedostatci rada RSSCloud sustava, koji ujedno predstavljaju prednosti PSHB sustava, jesu nemogućnost prijave povratnog sakupljača te preuzimanje podataka od strane sakupljača čime ovakvi sustavi nemaju sposobnost razmjernog rasta. U PSHB sustavima središte potiskuje podatke sakupljačima dok kod RSSCloud sustava sakupljač povlači spis snabdjevača.

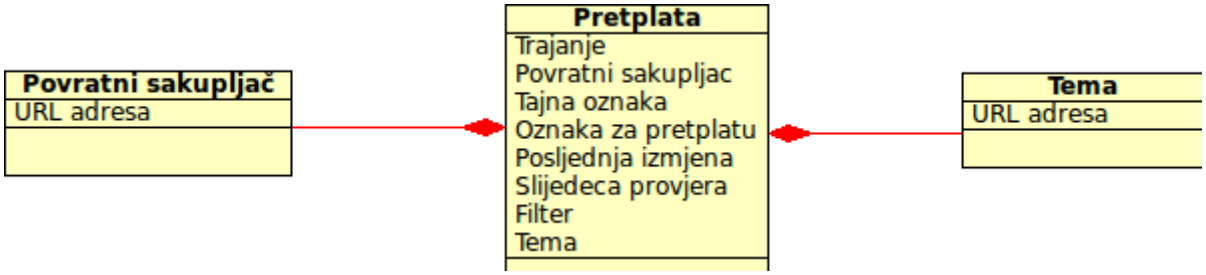
6. Programsko ostvarenje PubSubHubbub središta

Tijekom izrade rada ostvaren je programski sustav središta sukladno specifikaciji PubSubHubbub protokola uz dodatne mogućnosti opisane u narednim poglavljima. Sustav je ostvaren objektno-orijentiranim skriptnim programskim jezikom Python[1][37] koristeći programski radni okvir Django[38]. Poslužitelj na kojemu se izvodi navedeni sustav dostupan je preko URL adrese *bellpeace.webfactional.com*. U narednim poglavljima opisan je programski model sustava te procesi obrade pretplate i raspodjele podataka.

6.1. Programski model

Programski model sustava osmišljen je te ostvaren s ciljem izvođenja što manje koraka pri dohvaćanju podatala o pretplatama te pripadnim temama. Za svakog povratnog sakupljača vežu se pripadne pretplate čiji se sadržaj prima. Koristeći objektno-relacijsko mapiranje Django sustava, svaka pretplata, također, ima vezu na povratnog sakupljača. Svaka pretplata ima vezu na temu na koju se odnosi. Također, svaka tema ima vezu na sve pripadne joj pretplate. Na slici 6.1. prikazan je dijagram klasa koji prikazuje programsku arhitekturu ostvarenog sustava. Logička arhitektura sustava, odgovara PSHB sustavima, opisanim u poglavlju 5.

[1]Korištena je *JPython* inačica Python interpretatora pošto ista omogućuje istovremeno izvođenje više dretvi u istoimenim sustavima, što je nedostatak, često korištene, inačice *CPython*.



Slika 6.1: Prikaz rada RSSCloud sustava

Prikazani model te pripadno nazivlje koristiti će se pri opisu rada sustava u narednim poglavljima.

6.2. Ostvarivanje pretplate

Proces pretplaćivanja određen je po specifikaciji PubSubHubbub protokola uz dodatne mogućnosti te se ostvaruje slanjem odgovarajućih zahtjeva na URL adresu *bellpeace.webfactional.com/public_hub/suscribe*. Slanje HTTP POST zahtjeva za ostvarivanje pretplate omogućuje, uz parametre iz tablice 5.2., određivanje vrijednosti dodatnog parametra *hub.filter* koji služi za filtriranje sadržaja koji se potiskuje povratnim sakupljačima. Vrijednost spomenutog parametra jest JSON [39] zapis koji prati specifirani oblik, opisan u poglavlju 6.2. Ispravnost parametara zahtjeva provjerava se na svim parametrima osim spomenutog, kako bi ispravnost sustava bila sukladna specifikaciji. Ispravnost dodatnog parametra provjerava se pri potiskivanju sadržaja.

Kako bi se korisnicima sustava olakšao proces pretplaćivanja, središte, u slučaju neispravnosti parametara ili nedostatku istih, izvornom sakupljaču vraća odgovor u JSON oblik koji korisniku daje informacije o pogreškama pri pretplati. Sustav korisniku ne daje informacije samo o prvoj pogrešci koju pronađe, već o svim pogreškama. Na primjer, u slučaju da korisnik nije odredio URL adresu povratnog poslužitelja te krive vrijednosti o vrsti pretplate (pretplaćivanje/ukidanje), vraća se odgovor o neuspjeloj pretplati čije je tijelo poruke prikazano u tablici 6.1.

Tablica 6.1: Primjer JSON poruke koja opisuje pogreške u parametrima pretplate kada nije određen povratni poslužitelj te je krivo određena vrsta pretplate

```
{
    "hub.mode": ["1"],
    "hub.callback": ["0"]
}
```

Oblik zapisa JSON (engl. *Javascript Object Notation*), slično XML [14] obliku zapisa, omogućuje određivanje proizvoljnih struktura poruka koje se izmjenjuju između heterogenih sustava. Iako slabije čitljiv od strane ljudi, JSON oblik pogodan je za slanje poruka koje predstavljaju strukturirane podatke, za razliku od XML poruka koje služe za razmjenu spisa. JSON oblik predstavlja rječnik koji se zasniva na uređenim parovima ključ-vrijednost. Vrijednosti mogu biti novi rječnici, liste te pojedinačni entiteti. JSON oblik je strukturiran kao korijenski rječnik u kojeg mogu biti ugnježđene dodatne strukture, prethodno opisane. U primjeru u tablici 6.1. JSON poruka sadrži dva para ključ-vrijednost koji predstavljaju greške pri određivanju dva parametra zahtjeva. Vrijednost ključa predstavlja ime parametra dok je vrijednost, vezana za ključ, lista kodova grešaka. Za parametre koji moraju biti određeni pri zahtjevu, kod 0 predstavlja nedostatak istog, dok druge vrijednosti označuju drugu vrstu pogreške. Svi kodovi koji se mogu pojaviti za određeni parametar nalaze se u Dodatku A te na web stranici *bellpeace.webfactional.com/error_codes*.

Provjera pretplate odvija se sukladno specifikaciji protokola. U slučaju da povratni sakupljač nije dostupan, svakih sat vremena se šalje novi zahtjev s novom vrijednošću *hub.challenge* parametra. U slučaju da ne dobije odgovor u 24 sata, središte odustaje od pretplate.

Pri uspješnoj provjeri pretplate, sustav središta pokušava pronaći pripadne podatke povratnog poslužitelja te teme koji su određeni u zahtjevu. Ako isti ne postoje u sustavu, stvaraju se novi podaci koji predstavljaju spomenute entitete. Kako URL adrese jednoznačno određuju o kojim se entitetima radi, dohvaćanje dodataka izvodi se u vremenskoj složenosti *O(1)* [40]. Također, pri svakoj izmjeni podataka o pretplati, kao što je ponovna provjera zainteresiranosti za temu, osvježavaju se podaci o posljednjoj izmjeni. Kako bi se omogućile ponovne provjere zainteresiranosti, pri svakoj izmjeni osvježavaju se podaci o vremenu slijedeće provjere zainteresiranost. Vrijeme provjere se postavlja na vrijeme neposredno prije

isteka trajanja pretplate ili na dnevnoj bazi u slučaju da je pretplata trajna. Potonja provjera omogućuje prepoznavanje pretplata koje su trajno određene, a koje su u međuvremenu ukinute na strani pružatelja RSS/Atom usluga. Pronalazak takvih pretplata kao posljedicu ima brisanje istih, radi uštede na memorijskom prostoru.

6.2.1. Naknadne provjere pretplaćivanja

Prije isteka pojedinih pretplata, povratnim sakupljačima se šalju zahtjevi za provjerom pretplate kako bi se provjerila zainteresiranost za temu. Kako je opisano u prethodnom poglavlju, za svaku pretplatu postoje podaci o vremenu naknadnih provjera koje se koristi za obavljanje istih. Periodično provjeravanje izvodi se preko sustava obavljanja poslova u vremenskoj ovisnosti, pod imenom *cron-job* [41], operacijskog sustava Linux [41] na kojemu se izvodi sustav PSHB središta ostvarenog pri ovom radu. Svaku minutu sustav provjerava postojanje pretplata čije je vrijeme naknadne provjere prošlo ili jest jednako trenutnom. Pretplate se uređuju u niz na način da, gledajući od početka, svaka slijedeća pretplata ima vrijeme naknadne provjere veće od prethodne. Uređivanje u niz omogućeno je korištenjem predodređenih funkcija Django radnog okvira, koji koriste optimalne algoritme uređivanja u niz (engl. *sort*). Za svaku pretplatu obavlja se asinkrona provjera zainteresiranosti što predstavlja sposobnost na razmjeran rast opisanog podsustava. U trenutku kada se naiđe na pretplatu čije vrijeme naknadne provjere ne odgovara postavljenim uvjetima, zaustavlja se rad algoritma.

6.2.2. Provjera stanja pretplate

Izvorni sakupljači, koji su inicirali proces pretplaćivanja, mogu provjeriti stanja svih pretplata čiji se sadržaj potiskuje određenom povratnom sakupljaču. Slanjem HTTP GET zahtjeva na URL adresu *bellpeace.webfactional.com/public_hub/subscriptions* određujući URL adresu sakupljača te tajnu oznaku za autentificirano potiskivanje, izvorni poslužitelj dobiva podatke o svim pretplatama pripadnog povratnog sakupljača. Popis parametara, uz pripadni opis, se nalazi u tablici 6.2.

Tablica 6.2: Parametri za provjeru stanja pretplate, uz pripadni opis

PARAMETAR	OPIS
hub.callback	URL adresa povratnog sakupljača
hub.secret	tajna oznaka za autentificiranu raspodjelu podataka

Dobiveni podaci, uz URL adresu teme koja se prati, sadrže i informacije o datumu posljednje izmjene podataka o pretplati te datumu isteka pretplate u JSON obliku. Primjer mogućeg izgleda takve poruke nalazi se u tablici 6.3.

Tablica 6.3: Primjer JSON poruke koja sadrži informacijeo stanjima pretplata određenog povratnog sakupljača

```
{
    "susbscriptions: [{"
      "topic": "http://zp-pshbpublisher.appspot.com/feed",
      "lastUpdate": "2011-05-26 08:41:49.420316",
      "expireDate": "2011-05-27 08:41:49.420316"
    }]
}
```

Parametar oznake autentificirane raspodjele podataka posjeduje samo izvorni sakupljač čime je očuvana tajnost podataka.

6.2.3. Ukidanje pretplate

Ukidanje pretplate izvodi se na identičan način kao i ostvarivanje pretplaćivanja. Ako je određen parametar *hub.filter*, isti se zanemaruje. Pri primitku zahtjeva za ukidanje, sustav pokušava pronaći podatke o entitetima određenim u zahtjevu. U slučaju da isti ne postoje, sustav vraća odgovor sukladan pogreškama koje je pronašao. U slučaju ispravnog zahtjeva, brišu se podaci o pretplati koja je određena zahtjevom. Također, brišu se podaci o pripadnom povratnom poslužitelju te temi, ako ne postoje pretplate koje se vežu na iste. Opisane dodatne radnje omogućuju optimalno korištenje memorijskog prostora. Sve radnje pronalaska tj. dohvaćanja podataka izvode se u vremenskoj složenosti *O(1)*.

6.3. Raspodjela podataka

Objavljivač sadržaja, nakon što izmjeni sadržaj spisa snabdjevača, obavještava središte slanjem odgovarajućih zahtjeva, kako je opisano u poglavlju 5.2., na URL adresu *bellpeace.webfactional.com/public_hub/publish*. Središte dohvaća sadržaj spisa snabdjevača na način da prati sva HTTP preusmjeravanja. U slučaju da je preumjeravanje trajno, URL adresa spisa snabdjevača, tj. teme, trajno se sprema kako se slijedeći put ne bi morala nepotrebno pratiti sva preusmjeravanja. Za temu, koja odgovara URL adresi snabdjevača, pronalaze se sve pretplate te pripadni im povratni sakupljači. Neposredno prije paralelne raspodjele podataka, dodaju se zaglavlja za autentificiranu raspodjelu ako je isto određeno pri pretplati. Također, obavlja se filtriranje sadržaja sukladno vrijednošću *hub.filter* parametra, ako je isti određen.

6.3.1. Filtriranje sadržaja

U slučaju da je vrijednost filtra ispravna, obavlja se filtriranje, dok se u suprotnome ne obavlja potiskivanje podataka pripadnom povratnom sakupljaču. Filter jest predstavljen JSON oblikom u kojem su određene akcije koje se moraju sprovesti nad sadržajem, uz pripadne paramatre. U tablici 6.4. prikazane su radnje te pripadni opis.

Tablica 6.4: Dopuštene akcije za određivanje filtriranja sadržaja

AKCIJA	**OPIS**
exists	navodi se se koje XML elemente spis mora posjedovati
contains	navode se XML elementi te ključne riječi koje navedeni elementi u svom sadržaju moraju posjedovati; može biti određeno za sve pojave ili za barem jednu pojavu elementa
length	navode se XML elementi, duljina te odnos koji duljina sadržaja elementa mora ispunjavati; može se odrediti i odnos za duljinu cijelog spisa
exclude	navode se XML elementi koji neće biti uključeni u sadržaj koji će se raspodjeliti

Pri određivanju filtra, moguće je odrediti akcije za RSS spise ključnom riječi *rss*, Atom spise ključnom riječi *atom* te za obje vrste spisa ključnom riječi *all*.

Svaka ključna riječ predstavlja ključ dok je vrijednost lista koji sadrži parove ključ-vrijednost koji predstavljaju akciju te parametre akcije, respektivno.

U slučaju da su određeni ugnježđeni XML elementi tada se u filtru isti navode kao niz znakova koji sadrži put do ugnježđenog elementa krećući od jedne razine iznad korijenskog elementa RSS ili Atom spisa. Na primjer, element *topic* pri *item* elementu RSS spisa u JSON obliku filtra poprima vrijednost *item/title*. Jednostavni primjer filtra u JSON obliku prikazan je u tablici 6.5.

Tablica 6.5: Primjer jednostavnog filtra u JSON obliku

```
{
    "rss":[
        "exists" : ["title", "item/topic"]
    ]
}
```

U filtru se određuju pravila za RSS spise koje predstavlja raspodjelu samo onih sadržaja koji imaju element *title* te element *topic* u elementu *item*.

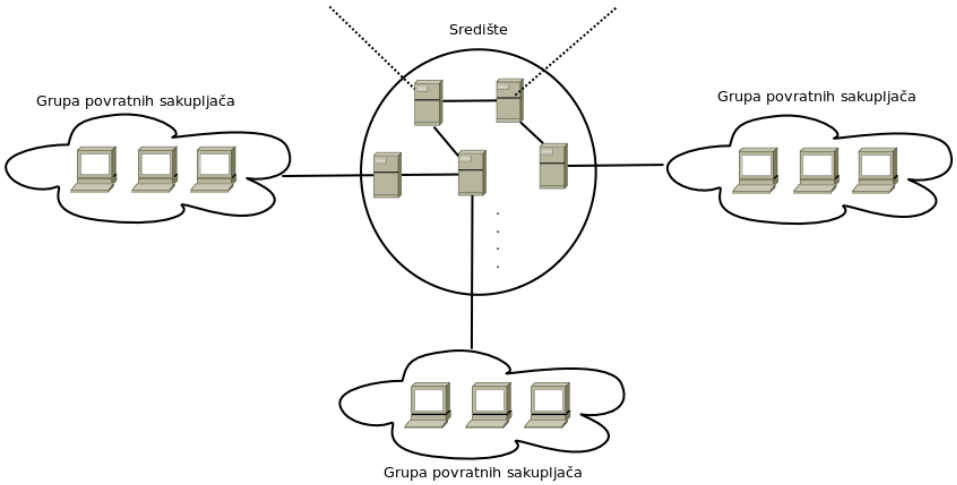
Poredak akcija pri određivanju filtra nije bitan pošto središte uvijek obavlja akcije istim redoslijedom. Prvo se provjerava *exists* pravilo zatim *contains* te *length*. U slučaju negativnog rezultata bilo koje od navedenih akcija prekida se filtriranje te se sadržaj ne potiskuje pripadnom povratnom sakupljaču. Ako su navedene akcije vratile pozitivan odgovor o spisu, tada se nad istim obavlja *exclude* akcija. U slučaju da pojedina aktivnost, određena u filtru, ne odgovora prethodno opisanom obliku ili nije navedena, ista se preskače.

6.4. Slijedeća generacija sustava

Slijedeća generacija ostvarenog sustava svladati će probleme centralizacije središta te omogućiti korisnicima jednostavnije određivanje proizvoljnih filtera. Također, moguće povećanje broj korisnika zahtijevati će preinake u arhitekturi sustava, kako bi sustav mogao pružati usluge uz dosadašnju kvalitetu istih. Planirane nadogradnje opisane su u slijedećim poglavljima.

6.4.1. Raspodijeljeni sustav središta

Kako bi se odgovorilo na povećanje broja korisnika tj. povratnih sakupljača, sustav središta se od jednog računala razvija u zaseban raspodijeljeni sustav. Svako računalo brine o određenom skupu povratnih sakupljača te čuva podatke o pretplatama istih. Pri procesu ujednačavanja opterećenja (engl. *load balancing*) moguće je da pojedina računala dohvaćaju sadržaje koji se moraju raspodjeliti povratnim sakupljačima koja nisu u njihovoj domeni odgovornosti. Tada se dobiveni podaci šalju ka računalima koja su za iste sadržaje odgovorna, koristeći tablice preusmjeravanja ili princip preplavljivanja [3]. Ilustrativni model opisanog sustava prikazan je na slici 6.2



Slika 6.2: Ilustrativni model raspodijeljenog PSHB središta

Pri korištenju tablica preusmjeravanja, računalo koje posjeduje sadržaj koji mu logički ne pripada šalje isti računalu koje je određeno tablicom preusmjeravanja. Slijedeće računalo šalje sadržaj drugom računalu koristeći svoju tablicu preusmjeravanja. Proces se nastavlja dok računalo, odogovorno za spomenuti sadržaj, ne primi isti. Ostvarivanje tablice preusmjeravanja ovisi o posjedničkim potrebama, no često je ostvareno kao raspodijeljena tablica raspršivanja [3]. URL adrese tema tj. snabdjevača se, određenim funkcijama sažimanja (engl. *hash*), mapiraju u jedinstvene oznake. Svako računalo preuzima odgovornost za podskup ključeva. Tablica preusmjeravanja se određuje tako da se upit nad ključem iz odgovarajućeg intervala prosljeđuje ka odgovarajućem, slijedećem računalu. Slijedeće računalo je određeno kao rezultat modulo funkcije zbroja oznake trenutnog računala i ključa te ukupne veličine prostora ključeva.

Pri korištenju principa preplavljivanja, računalo koje je primilo sadržaj šalje isti svim svojim susjedima. Svi susjedi šalju primljeni sadržaj svim svojim susjedima osim onom od kojega su primili sadržaj. Na ovaj način je osiguran primitak sadržaja od strane odgovarajućeg računala, no povećava se količina komunikacije zbog čega je prethodno opisani sustava tablica preusmjeravanja korišteniji u primjeni.

Slijedeća preinaka sustava, s ciljem ostvarivanja viđe razine sposobnosti razmjernog rasta, preorganozirati će prethodno opisani sustav na način da će za skup povratnih sakupljača biti odgovoran skup računala, umjesto dosadašnjeg pojedinačnog računala. Ovakav skup računala predstavlja administrativnu domenu [3].

6.4.2. Razvoj upitnog jezika za filtriranje

Mjesto sadašnjeg JSON oblika filtra, slijedeća generacija sustava za filtriranje će koristiti posjednički razvijen upitni jezik po uzoru na jezik SQL (engl. *Standard Query Language*) [42]. Kako je spomenuti jezik široko korišten, korisnicima sustava biti će olakšano razvijanje filtera. Baza podataka nad kojom će se vršiti upit biti će dohvaćeni sadržaj. Opisani jezik omogućiti će proizvoljno uređivanje sadržaja za razliku od dosadašnjeg uklanjanja određenih XML elemenata. Biti će omogućeno dupliciranje, konkatenacija te ugnježđivanje određenih elemenata čime će korisnici pretvarati sadržaj u željeni oblik bez razvijanje vlastite programske podrške te trošenja procesorskog vremena.

6.4.3. Pomak od RSS i Atom sustava

Kako su RSS i Atom zapisi zasnovani na XML standardu, slijedeća generacija sustava omogućiti će obradu i raspodjelu proizvoljnih XML spisa. Kako proces filtriranja kao osnovne jedinice koristi XML elemente, spomenuta nadogradnja neće zahtjevati korjenite preinake podsustava za filtriranje sadržaja. Na ovaj način se dopušta korištenje vlastitih XML struktura spisa pri snabdjevačima što omogućuje komunikaciju entiteta koji nisu samo dio RSS ili Atom sustava.

7. Zaključak

Razvijanjem Interneta i World Wide Weba počelo je novo razdoblje širenja i raspodjele informacija. Brzo i jednostavno korištenje postavilo je temelj za privlačenje velikog broja korisnika. Povećanje broja korisnika kao posljedicu je imalo povećani broj izvora informacija. Veliki broj izvora informacija otežao je korisnicima pronalazak i grupiranje informacija. Iako su spomenuti nedostaci djelomično uklonjeni pojavom web tražilica i sličnih sustava, klijent-poslužitelj model rada Weba imao je ograničenje neprestanog provjeravanja korisnika za objavom novih informacija.

Razvoj RSS i Atom sustava, pod utjecajem koncepta Weba u stvarnom vremenu, omogućio je korisnicima pretplaćivanje na određene sadržaje dok bi provjeravanje dostupnosti novih informacija obavljao klijentski programski sustav sakupljača. Iako napredni iz perspektive korisnika, spomenuti sustavi su, kao i sustavi tipičnog klijent-poslužitelj modela, radili na sustavu povlačenja. Kao rješenje spomenutih problema razvijen je PubSubHubbub (PSHB) protokol.

PSHB protokol, temeljen na protokolu HTTP, omogućuje nadogradnju RSS i Atom sustava tj. razvijanje sustava klasičnog objavi-pretplati modela. Korisnici se prijavljuju na teme čiji im se novi sadržaji potiskuju neposredno nakon objave istih. Uvođenjem novog entiteta, središta, redundantan rad sakupljača i objavljivača delegiran je istom. Središte zaprima pretplate te pri objavi novih sadržaja istovremene potiskuje podatke sakupljačima. PSHB protokol omogućuje razvoj sustava sa sposobnošću razmjernog rasta. Buduće inačice PSHB sustava, koje će imati raspodijeljeno ostvarenje središta, ostvariti će još veću razinu sposobnosti razmjernog rasta. Zbog veličine Weba koja neprekidno ide u smjeru povećanja, PSHB sustavi predstavljaju temelj raspodjele podataka u budućnosti.

Literatura

[1] Alen Bažant, Gordan Gledec, Željko Ilić, Gordan Ježić, Mladen Kos, Marijan Kunštić, Ignac Lovrek, Maja Matijašević, Branko Mikac, Vjekoslav Sinković *Osnovne arhitekture mreža*, Element, 2. izdanje, 2007.

[2] Louis Rosenfeld, Peter Morville, *Information Architecture for the World Wide Web*, O'Reilly Media, 2. izdanje, 2002.

[3] Andrew S. Tannenbaum, Maarten Van Steen, *Distributed Systems: Principles and Paradigms*, Prentice Hall, 2. izdanje, 2007.

[4] Ted Roden, *Building the Realtime User Experience*, O'Reilly Media, 2011.

[5] David E. Bakken, *Middleware*, Washington State University, 2011.

[6] Dejan S. Milojicic, Vana Kalogeraki, Rajan Lukose, Kiran Nagaraja1, Jim Pruyne, Bruno Richard, Sami Rollins 2, Zhichen Xu, *Peer-to-Peer Computing*, HP Laboratories Palo Alto, 2003.

[7] Jian Liang, Rakesh Kumar, Keith W. Ross, *Understanding Kazaa*

[8] Wayne Jansen, Tom Karygiannis, *Mobile Agent Security*, National Institute of Standards and Technology

[9] T. Berners-Lee, R. Fielding, U.C. Irvine, L. Masinter, *Uniform Resource Identifiers (URI): Generic Syntax*, http://www.ietf.org/rfc/rfc2396.txt, 1998.

[10] J. Gettys, R. Fielding, J. Mogul, L. Masinter, H. Frystyk, P. Leach, T. Berners-Lee, *Hypertext Transfer Protocol–HTTP/1.1*, http://www.w3.org/Protocols/rfc2616/rfc2616.html, 1999.

[11] T. Berners-Lee, D. Connolly, *Hypertext Markup Language-2.0*, http://www.rfc-editor.org/rfc/rfc1866.txt, 1995.

[12] Information Sciences Institute University of Southern California, *Transmission Control Protocol*, http://www.ietf.org/rfc/rfc793.txt, 1981.

[13] Leonard Richardson, Sam Ruby, *RESTfull Web Services*, O'Reilly Media, 2007.

[14] Tim Bray, Jean Poli, C.M. Sperberg-McQueen, Eve Maler, Francois Yergeau, *Extensible Markup Language (XML) 1.0*, WRC, 5. izdanje, 2008.

[15] John W. Rittinghouse, James F. Ransome, *Cloud Computing: Implementation, Management and Security*, CRC Press, 2010.

[16] David E.Y. Sarna, *Implementing and Developing Cloud Computing Applications*, CRC Press, 2011.

[17] Pin Nie, *An open standard for instant messaging: eXtensible Messaging and Presence Protocol (XMPP)*, Helsinki University of Technology

[18] Simon Horman, *SSL and TLS: An Overview of A Secure Communications Protocol*, 2005.

[19] Myung-Sun Kim, *Simple Authentication And Security Layer Incorporating Extensible Authentication Protocol*, University Of New Hampshire, 2005.

[20] Simone Leggio, *SIP for Instant Messaging and Presence Leveraging Extensions*, University Of Helsinki

[21] Jonathan B. Postel, *Simple Mail Transfer Protocol*, http://james.apache.org/server/rfclist/smtp/rfc0821.txt, 1982.

[22] All Streaming Media.com, http://all-streaming-media.com/articles/Streaming-Media-Intro_Streaming-protocols.htm, 06-05-2011.

[23] H. Schulzrinne, A. Rao, R. Lanphier, *Real Time Streaming Protocol (RTSP)*, http://www.ietf.org/rfc/rfc2326.txt, 1988.

[24] J. Postel, *User Datagram Protocol*, http://www.ietf.org/rfc/rfc768.txt, 1980.

[25] *RSS 2.0*, http://cyber.law.harvard.edu/rss/rss.html, Berkman Center, 2003.

[26] Will Richardson, *RSS: A quick guide for educator*, http://weblogged.com/wp-content/uploads/2006/05/RSSFAQ4.pdf, 13-4-2011.

[27] M. Nottingham, R. Sayre, *The Atom Syndication Format*, http://www.ietf.org/rfc/rfc4287.txt, 2005.

[28] David Flanagan, *JavaScript: The Definitive Guide*, O'Reilly Media, 5. izdanje, 2006.

[29] Alex Russell, Greg Wilkins, David Davis, Mark Nesbitt, *Bayeux Protocol– Bayeux 1.0.0*, http://svn.cometd.com/trunk/bayeux/bayeux.html, 2007.

[30] Ian Hickson, *The Web Socket protocol*, http://tools.ietf.org/html/draft-hixie-thewebsocketprotocol-54, 2009.

[31] Ian Hickson, *HTML5*, http://dev.w3.org/html5/spec/Overview.html, 2011.

[32] Daniel Honigman, *Lifestreams*, http://www.webershandwick.com/resources/ws/flash/Lifestre 10-4-2011.

[33] B. Fitzpatrick, B. Slatkin, M. Atkins, *PubSubHubbub Core 0.3 – Working Draft*, http://pubsubhubbub.googlecode.com/svn/trunk/pubsubhubbub-core-0.3.html, 20-05-2011.

[34] D. Eastlake, P. Jones, *Secure Hash Algorithm 1 (SHA1)*, http://www.faqs.org/rfcs/rfc3174.html, 2001.

[35] James Lewin, *Word-Press goes real-time with rssCloud*, http://www.podcastingnews.com/content/2009/09/wordpress-goes-realtime-with-rsscloud/, 23-4-2011.

[36] Information Sciences Institute University of Southern California, *Internet Protocol*, http://www.faqs.org/rfcs/rfc791.html, 1981.

[37] Mark Lutz, *Programming Python*, O'Reilly Media, 5. izdanje, 2010.

[38] *Django*, https://www.djangoproject.com/, 12-3-2011.

[39] D. Crockford, *The application/json Media Type for JavaScript Object Notation (JSON)*, http://www.ietf.org/rfc/rfc4627.txt, 2006.

[40] Thomas H. Cormen, Charles E. Leiserson, Ronald R. Rivest, Clifford Stein, *Introduction to Algorithms*, MIT Press, 2. izdanje, 2002.

[41] Machtelt Garrels, *Introduction to Linux*, 2007.

[42] James Hoffman, *Introduction to Structured Query Language*, 2001.

Dodatak A
Tablica kodova greški pri pretplaćivanju

U slijedećoj tablici prikazani su kodovi grešaka pri pretplaćivanju na PSHB sustav ostvaren u ovom radu te pripadni opisi.

PARAMETAR	KOD	OPIS
hub.mode	0	izostanak
	1	nedopuštena vrijednost
hub.callback	0	izostanak
	1	URL adresa posjeduje fragment
	2	kriva URL shema
hub.topic	0	izostanak
	1	URL adresa posjeduje fragment
hub.verify	0	izostanak
	1	nedopuštena vrijednost
hub.lease_seconds	1	vrijednost nije cijeli broj
	2	vrijednost manja od nule

Raspodijeljeni sustav za analizu tijekova aktivnosti na Webu

Sažetak

Razvoj World Wide Web informacijskog sustava, temeljenog na modelu komunikacije klijent-poslužitelj, privukao je sve veći broj korisnika. Povećanje broja korisnika kao posljedicu je imalo povećanje broja izvora informacija. Kako bi korisnicima bilo omogućeno praćenje više izvora informacija sa sigurnošću da iste neće propustiti zbog učestalog osvježavanja podataka, razvijeni su RSS i Atom sustavi. Spomenuti sustavi su prividno potiskivali podatke korisnicima po modelu objavi-preplati, no unutrašnji rad je predstavljao komunikaciju na načelu učestalog povlačenju podataka. Kao nadogradnja na spomenute sustave, razvijen je PubSubHubbub protokol, koji omogućuje potiskivanje podataka u stvarnom vremenu. Pri PubSubHubbub sustavima nema redundantog rada poslužitelja ni programa sakupljača, već je za raspodjelu podataka zaslužan novi entitet, središte. PubSubHubbub protokol omogućuje razvoj sustava zasnovanih na modelu komunikacije objavi-pretplati te je nastao pod utjecajem koncepta Weba u stvarnom vremenu.

Ključne riječi: World Wide Web, Web u stvarnom vremenu, PubSubHubbub, RSS, Atom, HTTP, REST, raspodijeljeni sustavi, objavi-pretplati, klijent-poslužitelj

Distributed system for analyzing Web activity streams

Abstract

World Wide Web development and growth led to increased number of users. Increased number of users led to development of new technologies and increased number of information providers. It became difficult for users to keep track of new information availability, so technologies were developed to push information towards users. Communication model evolved from client-server model to increasingly popular publish-subscibe model. One of the first, previously described, technologies developed were RSS and Atom which represented publish-subscribre data distribution from user perspective but still based their work on frequent data polling. As a upgrade, PubSubHubbub protocol was developed, which enables authentic publish-subscribe distributed systems development. RSS and Atom deficiencies, like redundant server and aggregator work, network traffic overhead, were overcomed by introducing hub middleware which purpose is real time data distribution. PubSubHubbub protocol was developed under the influence of emerging real-time Web technologies and concepts.

Keywords: World Wide Web, Real-time Web, PubSubHubbub, RSS, Atom, HTTP, REST, distributed systems, publish-subscribe, client-server