Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Peter Kacin

Deljenje datotek v porazdeljenem okolju

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: dr. Andrej Brodnik

Ljubljana 2012

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00241/2012
Datum: 02.04.2012

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **PETER KACIN**

Naslov: **DELJENJE DATOTEK V PORAZDELJENEM OKOLJU**
FILE SHARING IN A DISTRIBUTED ENVIRONMENT

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

V sodobnem svetu že dolgo ne obstaja meja med računalnikom in računalniškim omrežjem, ampak imamo opravka z enim samim porazdeljenim okoljem. V tem okolju ima uporabnik opravka s svojimi datotekami, za katere želi, da so mu povsod na voljo. Poleg tega želi svoje datoteke ponuditi drugim uporabnikom na vpogled ali za popravljanje.

Preučite pomen in vlogo izreka CAP in njegovo vlogo pri izdelavi podobnih porazdeljenih sistemov. Poleg tega preverite, kako lahko z matriko dostopnosti in njeno izvedbo seznamom ACL uredite nudenje dostopa do deljenih datotek.

Mentor:
doc. dr. Andrej Brodnik

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:
prof. dr. Nikolaj Zimic

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Peter Kacin, z vpisno številko **63050052**, sem avtor diplomskega dela z naslovom:

Deljenje datotek v porazdeljenem okolju

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom prof. dr. Andreja Brodnika
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI"

V Ljubljani, dne 23. septembra 2012 Podpis avtorja:

Zahvala

Zahvaljujem se mentorju dr. Andreju Brodniku za strokovno pomoč in nasvete pri izdelavi diplomske naloge. Zahvaljujem se vsem zaposlenim na Arnesu, ki so mi omogočili izvedbo praktičnega dela naloge. Rad bi se zahvalil tudi staršem, bratu in sestri ter vsem ostalim, ki ste me podpirali v času študija.

Kazalo

Povzetek

Cilj diplomske naloge je bil implementacija storitve, ki uporabnikom omogoča shranjevanje in deljenje datotek na oddaljenem mestu. Pri implementaciji storitve smo se osredotočili na uporabniško izkušnjo. Za dobro uporabniško izkušnjo je pomembno, da uporabnik lahko upravlja z datotekami tako, kot je vajen z lokalnimi datotekami.

Če celoten sistem, ki omogoča storitev, ni vedno povezan, lahko pride do nedosegljivosti in/ali neusklajenosti datotek. Ta problem je podrobneje obravnavan v okviru izreka CAP, ki pravi, da je nemogoče doseči visoko dosegljivost skupaj z nedeljivo skladnostjo ob prisotnosti delitve omrežja.

Obravnavamo dva načina nadzora dostopa do datotek. Pri seznamu zmožnosti se pravice za dostop do datotek hranijo pri uporabniku, pri seznamu nadzora dostopa pa pri datotekah. Na lokalnih napravah z načini nadzora dostopa v splošnem ni večjih težav. Težave se lahko pojavljajo, ko se povezujejo naprave z različnimi implementacijami le teh. V takšnih primerih je težavno zagotoviti učinkovit nadzor dostopa, ki bo združljiv z obema napravama.

V nalogi opisujemo nekaj primerov rešitev za implementacijo storitve. Nobeden ni bil idealen. Težave so se kazale predvsem pri nadzoru dostopa. Pri rešitvi z uporabo NFS je bila težava v tem, da odjemalec s korenskim dostopom lahko pride do poljubnega UID in tako dostopa do poljubne datoteke. Pri ostalih treh primerih rešitev, sistemi uporabljajo svojo implementacijo nadzora dostopa z izjemo git. Slednji v končni implementaciji ravno tako uporablja svojo implementacijo nadzora dostopa. Obravnavani sistemi

s svojo implementacijo nadzora dostopa omogočajo upravljanje le preko spletnega vmesnika.

Prikazali smo dva primera implementacije. V obeh nam je uspelo zadostiti vsem merilom ustreznosti, ki smo jih definirali. Storitvi, ki sta bili implementirani sta uporabni. Ker naprave nimajo vedno medmrežne povezljivosti, smo izbrali odjemalce, ki vzdržujejo lokalne kopije datotek. Na ta način smo dosegli dosegljivost datotek tudi v primeru delitve omrežja. Pri tem smo se morali sprijazniti z določeno stopnjo neusklajenosti datotek, ki se lahko pojavi pri delu brez povezave. Ko je sistem povezan, je mogoče dosegati tudi dobro usklajenost datotek. Na koncu še vedno ostaja odprto vprašanje, kako upravljati s pravicami dostopa iz datotečnega sistema naprave uporabnika in ne posredno preko ločenega mehanizma.

Ključne besede:

Deljenje datotek, porazdeljen sistem, izrek CAP, nadzor dostopa, uporabniška izkušnja

Abstract

The goal of this diploma thesis was to implement a service, which allows users to save and share data at a remote location. In the implementation, we focused on user experience. For a good user experience, it is very important that user can manage files in the same way as he is accustomed to dealing with local files.

If a whole system allowing a service is not fully connected, unavailability and/or inconsistency may appear. That problem is further discussed in the context of CAP theorem, which says, that it is impossible to reach high availability together with atomic consistency in the presence of network partitions.

We discuss two ways of file access control. Rights for file access are stored together with a user as a capability list and with files as an access control list. Access control mechanisms on local devices don't cause many problems. However, problems may appear, when different devices using different implementations of access control mechanisms connect among themselves. In such cases it is difficult to assure effective access control that would be compatible with both devices.

We describe some examples of service implementation solution, but none of them is ideal. Problems mainly occur in access control mechanisms. In the NFS solution, client with root access can get arbitrary UID, which makes it capable to access any file. Other three solutions use their own implementation of access control mechanism with the exception of Git. The latter in final implementation also uses its own implementation of access control

mechanism. Discussed systems using their own implementation of access control mechanism only allow management through a web interface.

We present two examples of service implementation. In both, we succeed meeting all suitability criteria as we defined them. Services that we implemented are useful. We have chosen clients that maintain local copies of files, since they don't always have internet connectivity. This way we made files available even during network partition. However, we had to accept a possibility for some degree of inconsistency, which may appear while using service. In case of full connectivity, we can also achieve good consistency of files. At the end there still remains one important question, how to manage access rights from the file system of user device and not indirectly, using separate mechanism.

Keywords:

File sharing, distributed system, CAP theorem, access control, user experience

Poglavje 1

Uvod

Pri problemu dostopa do datotek iz različnih naprav in njihovemu deljenju s poljubnimi napravami oziroma uporabniki lahko govorimo o dveh možnih rešitvah. Prva rešitev je dodelitev neposrednega dostopa do datotek, druga pa je prenos datotek na drugo napravo in dodelitev dostopa do podatkov preko te naprave, torej posreden dostop. Da je deljenje datotek sploh možno, morata vozlišči, ki si želita deliti datoteke, delovati in biti povezani.

Pri napravah, s katerimi dandanes rokujejo uporabniki, to ni vedno mogoče. Mobilnim napravam naprimer lahko zmanjka energije za delovanje (prazna baterija), lahko izgubijo povezljivost z baznimi postajami ali drugimi dostopnimi točkami. Podobne težave imajo stacionarne naprave. Lahko pride do izpada medmrežja ali električnega omrežja, uporabniki jo lahko zaradi neuporabe izključijo, lahko se pojavijo okvare, ki so navadno pogostejše pri nenamenskih napravah, ki niso namenjene neprestanemu delovanju. Zaradi velikih možnosti za izpad uporabniških naprav, je smiselno podatke podvajati na napravo, ki je skoraj vedno prižgana in ima skoraj vedno povezljivost v medmrežje. Beseda skoraj v prejšnji povedi je na mestu, ker je tudi strežnik, ki najbolj ustreza opisu, realna naprava in kot taka podvržena napakam in izpadom. Tukaj lahko govorimo o 99,999% (*five nines* [14]) ali celo o 99,99999% dosegljivosti.

Pri zasnovi sistema se bomo zaradi čimboljše dosegljivosti osredotočili na

rešitve, pri katerih je osnovna kopija datotek na strežniku. S podvojevanjem podatkov na zunanjo napravo, so datoteke za druge uporabnike dostopne tudi, ko uporabnik ni povezan v medmrežje, torej pri delitvi omrežja. V sistemu, pri katerem so vozlišča uporabniške naprave, ki nimajo vedno povezljivosti, je to kar pogost pojav, ki ga je potrebno pri zasnovi sistema upoštevati. S podvajanjem podatkov se pojavi dodatno vprašanje in sicer kako se bo razreševala neusklajenost podatkov. Če se podatki spremenijo na eni kopiji, bi se to za popolno usklajenost moralo takoj poznati tudi na vseh ostalih, kar je v realnem svetu nemogoče že zaradi omejene hitrosti potovanja informacij.

Poleg vprašanja z dosegljivostjo, se pojavi še vprašanje z nadzorom dostopa do datotek. Uporabniki večine datotek ne želijo deliti s širšo javnostjo, zato je potrebno pri zasnovi sistema, med drugimi, rešiti vprašanje upravljanja z dostopom do datotek.

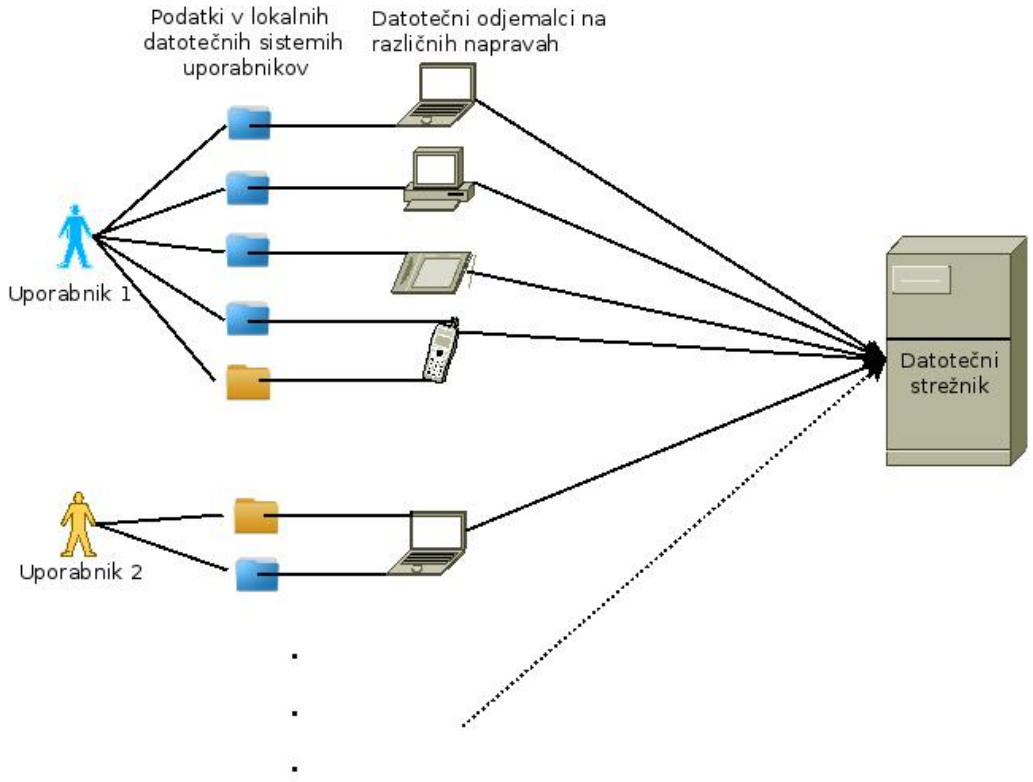
Poglavje 2

Definicija problema

Ko dostopamo do podatkov lokalno, se navadno ne sprašujemo o dosegljivosti. Dosegljivost je lahko vprašljiva, če dostopamo do oddaljenih podatkov. Če je namreč povezava z oddaljenim mestom, kjer se nahajajo podatki, prekinjena, nam postanejo nedosegljivi. Dosegljivost je mogoče povečati s podvajanjem podatkov. V primeru izpada povezave do enega vozlišča s kopijo podatkov, lahko še vedno dostopamo do drugih vozlišč, ki hranijo kopijo enakih podatkov. Če dostopamo do kopije, ki podatkov še nima sinhroniziranih z ostalimi kopijami, lahko pride do neusklajenosti. Pri dostopu do oddaljenih podatkov je potrebno rešiti tudi problem dodeljevanja pravic dostopa do podatkov. Uporabniki namreč uporabljajo naprave z različnimi implementacijami nadzora dostopa, ki med seboj niso vedno združljive. Za uporabnike pa je predvsem pomembno, da lahko z datotekami lahko enostavno upravljajo.

2.1 Usklajenost podatkov

Večina spletnih storitev, še posebno baze podatkov, težijo k usklajenosti podatkov. S tem namenom je bil raziskan in razvit model ACID ([2]). Pri modelu ACID je pomembna nedeljivost operacij (*Atomicity*), skladnost podatkov s predhodno zaključenimi operacijami (*Consistency*), izolacija neizvršenih operacij (*Isolation*) in obstojnost izvršenih operacij (*Durability*).



Slika 2.1: Skica sistema.

Za nekatere storitve je visoka stopnja usklajenosti podatkov glavni cilj. Pri bančnih transakcijah naprimer lahko pride do precej škode, če podatki niso usklajeni ([2]).

Če v sistemu obstaja le ena različica datotek, do neusklajenosti datotek ne prihaja. Nevarnost neusklajenosti se pojavi, ko se podatki podvajajo na več mestih. Če se spremenijo podatki na enem mestu in se spremembe ne prenesejo na ostala mesta, kjer so podvojeni, pravimo, da so podatki neusklajeni. V primeru sistema, ki ga želimo implementirati, lahko pride do neusklajenosti datotek, če odjemalci vzdržujejo lokalne kopije. V tem primeru lahko na odjemalce gledamo tudi kot na vozlišča sistema, pri katerem je v primeru odsotnosti medmrežne povezave prisotna delitev omrežja na dve ali več množic, ki med seboj ne morejo komunicirati. Če v takem stanju sistema pride do spreminjanja datotek, se lahko pojavi neusklajenost podatkov. To v praksi pomeni, da se lahko v različnih množicah vozlišč, ki med seboj ne morejo komunicirati, nahajajo različne množice in različice datotek. Za uporabnika to vsekakor ni dobra novica, zato je pri implementaciji sistema pomembno, da so te razlike čimmanj opazne. Če je le mogoče, je potrebno sistem zastaviti tako, da lahko vsa ta neskladja pri različicah datotek razrešuje sistem brez intervencije uporabnika.

2.2 Dosegljivost in latenca

Od spletnih storitev se pričakuje visoko dosegljivost. Vsaka zahteva bi se morala izvršiti in prejeti odgovor. Cilj spletnih storitev dandanes je tolikšna dosegljivost, kolikor je zanesljivo omrežje, v katerega so povezane ([16]).

Dosegljivost je ena od najbolj zaželjenih lastnosti storitve, ki jo skušamo realizirati. Za doseganje dosegljivosti storitve je pomembno, da je vsaj del sistema povezan. Idealno bi seveda bilo, da bi bila vsa vozlišča delujoča in povezana. S tem bi zagotovili dosegljivost sistema, lahko pa tudi usklajenost podatkov. Kljub temu, da je mogoče zagotoviti dosegljivost sistema, ima to lahko za posledico povečano latenco. Če od sistema zahtevamo visoko stopnjo

usklajenosti, je latenca lahko zelo velika in se tipično povečuje z večanjem medsebojne oddaljenosti vozlišč, kar je povezano z omejeno hitrostjo potovanja same informacije o spremembi. Tudi samo izvajanje algoritma, ki nam to zagotavlja, povzroča dodatno latenco.

Z zmanjšano zahtevano stopnjo usklajenosti podatkov je mogoče zmanjšati tudi latenco in v primeru delitve omrežja ohraniti dosegljivost. Najboljšo dosegljivost in najmanjšo latenco lahko dosežemo z uporabo lokalne kopije datotek. V tem primeru so datoteke uporabniku dosegljive vedno, medtem ko je latenca tipično takšna, kot je latenca dostopa do ostalih lokalnih datotek. Slaba stran lokalne kopije je, da s tem povečamo možnost pojava neusklajenosti datotek, še posebno pri izpadu medmrežne povezave oziroma delitvi omrežja.

2.3 Odpornost na delitev omrežja

Pri porazdeljenih sistemih je logično pričakovati tudi določeno stopnjo odpornosti na napake. V primeru izpada nekaterih vozlišč ali povezav med njimi je pomembno, da storitev še vedno deluje. Ena od želenih lastnosti porazdeljenega sistema je odpornost na delitev omrežja. V nekaterih primerih lahko modeliramo tudi izpad vozlišča sistema, kot posebno delitev omrežja, pri kateri je to vozlišče ločeno od ostalega sistema ([16]).

Pri odpornosti na delitev omrežja gre za sposobnost delovanja storitve v primeru razpada omrežja med vozlišči sistema na dve ali več množic, ki medsebojno ne morejo komunicirati. V takih trenutkih storitev navadno ne deluje s polno funkcionalnostjo, kljub temu pa lahko zagotovimo določeno stopnjo dosegljivosti in usklajenosti datotek. Če imamo enega odjemalca, ki dostopa do enega odložišča na strežniku, gre v primeru izpada povezave med njima za delitev omrežja. Pri tem se nam lahko zgodita dve skrajni možnosti. Če je odjemalec implementiran tako, da podatke bere neposredno s strežnika, so v primeru delitve omrežja datoteke nedosegljive. So pa zato podatki usklajeni, saj obstaja samo ena različica podatkov, ki bo dosegljiva,

ko delitve omrežja ne bo več. Druga skrajnost je lokalna kopija datotek. V primeru lokalne kopije datotek bodo le te vedno dosegljive. Če med nastopom delitve omrežja datoteke spreminjamo, se pojavi neusklajenost datotek med lokalnim odložiščem in tistim, ki se nahaja na strežniku.

2.4 Upravljanje z dostopom

Pri zasnovi sistema se poraja vprašanje upravljanja z dostopom do datotek. Poleg možnosti dodelitve javnega dostopa bi radi imeli možnost dodelitve dostopa za specifičnega uporabnika ali skupino uporabnikov. Poleg tega bi radi uvedli več nivojev dostopa. Zanima nas predvsem dostop za branje datotek in dostop za pisanje v datoteke. Sistemi za upravljanje z dostopom navadno poznajo še nivo dostopa za izvrševanje datoteke ([35, str. 434-435]). Te vrste dostopa za izvedbo deljenja sicer ne bi potrebovali. Morda je zanimiva zgolj uporaba pri posebnih datotekah — mapah za pravice izpisa vsebine in dostopa do vsebine mape. S stališča uporabnika je to sicer nepomembno, važno je le, da lahko poljubnemu uporabniku dodeli dostop za branje in/ali dostop za pisanje. Obstajajo še druge vrste pravic (npr. lastništvo, možnost dodeljevanja pravic ipd.). V tem delu jih ne bomo obravnavali, ker njihova definicija ni enotna med različnimi operacijskimi sistemi.

Eden najbolj razširjenih načinov za upravljanje dostopa je ACL (*Access Control List*). Pri modelu ACL je vsakemu objektu pripet seznam dovoljenj dostopa. Vsak ACE (*Access List Entry*) zapis na seznamu je sestavljen iz osebka in operacij, ki mu jih je nad tem objektom dovoljeno izvajati. Večina današnjih operacijskih sistemov podpira uporabo načina ACL. Kot rečeno, mi se bomo osredotočili na operaciji branja in pisanja.

2.5 Uporabniška izkušnja

Z uporabniškega stališča je najpomembnejše, da je dostop do datotek mogoč kadarkoli in kjerkoli. Zaradi te zahteve bodo podatki prestavljeni na strežnik,

ki je skoraj vedno dosegljiv (glej sliko 2.1). Posledično bodo skoraj vedno dosegljive datoteke, če le ima odjemalec povezavo z medmrežjem.

Da bo upravljanje z datotekami za uporabnika čimbolj preprosto, je smiselno oddaljene datoteke vključiti v lokalni datotečni sistem in tako omogočiti delo z njimi na enak način, kot je uporabnik vajen pri lokalnih datotekah. Ker pri uporabniških napravah ni nenavadna uporaba brez medmrežne povezave, bi bilo dobro zagotoviti, da so datoteke še vedno vidne, tudi ko ni povezave. Še bolje bi bilo zagotoviti tudi možnost njihovega spreminjanja. To sicer povzroči težave z neusklajenostjo datotek, ki naj bo za uporabnike čimmanj opazna.

Za dobro uporabniško izkušnjo bo potrebno rešiti še problem deljenja oziroma upravljanja dostopa do datotek. Najelegantnejša rešitev bi bila, da bi lahko dostop dodeljevali kar v lokalnem upravljalniku datotek.

Uporabnikom bi radi omogočili uporabo storitve s čimveč različnimi napravami, ki jih poganjajo različni operacijski sistemi. Poskusili bomo implementirati sistem, za katerega obstajajo odjemalci, ki se lahko namestijo na najbolj uporabljenih namiznih, kot tudi mobilnih operacijskih sistemih. Za lažjo predstavo o arhitekturi sistema, ki ga nameravamo implementirati je v pomoč Slika 2.1.

Poglavje 3

Teoretično ozadje

V prejšnjem poglavju so bile definirane lastnosti, ki bi jih bilo dobro dosegati pri implementaciji storitve. Poleg tega je bil predstavljen tudi uporabniški pogled na storitev, ki je ravno tako zelo pomemben vidik pri zasnovi sistema. Za uspešnost storitve je namreč zelo pomembna dobra uporabniška izkušnja, ki ima pozitiven učinek na število uporabnikov storitve. V nadaljevanju sledi predstavitev teoretičnega ozadja, ki omogoča oziroma omejuje implementacijo željenih lastnosti. Kot je razvidno iz prejšnjega poglavja, izgledajo usklajenost kopij podatkov, njihova dosegljivost, latenca in odpornost storitve na delitev omrežja med seboj precej povezane.

3.1 Izrek CAP in latenca

Na simpoziju o principih porazdeljenega računanja (PODC) v Portlandu leta 2000 je Eric Brewer podal predpostavko, da je za spletno storitev nemogoče hkrati zagotavljati naslednje tri lastnosti ([16]):

- usklajenost,
- dosegljivost in
- odpornost na delitev omrežja.

Vse tri lastnosti so zaželjene in velikokrat tudi pričakovane od realnih spletnih storitev ([16]). Najprej si poglejmo, kaj razumemo pod vsakem od teh treh terminov.

Najbolj naravna predstavitev ideje o usklajeni storitvi je z nedeljivim podatkovnim objektom. Pri tej vrsti usklajenosti mora obstajati popolen vrstni red izvrševanja operacij, tako, da izgleda, kot da bi bile vse operacije izvršene v trenutku. Pomembna lastnost nedeljivega za branje in/ali pisanje deljenega pomnilnika je, da vsaka operacija branja, ki se začne po končani operaciji pisanja, vrne vrednost sistema, ki je bila definirana z zadnjim pisanjem ([35, str. 707-733]).

Da bi bil porazdeljen sistem zvezno dosegljiv, mora vsaka prejeta zahteva z delujočega vozlišča prejeti odgovor. V nekaterih pogledih je to definicija šibke dosegljivosti, saj nikjer ni določeno, koliko časa ima ima storitev na voljo za odgovor. Po drugi strani pa, ko v sistemu dopuščamo delitev omrežja, lahko na definicijo gledamo kot definicijo visoke dosegljivosti. Tudi ob izpadu določenih povezav v sistemu, se mora vsaka zahteva zaključiti ([2, 16]).

Pri odpornosti na delitev omrežja je pomembno, da storitev deluje tudi v primeru izpadov določenih povezav med vozlišči. Še več, v primeru delitve omrežja, le to razpade na dve ali več podmnožic vozlišč, ki medsebojno ne morejo komunicirati, saj se vsa medsebojna sporočila izgubijo. Nedeljiv deljen pomnilnik zahteva, da so vse operacije nedeljive, kljub temu, da se poljubno število sporočil izgubi, dosegljivost storitve pa zahteva, da vsaka zahteva prejme odgovor. Nobena napaka, ki je manjša od odpovedi celotnega omrežja, ne sme povzročiti napačnega odziva sistema ([2, 16]).

3.1.1 Asinhrona omrežja

Pri asinhronem modelu omrežja ni prisotne ure, vozlišča se lahko odločajo le na podlagi prejetih sporočil in lokalnih izračunov.

Seth Gilbert in Nancy Lynch sta dokazala [16] izrek CAP:

Izrek 3.1 *Pri modelu asinhronega omrežja je nemogoče implementirati podatkovni objekt za branje in pisanje, ki zagotavlja naslednje lastnosti:*

- *dosegljivost in*
- *nedeljivo skladnost*

v vseh izvedbah poštenega algoritma (vključno s tistimi, pri katerih so sporočila izgubljena).

Dokaz. Trditev lahko dokažemo s protislovjem. Denimo, da obstaja algoritem A, ki zadošča trem kriterijem: nedeljivi skladnosti, dosegljivosti in odpornosti na delitev omrežja. Zamislimo si izvedbo algoritma A, pri kateri obstaja zahteva, ki vrne neusklajen odgovor. Privzemimo, da je omrežje sestavljeno iz vsaj dveh vozlišč. Ta so razdeljena v dve disjunktivni, neprazni množici: $\{G_1, G_2\}$. Glavna ideja dokaza je predpostavka, da se vsa sporočila med G_1 in G_2 izgubijo. Če se nato zgodi pisanje v G_1 in se kasneje zgodi pisanje v G_2, potem operacija branja ne more vrniti rezultata predhodne operacije pisanja ([16]).

□

Težava, ki se kaže v asinhronih omrežjih je v tem, da posamezno vozlišče nima informacije o izgubljenih sporočilih, tako se ne zaveda niti delitve omrežja. Posamezno vozlišče nima podatka, da je nekje v sistemu prišlo do pisanja, dokler o tem ne prejme obvestila. Iz tega sledi, da pri asinhronih omrežjih sploh ni mogoče zagotavljati nedeljive usklajenosti v primeru da obstaja več kopij podatkov.

Kljub temu, da vseh treh želenih lastnosti skupaj pri asinhronih omrežjih ne moremo zagotoviti, je mogoče zagotoviti [16] katerekoli ostali dve :

- **usklajenost in odpornost na delitev omrežja:** če ne zahtevamo dosegljivosti, je zagotavljanje usklajenosti in odpornosti na delitev omrežja lahko doseči. Sistem, ki ignorira vse zahtevke ustreza temu.
- **usklajenost in dosegljivost:** v odsotnosti delitve omrežja je možno doseči tako usklajenost, kot dosegljivost. Algoritem s centraliziranim shranjevanjem podatkov naprimer ustreza temu primeru.

- **dosegljivost in odpornost na delitev omrežja:** če usklajenost ni pomembna, potem lahko sistem vrne začetno vrednost vozlišča in s tem zadosti zahtevi o dosegljivosti in odpornosti na delitev omrežja. Vseeno lahko tudi pri takem sistemu zagotovimo šibko usklajenost s tem, da vozlišče vrne zadnjo zapisano vrednost, kar je navadno bolje od začetne vrednosti.

3.1.2 Delno sinhrona omrežja

Iz zaključkov v prejšnjem poglavju se zdi, da vseh želenih lastnosti storitve ne bo mogoče doseči. Še največ optimizma nam dopušča dejstvo, da večina omrežij v realnem svetu ni popolnoma asinhronih. Spletne aplikacije in storitve za prenos podatkov večinoma uporabljajo protokol TCP ([23]), ki je povezavni protokol. Z vzpostavitvijo povezave omogoča zanesljiv prenos podatkov med napravama. Storitve, pri katerih je pomemben zanesljiv prenos, ne delujejo najbolje v asinhronih omrežjih. Težava v tem, da vozlišče ne more zaznati delitve omrežja in izgubljenih paketov. Če vozliščem dodamo ure, ki se povečujejo z enako frekvenco, je mogoče implementirati algoritem za zaznavo delitve omrežja, kot tudi izgube paketov. Preprost algoritem za to je naprimer, da za sporočilo določimo čas v katerem mora biti potrjena dostava, če v tem času ni potrditve, privzamemo, da je sporočilo izgubljeno ([16, 23]).

Kljub dodanim uram, s čimer vozlišča lahko dobijo več informacij o stanju sistema, sta avtorja dokaza za izrek CAP v asinhronem omrežju, dokazala, da velja [16] tudi v delno sinhronih omrežjih :

Izrek 3.2 *Pri modelu delno sinhronega omrežja je nemogoče implementirati podatkovni objekt za branje in pisanje, ki zagotavlja naslednje lastnosti:*

- *dosegljivost in*
- *nedeljivo skladnost*

v vseh izvedbah (tudi teh, pri katerih so sporočila izgubljena).

Dokaz. Dokaz je podoben tistemu iz izreka 3.1. Sledili bomo enaki metodologiji: Razdelimo omrežje na dve komponenti $\{G_1, G_2\}$, zamislimo si dopustno izvedbo algoritma A, pri kateri se zgodi pisanje v eno komponento, čemur sledi branje iz druge komponente. To branje ne more vrniti usklajenih podatkov ([16]).

□

Kljub temu, da vseh treh lastnosti skupaj ni mogoče doseči, je v primeru, ko so vsa sporočila dostavljena, mogoče doseči dosegljivost in nedeljivo usklajenost. Tudi v primeru izgube sporočil, lahko vozlišče vrne zadnjo različico podatkov, ki jo ima. Če delitev omrežja ni dolgotrajna, so tudi nekoliko starejši podatki še vedno lahko uporabni. S tega vidika lahko govorimo o stanju šibke usklajenosti, ki dovoljuje tudi streženje nekoliko zastarelih podatkov. Vozlišča storitve, pri katerih je dovoljena šibka usklajenost, se medsebojno usklajujejo, vendar lahko na vsako vozlišče pišemo ali z njega beremo tudi, ko podatki niso popolnoma usklajeni. Zaradi delitve omrežja ali zaradi samega principa delovanja storitve, ki lahko zaradi zmanjšanja latence dovoljuje nekoliko šibkejšo usklajenost podatkov ([1, 2, 16, 17]).

Delitev omrežja je navadno razmeroma redek pojav. Ko delitve omrežja ni, lahko zagotovimo tako dosegljivost, kot nedeljivo usklajenost. Izrek CAP je bil v preteklosti velikokrat narobe razumljen, kot omejitev, da lahko v sistemu dosežemo le dve od treh želenih lastnosti. Precej razširjeno mišljenje je, da se delitvam omrežja ni mogoče izogniti, zato je potrebno žrtvovati ali dosegljivost ali usklajenost. Seveda je to nujno ko se pojavi delitev omrežja, medtem ko je pri vseh ostalih primerih v delno sinhronem omrežju mogoče doseči nedeljivo usklajenost skupaj z doseglivostjo. Če je zasnova sistema takšna, da so delitve omrežja redke, je lahko storitev večino časa dosegljiva in nedeljivo usklajena ([1, 2]).

Načrtovalci sistemov se kljub temu dostikrat odpovedujejo nedeljivi usklajenosti tudi, ko delitve omrežja ni. Glavni razlog je bržkone zagotavljanje manjše latence. Izrek CAP namreč v odsotnosti delitve omrežja ne omejuje ne dosegljivosti, niti usklajenosti. Povečevanje latence je povezano z zagota-

vljanjem nedeljive usklajenosti in se povečuje z večjo razpršenostjo oziroma medsebojno oddaljenostjo vozlišč. Zaradi omejene hitrosti potovanja informacije, algoritem za zagotavljanje boljše ali celo nedeljive usklajenosti potrebuje nekaj časa za izvajanje. To zakasni odgovor vozlišča in se uporabniku storitve odraža kot povečanje latence ([1, 2]).

Zastavlja se vprašanje zakaj je podvajanje podatkov sploh potrebno. Pri zagotavljanju visoke dosegljivosti, je potrebno upoštevati tudi možnost izpada vozlišča ali delitve omrežja. To samo po sebi vodi v potrebo po določeni stopnji podvajanja podatkov. Tukaj velja izpostaviti razliko tega kompromisa, ki je posledica same možnosti za izpad vozlišča od kompromisa, kot posledice izreka CAP, ki nastopi, ko je delitev omrežja že prisotna. Za zagotavljanje čimvišjega možnega nivoja dosegljivosti, je dobro podatke podvajati tudi izven podatkovnega centra, kar povečuje možnost za dosegljivost ob izpadu celotnega podatkovnega centra ([1]).

3.1.3 Podvajanje podatkov

Za podvajanje podatkov obstajajo trije načini: posodobitve so poslane (i) na vsa vozlišča hkrati, (ii) najprej na vnaprej določeno glavno vozlišče in (iii) najprej na poljubno vozlišče. Vse tri možnosti so lahko implementirane na različne načine, pri vseh pa je potreben kompromis med latenco in usklajenostjo. V naslednjih rezdelkih si bomo podrobneje ogledali vsakega od treh načinov ([1]).

Posodobitve se pošljejo vsem vozliščem sočasno

V sistemu, pri katerem se pošljejo posodobitve vsem vozliščem hkrati, obstaja velika nevarnost neusklajenosti. Težava je v tem, da se lahko različna vozlišča posodobijo v različnem vrstnem redu. Problem je še bolj viden pri sočasnih posodobitvah. V takem sistemu brez dodatnih mehanizmov ni mogoče doseči nedeljive usklajenosti. Problem neusklajenosti je mogoče rešiti s predobdelavo posodobitev ali s protokolom, s katerim se vozlišča dogovorijo o vrstnem redu posodobitev. Pri obeh načinih se pri sistemu poveča latenca. Pri prvem

načinu je izvor latence preusmeritev posodobitev na na dodatno komponento, v drugem latenco poveča sam protokol za dogovor ([1]).

Posodobitve se najprej izvedejo na glavnem vozlišču

Drugi način za podvajanje je vpeljava glavnega vozlišča, ki se posodobi prvo, šele nato se posodobijo ostala vozlišča. Glavno vozlišče v tem primeru obdeluje in razrešuje vse zahteve za pisanje. Ko jih obdela, jih pošlje še ostalim vozliščem. V vrstnem redu, ki ga glavno vozlišče določi, se nato posodobijo. Za podvajanje na ostalih vozliščih imamo tri možne implementacije ([1]):

- **sinhrono podvajanje:** pri sinhronem podvajanju glavno vozlišče pošlje podatke ostalim in počaka, dokler vsa vozlišča ne potrdijo posodobitve. Čeprav ta način ohranja usklajenost podatkov, se lahko pojavi precej velika latenca. Sploh v primeru, ko se vozlišča ne nahajajo v istem podatkovnem centru, saj je latenca pogojena z vozliščem, ki se najpočasneje posodobi.

- **asinhrono podvajanje:** posodobitve obravnavamo, kot da so že izvršene, preden se uveljavijo. V takem sistemu ni zagotovila, da so bile posodobitve razširjene po sistemu. Različne implementacije se različno spopadajo s kompromisom med usklajenostjo in latenco:

  - če se vsa branja izvajajo na glavnem vozlišču, ni težav z usklajenostjo. Se pa pojavljajo različni viri latence. Glavno vozlišče je lahko dosti bolj oddaljeno, kot najbližje vozlišče. Težava se pojavi tudi pri izpadu ali preobremenjenosti glavnega vozlišča, saj je branje onemogočeno ali zakasnjeno, dokler glavno vozlišče ne obratuje normalno.
  - če dovolimo branje s kateregakoli vozlišča, se latenca lahko precej zmanjša. Je pa zato večja nevarnost neusklajenosti, saj imajo lahko vozlišča različne kopije podatkov, če se posodobitve še niso razširile po celotnem sistemu. Zmanjšano usklajenost je mogoče

omejiti z uporabo zaporednih številk posodobitev in implementacijo sekvenčne usklajenosti.

v obeh primerih je lahko povečana latenca pri pisanju, saj je lahko glavno vozlišče precej oddaljeno.

- **kombinacija sinhronega in asinhronega podvajanja:** posodobitve se razpošljejo sinhrono določenemu številu vozlišč, ostalim pa asinhrono. Tudi tukaj sta stopnja usklajenosti in latenca povezani z implementacijo. Pogosto se pri teh sistemih uporablja protokol sklepčnosti, pri katerem poteka sinhrono posodabljanje na W vozliščih, branje pa z R vozlišč, iz katerih se izbere najnovejša različica. Če je sistem nastavljen na $R + W > N$, pri čemer je N število vseh vozlišč, se branje zgodi iz vsaj enega vozlišča, ki je bilo posodobljeno na sinhron način. V tem primeru težav z neusklajenostjo ni. Je pa latenca pogojena z odzivom najpočasnejšega vozlišča, ki je udeleženo pri branju. Če je sistem nastavljen na $R + W \leq N$, se zaradi manj udeleženih vozlišč tipično latenca zmanjša. V takšni nastavitvi se lahko ponovno pojavi neusklajenost.

Posodobitve se izvajajo na poljubnih vozliščih

Pri tej vrsti podvajanja se najprej posodobitev zgodi na poljubnem vozlišču, šele nato se razširi na ostala vozlišča. Lahko se zgodi tudi posodobitev istega podatkovnega objekta na dveh različnih vozliščih hkrati. Kompromis med usklajenostjo in latenco je zopet odvisen od implementacije ([1]):

- **podvajanje na ostala vozlišča poteka sinhrono:** s sinhronim podvajanjem je zagotovljena dobra usklajenost. Težave z latenco so podobne, kot pri sinhronih posodobitvah z glavnega vozlišča, dodatno latenco lahko tukaj povzroča odkrivanje in razreševanje sočasnih posodobitev.

- **asinhrono podvajanje:** latenca je lahko v tem primeru minimalna,

pojavljajo pa se težave z neusklajenostjo, saj nimamo nobenega zagotovila, kdaj se bo na katerem vozlišču zgodila posodobitev.

Pri vsakem podvajanju podatkov naletimo na potrebo po kompromisu med usklajenostjo in latenco. Če se vozlišča sistema nahajajo v istem podatkovnem centru, je vpliv na latenco znatno manjši, tako da zaradi sprejemljive latence mogoče ni potrebno žrtvovati usklajenosti. Če pa podatke podvajamo čez globalno omrežje, se kompromisu med usklajenostjo podatkov in latenco zelo težko izognemo. Zmanjšana usklajenost v teh primerih ni posledica omejitve izreka CAP, saj zmanjšujemo usklajenost, da bi zagotovili manjšo latenco. Navadno pa so sistemi z zmanjšano zahtevano usklajenostjo bolj odporni tudi na delitev omrežja ([1]).

3.1.4 Spopadanje z delitvijo omrežja

Pri delitvi omrežja ni globalnega obvestila o delitvi omrežja, pač pa ga mora vsako vozlišče zaznati samo. Lahko se zgodi celo, da nekatera vozlišča zaznajo delitev, druga pa ne. Navadno se delitev omrežja zazna tako, da po tem, ko vozlišče na zahtevo ne prejme odgovora drugega vozlišča, po nekem pretečenem času zaključi, da je delitev prisotna. Za sklepanje dobrega kompromisa med dosegljivostjo in usklajenostjo je zaznava delitve ključnega pomena. Poleg zaznave delitve in s tem proženje drugačnega načina delovanja, je pomemben tudi postopek za regeneracijo po tem, ko je komunikacija med vozlišči ponovno mogoča ([2]).

Ko sistem zazna delitev, ima dve možnosti. Lahko omeji določene operacije, s čimer zmanjša dosegljivost, druga možnost pa je, da shranjuje dodatne informacije o operacijah, kar mu bo kasneje koristilo pri regeneraciji od delitve omrežja. Načrtovalec sistema se mora odločiti, ali omejiti operacije, ali tvegati, da bo z njimi povzročil neusklajenost, z namenom, da jo ob odsotnosti delitve omrežja obnovil. Če se podatki lahko enostavno združijo naprimer, je obnovitev usklajenosti preprosta.

Posebna obravnava pri pojavu delitve omrežja postavlja raziskovalcem

nove izzive pri raziskovanju izvrševanja operacij brez povezave. Na delo brez povezave lahko gledamo kot na daljšo delitev omrežja.

Ko delitev omrežja ni več prisotna, je čas za razreševanje morebitnih neskladij, ki so nastala. Sistem v tem trenutku pozna trenutno stanje in zgodovino. Zgodovina je pomembnejša, saj je iz nje razvidno, katere operacije so kršile usklajenost. Iz teh podatkov je potrebno popraviti stanji na obeh straneh v pravilno usklajeno stanje. Pri večini sistemov nekaterih neskladij sistem ne more združiti. Pri takšnih sistemih je navadno potreben poseg uporabnika. Spet pri drugih lahko sistem sam razrešuje neskladja z izborom omejenega nabora operacij, ki jih dovoljuje med delitvijo omrežja. Z zakasnitvijo določenih operacij, ki bi lahko povzročale nerazrešljiva neskladja, je to razmeroma enostavno doseči. Za omogočanje samodejnega razreševanja neskladij, je najbolje dovoliti le zamenljive operacije, to je takšne, katerih vrstni red ne spremeni končnega rezultata. Druga možnost je med delitvijo omrežja dovoliti le operacije, ki monotono povečujejo stanje tako, da je ob razreševanju neskladij končno stanje unija obeh množic ([1, 31]).

3.2 Matrika dostopnosti

V sistemih, v katerih imamo različne osebke, ki dostopajo do virov, se navadno pojavlja potreba po zaščiti dostopa. V primeru te naloge, so osebki uporabniki, viri pa njihove datoteke. Če bi bili vsi uporabniki pošteni in nezmotljivi ter dostopali le do svojih datotek in datotek ostalih uporabnikov, ki bi jim to dovolili (npr. ustno), potrebe po zaščiti dostopa ne bi bilo. V realnosti poštenosti vseh in celo ob vseh priložnostih ni pričakovati. Zato se kaže izrazita potreba po mehanizmu, s katerim lahko zaščito dostopa vgradimo že v sam sistem. Z mehanizmom želimo preprečiti oziroma vsaj omejiti škodo, ki si jo lahko uporabniki med seboj povzročijo. Škoda nastane lahko na različne načine. Že samo branje ali kopiranje datotek lahko povzroči nekomu škodo, če te niso namenjene javnosti. Škoda je lahko še večja, če se pobrišejo ali spremenijo datoteke, ki temu niso namenjene. V skrajnem pri-

meru lahko uporabniki s svojim ravnanjem povzročijo tudi omejeno delovanje ali celo sesutje sistema ([15]).

Ideja modela sistema za upravljanje dostopa do virov je v vpeljavi različnih okolij oziroma domen, ki imajo različne pravice dostopa do virov. V našem primeru do virov dostopajo uporabniki, zato potrebujemo vsaj za vsakega uporabnika svojo domeno, lahko pa je domen tudi več. Pri najbolj preprosti rešitvi problema dostopa ima vsak uporabnik svojo domeno, medtem ko so njegove datoteke dostopne samo v tej domeni. Vsak uporabnik torej lahko dostopa le do svojih datotek. Cilj naloge je v omogočanju dostopa do lastnih datotek tudi drugim uporabnikom (npr. prijateljem, sodelavcem,...). Tak sistem zaščite uporabnike preveč omejuje, zato bo potrebno dodati v sistem tudi deljenje virov med domenami. V ta namen je nujno potrebna vpeljava sistematičnega načina za opis pravic dostopa uporabnikov do datotek ([15]).

Lampson predlaga [15] idealizirani sistem, ki ga poimenuje sistem objektov. Sestavljajo ga tri glavne komponente: množica objektov z oznako X, množico domen, ki jo označi z D in matrika dostopnosti, označeno z A. V tem sistemu želimo omejiti dostop do objektov. Objekti so v splošnem lahko kakršnakoli stvar, ki potrebuje zaščito dostopa, v primeru naloge so to datoteke. Domene so entitete, ki imajo dostop do objektov. Njihova glavna lastnost je, da imajo lahko različne dostope za različne objekte.

Matrika dostopnosti A določa dostop domen do objektov. Za vsako domeno je v matriki svoja vrstica, stolpci matrike pa pripadajo objektom. Vsak element matrike A_{ij} določa dovoljenja dostopa domene i do objekta j. Dovoljene dostope določa množica nizov, imenovanih prilastki dostopa. Dandanes so tipični prilastki `branje`, `pisanje` in `izvajanje`. Pravimo, da ima domena i dostop za x do elementa j, če je x na seznamu elementa A_{ij}. Vsak prilastek ima pripet še bit imenovan zastavica za kopiranje, ki določa, če domena sme kopirati dovoljenja dostopa za ta objekt na drugo domeno ([15]).

Dodajanje in brisanje vnosov matrike poteka po določenih pravilih:

- lastništvo objekta omogoča dodajanje poljubnega dostopa za ta objekt ostalim domenam (z ali brez zastavice za kopiranje),

A	Domena 1	Domena 2	Domena 3	Datoteka 1	Datoteka 2	Datoteka 3
Domena 1	Lastništvo Nadzor	Klic			Lastništvo+ Branje+ Pisanje+	Branje
Domana 2	Lastništvo			Branje	Branje+ Pisanje+	Lastništvo+
Domena 3		Lastništvo+ Nadzor	Lastništvo+ Nadzor	Pisanje+	Branje Pisanje	

+ zastavica za kopiranje nastavljena

Slika 3.1: Primer matrike dostopnosti.

- če ima domena zastavico za kopiranje pri določenem prilastku objekta, lahko dodaja ta prilastek za ta objekt ostalim domenam,
- dostop za nadzor domene omogoča brisanje poljubnih prilastkov v tej domeni (brisanje prilastkov dostopa za celotno vrstico domene).

Primeri s slike 3.1:

- domena 2 lahko doda prilastke dostopa za datoteko 3 domeni 3 (ker ima za to datoteko lastniški dostop),
- domena 3 lahko doda prilastek 'pisanje' za datoteko 1 domeni 2 (ker ima pri tem prilastku zastavico za kopiranje),
- domena 3 lahko briše prilastke dostopa za domeno 2 in 3 (ker ima za ti domeni dostop za nadzor).

Zastavica za kopiranje je namenjena omejevanju omogočanja dostopa. Domena ki ima prilastke dostopa brez zastavice, tako ne more dodeljevati dostopa naprej. Pojavlja se tudi vprašanje odvzemanja pravic dostopa v primeru lastništva objekta. Če lastniku dovolimo brisanje prilastkov dostopa, je smiselno uvesti še zaščito pred brisanjem prilastkov ostalih lastnikov objekta. Sicer bi se lahko zgodilo, da en lastnik onemogoči dostop drugemu lastniku objekta, kar navadno ni najbolj zaželjeno.

Z vidika nadzora dostopa se pojavlja še ena dilema. Ali domena, ki je lastnica objekta oziroma ima pri prilastkih dostopa do tega objekta nastavljene zastavice za kopiranje, lahko dodeljuje dostop poljubnim domenam, ali samo domenam, katerih lastnik je. O tem se mora odločiti načrtovalec sistema.

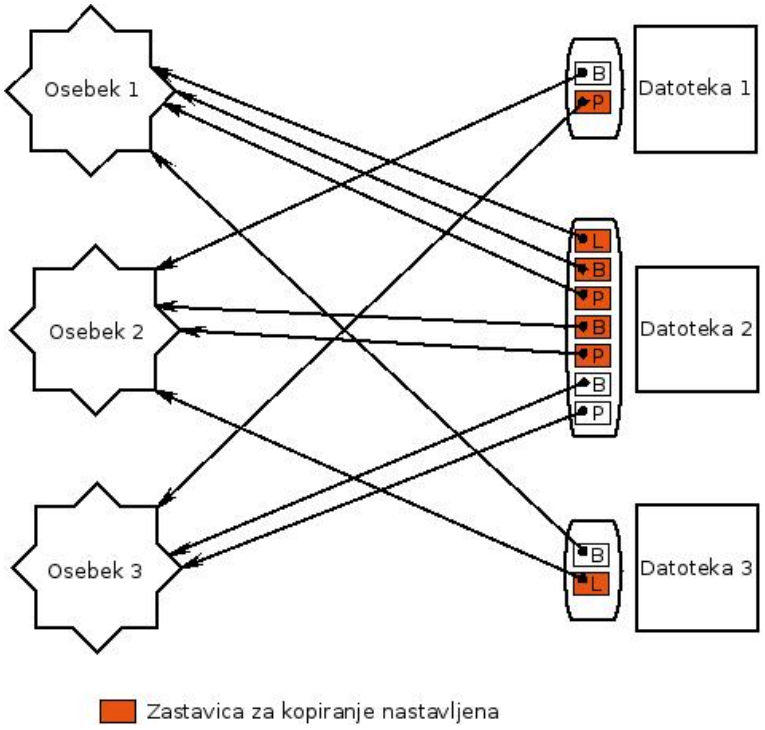
Shranjevanje matrike dostopnosti v obliki tabele, kot je na sliki 3.1 je precej potratno. Zapisi v tabeli so namreč tipično redko posejani. Za boljšo izkoriščenost prostora, bi lahko matriko A predstavili s tabelo s tremi stolpci. Vsako neprazno polje s slike 3.1, bi predstavili s trojico: d, x in $A_{d,x}$.

Tudi predstavitev s takšno tabelo je precej potratna. Navadno namreč niso aktivne vse domene in objekti, zato bi takšna tabela v glavnem pomnilniku zasedala precej nepotrebnega prostora. Poleg tega pa je še čas iskanja po takšni tabeli sorazmeren z njeno velikostjo, kar bi v primeru v tistem trenutku nepotrebnih zapisov v tabeli povečalo čas dostopa do objekta.

Očitna sta še dva mogoča načina implementacije. Predstavitev matrike dostopnosti po stolpcih ali po vrsticah. Pri predstavitvi matrike s stolpci je mogoče predstaviti matriko kot seznam nadzora dostopa (*Access Control List*, tudi ACL), ki ga pripnemo vsakemu objektu, vsak vnos seznama pa predstavlja pravice, ki jih ima določena domena nad tem objektom. Druga možna predstavitev je s seznamom zmožnosti (*Capability List*). Seznam zmožnosti se vzdržuje pri osebkih oziroma domenah. Seznam zmožnosti določa do katerih objektov in kakšen način dostopa je mogoč ([15]).

3.2.1 Seznam nadzora dostopa

Ena od možnih implementacij je shranjevanje prilastkov dostopa pri objektu. V splošnem je ideja v tem, da za vsak objekt x do katerega želi domena d dostopati, pokliče proceduro $A_x(d)$, ki želen način dostopa do objekta dovoli ali pa ga zavrne. Procedura je določena s strani lastnika objekta. Ime domene ni praktično shranjevati, zato je smiselna predstavitev domene z številko, ki jo določi sistem. Te številke tako ni mogoče ponarediti, saj na preslikavo, ki jo naredi sistem, uporabnik nima vpliva. Vsak uporabnik, ki kliče proceduro,



Slika 3.2: Seznam nadzora dostopa.

od sistema dobi unikaten ključ dostopa. Lastnik objekta, do katerega želi uporabnik dostopati, proceduro nastavi tako, da odobri ali zavrne dostop glede na ključ dostopa. Ker uporabnik ključa ne more ponarediti, ga lahko brez bojazni za zlorabo posreduje ostalim uporabnikom, ki mu lahko s tem omogočijo dostop do svojih datotek ([15]).

Manj splošen način predstavitve matrike dostopnosti pri objektu je predstavitev s seznamom nadzora dostopa. Vsak vnos seznama je sestavljen iz para (ključ, prilastki dostopa). Če je ključ, s katerim dostopamo, na seznamu, dobimo dostop do objekta, ki ga določajo prilastki dostopa pri tem vnosu. Seznam nadzora dostopa za vsak objekt posebej, je precej nerodna implementacija. Zato sistemi navadno uporabljajo posebne objekte, ki vsebujejo druge objekte. Ti objekti so navadno poznani kot mape in omogočajo razporeditev objektov v drevesno strukturo ([15, 24]).

Ena od pomanjkljivosti predstavitve matrike dostopnosti s seznamom

nadzora dostopa je v tem, da je predstavljena pri objektu in tako uporabnik ne ve, do katerih objektov lahko dostopa, oziroma kateri objekti sploh obstajajo. To se lepo vidi na sliki 3.2, ki prikazuje implementacijo matrike A s seznamom nadzora dostopa. Puščice za odobritev dostopa potekajo od objektov proti osebkom. Brez informacije o obstoju objekta je do njega nemogoče dostopati. Za razrešitev tega problema je potreben ločen sistem, ki da uporabniku informacijo o obstoječih objektih in tako sploh možnost za dostop ([15, 24]).

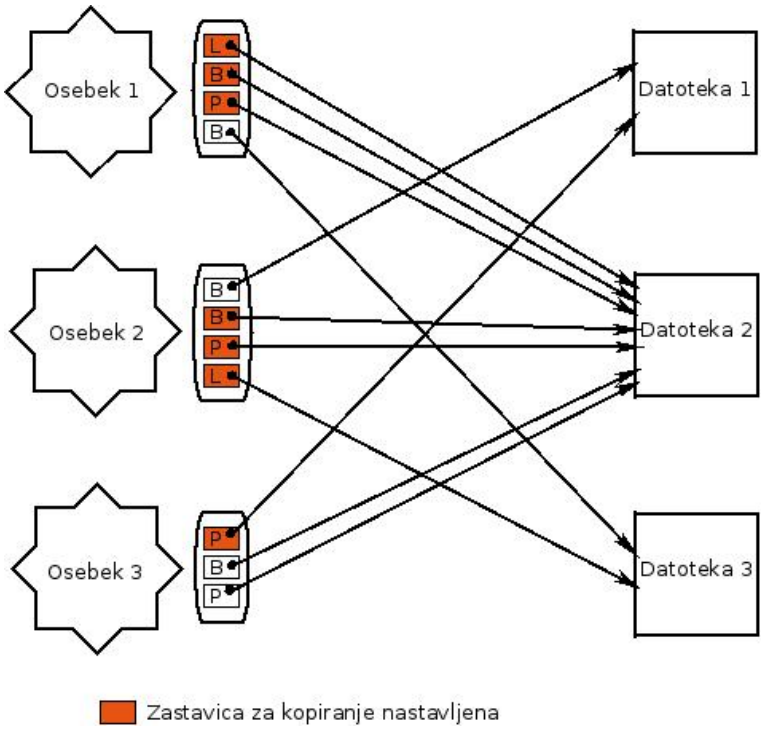
3.2.2 Seznam zmožnosti

Druga možnost implementacije je predstavitev matrike dostopnosti s seznamom parov $(x, A_{d,x})$. Vsak tak par predstavlja zmožnost dostopa domene d do objekta x.

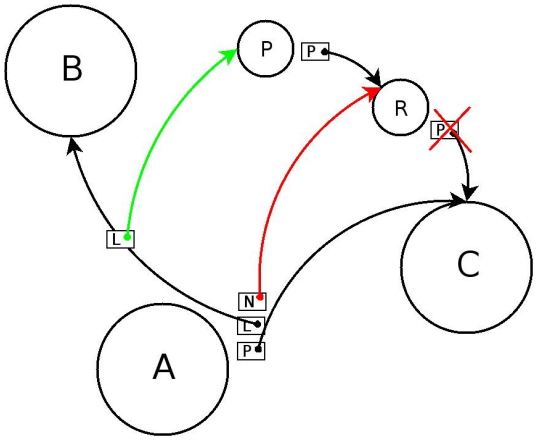
Tej predstavitvi matrike A zato pravimo seznam zmožnosti. Posedovanje zmožnosti domene v tem primeru že predstavlja avtorizacijo dostopa do objekta. Kot se lepo vidi iz slike 3.3, za dostop do objekta ni potreben noben dodaten seznam objektov, saj dostop do objekta določa že sama zmožnost. Če zmožnosti za objekt x ni v seznamu zmožnosti domene d, potem domena d (brez dodatnih mehanizmov) objekta x ne vidi ([15, 24]).

Ena od pomanjkljivosti, ki se omenja pri seznamu zmožnosti, je nezmožnost omejevanja dodeljevanja dostopa drugim uporabnikom. Sam model seznama zmožnosti nima te pomanjkljivosti. Omejitev lahko nastane pri implementaciji. Če sistem zastavimo tako, da lahko določen osebek prenaša dovoljenja dostopa do objektov na druge osebke, le v primeru da ima lastništvo nad temi objekti, potem lahko širjenje dovoljenj dostopa omejimo s primerno definiranimi zmožnostmi pri osebkih. Če ima osebek dostop za nadzor nad osebkom, ki mu dodeljuje zmožnosti, mu jih lahko tudi poljubno odvzema ([24]).

Pri pomanjkljivostih seznama zmožnosti se dostikrat omenja tudi avtorjeva nezmožnost razveljavitve podeljene zmožnosti dostopa do objekta. Tudi temu se je moč izogniti pri primerni zasnovi sistema in definiciji zmožnosti.



Slika 3.3: Seznam zmožnosti.



Slika 3.4: Podelitev zmožnosti dostopa, ki omogoča razveljavitev.

Predpostavimo, da lastništvo nad določenim osebkom omogoča tudi vpogled v njegov seznam zmožnost in s tem kopiranje vnosov. V tem primeru za realizacijo možnosti razveljavitve podeljene zmožnosti, potrebujemo dva nova osebka. Enega za posredovanje (P) zmožnosti, drugega pa za možnost razveljavitve (R) zmožnosti (glej sliko 3.4).

Denimo, da imamo tri osebke: A, B in C. Osebek A ima zmožnost pisanja za osebek C, hkrati ima lastniški dostop za B. Svojo zmožnost pisanja za C bi želel dodati tudi osebku B, vendar obstaja možnost, da si v prihodnosti premisli, zato želi imeti možnost preklica podeljene zmožnosti. V ta namen lahko osebek A ustvari dva nova osebka P in R. P bo služil za posredovanje zmožnosti, R pa za njeno razveljavitev. Osebku R mora dodati zmožnost pisanja za C, P-ju pa zmožnost pisanja na R. S prenosom lastništva osebka P na osebek B, dobi B preko P in R zmožnost za pisanje za osebek C. Če želi A razveljaviti podeljeno zmožnost pisanja B-ju, lahko to stori z brisanjem zmožnosti pisanja osebku R nad C-jem. Da lahko to zmožnost briše, mora imeti tudi zmožnost za nadzor R-ja. osebek B s tem izgubi možnost pisanja na C, saj do njega nima več niti posredne reference ([24]).

Kot vidimo, sta si seznam nadzora dostopa in seznam zmožnosti v splošnem zelo podobna. Kot glavno razliko velja ponovno izpostaviti potrebo po

ločenem mehanizmu za omogočanje vidnosti objektov osebkom. Ker je seznam nadzora dostopa definiran pri objektu, potrebujejo osebki referenco na objekte, da sploh lahko zaprosijo za dostop do njih. Pri seznamu zmožnosti je že sama zmožnost referenca na objekt in hkrati avtorizacija za dostop. Večina ostalih lastnosti obeh sistemov je odvisnih od same implementacije.

Poglavje 4

Primeri rešitve

Za prenos datotek med napravami je na voljo kar nekaj protokolov. Specifikacije nekaterih so javno dostopne in jih tako lahko vsakdo implementira, spet drugi so bili razviti za točno določeno storitev in njihove podrobnosti niso javno objavljene. V tem poglavju si bomo pogledali nekaj mogočih rešitev za implementacijo želene storitve. Pri implementaciji bomo pozorni na tri lastnosti sistema: (i) dober kompromis med usklajenostjo in dosegljivostjo datotek ter odpornostjo na delitev omrežja, (ii) nadzor dostopa in (iii) preprostost uporabe.

4.1 NFS

Kratica NFS pomeni *Network File System* oziroma omrežni datotečni sistem. NFS je odprt standard, ki ima več različic, prva javno dostopna je NFSv2, sledile so NFSv3 in NFSv4 definirana v RFC 3530 ([32]), ki je bila razširjena z različico NFSv4.1 definirano v v RFC 5661 ([33]). Ker je standard javno objavljen, ga lahko implementira vsakdo. Ideja protokola je v omogočanju pripenjanja dela diskovnega prostora z oddaljene strežniške naprave, tako da z njim lahko odjemalec upravlja kot z lokalnim diskovnim prostorom. Na ta način je dosežena možnost preprostega deljenja datotek čez omrežje.

NFSv4 je začel podpirati tudi porazdeljevanje podatkov na več strežnikov,

na način, ki je za odjemalca neopazen, saj je uporabljen skupni imenski prostor. Z različico 4 je omogočeno tudi podvajanje podatkov, kar omogoča doseganje dobre dosegljivosti, lahko pa nastanejo težave z usklajenostjo. To se da deloma omiliti z uporabo zaklepanja virov, ki je tudi ena od novih možnosti pri različici NFSv4. NFS ne podpira operacij brez povezave, ki so za prenosne naprave zelo zaželjene, saj omogočajo neko omejeno obliko dosegljivosti tudi ko naprava ni povezana s storitvijo ([32]).

Ko imamo opravka z deljenjem datotek, se postavi tudi vprašanje varnosti oziroma zaščite dostopa do datotek pred nepooblaščenimi osebami. Za preverjanje pristnosti odjemalca so pri različici NFSv4 na voljo 3 možnosti: Kerberos 5, SPKM3 in AUTH_SYS. Odjemalec in strežnik se dogovorita, katera od teh bo uporabljena ([32]).

Za nadzor dostopa je uporabljen model ACL, ki za razliko od starejših različic, kjer se je uporabljal tradicionalni UNIX nadzor dostopa, združljiv tudi z Windows okoljem ([32]). Ko se diskovni prostor pripne na odjemalčev sistem, poteka zaščita dostopa preko lokalnega sistema za nadzor dostopa. Odjemalec lahko dostopa do datotek, če imajo datoteke za njegov UID (identifikacijska številka uporabnika) dovoljenja za želeno vrsto dostopa. Izjema so datoteke, za katere ima dovoljenje za dostop uporabnik z UID 0, oziroma uporabnik s korenskim dostopom. Do teh datotek tudi s korenskim računom na odjemalcu ni mogoče dostopati, saj se UID 0 odjemalca pretvori v 500, ki ima dostop le do datotek s pravicami za javni dostop. Če imamo na odjemalcu korenski dostop, lahko ustvarimo račun z poljubnim UID in na tak način lahko pridobimo dostop do poljubnih datotek ([4, 36]).

Ta način nadzora dostopa je neučinkovit, če nimamo poštenih uporabnikov. Eden od načinov bi sicer bil ta, da bi vsakemu uporabniku dodelili svoj del diskovnega prostora. Za deljenje datotek med uporabniki bi lahko uporabili del prostora, do katerega bi imeli dostop vsi oziroma le določena skupina uporabnikov. Žal ne omogoča zaščite dostopa datotek, kot je to na lokalnem sistemu, kjer razen uporabnika s korenskim dostopom, ostali uporabniki ne morejo dostopati do datotek ostalih uporabnikov, za katere niso

pridobili pravic, čeprav uporabljajo vsi isti diskovni prostor.

Pri rešitvi NFS poteka dostop do oddaljenega diskovnega prostora neposredno. Odjemalci navadno ne vzdržujejo lokalne kopije, zato pride v primeru delitve omrežja do nedosegljivosti datotek. To vsekakor kvari uporabniško izkušnjo, saj je dosegljivost datotek za uporabnike zelo pomembna.

4.2 WebDAV

WebDAV je razširitev protokola HTTP, zato si bomo najprej ogledali značilnosti in delovanje le tega. HTTP je aplikacijski protokol za porazdeljene informacijske sisteme. Kratica HTTP pomeni *hypertext transfer protocol* ali protokol za prenos hiperteksta. Hipertekst je množica objektov, ki z logičnimi povezavami med vozlišči sestavljajo graf. Na osnovi protokola HTTP poteka komunikacija in prenos podatkov v svetovnem spletu.

HTTP paket je sestavljen iz vrstice za zahtevek, sledijo zaglavja od katerih je obvezno `host` (od različice HTTP/1.1), ostala so poljubna. Temu sledi prazna vrstica in sporočilo. V vrstici za zahtevek je ime predpisane metode, ki ji sledi URI zahtevka (pot do vira). V prvi različici standarda je bila predpisana le metoda GET, ki še vedno omogoča prenos želene datoteke s strežnika na odjemalca. V standard HTTP/1.0 so dodali še metodi HEAD in POST. Prva je namenjena prenosu zaglavja HTTP paketa brez vsebine, druga pa je namenjena spreminjanju oziroma prenosu vsebine na strežnik ([12]).

Ker se je pokazala potreba po dodatnih metodah, so sprejeli še standard 1.1, v katerem so definirali pet dodatnih metod ([12]):
PUT: za nalaganje točno določenega vira na strežnik,
DELETE: za brisanje vira s strežnika,
TRACE: strežnik v odgovoru vrne zahtevo odjemalca,
OPTIONS: za poizvedbo o podprtih medodah strežnika in
CONNECT: za vzpostavitev TCP/IP tunela, uporabno za šifriranje komunikacije po protokolu TLS in SSL.

Od strežnika je zahtevana podpora za metodi GET in HEAD ter, če je le možno, za metodo OPTIONS. Sicer pa lahko podpira poljubno število poljubnih metod. To omogoča enostavno razširitev standarda brez težav z združljivostjo z obstoječo infrastrukturo ([12]).

Ta prilaglodljivost protokola je vodila tudi v vpeljavo razširitve WebDAV (*Web Distributed Authoring and Versioning* oziroma protokol za mrežno porazdeljeno avtorstvo z beleženjem različic). Beleženje različic je bilo sicer dodano kasneje, kot razširitev, medtem ko se osnovni protokol osredotoča na avtorstvo oziroma upravljanje z viri. V razširitvi WebDAV so bila definirana nova zaglavja in metode, ki so olajšale delo z datotekemi ([11]):
PROPFIND: za pridobitev lastnosti vira, tudi vsebino mape,
PROPATCH: za brisanje več lastnosti vira z enim samim nedeljivim opravilom,
MKCOL: za ustvarjanje zbirke (mape),
COPY: za kopiranje vira z enega URI na drugega,
MOVE: za premikanje vira z enega URI na drugega,
LOCK: za zaklep vira in
UNLOCK: za odklep vira.

Kot je razvidno iz opisa metod, razširitev WebDAV dodaja metode podobne ukazom za delo z datotekami kot jih uporabljajo operacijski sistemi. Navadno so WebDAV odjemalci implementirani tako, da lahko uporabnik z oddaljenimi datotekami ravna tako, kot da bi bile lokalno na računalniku. Odjemalec sam poskrbi za sinhronizacijo lokalnih datotek s stanjem na strežniku ([11]).

Tako pri protokolu HTTP, kot pri njegovi razširitvi WebDAV v osnovni implementaciji podatki med odjemalcem in strežnikom potujejo v nešifrirani obliki. Temu se lahko izognemo z uporabo šifrirnega protokola SSL oziroma TLS. Z uporabo metode CONNECT se ustvari tunel med posrednikom na odjemalčevi in posredniku na strežniki strani. Na ta način zaščitimo uporabniške podatke in ostalo komunikacijo med odjemalcem in strežnikom. Po-

leg tega se strežnik predstavi s svojim SSL certifikatom in se je tako mogoče prepričati, da komunikacija resnično poteka z želenim strežnikom, kar izboljšuje varnost pred napadi. Prav tako lahko na enak način strežnik sklepa o pristnosti odjemalca ([30]).

Pri protokolu WebDAV je arhitektura odvisna predvsem od implementacije odjemalca. Odjemalec lahko bodisi dostopa neposredno do oddaljenega odložišča, bodisi z vzdrževanjem lokalne kopije s sinhronizacijo med lokalnim in oddaljenim odložiščem. Z izbiro odjemalca je tako določena glavnina kompromisa med dosegljivostjo, usklajenostjo in odpornostjo na delitev omrežja.

Če odjemalec dostopa do strežnika neposredno, potem si lahko obetamo dobro usklajenost podatkov, ker obstaja le ena kopija datotek. Tudi dosegljivost ni problematična, dokler ne pride do delitve omrežja(ali izpada oziroma preobremenjenostjo glavnega vozlišča). Ko nastopi delitev omrežja, to je ko ni povezave med odjemalcem in strežnikom, je storitev nedosegljiva.

Druga možnost je izbira odjemalca, ki vzdržuje lokalno kopijo datotek. V tem primeru imamo odlično dosegljivost, saj je kopija datotek tudi v lokalnem datotečnem sistemu. V tem primeru že zaradi same replikacije podatkov obstaja možnost neusklajenosti. Ta pa je neizbežna, ko nastopi delitev omrežja, datoteke na lokalnem ali oddaljenem odložišču pa se spremenijo. Če je odjemalec implementiran tako, da zna dobro razreševati neskladja, ki se pojavljajo ob delitvah omrežja, je ta rešitev za naprave, ki nimajo vedno povezljivosti, smiselnejša.

Za nadzor dostopa do virov, WebDAV uporablja svojo implementacijo seznama nadzora dostopa. Preverjanje pristnosti uporabnika poteka z dostopom na določen naslov na način, ki ga podpira protokol HTTP ([12]). Nadzor dostopa je odvisen od implementacije strežnika in je ločen od mehanizma za nadzor dostopa lokalnega datotečnega sistema. S primerno implementacijo sistema, bi bilo mogoče mehanizem za nadzor dostopa vgraditi tudi v odjemalca ([7, 10]).

Pri rešitvi WebDAV je mogoče izbrati odjemalca, ki s sinhronizacijo z oddaljenim odložiščem, vzdržuje lokalno kopijo datotek. Na ta način lahko

uporabnik dostopa do datotek tudi v prisotnosti delitve omrežja.

4.3 Git

Git je porazdeljen sistem za nadzor različic zbirke datotek. Beleži zgodovino sprememb zbirke vključno z možnostjo vrnitve zbirke na poljubno predhodno stanje. To stanje lahko pomeni drugačno zbirko datotek ali samo spremembo njihove vsebine. V porazdeljenih sistemih za nadzor različic ima vsakdo popolno kopijo zbirke, vključno z njeno zgodovino, kar mu omogoča izvajanje operacij za nadzor različic nad lokalno kopijo. Uporaba takega sistema ne zahteva centralnega odložišča, odjemalec je hkrati lahko tudi strežnik drugemu odjemalcu in obratno ([13, 39]).

Za beleženje sprememb datoteke jo je potrebno dodati na seznam datotek zbirke. Dodajanju datotek je namenjen ukaz `add`. Z ukazom `commit` se datoteke odda v odložišče. Zbirko datotek je mogoče povrniti na katerokoli prejšnje stanje, ki je bilo označeno z ukazom `commit` ([6, 39]).

Lokalno odložišče lahko kadarkoli sinhroniziramo tudi z oddaljenimi odložišči. Oddaljena odložišča lahko kloniramo (identična kopija vključno z zgodovino). Lastnik odložišča lahko sinhronizira lokalno odložišče z oddaljenim z ukazom `pull`, ali oddaljenega s svojim lokalnim z ukazom `push` ([6, 39]).

Git omogoča tudi razvejanje. To pomeni, da lahko obstaja več različic zbirke datotek. Ta možnost je uporabna predvsem pri razvoju programske opreme, kjer je mogoče neko novo funkcionalnost kodo razvejati na novo vejo in tako s paralelnim razvojem ni vpliva na glavno zbirko datotek ([6, 39]).

Git podpira način delovanja brez povezave. Vsako vozlišče sistema shranjuje zgodovino sprememb lokalnega odložišča. S pomočjo zgodovine lahko tudi razrešuje morebitna neskladja, ki nastajajo pri delu brez povezave. Pri vozliščih, ki so povezana, lahko s spomočjo zgodovine sinhronizira vse kopije datotek na vozliščih v usklajeno stanje. S takšno zasnovo sistema lahko dosežemo odlično dosegljivost storitve, saj lahko datoteke uporabljamo tudi, ko ni povezave z ostalimi vozlišči. Tu se sicer neizbežno pojavi zmanjšanje

usklajenosti, vendar se ta obnovi, ko delitev omrežja ni več prisotna.

Prenos podatkov med napravama poteka po protokolu Git. Za zagotavljanje integritete medsebojne komunikacije, le ta poteka preko tunela SSH. Protokol SSH omogoča tako preverjanje pristnosti, kot tudi zaščito integritete komunikacije med vozliščema. Za povezavo odjemalca s strežnikom je potrebno le dodati javni ključ odjemalca na predvideno mesto na strežniku, to je v domači mapi uporabnika, preko katerega bo dovoljen dostop do sistema. Nadzor dostopa poteka po standardnem UNIX ACL mehanizmu, novejši sistemi UNIX, podpirajo tudi uporabo razširjenega seznama nadzora dostopa, ki omogoča dodeljevanje dostopa tudi posameznim uporabnikom in različnim skupinam ([6, 39]).

Pri rešitvi Git lahko uporabniki dostopajo do datotek tudi, ko nimajo povezljivosti v medmrežje. To je mogoče, ker do datotek vedno dostopamo preko lokalno kopije, ki jo z odjemalcem lahko sinhroniziramo z oddaljenim odložiščem. Programska oprema Git vsebuje tudi odjemalca, ki pa je brez grafičnega vmesnika, sinhronizacijo je potrebno sprožiti ročno. Programska oprema Git, sama po sebi ne nudi uporabniške izkušnje, ki si jo želi večina uporabnikov. V kombinaciji z alternativnim odjemalcem in mehanizmom, ki omogoča nadzor datotek, bi lahko bil primerna rešitev tudi za širši krog uporabnikov.

4.4 Dropbox

Dropbox je storitev namenjena shranjevanju in deljenju datotek na oddaljenem mestu. Podjetje ki ponuja gostovanje, je razvilo tudi istoimenski odjemalec za sinhronizacijo datotek z lokalne naprave, z datotekami na oddaljenem mestu. Odjemalec je podprt na večini namiznih okoljih, kot tudi na okoljih najpogostejših mobilnih naprav. Za uporabnika upravljanje z datotekami poteka tako, kot z vsemi ostalimi lokalnimi datotekami. Odjemalec poskrbi, da se spremembe lokalne kopije datotek prenesejo tudi na oddaljeno mesto. Poleg prenosa datotek preko odjemalca, je mogoče do datotek

dostopati in jih nalagati tudi preko spletnega vmesnika ([9]).

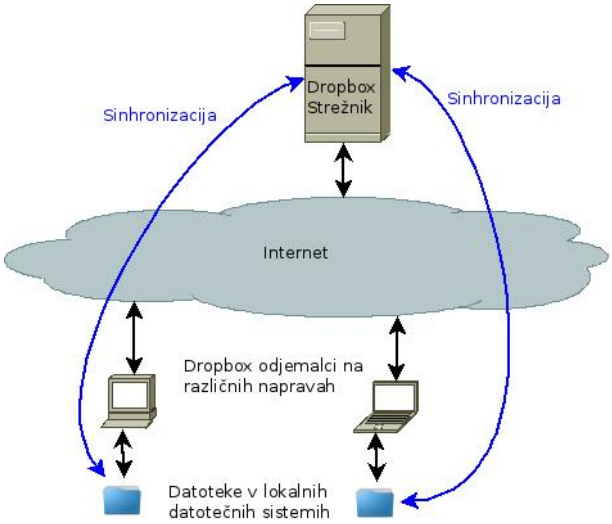
Za sinhronizacijo uporablja knjižnico `librsync`, ki uporablja rsync remote-delta algoritem. Algoritem omogoča prenos sprememb med datotekama in ne potrebuje obeh kopij datotek na istem mestu. To mu omogoča učinkovitost pri prenosu, saj se med sinhronizacijo ne prenašajo celotne datoteke, ampak le spremembe. Z vidika odjemalcev je arhitektura Dropboxa taka, da se odjemalci sinhronizirajo s centralnim strežnikom (glej sliko 4.1). Dropbox podpira tudi lokalno sinhronizacijo. Odjemalci, ki so povezani v lokalno omrežje, se lahko tudi medsebojno sinhronizirajo, kar pospeši proces sinhronizacije ([9, 37]).

Prenos datotek poteka po šifrirani povezavi SSL, tudi podatki so shranjeni v šifrirani obliki z uporabo AES-256 šifriranja. Za shranjevanje datotek na oddaljenem mestu Dropbox uporablja Amazonov sistem za shranjevanje S3 ([9]).

Dropbox podpira tudi nadzor različic. To doseže z zgodovino sprememb, ki jo beleži. Zgodovina omogoča lažje in učinkovitejše razreševanje neskladij v primeru dela brez povezave ali sočasnih operacij. Kot vidimo, storitev ohranja dosegljivost datotek tudi v primeru delitve omrežja. Neusklajenost, ki nastane pri delu brez povezave, se razrešuje, ko se uporabnik ponovno poveže z medmrežjem.

Način, s katerim je dosežen nadzor dostopa do datotek in map, ni javno objavljen. Kot kaže, pri OwnCloudu zaščita dostopa do datotek ni povezana z zaščito dostopa do lokalnih datotek. Če namreč spremenimo pravice za dostop do datotek ali map v lokalnem datotečnem sistemu se to pri drugih uporabnikih Dropbox-a ne pozna.

Dodeljevanje dostopa poteka preko spletnega vmesnika. Z ostalimi uporabniki je mogoče deliti mape. Lastnik deljene mape lahko dodaja in odvzema dostop drugim uporabnikom. Ko ima uporabnik dostop do deljene mape, lahko bere in piše v katerokoli datoteko v tej mapi. Za vsakega uporabnika je določeno, do katerih map ima dostop in te mape se sinhronizirajo z lokalnim odložiščem. Ker lastnik lahko prekliče zmožnost dostopa ostalim



Slika 4.1: Arhitektura Dropboxa.

uporabnikom, mora biti implementiran tudi mehanizem, ki omogoča preklic, kot naprimer s slike 3.4. Datoteke ali mape je mogoče s povezavo javno objaviti. Dostopne so vsakomur, ki pozna naslov za dostop.

Uporabniki lahko urejajo datoteke brez bojazni, da bi se katera od različic prepisala. Zgodovina sprememb omogoča tudi obnovitev izbrisanih datotek, ki jih lahko obnovimo s poljubnim odjemalcem, ki je imel dostop do te datoteke ([9]).

Dropbox je lastniška programska oprema in ne omogoča postavitve na lastnem strežniku. Celotna varnost je predvsem v rokah podjetja, ki ponuja storitev. Če sami ne poskrbimo za dodatno zaščito, kot naprimer šifriranje datotek, preden jih naložimo na oddaljeno mesto, lahko le zaupamo, da so datoteke s samo storitvijo primerno zaščitene.

Poglavje 5

Implementacija

V prejšnjem poglavju smo si pogledali nekaj možnih sistemov, s katerimi bi v grobem lahko izpolnili pričakovanja uporabnikov. Nobeden od teh sistemov ne izpolnjuje uporabniških želja v celoti. Kljub temu, bomo v tem poglavju vzpostavili storitev, ki se bo v čimvečji meri približala zastavljenemu cilju. Storitev bomo realizirali z implementacijama, ki uporabljata različne tehnologije in ju medsebojno primerjali po uspešnosti pri doseganju meril ustreznosti, ki jih bomo definirali v naslednjem razdelku.

5.1 Merila ustreznosti

Poleg zahtev oziroma želja, ki jih imajo do storitve uporabniki, imamo nekaj zahtev do sistema tudi z načrtovalskega vidika. Kot načrtovalci sistema, želimo izpolniti naslednja merila ustreznosti:

- postavitev na lastni strojni opremi,
- preverjanje pristnosti uporabnika z uporabo obstoječe baze uporabnikov LDAP ([41]),
- upravljanje z datotekami na strežniku, kot da bi se nahajale na lokalni napravi,
- deljenje datotek z ostalimi uporabniki,

- podprtost v različnih okoljih in

- integriteta komunikacije med odjemalcem in strežnikom.

Od primerov rešitev, ki smo si jih pogledali v prejšnjem poglavju, bo pri dveh ta merila težko doseči. Dropbox je lastniška programska oprema in ne omogoča postavitve na lastni strojni opremi. Pri rešitvi z uporabo NFS, pa bi imeli težave z učinkovito zaščito dostopa do datotek skupnega diskovnega prostora. Poleg tega v praksi nastajajo težave z delovanjem na različnih operacijskih sistemih, zaradi slabe podprtosti standarda pri odjemalcih ne-UNIX okolij. Še posebno pri mobilnih napravah (iOS, Windows Phone, itd.).

Za implementacijo bomo zato poiskali rešitev, ki za prenos podatkov med naprami uporablja protokol WebDAV in rešitev z uporabo protokola Git. Poskusili bomo v čimvečji meri izpolniti merila ustreznost, ki smo si jih zadali. Na koncu poglavja bo sledila evalvacija doseganja meril ustreznosti obeh implementacij.

Kot WebDAV rešitev smo namestili sistem OwnCloud. Po opisu funkcionalnosti na domači spletni strani ([27]), lahko sklepamo, da sistem OwnCloud, ki temelji na protokolu WebDAV, omogoča doseganje vseh zastavljenih meril ustreznosti. V nadaljevanju bomo preverili, kako je s tem v praksi.

Druga rešitev, ki smo jo namestili je sistem Git. Programska oprema Git (odjemalec in strežnik) sama po sebi ne omogoča izpolnitev zastavljenih lastnosti sistema. Odprto ostaja vprašanje vključitve že obstoječih uporabnikov v sistem. Za doseganje tretje točke pri želenih lastnostih, bo potrebno najti še odjemalca za samodejno sinhronizacijo lokalnih datotek s stanjem na strežniku. Poenostaviti bi bilo potrebno tudi omogočanje dostopa drugim uporabnikom. Za doseganje želenih lastnosti bo potrebno poiskati dodatno programsko opremo, ki bo osnovnemu sistemu povečala fonkcionalnost do te mere, da doseže zastavljene lastnosti in izboljša uporabniško izkušnjo.

V naslednjem razdelku si bomo pogledali, na kak način lahko postavimo oba sistema, da bi v čimvečji meri zagotovili izpolnjevanje meril ustreznosti.

5.2 Osnovna namestitev

Namestitev obeh sistemov bo potekala na operacijskem sistemu CentOS 6.3, ki je ena od distribucij Linux, namenjena za poslovno uporabo. CentOS temelji na distribuciji Red Hat in teži k popolni binarni združljivosti ([3]). Linux je bil izbran, ker je to najbolj razširjen odprtokoden operacijski sistem, ki v strežniškem tržnem deležu presega tudi vse zaprtokodne operacijske sisteme.

OwnCloud

Namestitev sistema OwnCloud je sorazmeroma preprosta. Na strežnik je potrebno namestiti programsko opremo, ki jo potrebuje OwnCloud za delovanje. Spletni vmesnik je napisan v programskem jeziku PHP, zato je potrebna programska oprema za poganjanje kode PHP, strežnik HTTP in baza SQL. Na domači spletni strani programske opreme OwnCloud je na voljo paket z izvorno kodo. Paket je potrebno prenesti na strežnik, razpakirati in prestaviti dobljeno mapo na želeno mesto v datotečnem sistemu ([28]).

Mapa z nastavitvenimi datotekami, mapa za datoteke uporabnikov in mapa za razširitve, morajo za pravilno delovanje imeti nastavljeno lastništvo na UID, ki ga uporablja spletni strežnik. Za uporabo spletnega vmesnika, je potrebno glavno mapo OwnCloud-a umestiti v imenski prostor spletnega strežnika ([28]). Preko spletnega vmesnika poteka osnovna nastavitev sistema (glej sliko 5.1).

Poleg skrbniškega računa, lahko v tem koraku nastavimo alternativno lokacijo za mapo z datotekami uporabnikov in podatkovno bazo, ki bo uporabljena, ter podatke za dostop do nje.

S tem je namestitev na strežniku končana. V spletni vmesnik se lahko prijavimo s skrbniškim računom in tam nadaljujemo nastavljanje. S skrbniškim računom sedaj že lahko dostopamo do prostora na strežniku, tudi preko večine (nekateri imajo zaradi nezdružljive implementacije težave) WebDAV odjemalcev.

Slika 5.1: OwnCloud — osnovne nastavitve.

Git

Pri večini novejših distribucijah operacijskega sistema Linux je git že prisoten v odložiščih s programsko opremo. Pred namestitvijo je potrebno namestiti knjižnice, od katerih je Git odvisen, nato ga lahko v namestimo iz ukazne lupine s preprostim ukazom. Ko je programska opema Git nameščena na strežnik, lahko ustvarimo novo odložišče z ukazom `git init`. Za oddaljen dostop do odložišča, je v datoteko za preverjanje pristnosti SSH uporabnika na strežniku potrebno dodati še javni del ključa s katerim bo mogoče dostopati do odložišča. S tem je osnovna namestitev na strežniku končana ([6]).

Na strani odjemalca so zahteve podobne. Za dostop do strežnika je potrebno namestiti programsko opremo git, ki med drugim vključuje tudi odjemalca. Preverjanje pristnosti uporabnika poteka preko javnega dela ključa, ki je nameščen na strežniku in zasebnega dela, ki je shranjen pri uporabniku. Za dostop do oddaljenega odložišča je v zbirko zasebnih ključev uporabnika potrebno dodati zasebni del para ključa, katerega javni del je nameščen na strežniku. S tem je tudi namestitev na uporabnikovi strani končana.

5.3 Vključevanje obstoječih uporabnikov v sistem

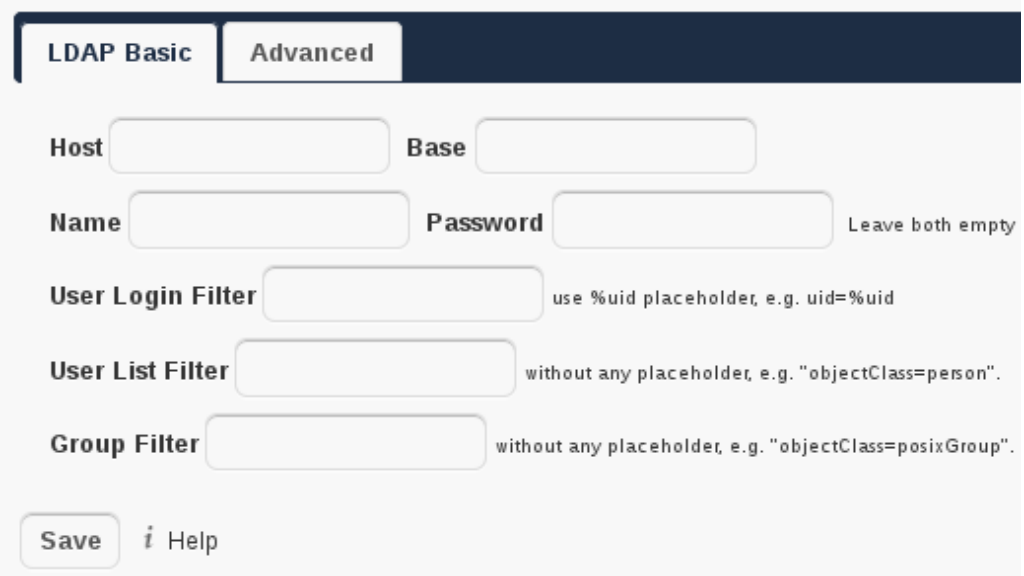
Če baza uporabnikov že obstaja, je smiselno preverjanje pristnosti nastaviti tako, da uporabi to bazo. Obstoječim uporabnikom za dostop tako ne bo potrebno nastavljati novega računa. Za potrebe te naloge bomo vključili v sistem uporabnike, ki imajo račun na strežniku LDAP ([41]).

OwnCloud

Pri sistemu OwnCloud obstaja razširitev ([29]), ki omogoča preverjanje pristnosti uporabnikov preko vnosov baze LDAP. Za vključitev razširitve se je potrebno prijaviti v spletni vmesnik s skrbniškim računom. Razširitev vključimo tako, da v meniju izberemo *Nastavitve*, nato *Aplikacije*, poiščemo razširitev *LDAP user and group backend*, in vključimo s klikom na gumb *Enable*. Na enak način lahko vključujemo oziroma izključujemo tudi ostale razširitve.

V meniju nastavitve se pojavijo možnosti za nastavitev preverjanja pristnosti LDAP (glej sliko 5.2). Zavihek *LDAP Basic* omogoča osnovne nastavitve za preverjanje pristnosti preko vnosov strežnika LDAP. Zavihek *Advanced* pa ponuja napredne možnosti nastavitev, kot so: uporaba nestandardnih vrat, uporaba šifrirane povezave, nastavitev omejitve diskovnega prostora na podlagi prilastka strežnika LDAP, itd. Ko vnesemo ustrezne podatke za nastavitev, shranimo spremembe s klikom na gumb *Save*. Če je LDAP strežnik dosegljiv in so vnešene nastavitve z njim skladne, bi preverjanje pristnosti na podlagi baze LDAP moralo delovani za dostop do spletnega vmesnika, kot tudi pri dostopu do strežnika z WebDAV odjemalcem.

Pri preizkusu delovanja smo opazili, da se lahko LDAP uporabniki preslikajo v istega uporabnika, kot je bil definiran preko skrbniškega računa spletnega vmesnika. Da bi se izognili možnosti dostopa dveh različnih uporabnikov do enega uporabniškega računa storitve, se je smiselno držati uporabe le enega načina dodeljevanja računov.

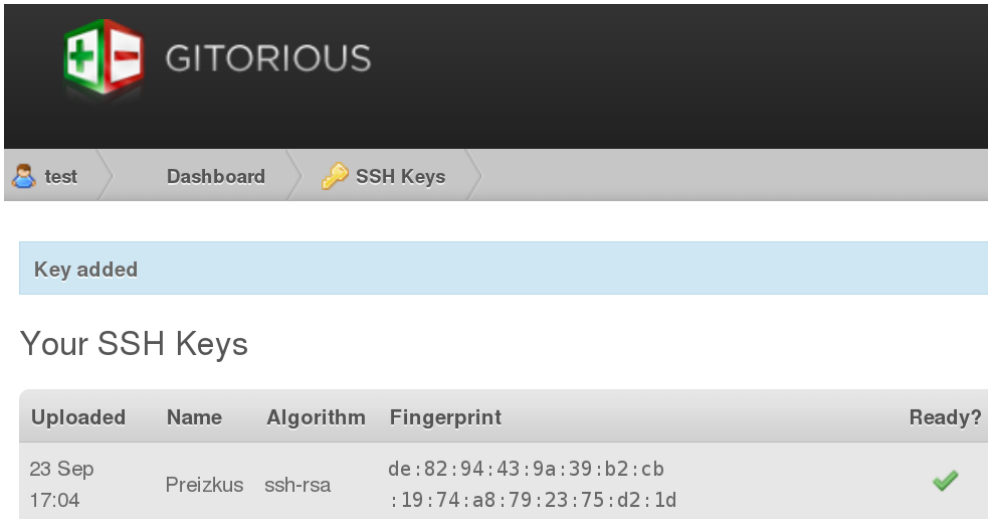


Slika 5.2: Nastavitve preverjanja pristnosti preko strežnika LDAP pri OwnCloud-u.

Git

Pri sistemu git je nekoliko težje zagotoviti način za preverjanje pristnosti uporabnika prek strežnika LDAP. Ker preverjanje pristnosti pri Gitu poteka po protokolu SSH, je potrebno najti način, ki preko LDAP preverjanja pristnosti omogoča nalaganje javnega dela ključa SSH na strežnik. Problem lahko rešimo s spletnim vmesnikom, ki omogoča nalaganje ključa SSH in preverjanja pristnosti preko strežnika LDAP. Preko spletnega vmesnika bi želeli dodajati tudi nova odložišča in dodeljevati pravice dostopa do njih, saj tega v osnovni namestitvi Git ne moremo početi drugače, kot z ročnimi nastavitvami na strežniku.

Na spletu je dostopnih precej različnih spletnih vmesnikov, ki omogočajo pregledovanje oddaljenega odložišča: Gitalist ([18]), Gitweb ([22]), Cgit ([5]), Gitlab ([19]), Gitorious ([21]), Gitolite ([20]), itd. Od naštetih le dva podpirata [19, 21] upravljanje s ključi SSH. Oba spletna vmesnika podpirata tudi preverjanje pristnosti LDAP in razširjata funkcionalnosti osnovne git



Slika 5.3: Upravljalec ključev v spletnem vmesniku Gitorious.

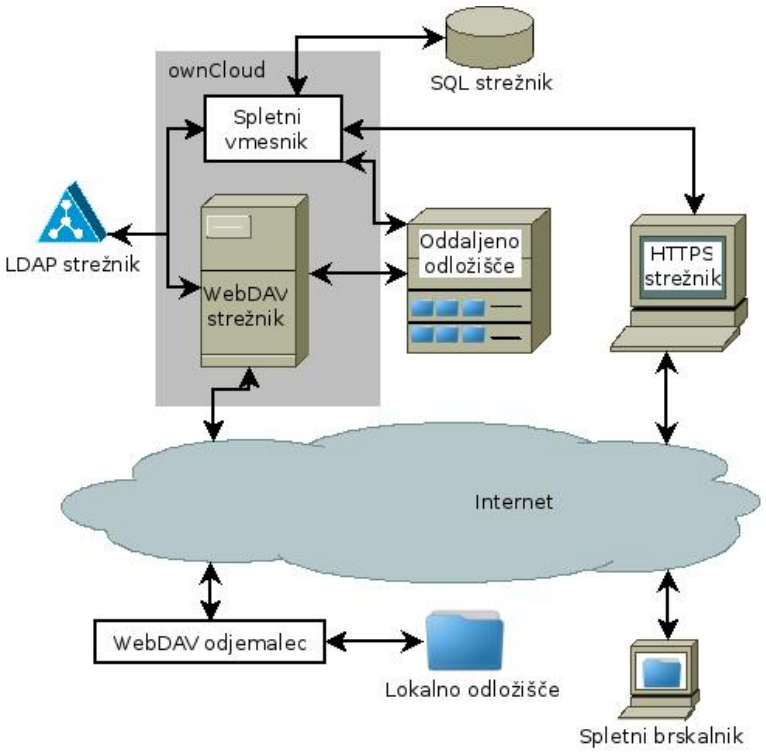
namestitve. Z vključitvijo enega od njiju v sistem bomo tako zadostili skoraj vsem merilom ustreznosti. Za naš sistem smo izbrali Gitorious. Namestitev ni enostavna, zato se v podrobnosti tu ne bomo spuščali. Bralec si lahko več o tem prebere na domači spletni strani ([21]). Manjka še odjemalec, ki omogoča samodejno sinhronizacijo lokalnega odložišča s strežnikom.

5.4 Upravljanje z oddaljenimi datotekami

Upravljanje z datotekami na oddaljenem mestu omogoča primeren odjemalec. Za boljšo uporabniško izkušnjo je smiselno izbrati tak odjemalec, ki omogoča umestitev oddaljenih datotek v lokalni datotečni sistem. Tak odjemalec je lahko implementiran na dva načina: z neposrednim dostopom do datotek na strežniku ali z vzdrževanjem lokalne kopije datotek.

OwnCloud

Za protokol WebDAV obstaja veliko različnih odjemalcev. Eni do oddaljenega mesta dostopajo neposredno, drugi vzdržujejo lokalno kopijo datotek

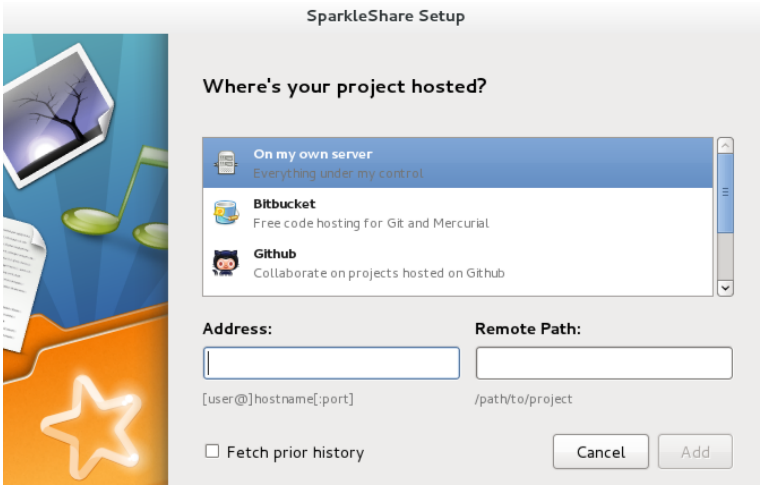


Slika 5.4: Arhitektura sistema OwnCloud.

in skrbijo za sinhronizacijo. Za uporabnike je zelo pomembna dosegljivost datotek, zato se zdi implementacija, ki vzdržuje lokalno kopijo datotek boljša rešitev, kljub temu, da lahko datoteke postanejo neusklajene, če jih spreminjamo, medtem ko nimamo povezave s strežnikom.

Na domači spletni strani OwnCloud ([26]) so na voljo odjemalci za najpogostejše operacijske sisteme namiznih naprav in odjemalca za mobilne naprave z operacijskim sistemom Android in iOS. Odjemalci so implementirani tako, da s sinhronizacijo vzdržujejo lokalno kopijo datotek, kar omogoča tudi delo brez povezave. Za povezovanje z oddaljenim odložiščem strežnika, bi lahko uporabili alternativne odjemalce, ki so na voljo tudi za manj razširjene operacijske sisteme. Ker je namenski odjemalec Owncloud Sync mogoče namestiti na večini uporabniških naprav, se tu v alternative ne bomo spoščali.

Nadzor dostopa pri sistemu OwnCloud poteka preko spletnega vmesnika.



Slika 5.5: Dodajanje novega oddaljenega odložišča pri odjemalcu SparkleShare.

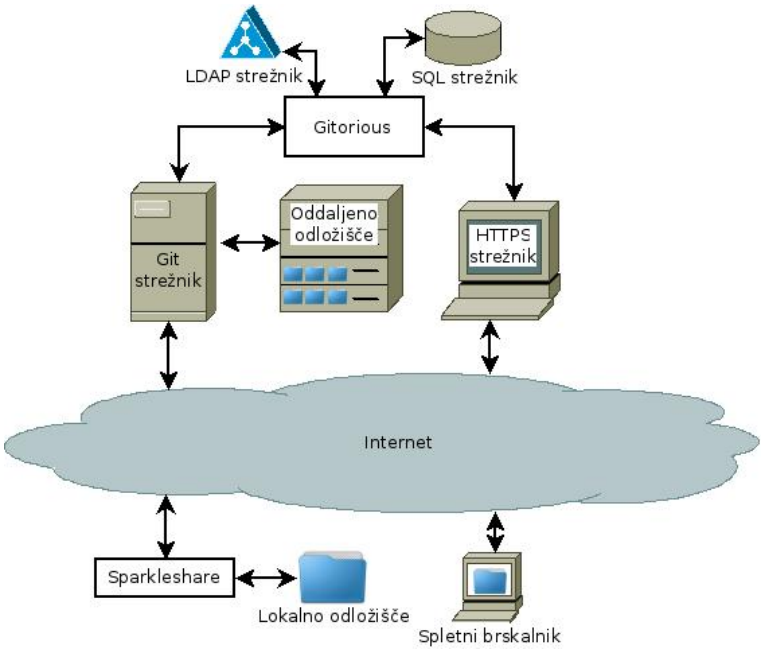
V spletnem vmesniku lahko dostop do datotek dodeljujemo uporabnikom skupine in skupini kateri pripadamo. Mogoče je tudi deljenje preko javne povezave. V tem primeru je datoteka za branje dostopna vsem, ki poznajo njeno povezavo.

Git

Tudi v sklopu programske opreme Git je na voljo odjemalec. Odjemalec je sicer zmogljiv, vendar nima grafičnega vmesnika. Za večino navadnih uporabnikov je tak način uporabe prezahteven. Da bi uporabniku izboljšali uporabniško izkušnjo, bomo poiskali takšnega odjemalca, ki omogoča upravljanje z datotekami iz grafičnega vmesnika za upravljanje z lokalnim datotečnim sistemom.

Ker je osnovni namen programske opreme Git nadzor različic izvorne kode, je tudi poudarek pri implementaciji večine odjemalcev na enostavnosti nadzora in preglednosti različic. Poleg tega ima večina ločen uporabniški vmesnik ([8, 25, 34]).

Odjemalec SparkleShare ([38]) se osredotoča na deljenje datotek. Pod-



Slika 5.6: Arhitektura sistema, ki temelji na programski opremi Git, spletnemu vmesniku Gitorious in odjemalcu SparkleShare.

prt je na namiznih okoljih Windows, Mac OS in Linux. Na voljo je tudi v trgovinah za aplikacije mobilnih naprav z operacijskim sistemom iOS in Android. S samodejno sinhronizacijo vzdržuje lokalno kopijo datotek usklajeno z oddaljenim odložiščem. Odjemalec Sparkleshare pri namestitvi ustvari nov par javnega in zasebega ključa in mapo *SparkleShare* v lokalnem datotečnem sistemu. V tej mapi se po končani namestitvi nahaja javni del ključa, ki ga je potrebno dodati na strežnik, do katerega želimo dostopati.

Na strežnik, ga v našem primeru lahko prenesemo preko spletnega vmesnika, kar prikazuje slika 5.3. Ko je javni del ključa nameščen na strežniku, lahko v Sparkleshare-u oddaljeno odložišče dodamo na seznam odložišč za sinhronizacijo. V meniju izberemo *Add Hosted Project* in vnesemo pot do oddaljenega odložišča in potrdimo s klikom na gumb *Add* (glej sliko 5.4). Če je bil javni del ključa nameščen na strežniku in če je bila vnešena prava pot do oddaljenega odložišča, je odjemalec v mapo *SparkleShare* dodal mapo s

kopijo oddaljenega odložišča. S tem je tudi nastavitev odjemalca pri implementaciji sistema Git zaključena.

Tako kot pri sistemu OwnCloud, tudi pri sistemu Git, nadzor dostopa poteka preko spletnega vmesnika. Spletni vmesnik Gitorious omogoča dodeljevanje dostopa za branje uporabnikom in skupinam. Za pisanje je potreben lastniški dostop do oddaljenega odložišča. Lastnik lahko prenese lastništvo odložišča na skupino in s tem dodeli vsem uporabnikom skupine dostop za branje.

5.5 Evalvacija

Tema tega razdelka je pregled implementacije obeh sistemov. Arhitektura končnih implementacij je vidna na slikah 5.5 in 5.6. Pri vsakem od definiranih meril ustreznosti si bomo na kratko pogledali, ali nam ga je uspelo doseči:

- postavitev na lastni strojni opremi: oba sistem nam je uspelo postaviti na lasten strežnik.

- preverjanje pristnosti uporabnika z uporabo obstoječe baze uporabnikov LDAP: sistem OwnCloud podpira preverjanje pristnosti preko strežnika LDAP za spletni vmesnik in za dostop z odjemalcem. Pri sistemu Git preverjanje pristnosti poteka po protokolu SSH, vendar pa lahko pri spletnem vmesniku, preko katerega je omogočeno nalaganje javnih ključev, preverjamo pristnost s strežikom LDAP.

- upravljanje z datotekami na strežniku, kot da bi se nahajale na lokalni napravi: pri implementaciji obeh sistemov smo našli odjemalce, ki s sinhronizacijo z oddaljenim odložiščem vzdržujejo lokalno kopijo datotek.

- deljenja datotek z ostalimi uporabniki: pri obeh sistemih je mogoče dodeljevati pravice dostopa do datotek le preko spletnega vmesnika.

- podprtost v različnih okoljih: oba sistema lahko uporabljajo uporabniki na vseh najbolj pogostih operacijskih sistemih.
- integriteta komunikacije med odjemalcem in strežnikom: integriteto komunikacije med odjemalcem in strežnikom zagotavlja protokol SSH pri sistemu Git in protokol SSL pri sistemu OwnCloud.

Poglavje 6

Zaključek in odprta vprašanja

Zastavljene cilje oziroma željo po brezhibni uporabniški izkušnji smo v tej nalogi le delno dosegli. Spoznali smo, da v primeru delitve omrežja, izrek CAP omejuje doseganje visoke dosegljivosti skupaj z nedeljivo usklajenostjo. Kompromis med njima je potrebno nastaviti pri implementaciji vsake storitve posebej. Pri nadzoru dostopa do datotek nam tudi ni uspelo najti idealnega sistema. Dodeljevanje dostopa je bilo potrebno realizirati s spletnim vmesnikom, saj noben sistem ni omogočal učinkovite vgradnje nadzora dostopa v lokalni datotečni sistem odjemalca.

6.1 Omejitve izreka CAP

Izrek CAP pravi, da je v porazdeljenem sistemu nemogoče doseči nedeljivo usklajenost, dosegljivost in odpornost na delitev omrežja sočasno [16]. Ko so vozlišča sistema povezana, ni potrebno žrtvovati ne dosegljivosti, ne usklajenosti. Posledice izreka CAP postanejo aktualne v trenutku ko nastopi delitev omrežja. Na to mora biti načrtovalec sistema pozoren že pri zasnovi. Od narave storitve je odvisno, katera lastnost je bolj pomembna. Če je za storitev pomembna dobra dosegljivost, je potrebno sistem nastaviti tako, da je v primeru delitve omrežja dovoljena manjša stopnja usklajenosti v zameno za visoko dosegljivost. Če je pomembna usklajenost datotek, je med

delitvijo omrežja potrebno zmanjšati dosegljivost storitve (npr. onemogočiti dostop za pisanje). Če imamo opravka s storitvijo, kjer sta zelo pomembni lastnosti visoke dosegljivosti in usklajenosti nastopijo večje težave. Delitve omrežja so v realnem svetu vedno prisotna, tako ne moremo žrtvovati odpornosti na delitev. V tem primeru lahko samo zmanjšamo možnost nastanka delitev omrežja. To lahko naprimer storimo s postavitvijo vozlišč v isti podatkovni center, uporabo boljše strojne (tudi programske) opreme, z uporabo več fizičnih povezav itd.

Kljub vsem preventivnim ukrepom pred delitvami omrežja, je možnosti za njen nastanek nemogoče popolnoma izničiti. V realnem svetu je tako pri vsaki storitvi potrebno računati na pojav delitev in biti posledično pripravljen na kompromis med visoko dosegljivostjo in nedeljivo usklajenostjo.

6.2 Omejitve pri dodeljevanju dostopa

Nobena od opisanih mogočih implementacij ni prepričljivo rešila problema dodeljevanja dostopa do datotek. Pri sistemu z uporabo NFS, je težava v tem, da odjemalec z korenskim dostopom lahko ustvari račun z UID-jem, ki je enak tistemu, ki ima dovoljenje za želen dostop do poljubne datoteke pripetega diskovnega prostora (z izjemo UID-ja 0). Sistem z uporabo WebDAV-a uporablja svojo implementacijo seznama nadzora dostopa. Po tej plati dosti obeta, vendar je v praksi težava v različni podprtosti standarda in slabi združljivosti različnih strežnikov in odjemalcev.

Pri sistemu, ki uporablja Git, se med odjemalcem in strežnikom vzpostavi SSH povezava. Certifikat, ki ga odjemalec uporabi pri vzpostavitvi povezave, določa kateri UID se dodeli procesu, ki dostopa do oddaljenega odložišča. Pri takem načinu dostopa do oddaljenega odložišča, je dostop do datotek zaščiten z ACL sistemom do katerega dostopamo. Slabost tega načina je, da je potrebno pred dostopom prenesti javni ključ odjemalca na sistem, do katerega želi dostopati.

Dropbox uporablja lasten sistem za nadzor dostopa. Sistem je nekoliko

omejen, saj za posamezne datoteke ni mogoče nastavljati dostopnih pravic. Dostop za branje in pisanje lahko določimo le celotni mapi. Sistem ima še eno pomanjkljivost. Dodeljevanje dostopa je mogoče le preko spletnega vmesnika. Vprašanje učinkovitega nadzora dostopa do datotek ostaja odprto.

6.3 Uporabniška izkušnja

Pri obeh implementiranih sistemih je bil poudarek na dosegljivosti datotek. V ta namen smo izbrali odjemalca, ki hranita lokalno kopijo datotek. Datoteke so uporabniku dosegljive tudi, ko nima povezave z medmrežjem. Odjemaleca uskladita kopiji datotek, ko se povezava ponovno vzpostavi. Nekoliko manj prijazna uporabniška izkušnja je, ko želi uporabnik datoteke deliti z drugimi uporabniki. Za dodelitev pravic za dostop do datotek mora uporabiti spletni vmesnik.

Literatura

[1] D.J. Abadi, "Consistency Tradeoffs in Modern Distributed Database System Design", Computer, Februar 2012, str. 37-42.

[2] E. Brewer, "Pushing the CAP: Strategies for Consistency and Availability", Computer, Februar 2012, str. 23-29.

[3] CentOS, "CentOS: The Comunity ENTerprise Operating System", Dostopno na: https://www.centos.org/, Dostopano: 20.9.2012.

[4] CentOS, "CentOS Deployment Guide - Securing NFS" Dostopno na: http://www.centos.org/docs/5/html/Deployment_Guide-en-US/s1-nfs-security.html, Dostopano na: 20.9.2012.

[5] Cgit, "Cgit", Dostopno na: http://hjemli.net/git/cgit/about/, Dostopano: 20.9.2012.

[6] S. Chacon, "Pro Git", Dostopno na: http://git-scm.com/book, Dostopano: 20.9.2012.

[7] G. Clemm, IBM, J. Reschke, greenbytes, E. Sedlar, Oracle Corporation, J. Whitehead, U.C. Santa Cruz, "Web Distributed Authoring and Versioning (WebDAV) Access Control Protocol", RFC 3744, Maj 2004.

[8] D. Drager, "5 Windows Git Clients to 'Git' the Job Done", Oktober 2010, Dostopno na: http://www.makeuseof.com/tag/5-windows-git-clients-git-job/, Dostopano: 20.9.2012.

[9] Dropbox, "Dropbox", Dostopno na: www.dropbox.com, Dostopano: 20.9.2012.

[10] L. Dusseault, "WebDav: Next Generation Collaborative Web Authoring", Prentice Hall, 1. izdaja, 2003, str. 338-347.

[11] L. Dusseault, Ed., CommerceNet, "HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV)", RFC 4918, Junij 2007.

[12] R. Fielding, UC Irvine, J. Gettys, Compaq/W3C, J. Mogul, Compaq, H. Frystyk, W3C/MIT, L. Masinter, Xerox, P. Leach, Microsoft, T. Berners-Lee, W3C/MIT, "Hypertext Transfer Protocol – HTTP/1.1", RFC 2616, Junij 1999.

[13] Git, "Git" Dostopno na: http://git-scm.com/ Dostopano: 20.9.2012.

[14] S. Hull, 'Reality Check: Five Nines: A Telecom Myth", Januar 2009, Dostopno na: http://www.cable360.net/ct/operations/testing/Reality-Check-Five-Nines-A-Telecom-Myth Pridobljeno: 20.9.2012.

[15] B. W. Lampson, "Protection", Proceedings of the 5th Princeton Conference on Information Sciences and Systems, Princeton, 1971, str. 437.

[16] N. Lynch, S. Gilbert, "Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-tolerant Web Service", ACM SIGACT News, Junij 2002, str. 51-59.

[17] N. Lynch, S. Gilbert, "Perspectives on the CAP Theorem", Computer, Februar 2012, str. 30-35.

[18] Gitalist, "Gitalist", Dostopno na: http://www.gitalist.com/, Dostopano: 20.9.2012.

[19] Gitlab, "Gitlab", Dostopno na: http://gitlabhq.com/, Dostopano: 20.9.2012.

[20] Gitolite, "Gitolite", Dostopno na: https://github.com/sitaramc/gitolite#readme, Dostopano: 20.9.2012.

[21] Gitorious, "Gitorious", Dostopno na: http://gitorious.org/, Dostopano: 20.9.2012.

[22] GitWeb, "GitWeb", Dostopno na: http://sourceforge.net/apps/trac/sourceforge/wiki/GitWeb repository browser, Dostopano: 20.9.2012.

[23] Information Sciences Institute, University of Southern California, "Transmission Control Protocol", RFC 793, September 1981.

[24] M. S. Miller, J. S. Shapiro, "Paradigm Regained: Abstraction Mechanisms for Access Control", Advances in Computing Science - ASIAN 2003, str. 224-242.

[25] D. Oh, "6 Useful Graphical Git Client for Linux", Januar 2012, Dostopno na: http://maketecheasier.com/6-useful-graphical-git-client-for-linux/2012/01/18, Dostopano: 20.9.2012.

[26] Owncloud, "Downloads", Dostopno na: https://owncloud.com/download, Dostopano: 20.9.2012.

[27] OwnCloud, "OwnCloud" Dostopno na: http://owncloud.org, Dostopano: 20.9.2012.

[28] OwnCloud, "OwnCloud Installation", Dostopno na: http://owncloud.org/support/install/, Dostopano: 20.9.2012.

[29] OwnCloud, "OwnCloud LDAP backend", Dostopno na: http://owncloud.org/support/ldap-backend/, Dostopano: 20.9.2012.

[30] E. Rescorla, RTFM, Inc., "HTTP Over TLS", RFC 2818, Maj 2000.

[31] M. Shapiro, C. Baquero, N. Preguiça, M. Zawirski, "Convergent and Commutative Replicated Data Types," Bulletin of the EATCS, Junij 2011, str. 67-88.

[32] S. Shepler, B. Callaghan, D. Robinson, R. Thurlow, Sun Microsystems, Inc., C. Beame, Hummingbird Ltd., M. Eisler, D. Noveck, Network Appliance, Inc., "Network File System (NFS) version 4 Protocol", RFC 3530, April 2003.

[33] S. Shepler, Ed., Storspeed, Inc., M. Eisler, Ed., D. Noveck, Ed., NetApp, "Network File System (NFS) Version 4 Minor Version 1 Protocol", RFC 5661, Januar 2010.

[34] Shiningthrough, "Mac OS X Git Clients Roundup", September 2010, Dostopno na: http://shiningthrough.co.uk/Mac-OS-X-Git-Clients-Roundup, Dostopano: 20.9.2012.

[35] A. Silberschatz, P.B. Galvin, G. Gagne, "Operating System Concepts with Java, 7. izdaja, 2007.

[36] C. Smith, "Linux NFS-HOWTO", poglavje "Security and NFS", Dostopno na: http://nfs.sourceforge.net/nfs-howto/ar01s06.html, Dostopano: 20.9.2012.

[37] Sourceforge, "Librsync", Dostopno na: http://librsync.sourceforge.net/, Dostopano: 20.9.2012.

[38] SparkleShare, "Sparkleshare", Dostopno na: http://sparkleshare.org/, Dostopano: 20.09.2012.

[39] L. Vogel, "Git Tutorial", Dostopno na: http://www.vogella.com/articles/Git/article.html, Dostopano: 20.9.2012.

[40] WinSCP, "Understanding SSH", Dostopno na: http://winscp.net/eng/docs/ssh, Dostopano: 20.9.2012.

[41] K. Zeilenga, Ed., "Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map", RFC 4510, Junij 2006.