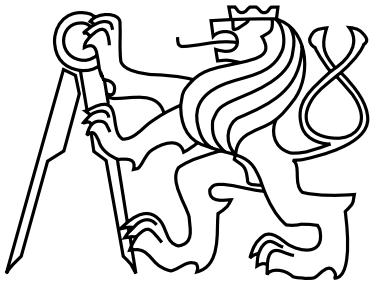
Na tomto místě bude oficiální zadání vaší práce

- Toto zadání je podepsané děkanem a vedoucím katedry,
- musíte si ho vyzvednout na studiijním oddělení Katedry počítačů na Karlově náměstí,
- v jedné odevzdané práci bude originál tohoto zadání (originál zůstává po obhajobě na katedře),
- ve druhé bude na stejném místě neověřená kopie tohoto dokumentu (tato se vám vrátí po obhajobě).

České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra řídicí techniky



Diplomová práce

Distribuované úložiště dat

Martin Volf

Vedoucí práce: Ing. Peter Macejko

Studijní program: Otevřená Informatika, Magisterský

Obor: Počítačové inženýrství

8. ledna 2014

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.

V Úvalech u Prahy dne 31. 12. 2013 .....

Abstract

The goal of this paper is to examine the existing distributed systems for data storage and synchronization of data between computers. Based on gained experiences, develop concept of distributed data storage system and it's pilot implementation. The proposed system must be secure both in terms of data privacy and terms of availability of data.

Abstrakt

Cílem této práce je prozkoumat existující distribuované systémy pro ukládání dat a synchronizaci dat mezi počítači. Podle získaných poznatků vytvořit návrh a pilotní implemetaci distribuovaného systému pro ukládání a synchronizaci dat. Navržený systém musí být bezpečný, jak z hlediska utajení dat, tak z hlediska přístupnosti dat.

Obsah

1 Úvod **1**

2 Popis problému, specifikace cíle **3**
2.1 Bližší popis problému . . . 3
2.2 Specifikace cíle . . . 4

3 Rešerše existujících řešení **5**
3.1 OwnCloud . . . 5
3.1.1 Struktura systému . . . 5
3.1.2 Výhody . . . 5
3.1.3 Nevýhody . . . 6
3.2 Wuala . . . 6
3.2.1 Struktura systému - do roku 2010 . . . 7
3.2.2 Výhody . . . 7
3.2.3 Nevýhody . . . 8
3.3 Google File systém . . . 8
3.3.1 Struktura systému . . . 8
3.3.2 Výhody . . . 9
3.3.3 Nevýhody . . . 9
3.4 Drop box . . . 9
3.4.1 Struktura systému . . . 10
3.4.2 Výhody . . . 10
3.4.3 Nevýhody . . . 10
3.5 Zhodnocení systémů . . . 11
3.5.1 Návrh „ideálního“systému . . . 11

4 Analýza a návrh řešení **13**
4.1 Obecné požadavky . . . 13
4.1.1 Multiplatformní systém . . . 13
4.2 Funkční požadavky . . . 13
4.2.1 Požadavky na systém ze strany Klienta . . . 13
4.2.1.1 Dostupnost . . . 13
4.2.1.2 Spolehlivost - data se neztratí . . . 13
4.2.1.3 Jednoduché ovládání . . . 13
4.2.1.4 Bezpečnost dat . . . 14

4.2.1.5 Automatická aktualizace/synchronizace souborů na lokálních úložištích . . . 14
4.2.1.6 Možnost sdílení dat . . . 14
4.2.1.7 Verzování dat, krátkodobá historie dat . . . 14
4.2.2 Požadavky na systém ze strany provozovatele . . . 14
4.2.2.1 Minimalizace dat, co nejméně dat na úložištích . . . 14
4.2.2.2 Minimalizace nároků na rychlost komunikace . . . 14
4.2.3 Struktura systému . . . 14
4.3 Datový server . . . 15
4.3.1 Funkce serveru . . . 15
4.3.1.1 Inicializace / synchronizace . . . 15
4.3.1.2 Ulož data . . . 15
4.3.1.3 Načti data . . . 15
4.3.1.4 Smaž data . . . 16
4.3.1.5 Monitoring vytížení . . . 16
4.4 Databázový server . . . 16
4.4.1 Struktura databáze . . . 16
4.4.2 Funkce serveru . . . 16
4.4.2.1 Synchronizace . . . 16
4.4.2.2 Monitoring . . . 16
4.4.2.3 Ukládání dat . . . 17
4.4.2.4 Načítání dat . . . 18
4.4.2.5 Mazání dat . . . 18
4.5 Access server . . . 18
4.5.1 Funkce serveru . . . 19
4.5.1.1 Nahrání souboru . . . 19
4.5.1.2 Načtení souboru . . . 19
4.5.1.3 Smazání souboru . . . 19
4.5.1.4 Ostatní požadavky . . . 19
4.6 Klientská aplikace . . . 19
4.6.1 Funkce aplikace . . . 20
4.6.1.1 Nahrání souboru . . . 20
4.6.1.2 Načtení vzdáleného souborového systému . . . 20
4.6.1.3 Nahrání souboru na server . . . 20
4.6.1.4 Načtení souboru . . . 20
4.6.1.5 Vyhledání veřejného souboru . . . 20
4.6.1.6 Ostatní funkce . . . 20
4.7 Notify server . . . 21
4.8 Komunikační model . . . 21
4.8.1 Klientská aplikace . . . 21
4.8.2 Access server . . . 21
4.8.3 Data server . . . 23
4.8.3.1 Struktura komunikace při přenosu dat: . . . 24
4.8.3.2 Struktura komunikace při porovnání souborů: . . . 24
4.8.4 Databázový server . . . 24
4.8.4.1 Struktura komunikace při synchronizaci . . . 24

4.8.4.2 Struktura komunikace s klientem . . . 25
4.8.5 Notify server . . . 26

5 Realizace **27**
5.1 Vývojové a aplikační prostředí . . . 27
5.2 Komunikace . . . 27
5.3 Datový server . . . 28
5.3.0.1 Inicializace / synchronizace . . . 28
5.3.0.2 Ulož data . . . 29
5.3.0.3 Načti data . . . 29
5.3.0.4 Smaž data . . . 29
5.4 Databázový server . . . 29
5.4.1 Databáze . . . 29
5.4.2 Popis funkcí serveru . . . 30
5.4.2.1 Synchronizace mezi databázovými servery . . . 30
5.4.2.2 Inicializace / Synchronizace datových serverů . . . 30
5.4.2.3 Monitoring . . . 30
5.4.2.4 Ukládání dat . . . 31
5.4.2.5 Načítání dat . . . 31
5.4.2.6 Mazání dat . . . 32
5.5 Access server . . . 32
5.5.1 Nahrání souboru . . . 33
5.5.2 Načtení souboru . . . 33
5.5.3 Smazání souboru . . . 33
5.6 Klientská aplikace . . . 34
5.6.1 Komunikace s uživatelem . . . 34
5.6.2 Registrace uživatel . . . 34
5.6.3 Přihlášení uživatel - načtení seznamu souborů . . . 35
5.6.4 Nahrání souboru do systému . . . 36
5.6.5 Načtení souboru . . . 37
5.6.6 Vyhledání veřejného souboru . . . 37
5.7 Notify server . . . 37

6 Testování **39**
6.1 Stabilita, spolehlivost . . . 39
6.2 Testy bezpečnosti . . . 40
6.3 Testy multiplatformnosti . . . 40
6.4 Rychlostní testy . . . 40

7 Závěr **43**

A Seznam použitých zkratek **47**

B Instalační a uživatelská příručka **49**
B.1 Instalace v systému Windows . . . 49
B.2 Instalace v systému Linux . . . 49
B.3 Nastavení jednotlivých částí systému . . . 50
B.3.1 Klientská aplikace . . . 50
B.3.1.1 config.cfg . . . 50
B.3.2 Access server . . . 50
B.3.2.1 config.cfg . . . 50
B.3.3 Datový server server . . . 50
B.3.3.1 config.cfg . . . 51
B.3.4 Databázový server server . . . 51
B.3.4.1 config.cfg . . . 51
B.4 Provozní příručka . . . 52
B.4.1 Spuštění systému . . . 52
B.4.2 Použití . . . 52

C Obsah přiloženého CD **53**

Seznam obrázků

3.1 Struktura systému OwnCloud 6
3.2 Struktura systému Wuala do roku 2010 7
3.3 Struktura systému Google File systém 8
3.4 Struktura systému DropBox 10

4.1 Struktura systému 15
4.2 Model systémové databáze 17
4.3 Komunikační schéma systému 22

5.1 Registrační formulář klientské aplikace 35
5.2 Přihlašovací okno aplikace 35
5.3 Hlavní okno aplikace 36
5.4 Dialog načtení souboru 37

Seznam tabulek

6.1 Měření rychlostí systému 41
6.2 Tabulka rychlostí ostatních systémů 42

Kapitola 1

Úvod

Způsob ukládání a přenosu dat je dlouhodobý problém, který řeší jak normální uživatelé PC při ukládání fotek z dovolené, tak velké nadnárodní korporace, které potřebují nějakým způsobem uchovat data, se kterými pracují při produkci, nebo tajné dokumenty, které vypovídají o historii, popřípadě budoucím rozvoji firmy. Doposud nebyl nalezen ideální systém pro ukládání dat, jelikož zde figurují faktory, které se navzájem vyvracejí.

Uživatelé na systémy pro ukládání dat kladou následující podmínky.

První velký problém je zabezpečení dat. Jak zajistit, aby se k datům nedostal nikdo nepovolaný. To mohu zajistit použitím lokálního úložiště u sebe doma nebo ve firemní síti.

Tím ale vznikne problém přístupnosti dat. Pokud data mám na datovém úložišti doma nebo ve firmě, jak se k datům dostanu, když jsem na dovolené nebo na služební cestě, tak aby data nebyla napadnutelná nikým dalším.

Druhým velkým problémem je spolehlivost zařízení. Pokud mám data na běžném počítači, disku, spolehlivost není vysoká a není nic horšího než data ztratit. Pokud si pořídím „spolehlivé“ úložné zařízení, u kterého výrobce garantuje vysokou výdrž nebo okamžité řešení problému v případě výpadku, musím počítat s vysokou pořizovací cenou.

Tím se dostáváme k dalšímu kritériu, což je cena. Toto kritérium je velmi důležité pro každého uživatele. Mezi další patří náročnost na provoz, užívání systému a další.

Cílem všech provozovatelů, datových úložišť je tedy vyřešit co nejlépe tyto nároky uživatelů, při co nejnižší pořizovací ceně.

Každý provozovatel se specializuje na určitý typ klientů, pro kterého jsou důležité jen některé z této množiny nároků na systém. Buď je systém bezpečný a spolehlivý, i když drahý. Což je přijatelné řešení pro velké firmy. Nebo se na bezpečnost, či spolehlivost tolik nedbá, ale je nižší pořizovací cena.

Řešením umožňujícím splnit všechny tyto podmínky, při relativně nízkých pořizovacích nákladech, je využití distribuovaného datového úložiště (cloudového úložiště[12]). To pracuje na principu duplikace všech částí systému, i dat samotných mezi několik počítačů, čímž zaručuje spolehlivost systému při nízkých pořizovacích nákladech, jelikož jednotlivé počítače nemusí být speciální „spolehlivá“ zařízení. Problém zabezpečení, před zneužitím třetí osobou se dá vyřešit pomocí omezení přístupu a šifrování dat. Tím umožníme systému běžet kdekoliv, tudíž vyřešíme i problém přístupnosti.

Kapitola 2

Popis problému, specifikace cíle

2.1 Bližší popis problému

Nyní si podprobněji popíšeme aspekty této problematiky a jejich možná řešení.

Jak již bylo řečeno v úvodu, v oblasti datových úložišť je několik podstatných prvků, které je nutné zajistit. Spolehlivost, zabezpečení, dostupnost a cena, jelikož peníze jsou vždy až na prvním místě.

- **Spolehlivost** Tento problém se skládá ze dvou částí. První je spolehlivost systému ze strany přístupnosti dat. Data by měla být dostupná a to vždy. Nemělo by dojít k situaci, že z důvodu výpadku nějaké komponenty systému by data nebyla přístupná (i jen dočasně). Druhá část je odolnost před ztrátou dat. I v případě zničení zařízení se nesmí data ztratit. Toto je asi nejdůležitější prvek každého datového úložiště. Oba tyto problémy jsou obtížně vyřešitelné a není možné garantovat jejich stoprocentní splnitelnost.

  Vyřešit tyto problémy lze několika způsoby. Pořízením kvalitního počítače(serveru), který je určený pro nepřetržitý provoz s vysokou spolehlivostí. Toto řešení je však velice nákladné, jelikož takové počítače jsou řádově dražší než klasická PC. I toto řešení má však své vady. Pokud vypadne elektrický proud, nebo je přerušený přístup k datové síti, nepomůže ani drahý HW.

  Druhou možností je distribuce mezi více počítačů, které běží na sobě nezávisle a jsou synchronizovány. Data pak jsou umístěna na několika z nich zároveň. V případě, že jsou stroje i geograficky na různých místech, řeší to oba problémy s největší možnou mírou.

- **Dostupnost** Na tento problém můžeme mít dva různé pohledy. Pohled z dostupnosti časové, aby byl soubor přístupný kdykoliv, ten je však obsažen již v problému spolehlivosti. Druhý pohled je ze strany přístupnosti z hlediska lokality. Data by měla být přístupná odkudkoliv. Ať již jste ve firmě, doma, nebo někde na cestách. Tento problém se dá vyřešit poměrně jednoduchým postupem, přípojením zařízení na internet a zařízením přístupu uživatelů k datům. Tím však narážíme na další problém a to je zabezpečení.

- **Zabezpečení** Zabezpečení je specifický problém. Jelikož není možné vytvořit dokonalé zabezpečení. Vždy k datům bude mít někdo přístup a ten někdo je může ukrást. Další faktor tohoto problému je, že čím více zabezpečíme data, tím více znesnadníme jejich dostupnost. Jelikož tento problém se netýká většiny uživatelů, spíše zákazníků firemních, není u většiny veřejných datových úložišť řešen, nebo je řešen pouze okrajově. Z toho důvodu velké firmy si vytvářejí vlastní datová úložiště a menší se spokojí se sdílenými diskovými poli, nebo něčím podobným.

  Proto většina existujících datových úložišť tento problém neřeší a bezpečnost dat řeší jen okrajově. Většinou pouze omezením přístupu ostatních uživatelů. To však nezamezuje vlastníkům systému k přístupu k datům. Řešením je šifrování dat, což však z různých důvodů většina systémů nepodporuje.

Řešením na všechny tyto problémy je využití šifrovaného cloudového systému. Ten zajistí bezpečnost, spolehlivost i dostupnost dat. Navíc je možné takovýto systém provozovat i na běžných PC, tudíž cena systému není příliš vysoká.

2.2 Specifikace cíle

Cílem této práce je seznámit se s aktuálně dostupnými datovými úložišti. Následně dle získaných informací navrhnout a vytvořit pilotní verzi distribuovaného datového úložiště, které by co nejlépe vyhovovalo potřebám firemního prostředí. Což znamená, že se zaměřím na dostupnost dat, bezpečnost a spolehlivost. Cena provozu systému, není v tomto případě hlavním kritériem. Je sice nutné systém optimalizovat, avšak ne na úkor ostatních kritérií.

Jelikož většina firem funguje na uzavřených sítích a komunikace s vnějším světem probíhá pomocí přístupových serverů, je nutné zajistit, aby data byla dostupná i z vnější sítě a to bezpečnou cestou.

Kapitola 3

Rešerše existujících řešení

Analyzoval jsem několik nejvýznamnějších existujících řešení datových úložišť. Ke každému jsem doplnil zhodnocení, kde mají dle mého názoru své přednosti a kde zase nedostatky.

3.1 OwnCloud

Systém OwnCloud [3] není cloudové úložiště, ačkoli tomu název napovídá[4]. Jedná se o vzdálené úložiště dat, synchronizační systém a systém pro sdílení dat. Důvod, proč se nejedná o cloudové úložiště dat, je že celý systém běží na jediném serveru 3.1.

O zabezpečení dat se stará ochrana přístupu k datům pomocí oprávnění na data nahlížet. Tato vlastnost umožňuje snadné sdílení souborů. Přístup k serveru probíhá pomocí zabezpečeného spojení, ať už při použití webového rozhraní, nebo klientské aplikace. Dále je také možnost data šifrovat, ale je omezena na použití klientské aplikace, kde se data zašifrují podle hesla uživatele a na serveru jsou dále nečitelná. S tím ovšem odpadá možnost data sdílet.

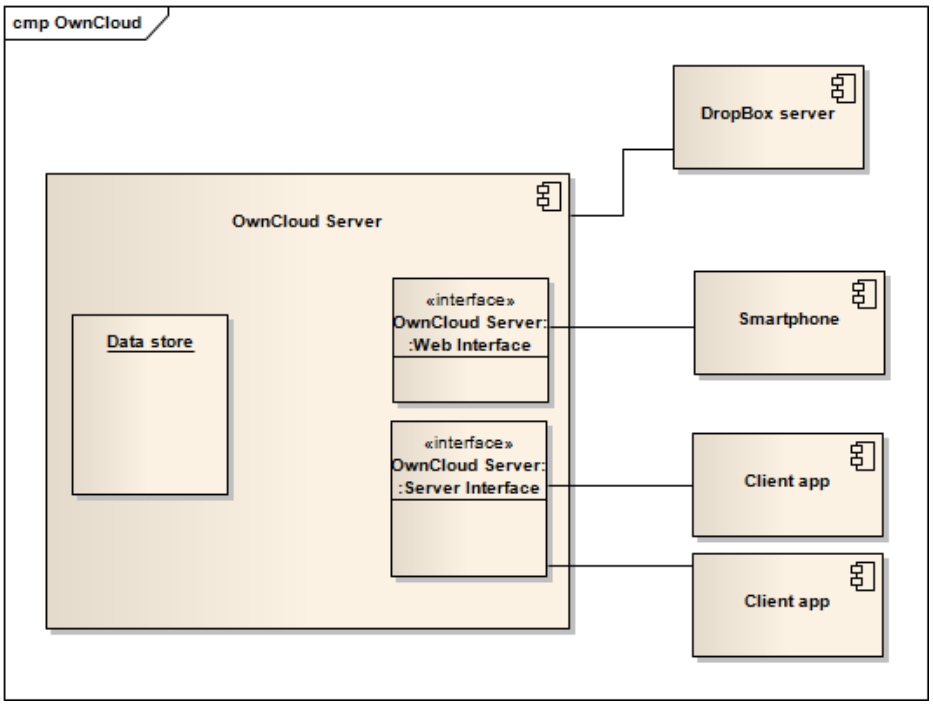
Výhodou systému je, že se neomezuje jen na synchronizaci souborů, umožňuje synchronizaci kalendářů a galerií obrázků. Dále umožňuje připojit jiná cloudová úložiště do systému (například DropBox) a sjednocuje přístup k souborům.

3.1.1 Struktura systému

Obrázek struktury systému: 3.1.

3.1.2 Výhody

- podporuje adresářovou strukturu souborů na serveru
- snadné sdílení mezi jednotlivými uživateli, jejich skupinami, veřejný přístup
- dobrý koncept zabezpečení souborů, přístupová práva, šifrování souborů



Obrázek 3.1: Struktura systému OwnCloud

3.1.3 Nevýhody

- není distribuované úložiště - problém se spolehlivostí
- obsahuje „single point of failure“

3.2 Wuala

Systém založený jako open source projekt, který později převzala firma LaCie [2]. V úvodu začal jako systém distribuovaného úložiště, které využívalo jak zdroje serverových úložišť, tak klienti mohli poskytnout místo pro datové úložiště na svých počítačích [9].

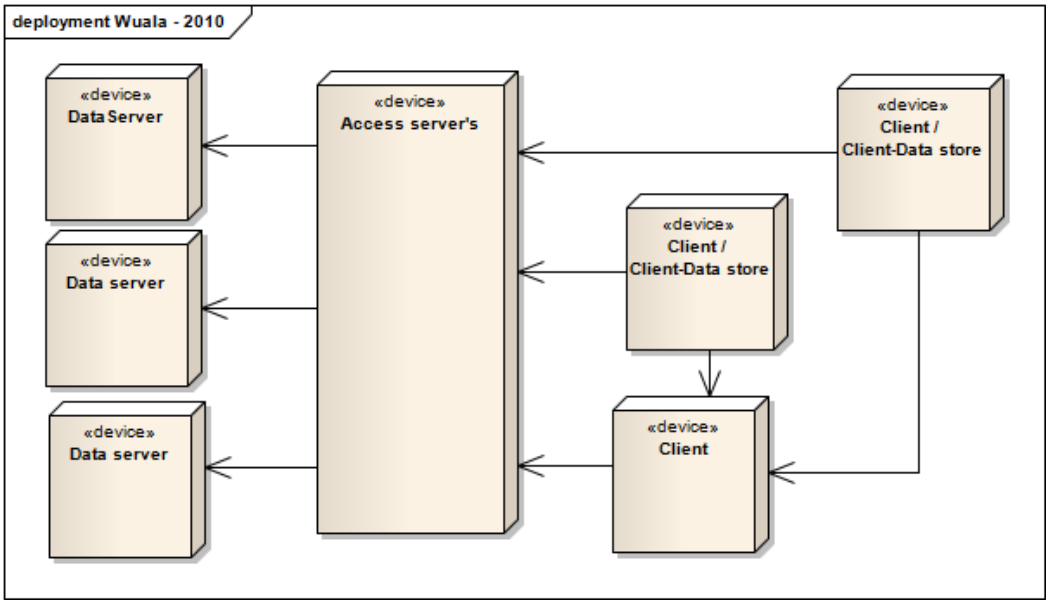
Jednalo se tedy o hybrid mezi client-server cloudovými úložišti a peer to peer úložišti 3.2. Wuala se od počátku snažila být bezpečný systém. Všechny uživatelské soubory byly kryptovány, už na straně klienta, klič k zašifrování souboru je odvozen ze souboru samotného, což pomáhá detekovat soubor, který již v úložišti existuje. Nemusí se tedy odesílat znovu a ušetří to datový prostor úložiště. Díky šifrování mohla být data uložena kdekoliv, ale přístupná byla pouze majiteli. Komunikace systému probíhá pomocí protokolu TCP v případě malých souborů. U velkých souborů je soubor rozdělen na části a odeslán pomocí UDP, což urychluje odeslání souborů mezi různé servery. Klientské datové úložiště bylo využíváno pouze pro velké soubory, kde fungovalo hlavně jako souborová cache, při stahování těchto velkých souborů.

Po převzetí projektu firmou LaCie (rok 2010), se o vývoji už mnoho neví. Wuala upouští od využívání uživatelských datových prostor a data jsou uložena pouze na serverech firmy. Přechází tedy ke konceptu klient-server, který využívají i ostatní úložiště. Komunikace se změnila, nadále se již používá pouze TCP komunikace. Data jsou uploadovány vcelku na jeden server. Při stahování se navazují až 3 současná spojení, kde každé spojení přenáší jednu část souboru, po té se uzavře. Pro další část se otevírá nové spojení. Jediné co zůstává je důraz na soukromí klienta, kde všechny soubory jsou před odesláním na server šifrovány.

Další nevýhodou je, že díky hashům souborů je možné určit, kteří uživatelé mají uložená stejná data. Dále, aby bylo možné vytvořit zamezení duplikací dat na serverech, bylo nutné šifrovat soubory podle klíče, který je vytvořen ze souboru. Jinak by každý uživatel mohl soubor zašifrovat jiným heslem a odstranění duplikací by nemohlo být provedeno. To vede ke zmenšení bezpečnosti, nebo soukromí uživatelů.

Důvod změny architektury je neznámý, možností se nabízí několik, od poklesu cen úložišť, přes velkou nespolehlivost klientských úložišť, po náročnou správy tohoto typu úložiště, kde hrozila jeho nespolehlivost.

3.2.1 Struktura systému - do roku 2010



Obrázek 3.2: Struktura systému Wuala do roku 2010

3.2.2 Výhody

- šifrování dat na straně klienta
- zajímavý koncept řešení

3.2.3 Nevýhody

- náročná správa úložiště – zajištění konzistence na velmi nespolehlivém HW
- rozdílná doba přístupu
- náročnost na výpočetní výkon na straně klienta

3.3 Google File systém

Distribuovaný file systém, který si klade za nároky spolehlivost a rychlost při obrovském množství dat a přistupujících zařízení [11]. Dále se předpokládá, že datová úložiště jsou nespolehlivé systémy. Byl navržen s ohledem na potřeby Google 3.3. Specializuje se na ukládání obrovských souborů, ve velikostech od jednotek GB, až po několika TB soubory. Proto je ukládání řešeno pomocí takzvaných chunků. Soubor je rozdělen na tyto chunky, které mají pevnou velikost. Což výborně vyhovuje potřebám pro čtení a zapisování, jelikož soubory se většinou nepřepisují. Jednou se zapíšou a pak sekvenčně čtou. Změny souborů většinou probíhají připsáním dat na konec souboru, do posledního chunku, nebo se přidají další. Každý tento chunk se pak ukládá na více datových serverů.

3.3.1 Struktura systému

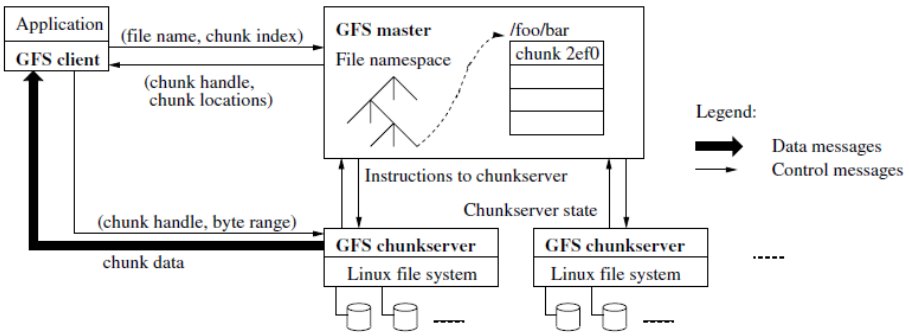


Figure 1: GFS Architecture

Obrázek 3.3: Struktura systému Google File systém

Systém se skládá z takzvaných chunk serverů (datové servery), z nichž jeden je označený jako Master. Master se stará o veškerá systémová metadata - kde je co uloženo, adresářovou strukturu, linkování mezi soubory a chunky. Master se dále stará o monitoring celého systému a úkoluje jednotlivé chunk servery. Klienti se Mastera dotazují, na metadata, lokaci kde mají hledat soubory (jednotlivé chunky), data samotná pak už stahují (zapisují) přímo z chunk serverů. Integrita dat je zajištěna pomocí kontrolních součtů, ty jsou pak uloženy na Masteru. Pokud na chunku mají data jiný kontrolní součet, nepovolí se čtení a data jsou aktualizována.

Chunky jsou veliké 64MB. Tato velikost byla zvolena kvůli minimalizování komunikace s Masterem a úspoře velikosti metadat na Masteru. Kompromis mezi malými soubory, kterých by bylo mnoho (tvořili mnoho metadat) a velkými soubory, které by se načítali dlouho.

Metadata tvoří tři skupiny: adresářový strom, propojení souborů a chunků a kde jsou chunky uloženy. První dvě skupiny jsou logovány do souboru, který se rozesílá na jiné servery, aby při pádu Mastera nebyl problém ho obnovit. Třetí skupina se při startu Mastera načítá z jednotlivých chunk serverů. Master drží metadata přímo v paměti.

3.3.2 Výhody

- vhodné pro velké množství dat
- rychlé, jednoduché
- dokáže si poradit s nespolehlivým HW
- vhodné pro velké zatížení, stovky až tisíce serverů a klientů
- podporuje adresářovou strukturu souborů na serveru

3.3.3 Nevýhody

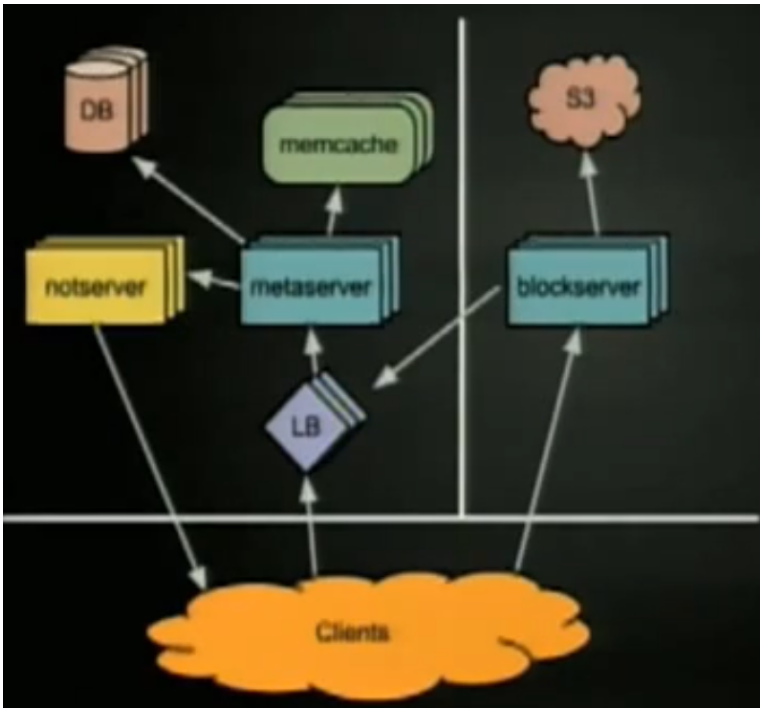
- pouze jeden Master – vzniká úzké hrdlo aplikace
- vyžaduje velmi rychlou síťovou komunikaci – velké chunky
- není zabezpečení dat
- nešifrovaná komunikace

3.4 Drop box

V dnešní době jedno z nejpožívanějších distribuovaných úložišť dat. Jde o architekturu 3.4 klient server [1]. Pracuje na principu synchronizace lokálního adresáře. Využívá toho, že nepotřebuje žádnou cache, jelikož cache zajišťuje klientská aplikace [10].

Samotný systém, nemá vlastní datové úložiště, ale využívá Amazon S3. Klientská aplikace neukládá soubor vcelku, ale po částech (chunk) o velikosti 4MB. Z těch se vytváří hash a provádí se kontrola duplicitních souborů, každý soubor je v datovém úložišti uložen jen jednou. Systém do datového úložiště posílá soubor pouze jednou na nějaký komunikační server. O jeho distribuci už se stará systém S3.

Co se týče zabezpečení, komunikace probíhá prostřednictvím zabezpečeného spojení (SSL), ale samotná data nejsou nijak šifrována. Aplikace podporuje sdílení souborů, i celých složek a umožňuje jejich verzování, uchovává předchozí verze, ne však starší než 30 dnů.



Obrázek 3.4: Struktura systému DropBox

3.4.1 Struktura systému

Databázové servery jsou klasické SQL like servery, které jsou mezi sebou aktualizovány.

Zajímavostí je, že systém udružuje po celou dobu běhu klientské aplikace aktivní spojení mezi klientem a serverem, prostřednictvím „notserverů“.

Servery jsou permanentně monitorovány. V případě výpadku některého ze serverů, především pokud jde o server řídící synchronizaci, je okamžitě nahrazen jiným serverem.

3.4.2 Výhody

- verzování souborů
- snadné sdílení souborů mezi uživateli
- podporuje adresářovou strukturu souborů na serveru
- udržuje permanentní komunikaci s klientem, může ho upozorňovat na případně změny

3.4.3 Nevýhody

- soubory nejsou šifrovány
- nemá vlastní datové úložiště

3.5 Zhodnocení systémů

Zde si shrneme jednotlivé systémy a jejich výhody. Nakonec zvážíme, co by bylo dobré z jakého systému vzít, aby se vytvořil „ideální“ systém pro ukládání dat.

- **OwnCloud** - Hlavní přednost tohoto systému vidím v možnostech ukládání a sdílení souborů. Jejich řazení do adresářů a možnost zabezpečení pomocí šifrování. Horší je to však s konceptem systému, kde se nejedná o cloudové úložiště, jelikož celý systém běží na jediném počítači. Tím vzniká single point of failure.
- **Wuala** - Tento systém má sice zajímavý koncept řešení, ovšem v podstatě nic z toho co bylo využíváno do roku 2010 bych nevyužíval. Využití klientů jako datových serverů je sice zajímavý nápad, ovšem spolehlivost těchto úložišť je natolik špatná, že bych od tohoto nápadu upustil. Datové úložiště u klienta může být velmi často nedostupné a není možné zaručit, ani odhadnout jak často bude k dispozici. Datový server v tomto případě může být častěji nepřístupný, než přístupný. Systém šifrování se mi taky nezdá nijak moc věrohodný.
- **Google File Storage** - Na tomto systému je zřetelně vidět, že byl vytvořen specificky pro potřeby googlu. Není nijak zabezpečen, ani není příliš flexibilní v oblasti operací s daty. Mazání, změny a podobně jsou v případě uživatelského přístupu běžným jevem a tento systém je příliš nepodporuje. Zajímavý nápad však vidím v dělení souborů na chunky a operace s menšími soubory.
- **DropBox** - Tento systém se mi zdá pro využití jako uživatelské datové úložiště skoro dokonalý. Velmi dobrý nápad je využití notify serveru, který umožňuje klientovi připojenému z více míst, automatické aktualizace souborového systému. Jediným závažným nedostatkem vidím nedostatečné zabezpečení.

3.5.1 Návrh „ideálního“ systému

Ideální systém bych si představoval s využitím souborového přístupu a zabezpečení, které podporuje OwnCloud. Systém adresářů a sdílení dat mezi uživately či skupinami uživatelů.

Architektury DropBoxu - využití přístupových, notify a databázových serverů. S tím, že klient bude pouze komunikovat prostřednictvím přístupových serverů a systém s ním pomocí notify serveru.

Ukládání souborů bych založil na chuncích jako má Google File Storage. I když v mém případě bych chunky vytvořil menší a s možností proměnlivé velikosti.

Myšlenku využití klientů jako datových úložišť, bych úplně nezavrhoval, ale neintegroval bych datové úložiště přímo do klienta. Umožnil bych pouze vytovřit datový server na počítači, kde běží klientská aplikace, v případě že se nám to bude hodit.

Kapitola 4

Analýza a návrh řešení

4.1 Obecné požadavky

Požadavky, které nemají vliv na funkčnost programu, avšak jsou v zadání, z důvodů kompatibility nebo z důvodů bezpečnosti aplikace.

4.1.1 Multiplatformní systém

Je nutné aby systém dokázal běžet jak na Linuxu tak Windows. To vyžaduje zvolit vhodný programovací jazyk. Pro mojí úlohu jsem zvolil C++, s využitím frameworku Qt [7]. Ostatní jazyky se nezdají tak vhodné, hlavně díky jejich náročnosti.

4.2 Funkční požadavky

4.2.1 Požadavky na systém ze strany Klienta

4.2.1.1 Dostupnost

Data musí být vždy dostupná, nesmí nastat situace, že by data byla pro výpadek nedostupná. Řešením je eliminovat úzké hrdlo, což zařídíme pomocí duplikací serverů. Tím docílíme toho, že pokud kterýkoliv server vypadne, systém poběží dál.

4.2.1.2 Spolehlivost - data se neztratí

Data se nesmí z důvodu výpadku datového serveru ztratit. Taktéž vyřešíme tím, že bude datových serverů víc a data budou uložena na několika serverech najednou.

4.2.1.3 Jednoduché ovládání

Tento požadavek v práci příliš neřeším, jde pouze o architekturu klientské aplikace, návrh GUI a přizpůsobení pro rychlou a snadnou práci. Což není hlavním předmětem této práce.

4.2.1.4 Bezpečnost dat

Data nesmí přečíst nikdo, komu nejsou určena. To znamená potřebu šifrování komunikace i samotných dat, pomocí uživatelského klíče. Šifrování dat musí probíhat již na klientském zařízení.

4.2.1.5 Automatická aktualizace/synchronizace souborů na lokálních úložištích

V případě, že je uživatel připojen pomocí více zařízení, je nutné v případě změny v datech, kterou vyvolalo jedno z těchto zařízení, upozornit ostatní zařízení na tuto změnu.

4.2.1.6 Možnost sdílení dat

Uživatel bude moci sdílet data s jednotlivými uživateli, v uživatelských skupinách, i jako veřejná data, která jsou přístupná všem uživatelům. V případě sdílení dat s jiným uživatelem, nebo ve skupině, nebudou data šifrována soukromým klíčem, ale klíčem zvoleným uživatelem. Data sdílená se všemi uživateli nebudou šifrována vůbec.

4.2.1.7 Verzování dat, krátkodobá historie dat

Každý soubor si bude uchovávat krátkodobou historii. V případě změny souboru, bude možné soubor obnovit v minulé verzi.

4.2.2 Požadavky na systém ze strany provozovatele

4.2.2.1 Minimalizace dat, co nejméně dat na úložištích

Data by měla být ukládána v co nejmenším počtu kopií. Tento požadavek nesmí jít na úkor spolehlivosti, tím myšleno replikace na několika serverech. Jde o to neukládat znovu data, která server již obsahuje, popřípadě je později odstranit a využívat pouze jednu kopii.

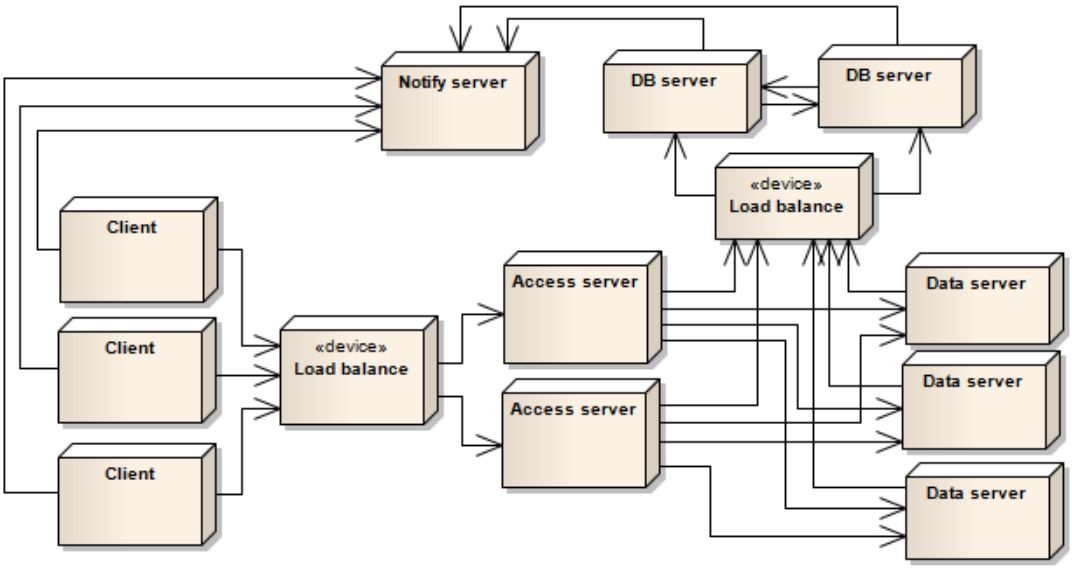
4.2.2.2 Minimalizace nároků na rychlost komunikace

Pro vyřešení požadavku je nutné minimalizovat množství dat, která jsou přenášena mezi soubory. To vyřešíme komprimací přenášených dat a vytvořením vlastního protokolu zpráv. Kupříkladu XML je pro mne nevhodné, jelikož potřebuje velké množství dat, které jsou nutné pouze pro strukturování přenášené informace.

4.2.3 Struktura systému

Jelikož navrhuji systém pro firemní použití, dá se očekávat, že většina počítačů, na kterých systém poběží, bude součástí firemní sítě, oddělená od vnější sítě firewallem a přístupovými bránami. Je tedy nutné s tím počítat a návrh tomu přizpůsobit.

V návrhu systému dále počítám s využitím load balance serverů, které budou při využití několika access nebo databázových serverů, rovnoměrně rozkládat zátěž na jednotlivé servery. 4.1



Obrázek 4.1: Struktura systému

4.3 Datový server

Jedná se o jednoduchý server, který ukládá souborové chunky do přiděleného adresáře dále je pak čte či maže. Server ke svému běhu potřebuje znát seznam svých chunků a seznam databázových serverů, kterým se má při startu hlásit. Po spuštění provede inicializaci a poté už jen poslouchá na nastavených portech, čeká na instrukce od access serveru, zda má vykonat nějakou další akci.

4.3.1 Funkce serveru

4.3.1.1 Inicializace / synchronizace

Probíhá při startu serveru. Datový server odešle zprávu databázovému serveru, oznámí že běží a je možné s ním pracovat. Součástí zprávy také bude seznam chunků, které má datový server k dispozici. Databázový server seznam překontroluje a vrátí datovému serveru informaci o tom jaké chunky má smazat, nebo doplnit.

4.3.1.2 Ulož data

Vykonává se po vyvolání access serverem. Součástí této zprávy je i datový chunk, který se má uložit. Server tento chunk, uloží a zapíše si jméno souboru do seznamu uložených chunků.

4.3.1.3 Načti data

Vykonává se po vyvolání access serverem. Součástí této zprávy je jméno chunku, který se má načíst. Server načte chunk a odešle ho zpět access serveru.

4.3.1.4 Smaž data

Vykonává se po vyvolání access serverem. Součástí této zprávy je jméno chunku, který se má smazat. Server chunk smaže a vymaže si ho i z listu uložených chunků.

4.3.1.5 Monitoring vytížení

Server trvale sleduje kolik má aktivních připojení a kapacitu úložného prostoru.

4.4 Databázový server

Databázový server obsahuje všechny potřebné informace o uživatelích, uložených datech, které systém potřebuje k běhu. Tento server bude kontaktován pouze datovými servery, při inicializaci / synchronizaci a access servery, které budou vyřizovat žádosti klientských aplikací. Bude odpovídat za ověření identity uživatele, lokalizaci dat a jejich synchronizaci. Jelikož datových serverů bude víc, bude mezi nimi probíhat synchronizace. Dále pak bude server při změně dat uživatele vyvolávat funkce notify serveru, který předá oznámení o změně všem zařízením přihlášeným daným uživatelským účtem.

4.4.1 Struktura databáze

Model struktury databáze. 4.2

4.4.2 Funkce serveru

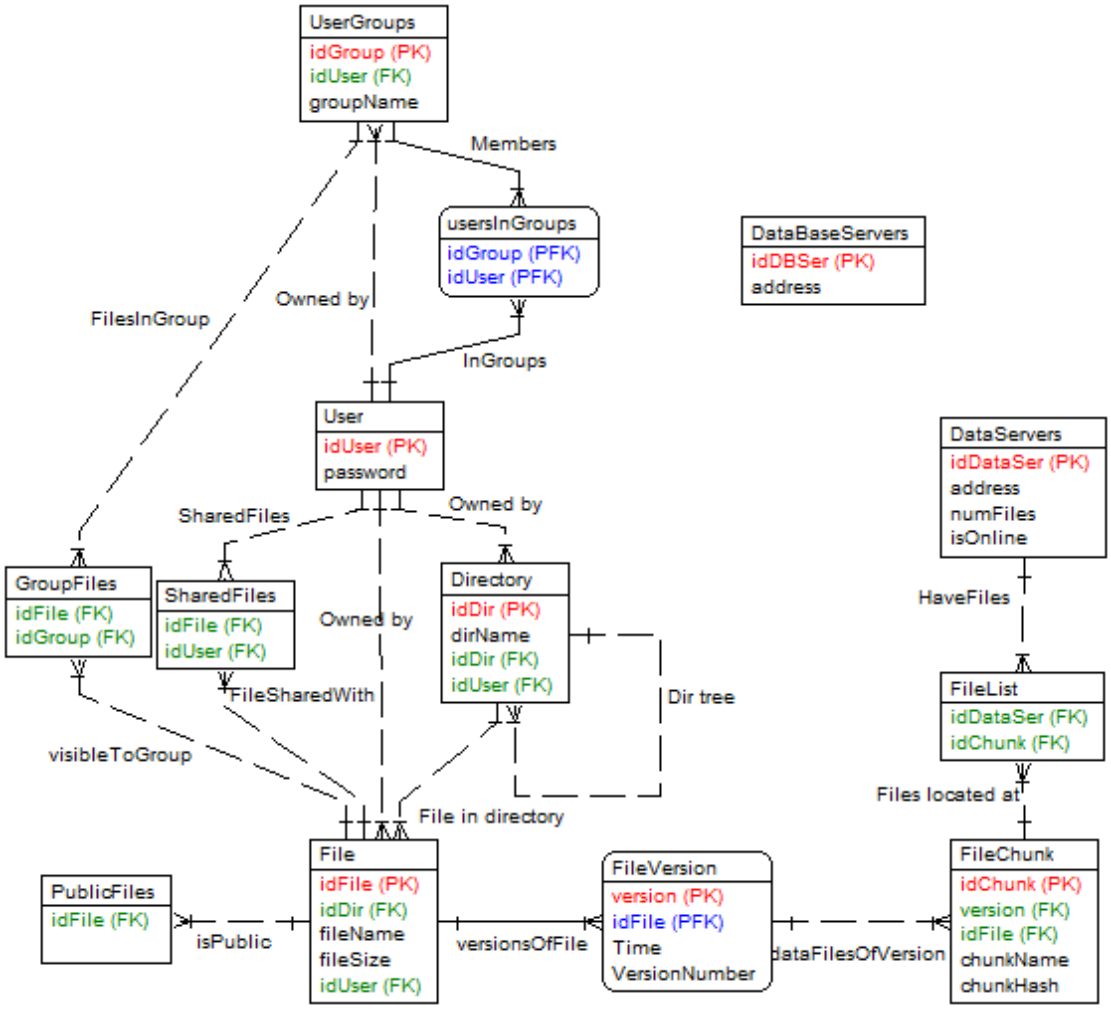
V této části si postupně popíšeme jednotlivé funkce databázového serveru, jak jsou navrženy.

4.4.2.1 Synchronizace

Je nutné vytvořit různé funkce synchronizace. První je synchronizace Datového serveru, kdy se server přihlásí, že je aktivní a databázový server zkontroluje soubory na něm uložené, popřípadě soubory, které jsou navíc, nechá smazat. Druhou je synchronizace mezi Databázovými servery. Z důvodu odstranění single point of failure, je nutné provozovat více Databázových serverů najednou a všechny servery musí obsahovat stejná data.

4.4.2.2 Monitoring

Jde o monitoring vytíženosti datových serverů. Je nutné data ukládat na nejméně vytížené servery, aby nedošlo k přetížení nebo přeplnění některých serverů, když ostatní jsou volné. Další funkcí monitoringu je vyhodnocování nepotřebných dat. Databázový server bude porovnávat hashe jednotlivých chunků. Pokud se budou shodovat, dotáže se datového serveru, zda jsou tyto chunky skutečně stejné. Pokud budou stejné, upraví si data v databázi tak, aby se dal jeden z chunků smazat. Tento chunk nechá smazat z datového serveru. Tato funkce se bude automaticky spouštět po nastavené době.



Obrázek 4.2: Model systémové databáze

4.4.2.3 Ukládání dat

- **Vytvoření souboru**
  Server byl dotázán na vytvoření nového souboru, je vytvořen záznam v datové struktuře a přiděleny servery na které se mají data uložit, seznam serverů pošle access serveru. Pokud je soubor označený pro sdílení, zaregistruje ho do příslušných struktur (sdílený se skupinou, veřejný atd.).

- **Update souboru**
  Server porovná jméno souboru s již existujícími v datovém úložišti, zda soubor již neexistuje. Pokud se shoduje jeho název s existujícím souborem, jde o update souboru. Pro daný soubor vytvoří novou verzi a vybere datové servery, kde bude nová verze uložena.

- **Vytvoření adresáře**
  Přidá virtuální adresář do tabulky. Adresáře budou všechny vycházet z kořenového adresáře home.

- **Vytvoření uživatele**
  V případě registrace uživatele je nutné vytvořit uživatele v databázi a vytvořit mu domovský adresář, který je výchozí pro uživatelova data.

4.4.2.4 Načítání dat

- **Načtení souborového stromu**
  Vrátí uživatelův seznam souborů, včetně adresářové struktury.

- **Načtení souboru**
  Soubor je identifikován svým indexem a svou verzí.

- **Načtení historie souboru**
  Jednotlivé verze souboru nejsou klasicky přenášeny v seznamu souborů uživatele, o historii bude nutno specificky zažádat.

- **Vyhledání veřejného souboru**
  Tato funkce bude hledat veřejné soubory, prohledávání bude probíhat dle jména souboru.

4.4.2.5 Mazání dat

- **Smazání souboru**
  Smazat soubor může jen vlastník souboru, vyhledání proběhne dle uživatele a indexu souboru.

- **Smazání adresáře**
  Smazat je možné jen prázdný adresář, hlavně z důvodů bezpečnosti.

- **Automatické mazání starých verzí souboru**
  Verze souboru budou udržovány jen po určitou dobu. To z důvodu kapacitních. Po uplynutí této doby budou soubory smazány.

4.5 Access server

Server, který vytváří rozhraní mezi systémem, a klientskou aplikací. Hlavní funkce tohoto serveru je urychlení distribuce souborů mezi uživatelskou aplikací a datovými servery. O tuto funkčnost by se buď musela starat strana klienta nebo databázový server. V případě distribuce na straně klienta by mohl být problém s rychlostí připojení a potřeby přímé viditelnosti na datové servery. Předpokládám totiž vysokorychlostní spojení mezi servery, několikrát rychlejší než průměrná rychlost internetové přípojky. Distribuce jiným ze serverů by zbytečně vytěžovala tyto servery a mohla způsobit blokování systému. Dalším důvodem existence tohoto serveru, je odstínění komunikace databázového serveru od klientských aplikací

(ty vůbec netuší, že tyto servery existují), server tudíž vytváří prostředí podobné demilitarizované zóně.

Nevýhodou tohoto řešení je, že access server je úzké hrdlo systému. V případě jeho výpadku, by byl systém nedostupný. To se však snadno dá řešit využitím více přístupových serverů najednou. Tím docílím, že při výpadku jednoho nebo více serverů mám stále další náhradní a systém poběží dál.

4.5.1 Funkce serveru

Nyní si popíšeme složitější funkce tohoto serveru.

4.5.1.1 Nahrání souboru

Access server dostane požadavek od klientské aplikace na uložení souboru na server. Aplikace informace o souboru předá databázovému serveru, který si soubor zařadí a vrátí access serveru na jaké datové servery má jednotlivé části souboru uložit. Ten následně otevře nový komunikační port, pro tohoto klienta, kde čeká na části souboru. Ty pak distribuuje na příslušné datové servery.

4.5.1.2 Načtení souboru

Access server dostane požadavek od klientské aplikace na načtení souboru ze serveru. Kontaktuje databázový server, který mu vrátí seznam serverů, kde by se měly nacházet jednotlivé souborové chunky. Klientovi pouze oznámí z kolika chunků se soubor skládá. Access server otevře nový komunikační port, kde čeká dokud nebude klient připraven. Na žádost stáhne jeden souborový chunk a předá ho klientovi. Toto se opakuje dokud nejsou staženy všechny části souboru.

4.5.1.3 Smazání souboru

Access server dostane požadavek od klientské aplikace na smazání souboru. Informace předá databázovému serveru. Ten vrátí adresy serverů, kde jsou jednotlivé souborové chunky. Access server odešle na všechny datové servery žádosti o odstranění daných souborových chunků.

4.5.1.4 Ostatní požadavky

Server musí reagovat i na další požadavky, jako vytvoření adresáře, přejmenování souboru atd. Tyto požadavky však nijak neobsluhuje, pouze je předá databázovému serveru, který je obslouží.

4.6 Klientská aplikace

Aplikace na straně klienta, která má za úkol simulovat vzdálený filesystém. Poskytuje uživateli adresářovou strukturu a jména souborů umístěných na vzdáleném serveru. Aplikace umožňuje stahování a nahrávání souborů na server.

4.6.1 Funkce aplikace

4.6.1.1 Nahrání souboru

Při startu aplikace se musí uživatel přihlásit. Klientská strana odešle data k ověření a stáhne uživatelův seznam souborů.

4.6.1.2 Načtení vzdáleného souborového systému

Aplikace po spuštění zažádá server o poskytnutí souborového systému daného uživatele. Po obdržení dat systém uživateli zobrazí adresářový a souborový systém.

4.6.1.3 Nahrání souboru na server

Aplikace načte uživatelem vybraný soubor, podle potřeby ho zašifruje. V případě velké velikosti souboru, bude rozdělen na menší části. Následně odešle žádost o uložení access serveru. Po vyřízení žádosti odešle soubor access serveru, který se postará o zapsání souboru na datové servery.

4.6.1.4 Načtení souboru

Aplikace dle ID souboru (zná ze seznamu souborů) zažádá access server o jeho poskytnutí, tento server vyhodnotí žádost (ověří oprávnění). Pak odešle soubor, nebo jeho části klientské aplikaci, která soubor z jednotlivých částí složí, dešifruje a uloží na požadované místo.

4.6.1.5 Vyhledání veřejného souboru

Uživatel zadá jméno souboru / uživatele, systém odešle žádost access serveru, ten žádost vyhodnotí a vrátí seznam možných souborů. Pokud uživatel bude chtít některý soubor stáhnout, pokračuje se voláním načtení souboru.

4.6.1.6 Ostatní funkce

Princip dalších funkcí je primitivní a klientská aplikace pouze instruuje access server, aby operaci provedl. Nebudu je tedy zde rozepisovat. Jedná se o:

- Vytvoření adresáře
- Smazání adresáře
- Smazání souboru
- Změna jména adresáře/souboru

4.7 Notify server

Notify server slouží k udržování spojení s klientskými aplikacemi. To je nutné z důvodu automatické synchronizace mezi serverem a klientskými aplikacemi. V případě, že uživatel je připojen pomocí dvou aplikací a pomocí jedné provede změnu v datech, server musí kontaktovat druhou aplikaci a na tuto změnu ji upozornit.

Tuto funkci by samozřejmě mohl vykonávat access server, ale jelikož je předpokládané vyšší zatížení access serverů při přenosu souborů, je lepší na tuto funkčnost vyhradit další server.

4.8 Komunikační model

[tp] Komunikačné model systému 4.3.

4.8.1 Klientská aplikace

Klientská aplikace komunikuje s access serverem a notify serverem.

Komunikace s notify serverem bude realizována pomocí SSL spojení. Z hlediska účelu, informování o nových datech uživatele, je nutné pouze říci notify serveru jaký uživatel je z tohoto počítače připojen. Dále se bude spojení udržovat po celou dobu běhu aplikace a pouze poslouchat, jestli notify server nezašle oznámení o změně, na který klientská aplikace zareaguje vyžádáním nového seznamu souborů. Před ukončením klientské aplikace, aplikace oznámí svůj konec a přeruší spojení.

Komunikace s access serverem bude realizována pomocí zabezpečeného spojení SSL a i pomocí nešifrovaného TCP. Zabezpečení je nutné pro přenos komunikace s Databázovým serverem. Jelikož jsou přenášena data o uživateli a jeho souborech. TCP spojení bude využíváno pro přenos souborů jako takových. Jelikož jsou soubory šifrovány již na straně klienta, není třeba zabezpečovat i jejich přenos. V tomto typu komunikace nebude přenášeno nic, co by bylo možné zneužít.

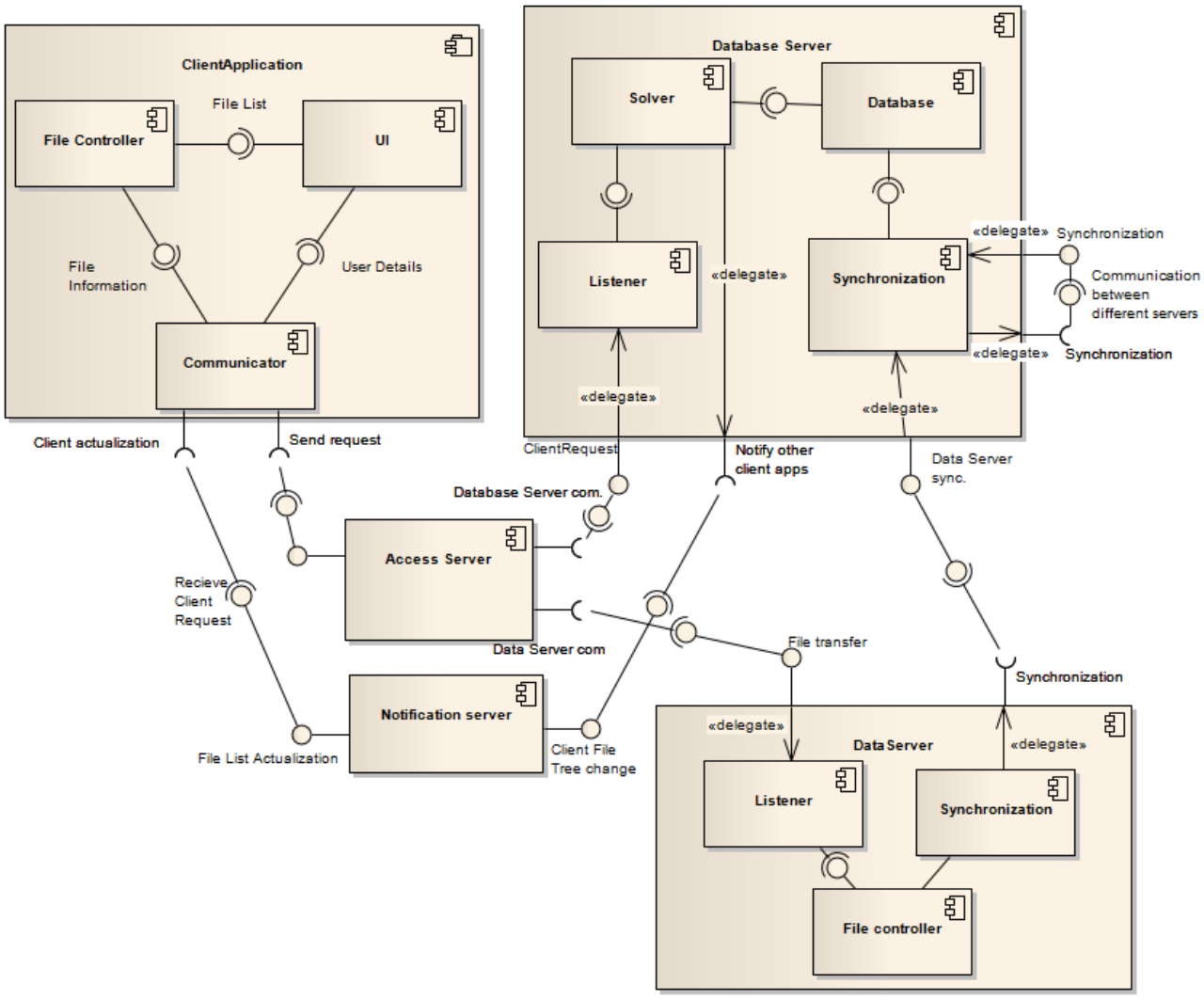
4.8.2 Access server

Access server je z hlediska komunikace nejdůležitější část systému. Vytváří rozhraní mezi klientem a všemi servery. Dále funguje jako demilitarizovaná zóna. Je tedy nutné, aby měl přímou viditelnost na všechny ostatní části aplikace. Jedinou výjimkou je notify server.

Způsoby komunikace má dva. TCP pro přenos dat a SSL pro komunikaci s databázovým serverem. Jediná data, která se předávají access serveru jsou data potřebná k přenosu souborů. Ty access serveru říkají odkud brát chunky souboru, který si klient vyžádal, nebo kam je uložit v případě, že klient chce ukládat soubor.

Struktura dat předávaných access serveru:

- návratový kód - signalizuje úspěšnost operace, popřípadě k jaké chybě došlo
- počet replikací



Obrázek 4.3: Komunikační schéma systému

- jméno chunku
- adresy datových serverů kam má být chunk uložen

  V případě, že je chunků více se předchozí dvě položky opakují podle potřeby.

Pokud byl databázový server úspěšný, klientovi bude zasláno:

- návratový kód
- adresa přístupového serveru
- port na kterém poslouchá pro přenos dat

Další nutná komunikace pro přenos nastává v případě, že je datový server nedostupný. Access server v tomto případě kontaktuje databázový server s tím, že chce vyměnit datový server.

Odeslaná zpráva:

- typ instrukce
- jméno souboru
- adresa

Odpověď:

- návratový kód
- nová adresa

4.8.3 Data server

Datový server je server pro ukládání dat. Komunikuje s access serverem a databázovým serverem. Po startu datový server odešle zprávu databázovému serveru, kterému zašle seznam souborů, které má uložené v datovém úložišti. Databázový server mu vrátí seznam instrukcí, které má vykonat, aby zaktualizoval svůj stav. Které soubory smazat, nebo si vyžádat od jiného serveru.

Dále server poslouchá na nastaveném portu (listener) a vykonává požadavky, které na něj má access server, respektive klientská aplikace komunikující prostřednictvím aceess serveru.

Poslední funkcí datového serveru je porovnání dvou souborů. Tato instrukce bude probíhat přes standardní port, na kterém datový server poslouchá.

4.8.3.1 Struktura komunikace při přenosu dat:

Příchozí zpráva:

- typ instrukce
- jméno souboru
- data - pouze při nahrávání souboru

Odpověď:

- návratový kód
- jméno souboru - pouze při načítání souboru
- data - pouze při načítání souboru

4.8.3.2 Struktura komunikace při porovnání souborů:

Příchozí zpráva:

- typ instrukce
- jméno souboru 1
- jméno souboru 2

Odpověď:

- návratový kód

4.8.4 Databázový server

Databázový server komunikuje pouze prostřednictvím zabezpečené komunikace SSL. Jelikož uchovává citlivá data, která by neměla být veřejně dostupná, je nutné komunikaci zabezpečit.

Databázový server je mozkem celé aplikace, ví o všech uživatelích a všech souborech, které jsou v systému uloženy. Komunikuje s ním klient, prostřednictvím access serveru a datové servery, které se s jeho pomocí synchronizují, či inicializují. Další formou komunikace je mezi serverová komunikace mezi databázovými servery, pomocí které si synchronizují data.

4.8.4.1 Struktura komunikace při synchronizaci

Příchozí zpráva:

- typ instrukce
- data k synchronizaci

Odpověď:

- návratový kód

4.8.4.2 Struktura komunikace s klientem

Registrace uživatele

- typ instrukce
- uživatelské jméno
- heslo

Odpověď:

- návratový kód

Žádost o seznam souborů (přihlášení)

- typ instrukce
- uživatelské jméno
- heslo

Odpověď:

- návratový kód
- adresáře -seznam
- skupiny - seznam
- soubory - seznam

Vytvoření, přejmenování, smazání adresáře

- typ instrukce
- uživatelské jméno
- heslo
- cesta ve virtuální adresářové struktuře (při přejmenování nebo smazání jde o identifikátor adresáře)
- jméno

Odpověď:

- návratový kód

Vytvoření souboru

- typ instrukce
- uživatelské jméno
- heslo
- jméno souboru
- cesta ve virtuální adresářové struktuře
- velikost souboru
- typ sdílení
- s kým sdílet (seznam uživatelů, nebo skupin se kterými chcemi soubor sdílet)
- seznam souborových částí

Odpověď: Seznam chunků a datových serverů, kam mají být jednotlivé chunky umístěny. Popsáno v kapitole Access serveru.

Stažení, smazání souboru

- typ instrukce
- uživatelské jméno
- heslo
- identifikátor souboru

Odpověď: Seznam chunků a datových serverů, kam mají být jednotlivé chunky umístěny. Popsáno v kapitole Access serveru.

4.8.5 Notify server

Notification server má jediný úkol. Udržovat spojení s klientskou aplikací a v případě změny dat na serveru, na tuto skutečnost upozornit klientskou aplikaci. O této změně je informován databázovým serverem. Server bude implementovat pouze zabezpečené spojení SSL. Komunikaci navazuje klient, který se připojí na server a dále už jen čeká. Databázový server pouze zasílá na Notify server informace o uživatelích.

Interface spojení s databázovým serverm

- Uživatelské jméno

Interface spojení s klientem - vstup:

- Uživatelské jméno

výstup:

- Zpráva o změně souborů na serveru

Kapitola 5

Realizace

V rámci této diplomové práce, byl realizován pouze pilotní implementace tohoto navrženého distribuovaného datového úložistě. Nejsou tedy hotovy všechny části návrhu, pouze funkčnost nutná k základní funkci datového úložiště. To jest spolehlivé a bezpečné ukládání souborů. Realizováno je vytváření adresářové struktury, ukládání, stahování a mazání souborů. Dále pak funkce synchronizací jednotlivých serverů.

5.1 Vývojové a aplikační prostředí

Prvním důležitým úkolem bylo vybrat vývojové a provozní prostředí aplikace. Jelikož má jít o multiplatformní aplikaci, nebylo možné použít jakýkoliv programovací jazyk, nebo jinou komponentu. Nakonec jsem zvolil programovací jazyk C++, s využitím knihoven Qt. Knihovna Qt [7] je OpenSource projekt, který velmi usnadňuje práci v C++, při zachování rychlosti kterou C++ umožňuje. Další výhodou, tohoto řešení je, že se zaměřuje na všechny nejpoužívanější operační systémy, jak pro PC, tak pro mobilní zařízení. Jedinou nevýhodou je, že má velmi okrajově řešené funkce pro zabezpečení. Je tedy nutné pro šifrování a zabezpečenou komunikaci využívat jinou knihovnu.

K tomu jsem zvolil OpenSSL. Taktéž jde o OpenSource projekt [6], který má dlouhou historii a je v praxi velmi používaný. Sice je primárně vyvíjen a používán na operačních systémech linuxového typu, ale zachovává standart C++ STL a je možné ho použít i v systémech Windows a dalších.

Poslední důležitá část systému je databáze. Zde jsem využil PostgreSQL [8], s kterým jsem již měl pozitivní zkušenosti. Jde o OpenSource projekt, který funguje na všech rozšířených operačních systémech.

5.2 Komunikace

Jak již bylo v návrhu určeno, bylo třeba vytvořit dvě komunikační cesty SSL a klasické TCP. TCP komunikace je tvořena za pomocí klasické STL C++. SSL komunikace je řešena za použití knihoven OpenSSL [5].

Ač využívám TCP a SSL, které je postavené nad TCP komunikací, je lepší ověřovat, zda data při přenosu dorazila celá. Pokud dojde k chybě již při komunikaci, nemá smysl na daný požadavek reagovat, jelikož by došlo k chybě později. Z toho důvodu při přenosu dat postupuji tak, že při přenosu první 4 byty reprezentují délku dat v bytech, následně odešlu zprávu samotnou.

Dále bylo nutné zvolit strukturu komunikace. Zde se nabízelo například XML. Jednoduché řešení, na které existuje mnoho různých parserů a je kompatibilní se všemi systémy. Má však velkou nevýhodu a tou je velká datová zátěž. Jelikož XML vyžaduje velké množství dat, které jsou nutné jen k přenosu živých dat. Zvolil jsem proto přenos pomocí jednoduchého protokolu, který data bere jako binární tok.

Princip přenosu je jednoduchý, uvažuji 3 možné druhy dat.

- **byte** - vhodné na přenos signálů, chybových hlášek a ostatních nastavení
- **integer** - přenos číslic. Je rozložen na 4 byty, pomocí funkce **intToChar**. Následně při přijmu změněn zpět na číslo funkcí **charToInt**.
- **datové pole** - je reprezentováno svou délkou (integer) a daty samotnými. Použito pro přenos dat a textových řetězců

Zde by mohl vzniknout problém v jiném uspořádání dat na různých operačních systémech, little/big endian, hlavně v případě integeru. Proto jsem vytvořil vlastní funkce pro převod integeru **intToChar** a **charToInt**. Tyto funkce převádí integer na 4 byty a zpět, způsobem, který zajistí stejný výsledek na jakémkoliv operčním systému. Tím zajistím kompatibilitu mezi různými systémy. V případě přenosu seznamu, například seznam souborových chunků, přenáším seznam pomocí počtu jeho entit. Tento způsob komunikace, generuje minimum dat využívaných jen k přenosu a následnému parsování, díky tomu je efektivnější.

5.3 Datový server

Realizace datového serveru zahrnuje všechny funkce, kromě funkce monitoringu. V této kapitole si postupně popíšeme jednotlivé funkce, jak byly vytvořeny.

5.3.0.1 Inicializace / synchronizace

Při startování serveru aplikace načte konfigurační soubor a soubor se seznamem chunků na serveru uložených, pokud existuje. Zkusí se spojit s databázovým serverem. Pokud spojení není úspěšné aplikace končí. Databázový server vrátí chunky, které má datový server smazat, jelikož jsou navíc, měly být smazány v době kdy byl datový server vypnut. Situace, že by nějaký chunk chyběl není prakticky možná, jelikož při nahrávání chunků na jednotlivé datové servery je tato možnost ošetřena.

Po té, co je provedena údržba, datový server začne poslouchat na stanoveném portu.

Komunikace s přístupovými servery není stálá, je řešena pomocí žádostí. Jakmile je žádost vyřízena, je spojení ukončeno. Každý příchozí požadavek vytvoří vlastni thread, který bude požadavek obsluhovat. Server na počátku má nastavený thread pool, o velikosti kterou si uživatel nastavil. Pokud by bylo požadavků najednou více, než je kapacita thread

poolu, thread pool nezvětšuji, ale vytvořím nový thread ručně. Tento thread se po vyřízení požadavku zruší.

5.3.0.2 Ulož data

Po obdržení zprávy od access serveru, systém podle prvního znaku ve zprávě rozpozná že jde o požadavek na uložení chunku. Ze zprávy dále extrahuje jméno chunku a data. Chunk následně uloží do adresáře nastaveného v konfiguračním souboru. Dále si jméno zapíše do souboru fileList.txt, kde si udržuje seznam všech chunků uložených na serveru. Vrátí přístupovému serveru návratový kód, zda se uložení podařilo, a tím končí.

5.3.0.3 Načti data

Po obdržení zprávy od access serveru, systém podle prvního znaku ve zprávě rozpozná že jde o požadavek na načtení chunku. Zpráva dále obshuje jméno chunku. Systém se pokusí načíst chunk. Pokud je to možné vytvoří odpověď ve formě: návratový kód, jméno chunku, data. V případě, že data nejsou přítomna vrátí chybu.

5.3.0.4 Smaž data

Při příchodu zprávy, která požaduje smazání chunku, systém chunk smaže, následně jméno vyhledá ve fileList.txt odkud ho také odstraní. Přístupovému serveru vrátí návratový kód o úspěšnosti operace.

5.4 Databázový server

Databázový server je mozek celé aplikace. V podstatě se dá říci, že řídí chod všech ostatních serverů. V této kapitole si postupně podrobně popíšeme vlastnosti tohoto serveru.

Jelikož komunikace celého systému, funguje na principu zpráv, na které databázový server zareaguje a ukončí spojení. Neexistuje žádná paměť, který uživatel komunikuje z jakého počítače. Proto všem uživatelským operacím (ukládání, načítání a mazání dat) předchází ověření přístupu a s tím spojené ověření oprávnění vykonat danou operaci. Proto je součástí každé zprávy uživatelské jméno a hash hesla. Ten se ověří oproti databázi. Podle uživatelského ID u jednotlivých funkcí ověřuji jeho oprávnění vykonat danou operaci.

5.4.1 Databáze

Pro ukládání dat serveru je použita databáze PostgreSQL 9.2. Jde o objekto-relační databázi, která má všechny vlastnosti ACID. Její velkou výhodou je, že funguje pod všemi operačními systémy od Linuxu, po Mac OS. Jde o stabilní databázi, která je stavěná i pro velké objemy dat.

S databází se komunikuje prostřednictvím modulu, který nabízí Qt. Připojení k databázi probíhá po přijetí klientského požadavku. To je nutno z důvodu, že modul není threadově bezpečný. Navíc by mohly být problémy při využití transakcí, které jsou nutné při vkládání souborů, či jejich mazání, aby byla zabezpečena konzistence databáze. Parametry pro připojení k databázi, bere aplikace z konfiguračního souboru *config.cfg*.

5.4.2 Popis funkcí serveru

5.4.2.1 Synchronizace mezi databázovými servery

Tato funkce je asi nejdůležitější funkcí databázového serveru, v případě využití více databázových serverů.

Tato funkce je aktivována pokaždé, když byla v databázi provedena nějaká změna, přidání či odebrání záznamů. Program se podívá do databázové tabulky **DatabaseServers**. V této tabulce jsou uvedeny adresy všech ostatních databázových serverů, které jsou součástí celého systému. Vezme adresy všech serverů a rozešle instrukce v podobě SQL příkazů na všechny adresy. Jednotlivé servery zprávu zpracují a přidají si ji do svých databází. Následně potvrdí, zda se podařilo aplikovat příkazy nebo ne.

Z důvodu mezi serverové synchronizace, nebylo možné používat číselné identifikátory jednotlivých položek v databázi. Generování identifikátorů popíši později u jednotlivých případů. Číselné identifikátory jsou použité pouze ve dvou případech a to u tabulky Databázových serverů a Datových serverů. V případě Databázových serverů, není problém, jelikož jednotlivé záznamy do ní přidává ručně administrátor. U datových serverů, jsou sice záznamy přidávány automaticky, ale nový server se nepřidává příliš často a nebezpečí z možného přidání dvou serverů naráz je velmi malé. Tím však získám velikou úsporu ve velikosti záznamů v tabulce *FileList*, kde jsou identifikátory fileChunků a Datových serverů. Tudíž jsem se rozhodl toto nebezpečí postoupit s tím, že při iniciálním spuštění se musí Datové servery spouštět postupně.

5.4.2.2 Inicializace / Synchronizace datových serverů

Tato synchronizace probíhá při každém spuštění datového serveru. Datový server zašle po svém startu svojí adresu a seznam souborů, pokud nějaké má. Databázový server si ze zprávy vyčte adresu datového serveru. Prověří databázi, zda takový server existuje.

Pokud v databázi není záznam o serveru s touto adresou, vytvoří nový záznam a vrátí potvrzení.

Pokud záznam nalezne, vyhledá podle ID záznamu, seznam souborů, které má datový server obsahovat. Porovná seznam z databáze se seznamem, který zaslal datový server. V případě, že nějaký soubor na datovém serveru přebývá, zařadí ho databázový server do seznamu souborů ke smazání. Situace, že by server měl obsahovat soubor, který tam není, by neměla bez vnějšího zásahu nastat. Jelikož je při ukládání souborů kontrola, zda se uložení zdařilo. V případě že nikoliv, je soubor nahrán na jiný server a změněn záznam v databázi.

Seznam souborů k smazání zašle po zpracování všech požadavků zpět datovému serveru. Nakonec si nastaví příznak, že je daný datový server aktivní.

5.4.2.3 Monitoring

Funkce monitoringu nejsou v realizaci dokončeny. Realizován je pouze jednoduchý aparát na rovnoměrné vytížení serverů. Funkce funguje na principu počítání množství chunků na jednotlivých serverech. Při ukládání nového souboru vybírám aktivní servery, které mají nejmenší počet uložených souborů.

5.4.2.4 Ukládání dat

- **Vytvoření uživatele**
  Tato operace se vykoná místo ověření uživatele. Jako první systém zkontroluje zda neexistuje uživatel s tímto uživatelským jménem. Jelikož uživatelské jméno je pro systém unikátní identifikátor daného uživatele. Pokud neexistuje, vytvoří systém záznam o uživateli a vytvoří mu jeho domovský adresář.

- **Vytvoření adresáře**
  Funkce přidá virtuální adresář do tabulky. Jak již bylo řečeno dříve, je nutné vytvořit unikátní identifikátor adresáře. Ten je tvořen uživatelským jménem, jménem tohoto adresáře a hashem rodičovského adresáře. Z tohoto řetězce je vytvořen nový hash, který funguje jako unikátní identifikátor. Z tohoto důvodu platí omezení, že nesmí být v jednom adresáři více podadresářů se stejným jménem, jelikož vytvořené hashe by byly stejné.

- **Vytvoření souboru**
  Tato funkcionalita je v aplikaci implementována s jistými omezeními. Není implementováno verzování souborů a jejich sdílení. Z pohledu databázového serveru by však neměl být problém tuto funkci doplnit. Jde pouze o přidání dalších záznamů do databáze na příslušná místa. Bylo by však nutné upravit i další části systému a z časových důvodů jsem to již nestihl dokončit.

  Jako první se otestuje, zda se soubor se stejným jménem již nenachází v daném adresáři. Potom by šlo o novou verzi. Systém zkusí vyhledat soubor podle jména v tomto adresáři. Pokud je záznam nalezen, vrátí se chyba, aby soubor nebyl přepsán. V druhém případě jde o nový soubor a začne se ukládat do databáze.

  Nejdříve se vygeneruje ID souboru. To se skládá ze jména souboru, hashe rodičovského adresáře a uživatelského jména. Z těchto dat se vygeneruje hash a ten se použije jako ID. Dále se pokračuje s verzí souboru. Jelikož jde o nový soubor, verze je vždy jedna. ID verze je hash čísla verze a ID souboru.

  Pak vytvořím záznamy jednotlivých chunků. Pro každý chunk vybírám zvlášť datové servery, kam má být chunk uložen. Počet serverů se řídí nastavením v konfiguračním souboru. Datové servery, které jsou uvažovány při výběru, musí být v daný moment označené jako aktivní. Pokud server nemá dostatečný počet dostupných datových serverů, které jsou potřebné pro uložení souboru, ohlásí chybu a operaci neprovede. Lepší je, když systém neprovede akci, než aby ji provedl špatně, nebo nedostatečně.

  Během procesu ukládání záznamů jednotlivých chunků aplikace sestavuje směrovací tabulku pro access server, tu mu po úspěšném dokončení odešle.

- **Update souboru**
  Verzovaní souborů a s ním spojený update souboru, není implementován.

5.4.2.5 Načítání dat

- **Načtení souborového stromu**
  Tato funkce funguje jako funkce ověření přihlášení uživatele a načtení souborového

stromu zároveň. Po ověřeni uživatelského oprávnění systém postupně prochází adresáře, soubory a skupiny uživatele. Do odpovědi pro klienta zapisuje:

  - **typDat** - d adresář, f - soubor, g - skupina
  - **id**
  - **jméno**

- **Načtení souboru**
  Načtení souboru spočívá ve vyhledání všech chunků, které patří k souboru a serverů na kterých jsou jednotlivé chunky uloženy. Následně sestaví seznam chunků s adresami datových serverů, kde jsou jednotlivé chunky uloženy. Tento seznam následně zašle zpět access serveru. Načtení vybrané verze není podporováno.

- **Načtení historie souboru**
  Tato funkce také, stejně jako zbytek podpory verzování, není implementována.

- **Vyhledání veřejného souboru**
  Sdílení není v této práci realizováno.

5.4.2.6 Mazání dat

- **Smazání souboru**
  Aplikace po přijetí požadavku ověří, zda uživatel je vlastník daného souboru, tím jestli má právo tento soubor vymazat. Pokud ano, vyhledá všechny souborové chunky a servery na kterých jsou uloženy. Vytvoří seznam pro access server, podle kterého je access server nechá vymazat. Poté odstraní daný soubor z databáze. V případě úspěšného vymazání odešle seznam access serveru, aby nechal soubory vymazat z jednotlivých datových serverů.

- **Smazání adresáře**
  Po obdržení požadavku aplikace zkontroluje, jestli je adresář prázdný, pokud ano vymaže ho.

- **Automatické mazání starých verzí souboru**
  Tato funkce není v rámci této práce realizována.

5.5 Access server

Access server funguje jako přístupová brána klientských aplikací do systému. Jeho další úlohou je distribuce souborů mezi jednotlivé datové servery.

Server při startu načte konfigurační soubor a vytvoří listener zabezpečeného spojení SSLv3. Zde čeká na připojení klientských aplikací. Po připojení jsou jednotlivé požadavky obslouženy vždy ve vlastním threadu. Server stejně jako ostatní servery má thread-pool nastavitelné velikosti. V případě, že je připojených více klientů, než je velikost thread-poolu, jsou pro každého klienta vytvářeny vlastní thready, které jsou po dokončení požadavku ukončeny.

Po přijetí požadavku server deleguje tento požadavek na databázový server. Jediné co ho na každé žádosti zajímá, je typ instrukce. Pokud se totiž jedná o nějakou operaci se soubory, tj. o nahrávání, načtení nebo smazání souboru, vyřizuje server následně komunikaci s datovými servery. Je tedy nutné tyto žádosti rozenat. Rozpoznání probíhá pomocí prvního bytu každé žádosti (typ instrukce). Jednotlivé komunikace u těchto instrukcí jsou popsány níže.

V případě, že jde o jinou instrukci přeposlě odpověď databázového serveru zpět klientovi a končí spojení.

5.5.1 Nahrání souboru

Po přijetí této žádosti, access server čeká na odpověď od databázového serveru. V odpovědi obdrží seznam souborových chunků, které budou nahrány na datové servery. Ke každému chunku je připojen seznam adres datových serverů, kam má být odeslán. Tento seznam si zpracuje a vytvoří nový listener kde očekává přenos daného souboru. Po vytvoření listeneru, klientovi navrátí svou IP adresu a port kde naslouchá. Pak už jen čeká na zprávy od klienta na daném portu.

Po přijetí zprávy, access server zprávu přečte, zjistí zda jde skutečně o zprávu nahrání chunku a jméno chunku, který je přenášen. Podle jména vyhledá daný chunk v seznamu a zjistí na jaké servery má být odeslán. Pak chunk rozesílá na jednotlivé datové servery. V případě, že je některý ze serverů nedostupný, nebo se nepodaří na něj data uložit, access server odešle požadavek o výměnu datového serveru na databázový server. Po obdržení nového serveru se znovu pokusí chunk uložit. Takto opakuje dokud se to nepodaří. V případě, že neobdrží nový server, pokusí se uložit chunk na ostatní servery kam měl být původně uložen. V případě, že se nepodaří chunk uložit na žádný server, ukončí spojení s tím, že ukládání selhalo. S tím ukončí veškerou komunikaci ukládání tohoto souboru, jelikož jedna jeho část se nepodařila uložit.

Tento postup opakuje, dokud neodešle všechny chunky, nebo klient neukončí spojení.

5.5.2 Načtení souboru

Při tomto požadavku se postupuje podobně jako při nahrávání souboru. Rozdíl je ve zprávě klientovi, tomu se navíc zasílá i počet chunků, ze kterých se daný soubor skládá.

Dále se klient dotazuje na zaslaný listener, pokaždé access server zažádá o jeden chunk ze seznamu. Pokud soubor od datového serveru nebodrží, ptá se dokud daný chunk postupně na všech serverech, které dostal v seznamu. Pokdu soubor nedostane ani od jednotho serveru, oznámí chybu a končí komunikaci.

Tímto způsobem se klient dotáže na všechny chunky. Po odeslání všech souborů server uzavře listener a končí komunikaci s klientem.

5.5.3 Smazání souboru

V tomto případě access server v odpovědi od databázového serveru obdrží jak operace dopadla a seznam chunků společně s adresami serverů kde jsou umístěny. Rozdíl zde je v tom,

že access server v případě kladné odpovědi odešle výsledek klientovi ihned. Tím s ním končí komunikaci.

Následně se pro každý chunk, pokusí kontaktovat všechny servery na kterých je uložen a zašle jim požadavek na smazání daného chunku. Úspěšnost této operace není nijak kontrolována. K selhání může dojít pouze ze dvou důvodů. První důvod je, že je datový server vypnutý. Což nám nevadí, jelikož soubor bude smazán při startu tohoto datového serveru. Druhým důvodem je, že soubor na daném serveru není, což je velmi nepravděpobné, ale také nám to v tomto případě nevadí.

5.6 Klientská aplikace

Tato část systému poskytuje rozhraní mezi uživatelem a systémem. Umožňuje uživateli se přihlásit do systému, ukládat soubory a následně je načítat. Nejduležitější činností klientské aplikace je zabezpečení souborů před přečtením třetí osobou, což je realizováno pomocí šifrování.

5.6.1 Komunikace s uživatelem

Klientskou aplikaci bylo možné realizovat dvěmi způsoby.

Prvním bylo integrovat ji do prostředí operačního systému, aby se tvářila jako virtuální disk. Uživatel by pak se systémem komunikoval pomocí standardních prostředků, které poskytuje operační systém ke správě souborů. Tím by se však omezila funkčnost na specifický operační systém. Další výhody tohoto řešení, jako kupříkladu možnost startu aplikace při startu operačního systému, by byly vykoupenu nutností zadat přihlašovací údaje do konfiguračního souboru.

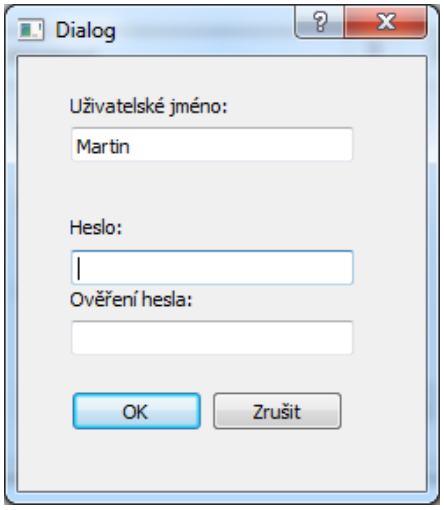
Tento způsob se mi nezdál příliš lákavý, proto jsem se rozhodl pro druhou variantu a tou je vytvoření vlastního GUI. Další výhodou je, že v tomto případě bylo možné i klientskou aplikaci naprogramovat jako multiplatformní aplikaci.

V případě potřeby je navíc možné tuto aplikaci snadněji integrovat do prostředí nějakého operačního systému.

5.6.2 Registrace uživatel

Registraci uživatele je možné vyvolat při startu systému, před přihlášením uživatele. Uživatel se registruje zadáním uživatelského jména a hesla 5.1. Heslo není možné později měnit. Po potvrzení systém zkontroluje, zda bylo heslo zadáno správně, dvakrat stejné heslo. Po kontrole, odešle požadavek na přístupový server.

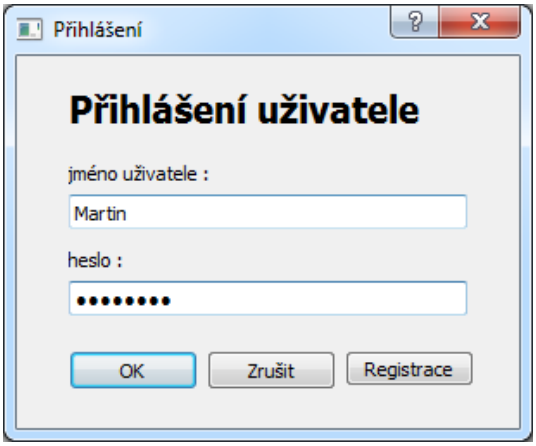
Pokud registrace proběhne, je umožněno uživateli se přihlásit. Pokud už uživatel existuje, objeví se příslušná hláška.



Obrázek 5.1: Registrační formulář klientské aplikace

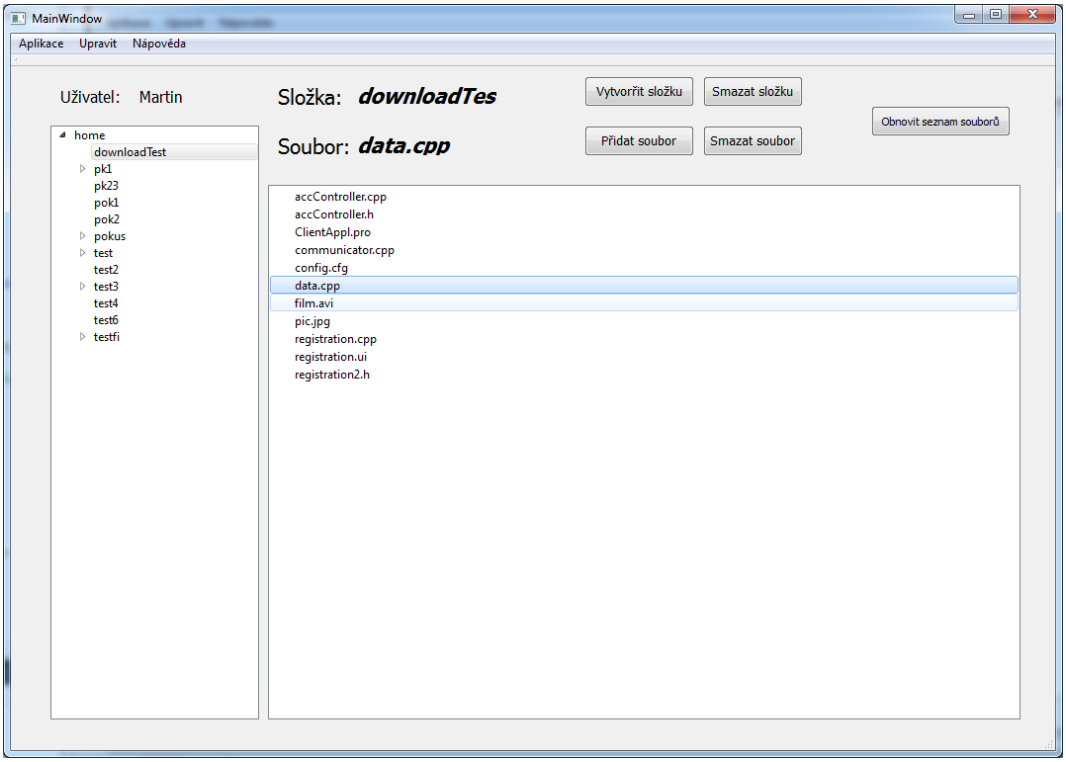
5.6.3 Přihlášení uživatel - načtení seznamu souborů

Při startu systému se uživateli zobrazí přihlašovací okno 5.2 a uživatel se musí přihlásit svým uživatelským jménem a heslem.



Obrázek 5.2: Přihlašovací okno aplikace

Pokud uživatelské jméno nesouhlasí, uživatel bude o této skutečnosti informován chybovou hláškou. Jinak se mu po úspěšném přihlášení zobrazí dialog hlavního okna. Při úspěšném přihlášení aplikace obdrži od databázového serveru, seznam souborů, adresářů a skupin. Soubory si aplikace uloží do interního seznamu. Vygeneruje strom adrešářů a zobrazí je v hlavním okně 5.3. Na skupiny aplikace nereaguje. Soubory se zobrazí při vybrání adresáře, do kterého soubory patří. Struktura hlavního okna je zobrazena na obrázku, adresářový strom okno nalevo, seznam souborů pravé okno.



Obrázek 5.3: Hlavní okno aplikace

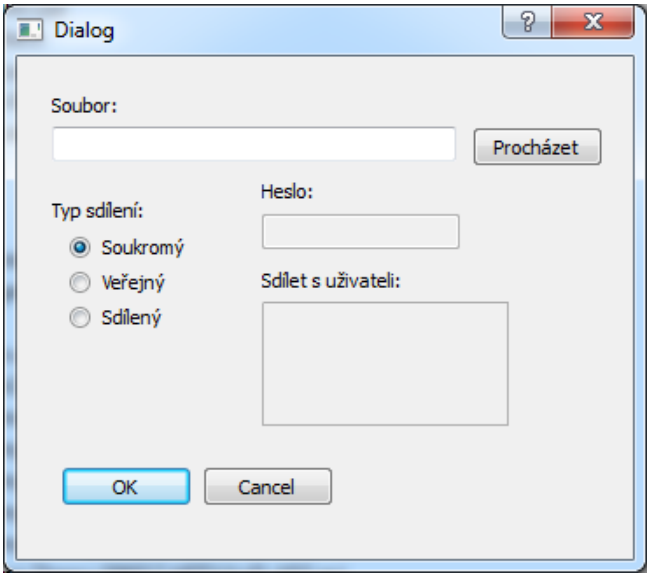
5.6.4 Nahrání souboru do systému

Pro vyvolání musí uživatel vybrat adresář ve virtuální adresářové struktuře systému a vyvolat dialog načtení souboru. Zde vybere soubor, který chce nahrát do systému. Ostatní nastavení v dialogu, jsou nefunkční, jelikož se týkají sdílení souborů, které není implementováno.

Po potvrzení začne aplikace načítat zadaný soubor. Načítání velkých souborů probíhá po 4MB částech, menší než 4MB jsou načteny najednou. Jednotlivé části (chunky) jsou po načtení zašifrovány, pomocí šifry AES. Šifrovací klíč je odvozen z hesla uživatele, vytvořením hashe MD5. MD5 je sice slabý hash, který se dá prolomit, ale tento hash není nikam odesílán ani ukládán, proto pro mé potřeby dostatečně vyhovuje. Dále se pro každý chunk vytvoří jeho hash. Nakonec chunky uloží do „cache“ adresáře. Jméno každého chunku je vytvořeno tímto klíčem.

- hash - kódovaný pomocí BASE64
- datum a čas uložení
- uživatelské jméno
- pořadí chunku

Po úspěšném zpracování souboru, je odeslána žádost o uložení souboru na databázový server. Klientská aplikace obdrží odpověď, o úspěšném či neúspěšném provedení. V případě



Obrázek 5.4: Dialog načtení souboru

úspěšného provední je součástí odpovědi adresa a port, kam má aplikace zasílat jednotlivé chunky. Chunky jsou odesílány po jednom. Po úspěšeném odeslání všech částí, klienstká aplikace tuto skutečnost oznámí uživateli.

5.6.5 Načtení souboru

Tuto funkci vyvolává uživatel vybráním souboru a přikazem k jeho stažení. Aplikace dle ID souboru (zná dle seznamu souborů) zažádá databázový server o jeho poskytnutí, tento server vyhodnotí žádost (ověří oprávnění). V případě úspěšné žádosti aplikace obdrží zpět adresu a port na přístupový server, kam se má připojit na datový přenos a počet částí souboru, které má očekávat.

Pak se připojuje na přístupový server a pokaždé obdrží jednu část souboru. Tyto části si aplikace ukládá do cache adresáře. Po dokončení stahování začne části dešifrovat a skládat do výsledného souboru. Pořadí souborů se řídí posledním číslem v jméně chunků.

5.6.6 Vyhledání veřejného souboru

Tato funkčnost není v rámci této práce realizována.

5.7 Notify server

Tento server není nezbytně nutný k fungování aplikace. Jeho účel je pouze k zlepšení uživatelského komfortu. Jelikož realizace je pouze o pilotní implementace, nebyl tento server realizován.

Realizace tohoto serveru by neměla být problém pro kohokoliv kdo by v této práci pokračoval. Většina modulů potřebných k fungování tohoto serveru byla použita v některém ze serverů. Pro vytvoření je pouze potřebné tyto moduly poskládat dohromady a upravit klientskou aplikaci a databázový server, aby s tímto serverem komunikovali a využívali jeho funkce.

Kapitola 6

Testování

Testování probíhalo převážně manuálním způsobem, simulováním různých situací. Jelikož celý systém je několik aplikací, které spolu komunikují a data ukládaná na databázový server musí být unikátní, není možné efektivně testovat systém pomocí automatických testů. Na tomto systému bylo nutné testovat několik faktorů. Stabilitu, spolehlivost, bezpečnost, rychlost a multiplatformnost.

V této kapitole si postupně popíšeme průběhy testů, jakým způsobem byly realizovány a jejich výsledky.

6.1 Stabilita, spolehlivost

Testy probíhaly na dvou počítačích. Při testu jsem využíval jednu klientskou aplikaci a jeden access server. Access server nemělo smysl duplikovat, jelikož při pádu se ztratí všechny probíhající přenosy a klientské aplikace zahlásí u daných operací chybu. Je tedy nutné vyvolat operaci znovu. Ta pak poběží pomocí jiného access serveru, ale nehrozí žádná ztráta dat. O přesměrování na jiný access server se postará load balance server, který není součástí této práce, není tedy důvod toto testovat.

Dále se využívalo dvou databázových a datových serverů. Při testu byl nastaven počet replikací dat nahrávaných do systému na 1 replikaci. Databázové servery, byly synchronizovány a byla nastavena jejich synchronizace.

Při testování stažení souboru byly soubory na, kterých byl test proveden, nahrány předem s nastavením počtu replikací na 2. Test pak probíhal vypnutím jednoho serveru, na kterém byly uloženy data. Pomocí debuggeru jsem se podíval, který server je první na řadě pro získání dat a ten jsem vypnul.

Testování spočívalo ve vypínání jednotlivých serverů, za běhu nejlépe během přenosu souboru.

- **Databázový server** Na databázových serverech fungovala synchronizace správně. Při vypnutí jednoho serveru a ručním přesměrování access serveru na druhý server, systém běžel dál. V praxi se o přesměrování bude starat load balance server. Jediný rozdíl byl v rychlosti provedení operací, jelikož při operacích vyžadujících synchronizaci se čekalo na time-out synchronizace. Tato doba se podstatně zkracuje, pokud pokud je počítač s danou adresou úplně nedostupný, ne pouze vypnutý server.

- **Datový server** Tento test probíhal při přenosu souboru. Jak uploadování, tak stahování souboru. Zde se v případě výpadku serveru, prodloužila doba nahrání souborů, rovněž díky timeoutům. Při pokusu o uložení dat na datový server, ukládání selhalo a datový server byl vyměněn za jiný. Rychlostně ovlivněn byl pouze přenos, při kterém server poprvé selhal, další komunikace běžela původní rychlostí.

  Protože test probíhal pouze s jednou replikací, z důvodu nedostatku počítačů, byl soubor který byl ukládán po vypnutí serveru nedostupný. Několik části bylo uloženo na serveru, který byl vypnut. Tento problém však bude v praxi řešit více replikací jednotlivých chunků. Nemělo by k němu tedy dojít.

  Při testu načítání souborů, byly soubory obsaženy na obou datových serverech. Po zjištění nedostupnosti jednoho serveru, byla data získána z druhého serveru.

- **Access server** Při vypnutí tohoto serveru během komunikace, daná operace selhala a klientská aplikace ohlásila chybu. Operaci bylo nutné opakovat.

6.2 Testy bezpečnosti

Testy bezpečnosti byly provedeny sledováním komunikace. K tomu jsem použil program WireShark, kde jsem si odchytil celou komunikaci, mezi klintskou aplikací, access serverem a databázovým serverem. Komunikace byla šifrována pomocí SSL, nebylo tedy možné získat z ní žádné informace o jejím obsahu.

Soubory byly taktéž cryptovány a nebyly nalezeny žádné shody mezi originálním souborem a jednotlivými chunky. Při komunikaci pro přenos souborů, bylo možné zjistit jaký soubor se přenáší (jeho jméno vygenerované aplikací) a jeho šifrovaný obsah. Ten však případnému útočníkovi, pokud nezná uživatelovo heslo nijak nepomůže.

6.3 Testy multiplatformnosti

Všechny části systému byly během testů přeloženy a spuštěny na systémech Windows XP SP3, Windows 7 a Linux Ubuntu 12.10.

Při tomto testu bylo testováno zda systém dokáže fungovat když jsou jednotlivé části spuštěné na odlišných systémech. Kupříkladu Klientská aplikace, běžící pod linuxem a ostatní servery pod Windows. Postupně byly otestovány všechny možné kombinace. Pouze databázový server vždy běžel na systému Windows, jelikož systém Linux neměl během testů funkční databázi.

Během testu se neprojevily žádné problémy, ani zpoždění.

6.4 Rychlostní testy

Rychlostní testy mé aplikace jsem dělal doma na domácí síti. K testování jsem použil 4 počítače. Jeden pro klientskou aplikaci, druhý na databázový server. Na posledních dvou běžely zárověň datový a databázový server. Při testu se data ukládaly ve dvou replikacích.

Upload souboru				
Soubor	Velikost (MB)	Šifrování (s)	Celkový čas přenosu (s)	Celková rychlost (kB/s)
1	176,6	24	52	3385,8
2	186,4	25	53	3177,5
3	170,3	23,8	51	3338,7
4	1515,6	210	446	3398,3
Download souboru				
Soubor	Velikost (MB)	Stahování (s)	Celkový čas přenosu (s)	Celková rychlost (kB/s)
1	176,6	37	43	4094,5
2	186,4	38,5	44	3827,5
3	170,3	34	40	4256,8
4	1515,6	307	356	4257,4
Přímé kopírování: klientský počítač -> datový server				
Soubor	Velikost (MB)	Stahování (s)	Celkový čas přenosu (s)	Celková rychlost (kB/s)
1	176,6	-	19	9266,5
2	186,4	-	20	9320,5
3	170,3	-	18	9459,5
4	1515,6	-	156	9715,6

Tabulka 6.1: Měření rychlostí systému

Použité počítače:

- AMD Athlon 7750 2,70GHz, 4GB RAM, disk ST3250824AS, Windows XP - SP3 (Klientská aplikace)
- Intel Core I3 530 2,93GHz, 2GB RAM, disk ST3250318AS, Windows XP - SP3 (Access server)
- Notebook Toshiba Tecra A10 - Intel Core I3 M330 2,13GHz, 4GB RAM, disk HTS545032B9A300, Windows 7 (Databázový a Datový server)
- Intel Core I7 960 3,20GHz, 12GB RAM, disk WD1002FAEX, Windows 7 - 64b (Databázový a Datový server)

Klient byl propojen se zbytkem počítačů sítí o rychlosti 100Mb/s, ostatní servery mezi sebou komunikovaly rychlostí 1Gb/s. Tím jsem co nejlépe simuloval podmínky reálného použití. Jelikož předpoklad je pomalejší připojení klienta, než je rychlost komunikace mezi servery.

V tabulce naměřených rychlostí systému 6.1 jsem měřil několik veličin. Šifrování reprezentuje čas potřebný k šifrování a vytvoření chunků souboru. Celkový čas přenosu je součet času šifrování a času přenosu dat. Celková rychlost je počítána z velikosti souboru a celkového času přenosu.

Rychlosti při přímém kopírování jednotlivých souborů, byly okolo 9500 kB/s. Což je daleko vyšší rychlost, než při použití mého systému. Na druhou stranu můj systém data

Jméno systému	**Přenosová rychlost (kbit/s)**
DropBox	3076
Wuala	1453
Windows Live Mash	2103
Apple iCloud	3204
Ubuntu One	2405
Systém Petra Tučka	9823

Tabulka 6.2: Tabulka rychlostí ostatních systémů

ukládal na dva počítače a navíc je šifroval. Dále šifrování zabíralo cca polovinu času, což bylo dáno především rychlostí pevného disku na klientském počítači.

Pro další porovnání jsem zahrnul výsledky testů 6.2 z práce Petra Tučka [13]. Jelikož on testoval na kolejní síti, která je připojena k internetu přípojkou o rychlosti 100Mb/s. V případě mého testu by byla ostatní řešení silně znevýhodněna rychlostí internetu, ta je mnohem pomalejší než rychlost lokální sítě.

Při porovnání je patrné, že si můj systém nestojí špatně ani v případě placené konkurence. Rychlostně je ve většině případů lepší nebo alespoň stejně rychlý.

Kapitola 7

Závěr

V této práci jsem se zaměřil na několik cílů. Prozkoumat současná dostupná datová úložistě a jejich vhodnost pro firemní použití. Následně ze získaných informací navrhnout a vytvořit pilotní implementaci distribuovaného datového úložiště.

Současná datová úložiště jsou především zaměřena na využití běžným uživatelem, ne na firemní užití. Mají mnoho užitečných funkcí, které využije každý, ale chybí jim klíčové funkce, které ocení firmní zákazník. Skoro všechny systémy umožňují sdílet data mezi uživateli, poskytují vysokou spolehlivost a rychlost při přenosu dat. Nedostatky mají především v zabezpečení dat, jelikož na to uživatel z řad veřejnosti neklade tak velké nároky.

V této práci jsem navrhl bezpečné datové úložistě, které v návrhu zahrnuje i většinu z funkcí, které nabízejí ostatní datová úložiště. Můj systém však oproti ostatním nabízí i dobré zabezpečení dat. Data jsou chráněna jak při přenosu, tak při uskladnění v systému pomocí šifrování.

V rámci implementace, byl realizován pilotní projekt. Ten umožňuje uživateli základní práci se soubory a adresáři v rámci systému. Nabízí dobrou spolehlivost a vysokou bezpečnost dat, při zachování vysoké rychlosti přenosu dat. Jde však stále o pilotní implementaci, která nemá všechny funkce které by byly potřebné pro využití v praxi. Chybí možnosti sdílení, verzování a další funkce. Ty nejsou sice bezpodmínečně nutné, ale velmi zpříjemní užívání systému. Doplnit tuto funkčnost není z principu těžké, jelikož pilotní implementace s touto funkčností počítá, je to pouze časově náročné a nebylo již v mých silách je dokončit v rámci této práce.

Tím se dostáváme k možnostem dalšího rozšíření systému. Možností je mnoho, jako první se nabízí doplnit již zmiňované funkce. Pokusit se o vyřešení problému při výpadku jednotlivých serverů, kdy systém čeká na time-out, čímž se zpomalí. Další možností je vytvoření klientských aplikací, které budou přímo integrovatelné do file systémů jednotlivých operačních systémů, nebo pro mobilní přístoroje (smart phone, tablety, atd..).

Literatura

[1] DropBox - Kevin Modzekewski *How we have scaled DropBox* . 10/9/2012 . [cit. 29. 12. 2013]
<http://www.quora.com/Dropbox/What-is-Dropboxs-architecture >

[2] Wuala [cit. 29. 12. 2013]
<www.wuala.com >

[3] OwnCloud [cit. 29. 12. 2013]
<www.owncloud.org >

[4] OwnCloud - Frank Karlitschek *The technology behind ownCloud* . 13/12/2011 . [cit. 29. 12. 2013]
<https://owncloud.com/blog/the-technology-behind-owncloud-2 >

[5] Varun Gupta *Secure Server Client using OpenSSL in C.* 16/8/2010 . [cit. 29. 12. 2013]
<http://simplestcodings.blogspot.cz/2010/08/secure-server-client-using-openssl-in-c.html >

[6] OpenSSL [cit. 29. 12. 2013]
<http://www.openssl.org/ >

[7] Qt Project [cit. 29. 12. 2013]
<http://qt-project.org/ >

[8] PostgreSQL [cit. 29. 12. 2013]
<http://www.postgresql.org/about/ >

[9] Thomas Mager, Ernst Biersack and Pietro Michairdi *A Measurement study of the Wuala On-line Storage Service* . 2012

[10] Idilio Drago, Marco Mellia, Maurizio M. Munafò, Anna Sperotto, Ramin Sadre, Aiko Pras *Inside Dropbox: Understanding Personal Cloud Storage Services* . 2012

[11] Sanjay Ghemawat, Howard Gobioff, Shun-Tak Leung, Google *The Google File system* . 2003

[12] Gurudatt Kulkarni, RaniWaghmare, Rajnikant Palwe, Vidya Waykule, Hemant Bankar, Kundlik Koli *Cloud Storage Architecture* . 2003

[13] Bc. Petr Tuček *Distribuované úložiště dat* . 2/1/2012

Příloha A

Seznam použitých zkratek

PC Personal computer

HW Hardware

TCP Transmission Control Protocol

UDP User Datagram Protocol

GFS Google File Storage

MB, GB, TB Megabyte, Gigabyte, Terabyte

SSL Secure Socket Layer

SQL Structured Query Language

DB Database

GUI Graphical user interface

XML Extensible markup language

STL Standard Template Library

ACID Atomicity, Consistency, Isolation, Durability

ID Unikátní identifikátor

IP Internet Protocol

SP3 Service Pack 3

Příloha B

Instalační a uživatelská příručka

Systém byl vytvořen a testován na operačních systémech Windows 7 a Linux Ubuntu 12.04. Využité knihovny:

- Qt SDK 4.8.4 (4.8.5)
- Open SSL 1.0.1e

Ve Windows byl systém překládán pomocí překladače Microsoft Visual Studio 2010.

B.1 Instalace v systému Windows

Instalace prostředí ve Windows spočívá v nakopírování přiložených .dll knihoven do adresáře aplikace společně s .exe souborem a config.cfg. Jen na počítači kde poběží databázový server, je nutné nainstalovat databázový stroj PostrgeSQL 9.2, nebo vyšší a nastavit dle instrukcí níže.

B.2 Instalace v systému Linux

Instalace v prostředí linux vyžaduje nainstalování Qt libraries 4.8.* . Ty jsou dostupné na adrese: http://qt-project.org/downloads . V Ubuntu stačí instalace pomocí balíčkového systému, balíček Qt SDK.

Dále je nutné nainstalovat OpenSSL verze: 1.0.1e. Možné stáhnout na adrese: http://www.openssl.org/source/ . Instalace probíhá provedením standardních příkazů (./config , make , make install).

Na počítači kde poběží databázový server je nutné nainstalovat PostreSQL 9.2, nebo vyšší a nastavit dle instrukc níže.

Součástí práce není přeložená verze aplikace, je tudíž nutné vytvořit funkční verzi ze zdrojových souborů jejím přeložením pomocí překladače gcc.

Pro přeložení je na CD **makefile** ke každé části systému. Adrsáře obsahující makefile a adresáře se zdrojovými soubory je nutné nakopírovat do jednoho adresáře. Makefile musí být v jiném adresáři než jsou zdrojové soubory. Pak v adresáři obsahujícím makefile zavoláme příkaz **make**.

B.3 Nastavení jednotlivých částí systému

Při nastavování konfiguračních souborů musí mít všechny servery jednoho typu (access, data ...) nastavený stejný port na, kterém budou poslouchat.

B.3.1 Klientská aplikace

Nastavení systému probíhá vyplněním souboru "config.cfg". Struktura souboru se nesmí měnit.

B.3.1.1 config.cfg

```
access server address:
192.168.100.1
access server port:
10120
cache path:
./CACHE/
```

B.3.2 Access server

Nastavení systému probíhá vyplněním souboru "config.cfg". Struktura souboru se nesmí měnit.

Server dále vyžaduje vytvoření serverového certifikátu. Ten musí být uložen v adresáři kde je aplikace umístěna. Jméno certifikátu musí být: "mycert.pem"a klíče "mycert.key".

B.3.2.1 config.cfg

```
Adress of this server:  - adresa musí ukazovat přímo na tento
192.168.100.1             počítač, ne na load balance server
Access server port:
10120
Number of threads in pool:
4
DataBase server adr: - adresa databázového serveru,
192.168.100.1          nebo load balance serveru databázových serverů
DataBase server port:
10100
Data server port:
13533
```

B.3.3 Datový server server

Nastavení systému probíhá vyplněním souboru "config.cfg". Struktura souboru se nesmí měnit.

B.3.3.1 config.cfg

```
data server address:
192.168.100.1
database server address:  - adresa databázového serveru,
192.168.100.1               nebo load balance serveru databázových serverů
Storage directory path:
./DATA/
Number of threads in pool:
4
Database server port:
10100
Listening port:
13533
```

B.3.4 Databázový server server

Narozdíl od ostatních serverů, tento server ke svému běhu potřebuje přístup k databázi. Je tedy nutné nainstalovat databázi PosgreSQL verze 9.2 nebo vyšší.

V databovém stroji je nutné vytvořit databázi a uživatele, který v ní bude mít oprávnění pro čtení, vkládání, update a mazání záznamů. V dané databázi je nutné provést skript "dbInit.sql", který je uložen na CD.

Pokud bude užíváno více databázových serverů, je nutné aby do tabulky "DataBaseServers"byly přidány záznamy, kde ve sloupci "address"je umístěna adresa ostatních serverů. Pokud používáme jeden server, nic nevyplňujeme.

Přístupové údaje vyplníme společně s ostatními informacemi do souboru "config.cfg". Pokud užíváme více Databázových serverů, je nutné aby všechny servery měly nastaven stejný port na kterém budou poslouchat.

Server vyžaduje vytvoření serverového certifikátu. Ten musí být uložen v adresáři kde je aplikace umístěna. Jméno certifikátu musí být: "mycert.pem"a klíče "mycert.key".

B.3.4.1 config.cfg

```
Database machine host name:
localhost
Database name:
cloud
Database user:
postgres
Database password:
pass
Number of threads in pool:
4
Number of data replications:
1
```

```
Listening port:
10100
```

B.4 Provozní příručka

B.4.1 Spuštění systému

Jednotlivé servery se musí spouštět v sekvenci Databázové servery, Datové servery, Access servery. Při prvním spuštění datových serverů, je nutné spouštět servery postupně, jinak by mohlo dojít k chybě.

B.4.2 Použití

Při prvním spuštění je nutné se zaregistrovat. Uživatel zadá uživatelské jméno a heslo. Heslo není možné později změnit.

Po úspěšné registraci se uživatel pomocí uživatelského jména a hesla přihlašuje do systému. Po přihlášení se objeví v levé části obrazovky strukturu adresářů, v pravé soubory v aktivním adresáři.

Iniciálně je uživateli vytvořen adresář home, uživatel nemůže vytvářet adresáře, nebo soubory mimo adresář home, nebo jeho podadresáře.

Příloha C

Obsah přiloženého CD

```
|   readme.txt
|
+---bin
|   +---Linux  // obsahuje makefile pro prelozeni v prostredi linux
|   |   +---AccessServerRun
|   |   |       config.cfg
|   |   |       Makefile
|   |   |       mycert.key
|   |   |       mycert.pem
|   |   |
|   |   +---ClientApplRun
|   |   |       config.cfg
|   |   |       Makefile
|   |   |
|   |   +---DataServerRun
|   |   |       config.cfg
|   |   |       Makefile
|   |   |
|   |   +---DBServerRun
|   |   |       config.cfg
|   |   |       Makefile
|   |   |       mycert.key
|   |   |       mycert.pem
|   |   |
|   |   \---DB_script
|   |           dbInit.sql
|   |
|   \---Win  // obsahuje binarní soubory spustitelné v prostředí Windows
|       +---AccessServer
|       |       AccessServer.exe
|       |       config.cfg
|       |       mycert.key
```

```
|       |       mycert.pem
|       |
|       +---ClientAppl
|       |       ClientAppl.exe
|       |       config.cfg
|       |
|       +---DataServer
|       |       config.cfg
|       |       dataServer.exe
|       |
|       +---DBServer
|       |       config.cfg
|       |       DataBaseServer.exe
|       |       mycert.key
|       |       mycert.pem
|       |
|       +---DB_script
|       |       dbInit.sql
|       |
|       +---dlls    // knihovny potřebné ke spuštění aplikací
|       |       libeay32.dll
|       |       libssl32.dll
|       |       msvcp100d.dll
|       |       msvcr100d.dll
|       |       QtCored4.dll
|       |       QtGuid4.dll
|       |       QtSqld4.dll
|       |
|       \---install  // instalační soubor databázového stroje
|               postgresql-9.3.1-1-windows.exe
|
+---source  // zdrojové soubory jednotlivých aplikací
|   +---AccessServer
|   |       AccessServer.pro
|   |       clientssl.cpp
|   |       clientssl.h
|   |       config.cfg
|   |       connectData.cpp
|   |       connectData.h
|   |       connectionaccept.cpp
|   |       connectionaccept.h
|   |       data.cpp
|   |       data.h
|   |       filetransfer.cpp
|   |       filetransfer.h
|   |       main.cpp
```

```
|   |       main.h
|   |       mycert.key
|   |       mycert.pem
|   |       serverssl.cpp
|   |       serverssl.h
|   |
|   +---ClientAppl
|   |       cfiletransport.cpp
|   |       cfiletransport.h
|   |       ClientAppl.pro
|   |       clientssl.cpp
|   |       clientssl.h
|   |       communicator.cpp
|   |       communicator.h
|   |       config.cfg
|   |       connectData.cpp
|   |       connectData.h
|   |       createdir.cpp
|   |       createdir.h
|   |       createdir.ui
|   |       createfile.cpp
|   |       createfile.h
|   |       createfile.ui
|   |       data.cpp
|   |       data.h
|   |       defs.h
|   |       infdial.cpp
|   |       infdial.h
|   |       infdial.ui
|   |       login.cpp
|   |       login.h
|   |       login.ui
|   |       main.cpp
|   |       mainwindow.cpp
|   |       mainwindow.h
|   |       mainwindow.ui
|   |       registration.cpp
|   |       registration.h
|   |       registration.ui
|   |       uiLoadFile.ui
|   |
|   +---DataServer
|   |       clientssl.cpp
|   |       clientssl.h
|   |       config.cfg
|   |       dataServer.pro
```

```
|   |       fileController.cpp
|   |       fileController.h
|   |       main.cpp
|   |       main.h
|   |
|   +---DBModel
|   |       cloud.dm2
|   |       dbInit.sql
|   |       dbModel.png
|   |
|   \---DBServer
|           clientssl.cpp
|           clientssl.h
|           config.cfg
|           connectionaccept.cpp
|           connectionaccept.h
|           DataBaseServer.pro
|           main.cpp
|           main.h
|           mycert.key
|           mycert.pem
|           serverssl.cpp
|           serverssl.h
|
\---text  // textová část práce
    |   prohlaseni.pdf
    |   volfmar5_thesis2013.pdf
    |   zadani.pdf
    |
    \---source  // zdrojové soubory textové části práce
        |   k336_thesis_macros.sty
        |   volfmar5_thesis2013.tex
        |
        \---figures // obrázky obsažené v textové části práce
                Communication.png
                dbModel.png
                DropBox.png
                funkcni_pozadavky.png
                GFS.png
                login.png
                LogoCVUT.eps
                LogoCVUT.pdf
                mainWindow.png
                obecne_pozadavky.png
                OwnCloud.png
                registration.png
```

strukturaAplikace.png
upload.png
use_case.png
use_case2.png
wuala_2010.png