Sveučilište u Zagrebu
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
Zavod za automatiku i procesno računarstvo

Raspodijeljeni sustav za pohranu i dohvat podataka

Diplomski zadatak br. 1493
Ivan Voras
0036380923

Zagreb, 2006.

Zahvaljujem se prof. Mariu Žagaru za vodstvo i strpljivost, Kristijanu Zimmeru za šansu i Sonji Miličić što je bila u blizini.

1. Sažetak

U ovom radu proučene su tehnologije raspodijeljenih partnerskih mrežnih sustava za pohranu i dohvat podataka. Ovi sustavi se sastoje od čvorova međusobno povezanih na visokoj razini, bez obzira na njihovu fizičku povezanost u računalne mreže, u virtualne "nadmreže". Kako su svi čvorovi partnerske nadmreže ravnopravni (ne postoji podjela na poslužitelje i klijente), ove tehnologije posjeduju neke jedinstvene probleme: međusobno otkrivanje čvorova i nestalna aktivnost čvorova. Predstavljeni su različiti načini rješavanja ovih problema, te predložena arhitektura partnerskog sustava koji se temelji na TCP/IP protokolima i obavlja razmjenu podataka strukturiranih u rječnike. Implementirano je prototipno rješenje u programskom jeziku Java, te izrađena tehnička i korisnička dokumentacija.

A distributed system for data storage and retrieval – abstract

This work studies technologies applied in building distributed peer-to-peer network systems for data storage and retrival. These systems consist of arbitrary number of network nodes connected on a higher layer and their connectivity is not influenced by physical topologies of networks of which they are a part of – they form "overlay networks." All nodes in a peer-to-peer overlay network have equal functionality (there are no "server nodes"). This equality of nodes presents some unique problems: mutual discovery of active nodes and varied connectivity as nodes join and depart the network. Various solutions for these problems are presented and evaluated and an architecture for a peer-to-peer system is proposed. The proposed system uses TCP/IP protocols for inter-node communication and allows storage and retrieval of data records in the form of dictionaries (mappings). A prototype of the system is implemented in Java programming language, and technical and user documentation is provided in this work.

2. Sadržaj

3. Uvod

Pohrana podataka je jedan od najznačajnijih zadataka bilo kojeg računalnog sustava. Pažnja posvećena ovoj temi vidljiva je u tome što je tijekom godina osmišljen veliki broj vrlo različitih sustava koji omogućavaju upravljanje, pohranu i dohvat digitalnih podataka raznih vrsta, te što ona još uvijek predmet aktivnog izučavanja. Pri razmatranju sustava za pohranu podataka uvijek se iznova pojavljuju dva važna problema: organizacija podataka te ograničenja kapaciteta uređaja koji pohranjuju podatke na fizičkom sloju.

Pojavom svestrano korištenih računalnih mreža tijekom zadnjih desetljeća postalo je jasno da budućnost pohrane podataka u općenitom slučaju ne leži u monolitnim sustavima velikih kapaciteta nego u sustavima raspodijeljenima na većem broju neovisnih lokacija, potencijalno nehomogenima, i povezanima putem standardnih tehnologija za stvaranje računalnih mreža. Rastom perfomansi, mogućnosti i podatkovnog kapaciteta računala (bilo koje veličine ili primjene) sve više se smanjuje potreba za dediciranim računalima čiji je jedini zadatak posluživanje podataka. Podaci danas mogu biti bez problema i uvođenja dodatne kompleksnosti pohranjeni na računalima na kojima su stvoreni, ili na računalima na kojima ih je iz raznih razloga "prikladnije" čuvati.

Sukladno opisanom tehnološkom razvoju, cilj ovoga rada je stvoriti sustav za pohranu i dohvat podataka koji ima osobine "partnerskog" (*peer-to-peer*) načina rada umjesto u praksi uobičajenog klijentsko-poslužiteljskog (*client-server*). Osobine koje se žele postići ovim načinom rada su veća otpornost na pogreške i ispade u radu pojedinačnih računala, lokalizacija podataka na računalima na kojima su nastali (ili na kojima su "bitni") te povećanje ukupnog dostupnog kapaciteta za pohranu podataka.

Jedan od ciljeva implementacije sustava u ovom radu, uz korektnost i lakoću budućeg održavanja i nadograđivanja, je izrada modularnog sustava koji će dugoročno biti moguće koristiti na računalima različitih kapaciteta. Iako je ova početna implementacija načinjena prvenstveno za računalne sustave relativno velikih kapaciteta (*desktop* vrste), pri izradi je posvećena pažnja da se koriste principi koji će omogućiti kasniju prilagodbu ili nadogradnju za izvođenje na računalima manjih kapaciteta (ali koji ipak zadovoljavaju minimalne zahtjeve kao postojanje Java okruženja sa adekvatnim mogućnostima).

Dalje u ovom radu će računala koja su dio računalne mreže u razmatranju biti zvana “čvorovi” da se naglasi općenitost principa i mogućnost primjene na uređajima manjih kapaciteta ili neuobičajene izvedbe.

4. Partnerski način rada i partnerske mreže

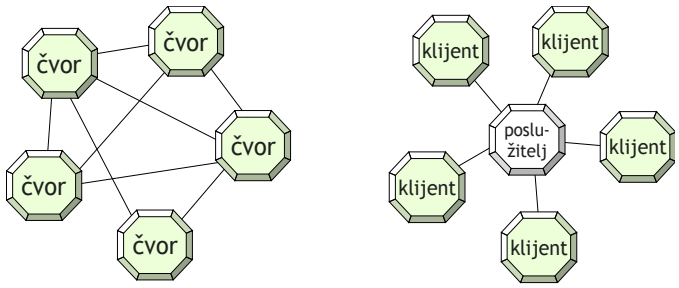
Rast popularnosti partnerskih računalnih mreža dogodio se pojavom alata i servisa za slobodnu globalnu razmjenu datoteka. Prvi od ovih sustava bio je *Napster*, a po njegovom gašenju stvoreni su brojni drugi sustavi iste ili slične namjene i načina rada, među kojima su poznati *Gnutella*, *Kazaa* i *FreeNet*. Zajednička osobina svih ovih sustava je da im je osnovna namjena (izvan koje se nisu dalje razvijali) razmjena datoteka između vrlo velikog broja korisnika / čvorova, od kojih proizvoljan broj korisnika može imati vlastitu (lokalnu) kopiju datoteke te se dohvat sadržaja jedne datoteke obavlja primanjem dijelova datoteke od više čvorova istovremeno.

Definicija partnerskih računalnih mreža[1] je: "*... Raspodijeljeni sustavi koji se sastoje od međusobno povezanih čvorova koji se mogu samostalno organizirati u mrežne topologije sa svrhom dijeljenja raspoloživih resursa kao što su korisnički podaci, procesorsko vrijeme, kapacitet za pohranu podataka ili mrežna propusnost, te koji se mogu samostalno adaptirati na ispade funkcionalnosti i nepredvidive dolaske i odlaske čvorova na mreži, uz zadržavanje prihvatljive razine prospojenosti i performansi bez potrebe za nadzorom, kontrolom i podrškom iz jednog središnjeg mjesta.*"

Iz definicije je vidljivo da partnerski način rada mreže čvorova ima slijedeće osobine:

- Svaki čvor je ravnopravan, uključujući mogućnosti prihvaćanja upita o podacima od strane korisnika ili drugih čvorova
- Komunikacija između čvorova je izravna (bez međukoraka kao što su poslužitelji)
- Čvorovi samostalno prikupljaju informacije o dostupnosti drugih čvorova
- Pojedinačni čvorovi imaju u svom lokalnom sustavu za pohranu na raspolaganju samo dio podataka, odn. podskup ukupnih podataka dostupnih na mreži

Navedene osobine su dijametralno suprotne osobinama klasičnih klijentsko-poslužiteljskih sustava, u kojima postoji jasna razlika između čvorova koji pohranjuju i nude sadržaj odn. podatke (poslužitelji) te čvorova koji podatke potražuju, obrađuju ili stvaraju (klijenti).



Slika 1. Usporedba strukture mrežne povezanosti kod partnerskih mreža i klijentsko-poslužiteljskih mreža

Decentraliziranost partnerskih mreža donosi probleme koji ne postoje u klijentsko-poslužiteljskim mrežama: praćenje aktivnih čvorova u mreži (jer se čvorovi mogu dinamički priključivati u mrežu i odlaziti iz mreže) te propagiranje korisničkih upita kroz mrežu (jer svaki čvor ima samo dio podataka). Različite implementacije partnerskih sustava rješavaju ove probleme na različite načine. Neka od ovih rješenja odstupaju od teoretski čiste implementacije partnerskog principa rada u korist brzine ili praktičnosti rješenja, te unose neke od slijedećih ograničenja ili modifikacija:

- Prisutan je središnji poslužitelj koji prikuplja informacije o čvorovima (status čvorova)
- Središnji poslužitelj prikuplja informacije i o tome koje podatke čvorovi posjeduju
- Korisnički upiti za podacima moraju sadržavati neku dodatnu informaciju o tome gdje se sadržaj nalazi

Sustavi partnerskih mreža koji posjeduju neke od navedenih ograničenja se nazivaju i "hibridnim partnerskim mrežama" jer nastala arhitektura ima neke značajke klijentsko-poslužiteljskih mreža. Važno je napomenuti da sustavi partnerskih mreža koji imaju koncept "posebnih" čvorova (*supernodes*) koji sadrže podatke o obližnjim čvorovima (u nekoj proizvoljno određenoj regiji) i/ili koji služe kao poveznici između mreža (*gateways*) ne smatraju hibridnim ukoliko je određivanje odn. proglašavanje čvora "posebnim" dinamičko, odn. u slučaju ispada "posebnog" čvora neki drugi, slučajni čvor postaje "poseban."

4.1. Partnerske mreže i *grid* sustavi

Partnerske i *grid* mreže su dva pristupa izgradnji raspodijeljenih sustava, sa fokusom na različite osobine. Uobičajena namjena *grid* mreža je stvaranje sustava koji nude visoke

performanse ali su u izvedbi ograničeni na poznata okruženja i homogene konfiguracije te unaprijed predvidljive komunikacijske mogućnosti. Tehnologije partnerskih mreža ne postavljaju usporedivo veliki naglasak na performanse sustava, nego na pouzdanost i mogućnosti samostalnog reagiranja na dinamične događaje u mreži (kao što su dolazak i odlazak čvorova). Ipak, ova dva pristupa počinju konvergirati jer se, zbog smanjenja troškova održavanja, u *grid* mrežama istražuju mogućnosti samostalne konfiguracije, a sa druge strane partnerske mreže počinju biti korištene za širi skup primjena od onih sa kojima je razvoj započeo[2].

4.2. Pregled nekih poznatih i povijesnih partnerskih mreža

Implementacije teoretski čistog modela partnerske mreže u praksi su rijetke, uglavnom zbog teško rješivog problema otkrivanja topologije odn. mrežne prospojenosti čvorova. Sustavi koji se aktivno koriste na velikom broju čvorova koriste široko rasprostranjenu infrastrukturu baziranu na TCP/IP protokolima te zbog jednostavnosti upotrebe ipak moraju koristiti standardne infrastrukturne servise kao što su DNS informacijski poslužitelji. Različiti sustavi rješavaju ove probleme na različite načine.

4.2.1. Povijesni i rani sustavi

Unatoč tome što pojam partnerskih mreža još nije postojao, tragovi ove ideje i začetci budućih sustava se mogu vidjeti u sustavima koji datiraju još iz samih početaka Interneta. U retrospektivi, sustavi kao što su e-mail poslužitelji (protokol SMTP), mrežni diskusijski poslužitelji (Usenet, protokol NNTP) i raspodijeljeni sustav za tekstualni razgovor u realnom vremenu (*Internet Relay Chat* - IRC) imaju osobine čije postojanje ih može smjestiti u kategoriju partnerskih mreža.

Programi za razmjenu e-mail poruka (e-mail poslužitelji), osobito rane implementacije kao što je Sendmail, bili su namijenjeni izvođenju na svakom računalu koje je bilo povezano u mrežu. Svaka instanca poslužitelja na računalu bi od (lokalnog) korisnika dobivala e-mail poruke za daljnju dostavu, a od drugih računala u mreži bi primala poruke za dostavu korisniku. Dostava poruka korisnicima na udaljenim računalima bi nerijetko išla putem nekoliko posredničkih računala i nije ovisila o fizičkoj topologiji mreže.

Mrežni diskusijski poslužitelji (Usenet) su tradicionalno međusobno povezani, te iako korisnici poruke šalju i primaju na njima bliskim poslužiteljima, ove poruke se propagiraju kroz mrežu povezanih poslužitelja na način sličan onom na koji se propagiraju informacije o sadržajima u

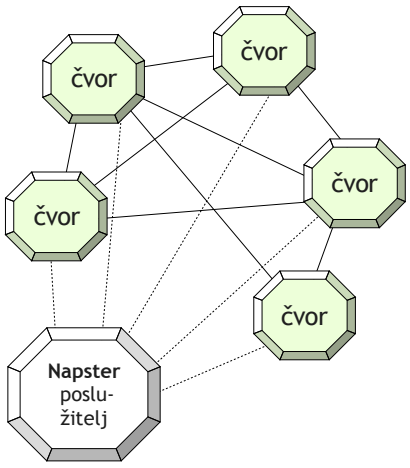
partnerskoj mreži (te se ovdje nailazi na iste probleme, kao što je višestruko dostavljanje iste poruke). Razlika je ipak u tome što je mrežu poslužitelja potrebno unaprijed konfigurirati.

IRC poslužitelji su međusobno povezani kao i Usenet poslužitelji, te se poruke propagiraju do svih poslužitelja u realnom vremenu. Dodatno, IRC podržava izravno spajanje i komunikaciju dva korisnika (*Direct Client-to-client* – DCC). Ove mogućnosti se vrlo često koriste za stvaranje *ad-hoc* partnerskih mreža, u kojoj se informacije o dostupnosti sadržaja razmjenjuju u globalnom sustavu (na jednom "kanalu"), a sami korisnički podaci pomoću izravne komunikacije dva korisnika. Ovaj način korištenja IRC mreža uživa gotovo jednaku popularnost za razmjenu multimedijskih sadržaja kao i "prave" partnerske mreže.

4.2.2. Napster

Prvi popularni sustav partnerskih mreža za razmjenu datoteka (i ujedno prvi koji je donio lošu reputaciju ovakvim sustavima) je *Napster*. Nastao 1999. godine, njegova svrha je bila razmjena glazbenih i multimedijalnih datoteka. Rad Napstera temeljen je na postojanju jednog središnjeg poslužitelja koji prima i pohranjuje informacije o svim spojenim čvorovima te sadržajima koji ovi nude, što je je omogućilo vrlo brza i precizna pretraživanja (izvođena lokalno na poslužitelju). Sami podaci – datoteke odn. dijelovi datoteka – bili su prenošeni u izravnoj komunikaciju između više korisnika istovremeno kao i kod drugih partnerskih mreža.

Zbog toga što ovisi o postojanju središnjeg poslužitelja, Napster danas ne bi bio svrstan u kategoriju pravih partnerskih mreža (odn. bio bi u istom svojstvu u kojem su starije tehnologije kao IRC).



Slika 2. Povezanost čvorova u Napster mreži

Upravo postojanje ove kritične točke (poslužitelja) je omogućilo brzo zaustavljanje (gašenje) Napsterove mreže.

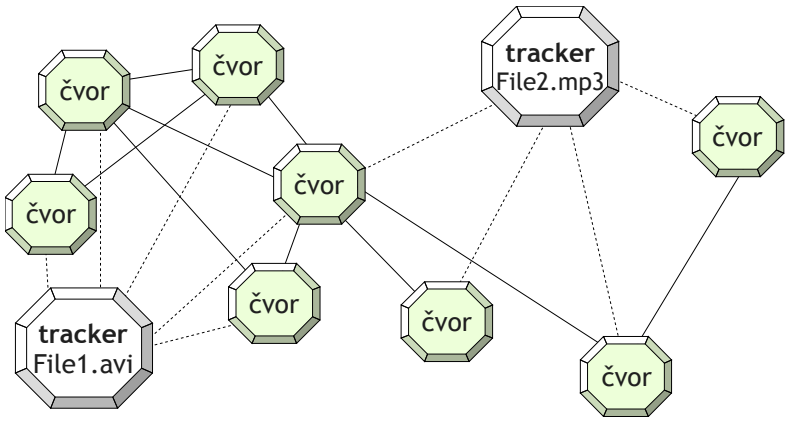
4.2.3. Gnutella

Povijest Gnutelle počinje ne sasvim legalnim događajima vezanim uz prvu inačicu programa, no danas postoji veliki broj legalnih Open-source klijenata koji komuniciraju istim protokolom i omogućavaju spajanje u "Gnutella mrežu." Za razliku od Napstera, rad Gnutella mreže je od početka potpuno decentraliziran. Novi čvorovi pri početku rada testiraju dostupnost unaprijed poznatih čvorova (lista poznatih čvorova se dinamički osvježava) te izabiru nekoliko čvorova na koje se priključuju. Izbor čvorova za priključivanje se izvodi tako da optimizira dostupnost podataka te jedan čvor u pravilu nije spojen na više od deset drugih čvorova istovremeno. Korisnički upiti se propagiraju kroz mrežu spojenih čvorova, te se rezultati korisniku prikazuju redoslijedom kojim pristižu čvoru koji je postavio upit. Pretraživanje mreže stoga nije deterministički precizno niti ograničeno u vremenu, nego ovisi o trenutnoj topologiji mreže.

Kasnije verzije protokola definiraju "posebne" čvorove koji su istovremeno spojeni na vrlo veliki broj drugih čvorova (ovisno o kapacitetu mrežne veze) te mogu ubrzati dohvat rezultata upita jer upite propagiraju na veliki broj čvorova istovremeno. Gnutella mreža je aktivna i danas, te je jedna od najpopularnijih partnerskih mreža za razmjenu datoteka.

4.2.4. BitTorrent

Danas najpopularniji sustav za razmjenu datoteka putem stvaranja partnerskih mreža razvijen je s ciljem da autorima sadržaja olakša distribuciju velikih datoteka iskorištavanjem mrežnih kapaciteta klijenata koji su započeli preuzimanje datoteke za daljnju distribuciju preuzetih dijelova datoteke. Ovo je hibridni sustav u kojem središnji poslužitelj sadrži listu čvorova koji su u procesu preuzimanja datoteka (te informacije o tome koje dijelove datoteka imaju pojedini čvorovi), no za razliku od Napstera ne postoji samo jedan globalni poslužitelj već svaki autor odn. distributer sadržaja može uspostaviti svoj poslužitelj. Ovi poslužitelji sadrže samo informacije o datotekama i čvorovima koji ih preuzimaju, ali ne i sadržaj datoteka, koji se između čvorova prenosi u partnerskom načinu rada. Zbog toga što nemaju sadržaj datoteka nego samo "prate" metapodatke o datotekama i čvorovima, ovi poslužitelji se nazivaju *trackers*.



Slika 3. Primjer BitTorrent partnerske mreže sa dvije datoteke prijavljene kod dva *trackera*

Kako različite datoteke mogu biti prijavljene kod različitih *trackera*, ne postoji jedna globalna BitTorrent mreža, već se radi o nizu malih nezavisnih grozdova čvorova koji su okupljeni oko *trackera*. Unatoč tome što postoje kritične točke u funkcioniranju mreže čiji ispad ili prekid rada onemogućavaju dostupnog sadržaja, u praksi se jednostavnost uspostavljanja novih *trackera* pokazala dovoljnom za osiguranje nastavka rada i popularnosti sustava. Za razliku od praktički svih drugih partnerskih mreža koje se koriste za distribuciju velike količine sadržaja, BitTorrent mreže ne posjeduju mogućnosti izravnog pretraživanja sadržaja dostupnog na *trackerima*, nego se oslanjaju na eksternu distribuciju ovih informacija putem malih datoteka, tzv. "torrenta."

4.3. Principi rada i tehnologije partnerskih mreža

Koristan pojam pri razmatranju partnerskih mreža je "nadmreža" (*overlay network*) koji se koristi za označavanje mreže povezanih čvorova čija međusobna povezanost ne ovisi o topologiji fizičke računalne mreže koja ih povezuje niti, u teoriji, o protokolu kojim pojedini čvorovi komuniciraju. Čvorovi pripadaju nekoj nadmreži ukoliko postoji pojam pripadnosti i mogućnost međusobne komunikacije čvorova.

Funkcioniranje partnerske mreže može se podijeliti u nekoliko zasebnih, uvijek prisutnih koraka:

- Formiranje nadmreže, odn. otkrivanje virtualne topologije nadmreže
- Pretraživanje nadmreže u potrazi za čvorovima koji sadrže željene podatke
- Prijenos korisničkih podataka (*payload*) između čvorova

Navedeni koraci su ujedno i kronološki poredan redoslijed operacija sa stajališta novog čvora koji se priključuje u partnersku mrežu.

4.3.1. Formiranje nadmreže

Prvi korak u stvaranju partnerske mreže je međusobno otkrivanje čvorova koji sudjeluju u mreži. Kako je broj i fizički smještaj čvorova u nadmreži proizvoljan i neovisan od njihovog fizičkog smještaja i topologije mreže pomoću koje su povezani, međusobno otkrivanje čvorova se ne može oslanjati na mehanizme i podatke koji su dostupni iz TCP/IP- i nižih mrežnih slojeva mreža.

U danas korištenim partnerskim mrežama implementirano je nekoliko načina na koji čvorovi koji se tek priključuju u nadmrežu mogu otkriti druge čvorove prisutne u nadmreži:

- Pomoću pred-definirane liste adresa čvorova koji "uvijek postoje"
- Pomoću pred-definirane adrese poslužitelja koji sadrži adrese aktivnih čvorova (*hub*)
- Uz mogućnost dohvata liste čvorova ili poslužitelja putem eksternih protokola (kao što su HTTP ili FTP)
- Uz mogućnost korisničkog unosa čvora za čije postojanje korisnik saznaje na neki drugi način
- Uz mogućnost korištenja ograničenih mogućnosti otkrivanja čvorova koji su mogući iz nižih slojeva odn. TCP/IP mreže.

Zbog jednostavnosti implementacije i korištenja, najčešće se koriste prva dva načina, pri čemu su predefinirane adrese obično ugrađene u sam klijentski program koji omogućuje rad u partnerskoj mreži. Unatoč tome što ovaj način implementacije otvara priliku za onemogućavanje rada partnerske mreže utjecanjem na ključne čvorove ili poslužitelje, u praksi se pokazao dovoljno otporan uz često osvježavanje programskih verzija ili ako su klijentski programi otvorenog koda. Primjeri partnerskih sustava koja koriste ovaj način rada su *Napster* i *Kazaa*.

Po popularnosti slijede treći i četvrti način, u kojem se lista dostupnih čvorova i poslužitelja čuva odvojeno od samih klijentskih aplikacija, te ponekad i distribuira drugačijim metodama. Na ovaj način rade sustavi kao *BitTorrent* i *DC++*.

Samostalno otkrivanje drugih čvorova korištenjem funkcionalnosti mrežnih slojeva na kojem se nadmreža temelji (najčešće TCP/IP) se koristi rijetko, i uglavnom u specijaliziranim rješenjima namijenjenim korištenju u malim zatvorenim mrežama (lokalnim mrežama, LAN). Ovaj način otkrivanja čvorova se ponekad koristi kao pomoćni način u slučaju kada drugi implementirani

načini ne uspiju pronaći pogodne čvorove u nadmreži. Jedan od sustava koji ga koristi u tom obliku je *Gnutella*.

Različiti načini otkrivanja čvorova u nadmreži određuju detalje topologije nadmreže. Načini otkrivanja koji se oslanjaju na postojanje središnjih poslužitelja ili "uvijek prisutnih" čvorova ("hibridne" partnerske mreže) najčešće rezultiraju topologijom u kojoj su svi čvorovi u stalnom kontaktu sa ovim poslužiteljima ili čvorovima te stoga imaju jednostavan i brz način dohvata informacija o drugim čvorovima nadmreže ili čak o sadržajima koji ovi nude (ukoliko arhitektura protokola omogućava prikupljanje ovih informacija na ovaj način). Ovi "posebni" čvorovi obično primaju veći broj pojedinačnih upita od "običnih" čvorova te moraju imati više procesorske snage i bolju mrežnu propusnost[3]. Mrežne topologije sustava koji koriste ovaj način rada imaju kombinirane osobine zvjezdaste topologije (veze prema "posebnim" čvorovima) i međusobno prospojene (*mesh*) topologije. Potpuno decentralizirane partnerske mreže kod kojih su svi čvorovi jednaki nemaju osobina zvjezdaste topologije već samo nepravilno prospojenu strukturu.

4.3.2. Pretraživanje nadmreže

Najčešća svrha partnerskih mreža je pohrana i dohvat informacija. Za uspješno ispunjavanje ove svrhe informacije moraju biti strukturirane te mora biti moguće uspostaviti kriterije po kojima su one pohranjene na određeni čvor i po kojima će ih korisnici moći pronaći.

Mogućnosti pretraživanja partnerskih mreža uvelike ovise o načinu na koji je izvedeno formiranje nadmreže, odn. njenoj topologiji. U slučaju postojanje središnjeg poslužitelja moguće je održavati precizan popis svih čvorova koji sudjeluju u nadmreži, što osigurava da korisnički upit može biti proslijeđen do svih čvorova, te da je vrijeme potrebno da svi čvorovi odgovore donekle predvidivo. Suprotno tome, kod potpuno decentralizirane partnerske mreže korisnički upit se mora nedeterministički propagirati od jednog čvora do čvorova s kojima je ovaj povezan (koji su mu "susjedni"), te nije moguće garantirati da će upit stići do svih čvorova (što se može dogoditi u slučaju ispada dijela čvorova ili zbog nedostatka resursa kao što je mrežna propusnost) niti se može predvidjeti vrijeme potrebno da upit obiđe do svih čvorova. Iz ovih razloga u praktične implementacije partnerskih mreža se ponekad uvode "posebni" čvorovi, koji imaju više resursa na raspolaganju, te su povezani sa neuobičajeno velikim brojem drugih čvorova. U ovom slučaju, čvorovi koji su izvor upita za podacima preferiraju slati upite izravno ovim "posebnim" čvorovima umjesto da ih propagiraju kroz "obične" čvorove. Način proglašavanja pojedinih

čvorova "posebnim" može biti statičan (unaprijed predviđen od autora sustava ili od korisnika) ili dinamičan (ponekad sa testiranjem pogodnosti čvorova tijekom rada sustava).

4.3.3. Načini pretraživanja nadmreža

U slučaju partnerskih mreža sa velikim brojem čvorova, od velike važnosti je osigurati što brže pretraživanje nadmreže za korisnički specifiranim podacima.

Ako su podaci jednostavnog oblika, kao što su imena datoteka/ili ključne riječi koje ih opisuju, moguće je koristiti vrlo efikasne tehnike zasnovane na raspodijeljenim *hash* tablicama (*distributed hash table*). U ovim tehnikama imena datoteka se prevode u binarne nizove ograničene duljine putem funkcija jednosmjerne kompresije (*hash* funkcije). Čvorovima u nadmreži se pridjeljuju određeni dijelovi prostora ključeva, te ovi time postaju "odgovorni" za podskupove ključeva. Uvode se pravila koja uređuju međusobnu povezanost čvorova na način da se za određeni ključ može precizno odrediti "smjer" pretraživanja nadmreže, čime se kompleksnost pretraživanja (s obzirom na broj čvorova koje treba pretražiti, od ukupnih N) može smanjiti na O(log N) [4]. Neka od mogućih pravila raspodjeljivanja dijelova prostora ključeva na čvorove uključuju razmatranje ovog prostora kao točaka na kružnici, sa pripadnom metrikom, ili kao binarnog stabla. Ovaj način pretraživanja se može koristiti samo u slučaju da klijenti unaprijed znaju cijela imena datoteka ili točne ključne riječi kojima je generiran ključ, ali ga nije moguće koristiti u slučaju potrebe pretraživanja dijelova imena datoteka ili ključnih riječi.

Općeniti način pretraživanja u slučaju kompleksnih podataka ili kompleksnih zahtjeva za pretraživanjem je propagiranje upita kroz sve čvorove nadmreže. Ovo pretraživanje mora osigurati da upit pouzdano dospije do svih čvorova u nadmreži, te da rezultati pretraživanja dospiju do čvora na kojem je korisnik pokrenuo upit.

5. Arhitektura raspodijeljenog partnerskog sustava za pohranu i dohvat podataka

Cilj ovog rada je izrada partnerskog sustava za pohranu i dohvat podataka sa ravnopravnim čvorovima, te je u tu svrhu osmišljena prikladna arhitektura sustava. Načinjeni sustav implementiran je kao konzolna aplikacija u jeziku Java, verzije 1.5, uz korištenje razvojnog alata NetBeans 5. Radni naziv sustava tijekom njegovog nastanka je "*Cogent*[1]." U kontekstu projekta definiraju se pojmovi "Cogent čvor" i "Cogent mreža."

Cogent čvor je definiran kao jedna instanca Cogent procesa vezana za jednu IP adresu, te koja može komunicirati sa drugim instancama vezanima za druge IP adrese. Svaki Cogent čvor posjeduje vlastitu konfiguraciju i mehanizme pohrane i dohvata podataka. Čvorovi su povezani izravno ("susjedni su") ako postoji mogućnost i želja uspostavljanja izravne TCP/IP veze među njima, a neizravno ako oba čvora izravno povezana na treći čvor koji ima mogućnost posredovanja informacija između prva dva čvora.

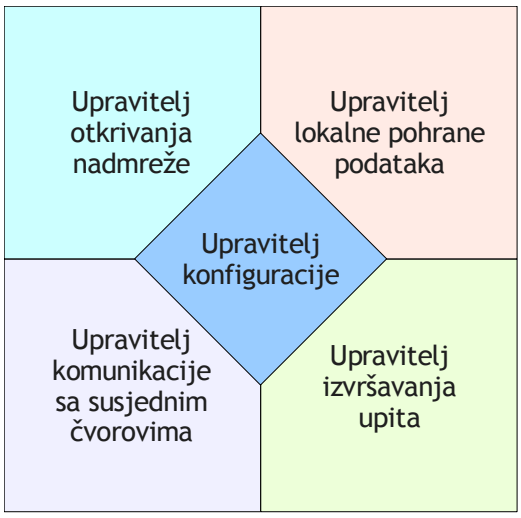
Cogent mreža je definirana kao nadmreža čvorova na kojima se izvode instance procesa Cogent vezanima uz vlastite IP adrese, odn. nadmreža Cogent čvorova. Dva čvora pripadaju Cogent mreži ako su izravno ili neizravno povezana. Drugim rječima između svaka dva čvora u Cogent mreži postoji barem neizravna povezanost. Cogent mreža koja se sastoji od samo jednog čvora je degenerirani slučaj, ali je dozvoljen.

Svaki Cogent čvor se sastoji od više modula od kojih svaki obavlja specifičnu funkciju unutar jednog čvora raspodijeljenog partnerskog sustava:

- Upravitelj konfiguracije
- Upravitelj otkrivanja nadmreže
- Upravitelj lokalne pohrane podataka
- Upravitelj komunikacije sa susjednim čvorovima u nadmreži
- Upravitelj izvođenja upita

Suradnja modula se odvija po logičnoj potrebi i može biti predstavljena slijedećim dijagramom:

1 Riječ "cogent" dolazi iz engleskog jezika gdje ima značenje "appealing to the intellect or powers of reasoning, convincing." Odabrana je kao (neslužbeni) naziv projekta jer dobro opisuje autorovo mišljenje o partnerskim mrežnim sustavima.



Slika 4. Arhitektura sustava Cogent (moduli koji su u dodiru surađuju u radu sustava)

Iako dijagram na slici 4 dobro predstavlja model međusobne suradnje modula, nisu svi moduli jednako kompleksni, niti jednake važnosti. Na primjer, upravitelj izvođenja upita je najkompleksniji modul, dok je upravitelj konfiguracije mnogo jednostavniji.

5.1. Upravitelj konfiguracije

Zadatak upravitelja konfiguracije je osiguravanje radnog okruženja u kojem se odvija rad ostalih modula sustava. Ovo je prvi modul koji se inicijalizira i pokreće.

Konfiguracijski podaci se sakupljaju iz tri moguća izvora:

- Parametri Java okruženja i operacijskog sustava
- Parametri spremljeni u konfiguracijskoj datoteci aplikacije
- Parametri upisani na naredbenom retku pri pokretanju aplikacije

U slučaju da se ista osobina može konfigurirati putem više izvora, redoslijed prioriteta određuje koja vrijednost će biti aktivna i korištena tijekom izvođenja aplikacije (konfiguracijska datoteka ima najniži prioritet, naredbeni redak najviši).

Nakon inicijalizacije upravitelja konfiguracije, moduli iz njega preuzimaju sve konfiguracijske vrijednosti, te aplikacija dalje može funkcionirati bez izravne interakcije sa mehanizmima konfiguriranja nižeg nivoa. Ova osobina dopušta moguću zamjenu konfiguracijskog upravitelja sa alternativnim upraviteljem koji konfiguracijske vrijednosti može dohvaćati iz nekog zamjenskog izvora ili koji će ih imati izravno upisane u izvorni kod (odn. neće dozvoljavati konfiguraciju).

5.1.1. Okruženje Jave i operacijskog sustava

Iz okruženja u kojem se izvodi Java proces se dohvaćaju sistemske informacije kao vrsta operacijskog sustava (Windows ili Unix), oblik putanje datoteka (znakovi za razdvajanje djelova putanje) i lokacija korisničkog direktorija (*home directory*).

5.1.2. Konfiguracijska datoteka

Podrazumjevani (*default*) smještaj konfiguracijske datoteke se određuje iz početnog korisničkog direktorija (*user home directory*) i glasi `${HOME}/.cogent/Cogent.ini`. Format i sintaksa ove datoteke su inspirirani formatom i sintaksom .INI datoteka poznatih iz operacijskog sustava Windows (detalji u poglavlju 10.1). Korištenje ovog formata za konfiguracijsku datoteku je odabrano zbog njegove velike jednostavnosti, te lake i efikasne implementacije procesiranja. Ovdje upisane konfiguracijske vrijednosti trebaju predstavljati testirane i "uobičajene" vrijednosti koje će biti iskorištene u normalnom radu sustava. Detaljna strukturu opcija je dokumentirana u poglavlju o korištenju sustava (korisnička dokumentacija).

5.1.3. Naredbeni redak

Većina vrijednosti konfigurirane putem konfiguracijske datoteke može se izmijeniti korištenjem naredbenog retka, s tim da su na ovaj način zadane vrijednosti aktivne samo za jednu instancu procesa (ne pohranjuju se). Oblik parametara na naredbenom retku je formata uobičajenog za programe u Unix okruženju: radi se o prekidačima (*switches*) koji započinju crticom iza koje slijedi jedno slovo, te opcionalno razmak i dodatne vrijednosti.

5.2. Upravitelj otkrivanja nadmreže

Zbog smanjenja potrebnih resursa na jednom Cogent čvoru ali i proučavanja dinamičnog ponašanja u partnerskih mrežama, maksimalni broj čvorova koji mogu biti susjedni jednom čvoru (odn. koje će čvor prihvatiti za susjedne) je ograničen. Ovo ograničenje je moguće konfigurirati putem konfiguracijske datoteke (opcija `max_neighbour_nodes` u grupi `system`) ili putem naredbenog redka (parametar "`-n`"). Očekivano ponašanje sustava u ovisnosti o ovom ograničenju je da će se povećanjem ograničenja povećati brzina pretraživanja nadmreže, ali uz veće opterećenje čvorova (osobito onog koji je prvi primio upit).

U sustavu Cogent implementirana su dva načina pomoću kojeg novi čvorovi mogu otkrivati postojeće čvorove u nadmreži te se tako uključiti u rad nadmreže:

- Putem korisnički zadanog popisa postojećih aktivnih čvorova
- Putem ograničenog otkrivanja čvorova na trenutnom segmentu lokalne TCP/IP mreže

Uobičajen način rada je da se ova dva načina koriste istovremeno. Ukoliko korisnik zada premali broj početnih čvorova, upravitelj otkrivanja nadmreže će pokušati pronaći dovoljni broj lokalnih Cogent čvorova da popuni kvotu. Popis čvorova se sustavu predaje putem konfiguracijske datoteke ili naredbenog retka. U konfiguracijskoj datoteci naziv opcije je initial_`neighbours` u grupi `cogent`, čemu odgovara parametar "`-i`" u naredbenom retku.

Otkrivanje čvorova na lokalnom segmentu TCP/IP mreže odvija se emitiranjem UDP mrežnih paketa nespecifirane adrese (*UDP broadcast*) sa odgovarajuće konfiguriranim mrežnim portom (isti kao za TCP). Sadržaj ovih paketa je tekstualan, u formatu nalik onom korištenom u Windows .INI datotekama. Paketi za otkrivanje postojanja i statusa lokalnih čvorova se šalju prilikom pokretanja čvora, gašenja čvora te, opcionalno, periodički tijekom rada čvora.

Zbog potrebe da radi samostalno i kontinuirano tijekom aktivnosti Cogent čvora, upravitelj otkrivanja nadmreže je implementiran u zasebnoj niti (*thread*) izvođenja, te zbog zaštite od problema zahtjeva pažnju pri pristupanju njegovim svojstvima i podacima.

5.3. Upravitelj lokalne pohrane podataka

Svaki Cogent čvor posjeduje određene lokalno pohranjene podatke koji se mogu podijeliti u dvije skupine:

- Interni podaci sustava potrebni za rad Cogent čvora
- Korisnički podaci (*payload*) koji se mogu kreirati i pretraživati

Interni podaci uključuju definicije tipova podataka te metainformacije i statistike o radu čvora i mreže, te su detaljnije opisane u poglavlju o detaljima implementacije. Korisnički podaci ovise o namjeni i funkcioniranju Cogent mreže, te na njih u većoj mjeri utječu korisničke potreba.

Obje skupine podataka se pohranjuju lokalno u obliku baze podataka HSQLDB oblika. Radi se o ugrađenoj (*embedded*) bazi podataka napisanoj u Javi koja ne zahtjeva posebno održavanje. Svi potrebni dijelovi baze podataka se inicijaliziraju pri prvom pokretanju Cogent čvora. Korištenje ugrađene baze podataka umjesto razvoja vlastitog načina pohrane podataka ima za prednosti brži razvoj aplikacije, velike mogućnosti manipulacije podacima i strukturom te veću otpornost na pogreške.

5.4. Upravitelj komunikacije sa susjednim čvorovima u nadmreži

Otkrivanje (lokalnih) čvorova u nadmreži se odvija korištenjem UDP paketa, no komunikacija između susjednih čvorova se odvija XML-RPC protokolom temeljenom na TCP-u. Radi se o otvorenom protokolu koji koristi HTTP protokol na nižem nivou za razmjenu podataka koji su serijalizirani u XML obliku. Po svojoj izvedbi, protokol služi pozivanju udaljenih procedura (*remote procedure call*). Svaki Cogent čvor je istovremeno poslužitelj i klijent za XML-RPC protokol, te zbog potrebe stalne spremnosti za komunikaciju ovaj upravitelj je implementiran kao posebna nit izvođenja.

XML-RPC je izabran zbog toga što je jednostavan za implementaciju, ali ipak nedvosmisleno definiran i vrlo velikih mogućnosti. Za razliku od nekih drugih protokola temeljenih na razmjeni podataka u XML obliku (kao što je SOAP), XML-RPC ne zahtjeva prethodnu definiciju složenih tipova podataka (strukture i nizovi) koji se prenose u pozivima i oglašavanje podržanih (odn. postojećih) funkcija i tipova podataka, ali su svi podaci i definicije funkcija strogo tipizirani. Ove osobine omogućavaju efikasno stvaranje RPC poslužitelja i klijenata te obavljanje RPC poziva koji koriste korisnički definirane strukture (strukture koje nisu unaprijed definirane pri pokretanju sustava). Kako postoji ugrađeni mehanizam za obavještavanje o pogreškama poziva i izvođenja, osigurano je da se klijent i poslužitelj razumiju, ili barem da su vrlo brzo obavješteni o razlikama u tumačenju predanih vrijednosti, što se dobro uklapa u strogo tipizirane programske jezike kao što je Java (drugim rječima, ne postoji mogućnost za "tihe" greške). Protokol ne održava stalnu TCP vezu između čvorova (*stateless* je), što smanjuje zahtjeve za resursima.

5.5. Upravitelj izvođenja upita

Upravitelj izvođenja upita je modul koji prima XML-RPC zahtjeve od drugih čvorova (odn. djeluje kao XML-RPC poslužitelj), te odgovara na njih. Zbog mogućnosti potencijalno zahtjevnog radom koji ne smije ometati ostale module ovaj modul je implementiran kao zasebna nit izvođenja u procesu. Postoje nekoliko vrsta mogućih upita:

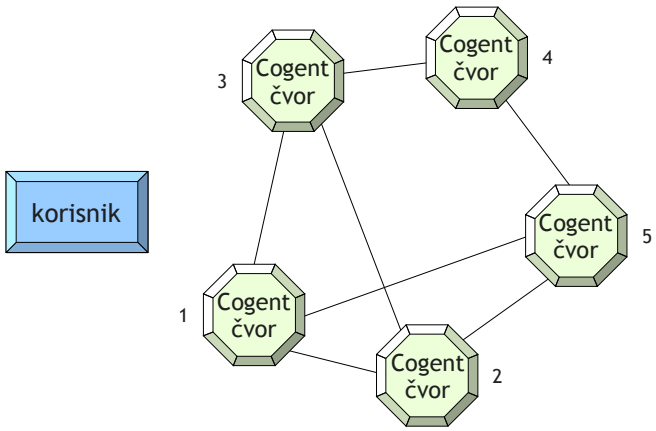
- Upit za podacima
- Upit za proslijeđivanje podataka
- Upit za izmjenu podataka
- Upit o statusu čvora

Upit za podacima je najsloženiji, jer osim sadržaja upita (u tekstualnom obliku) sadrži još informacije o tome kako i da li propagirati upit susjednim čvorovima, te u kojem obliku se očekuje odgovor. Slijedi upit za proslijeđivanje podataka, u kojem se podaci koji su rezultirali upitom prosljeđuju od trenutnog čvora ili kroz trenutni čvor prema odredišnom čvoru. Upit za izmjenu podataka je poseban po tome što se ne propagira prema drugim čvorovima, te se izmjena podataka (što uključuje i upis novih podataka) odvija samo lokalno. Upit o statusu čvora jednostavno vraća informacije o funkcioniranje čvora, kao što je broj "slobodnih mjesta" za susjedne čvorove. Opisi upita, njihovih parametara i rezultata se nalazi u poglavlju o detaljima implementacije.

5.6. Opis izvođenja upita u nadmreži

Jedna od osnovnih ideja pri stvaranju Cogent sustava bila je da pri postavljanju upita ne postoji razlika između običnih Cogent čvorova i onih putem kojih korisnici zadaju upite. Sa stajališta komunikacije između čvorova, na isti način (putem XML-RPC protokola) se predaju početni korisnički upit i upiti propagirani između čvorova. Osim po nekim indirektnim osobinama, Cogent čvorovi ne mogu razlikovati da li je upit koji dobijaju dolazi od korisnika ili od drugog Cogent čvora.

Primjer Cogent mreže koja se sastoji od 5 čvorova i sa ograničenjem na maksimalno tri susjeda je prikazan na slici 5.



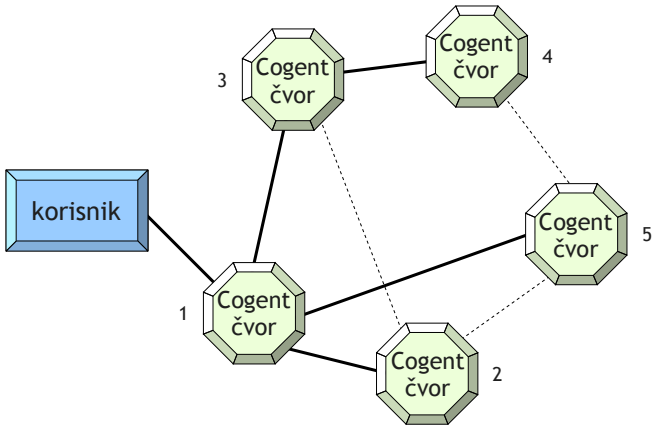
Slika 5. Primjer jednostavne Cogent mreže sa pet Cogent čvorova i neaktivnim korisnikom

Korisnik može postaviti upit na bilo koji čvor u Cogent mreži. Nakon što čvor primi uput, prvo se provjerava da li je ovaj označen za propagiranje, i ako je proslijeđuje se susjednim čvorovima. Tek nakon toga upit se izvodi lokalno i podaci iz lokalne pohrane šalju korisniku, ako postoje.

Ovaj redoslijed je izabran da omogući što ranije i što brže propagiranje upita kroz nadmrežu, te se upit može izvoditi lokalno za vrijeme dok ga drugi čvorovi primaju i obrađuju.

Rezultati upita se korisniku šalju asinkrono, kako obrade upita završavaju na udaljenim čvorovima. Implementirana su dva načina na koji se rezultati nakon izvršavanja upita dostavljaju klijentu: izravno i neizravno. Izravnim načinom svaki čvor pojedinačno kontaktira korisnika istim protokolom kao za komunikaciju između čvorova te dostavlja svoje podatke. Neizravnim načinom podaci se šalju prema korisniku istim putem (obrnutim redoslijedom čvorova) kojima je čvoru došao upit. Izbor načina dostave rezultata vrši korisnik prilikom postavljanja upita.

Kako je način stvaranja topologije nadmreže proizvoljan s obzirom na redoslijed otkrivanja čvorova, rezultirajuća topologija može imati cikluse. Da se spriječi kruženje upita pri propagiranju nadmrežom koristi se kombinacija dva mehanizma. Prvi je pridruživanje jedinstvenog identifikacijskog niza znakova svakom upitu (niz generira čvor ili korisnik koji stvara upit) koji se privremeno pohranjuje u svakom čvoru koji je upit primio. Dobije li čvor upit koji je već obradio, ignorirati će ga. Drugi mehanizam je stvaranje uređene liste čvorova putem kojih je upit došao do trenutnog čvora i distribuiranje liste zajedno s upitom. Upit neće biti proslijeđen susjednim čvorovima koji se nalaze u ovoj listi te se time izbjegava stvaranje jednostavnih petlji u propagiranju. Ovim se efektivno simulira stvaranje razapinućeg stabla (*spanning tree*) koje sadrži sve čvorove najviše jednom.



Slika 6. Propagiranje upita poslanog na čvor 1 kroz Cogent mrežu

U primjeru na slici 6 korisnik šalje upit na čvor "1", koji ga propagira svojim susjednim čvorovima "2", "3" i "5". Čvor "3" upit propagira čvoru "4" jer je to jedini njemu susjedni čvor koji još nije posjećen. Upit dolazi na čvor "4" sa listom prijeđenih čvorova koja glasi "korisnik, 1,

3" te se nakon obrade rezultat šalje čvoru "3" sa listom koja glasi "korisnik, 1", te čvoru "1" sa listom koja sadrži samo jedan element "korisnik" i na poslijetku korisniku, sa praznom listom.

Lista čvorova se koristi i u još dva slučaja: pri korištenju neizravnog načina vraćanja rezultata upita i pri ispadu međučvora u trenutku vraćanja rezultata korisniku. Kada se rezultat upita vraća korisniku putem kojim je došao, lista posjećenih čvorova se prolazi u suprotnom smjeru te se čvorovi uklanjaju iz liste pri nailasku. U slučaju da se pri neizravnom vraćanju rezultata neki od čvorova iz liste ne može pronaći (npr. ako je čvor istupio iz mreže ili se dogodila pogreška u radu mreže), isprobavaju se slijedeći čvorovi iz liste dok se ne nađe aktivni čvor ili dok se lista ne isprazni (u kojem slučaju se rezultat odbacuje).

5.7. Opis podataka koji se pohranjuju i dohvaćaju u nadmreži

Podaci koji se pohranjuju i dohvaćaju u Cogent nadmreži su dinamičko stvoreni riječnici (*dictionary*) koji imaju znakovne nizove za ključeva i vrijednosti koje mogu biti znakovne, cjelobrojne ili brojevi sa pomičnim decimalnim zarezom. Rječnici imaju svoje "tipove." Tip rječnika određuje koje sve ključeve rječnik posjeduje te kojeg tipa su vrijednosti pojedinih ključeva. Ključevi i nazivi tipova rječnika su imenovani nizom znakova koji može sadržavati alfanumeričke i ograničene interpunkcijske znakove, te se ne razlikuju po veličini slova. Svaki čvor sadrži proizvoljan broj rječnika proizvoljnih tipova, te ne mora sadržavati rječnike svakog tipa.

```
termo_zapis
     lokacija    : string
     vrijeme     : integer
     temperatura : float

status_uredjaja
     naziv       : string
     status      : integer
```

Slika 7. Primjer tipova rječnika "termo_zapis" i "status_uredjaja"

Upiti za podacima služe dohvaćanju jednog ili više rječnika istog tipa. Da bi korisnik uspješno dohvatio rječnike mora prethodno biti upoznat sa dostupnim tipovima rječnika, ali svaki čvor može vratiti popis tipova rječnika koje sadrži. Dobije li čvor upit za podacima rječnika nepoznatog tipa, upit će biti ignoriran.

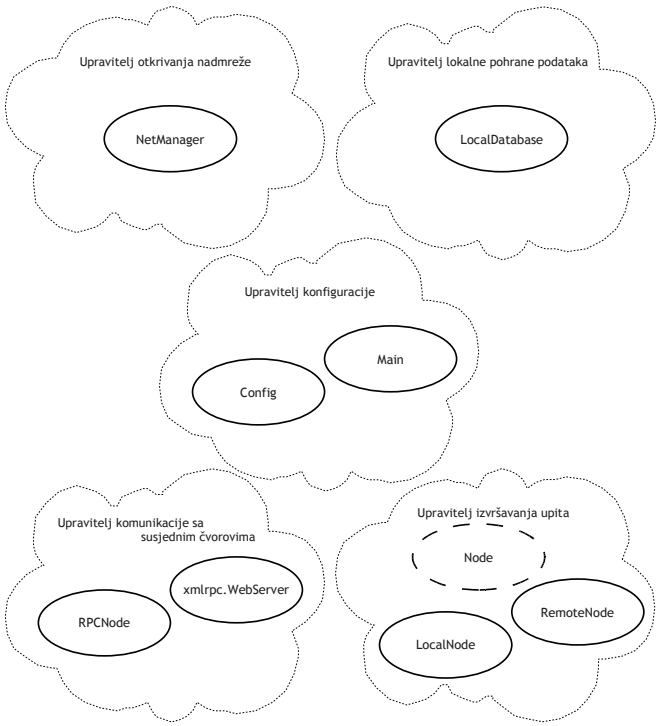
6. Detalji implementacije

Sustav Cogent implementiran je u programskom jeziku Javi, u okruženju Java 1.5. Korišten je razvojni sustav NetBeans5, te je struktura direktorija i način prevođenja programa prilagođen ovom sustavu. Korisna osobina NetBeans direktorija sa projektom je što projekt moguće prevoditi i izvan sustava standardnim alatom za prevođenje Java projekata "`ant`."

`/Cogent`	Osnovni direktorij projekta Cogent
`/build`	Sadrži prevedene datoteke klasa (privremeni direktorij)
`/dist`	Sadrži prevedeni projekt zapakiran u .jar datoteku i potrebne dodatne biblioteke za izvođenje projekta (privremeni direktorij)
`/etc`	Sadrži primjere konfiguracijskih datoteka
`/libs`	Sadrži dodatne biblioteke potrebne za prevođenje i izvođenje projekta
`/nbproject`	Sadrži opis NetBeans projekta (interne datoteke koje koristi NetBeans)
`/src`	Osnovni direktorij za izvorni kôd
`/cogent`	Java paket "cogent" koji sadrži izvorni kôd sustava

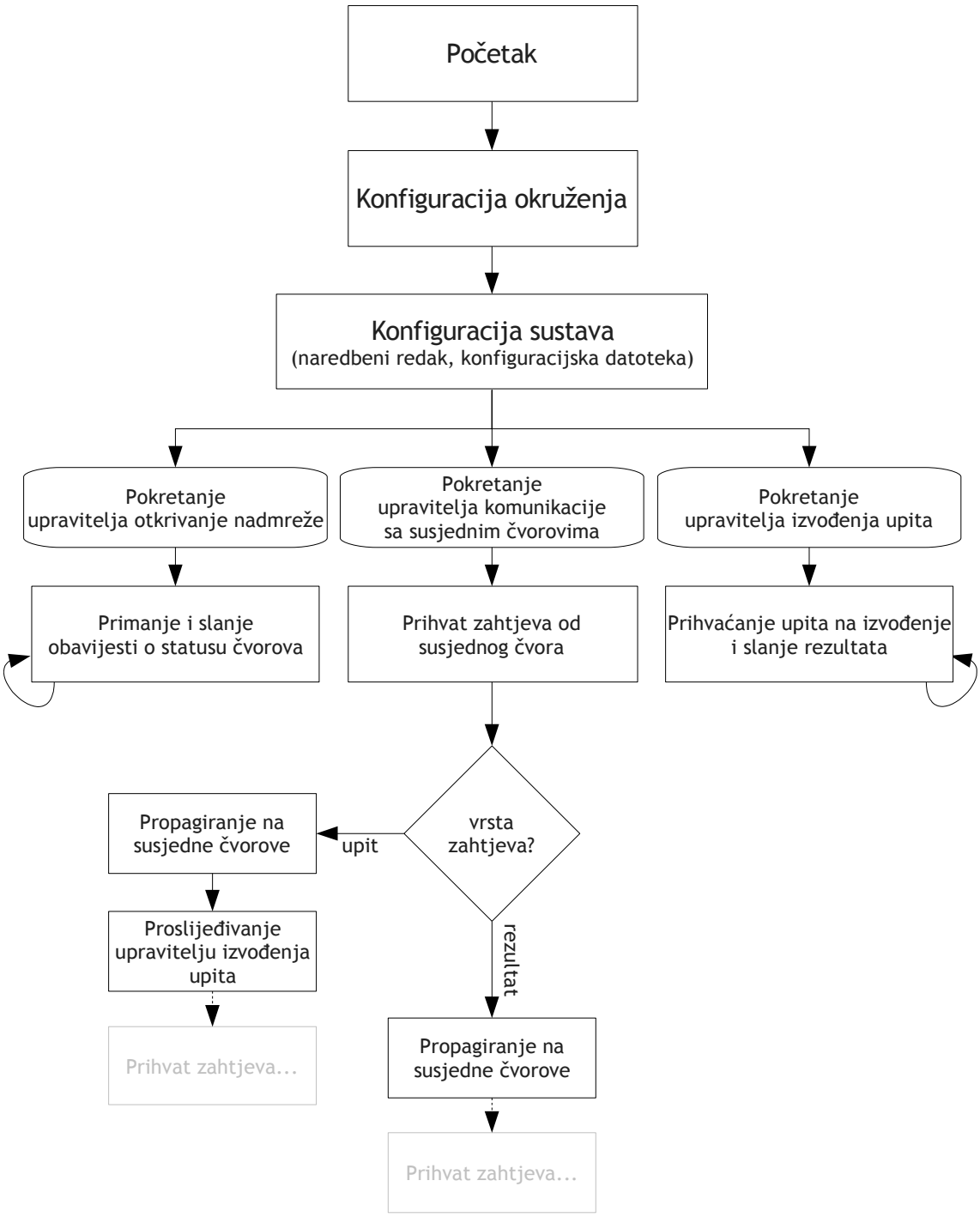
Slika 8. Opis direktorija prisutnih u razvojnom okruženju

Klase koje čine projekt Cogent su prikazane na slici 9:



Slika 9. Java klase koje čine jezgru modula sustava Cogent

Na slici 10 prikazana je shema rada instance Cogent procesa na jednom čvoru, od pokretanja procesa do stabilnog stanja u kojem upravitelji obavljaju svoje zadatke u u petlji.



Slika 10. Shema rada Cogent čvora

Kako se tri aktivna modula (upravitelji otkrivanja nadmreže, komunikacije sa susjednim čvorovima i izvođenja upita) izvode u zasebnim nitima izvršavanja, osnovna nit procesa ostaje slobodna. Za potrebe testiranja implementirano je malo sučelje u obliku rudimentarnog

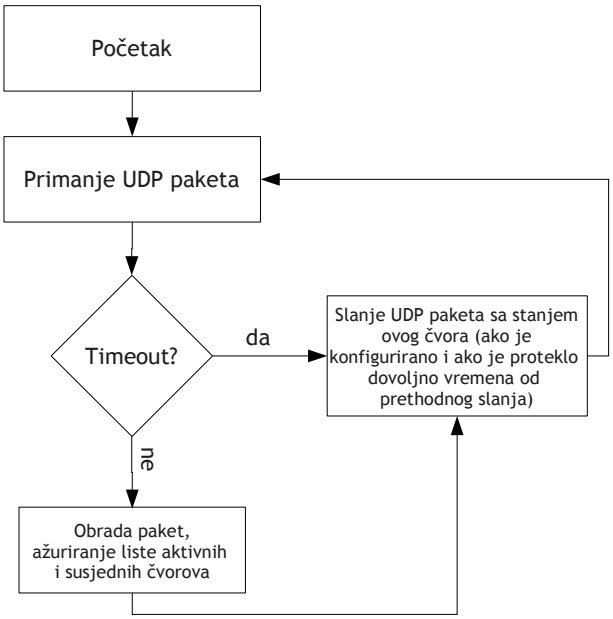
naredbenog redka unutar aplikacije koje se može koristiti za ispitivanje stanja čvora. Ovo sučelje se koristi samo ako je uključeno putem naredbenog redka pri pokretanju čvora.

6.1. Opis rada modula

Podjelom na module u zasebnim nitima izvršavanja osigurano je izvođenje svih važnih dijelova Cogent čvora neovisno o drugim komponentama, te je sva kompleksnost modula lokalizirana unutar njihovog vlastitog izvršnog konteksta. Za razmjenu podataka između modula koristi se sinkronizacijski mehanizmi ugrađeni u jezik.

6.1.1. Upravitelj otkrivanja nadmreže

Rad upravitelja otkrivanja nadmreže se sastoji od jedne petlje u kojoj se čeka na dolazak UDP paketa sa informacijama od drugih čvorova, uz vremensko ograničenje (*timeout*). U slučaju da je konfigurirano automatsko slanje obavijesti o statusu trenutnog čvora, stvara se i na nespecifiranu adresu šalje UDP paket sa informacijama oblikovanim u jednostavan format nalik Windows .INI datotekama. Na slici 11 je prikazana shema rada ovog modula.



Slika 11. Shema rada upravitelja otkrivanja nadmreže

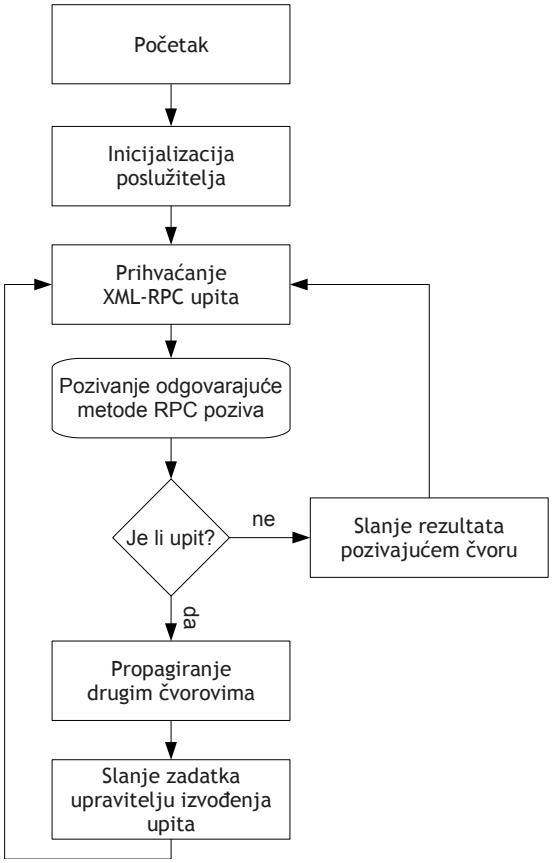
Vrste UDP paketa koji se koriste u oglašavanju aktivnosti Cogent čvorova su:

- Oglašavanje aktivnosti čvora
- Upit o aktivnosti drugih čvorova u mreži
- Oglašavanje prestanka aktivnosti čvora

Svaki Cogent čvor šalje i prima sve navedene vrste paketa.

6.1.2. Upravitelj komunikacije sa susjednim čvorovima

Namjena upravitelja komunikacije sa susjednim čvorovima je uspostavljanje XML-RPC poslužitelja u svakom čvoru, te izvršavanje RPC poziva. Nakon inicijaliziranja poslužitelja rad modula je jednostavna petlja koja prihvaća TCP veze, implementira HTTP te XML-RPC protokole te poziva odgovarajuće metode koje obavljaju traženu funkcionalnost.

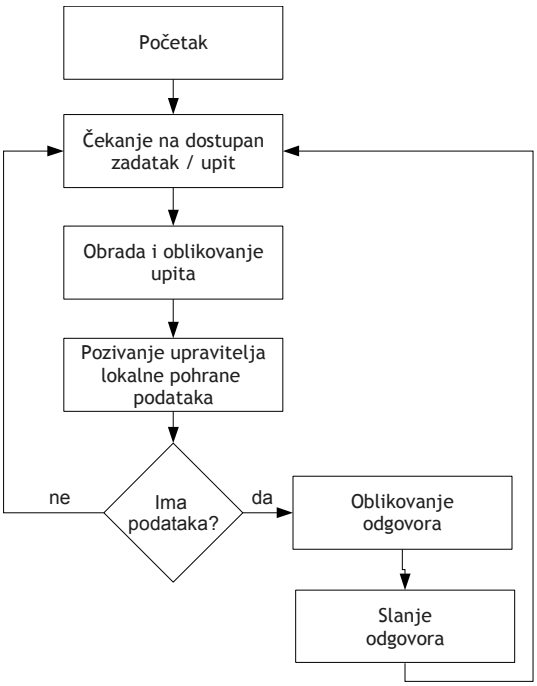


Slika 12. Shema rada upravitelja komunikacije sa susjednim čvorovima

RPC pozivi koji zahtjevaju izvođenje upita ne izvode upite samostalno, već ih (nakon mogućeg slanja na propagiranje mrežom) predaje na izvršavanje upravitelju izvođenja upita.

6.1.3. Upravitelj izvođenja upita

Upravitelj izvođenja upita posjeduje listu zadataka koje obrađuje, iz koje uzima jedan po jedan zadatak na izvršavanje. Zadaci su upiti koje je čvor dobio putem upravitelja komunikacije sa susjednim čvorovima.



Slika 13. Shema rada upravitelja izvođenja upita

Upiti se pri primitku preoblikuju na način koji je prikladan za dohvat podataka od upravitelja lokalne pohrane podataka te se poziva ovaj upravitelj. Pronađeni podaci se oblikuju prikladno za slanje odgovora te se naposlijetku odgovor šalje čvoru koji je poslao upit (ili korisniku).

6.2. Opisi protokola, funkcija i struktura podataka

U radu sustava Cogent koristi se nekoliko protokola, funkcija i struktura podataka, koji su dokumentirani u ovom poglavlju za referencu pri nadogradnji projekta i za korištenje u projektima koji trebaju biti kompatibilni s njim.

6.2.1. UDP paketi korišteni pri otkrivanju nadmreže

Za otkrivanje nadmreže koristi se slanje UDP mrežnih paketa na nespecifiranu adresu (*broadcast*) koje primaju svi Cogent čvorovi na lokalnom segmentu TCP/IP mreže. Za smanjenje problema u prijenosu ovi paketi moraju biti ograničeni u veličini, najčešće na 1400 okteta. Sadržaj paketa je niz znakova kodiranih u UTF-8 obliku [5] strukturiranih u formatu nalik onom korištenom za Windows ".INI" datoteke (format je opisan u poglavlju 10.1).

Paketa za oglašavanje aktivnosti čvora služi ažuriranju liste aktivnih čvorova tijekom rada nadmreže; njegova struktura je prikazana na slici 14:

```
[cogent]
type=announce
node_address=<ip_address>
```

```
start_time=<time>
free_neighbour_slots=<n>
```

Slika 14. Struktura paketa za oglašavanje aktivnosti čvora

Vrijednost `<ip_address>` je IP adresa (za verziju protokola IPv4) čvora (koja se mora slagati sa adresom pošiljatelja čvora), a `<time>` je cijeli broj koji sadrži oznaku vremena (*time stamp*) u Unix obliku (broj sekundi od ponoći 1.1.1970.). Vrijednost označena sa <n> je cijeli broj koji sadrži broj slobodnih mjesta za susjedne čvorove. Procjenjuje se da će veličina ovog paketa u primjeni biti oko 70 okteta.

Paket koji sadrži upit o aktivnosti drugih čvorova u nadmreži šalju čvorovi koji žele saznati status drugih aktivnih čvorova na mreži. Čvorovi koji prime ovaj paket odgovaraju izravno pošiljatelju (ne koristi se nespecifirana adresa) da bi se smanjilo opterećenje mreže. Struktura ovog paketa (duljine otprilike 24 okteta) je prikazana na slici 15:

```
[cogent]
type=whoisthere
```

Slika 15. Struktura paketa za upit o aktivnosti čvorova u nadmreži

Oglašavanje prestanka aktivnosti čvora uzrokuje ažuriranje lista aktivnih i susjednih čvorova u Cogent čvorovima koje prime paket sa strukturom prikazanom na slici 16 (duljine otprilike 50 okteta):

```
[cogent]
type=offline
node_address=<ip_address>
```

Slika 16. Struktura paketa za oglašavanje prestanka aktivnosti čvora

6.2.2. Opis RPC funkcija čvorova

Svaki Cogent čvor nudi na izvršavanje putem XML-RPC protokola funkcije čijim pozivanjem se obavljaju sve važnije operacije na čvorovima i nadmreži. Pozivanje svih funkcija je dozvoljeno od strane čvorova ili od vanjskih klijenata koji koriste usluge nadmreže. RPC funkcije sustava Cogent su smještene u imenski prostor (*namespace*) "cogent." U okviru XML-RPC protokola (koji je nadgradnja HTTP protokola), sve funkcije trebaju biti posluživane sa putanje "/".

Uz čvorove koji sudjeluju u Cogent nadmreži, klijentski sustav koji postavlja upit mora postaviti XML-RPC poslužitelj koji nudi samo jednu funkciju, `processResponse()` koja prihvaća rezultat upita.

Slijede definicije funkcije u obliku nalik onome koji se koristi za deklaracije u jeziku C, ali sa korištenjem tipova podataka koji su standardizirani u specifikaciji za XML-RPC (na primjer: "struct" označava dinamičku strukturu odn. rječnik).

```
struct cogent.getNodeStatus();
```

Vraća rječnik sa statističkim podacima o rada čvora. Podaci u rječniku su:

- `int start_time` – Vrijeme početka rada čvora, u Unix obliku
- `int max_neighbours` – Maksimalni broj čvorova koje ovaj čvor može imati za susjede
- `int num_neighbours` – Broj čvorova koje čvor smatra susjednima
- `int num_rpc` – Broj RPC poziva koje je čvor obradio
- `int num_queries` – Broj upita koje je čvor obradio (odnosi se samo na upite o podacima, ne uključuje druge RPC pozive)

```
array cogent.getSupportedTypes();
```

Vraća niz znakovnih nizova (*array of strings*) sa imenima tipova rječnika koje ovaj čvor posjeduje.

```
struct cogent.getTypeDetails(string type_name);
```

Vraća definiciju tipa rječnika u obliku strukture (rječnika) čiji su ključevi isti kao ključevi tipa rječnika a vrijednosti nazivi tipovi pojedinih polja (string, integer ili float). Parametar `type_name` je naziv tipa rječnika.

```
boolean cogent.addType(string type_name, struct definition);
```

Dodaje novi tip rječnika u čvor. Parametar type_name je naziv tip rječnika, a definition definicija rječnika, rječnik čiji su ključevi isti kao ključevi tipa rječnika a vrijednosti nazivi tipova pojedinih polja (string, integer ili float). Vraća *true* ako je dodavanje uspjelo a *false* ako nije (u slučaju da tip rječnika već postoji).

```
boolean cogent.addData(string type_name, struct data);
```

Dodaje podatke u lokalnu pohranu. Dodavanje podataka na ovaj način se ne propagira na druge čvorove Cogent mreže. Parametri funkcije su:

- `type_name` je naziv tipa rječnika za koji se dodaju podaci
- `data` je rječnik odgovarajućeg tipa koji sadrži podatke za dodavanje

Funkcija vraća *true* ako je dodavanje podataka uspjelo, *false* ako nije.

```
boolean cogent.fullQuery(string type_name, string filter, string query_id,
    boolean propagate, array return_path, boolean direct_return);
```

Ova funkcija služi slanju upita u Cogent mrežu. Parametri su:

- `type_name` je naziv tipa rječnika na koji se upit odnosi
- `filter` je filterski izraz koji određuje koji podaci će se vratiti (može biti prazan, u kojem slučaju se vraćaju svi podaci)
- `query_id` je jedinstveni identifikator upita, proizvoljan niz alfanumeričkih znakova koje stvara korisnički program koji postavlja upit
- `propagate` određuje da li će se upit propagirati na susjedne čvorove
- `return_path` je uređena lista adresa čvorova (IP adresa kao niz znakova) putem kojih je upit stigao do trenutnog čvora. U slučaju prvog čvora ova lista mora sadržavati jedan element – IP adresu klijenta koji prima rezultat.
- `direct_return` određuje da li će se rezultat vratiti izravno čvoru (ili korisniku) koji je inicirao upit, ili će se rezultat slati istim putem kojim je upit došao (iz parametra return_path)

Rezultat funkcije je *true* ako je upit ispravan i poslan na izvršavanje, *false* ako se dogodila pogreška.

Ova funkcija nudi sve mogućnosti slanja upita u Cogent mrežu. Koristi se i interno pri propagiranju upita kroz mrežu.

```
boolean cogent.query(string type_name, string filter, string query_id);
```

Funkcija služi jednostavnom slanju upita u Cogent mrežu, i namijenjena je za korištenje umjesto funkcije `fullQuery()` zbog jednostavnosti. Dodatni parametri koji se predaju u `fullQuery()` a ne postoje u `query()` su zamijenjeni podrazumjevanim vrijednostima:

- `propagate` je *true*
- `return_path` sadrži IP adresu pošiljatelja upita
- `direct_return` je *true*

Rezultat funkcije je *true* ako je upit ispravan i poslan na izvršavanje, *false* ako se dogodila pogreška.

`boolean cogent.processResult(string query_id, array result, array return_path)`
Namjena ove funkcije nije pozivanje od strane korisnika, već suprotno – ovu funkciju pozivaju čvorovi koji su završili sa izvođenjem upita na čvorovima koji su upit poslali. Svaki čvor koji može poslati (ili propagirati) upit treba implementirati ovu funkciju putem koje će primiti rezultat upita. Parametri funkcije su: • `query_id` je jedinstveni identifikator upita kojeg je na proizvoljan način stvorio klijentski program koji je postavio upit • `result` je lista rječnika koji su rezultat upita, odn. koji odgovaraju filterskom izrazu upita • `return_path` je lista znakovnih IP adresa čvorova putem kojih je upit stigao do čvora koji šalje rezultat. Ako lista sadrži više od jednog elementa, rezultat se propagira slijedećem elementu iz liste, te taj element uklanja Funkcija vraća *true* ako je rezultat uspješno primljen, *false* ako nije.

Čvorovi Cogent nadmreže međusobno surađuju pomoću navedenih funkcija. Korisnički program koji postavlja upit u nadmrežu može postaviti svoj upit na bilo koji čvor nadmreže te će ovaj (uz uvjet da je tako zatraženo) biti propagiran do drugih čvorova. Rezultati upita će korisničkom programu stići putem poziva funkcije `processResult()` koju ovaj mora implementirati.

7. Korisnička dokumentacija

Postavljanje Cogent sustava uključuje kopiranje datoteka sa distribucijom sustava te opcionalno stvaranje i izmjena konfiguracijske datoteke. Struktura direktorija sa datotekama sustava i biblioteka potrebnih za izvršavanje je prikazana na slici 17:

`Cogent.jar`	Izvršna datoteka sustava sa prevedenim Java kodom
`lib/`	Direktorij sa dodatnim bibliotekama potrebnim za rad
`lib/hsqldb.jar`	Biblioteka HSQLDB sustava
`lib/java-getopt-1.0.12.jar`	Biblioteka *Getopt* za analizu parametara naredbenog retka
`lib/protomatter-1.1.8.jar`	Biblioteka *Protomatter* koja sadrži podršku za praćenje rada aplikacija zapisivanjem poruka (*logger*)
`lib/commons-codec-1.3.jar`	Biblioteka za podršku
`lib/xmlrpc-2.0.jar`	Biblioteka sa XML-RPC podrškom
`lib/ini4j-1.0.jar`	Biblioteka sa podrškom za učitavanje i zapisivanje podataka u obliku inspiriranom Windows .INI formatom

Slika 17. Biblioteke potrebne za rad sustava Cogent

Pokretanje instance sustava (odn. jednog čvora) se izvodi pokretanjem interpretera Java jezika sa opcijom izvršavanja kôda iz biblioteke `Cogent.jar`.

7.1. Parametri naredbenog redka

Naredba kojom se pokreće Cogent sustav odn. čvor je slijedećeg oblika:

```
java -jar Cogent.jar <parametri>
```

Parametri koji se mogu koristiti pri pokretanju su:

`-a <ip_address>`	IP adresa koju će koristiti ovaj čvor (*bind address*)
`-p <port>`	Mrežni port kojeg će koristiti ovaj čvor (uobičajeno 8888).
`-c`	Uključuje pokretanje jednostavnog sučelja za ispitivanje rada instance Cogent čvora (*debug console*)
`-d <directory>`	Direktorij koji će čvor koristiti za konfiguracijsku datoteku, lokalnu pohranu podataka i zapise o radu sustava (*log*)

`-n <max_nodes>`	Maksimalni broj susjednih čvorova
`-v`	Uključuje ispisivanje veće količine informacija u zapise o radu sustava (*log*) i uključuje ispis ovih informacija na terminal na kojem je sustav pokrenut (na *stderr*)

Većina parametara koji se mogu konfigurirati putem naredbenog retka se mogu konfigurirati u konfiguracijskog datoteci. Parametri u naredbenom retku imaju prednost nad konfiguracijskom datotekom.

7.2. Konfiguracijska datoteka

Konfiguracijska datoteka koju koristi Cogent sustav je u formatu inspiriranom Windows .INI datotekama, te može sadržavati slijedeće parametre:

Konfiguracijska opcija	**Tip opcije**	**Opis**
Grupa: `[system]`		
verbose	boolean (“true” ili “false”)	Određuje da li ispisivati dodatne informacije o radu sustava u svrhu analiziranja sustava ili otklanjanja pogrešaka (inicijalna vrijednost=”false”)
local_dir	string	Direktorij u kojem će biti pohranjeni podaci vezani za rad Cogent čvora
network_port	integer	Mrežni port za korištenje u TCP i UDP protokolima koji će Cogent čvor koristiti. Svi Cogent čvorovi u jednoj nadmreži moraju koristiti isti mrežni port. (inicijalna vrijednost=8888)
max_neighbour_nodes	integer	Maksimalni broj susjednih čvorova ovom čvoru (inicijalna vrijednost=3)
Grupa: [cogent]		
broadcast_interval	integer	Interval (u sekundama) kojem će stanje Cogent čvora biti slano putem UDP *brodacast* paketa na lokalnoj mreži (0 = nikada)

initial_neighbours	string	Lista IP adresa, odvojenih zarezima, od Cogent čvorova koji će činiti inicijalne susjede trenutnom čvoru. Cogent čvorovi mogu odbiti susjedstvo u slučaju da već imaju maksimalni broj susjeda.

7.3. Opis upita

Svaki upit za podacima poslan u Cogent mrežu imaju dva dijela:

- Naziv tipa rječnika na koji se odnosi
- Filterski izraz

Naziv tipa rječnika je običan niz znakova koji može sadržavati alfanumeričke znakove te znakove "_" i "!". Filterski izraz je logički izraz koji određuje koji podaci se žele dohvatiti, te sadrži operacije koje uključuju ključeve rječnika i korisnički zadane konstante. Izraz može sadržavati proizvoljan broj ključeva, operacije usporedbe, logičke operacije "i" i "ili", te zagrade. Sintaksa filterskih izraza je uobičajena *infix* notacija. Primjeri filterskih izraza (za tipove rječnika iz primjera u slici 7) su:

```
1. lokacija="C7-11" and vrijeme>1149813203
2. temperatura<13 and (lokacija="D255" or
   lokacija="D245")
3. status=65535
```

Slika 18. Primjeri filterskih izraza

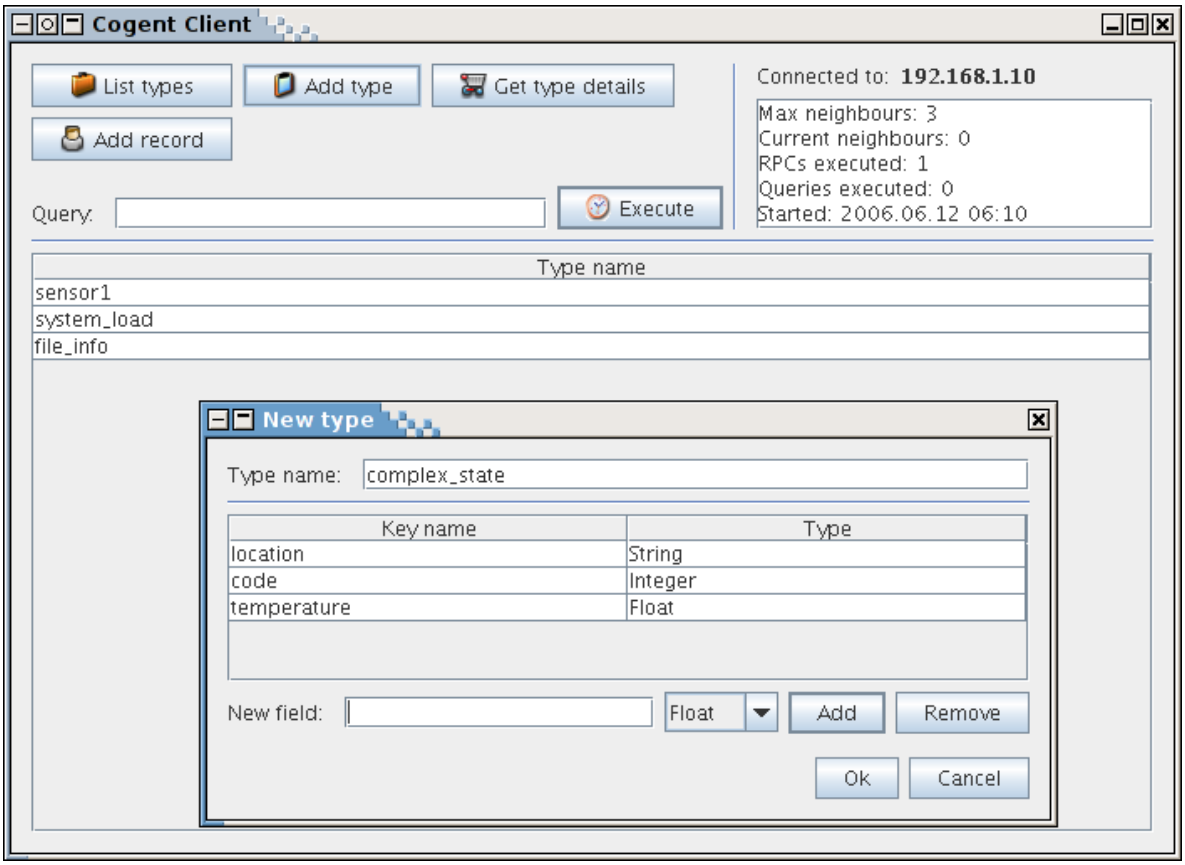
Filterski izrazi mogu koristiti slijedeće operatore i podizraze:

- `(, )` - Zagrade
- `NOT` - Operacija negacije logičkih izraza
- <, >, <=, >=, = - Operatori uspoređivanja
- `AND`, `OR` - Operatori logičkih izraza za kombiniranje podizraza

Filterski izrazi određuju koji će se rječnici vratiti čvoru ili korisniku koju postavlja upit kao rezultati upita.

7.4. Klijentska aplikacija

Za testiranje i razvoj sustava Cogent načinjena je jednostavna grafička klijentska aplikacija koja se povezuje na proizvoljno zadani čvor ("trenutni čvor") te omogućava dodavanje i dohvat podataka i tipova. Primjer sučelja aplikacije prikazan je na slici 19.



Slika 19. Sučelje klijentske aplikacije

Elementi grafičkog sučelja aplikacije su:

- "List types" - prikazuje popis dostupnih tipova rječnika koji su prisutni na trenutnom čvoru
- "Add type" - prikazuje novi dijaloški prozor za dodavanje tipa rječnika (prikazano na slici)
- "Get type details" - dohvaća i prikazuje definiciju odabranog tipa rječnika
- "Add records" - dodaje podatak određenog tipa u trenutni čvor
- "Execute" - Izvršava upit unesen u polje "Query"

Aplikacija je napisana u Javi te koristi SWING grafičko sučelje.

8. Predloženi načini korištenja

Najznačajniji razlozi za izgradnju sustava kao što je Cogent – raspodijeljena baza podataka koja koristi partnerski način rada neovisnih čvorova – su smještanje na, i dohvat podataka sa, računala na kojima su ti podaci stvoreni ili na kojima su po raznim kriterijima "bitni". Kako su svi čvorovi nadmreže povezani u nadmrežu, upit postavljen u bilo kojem čvoru će vratiti podatke koji se nalaze na bilo kojem čvoru, te je tako odvojeno pretraživanje podataka od lokacije na kojem su podaci pohranjeni.

Jedan od mogućih načina korištenja sustava je u mreži čvorova sa senzorima koji su smješteni u prostoru, gdje svaki čvor sakuplja informacije i pohranjuje ih lokalno. Povezivanjem čvorova u nadmrežu se omogućuje postavljanje jednog upita koji dohvaća informacije o podacima sa svih čvorova. U ovom slučaju senzori ne moraju biti fizikalni senzori već bilo kakvi izvori podataka, na primjer oni dobiveni nadzorom stanja računala (količina slobodne memorije, opterećenje procesora i sl.).

Neki drugi načini korištenja sustava mogu biti: pohrana velike količine podataka koja ne može biti pohranjena u samo jedan čvor te pohrana kompleksnih informacija ili izvršavanje kompleksnih upita koje traje predugo kada bi se koristilo samo jedno računalo.

8.1. Daljnji razvoj

Postoje efektivno dva smjera daljnjeg razvoja sustava Cogent: povećanje mogućnosti pohranjivanja podataka i kompleksnosti upita te povećanje mogućnosti korištenja i ugradnje u kompleksnije sustave koji su izvori ili korisnici podataka.

Efikasnost pohrane podataka se može povećati uvođenjem dodatnih informacija o rječnicima koji omogućuju optimizacije kao što je indeksiranje podataka. Dodavanje složenih mogućnosti filtriranja u upitima bi moglo dozvoliti upite koji dohvaćaju kompleksne vrste podataka ili podatke u nekom međuodnosu između čvorova.

Za praktičnu primjenu sustava moguće ga je ili ugraditi u neki veći sustav koji služi kao izvor podataka za čvor ili proširiti sa izvorima podataka. Trenutno se koristi nekoliko biblioteka koje implementiraju različite mogućnosti sustava, čija je veličina otprilike 1.1MB. Za ugradnju u računala sa manjim kapacitetima moguće je smanjiti količinu dodatnog koda potrebnih za rad sustava (uključujući i premještanje kôda iz biblioteka u osnovnu arhivu sustava) na ispod 500kB.

9. Zaključak

Partnerske mreže nude jedinstveni način raspodijeljene pohrane i dohvata podataka, uz neke jedinstvene probleme.

Povijesno, namjena partnerskih sustava i mreža je bila ograničena na razmjenu datoteka, te je i danas najveći broj korištenih partnerskih sustava ovog tipa. Do ideje modernih sustava se došlo postepeno, evolucijom postojećih sustava koji su bili korišteni za razmjenu korisničkih informacija između velikog broja računala (kao što su e-mail sustavi, Usenet i IRC sustavi). Prvi široko popularni partnerski sustav bio je Napster, no njegova arhitektura je bila centralizirana što je doprinijelo njegovom brzom gašenju. Ono što je ipak bitno kod Napstera je da su datoteke između pojedinačnih klijenata razmjenjivane izravno, na način koji podsjeća na "čiste" partnerske sustave. Slijedeći sustavi su nastojali izbjeći oslanjanje na centralizirane sustave i poznavanje unaprijed prisutnih čvorova primarno radi povećanja otpornosti mreže na ispade pojedinačnih čvorova. Unatoč tome, teoretski čiste partnerske mreže nisu raširene jer se moraju oslanjati na postojeće servise TCP/IP mreža, kao što su DNS poslužitelji. Međusobno povezani čvorovi partnerskog sustava čine nadmrežu koja ne ovisi o njihovim fizičkim lokacijama i topologiji.

U ovom radu predložena je arhitektura raspodijeljenog partnerskog sustava sa potpuno neovisnim čvorovima, te je implementiran odgovarajući prototip. Otkrivanje susjednih čvorova u nadmreži odn. izgradnja topologije nadmreže se u ovom sustavu odvija na dva načina: sa korisnički definiranim susjednih čvorova i ograničenim dinamičkim otkrivanjem susjeda na lokalnom TCP/IP segmentu mreže slanjem UDP mrežnih paketa na nespecifiranu adresu (*broadcast*). Komunikacija između čvorova se odvija putem XML-RPC protokola. Podaci koji se pohranjuju, pretražuju i razmjenjuju su strukturirani u rječnike, a upiti za podacima sadrže filterske izraze koji određuju koji podaci će uključeni u rezultat upita. Upiti koji stignu na jedan čvor se propagiraju do ostalih čvorova u nadmreži, te se rezultati pojedinačnih čvorova vraćaju čvoru ili klijentu koji je upit postavio.

Ovakva partnerska mreža može biti korištena na nekoliko načina, među kojima su pohrana i prikupljanje informacija od digitalnih senzora postavljenih na različitim lokacijama i povezanih u odgovarajuću mrežu, te u slučaju da je količina informacija prevelika za pohranu na jednom čvoru ili je izvršavanje upita prekompleksno. Raspodijeljene partnerske mreže posjeduju veliku skalabilnost kapaciteta za pohranu podataka i procesorske snage, te su stalni predmet izučavanja.

10. Dodatci

U ovom poglavlju su opisane neke od pomoćnih tehnologija korištenih u izradi rada.

10.1. Format zapisa podataka inspiriran Windows ".INI" formatom

Podaci u ovom formatu su niz znakova u UTF-8 kodiranju oblikovanih na slijedeći način:

- Sadržaj se interpretira kao niz znakova strukturiranih u redke koji su odvojeni znakovima "\n", "\r" ili "\r\n" (u notaciji korištenoj kod jezika C)
- Znakovi praznog prostora (*whitespace*) ne utječu na strukturu i sadržaj redaka
- Komentari započinju znakom ";" koji se može pojaviti bilo gdje unutar redka (osim ako se pojavi unutar navodnicima odijeljenog niza, gdje nema značenje započinjanja komentara) i traju do kraja redka. Komentari se smatraju znakovima praznog prostora.
- Redci sadržavaju ili podatke u obliku "naziv=vrijednost" ili oznaku početka grupe vrijednosti u obliku "[naziv_grupe]"
- Nazivi vrijednosti su nizovi znakova iz skupa alfanumeričkih znakova i znakova "_", "-", ".", ".". Vrijednosti su proizvoljni nizovi znakova čija definicija ovisi o konkretnoj primjeni
- Grupe vrijednosti grupiraju niz redaka koji sadrže podatke. Nazivi grupe vrijednosti mogu sadržavati znakove
- Mora postojati barem jedna grupa vrijednosti

10.2. Protokol XML-RPC

XML-RPC je protokol namijenjen pozivanju udaljenih procedura / funkcija nastao 1999. iz potrebe za standardiziranim načinom komuniciranja između procesa na različitim računalima uz korištenje tehnologija zasnovanim na HTTP i XML standardima [6]. Protokol je nastao iz potrebe za otvorenim rješenjem koje je jednostavno, robusno i upotrebljivo u vrijeme kada nije bilo odgovarajućeg standarda a većina napora se odvijala u smjeru izrade nestandardnih vlasničkih rješenja. Iako je kasnije po uzoru na XML-RPC standardiziran protokol SOAP, ovaj protokol zbog svoje kompleksnosti nije bio dobro primljen među Open-source projektima, te je XML-RPC protokol ostao popularan i korišten do danas.

Komunikacija XML-RPC-om se odvija između poslužitelja i klijenata, transportom poruka u XML obliku putem mehanizama protokola HTTP. U XML porukama su podaci sa strogo definiranim tipovima, te informacije o nazivima udaljenih procedura koje se pozivaju i

pogreškama koji nastanu u izvođenju procedura. Jednostavni tipovi podataka podržanih u protokolu su:

- "`int`" ili "`i4`" : predznačeni 32-bitni cjelobrojni tip
- "`boolean`" : logička vrijednost (istina / neistina)
- "`string`" : niz znakova
- "`double`" : broj u IEEE 754 obliku sa pomičnim zarezom (64-bitna)
- "`dateTime.iso8601`" : datum i/ili vrijeme
- "`base64`" : proizvoljna binarna vrijednost kodirana po *base64* standardu

Jednostavni tipovi podataka se mogu grupirati u složene tipove na dva načina:

- "`struct`" : rječnik (*dictionary*) koji sadrži nizove znakova za ključeve i proizvoljne vrijednosti
- "`array`" : niz odn. lista koja sadrži proizvoljan broj proizvoljnih vrijednosti

Složeni tipovi podataka (*struct* i *array*) mogu sadržavati bilo koje druge vrijednosti, uključujući i druge složene vrijednosti.

Primjer XML-RPC poziva, uključujući i HTTP zaglavlja je prikazan na slici 19.

```
POST / HTTP/1.0
User-Agent: xmlrpclib/0.7 (FreeBSD)
Content-Type: text/xml
Content-length: 181

<?xml version="1.0"?>
<methodCall>
   <methodName>examples.getStateName</methodName>
   <params>
      <param>
         <value><int>41</int></value>
      </param>
   </params>
</methodCall>
```

Slika 20. Primjer XML-RPC poziva

Primjer na slici 19 je poziv XML-RPC funkcije "`examples.getStateName`" koja prima jedan cjelobrojni parametar.

10.3. Baza podataka HSQLDB

Čvorovi sustava Cogent koriste bazu podataka HSQLDB za lokalnu pohranu podataka. HSQLDB je jednostavna baza podataka napisana u potpunosti u Javi te namijenjena ugrađivanju u veće programe bez potrebe za posebnim poslužiteljem i održavanjem [7]. Podržane su osnovne mogućnosti relacijskih baza podataka i jezika SQL, uz standardno sučelje za korištenje – JDBC.

11. Literatura

[1] "*A survey of peer-to-peer content distribution technologies*" - Stephanos Androutsellis-Theotokis, Diomidis Spinellis - *ACM Computing Surveys*, Prosinac 2004.

[2] "*On death, taxes, and the convergence of peer-to-peer and grid computing*" - I. Foster, A. Iamnitchi - *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems* (*IPTPS '03*), Berkley, CA, USA, Veljača 2003.

[3] "*Can heterogeneity make Gnutella scalable?*" - Q Lv, S Ratnasamy, S Shenker - *Proceedings of the 1st International Workshop on Peer-to-Peer Systems* (*IPTPS '02*), MIT Faculty Club, Cambridge, MA, USA, Ožujak 2002.

[4] "*Looking up data in P2P systems*" - Hari Balakrishnan, M. Frans Kaashoek, David Karger, Robert Morris, Ion Stoica – *Communications of the ACM*, Veljača 2003.

[5] "*RFC 3629: UTF-8, a transformation format of ISO 10646*" - Francois Yergeau et al., IETF Standards track, Studeni 2003.

[6] "*XML-RPC Specification*" - Dave Winer, http://www.xmlrpc.com/spec (pristup lipnja 2006., zadnja izmjena sadržaja lipnja 2003.)

[7] "HSQLDB" - HSQLDB development group, http://www.hsqldb.org/ (pristup lipnja 1006., zadnja izmjena sadržaja travnja 2006.)

12. O autoru

Ivan Voras je rođen 1981. godine u Slavonskom Brodu, gdje je pohađao osnovnu školu "Bogoslav Šulek" te Prirodoslovno-matematičku gimnaziju Matija Mesić. Uz veliku podršku profesora informatike i profesorice fizike, osvaja brojne gradske i županijske nagrade iz ova dva područja, te tri državne nagrade: dva puta iz fizike (u prvom i četvrtom razredu) i jednom iz informatike (u trećem razredu), u svim slučajevima u kategoriji samostalnih (praktičnih) radova. Jedna od ovih nagrada, prvo mjesto na natjecanju iz fizike, mu je omogućila izravan upis na *Fakultet elektrotehnike i računarstva* u Zagrebu.

Nedugo nakon upisa 2000. godine priključuje se mladom FER-ovom razvojnom timu za web i CMS aplikacije ("FERweb tim") i ubrzo postaje tehnički voditielj tima te administrator poslužitelja i drugih računala. U tom svojstvu sudjeluje u razvoju prve generacije CMS-a na www.fer.hr te njegovih inačica od kojih je jedna instalirana kod CARNet-a a jedna je otkupljena od Plive d.d. za daljnji interni razvoj, te 2006. godine i druge generacije. Tijekom rada na CMS-u CARNet-a sudjeluje i u brojnim povezanim aktivnostima kao što su CARNet-ovi referalni centri, časopis *Edupoint*, *eLA* i druge.

Tijekom studija bavi se mnogim i raznovrsnim stvarima, među kojima su vođenje i održavanje "spontane" studentske računalne mreže u studentskom domu *Ante Starčević*, predavanje na lokalnim i međunarodnim konferencijama, povremeno pisanje za časopise Mreža i Bug, te aktivno sudjelovanje u brojnim open-source projektima (najviše na projektu operacijskog sustava FreeBSD). Profesionalni interesi su mu trenutno raspodijeljeni mrežni sustavi za pohranu podataka.

U ono malo slobodnog vremena koje ponekad nađe najviše voli gledati stare crno-bijele horrore i SF filmove u dobrom društvu.

12.1. Reference

- Treće mjesto, Državno natjecanje iz fizike 1997. sa radom "Eksperimentalno određivanje brzine vrtnje tekućine u magnetskom polju"
- Posebno priznanje, državno natjecanje iz fizike 1998. sa radom "Eksperimentalni dokaz i primjena elektrofotografije"
- Nagrada, Državno natjecanje iz informatike, 1999. sa radom "NT+ Internet serveri"

- Prvo mjesto, Državno natjecanje iz fizike, 2000. sa radom "Izučavanje determinističkog kaosa"
- Član i kasnije voditelj školskog GLOBE tima 1997.-2000. godine (www.globe.gov)
- Voditelj i urednik školskog e-zine časopisa Trag (1999.-2000.)
- Dobitnik Državne stipendije za nadarene studente (2000.-2005.)
- Član i tehnički voditelj FER-ovog razvojnog tima za www.fer.hr ("FERweb")
- Voditelj FERweb tima tijekom razvoja i korištenja CMS-a na www.carnet.hr i pri početnim prilagodbama za Plivu
- Posebno priznanje dekana za razvoj FER-ovog CMS-a (2001.)
- Voditelj FERweb tima tijekom razvoja nove generacije CMS-a (2006.)
- Predavač na konferenciji DORS/CLUC 2003., 2004., 2005. i 2006. godine
- Predavač na konferenciji CUC 2005. godine
- Honorarni autor za časopise Mrež@ i BUG (2004.-2006.)
- Dobitnik stipendije Google Inc. "Google's Summer of Code" 2005. i 2006. godine
- Aktivni suradnik na projektu FreeBSD (operacijski sustav Unix vrste)
- Autor brojnih open-source alata, osam od kojih su prihvaćeni na SourceForge.net

12.2. Bibliografija

- "*Eksperimentalno određivanje brzine vrtnje tekućine u magnetskom polju*" - Ivan Voras, Josip Đermić, Marina Gojković (prof.) - Državno natjecanje iz fizike, 1997.
- "*Eksperimentalni načini dobivanja električne energije*" - Ivan Voras, Zvonimir Mesić, Marina Gojković (prof.) – Državno natjecanje iz fizike, 1998.
- "*Eksperimentalni dokaz i primjena elektrofotografije*" - Ivan Voras, Marina Gojković (prof.) - Državno natjecanje iz fizike, 1999.
- "*Izučavanje determinističkog kaosa*" - Ivan Voras, Marina Gojković (prof.) - Državno natjecanje iz fizike, 2000.
- "*Distributing Web-based Content Management System – FERweb*" - Ivan Voras, Kristijan Zimmer, Mario Žagar, proceedings of Information Technology Interfaces conference, 2005.

- "*On recording and presentation of measurement data acquired via web services*" - Sonja Miličić, Ivan Voras, Hrvoje Keko, proceedings of MIPRO conference, 2006.
- "*A Hierarchical File System Interface to Database-based Content Management System*" - Ivan Voras, Kristijan Zimmer, Mario Žagar, proceedings of Information Technology Interfaces conference, 2006.
- "*Network Distributed File System in User-space*" - Ivan Voras, Mario Žagar, proceedings of Information Technology Interfaces conference, 2006.