Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

FIIT-5212-72359

Michal Slovík

ZDIEĽANÝ MOBIL

Bakalárska práca

Študijný program: Informatika

Študijný odbor: 9.2.1 Informatika

Miesto vypracovania: Ústav informatiky, informačných systémov a softvérového inžinierstva (FIIT)

Vedúci práce: Ing. Ján Lang, Phd.

Máj 2016

Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

ZDIEĽANÝ MOBIL

Študijný program:	Informatika
Autor:	Michal Slovík
Vedúci záverečnej práce:	Ing. Ján Lang, PhD.
Miesto a rok predloženia práce:	Bratislava 2016

Služby poskytované telekomunikačnými operátormi môžu byť náročné pre vybavenie koncových zariadení. Široká funkcionalita poskytovaná súčasnými zariadeniami ponúka komfort pri komunikácii, práci či zábave. Dostupnosť komunikačných rozhraní a alternatívnych komunikačných sietí umožňuje použitie týchto zariadení aj v tzv. zdieľanom režime.

Táto práca sa venuje analýze existujúcich aplikácií, ktoré umožňujú takéto zdieľanie telekomunikačných služieb. Analýza ponúka široký prehľad o súčasnom stave a predkladá nové možnosti zdieľania. Zo zistenej analýzy sa pokračovalo v navrhovaní a implementovaní vlastnej aplikácie, ktorá umožňuje zdieľanie telekomunikačných služieb, napríklad telefonovanie, odosielanie a prijimanie SMS správ. Dôraz je kladený na zdieľanie služieb mobilným zariadením s operačným systémom Android medzi viacerých klientov. Záverom práca ponúka ďalšie možnosti rozvoja aplikácie.

Kľúčové slová: zdieľanie, telekomunikačné služby, android, multi-klient

Slovak University of Technology in Bratislava
FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES

SHARED MOBILE

Study Programme:	Informatics
Author:	Michal Slovík
Supervisor:	Ing. Ján Lang, PhD.
Place and year of submission:	Bratislava 2016

It can be demanding to work with terminal equipment by telecommunication operators' services. The broad functionality provided by the current equipment offers a bigcomfort in communication, work and entertainment as well. Availability of alternative communication interfaces and communication networks allowus to use these devices also in a so-called "shared mode".

This work analyzes all the existing applications that allow sharing telecommunications services.The analysis provides a broad overview of the current state and presents new sharing options. It is then followed by a design and implementation of custom application that allows sharing the telecommunications services, such as: phone calls, sending and receiving SMS.The emphasis is put on sharing services mobile device with the Android operating system which allows sharing among multiple clients. In conclusion the thesis offers additional possibilities for the development of such applications.

Keywords: sharing, telecommunication services, android, multi-client

Vyhlásenie autora

Podpísaný Michal Slovík čestne vyhlasujem, že som bakalársku prácu Zdieľaný mobil vypracoval na základe poznatkov získaných počas štúdia a informácií z dostupnej literatúry uvedenej v práci.

Vedúcim mojej bakalárskej práce bol Ing. Ján Lang, PhD.

Bratislava, dňa 10.5.2016

.....

podpis autora

Poďakovanie

Chcem sa poďakovať vedúcemu bakalárskej práce, ktorým bol Ing. Ján Lang, PhD., za odborné vedenie, rady a pripomienky, ktoré mi pomohli pri vypracovaní tejto bakalárskej práce.

Obsah

Zoznam obrázkov a tabuliek

Zoznam skratiek a značiek

WWW - World Wide Web
SMS - Short message service
MMS - Multimedia Messaging Service
GSM - Global System for Mobile Communications
P2P - Rovný s rovným (angl. Peer to peer)
AP - Access Point
IPv6 - Internet Protocol version 6
IPv4 - Internet Protocol version 4
RTP - Real time protocol

Úvod

Zdieľanie je v dnešnom svete neoddeliteľnou súčasťou nášho každodenného života. Medzi priateľmi, ale aj v pracovnom prostredí chceme a potrebujeme zdieľať informácie. Odosielať správy, fotografie, hudbu, súbory do práce či rozpracované aplikácie.

Keď pracujeme spoločne na väčších či menších projektoch, je veľmi užitočné rozdeliť si prácu na menšie časti a takto môžu viacerí ľudia naraz pracovať na celej práci, a táto činnosť sa nazýva kolaborácia. Zdieľanie má svoje uplatnenie aj v prípade kolaboratívneho prístupu, kolaboratívnej tvorby a kooperatívneho prístupu pri vývoji softvéru. Vývoj softvéru je neodmysliteľne kolaboratívny proces. Medzi nástroje, ktoré vyvíjajú softvér kolaboratívnym prístupom patrí napríklad Git[1].

Ďalším veľkým priestorom pre zdieľanie dát je cloud[2] , ktorý spočíva v zdieľaní hardvérových a softvérových prostriedkov pomocou siete [23]. K takémuto cloud-u dokáže pristupovať akékoľvek podporované zariadenie. To znamená ak by šlo o aplikáciu napríklad Dropbox[3] dokážem pomocou počítača uložiť akékoľvek dáta na svoj cloud a tieto dáta dokážem neskôr prečítať aj z iného zariadenia, napríklad mobilu. Prístup k zdieľaným dátam a prostriedkom je v súčasnosti podporovaný aj smart mobilnými zariadeniami. Konkrétne na Slovensku sa výrazne zvýšila vybavenosť mobilnými telefónmi. Podľa výsledkov Sčítania obyvateľov, domov a bytov 2011 sa oproti roku 2001 zvýšil počet mobilných telefónov takmer o jeden milión[4]. Je teda praktickejšie zdieľať informácie, údaje a rôzne dáta priamo cez mobilné telefóny.

Priamo pri zdieľaní môžeme hovoriť o mnohých variantoch zdieľania. Prvé praktické využitie zdieľania, je samotné zdieľanie internetového pripojenia pomocou technológie hot-spot (preklad: prístupový bod), kde ako v prípade obyčajného prístupového bodu, vytvoríme uzol, ku ktorému je možné pripojiť sa ako ku hociktorému sieťovému adaptéru. Zároveň však tento mobilný telefón zdieľa svoje pripojenie na internet (napríklad cez mobilné dáta) do tohto uzla. Takýmto spôsobom môže mobilný telefón zdieľať svoje internetové pripojenie.

Ďalším príkladom je zdieľanie samotných dát vytvorených priamo v mobile, ako už bolo spomenuté existuje možnosť takéhoto zdieľanie cez cloud, ale pri takomto zdieľaní je potrebné mať internetové pripojenie. Čo ak by sme chceli zdieľať iba aktuálne vytvorenú fotografiu na svoj počítač a nie som práve pripojený na internet? Jednou z mnohých

[1] https://github.com/

[2] angl. "mrak" http://slovniky.lingea.sk/

[3] https://www.dropbox.com

[4] slovak.statistics.sk/wsp/portal

možností je zdieľanie celého mobilného telefónu do počítača pomocou aplikácie AirDroid[16]. Táto aplikácia dáva počítaču plnú kontrolu nad mobilným telefónom. Dokážeme posielať nielen dáta, ale aj SMS správy, poznámky či priamo telefonovať z počítača a teda aj prijímať hovory.

Analýza potenciálu mobilných zariadení vystrieda analýza existujúcich riešení. Z týchto existujúcich aplikácií vyšpecifikujeme najdôležitejšie vlastnosti, ktoré uplatňujeme pri našej aplikácii. V návrhu našej aplikácie uvádzame funkcionálne a nefunkcionálne požiadavky znázornené prehľadnými diagramami. Návrh vystrieda implementácia, ktorá opisuje nasadenie týchto požiadaviek do implementačného prostredia. V implementácii sa zamýšľame aj nad alternatívami nášho riešenia, ktoré by mohli poskytovať výhody aj nevýhody. Evalvácia testuje či stanovené požiadavky sú dostatočné pre používateľa, a zároveň definujú presne hranice možností našej aplikácie.

Záver práce dáva čitateľovi prehľad o celej práci, o jednotlivých krokoch vývoja aplikácie o jej schopnostiach a možnostiach. V závere sú uvedené aj alternatívne riešenia a možné doplnenia našej aplikácie.

1 Súvisiace práce

Slovo "zdieľanie" ako také je čechizmus odvodený od slovesa sdílet[5]. Pretože sa aj my budeme pokúšať zdieľať rôzne telekomunikačné služby (vytváranie a prijímanie hovorov, odosielanie a prijímanie SMS správ , ...) je vhodné oprieť sa o predošlé skúsenosti iných ľudí.

V tejto časti by sme si ukázali práce, ktoré úzko súvisia s našou témou, využívajú služby poskytované mobilnými operátormi na zdieľanie. Tieto práce sme kategorizovali podľa, toho čo presne zdieľajú a medzi akými zariadeniami.

Okrem všeobecných špecifikácií jednotlivých aplikácií sa budeme zameriavať na niekoľko konkrétnych bodov. Napríklad schopnosť zabezpečenia svojho zdieľania, či už šifrovaním alebo heslom. Ďalším faktorom bude rozšíriteľnosť softvéru, teda či aplikácia dáva voľnosť používateľovi v doplnení svojich predstáv. Niekedy rozhodujúcim kritériom je aj robustnosť, od ktorej sa odvíja aj cenová hladina softvéru, pričom na druhej strane robustnejšia aplikácia ponúka oveľa väčšie možnosti ako lacnejší a menší variant. Faktorom, pomocou ktorého sú rozdelené aplikácie, je rozsah zdieľania. Začneme zdieľaním súborov po sieti.

1.1 Zdieľanie súborov

Táto sekcia je venovaná odosielaniu a prijímaniu súborov pomocou rôznych alternatív. Tieto alternatívy sa snažia napodobniť počítačové verzie internetového sieťového pripojenia a s tým spojeného odosielania a prijímania súborov. Prvá alternatíva je peer-to-peer variant viac známy ako BitTorrent, kde každý uzol je zároveň príjemcom, ale aj odosielateľom dát.

1.1.1 Wireless Peer-to-peer zdieľanie súborov

Zdieľanie súborov pomocou Peer-to-Peer služby je veľmi populárne [9]. Ako uvádzajú Ze Li a Haiying Shen 50 % stiahnutých súborov a 80 % všetkých súborov, ktoré boli uložené na internet pracovali so službou Peer-to-peer [10]. So stúpajúcim využívaním mobilných telefónov[6] má táto technológia veľké uplatnenie. Virtuálna mobilná Peer-to-Peer sieť ponúka práve možnosť zdieľania súborov a dát medzi jednotlivými mobilnými zariadeniami [1].

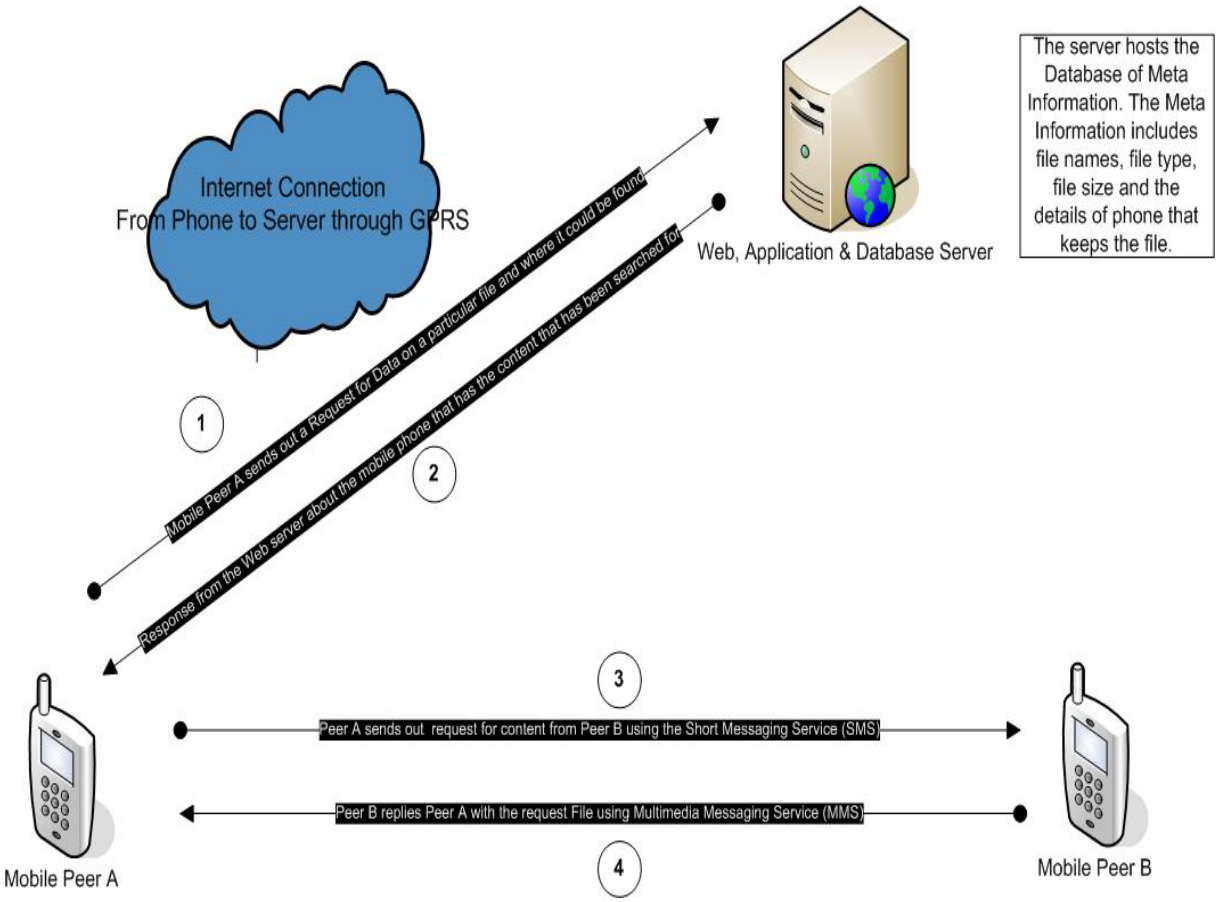
V mobilnom P2P(peer-to-peer) sieťovom modeli, ktorý opisuje skupina autorov [1] ,

[5]slovniky.lingea.sk

[6]slovak.statistics.sk/wsp/portal

pracujú s druhou generáciou (2 / 2.G) GSM mobilnou sieťou. Chceli napodobniť internetovú P2P(peer-to-peer) sieťovú variantu. Avšak boli obmedzení viacerými nedostatkami. Operátori zabraňujú využívanie mobilov ako samostatných jednotiek s IP adresami [13]. Operátori taktiež kontrolujú dáta, ktoré sú posielané. A v niektorých sieťach nie je možné odosielanie hlasu a dát paralelne [26]. Kvôli týmto obmedzeniam sa vyhli reprezentovaniu, kde by všetky telefóny boli prepojené priamo medzi sebou. Využili web server ako mobilného zástupcu (angl. mobile proxy). Tento server využíva databázu meta-údajov o súboroch a mobilných zariadeniach, ktoré sú v sieti. Ako komponenty pre odosielanie súborov sa využívajú SMS a MMS.

Ako funguje odosielanie súborov je znázornené na obrázku č. 1. Ako prvé odošle mobil A žiadosť o konkrétne dáta, a kde ich má hľadať. Web server mu odpovie, ktorý telefón obsahuje hľadaný súbor. Mobil A odošle žiadosť mobilu B, vďaka SMS, ktorý určil web server ako vlastníka hľadaného súboru a mobil B odošle pomocou MMS konkrétny súbor. Podobnou alternatívou je využite SIP protokolu pre podobné účely [9].



Obrázok 1: Odosielanie a prijímanie súborov použitím mobilu [1]

1.1.2 Využívanie SIP protokolu na zdieľania súborov

Ako v predošlom prípade pre zdieľanie súborov, tak v tomto prípade Liang Li a Xinshe Wang [9] využívajú P2P sieť implementovanú v 3G mobilnej sieti. Využívajú SIP protokol pre doručovanie kontrolných správ. Najskôr niekoľko slov o samotnom SIP protokole.

SIP je textový protokol odvodený od HTTP, teda je veľmi podobný webovej komunikácii [6]. Možno vzniká otázka,prečo je potrebné použiť tento protokol. Odpoveďou sú ASCII znaky, ktoré využíva SIP protokol vo svojich správach. Tým pádom nie je potrebné žiadne binárne konvertovanie ani dekódovanie, pretože SMS správy využívajú ASCII znaky [3]. Peer-to-peer metóda bola opísaná v predošlej podkapitole. Takže môžeme prejsť k samotnému odosielaniu súborov.

Každý mobil, ktorý chce odosielať a prijímať súbory, musí byť evidovaný na SIP serveri, ktorý je vedený internetovým operátorom [6]. Počas registračného procesu sa vymieňajú informácie aj o autentifikácii, aby bola celá komunikácia zabezpečená. Po tejto registrácii je možné zdieľať a sťahovať súbory. Pre zdieľanie súborov odošle žiadosť na centrálny server, ktorý je pod správou SIP servera. Pre sťahovanie súborov sa najskôr odošle žiadosť centrálnemu serveru a ten odpovie, kto vlastní súbor a teda ku komu má smerovať žiadosť o stiahnutie súboru [9].

Problémom pri tomto zdieľaní je samotný SIP protokol, keďže neobsahuje žiadosti na centrálny server a jeho odpovede, je potrebné hlavičku tohto protokolu doplniť o vlastné informácie.

V našej práci je užitočné vedieť, ako odosielať súbory, ale ak chceme zdieľať telekomunikačné služby (napr.: prijímanie a vytváranie hovorov, telefonovanie v reálnom čase, odosielanie SMS správ...) je potrebné uvažovať nad iným riešením. Najskôr chceme ukázať zdieľaniu zvuku, presnejšie hlasu.

1.2 Zdieľania hlasu a hlasových správ

Myšlienka vzniku posielania hlasu z jedného mobilu na druhý pomocou Bluetooth / Wifi je relatívne jednoduchá [22]. Hlas posielaný telefónom je už v tejto chvíli podporovaný službou GSM alebo aj internetovou službou, ktorá je lacnejšia, nazývaná VoIP[7]. VoIP podporuje viacero protokolov, ktoré sú od seba rôzne odlíšené. Zameriame sa na tie najpoužívanejšie.

1.2.1 Posielanie hlasu cez internetový protokol (VoIP) medzi mobilnými zariadeniami

Ako už bolo spomenuté, podpora odosielania a prijímania hlasu technológiou GSM [8] a 3G mobilná komunikácia [24] avšak za vyššiu cenu oproti IP telefonovaniu. V [22] si dali za úlohu vytvoriť rovnakú službu akú ponúkajú bežní operátori pre mobilné telefóny bezplatne. Autori tohto nápadu využívali Ad hoc a Peer-to-peer technológiu, ktorá používa Wifi sieť, teda vyhli sa použitiu centrálnej databáze, a tiež sa vyhýbajú potrebe registrovať sa do akejkoľvek inej služby.

Najskôr potrebujeme prepojenie medzi jednotlivými mobilmi. Toto spojenie je možné vytvoriť alebo pomocou bluetooth alebo bezdrôtovej siete. Bluetooth má určite obmedzenia, hlavne čo sa týka veľkosti rozšírenia, keďže jeho účinný dosah je maximálne 30 metrov bez prekážok [2]. Základný koncept, ktorý použili v [22] je založený na posielaní hlasu cez WLAN (angl. wireless local area network), to znamená, že komunikácia medzi dvoma telefónmi je bezplatná, ale vyžaduje si pripojenie oboch zariadení do bezdrôtovej siete. Systém , ktorý vytvorili v [22] dovoľuje vyhľadať iného používateľa v dosahu Wifi a nadviazať bezplatné VoIP hlasové spojenie. Taktiež umožňuje vytvoriť virtuálne spojenie cez AP (access point) a samozrejme ponúka možnosť použiť GSM v prípade, že žiadne Wifi pripojenie nie je možné.

Takže v prípade, že chce vykonať používateľ A hovor, najskôr sa prekonvertuje jeho číslo, a číslo na ktoré chce vykonať hovor na IP adresy (konkrétne IPv6) pomocou ich algoritmu [22]. Potom sa používateľ A pokúsi vytvoriť spojenie pomocou Peer-to-peer, pričom využíva ich unikátne IP adresy. Ak je používateľ B v dosahu bezdrôtovej siete, a potvrdí spojenie (teda príjme hovor), spojenie bude vytvorené. Ak by používateľ B nebol v dosahu bezdrôtovej siete, pokúsi sa vytvoriť AP a pomocou virtuálneho mostu vytvoriť spojenie. A ak by nebol používateľ B ani v tejto sieti, tak IP adresy sa zahodia

[7] angl. Voice Over Internet Protocol www.voip-info.org

[8] skr. Global System for Mobile

a vykoná sa bežný GSM hovor [22]. Takto dokážeme nadviazať spojenie, ale potrebujeme ešte odosielať a prijímať hlas. Potrebujeme teda konkrétny VoIP protokol.

V článku [22] na spracovanie hlasu využívali *J2ME development* architektúru. Táto architektúra známa aj ako *Java ME* pracuje s grafikou, zvukom, sieťou a používateľskými vstupmi a aj bezpečnostnými prvkami [15]. Pre zvukový vstup využíva MIDI, zvukový komunikačný protokol, ktorý dovoľuje počítaču ale aj iným zvukovým prístrojom komunikovať v reálnom čase prostredníctvom sériového rozhrania [8]. Týmto rozhraním dokážu získavať hlas z rozhrania mobilného telefónu pomocou mikrofónu a odoslať ho ďalšiemu telefónu, ktorý ho prezentuje na svojom rozhraní, najčastejšie reproduktor. Toto by nám vedelo pomôcť pri zdieľaní samotného hlasu medzi mobilmi, avšak my chceme zdieľať túto službu najskôr medzi počítačom a mobilom. A tak sa pomaly dostávame k UDP mediálnemu toku[9].

1.2.2 Zdieľanie hlasu naprieč rozdielnymi zariadeniami

Ako už názov prezrádza, budeme sa snažiť ukázať ako pracuje VoIP v spolupráci s rôznymi iným protokolmi, napríklad UDP alebo RTP. Tiež si predstavíme aplikácie, ktoré využívajú tieto protokoly na telekomunikačné služby.

VoIP komunikácia, ktorú opisujú [20], sa skladá z posielania signálov a média. Akékoľvek médium sa odosiela pomocou protokolov na odosielanie rozdielnych mediálnych prvkov ako zvuk alebo video, napríklad RTP alebo RTCP [19]. Signálové protokoly sú zodpovedné za vznik, zachovanie a rýchle uzavretie spojenia. Medzi hlavné signálové protokoly patria H.323, SIP/SDP (RFC3261 [17]) a XMPP/ Jingle [12]. V krátkosti si prejdeme ako fungujú samostatne a potom ako spolupracujú.

Najskôr teda protokol RTP[10], ktorý pracuje s dátami v reálnom čase ako napríklad zvuk alebo video. Tento protokol dopĺňa príbuzný protokol RTCP[11], ktorý je doplnený o interaktívnu charakteristiku, napr.: video konferencia alebo push-to-talk službu a používa na posielanie pravidelných kontrolných paketov. Oba tieto protokoly sú definované v RFC 3550 [19] a RFC 3551 [5]. To boli protokoly vyššej vrstvy, konkrétne aplikačnej. Oba tieto protokoly môžu pracovať aj nad UDP aj nad TCP protokolmi transportnej vrstvy [20].

Pre komunikáciu sa preferuje využitie UDP protokolu nad TCP protokolom, pretože odosielanie a prijímanie videa a zvuku sú "citlivé" na oneskorenia, ktoré vznikajú pri tom ako TCP nadväzuje spojenie a neustále kontroluje spojenie a každý chybu znova odosiela,

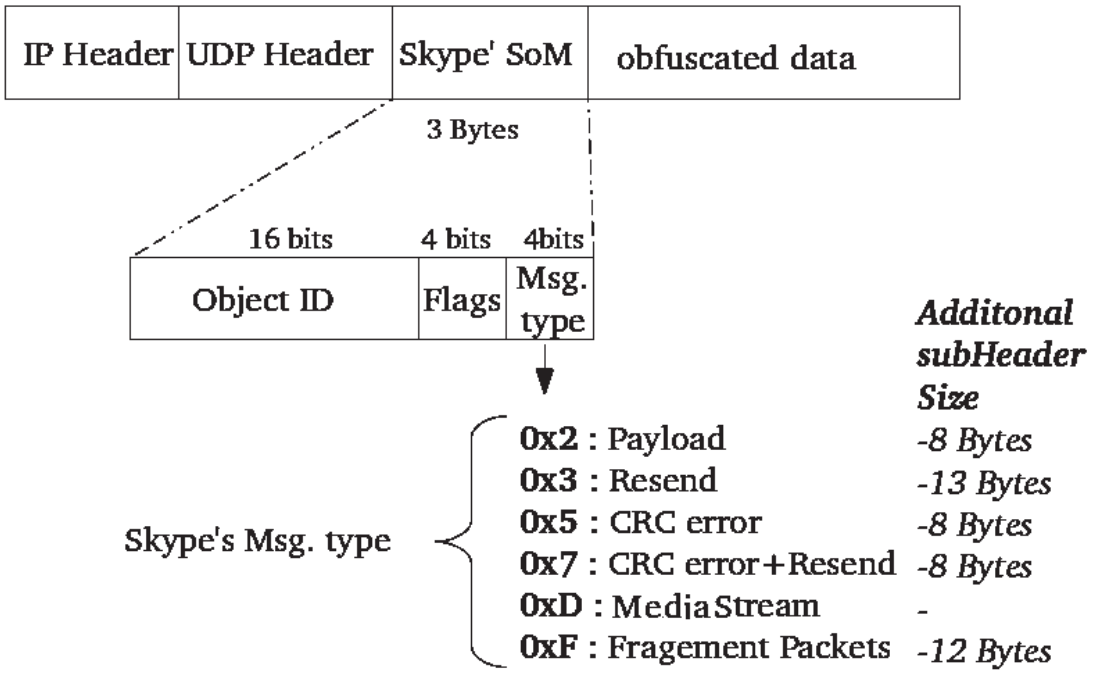
[9] angl. UDP media stream

[10] angl. real time protocol

[11] angl. real-time transport control protocol

ale hlavne TCP nepodporuje *multicasting* [20]. Iba čistý UDP protokol by nestačil, keďže je nespoľahlivý a nezaručuje doručenie všetkých paketov a ani ich odoslanie v správnom poradí. Tieto nevýhody riešia práve RTP a RTCP protokoly.

V aplikácii, ktorú opisujú [20], použili práve tieto protokoly RTP a RTCP, ale využili aj časť aplikácie Skype, ktorá používa na odosielanie mediálnych paketov vlastný zabezpečený protokol[12], pričom viditeľné sú iba prvé tri bajty jeho hlavičky, zvyšok správy je šifrovaný. Túto hlavičku je možné vidieť na obrázku č. 2. Práve tieto tri bajty využili z aplikácie Skype. Takýmto upraveným spôsobom, boli schopní vytvoriť tok (angl. stream) mediálnych dát buď pomocou RTP protokolu, alebo pomocou upraveného protokolu, ktorý využíva Skype [20]. Veľmi podobný spôsob by bol vhodný pre našu aplikáciu. Vytvoriť jednoduchý tok zvukových dát (hlas) z jedného uzla (napr.: počítač v našom prípade) a podobný tok opačným smerom. Takýmto spôsobom by bolo možné vykonávať telekomunikačnú službu z akéhokoľvek zariadenia, ktoré podporuje RTP alebo RTCP.



Obrázok 2: Hlavička paketu programu Skype [20]

Existuje oveľa viac VoIP protokolov[13], ktoré ponúkajú rôzne výhody aj nevýhody. Teraz si ideme ukázať aplikácie, ktoré sú nasadené do bežného prostredia.

[12] www.skype.com

[13] http://searchunifiedcommunications.techtarget.com/tutorial/VoIP-protocol-listing-A-glossary-of-VoIP-protocols-and-standards

2 Existujúce aplikácie a nástroje zdieľania služieb a prostriedkov

V tejto kapitole sa budeme venovať využívaniu aplikácii na zdieľanie služieb v telekomunikácii (napr.: Skype, AirDroid, GoogleTalk, ...) alebo zdieľanie prostriedkov, pričom za prostriedky rozumieme diskové miesto, ale aj vzdialené rozhranie, t.j. zdieľanie obrazovky (WhatApp, JoinMe, TeamViewer) či zdieľanie diskového priestoru (GitHub, DropBox, Mega ...) .

V nasledujúcej časti si uvedieme aplikácie, ktoré sprostredkujú hovor či už medzi mobilnými zariadeniami navzájom alebo medzi počítačom a mobilom.

2.1 Zdieľanie prostriedkov mobilnými telefónmi

Aj keď zdieľanie prostriedkov nepatrí priamo do množiny záujmov tejto práce, mobilné aplikácie, ktoré pracujú s týmito prostriedkami môžu používať podobné postupy pre komunikáciu či dokonca riešenie problémov zdieľania.

Medzi zdieľanie prostriedkov zahŕňame zdieľanie sieťového pripojenia, či zdieľanie výkonu alebo pamäťového priestoru. V našom prípade môžeme zdieľať GSM sieť pre telefonovanie. Počítač môže zdieľať svoju internetovú sieť a poskytovať ju mobilu.

Samotný mobil môže prijímať rôzne zdieľane prvky ako napríklad vzdialené diskové miesta a pristupovať k nim pomocou FTP[14] pripojenia [25], ale dokáže aj zdieľať svoju obrazovku a s touto zdieľanou obrazovkou dokáže pracovať iné zariadenie.

2.1.1 JoinMe

Aplikácia, pomocou ktorej vieme zdieľať obrazovku je JoinMe [11]. Hlavnou výhodou tejto aplikácie je schopnosť podporovať spojenie viacerých používateľov medzi sebou, takzvaný *multiklient* (až 250 používateľov pri Enterprise verzii [11]). Dokáže pracovať naprieč rôznymi operačnými systémami, takže nie je problém zdieľať pracovnú plochu počítača a pracovať s ňou v mobile.

Samozrejme na použitie všetkých týchto služieb (okrem Trial verzie), ktoré poskytuje táto aplikácia, je potrebné sa zaregistrovať a používať ich stránku a tiež je potrebné na základe toho, do akej miery budeme túto aplikáciu využívať, uhradiť čiastku na to

[14] File Transfer Protocol

potrebnú [11]. Obrázok č.3 znázorňuje prihlasovací formulár, pomocou ktorého sa prihlásime do existujúcej služby zdieľanej obrazovky(značené zelenou), alebo vytvoríme nové vlastné zdieľanie obrazovky (označené červenou).

Zo sledovaných vlastností táto aplikácia nemá možnosť rozšírenia. Znovupoužiteľnosť tejto aplikácie je teda veľmi malá. Zabezpečenie je formou 256-bitového SSL šifrovania a zaručuje, že žiadne dáta nie sú počas hovoru ukladané na server. Vzhľadom na robustnosť aplikácie je zaručená jej spoľahlivosť aj pri vyššom zaťažení.

Obrázok 3: V programe JoinMe máme možnosť vybrať si medzi vytvorením pripojenia alebo pripojením sa k existujúcej komunikácii [11]

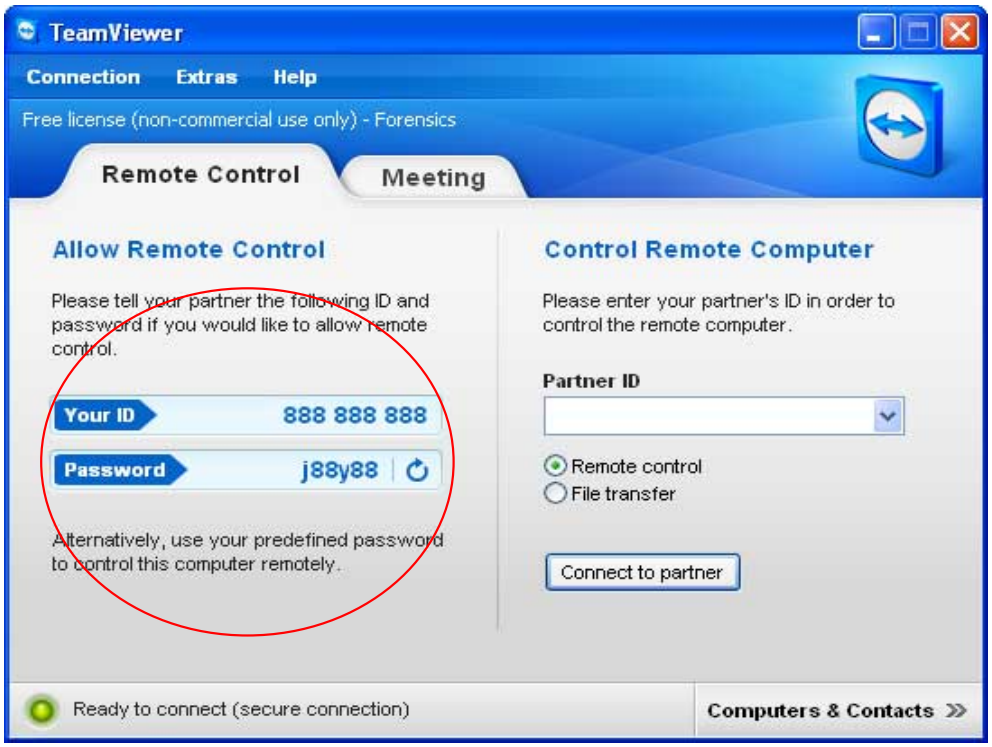
2.1.2 TeamViewer

Ďalším užitočným nástrojom pre zdieľanie prostriedkov je TeamViewer [15] , ktorý okrem zdieľania dát a informácií sprístupňuje aj vzdialené ovládanie cudzieho počítača, teda znova ide o zdieľanie pracovnej plochy a jej využívanie iným používateľom [16]. Medzi zdieľaním prostriedkov treba aj predstaviť zdieľanie súborov, ktoré tieto programy podporujú, avšak rozšírenejšia alternatíva je využitie Cloudových úložísk [23].

Obrázok č.4 ukazuje, ako je možné zdieľať obrazovku na rôznych zariadeniach. V prípade, že chceme vytvoriť zdieľanie svojej plochy, musíme zadať ID tejto relácie a prístupové heslo tejto relácie (na obrázku znázornené v červenom kruhu). V prípade, že sa chceme pripojiť k relácii zdieľania obrazovky, zadáme ID relácie, ktorej obrazovku chceme zdieľať a nastavíme režim (buď vzdialené ovládanie, prezentovanie alebo

[15]http://www.teamviewer.com/en/
[16]http://www.teamviewer.com/en/

odosielanie súborov) a následne bude potrebné zadať heslo relácie.



Obrázok 4: Zobrazenie možností vytvorenia pripojenia alebo pripojiť sa k aktuálnemu [4]

Táto aplikácia sleduje ako hlavný cieľ zdieľanie obrazoviek. Takže ako hlavný faktor má vysokú prenosnosť na rôzne zariadenia s rozličnými operačnými systémami. Zabezpečenie celej aplikácie je taktiež dosť podstatný prvok vzhľadom na zdieľanie obrazovky. Využívajú RSA 2048(verejný/privátny kľúč) a AES-256 [17] bitové na šifrovanie [4]. Vzhľadom na tieto vlastnosti je aj softvér patrične robustný a málo znovupoužiteľný.

2.1.3 GitHub

Medzi najzaujímavejšie zdieľané úložisko pre spoločnú tvorbu softvéru je nesporne Git, čo je viditeľné už len tým, že až 30 miliónov projektov bolo vytvorených aj za pomoci tohto zdieľaného prostriedku[18]. Čo sa týka znovupoužiteľnosti je Git vynikajúci nástroj, ktorý ponúka rozsiahle možnosti spolupráce a vývoja.

Samozrejme existujú aj iné, jednoduchšie alternatívy (DropBox, Mega, OneDrive

[17] Advanced Encryption Standard
[18] https://github.com/

a iné). Avšak praktické využitie Cloudového úložiska je pre našu konkrétnu potrebu minimálne. Ešte sa zastavíme pri jednom úložisku a to konkrétne Mega[19]

2.1.4 Mega

Ide o začínajúce cloudové úložisko (2013), ktoré ponúka možnosť pripojiť sa k tvorbe tohto softvéru. Avšak oveľa zaujímavejšie sú výhody tejto aplikácie. Za najpodstatnejší prvok si vzali za úlohu bezpečnosť dát [20]. Ak teda chceme uložiť nejaké dáta na *cloud* a odoslať ich inej osobe. Dáta sa zašifrujú, a spolu s odkazom na dáta mu musíme poslať aj kľúč na dešifrovanie týchto dát. Pričom celá táto činnosť neobmedzuje používateľa pri práci. Čo sa týka prenosnosti na niektorých platformách (iOS, Android) na tých sa ešte stále pracuje.

2.2 Zdieľanie telekomunikačných služieb mobilného telefónu

Teraz sa pozrieme detailnejšie na aplikácie, ktoré využívajú telekomunikačné služby. Denne používanou telekomunikačnou službou je obyčajne telefonovanie. Uvažovať nad bežnou GSM službou poskytovanou našimi operátormi nie je účelom ani cieľom tejto práce, preto by sme sa venovali iným ako klasickým riešeniam.

Ako prvú aplikáciu pre prijímanie a vytvárania hovorov si uvedieme Skype. Táto aplikácia bola už spomína aj v predošlej kapitole 1.2.2.

2.2.1 Skype

Program Skype slúži na komunikovanie medzi ľuďmi navzájom, dokáže zdieľať zvuk a aj video[21]. Poskytuje tak pohodlné zdieľanie telekomunikačných služieb. Pracuje na princípe P2P medzi používateľmi, a tak prevádzkuje VoIP, vďaka ktorej vie vykonávať hovory z počítača aj do telefónnej siete, avšak táto služba je prevádzkovaná za poplatok [22]. Samotný softvér nepodporuje znovupoužitie. Bezpečnosť pri tejto aplikácii ostáva otázna, keďže neuvádzajú šifrovanie alebo inú metódu na zabezpečenie dát.

Taktiež spoplatnená služba je pridelenie telefónneho čísla používateľovi pri počítači a je možné sa dovolať na toto číslo z mobilnej siete [23]. Presne k tomuto sa chceme dopracovať

[19] https://mega.nz
[20] https://mega.nz/about
[21] www.skype.com
[22] pozri 18
[23] pozri 18

v našej práci, ale chceme, aby samotné poskytovanie tejto služby bolo bezplatné (hovor v mobilnej sieti spoplatnený je). Ďalší program nevyžaduje spoplatnenie služby vykonania hovorov, ale vyžaduje vytvorenie spojenia buď v jednej sieti, alebo pripojenie k internetu.

2.2.2 AirDroid

AirDroid ponúka zdieľanie celého mobilné zariadenia, zdieľanie hovorov, zdieľanie SMS správ, súborový systém, vzdialené ovládanie plochy (táto možnosť je už spoplatnená) [18]. Existuje niekoľko možností ako sa pripojiť s počítačom ku mobilu. Prvou je iniciácia z mobilu, ktorý začne vysielať na konkrétnu IP adresu a port, na ktorý sa môže počítač pripojiť pomocou webového prehliadača, nakoniec je potrebné zadať heslo, pre zabezpečenie spojenia [7]. Ďalšou možnosťou je prečítanie QR[24] kódu, ktorý vygeneruje webový prehliadač pomocou stránky web.airdroid.com a spojenie je potrebné dokončiť potvrdením vo webovom prehliadači [18]. Poslednou zaujímavou vecou je zdieľanie obrazovky jedna k jednej pomocou AirMirror služby, ktorá vyžaduje mať "rootnutý" mobil.

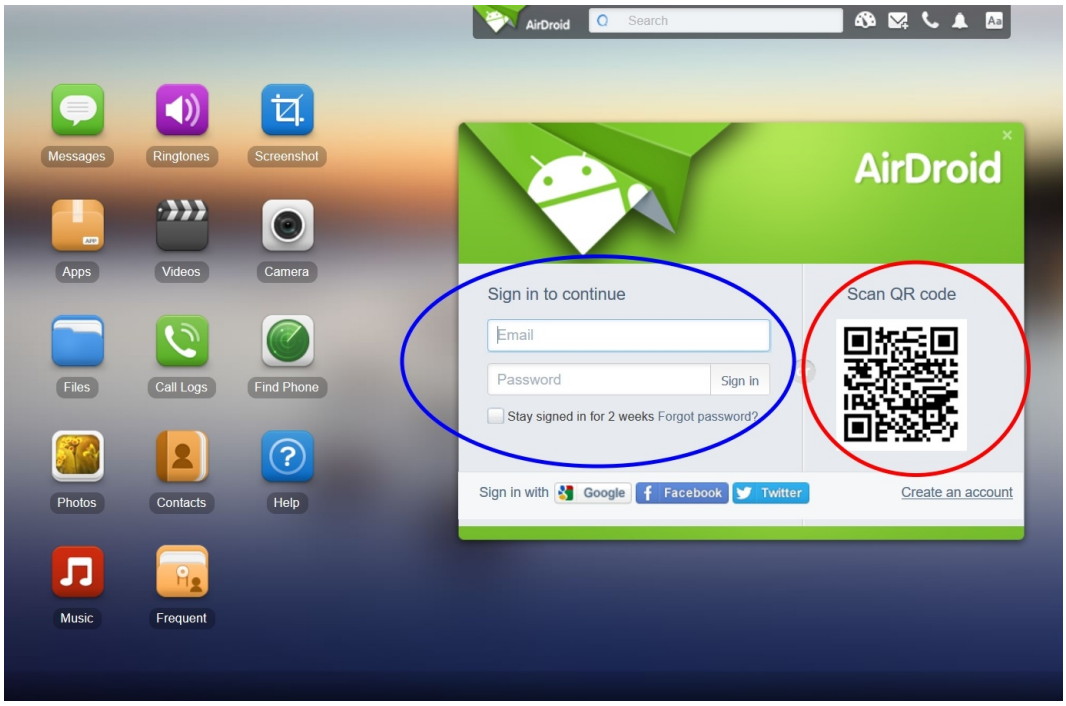
Mať "rootnutý" mobil znamená, že používateľ má úplný prístup k operačnému systému svojho telefónu, analógia s administrátorskými právami [21], avšak je tam riziko straty záruky stanovenej výrobcom. Na obrázku č.5 je možné vidieť ako vyzerá prihlasovací formulár pre aplikáciu vo webovom prehliadači. Máme na výber prihlásiť sa buď pomocou mena a hesla a podobne urobiť v mobile. Druhou možnosťou prihlásenia je prečítaním QR kódu pomocou mobilu (červenom krúžku). Po prečítaní nastane automaticky prihlásenie. Pri vykonaní hovoru však narazíme na závažný problém a to je nepodporovanie zariadení. Táto aplikácia [18] bola vyvinutá a pracuje na počítačoch s operačným systémom Mac OS X a mobiloch iOS, kde nie je problém so zdieľaním hovoru. Ale pri Androidoch alebo aj iných zariadeniach síce hovor vzniká, ale presmerovanie hlasu sa po zdvihnutí hovoru preruší a teda hovor nie je presmerovaný. Tejto téme sa budeme venovať v ďalšej kapitole.

2.2.3 Signal

Voľne šíriteľná aplikácia pre Android, ktorá poskytuje telekomunikačné služby hovorov a odosielanie a prijímanie textových alebo multimediálnych správ [14]. Pre samotné použitie nevyžaduje registráciu, potrebuje iba telefónne číslo kvôli presmerovaniu. Táto aplikácia dokáže vytvárať a prijímať hovory, odosielať a prijímať SMS alebo MMS správy. Výhodná je bezpečnostná komunikácia textových a multimediálnych správ, ktoré sú šifrované AES[25]

[24] Quick Response - prekl. kódy rýchlej reakcie, www.collinsdictionary.com

[25] Advanced Encryption Standard, http://www2.fiit.stuba.sk/ lhudec/PIS/pis.htm



Obrázok 5: Zobrazenie možností prihlásenia do služby AirDroid [18]

šifrou [14].

Aplikácia je do veľkej miery znovupoužiteľná. Nejde o robustný systém, čo môže pri vyššej zátaži robiť problém a nemusí byť úplne spoľahlivá. Najväčším negatívom je viazanosť na operačný systém Android. Ďalšia aplikácia sa snaží simulovať využitie štýlu podobnému odosielaniu klasických SMS správ.

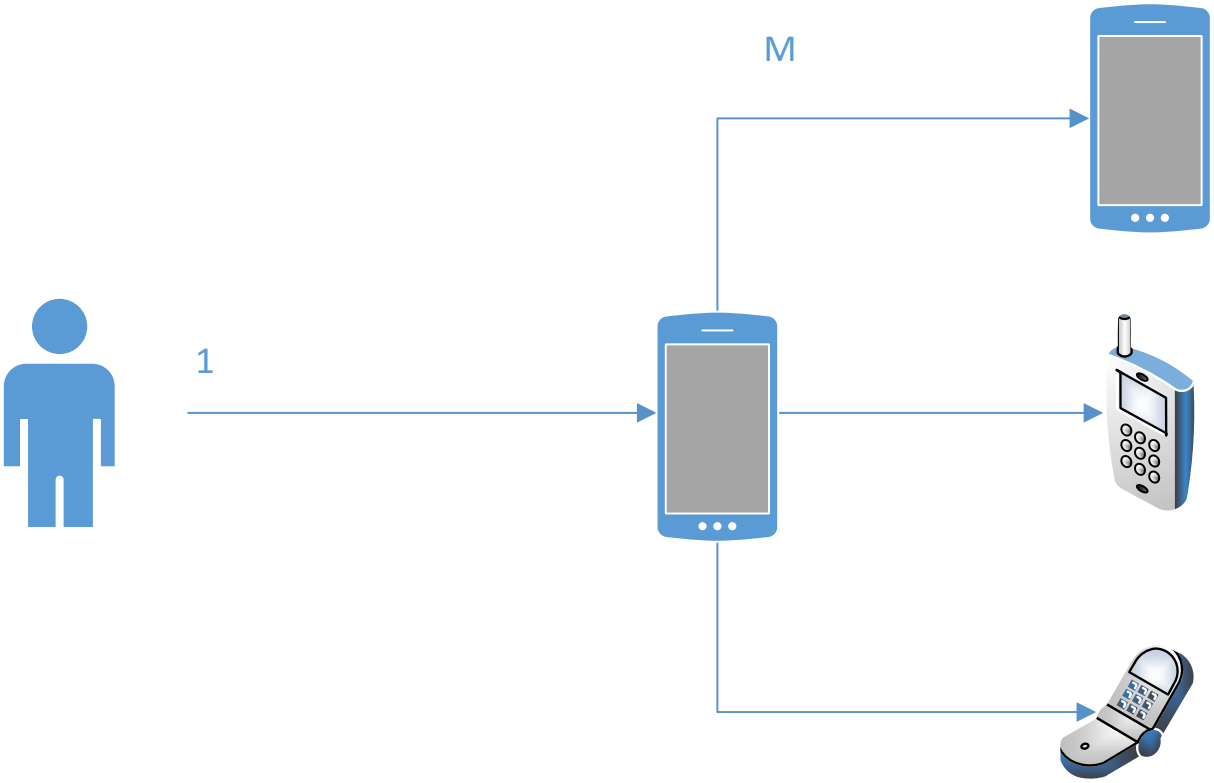
2.2.4 Messenger

Ide o aplikáciu od firmy Facebook[26]. Pre svoje fungovanie vyžaduje vlastnenie účtu na facebook-u a používateľ sa prihlasuje svojím účtom. Aplikácia umožňuje používateľovi dať pocit akoby osobe odosielal SMS správu, ale v skutočnosti iba posiela správu cez chat na facebook-u. Výhoda je aj možnosť hovorov. Takýmto spôsobom sa im podarilo úspešne nahradiť telekomunikačné služby, takže používatelia nepotrebujú využívať GSM sieť, stačí im pripojenie k internetu.

[26]https://www.messenger.com/features

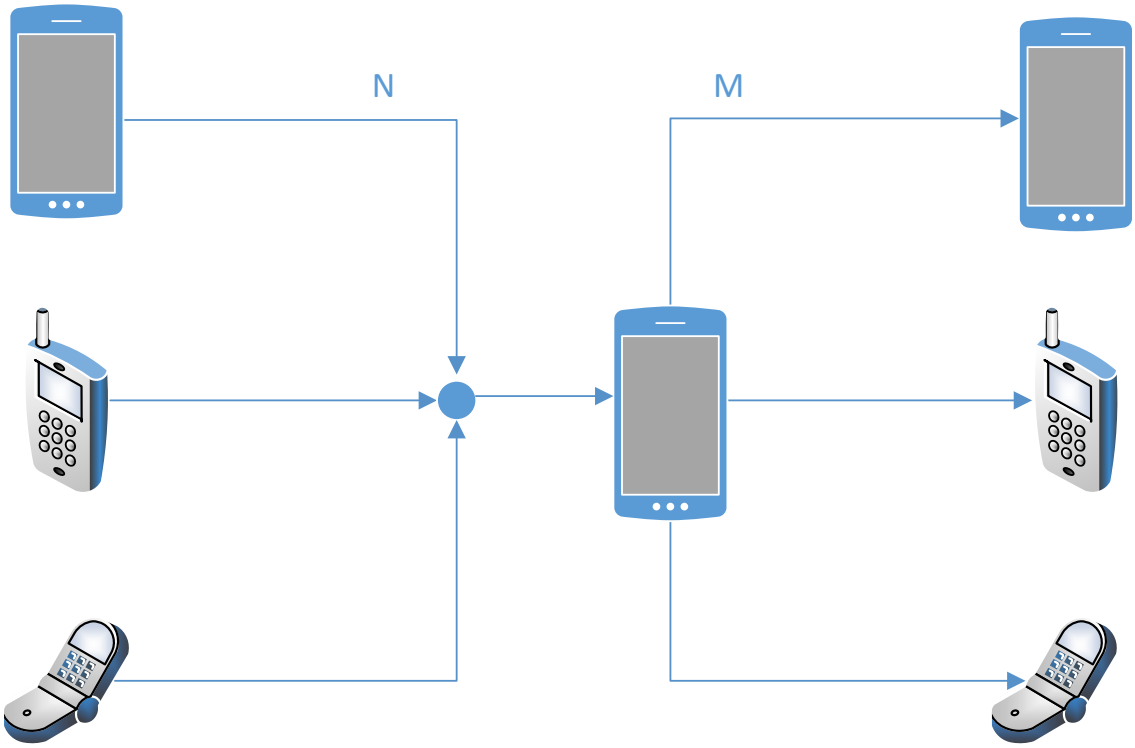
2.3 Zhodnotenie jednotlivých aplikácií

Všetky tieto aplikácie umožňovali rôzne druhy zdieľania, avšak niektoré mali určité nevýhody, napríklad za ich službu bolo potrebné platiť (Skype volania do mobilnej siete). Sledovali sme u týchto aplikácií ich možnosť znovupoužiteľnosti, čiže rozšírenia pre ďalšie použitie, bezpečnostné prvky týchto aplikácií. Zameriavali sme sa aj na spoľahlivosť týchto systémov. V neposlednom rade nás zaujíma prenosnosť, teda možnosť použitia aplikácie na rôznych zariadeniach. Ďalší je viditeľný rozdiel medzi aplikáciami, konkrétne v architektúre. Rozlišujeme dve základne, prvá (označíme písmenom A), ktorá zdieľa mobilné služby 1 ku viacerým (m) Obrázok č.6. Druhá (označíme písmenom B), ktorá zdieľa mobilné služby viacerých ku viacerim (n ku m) Obrázok č.7. Objekt znázornený v strede v oboch obrázkoch znázorňuje mobil alebo server, ku ktorému pristupujú.



Obrázok 6: Znázornenie architektúry typu A

Preto sme porovnali tieto aplikácie, tabuľka č. 1. Vieme, že každá z nich ponúka možnosť zdieľania služieb alebo prostriedkov. Rozdielne sú v viacerých parametroch: cena



Obrázok 7: Znázornenie architektúry typu B

za poskytovanie služieb, operačný systém, na ktorom dokážu pracovať, a ktorú architektúru využívajú v rámci svojho zdieľania, bezpečnostný prvok, a jej znovupoužiteľnosť.

Tabuľka 1: Porovnanie jednotlivých aplikácií

Názov	**JoinMe**	**TeamViewer**	**GitHub**	**Skype**	**AirDroid**	**Signal**
Bezplatne využívanie	X	X	X	X	X	X
Platené služby	X	X		X		
Windows	X	X	X	X		X
Mac OS X	X	X	X	X		
Linux		X	X	X		
Android	X	X	X	X		
iOS	X	X	X	X	X	
Windows Phone			X	X		
Architektúra(A / B)	typ B	typ B	X	typ A	typ A	typ A
Bezpečnosť	256-bitové SSL	AES-256	žiadne zistené	nezistené	nezistené	AES
Znovupoužiteľnosť	žiadne	žiadne	možné	žiadne	možné	možné

Tieto aplikácie teda poskytujú zdieľanie prostriedkov alebo služieb, ale často krát za

poplatok a za nutnosť prihlásenia do centrálneho servera. Iba dve aplikácie (JoinMe a TeamViewer) dokážu pracovať v architektúre typu B, to znamená, že vedia viacerí používatelia pristupovať k jednému zariadeniu (serveru) a toto zariadenie dokáže zdieľať svoje služby viacerým používateľom.

Ostatná väčšina analyzovaných aplikácií pracovala s architektúrou A. Napríklad aplikácia AirDroid dokáže prijímať zdieľanie iba od jedného mobilného zariadenia, ale dokáže cez neho využívať služby poskytované m používateľom. Samotné zdieľanie hovoru je pri AirDroide možné iba pre mobily s operačným systémom iOS.

Obe tieto architektúry by sme chceli využiť pri našej aplikácii. Ako počiatočnú by sme brali architektúru typu A, pričom by bolo možné rozšírenie aj o architektúru typu B. Cieľom tejto práce by bolo vytvorenie aplikácie, ktorá by umožňovala zdieľanie telekomunikačných služieb ako tieto aplikácie. Zároveň by sme sa pokúsili o bezplatné využitie takejto aplikácie, nakoľko značná časť týchto aplikácií svoje služby ponúka za poplatok. Aplikácia by mala byť dostatočne zabezpečená pred nežiadaným únikom informácií alebo narušením komunikácie.

2.4 Hardvérové obmedzenie presmerovania hovoru

Ako sme spomínali pri aplikácii AirDroid [14], sme narazili na závažný problém. Presmerovanie zvuku sa po začatí hovoru prerušilo, resp. ani nevykonalo. Narazili sme na problém, presmerovania hovoru v mobile. Abstraktná vrstva (obrázok v prílohe A), ktorá je zodpovedná za presmerovania výstupu z mikrofónu a vstupu do reproduktora sa pod operačným systémom Android nazýva RIL[27]. RIL daemon a Vendor RIL sú zodpovedné za komunikáciu s GSM. A tu vzniká náš problém, že väčšina Androidov má túto vrstvu neprístupnú, alebo je potrebné ju nahradiť inou. Pri operačnom systéme iOS pravdepodobne nahradzujú túto vrstvu vlastnou implementáciou a preto dokážu zdieľať pohodlne telekomunikačné služby.

Operačný systém Androidu ponúka možnosť pristupovať k tejto vrstve, ale iba v obmedzenej miere. Jeho trieda *android.telephony* ponúka možnosť vytočiť telefónne číslo alebo zisťovať aktuálny stav telefónu[28], ale zatiaľ neumožňuje presmerovanie mikrofónu a reproduktoru.

Iná trieda, konkrétne SmsManager[29] umožňuje pristupovať k tejto abstraktnej vrstve. Takýmto spôsobom dokážeme teda odosielať SMS správy rozličných veľkostí na rôzne čísla. V našej práci sa budeme venovať práve tejto telekomunikačnej službe.

[27] http://www.netmite.com/android/mydroid/development/pdk/docs/telephony.html
[28] http://developer.android.com/reference/android/telephony/package-summary.html
[29] http://developer.android.com/reference/android/telephony/SmsManager.html

3 Špecifikácia cieľov

Táto kapitola sa bude zaoberať definovaním cieľov, ktoré sa táto práca bude snažiť naplniť. Hlavným cieľom je zdieľanie služieb poskytovaných telekomunikačnými operátormi. Ako bolo uvedené v kapitole Existujúce aplikácie a nástroje zdieľania služieb a prostriedkov, tak takéto aplikácie existujú a sú voľne dostupne. V našej práci by sme chceli vylepšiť vlastnosti, ktoré tieto aplikácie majú. Zjednotiť ich pozitíva a snažiť sa čo najviac eliminovať ich nevýhody.

Medzi základné ciele patrí jednoduchosť ovládania a prístupu. To znamená aplikácia musí byť čo najjednoduchšia pre používateľa a dokáže intuitívne pracovať v grafickom rozhraní. K tomuto má napomôcť aj používateľská príručka a help umiestnený priamo v aplikácii.

Cieľu, ktorému sa niektoré aplikácie vyhli je architektúra *multiklient*. Výhodou takéhoto prístupu je schopnosť servera obsluhovať viac klientov naraz. Server by mal dokázať všetky tieto požiadavky obsluhovať bez zbytočnej odozvy alebo prípadnej chybovosti(vynechanie textu a podobne).

Bezpečnosť, ktorá je v dnešnej dobe čoraz viac oceňovaná by bola ďalším cieľom, ktorý by mala naša aplikácia spĺňať.

Nemalý podiel je aj samotný kód, ktorý by mal byť čo najviac udržateľný a znovupoužiteľný. Teda aby aj iný programátor mal možnosť doplniť vlastnú funkcionalitu podľa potreby. Tento cieľ by sme chceli naplniť použitím návrhových vzorov (Singleton) a architektonických štýlov (MVC) a využívaním existujúcim frameworkov[30] (JavaFX).

K tomuto cieľu prispieva aj možnosť pristupovať priamo ku kódom aplikácie. Obe časti aplikácie budú sprístupnené bez poplatkov pomocou Github [31]. Mobilná aplikácia bude dostupná aj pomocou internetového obchodu *Google play* [32] taktiež bezplatne.

[30]http://www.thefreedictionary.com/framework

[31]https://github.com/Mishco/BC_app_2

[32]https://play.google.com/store/apps/details?id=com.bc.michal.sms_chatpc#details-reviews

3.1 Stanovenie požiadaviek

- Spoľahlivosť celej aplikácie(serverovej aj klientskej časti), aby systém poskytoval kedykoľvek rovnaké výsledky pre zadané vstupy,
- Znovupoužiteľnosť, aby bola aplikácia schopná pripojiť k sebe ďalšie komponenty,
- Možnosť zapojenia viacerých klientov na jeden server, architektúra multiklient,
- Bezpečnosť celej komunikácie, či už pomocou zabezpečených protokolov, ale aj šifrovaním komunikácie (RSA),
- Umiestniteľnosť, ľahkosť s akou sa systém umiestňuje do konkrétneho prostredia,
- Dostupnosť, aplikácia musí byť dostupná bezplatne a jednoducho,
- Jednoduchá inštalácia, najvýhodnejšie pre používateľa je rýchlo a jednoducho pristupovať k aplikácii

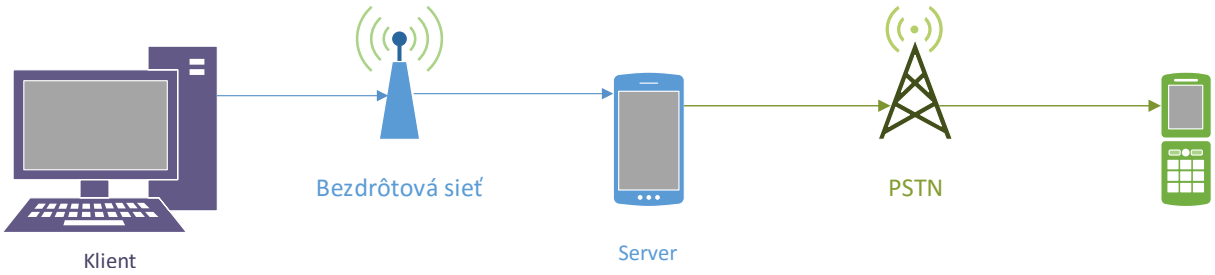
4 Návrh

Tato kapitola sa zaoberá návrhom celej aplikácie a jej zložením z dvoch častí (klient, server). V jednotlivých kapitolách si ukážeme aj diagramy pre konkrétne časti. Pre jednoduchosť nebudeme uvádzať všetky diagramy, ktoré by mala návrh softvéru obsahovať. Zameriame sa na zaujímavé časti.

4.1 Návrhový model

Ako už bolo spomínané, naša aplikácia sa bude skladať z dvoch častí. Pôjde o architektúru klient-server(angl. Client-Server) Teda o synchrónnu komunikáciu medzi desktopovou aplikáciou (počítač) a mobilnou aplikáciou. S tým, že v našom prípade budeme uvažovať aj o viacerých klientoch. Budeme uvažovať aj nad zapojením viacerých mobilných koncových zariadení, ktoré by mali zvýšiť spoľahlivosť systému (pri väčšom množstve požiadaviek možná distribúcia a prerozdelenie týchto požiadaviek).

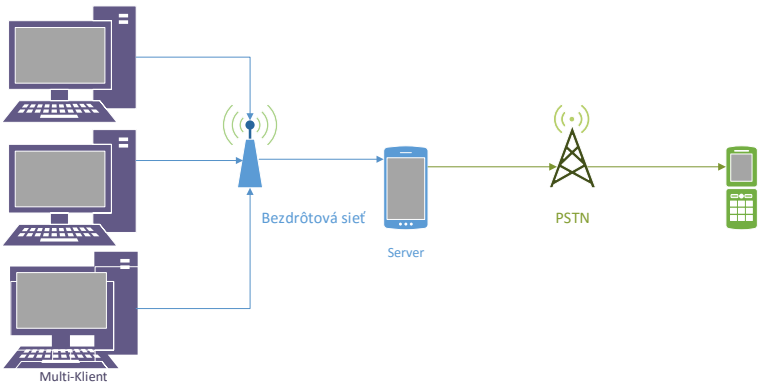
Obrázok č. 8 nám zobrazuje návrh aplikácie v štýle klient-server. Komunikácia medzi klientom a serverom prebieha pomocou bezdrôtovej siete(Wifi) a mobil posiela požiadavky ďalej pomocou PSTN (angl. public switched telephone network[33])



Obrázok 8: Návrhový model aplikácie v štýle klient-server

Avšak tu nekončíme, pretože chceme rozšíriť bežné aplikácie, ktoré dokážu takto pracovať. Preto rozšírime možnosť zapojenia viacerých klientov k jednému serveru (obrázok č. 9). A ďalšou alternatívou by bolo zapojenie viacerých serverov, ktoré by si medzi sebou posielali požiadavky v prípade väčšej výpočtovej záťaže.

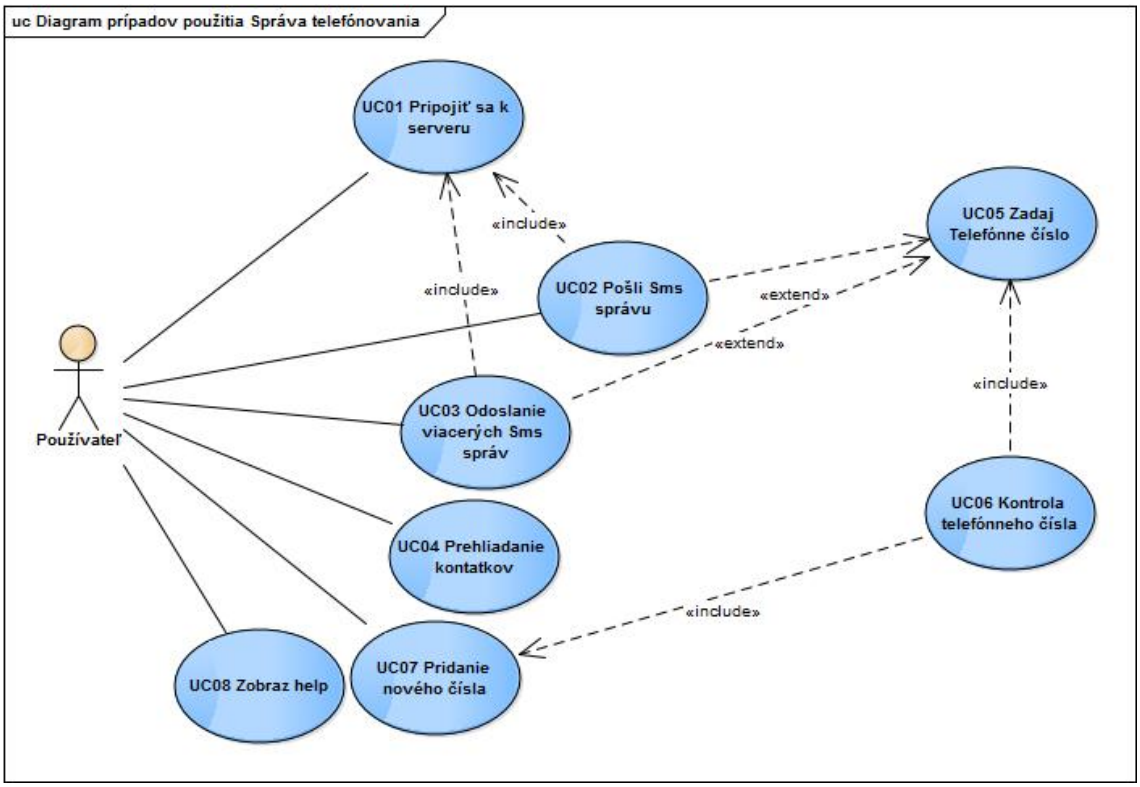
[33]http://searchnetworking.techtarget.com/definition/PSTN



Obrázok 9: Návrhový model aplikácie v štýle multiklient-server

4.2 Funkcionálne a nefunkcionálne požiadavky

Funkcionálne požiadavky sú činnosti, ktoré aplikácia poskytuje, sú uvedené na diagrame prípadov použitia (obrázok č. 10), ktorý opisuje jednotlivé činnosti aplikácie. Prípady použitia ukazujú rôzne možnosti dostupné pre používateľa a opisujú správanie systému v rôznych situáciách.



Obrázok 10: Diagram prípadov použitia pre zdieľanie telekomunikačnej služby

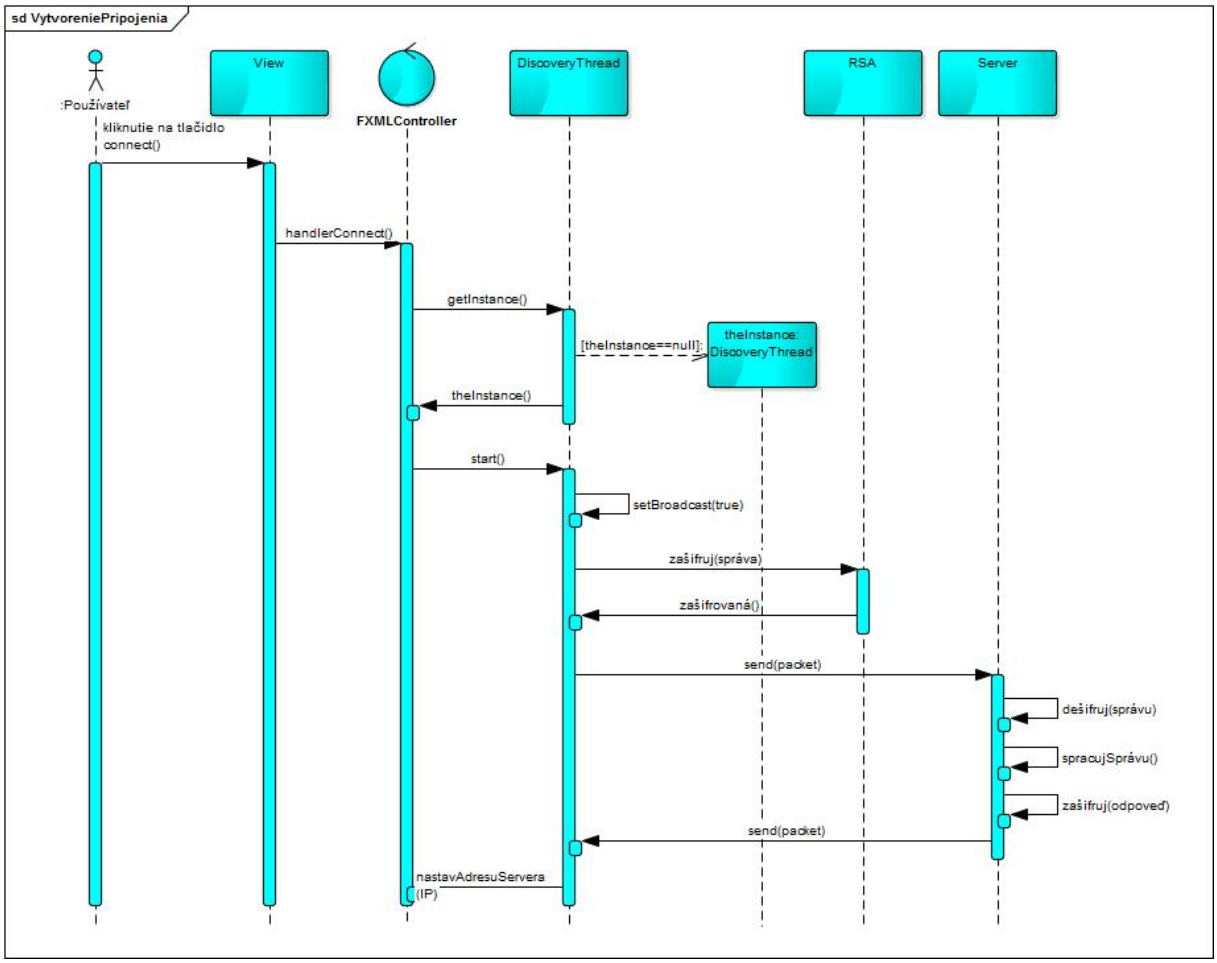
Medzi funkcionálne požiadavky patrí:

- Pripojenie sa k serveru, ktoré je nutné k ďalším činnostiam
- Odoslanie SMS správy na zvolené číslo
- Odoslanie viacerých SMS správ na jedno číslo
- Prehliadanie kontaktov (mien a telefónnych čísel)
- Pridanie nového čísla do kontaktov
- Zobraziť help s návodom na použitie

Následne si ukážeme najdôležitejšiu požiadavku a to vytvorenie spojenia so serverom. V kapitole Implementácia si ukážeme akým spôsobom sa odosielajú SMS správy na zvolené číslo a ako sa pristupuje ku telefónnym číslam.

4.3 Vytvorenie spojenia medzi klientom a serverom

Tento sekvenčný diagram (obrázok č. 11) je zobrazenie klientská časti aplikácie, ktorá slúži na spojenie so serverom. Zároveň sa pri vytváraní spojenia používa aj šifrovanie RSA, ktoré má zabezpečiť zašifrovanie prechádzajúcich správ.

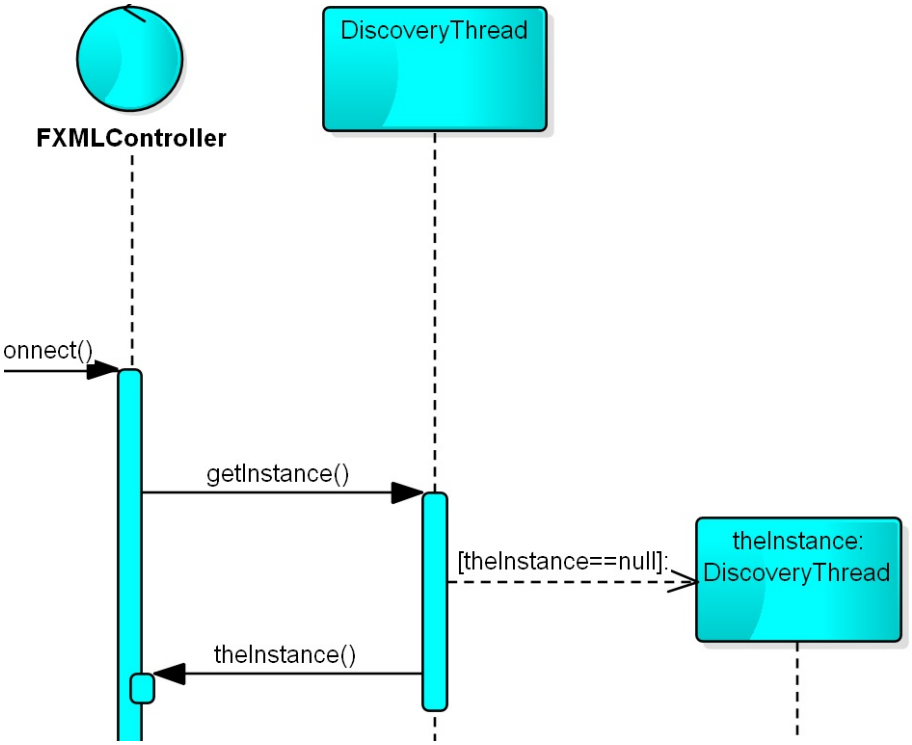


Obrázok 11: Sekvenčný diagram vytvorenia pripojenia s použitím návrhového vzoru Singleton a šifrovania RSA

Používateľ v prvom kroku stlačí tlačidlo connect na klientskej strane aplikácie. Trieda *FXMLController*, ktorá je súčasťou modelu MVC, zachytí kliknutie, vytvára inštanciu triedy *DiscoveryThread*. V tejto triede sa pomocou broadcastového vysielania snažíme zistiť adresu servera, ktorý čaká na akúkoľvek činnosť na danom porte. Pred samotným odoslaním sa správa, ktorá sa odosiela do celej siete zašifruje, aby ju mohol prečítať iba

server. Server si po zachytení a dešifrovaní správu prečíta a odpovie. Odpoveď sa znova dešifruje a odošle sa FXMLControleru, ktorý zistenú adresu Servera nastaví.

Na ďalšom obrázku(obr č. 12) vidíme výrez z predošlého diagramu, ide o použitie návrhového vzoru Singleton[34]. FXMLController získava inštanciu triedy *DiscoveryThread* pomocou metódy getInstance(), táto metóda mu vráti iba jedinú možnú inštanciu tejto triedy. Takýmto spôsobom je dosiahnutý návrhový vzor Singleton. Pri sieťových operáciach je vhodné, aby vždy existovala iba jediná inštancia danej triedy, ktorá pristupuje k sieti.



Obrázok 12: Použitie návrhového vzoru Singleton

[34]https://sourcemaking.com/design patterns/singleton

5 Implementácia a použitie

Aplikácia, ktorú sa budeme snažiť vyvíjať, bude pozostávať z niekoľkých častí. Budeme sa snažiť ukazovať, s akými problémami sme sa počas implementácie stretávali a ako sme ich riešili.

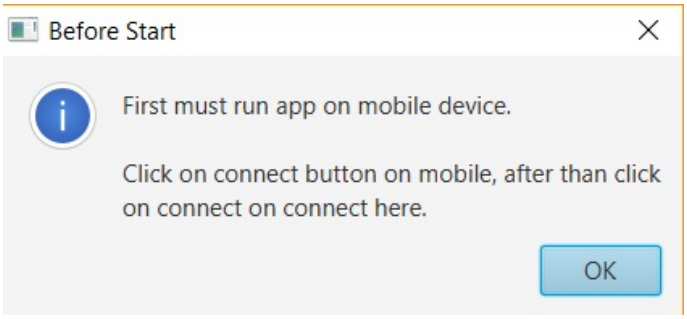
5.1 Kompatibilita a vývojové prostredia

Ako prvé bolo treba vyriešiť vhodné prostredie pre implementovanie oboch častí aplikácie (klientskej/desktopovej a serverovej/mobilnej). Pri klientovi sme zvolili programovací jazyk JAVA, konkrétne framework *JavaFX* (Java 1.8.0_73), vývojové prostredie NetBeans 8.0.2. JavaFX ponúka mnohé výhody, okrem iného priamo pracuje so vzorom MVC (angl. Model-View-Controler). Grafické rozhranie realizuje pomocou XML a dokáže jednoducho mapovať grafické prvky(view) s premennými(model). Ukážeme si aj lambda výrazy, ktoré výrazne zjednodušujú a spriehľadňujú kód.

Vzhľadom na rozšírenosť a prístupnosť sme pri serveri vybrali mobil s operačným systémom Android(*4.4.4 KitKat*), ide o relatívne starší typ(2013) a teda je určitá záruka pružnosti pri použití na novších zariadeniach. Programovací jazyk je tak isto Java, vývojové prostredie Android Studio 1.3.1. Najskôr si ukážeme, ako tieto jednotlivé časti aplikácie vyzerajú.

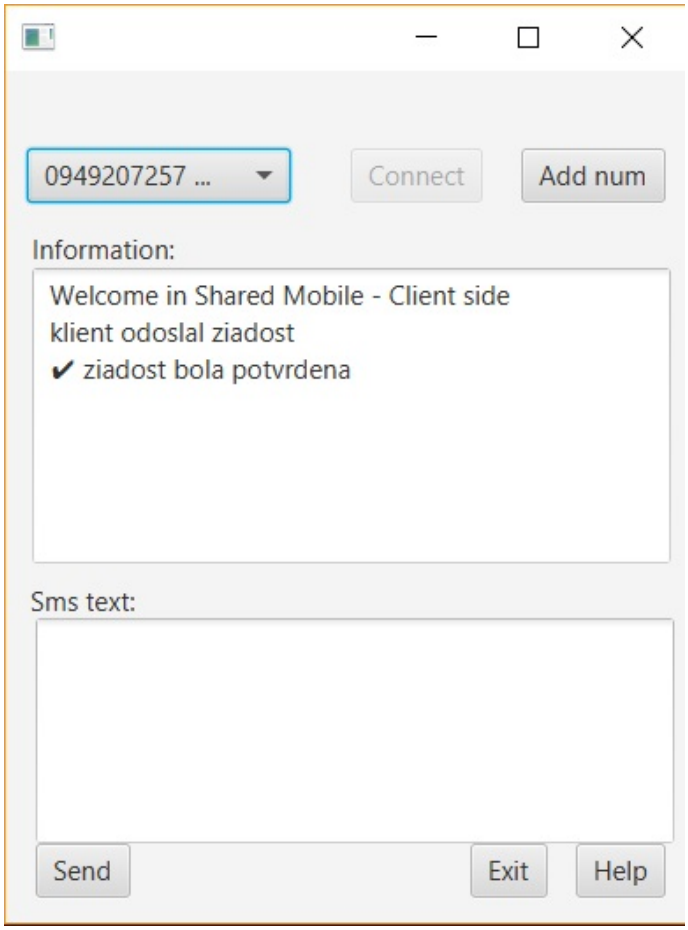
5.2 Používateľské rozhranie

Keďže sa aplikácia skladá z dvoch častí, serverovej(mobil) a klientskej(počítač), budeme pracovať s viacerými oknami. Hlavné rozhranie aplikácie je klient v počítači(obrázok č.15).



Obrázok 13: Upozornenie používateľa pre spustením desktopovej aplikácie

Pred spustením desktopovej aplikácie nás program upozorní, že ako prvá musí byť spustená aplikácia v mobile (obr. 13). Potom ako zapneme mobilnú aplikáciu a vytvoríme spojenie so serverom, máme k dispozícií zoznam čísel, na ktoré môžeme posielať správy. Tlačidlom *Add num* môžeme pridať nové čísla do zoznamu, ak sa tam už nenachádza. Help slúži na zobrazenie informácií o aplikácii a obsahuje aj stručný návod. Tlačidlo *Send* slúži na odoslanie textovej správy na označené číslo. V prípade, že textové pole (SMS text) ostáva prázdne, klient neposiela nič, a čaká na zadanie textu.

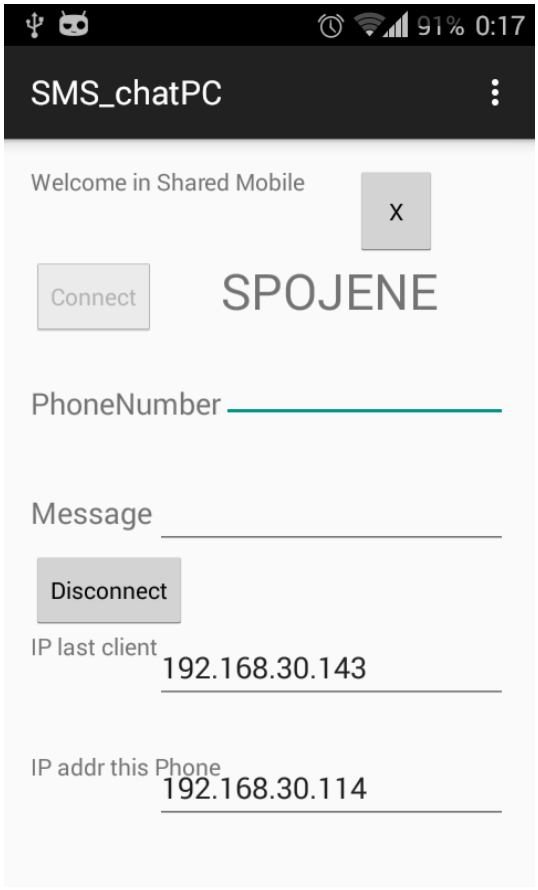


Obrázok 14: Hlavná obrazovka desktopovej aplikácie (klient)

Ďalšia časť je venovaná mobilnej aplikácii, ktorá slúži len pri vytvorení spojenia. Táto aplikácia je voľne dostupná[35]. Hlavná obrazovka obsahuje iba jedno tlačidlo connect, ktoré slúži na spustenie servera a začatie počúvania od akéhokoľvek klienta. Textové pole *debugMessage* slúži na zobrazenie aktuálneho stavu servera. Textové polia, ktoré nasledujú za *PhoneNumber* a *Message*, slúžia na vypísanie poslednej odoslanej správy na konkrétne číslo. Tlačidlo *Disconnect*, ktoré je pri spustení neprístupné môžeme použiť až

[35]https://play.google.com/store/apps/details?id=com.bc.michal.sms_chatpc#details-reviews

po vytvorení spojenia. Pomocou neho je možné ukončiť všetky spojenia medzi serverom a klientom(klientmi). Posledné textové polia sú venované jednotlivým IP adresám, konkrétne ide o adresu klienta, ktorý ako posledný poslal žiadosť o odoslanie SMS správy. Druhá IP adresa, je aktuálna adresa servera, teda mobilu.



Obrázok 15: Hlavná obrazovka mobilnej aplikácie (server)

5.3 Odoslanie SMS správy

Popri klientskej časti aplikácie sme začali pracovať aj na serverovej časti a to konkrétne využitím samostatnej knižnice *android.telephony.SmsManager;*. Táto knižnica dokáže odosielať SMS správy na zadané telefónne číslo. V ukážke algoritmu je ukázané ako dokáže odosielať aj jednoduchú SMS správu, ale aj zloženú, teda viac SMS správ naraz.

Algoritmus 1 SMS manager

```
1  // odoslanie samostatnej spravy
2  // na tel. cislo phoneNumber
3  SmsManager smsManager = SmsManager.getDefault();
4  smsManager.sendTextMessage(phoneNumber, null, textSms, null, null);
5
6  // odoslanie viacerych sprav pomocou parts
7  // takym sposobom, ze posle viac casti za sebou
8  // tieto casti zozbieral do StringBuilder LSMS
9  LSMS.append(receiveMsg.getBody());
10 ArrayList<String> parts = smsManager.divideMessage(LSMS.toString());
11 smsManager.sendMultipartTextMessage(receiveMsg.getNumber(), parts);
```

Tento algoritmus využíva voľne dostupnú triedu *android.telephony.SmsManager*[36]. Táto trieda okrem iných funkcionálnych vlastností, má medzi svojimi metódami, aj metódu *sendTextMessage()*, ktorá dokáže odoslať text zadaný medzi argumentami na telefónne číslo, ktoré je tiež zadané ako jeden z argumentov.

Pri odoslaní väčšej správy(jedna SMS správa môže mať maximálne 160 znakov), sa postupuje podobne. Najskôr sa zozbierajú všetky časti správy do premennej typu StringBuilder, ktorá má oproti klasickému typu String výhodu, že nevytvára vždy novú inštanciu a je synchronizovaná na rozdiel od StringBuffra. Keď prijmeme celú správu od klienta, táto sprava sa rozdelí na časti s maximálnou dĺžkou 160 znakov(pri špeciálnych znakoch je to ešte aj menej). Takto rozdelená správa sa pomocou metódy *sendMultipartTextMessage()* odošle na dané číslo.

[36]http://developer.android.com/reference/android/telephony/SmsManager.html

6 Evalvácia

V tejto časti by som rád uviedol testy, ktorými skúšame a overujeme správnosť aplikácie. Prvé testy slúžia na dokázanie funkčnosti mobilných služieb, teda odosielanie SMS správ a vykonanie hovoru. Ďalšie testovacie scenáre slúžia na overenie bezpečnosti, spoľahlivosti aplikácie.

6.1 Testovacie scenáre

1. Odoslanie samostatnej SMS správy
2. Odoslanie dvoch rozdielnych SMS-správ na dve rôzne čísla
3. Odoslanie čo najväčšieho množstva SMS-správ od jedneho klienta až po zahltenie
4. Maximálna dĺžka SMS správy a jej rozloženie na menšie správy
5. Dĺžka odozvy a priemerný čas potrebný pre odoslanie SMS správy
6. Platformovo nezávislý softvér
7. Koľko vieme vybaviť klientov naraz (multiklient využitie)
8. Bezpečnostný RSA prvok
9. Využitie CPU a pamäte mobilnej časti aplikácie
10. Predikcia distribúcie zaťaženia servera (príliš veľké zaťaženie)

6.1.1 Odoslanie samostatnej SMS správy

V tomto testovacom scenári sa vždy najskôr vytvorí nové spojenie so serverom. Testovanie prebehlo v dvoch sieťach, jednej lokálnej (hot-spot) a druhej verejnej Wifi sieti (školský eduroam). Počet opakovaní je ustálený na desať. Tabuľka porovnáva časové odozvy rozdielne pri lokálnej sieti oproti verejnej sieti.

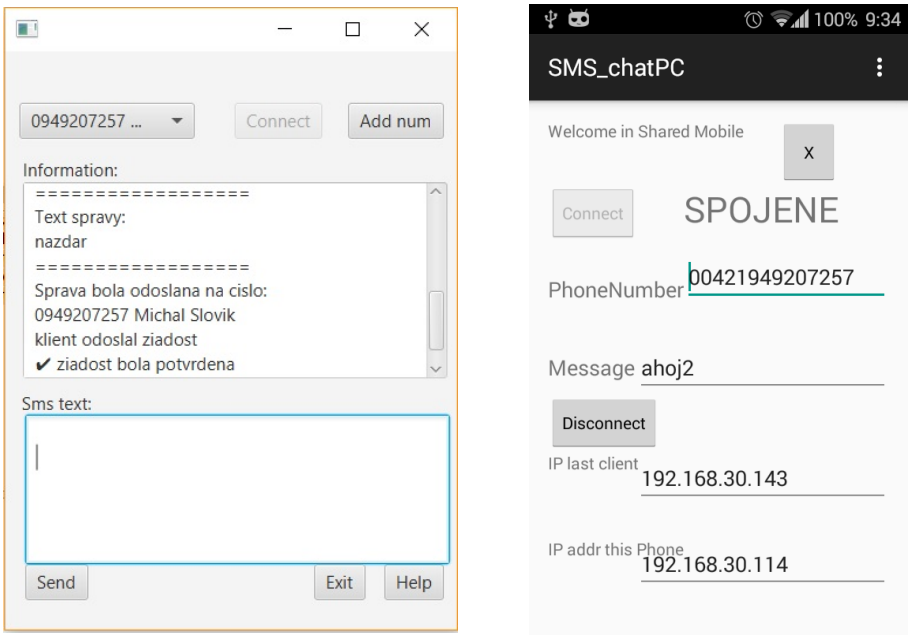
Tabuľka 2: Tabuľka časových odoziev v dvoch rôznych sieťach.

#	**HotSpot (ms), doba odozvy**	**Verejná sieť (ms)**
1	865	542
2	681	561
3	784	593
4	703	537
5	773	748
6	688	606
7	827	606
8	962	574
9	708	601
10	750	664
Priemer	**774.1**	**603.2**

V tabuľke môžeme vidieť desať opakovaných pripojení k serveru. Všetky pripojenia boli uskutočnené a SMS sa podarilo odoslať. V tomto prípade nás hlavne zaujímala doba odozvy pre vytvorenie spojenia. Klient musí broadcastom vyhľadať adresu server, server musí odpovedať a až tak vznikne spojenie. Všetky tieto správy sú šifrované, takže časové odozvy pribúdajú. Šifrovaniu a jeho dopadu sa budeme venovať neskôr. Zaujímavé je ešte povedať, že doba odozvy použitím hot-spotu bola pomalšia ako v prípade verejnej siete. Pravdepodobne je to spôsobené samotným klientom, ktorý je pripojený k viacerým sieťam (k lokálnej cez kábel, k bezdrôtovej cez wifi), a preto musí program prehľadávať obe siete. V prípade verejnej siete je potrebné hľadať server iba v jednej.

6.1.2 Odoslanie dvoch rozdielnych SMS na dve rôzne čísla

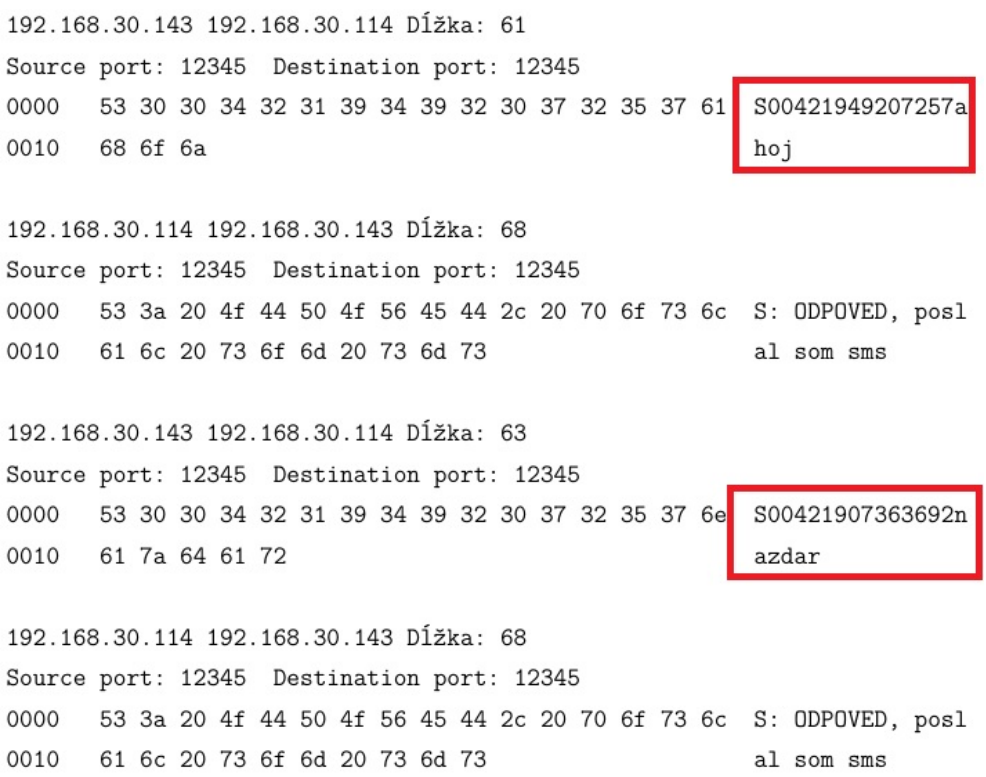
V prípade vytvorenia spojenia medzi klientom a serverom, môžeme začať posielať ľubovoľné SMS správy na ľubovoľné telefónne čísla. Po odoslaní SMS správy(stlačením tlačidla *send* alebo klávesou enter), bude odoslaná naša žiadosť na server. V prípade úspešného odoslania oboch SMS správ budú hlavné obrazovky oboch aplikácií vyzerať rovnako ako na obr.16.



Obrázok 16: Obrazovky jednotlivých zariadení. Vľavo je desktopová aplikácia. Vpravo sa nachádza mobilná aplikácia, ktorá zobrazuje poslednú odoslanú SMS správu

Ďalej môžeme vidieť (obr. č. 17) zachytenie celej komunikácie pomocou nástroja WireShark[37], ktorý umožňuje sledovať tok dát v sieti po jednotlivých paketoch. Vidíme teda štyri pakety, pričom prvý obsahuje číslo, na ktoré sa odosiela správa a samotný text správy. Nasleduje odpoveď od servera, po ktorom znova posielame SMS správu na iné číslo s iným textom. Táto správa je opäť potvrdená serverom. Správy, ktoré odosiela klient, sú zvýraznené červenou farbou.

[37]https://www.wireshark.org/

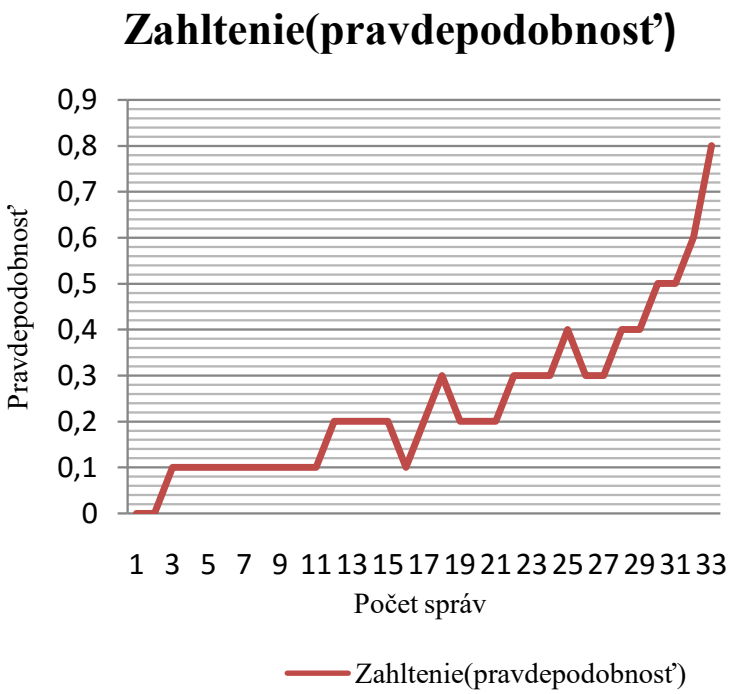


Obrázok 17: Wiresharkova komunikácia počas odosielania SMS správ

Na týchto záznamoch z Wiresharku vidíme ako prebiehala celá komunikácia počas odosielania SMS správ. Prvé dva riadky výseku vždy ukazujú IP adresu odosielateľa a IP adresu prijímateľa. Port je na oboch stranách aplikácie vždy rovnaký(12345). Dĺžka správ závisí od samotného textu správy, ktorú odosiela klient. Správa, ktorú klient odosiela sa skladá z príznaku 'S', ktoré signalizuje serveru, aby odoslal jednoduchú SMS správu. Ďalšia časť správy obsahuje telefónne číslo, na ktoré sa odosiela SMS správa. Zvyšok správy tvorí samotný text SMS.

6.1.3 Odoslanie čo najväčšieho množstva SMS správ od jedného klienta až po zahltenie

V prípade korektného spojenia jedného klienta sme sa pokúsili o odoslanie čo najväčšieho počtu SMS správ. K serveru je možné pristupovať aj z viacerých klientov, ale pre potreby tohoto testovacieho scenára nám postačuje jeden klient, ktorý sa pokúša odoslať čo najväčšie množstvo správ na rôzne čísla a takýmto spôsobom zahltiť server. Server by mal byť schopný správy prijímať, odosielať v správnom poradí a so správnym textom na zadané telefónne čísla.



Obrázok 18: Pravdepodobnosť zahltenia aplikácie po odoslaní SMS správ

V prvom prípade nastáva zlyhanie, ak používateľ odošle správu skôr ako príde potvrdenie predošlej SMS správy. V tom okamihu aplikácia ešte čaká na porte a nedokáže pristúpiť k sieti, čo sa prejaví zachytením výnimky a oznámením o chybe.

Ďalšia možnosť je neukončenie vlákna. V prípade, že používateľ odosiela veľké množstvo správ (viac ako desať SMS správ v priebehu 10 sekúnd), hrozí riziko preťaženia obsluhy vlákien a desktopová aplikácia znova zachytí výnimku o chybe.

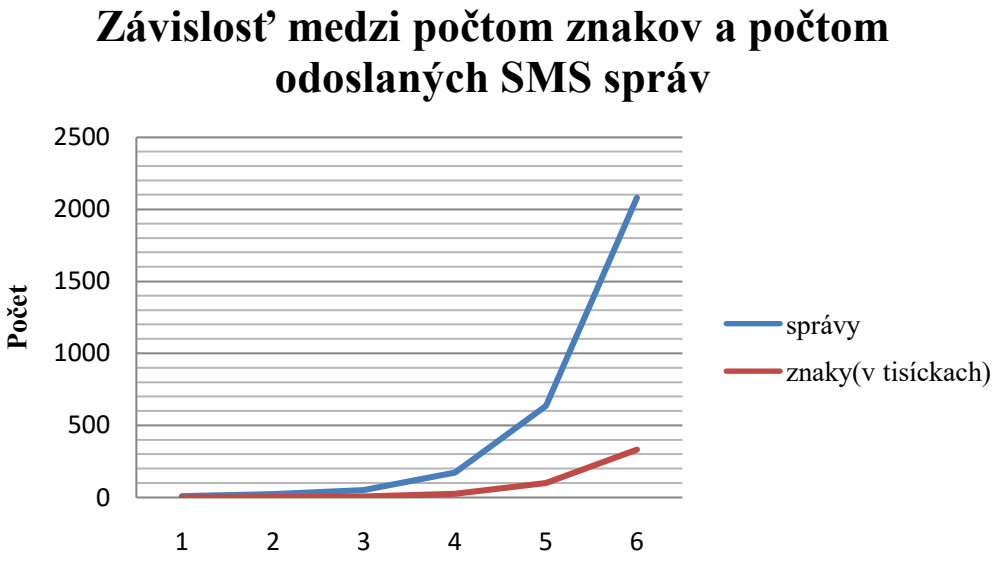
Posledným obmedzením je nedokončenie vlákien. Úspešne sme boli schopní odoslať 26 správ (dĺžka anglickej abecedy), pričom každá správa vytvorila nové vlákno, a aj keď sme dokázali odosielať ďalšie správy, pri 33. vlákne(teda 33. vytvorenej správe) nás aplikácia upozornila na výnimku v tomto vlákne.

6.1.4 Maximálna dĺžka SMS správy a jej rozloženie na menšie správy

Dĺžka jednej SMS správy je maximálna 160 ASCII znakov. V prípade, že správa obsahuje aj špeciálne znaky (napr.: mäkčene, dĺžne, ...) maximálna dĺžka správy klesá, pretože tieto znaky zaberajú viac bajtov.

Ako prvé skúšali sme odoslať správu, ktorej dĺžka bola viac ako 1000 znakov. Vďaka metóde sendMultipartTextMessage sa veľmi jednoducho podarilo odoslať 9 SMS správ, ktoré prijímateľ nakoniec dostal ako jednu celistvú správu.

Takýmto spôsobom sme zvyšovali počet znakov, ktoré by mal byť schopný odoslať. Na nasledujúcom grafe (obr. č. 19) je možné vidieť závislosť medzi počtom znakov a počtom správ potrebných na odoslanie takejto správy. Na grafe je znázornené, kde približne sa nachádza horná hranica počtu znakov, ktoré je klient schopný spracovať a odoslať, približne okolo 200-tisíc znakov.



Obrázok 19: Závislosť medzi počtom znakov a počtom odoslaných SMS správ

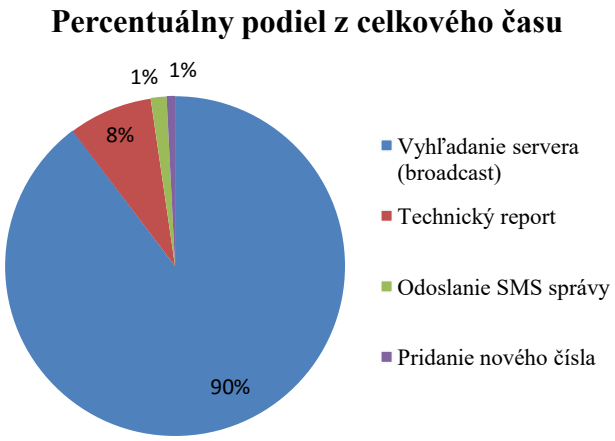
6.1.5 Dĺžka odozvy a priemerný čas potrebný pre odoslanie SMS správy

Pomocou logov programu vieme zistiť jednotlivé časy pri začiatku komunikácie. Na tabuľke č. 3 vidíme jednotlivé časy po desiatich meraniach aplikácie. Celkový čas počítame od vyhľadania servera pomocou broadcastu, cez prvú technickú správu, odoslanie jednej SMS správy a prípadne pridanie nového čísla. Komunikácia prebiehala medzi jedným klientom(počítač s operačným systémom Windows 10, Asus) a jedným serverom(mobil s verziou Androidu 4.4.4). Všetkých desať meraní úspešne odoslali SMS správu na zvolené číslo. Na grafe č. 22 vidíme, že najväčší podiel na celkovom čase má vyhľadanie servera, čo sa v bežnej prevádzke vykoná iba raz a následne iba odosielame SMS správy, ktoré zaberajú 1% času(2 až 5 ms).

Tabuľka 3: Tabuľka celkového času počas behu aplikácie a percentuálny podiel jednotlivých častí, počas desiatich meraní.

	Čas od spustenia(ms)	**Podiel vyhľadania**	**Dobá odpovede**	**Odoslanie SMS**
1	229	95%	4%	1%
2	281	73%	26%	1%
3	279	80%	16%	3%
4	250	92%	4%	3%
5	238	92%	5%	1%
6	241	95%	4%	1%
7	238	90%	5%	1%
8	297	95%	3%	2%
9	245	91%	9%	1%
10	227	95%	4%	1%

V tejto tabuľke môžeme vidieť počas desiatich pokusov jednotlivé časy ako dlho trvali jednotlivé činnosti. Napríklad v prvom riadku, aplikácia bežala 229 ms, z toho 95 % (217 ms)času venovala vyhľadaniu servera, 4% (10 ms)času tvorilo odoslanie testovacej správy a prijatie odpovede a 1% (2 ms)času tvorilo odoslanie SMS správy. Takýmto spôsobom sme vykonali desať testovaní. Priemerné hodnoty vyhľadávania servera tvoria 90% celkového času aplikácie. Na odoslanie technickej správy priemerne aplikácia spotrebuje 8% času a odoslanie SMS správy 2% času. Medián behu aplikácie bol 252,5 ms.



Obrázok 20: Percentuálny podiel na celkovom čase

6.1.6 Platformovo nezávislý softvér

Obe aplikácie už boli sprístupnené 26. apríla[38] [39] a ľudia tak mali možnosť tieto aplikácie skúšať na svojich zariadeniach. V tabuľke (tabuľka č.4) vidíme zoznam, na ktorých vidíme, aké zariadenia boli schopné spustiť obe aplikácie, vytvoriť spojenie a odoslať SMS správu.

Tabuľka 4: Tabuľka zobrazujúca typy zariadení pre časti server a klient

#	Klient (typ, operačny systém)	Server (typ, operačný systém)	stiahnutie	spojenie	odoslanie
1	Lenovo, Windows 10	Samsung, Android 4.4.4(KitKat)	OK	OK	OK
2	Dell Inspirion, Windows 8.1	Lenovo XL, Android 5.0.1(Lollipop)	OK	OK	OK
3	Toshiba, Windows 7	Sony Xperia, Android 4.1.2	OK	OK	OK
4	Acer Aspire, Windows 8.1	Samsung, Android 5.1	OK	OK	OK
5	Asus, Windows 8.1	Xiaomi, Android 6.0 (Marshmallow)	OK	OK	OK
6	Windows 10	Samsung GT, Android 4.4.2	OK	OK	OK
7	Toshiba, Ubuntu 12.2	Samsung, Android 4.4.4(KitKat)	OK	OK	X
8	Windows 10	Android 2.3.5	X	X	X
9	Windows 10	Lenovo, Android 5.1	OK	X	X

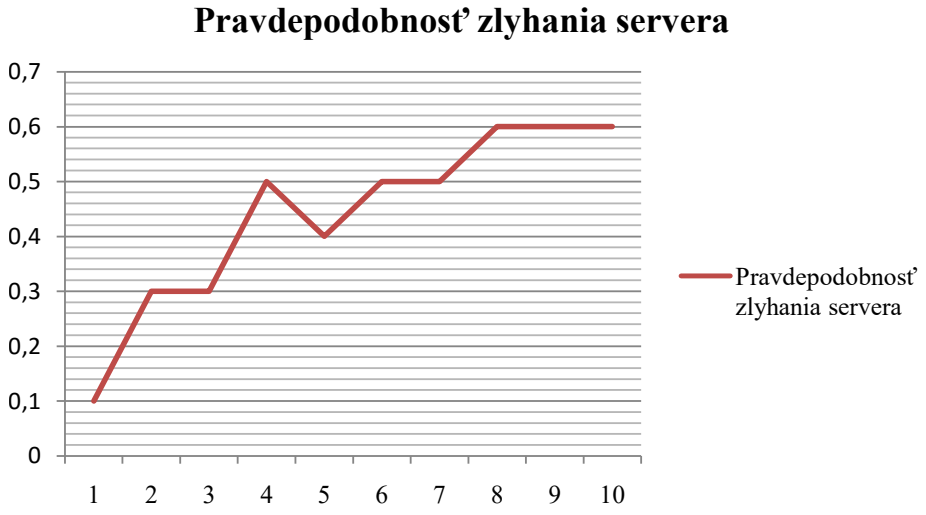
Tabuľka č. 4 nám ukazuje na akých zariadeniach sme boli schopní mobilnú aplikáciu stiahnuť. V prípade verzie Androidu nižšej ako 3.0 sme aplikáciu nedokázali ani stiahnuť (značené X). Vytvoriť spojenie sa podarilo väčšine zariadení, ktoré dokázali stiahnuť aplikáciu okrem jedného prípadu (9.riadok). Odoslať SMS správu bolo problematické aj pre aplikácie, ktoré pred tým nemali problém. Konkrétne test na riadku číslo 7 mal problém s právami na odoslania. Pri ostatných zariadeniach sa podarilo odoslať SMS správu.

[38] https://github.com/Mishco/BC_app_2

[39] https://play.google.com/store/apps/details?id=com.bc.michal.sms_chatpc#details-reviews

6.1.7 Koľko vieme vybaviť klientov naraz (využitie architektúry multiklient)

Server po stlačení tlačidla *connect* začína počúvať broadcastové vysielanie akýchkoľvek klientov. V prípade, že klient odošle žiadosť o pripojenie, server mu odpovie a ďalej čaká na broadcastové vysielanie. To znamená, že dokáže obsluhovať viac klientov.

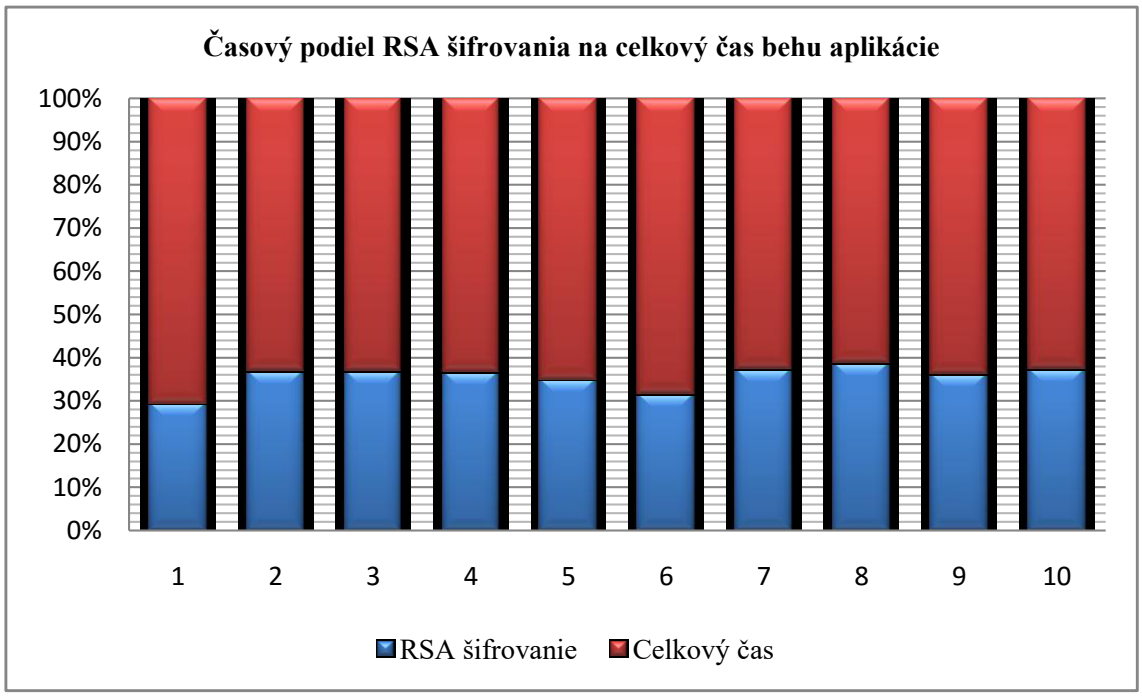


Obrázok 21: Pravdepodobnosť zlyhania servera vzhľadom na počet pripojených klientov

Vyskúšali sme test dvoch klientov na rozdielnych počítačoch. Ale z desiatich pokusov o pripojenie dva zlyhali, alebo boli chybne vytvorené a preto sme stanovili pravdepodobnosť chyby 0,2 pri dvoch klientoch. Takýmto spôsobom sme navyšovali počet klientov a získavali jednotlivé pravdepodobnosti, ale stále na dvoch zariadeniach (aplikácia umožňuje vytvárať viaceré inštancie samej seba), a teda z jedného počítača dokážu aj viacerí klienti odosielať SMS správy. Server dokázal spracovávať požiadavky, ale určitom počte klientov (8) sa ustálila pravdepodobnosť zlyhania. Buď nastala chyba pri spájaní s menším počtom klientov, alebo ak už prekročil hranicu počtu klientov nebol žiadny problém pripojiť aj ďalšieho klienta.

6.1.8 Bezpečnostný RSA prvok

Kvôli zabezpečeniu celej komunikácie využívame šifrovací algoritmus RSA. V našej aplikácii sme zvolili p = 17 a q = 19. Za *e* sme dosadili 65537, ktoré je považované za dostatočne bezpečné, ale zároveň optimálne pri rýchlosti výpočtu.



Obrázok 22: Časový podiel RSA šifrovania na celkovom času behu aplikácie

Graf (obr. č.22) nám zobrazuje podiel času počas šifrovania. Šifrovanie využívame na zabezpečenie komunikácie medzi klientom a serverom. Počas desiatich meraní sme zaznamenávali koľko času zaberá RSA šifrovanie. Vzhľadom na celkový beh aplikácie je to priemerne menej ako 40 %. V prvom prípade aplikácia bežala 988 ms z toho 316 ms zaberalo šifrovanie a dešifrovanie správ. RSA šifrovanie má 31 % podiel na celkovom čase behu aplikácie. Šifrovanie je teda určitým obmedzením (spomaľuje celkovú dobu aplikácie), avšak ak máme na zreteli bezpečnosť vždy je potrebné obetovať určitý čas.

6.1.9 Využitie CPU a pamäte mobilnej časti aplikácie

Všetko funkcionálne aj nefunkcionálne požiadavky si vyžadujú určité množstvo pamäte a výpočtového času procesora. Pri desktopovej aplikácii to síce tiež nesmieme zanedbať, ale podstatnejší ukazovateľ je pri mobilnej aplikácie. Na sledovanie jednotlivých veličín v rôznych časoch nám poslúžil nástroj *Android Monitor* implementovaný v Android Studiu. Sledovali sme využitie procesora počas spustenia samotnej aplikácie, počas vytvárania spojenia a počas odosielania SMS správy. Pri využití pamäťového priestoru sme sledovali maximálnu pridelenú pamäť, využitie pri spustení aplikácie, počas vytvárania spojenia a hlavne pri odosielaní SMS správy.

Tabuľka 5: Využitie procesorového výkonu

#	štart	spojenie	odoslanie SMS
1	5,02%	4,75%	2,35%
2	3,45%	5,15%	3,25%
3	3,51%	6,10%	3,23%
4	4,92%	5,00%	3,31%
5	3,51%	7,50%	3,50%
6	4,75%	6,45%	3,35%
7	4,83%	5,15%	3,24%
8	3,62%	6,32%	3,58%
9	4,95%	4,85%	4,02%
10	3,69%	5,45%	3,65%
Priemer	4,23%	5,67%	3,35%

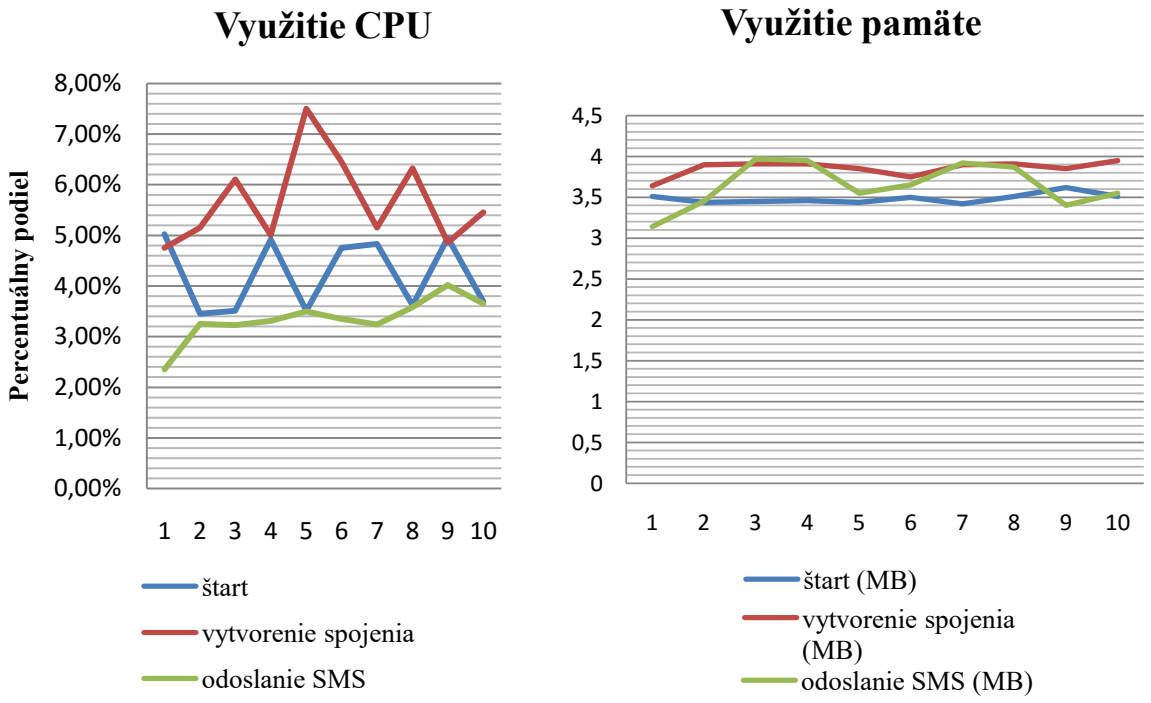
Tabuľka č. 5 nám ukazuje využitie procesora počas jednotlivých činností, teda pri štarte samotnej aplikácie, pri vytváraní spojenia a pri odosielaní SMS správy. Všetky tieto hodnoty boli čítané za pomoci nástroja Android Monitor. Aplikácia priemerne využíva 4,23 %procesorového výkonu na spustenie aplikácie. Na vytvorenie spojenie využíva 5,67 % procesora a na odoslanie SMS správy 3,35 %. V ďalšej tabuľke môžeme vidieť pamäťové využitie.

Tabuľka 6: Využitie pamäťového priestoru

#	štart (MB)	vytvorenie spojenia (MB)	odoslanie SMS (MB)	pridelené maximum
1	3,51	3,64	3,14	6,73 MB
2	3,44	3,9	3,45	5,95 MB
3	3,45	3,91	3,97	5,71 MB
4	3,46	3,91	3,95	5,74 MB
5	3,44	3,85	3,55	5,74 MB
6	3,5	3,75	3,65	5,72 MB
7	3,42	3,9	3,92	5,71 MB
8	3,51	3,91	3,87	5,74 MB
9	3,62	3,85	3,4	5,92 MB
10	3,51	3,95	3,55	6,25 MB
Priemer	3,486	3,857	3,645	

Na tabuľke č.6 je znázornené využitie pamäte počas jednotlivých krokov programu. Pri štarte aplikácie priemerne využíva 3,486 MB pamäte. Počas vytvárania spojenia využíva priemerne 3,857 MB pamäte. Pri odosielaní SMS správy priemerne využíva 3,645 MB pamäte.

Tieto hodnoty môžeme prehľadnejšie vidieť na obrázku č. 23.

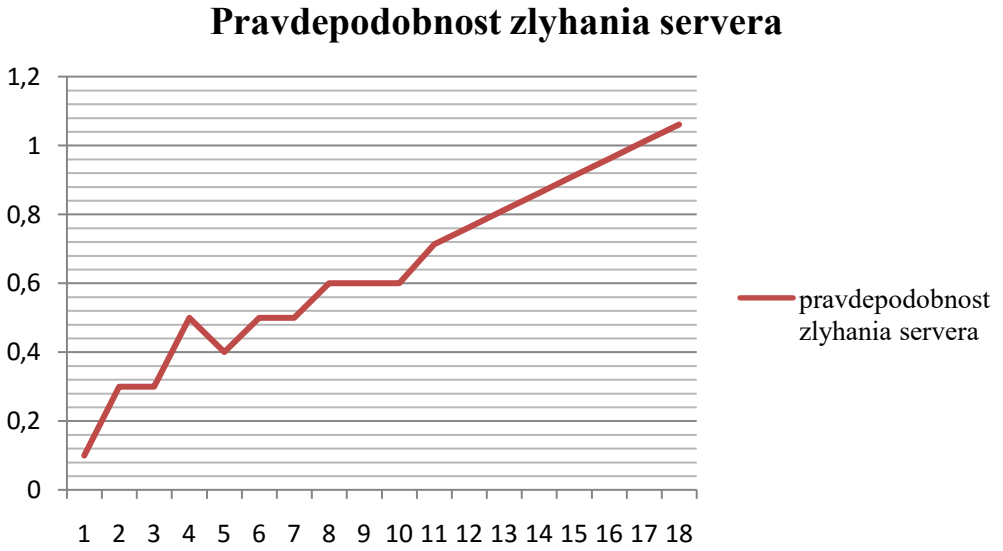


Obrázok 23: Využitie procesora a pamäťového priestoru v jednotlivých krokoch

6.1.10 Predikcia distribúcie zaťaženia servera (príliš veľké zaťaženie)

Táto požiadavka síce nepatrí medzi funkcionálne požiadavky a v našej aplikácii zatiaľ neviem riešiť distribúciu zaťaženia. Zapojenie viacerých mobilných koncových zariadení, v našom prípade serverov, by zvýšilo spoľahlivosť systému. V prípade veľkého zaťaženia by server mal mať schopnosť rozložiť svoje požiadavky a preposlať ich "kolegovi", teda ďalšiemu serveru. Kto zapne druhý server, dokedy má bežať a podobné požiadavky by musel riešiť nadradený systém. V tomto testovaní využijeme predikciu, to znamená, na základe pravdepodobnosti zlyhania servera vzhľadom na počet pripojených klientov budeme predpokladať ďalšie správanie.

Ak teda pri zapojení jedného servera a 10 klientov, máme pravdepodobnosť zlyhania servera 0,6, tak lineárne predikujeme správanie pri vyššom počte klientov.



Obrázok 24: Lineárna predikcia zlyhania servera pri väčšom počte klientov

Môžeme vidieť (obr. č.24), že pri 17 klientoch je pravdepodobnosť zlyhania rovná 1, čo znamená, že server určite zlyhá. Ak by sme chceli zabezpečiť distribúcii zaťaženia bolo by potrebné, aby nadradený systém (alebo samotný server), pred dosiahnutím tohto počtu pripojených klientov rozdelili požiadavky, znížili pravdepodobnosť zlyhania, a zvýšili tak spoľahlivosť celého systému.

Záver

Počas tohto roka sme hľadali a zisťovali dostupné riešenia zdieľania telekomunikačných služieb rôznymi prostriedkami. Medzi prvými sa ukázali najskôr veľmi jednoduché riešenia, ako napríklad AirDroid, TeamViewer a podobné. Postupne sme prechádzali ďalej a hľadali sme aplikáciu, ktorá by dokázala zdieľať svoje služby pre jedného používateľa, ale aj pre viacerých používateľov. Tieto aplikácie zatiaľ nedokázali pracovať s takouto architektúrou, a mnohé svoje služby ponúkajú za poplatok. Ďalšou nevýhodou bolo nemožnosť presmerovania hovorov na operačnom systéme Android v dôsledku neprístupnosti k RIL.

Ďalším hľadaním sme postupne objavili jednoduchšie aplikácie, ktoré nemajú tak rozsiahlu funkcionalitu,(napríklad Signal), ale ponúkajú zdieľanie mobilných služieb. Ďalším plusom tejto aplikácie je jej otvorený kód, ktorý nám umožňuje vidieť ako pracuje operačný systém Android s mobilnými službami.

V ďalšej časti sme sa venovali protokolom, ktoré spolupracujú pri zdieľaní multimediálnych služieb. Najskôr sme spomenuli RTP, ktorý dokázal streamovať video alebo zvuk na presne zadanú IP adresu a číslo portu a tak dokázalo akékoľvek iné zariadenie počúvať pomocou tejto IP adresy a portu. Presmerovaním streamu medzi počítačom a notebookom sme dokázali odosielať a prijímať zvukové dáta. Avšak ako sme zistili presmerovanie tohto streamu do vytvoreného hovoru nie je triviálne.

Ďalším protokolom, ktorý sme sledovali bol SIP protokol. Tento protokol umožňuje zdieľanie mobilných služieb (VoIP). Pracuje so vzdialeným serverom do ktorého sa prihlasujeme pomocou mena a hesla a dokážeme komunikovať s iným zariadením, ktoré je tiež pripojené na tento server. Protokol, ktorý pracuje na nižšej vrstve, ale ponúka možnosť zdieľania multimediálnych súborov je UDP.

Takto sme sa dostali k jednoduchému použitiu UDP paketov na odosielanie správ. Aplikácia, ktorá vznikla najskôr mala architektúru klient-server, pričom klient bol počítač a serverom náš zdieľaný mobil. Pri zdieľaný telekomunikačných služieb sme sa dostali k odosielaniu SMS správ. Pričom android poskytoval jednoduchý postup ako sa dostať k tejto triede a využívať ju. Ďalšou iteráciou sme zistili, že existuje možnosť odosielať SMS správu s rozsiahlym textom. Klient sme vyvíjali smerom, aby bol schopný vyhľadať server sám a pripojiť sa k nemu bez zásahu používateľa, takýmto spôsobom sme sa dostali k broadcastovému vysielaniu.

Broadcast ako taký prináša značnú výhodu aj v serverovej časti aplikácie, pretože server dokáže obsluhovať ľubovoľný počet klientov, stačí aby bol klient v tej istej sieti. A tak sa dostávame k architektúre multiklient. Umožňuje pripojenie viacerých klientov k

jednému serveru a následnému obsluhovaniu všetkých požiadaviek serverom.

Avšak broadcast a UDP pakety sú veľmi jednoducho dostupné a preto je potrebné ich zabezpečiť. Zabezpečenie sme realizovali RSA šifrovaním, ktoré je jednoduché na pochopenie a implementáciu a stále dostatočne bezpečné aj v dnešnej dobe. Jeho nevýhodou je spomaľovanie behu aplikácie, ale stále je to pre používateľa nepatrná zmena.

Aplikáciu sme po implementácii dali k dispozícii verejnosti a tak sme mohli získať cenné informácie k testovaniu. Testovanie aplikácie prebiehalo vo viacerých fázach.

Samozrejme sme testovaním narazili na určité ohraničenia našej aplikácie. RSA šifrovanie síce zabezpečuje komunikáciu, ale predlžuje celkový čas jednotlivých operácií programu. Počet klientov pripojených k jednému serveru je obmedzený pravdepodobnosťou zlyhania servera, ktorý po prekročení tohto limitu nezaručuje správne fungovanie. Počiatočne testovania sa zameriavali na splnenie funkcionálnych požiadaviek. Odoslanie jednoduchej ale aj zloženej SMS správy, odoslanie viacerých SMS správ do určitého limitu. Ďalším kritériom pri testovaní bolo efektivita využívania zdrojov. Výpočtový výkon, doba vykonávania jednotlivých častí programu či schopnosť pracovať v architektúre multi-klient. Na základe zistených výsledkov, by sme mohli usúdiť, že aplikácia spĺňa základne požiadavky. Dokáže pracovať aj v multi-klient architektúre a dokáže správne využívať hardvérové prostriedky.

Čo sa týka tejto práce, sú značné možnosti, ako by sa dalo postupovať. Medzi prvé kroky by malo patriť zdieľanie aj multimediálnych správ (MMS). Zdieľania klasického hovoru alebo dokonca vytvorenie či pripojenie sa k skupinovému hovoru môže byť zásadne ovplyvniteľné prístupom k triede RIL, pretože táto trieda dokáže priamo pristupovať k hovoru. Ďalšou alternatívou, ktorou by sa dalo pokračovať je možná distribúcia výkonu servera, ktorý by v prípade veľkej prevádzky vyhodnotil, že potrebuje pomôcť a iný server by obslúžil jeho nadbytočné požiadavky. Taktiež by sa dalo premýšľať aj nad rozšírením frontendu pre iné operačné systémy a iné zariadenia(tablety, smartphony, ...). Všetky tieto kroky boli približne začaté touto prácou, ktorá čerpala už z existujúcich aplikácií. Je teda veľká schopnosť rozšírenia rôznymi smermi.

Zoznam použitej literatúry

[1] Abiona, O.O. Oluwaranti, A.I. ; Anjali, T. ; Onime, C.E. ; Popoola, E.O. ; Aderounmu, G.A. ; Oluwatope, A.O. ; Kehinde, L. Architectural Model for Wireless Peer-to-Peer (WP2P) File Sharing for ubiquitous Mobile Devices. pp. 35–39.

[2] Bluetooth Marketing. Bluetooth Range. http://www.bluair.pl/bluetooth-range.

[3] Brown, J., Shipman, B., and Vetter, R. SMS: The short message service. *Computer 40*, 12 (2007), 106–110.

[4] GmBH, T. Team Viewer. https://www.teamviewer.com.

[5] H. Schulzrinne, S. C. RTP Profile for Audio and Video Conferences with Minimal Control. http://www.ietf.org/rfc/rfc3551.txt, 2003.

[6] Handley, M. SIP: Session Initiation Protocol.

[7] Hoffman, C. Control your Android from a Browser with airdroid. How-To Geek. http://www.howtogeek.com/105813/control-your-android-from-a-browser-with-airdroid/.

[8] Hoideker, J. Selathco 0.91 generated HTML: Obecné MIDI (General MIDI). http://www-kiv.zcu.cz/ herout/html_sbo/midi/5.html.

[9] Li, L., and Wang, X. P2P File-Sharing Application on Mobile Phones Based on SIP. In *2007 Innovations in Information Technologies (IIT)* (nov 2007), IEEE, pp. 601–605.

[10] Liu, F., Li, Z., and Yu, J. P2P applications identification based on the statistics analysis of packet length. In *Proceedings - 2009 International Symposium on Information Engineering and Electronic Commerce, IEEC 2009* (2009), pp. 160–163.

[11] LogMeIn, I. JoinMe. https://www.join.me/joinme.

[12] Ludwig, S., Beda, J., Saint-Andre, P., McQueen, R., Egan, S., and Hildebrand, J. Jingle. http://xmpp.org/extensions/xep-0166.html#howitworks, dec 2009.

[13] Mishra, A. Performance and architecture of SGSN and GGSN of general packet radio service (GPRS). In *GLOBECOM'01. IEEE Global Telecommunications Conference (Cat. No.01CH37270)* (2001), vol. 6, IEEE, pp. 3494–3498.

[14] OpenWhisperSystems. Signal. https://whispersystems.org/.

[15] Oracle. What is J2ME or Java ME? http://www.java.com/en/download/faq/whatis_j2me.xml

[16] Organo, A. How To Access Android Phone From PC With AirDroid. `http://www.eyes4tech.com/how-to-access-android-phone-from-pc-with-airdroid/`, nov 2011.

[17] Rosenberg, J. . H. S. . G. C. . A. J. . J. P. . R. S. . M. H. . E. S. SIP: Session Initiation Protocol. http://www.ietf.org/rfc/rfc3261.txt.

[18] Sand Studio. AirDroid. http://www.airdroid.com.

[19] Schulzrinne, H. C. R. F. J. RTP: A Transport Protocol for Real-Time Applications. http://www.ietf.org/rfc/rfc3550.txt.

[20] Sinam, T., Singh, I. T., Lamabam, P., Devi, N. N., and Nandi, S. A technique for classification of VoIP flows in UDP media streams using VoIP signalling traffic. In *2014 IEEE International Advance Computing Conference (IACC)* (feb 2014), IEEE, pp. 354–359.

[21] Spyappsmobile. Advantages of Rooting Your Android Device — SpyAppsMobile.com. http://spyappsmobile.com/advantages-of-rooting-your-android-device.

[22] Sundar, S. Voice over IP via Bluetooth/Wi-Fi Peer to Peer. *Advances in ...* (2012).

[23] system, C. Cisco Cloud Computing - Data Center Strategy, Architecture, and Solutions.

[24] The GSMA Foundation. GSM. http://www.gsma.com/aboutus/gsm-technology/gsm.

[25] Tolvanen, J., Suihko, T., Lipasti, J., and Asokan, N. Remote Storage for Mobile Devices. In *2006 1st International Conference on Communication Systems Software & Middleware*, IEEE, pp. 1–9.

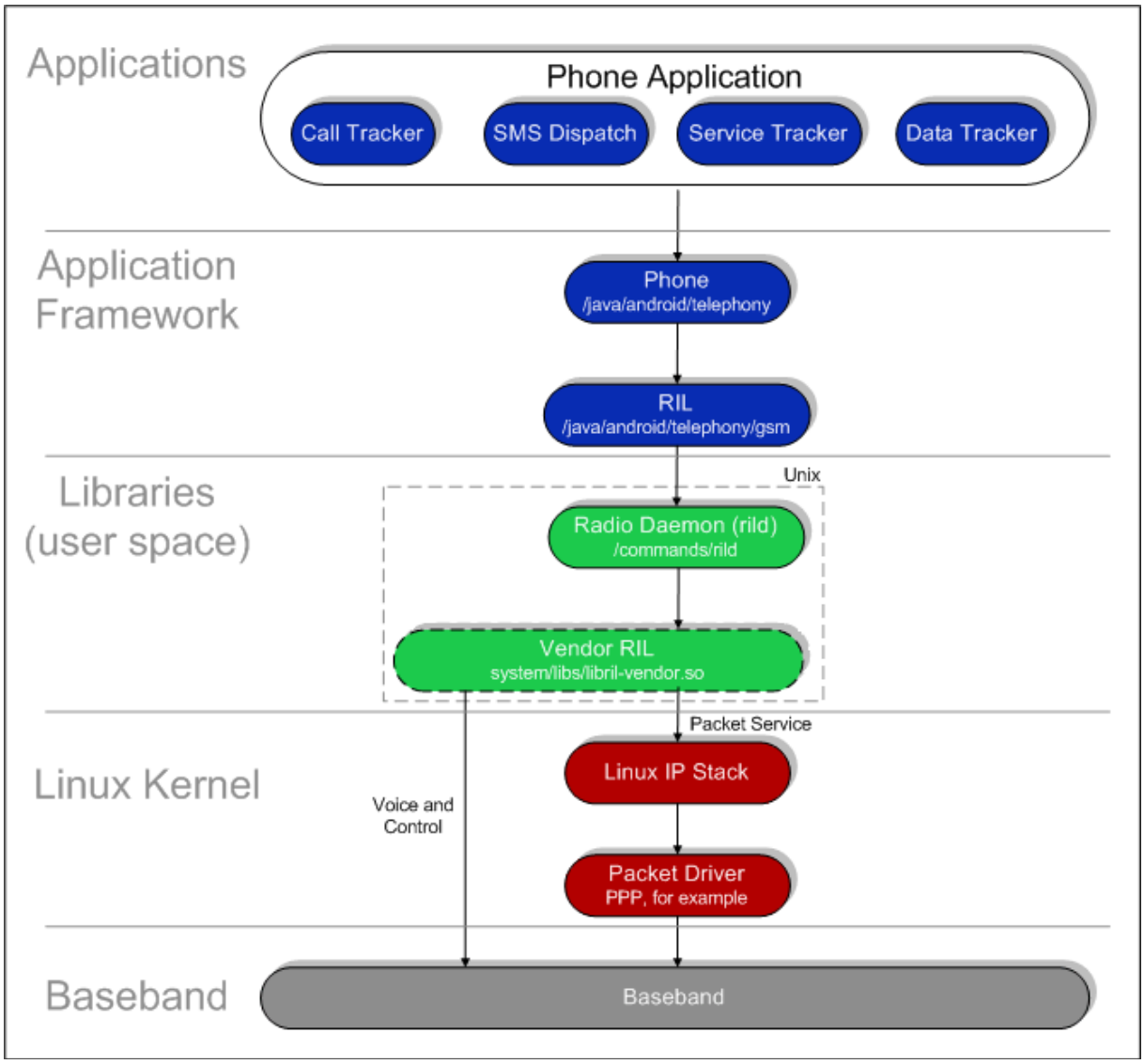
[26] uwe Andersen;Hermann De Meer;Jens O. Oberender, T. T. Simulative performance evaluation of a mobile peer-to-peer file-sharing system.

Prílohy

A Štruktúra elektronického nosiča

\
\Bakalarska_praca.pdf
\Manual.pdf
\Klient
\Klient\BC_app_client_1.0.jar
\Klient\numbers
\Klient\README.txt
\Klient\time.txt
\Klient\ZdrojovyKod**SMS_chatPC**\
\Server
\Server\BC_app_server_2.0.apk
\Server\README.txt
\Server\ZdrojovyKod**BC_app_client_1.0**\

B Abstraktná trieda RILD

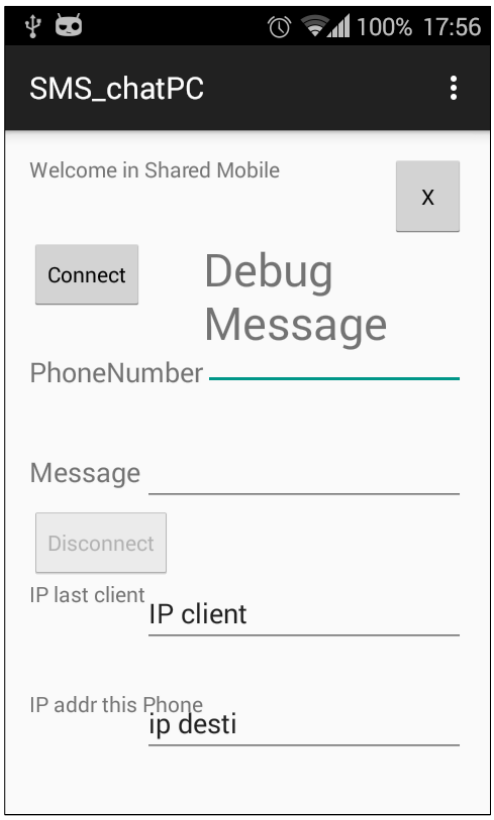


Obrázok B.1: RIL v kontexte operačného systému Android

C Používateľská príručka

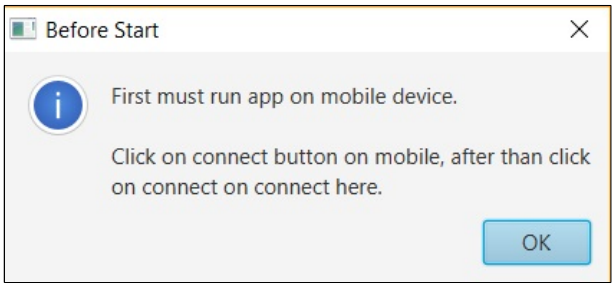
Ako prvé si stiahneme obe aplikácie. Serverovú do mobilu s operačným systémom Android (min. verzia 3.2).
Dostupné na `https://play.google.com/store/apps/details?id=com.bc.michal.sms_chatpc#details-reviews`
Druhá je dostupná taktiež online a to na adrese https://github.com/Mishco/BC_app_2 kde stiahneme celý balík .zip Oba časti aplikácie sú aj súčasťou priloženého CD.

- Krok 1: Pripojenie k rovnakej WiFi oboch zariadení
- Krok 2: Spustenie aplikácie v mobile
- Krok 3: Spustenie aplikácie v počítači.
- Krok 4: Kliknutie na connect v mobile.



Obrázok C.1: Úvodná obrazovka mobilnej aplikácie

- Krok 5: Kliknutie na connect v počítačovej aplikácii



Obrázok C.2: Upozornenie pred spustením obrazovky desktopovej aplikácie

- Krok 6: Čakanie na spojenie (maximálna pár sekund) ak sa spojenie nevytvorilo, je potrebné obe aplikácie reštartovať a začať znova.
- Krok 7: Spojenie beží

C.1 Odoslanie SMS správy

- Krok 1: Vytvorenie spojenia
- Krok 2: Vybratie tel. čísla v klientskej aplikácii. Ak nie je vo výbere žiadne vaše číslo, stačí ho pridať tlačidlom *add num*
- Krok 3: Napísanie textu a stlačenie tlačidla *send* (alebo klávesy enter).