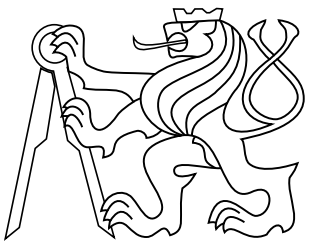
diplomová práce

Distribuované úložiště dat - klientská část

Bc. Jan Janura



Květen 2014

Vedoucí práce: Ing. Peter Macejko

České vysoké učení technické v Praze
Fakulta elektrotechnická, Katedra počítačů

Poděkování

Rád bych poděkoval svým rodičům, za podporu při studiu a psaní této diplomové práce. Dále bych chtěl poděkovat vedoucímu diplomové práce Ing. Peteru Macejkovi za konzultace, dobré rady ohledně práce a trpělivost.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškeré použité informační zdroje v souladu s metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Abstrakt

Práce se zabývá návrhem subsystému a následnou pilotní implementací zajišťující přístup jednotlivých klientů k uloženým datům. Při návrhu se zohledňují požadavky na redundantnost a rychlou aktualizaci změn. Výsledkem této práce je funkční program sloužící k jednoduchému přístupu do systému, který umožnuje bezpečné uložení dat do sítě, jejich načtení a naznačuje směr dalšího vývoje.

Klíčová slova

Klientská aplikace, FUSE, Access server

Abstrakt

The thesis daels with a design of subsystem and its implementation making possible for users to access their data. During the design, many requirements were taken into account (e.g. the redundancy, the ability to adapt to changes quickly). As a result of this thesis the program communicating with a system is presented. The system has functionalities as follows - safe storing of user's data and its reading through the network. The thesis also discusses next possibilities of development.

Keywords

Client application, access server

Obsah

1. Úvod **1**

2. Popis problému, specifikace cíle **3**
2.1. Bližší popis problému . . . 3
2.1.1. Zabezpečení . . . 3
2.1.2. Dostupnost . . . 3
2.1.3. Spolehlivost . . . 3
2.2. Specifikace cíle . . . 3

3. Rešerše existujících řešení **5**
3.1. Dropbox . . . 5
3.1.1. Použití . . . 5
3.1.2. Bezpečnost . . . 6
3.1.3. Sdílení a synchronizace . . . 6
3.1.4. Webové rozhraní . . . 6
3.1.5. Klientská aplikace . . . 7
3.1.6. Testování . . . 7
3.1.7. Přehled vlastností a nabízených služeb . . . 7
3.2. Ubuntu One . . . 8
3.2.1. Použití . . . 8
3.2.2. Bezpečnost . . . 9
3.2.3. Sdílení a synchronizace . . . 9
3.2.4. Webové rozhraní . . . 9
3.2.5. Klientská aplikace . . . 10
3.2.6. Testování . . . 10
3.2.7. Přehled vlastností a nabízených služeb . . . 10
3.3. CloudMe . . . 11
3.3.1. Použití . . . 11
3.3.2. Bezpečnost . . . 11
3.3.3. Sdílení a synchronizace . . . 11
3.3.4. Webové rozhraní . . . 12
3.3.5. Klientská aplikace . . . 12
3.3.6. Testování . . . 12
3.3.7. Přehled vlastností a nabízených služeb . . . 13
3.4. ownCloud . . . 13
3.4.1. Použití . . . 13
3.4.2. Bezpečnost . . . 14
3.4.3. Sdílení a synchronizace . . . 14
3.4.4. Webové rozhraní . . . 14
3.4.5. Klientská aplikace . . . 14
3.4.6. Testování . . . 15
3.4.7. Přehled vlastností a nabízených služeb . . . 15
3.5. Testování na referenčním stroji . . . 16
3.5.1. Dropbox . . . 17
3.5.2. UbuntuOne . . . 17
3.5.3. CloudMe . . . 17
3.5.4. ownCloud . . . 18

3.6. Zhodnocení 19

4. Analýza a návrh řešení **22**
4.1. Struktura systému 22
4.1.1. Klientská aplikace 22
4.1.2. Access server 22
4.1.3. Datové centrum 22
4.1.4. Komunikace 22
4.1.5. Indentifikace 22
4.1.6. Zabezpečení systému 23
4.2. Obecné požadavky 23
4.2.1. Multiplatformní systém 23
4.2.2. Volba operačního systému 23
4.3. Funkční požadavky z pohledu klienta 24
4.4. Nefunkční požadavky z pohledu klienta 24
4.5. Požadavky z pohledu provozovatele 24
4.5.1. Minimalizace přenášených dat 24
4.5.2. Rozchunkování souborů 24
4.5.3. Rozlišení jednotlivých uživatelů 24
4.6. Klientská aplikace 24
4.6.1. Filesystem in Userspace 26
4.6.2. Struktura klientské aplikace 27
4.6.3. Funkce aplikace 27
4.7. Access server 30
4.7.1. Klientský modul 30
4.7.2. Data center modul 30
4.7.3. User Handler 31
4.7.4. Route Handler 31
4.7.5. Modul vnitřní komunikace 31
4.7.6. Funkce access serveru 31
4.8. Komunikační model 32
4.8.1. Komunikace klienta s access serverem 32
4.8.2. Formát zpráv při komunikaci klienta s access serverem 32
4.8.3. Komunikace access serveru s datovým centrem 33
4.8.4. Komunikace mezi access servery 34

5. Realizace **35**
5.1. Vývojové prostředí 35
5.2. Funkce klientské aplikace 35
5.2.1. Fronta požadavků 36
5.2.2. Obsluha fronty požadavků 36
5.2.3. FUSE modul 36
5.2.4. Rozchunkování souborů 41
5.2.5. Šifrování souborů 41
5.2.6. Odesílání dat na AS 41
5.2.7. Registrace uživatele 43
5.2.8. Generování klíče pro šfirování 43
5.2.9. Přihlášení uživatele 44
5.2.10. Inicializace aplikace 44
5.2.11. Nahrání souborů 44

5.2.12. Mazání souborů . . . 45
5.2.13. Sdílení souborů mezi uživateli . . . 45
5.2.14. Synchronizace více uživatelů . . . 45
5.3. Access server . . . 46
5.3.1. Klientský modul . . . 46
5.3.2. User handler . . . 46
5.3.3. Route handler . . . 47
5.3.4. Data center modul . . . 47
5.3.5. Modul vnitřní komunikace . . . 48

6. Testování **49**
6.1. Testování klientské aplikace . . . 49
6.2. Testování access serveru . . . 49
6.3. Testování v rámci celého systému . . . 50
6.3.1. Tetstovací sestava . . . 50
6.3.2. Výsledky měření . . . 50

7. závěr **54**

Přílohy

A. Instalační a uživatelská příručka klientské aplikace **55**
A.1. Prostředí a běh . . . 55
A.2. Kompilace . . . 55
A.3. Nastavení . . . 56
A.4. Spuštění . . . 56

B. Instalační a uživatelská příručka access serveru **58**
B.1. Prostředí a běh . . . 58
B.2. Kompilace . . . 58
B.3. Nastavení . . . 58
B.4. Spuštění . . . 60

C. Obsah přiloženého CD **61**

Literatura **66**

Seznam obrázků

2. Celková koncepce systému 23
3. FUSE modul jádra a komunikace s FUSE knihovnou 25
4. Struktura klientské aplikace 28
5. Struktura access serveru 30

6. Dešifrování a čtení souboru 39
7. Zápis a šifrování dat do souboru 40
8. Generování klíče pro šifrování 43

9. Schéma testovací sestavy 51

Seznam tabulek

1. Přehled naměřených časů a rychlostí přenosu souborů u služby Dropbox 7
2. Přehled funkcionalit u jednotlivých verzí služby 8
3. Přehled naměřených časů a rychlostí přenosu souborů u služby Ubuntu One 10
4. Přehled funkcionalit u jednotlivých verzí služby 11
5. Přehled naměřených časů a rychlostí přenosu souborů u služby CloudMe 13
6. Přehled naměřených časů a rychlostí přenosu souborů u služby ownCloud 15
7. Přehled testovaných souborů na virtuálním stroji 16
8. Testování služby Dropbox na referečním stroji 17
9. Testování služby UbuntuOne na referečním stroji 17
10. Testování služby CloudMe na referečním stroji 17
11. Testování služby ownCloud na referečním stroji 18

12. Přehled testovaných dat a naměřených časů I 52
13. Přehled testovaných dat a naměřených časů II 52

Zkratky

FUSE	Filesystem in Userspace
AS	Access Server
ASCII	American Standard Code for Information Interchange
TCP	Transmission Control Protocol
AES	Advanced Encryption Standard
IP	Internet Protocol
PEM	Privacy Enhanced Mail

1. Úvod

Problém ukládání dat a zejména jeho způsob se řeší již po desetiletí. Spolu s rozvojem vědy a techniky se neustále zdokonaluje, zrychluje a přizpůsobuje nárokům koncových uživatelů. Od prvopočátků, kdy se data ukládala převážně na děrné štítky nebo později na magnetické pásky, doba pokročila k využívání ať již komerčních nebo nekomerčních datových úložišť, která jsou nyní moderní a získávají stále větší popularitu mezi běžnými uživateli. Dosud ovšem nebyl nalezen systém, který by nabízel ideální způsob ve všech směrech řešení, protože některé funkce nárokované na tyto systémy se vzájemně vylučují a hledá se ideální kompromis.

Mezi největší problémy při přenosu dat patří např. zabezpečení. Způsobů jak zabezpečit data proti nežádoucí osobě je hned několik. Jedno z řešení je využívání lokálního úložiště u sebe doma nebo v malé firemní síti, které dává garanci zamezení přístupu cizímu návštěvníkovi, ale když zaměstnanec potřebuje přistupovat k datům na služební cestě nebo se chceme pochlubit fotkami z dovolené příbuzným nebo známým? Nosit s sebou celé úložiště je nepohodlné a v případě firemního řešení i nereálné. To nás přivádí na druhý z hlavních požadavků na systém a to je přenositelnost a dostupnost uložených dat. Je žádoucí, aby měl uživatel přístup ke svým datům odkudkoliv, bez ohledu na to, jestli je fyzicky vzdálen od úložiště deset metrů nebo tisíc kilometrů.

Hlavní kritérium, podle kterého se většina uživatelů řídí, je však pořizovací cena celého systému. Pokud budeme chtít mít data perfektně zabezpečena proti zneužití, budeme chtít k datům přistupovat odkudkoliv a budeme požadovat vysokou spolehlivost, pak cena takového systému bude nepochybně několikrát vyšší než cena systému, který tyto požadavky vyřeší kompromisem nebo nějakou funkcionalitu podporovat vůbec nebude.

V dnešní době online virtuálního světa, kdy se sebemenší firma neobejde bez internetového připojení, se řeší tato problematika pomocí tzv. cloudového úložiště, která pracují tak, že uživatel nahraje data do jedné složky a ta se pak objeví ve všech propojených účtech. Tyto úložiště většinou podporují velký počet připojených koncových bodů a není v silách jednoho přístupového serveru, aby všechny požadavky vyřizoval sám. Pokud by tomu tak bylo, stává se toto místo úzkým hrdlem systému a v případě výpadku se stává celý systém nefunkční. Proto se využívá distribuovaného výpočtu, který je složitější na vývoj, správu i údržbu, ale umožňuje rozdělení zátěže na více strojů a zamezuje výpadek služby v momentě nedostupnosti některého z uzlů.

S problematikou cloudových úložišť souvisí i svěření svých citlivých dat cizímu poskytovateli, který je bude následně spravovat. Tyto všechny důvody vedou k vývoji systému, který půjde snadno nasadit, bude decentralizovaný a organizace, která se ho rozhodne využívat, ho bude moci i sama spravovat.

V první části své práce se budu zabývat řešerší několika současných synchronizačních služeb. Druhá část bude obsahovat problematiku integrace klientské části do operačního systému a řešení přístupového bodu do navrhovaného systému, který by měl být s ohledem na výše popsané problémy řešen distribuovaným systémem se zabezpečenou komunikací ke všem ostatním částem systému.

2. Popis problému, specifikace cíle

2.1. Bližší popis problému

Jak jsem již naznačil v úvodu své práce, v oblasti vývoje datových úložišť jsou čtyři hlavní aspekty, které je nutné brát v úvahu. Patří k nim zabezpečení, dostupnost, spolehlivost a cena. Každý z těchto aspektů se významnou měrou podílí na kvalitě úložiště a zároveň ovlivňuje další parametry, takže nelze jednoznačně určit nejlepší řešení.

2.1.1. Zabezpečení

Jedná se o velice složitý problém, na který lze pohlížet několika různých úhlů. První je z hlediska samotného uložení dat, kdy se snažíme informace na disku zašifrovat tak, aby i při fyzické ztrátě nebo odcizení úložiště byly informace neustále chráněny a nemohli být zneužity. Druhou možností pohledu na situaci je zabezpečení dat při komunikaci. Pokud jsou data uložená na nějakém serveru a klient si o ně zažádá, je žádoucí, aby nikdo třetí tyto informace během přenosu neodposlechl. Obecně platí, že čím více se data budeme snažit chránit, tím více omezíme jejich dostupnost.

2.1.2. Dostupnost

Stejně jako v případě bezpečnosti, nelze ani na dostupnost pohlížet jednotně. Můžeme mít na systém nárok z hlediska časové dostupnosti, kdy požadujeme přístup k souboru kdykoliv. Tento problém velice úzce souvisí se spolehlivostí systému. Druhý pohled na dostupnost je takový, že chceme k datům přistupovat odkudkoliv. To lze vyřešit vystavením úložiště na internet, ale to přináší problém s bezpečností a optimálním přístupem uživatelů k datům a problém bezpečnosti.

2.1.3. Spolehlivost

Dalším neméně důležitým faktorem je spolehlivost systému. Existuje mnoho mechanismů pro řešení spolehlivosti komunikačních cest nebo zálohování dat, která mají za úkol jediné. Zaručit dostupnost dat při výpadku některé komponenty i po krátkou dobu. Toto řešení je však velice složité a nikdy nejde zajistit 100% spolehlivost. Jedná se však o nejdůležitější část každého datového úložiště.

Vhodný způsob minimalizace všech zmíněných problémů je nasadit distribuované řešení, které vyřeší výše popsané problémy. Nutností bude mezi prvky systému začlenit šifrování pro zajištění bezpečnosti nebo šifrovat data v klientské aplikaci.

2.2. Specifikace cíle

Cílem této práce je seznámit čtenáře s několika aktuálně dostupnými úložišti a následně z informací nasbíraných při testování těchto produktů navrhnout a zrealizovat

pilotní implementaci přístupového bodu k celému systému. Řešení by mělo být navrženo tak, aby se co nejvíce blížilo firemnímu nasazení a nechalo se v případě potřeby rozumným způsobem rozšířit. Tento přístupový bod bude tvořen klientskou aplikací s přístupovým serverem.

3. Rešerše existujících řešení

V této kapitole budu testovat a porovnávat několik již existujících řešení. Testování probíhalo behem zimního sesmestru roku 2013/2014 a je možné, že některé informace již nejsou aktuální. Zaměřoval jsem se na poskytované služby a způsob přístupu k aplikaci. Jedná se o tyto následující úložiště:

- Dropbox
- Ubuntu one
- CloudMe
- ownCloud

Prvotní testování probíhalo na osobním notebooku s následujícími parametry:

- Systém Windows 7 64bit
- Procesor Inter(R) Core(TM)2 Duo CPU T 6500 2,1GHz
- 4GB operační paměť RAM
- Internetové připojení ke kolejní síti Pod-o-lee se symetrickým 1Gb/s připojením

3.1. Dropbox

Služba Dropbox byla založena v roce 2007 Drew Houstonem spolu se studentem MIT Arash Ferdoesiem a je typickým příkladem dnešního cloudového úložiště. Počet uživatelů, kteří službu využívají překonal hranici 100 milionů a každý den se nahraje přes miliardu souborů. Každý uživatel v průměru obsluhuje pět připojených zařízení a z těchto faktů vychází, že Dropbox obsluhuje přes půl miliardy synchronizovaných produktů a očekává se další nárůst. Podle Hustona, však produkt dostatečně nevyužívají mobilní operátoři, kteří by mohli prodávat extra kapacitu k tarifům, nezohledňovat datové přenosy s úložištěm nebo zvýhodňovat skupiny lidí, kteří sdílejí stejná data[1].

3.1.1. Použití

Nový uživatel, který se rozhodne službu využívat, si musí nejprve pomocí emailové adresy a hesla zaregistrovat účet (free verze nabízí kapacitu 2GB) na adrese https://www.dropbox.com/. Po nainstalování lokální aplikace do systému se vytvoří speciální složka, která bude sloužit pro synchronizaci dat s úložištěm na serveru a případně i s dalšími zařízeními, která jsou připojena k účtu. Služba je integrována do systému, takže se uživatel přes ikonu v oznamovací části dozví stav synchronizace nebo zde může nastavit různé parametry pro přenos dat. Ke službě lze přistupovat i přes webové rozhraní, které nabízí kromě běžné práce se soubory i možnost obnovovat smazané soubory a složky až do doby 30 dní nazpět. U souborů navíc lze obnovit jakoukoliv předchozí verzi v tomto časovém úseku. Úložiště je možno využívat i ve dvou placených variantách, které nabízí větší úložný prostor nebo neomezené doby obnovy změněných souborů[1].

3.1.2. Bezpečnost

Z hlediska bezpečnosti je nejrizikovější přístup přes webové rozhraní. Z tohoto důvodu Dropbox nabízí svým zákazníkům možnost využívat při přístupu přes webové rozhraní dvoufázové přihlašování a omezit tak bezpečností riziko spojené se zneužitím účtu.

Uživateli přijde také pokaždé email, když se k účtu připojí nějaké nové zařízení, které s účtem ještě nekomunikovalo. Webové rozhraní pak nabízí přehled zařízení, které s účtem spolupracovalo a uživatel tak má přehled alespoň o částečném pohybu svých dat.

Synchronizovaná data z lokální aplikace jsou během přenosu zabezpečena pomocí SSL a na serveru se šifrují heslem, které je generování přímo službou a uživatel ho nemůže nijak zvolit ani částečně ovlivnit.

Velkým problémem však je, že se uživatelské jméno a heslo zadává pouze při nastavení klientské aplikace, ale při komunikaci se používá ID klienta, který vygeneroval server a poslal ho zpět klientské aplikaci. V případě zjištění tohoto ID, které je uloženo v SQLite databázi a vložením do databáze na svém počítači, naklonujeme zařízení, ze kterého bylo ID odcizeno. Získáváme tak v budoucnu plnou kontrolu nad synchronizovanými daty aniž by se o tom poškozený uživatel jakkoliv dozvěděl[2].

3.1.3. Sdílení a synchronizace

Pokud se uživatel rozhodne, že bude s někým sdílet svá data, může k tomu využít webové rozhraní nebo integrovanou funkci v systému. Ta pouze vytvoří odkaz a přesměruje uživatele na webové rozhraní. Možnosti sdílení jsou v zásadě dvě. První nabízí možnost sdílet veřejně jakoukoliv složku, ke které se mohou řetězově připojovat další uživatelé. Druhá umožňuje s kýmkoliv sdílet data pomocí pozvánky přes email nebo veřejně vystavit soubor či složku na facebooku nebo twiteru. Data však nelze žádným způsobem dohledávat nebo prohledávat nadřazené adresáře.

Nejobvyklejší způsob jak synchronizovat data je nakopírovat soubory a adresáře do složky vytvořené službou, která zaznamená změnu a spustí se synchronizace se serverem. Služba uživateli v klientské aplikaci nabízí možnost nastavit synchronizaci pouze libovolných složek z defaultního adresáře, ale neumožňuje vybrat si konkrétní soubory, stejně jako neumožňuje přidružit libovolné složky ze systému[1].

3.1.4. Webové rozhraní

Webové rozhraní nabízí všechny potřebné funkce pro online správu souborů a adresářů. Uživatel zde má možnost:

- Vytváření nových souborů a složek
- Nahrávání souborů
- Mazání souborů a složek
- Přesouvání souborů a složek pomocí Drag and Drop
- Sdílení souborů a složek
- Obnova dřívějších verzí souboru
- Obnova smazaných souborů
- Přehrávání jednotlivých skladeb – neumožňuje přehrávání celých alb

- Vytváření fotoalb – chybí funkce pro automatické prohlížení fotek

3.1.5. Klientská aplikace

Klientská aplikace po nainstalování vytvoří v systému složku pro synchronizaci, ve které se nachází volba pro sdílení, funkce pro přesměrování na web nebo webový výčet předchozích verzí. V oznamovací části nabízí základní možnosti jak službu ovládat nebo nastavit základní parametry. Uživatel zde dostane informace o současné synchronizaci, odhadu zbývajícího času do konce synchronizace a může si zde nastavit rychlost přenosu pro nahrávání a stahování dat.

3.1.6. Testování

Dropbox patří na trhu s cloudovými úložišti k majoritním zástupcům, ale nenabízí oproti jiným službám žádné vlastnosti navíc. Při nahrávání souborů nemá problém synchronizovat jednotlivé soubory ihned, avšak při velkém množství malých souborů zpracovával data několik hodin a při tom tato data nebyla v polovině nahrávání dostupná ani přes webové rozhraní. Určitou výhodou může být opětovné nahrání trvale smazaných dat (konkrétně u 100 fotek), která se nahrála během desítek vteřin, a to i přes to, že na serveru byla trvale smazána. Služba uživateli nabízí stahování souborů ve formátu, v jakém jsou na serveru uloženy. V případě stahování celé složky ji služba zabalí do .zip archivu, avšak s velkým adresářem si služba neporadila a nahlásila chybu. Kladem zůstává její poměrně vysoká přenosová rychlost nahrávání u velkých souborů a oproti jiným testovaným službám rychlost operací ve webovém prostředí. Služba byla testována na přenos jednoho velkého souboru o velikosti 1,4GB, z pohledu uživatele běžného nahrávání sta fotek různých velikostí o celkové velikosti 331MB a 15761 souborů v podobě zdrojového kódu linuxového jádra. Naměřené časy s rychlostmi přenosu jsou v Tab. 1.

Počet souborů	Celková velikost	Čas nahrávání	Přenosová rychlost	Čas stahování	Přenosová rychlost
1	1.4GB	28 min	6.8 Mb/s	14 min	13.6 Mb/s
100	331 MB	3 min	14.8 Mb/s	3 min	14.8 Mb/s
15761	87.8 MB	1h 35min	126 kb/s	—	—

Tabulka 1. Přehled naměřených časů a rychlostí přenosu souborů u služby Dropbox

3.1.7. Přehled vlastností a nabízených služeb

- Nabízená základní kapacita 2GB, placená verze až 100GB
- Přístup k datům přes složku v systému nebo webové rozhraní
- Mazání souborů a složek
- Možnost částečné synchronizace
- Omezení při nahrávání souboru přes webové rozhraní na 300MB
- Multiplatformní podpora pro Windows, Mac OS X, Linux, Android, iOS nebo BlackBerry
- Verzování souborů 30 dní nazpět
- Sdílení souborů mimo komunitu
- Při přihlašování možnost dvoufázového potvrzení

- Šifrování souborů při ukládání algoritmem AES-256
- Šifrování při přenosu a synchronizaci pomocí Secure Socket Layer
- Pro vlastní ukládání dat služba používá Amazon Simple Storage Service
- Při placené verzi je stanovená hranice 300 000 souborů než se služba začne výrazněji zpomalovat

Kategorie	Free	Pro	Business
Cena	—	$99/rok	$795/rok pro 5 uživatelů $125/rok pro přidaného uživatele
Kapacita	2 GB	100 GB	Nelimitováno
Synchronizace	Ano	Ano	Ano
Sdílení	Ano	Ano	Ano
AES-256	Ano	Ano	Ano
SSL	Ano	Ano	Ano
Dvoufázové přihlašování	Ano	Ano	Ano
Neomezené obnovení souborů	Ne	Ano	Ano
Sdílení mimo komunitu	Ano	Ano	Ano

Tabulka 2. Přehled funkcionalit u jednotlivých verzí služby

3.2. Ubuntu One

Ubuntu One přestavuje další službu z řady cloudových úložišť. Provozuje ji firma Canonical primárně pro svůj systém Ubuntu a k datovému úložišti nabízí placenou službu streamování hudby v mobilních zařízeních a online nákup MP3 souborů v Ubuntu One Music Store s následným stažením do zařízení. Podstatnou změnou oproti jiným mnou testovaným systémům je přikupování kapacity postupně po částech a tím možnost přizpůsobit si prostor vlastním potřebám, bez zbytečného placení nevyužité kapacity[3] [4].

3.2.1. Použití

Pro využívání služeb, je nutné provést registraci účtu s prvotní kapacitou 5GB na webové adrese https://one.ubuntu.com/, kde se uživatel identifikuje emailovou adresou a heslem. Po nainstalování aplikace má uživatel možnost nastavit složky, které chce synchronizovat. Primárně se nastaví nově vytvořená služka UbuntuOne, kterou lze rozšířit o jakoukoliv další složku ze systému. Spuštěná aplikace se zobrazí v oznamovací oblasti a nabízí možnost sledovat aktuální nahrávání nebo nastavení limitu rychlosti pro přenos dat. Služba však není integrována do systému, takže uživatel nemá přehled o tom, které soubory nebo složky už má synchronizované, a které ještě na synchronizaci čekají a je odkázaný pouze na celkový stav procesu. Ke službě lze přistupovat i přes webové rozhraní, které nenabízí příliš komfortu pro práci se soubory, ale základní funkce pro editaci obsahu úložiště tu je. Služba také disponuje obnovou smazaných souborů jak z

ostatních zařízení, tak z webového rozhraní. Proces funguje tak, že pokud si uživatel smaže data na jednom zařízení, na ostatních se přesunou do složky Trash Folder, kde je možné je obnovit. Takto smazané soubory lze obnovit do 60 dnů. Po této době služba negarantuje obnovu všech souborů, protože jsou ze serveru mazána.

Se službou souvisí také Ubuntu One Music Store, kde si uživatel kupuje hudbu ve formátu MP3, která se nejdříve zkopíruje do datového úložiště Ubuntu One a následně pomocí synchronizace až do vlastního zařízení[3] [4].

3.2.2. Bezpečnost

Do roku 2012 nebyla služba příliš zabezpečena a jediný způsob jak chránit svá data bylo šifrování pomocí nějakého externího skriptu nebo programu. Nyní je tento problém částečně vyřešen a při využívání klientské aplikace se pro komunikaci mezi serverem a klientem používá SSL, šifrování na samotném serveru však stále chybí.

Při přístupu přes webové rozhraní trvá nepoměrnou dobu ověření přihlašovacích údajů. Velkým problémem je, že služba uživatele trvale přihlásí, a pokud se zapomene při opuštění služby odhlásit, automaticky je připojen při dalším navštívení úvodní stránky a nemusí zadávat přihlašovací údaje. Vzniká tak prostor pro neoprávněné zneužití účtu[3] [4].

3.2.3. Sdílení a synchronizace

Pokud se uživatel rozhodne podělit se o svůj obsah s někým dalším, záleží, zda bude sdílet soubor nebo celý adresář. V případě sdílení celé složky se odešle email s pozvánkou a odkazem na sdílenou složku. Podmínkou je, že uživatel musí mít účet, aby se k datům dostal. Tuto nevýhodu eliminuje druhý způsob sdílení nazvaný jako „publikování“. Ten vygeneruje odkaz, který lze jakýmkoliv způsobem distribuovat příjemcům. Má však nevýhodu v tom, že jej lze aplikovat pouze na jednotlivé soubory. Odkaz lze vygenerovat i v klientské aplikaci, ale sdílení složek umožňuje nastavit pouze webové rozhraní[3] [4].

Synchronizace dat probíhá standardním způsobem. Uživatel nakopíruje data do připravené složky, která zaznamená změnu dat a spustí se synchronizace. Aplikace nabízí možnost přidružit k defaultní složce i libovolnou složku z rodičovského adresáře, která se následně objeví vedle defaultní složky s volbou, zda ji chce uživatel synchronizovat nebo ne.

3.2.4. Webové rozhraní

Webové prostředí uživateli nenabízí takový komfort jako například Dropbox a nenabízí ani nástroj Drag and Drop, ale nejnutnější funkce pro editaci úložiště tu jsou. Rozhraní celkově působí jednoduše, avšak je účelné a pro základní práci plně dostačující. Jeho nevýhodou je přihlašování a absence funkce pro stažení celé složky. Výčet základních funkcí, které webové rozhraní nabízí je následující:

- Nahrání souboru
- Vytvoření složky
- Sdílení složek
- Publikování souborů
- Obnova smazaných souborů

- Stahování souborů

3.2.5. Klientská aplikace

Klientská aplikace po nainstalování do systému vytvoří složku pro synchronizaci, která obsahuje adresář pro sdílené soubory s účtem. Hlavním přístupovým bodem aplikace je oznamovací oblast systému. Detail ikony nabízí kromě sledování přenášených souborů nebo pozastavení synchronizace také otevření hlavního okna aplikace, ve kterém jsou všechny podstatné funkce pro nastavení služby. Uživatel vidí sdílené složky, připojená zařízení ke službě za dobu jejího používání nebo nastavení limitů rychlosti pro nahrávání a stahování dat.

3.2.6. Testování

Služba Ubuntu One patří na trhu s webovými úložišti mezi známější produkty, ale výsledky testování na třech typech souborů (velký ISO soubor, 100 fotografií a 15761 malých souborů) představovaly spíše podprůměr. Opět se potvrdil předpoklad, že se spousty malých souborů si služba příliš neporadí a nahrávaný čas překročil 4h.Oproti tomu jednotlivé soubory (fotky) nebo jeden velký soubor služba nahrála obstojně a čas se pohyboval v řádech minut. Naměřené testovací časy jsou zaznamenány v Tab. 3.

Další negativní vlasnost představuje možnost stahovat pouze soubory a nikoliv celé složky nebo mazání souborů přes webové rozhraní. Služba dokáže smazat najednou jen určité množství souborů a v případě většího požadavku po několika vteřinách zahlásí chybu a žádná změna se neprovede.

Počet souborů	Celková velikost	Čas nahrávání	Přenosová rychlost	Čas stahování	Přenosová rychlost
1	1.4GB	31 min	6.1 Mb/s	13 min	14.7 Mb/s
100	331 MB	15 min	2.94 Mb/s	—	—
15761	87.8 MB	4h 20min	45 kb/s	—	—

Tabulka 3. Přehled naměřených časů a rychlostí přenosu souborů u služby Ubuntu One

3.2.7. Přehled vlastností a nabízených služeb

- Základní kapacita 5GB s možností přikupování po 20GB
- Přístup k datům přes složku v systému nebo webové rozhraní
- Možnost rozšíření synchronizace o další složky ze systému
- Sdílení dat mimo komunitu
- Nastavení oprávnění pro sdílené soubory
- Obnova smazaných dat
- Podporované platformy Windows od verze XP, Mac OS, Android, iPhone
- Šifrování komunikace pomocí SSL
- Rozšíření úložiště o streamování hudby přes Ubuntu One Music Store

	Ubuntu One Free	Music Streaming
Cena	—	$3.99/měsíc nebo $39.99/rok
Kapacita	5 GB	+20 GB
Synchronizace	Ano	Ne
Podpora PC	Ano	Ne
Podpora mob. zařízení	Ano	Ano

Tabulka 4. Přehled funkcionalit u jednotlivých verzí služby

3.3. CloudMe

Další z řady cloudových úložišť pro online zálohování a synchronizování dat se jmenuje CloudMe. Jedná se o švédský produkt z roku 2011, od zakladatele Daniela Arthurssona, který je nyní majoritně vlastněn firmou Xcerion. Dříve byla tato služba provozována pod jménem iCloud, ale tento název a doménu iCloud.com koupila v roce 2011 firma Apple údajně za 4,5 milionů dolarů. Po přejmenování se služba rozdělila na CloudMe, což je klasické úložiště pro synchronizaci dat a cloudTop, které se zabývá virtuálním cloud desktopem, zahrnuje ukládání souborů a byl hlavním tahákem firmy před rozdělením. Nyní je služba používána ve 229 zemích s vlastními datovými servery ve Švédsku a zemích Evropské Unie[**icloud**].

3.3.1. Použití

Pro využívání cloudového úložiště je nutná registrace na adrese www.cloudme.com, která vyžaduje několik obligátních údajů včetně uživatelského jména a hesla pro budoucí přihlašování. Základní verze nabízí uživateli kapacitu 3GB prostoru zdarma a zpoplatněné kapacity 25, 100 a 500GB. Pro využívání klientské aplikace je nutné nainstalovat službu do systému, kde bude následně dostupná v oznamovací oblasti. CloudMe nabízí stejně jako jiní poskytovatelé podobných typů úložišť synchronizaci a online zálohu dat přes složku Blue Folder, připojení několika zařízení s různými operačními systémy nebo sdílení dat mimo komunitu. Nabízí také možnost přistupovat k datům přes webové rozhraní, které je velice uživatelsky příjemné a nabízí spoustu multimediálních funkcí např. pro přehrávání hudby nebo prohlížení celých alb. Nevýhodou služby oproti ostatním konkurentům je omezení velikosti nahrávaného souboru na 150MB.

3.3.2. Bezpečnost

Většina úložišť nenabízí velké množství možností při zabezpečení uživatelských dat a CloudMe není výjimkou. Je tu pouze možnost šifrovat data při přenosu pomocí SSL s SSL Extend Validation Certificate, ale jiný způsob jak by si uživatel ochránil svá data není. Má sice možnost využít integrace programu WinZip 17.5 jako software třetích stran do služby a může si soubory archivovat, ale nijak to neřeší problém, že data nelze zašifrovat přímo na serveru[5].

3.3.3. Sdílení a synchronizace

Sdílení uživateli nabízí tři způsoby, jak se podělit o svá data. První z nich nabízí běžné odeslání pozvánky přes email, Facebook nebo Twitter a následný přístup přes odkaz.

Druhou i třetí zajímavou možností je sdílet složku s kýmkoliv opět pomocí pozvánky, ke které se přikládá přístupové heslo. Kdo pak zná heslo, může do složky vkládat svá data, které se objeví i u vlastníka. Ve všech třech případech má uživatel možnost stáhnout si celý sdílený obsah v jednou zip archivu. Druhá varianta však vyžaduje jméno a heslo, které se zasílalo spolu s pozvánkou. Uživatel má možnost synchronizovat svá data přes nepřeberné množství zařízení jako je notebook s různým operačním systémem, mobilní telefon, tablet, ale také podporuje Samsung SmartTV nebo technologii WebDAV pro připojení služby jako síťového disku.

3.3.4. Webové rozhraní

Webové rozhraní působí na uživatele velice příjemně a intuitivně, k čemu dopomáhá i předpřipravená struktura složek pro klasické dokumenty, fotky nebo hudbu. Uživatel zde má možnost online vyhledávat mezi svými soubory, sdílet je s ostatními, vidět aktuální využití svého datového prostoru nebo hudební přehrávač s aktuálně přehrávanou skladbou ze svého úložiště. Rozhraní je velice rychlé a nabízí tak prostor pro online práci se svými soubory, které se mohou libovolně přesouvat, přejmenovávat nebo mazat, stejně jako v kterémkoliv souborovém manažeru. Pokud si chce uživatel prohlížet své obrázky nebo fotky, nabízí se mu možnost využít online vytvořeného alba, kde se nemusí starat o jejich přepínání a spustí si je jako prezentaci. Je zde také integrovaná služba pro prohlížení a online úpravu textových dokumentů od společnosti Zoho, a tak má uživatel veškerý komfort pohromadě a může odkudkoliv měnit svůj obsah úložiště.

3.3.5. Klientská aplikace

Klientská aplikace, které díky integraci do systému je dostupná v oznamovací části systému nebo přes pravé tlačítko myši v systémové nabídce, nabízí spoustu různých možností k nastavení. Je zde možnost vybrat si jeden ze čtyř jazyků (Angličtina, Němčina, Španělština a Švédština), lze nastavit limit pro upload dat při synchronizaci, ale nejzajímavější možností je výběr jakékoliv složky ze systému a přidat ji k úložišti (počet složek je omezen na tři). Aplikace uživateli usnadňuje přístup do jednotlivých složek přesměrováním přímo z menu aplikace.

Služba je také integrována do systému, kde nabízí pohodlné přidávání nebo sdílení souborů a složek do úložiště. Negativem však je nepřítomnost jakýkoliv stavových symbolů v synchronizované složce, které by uživatele informovali o aktuálním stavu synchronizace u souborů nebo složek a ten je odkázán pouze na informace v oznamovací oblasti nebo otevření vlastní aplikace, kde je zobrazen průběh nahrávání i s informacemi o aktuálních operacích.

3.3.6. Testování

Z pohledu uživatele se jedná o velice jednoduchou, rychlou a přehlednou službu pro synchronizaci dat. Její hlavní nevýhodou je omezení nahrávaného souboru na 150MB, což jí omezuje k nahrávání pouze malých souborů. Velikost souboru je ověřována jak v klientské aplikaci, tak u webového rozhraní a jediný způsob jak nahrávat větší soubory než 150MB je využít 100GB nebo 500GB účet, který tento limit nemá. Při uploadu většího počtu malých souborů služba nejprve vytvořila adresářovou strukturu a pak se prováděl vlastní upload souborů. Celková doba nahrávání se pohybovala okolo čtyř hodin, oproti tomu největší možný soubor o velikosti 150MB se nahrával pět minut. Přesné hodnoty s přenosovou rychlostí jsou uvedeny v Tab. 5.

Počet souborů	Celková velikost	Čas nahrávání	Přenosová rychlost	Čas stahování	Přenosová rychlost
1	150 MB	5 min	4 Mb/s	25 min	48 Mb/s
100	331 MB	9 min	4.9 Mb/s	48 s	48 Mb/s
15761	87.8 MB	4h 9min	48 kb/s	—	—

Tabulka 5. Přehled naměřených časů a rychlostí přenosu souborů u služby CloudMe

3.3.7. Přehled vlastností a nabízených služeb

- Free účet s kapacitou úložiště 3GB
- Premium účet s kapacitou úložiště 25, 100, nebo 500GB
- Online cloudové úložiště pro synchronizaci dat
- Přístup a synchronizace k datům přes složku v operačním systému
- Synchronizace tří libovolných složek z lokálního disku
- Ve free a 25GB premium účtu omezení velikosti nahrávaného souboru na 150MB
- Obnova smazaných dat
- Přístup k datům přes webové rozhraní
- Multiplatformní podporu operačních systémů Windows, Linux, Mac OS X, Android, iOS
- Podpora pro Google TV, Samsung Smart TV
- Šifrování při přenosu pomocí SSL s SSL Extend Validation Certificate
- Vytvoření síťového disku v systému pomocí WebDAV
- Sdílení dat mimo komunitu
- Integrace programu WinZip 17.5 pro archivaci souborů a mnoho dalších programů tvůrců třetí strany
- Online vytváření fotoalb
- Streamování a přehrávání hudby přes webové prostředí

3.4. ownCloud

OwnCloud je open-source webové úložiště pro synchronizaci, sdílení a zálohu dat, kontaktů nebo kalendáře mezi více zařízeními. Projekt založil v roce 2010 Frank Karlitschek s ambicemi konkurovat Dropboxu a po třech letech se dostal do fáze, kdy jím lze konkurovat velkým komerčním firmám. Pro vývoj byl zvolen široce podporovaný jazyk PHP a Javascript, a to umožňuje spustit úložiště na libovolný aplikační server s operačním systémem, který podporuje PHP. Tento přístup umožňuje firmám nasadit službu na vlastní server a mít data pod vlastní kontrolou. Nevýhodou však je pouze jeden centrální server[6][7].

3.4.1. Použití

Pro používání cloudového úložiště není potřeba žádná registrace, ale uživatel musí mít od administrátora serveru uživatelské jméno a heslo pro přihlášení, které ověřuje LDAP databáze. OwnCloud umožňuje stejné funkce a možnosti jako velké komerční služby. Synchronizuje data mezi více zařízeními přes defaultní složku, kterou lze rozšířit o libovolný adresář ze zařízení, umožňuje přístup k datům přes webové rozhraní, sdílení dat mezi uživateli, verzování souborů nebo jazykovou lokalizaci pro český nebo anglický jazyk. Smazané soubory a verze jsou udržovány vzhledem k volné kapacitě úložiště a

služba využívá maximálně 50% této volné kapacity. Při překročení limitu se trvale odstraní nejstarší vytvořená verze, u smazaných souborů nejstarší soubor vložený do koše[8].

3.4.2. Bezpečnost

Pro zajištění bezpečnosti během přenosu dat a komunikace mezi zařízením a serverem služba nabízí použití SSL certifikátů a následně šifrovaní na serveru. Šifruje se pomocí symetrické šifry a jako klíč se používá uživatelské heslo pro přihlášení. Tyto služby musí být povolené administrátorem a vzhledem k možnosti vlastního nasazení pod vlastní správou, není šifrování nezbytně nutné, ale spíše doporučené. Po povolení jsou všechna uložená data automaticky zašifrována, ale existuje několik případů dat, které se nešifrují[8]:

- Verze souborů vytvořené před povolením šifrování
- Smazané soubory, které byly v koši ještě před povolením šifrování
- Náhledy v galerii
- Vyhledávací indexy z aplikace pro full textové vyhledávání

3.4.3. Sdílení a synchronizace

OwnCloud umožňuje snadné sdílení celých složek nebo jednotlivých souborů mezi uživateli, tak i odeslání odkazu emailem komukoliv na internetu. Sdílená data je možné chránit heslem nebo nastavit datum, kdy sdílení vyprší a zamezit tak přístupu nechtěným osobám k těmto datům. Synchronizace mezi zařízeními se provádí přes klientskou aplikaci, kde se využívá nastavené složky, nebo přes webové rozhraní, kde uživatel může nahrávat jednotlivé soubory ručně.

3.4.4. Webové rozhraní

Vzhledem k tomu, že klientská aplikace slouží pouze pro synchronizaci souborů, ostatní akce jako nastavení sdílení nebo správa verzí se děje přes webové rozhraní. To po přihlášení nabízí přehledného souborového manažera, který intuitivně nabízí všechny potřebné funkce pro práci se soubory. Uživatel zde má přístup k verzím souboru, nastavuje sdílení, které lze zabezpečit heslem nebo může povolit různá rozšíření jako je šifrování, ukládání verzí souborů, čtení PDF dokumentů a mnoho dalších služeb.

3.4.5. Klientská aplikace

Klientská aplikace podporuje nejrozšířenější platformy pro PC (Windows, Linux, Max OS) a slouží převážně pro synchronizaci dat. Po nainstalování se nastaví webová adresa serveru, kde služba běží a po vyplnění přihlašovacího jména a hesla se nastaví složka, která se bude se serverem synchronovat. Služba je následně dostupná přes oznamovací panel systému. Na pozadí se kontrolují změny, které se provedly na serveru, a okamžitě se synchronizují. Při tomto procesu má uživatel možnost vidět aktuální stav synchronizace a k ní se udržuje seznam posledních padesáti synchronizovaných souborů od spuštění aplikace s časem, kdy k synchronizaci došlo. Aplikace také umožňuje nastavit přípony souborů, které se nemají synchronizovat a může se vyfiltrovat pouze určitý typ souborů. K defaultnímu adresáři je možno přidat libovolnou složku ze systému a nemusí se tak kopírovat sdílená data do jedné složky. Aplikace umožňuje pozastavení

synchronizace, nastavení limitů pro stahování a nahrávání, ale není integrována do systému, takže uživatel nemá ve své složce možnost vidět stav synchronizace a je odkázán na informace v aplikaci.

3.4.6. Testování

Služba nabízí klientskou i webovou část v dobrém provedení, ale při detailnějším testování služby docházelo k tomu, že ne všechny funkce fungovaly tak jak měly a jak by uživatel očekával. Např. u klientské aplikace se nepodařilo spárovat složku ze systému se složkou ve službě a vytvořila se pouze složka v úložišti, která se neměla s čím synchronizovat. Druhým závažným nedostatkem je rapidně snížená rychlost webového rozhraní při nahrávání souborů přes klientskou aplikaci. Zajímavou vlastností, kterou jsem využil zejména při testování je přehled padesáti posledních akcí, které se při synchronizaci děly, avšak výpis byl omezen pouze na dobu běhu aplikace a ne na celkový proces synchronizace.

OwnCoud si velice dobře poradí s velkým souborem a větší problémy nemá ani s nahráváním fotek, ale s velkým množstvím malých souborů si dokázala služba poradit až za více jak dva dny. Při stahování přes webové rozhraní služba stahuje soubory ve formátu, jakém jsou na serveru uloženy a složky zabaluje do zip archivu. Problém opět nastává při archivaci složek s větším počtem souborů, které trvalo i několik minut, ale přenosová rychlost se pak pohybovala okolo 150 Mb/s. Přesné hodnoty s přesnou přenosovou rychlosti jsou uvedeny v Tab. 6.

Počet souborů	Celková velikost	Čas nahrávání	Přenosová rychlost	Čas stahování	Přenosová rychlost
1	1.4GB	9 min	21.23 Mb/s	71 s	161.5 Mb/s
100	331 MB	9 min	4.9 Mb/s	56 s	47.2 Mb/s
15761	87.8 MB	59h	3.4 kb/s	24 min	480 kb/s

Tabulka 6. Přehled naměřených časů a rychlostí přenosu souborů u služby ownCloud

3.4.7. Přehled vlastností a nabízených služeb

- Zakladatel Frank Karlitschek v roce 2010
- Open source pro vlastní nasazení
- Kapacita úložiště limitována vlastním zařízením
- Přístup k datům přes složku nebo webový prohlížeč
- Možnost přidat libovolnou složku z PC k synchronizaci
- Filtrování synchronizovaných souborů
- Podporované systémy Windows, Linux, Mac OS, iOS, Android
- Zálohování dat s využitím max. 50% volné kapacity úložiště
- Obnova smazaných souborů využívá max. 50% volné kapacity úložiště
- Sdílení souborů mimo server přes email nebo odkazem
- Šifrování přenosu pomocí SSL certifikátů
- Programovací jazyk PHP a javascript
- LDAP databáze pro autentizaci uživatelů
- Přístup přes webový prohlížeč

3.5. Testování na referenčním stroji

K opětovnému testování výše popsaných služeb a jejich relevantnímu porovnání, byl použit virtuální referenční stroj v počítačové síti ČVUT FEL s následujícími parametry:

- Hardware - Dell PowerEdge R710[9] s 2x Intel Xeon X5667 a diskovým polem ClarionAX4[10]
- Systém Windows 7 Enterprise 64 bitová verze, Service Pack1
- Procesor QEMU Virtual CPU version (cpu64-rhel6) 3,06GHz počet procesorů: 2
- 4GB RAM
- Rychlost připojení k internetu 1Gb/s symetricky

Pro zaznamenání množství přenesených dat se použil program Wireshark a pro zaznamenávání vytížení procesoru byl použit program Perfmon, který je integrován v systému Microsoft Windows 7.

Aby nedocházelo k vzájemnému ovlivňování služeb předchozí apliakcí, byla každá služba instalována na čistý snapshot systému.

Při tomto testování se zaznamenával datový přenos pomocí programu Wireshark a zatížení procesoru se měřil programem Perfmon. Z důvodu zpětného porovnání naměřených hodnot při prvním testování a testováním na referenčním stroji se k testování zvolili stejné soubory. Pouze středně velké soubory byly doplněné o další fotky, aby naměřená rychlost byla směrodatnější. Informace o souborech jsou uvedeny v Tab. 7.

	Počet složekt	Počet souborů	Celková velikost
Big files	1	1	1,4 GB
Middle files	1	152	501 MB
Small files	1077	15761	87,8 MB

Tabulka 7. Přehled testovaných souborů na virtuálním stroji

Odlišností u testování na referenčním virtuálním stroji od testování na osobním počítači, bylo měření zatížení procesoru. Tyto hodnoty jsou spolu s přenosovou rychlostí zaznamenány v následujících tabulkáchTab. 8 Tab. 9 Tab. 10 Tab. 11 .

3.5.1. Dropbox

Druh souborů	Směr přenosu	Rychlost v Mbit/s	Využití procesoru [%]
Big files	UP	0,12	23
Big files	DOWN	53,7	59
Middle files	UP	5,56	18
Middle files	DOWN	26,3	31
Small files	UP	0,259	35
Small files	DOWN	—	—

Tabulka 8. Testování služby Dropbox na referečním stroji

3.5.2. UbuntuOne

Druh souborů	Směr přenosu	Rychlost v Mbit/s	Využití procesoru [%]
Big files	UP	0,32	30
Big files	DOWN	11,8	20
Middle files	UP	9,65	32
Middle files	DOWN	—	—
Small files	UP	0,058	15
Small files	DOWN	—	—

Tabulka 9. Testování služby UbuntuOne na referečním stroji

3.5.3. CloudMe

Druh souborů	Směr přenosu	Rychlost v Mbit/s	Využití procesoru [%]
Big files	UP	—	—
Big files	DOWN	—	—
Middle files	UP	4,46	7
Middle files	DOWN	19,89	35
Small files	UP	0,1	3
Small files	DOWN	—	—

Tabulka 10. Testování služby CloudMe na referečním stroji

3.5.4. ownCloud

Druh souborů	Směr přenosu	Rychlost v Mbit/s	Využití procesoru [%]
Big files	UP	28,9	50
Big files	DOWN	43	62
Middle files	UP	4,9	13
Middle files	DOWN	26,9	42
Small files	UP	0,012	4
Small files	DOWN	2,685	11

Tabulka 11. Testování služby ownCloud na referečním stroji

3.6. Zhodnocení

Mnou testovaná úložiště poskytují uživateli až na výjimky podobné funkce v obdobném provedení. Jedná se o nahrání souborů přes klientskou aplikaci nebo webové rozhraní na server, jejich synchronizaci s dalšími zařízeními a sdílení souborů ať mezi uživateli uvnitř služby anebo publikování dat mimo síť služby. Ve všech případech až na jeden (ownCloud) se jednalo o nahrávání souborů na vzdálený server, který měla pod správou jiná organizace. To služby výrazně omezuje z hlediska důvěry, jakým způsobem se data zpracovávají a kontrolují avšak pro běžného uživatele, který se chce podělit o své soubory nebo fotky s přáteli, jsou naprosto vyhovující.

Mezi všemi testovanými službami se neobjevila žádná, která by se svými naměřenými hodnotami výrazně lišila od prvního testování. Rozdíl zde byl u služeb, které si i přes trvalé smazání souborů uchovaly data ve své paměti a při opětovném nahrávání téhož souboru se pouze obnovily. To mělo za následek přenesení jen malého množství dat na server a nepřenášel se tak celý datový obsah.

Z naměřených dat je patrné, že všechny služby si bez větších potíží poradí s velkými soubory nebo klasickým nahráváním fotek což je pro běžného uživatele nejčastější způsob využití těchto úložišť, ale s velkým počtem malých souborů si žádná služba neporadila výrazně lépe. Oproti tomu služba, která se při nahrávání výrazně odlišuje od ostatních je ownCloud. Úložiště soubory nahrávalo víc jak 24h a pro tento typ souborů je naprosto nevyhovující.

	Google Disk	SkyDrive	DropBox	Ubuntu One
Obecné				
Jazyková lokalizace	●	●	○	●
Integrace do kontextového menu	●	●	○	●
Vlastní nasazení	○	○	○	○
Webová aplikace	●	●	●	●
Klientská aplikace Windows / Linux / Mac OS / Ostatní	●/○/●/○	●/○ /●/○	●/●/●/○	●/●/●/○
Mobilní aplikace Android / Windows Phone / iOS / Ostatní	●/○/●/○	●/●/●/○	●/●/●/●	●/○/●/○
Cloudové úložiště	●	●	●	●
Bezplatná kapacita [GB]	15	7	2	5
Maximální kapacita [GB]	16384	207	100	○
Maximální velikost souboru [MB]	10240	2048	300*4	○
Možnost rozšíření zdarma	○	○	○	○
Synchronizace				
Synchronizace libovolné složky	○	○	○	●
Synchronizace podsložek	●	●	●	○
Synchronizace jednotlivých souborů	○	○	○	○
Pozastavení synchronizace	●	○	●	●
Sledování přenosu dat	●*5	○	○	○
Sdílení				
Veřejné	●	●	○	●
Odkazem	●	●	●	●
Pomocí pozvánky	●	●	●	●
Pomocí mailu	●	●	●	●
Skupinové	●	●	○	○
Nastavení oprávnění	●	●	○	●
Zabezpečení				
Šifrování na straně serveru	○	○	AES	○
Šifrovaní na straně klienta	○	○	○	○
Šifrování přenosu	SSL	SSL	SSL	SSL
Přihlášení pomocí dvoufázového ověření	●	○	●	○
Další funkce				
Obnova smazaných souborů	●	●	●	60 dnů*11
Verzování	●	●*6	●	○
Uložení verzí	30 dnů*7	25 verzí	30 dnů	○
Nastavení rychlosti uploadu	○	○	●	●
Nastavení priorit pro upload	○	○	○	○
Nastavení rychlosti downloadu	○	○	●	●
Rychlosti				
Rychlost uploadu malých souborů [Mb/s]	0,05	0,23	0,25	0.058
Doba uploadu malých souborů	3h 57m	4h12m	15m	2h16m
Rychlost uploadu středně velkých souborů [Mb/s]	4,77	2,83	5,56	9,65
Doba uploadu středně velkých souborů	14m	21m31s	5m	2m30s
Rychlost uploadu velkého souboru [Mb/s]	21,2	4	0,127	0,32
Doba uploadu velkého souboru	10m	58m	90s	3m
Rychlost downloadu malých souborů [Mb/s]	0,08	0,621	-	-
Doba downloadu malých souborů	2h 18m	1h15m	-	-
Rychlost downloadu středně velkých souborů [Mb/s]	26,19	5,315	26,31	-
Doba downloadu středně velkých souborů	3m	51m	2m30s	-
Rychlost downloadu velkého souboru [Mb/s]	58,92	2,8	53,77	11,8
Doba downloadu velkého souboru	4m	1h15m	3m	18m
	Google Disk	SkyDrive	DropBox	Ubuntu One

1 - FreeBSD	5 - Pouze počet přenesených složek a souborů
2 - BlackBerry	6 - Pouze Office soubory
3 - Společný úložný prostor pro služby Google Disk, Gmail a Fotky Google+	7 - Verze souborů z aplikací Google Dokumenty, Tabulky a Prezentace jsou ukládány navždy
4 - Pouze přes webové rozhraní	8 - Ve výchozím nastavení, lze změnit

Obrázek 1. Přehledová tabulka všech služeb

SugarSync	MEGA	TeamDrive	Wuala	cloudMe	SpiderOak	BitTorrentSync	ownCloud
○	●	○	●	○	○	○	●
●	○	○	○	●	●	○	○
○	○	●	○	○	○	●	●
●	●	○	○	●	●	○	●
●/○/●/○	●/(○/○/○)*10	●/●/●/○	●/●/●/○	●/●/●/○	●/●/●/○	●/●/●/●*1	●/●/●/○
●/●/●/●*2	●/○*11/●/●*2	●/○/●/○	●/○/●/○	●/○/●/○	●/○/●/○	●/○/●/○	●/○/●/○
●	●	●	●	●	●	○	○
5	50	2	5	3	2	○	○
1000	4096	neomezeno	2048	500	100	○	○
-	neomezeno	-	40960	150	-	○	○*12
●	○	●	●	○	●	○	○
●	●	●	●	●	●	●	●
●	●	○	●	○	●	○	○
●	●	○	●	○	●	○	○
○	●	●	●	●	●	●	○
●	●	●	●	●	●	●	●
●	●	○	●	●	●	○	○
●	●	●	●	●	●	○	●
●	○	●	○	●	○	○	○
●	○	●	○	●	○	○	●
●	●	●	●	●	●	●	○
○	●	●	●	○	●	●	○
AES-128	○	○	AES-128	○	AES-256	○	●
○	RSA-2048, AES-128	RSA-2048, AES-256	RSA-2048, AES-128	○	RSA-3076, AES-256	AES-128	○
SSL (3.3) + TLS	SSL	○	SSL	SSL	SSL	○	○
○	○	○	○	○	○	○	○
	●	●		●		30 dnů*13	●
●	○	●	●	○	●	●	●
30 dnů	○	neomezeno	10 verzí	-	-	30 dnů*8	-
●*9	●	○	●	●	●	●	●
●	○	○	●	○	○	○	●
○	○	○	●	●	○	●	●
0,04	-	1,17	0.7	0,1	0,5	2,34	0,012
1h50m	-	10m	30m	3h55m	12m	5m	1d7h20m
7,15	1,32	8,6	2,2	4,46	1,358	22,9	4,93
7m15s	51m	8m	13m13s	17m	1h4m	3m	13m
-	63,28	9,09	28,5	-	1,483	33,5	28,92
-	3m	22m	6m	-	2h40m	6m	7m
0,33	-	1,31	2	-	0,4	2,3	2,68
5h13m	-	9m	7m	-	15m	5m	1m30s
2,67	6,34	20,35	10,8	19,89	2,28	22,64	26,94
30m41s	11m	4m	6m58s	2m30s	33m44s	3m	3m
2,9	35,6	10,98	7,3	-	2,45	28,61	43
1h20m	6m	18m	30m41s	-	2h18m	7m	5m
SugarSync	MEGA	TeamDrive	Wuala	cloudMe	SpiderOak	BitTorrentSync	ownCloud

9 - Pouze relativní nastavení
10 - Zatím ve vývoji (17.11.2013)
11 - bez 100% garance
12 - přes webové rozhraní 8MB jinak podle konfigurace
13 - Dobu lze nastavit

4. Analýza a návrh řešení

4.1. Struktura systému

Celý systém je rozdělen do tří základních prvků. Jedná se o tyto části:

- Klientská aplikace
- Access server
- Datové centrum

4.1.1. Klientská aplikace

Klientská aplikace je tvořena ze třech základních částí. První část se stará o komunikaci s hostitelským souborovým systémem. Druhá obstarává komunikaci s access serverem a poslední, třetí část, je zodpovědná za práci s metadaty.

Práce s metadaty je realizována pomocí samostatné knihovny. Tato knihovna udržuje informace o adresářové struktuře a vlastních souborech v ní obsažených.

4.1.2. Access server

Access server odstíňuje klientskou aplikaci od datového centra a zprostředkovává komunikaci mezi nimi. Tvoří tedy přístupový bod pro klientské aplikace, které se do systému připojují a zároveň poskytuje funkce potřebné pro sdílení a historii uživatelských akcí.

4.1.3. Datové centrum

Datové centrum tvoří vlastní úložiště dat. Skládá se z jednotlivých datových serverů, která jsou spojena do strukturované sítě pomocí DHT. Každé jednotlivé centrum je schopno obsluhovat požadavky od AS.

4.1.4. Komunikace

Komunikace mezi jednotlivými prvky systému je tvořena pomocí socketů. Pro výměnu informací mezi jednotlivými prvky systému se využívá informačních zpráv, které jsou specifické pro jednotlivé subsystémy a prováděnou operaci.

4.1.5. Indentifikace

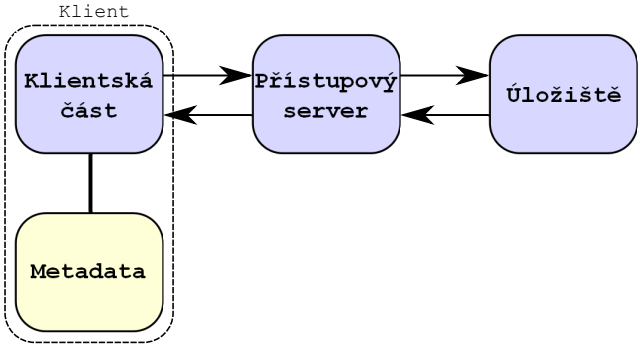
V rámci celého systému neexistuje žádná databáze uživatelských identifikátorů. Pro identifikaci uživatelů se používá jednoznačný identifikátor, který je generován klientskou aplikací při prvním přihlášení do systému. Tento identifikátor je vytvořen pomocí hashovaní funkce SHA-2 s celkovou délkou 80 bytů. Pro určení lokalizace souborů v rámci DHT datového centra se používá prvních 40 bytů identifikátoru daného souboru.

4.1.6. Zabezpečení systému

Jednotlivé soubory jsou rozděleny do chunků o maximální velikosti 4MB, které jsou následně zabezpečeny s využitím šifrovacího algoritmu AES-128. U každého souboru lze nastavit počet replikovaných souborů v rámci datového centra.

Komunikace je mezi jednotlivými prvky systému zabezpečena na úrovni protokolu TLS s PGP mechanizmem.

Celková koncepce systému je znázorněna na Obr. 2



Obrázek 2. Celková koncepce systému

4.2. Obecné požadavky

Jedná se o požadavky, které nemají vliv na funkčnost programu, ale jsou součástí zadání z důvodu kompatibility nebo bezpečnosti aplikace.

4.2.1. Multiplatformní systém

Pro vyvíjený systém je nutné zajistit, aby mohl běžet na různých operačních systémech. Pro tento účel je vhodné zvolit některou multiplatformní knihovnu. Knihovna POCO, která podporuje několik operačních systémů spolu s programovacím jazykem C++ bude vhodným nástrojem pro vývoj přístupového serveru. Ostatní programovací jazyky nenabízí oproti jazyku C++ výhodu v rychlosti, ani v objektovém přístupu. Pro tvorbu klientské aplikace, která bude přistupovat k funkcím operačního systému, bude vhodné využít jazyk C.

4.2.2. Volba operačního systému

Pro vývoj celého systému jsem zvolil GNU/Linux z důvodu širokého uplatnění při nasazování serverových řešení. Celý systém je navrhován tak, aby jej bylo možné kompilovat na více systémech.

4.3. Funkční požadavky z pohledu klienta

Jak vyplývá ze závěru testovaných úložišť, všechny systémy nabízí uživateli základní operace se soubory a adresáři, které najde v operačním systému. Jedná se o tyto následující operace, které v našem systému zohledníme:

- Vytvoření nového souboru
- Vytvoření nové složky
- Editace obsahu souboru
- Přejmenování souboru
- Přejmenování složky
- Smazání souboru
- Smazání složky
- Sdílení souborů
- Offline práce se soubory a následná aktualizace stavu

4.4. Nefunkční požadavky z pohledu klienta

Mezi nefunkční požadavky patří:

- Zabezpečení přenosu proti třetí osobě
- Šifrování souborů
- Replikování dat pro zajištění dostupnosti v případě výpadku

4.5. Požadavky z pohledu provozovatele

4.5.1. Minimalizace přenášených dat

V tomto případě se jedná o požadavek, který se uplatní při komunikaci mezi serverem a klientem anebo mezi samotnými přístupovými servery. Existuje mnoho formátů jako jsou např. JSON nebo XML, ale pro využití v komunikaci jsou naprosto nevhodné pro malou efektivitu, proto si v rámci svého subsystému vytvořím vlastní formát, který bude vyhovovat daným potřebám.

4.5.2. Rozchunkování souborů

Přenášet velké soubory mezi systémem a uživatelem by bylo velice nepohodlné z důvodu možného odpojení klientské aplikace před dokončením přenosu a opětovného startu přenosu od začátku při dalším spuštění. Rozchunkování bude mít i svoje uplatnění v rámci bezpečnosti a rozložení souborů na více datových serverů.

4.5.3. Rozlišení jednotlivých uživatelů

V rámci sítě access serverů bude nutné rozlišovat jednotlivé uživatele pomocí jednoznačných identifikátorů.

4.6. Klientská aplikace

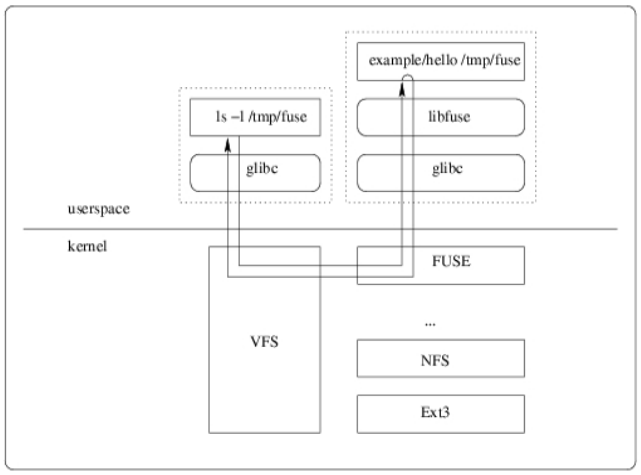
Klientská aplikace bude integrována do systému a jejím hlavním úkolem bude kromě interakce s uživatelem a následné promítnutí změn do správy metadat také komunikace s přístupovým serverem. Aplikace by měla být navržena s ohledem na možné cestování

uživatele sítí se zachováním všech jeho uložených dat. Přístup k aplikaci by měl být zabezpečený, aby nebylo možné získat data třetí osobou. Aplikace by měla poskytovat uživateli následující operace:

- Vytvoření souboru
- Vytvoření složky
- Přejmenování souboru
- Přejmenování složky
- Smazání souboru
- Smazání složky
- Čtení souboru
- Změna obsahu souboru
- Kopírování souborů
- Sdílení souborů s jinými uživateli

Pro vývoj klientské aplikace existuje několik způsobů realizace. Jedním z nich je implementace ovladače souborového systému v jádře systému. Tento způsob je velice náročný a musí se při něm dodržovat všechny zásady, které pro jádro platí. Navíc je třeba sledovat aktualizace jádra a modul upravovat. Nevýhodou takového systému je správa pouze od superuživatele (root)[11].

Druhá varianta vhodná pro realizaci naší aplikace se nazývá FUSE (Filesystem in Userspace). Jedná se o řešení, kdy se FUSE z pohledu jádra chová jako souborový systém, ale jeho funkce je odlišná. Nestará se o vyřízení požadavků a sestavení odpovědi od disku, ale spustí se jako běžný uživatelský program. Pomocí knihovny se komunikuje s jádrem systému přes modul FUSE, který je implementován v jádře Linuxu od verze 2.6.14.[12] Funkce tohoto systému je znázorněna na Obr. 3.



Obrázek 3. FUSE modul jádra a komunikace s FUSE knihovnou

Výhody tohoto řešení jsou:

- Snadná implementace
- FUSE nabízí podporu několika operačních systémů jako FreeBSD, OpenSolaris nebo Mac OS X
- API FUSE se nemění, a není tak třeba hlídat změny jádra OS

Nevýhody tohoto řešení jsou:

- Operace prováděné přes FUSE nejsou tak rychlé jako při implementaci přímo do jádra.

Z důvodu snadnější implementace a podpory několika operačních systémů jsem se rozhodl využít k řešení klientské aplikace FUSE. Její nevýhodou je nižší rychlost v porovnání s implementací do jádra, avšak tato nevýhoda pro nás není nijak zásadní, protože rychlost samotné implementace souborového systému nebude to, co bude náš systém omezovat.

4.6.1. Filesystem in Userspace

Pro implementaci filesystému existují dvě varianty jejich realizace. Pokud by se jednalo o velmi malé množství dat, které by aplikace udržovala, lze si je udržovat v paměti RAM. Výhodou takového systému by byla rychlost prováděných operací. Značnou nevýhodou je omezená velikost paměti, která by se mohla využit. Tato varianta je pro náš systém naprosto nevhodná, ale je vhodné zmínit, že takováto možnost existuje. Druhá varianta pracuje nad dvěma složkami, kde jedna slouží jako lokální cachce aplikace a druhá se zobrazuje jako připojený disk, nad kterým uživatel provádí příslušné operace.

Knihovna FUSE nabízí pro implementaci vlastního operačního systému 25 základních funkcí. Tento počet zahrnuje povinné o volitelné metody. Nyní nastíním několik funkcí se základním chováním, které budou potřebné pro klientskou aplikaci.

- `void* init(struct fuse_conn_info *conn);`

Tato metoda se volá jako úplně první při startu a inicializuje filesystém. Jedná se o volitelnou metodu.

- `void destroy(void* private_data);`

Tato metoda se volá po ukončení filesystemu. Opět se jedná o volitelnou metodu.

- `int getattr(const char* path, struct stat* stbuf);`

Tato metoda předává atriuty o souboru nebo složce, která je identifikovaná cestou (atribut path) do struktury stat. V případě shodné cesty se vrací 0, jinak se vrací příslušný kód chybové hlášky. Jedná se o povinnou metodu, která musí být implementovaná v každém systému.

- `int mknod(const char* path, mode_t mode, dev_t rdev);`

Tato metoda vytvoří prázdný soubor podle zadané cesty.

- `int mkdir(const char* path, mode\_t mode);`

Tato metoda vytvoří složku podle zadané cesty.

- `int rmdir(const char* path);`

Tato metoda smaže složku podle zadané cesty.

- `int unlink(const char* path);`

Tato metoda smaže soubor podle zadané cesty.

- `int rename(const char* from, const char* to);`

Tato metoda přejmenuje soubor nebo složku podle zadaných cest.

- `int open(const char* path, struct fuse_file_info* fi);`

Tato metoda slouží k otevření souboru na zadané cestě.

- `int readdir(const char* path, void* buf, fuse_fill_dir_t filler, off_t offset, struct fuse_file_info* fi);`

Tato metoda slouží k naplnění struktury dir_t pomocí funkce filler, která je v parametru funkce, podle zadané cesty.

- `int read(const char* path, char *buf, size_t size, off_t offset, struct fuse_file_info* fi);`

Tato metoda se stará o čtení ze souboru identifikovaného cestou. Při čtení je nutné naplňovat vnitřní buffer (parametr buf), který má maximální velikost 128kB. Parametry pro čtení jsou offset, který udává začátek čteného bloku v souboru a hodnota size, která udává množství přečtených dat. Funkce očekává na výstupu tolik přečtených dat, kolik si vyžádala.

- `int write(const char* path, char *buf, size_t size, off_t offset, struct fuse_file_info* fi);`

Tato metoda se stará o zápis do souboru identifikovaného cestou. Proces zapisování je shodný s procesem čtení s rozdílem, že funkce na svém výstupu očekává 0, pokud vše skončilo úspěšně nebo v opačném případě chybovou hlášku.

4.6.2. Struktura klientské aplikace

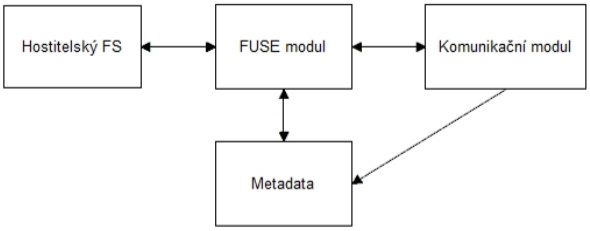
Pro správné a plynulé fungování klientské aplikace bude nutné od sebe oddělit část komunikující s access serverem a část, která se bude starat o vlastní práci se soubory. K této komunikaci bude vhodné zvolit frontu, přes kterou spolu budou jednotlivé části komunikovat. O správu metadat v aplikaci se postará knihovna, která je výstupem diplomové práce Tomáše Dyntara[13]. Struktura klientské aplikace je znázorněna na Obr. 4

4.6.3. Funkce aplikace

Start aplikace

Při startu aplikace bude důležité, aby se zajistila aktuálnost dat. Může se stát, že uživatel bude nad svými daty pracovat z několika míst a pokaždé by měl mít přístup ke stejným datům. To se nejlépe zajistí stažením user chunku z úložiště ještě před zpřístupněním aplikačního rozhraní.

Zároveň je nutné zajistit, aby uživatel měl výhradní přístup ke svým datům. Jedna z možností, jak tento problém vyřešit, je využití přihlašovacího jména a hesla a ověřování se proti user chunku. Uživatel by si k tomu musel buď šifrovací klíč, který bude potřeba pro zašifrování souborů zvolit nebo by se mu náhodně vygeneroval. Pořád ale nemáme jednoznačný identifikátor pro daný user chunk.



Obrázek 4. Struktura klientské aplikace

Jako lepší řešení se nabízí použití privátního klíče. Pokud vygenerujeme z privátního klíče hash, máme jednoznačný identifikátor, který se během využívání aplikace nezmění a můžeme se po přenosu klíče připojit z kteréhokoliv místa v síti. Pokud by se klíč pro šifrování a dešifrování souborů při prvním startu aplikace náhodně vygeneroval, musel by se někam uložit, aby k němu měl uživatel přístup při přemístění aplikace např. na nový počítač nebo by si ho uživatel musel přenášet s sebou. Nabízí se proto využít k tvorbě klíče pro šifrování stávající privátní klíč uživatele.

Nahrání souboru

Uživatel systémovými příkazy v klientské aplikaci vytvoří nebo zkopíruje soubor a ten se začne odesílat na access server. Dokud aplikace neobdrží potvrzení o úspěšném nahrání souboru na datový server, soubor se pro uživatele jeví jako neexistující, ale systém ho registruje a nelze vytvořit nový soubor se stejným jménem.

Stažení souboru

Ke stahování souborů z úložiště dojde v případě, kdy se uživatel rozhodne přenést svůj uživatelský účet na jiný stroj nebo jiné umístění v rámci svého systému než tomu bylo doposud. V takovém případě aplikace zajistí přenos všech datových chunků do lokální cache. Aby se minimalizoval počet přenášených dat a došlo k co nejrychlejšímu stažení požadovaných souborů, budou se stahovat pouze ty soubory, ke kterým uživatel otevře složku.

Druhým případem, kdy dochází ke stahování souborů do klientské aplikace je souběžná aktivita dvou uživatelů nad jedním účtem. V momentě, kdy uživatel A1 vytvoří nový soubor, access server se postará o aktualizaci změn a informuje klientskou aplikaci uživatele A2 o nutnosti stáhnutí souboru.

Třetí možností, kdy bude nutné stahovat soubory z datového úložiště, je zásah do chunků dat a jejich smazání z lokálního disku jinak než přes klientskou aplikaci.

Smazání souboru

Mazání souborů probíhá pouze v lokální cache uživatele. Při této operaci se odstraní záznam z metadat a odstraní se datové chunky, aby u uživatele zbytečně nezabíraly místo na disku. O této změně je nutné informovat AS, který informuje další připojené klienty pod totožným účtem.

"Check"souborů

Jedná se o funkcionalitu, která datovému serveru odešle seznam všech chunků v lokálním úložišti. Tato funkcionalita byla navržena místo klasického mazání souborů, kdy by se po každé změně musela provádět změna i na datovém úložišti. Výhodou je, že datové centrum není zatěžované častými akcemi od uživatele a v případě opětovného nahrání souboru nebude nutné posílat soubor znovu až na datové centrum.

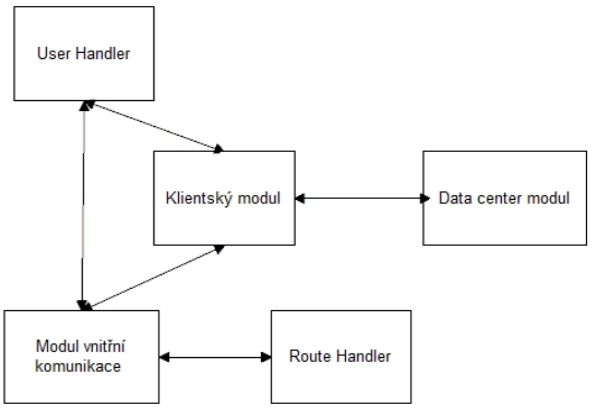
Sdílení souborů

Aplikace by uživateli měla poskytovat možnost sdílet soubory a složky s ostatními uživateli systému v několika úrovních.

4.7. Access server

Přístupový server bude mít za úkol zprostředkovávat komunikaci klientské aplikace s vlastním datovým centrem a zároveň bude sloužit jako vzájemné odstínění těchto dvou bodů. Nevýhodou tohoto řešení je, že se přístupový server stává úzkým hrdlem systému a při větší zátěži nebo výpadku serveru by se stal systém nedostupný. Tento problém lze řešit vybudováním distribuované sítě serverů a rozdělit zátěž mezi více strojů. Systém pak bude fungovat i při výpadku jednoho nebo více serverů. Musí se však zajistit komunikace přístupových serverů mezi sebou, aby bylo možné zajišťovat aktuálnost dat na všech klientských aplikacích. Struktura jednoho access serveru je znázorněna na Obr. 5

Využití tohoto řešení vychází z předpokladu, že velké množství přerušené komunikace přichází od klienta z důvodu častého odpojování aplikace nebo nestability připojení a není nutné touto komunikací zatěžovat datový server. Druhým důvodem je předpoklad, že datových serverů bude více než těch přístupových, a bylo by nutné zajistit jejich viditelnost v rámci sítě.



Obrázek 5. Struktura access serveru

4.7.1. Klientský modul

Tento modul bude obstarávat komunikaci s klientskou aplikací. Jeho hlavním úkolem bude přijímat požadavky od klientské aplikace a přeposílat je na datové centrum. Neméně důležitou funkcí bude předání požadavků modulu pro vnitřní komunikaci, který se postará o rozeslání zpráv na další AS.

4.7.2. Data center modul

Tento modul bude zajišťovat komunikaci s datovým serverem, na který bude odesílat požadavky v závislosti na informacích obdržených v klientským modulu. Tento modul bude rozdělen na klientskou a serverovou část, kdy klientská část bude zasílat požadavky na datové centrum a serverová část bude přijímat a odesílat příslušné soubory na server, který s ním naváže spojení. Mezi AS a datovým centrem se bude komunikovat pomocí zpráv popsaných v práci Martina Kudrnáče[14].

4.7.3. User Handler

Tento modul se bude starat o přístup k uživatelským informacím. Budou se udržovat následující informace:

- AS_UID - identifikátor uživatele v rámci sítě mezi AS
- IP adresa serveru
- Port serveru
- Další výskyty klienta

4.7.4. Route Handler

Tento modul se bude starat o přístup ke směrovacím informacím, které se využívají při rozesílání zpráv mezi access servery. V záznamu se udržují tyto informace:

- IP adresa serveru
- Port serveru
- Identifikator spojení (může obsluhovat serverová nebo klientská část spoje)
- Session

4.7.5. Modul vnitřní komunikace

Tento modul bude obsluhovat komunikaci mezi jednotlivými přístupovými servery. Protože bude docházet k nepravidelné komunikaci a předávání zpráv, musí se od sebe oddělit proces komunikace s předáváním zpráv k odeslání.

4.7.6. Funkce access serveru

Nyní nastíním složitější funkce access serveru.

Nahrání chunku

Access server dostane od klienta požadavek, že se chystá nahrávat chunk dat a čeká na další příjem dat.Po úspěšném přenosu celého chunku se naváže spojení k datovému centru. Z datového úložiště (nemusí být, a často to ani není uzel, na který jsme posílali požadavek) přijde odpověď, o možném zahájení přenosu. Pokud uzel potvrdí, že transfer proběhl v pořádku, předá se potvrzení o nahraném chunku klientské aplikaci a server si soubor ze své lokální cache smaže. Výhodou tohoto řešení je, že pokud dojde k přerušení přenosu během odesílání od klienta, datový server tím nebude zatěžovaný. Nevýhodou je čekání na nahrání celého souboru.

Stažení chunku

Stahování chunku probíhá velice podobně jako jeho nahrávání. Klientská aplikace vyšle na přístupový server požadavek, že chce získat chunk dat. Server naváže spojení k datovému úložišti a stejně jako v případě nahrávání, odpoví uzel, kde se příslušný chunk nachází a začnou se odesílat data. Po obdržení celého chunku AS potvrdí úspěšný přenos a klientské aplikaci odešle informaci spolu s parametry souboru, že s ní může zahájit přenos a začnou se odesílat data, která klient po úspěšném přijetí potvrdí.

Oznámení o novém uživateli

Tato funkcionalita má za úkol detekovat současné připojení uživatelů pod stejným účtem. Všem ostatním přístupovým serverům se v momentě připojení nového uživatele odešle zpráva s identifikátorem uživatele a každý server na tuto zprávu musí odpovědět buď negativně anebo pozitivně v případě, že je k němu klient připojen. Oba servery si zapamatují informace, kde se další klient nachází k dalšímu využití.

Oznámení o změně metadat

Jedná se o situace, kdy se v klientské aplikaci změní metadat a pod stejným uživatelským účtem pracuje více klientů. Je nutné tyto změny distribuovat informace nejen na datové úložiště, ale i k příslušným access serverům, které je následně odešlou klientům, kde se změna neprováděla. Tím se zajistí aktuálnost dat při běhu aplikace. Odesílání informací by mělo začínat až po potvrzení a úspěšném nahrání na datové centrum.

4.8. Komunikační model

Komunikace mezi jednotlivými prvky sítě je neoddělitelnou součástí tohoto systému. Je důležité, aby veškerá komunikace byla zašifrovaná a nebylo ji možné odposlechnout. K tomuto účelu jsme zvolili jednotný formát pro celý systém ve formě zabezpečených socketů pomocí TLS s ověřovacím mechanismem PGP.

Přístupový server tvoří bránu do celého systému a tomu odpovídá i jeho komunikace. Server odpovídá nejen na požadavky klienta, ale musí komunikovat i s datovými servery a v neposlední řadě i s okolními přístupovými servery.

4.8.1. Komunikace klienta s access serverem

Mezi klientem a přístupovým serverem se předpokládá častá komunikace, a proto by zvolený komunikační model měl být dostatečně jednoduchý a efektivní. Jedním z návrhů bylo využití některého stávajícího komunikačního protokolu, jako je například HTTPS. Jedná se o nejrozšířenější protokol s podporou šifrování a serverových operací (např. GET, POST, PUT), které by pro náš systém byly plně dostačující. Jeho výraznou nevýhodou je složitá realizace asynchronní komunikace, kterou budeme využívat. Z toho důvodu jsme tento návrh zavrhli a zvolili jsme otevřené TCP spojení přes šifrované sockety. Sockety nenabízí žádné předpřipravené operace jako například HTTPS, ale umožňují realizovat asynchronní přenos, který je pro nás důležitější a realizace informačních zpráv nebude nijak náročná. Mezi klientem a access serverem se budou posílat následující zprávy:

- Vyžádání uživatelského chunku
- Odeslání uživatelského chunku
- Oznámení o odesílání datového chunku
- Vyžádání chunku
- Oznámení o změně metadat

4.8.2. Formát zpráv při komunikaci klienta s access serverem

Odesílání chunku od klienta

Odeslání datového chunku a usre chunku se ve struktuře nijak lišit nebude, rozdílný bude pouze kód, s kterým se zpráva odešle. Po odeslání chunku s touto zprávou začne samotný přenos souboru.

- Identifikátor
- Kód operace
- Velikost chunku
- Počet replikací
- Hash souboru

Odpověd od serveru

- Kód operace
- Stavový kód

Vyžádání souboru

- Kód operace
- Hash souboru

Odpověd od serveru v případě nalezení chunku dat

- Kód operace
- Hash souboru
- Velikost souboru

Příjem souboru se potvrzuje odesálním zprávy "OK"

Odpověd od serveru v případě nenalezení chunku dat

- Kód operace
- Hash souboru
- Návratový kód

4.8.3. Komunikace access serveru s datovým centrem

Při komunikaci s datovým centrem se zohledňuje princip a mechanismy používané pro ukládání dat. Aby nedocházelo k zatěžování jednoho datového serveru, který by tak tvořil bránu k datovým serverům, neustálým přenosem souborů, které pro něho nejsou určené, přístupový server nejdříve odešle požadavek s kódem operace, kterou chce provádět na libovolný uzel v DHT síti a datové úložiště odpoví již ze serveru, kde jsou data uložena nebo kam se mají posílat. Bude nutné rozlišovat následující zprávy:

- Odesílání user chunku
- Odesílání datováho chunku
- Příjem chunku
- Update souborů

4.8.4. Komunikace mezi access servery

Při komunikaci mezi přístupovými servery se vychází z předpokladu, že serverů nebude velké množství a bude možné vytvořit takovou síť, kde bude možné komunikovat s každým přímo bez využití jakéhokoliv prostředníka. Tím se docílí nejrychlejšího možného přenosu zpráv z jednoho uzlu na druhý a nebude docházet ke zbytečnému zatěžování okolních serverů zprávami, které pro ně nejsou určené. Mezi přístupovými servery bude třeba rozesílají následující zprávy:

- Informace o dalších aktivních uzlech v síti při pokusu o připojení
- Přihlášení nového uživatele
- Odpojení uživatele
- Změna metadat v případě připojení více totožných uživatelů
- Přenos souboru v případě připojení více totožných uživatelů

5. Realizace

V analýze byly popsány všechny důležité prvky mého subsystému a nyní popíši jejich vlastní realizaci. V rámci mé diplomové práce došlo pouze k pilotní implementaci a některé funkce nebyly z různých důvodů realizovány.

5.1. Vývojové prostředí

Jak již bylo zmíněno v analýze, systém by měl mít multiplatformní podporu a být dostatečně rychlý. V dnešní době existuje velké množství programovacích jazyků, ale většina z nich není pro náš systém vhodná nebo nenabízí všechny potřebné funkce.

S přihlédnutím k těmto problémům jsem pro vývoj přístupového serveru zvolil jazyk C++ s využitím knihovny POCO. Jedná se o OpenSource knihovnu s podporou mnoha operačních systémů a nepřeberného množství funkcionalit týkajících se síťových řešení, vláken nebo samotného vývoje aplikací.

Nevýhodou této knihovny je absence PGP ověřovacího mechanismu pro komunikaci. Z tohoto důvodu je pro síťové řešení nutné nasadit GnuTLS knihovnu. Jedná se také o OpenSource projekt s podporou SSL, TLS a DTLS protokolů a technologií okolo nich.

Pro vývoj klientské části mého subsystému jsem zvolil FUSE. Tato knihovna je určena pouze pro čisté C a bylo nutné se rozhodnout, zda využívat jazyk C++ i zde a následně vytvořit API pro FUSE nebo využít podporovaný jazyk. Rozhodl jsem se využít čisté C z důvodu jednodušší implementace. Jazyk C++ by v tomto případě nenabízel žádné výraznější výhody.

Klientská aplikace bude mít za úkol šifrovat nahrávané soubory. Jazyk C neposkytuje sám o sobě žádné funkce pro šifrování a musí být doplněn o nějakou šifrovací knihovnu. Byla zvolena knihovna OpenSSL, která je primárně určena pro unixové systémy, ale lze jí použít i na systémech Windows a dalších. Jedná se rovněž o OpenSource knihovnu, která patří mezi nejrozšířenější šifrovací nástroje vůbec.

5.2. Funkce klientské aplikace

Hlavním úkolem klientské aplikace je komunikace s uživatelem, kterému musí umožnit základní operace se soubory jako je např. čtení, zápis nebo mazání. Aby nebylo možné data zneužít třetí osobou, musí být chráněna. Nejjednodušší a nejlepší způsob je data před uložením na disk zašifrovat a odesílat na server již zašifrované soubory. Třetí osoba pak nemá žádnou možnost zjistit obsah souborů při odposlechu nebo z lokální cache.

Celá aplikace se v systému tváří jako připojený disk reprezentovaný složkou v systému, ve kterém se provádějí běžné uživatelské operace. FUSE tyto operace monitoruje a mým úkolem bylo vhodně implementovat jejich funkčnost.

5.2.1. Fronta požadavků

Fronta požadavků v klientské aplikaci slouží pro uložení dotazů, které se mají odeslat na přístupový server a následně na datové centrum. Do fronty se ukládají záznamy, které nesou následující informace:

- **Cesta k souboru** - Cesta v rámci našeho FS, do které se zapisovalo, používá se jak indetifikátor
- **Jméno zapisovaného souboru** - Při změně obsahu souboru se liší s cestou k souboru
- **Velikost souboru** - Celková velikost souboru, v metodě fs_getattr se předává do FUSE
- **Velikost zapisovaného bloku** - Identifikátor ukončení spojení pokud je blok menší než 4096B
- **Typ požadavku** - Rozlišuje vyžadované a odesílaná chunky
- **Velikost pole chunků** - Počet datových chunků
- **Pole chunků** - Používá se pro přenos datových chunků a následné předání do správy metadat
- **Pole hashí** - Používá se pro přenos metadat chunků k odeslání na AS

Implementované FUSE metody vkládají do fronty jednotlivé požadavky na stáhnutí nebo odesální chunku, které jsou následně zpracovány obsluhou fronty. Fronta se nevyprazdňuje po odeslaném chunku, ale musí se odeslat celé pole chunků nebo hashí, a až pak je požadavek odstraněn z fronty. Je to z důvodu, aby se jednotlivé soubory zpracovávaly postupně a nedocházelo k promíchávání chunků od různých souborů. Druhým důvodem je odlišná struktura datových chunků a chunků metadat, kvůli které by se musela vytvořit nová struktura nesoucí informace o obouch strukturách současně. Konkrétní obsah jednotlivých struktur je popsán v práci Tomáše Dyntara[13].

5.2.2. Obsluha fronty požadavků

Frontu požadavků obsluhuje samostatné vlákno, které se stará i o odesílání dat na AS od kterého očekává i odpověd. Pokud je fronta prázdná, vlákno je uspané a čeká na probuzení příchodem nového požadavku. Princip zpracování jednotlivých požadavků se řídí podle velikosti pole chunků v uložené struktuře. Pokud není toto pole prázdné, odesílají se datové chunky jeden po druhém. V opačném případě se jedná o metadata nebo požadavky o stáhnutí souboru z datového centra.

Vlákno má také za úkol předávat po odeslání všech chunků dat příslušné informace do knihovny pro správu metadat, která vrátí nově vygenerovaná metadata pro odeslání. Tyto metadata se nezpracovávají hned, ale vloží se na konec fronty.

5.2.3. FUSE modul

Pro implementaci vlastního souborového systému pomocí FUSE není nutné využívat všech nabízených metod, ale lze si vybrat jen ty, které budou pro náš systém potřeba. V analýze jsem obecně popsal funkce, které v našem systému využijeme a nyní se zaměřím na jejich detailnější popis v rámci klientské aplikace.

fs_getattr

Jedná se o nejdůležitější metodu systému. Jejím hlavním úkolem je předávat informace o souborech a složkách v systému do knihovny FUSE. Protože se souborovým systémem pracujeme nepřímo a čekáme na potvrzení odeslaných dat, musí se při zjišťování, zda soubor existuje nebo jaké má vlastnosti, prohledávat i fronta požadavků, která funguje i jako část filesystému, který je udržovaný v operační paměti.

Neméně důležitou funkcí této metody je vytváření posledního chunku dat při zapisování do souboru. Více se této funkcionalitě budu věnovat u metody fs_write

Při zjišťování atributů o souboru nebo složce, se kontroluje, zda jsou v lokální cahce stažené všechny chunky dat potřebné pro otevření další části filesystému nebo pro otevření konkrétního souboru. Pokud nějaký chunk schází, přidá se do fronty požadavek na jeho stáhnutí.

fs_readdir

Tato metoda se používá při zobrazování obsahu adresáře. Protože ne všechny soubory mohou být odeslané na datové centrum, nebo aplikace ještě nestáhla všechny soubory, zobrazují se jen ty soubory, které jsou plně dostupné. Toho se docílí přes knihovnu pro správu dat, která poskytuje funkci na výpis adresáře a možnost získat všechny hashe pro jednotlivé soubory. Pomocí těchto hashí se zkontroluje, jestli soubor v lokální cache existuje.

Pokud se v této metodě objeví nějaký soubor, který není dostupný, přidají se do fronty požadavky na stáhnutí chybějících částí.

fs_mknod

Tato metoda se nechá rozdělit ve své funkčnosti do dvou částí. První z nich vytváří klasický prázdný soubor. Protože neznáme další kroky, které budou následovat, je nutné z tohoto prázdného souboru vytvořit hash a předat ho do knihovny pro správu metadat, která vrátí nově vygenerované chunky metadat. Ty se pak vloží do fronty pro odeslání na AS.

Druhá část se stará o vytvoření dočasného souboru. Ten vzniká při úpravě stávajícího souboru, který je uložený na úložišti. Opět se vytvoří pouze prázdný soubor, ale protože po zápisu do něj dojde k přejmenování na původní jméno, nikam se neodesílá a pouze se zapamatuje jeho cesta.

fs_open

Tato metoda se stará o otevření souboru. Metoda zkontroluje, zda je cesta, která se má otevřít, dostupná v metadatech nebo frontě požadavků. Fronta se musí kontrolovat, protože FUSE po dokončení zápisu volá tuto metodu společně s metodou read. Tím se ověřuje dostupnost souboru na disku.

fs_mkdir

Tato metoda vytváří v našem systému složku. Jedná se pouze o změnu metadat předané do jejich správy. Odpovědí jsou nově vygenerované chunky metadat, které se předají do fronty pro zpracování.

fs_rename

Tato metoda přejmenovává soubor nebo složku. Po přejmenování dojde ke změně metadat a vygenerování nových souborů, které se předávají do fronty požadavků k odeslání.

fs_rmdir

Tato metoda odstraní složku ze systému na zadaném umístění. Pro odstranění složky je nutné, aby neobsahovala žádný soubor. V takovém případě se ze systému úspěšně odstraní. Pokud bude obsahovat nějaký soubor, metoda vrátí chybovou hlášku, že složka není prázdná. Při úspěšném odstranění dojde k vygenerování nových chunků metadat, která se musejí vložit do fronty pro odeslání na AS.

fs_unlink

Tato metoda se stará o odstranění dat souboru ze systému. Při tomto odstranění dojde ke smazání datových chunků z lokální cache a úpravě metadat. Pro tuto operaci je volaná knihovní funkce meta_deleteFile(const char* _path). Ta vrátí nově vygenerované chunky metadat, které se musejí předat do fronty požadavků k odeslání.

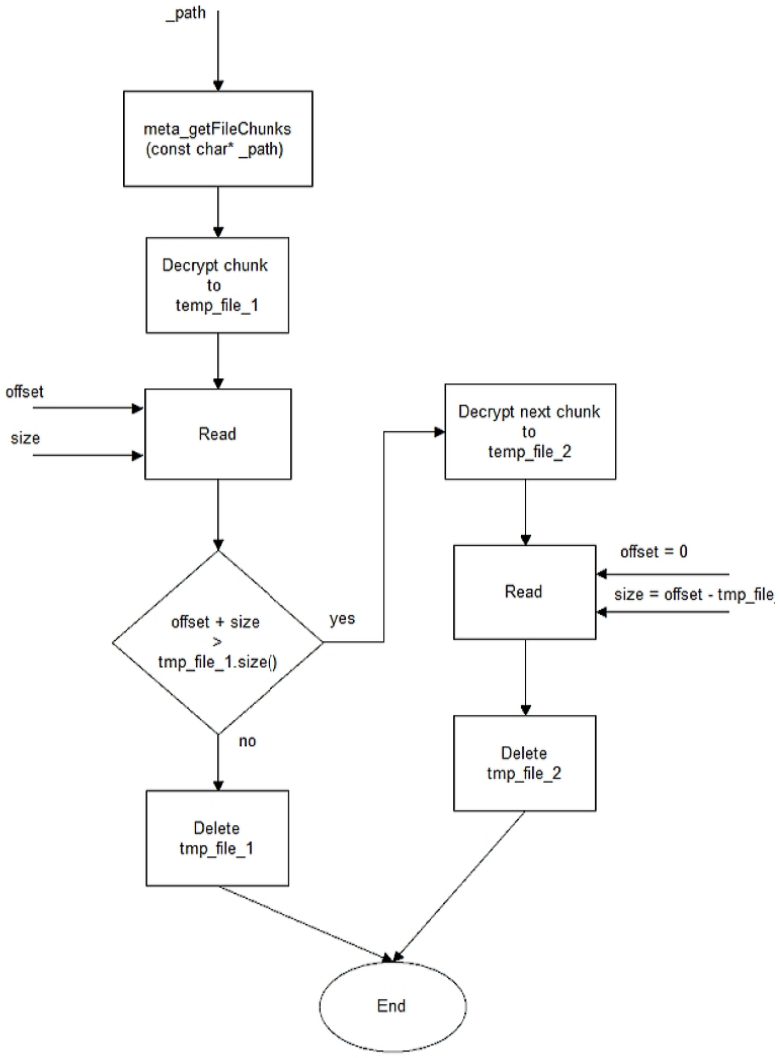
fs_read

Tato metoda se stará o čtení souborů. Spolu s write se jedná o nejsložitější a nejproblematičtější část systému. Read načítá ze souboru informace do interního bufferu, který může mít velikost až 128kB. Pro přečtení jednoho souboru o velikosti řádově MB je tato metoda volaná několikrát. Pozice pro začátek čtení udává offset a množství čtených dat určuje parametr size.

Funkce na svém výstupu žádá zapsání přečtených dat do bufferu a očekává, že jejich velikost bude stejná, jako je velikost parametru size. Cílem v této metodě ty je přečíst vždy takový blok dat, o který si FUSE požádalo.

V této funkci se nejdříve vypočítá, který chunk se má přečíst a následně se celý soubor dešifruje. Před přečtením požadovaného počtu dat, je nutné v souboru ještě provést příslušný posun. Při tomto posunu může dojít k situaci, kdy počet požadovaných dat přeteče do dalšího souboru. V takové situaci se otevře druhý soubor a přečte se zbytek dat.

Dešifrování souboru se provádí do dočasného souboru, který se po dokončení operace opět smaže. Průběh dešifrování a čtení bloku dat ukazuje Obr. 6



Obrázek 6. Dešifrování a čtení souboru

fs_write

Tato metoda se stará o zápis do souboru. Podobně jako u read se využívá přenosu dat po blocích. Tentokrát je velikost bloku 4096B. Pokud je množství dat, které se má zapsat do souboru větší než tento blok, je funkce volaná opakovaně.
Ukázka zápisu do souboru o velikosti 12720B:

1. size=4096, offset=0
2. size=4096, offset=4096
3. size=4096, offset=8192
4. size=432, offset=12288

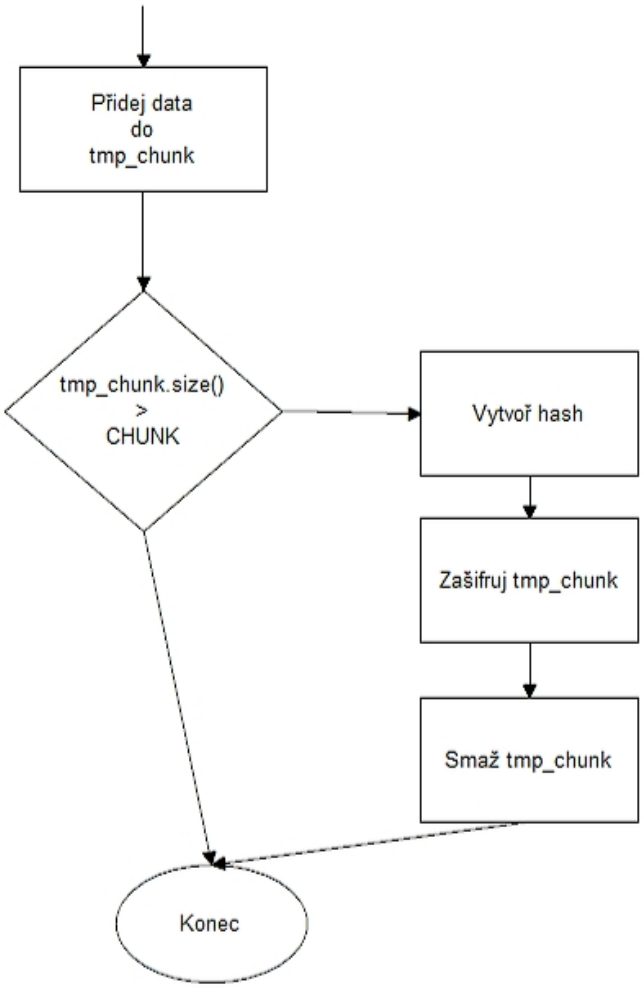
V této metodě bylo nutné vyřešit jakým způsobem zašifrovat data. Šifrování každého bloku by vedlo na velice problematické čtení a je vhodné z důvodu rychlosti zápisu i čtení zašifrovat větší množství dat dohromady. Proto jsem zápis realizoval tak, že

se data zapisují do dočasného souboru, který je po ukončení zápisu v metodě getattr zašifrován pod jménem s hashí a dočasný soubor je smazán.

Pokud se jedná o soubor větší, je nutné provést rozchunkování již při samotném zápisu do souboru. Pokud je aktuální velikost zapsaných dat v souboru větší než mez zvolená pro velikost chunku, vytvoří se hash, zašifruje se soubor a dočasný soubor se smaže kvůli dalšímu zápisu.

V případě, že se jednalo o první vytvořený chunk(platí i pro metodu getattr), tak se nevytváří nová struktura, ale pouze se přepíší hodnoty ve stávajícím záznamu. V dalších zápisech se musí pole chunků vždy zvětšit o jeden záznam.

Proces ukládání dat do souboru je znázorněn na Obr. 7



Obrázek 7. Zápis a šifrování dat do souboru

5.2.4. Rozchunkování souborů

Mezi nefunkční požadavky klientské aplikace patří rozdělení souborů do chunků dat, které se budou odesílat na datové centrum. Rozdělování a šifrování souborů probíhá v metodě write, která realizuje zápis dat do souborů. Pro všechny soubory byla zvolena jednotná velikost chunku 4MB. Pokud se zapisuje soubor, který je menší, velikost chunku odpovídá velikosti souboru, v opačném případě se se k souboru zřetězí víc chunků v pořadí, v jakém se za sebou šifrovali.

5.2.5. Šifrování souborů

S rozchunkováním velice úzce souvisí i šifrování souborů. Každý chunk je samostatně šifrován a v případě čtění dešifrován. Vzhledem k principu fungování FUSE a flexibilní velikosti vnitřního bufferu se šifrování provádí nad dočasným souborem, který je po zašifrování smazán. Nabízela se varianta šifrovat soubory přímo při zapisování, ale tento způsob by byl velice komplikovaný z hlediska čtení a značně by zatěžoval systém při přístupu k větším souborům.

Před samotným šifrováním je nutné vytvořit ze souboru hash. Pro tento účel se využívá funkce

`genHashSha256_64ext(char* _output, char* _filePath, char* _username, char* _timestamp)`
Tato funkce je obsažena v knihovně sfunc a stejně jako knihovna pro správu metadat, je výstupem diplomové práce Tomáše Dyntara [13]. Vstupními parametry této funkce jsou uživatelské jméno, která se zadává při registraci, časová známka a soubor, z kterého se má hash vytvořit. Ve všech případech, kdy se v aplikaci generuje hash z datových chunků, je vstupem dočasný soubor.

Pro účely šifrování a dešifrování se opět využívají funkce z sfunc knihovny, které pracují nad celýni soubory. Jendá se metody:

```
int encryptFileAES128CTR(const char* _readFilePath, const char* _writeFilePath, const unsigned char* _key)
```

```
int decryptFileAES128CTR(const char* _readFilePath, const char* _writeFilePath, const unsigned char* _key).
```

Vstupem těchto metod jsou cesty k souboru, který se má šifrovat/dešifrovat, a do kterého se májí data z výstupních operací zapsat. V případě šifrování je vstupem dočasný soubor, do kterého se zapisovala data a cesta do lokální cache s hashem.

5.2.6. Odesílání dat na AS

V analýze jsem popsal, jakým způsobem by měla probíhat komunikace s AS a nyní nastíním vlastní realizaci.

Protože se ke komunikaci používají zabezpečené TCP sockety, máme zajištěno, že se jednotlivé zprávy při komunikaci neztratí a budou doručeny. Není proto nutné všechny zprávy ještě potvrzovat.

Rozlišení zpráv se provádí podle prvních čtyř bytů, které určují, o jakou operaci se jedná. Jednotlivé zprávy mají následující podobu:

Odeslaní uživatelského chunku
USCH:HASH:VELIKOST_SOUBORU:POČET_REPLIKACÍ

Odeslání datového chunku
SEND:HASH:VELIKOST_SOUBORU:POČET_REPLIKACÍ

Příjem chunku
RECV:HASH

Příjem potvrzení odeslaného chunku
RECV:OK

Potvrzení přijatého chunku
SEND:OK

Příjem oznámení o chunku, který neexistuje
RECV:HASH:NOT_EXIST

Nyní nastíním vlastní průběh komunikace

Odesílání chunku

1. Odeslání zprávy SEND:HASH:VELIKOST_SOUBORU:POČET_REPLIKACÍ pokud se jedná o datový chunk
2. Odeslání zprávy USCH:HASH:VELIKOST_SOUBORU:POČET_REPLIKACÍ pokud se jedná o user chunk
3. Přenos dat
4. Příjem potvrzení od AS

Vyžádání souboru

1. Odeslání zprávy RECV:HASH
2. Klient přijme RECV:HASH:VELIKOST_SOUBORU pokud soubor v datovém centru je
3. Klient přijme RECV:HASH:NOT_EXIST pokud soubor v datovém centru není
4. Přenos souboru
5. Potvrzení příjmu SEND:OK – potvrzuje se odeslání dat z AS ke klientovi

Komunikace klientské aplikace s access serverem je rozdělena do dvou vláken. Jedno se stará o odesílání požadavků na AS, a druhé přijímá odpovědi. Protože se komunikuje pouze přes jeden otevřený socket, vlákno, které odeslalo požadavek, čeká, než se data v systému zpracují a následně je probuzeno odpovědí od přijímacího vlákna. Vlákno čeká na odpověď, protože potřebuje předat informace o úspěchu operace zpátky k frontě požadavků.

5.2.7. Registrace uživatele

Uživatel musí mít možnost provést registraci před prvním spuštěním. V našem systému se nejedná o klasickou registraci, kdy se záznam o jméně a hesle udržuje v nějaké databázi, ale o distribuované řešení, kdy systém nemá žádné konkrétní informace o svých klientech.

Z konceptu systému vychází řešení registrovat uživatele přes jeho vlastní privátní klíč, ze kterého se vytvoří unikátní hash a ta bude reprezentovat user chunk. Tento chunk je kořenovou strukturou pro práci s metadaty. K tomuto účelu se používá funkce z knihovny sfuncs.

```
void genPrivateKeyHash(char* _output, char* _filePath)
```

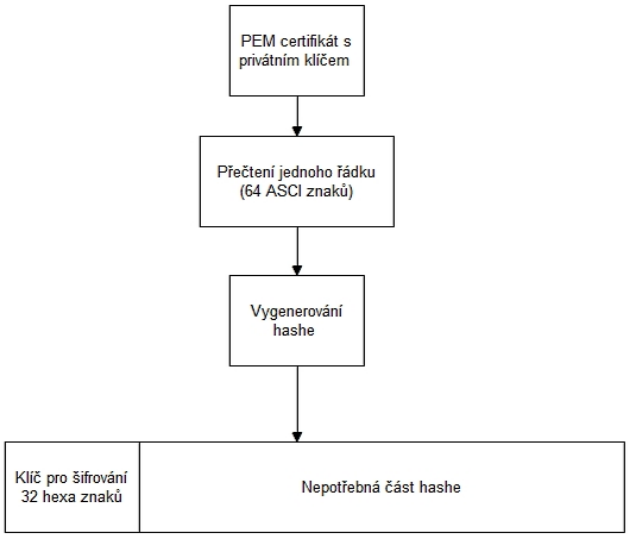
Tato metoda vezme privátní klíč uživatele a vygeneruje z něj hash, který se následně používá pro identifikátor user chunku.

5.2.8. Generování klíče pro šfirování

S registrací úzce souvisí také vygenerování klíče pro šifrování souborů, které musí být neměnné po celou dobu existence uživatele v systému. V analýze jsem zmínil možnosti, jak klíč generovat a nyní ukážu vlastní realizaci v aplikaci.

Aby byl vygenerovaný klíč neměnný, musí se vytvořit z privátního klíče uživatele. Nelze pro šifrování zvolit klíč vygenerovaný z celého PK, protože pak by kdokoliv mohl odposlechnout přenos user chunku a měl by šifrovací klíč pro všechna data.

Z toho důvodu jsem použil pouze první řádek s PK, který je ve formátu PEM. Tento řádek obsahuje 64ASCII znaků, ze kterých se vytvoří hash a z ní se vezme prvních 32 hexadecimálních znaků, které budou sloužit jako šifrovací klíč pro všechny operace. Podmínkou pro vygenerování tohoto klíče je, že PK nesmí být chráněný heslem. Proces vytvoření klíče je na Obr. 8



Obrázek 8. Generování klíče pro šifrování

5.2.9. Přihlášení uživatele

Pokud uživatel v minulosti provedl úspěšnou registraci, systém si ho pamatuje a nyní se ke svému účtu může odkudkoliv přihlásit. Potřebuje k tomu pouze privátní klíč, se kterým se registroval, a musí znát přihlašovací údaje. Po úspěšném vstupu do aplikace se z datového úložiště stáhne uživatelský chunk, ze kterého se načtou všechny ostatní informace potřebné pro fungování aplikace.

5.2.10. Inicializace aplikace

Aplikace pro svoji funkčnost potřebuje aktivní připojení k AS. Pokud se nepodaří spojení navázat, aplikace skončí s hlášením, že se nelze připojit k AS. Při samotné inicializaci se po kontrole stavu tohoto připojení žádá datové centrum o stáhnutí user chunku, aby se zajistila aktuálnost dat v lokálním úložišti. Po úspěšném stažení chunku se zkontroluje, zda v lokální cache je příslušný rootovský chunk. Ten se z user chunku dostane pomocí knihovní funkce:

```
const char* meta_user_getRootMetaChunkHash();
```

Přes klasické souborové funkce se pak v lokální cache zjistí, zda takový chunk existuje a pokud ne, vyžádá se od datového centra. Do doby, než aplikace zjistí, že má ve své cache user chunk i rootovský chunk, není spuštěna a čeká na dokončení všech zmíněných operací.

Výše popsaný postup se využívá, pokud od datového centra přijde kladné odpověď spolu s user chunkem, který se předá do správy metadat pomocí:

```
int meta_user_loadChunk(const char* _fileName)
```

Když ale datové centrum odpoví, že příslušný user chunk nemá, probíhá registrace uživatele, po které se volá funkce:

```
HashArray meta_firstLaunch(const char* _username, unsigned long long _storageSize,
const char* _lastConnect);
```

Tato funkce vytvoří user chunk a rostovský chunk, které se musí odeslat na datové centrum.

5.2.11. Nahrání souborů

FUSE aplikaci nabízí stejné možnosti jako souborový systém operačního systému. Uživatel není omezen na pouhé kopírování souborů složky, ale může využít systémových prostředků pro vytváření nebo mazání souborů přímo v mapovaném disku. Nahrávání souborů lze rozdělit do dvou variant.

V prvním případě se jedná o vytváření souboru přímo v disku našeho systému. Vytvoří se nový soubor s nulovou velikostí, který se předá do fronty požadavků pro odeslání. Protože neznáme další kroky uživatele a rozhodli jsme se uchovávat i prázdné soubory, musí se tato operace provést, i když v souboru nic není. Soubor lze následně klasicky upravovat a po uložení se vytvoří nový s novým hashem, který se opět odešle na AS, které se postará o další zpracování.

Druhý případ nahrání souboru na DS spočívá v klasickém zkopírování souborů na disk. Protože FUSE pracuje v posloupnosti sekvenčních příkazů, je možné kopírovat více souborů najednou. Tyto soubory jsou jeden po druhém systémem zpracovávány a řazeny na konec fronty.

V klientské aplikaci se při této operaci využívají následující FUSE metody:

`getattr(const char* path, struct stat* stbuf)` Pro zjištění artributů před a po vytvořením souboru

`mknod(const char* path, mode_t mode, dev_t rdev)` Pro vytvoření prázdného souboru

`rename(const char* from, const char* to)` Pro přejmenování dočasného souboru vytvořeného v mknod

`write(const char* path, char *buf, size_t size, off_t offset, struct file_info* fi)` Pro zápis dat do souboru

5.2.12. Mazání souborů

Opět se jedná o klasickou systémovou operaci, která je běžná ve všechny souborových systémech. Při této akci se smaže příslušný soubor z metadat pomocí funkce:

`HashArray meta_deleteFile(const char* _path);` Funkce vrátí nově vygenerovaná metadata, která se vloží do fronty pro odeslání na AS. Při tomto odstraňování se také smažou všechny datové chunky, které k souboru náležely.

5.2.13. Sdílení souborů mezi uživateli

FUSE pro sdílení souborů mezi uživateli nenabízí žádnou možnost, a proto není tato funkcionalita implementována.

5.2.14. Synchronizace více uživatelů

V analýze jsem také zmiňoval předávání změn metadat na AS, které by v případě připojení více klientů pod stejným uživatelským účtem rozeslalo tuto změnu. Při současném návrhu by se jednalo o nutnost rozesílat celý rootovský chunk a tím přepisovat obsah lokálního úložiště. Toto řešení se by nebylo moc vhodné, a proto není tato funkcionalita implementována.

Řešením této situace by bylo rozesílání pouze chunků s provedenými změnami. Po příjmu takového chunku by se provedla korektura stávajících metadat a nedocházelo by k přepisování všech informací. Takovou funkcionalitu knihovna pro správu metadat neposkytuje.

5.3. Access server

Hlavním úkolem access severu je zpracovávat požadavky od klienta a zprostředkovávat komunikaci mezi ním a datovým centrem. V analýze byly popsané jednotlivé požadavky na funkčnost a nyní přiblížím jejich vlastní realizaci. Jako programovací byl vybrán jazyk C++ s využitím knihovny POCO, která se používá zejména pro vytváření vláken a realizaci výlučného přístupu.

5.3.1. Klientský modul

Funkce modulu

Klientský modul zajišťuje komunikaci s klientskou aplikací pomocí zpráv popsaných u realizace klientské aplikace. Úspěšně přijaté požadavky se předávají ke zpracování do modulu, který realizuje komunikaci s datovým centrem. Pro každého nově připojeného klienta spustí nové vlákno a proběhne handshake pro ověření připojovaného klienta. Toto vlákno se stará o odesílání odpovědí zpět ke klientovi. Spolu s ním se spustí vlákno, které přijímá požadavky od klienta. Tyto vlákna spolu komunikují přes frontu zpráv.

Rozlišování klientů

Protože klientská aplikace jako první odesílá na server zprávu, která se dotazuje na user chunk, jsou klienti rozlišovaní podle hashe obsaženého v této zprávě. O jejich správu se stará User handler.

Fronta zpráv

Protože access server může přijímat požadavky nejen od klienta, ale také od okolních serverů, bylo nutné zajistit mechanismus, který se o toto postará. Byla vytvořena fronta zpráv, do které se vkládají zprávy pro příslušný access server. Vlákno, které odesílá zprávy klientovi, čeká na probuzení nově příchozím požadavkem (zprávou), které se odešle na klientskou aplikaci. Ve frontě jsou udržovány informace o ID klienta, kterému se má zpráva odeslat a samotná zpráva.

Odeslání souboru na datové centrum

Klientský modul přijme od klienta zprávu, a podle příslušného kódu operace předá informace do Data center modulu. Ten se postará o zpracování a vrátí klientskému modulu hodnotu odpovídající výsledku operace. Dokud není tento proces dokončen, vlákno čeká.

5.3.2. User handler

User handler se stará o správu uživatelských informací a o přístup k nim. Jedná se ve své podstatě o datový kontejner a prostředky zaručují výlučný přístup k datům. Obsahuje následující funkce:

- `int getRouteTableSize();` - Vrací počet uložených záznamů

- `void setNewRecord(string ip, int port, string flag, gnutls_session_t &session);` - Přidá do tabulky nový záznam
- `int haveRecord(string &ip, int &port);` - Prochazí route table, a když narazí na shodný záznam vrací index kde se nachazí, jinak -1
- `gnutls_session_t getSession(int pozice);` - Vrací session na pozici
- `RouteTableRec getRecord(int pozice);` - Vrací záznam s prvky popsaný v analýze
- `string getViaFlag(string &ip, int &port);` - Vrací přes jakou komunikační část (klient/server) se má zpráva odeslat
- `void removeRecord(int index);` - Odstraní záznam na pozici

5.3.3. Route handler

Stejně jako v případě user handleru se jedná o modul, který uchovává informace popsané v analytické části a zaručuje výlučný přístup. Využívá se k udržování směrovacích informací k ostatním serverům v síti.

5.3.4. Data center modul

Jedná se modul, který komunikuje s datovým centrem a je složen ze čtyř částí. Jedná se o nejvíce vytěžovanou část serveru. Složení modulu je následující:

Serverová část pro odesílání

Protože se jedná o část serveru, která je pro všechny klienty připojené na server stejná, je zpracovávána vlastním vláknem. Toto vlákno se spustí ihned po zapnutí serveru a neukončuje se po celou dobu běhu programu. Jeho hlavním úkolem je přijímat odpovědi od datového centra a odesílat mu podle přijatých informací chunky dat. Po odeslání chunku dat se předá zpráva s výsledkem operace do fronty zpráv, kde si jej klientský modul vyzvedne a zpracuje.

Serverová část pro příjem

Stejně jako předcházející část pracuje v samostatném vláknu a má i podobnou funkčnost. Rozdíl je v tom, že tato část se stará o příjem souborů od datového centra. Protože se jedná o část, která bude pravděpodobně neustále komunikovat s datovým centrem, je každý server spuštěný na svém portu.

Klientská část pro příjem a odesílání souborů

Jedná se krátkodobé spojení, které pouze odešle požadavek na datové centrum, které následně odpoví na jeden ze serverů. Jedná se předání zprávy od klientské aplikace na datové centrum.

5.3.5. Modul vnitřní komunikace

Tento modul má za úkol zajistit komunikaci mezi jednotlivými servery. Protože jsme vycházeli z předpokladu, že serverů nebude velké množství, zvolil jsem komunikaci každého s každým. Tím se zajistí nejrychlejší přenos informací z jednoho uzlu na druhý. Nevýhodou je větší množství komunikačních cest. V rámci tohoto modulu jsou implementované následující funkcionality:

Připojení nového uzlu

Klientská část spojení odešle na jeden z aktivních uzlů v síti dotaz s připojením, a pokud úspěšně proběhne handshake, aktivní uzel odešle nazpět zprávu, která obsahuje IP adresy a porty všech ostatních uzlů v síti, na které se má nový uzel připojit. K této komunikaci se používají následující dvě zprávy:

Žádost
CONNECT:IP_ADRESA_ZADAJICHO_SERVERU:PORT_ZADAJICHO_SERVERU

Odpověď
ACCEPT:IP:PORT:POCET_DVOJIC:IP:PORT....

Odeslání souboru

Pokud klient odešle zprávu, která identifikuje odeslání souboru na datové centrum, počká se, až centrum potvrdí uložení chunku, a ten je odeslán na příslušný AS server. Využívá se k tomu následující formát zprávy

SEND_FILE:UID:JMENO_SOUBORU:VELIKOST_SOUBORU

Z časových důvodů nebyly tyto funkcionality plně dokončeny. Jedná se o problém, který vyžaduje, větší časový prostor a pro samotné fungování aplikace jako celku nemá výrazný vliv. Na tuto funkcionalitu není připravena ani klientská aplikace.

6. Testování

V rámci této kapitoly se budu zabývat testováním pilotní implementace svého subsystému a testováním finální podoby celého systému.

Veškeré prvotní testování probíhalo lokálně na localhostu bez, síťové komunikace. Až testování celého systému se provedlo na virtuálních strojích v rámci sítě FEL.

6.1. Testování klientské aplikace

Testování klientské aplikace probíhalo v několika fázích. První testy se zabývaly základní funkčností metod týkající se FUSE a převážně probíhaly nad aplikací bez připojení k AS. Při těchto testech se nevyužívalo šifrování souborů, aby bylo dobře znatelné, jak probíhá především zápis a čtení ze souboru.

Ve druhé fázi testování jsem k metodám FUSE přidal knihovní funkce pro správu metadat a testoval, jak se systém chová s těmito funkcemi. V této fázi se vyskytlo několik nesrovnalostí při integraci knihovny, které byly postupně odstraněny.

Poslední fáze testování se již týkala šifrování dat a komunikace s AS. V tomto bodě vznikl největší problém při vývoji celého subsystému. Protože většina předchozích testů probíhala na souborech do velikosti 8MB, nepodařilo se objevit odlišné chování FUSE týkající se zápisů velkých souborů. To bylo způsobené i tím, že hranice 8MB není fixně stanovené a pokaždé se liší.

Tento problém se týkal detekce ukončení zápisu, který v prvotní verzi probíhal pomocí funkce getattr, která se u menších souborů vždy volala až po ukončení celého zápisu. Protože dokumentace neposkytuje žádný návod, jak detekovat ukončení zápisu, byl zvolen tento princip. Jak se později ukázalo chybně. Chyba byla odstraněna za pomocí detekce zapisovaného bloku do souboru. Jakmile se velikost bloku sníží pod velikost 4096B, jedná se o poslední zapisovaný blok a zápis do souboru byl ukončen. Nevýhodou toho řešení je že nedokáže pracovat se soubory, které jsou zarovnané na tuto velikost.

6.2. Testování access serveru

Pro testování access serveru jsem v prvotních chvílích používal jednoduchého testovacího klienta, který prováděl vždy nějakou specifickou operaci, jako bylo např. odeslání nebo příjem jednoho souboru. Stejným způsobem probíhalo i testování komunikace s datovým centrem.

Protože se v klientské aplikaci ukázalo, že nejsou dostatečně vyvinuty prostředky pro rozesílání změn na další klienty, není funkcionalita propojení sítě dostatečně otestována, a je pouze ověřeno navázání komunikace mezi servery. Testování proběhlo pouze lokálně a vzájemně se propojily tři servery.

6.3. Testování v rámci celého systému

Testování celého systému probíhalo nejdříve na osobních počítačích mezi sítěmi kolejePodolí, kde byl umístěn access server, a Masarykovou kolejí, kde bylo umístěno datové centrum. V rámci těchto testování se ověřila základní funkčnost celého systému a systémy se přenesly na virtuální servery v rámci sítě FEL.

6.3.1. Tetstovací sestava

Protože nebylo nutné využívat více access serverů, testování proběhlo pouze na jednom serveru, který byl umístěn v síti FEL na Karlově náměstí s následujícími parametry:

- 1xVCPU
- 512MB RAM
- 20GB HDD
- Debian 7.0.3 64b
- Veřejná IP adresa

Vlastní konfigurace serveru, na kterém jsou virtuální stroje spuštěny je následující:

- Procesor - 2 x Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
- Operační paměť - 128 GB DDR3/RDIMM 1600
- Síťové připojení - 2 x 10 GbE
- HDD - 4 x 600 GB 3.5"@ 15k SAS (RAID 5)SW
- Veřejná IP adresa

Software

- Linux cn01 3.2.0-4-amd64 1 SMP Debian 3.2.51-1 x86_64 GNU/Linux
- QEMU emulator version 1.1.2 (qemu-kvm-1.1.2+dfsg-6, Debian), Copyright (c) 2003-2008 Fabrice Bellard
- libvirtd (libvirt) 0.9.12.3

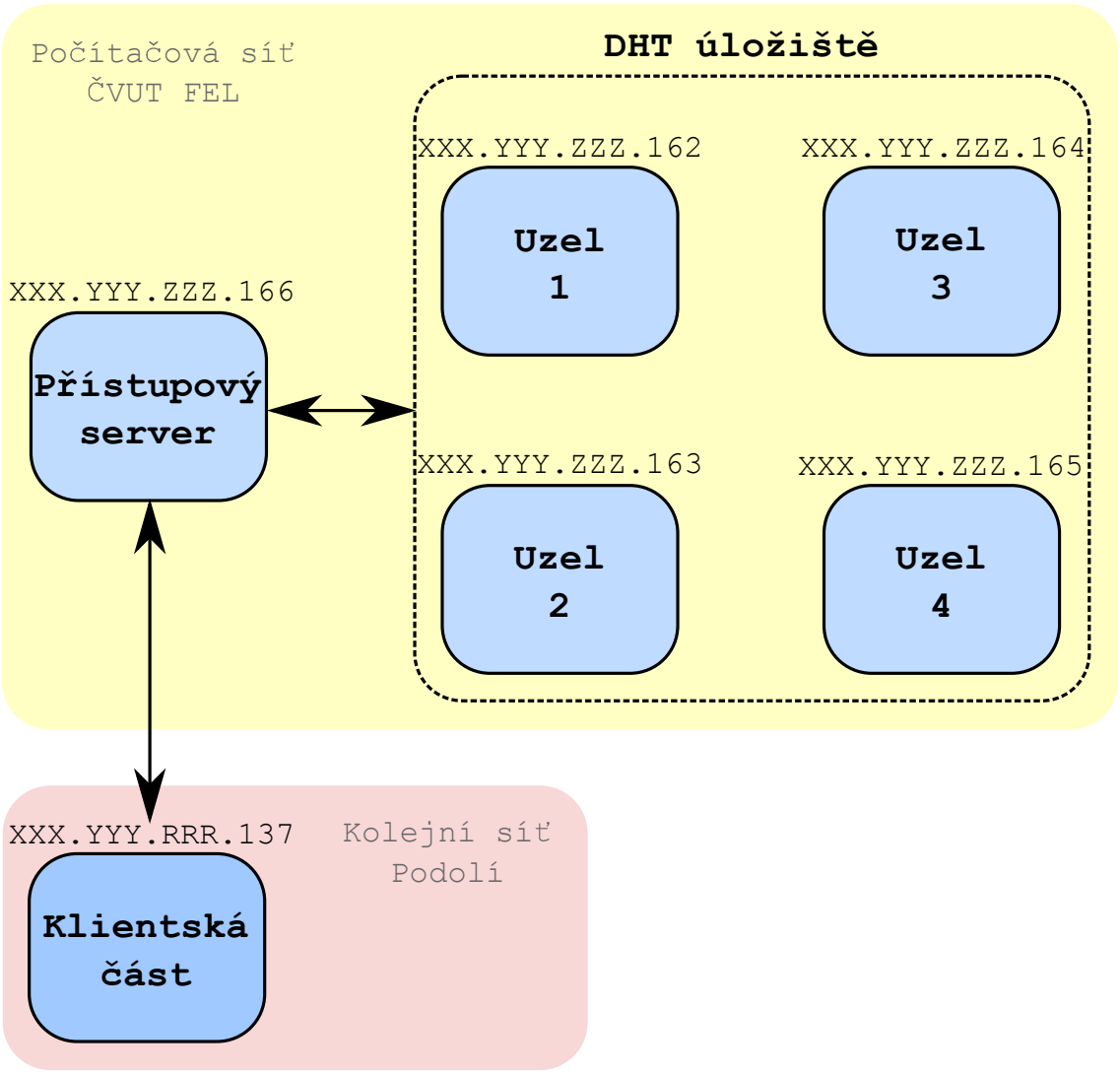
Při testování byla klientská aplikace spuštěna na mém osobním počítači s v síti koleje Podolí s následujícími parametry:

- Procesor: Intel(R) Core(TM)2Duo CPU T6500 2,1GHz
- RAM: 4GB DDR2
- HDD: 20GB
- Operační systém: Ubuntu 12.04LTS 64bit
- Připojení k internetu 1Gb/s symetricky

Schéma propojení jednolivých uzlů je na Obr. 9

6.3.2. Výsledky měření

Před samotným testováním byl předpoklad, že je systém navržen spíše pro větší soubory, a tento předpoklad se naplnil. Nahrání souborů do klientské aplikace nečinil výraznější problém, ale protože dochází k šifrování a rozdělování souborů do chunků, je nutné počítat s časovou prodlevou. Kopírování souboru o velikosti 335MB trvala přes minutu s rychlostí zápisu 5,9MB/s, při kopírování fotografií docházelo jen k minimální prodlevě. Po dobu toho kopírování nedocházelo k žádnému odesílání dat na server, což je způsobené odesíláním chunků per soubor a aplikace čeká na nahrání celého souboru.



Obrázek 9. Schéma testovací sestavy

Měřením jsme si také ověřili, že bezpečností mechanismus v podobě handshaku výrazně brzdí celou komunikaci a prodlužuje dobu přenosu jednotlivých chunků. Výrazný vliv na přenosovou rychlost má také sekvenční zpracování jednotlivých chunků na klientské aplikaci. Celkové testování proběhlo ve dvou blocích, kde se nejdříve vyzkoušela funkčnost na souborech různých typů a velikostí a ve druhém bloku se měřila rychlost u větších souborů.

První blok měření

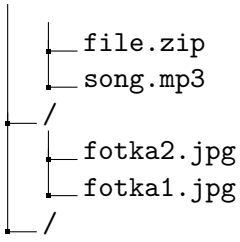
Pro tento blok měření se zvolili typické soubory, které se nejčastěji ukládají. Jednotlivé typy dat s časem přenosu jsou v Tab. 12.

Pro proces nahrávání se stahovala celá adresářová struktura, která měla následující podobu:

/
└── /

Typ souboru	Velikost	Čas nahrávání
Textový soubor	250kB	12s
Zvukový soubor	6,3 MB	15 s
Soubor ve formátu ZIP	10,9 MB	15s

Tabulka 12. Přehled testovaných dat a naměřených časů I



Protože se jednalo o vnořené adresáře, bylo nutné strukturu po zapnutí aplikace projít, aby se vytvořili požadavky na stáhnutí jednotlivých chunků.

Při testování klientské aplikace se objevil problém s nahráváním větších souborů nebo většího množství malých souborů najednou (např. zkopírováním složky) při kterém docházelo k nekorektnímu ukončování aplikace. Tento problém byl způsobený špatnou detekcí ukončení zápisu do souboru. Ale i tento test posloužil k ověření jednotlivých funkcí ve všech částech našeho subsystému.

Druhý blok měření

V tomto bloku se testovaly výrazně větší soubory, než tomu bylo v prvním případě. Jednalo se o test, který výrazně ukázal jednotlivé problémy systému. Jeho výsledky jsou zaznamenány v Tab. 13

Směr	Typ souboru	Velikost	Čas nahrávání	Přenosová rychlost
Nahrávání	Složka s 12 fotkama	23,4 MB	5m	0,62 Mb/s
Nahrávání	Soubor ZIP	23,1 MB	1m	3 Mb/s
Nahrávání	Soubor ZIP	51,3 MB	1m 30s	4,56 Mb/s
Nahrávání	Soubor ZIP	335,4 MB	4 min	11,18 Mb/s
Stahování	—	434,3 MB	7m 30s	7,72 Mb/s

Tabulka 13. Přehled testovaných dat a naměřených časů II

Jak nám potvrdil druhý test, systém je realizován pro přenos větších souborů, se kterými si poradí rychleji než v případě většího množství menších souborů. To je důsledkem zpracování dat na klientské aplikaci, která pro každý nově vytvořený soubor vytváří také nový user chunk a rootvský chunky. V případě, že se jedná pouze o jeden soubor, třeba i větší velikosti, nebyl výrazný problém s jeho zpracováním, avšak naměřené časy ukazují, že je prostor pro optimalizaci.

Výsledkem testování bylo úspěšné ověření fungování celého systému od správy metadat, přes klientskou aplikaci a access server až k datovému centru. Naměřené hodnoty ukazují, že pro optimální fungování systému s uspokojivými časy přenosu bude nutné upravit principy při odesílání souborů a práci s metadaty, která také značně ovlivňuje propustnost celého systému.

7. závěr

Tématem této práce byl subsystém týkající se klientské části, který by měl realizovat přístup k datům uložených v systému. V úvodu své práce jsem otestoval a zhodnotil čtyři současná úložiště. Poznatky nasbírané při tomto testování jsem využil při analýze a návrhu klientské aplikace a access serveru. V následné realizaci jsem implementoval tyto dvě části, které byly nakonec otestovány.

Výsledkem mé práce jsou pilotní implementace klientské aplikace a access serveru, které podporují základní operace jako vložení a nahrání souboru nebo jejich stažení z datového centra. Z časových důvodů se nepodařilo implementovat všechny funkce zmíněné v analýze. Kromě časového faktoru ovlivnil pilotní implementaci fakt, že pro přenos změn mezi jednotlivými klienty je potřeba navrhnout takový mechanismus, který bude spolupracovat s metadaty a půjde jednoduše distribuovat po síti.

Při realizaci byl největší problém s knihovnou FUSE. Jedná se o velice vhodný nástroj pro implementaci vlastního souborového systému, ale postrádal jsem bližší popis chování celé knihovny. Aplikace založená na této knihovně bezesporu splní všechny požadované nároky na práci se soubory, ale nenabízí žádné možnosti pro sdílení dat mezi uživateli nebo ovlivňování dalších vlastností systému. Pro další realizaci bych volil FUSE v kombinaci s klasickou desktopovou aplikací, která by se starala o sdílení mezi uživateli, histori dat apod.

V případě realizace access serveru se jedná o velice omezenou funkčnost na předávání dat mezi klientskou aplikací a datovým centrem. Pro komunikaci mezi jednotlivými uzly by bylo vhodné využít některou z DHT knihoven, anebo vybudovat knihovnu vlastní což je časově velice náročný úkol, který nelze realizovat během jednoho nebo dvou semestrů studia.

Při této práci jsem si prohloubil své znalosti týkající se programování a zejména pak programovacích jazyků C a C++. Vyzkoušel jsem si práci v týmu s kolegy Bc. Martinem Kudrnáčem a Bc. Tomášem Dyntarem, kteří se spolupodíleli na vývoji celého systému.

Příloha A.

Instalační a uživatelská příručka klientské aplikace

A.1. Prostředí a běh

Systém byl vyvíjen a testován na operačním systému Ubuntu 12.04LTS 64bit a testován na systému Debian 7.0.3 64bit.

Protože v programu nebyly použity žádné specifické funkce pro unixové prostředí, měl by být program použitelný i pro platformu Windows, do které je nutné doinstalovat knihovnu Pthreads-w32 nebo Pthreads-w64 v závislosti na architektuře systému.

Klientská aplikace je závislá na knihovně FUSE, GnuTLS, sfunc a MetadataManagement.

A.2. Kompilace

- fuse verze 2.9.3
- m4
- gmp
- nettle
- gnutls
- libgnutls-dev
- libgnutls28
- OpenSSL (vyžaduje knihovna sfunc)

Všechny tyto knihovny jsou přibalené k programu.

Po instalaci výše uvedených balíčku je třeba přidat knihovnu MetedataManagement mezi dynamické knihovny příkazem:

```
export LD_LIBRARY_PATH=*********:/usr/local/lib/
```

hvězdičky v příkazu se musí nahradit cestou, kde je knihovna uložena.

Vlasní kompilace programu je velice jednoduchá a provádí se příkazem

```
make
```

A.3. Nastavení

Konfigurace klientské aplikace se provádí pomocí konfiguračního souboru, který musí mít název config.cfg. Jeho struktura je následující
**** CONFIG FILE ****
username=
privateKeyPath=
cachePath=
numberOfReplication=
ip=
privatePGPkeyPath=
publicPGPkeyPath=
**** CONFIG FILE ****

Konkrétní ukázka
**** CONFIG FILE ****
username=USER_ROOT
privateKeyPath=/home/honza/workspace/Klient/cert/privateKey.key
cachePath=/home/honza/workspace/Klient/src/slozka
numberOfReplication=0
ip=147.32.93.137:8000
privatePGPkeyPath=cert/secret2.asc
publicPGPkeyPath=cert/public2.asc
**** CONFIG FILE ****

username - udává jaké uživatelské jméno se bude v aplikaci používat (v nynějším stavu systému nemá specifický význam)

privateKeyPath - určuje cestu k privátnímu klíči

cachePath - určuje cestu k lokální cahce aplikace (složka musí existovat)

numberOfReplication - udává počet replikací pro celý systém

ip - ip adresa a port AS na který se bude aplikaci připojovat, formát ip:port

privatePGPkeyPath - cesta k príváním klíči pro PGP

publicPGPkeyPath - cesta k veřenému klíči pro PGP

A.4. Spuštění

Ke spuštění aplikace je potřeba mít vytvořené dvě prázdné složky. Jedna bude sloužit jako lokální cache a druhá bude sloužit pro naší aplikaci.

Aplikace při svém spuštění vypíše dialog při stahování user chunku a následně se budou zobrazovat výpisy od knihovny FUSE.

Spuštění aplikace se provádí následujícím příkazem:

```
./klient název\_systémové\_složky absolutní\_cesta\_do\_cache -d
```

Ukázka spuštění

```
make

export LD_LIBRARY_PATH=/home/klient/lib/:/usr/local/lib/

./klient fuse $PWD/cache -d
```

Příloha B.

Instalační a uživatelská příručka access serveru

B.1. Prostředí a běh

Systém byl vyvíjen a testován na operačním systému Ubuntu 12.04LTS 64bit a testován na systému Debian 7.0.3 64bit.

Protože v programu nebyly použity žádné specifické funkce pro unixové prostředí, měl by být program použitelný i pro platformu Windows.

Aplikace je závislá na knihovně POCO a GnuTLS.

B.2. Kompilace

- Poco
- m4
- gmp
- nettle
- gnutls
- libgnutls-dev
- libgnutls28
- OpenSSL (vyžaduje knihovna sfunc)

Všechny tyto knihovny jsou přibalené k programu.

Vlasní kompilace programu je velice jednoduchá a provádí se příkazem

```
make
```

B.3. Nastavení

Konfigurace klientské aplikace se provádí pomocí konfiguračního souboru, který se zadává jako parametr při spouštění. Jeho struktura je následující
**** CONFIG FILE ****
IP_address
ServerName
cachePath
CertFileServer
KeyFileServer
RingFile

DataCenterSendPort
DataCenterRecvPort
DataCenterIP
DataCenterPort
InnerPort
DestInnerAddress

Konkrétní ukázka
**** CONFIG FILE ****
IP_address=127.0.0.1:8000
ServerName=AS_SERVER
cachePath=/home/honza/server/cache
CertFileServer=cert/public.asc
KeyFileServer=cert/secret.asc
RingFile=cert/ring.gpg
DataCenterSendPort=6666
DataCenterRecvPort=6555
DataCenterIP=147.32.81.162
DataCenterPort=8180
InnerPort=8080
DestInnerAddress=0

IP_address - ip adresa, na které se server spustí

ServerName - jméno serveru, slouží pouze pro identifikaci serveru v rámci sítě, jinak nemá žádný speciální význam

cachePath - cesta do cache, kam se budou ukládat soubory při komunikaci mezi klientem a datovým centrem (složka musí existovat)

CertFileServer - cesta k veřejnému klíči pro PGP

KeyFileServer - cesta k privátnímu klíči pro PGP

RingFile - cesta k ringu pro PGP

DataCenterSendPort - port, na kterém bude access server komunikovat s datovým centrem při odesílání souborů

DataCenterRecvPort - port, na kterém bude access server komunikovat s datovým centrem při přijímání souborů

DataCenterIP - IP adresa datového centra, na který se access server dotazuje

DataCenterPort - port datového centra, na který se access server dotazuje

InnerPort - port, na kterém bude server naslouchat pro vnitřní komunikaci

DestInnerAddress - IP adresa, na kterou se bude server připojovat v rámci vnitřní sítě mezi AS, pokud je 0 server se nikam nepřipojuje

B.4. Spuštění

Ke spuštění aplikace je potřeba mít vytvořenou složku, která bude sloužit jako cache. Aplikace při spuštění vypíše informace o nastartování jednotlivých serverů a čeká na připojení jednotlivých uživatelů. Spuští se následuícím příkazem:

```
./AS_server název_configuracniho_souboru
```

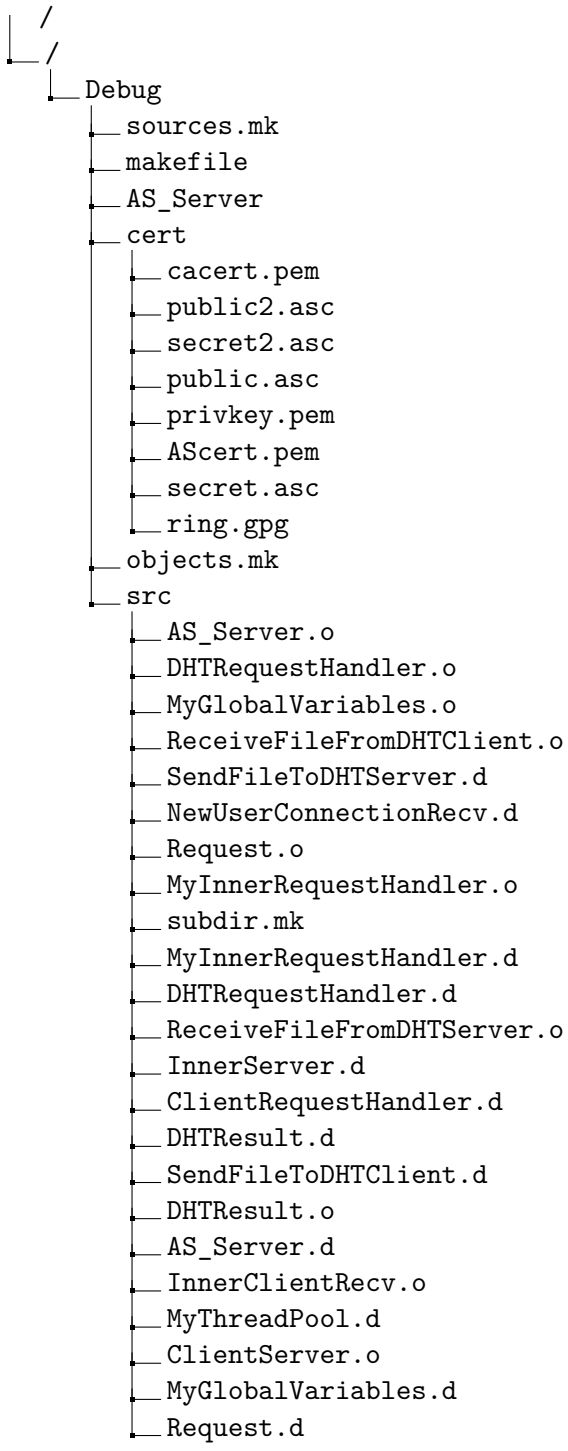
Ukázka spuštění

```
make
```

```
./AS_server config.cfg
```

Příloha C.

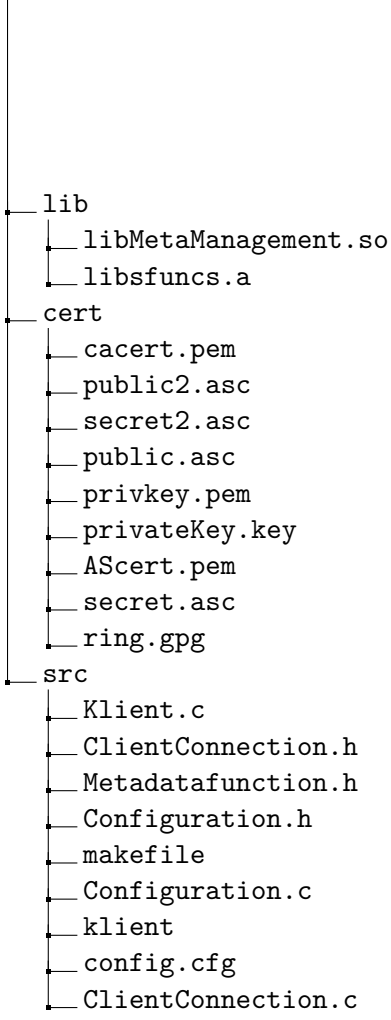
Obsah přiloženého CD



```
        ├── ReceiveFileFromDHTServer.d
        ├── IpPort.o
        ├── ReceiveFileFromDHTClient.d
        ├── RequestQueueHandler.o
        ├── NewUserConnectionRecv.o
        ├── InnerServer.o
        ├── NewUserConnection.o
        ├── InterCommunication.o
        ├── InnerServerRecv.o
        ├── IpPort.d
        ├── ClientRequest.d
        ├── MyInnerConnectionHandler.d
        ├── UserTableHandler.d
        ├── SendFileToDHTServer.o
        ├── RouteTableHandler.d
        ├── ClientRequest.o
        ├── UserTableHandler.o
        ├── RouteTableHandler.o
        ├── ClientRequestHandler.o
        ├── ClientServer.d
        ├── MyThreadPool.o
        ├── Configuration.o
        ├── MyInnerConnectionHandler.o
        ├── RequestQueueHandler.d
        ├── InnerClient.d
        ├── InnerServerRecv.d
        ├── SendFileToDHTClient.o
        ├── InterCommunication.d
        ├── UserInfo.d
        ├── RouteTableRec.d
        ├── Configuration.d
        ├── NewUserConnection.d
        ├── UserInfo.o
        ├── InnerClientRecv.d
        ├── InnerClient.o
        └── RouteTableRec.o
    ├── Config2.cfg
    ├── Config.cfg
    ├── cert
    │   ├── public2.asc
    │   ├── secret2.asc
    │   ├── public.asc
    │   ├── secret.asc
    │   └── ring.gpg
    └── src
        ├── UserInfo.h
        ├── InterCommunication.cpp
        ├── InnerClientRecv.h
        ├── InnerServer.cpp
        └── NewUserConnection.h
```

```
├── ClientRequest.cpp
├── UserTableHandler.cpp
├── ReceiveFileFromDHTServer.h
├── Configuration.cpp
├── RouteTableRec.h
├── ReceiveFileFromDHTClient.cpp
├── Request.cpp
├── UserTableHandler.h
├── ClientRequest.h
├── IpPort.cpp
├── MyInnerRequestHandler.cpp
├── SendFileToDHTServer.cpp
├── RouteTableHandler.cpp
├── InnerServer.h
├── AS_Server.cpp
├── NewUserConnectionRecv.h
├── RequestQueueHandler.cpp
├── Configuration.h
├── NewUserConnection.cpp
├── MyGlobalVariables.cpp
├── SendFileToDHTServer.h
├── ClientRequestHandler.h
├── IpPort.h
├── RouteTableHandler.h
├── NewUserConnectionRecv.cpp
├── SendFileToDHTClient.h
├── Request.h
├── DHTRequestHandler.cpp
├── ReceiveFileFromDHTServer.cpp
├── DHTResult.cpp
├── MyInnerConnectionHandler.cpp
├── InnerClient.cpp
├── SendFileToDHTClient.cpp
├── ClientServer.h
├── RouteTableRec.cpp
├── UserInfo.cpp
├── InnerServerRecv.cpp
├── MyInnerRequestHandler.h
├── InnerClientRecv.cpp
├── ClientServer.cpp
├── NewUserConnectionRecv.cpp
├── MyGlobalVariables.h
├── InterCommunication.h
├── DHTRequestHandler.h
├── ClientRequestHandler.cpp
├── RequestQueueHandler.h
├── DHTResult.h
├── MyInnerConnectionHandler.h
├── ReceiveFileFromDHTClient.h
└── InnerClient.h
```

```
        InnerServerRecv.h
/
    gnutls-3.1.22.tar.xz
    gnutls-3.1.0.tar.xz
    gmp-5.1.3.tar.bz2
    nettle-2.7.1.tar.gz
    poco-1.4.6p2-all.tar.gz
/
    felthesis
        acknowledgement.tex
        ch03.tex
        spolecnynavrh-eps-converted-to.pdf
        ch04.tex
        spolecnatabulka.jpg
        Klient_structure.pdf
        testovani.eps
        spolecnynavrh.eps
        testovani-eps-converted-to.pdf
        janurja.bib
        ch06.tex
        fuse_structure.pdf
        ch05.tex
        declaration.tex
        abstract.tex
        abbreviations.tex
        lev.pdf
        cipher_key.pdf
        fuse_read.pdf
        fuse_write.pdf
        app02.tex
        app01.tex
        janurja.pdf
        janurja.bib
        app03.tex
        As_structure.pdf
        janurja.tex
        ch01.aux
        ch01.tex
        ch02.tex
        janurja.synctex.gz
        template.tex
        fuse_structure.png
        template.bib
/
    Debug
        sources.mk
        makefile
        objects.mk
        src
            subdir.mk
```


lib
libMetaManagement.so
libsfuncs.a
cert
cacert.pem
public2.asc
secret2.asc
public.asc
privkey.pem
privateKey.key
AScert.pem
secret.asc
ring.gpg
src
Klient.c
ClientConnection.h
Metadatafunction.h
Configuration.h
makefile
Configuration.c
klient
config.cfg
ClientConnection.c

Literatura

[1] Cnews.cz. "Dropbox má 100 mil. uživatelů. Každý den uloží miliardu souborů - MWC 2013." In: *Dropbox má 100 mil. uživatelů. Každý den uloží miliardu souborů - MWC 2013.* (2013). URL: http://www.cnews.cz/dropbox-ma-100-mil-uzivatelu-kazdy-den-ulozi-miliardu-souboru-mwc-2013.

[2] Živě.cz. "Hackujeme Dropbox". In: *Hackujeme Dropbox: jak vyloupit cizí data.* (2013). URL: http://www.zive.cz/clanky/hackujeme-dropbox-jak-vyloupit-cizi-data/sc-3-a-156592/default.aspx.

[3] Ubuntu One. *Ubuntu One help font family.* URL: https://one.ubuntu.com/help/faq/ (cit. 16. 01. 2014).

[4] Petr Vaníček. "Putování cloudovými službami – Ubuntu One". In: *Putování cloudovými službami – Ubuntu One. Osobní blog Petra Vaníčka* (2012). URL: http://www.e-vanicek.cz/2012/03/putovani-cloudovymi-sluzbami-ubuntu-one.html.

[5] CloudMe. *CloudMe font family.* URL: https://www.cloudme.com/en/pricing (cit. 16. 01. 2014).

[6] Root.cz. "OwnCloud: vlastní cloudové úložiště". In: *OwnCloud: vlastní cloudové úložiště* (2012). URL: http://www.root.cz/clanky/owncloud-vlastni-cloudove-uloziste/.

[7] Samuraj-cz. "OwnCloud 5.0 - internetové úložiště ve firmě". In: *OwnCloud: vlastní cloudové úložiště* (2012). URL: http://www.samuraj-cz.com/clanek/owncloud-5-0-internetove-uloziste-ve-firme/.

[8] *OwnCloud User Manual.* URL: http://doc.owncloud.org/server/5.0/ownCloudUserManual.pdf.

[9] Dell Inc. *PowerEdge R710 Rack Server font family.* URL: http://www.dell.com/us/business/p/poweredge-r710/pd (cit. 02. 02. 2014).

[10] EMC Corporation. *CLARiiON AX4 font family.* URL: http://www.emc.com/products/detail/hardware/clariion-ax4.htm (cit. 02. 02. 2014).

[11] Petr Krčmář. *FUSE: celý svět je souborový systém font family.* URL: http://www.root.cz/clanky/fuse-cely-svet-je-souborovy-system/ (cit. 04. 10. 2011).

[12] *Filesystem in Userspace.*

[13] Tomáš Dyntar. *Distribuované úložiště dat - správa metadat.* Diplomová práce. Praha, 2014.

[14] Martin Kudrnáč. *Distribuované úložiště dat - správa dat.* Diplomová práce. Praha, 2014.

[15] Martin Volf. *Distribuované úložiště dat.* Diplomová práce. Praha, 2014.

[16] Tuček Petr. *Distribuované úložiště dat.* Diplomová práce. Praha, 2012.

[17] Živě.cz. "Dropbox". In: *Dropbox: Mýty, fakta a tipy z praxe. Jak na počítač.* (2012). URL: Dostupn%C3%A9%20z:%20http://jnp.zive.cz/dropbox-myty-fakta-a-tipy-z-praxe.

[18] Dropbox. "Everything you need to work smarter. Dropbox". In: *Everything you need to work smarter. Dropbox* (2013). URL: `https://www.dropbox.com/business/pricing`.

[19] Dropbox.com. *Help Center.font family.* URL: `https://www.dropbox.com/help` (cit. 07. 01. 2014).

[20] Zive.cz. "Apple koupil doménu iCloud, zaplatil za ní 4,5 milionu dolarů." In: *Apple koupil doménu iCloud, zaplatil za ní 4,5 milionu dolarů.* (2011). URL: `http://www.zive.cz/bleskovky/apple-koupil-domenu-icloud-zaplatil-za-ni-45-milionu-dolaru/sc-4-a-156840/default.aspx`.