UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

**Mojca Ciglarič**

# Usmerjanje ponavljajočih se poizvedb v vsebinskih omrežjih

doktorska disertacija

mentor: doc. dr. Tone Vidmar

Ljubljana, 2003

# KAZALO

# POVZETEK

Vsebina doktorske disertacije sodi na področje računalniških znanosti, v ožje področje računalniških struktur, sistemov in omrežij, še natančneje na področje porazdeljenega iskanja v vsebinskih omrežjih z arhitekturo enakovrednih računalnikov (sistemi tipa enak z enakim oz. *peer-to-peer*).

Porazdeljene vsebine so vsebine, ki se ne nahajajo le na specializiranih strežnikih, ampak tudi na končnih - osebnih računalnikih. Problematika sistema s porazdeljenimi vsebinami se nanaša na zagotavljanje dostopa do vsebin ne glede na njihovo nahajališče, na nadzor in upravljanje z vsebino od trenutka njene pojavitve v sistemu, ter na omogočanje učinkovitega sodelovanja med porazdeljenimi uporabniki.

Disertacija obravnava področje usmerjanja poizvedovalnih sporočil v nestrukturiranih vsebinskih omrežjih z arhitekturo enak z enakim, osredotoča pa se predvsem na zmanjšanje števila redundantnih prenosov, ki ga lahko dosežemo, če vemo, da se vsaj del poizvedb večkrat ponovi.

V uvodu je predstavljena osnovna problematika porazdeljenega iskanja in prikazana povezava z usmerjanjem poizvedovalnih sporočil. Nakazana je potreba po učinkovitejših mehanizmih za vsebinsko usmerjanje zlasti v sistemih, kjer želijo udeleženci ohraniti visoko lokalno avtonomijo.

Drugo poglavje opisuje značilnosti vsebinskih omrežij, pri čemer se osredotoča na združevanje in umeščanje vsebin. Glede na te lastnosti so vsebinska omrežja klasificirana v štiri družine. Za vsako družino je opredeljena težavnost izvedbe usmerjanja, pri čemer so kot najbolj problematična izpostavljena sintaktična vsebinsko neobčutljiva omrežja, krajše vsebinska omrežja tipa A.

Tretje poglavje predstavlja značilnosti sistemov z arhitekturo enak z enakim. Opisane so njihove splošne značilnosti in področja uporabe, več poudarka pa je na tistih lastnostih, ki so odločilne, da se na določenih področjih uporabe odločimo ravno za to arhitekturno obliko: razpršenost, razširljivost, anonimnost, dinamičnost, sposobnost samoorganizacije, avtonomnost in varnost.

Četrto poglavje se osredotoča na usmerjanje v izbranih omrežjih. Predstavljena je analiza lastnosti topologije navideznega omrežja: potenčni zakoni in pojav majhnega sveta. Glede na te lastnosti je predlagan najustreznejši algoritem za generiranje umetnih topologij, na katerih se izvajajo simulacije usmerjanja. Predstavljen je tudi zgoščen pregled najpomembnejših raziskovalnih dosežkov na področju usmerjanja v preučevanih omrežjih.

V petem poglavju je opisana osnovna različica usmerjanja – poplavljanje, nato pa sta predlagani dve izboljšavi poplavljanja, ki izkoriščata prej opisane topološke značilnosti in dejstvo, da se poizvedbe pogosto ponavljajo. To sta usmerjanje s pomnjenjem posredovanih odgovorov in usmerjanje z izmenjavo metapodatkov. Vozlišče si zapomni odgovore, ki jih posreduje svojim sosednjim vozliščem. Ko naslednjič prejme poizvedbo, ki se ujema s shranjenimi metapodatki iz odgovora, je ne poplavi vsem svojim sosedom, ampak le tistemu, od koder je odgovor prispel prejšnjič. V drugem načinu si vozlišča zbrane metapodatke izmenjujejo med seboj in tako po vzoru osnovnega principa porazdeljenega usmerjanja gradijo najkrajše poti do posameznih vsebin. Poudariti je potrebno le, da tu ne gre za usmerjanje prometa do znanega cilja, ampak gre za vsebinsko usmerjanje, torej za usmerjanje do znane vsebine, pri čemer ciljno vozlišče ni pomembno. V nasprotju s sistemi, ki indeksirajo celoto ali del dosegljivih vsebin, se tu ne shranjuje informacija o ciljnem vozlišču, ampak o usmerjanju, torej le o sosednjem vozlišču. Opisano je tudi, kako predlagani izboljšavi usmerjanja premostita težave zaradi dinamičnosti navideznega omrežja, saj stalno prihajanje in odhajanje vozlišč močno vpliva na učinkovitost usmerjanja. Nazadnje navajamo še naša pričakovanja glede obnašanja predlaganih različic usmerjanja.

V šestem poglavju je predstavljen konceptualni model, ki služi kot osnova za izvedbo simulacij: formalno opišemo topologijo omrežja, usmerjanje, porazdelitev in ponavljanje poizvedb ter porazdelitev in repliciranost vsebin. Posamezne vsebine so lahko enako popularne, ali pa so nekatere med njimi bolj popularne. Poizvedbe, ki se generirajo med simulacijo, so lahko porazdeljene uniformno po vsem spektru vsebin, ali pa proporcionalno glede na popularnost vsebin. Vsebine so lahko po vozliščih razmeščene uniformno, lahko pa so bolj popularne replicirane na večje število vozlišč. Izbrana kombinacija parametrov določa tudi ponovljivost poizvedb. Preskusili smo štiri tipe topologij: naključni graf, večkratno povezan obroč, večkratno povezan obroč z

naključnim prevezovanjem povezav in topologijo GLP, ki sledi potenčnim zakonom in pojavu majhnega sveta. Eksperimentirali smo tudi z velikostjo topologije: v veliki je domet poizvedbe manjši od najdaljše izmed najkrajših poti, v majhni pa je domet poizvedbe večji od te vrednosti.

Sedmo poglavje zajema oceno modela – njegovo validacijo in verifikacijo, navaja pa tudi metrike, s katerimi vrednotimo rezultate simulacij tako s stališča obremenjenosti omrežja kot tudi zadovoljstva uporabnika.

V osmem poglavju predstavljamo najpomembnejše rezultate simulacij. Pokažemo, da se z uporabo predlaganih izboljšav usmerjanja poizvedovalni promet navideznega omrežja lahko zmanjša tudi za več velikostnih razredov, pri čemer se kakovost storitve za uporabnika ne poslabša. Na več različicah navideznih omrežij (različne topologije in porazdelitve vsebin oz. poizvedb) prikažemo in komentiramo podobnosti in razlike v zmanjšanju količine prometa. Rezultate ocenimo tudi s pomočjo prej naštetih metrik, nazadnje pa podajamo smernice za uporabo predlaganih mehanizmov.

V sklepu ponovno ovrednotimo glavne ugotovitve disertacije ter navajamo nekaj možnosti za nadaljnje raziskovanje tega zanimivega in hitro razvijajočega se področja.

Disertacija prinaša naslednje prispevke znanosti:

- Analizirali smo obnašanje in topološke lastnosti vsebinskih omrežij z arhitekturo enakovrednih računalnikov ter ugotovili, katere od teh lastnosti lahko izkoristimo za zmanjšanje količine prometa.
- Predlagali smo dve različici učinkovitejšega vsebinskega usmerjanja ponavljajočih se poizvedb, kjer vozlišča upoštevajo prej posredovane odgovore, usmerjevalne metapodatke pa si med seboj lahko tudi izmenjujejo.
- Glede na rezultate simulacij in analize obnašanja predlaganih različic usmerjanja je bilo ugotovljeno, da je v modeliranih sistemih osnovno poplavljanje smiselno nadgraditi s predlaganimi izboljšavami.

# ABSTRACT

The contents of this dissertation fit in the field of computer science, more precisely in the area of computer systems, structures and networks, within which the dissertation deals with distributed search in content networks built on peer-to-peer architecture.

Distributed contents are those that are not located on specialized servers only, but also on several end computers. The main research issues in systems with distributed contents are: enabling content accessibility without regard to its location; management and control over the contents during their existence in the system, and effective collaboration among the system users.

The dissertation deals with query message routing in unstructured content networks with peer-to-peer architecture. Knowing that the quantity of repeated queries is considerable, it focuses predominantly on achieving lower number of redundant query transfers.

The introduction outlines the main problems of distributed search and relates them to query message routing. The need for more effective content based routing in systems with high local autonomy is exposed.

The second chapter describes content network properties, focusing on content placement and content aggregation. With regard to these two characteristics, the content networks are classified in four families. For each family, the level of query routing difficulty is assessed and syntactic content oblivious networks, shortly type A content networks, are found to be the most challenging.

The third chapter reviews the characteristics of peer-to-peer architecture and its implementation areas. Especially those properties are emphasized that strongly affect the choice for or against the peer-to-peer architecture: decentralization, scalability, anonymity, dynamicity, self-organization, desired level of autonomy and security.

The fourth chapter focuses on query routing in the selected type of network overlays. We analyze the overlay topology properties, namely power laws and small world property.

Afterwards, we choose an algorithm suitable for generating artificial topologies with these properties, which we would use for the simulation purposes.

We also present an overview of the most important research achievements concerning distributed search in unstructured peer-to-peer systems.

In the fifth chapter, we first describe the basic routing mechanism – flooding. Then we suggest two improvements, based on the aforementioned topological properties and the fact that some queries are frequently repeated. We call the first improvement *remembering past answers* and the second one *metadata exchange*. Each node stores metadata about the answers that were passed to other nodes. When a node receives a subsequent query, it tries to find a match within the stored metadata. If a match is found, the query is not flooded to all of the node's neighbors, but rather it is routed to the neighbor from whom the matching answer was obtained earlier. When using metadata exchange, the neighbor nodes exchange their metadata collections and build shortest paths towards the contents within their neighborhood in a distributed fashion.

We have to stress that we are talking about content routing here, namely routing towards the desired contents, not towards a known destination node. In content routing, the destination address is not important since the query is satisfied as soon as the contents are found at any node. Contrary to the systems indexing all or a part of the system contents, information regarding the destination node is not stored - only routing information, i.e. information on the neighbor node is stored.

We also describe how our routing scheme deals with network dynamicity and state our expectations regarding improved routing behavior.

In the sixth chapter, we formally present a conceptual model of the described system, serving as a base for the simulation execution. We explain network topology, routing strategies, query distribution and replication, contents distribution and replication. The contents may be either equally or not equally popular. The queries may target the contents either uniformly or proportionally to their popularity. The contents may be either replicated uniformly or may the more popular contents have also more copies within the system.

We have performed the simulation on four types of topologies: random graph, lattice, lattice with randomly reconnected links, and GLP topology, which follows the power law

and small world requirements. We have also experimented with two topology sizes – in the bigger one, the reach of the query is less than the longest of shortest paths in the graph, while in the smaller one these two quantities are roughly the same.

In the seventh chapter, we present model validation and verification and describe a couple of metrics used for interpretation of the simulation results.

In the eighth chapter, we present the most important simulation results. We show that the suggested improvements can reduce the total number of query hops for up to two orders of magnitude, compared to the basic flooding, whereas from the user's point of view, the quality of service remains the same. We present the differences in routing behavior over several combinations of system parameters and topology properties. We also evaluate the results by means of the metrics described earlier. At last, we give some suggestions for practical use of the presented routing improvements.

In the conclusions, we evaluate our contributions again and suggest possible directions for further research of this interesting and quickly developing area.

Our main contributions to science are:

- We evaluated the behavior and topological properties of content networks with peer-to-peer architecture and considered their role in traffic reduction.
- We suggested two improvements for routing of repeated queries, where the nodes consider previously forwarded answers and may also exchange these metadata with their neighbor nodes.
- According to the simulation results and the analysis of suggested routing modifications we observed that in the modeled systems, it is advisable to upgrade the basic flooding with the suggested modifications.

# 1 UVOD

Sistemi z arhitekturo enak z enakim se v zadnjih dveh letih postali izredno zanimivi za široko uporabo zaradi svoje preprostosti, prilagodljivosti, sposobnosti samoorganiziranja in porazdeljevanja bremen, pa tudi zaradi odpornosti na napake in razpoložljivosti, ki jo dosegajo zaradi masovne repliciranosti. Za uporabnika je neprecenljiva njihova sposobnost združevanja in izkoriščanja ogromne količine virov, predvsem komunikacijskih poti ter pomnilnih in procesorskih kapacitet, ki jim nadomeščajo nedosegljive močne in drage strežnike.

Skladno s povedanim skupina Gartner [1] ugotavlja, da se oblikujejo vedno večje potrebe po upravljanju s porazdeljenimi vsebinami. Meni, da bodo učinkovito upravljanje s porazdeljenimi vsebinami omogočile formalne omrežne rešitve (vsebinska omrežja) z arhitekturo tipa enak z enakim s podatkovno osredotočenim modelom. Najpomembnejša naloga porazdeljenega upravljanja bo iskanje porazdeljenih virov (podatkov, vsebin), poleg tega pa tudi zagotavljanje avtonomije lastnikom podatkov, varnost, administracija in podobno.

Trenutno poglavitne izzive za delo na področju sistemov za arhitekturo enak z enakim predstavljajo učinkovitost, zagotavljanje sprejemljivega nivoja zmogljivosti in varnosti, saj so sistemi tipično svetovnih razsežnosti, njihovi udeleženci pa praviloma nezanesljive narave, kar implicira visoko dinamičnost sistema. Poleg nezanesljive narave je značilnost udeležencev tudi njihova zahteva po visoki avtonomiji, ki preprečuje ali močno omejuje uporabo marsikaterega strožjega, a učinkovitejšega mehanizma.

V disertaciji se osredotočamo na problem učinkovitosti iskanja v vsebinskih omrežjih arhitekture enak z enakim, kjer so udeleženci visoko avtonomni, navidezno omrežje pa posledično nizko strukturirano. Ugotavljamo, da v literaturi ni zaslediti poskusov, kako izkoristiti visoko stopnjo ponovljivosti nekaterih poizvedb, ki bi jo lahko izkoristili za zmanjšanje redundantnosti usmerjanja. Zato bomo v disertaciji predlagali dva nova

načina usmerjanja poizvedb in s pomočjo simulacij ovrednotili njune prednosti in slabosti.

## 1.1 Porazdeljeno iskanje

Dober mehanizem za izvajanje porazdeljenega iskanja omogoča uporabnikom, da hitro najdejo lokacijo želene vsebine in da pri tem kar se da racionalno uporabljajo vire celotnega sistema. V sistemih za izmenjavo vsebin z arhitekturo enak z enakim je tak cilj težko dosegljiv zaradi nezanesljivosti posameznih vozlišč, velikosti sistema in podobno.

Uporabniki na posameznih vozliščih generirajo poizvedbe in prejemajo odgovore na poizvedbe. Rezultati so lahko želene vsebine, metapodatki o želenih vsebinah ali pa kazalci na vsebine oziroma podatek o njihovi lokaciji. Vsebine, ki se hranijo v sistemu, so lahko objekti kakršnekoli vrste: datoteke z glasbo, slikami, dokumenti, spletne strani, novičarski prispevki, pa tudi podatki, shranjeni v relacijski podatkovni bazi ali elektronski preglednici. Temu ustrezno so raznolike tudi poizvedbe: lahko jih predstavlja seznam ključnih besed, regularni izraz nad ključnimi besedami, pri tem pa lahko iskanje tudi omejimo na posamezne dele sporočila: glava, naslov oziroma ime, telo, metapodatki...

Iskalni mehanizem troplastno vpliva na sistem: na izbiro tipa topologije, na umeščanje vsebin in metapodatkov in na usmerjanje sporočil. Vse te tri značilnosti morajo se morajo podrediti zahtevi, naj iskanje poteka čimbolj učinkovito. Če je sistem nestrukturiran (torej se udeleženci med seboj lahko povezujejo poljubno), najdeni odgovori pomembno vplivajo na izbrana sosednja vozlišča. V strukturiranem sistemu udeleženci o povezovanju ne morejo odločati sami, vendar algoritem povezovanja išče takšne topologije, da iskanje lahko poteka učinkovito. Umeščanje vsebin mora ravno tako zagotavljati, da bo vsebina enostavno dostopna.

Iskalni mehanizem mora dodatno upoštevati tudi vsebinske zahteve sistema. Jezik za definiranje poizvedb mora biti dovolj izrazen, da uporabnik s poizvedbo lahko zajame dovolj široko množico vsebin, ter dovolj precizen, da uporabnik množico potencialnih rezultatov lahko zoži do obvladljive velikosti. Potrebno je vedeti, kaj se razume pod

zadovoljivim odgovorom: zadošča ena, poljubno katera vsebina, ki ustreza pogojem? Moramo poiskati vse takšne vsebine, ali pa morda le vnaprej določeno število?

V današnjih sistemih ločimo več oblik iskanja:

- Iskanje vsebin s pomočjo identifikatorjev (najpogosteje so to umetno generirana sekvenčna števila ali pa digitalni izvlečki) je dokaj okorno, saj mora uporabnik pred generiranjem poizvedbe ugotoviti, kakšen identifikator ima iskana vsebina. Tudi če v sistemu obstaja osrednje kazalo, v katerem lahko uporabnik najde ta podatke, je stvar nerodna, kadar ne more dovolj točno definirati, kaj išče. Enake slabosti ima tudi iskanje vsebin s pomočjo njihovih digitalnih izvlečkov. Oba principa sta danes najpogosteje uporabljena v eksperimentalnih strukturiranih sistemih enak z enakim, ki so sicer izredno učinkoviti.
- Iskanje vsebin s pomočjo seznama ključnih besed je dovolj intuitivno in zato najbližje povprečnemu uporabniku. Učinkovitejše je, če lahko iskanje omejimo na posamezne metapodatke: ime, velikost, tip, vsebina...
- Iskanje vsebin s pomočjo regularni izrazov je podobno kot s pomočjo ključnih besed, le da lahko zahtevo podrobneje strukturiramo. Za procesiranje te zahteve pa je zato potrebno tudi nekaj več virov.
- Strukturiran povpraševalni jezik se naj bi uporabljal, kadar so vsebine v sistemu strukturirane in s samimi ključnimi besedami ne bi mogli najti želenih vsebin, na primer v porazdeljeni relacijski podatkovni bazi z arhitekturo enak z enakim. Takšni sistemi so danes komaj v začetku eksperimentalne faze in zato je pri izvedbi povpraševanj še mnogo neznank.

## 1.2 Usmerjanje

Usmerjanje je pojem, ki ga dobro poznamo z omrežne plasti v računalniških komunikacijah. V disertaciji pa obravnavamo **vsebinsko usmerjanje na aplikacijski plasti**, zato bomo v nadaljevanju pojasnili podobnosti in razlike med osnovnimi principi obeh izvedb usmerjanja.

V porazdeljenem sistemu posamezni procesi ali aplikacije komunicirajo med seboj zgolj s sporočili. Tudi poizvedovanje je izvedeno tako, da si vozlišča – aplikacijski procesi med seboj izmenjujejo poizvedovalna sporočila in sporočila – odgovore. Na aplikacijski plasti vozlišč ne ločujemo na lokalne infrastrukture (izvori in ponori) in prenosna vozlišča, pač pa imajo tipično vsa vozlišča vse funkcije: generirajo sporočila (so izvori), sprejemajo sporočila (so ponori) in posredujejo sporočila (so usmerjevalniki).

Če mora vozlišče sprejeto sporočilo posredovati naprej, se mora najprej odločiti, kateremu ali katerim od sosednjih vozlišč ga bo posredovalo. Pravila, na podlagi katerih vozlišče sprejema tovrstne odločitve, imenujemo usmerjevalni mehanizem. Preprost primer usmerjevalnega mehanizma je poplavljanje: vozlišče posreduje sporočilo vsem sosednjim vozliščem, razen tistemu, od kogar je prišlo to sporočilo.

Sporočila lahko nosijo ciljni naslov, kamor so namenjena: tipično je to natanko eno, točno določeno vozlišče v sistemu. Vozlišče, ki posreduje sporočilo, ga usmerja tako, da bodo prenosi po poti do ciljnega vozlišča čimbolj učinkoviti. Govorimo o **usmerjanju na podlagi naslova**.

Na aplikacijski plasti pa srečamo drugo obliko usmerjanja: **vsebinsko usmerjanje**. Pri vsebinskem usmerjanju poizvedovalna sporočila ne nosijo ciljnega naslova, vendar pa je iz njih razvidno, kaj iščejo. Vozlišča takšna sporočila usmerjajo glede na njihovo vsebino: usmerjevalne odločitve sprejmejo šele, ko pregledajo vsebino sporočila. Poizvedovalna sporočila na primer usmerjajo tako, da bodo čimprej prišla do enega izmed vozlišč z želeno vsebino, najraje do najbližjega. Poudarimo: ne želimo doseči točno določenega vozlišča, ampak **katerokoli vozlišče** s točno **določeno vsebino**.

Zasnova usmerjevalnega mehanizma močno vpliva na kakovost storitve, ki jo nudi sistem. Prav tako kot lahko neprimeren mehanizem odbije potencialne uporabnike s svojo togostjo in nizko učinkovitostjo, se dober mehanizem zrcali v uspešnosti in učinkovitosti povpraševanja in s tem vabi v sistem tudi nove uporabnike.

# 2 VSEBINSKO OMREŽJE

Pojem vsebinskih omrežij se je v računalniški literaturi uveljavil šele v zadnjem letu ali dveh vzporedno z razmahom aplikacij za medsebojno izmenjavo datotek. Kung in Wu v [6] definirata vsebinsko omrežje kot navidezno omrežje nad omrežjem IP, ki podpira vsebinsko usmerjanje. Vozlišča navideznega omrežja usmerjajo sporočila, lahko pa tudi shranjujejo vsebine in jih nudijo ostalim vozliščem. Vsebinsko usmerjanje sporočil implicira, da se vozlišče odloči za smer, v katero bo posredovalo sporočilo, na temelju vsebine sporočila in ne na podlagi ciljnega naslova, kot smo vajeni na omrežni plasti.

Osnovne naloge vsebinskih omrežij so shranjevanje vsebin (podatkov, datotek, ...), omogočanje iskanja vsebin in upravljanje, seveda pa tudi zagotavljanje avtonomnosti, varnost, administracija (v sistemih, ki to omogočajo) in podobno.

Po osnovnih značilnostih vsebinska omrežja delimo v štiri družine, od katerih ima vsaka svoje prednosti in slabosti in posledično svoje specifično področje uporabe. V nadaljevanju po [6] povzemamo delitev vsebinskih omrežij in njihovo taksonomijo.

Prvi kriterij, po katerem delimo vsebinska omrežja, je način združevanja (agregiranja) vsebin. Združevanje vsebin v skupine je pomembno iz dveh razlogov: zaradi umeščanja vsebin na vozlišča in zaradi vsebinskega usmerjanja sporočil. Obe funkciji, umeščanje vsebin in usmerjanje, se namreč lahko izvajata bodisi glede na posamezno vsebino bodisi glede na skupino, v katero sodi vsebina. Združevanje vsebin v skupine pa omogoča mnogo večjo skalabilnost vsebinskega omrežja tudi pri izredno veliki množici vsebin.

Drugi kriterij pa je umeščanje vsebin na vozlišča. Strategija umeščanja vsebin neposredno vpliva na učinkovitost usmerjanja in tudi na velikost morebitnih indeksov ali količino drugih metapodatkov, potrebnih za izvajanje usmerjanja.

## 2.1 Združevanje vsebin

Združevanje vsebin pomeni po eni strani določanje vsebinske skupine, v katero sodi posamezna vsebina, po drugi strani pa tudi izbor vsebin, ki smo jim določili isto vsebinsko skupino, izmed množice vseh dostopnih vsebin.

O **semantičnem** združevanju govorimo, kadar imajo vsebinske skupine nek pomen – torej v skupine združujemo vsebine, ki so pomensko sorodne. V skupino sesalcev bi na primer združili vsebini *pes* in *krava*, v skupino žuželk pa *pajek* in *muha*.

O **sintaktičnem** združevanju pa govorimo, kadar vsebine združujemo na podlagi rezultata neke funkcije, ki ni povezana s pomenom (na primer zgoščevalna funkcija, ki kot argument vzame ime datoteke). Rezultat je le niz bitov, ki o vsebini ne povedo ničesar, tako kot tudi s primerjavo nizov bitov različnih vsebin ne moremo pridobiti nobene nove informacije o teh vsebinah. Seveda pa za vsako skupino vsebin lahko določimo meje intervala in v izbrano skupino uvrstimo vse vsebine, katerih rezultati padejo v ta interval. Primer: v isto skupino vsebin bi z uporabo funkcije, ki prešteje število znakov, lahko združili vsebini *pes* in *zob*, v drugo skupino pa *mačka*, *modem* in *5FX3f*.

V nekaterih sistemih pa vsebin sploh ne združujemo v skupine, ampak se umeščanje vsebin, iskanje po vsebinah in posledično tudi usmerjanje sporočil do določene vsebine izvaja za vsako vsebino posebej, ne glede na njeno sintakso ali semantiko. Take sisteme po Kungovi definiciji uvrščamo v skupino s sintaktičnim združevanjem.

## 2.2 Umeščanje vsebin

Umeščanje neke vsebine na vozlišče ali več vozlišč vsebinskega omrežja se lahko izvede bodisi kot funkcija same vsebine bodisi neodvisno od vsebine.

Omrežja z vsebinsko občutljivim umeščanjem ali krajše **vsebinsko občutljiva** omrežja so torej tista, kjer se umeščanje vsebin izvaja kot funkcija vsebine, glede na sintakso ali semantiko vsebine. Lahko rečemo, da na posamezno vozlišče ali skupino vozlišč umestimo le vsebine z določenim pomenom, ključnimi besedami ali pa z istim rezultatom zgoščevalne funkcije. Vsebinsko občutljivo umeščanje se izkaže za zlasti

učinkovito v semantičnih vsebinskih omrežjih, saj lahko s pametnim umeščanjem dosežemo skladnost med vsebinsko in topološko hierarhijo in posledično številne koristi:

- Učinkovito izvajanje poizvedb: poizvedba se usmerja glede na višjenivojsko skupino vsebin. Ko jo najde, se usmerja natančneje v smer prave podskupine in nazadnje do ustrezne vsebine. Zaradi skladnosti med vsebinsko in topološko hierarhijo pa vemo, da do tja ni več daleč.
- Vozlišča lahko vzdržujejo le majhno količino metapodatkov, potrebnih za usmerjanje poizvedb (zaradi vsebinske hierarhije) – torej le usmerjanje do podskupin za svoje vsebinsko področje in usmerjanje do glavnih vsebinskih skupin.
- Možno je učinkovito preiskovanje pomenske bližine oziroma pomensko sorodnih vsebin (*semantic proximity search*), saj jih ni težko identificirati – umeščene so na bližnjih vozliščih.
- Če so določene vsebine zelo zaželene, po njih uporabniki poizvedujejo zelo intenzivno. Takrat lahko nastopijo težave zaradi prevelikega števila dostopov na majhen del omrežja, s tem pa do zasičenja komunikacijskih poti in posledične nedostopnosti ravno tega dela omrežja. Po drugi strani pa nam informacija o zaželenosti določene skupine vsebin pove tudi to, kako zmogljivo komunikacijsko pot do te skupine moramo zagotoviti, da bo sistem kot celota lahko nemoteno deloval.

V nekaterih vsebinsko občutljivih sistemih je dovoljena ali celo zaželena replikacija vsebin. Za dodatne kopije vsebin je možno v nekaterih obstoječih sistemih zahteve glede umeščanja podatkov nekoliko zrahljati in vsebine na ta način bolj približati tistim lokacijam, kjer jih uporabniki pogosto potrebujejo.

Pri **vsebinsko neobčutljivem umeščanju** lahko vsebinske skupine namestimo na katerokoli vozlišče, ne glede na njihov pomen. Pri poizvedovanju zato tako omrežje ne more ločiti bolj obetavne smeri iskanja ob manj obetavnih. Poizvedba je očitno uspešnejša, če obišče več vozlišč.

Vsebinsko omrežje mora imeti izdelan mehanizem, s pomočjo katerega lahko najde pot do posameznih vsebin. Arhitekturno preprosta rešitev je vzpostavitev osrednjega

strežnika, kjer se registrirajo vse vsebine v sistemu in kamor se naslavlja vse poizvedbe. Strežnik poizvedovalcu pošlje odgovor, namreč lokacijo iskane vsebine, ta pa potem lahko pošlje zahtevo za vsebino na ciljno vozlišče.

Izvedljive pa so tudi bolj razpršene rešitve: vozlišča lahko razglašajo, kaj v svojem skladišču vsebin nudijo svojim sosedom. Sosedje si tako lahko oblikujejo indekse oziroma kazala, s pomočjo katerih lažje usmerjajo poizvedbe po svoji soseščini. Lahko pa vozlišča, ki želijo najti neko vsebino, o tem obveščajo svoje sosede. Sosedje nato po svoji presoji posredujejo poizvedbo naprej ali pa odgovorijo nanjo.

V omrežjih z vsebinsko neobčutljivim umeščanjem torej neko vsebino lahko umestimo na katerokoli vozlišče, ne glede na njen pomen, vsebinsko skupino ali topologijo omrežja. Tudi topologija navideznega omrežja se zato lahko oblikuje neodvisno od vsebin. To udobje pa po drugi strani odtehta cena medsebojnega obveščanja vozlišč, točneje prenosi poizvedb in metapodatkov o vsebinah.

## 2.3 Značilnosti posameznih skupin vsebinskih omrežij

Kot smo že omenili, razvrščamo vsebinska omrežja glede na način združevanja vsebin in na način umeščanja vsebin v štiri skupine, kar prikazuje tudi Tabela 2.1. V tem razdelku bomo predstavili značilnosti vsake izmed njih, pri čemer se bomo osredotočili na za nas najbolj zanimivo lastnost – način usmerjanja sporočil.

| | **SINTAKTIČNO ZDRUŽEVANJE VSEBIN** | **SEMANTIČNO ZDRUŽEVANJE VSEBIN** |
|---|---|---|
| **VSEBINSKO NEOBČUTLJIVO UMEŠČANJE** | **Tip A**<br>Sintaktična vsebinsko neobčutljiva omrežja | **Tip C**<br>Semantična vsebinsko neobčutljiva omrežja |
| **VSEBINSKO OBČUTLJIVO UMEŠČANJE** | **Tip B**<br>Sintaktična vsebinsko občutljiva omrežja | **Tip D**<br>Semantična vsebinsko občutljiva omrežja |

*Tabela 2.1: Štiri družine vsebinskih omrežij.*

### 2.3.1 Sintaktična vsebinsko neobčutljiva omrežja – tip A

Vsebinska omrežja, ki uporabljajo sintaktično združevanje ali pa vsebin sploh ne združujejo v skupine, dobro podpirajo anonimnost vsebin. Pojem anonimnost vsebin označuje lastnost omrežja, da skriva semantiko ali pomen vsebin. Drugače povedano, tudi če poznamo lokacijo vsebine ali pa vemo za njeno pripadnost neki vsebinski skupini, na podlagi tega ne moremo ugotoviti ničesar o njenem pomenu. In obrnjeno – tudi če poznamo pomen in vsebinsko skupino, nam to ne pove ničesar o lokaciji vsebine.

Tovrstna omrežja imajo visoko odpornost na napade, katerih cilj je onemogočiti dostop do neke vsebine ali skupine vsebin.

Usmerjanje v sintaktičnih vsebinsko neobčutljivih pa je težavno. Poizvedba z večjo verjetnostjo najde želeno vsebino, če obišče več vozlišč, obenem pa omrežja nočemo zadušiti v preveliki količini poizvedovalnih sporočil. Zlasti v omrežjih z velikim številom vozlišč namreč preprosto ni možno, da bi vsaka poizvedba obiskala vsa vozlišča.

Zato se v obstoječih sistemih te vrste (največ jih ne uporablja nikakršnega združevanja vsebin) pogosto uvede osrednji strežnik, ki služi kot kazalo vsebin za celoten sistem - tako deluje na primer spletni iskalnik Google [72], podoben princip pa je uporabljal tudi že pokojni sistem za izmenjavo glasbenih datotek Napster [73]. Žal pa je osrednji strežnik občutljiva točka celotnega sistema in če ta preneha delovati, sistem popolnoma razpade.

Druga možna izvedba iskanja je posredovanje poizvedovalnih sporočil od vozlišča do vozlišča. Najbolj robusten način usmerjanja sporočil je poplavljanje – vozlišče, ki na poizvedbo ne zna odgovoriti, jo posreduje vsem svojim sosedom. Če lahko vozlišče svoje sosede klasificira glede na to, kdo bo verjetneje odgovoril na poizvedbo, je izvedljivo tudi iskanje v globino: poizvedbo vozlišče posreduje najbolj obetavnemu sosedu. Če ta ne more odgovoriti nanjo, vozlišče nato poizvedbo posreduje naslednjemu sosedu. Če noben izmed sosedov ne more odgovoriti na poizvedbo, vozlišče odgovori sosedu, od kogar je prejelo poizvedbo, da nanjo ne more odgovoriti. Opisani mehanizmi so zelo potratni – bodisi časovno, bodisi glede zahtev po širini komunikacijskih poti in imajo nizko skalabilnost.

### 2.3.2 Sintaktična vsebinsko občutljiva omrežja – tip B

Sintaktična vsebinsko občutljiva omrežja so organizirana tako, da obidejo težave, povezane s slabo skalabilnostjo in občutljivostjo osrednjih strežnikov. Topologija navideznega omrežja je strogo predpisana, tipičen primer je hiperkocka, lahko pa jo nadomestijo tudi razni obroči ali navidezni večdimenzionalni koordinatni prostor. Sosednja vozlišča imajo pogosto del naslova med seboj enak. Tudi vsebine imajo svoje identifikatorje, ki jih dobijo z uporabo zgoščevalne funkcije (*hash*) na primer nad imenom datoteke, ključem lastnika vsebine in podobno.

Sistem ocenjuje ujemanje med identifikatorjem vsebine in identifikatorjem vozlišča, pri čemer se načini ocenjevanja od sistema do sistema zelo razlikujejo. Umeščanje je vsebinsko občutljivo, zato se vsebino vedno umesti na vozlišče (lahko tudi na večje število vozlišč), kjer je stopnja ujemanja največja. Koncept iskanja vsebine je zaradi takšnega načina umeščanja povsem trivialen: poizvedovalno sporočilo mora vsebovati identifikator iskane vsebine in vozlišča ga usmerjajo v smeri vedno večjega ujemanja z identifikatorji sosednih oziroma naslednjih vozlišč.

V zadnjih dveh letih so tovrstni eksperimentalni sistemi kar vzcveteli – omenimo naj najbolj citirane: Past – Pastry [47], OceanStore – Tapestry [48], CAN [45], Chord [46].

Kljub temu pa ne moremo prezreti dejstva, da so vsebinsko občutljivi sistemi kljub učinkoviti izvedbi iskanja vsebin zaradi svoje strogosti in formalne strukturiranosti manj primerni, če že ne celo neprimerni za uporabo v ohlapnih sistemih, kjer imajo vozlišča visoko lokalno avtonomijo, se dinamično vključujejo in spet izstopajo iz sistema ter želijo imeti večji nadzor tako nad "svojo" vsebino, ki jo nudijo ostalim vozliščem, kot tudi nad morebitno vsebino ostalih vozlišč, ki bi se morala po ravnokar opisanih mehanizmih umestiti na njihovo lokacijo.

### 2.3.3 Semantična vsebinsko neobčutljiva omrežja – tip C

V semantičnih vsebinsko neobčutljivih omrežjih se vsebine združujejo v skupine na podlagi njihovega pomena, vendar pa nastale vsebinske skupine lahko umestimo na katerokoli vozlišče. Podobno kot pri sintaktičnih vsebinsko neobčutljivih omrežjih iskanje poteka bodisi s potratnim poplavljanjem poizvedovalnih sporočil ali pa z

oglaševanjem, torej z obveščanjem ostalih vozlišč o vsebinah in vsebinskih skupinah, ki so na voljo. Lažje izvedljivo pa je iskanje pomensko sorodnih vsebin: ko najdemo neko vsebino iz določene vsebinske skupine, smo namreč našli celo vsebinsko skupino in za dostop do ostalih vsebin iz iste skupine ponovno iskanje ni potrebno.

### *2.3.4 Semantična vsebinsko občutljiva omrežja – tip D*

Zadnja skupina vsebinskih omrežij združuje dobre lastnosti semantičnega združevanja vsebin, namreč možnost preiskovanja sorodnih vsebin, in vsebinsko občutljivega umeščanja, namreč deterministočnost nahajališča vsebinske skupine. Sistemi so zahtevnejši za vzpostavljanje in upravljanje, njihova prednost pa je visoka učinkovitost. Usmerjanje poteka v smeri vsebinske hierarhije, skladnost med topološko in vsebinsko hierarhijo pa je visoka.

Največja slabost semantičnih vsebinskih omrežij je ranljivost za napad, katerega namen je odstranitev posamezne vsebinske skupine, na nekaterih potencialnih področjih uporabe pa se bi kot slabost izrazila tudi prenizka lokalna avtonomija.

## 2.4 Problematika usmerjanja v vsebinskih omrežjih

Ogledali smo si različne tipe vsebinskih omrežij in opisali njihove lastnosti. Ugotovili smo, da je usmerjanje poizvedb bistveno lažje, kadar je znan tudi postopek umeščanja vsebin. Če vemo, kam smo umestili neko vsebino, jo namreč tudi iščemo na tistem mestu. Usmerjanje se delno olajša tudi, če se uporablja semantično združevanje vsebin, saj lažje najdemo sorodne vsebine - potem ko smo našli vsaj eno iz vsebinske skupin.

Največji izziv za usmerjanje predstavljajo sintaktična omrežja, neobčutljiva za vsebino (družina A). Vsebinske skupine se namreč tvorijo z neko umetno funkcijo, ki iskanja vsebin nič ne olajša, ali pa se združevanje vsebin sploh ne uporablja. Umeščanje vsebin je neobčutljivo, torej se na izbranem vozlišču lahko nahaja katerakoli vsebina. In obratno: izbrana vsebina se lahko nahaja na kateremkoli vozlišču. Med vsebinsko hierarhijo, če ta sploh obstaja, in topologijo navideznega omrežja ni prave povezave, zato tudi za vzpostavljanje povezav med vozlišči navadno ni posebnih omejitev. Prednost tovrstnega omrežja je relativno visoka odpornost proti napadu na določeno vsebino ali

vsebinsko skupino, fleksibilnost topologije in visoka stopnja lokalne avtonomije. Slabosti pa so povezane predvsem z zahtevnostjo usmerjevalnih postopkov in njihovo visoko redundanco in relativno nizko učinkovitostjo na danes običajno zmogljivih komunikacijskih poteh.

Zaradi naštetih razlogov se bomo v tem delu osredotočili na usmerjanje v sintaktičnih omrežjih, neobčutljivih za vsebino. Čeprav problematike usmerjanja ne bomo rešili v celoti, bomo pokazali, da lahko z razmeroma skromnimi sredstvi močno zmanjšamo kumulativno obremenitev vsaj v primerih, ko se nekatere poizvedbe ponavljajo.

# 3 SISTEMI ENAK Z ENAKIM

V prejšnjem razdelku smo opisali namen in način delovanja vsebinskih omrežij. V tem pa predstavljamo značilnosti sistemov in aplikacij, ki jih lahko opišemo z izrazom enak z enakim ali P2P (*peer-to-peer*).

Vsebinska omrežja in sistemi enak z enakim se v mnogočem prekrivajo in imajo med seboj mnogo sorodnega. Večina vsebinskih omrežij ima arhitekturo enak z enakim, kar pomeni, da imajo udeleženci med seboj enako funkcionalnost, vendar pa sam izraz vsebinsko omrežje v prvi vrsti označuje namembnost in funkcionalnost, ne pa zgradbo in obliko sistema. Za vsebinska omrežja in za sisteme enak z enakim je značilno in zelo pomembno navidezno omrežje, ki se gradi na aplikacijski plasti. Mnogo sistemov enak z enakim podpira vsebinsko usmerjanje, nekatere oblike teh sistemov pa vendarle ne gradijo navideznega omrežja in tako tudi ne uporabljajo vsebinskega omrežja (na primer sistem za porazdeljeno računanje seti@home).

V splošnem označujemo s pojmom vsebinska omrežja sisteme, katerih glavna naloga je hranjenje in omogočanje dostopa do vsebin, medtem ko pri sistemih enak z enakim uveljavljeni izraz *izmenjava datotek* označuje le eno izmed več različnih področij uporabe. Nasprotno z izrazom enak z enakim v splošnem opisujemo arhitekturo sistema, ne pa njihovega namena.

Po definiciji, ki jo navajajo Milojicic in soavtorji v [5], **z izrazom enak z enakim opisujemo sposobnost reševanja osnovnih nalog sistema na značilen decentraliziran način**. Razpršeni viri, ki jih udeleženci uporabljajo pri reševanju svojih osnovnih nalog, so lahko procesorske zmogljivosti, pomnilni prostor, vsebine, komunikacijske kapacitete, človeški viri. Najznačilnejše funkcije tovrstnih sistemov so:

- izmenjava datotek ali podatkov v širšem smislu (vsebin),
- porazdeljeno računanje,
- komuniciranje in sodelovanje, skupinsko delo

- na nek način pa tudi storitve platforme ali vmesne plasti (*middleware*). V takem primeru sistem enak z enakim predstavlja plast, ki popolnoma zakriva podrobnosti spodaj ležečega sistema, implementira osnovne protokole in funkcionalnost za delovanje aplikacij enak z enakim in nudi uporabnikom ali tudi aplikacijam na višjih plasteh le vmesnik do svojih storitev.

Kljub temu pa definicija ne izključuje ohranjanja centralizacije v kakem delu ali več manjših delih sistema, če je to smiselno.

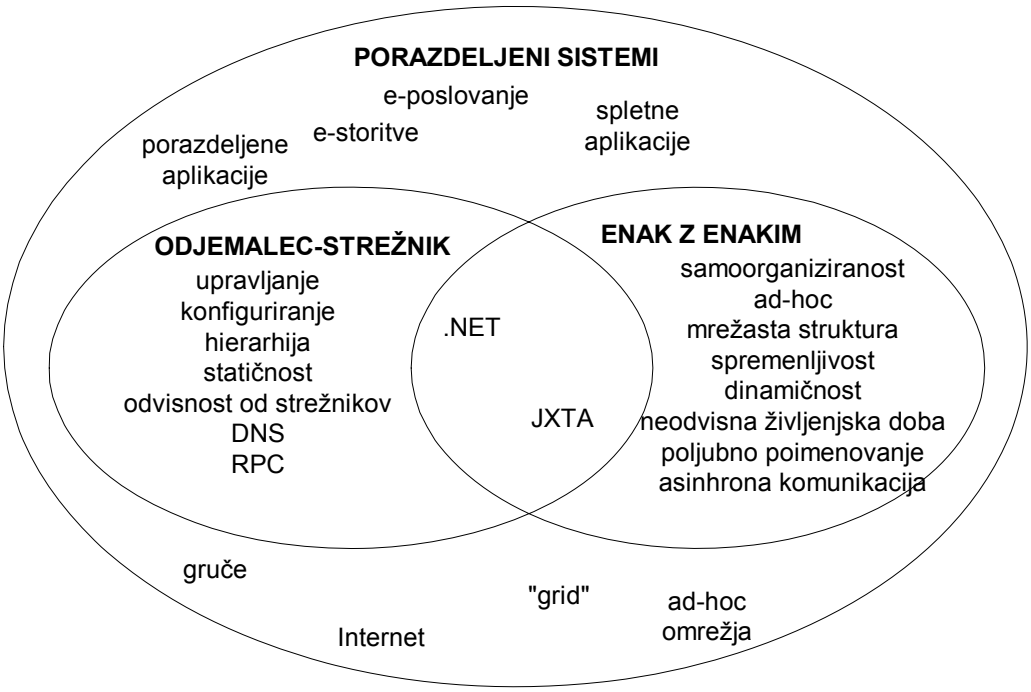
V nadaljevanju bomo z več vidikov predstavili in opisali arhitekturo enak z enakim in lastnosti sistemov na splošno, pa tudi glede na področja uporabe.

## 3.1 Splošne lastnosti

Posamezni končni računalniki - vozlišča, ki sodelujejo v sistemu, so med seboj enakovredni, tako po funkcionalnosti, kot v grobem tudi po zmogljivostih. So visoko avtonomni, kar pomeni, da se ne pustijo nadzirati kateremu od ostalih ali pa neki osrednji avtoriteti, ki bi nadzirala tudi ostala vozlišča. Zaradi tolikšne avtonomije se seveda postavi vprašanje, s kolikšno zanesljivostjo lahko sploh posamezno vozlišče zaupa ostalim in se zanese nanje, da bodo dobro opravljali svojo nalogo – zagotoviti vire, posredovati zahteve, omogočiti dostop do vsebin in podobno. Posamezno vozlišče ne more biti prepričano, da bo njegov sosed pravilno usmerjal sporočila, shranil dokument, ki mu ga pošlje, in ga na zahtevo tudi vrnil.

Posledično je v tovrstnih sistemih nujna višja stopnja redundance kot v klasičnih centraliziranih ali porazdeljenih sistemih.

Slika 3-1 prikazuje odnose med sistemi enak z enakim, odjemalec - strežnik in porazdeljenimi sistemi na splošno. Čeprav navajamo njihove tipične lastnosti, naj vendarle opozorimo, da bi bilo zelo težko ali celo nemogoče med njimi potegniti ostro ločnico. Sistem enak z enakim je na primer možno zasnovati tudi tako, da ga upravljamo in organiziramo od zunaj, pri arhitekturi odjemalec – strežnik pa si lahko zamislimo svoj način poimenovanja in ne uporabimo imenskega sistema domen.




*Slika 3-1: Mesto sistemov enak z enakim med porazdeljenimi sistemi.*

Oba sistema lahko služita kot podlaga za klasične in nove vrste aplikacij, oba lahko implementiramo na različnih platformah. Priznati moramo, da sam koncept direktnega povezovanja med enakimi udeleženci ni nov, poznamo ga že vsaj od rojstva klasične telefonije. Tudi dejstvo, da razpršenost virov načeloma prispeva k višji razpoložljivosti in skalabilnosti, je bilo znano še pred pojavom sistemov enak z enakim.

Naštejmo sedaj tiste novosti, ki jih prinaša šele koncept enak z enakim:

- Težišče dogajanja in izvajanja se prenese iz središča (prej na strežniku) na rob omrežja – na končne, uporabniške računalnike, katerih število lahko doseže tudi nekaj milijonov.
- Med seboj se povezujejo neposredno in nenačrtovano, kar zvišuje možnost zagotavljanja zasebnosti in anonimnosti.
- S stališča sodelovanja se tolerira in celo predvideva pogoste vstope v sistem in izstope iz njega med delovanjem, zato jih je potrebno ustrezno obravnavati (podpreti).
- Zaradi združevanja cenejših virov so skupni stroški lastništva nižji.

- Novi so tudi algoritmi; nekateri že znani koncepti porazdeljenih sistemov pa so potrebovali večje ali manjše prilagoditve.

Dejavniki, ki so omogočili nastanek in izredno hitro uveljavitev koncepta enak z enakim, so infrastrukturnega značaja (široka prisotnost in dostopnost komunikacijskih omrežij, zlasti interneta, ter veliko število računalnikov, prenosnikov, dlančnikov in podobnih naprav), pa tudi arhitekturni (zlasti vedno večja razpršenost komponent).

Neposredne koristi, ki jih za uporabnika prinaša sodelovanje v takem sistemu, pa so predvsem naslednje:

- Dostop do izredno velike količine virov (procesorske moči, pomnilnega prostora, datotek,…) - tako doma kot na delovnem mestu.
- Možnost neposrednega stika z drugimi uporabniki brez posrednikov ali vsaj z minimalnim posredovanjem.
- Lažja in učinkovitejša komunikacija in sodelovanje.
- Možen je večji razpon velikosti sistema, višja je razpoložljivost, višja je lahko tudi stopnja anonimnosti.

Ključna lastnost sistemov enak z enakim je njihova decentralizirana narava, ki se odraža na algoritmih, zasnovi aplikacij, podatkovnih strukturah, skalabilnosti in razpoložljivosti.

## 3.2 Področja uporabe

V tem razdelku bomo opisali, na katerih področjih uporabe so sistemi enak z enakim še zlasti primerni za implementacijo.

### *3.2.1 Izmenjava datotek*

Na področju izmenjave datotek je bila arhitektura enak z enakim do sedaj najbolj učinkovita in največ uporabljana. Sistemi za shranjevanje in izmenjavo vsebin nudijo uporabnikom okolje za iskanje, referenciranje in pridobivanje datotek, nudijo lahko tudi okolje za shranjevanje datotek, kjer skrbijo za čimbolj učinkovito lociranje in tudi morebitno replikacijo, zagotavljajo določeno stopnjo anonimnosti udeležencem sistema

in omogočajo upravljanje sistema (resnici na ljubo moramo priznati, da nekateri sistemi omogočajo le minimum upravljanja, spet drugi kar visoko stopnjo upravljanja, večinoma pa gre še največkrat za samoupravljanje, kar pomeni, da so nastavitve že del sistema in jih ni mogoče spreminjati).

V grobem lahko v sistemih za izmenjavo datotek identificiramo tri modele zasnove, ki jih bomo natančneje opisali v nadaljnjih podpoglavjih:

1. Centraliziran model z osrednjim kazalom: podatki o datotekah se zbirajo in iščejo centralno, v kazalu osrednjega strežnika.
2. Model s poplavljanjem poizvedb: nič ni znanega o lokaciji datotek, zato je iskanje statistično gledano tem učinkovitejše, čim več udeležencev obišče poizvedba.
3. Model z definiranim umeščanjem datotek: datoteke se umeščajo na vozlišča po vnaprej znanih pravilih, ki omogočajo, da se jih kasneje tudi lažje in hitreje najde.

Tabela 3.1, ki jo deloma povzemamo po Milojicicu [4], nazorno prikazuje primerjavo različnih oblik skupne uporabe virov (predvsem datotek oziroma vsebin): sistem enak z enakim in dve njegovi "historični" alternativi, namreč porazdeljen datotečni sistem ter spletne strani. Navzlic velikim razlikam v organizaciji in implementaciji so z zornega kota uporabnika prisotne tudi znatne podobnosti.

Iz nje pa razberemo tudi, v kakšnih okoliščinah se je modro in kdaj ne odločiti za arhitekturo enak z enakim: argumenti proti so predvsem velika potreba po varnosti, zahteve po močnejši konsistentnosti, visoki razpoložljivosti, pa tudi nezaželenost anonimnosti.

V tabeli s pojmom visoka skalabilnost označujemo sposobnost sistema, da podpira tudi zelo veliko število uporabnikov (torej da je svetovnih razsežnosti). Z izrazom transparentnost pa označujemo predvsem transparentnost dostopa in lokacije: ali do oddaljenih vsebin lahko dostopamo na enak način kot do lokalnih in ali moramo za dostop do oddaljenih vsebin natančno poznati njihovo lokacijo.

Odpornost na napake zajema tudi širino spektra različnih možnih napadov, ki jim lahko podleže sistem. Ostali pojmi so natančneje predstavljeni v nadaljevanju.

| REŠITEV | PRIMER | NAMEN | INFRASTRUK-TURA | KONSIS-TENTNOST | RAZPR-ŠENOST | SKALA-BILNOST |
|---|---|---|---|---|---|---|
| **Porazdeljen datotečni sistem** | NFS, DFS | Splošna raba datotek | Gruče, WAN, intranet | Močna | Srednja | Visoka |
| **WWW** | Spletne strani | Dostop do informacij | Internet | Bralni dostopi | Srednja | Visoka |
| **P2P** | Gnutella, eDonkey | Izmenjava vsebin | Internet, namen-ska omrežja | Šibka | Visoka | Visoka |

| REŠITEV | ANO-NIMNOST | SAMOORGA-NIZIRANOST | STROŠKI LASNTIŠTVA | VARNOST | TRANSPA-RENTNOST | ODPORNOST NA NAPAKE |
|---|---|---|---|---|---|---|
| **Porazdeljen datotečni sistem** | nizka | Nizka | Visoki | Visoka | Visoka | Visoka |
| **WWW** | srednja | Srednja | Nizki | Nizka | Srednja | Srednja |
| **P2P** | Lahko visoka | Visoka | Zelo nizki | Zelo nizka | Nizka | Nizka |

*Tabela 3.1: Primerjava treh oblik skupne uporabe virov.*

### 3.2.2 Porazdeljeno računanje

Uporaba sistema enak z enakim za porazdeljeno računanje predstavlja udejanjanje že nekaj časa prisotne ideje o koristni izrabi prostih procesorskih ciklov in o doseganju visokih zmogljivosti s pomočjo standardnih komponent ali računalnikov. To je specifična oblika, ki jo literatura, na primer Oram [2], zaradi njenih lastnosti uvršča med sisteme enak z enakim kljub temu, da udeleženci med seboj neposredno ne komunicirajo.

Tipično delovanje v sistemu enak z enakim nadzira osrednji strežnik, ki razdeljuje naloge udeležencem. Udeleženec sprejme nalogo in jo reši, ko ima čas (sprožilec je na primer trenutek, ko se vključi ohranjevalnik zaslona), nato pa rezultate vrne osrednjemu strežniku. Ta sestavlja rezultate v celoto, pri čemer sumljive zavrže in isto nalogo ponovno odda drugemu udeležencu.

Vidimo, da je ta sistem hibriden, saj je močno centraliziran. Obenem pa naj opozorimo na razliko s sistemom odjemalec - strežnik. V sistemu s strežnikom in več odjemalci generirajo zahteve odjemalci in jih pošiljajo strežniku v izpolnitev, v sistemu enak z enakim pa strežnik generira zahteve in jih pošilja udeležencem, da jih izpolnijo.

Udeleženci imajo zelo visoko avtonomijo: naloge lahko rešujejo, kadar želijo in prenehajo lahko, kadar želijo. Vsi imajo med seboj enako funkcionalnost in izvajajo večji del celotnega dogajanja v sistemu. Zato jih uvrščamo med sisteme enak z enakim.

Nekateri avtorji vključujejo med sisteme enak z enakim za porazdeljeno računanje tudi tako imenovane *grid* sisteme, ki so med seboj tesneje povezani in so od zunaj lahko vidni in uporabni transparentno, kot en sam računalnik. Vendar pa v znanstveni skupnosti danes že prevladuje mnenje, da zaradi nižje avtonomnosti vozlišč *grid* sistemi sodijo v svojo kategorijo.

Največja omejitev za uporabo sistema enak z enakim za porazdeljeno računanje je dejstvo, da mora biti osnovni problem možno razbiti na delčke, ki so med seboj popolnoma neodvisni, saj bi komuniciranje med udeleženci prineslo prevelike zakasnitve. Primerni so problemi tipa SPMD[1], kjer je potrebno isti računski postopek izvajati z mnogo različnimi kombinacijami vhodnih podatkov, na primer različne simulacije ali ugotavljanje veljavnosti vsakovrstnih modelov: raziskave človeškega genoma, analiza signalov iz vesolja, iskanje zdravila proti raku ali aidsu, borzne kalkulacije...

Glavne težave sistemov za porazdeljeno računanje ležijo na področjih varnosti in organizacije. Finančne aplikacije denimo tečejo izključno za požarnimi pregradami, saj je vstop na internet preveč tvegan, seveda pa to dejstvo močno zožuje krog razpoložljivih računalnikov, ki jih je možno vključiti v sistem. Na področju organizacije se pojavljajo poskusi zaračunavanja storitev oziroma plačevanja uporabnikom, ki nudijo sistemu svoje procesorske zmogljivosti. Vendar pa noben od takih poskusov do danes še ni bil uspešen: največ rezultatov lahko pokažejo projekti, kjer udeleženci sodelujejo iz veselja ali entuziazma.

### 3.2.3 Skupinsko delo

Sistemi za skupinsko delo nudijo podporo uporabniškemu sodelovanju na aplikacijski plasti. Tipične aplikacije, ki sodijo v to skupino, so orodja za komuniciranje, na primer

[1] SPMD – angleška kratica za *Single Problem Multiple Data.*

takojšnje sporočanje in klepet, ter aplikacije za skupno uporabo, igre za več igralcev, in ne nazadnje kompleksna orodja za podporo skupinskemu delu, ki že imajo elemente pravega porazdeljenega operacijskega sistema.

Naštejmo še izzive, s katerimi se soočajo tovrstni sistemi. Lociranje udeležencev lahko poteka centralizirano s prijavljanjem in odjavljanjem, lahko pa bi bilo implementirano tudi porazdeljeno, če bi razvili dodaten sistem za prepoznavanje udeležencev in razvrščanje v skupine. Potrebno je tudi zagotavljanje pravilnega delovanja sistema v primeru napak in začasnih odklopov posameznih udeležencev, ki jim je potrebno po vrnitvi v sistem zagotoviti vsa sporočila, ki jim prej niso mogla biti dostavljena. Težave lahko predstavlja skalabilnost in posledično dolgi odzivni časi, do katerih pride zaradi fizične oddaljenosti udeležencev in njihovega velikega števila.

### *3.2.4 Platforma*

Operacijski sistemi postajajo za izvajanje aplikacij manj pomembni, kot so bili nekoč, namesto tega pa pridobivajo pomembnost rešitve vmesne plasti (*middleware*). Take rešitve, če so le kakovostne in dovolj celovito zasnovane, lahko že imenujemo platforma. Za sisteme enak z enakim sta danes takšni predvsem okolji .NET in JXTA. Okolje .NET zajema celovito podporo storitvam in v ta okvir sodi tudi podpora arhitekturi enak z enakim. JXTA pa je okolje, ki podpira primarne komponente sistemov enak z enakim: odkrivanje virov, poimenovanje, varnost, komuniciranje, združevanje virov, obenem pa poudarja prenosljivost aplikacij med različnimi sistemi.

## 3.3 Značilnosti sistemov enak z enakim

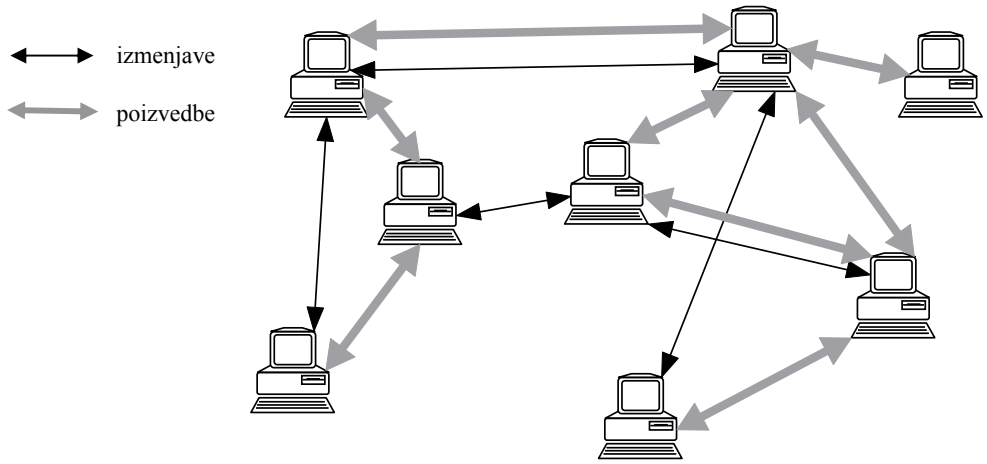
Namen vsakega računalniškega sistema je najprej nuditi podporo aplikacijam, ki izpolnjujejo želje uporabnikov. V tem razdelku bomo opisali, kateri so pogosti razlogi, ki načrtovalce privedejo do odločitve za arhitekturno rešitev enak z enakim, ali drugače povedano, katere probleme rešimo na najbolj eleganten način, če uporabimo sistem enak z enakim. Eden izmed množice motivov je na primer znižanje skupnih stroškov sistema. V centraliziranih sistemih z veliko odjemalci strežnik predstavlja največji strošek tako z vidika strojne kot tudi programske opreme. Prehod na sistem enak z enakim pomeni

razpršitev tega stroška na vse udeležence. Prihranek nastopi predvsem zato, ker udeleženci ponudijo v sistem svoje sicer neizkoriščene vire, na primer procesorske in pomnilne kapacitete, pa tudi zato, ker so osebni računalniki, ki predstavljajo tipične udeležence sistemov enak z enakim, relativno poceni v primerjavi z zmogljivimi večprocesorskimi strežniki.

Oglejmo si v nadaljevanju tiste značilnosti sistemov enak z enakim, ki same po sebi in v sinergiji z ostalimi značilnostmi najbolj močno opredeljujejo značaj in učinkovitost sistema enak z enakim kot celote.

### 3.3.1 Razpršenost

V klasičnem modelu odjemalcev in strežnikov so podatki, nadzor in izvajanje koncentrirani v središču – na strežniku. To dejstvo zagotavlja določeno udobje upravljalcu strežnika pri njegovem delu – na primer pri dodeljevanju pravic za dostop do sistema in pri zagotavljanju varnosti. Po drugi strani pa je strežnik potencialna tarča v primeru napada na sistem, lahko predstavlja ozko grlo in pogosto tudi vir neučinkovitosti ali mesto, kjer se kopičijo le malo izkoriščeni viri.




*Slika 3-2: Primer sistema z najvišjo stopnjo razpršenosti: poizvedbe in izmenjave lahko potekajo med poljubnimi pari udeležencev.*

Na drugi skrajnosti lestvice leži sistem s popolnoma razpršenimi podatki (primer prikazuje Slika 3-2) in izključno lokalnim, torej tudi popolnoma razpršenim nadzorom nad njimi. Vsi sodelujoči so po svojih vlogah in nalogah enakovredni, zato ni nikogar, ki

bi imel globalno gledano popoln pregled nad dogajanjem v sistemu. Tak sistem, imenujemo ga čisti enak z enakim, ima najvišjo stopnjo razpršenosti podatkov, nadzora in izvajanja.

Tabela 3.2 prikazuje na lestvici razpršenosti podatkov med že opisanima skrajnostma še dve vmesni stopnji, ki sta danes deležni velikega števila implementacij v sistemih za izmenjavo datotek: hibridni sistem z nizko stopnjo razpršenosti in sistem s super vozlišči, ki ima srednje visoko stopnjo razpršenosti.

Hibridni sistem ima nizko stopnjo razpršenosti nadzora in izvajanja, medtem ko so podatki že visoko razpršeni, torej prisotni na vseh udeležencih. Namenski strežnik navadno služi kot mesto, kjer se udeleženci registrirajo in sporočijo seznam vsebin, ki jih nudijo sistemu [73]. Tja udeleženci usmerjajo svoje poizvedbe in od tam dobijo odgovore. Šele nato se odločijo, ali želijo vzpostaviti stik z udeležencem, pri katerem se nahajajo iskane vsebine.

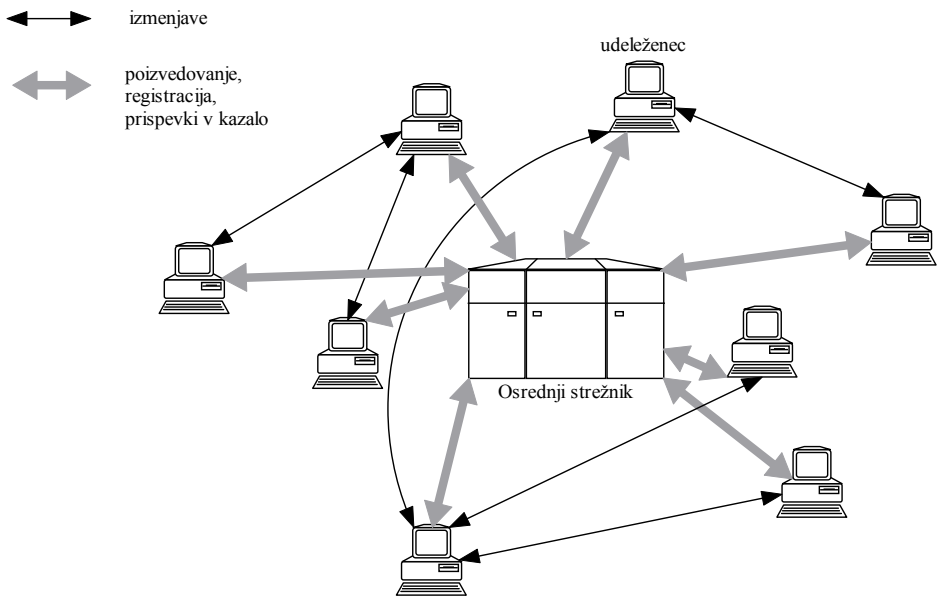
V sistemih, katerih primarna funkcija je porazdeljeno procesiranje, na primer seti@home [71] in njemu podobni sistemi, namenski strežnik razdeljuje udeležencem manjše naloge in skrbi za pravilno sestavljanje prispelih odgovorov. Če ugotovi, da kak odgovor morebiti manjka ali pa če dvomi v njegovo pravilnost, ustrezne naloge ponovno dodeli drugim udeležencem. Slika 3-3 prikazuje primer arhitekture hibridnega sistema.

| *RAZPRŠENOST* | |
|---|---|
| **VISOKA** | ▪ Čisti (nestrukturirani ali strukturirani) enak z enakim: vsi udeleženci so enakovredni in imajo popolnoma enake vloge. |
| **SREDNJA** | ▪ Super vozlišča, ultra vozlišča: v svoji soseščini imajo večjo vlogo (lokalni indeks), med seboj pa so enaka. |
| **NIZKA** | ▪ Hibridni enak z enakim: namenski strežnik lahko opravlja funkcijo indeksa, porazdeljuje delo, omogoča odkrivanje virov in povezovanje med udeleženci. |
| **NIČ** | ▪ Klasični sistem odjemalec – strežnik. |

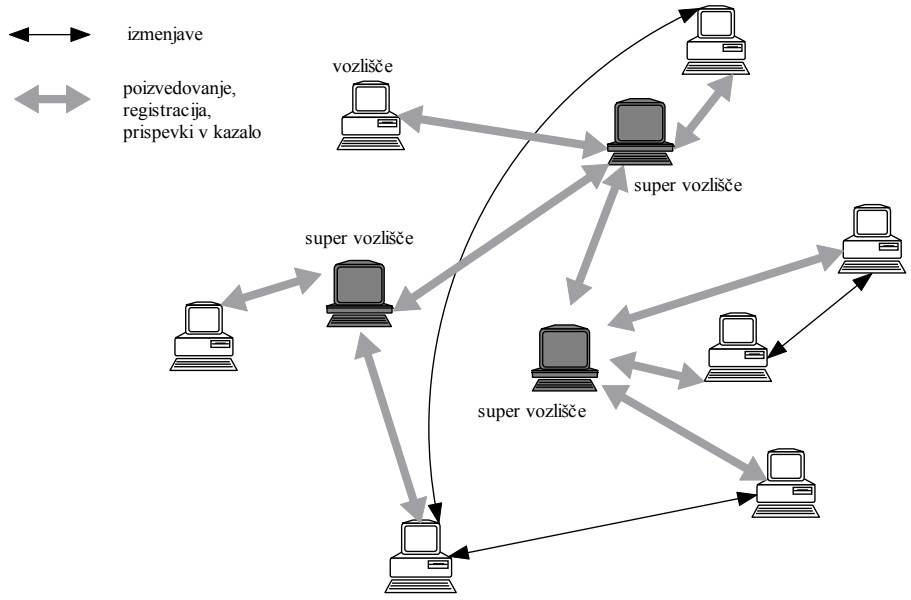
*Tabela 3.2: Stopnja razpršenosti podatkov in značilnosti sistema, ki jo implementira.*

V sistemu s super vozlišči je stopnja razpršenosti višja – v tabeli smo takšne sisteme označili s srednjo stopnjo razpršenosti. Podatki so visoko razpršeni, saj se nahajajo na vseh vozliščih. Glede nadzora in izvajanja pa so nekatera vozlišča "bolj enakovredna" kot druga – naj razložimo: sposobnejša in močnejša vozlišča z nadpovprečnimi komunikacijskimi kapacitetami se lahko razglasijo za super vozlišča. Topološka zahteva je, da se navadna vozlišča lahko povezujejo le na super vozlišča. Navadna vozlišča imajo nekoliko okrnjeno funkcionalnost, saj del njihovih nalog prevzame njihovo super vozlišče ([64], [62]).

Slika 3-4 prikazuje primer arhitekture sistema s super vozlišči. Super vozlišče posreduje poizvedbe od svojih navadnih vozlišč naprej na super vozlišča, s katerimi je povezano. Hrani tudi metapodatke o vsebinah na svojih navadnih vozliščih. Na prejete poizvedbe odgovarja v svojem imenu in v imenu svojih navadnih vozlišč in na ta način prihrani navadnim vozliščem nekaj procesiranja in veliko komunikacije. Šele ko se neko vozlišče odloči za prenos vsebine, vzpostavi neposredno povezavo z drugim navadnim vozliščem.




*Slika 3-3: Arhitektura hibridnega sistema z nizko stopnjo razpršenosti: samo izmenjave vsebin potekajo neposredno med udeleženci, za vse ostale funkcije pa se udeleženci obračajo na osrednji strežnik.*




*Slika 3-4: Arhitektura sistema s srednjo stopnjo razpršenosti:samo izmenjave vsebin potekajo neposredno med udeleženci, za ostale funkcije pa se udeleženci obračajo na svoje super vozlišče.*

Tako se vzpostavi dvonivojska hierarhija, kjer super vozlišča med seboj funkcionirajo kot čisti sistem enak z enakim, navadna vozlišča pa s super vozlišči tako, kot udeleženci v hibridnem sistemu enak z enakim z osrednjim strežnikom.

V velikih omrežjih je tudi število super vozlišč lahko zelo visoko, zato je razpršenost nadzora in izvajanja kljub lokalni centralizaciji še vedno definirana kot srednja. Takšna rešitev se je v praksi izkazala kot zelo učinkovita, saj predstavlja sprejemljiv kompromis med odrinjenostjo na rob za manj zmogljiva vozlišča in preprečevanjem pre-obremenjenosti komunikacijskega omrežja. Tak sistem ima v primerjavi s hibridnim enak z enakim bistveno nižjo ranljivost, obenem pa naleti pri velikem številu udeležencev na nivoju super vozlišč na enake težave glede usmerjanja in skalabilnosti kot čisti enak z enakim.

### 3.3.2 Razširljivost ali skalabilnost

Takoj ko nekatere funkcije osrednjega strežnika prenesemo na odjemalce oziroma ostale udeležence v sistemu, s tem pridobimo na razširljivosti sistema kot celote. Z besedo

razširljivost ali skalabilnost označujemo sposobnost sistema, da se učinkovito prilagaja večji ali manjši obremenitvi – različnemu številu udeležencev, različni količini zahtev, različnemu številu storitev, različnemu številu vozlišč, različni geografski porazdelitvi, različni velikosti omrežja in pomnilnih kapacitet, ter da ob tem ohranja sprejemljivo odzivnost in primerno kvaliteto storitev preko celotnega spektra konfiguracij sistema. Pri tem so majhni sistemi lahko prav tako pomembni kot veliki.

Osnovne metrike skalabilnosti, ki jih navajata Sun in Ni v [60], so bile razvite za ocenjevanje skalabilnosti paralelnih algoritmov na *k*-procesorskih računalnikih:

- **Pospešek** *S* (*speedup*) opisuje, kako narašča opravljeno delo na *k* procesorjih v primerjavi z delom na enem procesorju. V idealnem primeru bi bila to linearna funkcija:

  $$S(k) = k, \tag{3.1}$$

  v resnici pa bo vrednost navadno manjša od *k*.

- **Učinkovitost** *E* opisuje količino dela, ki jo opravi vsak izmed *k* procesorjev v primerjavi z delom na enem procesorju:

  $$E(k) = S(k) / k\ . \tag{3.2}$$

  V idealnem primeru bi bila ta vrednost enaka 1, realno pa bo vsak procesor opravil malo manj dela.

- **Skalabilnost** $\psi(k_1, k_2)$ od ene ($k_1$) do druge ($k_2$) konfiguracije meri razmerje med njunima učinkovitostma:

  $$\psi(k_1, k_2) = E(k_1) / E(k_2). \tag{3.3}$$

  Tudi ta ima v idealnem primeru vrednost 1, ki pa jo v praksi le redko dosegamo ali celo presegamo.

Te metrike so zelo koristne za samo razumevanje skalabilnosti, vendar pa so za splošne porazdeljene sisteme, med katere sodijo tudi sistemi enak z enakim, preveč poenostavljene. Ti sistemi ne izvajajo le enega algoritma, ampak več različnih nalog, potrebno je upoštevati njihovo različnost glede zahtevnosti; k učinkovitosti prispevajo tudi število uporabnikov, uporabljeni komunikacijski mehanizmi, zahteve glede kvalitete

storitev. Velikost sistema je kompleksna mera, saj zaobjema vse možne heterogene gradnike – procesorje, pomnilne zmogljivosti, komunikacijske poti, storitve, repliciranost in tako dalje.

Jogalekar in Woodside zato v [61] predlagata za kompleksne porazdeljene sisteme novo, splošno metriko, ki temelji na **produktivnosti** *F(k)*. Če se s spreminjanjem velikosti sistema produktivnost ohranja, potem za sistem lahko rečemo, da je skalabilen. Produktivnost sloni na treh temeljnih vrednosti:

- *R(k)*: prepustnost sistema - število odgovorov v sekundi, pri velikosti sistema *k*.
- *f(k)*: povprečna vrednost posameznega odgovora, ki jo dobimo glede na kvaliteto storitve, ko je velikost sistema enaka *k*. Funkcija *f* je lahko odvisna od poljubne mere: povprečna zakasnitev, verjetnost, da zakasnitev preseže nek prag, razpoložljivost, verjetnost izgube podatkov ali prekoračitve določenih vrednosti – odvisno od ciljev sistema. Zaloga vrednosti funkcije je lahko poljubna, pomembno je le, da za kvalitetnejši odgovor vrne večjo vrednost kot za manj kvalitetnega in da vrača pozitivna števila.
- *C(k)*: cena sistema velikosti *k*, preračunana na sekundo.

Produktivnost izrazimo kot razmerje med vrednostjo, pridobljeno v časovni enoti, in ceno sistema v časovni enoti:

$$F(k) = \frac{R(k)\, f(k)}{C(k)}. \tag{3.4}$$

Metrika skalabilnosti je izražena kot razmerje produktivnosti za dve konfiguraciji:

$$\Psi(k_1, k_2) = \frac{F(k_2)}{F(k_1)}. \tag{3.5}$$

Če se produktivnost ohranja, potem funkcija $\psi$ zavzema vrednosti okrog 1. Od posameznega primera je odvisno, kaj bo še sprejemljiva vrednost skalabilnosti.

S pomočjo teorije vrst Jogalekar in Woodside [61] razvijeta tudi analitične rešitve za skalabilnost devetih idealiziranih vrst sistemov, od katerih je za nas zanimiv zlasti "*sistem z neskalabilnim ozkim grlom in rastočo populacijo uporabnikov*". Ta je po svojih lastnostih podoben omrežjem enak z enakim, kjer ozko grlo predstavljajo

komunikacijske zmogljivosti. Ugotovila sta, da se skalabilnost takega sistema zmanjšuje s faktorjem $1 / k^2$, ko populacija raste.

Podobno ugotovitev so na svoji koži doživeli uporabniki sistemov za izmenjavo datotek, ki temeljijo na usmerjanju sporočil po principu poplavljanja. Preden je večina tovrstnih sistemov uvedla super vozlišča, so zaradi premajhnih komunikacijskih kapacitet velika omrežja začela razpadati na nepovezane komponente. Komunikacijske poti torej v takem sistemu predstavljajo ozko grlo. Z uvedbo super vozlišč se je količina sporočil zmanjšala, vendar je bil problem le prenesen na eno plast višje – v sistem super vozlišč. Z naraščanjem števila uporabnikov narašča tudi število super vozlišč in ne bo dolgo, ko se bodo težave ponovile. Edina rešitev je povečanje učinkovitosti usmerjevalnega mehanizma in s tem zmanjšanje skupnega števila prenosov sporočil.

Razširljivost sistema enak z enakim je navzgor omejena tudi s stopnjo centralizacije: s količino centraliziranih operacij, potrebnih za sinhronizacijo in usklajevanje; s količino podatkov, ki opisujejo stanje sistema in jih je potrebno nekje hraniti; s količino paralelnosti, ki je inherentna v obravnavani problematiki, končno tudi z uporabljenim programskim modelom in z zmogljivostmi centraliziranih komponent.

Današnji sistemi enak z enakim stremijo k zasnovi, s katero naj bi sistem zadovoljivo deloval tudi pri več milijonih uporabnikov, imeti bi moral torej zelo visoko skalabilnost.

Hibridni sistemi zahtevo po skalabilnosti rešujejo z zmogljivim osrednjim strežnikom, ki mora imeti izredno veliko pomnilno kapaciteto, visoko komunikacijsko zmogljivost in praktično 100% razpoložljivost. Tak strežnik je lahko zaradi večje zanesljivosti zgrajen v obliki gruče ali strežniške farme. Usklajevanje udeležencev zaradi centraliziranosti ni težavno. Seveda tudi tak strežnik predstavlja ozko grlo, ki pa ga je praviloma lažje širiti (na primer povečati gručo), kot bi bilo denimo povečati vse prenosne zmogljivosti v čistem nestrukturiranem sistemu enak z enakim svetovnih razsežnosti.

### 3.3.3 Anonimnost

V sistemih, kjer lahko do vsebin dostopa praktično kdorkoli, postaja anonimnost vedno pomembnejša. Cilj zagotavljanja anonimnosti je po eni strani preprečevanje možnosti cenzuriranja objavljenih vsebin, po drugi strani pa tudi zagotavljanje uporabnikom

sistema, da zaradi uporabe sistema ne bodo trpeli kakršnihkoli posledic (legalnih, moralnih,...) – ker uporaba pač ni sledljiva.

Včasih lahko med vrsticami najdemo namigovanja, da anonimnost potrebujejo le tisti uporabniki, ki bi radi neopaženo kopirali avtorsko zaščitene vsebine, zato poudarimo, da poznamo tudi številne popolnoma legitimne motive, ki podpirajo odločitev za zagotavljanje anonimnosti: preprečevanje cenzuriranja vsebin, svoboda izražanja in zaščita zasebnosti brez strahu pred morebitnim (legalnim) preganjanjem, preprečevanje zlonamernim udeležencem, da bi onemogočali objavo in distribucijo informacij,...

Definicija anonimnosti je po Pfitzmannu [11] naslednja: anonimnost je lastnost subjekta, da ga znotraj množice vseh možnih subjektov ne moremo identificirati.

Anonimnost je torej pogojena z množico vseh možnih subjektov, ki je lahko od primera do primera različna. Na primer: anonimnost pošiljatelja nekega sporočila je večja, če je možnih pošiljateljev veliko, in majhna, če je možnih pošiljateljev le malo.

Molnar in soavtorji v [2] kot očetje sistema enak z enakim Free Haven, ki naj bi omogočal čimbolj popolno anonimnost, razvrščajo oblike anonimnosti v šest skupin:

- **Anonimnost avtorja**: za posamezen dokument ali vsebino, ki je na voljo v sistemu, ni možno ugotoviti, kdo je njen avtor.
- **Anonimnost objavitelja**: za posamezen dokument ali vsebino, ki je na voljo v sistemu, ni možno ugotoviti, kdo jo je postavil v sistem oziroma objavil.
- **Anonimnost bralca**: ni možno ugotoviti, kdo vse je prenesel, bral, gledal ali kako drugače uporabljal neko vsebino.
- **Anonimnost strežnika**: za določeno vsebino ali dokument ni možno ugotoviti, na katerem strežniku oziroma na katerih vseh strežnikih se nahaja.
- **Anonimnost dokumenta**: strežnik ne ve, katere dokumente ali vsebine ima shranjene.
- **Anonimnost poizvedbe**: strežnik ne ve, s katerim dokumentom odgovarja na uporabnikovo poizvedbo.

Izraz *ni možno ugotoviti* v zgornjih navedbah je seveda potrebno razumeti z zdravo pametjo, kar pomeni, da ga lahko nadomestimo z izrazom *dovolj težko*.

Če hočemo v nekem sistemu uveljaviti določeno obliko anonimnosti, moramo vzpostaviti eno od oblik anonimnost komuniciranja: anonimnost pošiljatelja, anonimnost prejemnika ali celo obojestransko anonimnost.

Z anonimnostjo je tesno povezana lastnost **nepovezljivosti**: nepovezljivost dveh ali več stvari (subjektov, sporočil, dogodkov, akcij,...) v sistemu pomeni, da z vidika napadalca niso nič bolj in nič manj povezane, kot je že sicer njegovo a-priori znanje o njuni povezanosti. Drugače povedano, po standardu ISO IS 15408 [12] lastnost nepovezljivosti zagotavlja, da uporabnik sistema lahko uporabi več virov, ne da bi drugi mogli te dostope do virov povezati bodisi med seboj, bodisi z uporabnikom.

Anonimnost Pfitzmann dodatno definira s pomočjo nepovezljivosti: anonimnost pošiljanja ali sprejemanja sporočil je nepovezljivost kateregakoli od teh dveh dogodkov z identifikatorjem subjekta (pošiljatelja oziroma prejemnika). Šibkejša oblika anonimnosti je nepovezljivost pošiljatelja in prejemnika: morda lahko ugotovimo, kdo je poslal neko sporočilo ali kdo je sprejel neko sporočilo, ne pa kdo je komu poslal določeno sporočilo.

Najmočnejša zahteva pa predstavlja **neopaznost**: neopaznost je definirana kot lastnost objekta, da ga ni možno ločiti od ostalih objektov. Neopaznost komuniciranja pomeni, da »koristnih« sporočil ni možno ločiti od brezpredmetnega, navideznega (*dummy*) prometa ali naključnega šuma. Neopaznost pošiljatelja oziroma prejemnika zagotavlja, da ni možno ugotoviti, ali kateri izmed množice možnih pošiljateljev oziroma prejemnikov pošilja sporočila oziroma ali jih sprejema.

V nadaljevanju bomo našteli nekaj tehnik, s pomočjo katerih po Milojicicu [5] v sistemu enak z enakim lahko dosegamo določeno stopnjo nekaterih oblik anonimnosti.

- **Skupinsko oddajanje** (*multicast* in *broadcast*): ta oblika zagotavlja anonimnost prejemnika, saj se poizvedbe in vsebine pošiljajo na vse naslove iz skupine in morebitni napadalec in ostali člani skupine ne morejo ugotoviti, na kateri naslov je bilo sporočilo resnično namenjeno. Tehnika je najbolj učinkovita, če je večtočkovno oddajanje podprto že na nivoju komunikacijskega omrežja.
- **Hlinjenje pošiljateljevega naslova** (*spoofing*). Ta tehnika omogoča anonimnost pošiljatelja in je uporabna pri nepovezavnih protokolih (npr. UDP), vendar je potrebno protokol za to ustrezno prilagoditi. Dodatno težavo predstavljajo

številne požarne pregrade, ki zaradi velikega števila tovrstnih napadov filtrirajo neveljavne naslove. Zaradi naštetega hlinjenje ni ravno priporočljiva izbira.

- **Hlinjenje identitete.** Tudi hlinjenje identitete omogoča anonimnost pošiljatelja, vendar za nivo višje, na aplikacijski plasti: pošiljatelj se lahko predstavlja s tujo identiteto, torej izdaja za nekoga drugega. Vendar je stopnja anonimnosti nižja, saj tudi napadalec ve, da pošiljatelj lahko hlini identiteto.
- **Skrivne poti**. Namesto neposredne komunikacije pošiljatelj in prejemnik komunicirata preko enega ali več vmesnih vozlišč. Največkrat je s tem zagotovljena le anonimnost pošiljatelja. Vmesna vozlišča nastopajo kot navidezni ponori sporočil, vendar nato sporočila posredujejo naprej. Stopnja anonimnosti je odvisna od načina vzpostavljanja skrivne poti, od števila vmesnih vozlišč in od pogostosti spreminjanja vzpostavljenih poti.
- **Neprostovoljno umeščanje vsebin**. V sistemih, kjer umeščanje vsebin ni prepuščeno njihovemu lastniku, ampak pogojeno z nekim algoritmom, strežnik ni odgovoren za vsebine, ki so umeščene na njegovo lokacijo. Če so kriptirane (npr. v sistemu Freenet [9], [10]), strežnik niti ne more ugotoviti, za katere vsebine gre. S tem je zagotovljena anonimnost dokumentov.

V nestrukturiranih čistih sistemih enak z enakim, ki jih obravnava disertacija, se določena stopnja anonimnosti objavitelja dosega s pomočjo oddajanja sporočil na skupinske naslove, oziroma v terminologiji usmerjevalnih postopkov s pomočjo poplavljanja, pa tudi s pomočjo skrivnih poti. Anonimnost bralca se v nekaterih sistemih dosega s pomočjo skrivnih poti, anonimnost strežnika z neprostovoljnim umeščanjem, anonimnost dokumenta pa z enkripcijo.

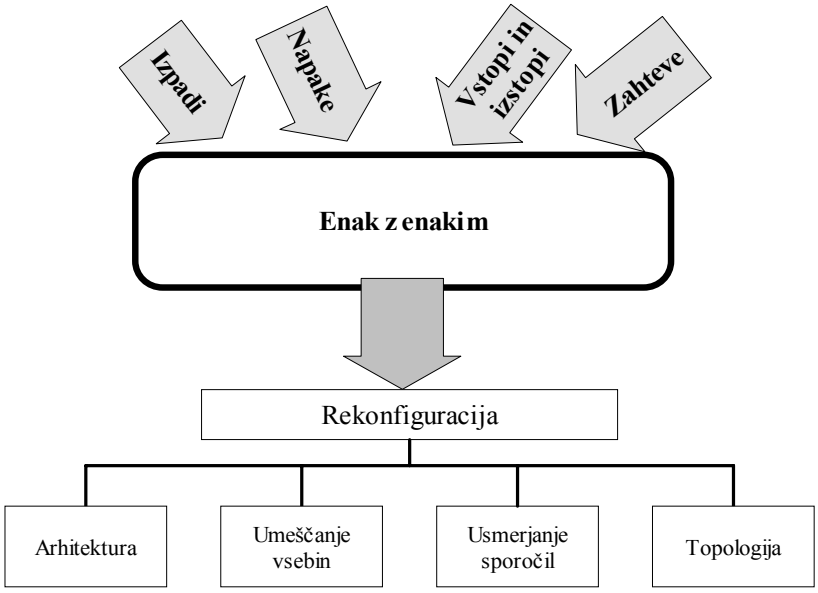
V idealnem sistemu bi si želeli zagotavljati vse ali čimveč oblik anonimnosti, obenem pa obdržati učinkovitost, razpršenost, odkrivanje virov brez posredovanja strežnika. Žal pa je zagotavljanje nekaterih oblik anonimnosti v nasprotju z omogočanjem nekaterih oblik funkcionalnosti sistema: na primer anonimnost strežnika močno omejuje učinkovitost iskanje, saj bi morali za vsak dokument točno vedeti, na katerih strežnikih se nahaja.

Zato se moramo pri načrtovanju omejiti bodisi glede anonimnosti, bodisi glede funkcionalnosti.

### 3.3.4 Dinamičnost in sposobnost samoorganizacije

Klasični porazdeljeni sistemi po vzpostavitvi obdržijo bolj ali manj fiksno obliko in arhitekturo, ki se spreminja le izjemoma. V nasprotju s tem je v sistemih enak z enakim pričakovano in popolnoma vsakdanje, da se udeleženci, skupine udeležencev ali celo večji deli sistema ves čas delovanja sistema na novo vključujejo v sistem in spet izstopajo iz njega. Uporabniki lahko pričakujejo, da se razpoložljive vsebine v sistemu menjajo, nekatere izginejo, obenem pa se pojavljajo nove.

Samoorganizacija sistema enak z enakim je nujno potrebna za zagotavljanje skalabilnosti, zanesljivosti delovanja in povezljivosti virov. Slika 3-5 prikazuje možne vplive in odgovore sistema nanje. Sistem se namreč glede velikosti (števila vozlišč in povezav), števila uporabnikov in obremenitve (števila zahtev) lahko spreminja popolnoma nepredvidljivo. Če bi bila organizacija centralizirana, bi bilo potrebno tekoče spremljanje vseh omenjenih in verjetno še kakih dodatnih kazalnikov, obenem pa bi bile nujne pogoste rekonfiguracije sistema. V primeru neprimernega trenutka rekonfiguracije bi lahko prihajalo do razpada sistema, do nedostopnosti virov in podobnih težav, obenem pa bi bilo mnogo rekonfiguracij popolnoma nepotrebnih.




*Slika 3-5: Vplivi na sistem enak z enakim in različni možni odgovori na potrebo po rekonfiguraciji.*

Sistem mora biti zasnovan tako, da vozlišča sama zaznavajo, kdaj je potrebna sprememba (največkrat topološka sprememba ali prilagajanje usmerjanja) v njihovi bližnji okolici in da v primernem trenutku to spremembo tudi izvedejo na način, ki je najbolj primeren. Pri obsežnih sistemih so pogoste napake in izpadi posameznih delov, zato je potrebno predvideti možnost samo-vzdrževanja in samo-okrevanja po napakah brez sodelovanja kake osrednje avtoritete. Prav tako je okrevanje potrebno v primeru napadov na omrežje, ker se v kratkem času iz sistema izključi večje število vozlišč.

Samoorganizacija se odvija na različnih področjih:

- **Arhitektura.** V sistemih s super vozliščih se kandidati za super vozlišča lahko prostovoljno javijo, če le izpolnjujejo dva pogoja: imeti morajo dovolj sposobno komunikacijsko povezavo in dovolj hitro morajo reagirati na zahteve, torej imeti dovolj visoko sposobnost procesiranja.
- **Umeščanje.** V sistemih z neprostovoljnim umeščanjem vsebin in v omrežjih z vsebinsko občutljivim umeščanjem se vsebine s pomočjo vsem vnaprej znanega algoritma umestijo na neko lokacijo v sistemu. Na podoben način se določa tudi, katere vsebine bodo v sistemu replicirani in na katerih lokacijah bodo umeščene replike. Algoritmi upoštevajo število zahtev po posameznih vsebinah in poskušajo umestiti replike čim bližje mestom, kjer se poraja največ zahtev.
- **Usmerjanje.** Za učinkovito usmerjanje sporočil v sistemu je potrebno konstantno preverjanje usmerjevalnih podatkov. Potreben je mehanizem za odstranjevanje zastarelih, neveljavnih podatkov in za čimbolj ažurno dodajanje novih, bolj svežih podatkov, ki omogočajo učinkovitejše usmerjanje. Skrajnost glede prilagodljivosti je usmerjanje s poplavljanjem: sporočilo vedno najde najučinkovitejšo pot, težave pa pri tem povzroča silno visoka redundantnost prenosov. Na drugem koncu lestvice pa bi bilo popolnoma prilagodljivo porazdeljeno usmerjanje po vzoru protokola RIP, ki trenutno ni implementirano v nobenem obstoječem sistemu, in kjer bi zaradi izmenjave velike količine usmerjevalnih podatkov spet morali sklepati kompromise med ažurnostjo usmerjanja in redundantnostjo prenosov. Več o tem bomo opisali v poglavju o simulacijah usmerjevalnih protokolov.

- **Topologija.** Nekaj avtorjev ([18],[29],[50]) je že pisalo o prilagajanju topologije navideznega omrežja potrebam ali interesom udeležencev, vendar je to področje še vedno zelo neraziskano. V navideznih omrežjih je možno vzpostaviti povezavo med poljubnima dvema udeležencema brez posebnih dodatnih organizacijskih ali cenovnih obremenitev. V primerjavi s fizičnimi omrežnimi topologijami je to velika prednost, ki bi lahko omogočala pametno oblikovanje in rast topologije navideznega omrežja v taki smeri, da bi že sama po sebi omogočala čim manjše število prenosov sporočil in s tem učinkovitejše iskanje. V obstoječih sistemih pa najdemo dve skrajnosti: bodisi je topologija vnaprej predpisana, kot na primer v visoko strukturiranih sistemih (večina vsebinsko občutljivih omrežij), ali pa je prepuščeno vsakemu udeležencu, da se poveže, kamor se želi.

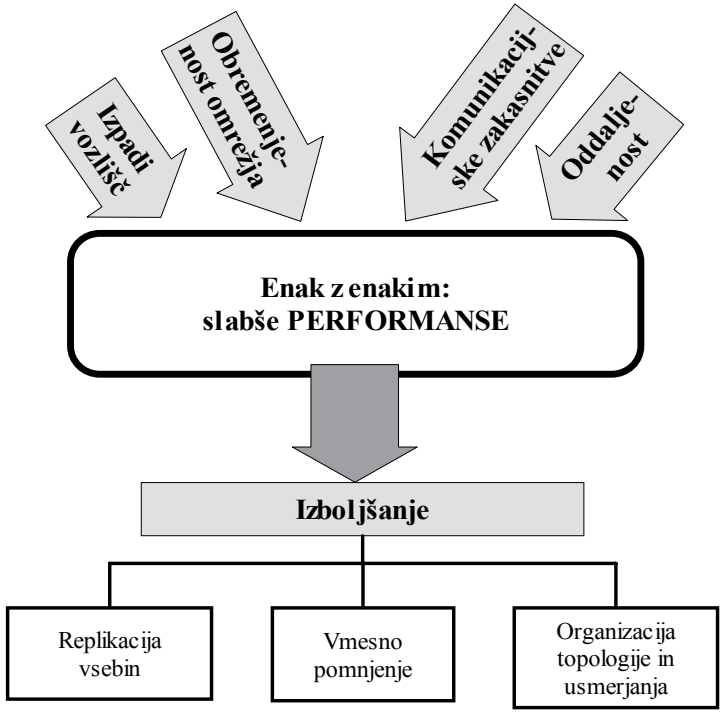
Sposobnost samoorganizacije je nujna predvsem za sisteme z višjo stopnjo razpršenosti, saj tu ni osrednje avtoritete, ki bi lahko izvajala centraliziran nadzor in reorganizacijo. V sistemih z nižjo razpršenostjo, na primer v hibridnih enak z enakim, je potreba po samoorganizaciji opazno nižja.

### *3.3.5 Zmogljivosti sistema*

Za načrtovalce sistemov enak z enakim zmogljivosti predstavljajo enega največjih problemov. Stremenje načrtovalcev k doseganju čim višjih zmogljivosti za različne problemske domene je tudi vzrok, da danes opažamo zelo široko množico različnih sistemskih zasnov.

Razpršenost osnovnih treh virov, namreč procesnih, komunikacijskih in pomnilnih kapacitet, pogojuje robne pogoje za doseganje višjih zmogljivosti. Za odzivni čas so odločilne zlasti omrežne zakasnitve, ki so pri omrežjih svetovnih razsežnosti vse prej kot zanemarljive. Ob visoki obremenitvi komunikacijskih poti, na primer zaradi velikega števila prenosov protokolarnih sporočil ali vsebin med udeleženci, je odzivnost še slabša.

Sistemi enak z enakim se za izboljšanje odzivnosti lahko poslužujejo ene ali več izmed treh ključnih tehnik, ki jih pregledno prikazuje Slika 3-6.




*Slika 3-6: Vplivi na slabšanje zmogljivosti sistema in tehnike za nevtraliziranje teh vplivov.*

- **Replikacija.** S pomočjo replikacije umeščamo kopije vsebin na vozlišča, ki so blizu izvorom poizvedb po teh vsebinah. Tako se zmanjša razdalja med izvorom in ponorom vsebin in posredno tudi dolžina verige posredovanj poizvedb in odgovorov, obenem pa odpravimo tudi težave, povezane z izginotjem vsebine, potem ko udeleženec – osnovni ponudnik vsebine zapusti sistem.

  V sistemih z neprostovoljnim umeščanjem vsebin je možno pravila za umeščanje replik zasnovati tako, da je njihova koristnost maksimizirana, da se torej umestijo čim bližje porabnikom vsebin. V sistemih z višjo avtonomijo udeležencev, kjer je umeščanje vsebin zgolj prostovoljno, je stopnja replikacije sicer visoka, vendar mesta replik niso nujno blizu porabnikom vsebin. Učinkovitost replikacije v takih sistemih je zato nižja.

- **Vmesno pomnjenje vsebin in sporočil** (*caching*). Nekatera vozlišča na poti od izvora do ponora lahko za krajši čas shranijo vsebine ali odgovore na poizvedbe in s tem zmanjšajo skupno število prenosov sporočil ali vsebin.

- **Ustrezna organizacija navideznega omrežja in usmerjanja sporočil.** Zaradi samoorganiziranosti navideznega omrežja se v njem izrazijo tudi lastnosti, ki so pogojene s socialnimi interakcijami udeležencev. Ena od takih lastnosti je topološka lastnost majhnega sveta, ki sta jo prva opisala Watts in Strogatz [24][25][26]. V sistemih enak z enakim so pojave majhnega sveta preučevali Ripeanu, Foster in Jovanović [16][19]. Druga pomembna topološka lastnost pa je obstoj potenčnih zakonov, ki so jih identificirali bratje Faloutsos [15]. Obe lastnosti bomo podrobneje opisali v poglavju o topologiji navideznega omrežja.

  Za doseganje dobrih odzivnih časov sistema je smiselno izkoristiti omenjene topološke značilnosti navideznega omrežja, jim prilagoditi način usmerjanja sporočil in s tem zmanjšati skupno število prenosov.

  Prav tako je smiselno v sistemih, kjer za vzpostavljanje topologije veljajo vnaprej določena pravila, ta pravila zasnovati tako, da bo tudi nastala topologija podpirala učinkovito posredovanje sporočil in vsebin.

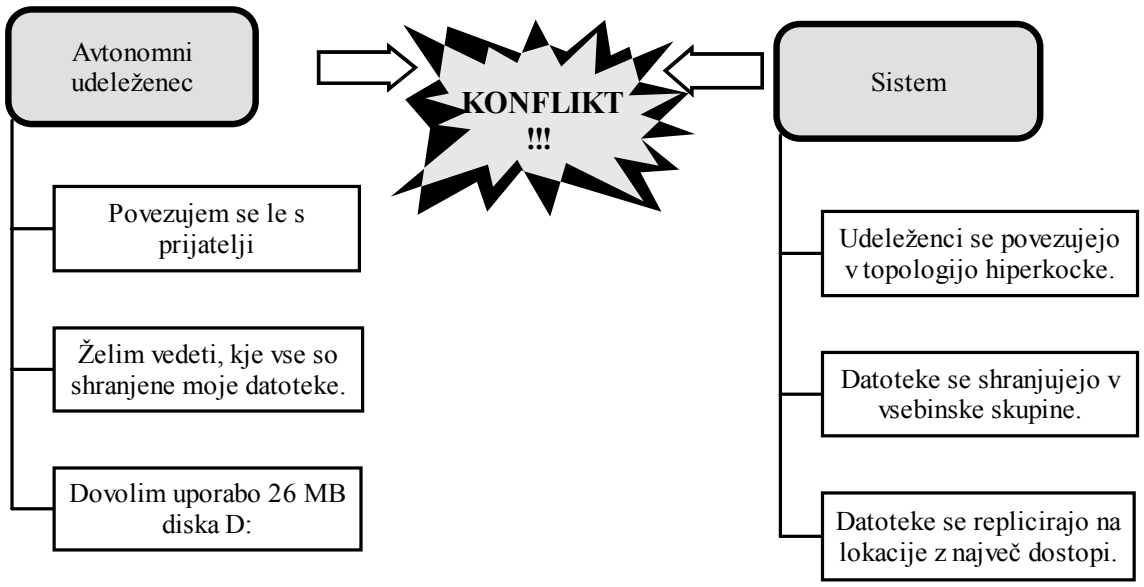
Kljub uporabi zgornjih treh tehnik učinkovitost sistemov enak z enakim še ni dosegla svojega vrhunca, saj je zlasti na področju organizacije omrežja in ustreznosti posameznih usmerjevalnih postopkov potrebno vložiti še precej raziskovalnih naporov. Uporaba učinkovitih mehanizmov pa je pogojena tudi s stopnjo avtonomnosti udeležencev, kar podrobneje opisujemo v naslednjem razdelku.

### *3.3.6 Avtonomnost*

V vsakem sistemu se morajo udeleženci podrediti določenim pravilom. Stopnja avtonomnosti udeleženca označuje, v kolikšni meri se je pripravljen podrejati pravilom. Drugače povedano, stopnja avtonomije opredeljuje, v kakšne sisteme se je pripravljen vključevati: v takšne, ki od njega ne zahtevajo več, kot je pripravljen dati ali storiti.

Končni uporabnik je sebičen: svoj namizni računalnik in vire je pripravljen nuditi sistemu enak z enakim le, če to ne bo šlo na račun kvalitete storitev, ki mu jo računalnik običajno nudi. Zato želi imeti nad svojimi viri nadzor, želi imeti moč, da jih umakne iz sistema, če se mu to zazdi potrebno, da jih omeji (na primer količino diska, ki ga nudi v sistem, ali del pasovne širine) ali da dovoli dostop do njih le udeležencem, ki jim zaupa.

Če mehanizmi sistema enak z enakim omejujejo obnašanje, ki ga uporabnik želi nadzirati, s tem omejujejo njegovo avtonomnost. Udeleženci se v takem sistemu ne počutijo dobro, saj jim ne nudi pričakovane funkcionalnosti, zato bodo iz njega čimprej izstopili. Primer takega konflikta prikazuje Slika 3-7: udeleženec sistema želi biti avtonomen, obenem pa pričakuje določeno stopnjo kvalitete storitev. Sistem želi pri določeni stopnji kvalitete storitev doseči čim večjo učinkovitost, torej čim boljšo izrabo virov in manj redundance. To najlaže doseže, če udeležencem postavi določena pravila, ki pa žal omejujejo njihovo avtonomnost.




*Slika 3-7: Konflikt med interesi udeleženca (visoka avtonomija, kvaliteta storitev) in interesi sistema (učinkovitost sistema kot celote).*

Pri zasnovi sistema je zato nujno upoštevati, komu je sistem namenjen, in ustrezno izbirati in prilagajati mehanizme (zlasti iskalne mehanizme in način umeščanja vsebin). Žal sta učinkovitost mehanizmov in avtonomnost udeležencev med seboj obratno sorazmerna, zato moramo v sistemih z visoko avtonomnostjo vzeti v zakup več redundance in nekaj slabše performanse.

Če je poudarek na hranjenju in pametni organizaciji vsebin ter na razpoložljivosti sistema, uveljavimo strikten nadzor nad umeščanjem vsebin in oblikovanjem topologije navideznega omrežja. Tako lahko zagotavljamo, da bomo vsako vsebino našli v omejenem času – če le obstaja v sistemu, vendar pa žrtvujemo avtonomijo vozlišč.

Primer so vsebinska omrežja z vsebinsko občutljivim umeščanjem Chord [46], Pastry [47], Tapestry [48], in CAN [45] in njihove nadgradnje.

Če pa je večji poudarek na avtonomnosti udeležencev in na široki, splošni uporabi, medtem ko garancija razpoložljivosti ni potrebna, si sistem ne more privoščiti strogega nadzora nad umeščanjem vsebin in oblikovanjem topologije. Tak sistem deluje pod popolnoma drugačnimi pogoji in kriteriji, primeri pa so Gnutella [62], Freenet [63], KazaA [64], Morpheus [66], eDonkey [74], ...

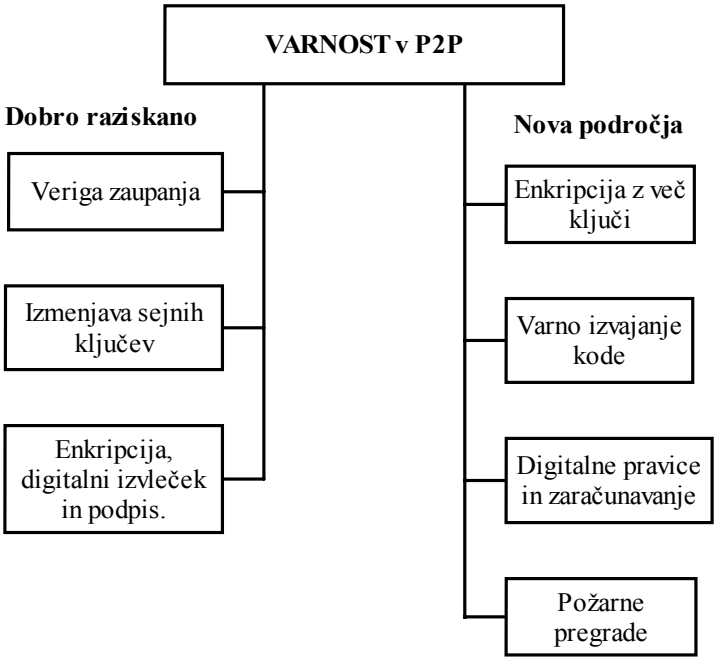
Z avtonomnostjo pa ni v tesni povezavi le učinkovitost, ampak tudi robustnost sistema. Robustnost sistema opisuje njegovo stabilnost v primeru nepredvidenih napak, predvsem pogostih vključevanj udeležencev v sistem in izstopov iz njega in posledične (ne)dostopnosti virov. Sistemi z bolj avtonomnimi udeleženci take napake lažje prebolijo, so torej stabilnejši in robustnejši. Formalni sistemi s togo organiziranostjo in nižjo avtonomijo udeležencev pa so okornejši tudi pri prilagajanju na nove razmere, torej manj robustni v visoko dinamičnem okolju.

Največja pridobitev za področje enak z enakim bi bil mehanizem, ki bi učinkovito in robustno urejal povezovanje (topologijo) in iskanje na tak način, da bi ohranil visoko avtonomnost udeležencev.

### *3.3.7 Varnost*

Na področju porazdeljenih sistemov je varnost že dobro razdelana problematika. Sistemi enak z enakim imajo glede varnosti podobne potrebe: gradnja verige zaupanja med udeleženci, sheme za izmenjavo sejnih ključev, enkripcija, digitalni izvlečki in podpisi. Vsak sistem uporablja le nekatere od naštetih mehanizmov, pač glede na svoj namen in zasnovo.

Pojavile pa so se tudi nove potrebe, ki jih v porazdeljenih sistemih do pojava arhitekture enak z enakim nismo poznali: kriptiranje z večjim številom tajnih ključev, izvajanje kode na gostitelju na varen način v ločenem prostoru, upravljanje z digitalnimi pravicami (predvsem avtorskimi), izvedljivost zaračunavanja uporabe sistema, komunikacija prek požarnih pregrad. Slika 3-8 prikazuje shemo s pregledom teh področij.




*Slika 3-8: Aktualni problemi na področju varnosti.*

Kriptiranje na način z enim javnim in več tajnimi ključi (*multi-key encryption*) se uporablja v sistemih z velikim poudarkom na anonimnosti, katerih cilj je med drugim tudi onemogočanje cenzuriranja objavljenih vsebin. Zato želijo objavljene vsebine še posebej zaščititi pred morebitnimi zlonamernimi posegi vanje.

Predvsem v sistemih za porazdeljeno računanje je potrebno, da udeleženci izvajajo programsko kodo, o kateri ne vedo nič. Da ne bi utrpeli kakih škodljivih posledic, je potrebno, da se ta koda izvaja popolnoma ločeno od ostalih virov udeleženca – v varnem okolju in na varen način.

V sistemih za izmenjavo datotek velik problem predstavlja nespoštovanje avtorskih pravic. Idejno problem lahko enačimo z nadzorom dostopa do vsebin: neko glasbeno datoteko lahko prenesejo le tisti uporabniki, ki so to plačali na ustrezen način. Nekateri avtorji predlagajo, da bi korak v pravo smer predstavljalo vključevanje t.i. vodnega tiska (steganografija) z zapisom o avtorskih pravicah v sporne glasbene ali filmske datoteke. Na podlagi tega zapisa bi ustrezno ukrepala aplikacija za izmenjavo vsebin. Vprašanje je tudi, ali bi bilo res pametno nadzor vključiti v sistem kot dodatni mehanizem, ali pa se lahko zanašamo na uporabnike, da bodo ravnali v skladu z zakoni.

O zaupanju nekemu udeležencu govorimo, kadar se prepričamo, da resnično ima identiteto, s katero se predstavlja. Nad zaupanjem se gradi ugled: uglednejši udeleženci so tisti, ki prispevajo v zakladnico celotnega sistema več lastnih virov. Neposredno od tod se poraja ideja o zaračunavanju uporabe sistema tistim udeležencem, ki v sistem ne prispevajo nič in imajo torej nizek ugled. Zaračunavanje v denarju je problematično, ker udeležencem načeloma ne zaupamo, četudi se v sistemu avtenticirajo, zato udeležencem najlažje »zaračunamo« njihovo neuglednost tako, da mu damo na primer nižjo prioriteto pri dostopanju do skupnih virov ali pa ga odrinemo na bolj oddaljen konec topologije.

Požarne pregrade predstavljajo oviro neposrednemu povezovanju enakovrednih udeležencev, saj so pogosto konfigurirane tako, da blokirajo povezave, ki bi jih zunanji udeleženci želeli vzpostaviti z udeleženci v varovanem območju. Možne so le povezave od znotraj navzven. Še večja težava nastopi, če sta oba udeleženca za požarnima pregradama: povezava med njima je možna le s pomočjo zunanjega posrednika.

## 3.4 Povzetek

Zamisel o arhitekturi enak z enakim se je porodila zaradi potrebe po večji razpršenosti, po podpori bolj neorganiziranemu obnašanju udeležencev, po anonimnosti in končno tudi po razpršitvi stroškov lastništva, zato so na teh področjih sistemi enak z enakim gotovo danes najprimernejša oblika organiziranosti. Za določene problemske domene so zelo primerni tudi kar se tiče zmogljivosti, skalabilnosti, varnosti, samoorganiziranosti in odpornosti na napake in usmerjene napade, saj jim razpršenost pri tem nudi določene prednosti. Ob tem pa nudijo le malo transparentnosti: uporabnik mora na primer posredovati pri izvedbi vključevanja v sistem, pri zagotavljanju metapodatkov, pri okrevanju po napakah in podobno.

Največja slabost sistemov enak z enakim je nedorečeno področje varnosti, saj posamezni udeleženci ničesar ne vedo drug o drugem. Tu bi bilo potrebno zgraditi sistem, ki bi omogočal vsakemu udeležencu graditi zaupanje in svoj ugled na podlagi prispevkov v sistem, zanesljivosti delovanja, širine komunikacijskih poti, procesorskih zmogljivosti in morda tudi ocene ostalih udeležencev.

Slabost obstoječih sistemov je tudi njihova nepovezljivost, saj imamo na različnih področjih mnogo med seboj podobnih si sistemov, ki pa med seboj niso združljivi. Deloma je za to kriva tudi mladost področja in odsotnost standardov, zato smo prepričani, da se bo stanje z vse večjo popularnostjo tovrstne arhitekture kmalu izboljšalo. Široka uporaba je tudi ključen dejavnik uspeha: ker se udeleženci nudijo storitve en drugemu, sistem z zmanjšanjem števila udeležencev pod neko kritično maso preprosto ugasne, saj preneha nuditi zadovoljive storitve (v nasprotju s sistemom odjemalec – strežnik, ki zadovoljivo funkcionira vse dokler deluje strežnik). In obratno: z večanjem števila udeležencev, ki nudijo svoje vire, so storitve sistema bolj kakovostne in privlačnejše, kar posledično v sistem privabi še več uporabnikov – meja je le skalabilnost sistema, predvsem algoritmov.

# 4 LASTNOSTI OMREŽIJ TIPA A

V poglavju 2 smo naredili izčrpen pregled vsebinskih omrežij, pri čemer smo napovedali, da se bomo v nadaljevanju še poglobili v lastnosti izbranih omrežij, torej vsebinskih omrežjih tipa A.

Na kratko ponovimo: vsebinska omrežja so navidezna omrežja na aplikacijski plasti, ki podpirajo vsebinsko usmerjanje sporočil. Vsebinsko usmerjanje sporočil pomeni, da se sporočilo prek omrežja ne usmerja na podlagi ciljnega naslova, temveč na podlagi vsebine sporočila. V vsebinskih omrežjih tipa A se vsebine združujejo v skupine sintaktično, pogosto pa se sploh ne združujejo. Umeščanje vsebin ni občutljivo za vsebino, posamezna vsebina se torej lahko nahaja kjerkoli v sistemu.

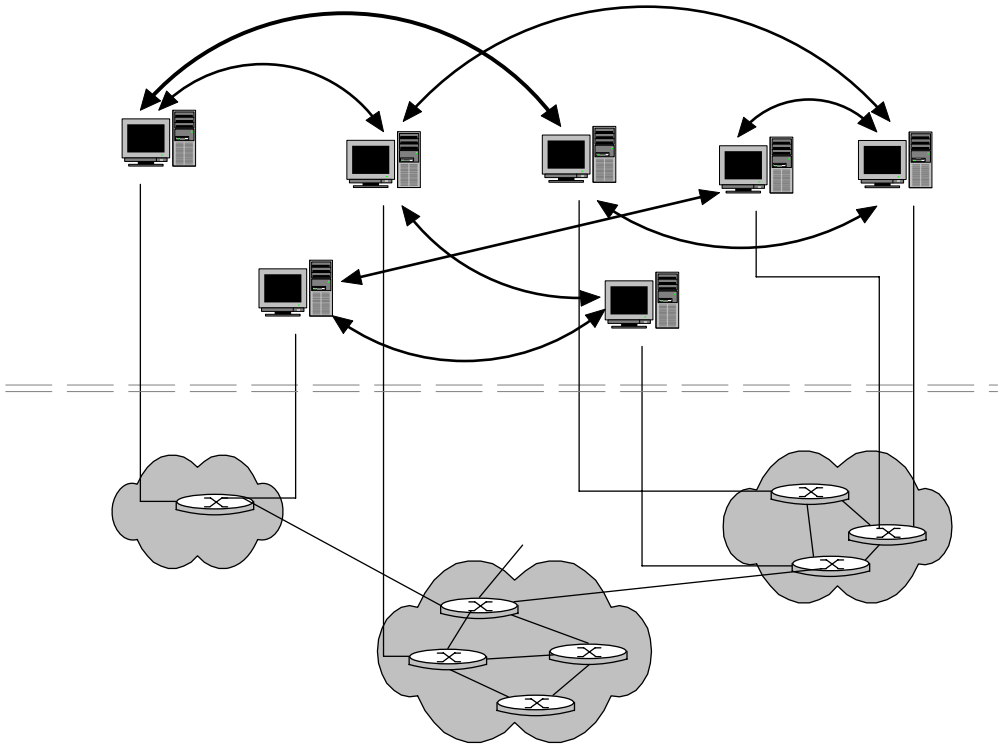
Velika večina današnjih sistemov enak z enakim za izmenjavo datotek predstavlja vsebinska omrežja tipa A. Mnogo vsebinskih omrežij ima arhitekturo enak z enakim.

Usmerjanje sporočil v omrežjih tipa A pogosto predstavlja težave s stališča zmogljivosti, saj je zaradi pomanjkanja usmerjevalnih informacij poizvedba toliko verjetneje uspešna, kolikor več vozlišč obišče. V vsebinsko občutljivih vsebinskih omrežjih je usmerjanje sporočil pogojeno z načinom umeščanja vsebin, ta pa je zasnovan ravno z namenom omogočiti učinkovito usmerjanje. V vsebinskih omrežjih s semantičnim združevanjem pa je usmerjanje zadovoljivo, ker je dovolj, da se najprej najde ustrezna skupina vsebin, nato pa šele znotraj nje iskana vsebina.

Vsebinska omrežja tipa A pa so zaradi svojih lastnosti posebno primerna za implementacijo splošno namenskih sistemov za izmenjavo datotek, v katere se lahko vključi vsak končni uporabnik in kjer hočejo udeleženci ohraniti visoko avtonomnost in nadzor nad "svojimi" vsebinami. Čeprav so omrežjem tipa A zaradi njihove nižje učinkovitosti nekateri avtorji izrazito nenaklonjeni [35], se moramo strinjati s preostalimi (na primer [16],[19],[30],[36],[57]), ki menijo, da bodo ravno zaradi avtonomnosti udeležencev vsebinska omrežja tipa A še dolgo časa najširše uporabljani sistemi enak z enakim, zato je iskanje načinov za doseganje boljših zmogljivosti še kako smiselno.

## 4.1 Topologija navideznega omrežja

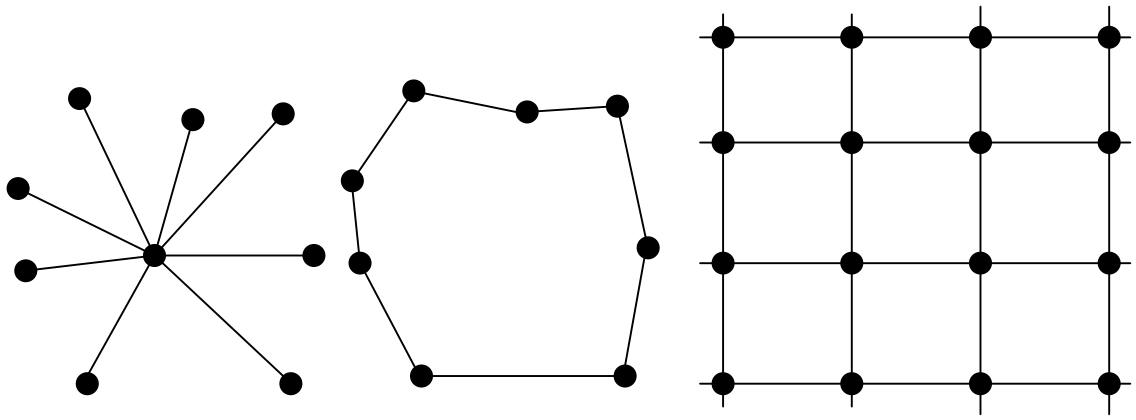
Navidezno omrežje navadno imenujemo množico aplikacijskih povezav med odjemalskimi procesi, ki tečejo v končnih, uporabniških računalnikih. Vozlišča navideznega omrežja predstavljajo procesi, ne nujno posamezni končni računalniki, saj lahko znotraj enega računalnika načeloma teče več istovrstnih procesov, ki jih obravnavamo kot samostojna vozlišča.



*Slika 4-1: Odnos med strukturo omrežja in strukturo navideznega omrežja.*

Navidezno omrežje ali "omrežje nad omrežjem" (angleško *overlay*) se oblikuje na aplikacijski plasti in ni odvisno od topologije omrežne plasti, torej od lokacije usmerjevalnikov in povezav med njimi. Primer odnosov med strukturo omrežja in strukturo navideznega omrežja prikazuje Slika 4-1. Navidezno omrežje predstavlja za omrežno plast manjšo obremenitev, kadar se njegove povezave lepo prilegajo fizičnim povezavam, oziroma večjo obremenitev, kadar mnogo povezav navideznega omrežja povezuje s stališča mrežne plasti močno oddaljene točke.

Za nas je najpomembnejša lastnost navideznega omrežja njegova topologija. Topologija v teoriji grafov predstavlja razpored vozlišč in povezav. Pravilne topologije so na primer zvezda, obroč, mrežasta struktura in podobno (Slika 4-2). Vendar pa se realna navidezna omrežja ne ponašajo z lastnostjo pravilnosti, saj največkrat nastajajo na videz stihijsko, neurejeno. Zaradi velikosti, ki lahko dosega od več deset tisoč do celo milijon vozlišč, jih je težko vizualizirati na smiseln način, lahko pa izračunamo nekaj njihovih zanimivih značilnosti.



*Slika 4-2: Primeri pravilnih topologij.*

- Število vozlišč grafa je podatek, ki nam pravzaprav o topologiji ne pove nič, koristen pa je kot mera velikosti grafa.
- Število povezav grafa. Tudi ta podatek nam ne pove dosti, saj je pomemben tudi razpored povezav. V povezavi s številom vozlišč pa lahko ugotovimo, kako poln je graf, oziroma kako močno povezan – kolikšno je razmerje med dejansko obstoječim številom povezav in vsemi možnimi povezavami na takem številu vozlišč.
- Stopnja vozlišč. Število vseh povezav nekega vozlišča imenujemo stopnja vozlišča. Povprečna stopnja vozlišča v grafu nam pove, kako gosto je nek graf povezan. Včasih nas bolj kot povprečna stopnja zanima frekvenčna porazdelitev stopenj v grafu, torej podatek, koliko vozlišč v grafu ima določeno stopnjo.
- Najkrajše poti. Najkrajša pot med dvema vozliščema je minimalno število skokov, ki je potrebno, da iz enega dosežemo drugo vozlišče. Zanima nas

povprečna najkrajša pot, včasih pa tudi najdaljša izmed najkrajših poti med vsemi možnimi pari vozlišč.

- Koeficient gručenja (*clustering coefficient*). Koeficient gručenja je mera, ki nam pove, v kolikšni meri so vozlišča v grafu povezana v manjše gruče, ki so gosteje povezane kot graf v celoti. Za posamezno vozlišče $v_i$ izračunamo koeficient gručenja kot razmerje med številom povezav med sosedi vozlišča $v_i$ in številom vseh možnih povezav (poln graf) med sosedi vozlišča $v_i$:

Mnogo omrežij iz realnega sveta, ne samo s področja računalništva, ampak tudi matematična, biološka in socialna, lahko uvrstimo med nehomogena omrežja, neodvisna od velikosti (*scale-free network*). V takih omrežjih ima manjši del vozlišč zelo veliko število povezav, večina preostalih vozlišč pa ima povezav le malo. Velja tako imenovan potenčni zakon:

$$P(k) \approx k^{-\alpha}, \tag{4.1}$$

kjer *P(k)* predstavlja verjetnost, da je vozlišče povezano s *k* drugimi vozlišči, $\alpha$ pa je konstantna potenca, značilna za določeno vrsto grafa. Potenčnim zakonom se pokorava topologija interneta, graf povezav svetovnega spleta (WWW) in tudi topologije navideznih omrežij nestrukturiranih vsebinskih omrežij z arhitekturo enak z enakim.

Raziskovalci domnevajo, da pride do pojava potenčnih zakonov zato, ker se v topologijo ves čas vključujejo nova vozlišča, ta pa se rada priključujejo na takšna vozlišča, ki so že sama dobro vpeta v omrežje, drugače rečeno, imajo že sama veliko število povezav.

V nadaljevanju bomo potenčne zakone razložili podrobneje, poleg tega pa bomo opisali še soroden topološki pojav, imenovan značilnost majhnega sveta.

### 4.1.1 Potenčni zakoni

Z analizo topologije navideznega omrežja lahko pridemo do kriterijev, kako zasnovati usmerjevalne protokole, da bodo čimbolj učinkoviti in čim manj redundantni. Zakonitosti, ki jih odkrivamo v realnih posnetkih topologij pa nam tudi pomagajo generirati umetne topologije, kar je pogosto neizogibno potrebno za analizo obnašanja ter testiranje in primerjavo protokolov.

Najbolj odmevne štiri potenčne zakone so leta 1999 predstavili bratje Faloutsos [15], odkrili pa so jih v topologiji interneta, ki so ga opazovali na nivoju avtonomnih sistemov oziroma administrativnih domen. Čeprav se internetna topologija razvija in raste na videz naključno, potenčni zakoni predstavljajo regularnost in predvidljivost tudi v takšni strukturi. Eksponenti potenčnih zakonov opisujejo značilnosti posameznih tipov grafov.

#### *4.1.1.1 Zakon ranga*

Vozlišča grafa uredimo po padajoči stopnji vozlišča. Rang vozlišča *v,* označimo ga z $r_v$, predstavlja mesto vozlišča v tem urejenem zaporedju. Zakon ranga pravi trdi, da je izhodna stopnja $d_v$ vozlišča *v* proporcionalna rangu $r_v$ vozlišča *v*, potenciranega na neko konstanto *R*:

$$d_v \propto r_v^R \tag{4.2}$$

*R* je konstanta, ki je značilnost posameznega sistema in je torej v različnih sistemih različna. Če narišemo graf odvisnosti izhodne stopnje $d_v$ od ranga $r_v$, kjer sta skali na obeh oseh logaritemski, dobimo graf, ki ima točke razvrščene skorajda na padajoči premici. S pomočjo linearne regresije lahko potegnemo premico skozi te točke in korelacijski koeficient se močno približa vrednosti 1, zato lahko rečemo da je odvisnost linearna. Nagnjenost premice opisuje konstanta *R*.

#### *4.1.1.2 Zakon stopnje*

Drugi potenčni zakon opisuje porazdelitev stopenj vozlišč v grafu (če je graf usmerjen, opisujemo le izhodne povezave). Frekvenca stopnje vozlišč $f_d$ predstavlja število tistih vozlišč v grafu, ki imajo izhodno stopnjo enako *d*.

Zakon stopnje trdi, da je frekvenca izhodnih stopenj vozlišč $f_d$ proporcionalna izhodni stopnji *d*, potencirani na konstanto *O*.

$$f_d \propto d^O \tag{4.3}$$

Posledično v takem grafu stopnja vozlišč ni poljubna, pač pa je v njem mnogo več vozlišč, ki imajo nižjo stopnjo kot vozlišč z višjo stopnjo. Tudi zakon stopnje lahko predstavimo s premico na grafu, kjer sta skali na obeh oseh logaritemski.

#### *4.1.1.3 Zakon skokov*

Tretji potenčni zakon opisuje oddaljenost med vozlišči, točneje velikost soseščine na oddaljenosti *h* ali manj skokov. *P(h)* predstavlja število vseh parov vozlišč v grafu, ki so med seboj oddaljena za največ *h* skokov, pri tem čemer so pari urejeni, zato vsakega

štejemo dvakrat: (*v1, v2*) in (*v2, v1*). Upoštevamo tudi pare (*v, v*), kjer je dolžina poti enaka 0. Če je *h* enak 0, potem *P(h)* predstavlja število vseh vozlišč v grafu. Če je *h* enak premeru $\delta$ grafa G (t.j. najdaljši najkrajši poti), dobimo maksimalno število parov $n^2$.

Graf odvisnosti števila parov vozlišč *P(h)* od števila skokov *h,* kjer sta skali na obeh oseh logaritemski, spet lahko aproksimiramo s padajočo premico. Zakon oddaljenosti pa nam pove, da je število vseh parov vozlišč *P(h)*, ki so oddaljena največ za *h* skokov, proporcionalno številu skokov *h,* potenciranemu na konstanto *H*, pri čemer pa mora biti število skokov *h* veliko manjše od premera grafa $\delta$.

$$P(h) \propto h^H; \quad h << \delta \ . \tag{4.4}$$

#### *4.1.1.4 Zakon lastnih vrednosti grafa*

Graf lahko predstavimo kot matriko, v kateri vsako vozlišče dobi svoj stolpec in vrstico, posamezno polje pa predstavlja povezavo. Če sta vozlišči povezani, je v polju vrednost 1, sicer pa 0. Če za takšno matriko lahko izračunamo lastne vrednosti, nam četrti potenčni zakon govori, da je lastna vrednost $\lambda_i$ povezovalne matrike proporcionalna indeksu *i* v zaporedju padajočih lastnih vrednosti, potenciranemu na konstanto *E:*

$$\lambda_i \propto i^E \tag{4.5}$$

### *4.1.2 Majhen svet*

Pojav majhnega sveta je prvi opisal S. Milgram, ko je preučeval socialna omrežja, kot na primer omrežje poznanstev med množico ljudi, kasneje pa so ta pojav posplošili na področja od biologije, sociologije, psihologije, fizike do računalništva. Največ sta ga preučevala Watts in Strogatz ([21][23][24][26]), ki sta mu tudi izbrala ime po izkušnji, ki jo je doživel že vsak izmed nas, ko v pogovoru z neznancem ugotovi, da imata skupnega znanca, prijatelja ali učiteljico iz osnovne šole.

Za graf pravimo, da ima lastnost majhnega sveta, če je relativno redko povezan (število povezav je reda velikost števila vozlišč), vendar v njem obstajajo tesneje povezane gruče vozlišč, obenem pa sta poljubni dve vozlišči med seboj povezani preko relativno nizkega števila vmesnih vozlišč. V socialnih grafih bi tesneje povezane gruče na primer pomenile skupine ljudi, ki se med seboj poznajo, poznanstev z ljudmi zunaj te gruče pa je manj.

Watts in Strogatz sta opisane lastnosti tudi formalizirala:

1. **Graf je redek**: število povezav je reda velikosti $\mathcal{O}(n)$, pri čemer je $n$ število vseh vozlišč v grafu. Spomnimo se, da je število vseh možnih povezav v grafu enako $n(n - 1) / 2$, torej $\mathcal{O}(n^2)$.

2. **Karakteristična dolžina poti *L*** je povprečna vrednost povprečnih najkrajših poti v grafu od posameznega vozlišča do vseh ostalih vozlišč $d(u)$, prek vseh vozlišč. Povprečna najkrajša pot od vozlišča $u$ do ostalih vozlišč je definirana kot:

$$d(u) = \frac{\sum_{v \in V \setminus \{u\}} d(u,v)}{n-1}; \; u, v \in V \,, \tag{4.6}$$

kjer $d(u,v)$ predstavlja dolžino najkrajše poti med vozliščema $u$ in $v$, n pa je število vseh vozlišč v grafu. Tako dolžino karakteristične poti izračunamo kot

$$L = \frac{\sum_{u \in V} d(u)}{n} \,. \tag{4.7}$$

Če v grafu $G$ velja lastnost majhnega sveta, potem je dolžina karakteristične poti v grafu $G$ primerljiva z dolžino karakteristične poti v naključnem grafu enake velikosti (glede števila vozlišč in povezav).

Karakteristično pot (tudi v tem delu) pogosto imenujemo povprečna najkrajša pot.

- Koeficient gručenja C opisuje, kako močno so med seboj povezana vozlišča znotraj gosteje povezanih gruč. Koeficient izračunamo tako, da za vsako vozlišče izračunamo, koliko njegovih sosedov je povezanih med seboj, to število pa delimo s številom vseh možnih povezav med temi vozlišči (kar pomeni s številom povezav na polnem grafu s takšnim številom vozlišč). Nato izračunamo povprečje teh vrednosti prek vseh vozlišč s stopnjo več kot 1. Vozlišča stopnje 1 izločimo, ker imajo le enega soseda in ne moremo govoriti o tem, kako močno je ta povezan sam s seboj.

  Naj bo $\Gamma(v)$ podgraf, ki vsebuje vsa vozlišča, povezana z vozliščem $v$. Koeficient gručenja $\gamma(v)$ vozlišča $v$ izračunamo kot

$$\gamma(v) = \frac{\left|E(\Gamma(v))\right|}{d_v(d_v - 1)},$$

kjer $\left|E(\Gamma(v))\right|$ predstavlja število obstoječih povezav med vozlišči. Koeficient gručenja $C$ celotnega grafa $G$ pa je izračunamo kot

$$C = \frac{\sum_{v \in V \setminus V^{(1)}} \gamma(v)}{|V| - |V^{(1)}|}, \tag{4.8}$$

kjer $V^{(1)}$ predstavljajo vozlišča stopnje 1.

Če v grafu $G$ velja lastnost majhnega sveta, potem je tu koeficient gručenja $C$ bistveno višji kot koeficient gručenja $C$ v naključnem grafu enake velikosti.

V številnih raziskavah in eksperimentih je bilo ugotovljeno, da imajo lastnost majhnega sveta grafi sodelovanj med pisci, znanstveniki, igralci, športniki, pa tudi grafi telefonskih klicev, hiperpovezav v svetovnem spletu, povezav v živčnem sistemu neke vrste črva, internetne topologije in navideznega omrežja enak z enakim.

Več raziskav ([16],[19]) ugotavlja, da tudi v topologiji samoorganizirajočih se omrežij enak z enakim veljajo potenčni zakoni in lastnost majhnega sveta. Menimo pa, da v zasnove obstoječih sistemov enak z enakim teh lastnosti ne izkoriščajo dovolj.

### 4.1.3 Generiranje topologije

Za generiranje umetnih topologij interneta in njemu podobnih omrežij je nekaj časa veljal za najboljšega postopek BA, ki sta ga leta 1999 predlagala Barabási in Albert [22] (imenuje se po njunih začetnicah): nova vozlišča se linearno raje povezujejo na tista obstoječa vozlišča, ki imajo višjo stopnjo. Verjetnost obstoja povezave med novim vozliščem in obstoječim vozliščem $v$ je v algoritmu BA definirana kot linearna preferenca:

$$\Pi(v) = \frac{d_v}{\sum_{k \in V} d_k}, \tag{4.9}$$

kjer je $d_v$ stopnja vozlišča $v$, $V$ je množica vseh obstoječih (že povezanih) vozlišč, suma pa predstavlja seštevek stopenj vseh obstoječih vozlišč.

Algoritem BA gradi omrežje tako, da začne z majhnim omrežjem ($m_0$ vozlišč je povezanih z $m_0$-$1$ povezavami), ki ga nato dograjuje po korakih do predvidene velikosti. V vsakem koraku algoritem izbira med dvema možnostma:

- z verjetnostjo $p$ doda algoritem $m \leq m_0$ novih povezav; nove povezave se dodajo med obstoječa vozlišča, ki imajo največjo linearno preferenco $\Pi(v_u)$;

- z verjetnostjo *1- p* algoritem doda v graf novo vozlišče; novo vozlišče je z grafom povezano z *m* povezavami; z novim vozliščem se povežejo vozlišča, ki imajo največjo linearno preferenco.

Z raziskovanjem generatorjev so se ukvarjali tudi drugi in predlagali nekatere drugačne generatorje ([53], [54], [55]), leta 2002 pa sta Bu in Towsley [56] predlagala posplošitev postopka BA, ki se je izkazala kot zelo učinkovita, saj generirane topologije najlepše sledijo potenčnim zakonom in imajo dobro izraženo lastnost majhnega sveta.

Postopek sta imenovala GLP (***g**eneralized **l**inear **p**reference*), preferenco pa sta predefinirala:

$$\prod(v) = \frac{d_v - \beta}{\sum_{k \in V}(d_k - \beta)}; \quad \beta \in (-\infty, 1). \tag{4.10}$$

$\beta$ je spremenljiv parameter, s pomočjo katerega določamo, kako močno se nove povezave povezujejo na vozlišča z visoko stopnjo. Parameter $\beta$ tako vpliva na preferenčno povezovanje med popularnimi vozlišči: če je manjši, je več možnosti povezovanja tudi za vozlišča z nizko stopnjo. Ker je $\beta<1$, imajo tudi vozlišča s stopnjo 1 verjetnost povezovanja večjo od nič.

Ker GLP trenutno velja za najboljši generator omrežij s potenčnimi zakoni in lastnostjo majhnega sveta, ga bomo uporabili tudi za generiranje topologij, na katerih bomo simulirali v disertaciji predlagane načine usmerjanja.

## 4.2 Sorodne raziskave

V nadaljevanju pregledno predstavljamo raziskave, ki se ukvarjajo s topologijami in usmerjanjem v vsebinskih omrežjih tipa A in v sorodnih omrežjih.

### *4.2.1 Raziskave usmerjanja v vsebinskih omrežjih tipa A*

Članek Daswanija in soavtorjev [33] predstavlja pregled odprtih problemov v sistemih enak z enakim za izmenjavo vsebin. Avtorji se osredotočajo predvsem na dve področji: porazdeljeno iskanje in varnost. Kot največji izziv za porazdeljeno iskanje navajajo navzkrižje med učinkovitostjo iskanja in avtonomijo udeležencev, sledi pa iskanje ravnotežja med kvaliteto storitve (npr. odzivni čas, število odgovorov) in obremenitvijo, ki jo za sistem predstavlja izvedba storitve.

Močna raziskovalna skupina na področju sistemov enak z enakim je *Stanford Peers* [31], ki dosega vidne rezultate na praktično vseh podpodročjih in tudi objavljajo na vseh pomembnejših konferencah.

Yangova v [36] predlaga tri učinkovite tehnike za iskanje v nestrukturiranih omrežjih enak z enakim. Vse tri temeljijo na zmanjševanju števila vozlišč, ki jih obišče poizvedba. Pri iterativnem poglabljanje se poizvedba pošlje samo sosedom. Če nihče ne vrne odgovora, se ponovno pošlje, tokrat sosedom sosedov, torej do globine 2. Če še vedno ni odgovora, se dalje iterativno povečuje globino do neke maksimalne globine. Bližnji odgovori se na ta način najdejo v primerljivem času, oddaljeni pa v daljšem kot pri osnovnem poplavljanju. Slabost je relativno velika redundanca in odzivni časi, učinkovitost pa je bistveno izboljšana le, če je večina odgovorov relativno blizu v primerjavi z maksimalno globino.

Pri usmerjenem iskanju v širino se poizvedba ne posreduje vsem sosedom, ampak le izbranim. Kriteriji so lahko: veliko število dobrih odgovorov iz prejšnjih poizvedb, veliko število sprejetih sporočil vseh vrst (naj bi izražalo stabilnost vozlišča), kratka vrsta čakajočih sporočil (tj. nezasičenost) in podobno. Tehnika je v povprečju učinkovita, problematična pa je zaradi preobremenjevanja "dobrih" vozlišč in težav z iskanjem bolj nenavadnih vsebin, ki ne ležijo na povprečno dobrih vozliščih.

Pri iskanju z lokalnimi indeksi vsako vozlišče vzdržuje indeks metapodatkov na vozliščih znotraj vnaprej definiranega polmera okrog sebe. Tako se lahko na enem samem vozlišču odgovori na poizvedbo v imenu mnogo vozlišč. Če odgovora ni, se poizvedba posreduje naprej, vendar je nekatera vozlišča ne procesirajo, ampak le posredujejo naprej – tista, katerih metapodatki so že bili preiskani na prejšnjem vozlišču. Slabost tega sistema je, da se morajo vozlišča ob vključevanju v sistem prijaviti vsem vozliščem znotraj polmera, ob vsaki spremembi lokalnih vsebin jim morajo poslati obvestilo o tem, prav tako pa mora vsako vozlišče periodično testirati, če so vsa vozlišča, ki jih indeksira sâmo, še živa.

Adamiceva s soavtorji v [30] raziskuje usmerjanje v omrežjih, v katerih veljajo potenčni zakoni. Zaradi močne povezanosti (velikega števila povezav) posameznih vozlišč v takšni topologiji predlagajo za usmerjanje poizvedbe strategijo naključnega sprehoda, pri čemer poizvedbo usmerjajo na vozlišča čim višje stopnje. Avtorji so pokazali, da je takšna rešitev dovolj učinkovita in ima izredno dobro skalabilnost. Žal pa je očitno, da

bodo vozlišča visoke stopnje pri takem načinu usmerjanja izredno obremenjena. V sistemu enak z enakim, kjer so viri dokaj enakomerno razporejeni po sodelujočih vozliščih, si takšnih ozkih grl pravzaprav ne moremo privoščiti.

Več avtorjev je zato poskušalo strategijo naključnega sprehoda razširiti ali dopolniti, pri čemer so za večje izboljšanje porazdelitve obremenjenosti žrtvovali nekaj učinkovitosti. Nastalo družino protokolov imenujemo epidemični ali tudi klepetavi protokoli, mehanizem usmerjanja pa širjenje čenč. Njihova osnovna ideja je namreč analogna zgodbici o širjenju čenč, ki nastane, če najbolj klepetavi sosedi zaupamo skrivnost: če poizvedbo pošljemo bolje povezanim vozliščem, bo hitreje dosegla večje število vozlišč. Sicer pa to večinoma niso deterministični protokoli, saj sosede, ki jim bomo posredovali poizvedbo, izbiramo bolj ali manj naključno.

Qin Lv s soavtorji v [57] predlaga usmerjanje s širjenjem obroča, ki je zelo podobno kot iterativno poglabljanje, vključno z vsemi prednostmi in slabostmi, ter različico usmerjanja z več vzporednimi naključnimi sprehodi. Tudi tu pride do težav z obremenjenostjo vozlišč visoke stopnje, zato avtorji predlagajo, da bi v nestrukturirane sisteme enak z enakim vgradili mehanizme, ki bi generirali topologijo z značajem bližje naključnemu grafu. Avtorji preučujejo tudi replikacijo vsebin in ugotavljajo, da bi bilo iskanje mnogo učinkovitejše, če bi bila v sistemu dovoljena aktivna replikacija, kar pomeni generiranje večjega števila kopij popularnih vsebin, ki bi jih umestili na poljubna vozlišča. Žal predlagane izboljšave posegajo v avtonomnost vozlišča, zato je vprašljivo, če bi taki sistemi v praksi doživeli primerljivo priljubljenost kot današnji, čeprav manj učinkoviti sistemi.

Crespo [32] pa predlaga uporabo usmerjevalnih indeksov kot namige, katere smeri v omrežju so za iskanje določene vsebine bolj perspektivne: na primer, da bi vozlišče vedelo, kolikšna množica dokumentov s posameznih področij se nahaja na različnih poteh, kamor lahko usmeri poizvedbo. Potem bi usmerilo vsako poizvedbo na tisto pot, ki ima največ dokumentov z iskanega področja. Predlagana strategija se je v simulacijah izkazala zelo dobro.

Portmann in Seneviratne v [51] predlagata različico, kjer naslednjih sosedov ne izbiramo naključno, ampak na osnovi njihovega števila povezav: vsakič, ko vozlišče prejme določeno poizvedbo, jo pošlje odstotku $p$ svojih sosedov, ki te poizvedbe še niso videli,

pri čemer najprej izbere sosede z nižjo stopnjo, torej z manjšim številom povezav. V simulaciji se je predlagana različica obnašala malenkost bolje kot nedeterministično širjenje čenč, vendar pa so bila močno povezana vozlišča manj obremenjena.

Joseph v [50] opisuje sistem Neurogrid: nestrukturiran sistem enak z enakim, ki spremlja odziv uporabnika na najdene odgovore in usmerja naslednje poizvedbe bolj intenzivno na tista vozlišča, ki v preteklosti že dajala koristne odgovore. Na ta način se gradi porazdeljen sistem ugleda med vozlišči, na podlagi katerega se izvaja tudi usmerjanje poizvedb. V Neurogridu vsebine vozlišč ne določa sistem, vozlišča so torej glede tega avtonomna, pač pa po najdenem odgovoru izvorno vozlišče vzpostavi neposredno povezavo z vozliščem odgovora, kar sčasoma pripelje do izredno gosto povezane topologije. To pa je pri resnični uporabi precej nerealno pričakovati in tudi v obstoječih sistemih takega povezovanja ne zasledimo, zato bi sistem potreboval dopolnitve.

**Nihče od omenjenih avtorjev ne poskuša izboljšati usmerjanja na podlagi ugotavljanja, koliko poizvedb se ponovi v dovolj kratkem času, da lahko za odgovor izkoristimo že prej najdeno pot**. Zato izhajamo iz našega dosedanjega dela [39]-[44]. Edina s tem povezana raziskava je eksperiment K. Sripanidkulchai [8], ki raziskuje popularnost poizvedb v sistemu Gnutella. Iz petih eksperimentov, trajajočih od 2 uri do 4 dni, je bilo ugotovljeno, da je porazdelitev popularnosti vsebin dvosegmentna Zipfova porazdelitev, za posamezen eksperiment pa je bil identificiran tudi eksponent v formuli porazdelitve. Dvosegmentna Zipfova porazdelitev pomeni, da so najbolj popularne vsebine med seboj po frekvenci poizvedb bolj izenačene kot manj popularne vsebine. Na podlagi te ugotovitve, ki je zelo odmevna in močno citirana v ostali literaturi, bomo tudi mi v simulacijah kot značilnost tovrstnih sistemov privzeli Zipfovo porazdelitev poizvedb. V isti raziskavi avtorji poskušajo zmanjšati potrebne komunikacijske kapacitete s pomočjo začasnega pomnjenja (*caching*), pri čemer navajajo, da bi pri uporabi okrog 4 MB pomnilnika lahko znižali celoten promet na manj kot tretjino.

### *4.2.2 Ostale sorodne raziskave*

Yangova v [37] medsebojno primerja hibridne sisteme enak z enakim, torej takšne, kjer je del funkcionalnosti centraliziran, medtem ko v [38] predlaga smernice za učinkovitejša omrežja z arhitekturo s super vozlišči. V obeh študijah so poleg matematičnega modela

tovrstnih sistemov za nas predvsem zanimivi eksperimentalno pridobljeni podatki o značilnostih in obnašanju uporabnikov v sistemih za izmenjavo datotek, ki so dovolj splošni, da jih lahko uporabimo tudi v nestrukturiranih sistemih (na primer povprečno število datotek, ki jih nudi posamezen uporabnik, povprečno število ključnih besed v poizvedbi, povprečna dolžina imena datoteke, povprečna frekvenca generiranja poizvedb za uporabnika, povprečna intenzivnost prihodov v sistem...).

Vaucher in soavtorji [59] sledijo obnašanje omrežja Gnutella in predstavljajo eksperimentalno pridobljene podatke iz tega omrežja. Glavna ugotovitev je, da so interakcije med posameznimi vozlišči v glavnem le kratkotrajne, torej da je omrežje dokaj dinamično. Podobne eksperimentalno pridobljene podatke predstavljajo tudi Saroiu in soavtorji [52].

V zadnjem času na področju vsebinskih omrežij in sistemov enak z enakim zasledimo poudarek na zagotavljanju anonimnosti in odpornosti proti cenzuriranju, pa tudi na zagotavljanju boljše skalabilnosti v nestrukturiranih sistemih. Liben-Nowell in soavtorji [14] raziskujejo značilnosti in posledice dinamičnosti omrežja v strukturiranem sistemu Chord, vsebinsko omrežje tipa B, in zanj postavijo model evolucije sistema.

V strukturiranih sistemih enak z enakim in vsebinskih omrežjih ostalih tipov, zlasti tipa B, ki indeksirajo vsebine s pomočjo porazdeljenih zgoščevalnih tabel (*distributed hash table*), kjer so shranjeni digitalni izvlečki vsebin, se raziskave ukvarjajo z učinkovitostjo razpošiljanja delnih indeksov ali tabel, in z doseganjem večje prijaznosti do uporabnika, saj vmesnik `(ključ, vrednost)` ni najbolj prožen. Hevristične tehnike izboljšujejo učinkovitost sistema tako, da usmerjajo iskanje le na del vozlišč, pri tem pa lahko zgrešijo pomembne vsebine.

Tang in soavtorji v [49] predlagajo strukturirano navidezno omrežje, ki naj bi se obnašal dovolj učinkovito tudi pri iskanju po celotni vsebini dokumenta, pri tem pa uporablja napredne algoritme za rangiranje zadetkov. Več avtorjev, npr. Keleher in ostali [17] neodvisno ugotavlja, da zgoščevalne tabele povzročajo izgubo informacij o lokalnosti (medsebojni bližini) vsebin, zato je usmerjanje manj učinkovito, kot bi lahko bilo. Zato predlagajo nad navideznim omrežjem še eno plast, ki naj bi poskrbela za boljšo učinkovitost in večjo prijaznost do uporabnika.

# 5 PREDLAGANO USMERJANJE

V tem poglavju bomo najprej predstavili namen, s katerim smo se lotili zasnove izboljšanih načinov usmerjanja. Opisali bomo, kaj sploh lahko spremenimo in kaj pričakujemo od spremembe načina usmerjanja. Natančneje si bomo ogledali osnovni mehanizem usmerjanja – poplavljanje, za katerega menimo, da bi ga lahko v sistemu s ponavljajočimi se poizvedbami delno spremenili in s tem dosegli značilno zmanjšanje skupnega prometa.

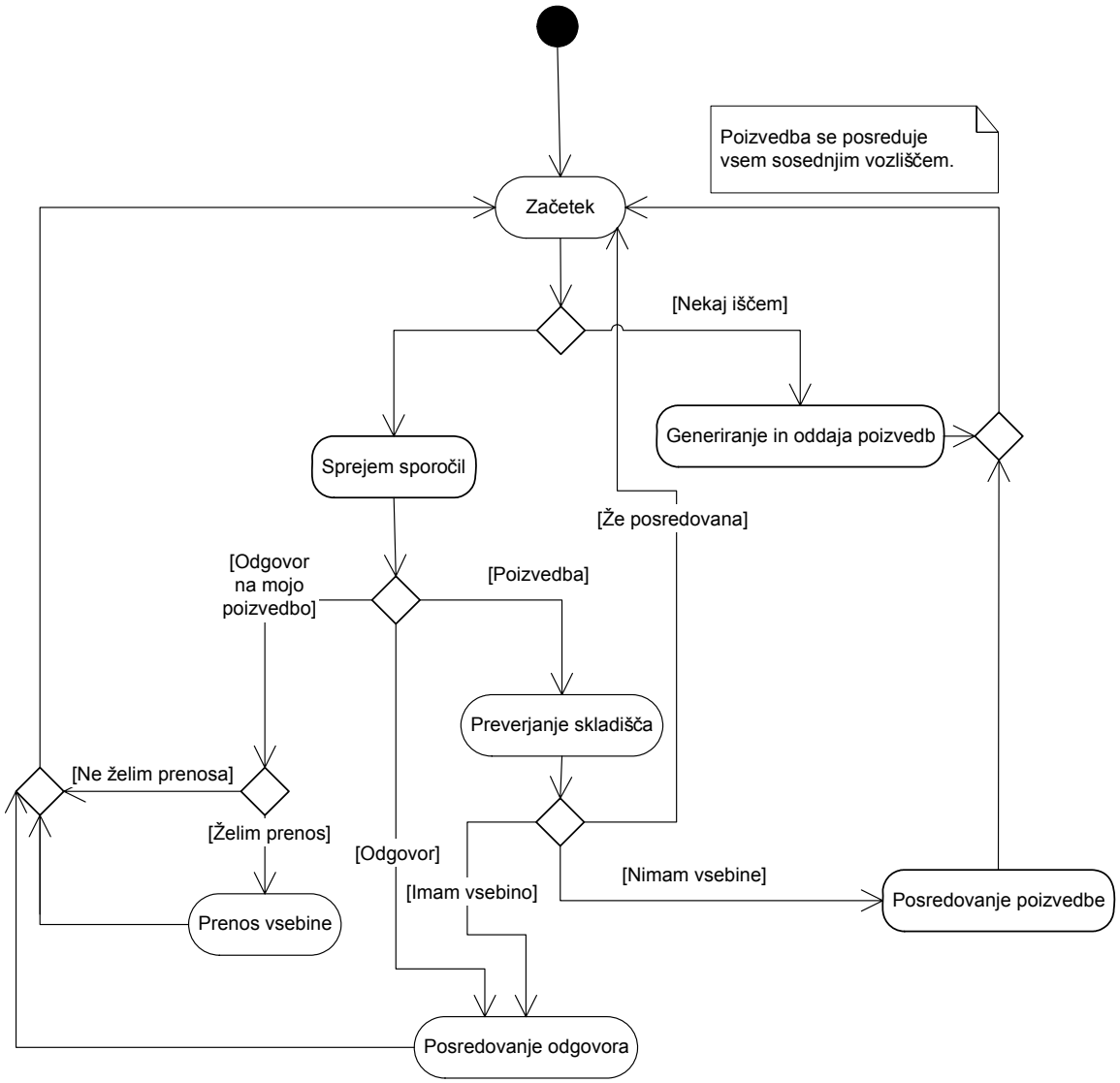
Po opisu poplavljanja navajamo dve predlagani izboljšavi in jih tudi podrobno komentiramo. Navedli bomo razloge, zaradi katerih pri uporabi predlaganih izboljšav pričakujemo zmanjšanje količine prenosov poizvedb. Za predlagani izboljšavi poleg osnovnih principov navajamo pričakovano koristnost in oceno potrebnih dodatnih virov za njihovo uresničitev.

## 5.1 Osnovno poplavljanje

Osnovno poplavljanje je robusten in zanesljiv način usmerjanja sporočil z znanimi slabostmi glede redundantnosti in požrešnosti po virih. V nadaljevanju tega razdelka bomo poplavljanje opisali podrobneje.

Vozlišče, ki prejme poizvedbo, v svojem lokalnem skladišču vsebin poskuša najti vsebino, ki bi ustrezala prejeti poizvedbi. Če jo najde, generira odgovor in ga vrne tja, od koder je prejel poizvedbo. Odgovor tipično vsebuje napotke, s pomočjo katerih poizvedujoče vozlišče lahko vzpostavi neposreden stik z vozliščem odgovora, ter metapodatke o najdeni vsebini. Če pa vozlišče ne najde odgovora, posreduje poizvedbo naprej vsem svojim sosedom razen tistemu, od katerega jo je prejelo. Možna je celo izvedba, pri kateri vozlišče v vsakem primeru (tudi ko je bil že generiran odgovor) posreduje poizvedbo svojim sosedom.

Vozlišče si ob vsakem posredovanju poizvedbe shrani identifikator poizvedbe. Če isto poizvedbo kasneje ponovno sprejme, je ne posreduje še enkrat. Pred posredovanjem tudi preveri, če še ni presegla svoje predpisane življenjske dobe.




*Slika 5-1: Koncept poplavljanja v obliki diagrama aktivnosti.*

Slika 5-1 prikazuje koncept poplavljanja v obliki visokonivojskega diagrama aktivnosti v standardni notaciji UML. Vozlišče generira in razpošlje novo poizvedbo ali pa sprejme sporočilo. Če sprejme poizvedbo, na podlagi edinstvenega identifikatorja poizvedbe (GUID) vozlišče ugotovi, če je poizvedbo sprejelo že kdaj prej in jo torej tudi že posredovalo svojim sosedom. V tem primeru je ne posreduje več nikamor, ampak se le vrne na začetek. Če gre za novo poizvedbo, vozlišče pogleda v svoje skladišče vsebin in

poskuša najti kako vsebino, ki ustreza poizvedbi. Če jo najde, generira odgovor in ga posreduje sosedu, od katerega je sprejelo poizvedbo, v nasprotnem primeru pa poizvedbo posreduje vsem svojim ostalim sosedom.

Če pa je bilo sprejeto sporočilo odgovor na poizvedbo, generirano s strani tega vozlišča, jo vozlišče ustrezno interpretira in lahko tudi sproži prenos vsebine. Če gre le za posredovanje odgovora, pa ga vozišče odpošlje ustreznemu sosedu.

Dobra lastnost poplavljanja je visoka robustnost in odpornost na napake, saj bo poizvedba vedno dosegla najbližje vozlišče z iskano vsebino po najkrajši poti. Res pa ga bo dosegla tudi po drugih poteh in dosegla bo tudi zelo veliko ostalih vozlišč. Oboje ima svoj vzrok v veliki redundantnosti prenosov. Žal pa ravno to pomeni tudi zelo veliko obremenitev za komunikacijske kapacitete, kar želimo z v nadaljevanju predlaganima načinoma usmerjanja izboljšati.

## 5.2 Izboljšava poplavljanja s pomnjenjem posredovanih odgovorov

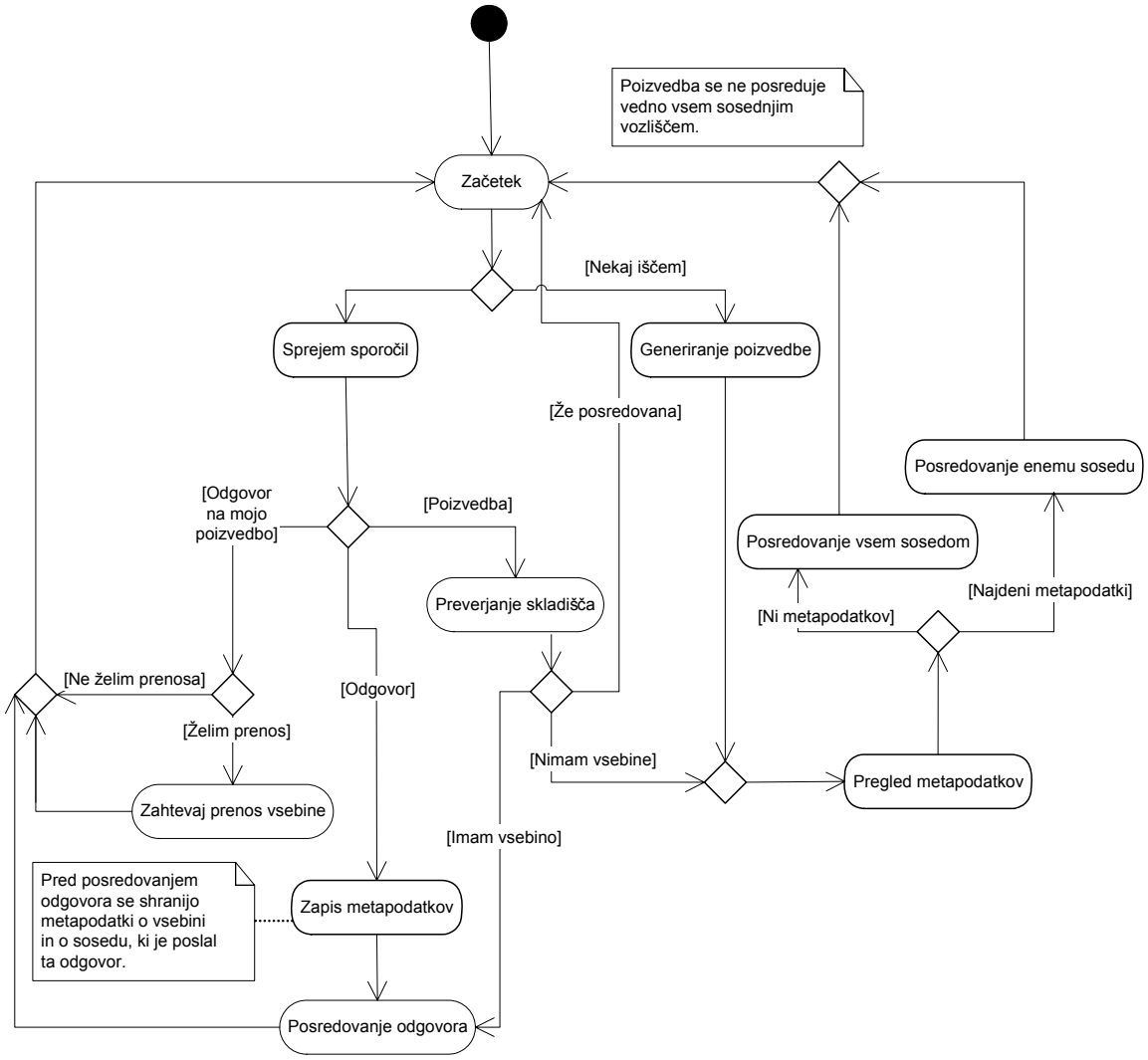
V prejšnjem razdelku smo opisali osnovno poplavljanje – usmerjevalni mehanizem, ki ga želimo spremeniti tako, da bo skupno število prenosov poizvedb v sistemu manjše, pri tem pa bomo ohranili kakovost iskanja. Drugače povedano, od izboljšanega usmerjanja pričakujemo, da zna najti najbližje vozlišče z iskano vsebino s povprečno manjšim številom redundantnih prenosov.

Pri osnovnem poplavljanju vozlišča vsako poizvedbo poplavijo, saj nimajo spomina za prej posredovane odgovore. Naš prvi predlog usmerjanja, izboljšano poplavljanje **s pomnjenjem posredovanih odgovorov,** pa izkorišča relativno visoko ponovljivost poizvedb. Vemo namreč, da so nekatere vsebine zelo zaželene in se poizvedbe po njih pogosto pojavljajo. Če z ustreznim mehanizmom dosežemo, da se vsaj te poizvedbe ne poplavljajo, ampak usmerijo le na vozlišče, od koder smo nazadnje dobili odgovor, sklepamo, da se bo obremenitev sistema že opazno zmanjšala.

Opišimo sedaj podrobneje predlagani mehanizem. Vozlišče skuša poizvedbe, za katere je že kdaj prej posredovalo odgovor, usmeriti na pot, po kateri je prišel ta odgovor. Tam bo poizvedba z veliko verjetnostjo spet našla odgovor. Odgovora ne bo našla le v primeru,

da je med tem časom ciljno vozlišče zapustilo omrežje, da je prekinjena kaka vmesna povezava na poti do cilja, ali da je ciljno vozlišče umaknilo želeni podatek oziroma vsebino. V tem primeru se po določenem času poizvedba vendarle poplavi.

Slika 5-2 prikazuje diagram aktivnosti za koncept usmerjanja s pomnjenjem posredovanih odgovorov. Če ga primerjamo z diagramom, ki prikazuje koncept poplavljanja, najprej opazimo novo aktivnost, ki nastopi pred posredovanjem odgovora.

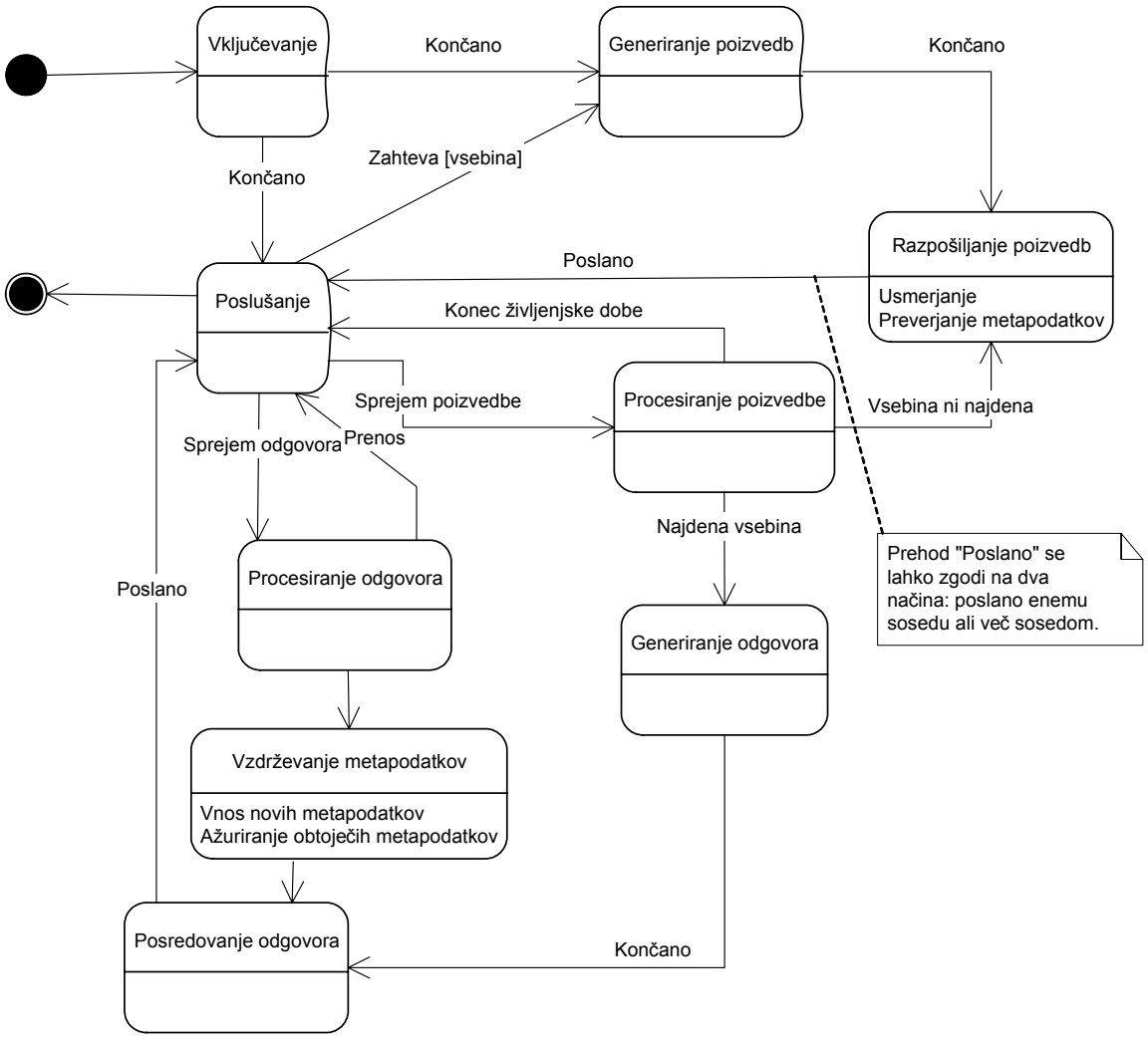



*Slika 5-2: Koncept izboljšanega poplavljanja s pomnjenjem posredovanih odgovorov.*

To je zapis metapodatkov: vsakič, ko vozlišče prejme odgovor na neko prej posredovano poizvedbo, shrani v svoje skladišče metapodatkov identifikacijo soseda, ki je posredoval ta odgovor in metapodatke o vsebini, ki jih nosi odgovor. S pomočjo teh metapodatkov

lahko kasneje vozlišče ustrezno usmerja poizvedbe, katerih iskalni pogoji ustrezajo shranjenim metapodatkom.

V naši implementaciji se med metapodatke vedno zapiše le eno usmerjanje za vsako vsebino. Shranjujemo vedno metapodatke o tistem odgovoru, ki je od ciljnega vozlišča prepotoval najmanjše število skokov – tako kasneje bomo usmerjali poizvedbe proti najbližjem vozlišču z želeno vsebino. Če dobimo več enakovrednih odgovorov, vedno shranimo najbolj svežega. Odgovore, ki so od ciljnega vozlišča potrebovali več skokov, le posredujemo dalje in jih v skladišče metapodatkov ne shranjujemo.




*Slika 5-3: Diagram stanj vozlišča za izboljšano usmerjanje s pomnjenjem posredovanih odgovorov.*

Tako poskušamo vzdrževati metapodatke čimbolj ažurne, saj vemo, da je navidezno omrežje dinamično. Če bi želeli na posamezno poizvedbo dobiti več kot en odgovor, bi bilo smiselno v skladišče metapodatkov shranjevati po več vnosov za posamezno vsebino.

Druga razlika med poplavljanjem in predlagano izboljšavo s pomnjenjem posredovanih odgovorov nastopi pri posredovanju poizvedb. Preden vozlišče odpošlje poizvedbo, pregleda svoje skladišče metapodatkov in med njimi poskuša najti metapodatke o taki vsebini, ki bi se ujemala s pogoji poizvedbe. Če jih najde, usmeri poizvedbo le na vozlišče, ki je posredovalo ta odgovor, sicer pa se poizvedba poplavi na vsa sosednja vozlišča (razen na tisto, ki je to poizvedbo posredovalo).

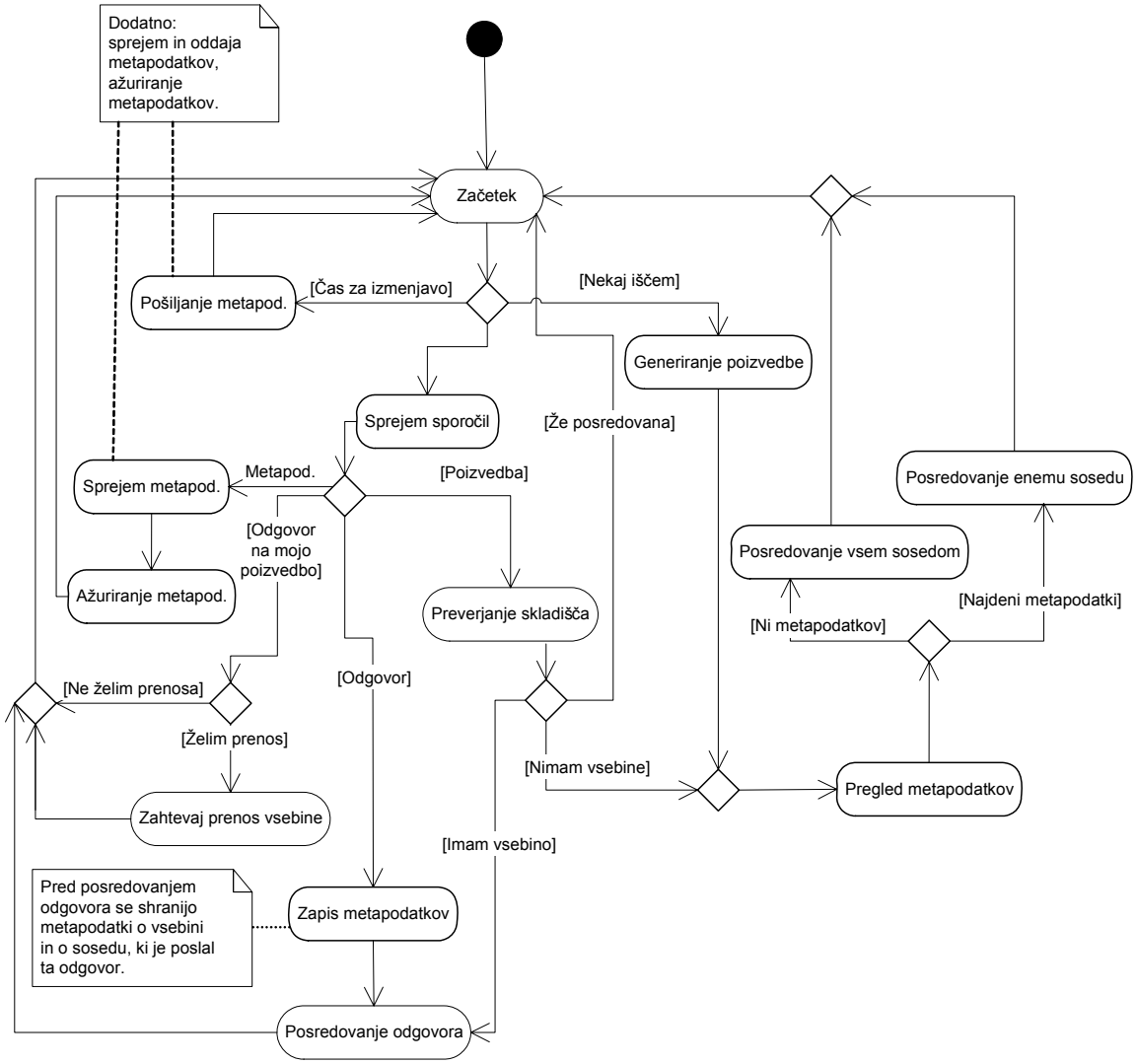
Za jasnejšo sliko si oglejmo še diagram stanj v notaciji UML (Slika 5-3). Diagram stanj prikazuje stanja, ki jih lahko zavzame vozlišče, ter prehode med njimi (kjer so možni). Tu ni poudarjena sekvenčnost, ampak bolj načini odzivanja vozlišča na zunanje dogodke ali sporočila. Stanje razpošiljanja poizvedb je na primer prisotno tu in pri poplavljanju, vendar je tu njegova vsebina bogatejša, prehod v stanje poslušanja pa je možen na dva načina: s pošiljanjem poizvedbe enemu sosedu ali pa s poplavo poizvedbe.

## 5.3 Izboljšava poplavljanja z izmenjavo metapodatkov

Izboljšano **poplavljanje z izmenjavo usmerjevalnih metapodatkov** nadgrajuje različico iz prejšnjega razdelka (usmerjanje s pomnjenjem posredovanih odgovorov) in se po svoji osnovni funkcionalnosti že približuje porazdeljenemu usmerjanju, kot ga poznamo iz omrežne plasti. Usmerjanje poizvedb poteka na enak način kot prej (Slika 5-2), torej bodisi poplavljanje, če se iskana vsebina ne ujame z do sedaj zbranimi metapodatki, bodisi posredovanje poizvedbe sosedu, ki je nazadnje posredoval ustrezen odgovor.

Dopolnjen pa je mehanizem izgradnje lokalne zbirke metapodatkov. Slika 5-4 prikazuje diagram aktivnosti, na katerem lahko izluščimo osnovno idejo in razliko od usmerjanja s pomnjenjem odgovorov. Prej je vozlišče zgolj shranjevalo metapodatke o posredovanih odgovorih, sedaj pa te metapodatke periodično pošilja vsem svojim sosedom in ravno tako periodično tudi sprejema metapodatke od svojih sosedov. V svojo zbirko

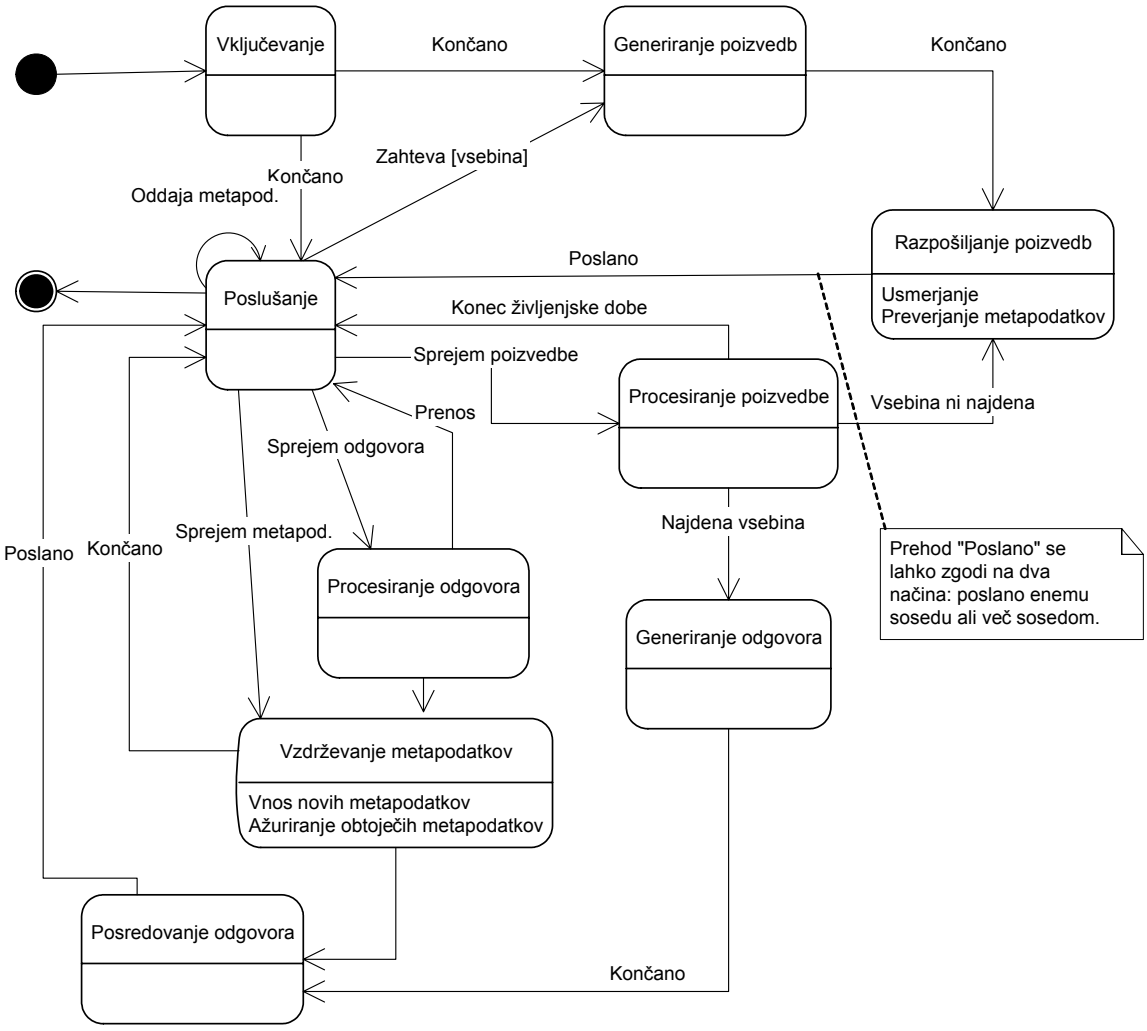
metapodatkov vstavi ustrezne nove podatke in tako lahko pravilno usmerja tudi tiste poizvedbe, za katere samo vozlišče ni videlo odgovora, pač pa so ga videla sosednja vozlišča ali celo sosedje sosedov. Tako vozlišča na porazdeljen način gradijo najkrajše poti za svojo okolico z radijem, enakim življenjski dobi poizvedbe.




*Slika 5-4: Diagram aktivnosti za usmerjanje z izmenjavo metapodatkov.*

Vozlišče mora namreč pri ažuriranju usmerjevalnih metapodatkov upoštevati tudi življenjsko dobo poizvedbe: shranjuje le usmerjanje do tistih vsebin, ki so znotraj radija, ki ga lahko doseže poizvedba. Tudi pri usmerjanju poizvedb mora upoštevati, koliko skokov je poizvedba že naredila, ter jo usmerjati le do vsebin, ki so v dosegu njene preostale življenjske dobe.

Prednost tega načina pred prejšnjim je predvsem vnaprejšnje konfiguriranje omrežja in posledično tudi pravilno usmerjanje do vsebin, po katerih do tedaj še ni bilo poizvedb. Slabost pa je večja količina prenosov, potrebnih za izmenjavo metapodatkov, in prav tako večja količina lokalnega pomnilnega prostora, potrebnega za hranjenje usmerjevalnih metapodatkov. Zaradi visoke dinamičnosti omrežja se lahko zgodi, da večjega dela usmerjevalnih metapodatkov sploh ne bomo uporabili: po manj zaželenih vsebinah ne bo še nobene poizvedbe, njihova vozlišča pa bodo že zapuščala omrežje.




*Slika 5-5: Diagram stanj za usmerjanje z izmenjavo metapodatkov.*

Oglejmo si še diagram stanj za naš drugi predlog usmerjanja. Slika 5-5 kaže diagram, zelo podoben diagramu usmerjanja s pomnjenjem posredovanih odgovorov. Najvidnejša razlika se pojavi pri prehodu iz stanja poslušanje neposredno v stanje vzdrževanja metapodatkov. Prehod nastopi ob prejemu usmerjevalnih metapodatkov od kakega izmed

sosednjih vozlišč. Ko je ažuriranje končano, vozlišče spet neposredno preide v stanje poslušanja.

## 5.4 Dinamičnost navideznega omrežja v predlaganih usmerjanjih

Inherentna lastnost sistemov enak z enakim je njihova dinamičnost: v sistem se stalno vključujejo nova vozlišča in iz njega izstopajo stara. Zaradi tega so vsak trenutek lahko na voljo novi viri, obenem pa ni več na voljo virov, do katerih smo pred nekaj trenutki še lahko dostopali.

### *5.4.1 Prihajanje in odhajanje vozlišč*

Novo vključena vozlišča od usmerjevalnega mehanizma ne zahtevajo nobene dodatne funkcionalnosti: usmerjane poizvedbe z drugih vozlišč se bodo v začetku novemu vozlišču izogibale, ob naslednjih poplavah pa bodo poizvedbe dosegle tudi novo vozlišče. To bo posredovalo prejete odgovore in se tako postopoma ustrezno konfiguriralo, obenem pa bodo usmerjanje prek njega dodajala tudi ostala vozlišča. Pri načinu usmerjanja z izmenjavo usmerjevalnih metapodatkov pa se bodo poizvedbe začele usmerjati prek njega že prej, takoj po prvi izmenjavi.

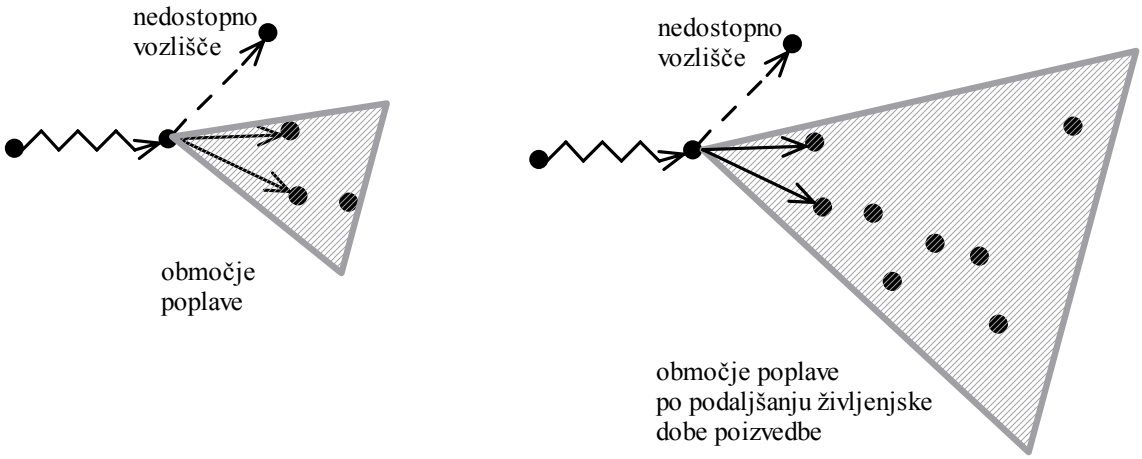
Težava z usmerjanjem pa nastopi ob izstopu vozlišč iz sistema, saj nekatere usmerjene poizvedbe ne morejo priti do cilja. Kljub temu se lahko še vedno zgodi, da vozlišče na podlagi metapodatkov usmeri poizvedbo na soseda, ki v sistemu ne obstaja več. Taka poizvedba gotovo ne bo našla odgovora. Edini mehanizem, ki omogoči, da taka poizvedba lahko vseeno najde odgovor, je njeno naknadno poplavljanje. Uporabnik bo sicer malo dlje čakal na odgovor, vendarle pa ga bo dobil. Vendar pa se je najprej potrebno odločiti, kdo bo sprožil zahtevo za poplavljanje poizvedbe. Različne možnosti predstavljamo v naslednjih razdelkih.

#### *5.4.1.1 Vozlišče, ki je ugotovilo, da sosed ni več dostopen*

Vozlišče, ki je ugotovilo, da sosed ni več dostopen, sproži zahtevo za poplavljanje. V tem primeru bo poplava poizvedbe v splošnem sprožena šele nekaj skokov od izvornega

vozlišča in lahko se zgodi, da bo njena življenjska doba (število skokov) že pri koncu, tako da tudi poplava ne bo zajela dovolj vozlišč za iskanje odgovora. Veliko takih poizvedb bo ostalo neodgovorjenih.

Ob tej priložnosti bi bilo možno seveda podaljšati preostalo življenjsko dobo poizvedbe in tako omogočiti poplavo popolne velikosti. Pomislek proti tej izvedbi pa je, da bi takšna "oddaljena" poplava ignorirala morebitne odgovore, ki so blizu izvora poizvedbe, in našla takšne, ki se nahajajo več skokov stran. Slika 5-6 predstavlja opisani različici poplave. Očitno je ta rešitev sicer izvedljiva, vendar pa je precej okorna in zato lahko poskusimo najti učinkovitejšo.




*Slika 5-6: Primerjava območja poplave pri nespremenjeni življenjski dobi poizvedbe in pri podaljšani življenjski dobi poizvedbe; oboje v primeru, ko eno od vozlišč na poti do rezultata postane nedostopno.*

#### *5.4.1.2 Izvorno vozlišče*

Poplavo lahko sproži **izvorno vozlišče**, če ne dobi odgovora v pričakovanem času. Težava pa je v tem, da izvorno vozlišče poizvedbo vdrugo sicer lahko pošlje vsem svojim sosedom, vendar je možno, da so tudi sosedje že prej posredovali odgovor nanjo in jo bodo zato posredovali naprej samo po poti, za katero smo ravnokar ugotovili, da ni uporabna. Zato bi bilo potrebno uvesti novo protokolarno sporočilo: poizvedba z obveznim poplavljanjem, ki bi jo poplavila vsa vozlišča, tudi tista, ki so že "skonfigurirana" za iskano vsebino. To bi med drugim prineslo težave glede zagotavljanja združljivosti z vozlišči, ki delujejo po osnovnem protokolu in ne poznajo

novega tipa sporočila. Potrebno bi bilo tudi zagotoviti preprečevanje možnosti zlorabe, saj bi lahko dovolj spretni in preveč neučakani uporabniki (ali razvijalci aplikacij) za vse poizvedbe že takoj zahtevali poplavljanje, kar bi seveda pomenilo korak nazaj glede agregirane obremenitve navideznega omrežja.

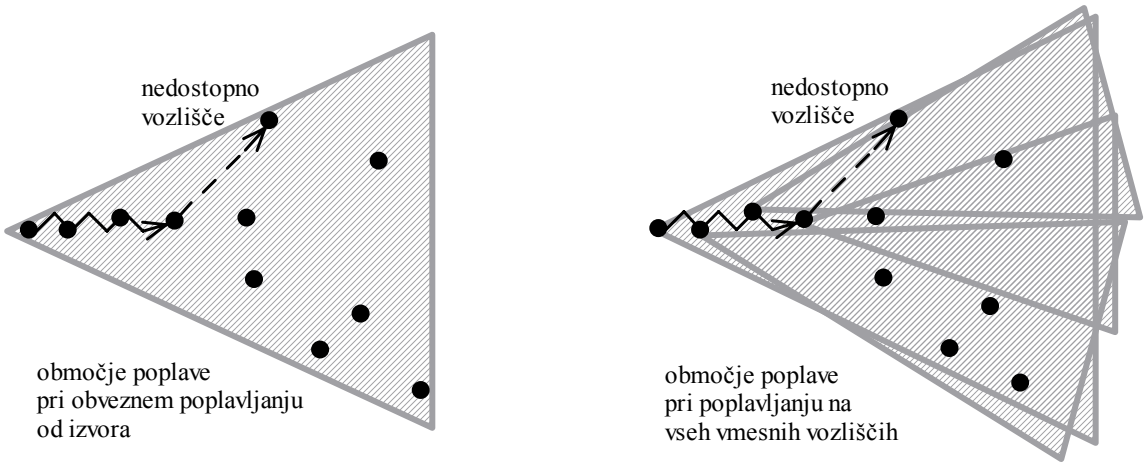
#### *5.4.1.3 Uporabnik*

Poplavo bi seveda lahko sprožil tudi uporabnik, ki dovolj dolgo ne bi dobil odgovora. Seveda je tu možnost zlorab še večja, poleg tega pa nalogo, ki bi jo lahko opravil sistem sam, a tem prelagamo na uporabnika. Uporabnika načeloma ne zanimajo detajli iskanja, kot na primer ali je bil odgovor najden s pomočjo poplave ali ne, zato to možnost lahko kar takoj izločimo.

#### *5.4.1.4 Vsa vmesna vozlišča*

Poplavo lahko sprožijo **vsa vozlišča od izvora** do vozlišča, ki je ugotovilo, da naslednji sosed ni več dostopen. Vozlišča so nekoč pred tem že videla odgovor na trenutno poizvedbo in si ga tudi zapomnila. Zapomnila so si tudi čas, ki je minil od trenutka takratnega posredovanja poizvedbe do sprejema odgovora. Zdaj lahko domnevajo, da mora odgovor prispeti po preteku podobnega časovnega intervala (upošteva naj se še nek smiseln, okoliščinam ustrezen varnostni faktor). Če po preteku tega časa odgovora ni, vsako vmesno vozlišče na poti od izvora poizvedbe sproži poplavo, če le poizvedbe ni poplavilo že prvič. Vozlišča, ki so poizvedbo poplavila že prej, odgovora še niso videla in zato ne morejo oceniti tega časovnega intervala, poleg tega pa s ponovno poplavo k rešitvi ne bi prinesla nič novega.

Dobra lastnost zadnjega načina je, da je časovno in glede števila prenosov poizvedbe popolnoma primerljiv s poplavo, ki jo sproži izvorno vozlišče, vendar ne zahteva uvedbe novega protokolarnega sporočila. Res pa zahteva več administracije in več virov na vsakem sodelujočem vozlišču. Največja prednost tega načina pa je bistveno boljši odzivni čas kot pri poplavi, ki jo zahteva izvorno vozlišče. Prepričajmo se.




*Slika 5-7: Primerjava poplave s strani izvornega vozlišča in poplave s strani vseh vmesnih vozlišč; oboje po odpovedi enega od vozlišč na poti do odgovora. Območji se prostorsko prekrivata (zajameta ista vozlišča), lahko pa pride do časovnih razlik.*

Denimo, da izvorno vozlišče domneva, da je vozlišče odgovora oddaljeno največ $t$ skokov in zato mora odgovor dobiti v času $2t$ skokov. Če odgovora v tem času ne dobi, sproži poplavo, med katero se najde alternativen odgovor, oddaljen $s$ skokov. Skupen čas, ki poteče od prvotne poizvedbe do sprejetja odgovora, je torej $2t + 2s$:

$$T_{izvor} = 2t + 2s \tag{5.1}$$

Če pa imajo tak mehanizem časovne kontrole vgrajena tudi vsa vmesna vozlišča, velja za vozlišče, ki je od izvornega oddaljeno en skok, naslednji razmislek: vozlišče pričakuje odgovor v času $(t - 1)$ skokov. Če ga v tem času ne dobi, sproži poplavo, ki po $(s – 1)$ skokih doseže vozlišče odgovora, odgovor pa se vrne po $2(t – 1) + 2(s – 1)$ skokih, nakar je potreben še en skok, da odgovor posredujemo do izvornega vozlišča. Upoštevati moramo tudi prvi skok, ki ga je opravila poizvedba, da je prišla do tega vozlišča, in tako je skupno število skokov, potrebnih, da odgovor doseže izvorno vozlišče, naslednje:

$$T_{en\ skok} = 1 + 2(t-1) + 2(s-1) + 1 = 2t + 2s - 2 \tag{5.2}$$

Če je vmesno vozlišče od izvornega odmaknjeno dva skoka, po enakem razmisleku ugotovimo, da se na izvorno vozlišče vrne odgovor po $2t + 2s – 4$ skokih, če pa je vmesno vozlišče odmaknjeno od izvornega $n$ skokov, odgovor doseže izvorno vozlišče po $2t + 2s – 2n$ skokih.

V splošnem primeru poplavo sprožijo vsa vmesna vozlišča, ki je niso sprožila že ob sprejemu poizvedbe. Izvorno vozlišče najprej doseže najbližji odgovor, torej velja

$$T_{vsa\ vozl.} = n + 2(t-n) + 2(s-n) + n = 2t + 2s - 2n \ . \qquad (5.3)$$

Zaradi naštetih razlogov smo se pri izvedbi simulacije odločili za poplavo s strani vseh vmesnih vozlišč.

Opozorimo naj še, da pri poplavi s strani vseh vmesnih vozlišč ni število prenosov sporočil nič večje kot pri obvezni poplavi, saj še vedno velja, da vozlišča pomnijo GUID – identifikator poizvedbe, in ko sprejmejo isto poizvedbo drugič, je ne posredujejo naprej.

### *5.4.2 Vzdrževanje usmerjevalnih metapodatkov*

Yang v študiji [36] empirično ugotavlja, da povprečen uporabnik v času svoje življenjske dobe oziroma ene seanse v sistemu enak z enakim generira približno povprečno deset poizvedb. Tudi če imamo v sistemu mnogo uporabnikov, lahko predpostavimo, da na vsakih deset generiranih poizvedb (ne glede na to, kdo jih je generiral), eden od njih zapusti sistem.

Vozlišča si pri obeh predlaganih načinih usmerjanja lokalno shranjujejo metapodatke, ki jih potrebujejo za učinkovitejše usmerjanje. Zaradi dinamične narave sistema moramo prav tako kot zgoraj tudi tu upoštevati, da povprečna življenjska doba vozlišča v sistemu ni neomejena. Zato smo implementirali tudi akcijo praznjenja skladišča metapodatkov: vsa vozlišča periodično brišejo lokalne metapodatke, ki so starejši kot je neka vnaprej določena omejitev, odvisna od povprečne življenjske dobe vozlišča.

## 5.5 Pričakovanja

Če izhajamo iz osnovnega poplavljanja, lahko količino prenosov poizvedb zmanjšamo tako, kot predlaga večina avtorjev, navajanih v razdelku 4.2: zmanjšamo število poti, po katerih pride poizvedba do posameznega vozlišča. S tem odpravimo čisto redundantne prenose, poizvedba pa še vedno doseže enako (ali vsaj skorajda enako) število vozlišč,

kot pri poplavljanju. S tem je tudi verjetnost, da bo poizvedba dosegla vozlišče z iskano vsebino, praktično enaka.

### 5.5.1 Izhodišče

Naše predlagane izboljšave pa izhajajo iz razmišljanja, da poizvedba na večini vozlišč ne najde iskane vsebine, zato bi bilo najbolje, da bi dosegla le tista vozlišča, na katerih se iskana vsebina nahaja. Seveda to lahko v nestrukturiranih sistemih ugotovimo le tako, da povprašamo vsako vozlišče, če morebiti ima iskano vsebino – torej poplavimo poizvedbo. Od tu naprej pa še upoštevamo ugotovitev, da pogosto različne poizvedbe iščejo iste vsebine. Če bi torej vedeli, kje je našla iskano vsebino prejšnja poizvedba, bi lahko usmerili naslednjo le na tista vozlišča. Dokler se topologija omrežja ne spremeni toliko, da ciljno vozlišče odide ali ni več dostopno po isti poti, bi lahko vse naslednje poizvedbe ustrezno usmerili in tako zmanjšali potrebno število prenosov poizvedbe na vrednost, ki je v velikostnem razredu življenjske dobe (števila skokov) poizvedbe.

Dobre lastnosti, ki jih pričakujemo od takšnega načina usmerjanja, so naslednje:

- zmanjšanje skupnega števila prenosov poizvedb,
- brez poslabšanja performans za končnega uporabnika (možno je celo izboljšanje zaradi manj skupnega prometa in s tem manjše obremenjenosti povezav),
- združljivost z obstoječimi načini usmerjanja (možna je vzporedna uporaba starega in novega načina – torej nekaj vozlišč z novim in nekaj s starim načinom, seveda potem skupno zmanjšanje števila prenosov ni tako veliko).

Pričakujemo pa, da bi z uvedbo predlaganega načina usmerjanja lahko naleteli tudi na nekatere težave:

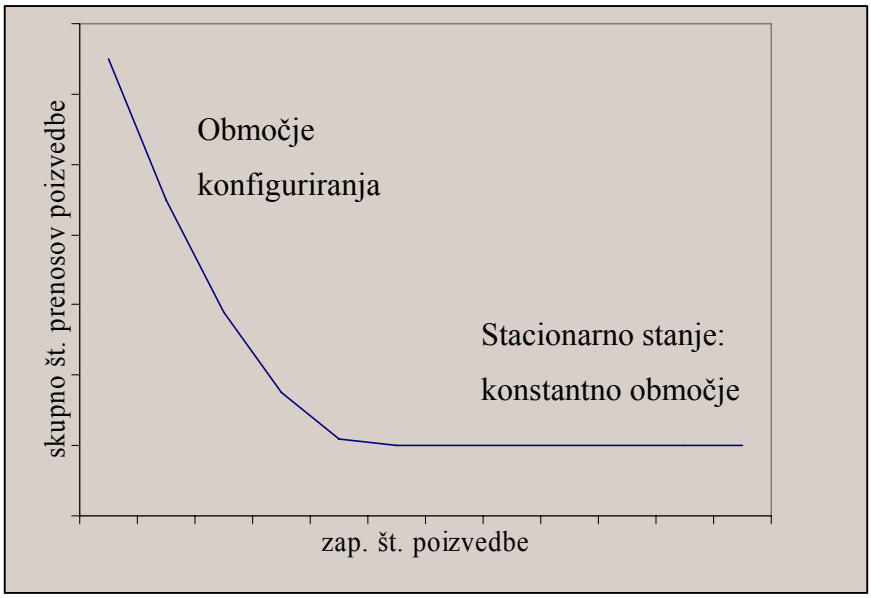
- Za učinkovito izvajanje usmerjanja vozlišča potrebujejo ustrezno količino pomnilnega prostora, kar za povprečne osebne računalnike sicer ne bi predstavljalo težav, vendar pa bi bilo problematično, če bi se usmerjanje uporabljalo za manj zmogljive, prenosne naprave, ki imajo precej bolj omejene vire.

- Pri neustreznih robnih pogojih (preveč dinamično omrežje glede na ponovljivost poizvedb) bi usmerjanje prineslo več škode kot koristi, saj bi se po nepotrebnem porabljali lokalni pomnilni in procesorski viri, konec koncev pa bi se poizvedbe večinoma poplavljale.

### 5.5.2 Konfiguriranje omrežja

Postopek konfiguriranja omrežja je proces, v katerem se vozlišča "učijo", oziroma si s pomočjo prenesenih poizvedb in odgovorov gradijo svoja skladišča usmerjevalnih metapodatkov. Konfiguriranje se pri uporabi predlaganih načinov usmerjanja dogaja ves čas delovanja sistema enak z enakim, ko v sistem vstopajo nova vozlišča. Da bomo bolje razumeli postopek konfiguriranja, si bomo na tem mestu podrobneje ogledali konfiguriranje ob "hladnem" zagonu sistema, ko nobeno vozlišče še nima nobenih usmerjevalnih metapodatkov. Enako velja vstopu nove vsebine v sistem, saj je do tedaj nobeno vozlišče ni poznalo in ne zna usmerjati poizvedb do nje.

Če bi vse poizvedbe iskale isto vsebino in se topologija navideznega omrežja ne bi spreminjala, bi se z vsako naslednjo poizvedbo povprečno število skokov zmanjševalo, dokler ne bi doseglo stacionarnega stanja - neke konstantne povprečne vrednosti.

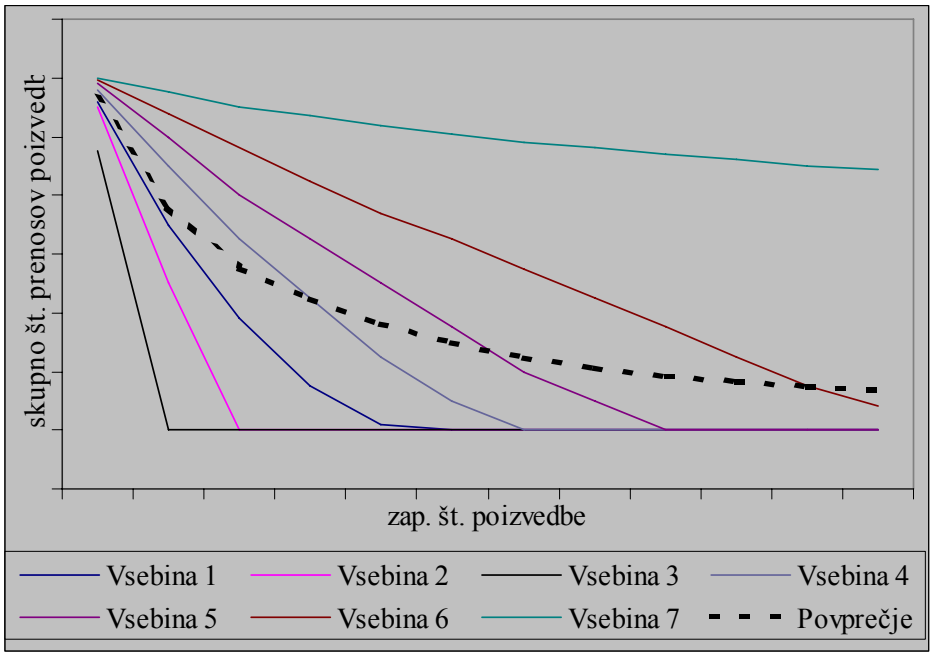



*Graf 5.1: Konfiguriranje omrežja: število prenosov poizvedbe v odvisnosti ob zaporednega števila poizvedbe.*

V primeru dinamične topologije bi bila osnovna oblika grafa enaka, le konstantna vrednost bi bila višja, saj bi bile v povprečje zajete tudi občasne poplave ali delne poplave (t. j. poplave le s strani nekaterih vozlišč na poti) zaradi pretrganih poti.

Če si to grafično predstavimo, bi funkcija števila prenosov poizvedbe v odvisnosti od zaporedne številke poizvedbe izgledala podobno, kot prikazuje Graf 5.1.

V splošnem sistemu z mnogo vsebinami se vozlišča konfigurirajo za nekatere vsebine hitreje, za druge pa počasneje - glede na to, kako pogosto se pojavljajo poizvedbe za posamezno vsebino. Število prenosov poizvedb za takšen sistem zato predstavlja povprečje več različno raztegnjenih krivulj, podobno kot to kaže Graf 5.2. Z barvnimi črtami so prikazane krivulje konfiguriranja za različno popularne vsebine, debelejše črtkano pa je nakazano povprečje njihovih vrednosti.




*Graf 5.2:Konfiguriranje omrežja pri mešanici poizvedb, ki iščejo različno popularne vsebine.*

Vidimo lahko, da bo povprečje nižje, če bo več popularnih vsebin, oziroma višje, če bo več nepopularnih vsebin. Ker ne poznamo matematičnega modela za topologije, v katerih veljajo potenčni zakoni in lastnost majhnega sveta, ne znamo analitično izračunati, kje nastopi t.i. konstantno območje, kako dolgo zaporedje poizvedb je potrebno, preden sistem doseže stacionarno stanje in povprečno koliko prenosov na poizvedbo lahko

pričakujemo v konstantnem območju. Pričakujemo, da bomo te vrednosti za izbrano kombinacijo robnih pogojev lažje ocenjevali, ko bomo imeli pred seboj rezultate simulacij.

V danih razmerah lahko izračunamo oceno za povprečje le, če privzamemo, da poznamo funkcijo *f'(x)*, ki predstavlja odvisnost števila prenosov poizvedbe od zaporednega števila ponovitev poizvedbe. Vemo, da je to padajoča diskretna funkcija, ki se ustali pri neki pozitivni vrednosti, večji od 1, podobno kot to predstavlja Graf 5.1.

$$f'(x) = \begin{cases} f(x), \text{če } x \in Z^+ \\ \text{ni definirano, sicer.} \end{cases} \tag{5.4}$$

Ko omrežje doživi mešanico poizvedb, ki povprašujejo po vsaki vsebini *i* skladno z njeno popularnostjo ($q_i$), skupno število prenosov poizvedb izračunamo kot povprečje funkcij $f_i(x)$. Funkcija $f_i(x)$ predstavlja odvisnost števila prenosov poizvedbe po vsebini *i* od zaporednega števila poizvedbe (v zaporedju vseh poizvedb) in je v primerjavi z $f(x)$ raztegnjena v smeri osi *x*: za popularnejše vsebine manj, za manj popularne pa bolj. Ker poznamo tudi popularnost $q_i$ posamezne vsebine *i*, lahko za $f_i(x)$ trdimo, da je enaka kot $f(x)$, raztegnjena s faktorjem $1 / q_i$ :

$$f_i(x) = f(\frac{1}{q_i} x). \tag{5.5}$$

Povprečno število prenosov $\overline{Y}$ , ki ga doživi omrežje in ga Graf 5.3 prikazuje z debelejšo črtkano črto, je torej povprečje vseh $f_i(x)$ prek vseh *N* različnih vsebin:

$$\overline{Y} = \frac{1}{N} \sum_{i=1}^{N} f(\frac{1}{q_i} x). \tag{5.6}$$

V popolnoma stacionarnem stanju ne-dinamičnega sistema bi bila po določenem številu poizvedb vsa vozlišča ustrezno skonfigurirana za usmerjanje do vseh vsebin. V takem stanju bi bilo povprečno število prenosov poizvedbe $\overline{Y}$ kar enako povprečju stacionarnih vrednosti števila prenosov poizvedbe prek vseh vsebin. V realnem svetu zaradi značilne visoke dinamičnosti omrežja (prihajanje in odhajanje vozlišč) do takega stanje nikoli ne bo prišlo.

S pomanjkanjem in z zahtevnostjo analitičnih modelov se srečujejo tudi drugi raziskovalci na področju nestrukturiranih sistemov enak z enakim, zato si večinoma

pomagajo s simulacijami ali pa s preverjanjem teorij na obstoječih odprtih omrežjih (največkrat na omrežju Gnutella), kadar je to izvedljivo.

### 5.5.3 Stacionarno stanje sistema

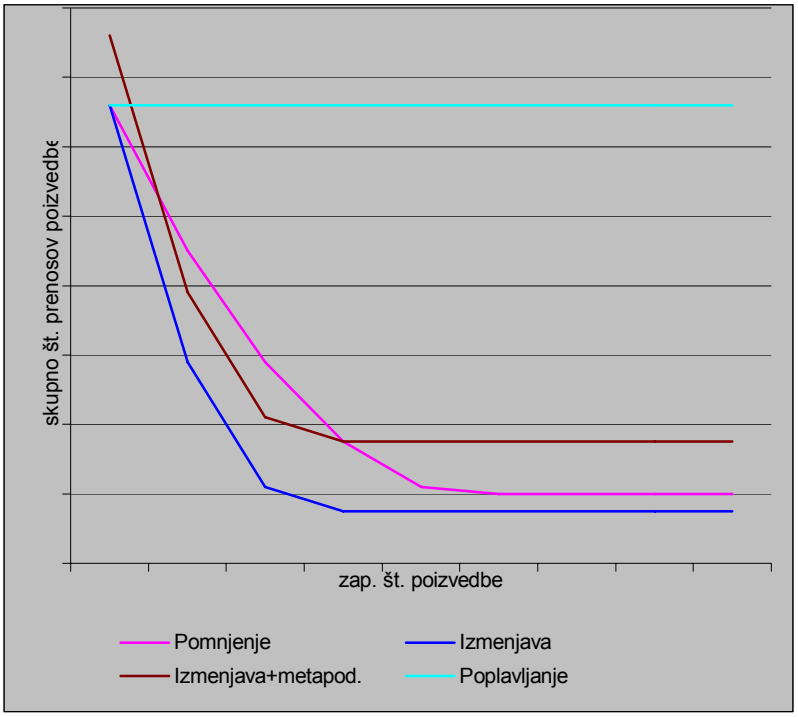
V stacionarnem stanju zavzame povprečno število prenosov poizvedbe konstantno vrednost. Še vedno pa pri obeh predlaganih usmerjanjih pričakujemo velika nihanja v številu prenosov za različne poizvedbe, pač glede na to, katero vsebino iščejo:

- Poizvedbe, ki iščejo popularne vsebine in take vsebine, ki so v sistemu že dovolj dolgo, da so se vozlišča ustrezno skonfigurirala, bodo opravile le nekaj skokov (v rangu življenjske dobe) in ne bodo nikoli poplavljene.
- Poizvedbe, ki iščejo malo manj popularne ali malo bolj sveže vsebine, lahko doživijo tudi kako delno poplavo, ko dosežejo vozlišče, ki te poizvedbe ne zna ustrezno usmeriti.
- Poizvedbe, ki iščejo zelo nepopularne ali pa zelo nove vsebine, bodo poplavljene s strani večine vozlišč, ki jih bodo dosegle.
- Dodatno pa lahko delne poplave doletijo tudi poizvedbe iz vseh gornjih treh skupin, če je prišlo do tako velike spremembe navidezne topologije, da so poti pretrgane in jih je potrebno ponovno najti.

Glede na sistem in na način usmerjanja je povprečno število prenosov sestavljeno kot utežena vsota števila prenosov nepoplavljenih, delno poplavljenih in popolnoma poplavljenih poizvedb, kar velja za oba predlagana načina usmerjanja. Pričakujemo pa, da bo med njima nastopila razlika v povprečnem številu skokov na poizvedbo.

Gotovo bo pri usmerjanju z izmenjavo metapodatkov učinkovitost bistveno višja in več poizvedb bo pravilno usmerjanih na celotni poti do ciljnega vozlišča. Do poplav (tudi delnih) bo prihajalo le pri odhodih vozlišč iz omrežja zaradi prekinjanja že znanih poti, ter ob prvem iskanju vsebine znotraj polmera življenjske dobe poizvedbe (število skokov). Medtem pa bo pri usmerjanju s pomnjenjem posredovanih odgovorov učinkovitost usmerjanja rahlo slabša, ker vozlišča lahko usmerjajo le poizvedbe po tistih vsebinah, za katere je bil že kdaj posredovan odgovor.

Za pošteno primerjavo in ovrednotenje obeh načinov usmerjanja je zato potrebno upoštevati tudi nemajhno obremenitev, ki jo predstavlja izmenjava usmerjevalnih metapodatkov. Denimo, da vozlišča metapodatke povprečno izmenjajo na vsakih $T$ generiranih poizvedb in da je vsako izmed $N$ vozlišč v povprečju povezano z $d$ sosedi. Na vsako generirano poizvedbo je tako potrebno prenesti še $Nd / T$ sporočil, ki predstavljajo izmenjavo metapodatkov. Primerjavo pričakovanega števila prenosov v fazi konfiguriranja in v stacionarni fazi za različne načine usmerjanja prikazuje Graf 5.3.




*Graf 5.3: Predvideno obnašanje različnih vrst usmerjanja: primerjava osnovnega poplavljanja s predlaganima načinoma usmerjanja. Pri usmerjanju z izmenjavo metapodatkov prikazujemo tudi različico, kjer so upoštevani tudi prenosi sporočil z metapodatki, ne samo poizvedb.*

# 6 SPLOŠEN MODEL SISTEMA

Model je abstrakcija realnega sistema, ki jo zgradimo z namenom, da bi realni sistem bolje razumeli in laže napovedovali njegovo obnašanje. V našem primeru želimo s pomočjo implementacije modela napovedati obnašanje različic vsebinskega omrežja pri uporabi predlaganih mehanizmov za usmerjanje poizvedb.

Ker so vsebinska omrežja med seboj lahko zelo različna glede na namen in način uporabe, želimo v model vključiti parametre, katerih vrednosti je moč prilagajati lastnostim izbranega vsebinskega omrežja. Predvsem so pomembni naslednji parametri: vrsta topologije navideznega omrežja, velikost topologije, način porazdelitve vsebin v sistemu in njihova stopnja repliciranosti, za nas najpomembnejši parameter pa je porazdelitev poizvedb in posredno tudi njihova ponovljivost.

## 6.1 Osnovni pojmi

Uporabniki v sistemu generirajo poizvedbe in sprejemajo odgovore. Odgovori vsebujejo kazalce na lokacije vsebin, ki ustrezajo poizvedbi. Vsebine v sistemu so lahko kakršnekoli, največkrat pa so to datoteke (dokumenti, glasba, film, slike, novice, spletne strani) ali pa podatki, shranjeni v relacijskih ali drugačnih podatkovnih bazah.

### *6.1.1 Poizvedba*

**Oblika**. Poizvedba ima lahko kakršnokoli obliko in sintakso, če je le primerna za določeno vrsto podatkov in če jo sistem razume. V sistemu za izmenjavo datotek je poizvedba lahko preprost seznam ključnih besed, lahko regularni izraz, iskanje pa lahko omejuje le na določene metapodatke (ime datoteke, tip datoteke…).

**Izraznost**. Jezik za opis poizvedbe mora imeti primerno izraznost glede na lastnosti celotnega sistema: iskanje dokumentov s pomočjo digitalnih izvlečkov ni primerno, če

želimo najti kak dokument, v katerem je omenjen Osama bin Laden – torej če želimo iskati po vsebini dokumenta. Učinkovito pa bo na primer, če želimo najti točno določeno skladbo Beatlov, ki smo jo prejšnji teden pomotoma izbrisali. Za iskanje po vsebini dokumenta pa bo primernejši seznam ključnih besed. Za iskanje po strukturiranih podatkih, denimo tabelah v relacijski podatkovni bazi, tudi seznam ključnih besed ni prava izbira: tu je potrebne močnejši, strukturiran povpraševalni jezik.

V našem modelu oblika poizvedbe in način zagotavljanja zadostne izraznosti poizvedbe nista opredeljena. Predpostavimo le, da je vozlišče sposobno za dano poizvedbo $p$ in vsebino $v$ ovrednotiti resničnost funkcije *ustreza*($v$, $p$): ali vsebina $v$ ustreza pogojem poizvedbe $p$.

### 6.1.2 Odgovor

V nekaterih primerih uporabniku zadošča, da poizvedba vrne natanko en odgovor, točneje kazalec na en podatek, datoteko oziroma poljubno vsebino, v drugih pa želi več odgovorov, vsaj neko vnaprej določeno število ali pa celo vse možne odgovore. Naš model ima poizvedbo za uspešno, ko je uporabnik prejel en odgovor nanjo. Temu je prilagojena tudi izvedba usmerjanja.

V vseh v nadaljevanju predstavljenih načinih usmerjanja sporočil potuje odgovor do izvornega vozlišča v nasprotno smer po isti poti, kot jo je prepotovala poizvedba.

## 6.2 Topologija

Protokole smo preizkušali oziroma simulirali na več vrstah in velikostih topologij. Vrsta topologije označuje način generiranja in njene lastnosti, ki pogojujejo tudi učinkovitost usmerjanja. V naslednjih razdelkih bomo opisali njihove lastnosti in izbrane topološke različice. Tabela 6.1 v številkah povzema njihove lastnosti.

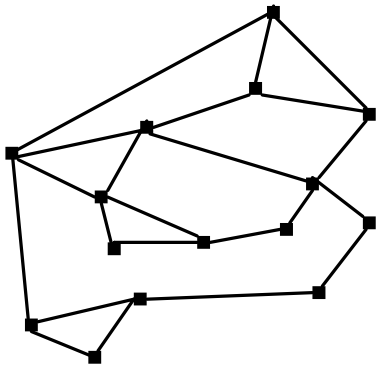
### 6.2.1 Naključni graf

Naključni graf ali naključno topologijo vzamemo kot osnovo za primerjavo z ostalimi topologijami, ki se skušajo čimbolj približati lastnostim, izmerjenim na topologijah realnih omrežij. Naključno topologijo generiramo tako, da generiramo zahtevano število vozlišč in vsakega sproti povežemo z naključno izbranim vozliščem iz množice že generiranih vozlišč. Zahtevano število povezav dosežemo tako, da v nadaljevanju naključno izbiramo pare vozlišč in jih povežemo med seboj. Če je izbrani par že povezan, ga ne povežemo še enkrat, saj nas podvojene povezave ne zanimajo.

| | Št. vozlišč | Št. povezav | Povp. stopnja | Povp. najkr. pot | Max. najkr. pot | Koef. gručenja |
|---|---|---|---|---|---|---|
| **NAKLJUČNI** | 155 | 258 | 3,32 | 4,40 | 9 | 0,0217 |
| **NAKLJUČNI** | 1527 | 2815 | 3,68 | 5,89 | 12 | 0,0016 |
| **2x POVEZAN OBROČ** | 155 | 310 | 4,00 | 19,62 | 39 | 0,5000 |
| **2x POVEZAN OBROČ** | 1527 | 3054 | 4,00 | 191,2 | 382 | 0,5000 |
| **2x POVEZAN OBROČ z 0,2 naključnim prevezovanjem** | 155 | 310 | 4,00 | 4,75 | 9 | 0,2825 |
| **2x POVEZAN OBROČ z 0,2 naključnim prevezovanjem** | 1527 | 3054 | 4,00 | 7,53 | 13 | 0,3850 |
| **GLP** | 155 | 259 | 3,34 | 3,40 | 9 | 0,1450 |
| **GLP** | 1527 | 2816 | 3,69 | 3,928 | 9 | 0,0820 |

*Tabela 6.1: Pregled preizkušenih topologij.*

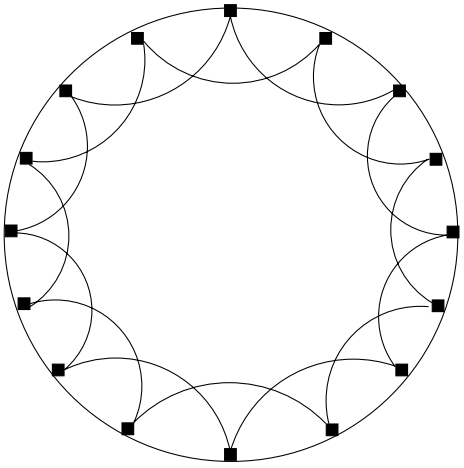
Slika 6-1 prikazuje majhno naključno topologijo s povprečno stopnjo vozlišč 3. Za naključne grafe sta značilna nizek koeficient gručenja in kratka povprečna najkrajša pot v grafu, o čemer se lahko prepričamo tudi s pomočjo prejšnje tabele.



*Slika 6-1: Primer naključne topologije na 15 vozliščih.*

### 6.2.2 Večkratno povezan obroč

Večkratno povezan obroč je pravilna, umetna topologija, ki se ponaša z visokim koeficientom gručenja. Zahtevano število vozlišč najprej povežemo v obroč, ki predstavlja primer enkrat povezanega grafa. Vsako vozlišče *i* je povezano z vozliščema *i*+1 in *i*-1. Za dvojno povezano topologijo povežemo vsako vozlišče še s sosedoma njegovih dveh sosedov, vozlišče *i* torej povežemo še z vozliščema *i*+2 in *i*-2; trojno povezano topologijo dobimo, če dvojno povezanemu obroču dodamo še povezavi *i*+3 in *i*-3 in tako dalje.



*Slika 6-2: Primer dvakratno povezanega obroča na 16 vozliščih.*

Slika 6-2 prikazuje primer takšne topologije. Njene značilnosti so redkost - število povezav je reda $O(n)$ - in pravilnost, poleg tega pa visok koeficient gručenja in dolga povprečna najkrajša pot, kar je razvidno tudi s slike. Koeficient gručenja je visok, ker med štirimi sosedi (vsakega) vozlišča obstajajo tri povezave, najkrajše poti pa so lahko zelo dolge, saj ni nobenih bližnjic na bolj oddaljene dele grafa.

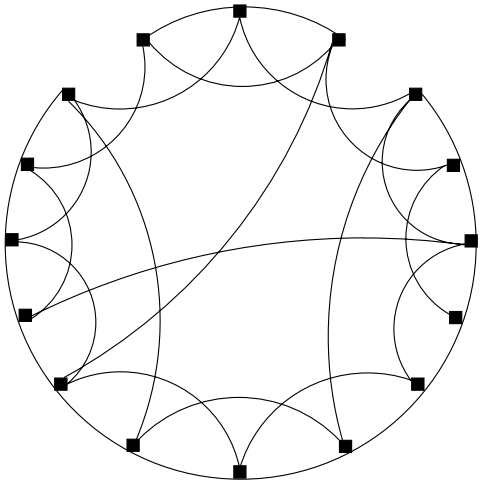
Slabost večkratno povezanega obroča je relativno dolga najkrajša pot med nasproti si ležečimi vozlišči: v večjem grafu, ki smo ga uporabili za simulacije, je povprečna najkrajša pot kar 191 skokov. Posledično sta seveda dolgi tudi najdaljša najkrajša pot v grafu in povprečna najkrajša pot. Večkratno povezan obroč zato ne izpolnjuje ene od zahtev topologije tipa majhen svet (*small world*), namreč dolžine povprečne najkrajše poti, ki bi morala biti primerljiva z dolžino v naključnem grafu. Z majhno modifikacijo,

ki jo predstavljamo v naslednjem razdelku, pa vendarle lahko na osnovi te topologije zgradimo boljši model topologije majhen svet.

### 6.2.3 Večkratno povezan obroč z naključnim prevezovanjem

V primerjavi z naključnim grafom ima večkratno povezan obroč bistveno daljšo povprečno pot ravno zaradi svoje pravilnosti. Če v sistem lahko uvedemo nekaj bližnjic med najbolj oddaljenimi pari vozlišč, se slika bistveno spremeni. Watts in Strogatz sta v odmevnem članku [24] predlagala, da naključno prevežemo določen odstotek povezav, kar pomeni, da eno od vozlišč na koncih povezave zamenjamo z drugim, naključno izbranim izmed preostalih vozlišč.

Ta model nam omogoča, da s pametno izbranim odstotkom prevezav $p$ dosežemo bistveno skrajšanje povprečne najkrajše poti, pri čemer pa se koeficient gručenja bistveno še ne zmanjša. Pri premajhnem $p$ je povprečna najkrajša pot še predolga, pri prevelikem pa se koeficient gručenja preveč zniža in topologija se po lastnostih približa naključnim grafom.



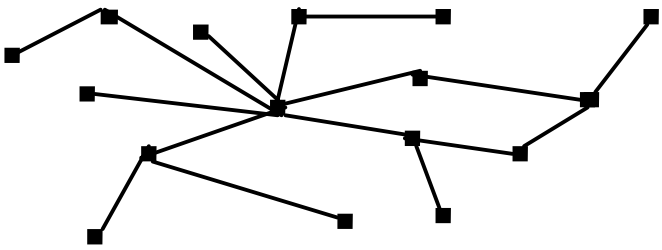
*Slika 6-3: Dvakratno povezan obroč, ki smo mu naključno prevezali 12.5% povezav (štiri povezave).*

Slika 6-3 prikazuje isto topologijo, le da smo štiri povezave naključno prevezali in s tem ustvarili štiri bližnjice, ki pripomorejo k bistveno krajši povprečni najkrajši poti v grafu. S primerno izbranim odstotkom glede na izbrano velikost grafa dosežemo lastnosti, ki so

značilne za topologijo majhnega sveta (bistveno višji koeficient gručenja in primerljiva povprečna najkrajša pot kot v naključnem grafu).

### 6.2.4 Topologija GLP

Topologija s posplošeno linearno preferenco ali krajše GLP (Generalized Linear Preference), ki smo jo podrobneje opisali v razdelku 4.1.3, predstavlja umetno generirano topologijo, ki se poskuša čimbolj približati lastnostim topologije interneta na nivoju avtonomnih sistemov. V več študijah ([15], [16], [19]) je bilo ugotovljeno, da v topologiji interneta veljajo potenčni zakoni, prav tako pa ima tudi lastnosti majhnega sveta. Ugotavljajo tudi, da so te lastnosti značilne tudi za topologije navideznih omrežij, zato lahko z ustreznimi nastavitvami parametrov generirane topologije GLP uporabimo tudi kot model navideznega omrežja.



*Slika 6-4: Primer topologije GLP na 16 vozliščih. Vidimo eno vozlišče z zelo visoko stopnjo in mnogo vozlišč s stopnjo ena.*

## 6.3 Usmerjanje

Namen tega dela je tudi s simulacijo pokazati razlike med predlaganimi načini usmerjanja poizvedb v nestrukturiranih omrežjih enak z enakim. Zato bomo s pomočjo simulacije prikazali obnašanje in lastnosti treh mehanizmov usmerjanja, ki smo jih podrobno opisali v razdelku 5, v različnih okoljih in okoliščinah:

- Osnovno poplavljanje
- Usmerjanje s pomnjenjem posredovanih odgovorov.
- Usmerjanje z izmenjavo usmerjevalnih metapodatkov.

## 6.4 Velikost

Velikost grafa merimo s številom vozlišč in povezav. Za naše simulacije smo generirali topologije dveh velikosti, manjšo okrog 150 vozlišč in večjo okrog 1500 vozlišč. Velikost grafa je izbrana tako, da premer večjega grafa (najdaljša najkrajša pot) v številu skokov presega povprečno življenjsko dobo sporočila, premer manjšega grafa pa ne. Tako dosežemo, da v manjši topologiji poplava doseže vsa vozlišča v grafu, v večji topologiji pa se to zgodi le redko.

Število povezav v generiranih grafih ni poljubno, ampak ga izberemo tako, da topologija tudi po ostalih lastnostih in metrikah ustreza eksperimentalnim rezultatom, pridobljenih na realnih, živih topologijah.V sistemih, ki uporabljajo protokol Gnutella, sta Ripeanu in Foster [16] v več zaporednih eksperimentih ugotovila, da se povprečna stopnja vozlišč v sistemih enak z enakim ne spreminja z velikostjo sistema in da znaša približno 3.4 povezav, zato bomo to vrednost privzeli tudi v simulacijah.

## 6.5 Porazdelitev in ponavljanje poizvedb

Ker je bistvo te naloge izboljšati usmerjanje ponavljajočih se poizvedb, je porazdelitev poizvedb za nas zelo pomemben podatek, saj posredno določa **delež ponovljenih poizvedb**, t.j. poizvedb, ki iščejo vsebino, po kateri je pred tem že poizvedovala kaka poizvedba. Zavedati pa se moramo, da je konec koncev **delež ponovitev odvisen od dolžine opazovanega časovnega intervala**: če bomo poizvedbe opazovali dovolj dolgo, se bodo nazadnje vse le še ponavljale, saj imamo v sistemu le končno množico razpoložljivih vsebin, po katerih povprašujejo poizvedbe. Zato je najbolj smiselna medsebojna primerjava načinov usmerjanja na rezultatih simulacij, izvršenih z enakim številom poizvedb.

Pod pojmom **porazdelitev poizvedb** si predstavljamo porazdelitev deležev poizvedb, ki povprašujejo po posamezni vsebini, med vsemi poizvedbami. Vsebine, ki so bolj zaželene in po njih v določenem časovnem obdobju povprašuje večje število poizvedb, imajo ta delež večji kot tiste, po katerih povprašuje manjše število poizvedb, zato ga bomo imenovali relativna popularnost ali tudi zaželenost.

Zaželenost $q_i$ vsebine $i$ v določenem časovnem obdobju izračunamo kot

$$q_i = \frac{Q_i}{Q}, \tag{6.1}$$

kjer $Q_i$ predstavlja število poizvedb, ki so v opazovanem časovnem intervalu povpraševale po vsebini $i$, $Q$ pa število vseh poizvedb v tem časovnem intervalu.

V naših simulacijah bomo predpostavili, da se v času trajanja simulacije porazdelitev poizvedb in z njo popularnost vsebin ne spreminjata.

### 6.5.1 Uniformna porazdelitev poizvedb

Za uniformno porazdelitev je značilno, da so vse vsebine približno enako zaželene. Relativne popularnosti so torej med seboj enake in za vsako vsebino velja

$$q_i = \frac{Q_i}{Q} = \frac{1}{D}, \tag{6.2}$$

pri čemer je $D$ število vseh različnih vsebin oziroma dokumentov v sistemu. V realnih sistemih uniformne porazdelitve praktično ne srečamo, saj so vedno določene vsebine bolj zanimive ali vsaj zanimive za širši krog kot druge.

### 6.5.2 Zipfova porazdelitev poizvedb

Več študij realnih sistemov je pokazalo, da porazdelitev poizvedb, ki jih generirajo "pravi", živi uporabniki, sledi Zipfovi porazdelitvi: če vse vsebine uredimo v zaporedje glede na njihovo relativno popularnost, ugotovimo, da je zaželenost posamezne vsebine proporcionalna z obratno vrednostjo njenega zaporednega mesta $i$, potenciranega na konstantno potenco $\alpha$, ki je od sistema do sistema različna. Zipfovo porazdelitev v splošni obliki ponazarja spodnja formula:

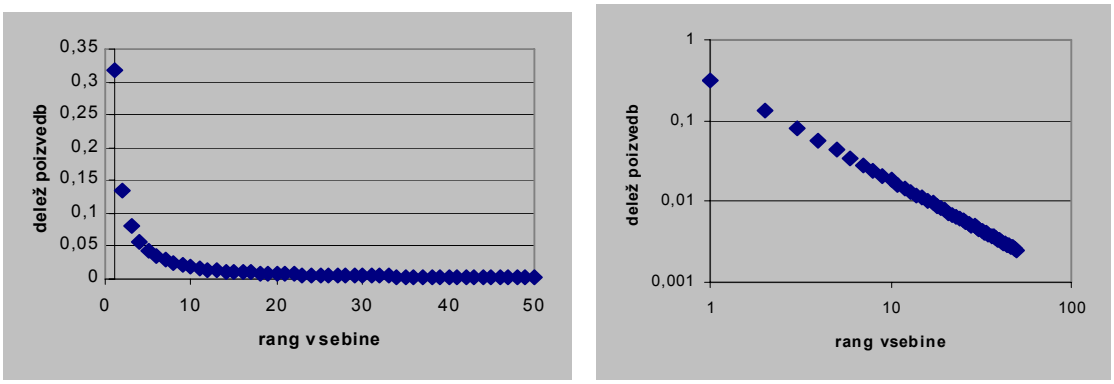
$$q_i \propto \frac{1}{i^{\alpha}} \tag{6.3}$$

Da dobimo pravi $q_i$ kot delež izmed vseh poizvedb, moramo vrednosti še normalizirati.

Sripanidkulchai v [8] ugotavlja to zakonitost v nestrukturiranem sistemu enak z enakim s protokolom Gnutella, namenjenim za izmenjavo datotek. Izkustveno je ugotovljeno, da je krivulja porazdelitve poizvedb sestavljena iz dveh segmentov. Prvi segment zajema približno 100 najpogostejših poizvedb (t. j. poizvedb po najpopularnejših vsebinah), ki so skoraj enako pogoste in bi jih lahko celo grobo aproksimirali z uniformno porazdelitvijo, ali pa točneje z Zipfovo formulo z nizko vrednostjo parametra $\alpha$ (blizu 0). Iz zbranih podatkov je izračunano, da se parameter $\alpha$ v drugem segmentu, ki zajema poizvedbe ranga od 100 do 100.000, giblje med 0,63 in 1,24. Primerljive vrednosti za popularnost vsebin bomo uporabili tudi mi v naši simulaciji.

Zipfova formula sodi v široko družino potenčnih zakonov, ki smo jih omenjali že pri lastnostih topologije. Njen obstoj nam na primeru porazdelitve poizvedb v vsebinskih omrežjih pove, da velika večina poizvedb povprašuje po morda treh ali petih, oziroma po majhni podmnožici razpoložljivih vsebin. Nasprotno pa po večjem delu vsebin poizveduje le malo poizvedb. Tako se pogosto ponavljajo poizvedbe po zaželenih vsebinah (na primer po aktualnih glasbenih ali filmskih datotekah), poizvedbe po manj zaželenih vsebinah pa so redke in imajo nizko ponovljivost.

Oglejmo si primer Zipfove porazdelitve poizvedb v sistemu s 50 različnimi dokumenti, če je konstantni faktor enak 1, za $\alpha$ pa vzamemo 1,24.




*Slika 6-5: Zipfova porazdelitev poizvedb – levo navaden graf, desno isti graf z logaritemsko skalo na obeh oseh. Parameter $\alpha$ opisuje naklon premice, ki jo potegnemo skozi točke na desnem grafu.*

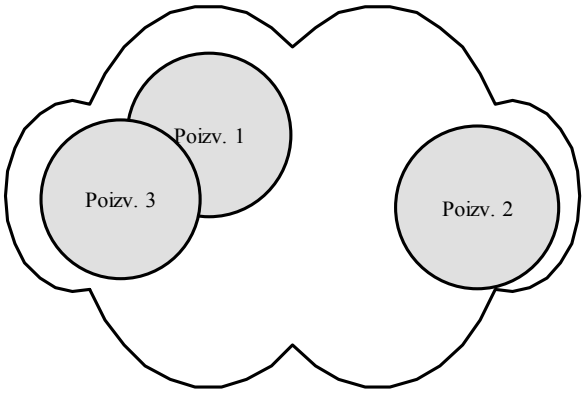
Vidimo lahko, da po najbolj zaželenem dokumentu povprašuje več kot tretjina poizvedb, medtem ko po dokumentih, ki so pod desetim mestom, poizveduje le odstotek ali še manj poizvedb.

### 6.5.3 Vpliv porazdelitve na ponavljanje poizvedb in usmerjanje

Omenili smo že, da porazdelitev poizvedb posredno vpliva na ponavljanje poizvedb. V naših simulacijah bomo primerjali sisteme z enakim deležem ponovljenih poizvedb, vendar različno porazdelitvijo in videli bomo, da so rezultati lahko zelo različni.

Pri uniformni porazdelitvi poizvedb se med ponovljenimi poizvedbami pojavljajo vsebine enakomerno. Z drugimi besedami povedano, manjše število poizvedb (na primer 2 ali 3) išče eno vsebino, primerljivo število drugih poizvedb spet drugo vsebino in podobno. Ker je omrežje veliko, doseg poizvedbe pa manjši od celotnega omrežja, ena sama poizvedba ne skonfigurira velikega dela omrežja (informacija o usmerjanju se zapiše le na vozliščih, ki posredujejo odgovor).

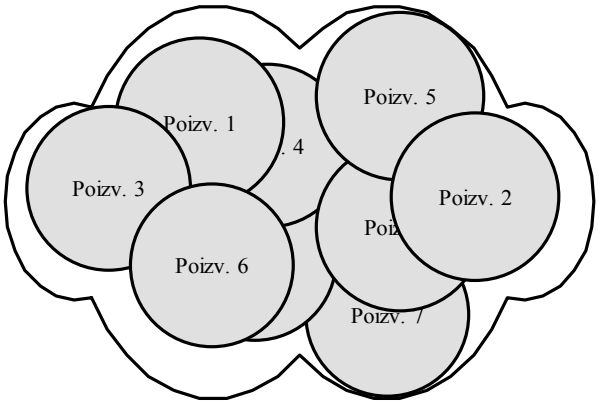
Slika 6-6 prikazuje situacijo, do katere prihaja pri uniformni porazdelitvi poizvedovanja. Če je število ponovitev poizvedbe za posamezno vsebino nizko, število vsebin veliko in povprečna življenjska doba vozlišča nizka, je lahko kljub visoki stopnji ponovljenih poizvedb učinkovitost usmerjanja le malo boljša od poplavljanja.




*Slika 6-6: V dinamičnem omrežju pri uniformni porazdelitvi poizvedb pridobi zaradi manjšega števila poizvedb po določeni vsebini usmerjevalne metapodatke le manjši del omrežja. V primeru na sliki so tri poizvedbe skonfigurirale vsaka svoje področje omrežja,*

*vendar pa je bila bolj učinkovita glede usmerjanja le poizvedba 3 na tistem delu, kjer se njeno področje prekriva s področjem poizvedbe 1.*

Pri Zipfovi porazdelitvi poizvedb pa je med ponovljenimi poizvedbami največ takih, ki poizvedujejo po manjšem številu najbolj zaželenih vsebin. Za vsako od teh vsebin je število poizvedb bistveno višje kot pri uniformnem poizvedovanju, zato se omrežje za usmerjanje do teh vsebin hitro skonfigurira in se pri naslednjih poizvedbah uporablja večji delež pridobljenih usmerjevalnih metapodatkov, kar posledično prinaša višjo učinkovitost usmerjanja.




*Slika 6-7:Pri Zipfovi porazdelitvi poizvedb so usmerjevalni metapodatki pri najpogostejših poizvedbah veliko bolje izkoriščeni, kar vodi do višje učinkovitosti usmerjanja.*

Pri Zipfovi porazdelitvi poizvedovanja ne smemo pozabiti, da višjo stopnjo ponovljenih poizvedb za bolj zaželene vsebine na drugi strani odtehta zelo nizka stopnja ponovljivosti ali celo odsotnost ponovljivosti poizvedovanja po manj zaželenih vsebinah. Te poizvedbe se morajo zato v dinamičnem sistemu skoraj vedno poplaviti. Vprašajmo se, ali to dejstvo znižuje povprečno učinkovitost usmerjanja? Gotovo je to dejavnik, ki povzroča več prenosov poizvedb, vendar nismo bistveno bolj na slabšem kot pri poizvedovanju z uniformno porazdeljenostjo, saj smo ravnokar pokazali, da pri nizki stopnji ponovljivosti poizvedb za posamezno vsebino učinkovitost ne more biti znatno boljša kot pri poplavljanju.

## 6.6 Porazdelitev in repliciranost vsebin

Vsebine ali datoteke, ki jih nudijo posamezna vozlišča, se večinoma nahajajo na več kot enem vozlišču, so torej replicirane. Stopnja repliciranosti $r_i$ za posamezno vsebino predstavlja delež vozlišč, ki jo nudijo, glede na vsa vozlišča:

$$r_i = \frac{N_i}{N}, \tag{6.4}$$

kjer $N$ predstavlja število vseh vozlišč, $N_i$ pa število vozlišč, ki nudijo vsebino $i$.

Višja stopnja repliciranosti za posamezno vsebino pomeni večjo verjetnost, da poizvedba že kmalu najde svojo ciljno vsebino in tako manj obremenjuje omrežje kot celoto. Za izvedbo simulacij visoke stopnje repliciranosti ne bodo zanimive, saj v tem primeru poizvedbe že kmalu najdejo odgovore in se zato ne morejo tako lepo izraziti lastnosti in razlike različnih načinov usmerjanja poizvedb. Tudi v realnih sistemih je povprečna stopnja repliciranosti vsebin relativno nizka, obenem pa je število različnih razpoložljivih vsebin zelo visoko.

### *6.6.1 Uniformna repliciranost*

V primeru uniformne repliciranosti so vse stopnje med seboj enake, vsi dokumenti se torej nahajajo na enakem številu vozlišč. Skrajna primera uniformne porazdelitve sta

$$r_i N = 1, \tag{6.5}$$

kjer se vsak dokument nahaja le na enem vozlišču in torej ne moremo govoriti o replikaciji v pravem pomenu besede, ter

$$r_i = 1, \tag{6.6}$$

kjer se vsak dokument nahaja na prav vseh vozliščih, gre torej za popolno replikacijo. Primer popolne replikacije za nas ni zanimiv, saj vozlišča ne bodo poizvedovala po datotekah, ki jih že imajo. V simulacijah bomo pri uniformni porazdelitvi predpostavljali nizko stopnjo replikacije.

Pripomnimo naj, da je v resnici uniformna porazdelitev manj verjetna, saj so gotovo bolj razširjene kopije tistih vsebin, ki so bolj popularne in po katerih je zato več povpraše-

vanja. Posledica povpraševanja je namreč navadno prenos vsebine na novo lokacijo in tako nastanek nove kopije. Na dolgi rok se število kopij ustali pri višji številki za bolj zanimive vsebine in pri nižji za manj zanimive.

### 6.6.2 Proporcionalna repliciranost

Če razvijamo zaključno misel iz prejšnjega razdelka, lahko zapišemo povezavo med popularnostjo posamezne vsebine $r_i$ in številom njenih kopij v sistemu, $q_i$:

$$r_i \propto q_i \,. \tag{6.7}$$

Število kopij posameznega dokumenta je sorazmerno številu poizvedb po tem dokumentu. Proporcionalno repliciranost lahko predpostavimo le v sistemih, kjer ugotovimo dvoje:

- da vozlišča pridobivajo večino novih vsebin s pomočjo poizvedovanja in
- da vsebine, ki jih prenesejo k sebi, nato večinoma tudi nudijo ostalim vozliščem.

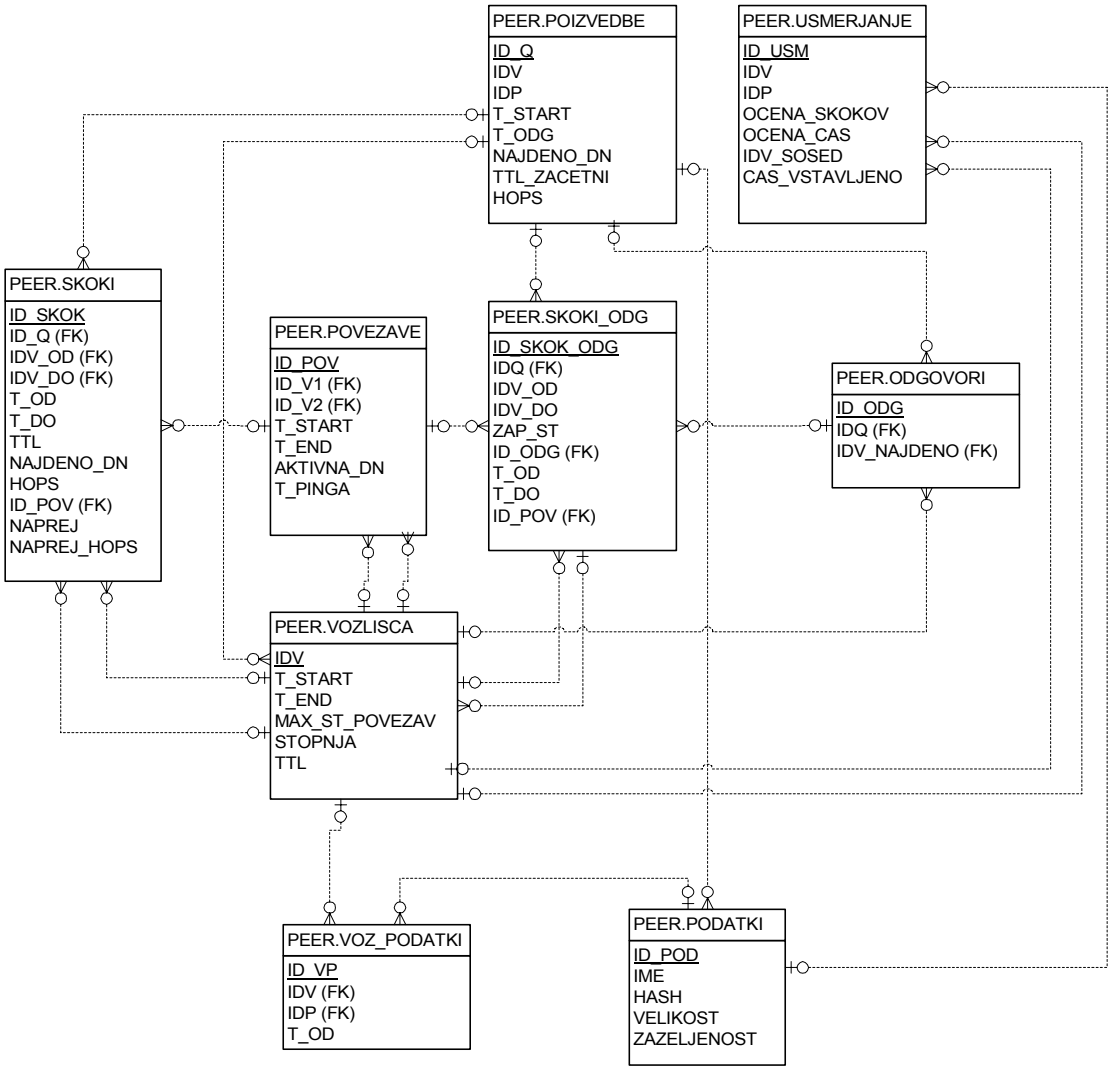
V nasprotnem primeru, ko torej nove vsebine vstopajo v sistem po drugih poteh, neodvisno od poizvedovanja, proporcionalna repliciranost ni nujno utemeljena.

Ker smo v naših simulacijah predpostavili porazdelitev poizvedb po Zipfu, bomo imeli v primeru proporcionalne repliciranosti tudi pri replike porazdeljene v skladu z Zipfovo porazdelitvijo.

Porazdelitev replik lahko povežemo s popularnostjo vsebine v nekem preteklem obdobju, po drugi strani pa tudi z načinom vstopanja novih vsebin v sistem. Če ta ni povezan s popularnostjo, je v nekaterih sistemih kljub temu bolj upravičeno predpostaviti uniformno porazdelitev replik. Zato bomo v simulacijah testirali obe varianti.

## 6.7 Izvedba simulacij

Kot osnovo za izvedbo simulacij smo izbrali podatkovni bazi Oracle 9i, v katero smo shranjevali vse podatke simulacij. Slika 6-8 prikazuje podatkovni model, ki smo ga zasnovali v ta namen..




*Slika 6-8: Podatkovni model baze, kjer se hranijo vsi podatki v zvezi s simulacijo. Na sliki nista prikazani tabeli Topologije in Rezultati, ker sta predvsem administrativnega značaja.*

Tabela PODATKI vsebuje seznam vseh vsebin, dostopnih v sistemu. V tabeli VOZLIŠČA je seznam vseh vozlišč, katerih povezave so razvidne iz tabele POVEZAVE. V tabeli VOZ_PODATKI izvemo, katero vozlišče ima kateri podatek oziroma vsebino. Generirane poizvedbe shranimo v tabelo POIZVEDBE, vse njihove skoke pa shranjujemo v tabelo SKOKI. Ko poizvedba najde vozlišče z iskano vsebino, se generira odgovor, ki ga zapišemo v tabelo ODGOVORI, njegove skoke do izvornega vozlišča pa zapisujemo v tabelo SKOKI_ODG. Usmerjevalne metapodatke pri usmerjanju z izmenjavo metapodatkov za vsa vozlišča shranjujemo v tabelo USMERJANJE.

Postopki za generiranje topologij, analizo topologij, izvedbo simulacij in analizo rezultatov so izvedeni delno v PL/SQL, delno v Javi. Oraclova podatkovna baza namreč dobro podpira sodelovanje z javanskimi programi: javanske procedure lahko shranimo v podatkovno bazo in jih uporabljamo, kakor da bi bile klasične bazne procedure, ali pa v javanski program vključimo razrede SQLJ, kjer lahko med javansko kodo pišemo SQL stavke za manipulacijo s podatki.

# 7 OCENA MODELA

Vsak model predstavlja abstrakcijo realnega sistema, s katero želimo izboljšati naše razumevanje delovanja sistema in v določenih okoliščinah tudi predvideti obnašanje sistema. Kako dobro se lahko abstrakcija prilagaja realnosti, je odvisno od poznavanja realnega sistema in od tega, kako realni sistem opazujemo in merimo njegove lastnosti.

Model ocenjujemo s pomočjo verifikacije in validacije njegovega delovanja. Verifikacija modela predstavlja ugotavljanje pravilnosti implementacije modela, torej pravilnosti in skladnosti delovanja s konceptualno definicijo. Validacija pa predstavlja ugotavljanje, ali konceptualna zasnova sploh ustreza realnosti.

## 7.1 Validacija konceptualnega modela

V tem razdelku bomo preverili in utemeljili domneve, na katerih sloni v prejšnjem poglavju predstavljeni model sistema enak z enakim. S tem bomo pokazali, da kljub določenim abstrakcijam model za naš namen še vedno dovolj dobro predstavlja realna vsebinska omrežja tipa A oziroma nestrukturirane sisteme enak z enakim.

Ker je naš namen opazovati in primerjati obnašanje treh protokolov za usmerjanje poizvedb, so bistvene tri lastnosti modela, ki morajo odražati lastnosti realnih sistemov, namreč topologija, porazdelitev poizvedb in porazdelitev vsebin.

### *7.1.1 Topologija*

Model sloni na domnevi, da v navidezni topologiji veljajo potenčni zakoni in lastnost majhnega sveta. Domnevo lahko preverimo tako, da se z modificirano aplikacijo vključimo v sistem enak z enakim in sledimo povezavam od vozlišča do vozlišča. To je invaziven poseg v omrežje, ki povzroča veliko prometa, obenem pa ostalim udeležencem ne prinaša nobenih koristi, ker modificirana aplikacija ne opravlja običajnih nalog

udeleženca sistema enak z enakim. Sami zato lastnosti topologije sistemov enak z enakim nismo preverjali, pač pa se opiramo na navedbe in eksperimente, opisane v razpoložljivi literaturi, predvsem [16] in [19], kjer avtorji pokažejo, da nestrukturirana vsebinska omrežja enak z enakim izkazujejo lastnosti majhnega sveta in potenčnih zakonov.

Pri generiranju topologije smo izbrali takšno kombinacijo vhodnih parametrov, da generirana topologija po tistih lastnostih, ki jih lahko izmerimo ali izračunamo, ustreza vrednostim, ki so eksperimentalno ugotovljene in navajane v literaturi (predvsem gre za povprečno stopnjo vozlišč), kot smo to navedli že v razdelku 6.4.

### *7.1.2 Porazdelitev poizvedb*

Pod pojmom porazdelitev poizvedb mislimo predvsem na to, po katerih vsebinah povprašujejo poizvedbe, točneje na frekvenco, s katero se pojavljajo poizvedbe po posamezni vsebini. Model predpostavlja Zipfovo porazdelitev poizvedb, ki jo eksperimentalno ugotavlja Sripanidkulchai [8] na omrežju Gnutella, kot smo navedli že v razdelku 6.5.2. Gnutella je odprt protokol za izmenjavo datotek v javni domeni, dokaj dobro dokumentiran in kot tak zelo primeren za preučevanje. Čeprav se je v zadnjem letu število njegovih uporabnikov močno zmanjšalo na račun odjemalcev, ki uporabljajo druge podobne protokole, je uporabniku vedno dostopnih nekaj tisoč vozlišč, kar je dovolj reprezentativen vzorec sistema za izmenjavo vsebin.

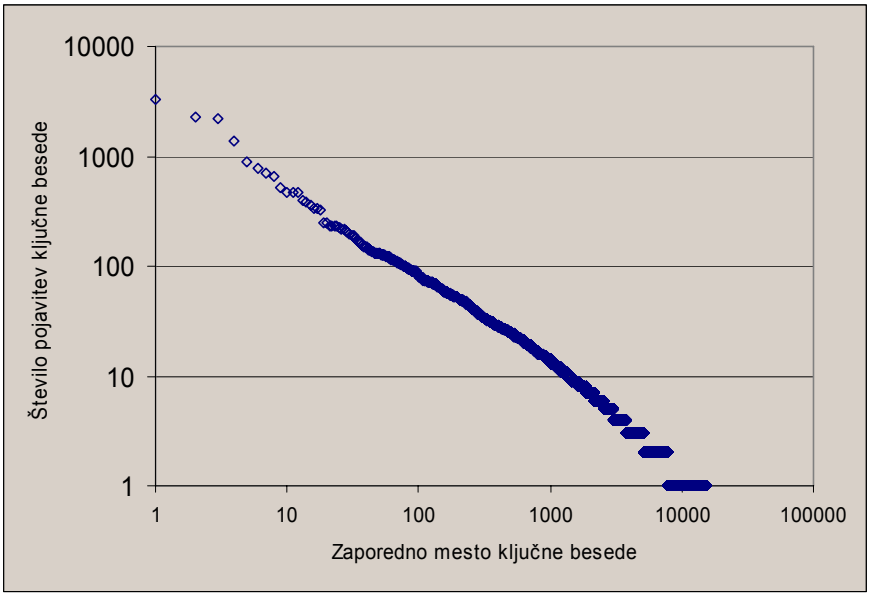
Za empirično validacijo modelirane porazdelitve poizvedb smo uporabili prirejeno implementacijo odjemalca po protokolu Gnutella, ki shranjuje prejete poizvedbe. Ključne besede iz teh poizvedb smo analizirali in s tem potrdili zgoraj citirane ugotovitve.

V treh poskusih smo prestregli 730 000 ključnih besed, od tega 75 000 različnih. Najpopularnejše ključne besede so bile pravzaprav podaljški imen datotek (iskanje po tipu) mp3, avi in mpg, ki so se v najdaljšem poskusu pojavile med 10 000 in 20 000 krat, ter angleški členi in vezniki (*the, and, an*) z nekaj manj pojavitvami. "Prave" ključne besede so se začele pod 2000 pojavitvami. Besedo *matrix* smo našteli 1311-krat , *love* 1550-krat, *Microsoft* smo zasledili 201-krat, *Linux* pa 198-krat; *Eminem* 439-krat, *Elvis* pa 134-krat. Na tisoče ključnih besed je imelo le eno ali dve pojavitvi.

Frekvence pojavljanja smo grafično predstavili na grafu z logaritemsko skalo na obeh oseh, pri čemer smo že vizualno lahko ugotovili, da so točke razporejene blizu ravne črte. S tem potrjujemo ugotovitev, da gre za Zipfovo porazdelitev; ugotoviti moramo le še ustrezno vrednost za potenco $\alpha$.

Ker so grafi med seboj navidez skoraj enaki, predstavljamo le enega izmed njih (Graf 7.1). S pomočjo linearne regresije smo točke aproksimirali s padajočo premico, katere naklon se je gibal v razponu od –0.96 do –1.26.

Ker smo vrednost –1.26 dobili pri največjem vzorcu, menimo da smo tudi za naše simulacije v Zipfovi formuli uporabili primerno vrednost $\alpha = -1.2$.




*Graf 7.1: Frekvence pojavljanja ključnih besed v poizvedbah, generiranih v omrežju Gnutella v juniju 2003.*

### 7.1.3 Porazdelitev vsebin

Porazdelitve vsebin eksperimentalno nismo mogli preveriti, zato svoje razmišljanje opiramo na način porazdelitve poizvedb. Zaradi velike popularnosti sistemov za izmenjavo datotek lahko upravičeno sklepamo, da uporabniki večino svojih datotek dobijo prek takšnega sistema, torej kot posledico generirane poizvedbe, ki ji sledi odgovor in nato največkrat še prenos datoteke k izvoru poizvedbe. Zato domnevamo, da

so vsebine porazdeljene po sistemu na enak način kot poizvedbe, t. j. tiste vsebine, po katerih poizveduje največ uporabnikov, so posledično tudi najbolj replicirane (ker so si jih uporabniki, ki so poizvedovali po njih, nato tudi prenesli).

Svoje razmišljanje lahko vsaj delno potrdimo tudi z navedbami Yangove v [37], ki je zbirala statistične podatke v hibridnem sistemu enak z enakim OpenNap. Ker je sistem bistveno bolj centralizirano usmerjen, je bilo možno zbrati več statističnih podatkov o obnašanju uporabnikov. Po drugi strani smo danes priča masovnemu prehajanju uporabnikov od enega do drugega sistema za izmenjavo datotek glede na to, kateri je v danem trenutku najhitrejši in nudi največ vsebin. Zato lahko predvidevamo, da gre za isto populacijo in da navedbe Yangove vsaj v grobem držijo tudi za uporabnike nestrukturiranih sistemov za izmenjavo vsebin.

Zanimiva podatka sta dva: Yangova namreč navaja, da se off-line, torej mimo sistema spremeni le 0.1% vsebin, vse ostale spremembe – torej 99.9% – se zgodijo kot posledica poizvedb. Druga navedba pa je, da ima 50% poizvedb za posledico tudi dejanski prenos datoteke.

Če smo v prejšnjem razdelku prepričali, da so poizvedbe porazdeljene po Zipfu, lahko zato z zadostno gotovostjo trdimo, da se na daljši rok tudi vsebine po sistemu porazdelijo na enak način, torej v skladu z Zipfovo porazdelitvijo.

Pripomnimo pa naj, da to velja le za javno dostopen sistem za izmenjavo datotek. Čim se bi spremenila namembnost sistema (npr. namenjen bolj za arhiviranje) ali populacija (npr. "sebična" populacija prenesenih vsebin ne bi več nudila ostalim uporabnikom), se lahko vstopanje novih vsebin in njihova porazdelitev popolnoma spremeni.

Zato si bomo kot alternativo Zipfove porazdelitve vsebin v simulacijah ogledali tudi uniformno porazdelitev vsebin.

## 7.2 Verifikacija modela

Verifikacija modela pomeni preverjanje skladnosti med implementacijo modela in konceptualnim modelom. S to aktivnostjo želimo pokazati, da računalniška simulacija res deluje tako, kot pričakujemo.

Programsko kodo vseh modulov smo že med programiranjem testirali na običajne načine, npr. testiranje različnih možnih poti skozi kodo. Nato smo testirali po metodi črne škatle z različnimi kombinacijami vhodnih parametrov: običajne vrednosti, mejne vrednosti, neveljavne vrednosti. Preverili smo, ali se lahko zgodijo vsi veljavni dogodki in ali se ne more zgoditi noben neveljaven dogodek. Nazadnje smo izvedli integracijsko testiranje in na majhnih primerih omrežja korak za korakom sledili celotno simulacijo, da smo se prepričali, da res poteka po pričakovani poti. Nazadnje smo testirali še izvajanje na velikih omrežjih, kjer pa smo zaradi obsežnosti preverjali le značilne vmesne točke.

### 7.2.1 Verifikacija generiranja topologije

Generirali smo različne večje in manjše topologije različnih vrst in si jih tudi vizualno predstavili. Izračunali smo parametre, ki opisujejo značilnosti generiranega grafa in preverili, da ustrezajo pričakovanim vrednostim (število vozlišč in povezav, povprečna stopnja, povprečna najkrajša pot, premer grafa, najdaljša najkrajša pot, koeficient gručenja). Nekatere od teh topologij lahko vidimo na slikah v razdelku 6.2.

### 7.2.2 Verifikacija generatorja naključnih števil

Za generiranje naključnih vrednosti, potrebnih med potekom simulacije, je bil uporabljen generator naključnih števil iz Oraclovega paketa DBMS_RANDOM, ki se lahko uporablja s podatkovno bazo Oracle 9i.

Generiranje naključnih vrednosti smo preverili s testom $\chi^2$. Ničelna hipoteza $H_0$ je bila: Generator vrača zaporedje neodvisnih spremenljivk, ki so enakomerno porazdeljene po intervalu [0,1]. Test smo izvajali pri 1000 prostostnih stopnjah. Izvedli smo teste na treh serijah po milijon vrednosti. Za vsako serijo smo generator inicializirali z drugo vrednostjo semena.

Rezultat je bil v vseh primerih enak: pri stopnji značilnosti $\alpha = 0.05$, ki je običajna za takšne teste, ne moremo zavrniti ničelne hipoteze o ustreznosti generatorja naključnih števil. Generator torej sprejmemo kot ustreznega.

### 7.2.3 Verifikacija porazdelitve vsebin in poizvedb

Porazdelitev vsebin smo testirali tako, da smo množici vsebin generirali popularnosti po Zipfovi formuli in nato preverili porazdelitev s pomočjo testa $\chi^2$, ustreznost potence $\alpha$ pa smo preverili s pomočjo linearne regresije nad točkami grafa z logaritemsko skalo na obeh oseh – podobno, kot smo delali pri validaciji modela in opisali v razdelku 7.1.2.

Za uniformno porazdelitev smo vsem vsebinam dodelili enako popularnost, zato ustreznosti porazdelitve nismo dodatno preverjali.

Verifikacijo porazdelitve poizvedb smo izvedli na enak način, le da smo predhodno izvedli simulacijo 100 generiranih poizvedb, prešteli vsebine, po katerih so poizvedovale te poizvedbe, in nad njimi preverili porazdelitev.

Vse porazdelitve so se izkazale za ustrezne.

### 7.2.4 Verifikacija postopkov usmerjanja

Postopke usmerjanja smo testirali na manjši topologiji GLP. Poleg pričakovanega obnašanja (vozlišče posreduje poizvedbo vsem sosedom ali izbranemu sosedu) smo pozornost posvečali zlasti robnim primerom: ko se najde odgovor, se poizvedbe ne posreduje naprej; kako usmerjajo vozlišča stopnje ena; preverjanje izteka življenjske dobe poizvedbe; shranjevanje usmerjevalnih metapodatkov, ...

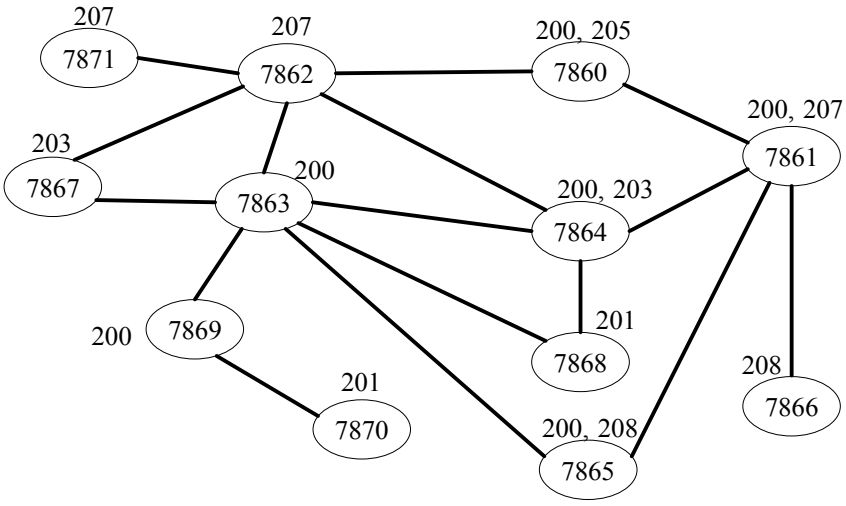
Pri verifikaciji postopkov usmerjanja smo preverili tudi, ali ima več zaporednih simulacij v enakih pogojih (topologija, porazdelitev vsebin in poizvedb, usmerjanje) tudi enake rezultate. Seveda rezultati niso enaki, število prenosov poizvedbe namreč že znotraj ene simulacije močno skače. Če pa gledamo gibajoče povprečje prek več poizvedb, dobimo skoraj identično obliko krivulje in posledično lahko trdimo, da je možno rezultate simulacije kadarkoli reproducirati.

V nadaljevanju predstavljamo primer sledenju izvajanja usmerjanja na vzorčni topologiji GLP z 12 vozlišči. Slika 7-1 in Slika 7-2 prikazujeta topologijo v grafični obliki in ustrezajoče zapise v podatkovni bazi.

| ID_POV | ID_V1 | ID_V2 |
|---|---|---|
| 15646 | 7861 | 7860 |
| 15654 | 7861 | 7864 |
| 15647 | 7862 | 7860 |
| 15648 | 7863 | 7862 |
| 15660 | 7863 | 7868 |
| 15649 | 7864 | 7862 |
| 15656 | 7864 | 7863 |
| 15657 | 7865 | 7861 |
| 15650 | 7865 | 7863 |
| 15651 | 7866 | 7861 |
| 15652 | 7867 | 7862 |
| 15653 | 7867 | 7863 |
| 15655 | 7868 | 7864 |
| 15658 | 7869 | 7863 |
| 15659 | 7870 | 7869 |
| 15661 | 7871 | 7862 |

| IDV | IDP |
|---|---|
| 7860 | 205 |
| 7860 | 200 |
| 7861 | 200 |
| 7861 | 207 |
| 7862 | 207 |
| 7863 | 200 |
| 7864 | 200 |
| 7864 | 203 |
| 7865 | 208 |
| 7865 | 200 |
| 7866 | 208 |
| 7867 | 203 |
| 7868 | 201 |
| 7869 | 200 |
| 7870 | 201 |
| 7871 | 207 |

*Slika 7-1: Povezave v preprostem GLP omrežju na 12 vozliščih (id_pov: identifikator povezave, id_v1in id_v2: identifikatorja vozlišč) in porazdelitev vsebin po vozliščih (idv: identifikator vozlišča, idp: identifikator vsebine).*




*Slika 7-2: Grafični prikaz topologije vzorčnega omrežja. V ovalih so oznake vozlišč (identifikatorji), ob vozliščih pa so identifikatorji vsebin, ki se nahajajo na posameznem vozlišču.*

#### *7.2.4.1 Primer poplavljanja*

Slika 7-3 prikazuje podrobne podatke o poizvedbi, katere izvajanje si bomo pogledali korak za korakom. Slika 7-4 pa prikazuje zapis vseh prenosov te poizvedbe, kot so shranjeni v podatkovni bazi.

| ID_Q | IDV | IDP | T_START | T_ODG | NAJDENO_DN | TTL_ZACETNI | HOPS |
|---|---|---|---|---|---|---|---|
| 13374 | 7867 | 201 | 7 | 11 | D | 5 | 2 |

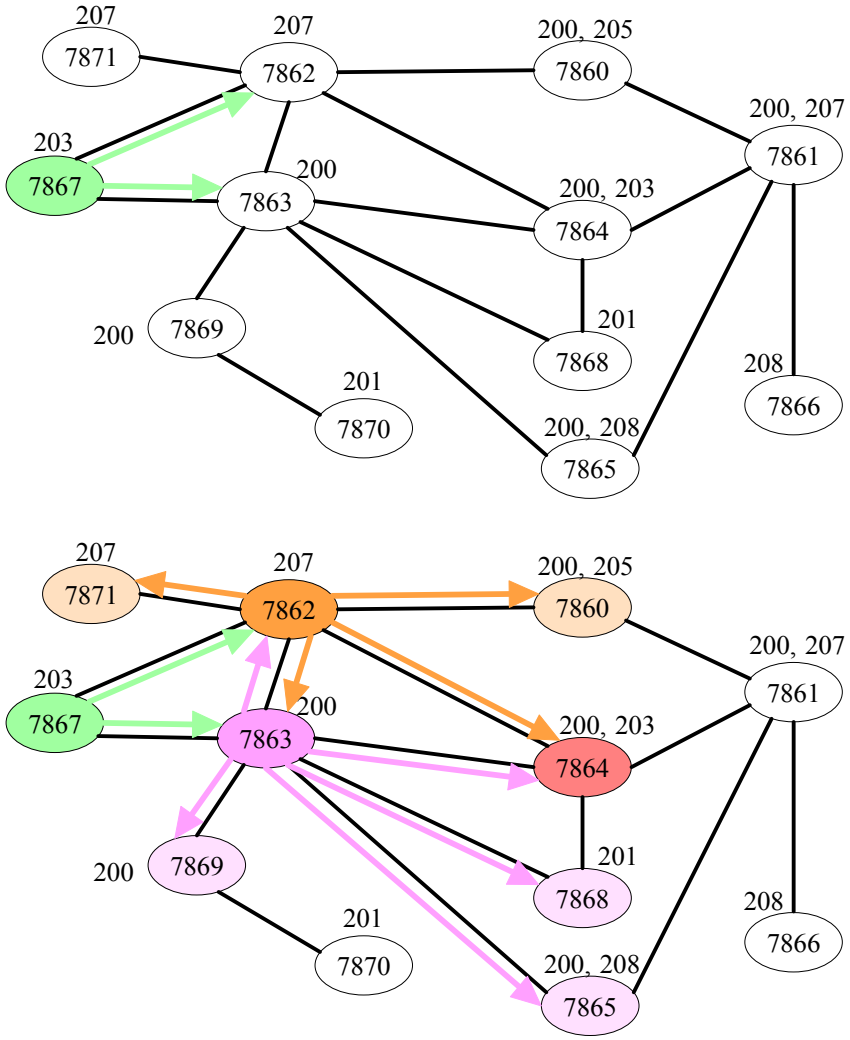
*Slika 7-3: Generirana poizvedba, ki jo je sprožilo vozlišče 7867, je poizvedovala po podatku 201, generirana je bila v času T=7, prvi odgovor je prejela v času T =11, do najbližjega odgovora sta bila potrebna 2 skoka, življenjska doba pa je bila 5 skokov.*

| ID_SKOK | ID_Q | IDV_OD | IDV_DO | T_OD | T_DO | TTL | NAJDENO_DN |
|---|---|---|---|---|---|---|---|
| 7312403 | 13374 | 7867 | 7862 | 7 | 8 | 4 | |
| 7312404 | 13374 | 7867 | 7863 | 7 | 8 | 4 | |
| 7312415 | 13374 | 7862 | 7871 | 8 | 9 | 3 | |
| 7312416 | 13374 | 7862 | 7864 | 8 | 9 | 3 | |
| 7312417 | 13374 | 7862 | 7863 | 8 | 9 | 3 | |
| 7312418 | 13374 | 7862 | 7860 | 8 | 9 | 3 | |
| 7312419 | 13374 | 7863 | 7869 | 8 | 9 | 3 | |
| 7312420 | 13374 | 7863 | 7862 | 8 | 9 | 3 | |
| 7312421 | 13374 | 7863 | 7864 | 8 | 9 | 3 | |
| 7312422 | 13374 | 7863 | 7865 | 8 | 9 | 3 | |
| 7312423 | 13374 | 7863 | 7868 | 8 | 9 | 3 | D |
| 7312425 | 13374 | 7864 | 7861 | 9 | 10 | 2 | |
| 7312426 | 13374 | 7864 | 7868 | 9 | 10 | 2 | D |
| 7312427 | 13374 | 7865 | 7861 | 9 | 10 | 2 | |
| 7312429 | 13374 | 7869 | 7870 | 9 | 10 | 2 | D |
| 7312430 | 13374 | 7860 | 7861 | 9 | 10 | 2 | |
| 7312436 | 13374 | 7861 | 7866 | 10 | 11 | 1 | |

*Slika 7-4: Zapis vseh prenosov izbrane poizvedbe (id_skok: identifikator skoka oziroma prenosa; id_q: identifikator poizvedbe; idv_od: identifikator vozlišča, ki je odposlalo poizvedbo; idv_do: identifikator vozlišča, ki je prejelo poizvedbo; t_od: čas začetka skoka; t_do: čas konca skoka; TTL: preostala življenjska doba; najdeno_DN: ali je bil najden odgovor pri prejemniku).*

Poizvedba je bila generirana ob času T=7 in vozlišče 7867 jo je takoj posredovalo obema svojima sosedoma (poplava), vozliščema 7862 in 7863. Ker noben od njiju ni našel pri

sebi vsebine 201, v naslednjem časovnem intervalu poizvedbo oba posredujeta naprej – poplavita vsem svojim sosedom, razen tistemu, od katerega sta poizvedbo sprejela. Slika 7-5 prikazuje zaporedje prenosov v prvih dveh korakih. Z zeleno barvo je označeno vozlišče, ki je izvor poizvedbe. Prav tako zeleno sta označena prenosa, ki predstavljata prvo poplavo in se zgodita na prvem koraku.

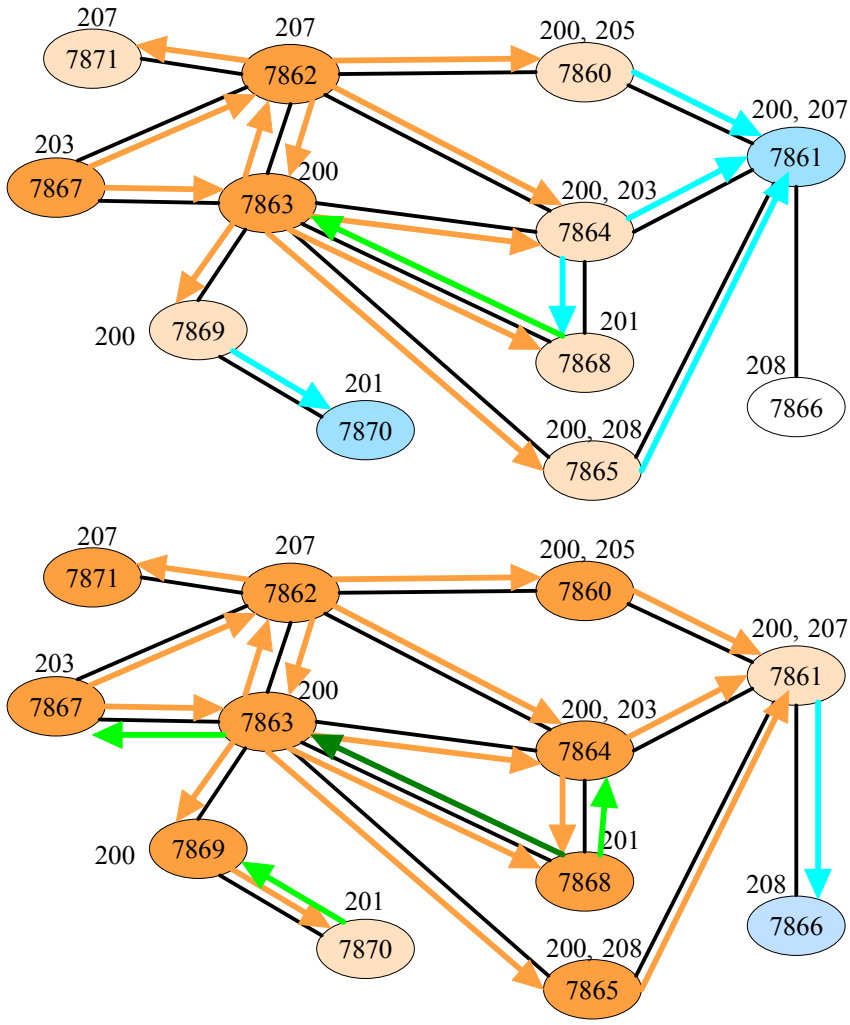



*Slika 7-5: Prva dva koraka prenosov poizvedb.V prvem koraku se zgodijo zeleno označeni prenosi, oranžni in vijolični pa v drugem koraku. Na vozlišču 7868 je bil že najden odgovor.*

Spodnji del iste slike prikazuje drugi korak, ko se vzporedno zgodita dve poplavi, od katerih vsako sproži eden od sosedov izvornega vozlišča. Prenosi enega so označeni z oranžno, drugega pa z vijolično barvo. Rdeče je obarvano vozlišče, ki je poizvedbo

prejelo od obeh. Opazimo lahko tudi, da je poizvedba dosegla vozlišče 7868, kjer se nahaja iskana vsebina, ter preverimo, da nobeno od vozlišč poizvedbe ni poslalo nazaj izvornemu vozlišču.

V tretjem koraku (Slika 7-6) preverimo, da vozlišče 7864, ki je poizvedbo v istem časovnem intervalu prejelo od dveh sosedov, le te ne posreduje nobenemu od njiju. Preverimo, da se posredovanje poizvedb zaključi tudi na vozlišču 7871, ki nima drugih sosedov kot tega, od katerega je prejelo poizvedbo. Prav tako poizvedbe ne posreduje več vozlišče 7868, ki je že generiralo odgovor. Zelena puščica predstavlja prenos odgovora nazaj po poti poizvedbe.




*Slika 7-6: Prenosi na tretjem (zgornji del slike) in četrtem koraku (spodnji del slike). Prejšnji prenosi so enotno označeni z oranžno barvo. Prenosi poizvedbe na zadnjem*

*koraku so modri, svetlo zelena puščica predstavlja prenos odgovora, temno zelena puščica pa prenos odgovora v prejšnjem koraku.*

Opazimo lahko, da le še vozlišče 7866 še ni prejelo poizvedbe, kar pa se bo zgodilo v naslednjem koraku, ki je prikazan na spodnjem delu iste slike. V četrtem koraku lahko preverjamo pravilnost prenosov odgovorov. Poizvedba v naslednjih časovnih intervalih več ne prenaša naprej, saj je dosegla že vsa vozlišča, čeprav ji življenjska doba še ni potekla. V tem časovnem intervalu je izvorno vozlišče že dosegel prvi odgovor, ki je bil oddaljen 2 skoka, in je povzročil zapis podatkov o najdbi odgovora (najdeno_DN), času odgovora (T_odg) in številu skokov (hops), ki jih prikazuje Slika 7-3.

Brez grafične predstavitve se prepričajmo le še v pravilnost zadnjih dveh prenosov drugega in tretjega odgovora: v petem koraku do vozlišča 7863 oziroma 7862, v šestem pa že do izvora – kar lahko razberemo v zadnjih dveh vrsticah, ki jih prikazuje Slika 7-7. Na cilj prideta oba v istem časovnem intervalu.

Za celotno izvedbo je bilo potrebnih 17 prenosov poizvedbe in 7 prenosov odgovorov.

| IDQ | IDV_NAJDENO | ID_ODG |
|---|---|---|
| 13374 | 7868 | 493178 |
| 13374 | 7868 | 493180 |
| 13374 | 7870 | 493181 |

| IDQ | IDV_OD | IDV_DO | ZAP_ST | ID_ODG | ID_SKOK_ODG | T_OD | T_DO |
|---|---|---|---|---|---|---|---|
| 13374 | 7868 | 7863 | 1 | 493178 | 1624934 | 9 | 10 |
| 13374 | 7863 | 7867 | 2 | 493178 | 1624941 | 10 | 11 |
| 13374 | 7868 | 7864 | 1 | 493180 | 1624942 | 10 | 11 |
| 13374 | 7870 | 7869 | 1 | 493181 | 1624943 | 10 | 11 |
| 13374 | 7864 | 7862 | 2 | 493180 | 1624945 | 11 | 12 |
| 13374 | 7869 | 7863 | 2 | 493181 | 1624951 | 11 | 12 |
| 13374 | 7862 | 7867 | 3 | 493180 | 1624952 | 12 | 13 |
| 13374 | 7863 | 7867 | 3 | 493181 | 1624953 | 12 | 13 |

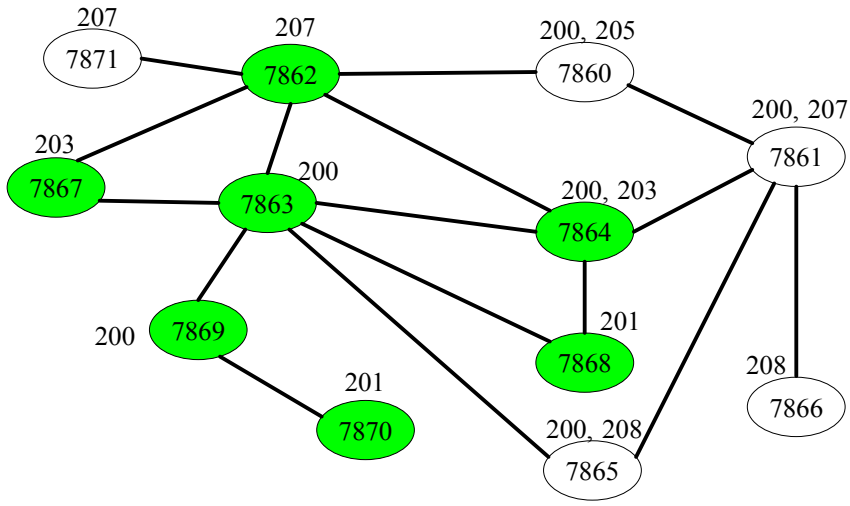
*Slika 7-7: Generirani odgovori zgoraj (idq: identifikator poizvedbe, na katero se nanaša odgovor; idv_najdeno: identifikator vozlišča, na katerem je bil najden odgovor; id_odg: identifikator odgovora) in njihovi prenosi na spodnjem izpisu(idq: identifikator poizvedbe; idv_od in idv_do: identifikatorja vozlišč - izvora in ponora; zap_st: zaporedna številka skoka posameznega odgovora; id_odg: identifikator odgovora;*

*id_skok_odg: identifikator skoka odgovora; T_od in T_do: začetni in končni čas prenosa).*

#### *7.2.4.2 Primer usmerjanja s pomnjenjem posredovanih odgovorov*

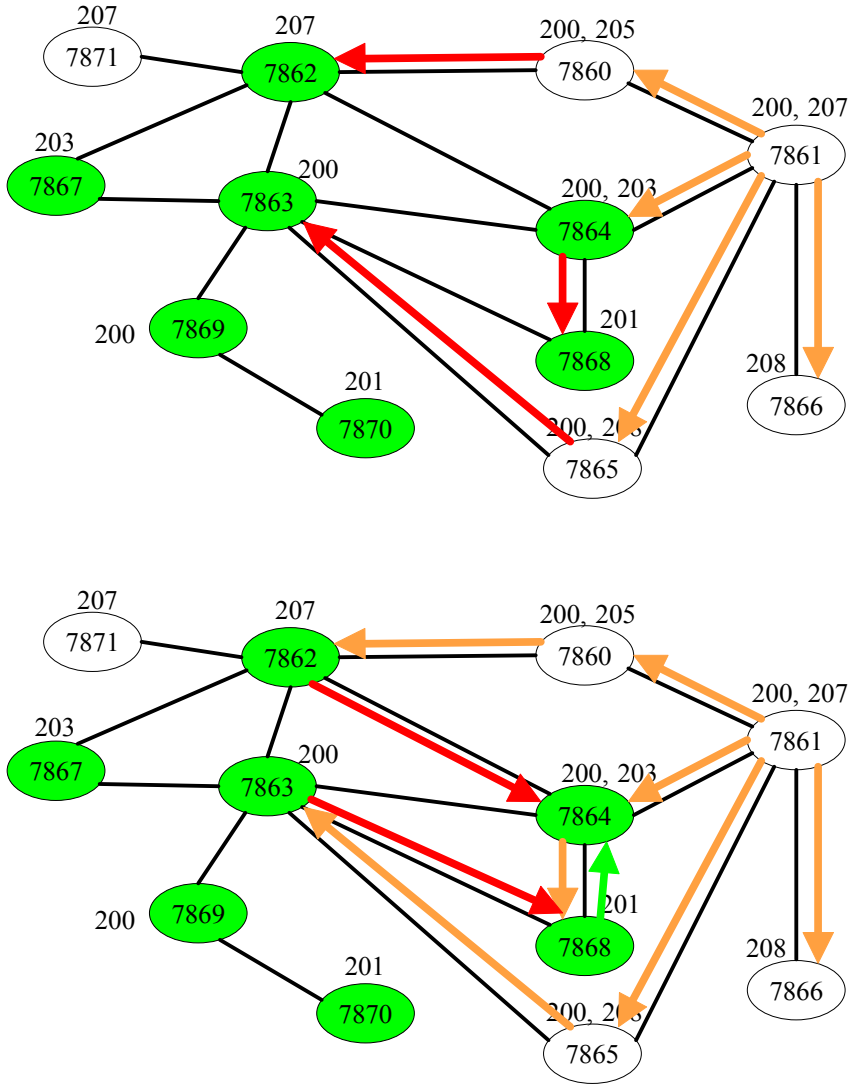
Ko se v omrežju prvič pojavi poizvedba po neki vsebini, je poplavljena na enak način, kot smo to prikazali v prejšnjem razdelku. Za vse naslednje poizvedbe pa nekaj vozlišč že zna usmerjati poizvedbe k želeni vsebini. V našem primeru so za vsebino 201 to vozlišča, prek katerih so potovala odgovora in jih prikazuje Slika 7-8.

V nadaljevanju bomo preverili pravilnost usmerjanja,poizvedbe, ki slej ko prej naleti na skonfigurirano vozlišče. Če poizvedbo po vsebini 201 generira vozlišče 7869, ki je že skonfigurirano (leži na poti, po kateri je potoval odgovor), gre poizvedba neposredno na vozlišče 7870, kjer se nahaja odgovor. Odgovor se vrne po isti – direktni poti. Potreben je torej en prenos poizvedbe in en prenos odgovora.




*Slika 7-8: V primeru pomnjenja so po začetni poplavi, enaki kot prej, za usmerjanje do vsebine 201 skonfigurirana zeleno obarvana vozlišča.*

Nato sprožimo poizvedbo po vsebini 201 na vozlišču 7861. To vozlišče še ni skonfigurirano, zato se bo poizvedba poplavila. Po prvem koraku je že dosegla prvo vozlišče, ki je že prej posredovalo odgovor – vozlišče 7864. To vozlišče poizvedbo usmeri neposredno do iskane vsebine na vozlišče 7868, medtem ko jo drugi dve vozlišči poplavita. Slika 7-9 opisano dogajanje predstavi še vizualno.




*Slika 7-9: Prenos poizvedbe po vsebini 201 v delno skonfiguriranem omrežju. Na zgornjem delu so z oranžno barvo označeni prenosi poizvedbe na prvem koraku, z rdečo pa na drugem koraku. Skonfigurirana vozlišča so zelena. Na spodnjem delu so z oranžno barvo označeni prenosi prvega in drugega koraka, z rdečo pa tretjega.*

S tem se prenosi poizvedbe končajo, v naslednjih korakih se do izvora le še preneseta generirana odgovora. Zapis vseh prenosov opisane poizvedbe in odgovorov, kot je zajet v podatkovni bazi, predstavljajo še Slika 7-10, Slika 7-11 in Slika 7-12.

Na predstavljenem primeru preverimo različna protokolarna pravil, na primer:

- Vozlišče, ki poizvedbo prejme vdrugo (potem, ko jo je že posredovalo naprej), je ne posreduje še enkrat (primer vozlišča 7864).

- Vozlišče, ki je kdaj prej že posredovalo odgovor z ustrezno vsebino, poizvedbe ne poplavi, ampak usmeri proti tej vsebini (primer vozlišča 7862, 7864).

| ID_SKOK | ID_Q | IDV_OD | IDV_DO | T_OD | T_DO | TTL | NAJDENO_DN |
|---|---|---|---|---|---|---|---|
| 7312562 | 13388 | 7861 | 7860 | 21 | 22 | 4 | |
| 7312563 | 13388 | 7861 | 7864 | 21 | 22 | 4 | |
| 7312564 | 13388 | 7861 | 7865 | 21 | 22 | 4 | |
| 7312565 | 13388 | 7861 | 7866 | 21 | 22 | 4 | |
| 7312568 | 13388 | 7864 | 7868 | 22 | 23 | 3 | D |
| 7312569 | 13388 | 7865 | 7863 | 22 | 23 | 3 | |
| 7312570 | 13388 | 7860 | 7862 | 22 | 23 | 3 | |
| 7312573 | 13388 | 7862 | 7864 | 23 | 24 | 2 | |
| 7312574 | 13388 | 7863 | 7868 | 23 | 24 | 2 | D |

*Slika 7-10: Vsi prenosi poizvedbe po vsebini 201, ki jo generira vozlišče 7861.*

| IDQ | IDV_NAJDENO | ID_ODG |
|---|---|---|
| 13388 | 7868 | 493205 |
| 13388 | 7868 | 493206 |

*Slika 7-11: Vsi generirani odgovori na poizvedbo po vsebini 201, ki jo generira vozlišče 7861.*

| IDQ | IDV_OD | IDV_DO | ZAP_ST | ID_ODG | ID_SKOK_ODG | T_OD | T_DO |
|---|---|---|---|---|---|---|---|
| 13388 | 7868 | 7864 | 1 | 493205 | 1625001 | 23 | 24 |
| 13388 | 7864 | 7861 | 2 | 493205 | 1625002 | 24 | 25 |
| 13388 | 7868 | 7863 | 1 | 493206 | 1625003 | 24 | 25 |
| 13388 | 7863 | 7865 | 2 | 493206 | 1625004 | 25 | 26 |
| 13388 | 7865 | 7861 | 3 | 493206 | 1625008 | 26 | 27 |

*Slika 7-12: Vsi prenosi odgovorov za poizvedbo po vsebini 201, ki jo generira vozlišče 7861.*

Navedimo še, da naslednja poizvedba po isti vsebini, generirana na vozlišču 7871, zahteva le tri prenose poizvedbe, saj jo lahko vsa vmesna vozlišča pravilno usmerijo.

### *7.2.4.3 Primer usmerjanja z izmenjavo metapodatkov*

Samo usmerjanje pri načinu z izmenjavo metapodatkov poteka zelo podobno kot v prejšnjem primeru, zato tu lahko grafično predstavitev opustimo. Ogledali si bomo le, kako poteka gradnja skladišča usmerjevalnih metapodatkov in njihova izmenjava.

Slika 7-13 prikazuje dva izseka usmerjevalnih metapodatkov. V našem primeru so si jih vozlišča izmenjala vsakih 5 časovnih intervalov. Preverimo lahko vstavljanje metapodatkov kot posledico prenesenih odgovorov in kot posledico izmenjave.

Prve tri vrstice (tiste, ki imajo atribut *cas_vstavljeno* manjši kot 5) v zgornji tabeli predstavljajo metapodatke, ki so bili zapisani kot posledica prenosov odgovorov – torej podatke, ki bi nam bili znani tudi, če bi šlo za usmerjanje s pomnjenjem posredovanih odgovorov, brez izmenjave metapodatkov. Slika 7-14 prikazuje prenose odgovorov, ki so povzročili te zapise. V nadaljevanju pa vidimo usmerjevalne metapodatke, ki so bili zapisani kot posledica izmenjave. Vsi imajo atribut *cas_vstavljeno* enak 5, ker so bili vstavljeni po prvi izmenjavi. V spodnjem delu iste slike pa vidimo še del zapisov, ki so nastali kot posledica kasnejših izmenjav (atribut *cas_vstavljeno* je enak 10 oziroma 15).

| IDV | IDP | OCENA_SKOKOV | IDV_SOSED | CAS_VSTAVLJENO |
|---|---|---|---|---|
| 7863 | 200 | 1 | 7869 | 3 |
| 7862 | 200 | 1 | 7864 | 4 |
| 7868 | 200 | 1 | 7863 | 4 |
| 7863 | 207 | 1 | 7862 | 5 |
| 7864 | 207 | 1 | 7862 | 5 |
| 7862 | 203 | 1 | 7864 | 5 |
| 7865 | 200 | 1 | 7863 | 5 |
| 7863 | 208 | 1 | 7865 | 5 |
| 7866 | 200 | 1 | 7861 | 5 |
| 7866 | 207 | 1 | 7861 | 5 |

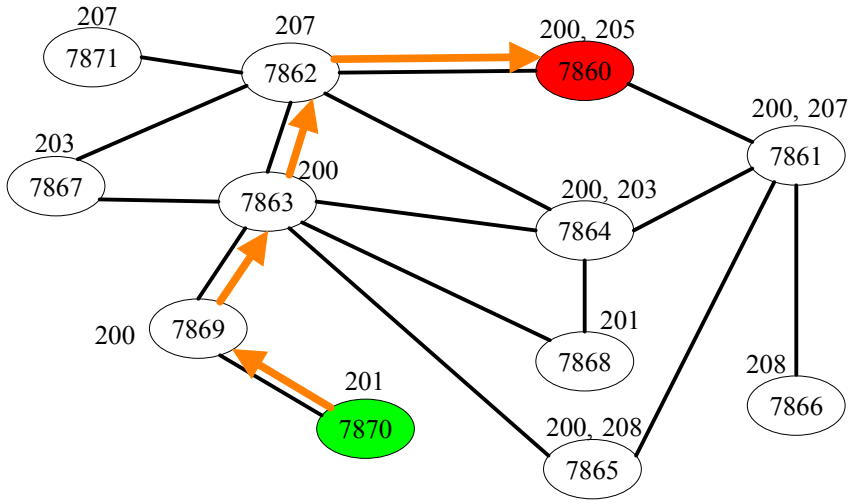
| IDV | IDP | OCENA_SKOKOV | IDV_SOSED | CAS_VSTAVLJENO |
|---|---|---|---|---|
| 7869 | 205 | 3 | 7863 | 10 |
| 7870 | 207 | 3 | 7869 | 10 |
| 7870 | 208 | 3 | 7869 | 10 |
| 7870 | 203 | 3 | 7869 | 10 |
| 7871 | 208 | 3 | 7862 | 10 |
| 7871 | 201 | 3 | 7862 | 10 |
| 7870 | 205 | 4 | 7869 | 15 |

*Slika 7-13: Del usmerjevalnih metapodatkov (idv: vozlišče, ki usmerja; idp: vsebina, do katere usmerja; ocena_skokov: predvideno število skokov do vsebine; idv_sosed: sosed, kateremu posredujemo poizvedbo; čas_vstavljeno: čas generiranja zapisa).*

| IDQ | IDV_OD | IDV_DO | ZAP_ST | ID_ODG | T_OD | T_DO | ID_SKOK_ODG |
|---|---|---|---|---|---|---|---|
| 13674 | 7869 | 7863 | 1 | 493490 | 2 | 3 | 1625449 |
| 13674 | 7864 | 7862 | 1 | 493493 | 3 | 4 | 1625452 |
| 13674 | 7868 | 7863 | 2 | 493494 | 4 | 5 | 1625457 |

*Slika 7-14: Prenosi odgovorov, ki so imeli za posledico prve tri zapise usmerjevalnih metapodatkov.*

V našem malem omrežju so zdaj skonfigurirana že vsa vozlišča. Najdaljša pot v grafu je tista, ki opredeljuje, kolikšno mora biti število izmenjav usmerjevalnih metapodatkov, da so dosežena vsa vozlišča. V naši topologiji je najdaljša pot od vozlišča 7870 do vozlišča, kjer se nahaja vsebina 205 – to sklepamo iz zadnje vrstice usmerjevalnih metapodatkov in nato še vizualno preverimo na shemi topologije (Slika 7-15). Ugotovimo, da se vsebina 205 nahaja le na vozlišču 7860 in da do tja res ne moremo po krajši poti kot v 4 skokih.




*Slika 7-15: Najkrajša pot od vozlišča 7870 do vsebine 205.*

## 7.3 Metrike za vrednotenje rezultatov

Večje simulacije generirajo med svojim izvajanjem zelo veliko podatkov, ki so za analizo precej nepregledni, pa tudi redundantni. Zato smo si že vnaprej izbrali množico metrik oziroma kriterijev za medsebojno primerjavo rezultatov simulacij. Po zaključku posamezne simulacije smo zagnali analizo rezultatov in nato shranili le rezultat analize,

nekakšen izvleček rezultatov simulacije. Čeprav se metrike med seboj precej prekrivajo, smo raje shranjevali več podatkov kot premalo, v nadaljevanju pa bomo v poglavju 8 predstavili le bolj zanimive rezultate in primerjave.

Metrike po vsebini lahko delimo v dve skupini:

- Z **uporabniškega** vidika nas zanimajo metrike, ki opisujejo kakovost storitve, ki jo nudi sistem. Če sedimo za računalnikom, vključenim v sistem enak z enakim, na mestu uporabnika, nas zanima predvsem ali bomo dobili odgovor na poizvedbo in če ga bomo dobili, kako hitro se bo to zgodilo. Metriki sta torej verjetnost uspeha in povprečna zakasnitev, izražena v številu skokov.
- Z vidika **obremenjenosti sistema** nas zanimajo metrike, ki opisujejo učinkovitost sistema: breme, njegovo razporeditev in porazdelitev v sistemu, količina režije in redundance, občutljive točke omrežja (najbolj obremenjena vozlišča) in podobno.

### *7.3.1 Osnovne metrike*

Osnovne metrike temeljijo na skupnem številu prenosov sporočil. Skupno število prenosov sporočil je zelo preprosta mera, ki pa nam kljub temu veliko pove o obremenitvi omrežja kot celote. Podrobneje lahko skupno število prenosov sporočil razdelimo na prenose poizvedb, prenose odgovorov in prenose metapodatkov, potrebnih za usmerjanje. Ostalih protokolarnih sporočil, ki so lahko potrebna za vzdrževanje topologije ali za povezovanje novih vozlišč v omrežje in podobno, v naši simulaciji nismo upoštevali, saj njihova količina ni odvisna od načina usmerjanja in bi bila v primerljivih omrežjih enaka. Njihova količina v današnjih sistemih enak z enakim je v primerjavi s količino prenosov poizvedb tudi bistveno (za več kot red velikosti) manjša.

Pri vrednotenju skupnega števila prenosov sporočil je nujno potrebno upoštevati tudi velikost omrežja in število povezav, saj lahko šele tako ugotovimo, ali ugotovljeno število prenosov pomeni za omrežje nevarnost zasičenja in prepolnitve morebitnih ozkih grl ali ne.

V to skupino zato sodijo tudi naslednje metrike:

- Število generiranih poizvedb - $P$.

- Število najdenih (generiranih) odgovorov - *O*. Odgovor se generira vsakokrat, ko poizvedba doseže vozlišče, kjer se nahaja vsebina, ki ustreza kriteriju poizvedbe.
- Število vseh skokov poizvedb - *SP*. Prenosi poizvedb predstavljajo pri poplavljanju in njegovih izvedenkah največjo obremenitev za omrežje.
- Število vseh skokov odgovorov - *SO*. Prenosi odgovorov navadno ne predstavljajo šibke točke protokola – le pri zelo učinkovitem usmerjanju poizvedb je njihovo število blizu ranga prenosov poizvedb.
- Povprečno število prenosov (skokov) za poizvedbo – *PSP* :

$$PSP = \frac{SP}{P} \tag{7.1}$$

- Povprečno število prenosov (skokov) za odgovor – *PSO* :

$$PSO = \frac{SO}{O} \tag{7.2}$$

  Tu naj opozorimo, da ena poizvedba lahko na različnih vozliščih najde večje število odgovorov. Nekateri so zelo blizu izvornega vozlišča, drugi pa oddaljeni več skokov, največ toliko, kolikor je življenjska doba poizvedbe ($TTL^2$). Povprečno število skokov za odgovor tako lahko niha le med 1 in TTL, v naših simulacijah 5 skokov. Za izvorno vozlišče pa je morda bolj zanimiva naslednja metrika, ki pove, koliko skokov je oddaljen najbližji odgovor.
- Povprečno število skokov (oddaljenost) do odgovora za uspešno poizvedbo – *PDO.* Ta parameter upošteva le uspešne poizvedbe, torej le tiste, ki so prejele vsaj en odgovor. Predstavlja pa povprečno število skokov, ki jih je opravil prvi sprejeti odgovor.
- Število uspešnih poizvedb – *PU* : Število poizvedb, ki so prejele vsaj en odgovor.

[2] Ponovno naj opozorimo, da tu ne gre za življenjsko dobo paketa - TTL, kot jo poznamo na omrežni plasti npr. v protokolu IP, ampak za podoben koncept na aplikacijski plasti, kjer življenjsko dobo sporočila merimo v številu skokov oz. prenosov med vozlišči navideznega omrežja.

- Število neuspešnih poizvedb – *PN* : Število poizvedb, ki niso prejele nobenega odgovora. Ker po izbranemu številu iteracij simulacijo "nasilno" prekinemo, zadnjih nekaj generiranih poizvedb vedno ostane brez odgovora.

Zanima nas ponovljivost poizvedb, zato opazujemo tudi največje število poizvedb, ki so iskale isti podatek, in povprečno število poizvedb po nekem podatku.

### 7.3.2 Izvedene metrike

V to skupino metrik smo razvrstili tiste, ki upoštevajo tudi naravo in namen obravnavanih sistemov.

#### *7.3.2.1 Cena poizvedbe*

V popolnoma nestrukturiranih sistemih se moramo sprijazniti z dejstvom, da je poizvedba toliko bolj uspešna, kolikor več vozlišč obišče. Vendar popolnoma zadošča, če poizvedba vsako vozlišče obišče le enkrat, saj naslednje prihode iste poizvedbe vozlišča preprosto ignorirajo. Vsak naslednji prihod poizvedbe lahko obravnavamo kot redundanco, ki je po nepotrebnem povečala celotno število prenosov. Portmann in Seneviratne v [51] po podobnem razmisleku definirata ceno poizvedbe *j* kot razmerje med številom skokov poizvedbe in doseženim številom vozlišč.

$$c_j = \frac{1}{r}\sum_{i=1}^{N} m_i \ . \tag{7.3}$$

V enačbi *r* predstavlja število vozlišč, ki jih je dosegla poplava, *N* je število vseh vozlišč v omrežju, $m_i$ pa je število sporočil, ki jih je posredovalo vozlišče *i*. Predpostavlja se, da je življenjska doba poizvedbe dovolj visoka, da poizvedba doseže vsa vozlišča.

Če se poizvedba posreduje samo po povezavah, ki tvorijo vpeto drevo na dani topologiji, potem vsako vozlišče prejme poizvedbo samo enkrat. V takem primeru je cena poizvedbe enaka 1. Višja cena pomeni, da je posamezno vozlišče poizvedbo prejelo v povprečju več kot enkrat.

V naši analizi definicijo cene poizvedbe prilagodimo na splošnejši model, kjer je življenjska doba sporočila poljubna. Tako se pogosto zgodi, da poizvedba ne doseže vseh

vozlišč v topologiji, Zato namesto $N$ – število vseh vozlišč – upoštevamo število vozlišč, ki jih poizvedba z izbrano življenjsko dobo lahko doseže z dovoljenim številom skokov.

V analizi rezultatov simulacije opazujemo naslednje vrednosti:

- Število vozlišč, ki jih je dosegla posamezna poizvedba – $r$.
- Cena posamezne poizvedbe - $c$.
- Število vseh poizvedb, ki jih posreduje posamezno vozlišče – $m$.

Za vse tri družine opazovanih vrednosti shranjujemo največjo vrednost preko vseh poizvedb oziroma vozlišč, prav tako najmanjšo vrednost, povprečje in mediano, ter še r 25. in 75. percentil.

#### *7.3.2.2 Učinkovitost in redundanca*

Učinkovitost in redundanca sta med seboj nasprotni si lastnosti usmerjanja. Če je usmerjanje učinkovitejše, je v sistemu manj redundance, velja pa seveda tudi nasprotno.

Redundanco merimo tako, da izračunamo odstotek nepotrebnih sporočil. Nepotrebna so tista, ki (kot smo omenili že v razdelku 7.3.2.1), dosežejo vozlišča, ki so poizvedbo že dobila po kaki drugi poti. Taka sporočila po nepotrebnem porabljajo sistemske vire, točneje prenosne zmogljivosti in konec koncev tudi nekaj procesorskega časa na pošiljatelju in prejemniku, pri tem pa ni od njih prav nobene koristi, saj jih prejemnik ignorira oziroma zavrže. Lv in soavtorji v [57] predlagajo, da redundanco izračunamo kot delež razlike celotnega števila prenosov izbrane poizvedbe $j$ in števila vozlišč, ki jih je ta poizvedba dosegla, v celotnem številu prenosov:

$$p_j = \frac{n_j - r_j}{n_j} \ , \tag{7.4}$$

kjer je $n_j$ skupno število prenosov poizvedbe $j$, $r_j$ pa število doseženih vozlišč. Če na primer $p$ zavzame vrednost 0.54, to pomeni, da je redundantnih 54 odstotkov prenosov sporočil.

Kot učinkovite prenose poizvedbe pa obravnavamo tiste, ki so dosegli vozlišče, kjer se nahaja rezultat in kjer se bo zato lahko generiral odgovor. Spremljamo lahko število učinkovitih prenosov za posamezno poizvedbo, ali pa podobno kot prej izračunamo odstotek učinkovitih prenosov:

$$d_j = \frac{n'_j}{n_j} \ , \tag{7.5}$$

kjer $n_j$ tako kot v prejšnji enačbi pomeni skupno število prenosov poizvedbe $j$, $n'_j$ pa število učinkovitih prenosov poizvedbe.

Metriki sta torej

- Redundantnost – $p$ in
- učinkovitost – $d$,

obe pa se nanašata na posamezno poizvedbo.

### 7.3.3 Najpomembnejše metrike

Čeprav smo v prejšnjih dveh razdelkih našteli kopico metrik, ki jih sledimo med izvajanjem simulacije, naj opozorimo, da se te med seboj močno prekrivajo in pravzaprav prikazujejo le malo različne poglede na uspešnost usmerjanja. Zato bomo največ pozornosti posvetili naslednjima:

- Število prenosov posamezne poizvedbe oziroma povprečno ali pa kumulativno število prenosov poizvedb v sistemu kot metrika za obremenitev sistema.
- Število uspešnih poizvedb in oddaljenost odgovora kot metrika za zadovoljstvo uporabnika.

# 8 REZULTATI

Ugotovili smo, da imajo navidezna omrežja razširjenih sistemov enak z enakim topološke lastnosti majhnega sveta in potenčnih zakonov, ki jih ustrezno posnema tudi umetno generirana topologija po metodi GLP.

Za vsako simulacijo smo generirali zaporedje večjega števila iteracij. V vsaki iteraciji smo najprej generirali novo poizvedbo, ki jo sproži naključno izbrano vozlišče, in ki poizveduje po vsebini, izbrani v skladu s porazdelitvijo popularnosti. Nato smo vse poizvedbe posredovali en skok naprej, prav tako vse obstoječe odgovore, generirali nove najdene odgovore in zaključili vse poizvedbe, ki jim je potekla življenjska doba. V primeru porazdeljenega usmerjanja smo v določenih časovnih intervalih še simulirali izmenjavo metapodatkov, torej ustrezno prilagodili usmerjanje.

V tem poglavju bomo prikazali predvsem tiste rezultate simulacij, ki so pomembnejši, značilnejši in zanimivejši za primerjavo posameznih načinov usmerjanja v različnih okoliščinah. Nekaj podrobnejših primerjav in numeričnih rezultatov pa lahko najdemo tudi v Prilogi.

Za primerjavo posameznih pogledov na rezultate simulacij bomo testirali več različnih sistemov, dve skrajnosti pa sta:

1. Sistem, kjer pričakujemo največ prenosov: to je sistem brez replikacije vsebin, z uniformno popularnostjo vsebin in tudi z uniformno frekvenco poizvedovanja. Tu od predlaganih izboljšav usmerjanja ne pričakujemo velikih koristi.
2. Sistem, ki je najbližje realnim nestrukturiranim sistemom enak z enakim, kakršne preučuje disertacija. To je sistem z nizko stopnjo replikacije vsebin, z Zipfovo porazdelitvijo vsebin in Zipfovo porazdelitvijo pogostosti povpraševanja.

Večino simulacij bomo izvedli na topologiji tipa GLP, saj ta predstavlja najboljši približek realnih topologij vsebinskih omrežij. Uporabljali bomo veliko in malo

topologijo; velika ima okrog 1500, mala pa okrog 150 vozlišč. Za primerjavo bomo nazadnje izvedli še nekaj simulacij na topologijah drugih tipov.

## 8.1 Usmerjanje na topologiji GLP

Topologija GLP je za nas najbolj zanimiva, saj simulacije, izvedene na njej, predstavljajo najboljši približek obnašanja protokolov v realnih sistemih. Tabela 8.1 podrobneje prikazuje lastnosti sistema, ki je služil kot osnova za simulacije.

| Lastnost | Velika topologija | Mala topologija |
|---|---|---|
| Št. vseh vsebin | 500 | 50 |
| Št. vsebin na enem vozlišču | 1 do 3 | 1 do 3 |
| Porazdelitev vsebin | Zipfova | Zipfova |
| Porazdelitev poizvedb (zaželenost vsebin) | Zipfova | Zipfova |
| Min. in max. $r_i$ (repliciranost) | 0 (0.00065) in 0.071 | 0 (0.0065) in 0.1 |
| Min. in max. $q_i$ (zaželenost) | 0.0000025 in 0.0044 | 0.00081 in 0.039 |

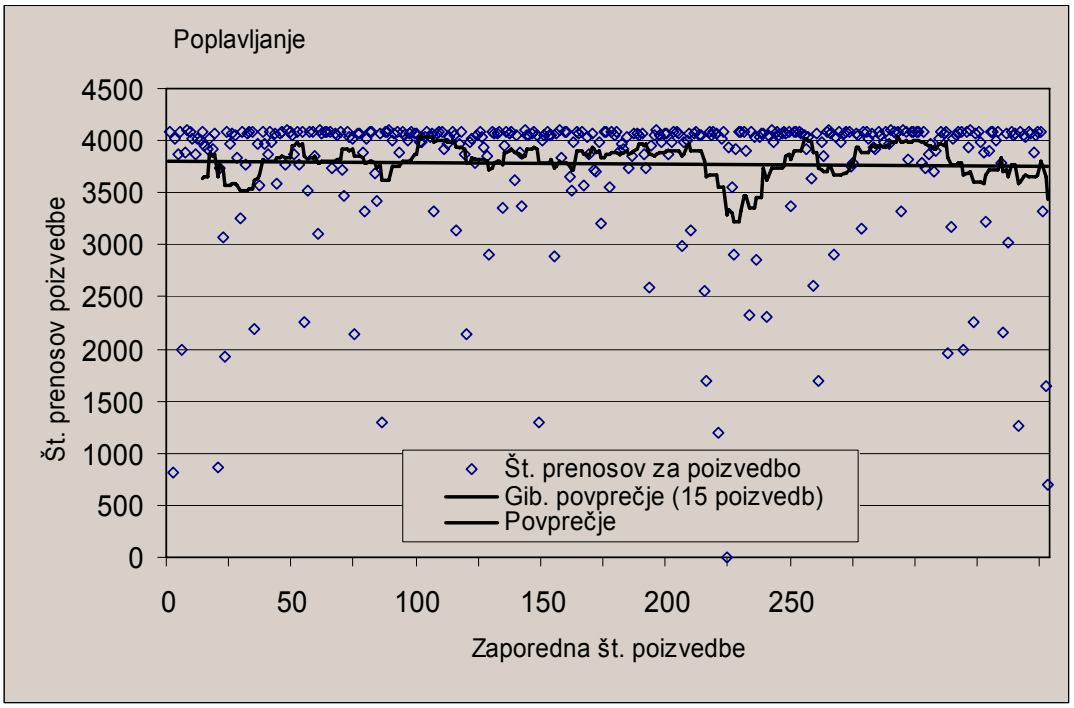
*Tabela 8.1: Lastnosti sistemov GLP, na katerih smo izvajali simulacije.*

Sistem je bil modeliran dinamično, kar pomeni, da smo upoštevali, da vozlišča občasno tudi izstopajo iz sistema; generirali smo povprečno en odhod vozlišča na 10 poizvedb v omrežju.

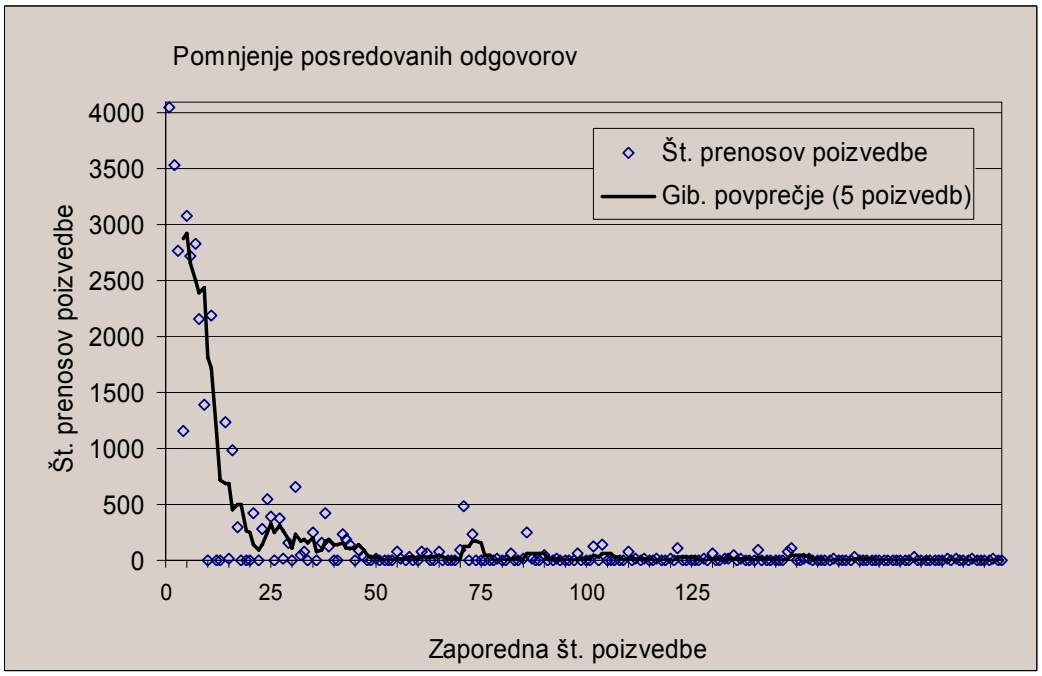
### *8.1.1 Poizvedbe po eni vsebini*

Za začetek si oglejmo, kako se sistem konfigurira, če vse poizvedbe poizvedujejo po isti vsebini. Izbrali smo najbolj replicirano vsebino, ki jo lahko najdemo na 18 vozliščih.

Graf 8.1 prikazuje osnovno različico usmerjanja - poplavljanje. Ne glede na to, koliko enakih poizvedb je sistem prenesel pred tem, največje število poizvedb zahteva okrog 4000 prenosov.




*Graf 8.1: Poplavljanje: število prenosov v zaporedju poizvedb, ki sprašujejo po isti vsebini.*

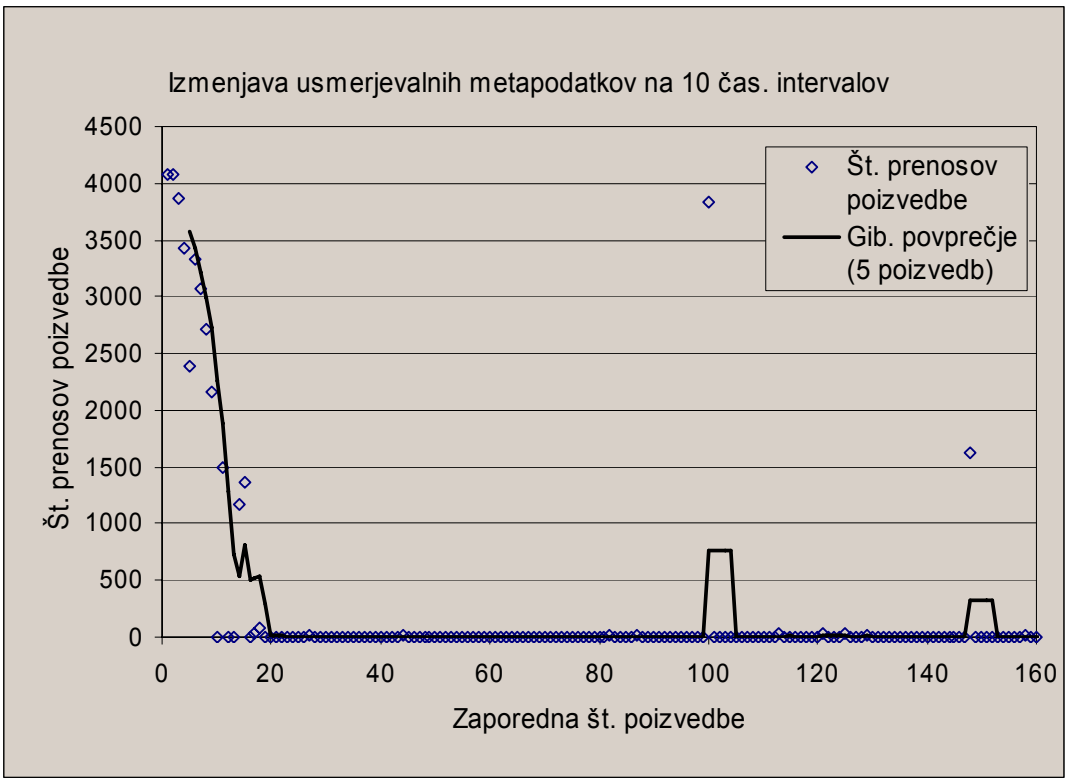



*Graf 8.2: Usmerjanje s pomnjenjem posredovanih odgovorov: število prenosov v zaporedju poizvedb, ki sprašujejo po isti vsebini.*

Graf 8.2 pa prikazuje konfiguriranje sistema pri uporabi usmerjanja s pomnjenjem posredovanih odgovorov.

Začetno število prenosov je enako kot pri poplavljanju, vendar hitro pade: po 20 poizvedbah je že za velikostni razred nižje, v rangu življenjske dobe poizvedbe pa se ustali po približno 50 poizvedbah. Kasnejši skoki in večja nihanja v številu prenosov se zgodijo zato, ker se zaradi dinamičnosti omrežja in odhajanja vozlišč pretrgajo že skonfigurirane poti in so zato potrebne vsaj delne poplave, da se najde alternativno pot ali lokacijo iskane vsebine.

Pri usmerjanju z izmenjavo metapodatkov (prikazuje ga Graf 8.3) je na prvi pogled slika zelo podobna usmerjanju s pomnjenjem posredovanih odgovorov. Opozorimo naj na bistvene razlike.




*Graf 8.3: Izmenjava usmerjevalnih metapodatkov: število prenosov v zaporedju poizvedb, ki sprašujejo po najbolj replicirani vsebini v sistemu.*

Zaradi izmenjave metapodatkov je konfiguriranje malo hitrejše, kar lahko razberemo zlasti na odseku po 20. poizvedbi. Z vsako izmenjavo vozlišče dobi informacijo o tem,

kako usmerjati poizvedbe do vozlišč, ki so za en skok bolj oddaljena od teh, do katerih je že znalo usmerjati. Obenem se vozlišča konfigurirajo tudi ob tem, ko posredujejo odgovore (enako kot v prejšnji različici). Ker je povprečna najkrajša pot v naši topologiji dolga 3.95 skoka, učinkujeta dve zaporedni izmenjavi metapodatkov skupaj z vmesnim konfiguriranjem zaradi posredovanih odgovorov že kot popolna konfiguriranost omrežja za usmerjanje do izbrane vsebine.

Ker so skonfigurirana vsa vozlišča, ne le tista, ki so posredovala odgovore, sistem pozna več alternativnih poti za usmerjanje in zato dinamičnost omrežja (odhajanje vozlišč in pretrgane poti) dolgo ne povzroči večjih skokov v številu prenosov. Prvi dvig števila prenosov opazimo šele po 100 poizvedbah, ko se je pretrgalo že večje število poti. Vendar se pa se te hitro spet vzpostavljajo in povprečje je se obdrži v velikostnem razredu življenjske dobe poizvedbe.

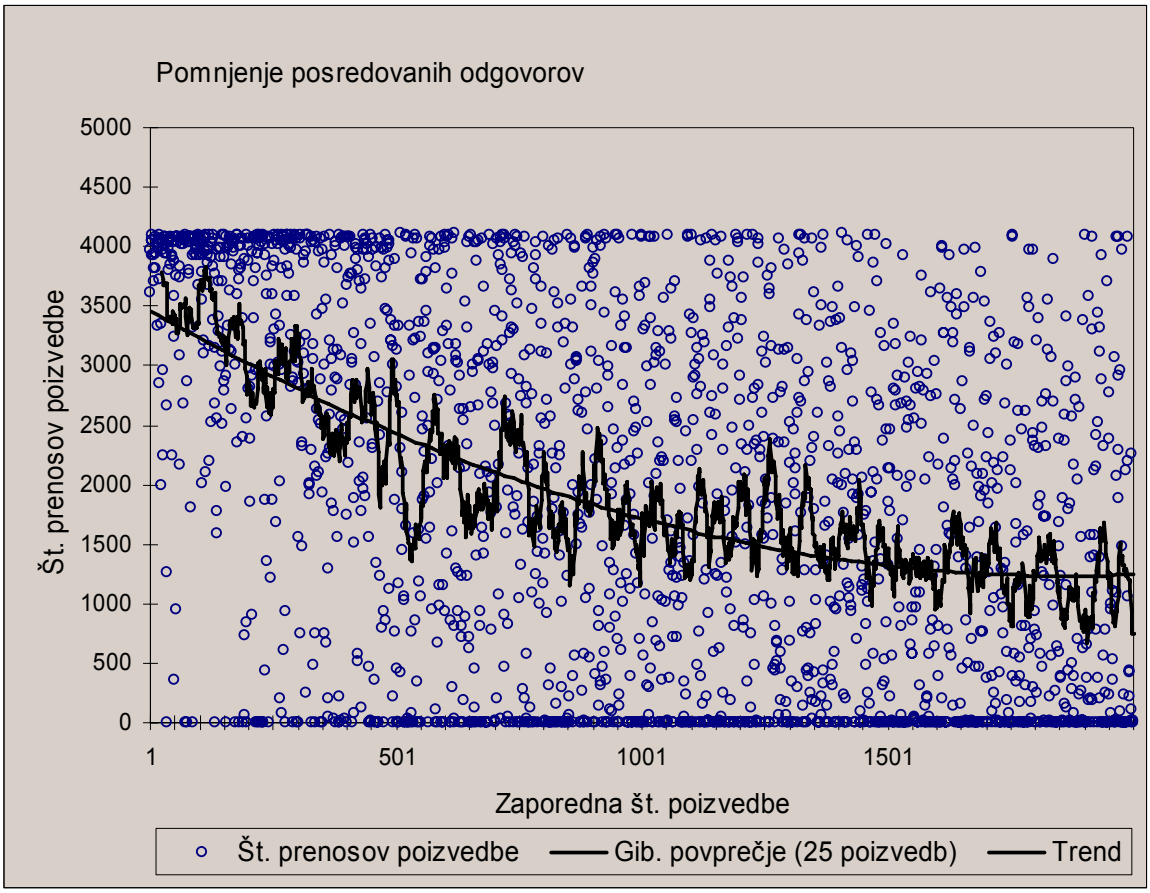
Ne smemo pozabiti, da tudi sama izmenjava metapodatkov povzroči veliko prenosov sporočil in tako predstavlja konstantno obremenitev sistema. Na predstavljenem omrežju, kjer je povprečna stopnja vozlišč 3.68, ob izmenjavi metapodatkov na vsakih 10 generiranih poizvedb to pomeni več kot 500 dodatnih prenosov na poizvedbo. Če sistem opazujemo skozi prizmo poizvedb po isti vsebini, je tak način usmerjanja videti preveč potraten. V naslednjem razdelku pa bomo pokazali, da v velikih sistemih z mešanico različnih poizvedb dodatno breme izmenjave metapodatkov glede na celotno količino prenosov poizvedb ne pomeni tako velike obremenitve.

Ostane nam še odgovor na vprašanje, kako se konfigurira sistem za manj popularne vsebine. Po naših predpostavkah so manj popularne vsebine tudi manj replicirane, ker je po njih poizvedovalo manj vozlišč, kar je imelo za posledico manj prenosov vsebine na nova vozlišča. Osnovna oblika krivulje je enaka, le da je bolj položna – počasneje doseže število prenosov v velikostnem razredu življenjske dobe poizvedbe. Povprečna dolžina najkrajše poti do vozlišča s tako vsebino je daljša kot pri vsebinah z višjo stopnjo replikacije in tudi globina, ki jo doseže poizvedba po manj replicirani vsebini, je večja. Zaradi večje globine tudi delne poplave povzročijo več prenosov poizvedbe. Primerjavo krivulj konfiguriranja za najbolj in najmanj popularno vsebino lahko najdemo v Prilogi (Graf 9.1).

### 8.1.2 Mešanica poizvedb po različnih vsebinah

Zdaj pa si na značilnem primeru omrežja oglejmo še, kako se obnaša sistem, ki doživi tipično mešanico poizvedb po različnih vsebinah. Poizvedbe po bolj zaželenih vsebinah se kmalu ponovijo, po manj zaželenih pa so večkrat edinstvene - v skladu z Zipfovo porazdelitvijo. Skupna krivulja konfiguriranja je raztegnjena v smeri osi *x*, kot smo pričakovali in komentirali že v poglavju 5.5.2.

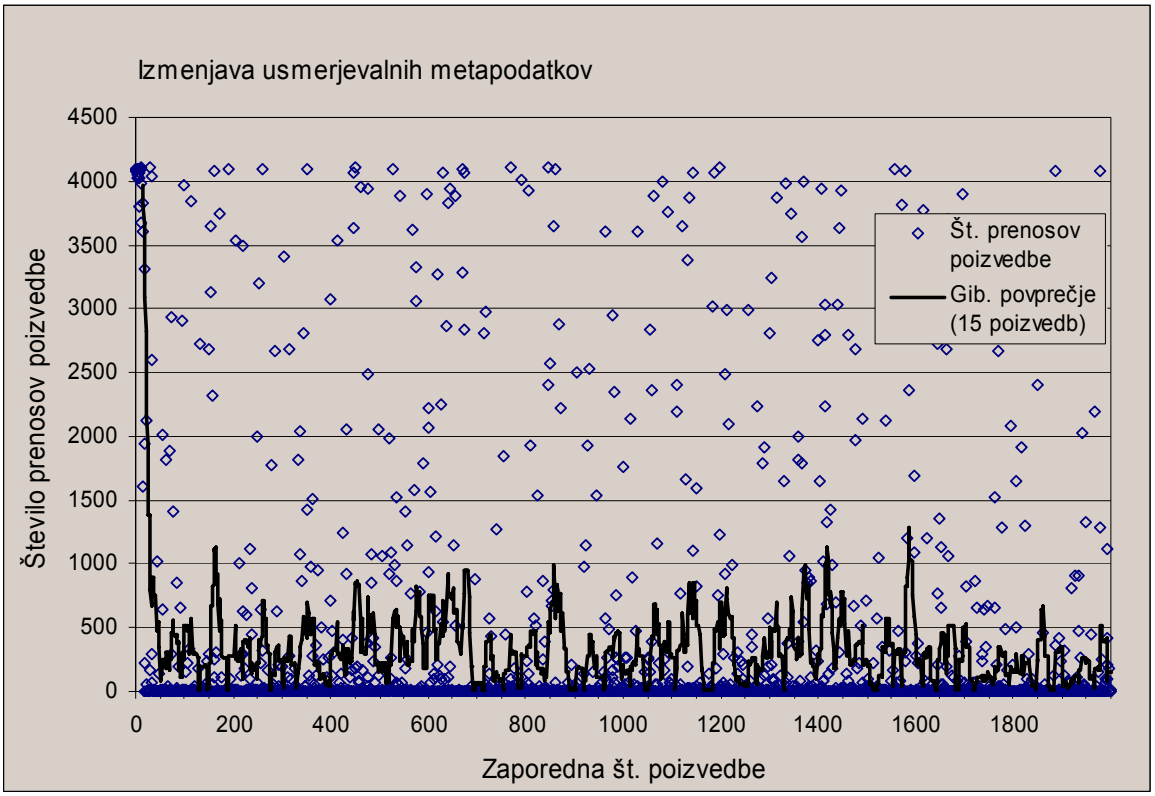
Pri poplavljanju se slika ne razlikuje od tiste iz prejšnjega razdelka, saj večina poizvedb zahteva okrog 4000 prenosov. Graf 8.4 pa prikazuje usmerjanje s pomnjenjem posredovanih odgovorov. Ker gre za mešanico poizvedb po različnih vsebinah, traja več časa, da se poizvedba po kaki vsebini ponovi, zato traja več časa, da sistem doseže stacionarno stanje, povprečno število prenosov pa je (tudi zaradi dinamičnosti omrežja) značilno višje kot pri konfiguriranju posameznih vsebin iz prejšnjega razdelka.




*Graf 8.4: Pomnjenje posredovanih odgovorov: število prenosov v zaporedju poizvedb, ki sprašujejo po mešanici različno popularnih vsebin.*

Točke, ki označujejo število prenosov za posamezno poizvedbo, so pri prvih 200 do 300 poizvedbah zgoščene okrog vrednosti 4000, vrednosti pod 1500 prenosov pa so na tem delu grafa zelo redke. Na desnem delu, med poizvedbami, ki so v zaporedju od 1500-tega mesta naprej, pa se slika obrne: do 4000 prenosov pride komaj kakšna, pač pa so se točke zgostile blizu osi *x*, torej v velikostnem razredu življenjske dobe poizvedbe.

Velja poudariti, da se omrežje že kmalu učinkovito usmerja poizvedbe za popularne vsebine, in sicer tako zaradi njihove višje frekvence poizvedb (ker se hitreje konfigurira) kot tudi zaradi višje repliciranosti takšnih vsebin. Težave pa nastopijo z manj popularnimi vsebinami: poizvedbe po njih so redke in tudi ko se del omrežja skonfigurira za usmerjanje do kake izmed njih, so poti daljše in se hitreje pretrgajo, pogosto še preden se jih sploh uporabi.




*Graf 8.5: Izmenjava usmerjevalnih metapodatkov: število prenosov v zaporedju poizvedb, ki sprašujejo po mešanici različno popularnih vsebin.*

Graf 8.5 pa prikazuje konfiguriranje sistema z izmenjavo usmerjevalnih metapodatkov. Šele tu lahko vidimo prednost takšnega usmerjanja. Konfiguriranje sistema je bistveno

hitrejše in tudi povprečje prenosov poizvedb je v stacionarnem stanju mnogo nižje kot pri usmerjanju s pomnjenjem posredovanih odgovorov (Tabela 8.2), pri čemer pa je uspešnost poizvedovanja praktično enaka!

| Usmerjanje | Povprečno št. prenosov posamezne poizvedbe | Oddaljenost odgovora | Delež odgovorjenih poizvedb |
|---|---|---|---|
| Poplavljanje | 3275 | 3.17 | 0.963 |
| Pomnjenje posredovanih odgovorov | 1930 | 3.01 | 0.975 |
| Izmenjava usmerjevalnih metapodatkov | 354 (vključno z izm. metapodatkov: 916) | 3.05 | 0.977 |

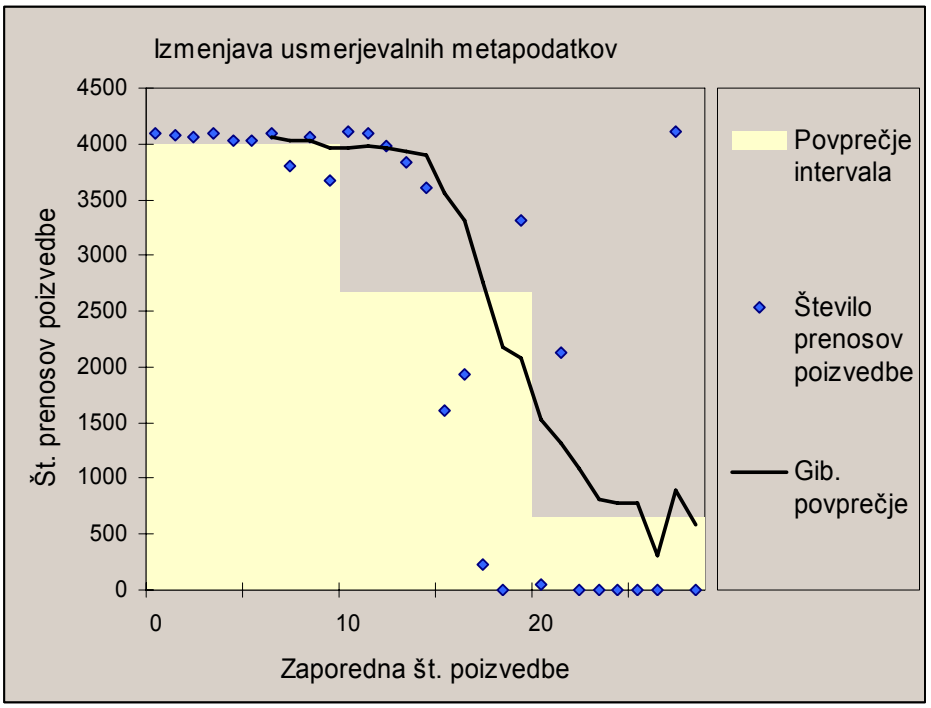
*Tabela 8.2: Primerjava predlaganih načinov usmerjanja.*

Iz tabele lahko tudi razberemo, da smo v primerjavi s poplavljanjem pri pomnjenju posredovanih odgovorov poizvedovalni promet zmanjšali za dobro polovico, pri izmenjavi metapodatkov pa kar za tri četrtine. Sistem še vedno najde dovolj odgovorov, ki so po številu skokov tudi najbližji viru poizvedbe in je torej glede tega primerljiv s poplavljanjem. Izkušnja uporabnika je torej enaka kot pri poplavljanju, lahko pa je celo boljša, če ima sistem zaradi manjše obremenitve tudi boljšo odzivnost.

S temi rezultati smo dokazali, da smo dosegli naš cilj: bistveno zmanjšati promet, ohraniti pa učinkovitost poizvedovanja. V nadaljevanju pa si bomo ogledali še nekatere značilnosti usmerjanja na različnih vrstah sistemov, saj s pomočjo tega bolje razumemo njihovo obnašanje in lažje napovedujemo, kako se bodo obnašali, tudi če pride do spremenjenih okoliščin.

Povprečna dolžina najkrajše poti v grafu je 3.95 skokov. Ne glede na že prenešene poizvedbe in odgovore zna vsako vozlišče po prvi izmenjavi metapodatkov ustrezno usmeriti vse poizvedbe, ki bodo našle odgovor na sosednjih vozliščih. Po drugi zna usmerjati že poizvedbe po tistih vsebinah, ki se nahajajo na sosedih sosedov, in tako dalje. Zato si podrobneje si oglejmo prvi del grafa, torej tisti del, kjer povprečje števila prenosov hitro pada.

Graf 8.6 prikazuje "povečavo" izseka od prve do tridesete poizvedbe, obenem pa z rumeno barvo nakazuje povprečje števila prenosov poizvedb na posameznem intervalu od ene do druge izmenjave metapodatkov.




*Graf 8.6: Podrobnejši prikaz začetnega konfiguriranja sistema ("povečava" podatkov s prejšnjega grafa).*

Že po treh izmenjavah metapodatkov tako omrežje doseže povprečne vrednosti, ki so značilne za stacionarno stanje. Na grafu vidimo, da tudi v stacionarnem stanju povprečno število prenosov precej niha, kar gre pripisati dinamičnosti omrežja in pretrganim potem.

Sistem učinkovito usmerja tudi poizvedbe za manj popularne vsebine, pretrgane poti pa sproti obnavlja (z vsako izmenjavo vsebin). Tako so le redko potrebne naknadne poplave – potrebne so le takrat, ko se pretrga kaka pot in se nato še pred naslednjo izmenjavo metapodatkov generira poizvedba, ki bi morala biti posredovana po tej poti.

## 8.2 Vpliv porazdelitve vsebin in poizvedb

Simulacije predlaganih načinov usmerjanja smo izvajali z različnimi porazdelitvami vsebin in poizvedb. Navedli smo že, da je v realnih sistemih ugotovljena Zipfova

porazdelitev zaželenosti vsebin in da zato sklepamo, da je takšna tudi porazdelitev vsebin. Če pa bi vsebine vstopale v sistem na drugačen način, ne zaradi poizvedovanja in posledičnih prenosov, bi bila lahko porazdelitev tudi drugačna. Za primerjavo smo izbrali uniformno porazdelitev vsebin, po katerih poizvedujemo bodisi z uniformno bodisi z Zipfovo porazdelitvijo poizvedb.

| | Porazdelitev vsebin – porazdelitev poizvedb | | |
|---|---|---|---|
| **Usmerjanje** | **Uni - uni** | **Uni - Zipf** | **Zipf - Zipf** |
| Pomnjenje | 4,17 | 4,49 | 10,48 |
| Izmenjava | 2,42 | 2,42 | 6,82 |

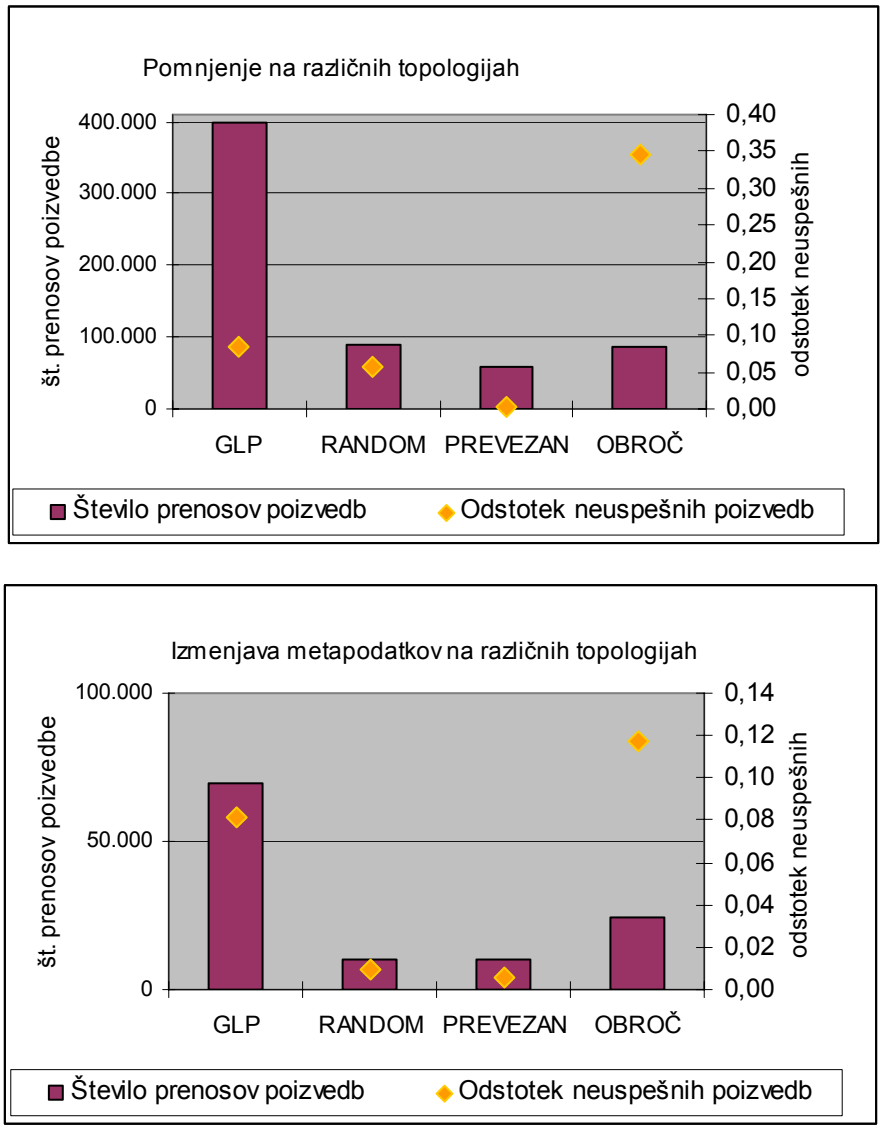
*Tabela 8.3: Povprečno število prenosov poizvedbe v stacionarnem stanju na mali topologiji GLP. Pri poplavljanju povprečje znaša 287.8 skokov.*

Rezultate prikazuje Tabela 8.3. Vidimo lahko, da se na uniformno porazdeljenih vsebinah povprečno število prenosov poizvedb praktično ne razlikuje, ne glede na to ali te povprašujejo po vsebinah enakomerno (uniformno) ali pa v skladu s popularnostjo po Zipfovi porazdelitvi.

Na prvi pogled preseneča višje število prenosov v sistemu, kjer so tudi vsebine porazdeljene po Zipfu. Vendar razlog lahko hitro odkrijemo: manj popularne vsebine so shranjene le na enem vozlišču in zaradi dinamičnosti sistema lahko kmalu postanejo nedosegljive. Poizvedbe po takšnih vsebinah se poplavijo in povzročijo nesorazmerno veliko število prenosov (stokrat več kot poizvedba, ki najde iskano vsebino), ki dvigujejo tudi skupno povprečje. Podrobnejšo sliko si lahko ogledamo v Prilogi (Graf 9.2, Graf 9.3, Graf 9.4). Če bi v izračunu upoštevali le odgovorjene poizvedbe, bi bilo povprečje na nivoju ostalih dveh konfiguracij.

## 8.3 Vpliv vrste in velikosti topologije

Zaradi svojih specifičnih lastnosti ima vrsta topologije velik vpliv na kumulativno obremenitev navideznega omrežja. Graf 8.7 primerja rezultate simulacij, ki so se izvajale s popolnoma enakimi vhodnimi parametri, le na različnih topologijah.




*Graf 8.7: Kumulativno število prenosov poizvedb in odstotek neuspešnih poizvedb za sisteme enakih lastnosti, le z različnimi vrstami topologij.*

Tako pri pomnjenju kot tudi pri usmerjanju z izmenjavo metapodatkov vidimo, da je topologija GLP izrazito zahtevna, kar se tiče skupnega števila prenosov. Razlog za to so vozlišča visoke stopnje, ki služijo kot bližnjica v bolj oddaljene dele grafa in zato omogočijo poizvedbi, da doseže višje število vozlišč. Posledično je tudi učinkovitost poizvedovanja zadovoljiva. Vidimo pa, da bi omrežje z naključno generirano topologijo in prevezan obroč ravno tako dosegla visoko učinkovitost pri bistveno manjšemu skupnemu številu prenosov, kar nam lahko služi kot navdih pri načrtovanju morebitnih novih vrst vsebinskih omrežij.

Opazimo lahko še, da se topologija obroča obnaša izrazito neučinkovito, saj je poizvedbi zaradi zelo dolgih poti v grafu dostopen le manjši del topologije. Samo usmerjanje zaradi omejene življenjske dobe poizvedbe tu učinkovitosti ne more izboljšati.

Simulacije smo izvajali na velikih in malih topologijah enakih lastnosti, pri čemer je bilo število vozlišč pri malih desetkrat manjše. Osnovne značilnosti usmerjanja so na obeh velikostih enake, vendar so razlike med posameznimi vrstami topologij pri velikih bolj poudarjene. Na obeh topologijah GLP je število prenosov poizvedb ne glede na način usmerjanja največje. Povsod so očitne slabosti topologije obroča. Prevezan obroč se odreže v malih topologijah bolje kot pri velikih, saj poizvedbe v malih dosežejo relativno večji delež vozlišč in so zato uspešnejše.

Tabela 8.4 nam osveži podatke o tem, kakšne so dolžine poti na malih topologijah:

| Topologija | Povprečna najkr. pot | Najdaljša najkr. pot |
|---|---|---|
| NAKLJUČNI GRAF | 4,40 | 9 |
| 2x POVEZAN OBROČ | 19,62 | 39 |
| 2x POVEZAN OBROČ z 0,2 naključnim prevezovanjem | 4,75 | 9 |
| GLP | 3,40 | 9 |

*Tabela 8.4: Dolžine poti na malih topologijah (155 vozlišč).*

Zavedati se moramo, da ima mala topologija dobrih 150 vozlišč in ker je življenjska doba poizvedbe 5 skokov, je primerljiva s povprečno najkrajšo potjo v grafu (razen pri obroču, ki ima bistveno daljšo povprečno najkrajšo pot). Večina poplavljenih poizvedb tako lahko doseže večino vozlišč, kar pri veliki topologiji ne velja (podrobnejši prikaz je v Prilogi - Graf 9.5)

## 8.4 Ostale značilnosti

V razdelku 5.5 smo teoretično pokazali, da se usmerjevalnih podatkov ni smiselno periodično brisati samo zato, da bi preprečili napačno usmerjanje. Pokazali smo, da bi na ta način zbrisali tudi mnogo še veljavnih podatkov. Simulacije so to domnevo potrdile,

saj je v primeru periodičnega brisanja precej višje povprečno število prenosov poizvedbe (Graf 9.6 in Graf 9.7 v Prilogi).

V razdelkih 7.3.2.1 in 7.3.2.2 smo opisali izvedene metrike, ki ocenjujejo ceno poizvedbe ter učinkovitost oziroma uspešnost poizvedovanja. Poglejmo, kaj nam povedo o usmerjanju v opazovanih sistemih.

Visoko povprečno število vozlišč, ki jih obišče posamezna poizvedba, samo po sebi ne pomeni nič slabega, saj nakazuje večjo verjetnost, da bo na enem izmed njih najden odgovor. Ker pa posledično pomeni večje število prenosov poizvedbe, ga vseeno poskušamo optimirati.

Želimo si, da bi bila obremenjenost vozlišč *m* čimbolj enakomerna, oziroma da vsaj navzgor ne bi preveč odstopala. Premočno obremenjena vozlišča namreč predstavljajo ozko grlo, ki povzroča slabo odzivnost celotnega sistema. Naše simulacije kažejo ne preveliko povprečno obremenjenost, vendar pa so maksimalne vrednosti od 10 do 15 krat višje od povprečja, kar pomeni, da so nekatera vozlišča izredno obremenjena.

| Metrika | Poplavljanje | Pomnjenje | Izmenjava |
|---|---|---|---|
| Povp. št. vozlišč, ki jih obišče poizvedba – *r* | 145 | 14.6 | 4.9 |
| Povp. št. posredovanih poizvedb na vozl. na poizv. (obremenjenost) - *m* | 1.86 | 0.16 | 0.05 |
| Povp. kolikokrat poizvedba doseže isto vozlišče (cena) - *c* | 1.94 | 1.11 | 1.02 |
| Delež nepotrebnih prenosov (redundanca) - *p* | 0.48 | 0.07 | 0.01 |
| Povp. št. najdenih vozlišč z odgovorom (učinkovitost) - *d* | 5.75 | 1.54 | 1.02 |

*Tabela 8.5: Izvedene metrike za vse tri različice usmerjanja na mali topologiji GLP.*

Cena *c* in redundanca *p* sta sorodni meri, pravzaprav redundanca izhaja neposredno iz cene. Želimo ju minimizirati, saj predstavljata prenose, ki so popolnoma odveč. Delež redundantnih prenosov sicer narašča hkrati s koeficientom gručenja; najvišji je pri topologiji dvakratno povezanega obroča in najnižji pri naključnem grafu. Koeficient gručenja nam namreč pove, kolikšen delež sosednjih vozlišč nekega vozlišča je

povezanih tudi med seboj, kar pomeni, da si med seboj neposredno izmenjujejo sporočila. Tako je večja verjetnost, da bo vozlišče isto poizvedbo dobilo od dveh ali več sosedov. Zaključimo lahko, da so nekatere topologije že po svoji naravi nagnjene k višji redundanci kot druge. Zato bi bilo pri načrtovanju novih modelov vsebinskih omrežij dobro upoštevati tudi vse svetle strani topologije naključnega grafa.

Če je naloga sistema, da uporabniku vrne zgolj en odgovor na poizvedbo, kot smo predpostavili v našem modelu, nam učinkovitost usmerjanja *d* ne prinaša nobene zanimive informacije. Včasih pa želi uporabnik čimveč različnih odgovorov na svojo poizvedbo, ali pa vsaj določeno število le teh. Metrika *d* nam v takem primeru pove, da si pri usmerjanju z izmenjavo usmerjevalnih metapodatkov ne moremo obetati, da bo uporabnik zadovoljen. Pomnjenje posredovanih odgovorov prinaša le malo boljše rezultate in če ne želimo uporabljati poplavljanja, bi bilo v takem primeru najbolje na različnih delih omrežja sprožiti več enakih poizvedo s pomočjo pomnjenja ali izmenjave metapodatkov, vse posamezne odgovore pa nazadnje zbrati in posredovati uporabniku.

## 8.5 Smernice za uporabo predlaganih mehanizmov

Iz prikazanih rezultatov lahko ugotovimo, da odločitev za nek način usmerjanja poizvedb ni preprosta ali izvedljiva samo na podlagi enega parametra, ampak moramo upoštevati več vidikov in lastnosti sistema in nato izbrati usmerjanje glede na rezultanto vseh sistemskih značilnosti. Zavedati se moramo tudi dejstva, da je vse lastnosti sistema izredno težko ovrednotiti pred njegovo implementacijo, mnogo lažje jih je eksperimentalno pridobiti in izmeriti na delujočem sistemu. Zato v tem razdelku podajamo le splošne smernice, ki napeljujejo in usmerjajo k neki odločitvi, ki pa jo mora konec koncev sprejeti načrtovalec sistema tudi na podlagi veliko izkušenj in nekaj intuicije.

### *8.5.1 Analiza značilnosti sistema*

Najprej je potrebno opraviti analizo značilnosti sistema, na podlagi katere bomo kasneje sprejeli odločitev za najprimernejši način usmerjanja. Pri analizi sistema si pomagamo z naslednjim seznamom vprašanj.

***TOPOLOGIJA:***

- Kakšen način vključevanja v sistem uporabljajo novo prihajajoča vozlišča?
- So znana pravila, ki dovoljujejo ali prepovedujejo povezavo na določena vozlišča novim vozliščem?
- So znana pravila, ki dovoljujejo ali prepovedujejo povezavo (prevezavo) na določena vozlišča obstoječim vozliščem?
- Je število povezav za posamezno vozlišče omejeno navzgor in / ali navzdol?
- V kakšnem velikostnem razredu pričakujemo število udeležencev?
- Kakšno dinamičnost pričakujemo? Kakšno povprečno življenjsko dobo vozlišča v sistemu pričakujemo in kolikšen delež stalno priključenih vozlišč?

Navadno v nestrukturiranih sistemih na zgornja vprašanja ne bomo našli točnih in jasnih odgovorov, vsaj ne na vsa. Če imajo vozlišča pri vzpostavljanju aplikacijskih povezav precej svobode, lahko z veliko verjetnostjo predvidevamo, da se bodo povezovala na tista vozlišča, od katerih si obetajo največ koristi. To pa so predvsem dobro povezana vozlišča (z višjo stopnjo) in vozlišča, ki nudijo veliko vsebin. V takem primeru je velika verjetnost, da bo nastala topologija ustrezala modelu GLP.

Topologije obroča ali prevezanega obroča prav po naključju ne bomo mogli pridelati; če bi imeli razlog za to, pa bi ga lahko kvečjemu predpisali v sam protokol. Dober razlog bi najverjetneje vključeval tudi vnaprejšnjo odločitev o načinu usmerjanja.

***NAČIN VSTOPANJA VSEBIN V SISTEM IN NAČIN POIZVEDOVANJA:***

- Kako in od kod udeleženci pridobivajo vsebine?
- Kolikšen delež vsebin dobijo udeleženci prek omrežja kot posledico poizvedovanja in kolikšen iz drugih virov?
- Ali predvidevamo, da bodo vse vsebine enako zaželene?
- Koliko različnih vsebin pričakujemo (vsaj velikostni razred: malo, srednje, zelo veliko) v celoti in koliko na posameznem vozlišču?

### 8.5.2 Razmislek pred odločitvijo

Kadar ne pričakujemo, da bo omrežje vključevalo na tisoče uporabnikov, je lahko odločitev za poplavljanje popolnoma sprejemljiva. Posebej to velja, kadar pričakujemo, da so vozlišča manj zmogljive naprave, ki si ne morejo privoščiti dovolj pomnilnih kapacitet za hranjenje posredovanih odgovorov in usmerjevalnih metapodatkov (na primer mobilne naprave - dlančniki ali telefoni). Pri zelo visoki dinamičnosti omrežja (zelo veliko vstopov v sistem in izstopov iz njega v časovni enoti, kratek čas prebivanja v sistemu, visoka spremenljivost lokalnih vsebin) predlagani izboljšavi poplavljanja tudi ne moreta delati čudežev in se obnašata tako kot poplavljanje. Ker pa sta bolj zahtevni glede lokalnih virov, in tudi glede števila prenosov sporočil (izmenjava usmerjevalnih metapodatkov! je odločitev za poplavljanje smiselna.

V manjših topologijah uvajanje kompliciranih usmerjevalnih postopkov težko upravičimo, saj le redko pride do tolikšne obremenjenosti omrežja, da bi se le to zasitilo. Le če bi poplavljanje delovalo na robu zmogljivosti sistema, bi pomnjenje prineslo ravno tolikšno zmanjšanje količine prometa, da bi sistem lahko sprejemljivo deloval.

Če predvidevamo topologijo GLP, se v vseh ostalih primerih lahko odločimo za pomnjenje posredovanih odgovorov. Zlasti veliko korist od te odločitve pa lahko pričakujemo v primeru, da imajo vsebine Zipfovo porazdelitev popularnosti in da vozlišča pridobivajo vsebine le prek sistema, torej da so vsebine tudi porazdeljene po vozliščih v skladu z Zipfovo porazdelitvijo. Če se vsebine na vozliščih ne spreminjajo z visoko dinamiko, uporabimo usmerjanje z izmenjavo metapodatkov, saj za same izmenjave ne bo potrebno prenašati toliko sporočil, da bi to skazilo celotno sliko. V primeru zelo visoke dinamike pa bo izmenjava metapodatkov preveč obremenila omrežje, zato zaradi bolj enostavne implementacije uporabimo pomnjenje posredovanih odgovorov.

Če pa vendarle lahko predvidimo, kako pogoste izmenjave metapodatkov bi morali implementirali, da bi sledile dinamiki spreminjanja lokalnih vsebin, je vredno razmisliti tudi o tej alternativi. Tovrstno usmerjanje bi bilo zlasti priporočljivo, če bi izmenjavo metapodatkov lahko implementirali tako, da bi si vozlišča ob vsaki izmenjavi poslala le inkrementalna sporočila, torej spremembe usmerjanja od prejšnje izmenjave. To bi

bistveno zmanjšalo velikost izmenjanih sporočil in tako dodatna obremenitev omrežja zaradi izmenjave ne bi bila več tako značilna.

V preostalih primerih je zanesljivo zadovoljiva odločitev pomnjenje posredovanih odgovorov. V simuliranih sistemih se je vedno obnesla bolje ali vsaj enako kot poplavljanje, obenem pa je bolj predvidljiva kot usmerjanje z izmenjavo metapodatkov.

# 9 SKLEP

Disertacija obravnava področje porazdeljenega iskanja v vsebinskih omrežjih z arhitekturo enak z enakim. Osredotoča se na vsebinska omrežja tipa A (po Kungu), ki so nestrukturirana in v njih zaradi tega usmerjanje poizvedovalnih sporočil predstavlja več težav kot v ostalih tipih vsebinskih omrežij.

V disertaciji smo podrobneje predstavili značilnosti arhitekture enak z enakim in njene uporabe v vsebinskih omrežjih. Analizirali smo lastnosti vsebinskih omrežij tipa A, pri čemer smo poudarili njihove topološke značilnosti na aplikacijski plasti, zlasti potenčne zakone in lastnost majhnega sveta. Poiskali smo tudi algoritem za generiranje umetnih topologij, ki čimbolj ustrezajo izmerjenim lastnostim realnih topologij.

Najbolj množično uporabljan mehanizem za usmerjanje poizvedb v razširjenih implementacijah vsebinskih omrežij tipa A je poplavljanje, ki pa je zelo redundantno. V literaturi najdemo več predlogov za racionalnejše usmerjanje, ki težijo k temu, da bi določeno vozlišče poizvedbo prejelo le od enega izmed svojih sosedov, ne od več sosedov. Mi pa smo želeli zmanjšati število redundantnih prenosov na račun dejstva, da se nekatere poizvedbe pogosto ponavljajo.

Zato smo predlagali dve izboljšavi usmerjanja, ki od vsakega vozlišča zahtevata, da shranjuje metapodatke o posredovanih odgovorih. Pri usmerjanju s pomnjenjem posredovanih odgovorov vozlišče naslednjo poizvedbo po isti vsebini usmeri le v smer, iz katere je prišel prejšnji odgovor. Usmerjanje z izmenjavo metapodatkov deluje podobno, le da si sosednja vozlišča poleg tega še med seboj periodično izmenjujejo metapodatke o svojih vsebinah in o usmerjanju do ostalih vsebin. Tako lahko pravilno usmerjajo tudi poizvedbe do vsebin, za katere do tedaj še ni bila generirana nobena poizvedba.

Predvideli smo, da bo navidezno omrežje po določenem številu poizvedb, potrebnih za konfiguriranje oziroma zbiranje metapodatkov, bistveno manj obremenjeno kot pri

uporabi osnovnega poplavljanja, zaradi pomanjkanja matematičnih modelov pa nismo znali točneje oceniti, kolikšno bo to zmanjšanje.

Nato smo postavili konceptualni model vsebinskega omrežja, opredelili smo lastnosti topologije, usmerjanja, velikost omrežja, porazdelitev in ponovljivost poizvedovanja ter porazdelitev in replikacijo vsebin. Model smo nato implementirali, izvedli validacijo in verifikacijo ter nato izvedli večje število simulacij na različnih topologijah, z različnimi vhodnimi parametri in z vsemi tremi vrstami usmerjanja.

Rezultati simulacij so potrdili naša pričakovanja. Zlasti pri kombinacijah vhodnih parametrov, ki so zelo blizu realnim sistemom za izmenjavo datotek, smo ugotovili, da uporaba pomnjenja posredovanih odgovorov zmanjša kumulativno število prenosov poizvedb na polovico do četrtino, uporaba izmenjave metapodatkov pa na desetino.

Zato smo priporočili uporabo obeh predlaganih usmerjanj v sistemih za izmenjavo vsebin in njim podobnih sistemih, kjer so vozlišča povprečno zmogljivi osebni računalniki. V sistemih, kjer pa vozlišča predstavljajo manj zmogljive naprave (dlančniki, mobilni telefoni in podobno), moramo biti zaradi večje zahtevnosti po lokalnih virih, zlasti pomnilnih, previdnejši. Odsvetujemo uporabo izmenjave metapodatkov, lahko pa se uporablja pomnjenje posredovanih odgovorov, če le skupno število vsebin v sistemu ni preveliko.

V disertaciji predstavljena problematika je zelo aktualna in se izjemno hitro razvija, pojavljajo se tudi nove implementacije sistemov enak z enakim in nove izvedbe porazdeljenega iskanja, ki odpirajo nova vprašanja in nova raziskovalna področja. Med možnostmi za nadaljnje raziskave naj omenimo le nekatere:

- Izvedba porazdeljenega iskanja, ko uporabnik zahteva več kot en odgovor ali pa točno določeno število odgovorov.
- Različici predlaganega usmerjanja, pri kateri pomnimo več odgovorov za isto vsebino in v skladišču usmerjevalnih metapodatkov hranimo več poti do iste vsebine.
- V literaturi nismo našli nobene empirične raziskave porazdeljenosti in repliciranosti vsebin.

- Popularnost vsebin realno ni konstantna, ampak se s časom spreminja. Zanimivo bi bilo simulirati dinamično popularnost: nekatere vsebine pridobivajo na zaželenosti in je veliko poizvedb po majhnem številu kopij, nato počasi število poizvedb in vzporedno tudi kopij vsebine narašča, poizvedovanje pa se zlagoma umiri in nato počasi pade.

# PRISPEVKI ZNANOSTI

Tema doktorske disertacije je upoštevati visoko stopnjo ponavljanja poizvedb ter topološke in druge lastnosti sistemov enakovrednih računalnikov, ki dovoljujejo in osmišljajo uvedbo določenih izboljšav v osnovni usmerjevalni postopek poplavljanja. S spremembo usmerjevalnega mehanizma smo zmanjšali količino celotnega prometa, ki ga mora prenesti omrežje, obenem pa ohranili učinkovitost poizvedovanja.

V disertaciji smo predlagali izboljšave usmerjanja, na različnih topologijah simulirali njihovo obnašanje in ocenili pričakovano zmanjšanje prometa. Uporabili smo vsebinsko usmerjanje, torej za usmerjanje do znane vsebine, pri čemer ciljno vozlišče ni pomembno. V nasprotju s sistemi, ki indeksirajo celoto ali del dosegljivih vsebin, nismo shranjevali informacij o ciljnem vozlišču, ampak o usmerjanju, torej le o sosednjem vozlišču.

Disertacija prinaša naslednje prispevke znanosti:

- Analiza obnašanja in topoloških lastnosti vsebinskih omrežij z arhitekturo enakovrednih računalnikov ter ugotovitve, katere od teh lastnosti lahko izkoristimo za zmanjšanje količine prometa.
- Izboljšanje usmerjanja ponavljajočih se poizvedb ob upoštevanju že posredovanih odgovorov in možnosti izmenjave usmerjevalnih podatkov med sosednjimi vozlišči.
- Glede na rezultate simulacij in analize obnašanja predlaganih različic usmerjanja je bilo ugotovljeno, da je v modeliranih sistemih osnovno poplavljanje smiselno nadgraditi s predlaganimi izboljšavami.

# LITERATURA IN VIRI

[1] Gartner Group, The Emergence of Distributed Content Management and Peer-to-Peer Content Networks, GartnerConsulting, Engagement #010022501, January 2001, http://marketplacena.gartner.com/010022501oth-NextPage.pdf

[2] A. Oram (ed.), Peer-to-Peer: Harnessing the Power of Disruptive Technologies, O'Reilly & Associates, 432 pages, 2001.

[3] S. Waterhouse, D. M. Doolin, G. Kan, Y. Faybishenko: Distributed Search in P2P Networks, IEEE Internet Computing, Feb. 2002.

[4] D. Brookshier (ed.): JXTA: Java P2P Programming, Sams Publishing, 423 pages, 2002.

[5] D. S. Milojičić, V. Kalogeraki, R. Lukose, K. Nagaraja, B. Richard, S. Rollins, Z. Xu: Peer-to-Peer Computing, Technical report HPL-2002-57, HP Laboratories Palo Alto, 2002.

[6] Kung, H. T., and Wu, C. H. (2002). Content Networks: Taxonomy and New Approaches, to appear in The Internet as a Large-Scale Complex System, K. Park and W. Willinger (eds), Oxford University Press, 2002.

[7] C. Rohrs, Query Routing for the Gnutella Network, LimeWire LLC, http://www.limewire.com/developer/query_routing/keyword%20routing.htm, Maj 2002.

[8] K. Sripanidkulchai: The Popularity of Gnutella Queries and its Implications on Scalability. http://www-2.cs.cmu.edu/~kunwadee/research/p2p/gnutella.html

[9] I. Clarke, O. Sandberg, B. Wiley, T. W. Hong: Freenet: a Distributed Anonymous Information Storage and Retrieval System, in Designing Privacy Enhanced Technologies, LNCS 2009, ed. By H. Federrath. Springer, New York 2001.

[10] I. Clarke, T. W. Hong, S. G. Miller, O. Sandberg, B. Wiley: Protecting Free Expression Online with Freenet, IEEE Internet Computing 6(1), 40-49 (2002).

[11] A. Pfitzmann, M. Koehntopp: Anonymity, Unobservability, and Pseudonymity – A Proposal for Terminology, in: H. Federrath (Ed.): Designing Privacy Enhancing Technologies, International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA, Proceedings. LNCS 2009, Springer 2001, pp. 1-9.

[12] ISO IS 15408 (Common Criteria for IT Security Evaluation), 1999, http://csrc.nist.gov/cc.

[13] N. Minar: Distributed Systems Topologies, Part I and Part II, O'Reilly Peer-to-Peer and Web Services Conference, November 2001, http://www.openp2p.com/pub/a/p2p/2001/12/14/topologies_one.html.

[14] D. Liben-Nowell, H. Balakrishnan, D. Karger: Observations on the Dynamic Evolution of Peer-to-Peer Networks, Proc. 1st Intl. Workshop on Peer-to-Peer Systems IPTPS'02, Cambridge Massachusets, March 2002.

[15] M. Faloutsos, P. Faloutsos, C. Faloutsos, On Power-Law Relationships of the Internet Topology, in L. Chapin (ed.), Proc. SIGCOMM '99, 251 – 262 (New York: ACM, 1999).

[16] M. Ripeanu, I. Foster: Mapping the Gnutella Network, IEEE Internet Computing special issue on Peer-to-Peer Networking, vol. 6(1), pages 50-57, February 2002.

[17] P. Keleher, B.Bhattacharjee, B. Silaghi: Are Virtualized Overlay Networks Too Much of a Good Thing? , Proc. 1st Intl. Workshop on Peer-to-Peer Systems IPTPS'02, Cambridge Massachusets, March 2002.

[18] A. Iamnitchi, M. Ripeanu, I. Foster: Locating Data in (Small World?) Peer-to-Peer Scientific Collaborations, Proc. 1st Intl. Workshop on Peer-to-Peer Systems IPTPS'02, Cambridge Massachusets, March 2002.

[19] M. A. Jovanovic, F. S. Anexstein, K. A. Berman, Modeling Peer-to-Peer Network Topologies through "Small-World" Models and Power Laws, in Proc. of IX. Telecommunications Forum TELFOR 2001, Belgrade, 2001.

[20] R. Albert, A.-L. Barabasi: Statistical Mechanics of Complex Networks. Reviews of Modern Physics, Vol. 74, January 2002, pp. 47–97.

[21] E. J. Newman, D. J. Watts, S. H. Strogatz: Random Graph Models of Social Networks. Proc. Natl. Acad. Sci. USA 99, 2566-2572 (2002).

[22] A.-L. Barabasi, R. Albert: Emergence of Scaling in Random Networks, Science Vol.286, 509-512, October 1999.

[23] S. H. Strogatz: Exploring Complex Networks, Nature Vol. 410, 268-276, March 2001.

[24] D. J. Watts, S. H. Strogatz: Collective dynamics of Small-World Networks, Nature Vol. 393, 440-442, June 1998.

[25] J. Kleinberg: The Small-World Phenomenon: An Algorithmic Perspective. In Proceedings of the 32nd ACM Symposium on Theory of Computing, 2000.

[26] D. J. Watts, Small Worlds – The Dynamics of Networks between Order and Randomness, Princeton University Press, Princeton, New Jersey. 1999.

[27] B. C. Neuman, Scale in Distributed Systems. In T. Casavant, M. Singhal, eds,: Readings in Distributed Computing Systems, pp. 463-489, IEEE Computer Society Press, 1994.

[28] M. T. Prinkley: An Efficient Scheme for Query Processing on Peer-to-Peer Networks, Technical Report, Aeolus Research Corporation, http://aeolusres.homestead.com, 2002.

[29] G. Pandurangan, P. Raghavan, E. Upfal: Building Low-Diameter Peer-to-Peer Networks, in Proc. of the 42nd Annual IEEE Symposium on the Foundations of Computer Science (FOCS), 2001.

[30] L. A. Adamic, R. M. Lukose, A. R. Puniyani, B. A. Huberman: Search in Power-Law Networks, Physical Review E, Vol.64, 046135 (2001).

[31] Stanford Peers: http://www-db.stanford.edu/peers/

[32] A. Crespo, H. Garcia-Molina: Routing Indices For Peer-to-Peer Systems, Proc. Intl. Conf. on Distributed Computing Systems, IEEE ICDCS 2002.

[33] N. Daswani, H. Garcia-Molina, B. Yang: Open Problems in Data Sharing Peer-to-Peer Systems, Proc. Intl. Conf. on Database Theory, IEEE ICDT 2003, LNCS Vol. 2572.

[34] P. Keyani, B. Larson, M. Senthil: Peer Pressure: Distributed Recovery from Attacks in Peer-to-Peer Systems, Proc. IFIP Workshop on P2P Computing, in conjunction with Networking 2002.

[35] J. Crowcroft, I. Pratt, Peer to Peer: Peering into the Future, Proc. of the IFIP-TC6 Networks 2002 Conference, www.cl.cam.ac.uk/Research/SRG/netos/publications.html.

[36] B. Yang, H. Garcia-Molina: Efficient Search in Peer-to-peer Networks, Technical Report 2001-47, Stanford University, na http://www.db-pubs.stanford.edu/.

[37] B. Yang,. H. Garcia-Molina: Comparing Hybrid Peer-to-Peer Systems, Proc. Very Large Databases VLDB'01, Roma, Italy, 2001.

[38] B. Yang,. H. Garcia-Molina: Designing a Super-Peer Network, Proc. 19th Intl. Conf. on Data Engineering ICDE 2003, March 2003. IEEE Computer Society. http://dbpubs.stanford.edu: 8090/pub/2002-13.

[39] M. Ciglarič, T. Vidmar: Position of modern peer-to-peer systems in the distributed systems architecture. Proc. 11th IEEE Mediterranean Electrotechnical Conference, MELECON 2002, Cairo, Egypt. IEEE, Piscataway 2002, str. 341-345.

[40] M. Ciglarič, T. Vidmar: So sistemi "enak z enakim" pravi porazdeljeni sistemi?. V: ZAJC, Baldomir (ur.). Zbornik konference ERK 2002, Portorož, Slovenija. Ljubljana: IEEE Region 8, Slovenska sekcija IEEE, zv. B, str. 43-46.

[41] M. Ciglarič, M. Trampuš, M. Pančur, T. Vidmar: Message routing in pure peer-to-peer networks. V: HAMZA, M. H. (ur.). 21st IASTED International Multi-Conference Applied Informatics, Innsbruck, Austria 2003. Zürich: ACTA Press, 2003, str. 813-817.

[42] M. Ciglarič, T. Vidmar: Query routing in content networks with pure peer-to-peer architecture. V: Proc. XXVI. International Convention, MIPRO 2003, Opatija, Croatia. CTE, 2003, str. 88-92.

[43] M. Ciglarič, T. Vidmar, M. Trampuš, M. Pančur: Content networks: distributed routing decisions in presence of repeated queries. V: IPDPS, Proc. International

Parallel and Distributed Processing Symposium, 2003, Nice, France. Los Alamitos, IEEE Computer Society, 2003, str. [1-6].

[44] M. Ciglarič, T. Vidmar: Management of peer-to-peer systems. V: IPDPS, Proc. International Parallel and Distributed Processing Symposium, 2003, Nice, France. Los Alamitos, IEEE Computer Society, 2003,str. [1-6].

[45] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker: A Scalable Content-Addresable Network. Proc. ACM SIGCOMM 2001.

[46] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, H. Balakrishnan: Chord: A Scalable Peer-to-Peer Lookup Service for internet Applications. In Proc. ACM SIGCOMM, August 2001.

[47] A. Rowstron, P. Druschel: Pastry: Scalable, distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems. In Proc. 18th IFIP/ACM Intl. Conf. on Distributed Systems Platforms (Middleware 2001), November 2001.

[48] B. Zhao, J. Kubiatowicz, A. Joseph: Tapestry: An infrastructure for Fault-tolerant Wide-area Location and Routing. Technical Report UCB/CSD-01-1141, U.C. Berkeley, 2001. http://www.cs.berkeley.edu.

[49] C. Tang, Z. Xu, M. Mahalingam: PeerSearch: Efficient Information Retrieval in Peer-to-Peer Networks, Technical Report HPL-2002-198, HP Laboratories Palo Alto, July 2002.

[50] S. Joseph: Neurogrid: Semantically Routing Queries in Peer-to-Peer Networks, Technical Report. www.neurogrid.net/NeuroGridSimulations.pdf. December 2002.

[51] M. Portmann, A. Seneviratne: Cost-Effective Broadcast for Fully Decentralized Peer-to-Peer Networks. Computer Communications, uncorrected proof, November 2002.

[52] S. Saroiu, P. Gummadi, S. Gribble: A Measurement Study of Peer-to-Peer File Sharing Systems. Proc. of the Multimedia Computing and Networking. January 2002.

[53] C. R. Palmer, J. G. Steffan: Generating Network Topologies That Obey Power Laws, in Proceedings of the Global Internet Symposium, Globecom 2000, November 2000.

[54] Medina, A., Lakhina, A., Matta, I., Byers, J.: BRITE: An approach to universal topology generation. Proc. MASCOTS'01, 2001.

[55] Jin, C., Chen, Q., Jamin, Q.: Inet 3-0: Internet Topology Generator, Technical report, Department of EECS, University of Michigan Ann Arbor, 2000.

[56] T. Bu, D. Towsley: On Distinguishing between Internet Power Law Topology Generators, in Proceedings of IEEE Infocom 2002.

[57] Q. Lv, P. Cao, E. Cohen, K. Li, S. Shenker: Search and Replication in Unstructured Peer-to-Peer Networks, Proceedings of 16th ACM International Conference on Supercomputing, ICS'02, New York, USA, junij 2002 (www.cs. princeton.edu/~qlv).

[58] Q. Lv, S. Ratnasamy, S. Shenker: Can Heterogeneity Make Gnutella Scalable? Proc. 1st Intl. Workshop on Peer-to-Peer Systems IPTPS'02, Cambridge MA, March 2002.

[59] J. Vaucher, G. Babin, P. Kropf, T. Jouve: Experimenting with Gnutella Communities, Proc. 4th Conference on Distributed Communities on the Web DCW'02, LNCS, Springer Verlag 2002.

[60] X. H. Sun, L. M. Ni: Scalable Problems and Memory-Bounded Speedup, Journal on Parallel and Distributed Computing, vol. 19, pp. 27 – 37, 1993.

[61] P. Jogalekar, M. Woodside: Evaluating the Scalability of Distributed Systems, IEEE Transactions on Parallel and Distributed Systems, Vol 11, pp. 589 – 603, 2000.

***Internetni viri – spletne strani sistemov enak z enakim:***

[62] Limewire in Gnutella, http://www.limewire.org (2002, 2003).

[63] Freenet, http://freenet.sourceforge.org (2002).

[64] Kazaa, http://www.kazaa.com (2002, 2003)

[65] FastTrack, http://www.fasttrack.nu (2002).

[66] Morpheus, http://www.morpheus.com (2002).

[67] Jabber, http://www.jabber.org (2003).

[68] Groove, http://www.groove.net (2003).

[69] Avaki, http://www.avaki.com (2002).

[70] Globus, http://www.globus.org (2002, 2003).

[71] Seti@home, http://setiathome.ssl.berkeley.edu (2002, 2003).

[72] Google, http://www.google.com (2002, 2003).

[73] Napster, http://www.napster.com (2001).

[74] eDonkey, http://www.edonkey.com (2002).

# IZJAVA

Izjavljam, da sem doktorsko disertacijo izdelala samostojno pod vodstvom mentorja doc. dr. Vidmar Toneta. Pomoč ostalih sem v celoti navedla v zahvali.

V Ljubljani, 17. novembra 2003

mag. Mojca Ciglarič, dipl. ing.

# ZAHVALA

Veliko ljudi je s svojimi nasveti, podporo, spodbudo in zgledom pripomoglo k temu, da se je rodila ta disertacija.

Na prvem mestu se zahvaljujem mentorju doc. dr. Tonetu Vidmarju za strokovno vodstvo pri delu in vse koristne nasvete, pa tudi za veliko potrpežljivosti in humorja, s čimer je pripomogel, da je kljub napetem urniku delo potekalo v sproščenem vzdušju in z visoko motivacijo.

Za komentarje pri oblikovanju naslova se zahvaljujem članu komisije prof. dr. Ljubu Pipanu, predsedniku prof. dr. Nikolaju Zimicu pa za temeljit pregled vsebine in številne koristne nasvete, ki so veliko pripomogli, da delo predstavlja bolj razumljivo in zaključeno celoto.

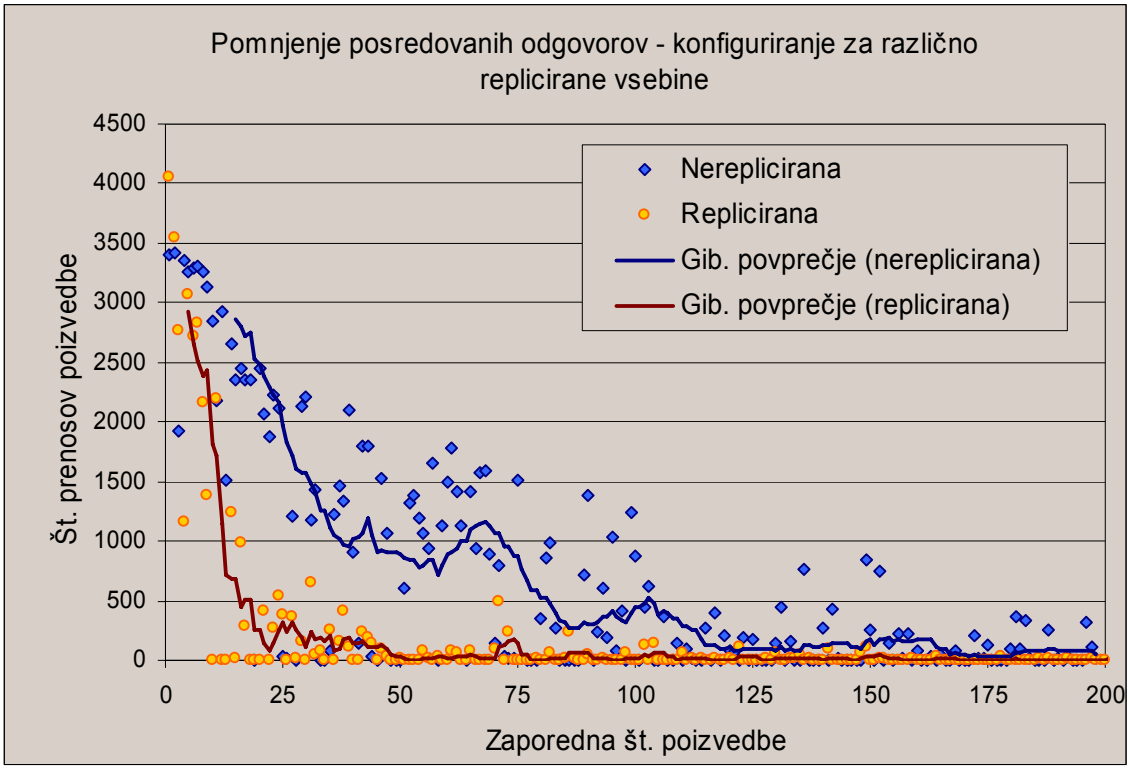
Sodelavcema in prijateljema, doktorskima kandidatoma mag. Matjažu Pančurju in mag. Mateju Trampušu se zahvaljujem za številne vsakodnevne diskusije, izmenjave izkušenj in medsebojno vzpodbujanje. Pot je bila v troje prijetnejša.

Študentu Damjanu Cvetku se zahvaljujem, da je prisluškoval omrežju Gnutella in mi priskrbel seznam iskanih ključnih besed.

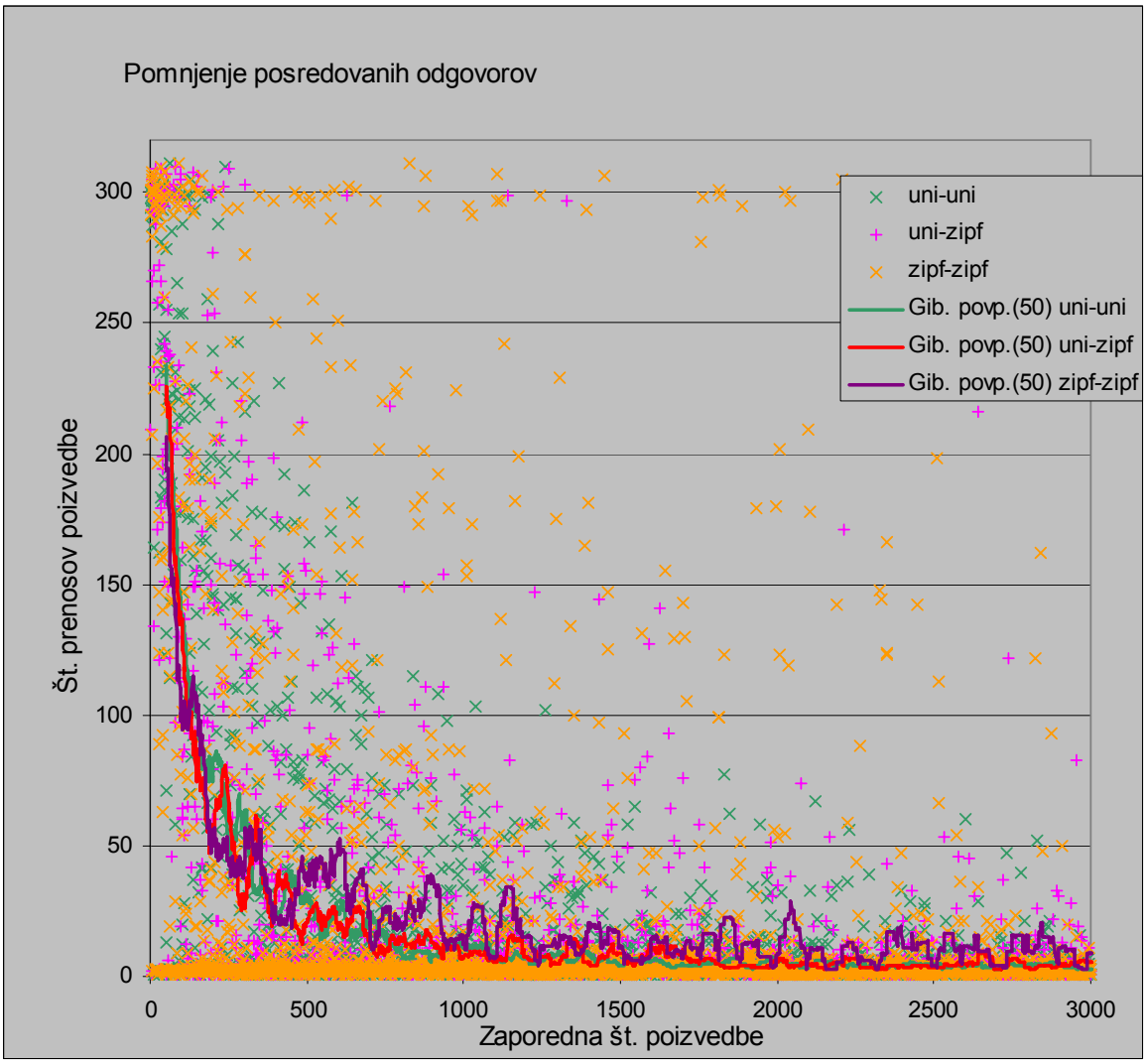
Zahvaliti se želim tudi svojim in moževim staršem ter varuški Miri Pust za vso spodbudo in neizmerno zaupanje vame, še zlasti pa za to, da so pogosto nase prevzeli del mojih materinskih dolžnosti.

Na zadnjem mestu pa se iz srca zahvaljujem še najpomembnejšim trem osebam v mojem življenju, možu Stanetu in sinovoma Tadeju in Timoteju, ki so me ves čas spodbujali in podpirali z veliko razumevanja in potrpežljivosti, obenem pa so se najbolj razveselili, ko je bilo delo vendarle končano.

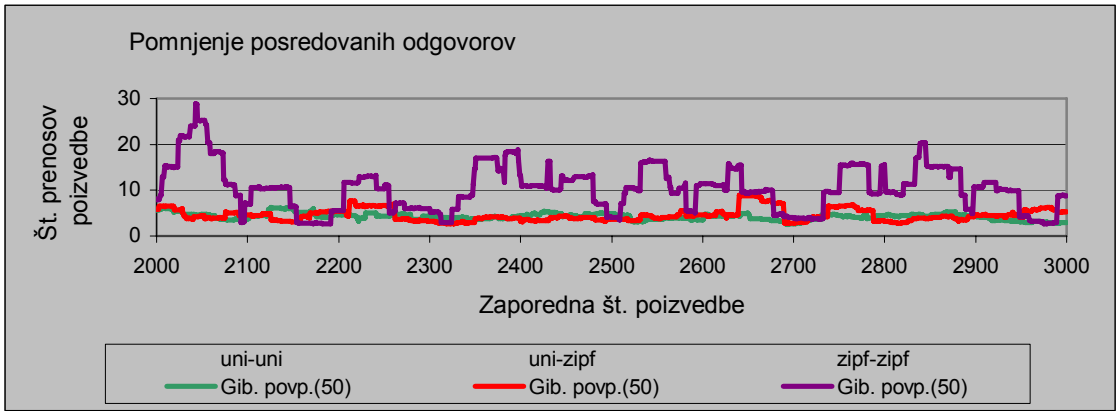
# PRILOGA - REZULTATI SIMULACIJ



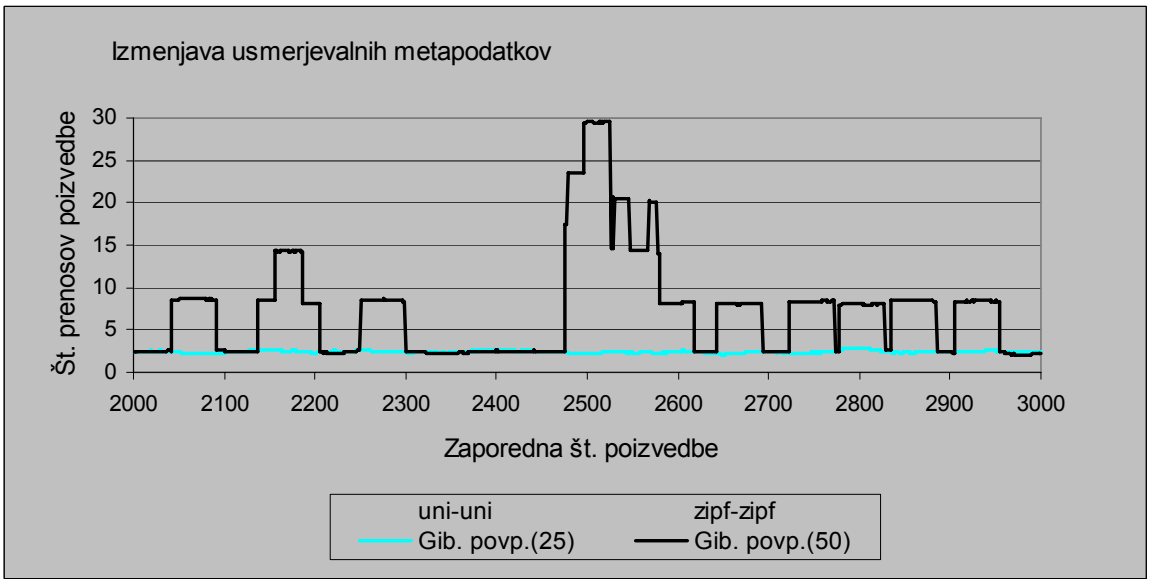

*Graf 9.1: Primerjava hitrosti konfiguriranja omrežja za poizvedbo po isti vsebini, če je ta vsebina bolj ali manj popularna in zato bolj ali manj replicirana. Graf predstavlja konfiguriranje na veliki topologiji GLP za najbolj in najmanj replicirano vsebino v sistemu (za prvo imamo 18 kopij, kar predstavlja 12% vozlišč, za drugo pa le eno kopijo, kar predstavlja 0.065% vozlišč).*



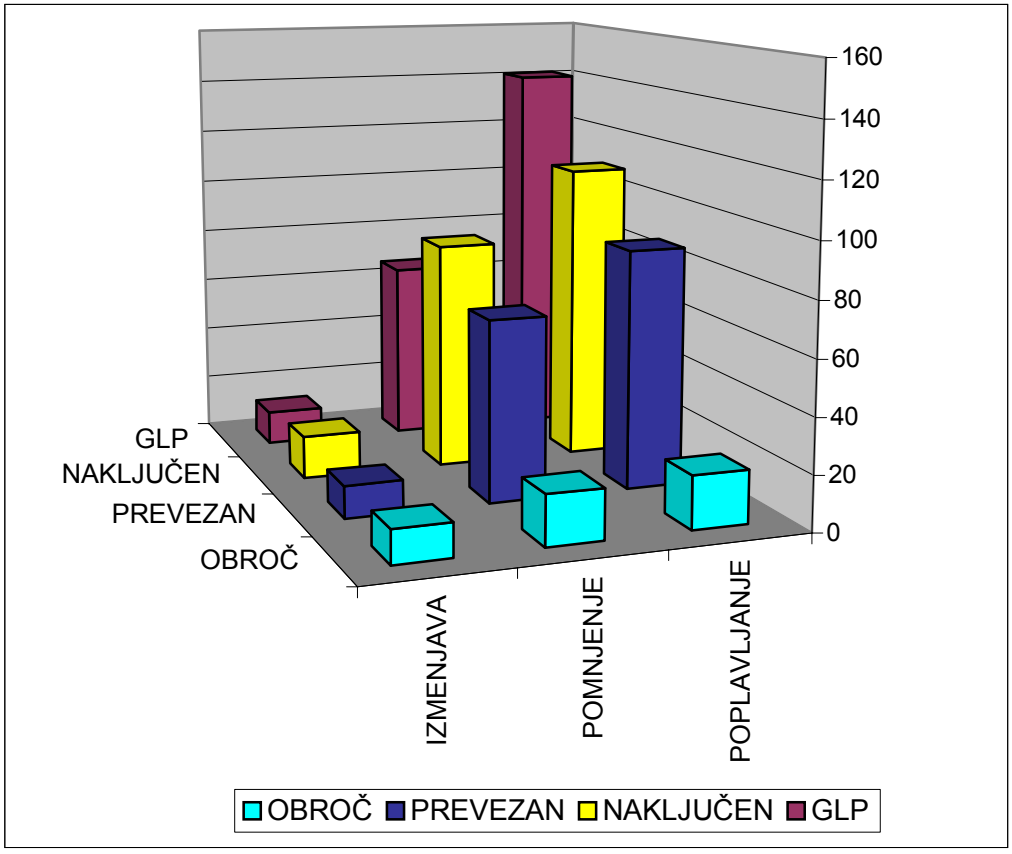

*Graf 9.2: Primerjava usmerjanja s pomnjenjem posredovanih odgovorov v sistemih z različno porazdelitvijo vsebin in poizvedb. Na prvem mestu je porazdelitev vsebin: uni – uniformna, zipf - zipfova porazdelitev zaželenosti. Na drugem mestu je porazdelitev poizvedb, torej način, kako te poizvedujejo po vsebinah: uni – uniformno (enakomerno); zipf – več po popularnih in manj po nepopularnih (v skladu z Zipfovo porazdelitvijo). V sistemu Zipf-Zipf je največ poplav, ker so manj replicirane vsebine večkrat nedostopne.*



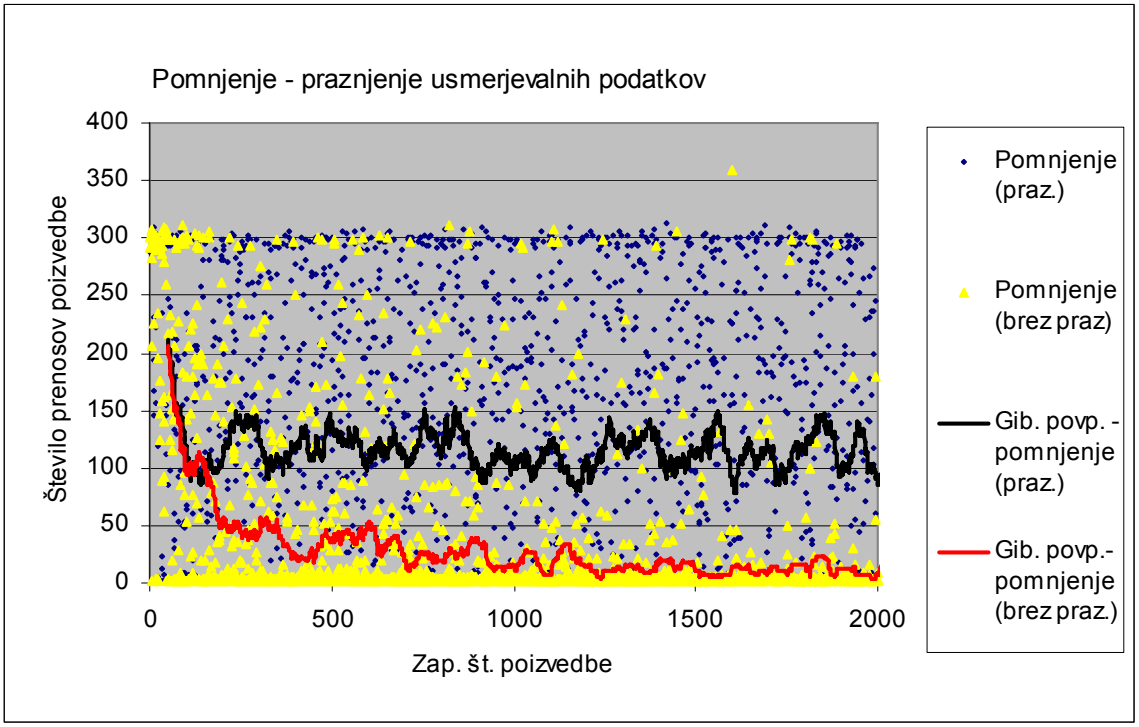

*Graf 9.3: Povečava izseka iz prejšnjega grafa. Jasno se vidijo velika nihanja v gibajočih povprečjih za konfiguracijo Zipf-Zipf, ki nastanejo zaradi nedostopnosti manj repliciranih vsebin. Ker je pri uniformni porazdelitvi vsebin njihova replikacija veliko bolj enakomerna, so pri drugih dveh sistemih ta nihanja bistveno manj opazna.*



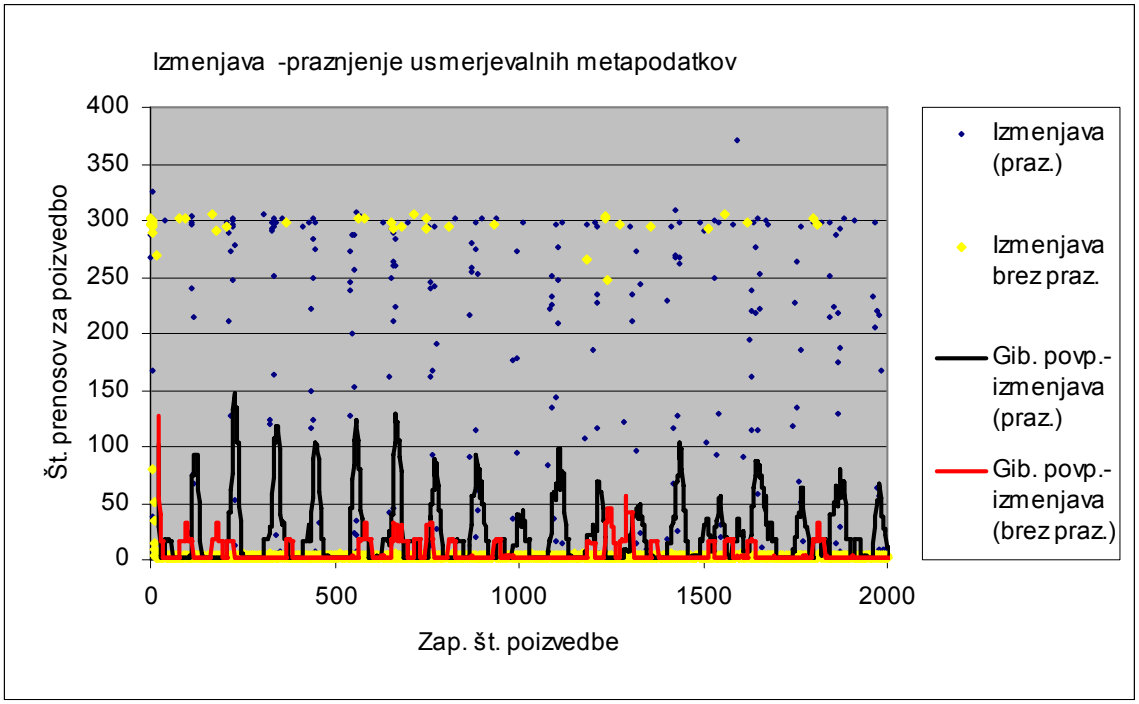

*Graf 9.4:Enak izsek iz grafa za izmenjavo usmerjevalnih metapodatkov. Sistema uni-uni in uni-zipf praktično sovpadata, zato na grafu prikazujemo le enega. V sistemu Zipf-zipf so prisotna velika nihanja iz istih razlogov kot pri usmerjanju s pomnjenjem posredovanih odgovorov.*



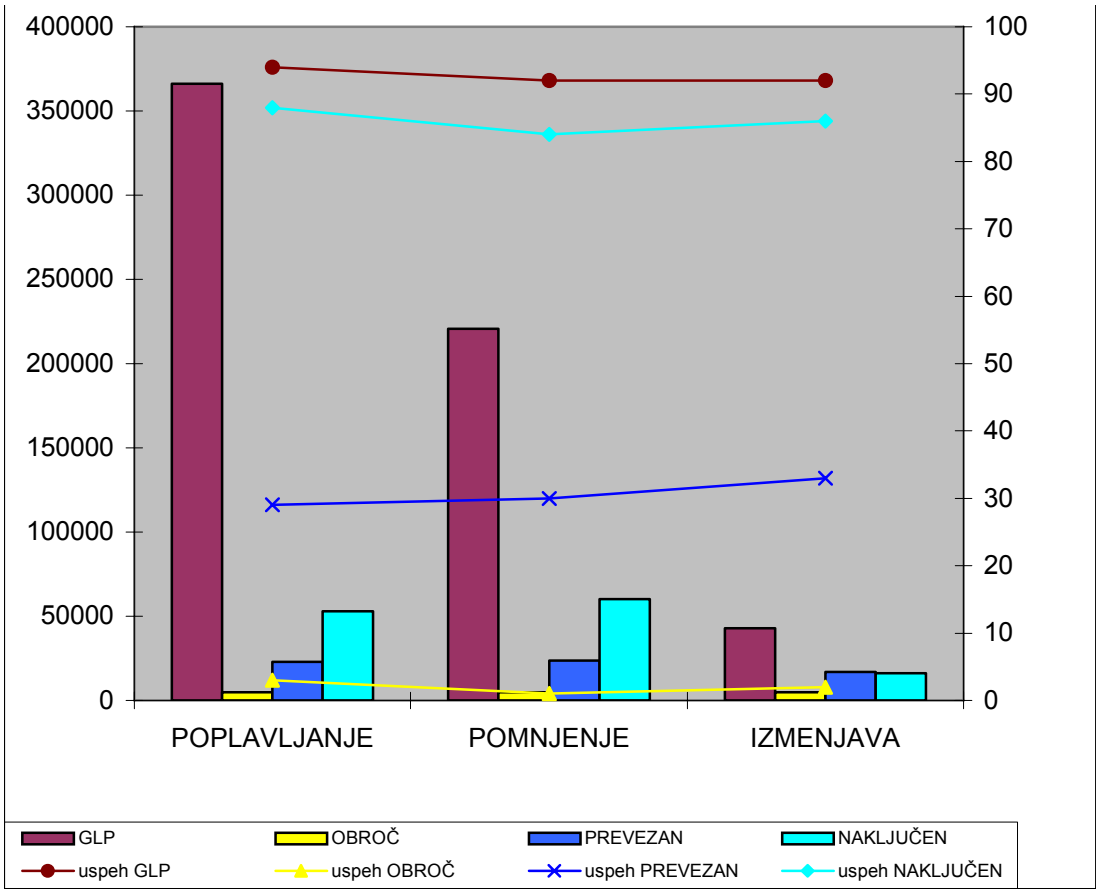

*Graf 9.5: Primerjava povprečnega števila vozlišč, ki ji doseže poizvedba v malih topologijah. Pri tem v topologiji obroča uspešnost poizvedovanja ne glede na način usmerjanja komaj doseže 50%, medtem ko se pri ostalih treh topologijah približa 100%. V velikih topologijah je slika podobna, le da topologija GLP tudi pri pomnjenju izstopa z izrazito višjim številom doseženih vozlišč.*




*Graf 9.6: Primerjava povprečnega števila prenosov poizvedbe pri pomnjenju, če praznimo domnevno zastarele metapodatke. Ker se hkrati z zastarelimi pobriše tudi mnogo še veljavnih, je skupno več škode kot koristi.*




*Graf 9.7: Primerjava povprečnega števila prenosov poizvedbe pri izmenjavi usmerjevalnih metapodatkov, če praznimo domnevno zastarele metapodatke. Opazimo lahko konice v številu prenosov na mestih, ki sledijo praznjenju metapodatkov (na vsakih 100 poizvedb), nato pa se omrežje po par izmenjavah zopet skonfigurira in število prenosov pade.*




*Graf 9.8: Primerjava kumulativnega števila prenosov za prvih 100 poizvedb in odstotka odgovorjenih poizvedb na različnih topologijah.*

**Primer zbranih podatkov: rezultati nekaterih simulacij na malih topologijah.**

| ID TOPOLOGIJE | VRSTA TOPOLOGIJE | ST VOZLISC | ID SIMULACIJE | VRSTA SIMULACIJE | ST POIZVEDB | ST ODGOVOROV |
|---|---|---|---|---|---|---|
| 22 | GLP | 154 | 412 | POMNJENJE | 300 | 1213 |
| 22 | GLP | 154 | 413 | POMNJENJE | 1000 | 4552 |
| 22 | GLP | 154 | 414 | POMNJENJE | 3000 | 6859 |
| 22 | GLP | 154 | 415 | POMNJENJE | 3000 | 1936 |
| 22 | GLP | 154 | 416 | PORAZDELJENO | 1000 | 964 |
| 22 | GLP | 154 | 432 | PORAZDELJENO | 1000 | 772 |
| 22 | GLP | 154 | 433 | PORAZDELJENO | 200 | 265 |
| 22 | GLP | 154 | 452 | POMNJENJE | 3000 | 6591 |
| 22 | GLP | 154 | 453 | POMNJENJE | 3000 | 5907 |
| 22 | GLP | 154 | 454 | POPLAVLJANJE | 3000 | 22004 |
| 22 | GLP | 154 | 455 | POMNJENJE | 3000 | 5168 |
| 22 | GLP | 154 | 456 | PORAZDELJENO | 3000 | 3049 |
| 22 | GLP | 154 | 457 | POMNJENJE | 3000 | 5127 |
| 22 | GLP | 154 | 458 | POMNJENJE | 3000 | 4882 |
| 22 | GLP | 154 | 459 | PORAZDELJENO | 3000 | 3037 |
| 22 | GLP | 154 | 460 | PORAZDELJENO | 3000 | 4267 |
| 22 | GLP | 154 | 461 | POMNJENJE | 3000 | 10872 |
| 22 | GLP | 154 | 472 | PORAZDELJENO | 3000 | 4508 |
| 22 | GLP | 154 | 473 | PORAZDELJENO | 3000 | 4374 |
| 22 | GLP | 154 | 474 | POMNJENJE | 3000 | 14292 |
| 46 | OBROC | 154 | 484 | PORAZDELJENO | 3000 | 2680 |
| 43 | OBROC | 154 | 478 | POMNJENJE | 3000 | 2848 |
| 44 | PREVEZAN | 154 | 481 | POMNJENJE | 3000 | 6138 |
| 44 | PREVEZAN | 154 | 482 | PORAZDELJENO | 3000 | 3132 |
| 42 | RANDOM | 154 | 475 | POMNJENJE | 3000 | 5141 |
| 45 | RANDOM | 154 | 483 | PORAZDELJENO | 3000 | 3141 |

| ST VSEH SKOKOV POIZVEDB | ST VSEH SKOKOV ODGOVOROV | POVP ST SKOKOV ZA POIZVEDBO | POVP ST SKOKOV ZA ODGOVOR | ST USPESNIH POIZVEDB | POVP ST SKOKOV ZA USPESNO | ST NEUSPESNIH POIZVEDB |
|---|---|---|---|---|---|---|
| 48627 | 4484 | 177,4708 | 3,69662 | 224 | 2,9330357 | 76 |
| 201811 | 16598 | 201,811 | 3,6463093 | 936 | 2,8023504 | 64 |
| 417911 | 22763 | 139,30367 | 3,3187054 | 2815 | 2,8181172 | 185 |
| 135625 | 6384 | 72,217785 | 3,2975207 | 867 | 2,828143 | 2133 |
| 20895 | 2695 | 20,895 | 2,7956432 | 939 | 2,7795527 | 61 |
| 80896 | 2300 | 80,896 | 2,9792746 | 750 | 2,9653333 | 250 |
| 7390 | 732 | 36,95 | 2,7622642 | 188 | 2,5 | 12 |
| 194830 | 20970 | 64,943333 | 3,1816113 | 2883 | 2,5629553 | 117 |
| 357177 | 19173 | 119,059 | 3,2458101 | 2429 | 2,6821737 | 571 |
| 860211 | 74692 | 286,737 | 3,3944737 | 2994 | 2,4752839 | 6 |
| 57191 | 15408 | 19,063667 | 2,9814241 | 2993 | 2,5065152 | 7 |
| 9319 | 7479 | 3,1063333 | 2,4529354 | 2995 | 2,4367279 | 5 |
| 55248 | 14917 | 18,416 | 2,9094987 | 2995 | 2,4851419 | 5 |
| 72329 | 13950 | 24,109667 | 2,8574355 | 2946 | 2,4243041 | 54 |
| 23284 | 7393 | 7,7613333 | 2,4343102 | 2953 | 2,40298 | 47 |
| 68179 | 12367 | 24,2285 | 2,8982892 | 2701 | 2,4590892 | 299 |
| 282449 | 36552 | 100,33712 | 3,3620309 | 2679 | 2,4774169 | 321 |
| 77264 | 12883 | 26,846421 | 2,8578083 | 2759 | 2,4142805 | 241 |
| 69421 | 12813 | 24,062738 | 2,9293553 | 2756 | 2,4698839 | 244 |
| 397531 | 48276 | 139,48456 | 3,3778338 | 2744 | 2,4435131 | 256 |
| 24639 | 6078 | 8,213 | 2,2679104 | 2648 | 2,2537764 | 352 |
| 85477 | 7730 | 28,492333 | 2,7141854 | 1961 | 2,4563998 | 1039 |
| 57337 | 18655 | 19,112333 | 3,0392636 | 2995 | 2,2220367 | 5 |
| 10410 | 7007 | 3,47 | 2,2372286 | 2983 | 2,1293999 | 17 |
| 89154 | 16703 | 29,718 | 3,2489788 | 2830 | 2,5922261 | 170 |
| 9879 | 7185 | 3,293 | 2,2874881 | 2973 | 2,1782711 | 27 |

| T_ZAC | T_KON | IZMENJAVA | PRAZNJENJE | ST SKOKOV MED | ODG MAX | ODG MIN |
|---|---|---|---|---|---|---|
| 1 | 300 | | | 177,4708 | 5 | 1 |
| 1 | 1000 | | | 201,811 | 5 | 1 |
| 1 | 3000 | | | 139,30367 | 5 | 1 |
| 1 | 3000 | | | 72,217785 | 5 | 1 |
| 1 | 1000 | 10 | | 20,895 | 5 | 1 |
| 1 | 1000 | 20 | | 80,896 | 5 | 1 |
| 1 | 200 | 20 | | 36,95 | 5 | 1 |
| 1 | 3000 | | | 64,943333 | 5 | 1 |
| 1 | 3000 | | | 119,059 | 5 | 1 |
| 1 | 3000 | | | 286,737 | 5 | 1 |
| 1 | 3000 | | | 19,063667 | 5 | 1 |
| 1 | 3000 | 10 | | 3,1063333 | 5 | 1 |
| 1 | 3000 | | | 18,416 | 5 | 1 |
| 1 | 3000 | | | 24,109667 | 5 | 1 |
| 1 | 3000 | 10 | | 7,7613333 | 5 | 1 |
| 1 | 3000 | 10 | 100 | 24,2285 | 5 | 1 |
| 1 | 3000 | 10 | 100 | 100,33712 | 5 | 1 |
| 1 | 3000 | 10 | 100 | 26,846421 | 5 | 1 |
| 1 | 3000 | 5 | 50 | 24,062738 | 5 | 1 |
| 1 | 3000 | | 50 | 139,48456 | 5 | 1 |
| 1 | 3000 | 20 | | 8,213 | 5 | 1 |
| 1 | 3000 | | | 28,492333 | 5 | 1 |
| 1 | 3000 | | | 19,112333 | 5 | 1 |
| 1 | 3000 | 20 | | 3,47 | 5 | 1 |
| 1 | 3000 | | | 29,718 | 5 | 1 |
| 1 | 3000 | 20 | | 3,293 | 5 | 1 |

| ODG_MED | R_MIN | R_MAX | R_50 | R_AVG | C_MIN | C_MAX | C_AVG |
|---|---|---|---|---|---|---|---|
| 2,933036 | 1 | 153 | 111 | 92,80292 | 1 | 2,924242 | 1,724592 |
| 2,80235 | 1 | 153 | 124 | 96,332 | 1 | 2,90604 | 1,794056 |
| 2,818117 | 1 | 154 | 87 | 75,503333 | 1 | 4,31579 | 1,536127 |
| 2,828143 | 1 | 153 | 88 | 40,046326 | 1 | 5,051546 | 1,46774 |
| 2,779553 | 1 | 153 | 3 | 11,929 | 1 | 2,101351 | 1,061527 |
| 2,965333 | 1 | 153 | 3 | 41,459 | 1 | 5,517483 | 1,266282 |
| 2,5 | 1 | 154 | 3 | 20,12 | 1 | 2,125874 | 1,12363 |
| 2,562955 | 1 | 154 | 4 | 36,713667 | 1 | 4,776316 | 1,268534 |
| 2,682174 | 1 | 154 | 52 | 64,482 | 1 | 3,609272 | 1,452845 |
| 2,475284 | 1 | 154 | 152 | 145,691 | 1 | 2,052632 | 1,941257 |
| 2,506515 | 1 | 154 | 3 | 12,410667 | 1 | 2,047619 | 1,092123 |
| 2,436728 | 1 | 153 | 2 | 2,776 | 1 | 2,13245 | 1,002597 |
| 2,485142 | 1 | 154 | 3 | 11,998 | 1 | 2,047297 | 1,088922 |
| 2,424304 | 1 | 153 | 3 | 14,615333 | 1 | 3,85124 | 1,109099 |
| 2,40298 | 1 | 153 | 2 | 4,9386667 | 1 | 6,190083 | 1,019723 |
| 2,459089 | 1 | 153 | 3 | 13,343284 | 1 | 6,191177 | 1,107387 |
| 2,477417 | 1 | 154 | 55 | 57,445115 | 1 | 4,233083 | 1,388474 |
| 2,414281 | 1 | 154 | 3 | 14,316887 | 1 | 7,75 | 1,118118 |
| 2,469884 | 1 | 154 | 3 | 13,02461 | 1 | 7,642336 | 1,111255 |
| 2,443513 | 1 | 154 | 91 | 77,277544 | 1 | 2,576389 | 1,511193 |
| 2,253776 | 1 | 21 | 2 | 4,0303333 | 1 | 3,882353 | 1,254824 |
| 2,4564 | 1 | 21 | 18 | 12,710667 | 1 | 4,05 | 1,828717 |
| 2,222037 | 1 | 131 | 3 | 12,460333 | 1 | 2,8 | 1,192455 |
| 2,1294 | 1 | 124 | 2 | 2,7636667 | 1 | 4 | 1,015086 |
| 2,592226 | 1 | 148 | 3 | 21,907667 | 1 | 2,115703 | 1,090273 |
| 2,178271 | 1 | 140 | 2 | 2,8823333 | 1 | 3,241379 | 1,007689 |

| C_50 | M_MIN | M_MAX | M_AVG | M_50 | P_MIN | P_MAX | P_AVG | P_50 |
|---|---|---|---|---|---|---|---|---|
| 1,927419 | 1 | 4459 | 362,8881 | 99 | 0 | 0,658031 | 0,382133 | 0,481172 |
| 2 | 2 | 15286 | 1310,461 | 11 | 0 | 0,655889 | 0,383798 | 0,5 |
| 1,693182 | 8 | 29831 | 2713,708 | 27 | 0 | 0,768293 | 0,29422 | 0,409396 |
| 1,894737 | 1 | 16009 | 941,8403 | 83 | 0 | 0,802041 | 0,268571 | 0,472222 |
| 1 | 1 | 1951 | 135,6818 | 11 | 0 | 0,524116 | 0,031626 | 0 |
| 1 | 2 | 7922 | 525,2987 | 12 | 0 | 0,818758 | 0,131451 | 0 |
| 1 | 1 | 675 | 53,94161 | 18 | 0 | 0,529605 | 0,064853 | 0 |
| 1 | 11 | 13586 | 1265,13 | 27 | 0 | 0,790634 | 0,154706 | 0 |
| 1,444444 | 9 | 28407 | 2319,331 | 28 | 0 | 0,722936 | 0,249757 | 0,307692 |
| 1,967105 | 11 | 83431 | 5585,786 | 26 | 0 | 0,512821 | 0,478201 | 0,491639 |
| 1 | 10 | 3532 | 371,3701 | 26 | 0 | 0,511628 | 0,059593 | 0 |
| 1 | 10 | 829 | 60,51299 | 23 | 0 | 0,531056 | 0,00138 | 0 |
| 1 | 8 | 3280 | 358,7533 | 27 | 0 | 0,511551 | 0,057465 | 0 |
| 1 | 10 | 4433 | 469,6688 | 26 | 0 | 0,740343 | 0,065729 | 0 |
| 1 | 11 | 2224 | 151,1948 | 26 | 0 | 0,838451 | 0,008982 | 0 |
| 1 | 9 | 7439 | 442,7208 | 50 | 0 | 0,83848 | 0,054198 | 0 |
| 1,393939 | 8 | 15837 | 1834,084 | 27 | 0 | 0,763766 | 0,223861 | 0,282609 |
| 1 | 13 | 8016 | 501,7143 | 53 | 0 | 0,870968 | 0,057156 | 0 |
| 1 | 14 | 6047 | 450,7857 | 46 | 0 | 0,86915 | 0,054577 | 0 |
| 1,647059 | 7 | 32939 | 2581,37 | 27 | 0 | 0,61186 | 0,28506 | 0,392857 |
| 1 | 99 | 228 | 159,9935 | 163 | 0 | 0,742424 | 0,083296 | 0 |
| 2,125 | 440 | 694 | 555,0455 | 552 | 0 | 0,753086 | 0,35499 | 0,529412 |
| 1 | 105 | 678 | 372,3182 | 368 | 0 | 0,642857 | 0,120089 | 0 |
| 1 | 19 | 145 | 67,5974 | 64 | 0 | 0,75 | 0,006239 | 0 |
| 1 | 13 | 1949 | 578,9221 | 509 | 0 | 0,527344 | 0,063569 | 0 |
| 1 | 15 | 233 | 64,14935 | 49 | 0 | 0,691489 | 0,004696 | 0 |

| D_MIN | D_MAX | D_AVG | D_50 |
|---|---|---|---|
| 1 | 8 | 2,37931 | 3 |
| 1 | 11 | 2,27234 | 2 |
| 1 | 10 | 2,036551 | 2 |
| 1 | 10 | 1,711426 | 10 |
| 1 | 6 | 1,014846 | 1 |
| 1 | 3 | 1,018617 | 1 |
| 1 | 8 | 1,3125 | 1 |
| 1 | 13 | 2,024931 | 1 |
| 1 | 12 | 2,132456 | 2 |
| 1 | 12 | 5,754837 | 6 |
| 1 | 12 | 1,611333 | 1 |
| 1 | 9 | 1,014005 | 1 |
| 1 | 12 | 1,577719 | 1 |
| 1 | 15 | 1,542062 | 1 |
| 1 | 13 | 1,022004 | 1 |
| 1 | 14 | 1,400591 | 1 |
| 1 | 15 | 3,444362 | 3 |
| 1 | 13 | 1,41763 | 1 |
| 1 | 14 | 1,3874 | 1 |
| 1 | 15 | 4,314835 | 4 |
| 1 | 3 | 1,00566 | 1 |
| 1 | 6 | 1,171167 | 1 |
| 1 | 13 | 1,792321 | 1 |
| 1 | 14 | 1,03282 | 1 |
| 1 | 15 | 1,657839 | 1 |
| 1 | 21 | 1,04266 | 1 |