VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

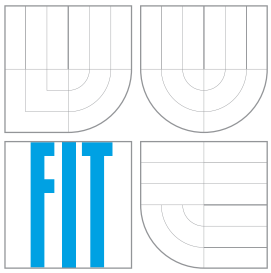
INTERAKTIVNÍ SDÍLENÍ DOKUMENTŮ

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE VIKTOR JÓBA
AUTHOR

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

INTERAKTIVNÍ SDÍLENÍ DOKUMENTŮ

INTERACTIVE DOCUMENT SHARING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

VIKTOR JÓBA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JOZEF MLÍCH

BRNO 2014

Abstrakt

Tato práce se zabývá sdílením a editací textových dokumentů v reálném čase přes počítačovou síť. Jednotný obsah dokumentu u všech členů sdílení je zajištěn pomocí algoritmu operačních transformací. Tato metoda je v práci více přiblížena. Výsledkem této práce je jednoduchý textový editor, který komunikuje na počítačové síti pomocí rozšířeného protokolu XMPP. V tomto editoru jeden uživatel vytvoří sezení sdílení dokumentu, do kterého pozve jednotlivé členy, kde následně mohou společnými silami tvořit formátovaný textový dokument, který je možné uložit ve formátu HTML, popřípadě exportovat do formátu PDF.

Abstract

The thesis deals with the real-time sharing and editing of text documents in the network. The synchronization of the document content for all users in the shared session is ensured by the algorithm of operational transformations. This method is described in the thesis in detail. The outcome of the thesis is a simple text editor that communicates over the network using the extended version of the XMPP protocol. One of the users creates a document sharing session in the editor and invites other users to cooperate on the rich text document that can be saved in HTML or exported into PDF.

Klíčová slova

Qt, operačné transformácie, xmpp, jabber, počítačová sieť, zdieľanie, reálny čas, úprava textu

Keywords

Qt, operation transformation, xmpp, jabber, computer network, sharing, real time, text editing

Citace

Viktor Jóba: Interaktivní sdílení dokumentů, bakalářská práce, Brno, FIT VUT v Brně, 2014

Interaktivní sdílení dokumentů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Jozefa Mlícha.

.....
Viktor Jóba
18. května 2014

Poděkování

Ďakujem panu Jozefovi Mlíchovi za vedenie pri tejto práci.

Obsah

1 Úvod **2**

2 Sieťové protokoly **4**
2.1 Sieťová komunikácia 4
2.2 Vlastný sieťový protokol 6
2.3 Protokol XMPP 6
2.4 Posielanie správ pomocou protokolu XMPP 7
2.5 Vlastné rozšírenia protokolu XMPP 8

3 Operačné transformácie **13**
3.1 Úvod do operačných transformácií 13
3.2 Atribúty jednotlivých zmien tvorených užívateľmi 14
3.3 Príklad a vysvetlenie jednoduchého skladania textu 14
3.4 Ďalšie možnosti operačných tranformácií 16

4 Implementácia **18**
4.1 Podrobnosti o sieťovej komunikácie 18
4.2 Podrobnosti o použitých operačných transformáciách 19
4.3 Diagram tried 21
4.4 Užívateľské rozhranie 22
4.5 Testovanie 23
4.6 Budúcnosť aplikácie 23

5 Záver **24**

A Obsah CD **26**

Kapitola 1

Úvod

V dnešnej dobe sa informačné technológie presadzujú v každej oblasti života. Tieto technológie sa používajú aj pre komunikáciu na diaľku. V prípade veľkých vzdialeností sa tým šetrí čas, prostriedky a strácajú sa negatívne vplyvy cestovania. V niektorých prípadoch ani nie je možné komunikovať osobne.

Pri nenáročnej komunikácii stačí e-mail alebo chat. Oproti osobnej komunikácií má tento spôsob komunikácie veľké množstvo nevýhod, ako príklad môžem uviesť chýbajúcu neverbálnu komunikáciu, alebo rýchlosť komunikácie oproti hovorenému slovu. Našťastie tento problém riešia video konferencie, ktorých úlohou je maximálne napodobniť komunikáciu užívateľov v jednej miestnosti.

Medzi úkoly, ktoré môžu užívatelia robiť v spoločnej miestnosti, a doteraz spomenuté technológie tieto úkoly nepodporujú, je napríklad zdieľanie predmetov. Aby sme sa priblížili o krok ďalej ku komunikácií cez internet maximálne napodobujúcej komunikáciu užívateľov v jednej miestnosti, môžme napríklad napodobiť zdieľanie papiera. Na papier sa dá písať a kresliť, avšak jeho nevýhoda je, že na ňom môže v jeden čas robiť úpravu málo ľudí, okrem prípadu, že je daný papier dostatočne veľký.

Cieľom tejto práce je navrhnúť a vytvoriť aplikáciu, ktorá umožní napodobniť práve spomínaný papier, avšak len písomnú formu, bez kreslenia. Oproti reálnemu papieru má však aj výhody. Na jednom dokumente môže spolupracovať väčšie množstvo užívateľov, dokonca aj na rovnakom mieste v texte, čo u fyzického papiera nie je možné, alebo aspoň pohodlné.

Zdieľanie dokumentov v reálnom čase, spolupráca na ich vytváraní a upravovaní nie je novinka. Podpora kolaborácie na dokumentoch je používaná napríklad v aplikáciach `google docs`, `gobby` alebo `coWord`. Hlavný dôvod, prečo som sa rozhodol vytvoriť aplikáciu, ktorá má mnoho konkurentov, je odstránenie závislosti na webových technológiách, prístup k dokumentu len vybraným užívateľom a možnosť formátovať text.

Kapitola 2 sa zaoberá bližším popisom sieťových komunikačných protokolom, ktoré sa dajú využiť pri zdieľaní dokumentov cez počítačovú sieť. Ďalej odôvodňuje, prečo sa rozhodlo v prospech protokolu XMPP, zhŕňa požiadavky na sieťový protokol pre editor zdieľaných dokumentov. Bližšie rozoberá protokol XMPP a rozšírenia, ktoré boli nevyhnutné pre podporu editora.

Kapitola 3 sa zaoberá prenosom a začleňovaním zmien v dokumente medzi užívateľmi. K tomuto účelu sa používajú takzvané operačné transformácie[4]. Princíp je popísaný na jednoduchom príklade, na základe algoritmu implementovaného v editore, ktorý je výsledkom tejto práce. Ďalej sa zaoberá hlavnými problémami, ktoré môžu nastať pri súčasnom editovaní dokumentu viacerými užívateľmi.

Kapitola 4 sa zaoberá detailami, ktoré boli využité pri implementácii editora. K operačným transformácia dopĺňa spôsob, akým je možné formátovať text a predávanie si správ medzi nimi. Pri sieťovej komunikácii rozoberá implementáciu sieťovej časti aplikácie na jednoduchom diagrame. Ďalej popisuje triedy a ich vzájomné vzťahy. Venuje sa aj popisu užívateľského rozhrania a testov operačných transformácií. Na záver popisuje budúce smerovanie aplikácie.

Kapitola 2

Sieťové protokoly

V tejto kapitole popisujem sieťové komunikačné protokoly využiteľné pri interaktívnom zdieľaní dokumentov.

V prvej časti tejto kapitoly popisujem model TCP/IP a vysvetľujem úlohu jednotlivých vrstiev modelu TCP/IP [8].

V podkapitole 2.2 popisujem možnosti vlastného neštandardného sieťového protokolu. Popisujem jednotlivé typy správ, ktoré takýto protokol musí implementovať. Na záver podkapitoly rozoberám nevýhody takéhoto prístupu.

V podkapitole 2.3 popisujem protokol XMPP [9]. Zaoberám sa identifikáciu jednotlivých užívateľov tohoto protokolu popisujem ich vzájomnú komunikáciu.

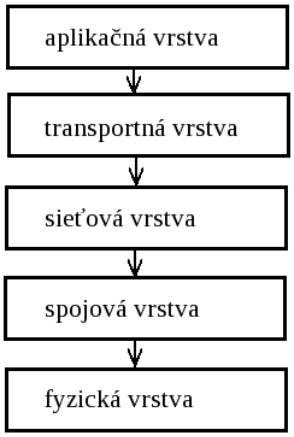
Ďalšie podrobnosti o komunikácií sú vysvetlené na príklade v podkapitole 2.4. Ďalej názorne ukazujem ukazujem spôsob, akým je možné jednoznačne identifikovať nie len užívateľa, ale aj klienta podporujúceho protokol XMPP, pre ktorý je správa určená.

Na záver kapitoly popisujem vlastné rozšírenia protokolu XMPP. Popisujem jednotlivé správy, ktoré je nutné implementovať. Ku každej správe prikladám názornú ukážku.

ďalej by sa dalo čerpať z [5], [2] a [1].

2.1 Sieťová komunikácia

Model sieťovej komunikácie[8] je pre jej zložitosť rozdelený do vrstiev. Každá vrstva využíva služby nižšej vrstvy. Komunikácia medzi rovnakými vrstvami na rôznych zariadeniach je riadená komunikačným protokolom. Architektúra je navrhnutá tak, aby bolo možné zmeniť komunikačný protokol na jednej vrstve bez dopadu na ostatné vrstvy. Model TCP/IP má 5 vrstiev, od najnižšej po najvyššiu sú to fyzická, spojová, sieťová, transportná, aplikačná vrstva. Na obrázku 2.1 sú zobrazené vrstvy a naznačený spôsob použitia vrstiev ostatnými vrstvami.



Obrázek 2.1: Zobrazenie vrstiev v modeli TCP/IP, prekreslené podľa [8].

Fyzická vrstva slúži k samotnému prenosu dát po fyzickom médiu, ako sú káble metalické či optické, v prípade rádiových vĺn vzduch. Komunikáciu riadi protokol navrhnutý špeciálne na dané médium a snaží sa odstrániť jeho nedokonalosti napríklad vhodným kódovaním, overovaním chýb a ich opravou, pokiaľ je to možné.

Spojová vrstva poskytuje spojenie medzi dvoma a viacerými systémami zapojenými v tej istej linke. Táto linka môže byť dvojbodová alebo mnohobodová. Radí jednotlivé rámce, stará sa o nastavenie parametru prenosu linky, detekuje neopraviteľné chyby.Formátuje fyzické rámce a priraďuje im fyzickú adresu. Na tejto vrstve fungujú zariadenia ako sú prepínače (switch), mosty(bridge).

Sieťová vrstva sa stará o smerovanie a adresovanie v sieti. Poskytuje spojenie medzi systémami, ktoré spolu priamo nesusedia. Najznámejší protokol pracujúci na tejto vrstve je IP. Data sú smerované cez smerovače (router). Na ceste od odosielateľa k príjemcovi sa môže vyskytnúť rada smerovačov. Smerovač slúži na prepojenie jednotlivých sietí do siete väčšej. Každé zariadenie v sieti vrátane smerovača má IP adresu, najčastejšie verzie 4 (IPv4), menej často verzie 6 (IPv6).

Transportná vrstva je implementovaná až v koncových zariadeniach. Poskytuje transportné služby spoľahlivým spojením TCP alebo nekontrolovaným nespoľahlivým spojením UDP. Výhoda protokolu UDP spočíva v rýchlosti prenášania dát, jeho nevýhoda je možnosť straty dát, preto sa využíva často na úlohy, kde strata dát neobmedzuje používanie služieb a vysoká rýchlosť je žiadaná. Príkladom je prenos videa a zvuku. Protokol TCP je využívaný častejšie, pretože overuje prijaté dáta a stratené dáta sa znova posielajú. Protokoly TCP a UDP sa adresujú portom, ktorý má hodnotu 0 až 65535. Porty 0 až 1023 sú tzn. dobre známe porty, väčšinou na ich otvorenie sú potrebné práva administrátora. Bežia na nich najčastejšie systémové aplikácie. Porty 1024 až 49151 sú registrované porty. Porty 49152 až 65535 nie sú registrované a využívajú ich rôzne aplikácie, často sú používané dočasne, dynamicky alebo privátne.

Aplikačná vrstva slúži na protokoly jednotlivých aplikácií. Známe protokoly sú napríklad FTP, HTTP, DHCP, DNS a podobne. Táto vrstva je vyhradená pre užívateľsky definované protokoly. Aplikačný protokol môže byť postavený aj na inom aplikačnom protokole pre ešte väčšiu abstrakciu od sieťovej komunikácie. Väčšinu sieťového prenosu vytvárajú aplikácie využívajúce služby aplikačnej vrstvy. Pre prenos informácií o dokumente je nutné

využiť niektorý z už existujúcich aplikačných protokolov, alebo navrhnúť vlastný. V ďalších podkapitolách sa budem zaoberať už len aplikačným protokolom.

2.2 Vlastný sieťový protokol

Vo fáze návrhu aplikácie bolo zvažované použiť rôzne aplikačné protokoly, na ktorých sa mohlo postaviť vlastný protokol, alebo doplniť existujúci protokol, poprípade vytvoriť celý aplikačný protokol vlastný. Požiadavky na aplikačný protokol pre editor boli hlavne možnosť naviazať spojenia a jeho udržovanie, správa užívateľov, v minimálnej miere aspoň ich pridávanie a odoberanie. Ďalej odosielanie zoznamu členov na požiadanie, rozposielanie zmien v dokumente, správa chýb a posielanie celého dokumentu.

Pri týchto požiadavkách je strata dát neprípustná a preto je na transportnej vrstve výhodné použiť protokol TCP[8]. Ďalšia výhoda protokolu TCP je automatické udržiavanie spojenia o ktoré sa už netreba starať. Veľmi dôležité je aj poradie prijímania správ. Protokol TCP automaticky radí dáta do poradia v akom boli odoslané.

Pridávanie členov mohlo prebiehať buď pripojením editora k vlastníkovi dokumentu, alebo výzvou vlastníka dokumentu, napríklad formou pozvánky. Odoberanie členov je možné odhlásením sa člena zo zdieľania dokumentu, poprípade jeho vyhodením vlastníkom dokumentu. Ďalej pri odoberaní a pridávaní členov a na vyžiadanie musí byť schopný vlastník dokumentu rozposlať aktuálny zoznam pripojených členov.

Ďalej protokol musí byť schopný umožniť preposlať celý dokument, napríklad novému pripojenému členovi, poprípade na vyžiadanie. Ak by nastala chyba v aplikácií, napríklad zlým interpretovaním prijatých dát v operačných transformáciách vo forme zmeny dokumentu, tak by mal protokol umožniť odoslanie celého dokumentu len od jedného zdroja - vlastníka, vďaka čomu by sa obsahy dokumentov každého člena synchronizovali.

Samotné zmeny v dokumente sú generované knižnicou operačných transformácií, popísanej v kapitole 3. Obsahujú informácie o tom, kto zmenu urobil, jednoznačný identifikátor zmeny, pozíciu na ktorej zmena nastala, počet znakov odobraných a počet znakov pridaných, samotný formátovaný text, napríklad vo formáte HTML a zoznam zmien ktoré užívateľ aplikoval. Takúto zmenu posiela každý člen vlastníkovi dokumentu a ten tieto zmeny rozposiela každému členovi. Takáto komunikácia je známa ako klient-server, kde vlastník dokumentu sa správa ako server.

Protokol TCP ani UDP neriešia naväzovanie spojenia v sietiach NAT. Toto je nutné riešiť na úrovni aplikačného protokolu. V prípade, že je nutné, aby užívateľ začať zdieľať dokument, ale jeho počítač s editorom zdieľaným dokumentov by bol pripojený do siete pomocou smerovača s funkciou NAT, je problematické sa k nemu napojiť, hlavne v prípade, ak by mal prevziať na seba úlohu servera.

2.3 Protokol XMPP

Aplikačný protokol XMPP[1] pôvodne vznikol ako protokol pre sieť jabber určenú na chat. Je postavený na jazyku XML, vďaka čomu je ľahko rozšíriteľný o nové možnosti. Tento protokol je popísaný aj v RFC[6][7]. O vývoj protokolu sa stará XMPP Standards Foundation. Rozšírenia nad rámec RFC sú vydávané v podobe XEP (XMPP extension protocol). Servery ktoré podporujú tento protokol štandardne načúvajú na TCP porte 5222. Pre vzájomnú komunikáciu serverov je vyhradený TCP port 5269.

[1] Extensible Messaging and Presence Protocol - rozšíriteľný protokol pre posielanie správ a zistenie stavu

Sieť využívajúca tento protokol nie je centralizovaná, jak je zvykom u iných IM sietí, ale je distribuovaná na servery po celom svete. Identifikátor užívateľa je podobný e-mailovej adrese, teda `meno@server`. Tento identifikátor sa volá Jabber ID, skrátene JID. Ďalej identifikátor obsahuje zdroj, teda popis klienta z ktorého je prihlásený. Takto vzniklý identifikátor je takzvaný FULL JID, ktorého tvar je `meno@server/zdroj`. Tento zdroj je potrebný pre jednoznačnú identifikáciu klienta, pretože na jednom účte môže byť užívateľ pripojený naraz z viacerých klientov. Pri naväzovaní komunikácie sa rozhoduje, s ktorým klientom sa bude komunikovať podľa zdroja, pokiaľ ho druhá strana zadá, alebo podľa priority. Poprípade sa môže spojenie naviazať so všetkými klientmi, ale následná komunikácia bude prebiehať s klientom, ktorý odpovie.

Užívateľ sa pripája k serveru, na ktorom je registrovaný a server je schopný overiť jeho identitu. V prípade, že by užívateľ chcel komunikovať s užívateľom registrovaným na inom servery, pripojí sa server užívateľa k serveru, na ktorom je registrovaný druhý užívateľ. Následne si serveri vymenia potrebné informácie, napríklad overenie existencie daného užívateľa, získanie jeho stavu, preposielanie správ od užívateľov a podobne. Sieť jabber ďalej podporuje viacužívateľské diskusie postavené na princípe miestností podobné miestnostiam známych z protokolu IRC, transporty pre možnosť pripojiť sa do iných sietí postavených na rôznych protokoloch a iné služby.

2.4 Posielanie správ pomocou protokolu XMPP

V tejto podkapitole názorne ukážem, ako vyzerajú jednoduché správy XMPP. V nasledujúcich ukážkach správ XMPP je JID odosielateľa odosielatel@domena.cz a jid príjemcu je prijemca@domena2.cz. Pri posielaní správy XMPP najprv odosielateľ pošle správu na svoj server, ktorý ju následne prepošle na server príjemcu. Táto správa sa buď na servery príjemcu v prípade jeho neprítomnosti uloží a doručí sa až pri jeho prihlásení, alebo sa doručí okamžite.

zdrojový kód 2.4.1 správa od odosielatel@domena.cz pre prijemca@domena2.cz

```
1 <message
2   from="odosielatel@domena.cz"
3   to="prijemca@domena2.cz"
4   type="chat"
5   id="nahodneID"
6   xml:lang="cz">
7     <body>Ahoj svet</body>
8 </message>
```

V ukážke XMPP správy 2.4.1 položka id obsahuje náhodný identifikátor, ktorý identifikuje aktuálny chat, vo väčšine klientov reprezentované ako jedno okno s chatom. Vďaka tomu je možné mať s jedným užívateľom viac chatov na rôzne témy. Položka `xml:lang="cz"` identifikuje jazyk správy, v tomto prípade český jazyk. Je možné správu poslať naraz vo viacerých jazykoch, pričom klient podporujúci protokol XMPP ukáže správu v preferovanom jazyku. Ak je príjemca prihlásený naraz pomocou viacerých klientov, tak túto správu doručí server príjemcu na klienta s najvyššou prioritou.

V ukážke XMPP správy 2.4.2 na riadku 3 je narozdiel od ukážky 2.4.1 príjemca doplnený o identifikátor klienta. V tomto prípade správy chodia len na identifikovaného klienta, ak je aktuálne klient s daným identifikátorom pripojený na server. V opačnom prípade sa správy

zdrojový kód 2.4.2 správa od odosielatel@domena.cz pre prijemca@domena2.cz/klient

```
<message
  from="odosielatel@domena.cz"
  to="prijemca@domena2.cz/klient"
  type="chat"
  id="nahodneID"
  xml:lang="cz">
    <body>Ahoj svet</body>
</message>
```

doručia na klienta s najvyššou prioritou.

2.5 Vlastné rozšírenia protokolu XMPP

Výhody protokolu XMPP oproti vlastnému sieťovému protokolu, sú hlavne možnosť autentifikácia na už existujúcom servery, bezproblémová komunikácia klientov aj v počítačovej sieti za sieťovým prvkom nat, ktorú je pomerne ťažké naviazať bez pomoci servera. Preto je najjednoduchšia cesta rozšíriť protokol XMPP o potreby editora zdieľaných dokumentov. V tejto podkapitole predstavím príklady správ, ktoré rozširujú protokol XMPP s krátkym popisom správ a ich účelu. Jednotlivé správy slúžia na overenia prítomnosti vhodného klienta, pripájanie a odpájanie klientov od zdielaného dokumentu, zoznam pripojených užívateľov zdieľajúcich dokument, posielanie zmien vzniklých v dokumente, ošetrovanie chýb a preposielanie celého dokumentu.

zdrojový kód 2.5.1 Klient žiada kontakt o overenie, či podporuje zdieľanie súborov.

```
<iq id='colabEditJobes' from='foo0@example.com/colab' to='foo1@example.
    com/colab' type='get'>
    <explore xmlns='http://teravik.cz/ns/collabEdit'/>
</iq>
```

V príklade správy 2.5.1 klient žiada kontakt o overenie, či podporuje zdieľanie súborov. Táto správa sa odosiela každému pripojenému klientovi kontaktu, ktorý sa užívateľ uložený na servery XMPP medzi kontaktmi. Správa sa posiela aj v prípade, že sa klient kontaktu pripojí.

zdrojový kód 2.5.2 Klient posiela správu, že zdieľanie dokumentov podporuje.

```
<iq id='colabEditJobes' from='foo1@example.com/colab' to='foo0@example.
    com/colab' type='result'>
    <explore xmlns='http://teravik.cz/ns/collabEdit' node='succesfull'/>
</iq>
```

Príklad 2.5.2 je veľmi podobný príkladu správy 2.5.1. Rozdiel je v časti explore, kde je pridaný atribut node='succesfull'. Táto správa je odpoveď na správu 2.5.1 v prípade, že klient podporuje zdieľanie súborov.

Príklad 2.5.3 je štandardná správa XMPP, ktorou štandardne odpovedajú klienti v prípade, že správu, ktorú prijali, nepodporujú.

zdrojový kód 2.5.3 Klient nepodporuje zdieľanie súborov.

```
1  <iq id='colabEditJobes' from='foo1@example.com/cb' to='foo0@example.com/
       colab' type='error' >
2       <explore xmlns='http://teravik.cz/ns/collabEdit'/>
3       <error code='503' type='cancel'>
4         <service-unavailable xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
5       </error>
6  </iq>
```

zdrojový kód 2.5.4 Vlastník dokumentu posiela správu, že pripojil ďalšieho člena do zdieľania dokumentu.

```
1  <iq id='colabEditJobes' from='foo0@example.com/colab' to='foo1@example.
       com/colab' type='set'>
2       <newUser xmlns='http://teravik.cz/ns/collabEdit' id="5" userName='
            volakdo@exmple.com'/>
3  </iq>
```

V príklade 2.5.4 vlastník dokumentu posiela správu všetkým členom zdieľania dokumentu, že pripojil ďalšieho člena. Parameter `id` v elemente `newUser` udáva prioritu tohoto člena. Táto priorita je jedinečná a slúži zároveň ako numerický identifikátor tohoto člena zdieľania dokumentu. Parameter `userName` v elemente `newUser` obsahuje JID[2] tohoto člena.

zdrojový kód 2.5.5 Vlastník dokumentu posiela správu, že člena zdieľania odpojil.

```
1  <iq id='colabEditJobes' from='foo0@example.com/colab' to='foo1@example.
       com/colab' type='set'>
2       <delUser xmlns='http://teravik.cz/ns/collabEdit' userName='
            volakdo@exmple.com'/>
3  </iq>
```

Správa v príklade 2.5.5 je veľmi podobná správe z príkadu 2.5.4. Mení sa len meno tagu z `newUser` na `delUser`. Táto správa oznamuje členom zdieľania, že vlastník dokumentu odpojil užívateľa. Preto sú parametri a použitie totožné so správou v príklade 2.5.4.

zdrojový kód 2.5.6 Člen zdieľania dokumentu posiela správu vlastníkovi dokumentu, že sa odpája.

```
1  <iq id='colabEditJobes' from='foo1@example.com/colab' to='foo0@example.
       com/colab' type='set'>
2       <bye xmlns='http://teravik.cz/ns/collabEdit'/>
3  </iq>
```

V príklade 2.5.6 člen zdieľania oznamuje vlastníkovi dokumentu, že sa odpája. Vlastník dokumentu následne pošle správu z prikladu 2.5.5 všetkým ostávajúcim členom zdieľania.

V príklade 2.5.7 vlastník dokumentu vyzýva užívateľa siete jabber, aby sa pripojil do zdieľania dokumentu. Element `connect` má parameter `docName`. Tento parameter obsahuje názov dokumentu.

[2] jednoznačný identifikátor užívateľa v sieti jabber

zdrojový kód 2.5.7 Vlastník dokumentu žiada užívateľa siete JABBER o spoluúčasť na zdieľaný dokumentu

```
1 <iq id='colabEditJobes' from='foo0@example.com/colab' to='foo1@example.
      com/colab' type='get'>
2     <connect xmlns='http://teravik.cz/ns/collabEdit' docName='
          foo0@example.com/resources:subor.txt' />
3 </iq>
```

zdrojový kód 2.5.8 Užívateľ siete JABBER prijíma pozvánku.

```
1 <iq id='colabEditJobes' from='foo1@example.com/colab' to='foo0@example.
      com/colab' type='result'>
2     <connect xmlns='http://teravik.cz/ns/collabEdit' docName='
          foo0@example.com/cola:subor.txt' result='yes'/>
3 </iq>
```

V správe 2.5.8 užívateľ, ktorý dostal správu z príkladu 2.5.7, odpovedá kladne a pripája sa do zdieľania dokumentu. V tagu `connect` je atribut `result`, ktorý je nastavený na hodnotu `yes`, ktorá značí, že sa klient pripája do zdieľania. V prípade nastavenia atributu `result` na hodnotu `no` užívateľ pozvánku odmieta.

zdrojový kód 2.5.9 Člen zdieľania dokumentu požaduje od majiteľa dokumentu kompletný zoznam pripojených členov.

```
1 <iq id='colabEditJobes' from='foo1@example.com/colab' to='foo0@example.
      com/colab' type='get'>
2     <userList xmlns='http://teravik.cz/ns/collabEdit'/>
3 </iq>
```

Člen zdieľania dokumentu si v príklade správy 2.5.9 žiada zoznam všetkých pripojených členom zdieľania. V tomto prípade je element `userList` prázdny.

zdrojový kód 2.5.10 Vlastník dokumentu posiela kompletný zoznam členov zdieľania dokumentu.

```
1 <iq id='colabEditJobes' from='foo0@example.com/colab' to='foo1@example.
      com/colab' type='result'>
2     <userList xmlns='http://teravik.cz/ns/collabEdit' >
3       <user name="uzivatel2@example.com" id="2">
4       <user name="uzivatel3@example.com" id="3">
5       <user name="uzivatel4@example.com" id="4">
6       <oldUsers name="uzivatel1@example.com" id="1">
7     </userList>
8 </iq>
```

Vlastník dokumentu odpovedá na správu 2.5.9 správou 2.5.10, ktorá sa od nej líši tým, že element `userList` obsahuje členov zdieľania. Vlastník dokumentu sa nikdy nepýta na zoznam užívateľov. Okrem vlastníka dokumentu nikto nesmie posielať zoznam pripojených členov. Element `userList` obsahuje elementy `user`. Každý element `user` označuje jedného člena zdieľania dokumentu. Parameter `name` obsahuje členov JID, parameter `id` obsahuje

prioritu člena, ktorá zároveň slúži ako jednoznačný numerický identifikátor.

zdrojový kód 2.5.11 Správa prenášajúca zmeny v dokumente s údajmi potrebnými pre operačné transformácie.

```
1  <iq id='colabEditJobes' from='foo0@example.com/colab' to='foo1@example.
       com/colab' type='set'>
2       <patch xmlns='http://teravik.cz/ns/collabEdit' messageId='5' sender=
            '1' pos='4' removed="3" added="1">
3         <text>add text</text>
4         <apliedChange id="4" message="1">
5         <apliedChange id="4" message="2">
6         <apliedChange id="4" message="3">
7       </patch>
8  </iq>
```

Správa v príklade 2.5.11 slúži na prenos informácii z operačných transformácií. Štruktúra správy odpovedá vstupným a výstupným parametrom knižnice operačných transformácií. Element `patch` obsahuje parametry `messageID`, ktorý značí poradové číslo správy, `sender`, ktorý je jednoznačný numerický identifikátor odosielateľa, `pos`, pozícia na ktorej vznikla zmena, `removed`, počet odstránenych znakov, `added`, počet pridaných znakov. Ďalej tento element obsahuje podelement `text`, obsahujúci formátovaný text v jazyku HTML. Ďalaj element `patch` obsahuje podelementy apliedChange, ktorý má parametry `id` a `message`, ktoré jednoznačne identifikujú správy, ktoré tento klient prijal od posledného odoslania správy z príkladu 2.5.11.

zdrojový kód 2.5.12 Chyba v dokumente

```
1  <iq id='colabEditJobes' from='foo0@example.com/colab' to='foo1@example.
       com/colab' type='error'>
2       <session xmlns='http://teravik.cz/ns/collabEdit'  />
3  </iq>
```

Správou v príklade 2.5.12 klient upozorňuje iných klientov, že na jeho strane nastala chyba, a preto je treba dokument reinicializovať od vlastníka dokumentu.

zdrojový kód 2.5.13 Vlastník dokumentu posiela členovi zdieľania dokumentu celý dokument po častiach.

```
1  <iq id='colabEditJobes' from='foo0@example.com/colab' to='foo1@example.
       com/colab' type='set'>
2       <document xmlns='http://teravik.cz/ns/collabEdit' parts="15" current
            ="3">
3         <content>tu je cely subor</content>
4         <owners>0,0,0,1,1,1,0,0,1,1</owners>
5       </document>
6  </iq>
```

Správa v príklade 2.5.13 slúži na posielanie celého dokumentu členom zdieľania. Túto správu posiela iba vlastník dokumentu, a to na vyžiadanie od klienta, alebo po neobnoviteľnej chybe v dokumente. Maximálna veľkosť správy je na jabber servery nastavená štandardne na 60KB, preto sa večšie dokumenty posielajú po častiach. Celkový počet častí,

na ktoré sa dokument rozdelil, je posielaný v elemente `document`, v parametry `patrs`. Aktuálna posielaná časť je v parametry `current`. Parameter `document` má podelement `content`, ktorý obsahuje text formátovaný pomocou jayzka HTML. Výsledný text dokumentu sa poskladá jednoduchým spojením reťazcov. Ďalším podelementom elementu `document` je `owners`. Tento podelement obsahuje jednoznačným numerický identifikátor autora znaku. Je zložený z čísel a oddeľovača čísiel, ktorým je čiarka. Pozícia čísla je rovnaká, ako pozícia znaku, ku ktorému patrí. Každý znak dokumentu má práve jedného autora.

zdrojový kód 2.5.14 Člen zdieľania dokumentu vyžaduje od vlastníka dokumentu obsah celého dokumentu.

```
1 <iq id='colabEditJobes' from='foo0@example.com/colab' to='foo0@example.
    com/colab' type='get'>
2     <document xmlns='http://teravik.cz/ns/collabEdit' />
3 </iq>
```

Správa v príklade 2.5.14 slúži na vyžiadanie si celého dokumentu klientom. Túto správu klient posiela pri chybe a pri pripojení sa už bežiaceho zdieľania dokumentu.

Kapitola 3

Operačné transformácie

V tejto kapitole sa venujem technológii, ktorá má na starosť prepočítavanie pozície prijatého textu tak, aby bol dokument odosielateľa textu a prijímateľa textu rovnaký po aplikovaní prijatého textu. Vďaka tomu sú dokumenty všetkých užívateľov synchronizované v reálnom čase.

V prvje podkapitole v krátkosti predstavujem operačné transformácie [4] a na jednoduchom príklade opisujem situáciu, pri ktorej sa bez operačných transformácií nezaobídem.

V podkapitole 3.2 popisujem jednotlivé atribúty vstupu a výstupu operačných transformácií. Ďalej vysvetľujem u každého atribútu dôvod, prečo by bez neho operačné transformácie nefungovali. Naznačujem spôsob formátovania textu, o ktorom samotné operačné transformácie nemusia uvažovať. Vďaka tomu je možné formátovať text bez zvýšenia zložitosti operačných transformácií.

V podkapitole 3.3 vysvetľujem na názornom príklade funkciu operačných transformácií. Príklad je jednoduchý, avšak ukazuje situáciu, ktorá nie je riešiteľná bez operačných transformácií. V tomto príklade sa pre jednoduchosť používajú len operácie pridávanie textu.

V poslednej podkapitole popisujem ďalšie možnosti operačných transformácií. Zaoberám sa problémom, ktoré môžu nastať pri mazaní textu. Ku každému problému popisujem spôsob riešenia tohoto problému. Na záver načrtávam spôsob, akým je možné operačné transformácie upraviť tak, aby mohlo editovať dokument viac užívateľov.

3.1 Úvod do operačných transformácií

Operačné transformácie [4] sú technológia, ktorá slúži na podporu kolaborácie v kolaboračných systémoch. Princíp operačných transformácií spočíva v tom, že každá zmena vytvorená účastníkom kolaborácie má pozíciu, na ktorej vznikla. Iný účastník mohol urobiť zmenu v tom istom čase na inej pozícií v texte. Keby sa tieto zmeny aplikovali ako klasický patch, tak by sa mohlo stať, že by texty u oboch účastníkov kolaborácie neboli rovnaké. Napríklad pri prázdnom dokumente, ak jeden užívateľ pridá text `A`, a druhý užívateľ v tom istom čase pridá text `B`, tak by obe zmeny vznikli na pozícii 0. Po aplikovaní patchu by sa prvému užívateľovi doplnil text B na pozíciu 0, čím by získal text `BA`, ale druhému užívateľovi by na pozíciu 0 pribudol text `A`, čím by mu vzniklo `AB`. Takto vznikol každému užívateľovi rôzny text. Ďalšia kolaborácia medzi týmito užívateľmi je nemožná. Pri použití operačných transformácií sa u jedného z užívateľov text pridá na pozíciu 1, čím vznikne rovnaký text u oboch užívateľoch. Ak má byť výsledok rovnaký text u oboch užívateľov, je jedno u ktorého užívateľa transformácia prebehne, ale nemôže prebehnúť u oboch.

3.2 Atribúty jednotlivých zmien tvorených užívateľmi

Na začiatku kapitoly bolo spomenuté, že ak jeden užívateľ posiela do prázdneho dokumentu text A a druhý užívateľ text B, tak musí vzniknúť rovnaký text, nezáleží na tom či vznikne text AB alebo BA. Problém je v tom, že užívatelia navzájom nevedia, ako ostatný užívatelia poskladali text a teda ho nemôžu na základe nich poskladať rovnako. Preto je nutné zaviesť prioritu užívateľov, tj. poradie, v ktorom sa to od jednotlivých užívateľov začne výsledný text skladať.

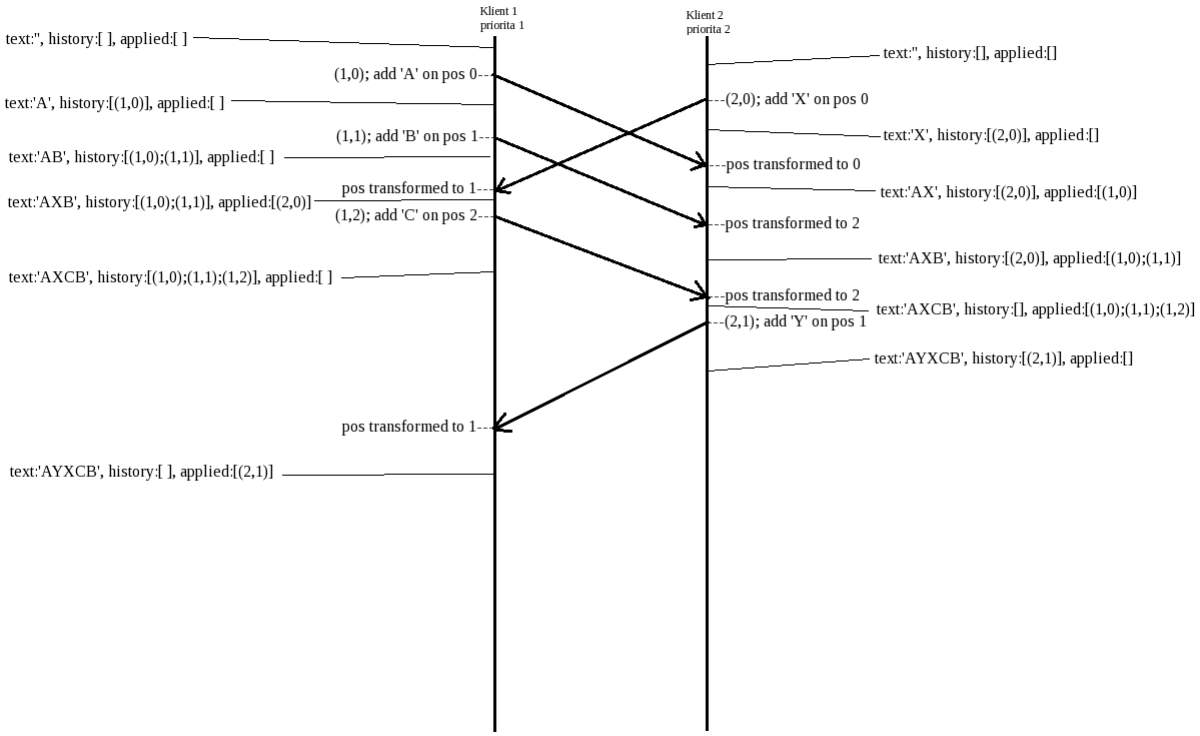
Ďalej je nutné identifikovať jednoznačne každú zmenu vytvorenú užívateľom, aby bola možná spolupráca s užívateľmi s rôzne rýchlim pripojením do počítačovej siete. Pre jednoznačné identifikovanie zmeny slúžia dva parametre. Prvý z nich označuje užívateľa, ktorý vytvoril zmenu. Druhý parameter identifikuje jednoznačne zmenu v rámci zmien užívateľa. Oba parametre sú typu celé číslo. Keďže priorita musí byť v rámci systému jedinečná, slúži zároveň aj pre identifikáciu užívateľa. Takže užívateľ s identifikátorom napríklad 5 má aj prioritu 5. Druhý parameter identifikujúci jednoznačne zmenu vytvorenú užívateľom je implementovaný ako číslo čítajúce poradie správ.

Keď už sú jednotlivé správy identifikované a vieme v akom poradí ich skladať, potrebujeme obsah jednotlivých správ. Dôležitá je pozícia, na ktorej urobil užívateľ zmenu. Pozícia sa berie z užívateľovho aktuálneho dokumentu, ako pozícia znaku, na ktorom urobil zmenu. Na túto pozíciu sa neaplikujú žiadne výpočty ani transformácie, tie sa aplikujú až na strane príjemcu. Ďalej pri každej zmene mohol užívateľ text pridať alebo odstrániť, poprípade oboje naraz, ak by text nahradzoval. Preto sú následné dva parametre počet pridaným znakov a počet odobraných znakov. Tieto dva parametre sa používajú pre výpočet transformácií na strane prijímateľa zmien. Samozrejme je potrebný aj text zmeny, tj. text, ktorý užívateľ pridal. Z tohoto textu nie je možné jednoducho odvodiť počet znakov pridaných v zmene, pretože tento text obsahuje formátovacie znaky, napríklad tagy HTML a podobne.

Posledným parametrom, ktorý obsahuje zmenová správa, je zoznam identifikátorov zmien (autor zmeny a poradové číslo zmeny), ktoré sa na strane klienta aplikovali od poslednej zmeny, ktorú tento užívateľ odoslal. Operačné transformácie majú v sebe históriu zmien, ktoré užívateľ urobil, ale protistrana ešte neaplikovala a poslala svoje zmeny. Vďaka tejto histórii je možné správne vypočítať novú pozíciu, na ktorej sa prijaté zmeny premietnu. Keď užívateľ prijme zoznam zmien, ktoré boli aplikované, tak ich najprv vymaže z histórie a až potom prepočítava pozíciu zmien.

3.3 Príklad a vysvetlenie jednoduchého skladania textu

Na obrázku 3.1 je znázornené posielania zmien v dokumente medzi dvoma klientmi. Po každej zmene je na obrázku zaznamenaný vnútorný stav operačných transformácií. Obrázok znázorňuje posielanie zmien v čase, ktoré by nebolo možné bez operačných transformácií aplikovať na dokument tak, aby mali obe strany komunikácie po prijatí všetkých zmien v dokumente rovnaké obsahy dokumentov. Klient `1` pošle dve zmeny, ktoré urobil v dokumente, zatiaľ čo prijme jednu zmenu, ktorú urobil klient `2`. Klient `2` za ten čas odošle jednu zmenu v dokumente a prijme dve zmeny od klienta `1`.



Obrázek 3.1: Posielanie zmien medzi užívateľmi a vnútorný stav operačných transformácií.

Na začiatku majú oba klienti prázdny dokument, históriu aj zoznam aplikovaných správ.

`Klient 1` pridá na pozícii `0` znak `A`. V ten istý okamžik pridá `klient 2` znak `X` na pozíciu `0`. Teraz má `klient 1` v dokumente text `A`, v histórii správu `(1,0)` kde 1 je priorita klienta a 0 je poradové číslo správy. Zoznam aplikovaných zmien má `klient 1` prázdny. `Klient 2` má v dokumente text `X`, v histórii správu `(2,0)` a prázdny zoznam aplikovaných správ.

`Klient 1` posiela správu `(1,1)`, v ktorej pridáva na pozíciu 1 znak `B`. Túto správu si pridá do histórie. Následne prijíma od `klienta 2` správu `(2,0)`. Táto správa sa transformuje tak, že sa prehľadajú všetky správy v histórii a posunie sa pozícia prijatého textu o znaky pridané pred tento text. V histórii je správa `(1,0)`, ktorá je na pozícii `0`, tak isto ako správa `(2,0)`. Keďže má správa `(1,0)` vyššiu prioritu, aj keď majú obe správy rovnakú pozíciu, ráta sa táto správa za správu s nižšou pozíciou, teda sa pozícia 0 správy `(2,0)` posunie o počet znakov správy `(1,0)`, tj. o 1 znak. Správa `(1,1)` je na pozícii 1 ako prepočítaná správa `(2,0)`, ale pri transformáciách je dôležitá pôvodná pozícia, teda u správy `(2,0)` sa uvažuje stále s pozíciou 0. Znak `X` sa pripisuje na pozíciu 1 u `klienta 1`, čím vzniká text `AXB`. Správa `(2,0)` bola aplikovaná do textu u `klienta 1`, preto sa pridáva do pola aplikovaných správ.

`Klient 2` prijíma od `klienta 1` správu `(1,0)`. V histórii `klienta 2` sa nachádza správa `(2,0)`. Tieto správy majú rovnakú pozíciu, ale správa v histórii má nižšiu prioritu, teda je považovaná za správu s vyššou pozíciou, preto sa pozícia prijatej správy nemení. Aktuálnu text sa mení na `AX`. Prijatá správa `(1,0)` sa pridáva do zoznamu aplikovaných zmien.

Následne prijíma `klient 2` správu `(1,1)` s pozíciou 1. V histórii je správa `(2,0)` s pozí-

ciou 0, teda sa bude meniť pozícia prijatej správy o dĺžku textu správy `(2,0)`, tj. o 1 znak. Výsledná pozícia správy `(1,1)` je 2, výsledný text u `klienta 2` je `AXB`. Správa `(2,0)` sa pridáva do pola aplikovaných správ.

Klient 1 posiela správu `(1,2)`, ktorá pridáva na pozíciu 2 znak `C`. Týmto vzniká u `klienta 1` text `AXCB`. Správa `(1,2)` obsahuje pole aplikovaných zmien, v ktorom je správa `(2,0)`. Pole aplikovaných zmien u `klienta 1` sa vyprázdni. Správa `(1,2)` sa ukladá do histórie.

Následne túto správu prijíma `klient 2`. Zo svojej histórie vymaže správy, ktoré prišli v správe `(1,2)` v poli aplikovaných správ. Tým sa história `klienta 2` vyprázdni. Pri prázdnej histórii nie je s čím porovnávať pozíciu v správe `(1,2)` a teda sa pozícia nemení, ale správa sa priamo aplikuje, tj. na pozíciu 2 sa pripíše znak `C`, čím vzniká text `AXCB`. V tento okamih je text u `klienta 1` aj text u `klienta 2` znova rovnaký.

`Klient 2` posiela správu `(2,1)` obsahujúcu text `Y` na pozícii 1. Správa obsahuje pole aplikovaných zmien a uloží sa do histórie `klienta 2`. Text u `klienta 2` je `AYXCB`. Následne `klient 1` túto správu prijme. Správy v zozname aplikovaných zmien sa vymažú z histórie `klienta 1`. Týmto sa vyprázdnila história `klienta 1` a teda transformácia správy od `klienta 2` neprebieha, ale priamo sa aplikuje. Text u `klienta 1` je `AYXCB`. Správa `(2,1)` sa uloží do zoznamu aplikovaných zmien.

Jednotlivé kroky sú znázornené na obrázku 3.1 aj s aktuálnym textom, históriou a zoznamom aplikovaných zmien. Ďalej vidieť na obrázku jednotlivé správy, ich pozície pri odosielaní a po transformácií samotný text (znak) posielaný v správe.

3.4 Ďalšie možnosti operačných tranformácií

Operačné transformácie musia okrem pridávania textu povoliť aj jeho mazanie. Pri mazaní môžu vzniknúť tri hlavné problémy.

Jeden klient vkladá text do bloku textu, zatiaľ čo druhý klient tento blok textu maže. Napríklad text `ABCDEF`, `klient 1` pridáva na pozíciu 3 text `XY` a `klient 2` maže text od pozície 2 dĺžky 3 znaky. `Klient 1` má v dokumente text `ABCXYDEF`', zatiaľ čo `klient 2` má text `ABF`. Riešenie je vymazať znaky tak, aby vznikol len text `ABF`, pretože pridávať text do už neexistujúceho bloku, odstavca alebo strany nemá zmysel. `Klient 1` musí vymazať obsah tej správy vo svojej histórii, ktorá sa zapisuje do mazaného bloku. Následne aplikuje správu, ktorá maže daný blok textu. `Klient 2` proste prijatú správu ignoruje.

Druhý problém nastane, ak `klient 1` maže blok, ktorý je v bloku mazanom `klientom 2`. Napríklad text `ABCDEF`, `klient 1` maže od pozície tri jeden znak, čím vzniká text `ABCEF`. `Klient 2` maže od pozície dva tri znaky, čím vzniká text `ABF`. Riešenie je pre `klienta 2` ignorovať zmenu od `klienta 1`, ale `klient 1` musí znížiť počet znakov, ktoré bude mazať, o počet znakov, ktoré už vymazal.

Tretí problém nastane keď sa mazané texty prekrývajú. Napríklad text `XABCY`, `klient 1` maže text `AB`, `klient 2` maže text `BC`. Riešenie je nechať iba text `XY`. Klienti si musia vypočítať počet znakov, ktoré mažú obaja a o tento počet znakov zmazať menej textu. `Klient 1` musí ešte pred mazaním posunúť pozíciu správy od klienta 2 o znaky, ktoré už vymazal, pretože sa mazaním textu zmenili pozície všetkých znakov v texte za týmto mazaním.

Doteraz som vysvetľoval operačné transformácie pre dvoch užívateľov. Pri spolupráci klient server má server zvlášť históriu pre každého klienta a klientom posiela už transformované správy. Pri komunikácii nezávislých klientov je to zložitejšie. Okrem toho že každý klient má históriu pre každý klient, treba rátať aj s tým, že od klienta 1 cez klienta 2 ku klientovi 3 môže prísť správa skôr ako priamo od klienta 1 ku klientovi 3. Keďže takáto sieť

klientov nie je riadená serverom, užívateľ sa môže odpojiť v čase, keď správu pošle len časti klientov a tým zničí celý dokument. Klienti, ktorým prišli od odpojeného klienta správy, majú iný dokument ako užívatelia, ktorý správu neprijali.

Kapitola 4

Implementácia

V prvej podkapitole popisujem, ako bola implementovaná komunikácia cez počítačovú sieť. Popisujem diagram znázorňujúci reálny prenos správ medzi jednotlivými klientmi a servermi.

V podkapitole 4.2 popisujem implementované operačné transformácie, spôsob predávania správ medzi nimi, funkciu vlastníka dokumentu. Ďalej rozoberám diagram, na ktorom je znázornená funkcia vlastníka dokumentu. Približujem rozdiel medzi vlastníkom dokumentu a ostatnými užívateľmi. Ku koncu podkapitoly opisujem spôsob, ktorým je možné operačné transformácie používať aj pre formátovaný text.

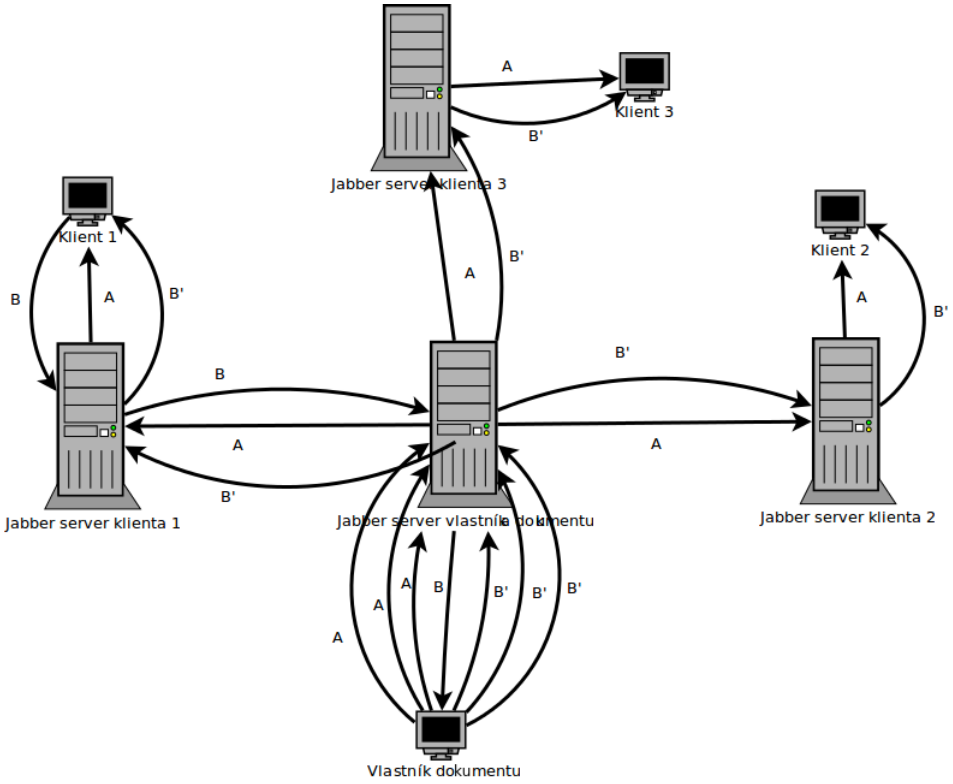
V podkapitole 4.3 popisujem hlavné triedy aplikácie a ich vzájomné vzťahy. Snažím sa opísať vnútornú štruktúru aplikácie bez zbytočných detailov. V podkapitole 4.4 popisujem užívateľské rozhranie aplikácie. Vysvetľujem úlohu jednotlivých častí hlavného okna.

V podkapitole 4.5 popisujem automatické testy, ktoré boli v rámci práce vytvorené. Tieto testy sú zamerané na operačné transformácie. V poslednej podkapitole píšem o plánoch do budúcnosti pre túto aplikáciu.

4.1 Podrobnosti o sieťovej komunikácie

Komunikáciu cez počítačovú sieť zaisťuje knižnica QXMPP. Táto knižnica využíva služieb frameworku Qt[3]. Framework Qt poskytuje radu tried pre komunikáciu cez počítačovú sieť, medzi nimi aj wrapper na scokety TCP. Znamená to, že klasické sockety TCP sú dostupné pomocou technológií Qt. Knižnica QXMPP implementu je protokol aplikačný XMPP[9]. Samostatne dokáže naviazať spojenie so serverom, udržiavať s ním spojenie a odpovedať štandardne na štandardné správy. V prípade neštandardnej správy prenecháva jej spracovanie rozšíreniam.

Zoznam jednotlivých správ, ktoré podporuje rozšírenie knižnice QXMPP, navrhnuté v rámci tejto práce, sa nachádza v kapitole 2.5. Jednotlivé správy sú zamerané na funkčnosť v sieti klient-server, kde za server sa považuje vlastník dokumentu, ktorý je objektom zdieľania. Jednotlivý klienti sa k vlastníkovi dokumentu nepripájajú ako je u serveru zvykom, ale vlastník dokumentu vyzýva klientov k tomu, aby sa pripojili k zdieľaniu a upravovaniu dokumentu. Jednotlivé správy sú navrhnuté pre možnosť byť vlastníkom len jedného dokumentu, čo je dosť obmedzujúce, ale na prvú verziu postačujúce.



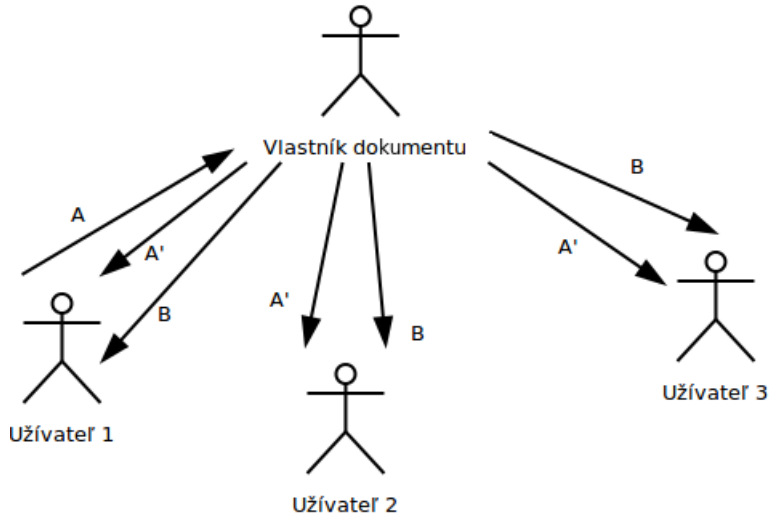
Obrázek 4.1: Komunikácia užívateľov cez počítačovú sieť.

Na obrázku 4.1 je znázornená komunikácia medzi štyrmi klientmi, pričom každý klient používa iný jabber server. Vlastník dokumentu posiela správu A všetkým užívateľom. Keďže plní funkciu servera, musí rozposlať túto správu zvlášť každému užívateľovi. Pošle pre každého užívateľa správu na svoj jabber server, ktorý túto správu pošle ďalej serverom klientov. Servery klientov pošlú túto správu klientom. Ďalšiu správu posiela klient 1. Adresuje ju len vlastníkovi dokumentu. Pošle ju na svoj jabber server, ktorý ju pošle ďalej na jabber server vlastníka dokumentu, a ten ju doručí vlastníkovi dokumentu. Server upraví správu na základe operačných transformácií (viz. kapitolu 3 a 4.2). Takto upravenú správu, `B'`, rozpošle všetkým užívateľom, ako v prípade správy A.

4.2 Podrobnosti o použitých operačných transformáciách

Operačné transformácie[4] boli popisované v kapitole 3. Pre implementáciu operačných transformácií v tejto práci je dôležitá trieda `QTextDocument`, ktorá obsahuje celý editovaný dokument. Samotný text je možné upravovať v ľubovolnej komponente Qt[3], ktorá podporuje takýto dokument. Operačné transformácie priamo pracujú s touto triedou dokumentu. Na základe vstupu operačných transformácií operačné transformácie upravia vstupný text, a na správnej pozícií ho vložia do dokumentu. Pri zmene textu dokument informuje operačné transformácie o tejto udalosti, a predá im potrebne parametre. Parametre sú pozícia, kde zmena vznikla, počet znakov pridaných a počet znakov zmazaných. Na výstup operačných transformácií sa vygeneruje správa, ktorú stačí vložiť na vstup operačných transformácií iného klienta.

Samotné rozhranie operačných transformácií sa používa veľmi jednoducho, stačí výstup operačných transformácií z jedného klienta vložiť na vstup operačných transformácií druhého klienta. Ďalej pri inicializácií operačných transformácií je potrebné predať odkaz na dokument. Problém nastáva pri zmene počtu užívateľov. V prípade dvoch rovnocenných klientov sa pred editovaním inicializuje v operačných transformáciách priorita klientov. Pri komunikácií typu klient-server je nutné server rozšíriť o možnosť pridávať a meniť užívateľov za chodu. Za server sa vždy považuje klient vlastniaci dokument a automaticky dostáva prioritu 0, ktorá je najvyššia možná.



Obrázek 4.2: Komunikácia užívateľov operačných transformácií

Na obrázku 4.2 je znázornená komunikácia medzi užívateľmi operačných transformácií. Vlastník dokumentu plní funkciu servera, ostatný užívatelia plnia funkciu klienta. Ako prvý zmení dokument `užívateľ 1`. Pošle správu `A` vlastníkovi dokumentu. Vlastník dokumentu túto zmenu aplikuje pomocou operačných transformácií do svojho dokumentu. Takto transformovanú správu posiela všetkým užívateľom. Transformovaná správa má rovnakú pozíciu, ako by zmenu v dokumente vytvoril sám vlastník dokumentu. Všetci užívatelia majú operačné transformácie prispôsobené tak, aby prijímali správy len od vlastníka dokumentu. To je spôsobené tým, že majú len jednu históriu. Vlastník dokumentu má históriu zvlášť pre každého klienta. Klienti aplikujú transformovanú právu `A'` pomocou svojich operačných transformácií, okrem `klienta 1`. `Klient 1` zistí, že je autor tejto zmeny a jednoducho ju zahodí. Pri správe B je situácie jednoduchšia. Zmenu vytvorí vlastník dokumentu a všetci klienti ju jednoducho aplikujú pomocou operačných transformácií.

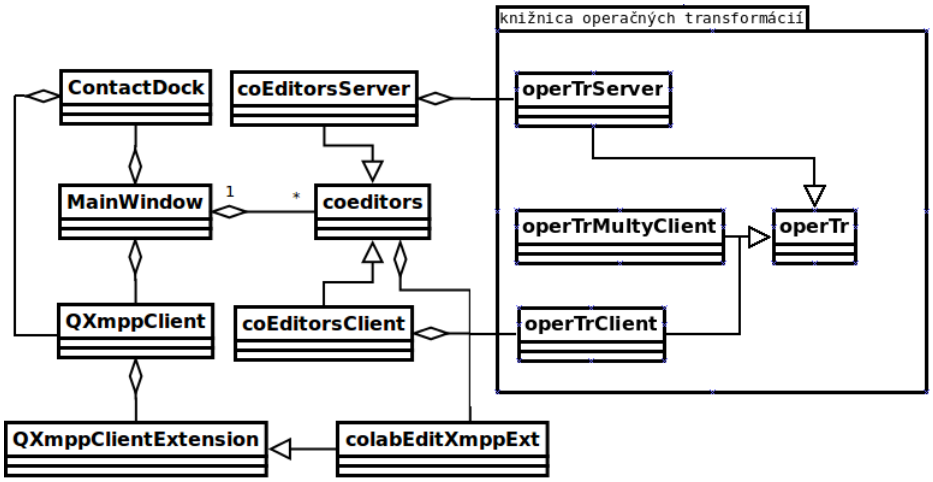
Operačné transformácie implementované v tejto práci podporujú formátovaný text. Je to možné vďaka tomu, že dokument vo frameworku Qt[3] podporuje importovať a exportovať vybrané časti textu ako text vo formáte HTML. Operačné transformácie požadujú pre výpočet pozície počet znakov textu. Tento počet znakov je počet znakov textu neformátovaného. Pozícia sa počíta rovnako ako pri neformátovanom texte. Pri vkladaní na správnu pozíciu sa o aplikovanie formátovania postará samotný framework Qt.

Tieto operačné transformácie nemajú žiadnu vyrovnávaciu pamäť a teda jednotlivé zmeny sú posielanie okamžite protistrane. Výhoda je minimálna latencia. Nevýhoda je vysoká sieťová a pamäťová náročnosť. Zmena každého znaku sa posiela zvlášť a ukladá sa

do histórie. Pri formátovaní HTML a posielaní v správe protokolu XMPP je takýto prístup príliš náročný na zdroje a pri pomalšom sieťovom pripojení sa môže výhoda nízkej latencie úplne vytratiť.

4.3 Diagram tried

Operačné transformácie a editor zdieľaného dokumentu sú napísané v jazyku C++. Využívajú služieb frameworku Qt[3] a knižnice QXMPP. Operačné transformácie aj editor zdieľaného dokumentu sú navrhnuté objektovo. Relácie medzi triedami sú znázornené na obrázku 4.3. Ako vidieť na obrázku, operačné transformáciu sú nezávislá knižnica, ktorú môžu používať aj iné projekty.



Obrázek 4.3: Diagram tried editora zdieľaného textu.

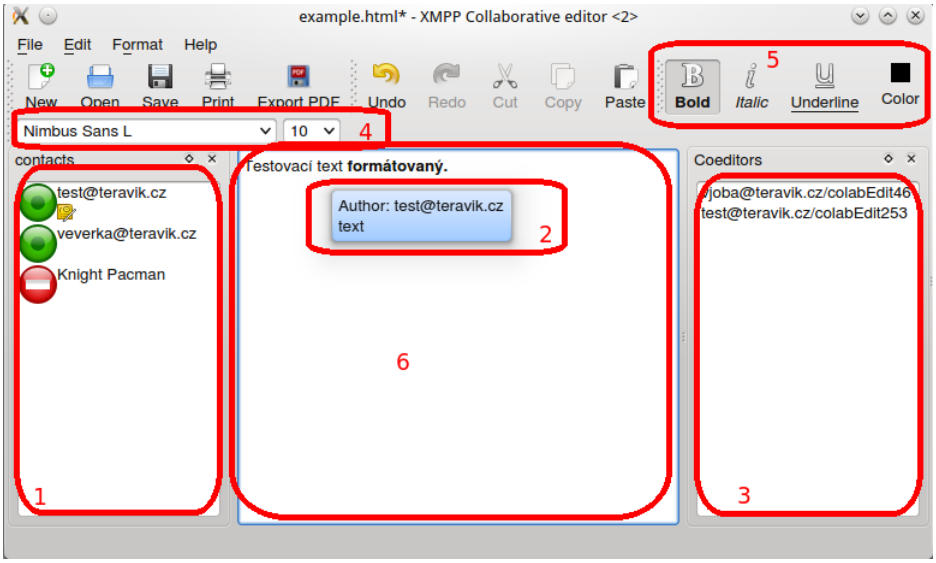
Na obrázku 4.3 visieť, že hlavná trieda, ktorá ovláda celú aplikáciu, je `MainWindow`. Táto trieda implementuje hlavné okno aplikácie. Obsahuje komponenty okna a samotné editovacie pole, s ktorým spolupracujú operačné transformácie.Ďalej obsahuje inštanciu typu `QXmppClient`, ktorej úlohou je sieťová komunikácia, a je implementovaná v knižnici QXMPP. K tejto inštancii pripája rozšírenie triedy `colabEditXmppExt`, ktoré implementuje správy popísané v kapitole 2.4.

Trieda `ContactDock` má za úlohu pripájanie a udržiavanie spojenia s jabber serverom, zobrazovať zoznam kontaktov užívateľa a pozívať užívateľov k zdieľaniu dokumentu. Táto trieda sa stará o prihlásenie užívateľa k jabber serveru aj v prípade, keď prihlasovacie údaje boli zadané do príkazového riadku.

Trieda `coeditors` zobrazuje zoznam užívateľov, ktorý aktuálne editujú zdielaný dokument. Na základe funkcie užívateľa sa používa zdedená trieda `coEditorsServer` alebo `coEditorsClient`. Tieto triedy obsahujú odkaz na inštanciu operačných transformácií, s ktorými spolupracujú a ovládajú ich. Ďalej priamo spolupracujú s triedou `colabEditXmppExt`. V podstate potomkovi triedy `coeditors` sú mosty medzi operačnými transformáciami a sieťovým prenosom.

4.4 Užívateľské rozhranie

Užívateľské rozhranie aplikácie je znázornené na obrázku 4.4. Skladá sa z hlavného okna, ktoré obsahuje hlavné menu, panely nástrojov, dokovacie okná, stavový riadok a textové pole s vlastným dokumentom. Dokovacie okná je možné z hlavného okna premiestniť na inú pozíciu v hlavnom okne, alebo úplne od neho odpojiť.



Obrázek 4.4: Kolaboratívny editor textu

Na obrázku 4.4 sú vyznačené hlavné časti aplikácie. Pod označením 1 je zoznam kontaktov užívateľa. Tento zoznam je uložený na jabber servery. V prípade, že užívateľ nie je prihlásený k svojmu jabber serveru, namiesto zoznamu kontaktov sa na tomto mieste nachádza formulár umožňujúci prihlásenie. Tento zoznam kontaktov je umiestnený v dokovacom okne. V zozname kontaktov vidieť ikonu, ktorá určuje stav kontaktu. Vedľa ikony sa nachádza meno tohoto kontaktu. Pod menom je umiestnená žltá ikona dokumentu, ktorá slúži na pozvanie kontaktu k editovaniu dokumentu. V prípade, že kontakt editovanie dokumentu nepodporuje, žltá ikona sa nezobrazí.

Ďalšie dokovacie okno obsahuje zoznam užívateľov, ktorý aktuálne editujú dokument. Na obrázku 4.4 je označený číslom 3. Toto dokovacie okno sa zobrazuje až po prihlásení na jabber server.

Dominantnou časťou hlavného okna je editor textu, na obrázku 4.4 označený číslom 6. Slúži na editáciu dokumentu a jeho formátovanie. V prípade, ak užívateľ nechá kurzor myši nad textom, zobrazí sa mu informačný text. Tento text nesie informácie o texte, na ktorom sa nachádza kurzor myši. Obsahuje meno užívateľa, ktorý text pod kurzorom myši napísal. Ďalej obsahuje maximálne 20 znakov, ktoré tento autor napísal v okolí kurzoru myši tak, aby vzniklý text bol súvislí výňatok z dokumentu.

Na panely nástrojov sa nachádzajú formátovacie nástroje pre formátovanie textu. Na obrázku 4.4 pod označením 4 sa nachádza komponenta na výber druhu písma. Vedľa nej sa nachádza komponenta na výber veľkosti písma. Ďalej na tomto obrázku pod označením 5 sa nachádzajú stavové tlačítka na zmenu vlastností písma ako je kurzíva, podtrhnutie a tučnosť písma. Nachádza sa tu aj tlačítko na zmenu farby písma.

4.5 Testovanie

Najkritickejšia časť celej práce vzhľadom na chyby sú operačné transformácie. Preto sú testované automatickými testami. Testy sú robené na pre prípad operačných transformácií, kde jedna inštancia operačných transformácií plní úlohu servera a druhý úlohu klienta. Druhý typ testov sa líši od prvého typu tým, že pre úlohu klienta sú použité dve inštancie operačných transformácií, pre úlohu servera ostáva jedna inštancia. Tieto testy pokrývajú všetky možné problémy, ktoré sa mi podarilo nájsť.

Testy sú napísané v jazyku C++ za pomoci frameworku Qt. Pri spustení testov sa v zložke, v ktorej sa spúšťajú, vytvoria súbory typu `png`. Každý obrázok obsahuje priebeh jedného testu. Test je znázornený časovým diagramom, kde každá zvislá čiara označuje jedného klienta. Medzi klientmi je znázornený prenos správy, ktorý obsahuje jednoduchý popis naznačujúci, čo je úlohou správy.

V prípade kompilácie testov s direktívou `SHOW_DEBUG_WINDOW` s hodnotou `true` sa pri spustení testov otvorí pre každý test okno, kde sa v prípade presunutia kurzora myši zobrazí text. Tento text ukazuje podrobné informácie o vnútornom stave operačných transformácií v jednotlivých okamihoch, vďaka čomu je možné jednoduchšie nájsť prípadnú chybu v operačných transformáciách.

Testov je celkom 54. Z toho je 26 testov základných medzi klientom a serverom. Ďalších 10 testov sú medzi dvoma klientmi a jedným serverom. Ostatných 18 testov sú testy pre špeciálny typ neriadených operačných transformácií, vhodný pre siete bez servera. Tento špeciálny typ operačných transformácií sa zatiaľ nikde nevyužíva.

Pri formátovanom texte sa výpočet nemení, rozdiel je iba v spracovaní textu pri vkladaní. O túto funkciu sa stará framework Qt, a preto nemá význam robiť testy zvlášť na formátovaný text. Ak by nastal problém pri formátovanom texte, ktorý pri neformátovanom nenastáva, chybu treba opraviť priamo vo frameworku Qt.

4.6 Budúcnosť aplikácie

Aplikácia bude v budúcnosti obohatená o funkcie, ktoré budú napodobovať maximálne ako je to možné priamu fyzickú komunikáciu užívateľov bez špeciálneho vybavenia. Presnejšie ide o možnosť do editora kresliť a podpora video hovoru. Operačné transformácie budú optimalizované. Bude pridaná podpora chatovacích miestností, ktoré sú súčasťou protokolu XMPP. V budúcnosti aplikácia prejde aj modernizáciou užívateľského prostredia. Plánovaný je aj port na operačný systém Android. Testy budú rozšírené aj na užívateľské rozhranie.

Medzi plánované zmeny patrí aj možnosť správy práv. O správu práv užívateľov chatovacích miestností sa stará jabber server. Tieto práva budú aplikované aj na dokument, ktorý upravujú užívatelia v miestnosti.

Pri portovaní do operačného systému Android bude treba vytvoriť viac druhov užívateľských rozhraní. Jeden typ bude klasický pre desktopy, ďalší pre tablety, ktorý bude podobný desktopovému. Posledný typ je najnáročnejší. Jedná sa o užívateľské rozhranie pre inteligentné mobilné telefóny. Inteligentné Mobilné telefóny menia rozlíšenie obrazovky podľa svojho natočenia, preto treba vytvoriť užívateľské rozhranie zvlášť pre každý prípad.

Kapitola 5

Záver

Problematika interaktívneho zdieľania dokumentov je popísaná v kapitole 3. Pri úprave dokumentov v reálnom čase je potrebné použiť technológiu operačné transformácie. Táto technológia je implementovaná v rámci tejto práce a slúži na výpočet správnej pozície pre vkladanie textu.

Výber vhodného protokolu pre komunikáciu po počítačovej sieti je zdôvodnený v kapitole 2. Rozhodovanie prebiehalo medzi protokolom XMPP a vlastným aplikačným protokolom. Zvolený bol protokol XMPP pre jeho rozšíriteľnosť, štandardizáciu a možnosť používať už existujúce servery.

V tejto bakalárskej práci bol vytvorený editor zdieľaného formátovaného textu, ktorý využíva na komunikáciu po počítačovej sieti aplikačný protokol XMPP. Tento editor podporuje základné formátovanie textu. Možné je pripojiť viacerých užívateľov k úprave jedného dokumentu. Podrobné informácie o implementácií sú dostupné v kapitole 4. Jeho výhodou oproti konkurencii je možnosť upravovať editovateľný text v reálnom čase s prihlásením z ľubovolného jabber účtu. Užívatelia nie sú pri práci nútený čakať, a môžu zároveň upravovať dokument aj v prípade siete s väčšou latenciou.

V prvej verzii musí vlastník dokumentu pozývať účastníkov zdieľania k pripojeniu sa k dokumentu, ale v ďalšej verzii je naplánované zdieľanie v rámci MUC (multy user chat) miestnosti. Samotné operačné transformácie sa blížia k možnosti využívať v rámci sietí P2P, ale vzhľadom na problémy pri pripájaní a odpájaní užívateľov mám podporu takýchto sietí v pláne len obmedzene. Ostatné plány do budúcnosti sú popísané v kapitole 4.6.

Aktuálne je možné aplikáciu spustiť pod operačným systémom Windows 7 a Linux s nainštalovaným frameworkom Qt4.8 alebo Qt5. Jeho konkurentom je napríklad Google Docs, ktorý dokáže všetko, čo dokáže editor v rámci tejto práce implementovaný. Okrem toho má aj iné výhody. Avšak Jeho nevýhodou je možnosť prihlásiť sa len s účtom od firmy Google, čo radu užívateľov odradzuje. Ďalšia konkurencia je editor Gobby, ktorý však nepodporuje formátovaný text.

Literatura

[1] *Iain Shigeoka. Instant Messaging in Java - The Jabber Protocols. Manning Publicatins Co., 2002. ISBN 1-930110-46-4.*

[2] *J.F. Kurose, K.W. Ross: Computer Networking: A Top-Down Approach Featuring the Internet, Adison-Wesley, 2003.*

[3] *Molkentin, D: The Book of Qt 4: The Art of Building Qt Applications, No Starch Press, 2007, 440p, ISBN10: 1593271476, ISBN13: 978159327147.*

[4] *Molli, Pascal, Oster, Gerald, Skaf-Molli, Hala and Imine, Abdessamad: Using the transformational approach to build a safe and generic data synchronizer. In: Tremaine, Marilyn M. and Simone, Carla (eds.) Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work 2003 November 9-12, 2003, Sanibel Island, Florida, USA. pp. 212-220.*

[5] *W.R. Stevens, B. Fenner, A.M.Rudoff: UNIX Network Programming. The Sockets Network API, Addison-Wesley, 2004.*

[6] Jabber Software Foundation: *Extensible Messaging and Presence Protocol (XMPP): Core.*

[7] Jabber Software Foundation: *Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence.*

[8] Kozierok, C. M.: *The TCP/IP Guide: A Comprehensive, Illustrated Internet Protocols Reference.* ISBN 978-159327-047-6.

[9] Saint-Andre, P.; Smith, K.; Troncon, R.: *XMPP: The Definitive Guide.* ISBN 978-0-596-52126-4.

Příloha A

Obsah CD