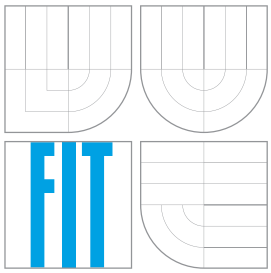
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

SDÍLENÍ ČÁSTI OBRAZOVKY

SCREEN SHARING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

SAMUEL RADIMÁK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. RUDOLF KAJAN

BRNO 2013

Abstrakt

Tato bakalářská práce je věnovaná popisu metod sdílení plochy a možnostem, které tato řešení poskytují. V práci jsou vysvětleny pojmy spojené se sdílením plochy. Dále jsou tu popsána jednotlivá existující řešení. Uvedené jsou též základní protokoly pro přenos obrazu a rozdíly mezi nimi. Větší část práce je věnována návrhu a implementaci aplikace, která umí sdílet plochu. Aplikace je rozdělena na dvě části, a to klient a server. Pro návrh je využit modelovací jazyk UML (Unified Modeling Language). Pro přenos samotného obrazu je využit protokol RFB (Remote Framebuffer), jehož popis se nachází v této práci. Výsledná aplikace byla podrobena testům, které jsou v této práci uvedeny taktéž. Vytvořená aplikace je porovnána s existujícími řešeními a načrtnuty jsou i oblasti, v kterých je možné tuto aplikaci vylepšit.

Abstract

The following bachelor thesis is dedicated to the description of methods for a desktop sharing and possibilities which these methods provide. In the beginning, principles connected with the concept of desktop sharing are explained. Moreover, there are described various solutions, which had already existed before my research. In addition, fundamental protocols for the transfer of the screen and main differences between them are mentioned. The most extensive part is devoted to the design and the implementation of an application for screen sharing. The application is split into two parts, a client and a server. The development is described with the use of UML (Unified Modeling Language). For the transport of the screen, RFB (Remote Framebuffer) protocol has been chosen. In advance, reduced description of this protocol is mentioned in this text. The developed application is tested and compared to other existing solutions. Finally, areas where improvements are possible are stated.

Klíčová slova

sdílení plochy, protokol RFB, remote framebuffer, C#, klient, server, WPF

Keywords

desktop sharing, RFB protocol, remote framebuffer, C#, client, server, WPF

Citace

Samuel Radimák: Sdílení části obrazovky, bakalářská práce, Brno, FIT VUT v Brně, 2013

Sdílení části obrazovky

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Rudolfa Kajana. Další informace jsou získany z odborné literatury a článků z různých informačních zdrojů. Všechny tyto zdroje, literární prameny a publikace, ze kterých jsem čerpal jsou v této práci uvedeny.

.....
Samuel Radimák
14. května 2013

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Rudolfovi Kajanovi za pomoc a užitečné rady při vytváření bakalářské práce.

Obsah

Kapitola 1

Úvod

Väčšinu nových projektov, nie len v oblasti informačných technológií, je nutné riešiť v týme. Týmová spolupráca je preto veľmi dôležitou súčasťou úspešného vytvorenia projektu. Nie vždy je ale možné, aby tým spolupracoval v plnej miere, za každých okolností. Vďaka tomu, že prístupnosť vysokorýchlostného Internetu je čoraz väčšia, sa do popredia dostávajú rôzne nástroje pre zjednodušenie spolupráce. Sú to tzv. kolaboračné nástroje (anglicky „collaboration tools“). Jedná sa o veľké množstvo nástrojov, programov, aplikácií a techník, ktoré umožňujú spoluprácu viacerých ľudí nad jedným problémom. Jednou zo skupín sú tzv. „real-time“ kolaboračné nástroje. Tie umožňujú spoluprácu v reálnom čase, ako napríklad komunikáciu, zdieľanie súborov či úpravu spoločných dokumentov. Tieto nástroje môžu nájsť uplatnenie pri práci nie len vo všetkých spoločnostiach, ale aj pri práci jednotlivcov.

Jedným z často využívaných kolaboračných nástrojov je zdieľanie plochy (anglicky „desktop sharing“). V najjednoduchšom prípade umožňuje zdieľanie plochy prepojenie dvoch počítačov. Jeden z nich je prezentujúci (účastník, ktorý zdieľa svoju plochu) a druhý len pasívne sleduje prácu, ktorú prezentujúci vykonáva. Pasívnych účastníkov prenosu môže byť viacero, čo pri spojení spolu s ďalšími komponentami multimediálnej komunikácie, ako napríklad prenos hlasu či správ, vytvára základ virtuálneho priestoru, kde sa môžu ľudia stretávať, socializovať a pracovať spoločne. Vo väčšom merítku sa jedná už o videokonferenciu. Definovaním pojmov a ďalším vysvetleniam sa zaoberá kapitola 2.

Existujúce riešenia umožňujú tiež ovládanie vzdialeného počítača. Poskytujú taktiež autentizáciu pre zabezpečenie komunikácie. Ovládanie vzdialeného počítača je štandardne aktívne, preto je ho nutné explicitne deaktivovať. Kapitola 3 sa zaoberá dostupnými riešeniami zdieľania plochy a poskytuje ďalšie informácie o využívaných technológiách. Širším popisom vybranej technológie prenosu obrazu sa zaoberá kapitola 4.

Cieľom tejto práce je návrh, realizácia a následné overenie funkcie vlastného riešenia, ktoré umožní zdieľanie plochy vzdialeného počítača. V rámci návrhu (kapitola 5) je postupne vybraný protokol, ktorý bude použitý pre prenos obrazu a je určený aj programovací jazyk, v ktorom bude výsledná aplikácia realizovaná. Pozornosť je taktiež venovaná zaisteniu správnej funkčnosti komunikácie v rôznych prípadoch. Overeniu funkčnosti implementovanej aplikácie je venovaná kapitola 6 a na koniec rôznymi možnosťami rozšírenia projektu sa zaoberá kapitola 7.

Kapitola 2

Zdieľanie obrazovky

Zdieľanie obrazovky je názov pre technológiu umožňujúcu používateľovi posielať dáta obrazu počítačovej obrazovky pomocou siete a ich zobrazenie na jednom či viacerých vzdialených zariadeniach.

Inými slovami, aplikácie umožňujúce zdieľanie obrazovky poskytujú možnosti, ako predstavovať prezentácie, dokumenty, obrázky, či akýkoľvek software spustený na počítači, zatiaľ čo vzdialene pripojení užívatelia v reálnom čase pozorujú, čo je zobrazované na obrazovke.

V súčasnosti je táto technológia spolu s technológiou vzdialeného ovládania využívaná najmä systémovými administrátormi. Administrátori riešia často jednoduché problémy, na ktoré táto technológia postačuje. V dnešnej dobe sa často stretávame aj s požiadavkami na kontrolu práce zamestnancov s firemnými zariadeniami. Zdieľanie obrazovky a jej monitorovanie je jednou z možností takejto kontroly.

Na strane zariadenia zdieľajúceho obraz je potrebný mechanizmus prenosu obrazu do klientovho zariadenia. Takisto je potrebná nejaká forma autentizácie používateľov, čím môže byť obmedzený prístup k zdieľanému obrazu. Na strane zariadenia prijímajúceho obraz je potrebné umožniť pripojenie, autentizáciu a zobrazenie prijatého obrazu.

Pre funkčnosť technológie je potrebné zaistiť prenos dát obrazu. Hlavnou časťou je teda protokol, ktorý zabezpečuje prenos obrazu zo serveru(zariadenie zdieľajúce obraz) na klienta (zariadenie zobrazujúce obraz). Existuje niekoľko takýchto protokolov, z toho najznámejšie sú:

- RFB (Remote FrameBuffer) protokol využívaný v systéme VNC (Virtual Network Computing),
- RDP (Remote Desktop Protocol) systému Windows.

Aplikácie pre zdieľanie obrazovky často vyžadujú veľkú šírku pásma pre prenos dát, preto je táto technológia využitá najlepšie tam, kde všetci účastníci majú dostupné dostatočne rýchle spojenie.

Zdieľanie obrazovky je teda charakterizované schopnosťou zdieľania obsahu počítačovej obrazovky v reálnom čase s jedným alebo viacerými vzdialene pripojenými používateľmi pomocou siete.

Kapitola 3

Dostupné existujúce aplikácie

V tejto časti budú porovnané existujúce riešenia. Uvádzané aplikácie umožňujú aj vzdialené ovládanie počítača. Hlavné znaky, ktoré charakterizujú populárne aplikácie a nástroje pre zdieľanie obrazovky sú:

1. *Zdieľanie zvolenej aplikácie.* Je to schopnosť zvoliť si špecifickú aplikáciu, dokument, prezentáciu, ktorá bude povolená pre zdieľanie s pripojenými osobami. Týmto je zabezpečené, že sa pripojení užívatelia nedostanú k častiam obrazovky, ktoré nie sú pre nich určené.

2. *Zdieľanie špeciálnej časti obrazovky.* Znamená to označenie špecifickej časti obrazovky, ktorá bude zdieľaná pripojeným používateľom. Prenášané bude čokoľvek nachádzajúce sa v tejto časti obrazovky.

3. *Zmena prezentujúceho.* Funkcia zmeny prezentujúceho umožňuje zdieľajúcemu vybrať ktoréhokoľvek pozorovateľa a učiniť ho novým prezentujúcim. Ako náhle ho zvolí, stáva sa novým prezentujúcim a má možnosť začať zdieľať svoju obrazovku s ostatnými používateľmi aplikácie.

4. *Maximálny počet účastníkov.* Jedná sa o maximálny počet účastníkov, ktorí sú súčasne pripojený k jednému zdieľajúcemu. Tento parameter býva v súčasných aplikáciách často obmedzovaný.

5. *Vzdialené ovládanie.* Je to umožnenie jednému alebo viacerým pozorovateľom zdieľanej obrazovky ovládnuť zariadenie zdieľajúce svoju obrazovku. Väčšinou sa jedná o zdieľanie prístupu k myši a klávesnici.

6. *Multiplatformnosť.* Dôležitým znakom je aj dostupnosť aplikácie pre rôzne platformy operačných systémov, ako napríklad Windows, Mac, Linux, atď.

7. *Rozšírené voľby.* Patria tu takisto možnosti komunikácie ako chat, VoIP (Voice over IP), tele konferencia, či podpora rôznych vizuálnych ukazovadiel a pomôcok.

8. *Nahrávanie.* Podpora nástrojov pre nahrávanie zobrazovanej zdieľanej obrazovky.

9. *Nutnosť sťahovania a inštalácie.* V súčasnosti existujú aj aplikácie, pre ktoré nie je potrebná inštalácia, či rôzne riešenia založené na zobrazení na internetových stránkach, tzv. web-based aplikácie.

3.1 Remote Desktop Connection

Remote Desktop Connection (RDC), takisto známy pod názvom Remote Desktop je klientská aplikácia od firmy Microsoft. Aplikácia je často používaná vďaka svojej dostupnosti, keďže je súčasťou každej verzie operačného systému Windows spoločnosti Microsoft. Po povolení služby na strane serveru je možné pripojenie klienta. Aktuálny používateľ prihlásený na serverovej strane je odhlásený zo svojho konta a nový používateľ je prihlásený s kompletnou kontrolou nad počítačom. Aplikácia využíva pre spojenie RDP (Remote Desktop Protocol).

3.1.1 RDP

Remote Desktop Protocol bol vyvinutý firmou Microsoft. Protokol je rozšírením rodiny protokolov ITU-T T.128 (International Telecommunication Union-Telecommunication Standardization Sector T.128), ktorej znenie je dostupné na adrese [2]. Pracuje na princípe klient-server, kde klient zobrazuje grafické používateľské prostredie, ktoré je vysielané zo vzdialeného počítača. RDP je vyvinutý pre podporu rôznych druhov sieťových topológií a viacero protokolov v LAN (Local Area Network). Server implicitne naslúcha na TCP (Transmission Control Protocol) porte 3389, ktorý mu bol pridelený organizáciou IANA (Internet Assigned Numbers Authority).

RDP je oveľa komplexnejší ako protokol RFB a umožňuje väčšiu mieru funkcionality, ako napríklad:

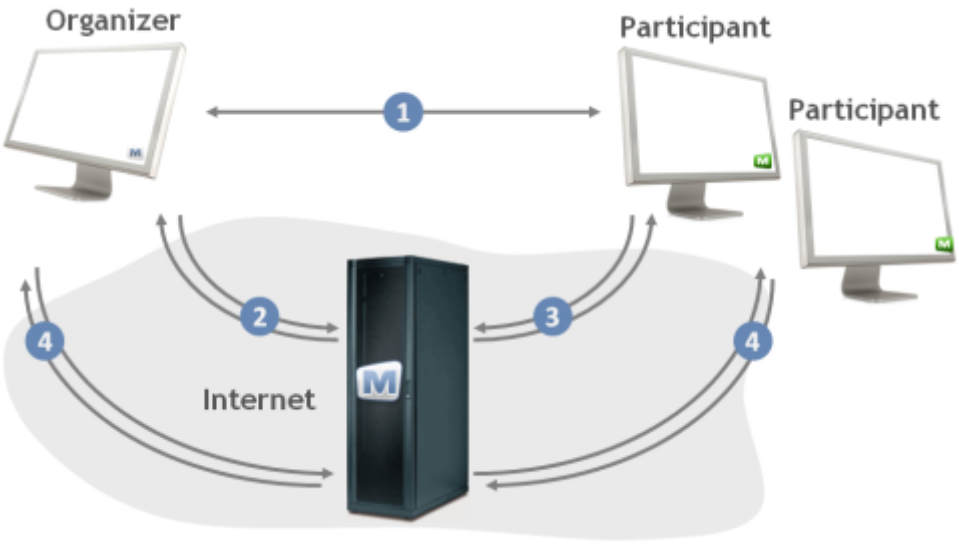
- tlač na vzdialenej tlačiarni,
- prenos zvuku,
- použitie lokálneho súboru na vzdialenom zariadení,
- viacnásobné pripojenie zariadení k jednému kanálu.

RDP nie je obmedzený len na operačný systém Windows. Existujú aplikácie využívajúce RDP aj pre iné operačné systémy. V operačnom systéme Mac OS X existuje taktiež implementácia klienta a nazýva sa Remote Desktop. Pre operačný systém Linux je vyvinutý klient s názvom rdesktop. Takisto existuje aj serverová časť implementácie, ktorá sa volá xrdp. Oba sú dostupné ako open source software. Existuje viacero ďalších implementácií tohto protokolu s rôznymi charakteristikami a obmedzeniami. Ich zoznam je možné nájsť na adrese [16].

Ďalšiu špecifikáciu protokolu je možné nájsť v [6] a [5].

3.2 Mikogo

Mikogo je aplikácia pre zdieľanie plochy a vzdialenú podporu vytvorená firmou BeamYourScreen GmbH. Software je dostupný pre rôzne operačné systémy ako Windows, Mac OS, Linux, iOS, Android. Poskytuje jednoduché riešenie on-line konferencií vďaka tomu, že aplikáciu je možné spustiť, sprístupniť tým zdieľanie plochy a umožniť klientom prístup k zdieľanému obrazu priamo z prehliadača. Domovská stránka aplikácie je dostupná na [7]. Použitá technológia komunikácie je proprietárna. Komunikácia aplikácie je zobrazená na obrázku 3.1 a funguje nasledovne:



Obr. 3.1: Architektúra Mikogo

1. Organizátor spojenia a klient sú v kontakte pomocou telefónu.
2. Organizátor spojenia umožní spustenie spojenia z osobného počítača, z Mikogo serveru príjme unikátny 9–číselný identifikátor, ktorý následne rozdistribuuje všetkým klientom.
3. Klienti otvoria internetový prehliadač, prejdú na hlavnú stránku aplikácie ([7]), kliknú na „Join Session“. Bez potreby inštalácie alebo registrácie jednoducho spustia software a zadajú 9-číselný identifikátor prijatý od organizátora.
4. Spojenie sa automaticky nadviaže a klienti môžu ihneď pozorovať organizátorovu zdieľanú plochu.

3.3 VNC

Virtual network computing (VNC) je systém pre zdieľanie obrazovky počítača, ktorý pre prenos informácií a komunikáciu využíva protokol RFB. Je ním možné takisto ovládať vzdialený systém pripojený pomocou siete a to vďaka tomu, že stlačenia tlačidiel klávesnice, ako aj myši sú prenášané sieťou od klienta k serveru. To dovoľuje technickej podpore prevádzať zmeny na počítačoch, serveroch, či iných sieťových zariadeniach bez fyzického kontaktu technikov so zariadeniami.

VNC pracuje na báze modelu klient-server. Klientská časť aplikácie je inštalovaná na lokálnom počítači a pripája sa k serverovej časti nachádzajúcej sa na vzdialenom zariadení. Server odosiela duplikát svojej obrazovky klientovi a prijíma príkazy prichádzajúce od klienta, ktoré postupne spracováva. VNC je nezávislé na platforme a kompatibilné s akýmkoľvek operačným systémom. Počítače musia byť pripojené k sieti s protokolovou sadou TCP/IP (Transmission Control Protocol/Internet Protocol) a mať otvorené porty pre prichádzajúcu komunikáciu.

VNC bol vytvorený ako open source projekt v Olivetti & Oracle Research Lab, no po ukončení projektu a zatvorení laboratória tvorcovia sformovali spoločnosť RealVNC, kde ďalej pokračovali na vývoji projektov využívajúcich VNC. V dnešnej dobe je pojem „VNC“ registrovanou značkou spoločnosti RealVNC Ltd.. Od tej doby bolo vytvorených mnoho projektov pre vzdialené ovládanie založených na VNC. Nižšie spomenuté produkty sú takisto založené na VNC, niektoré aplikácie docielili pridanú funkcionalitu.

3.3.1 RealVNC

RealVNC [9] je multiplatformná aplikácia umožňujúca vzdialenú kontrolu medzi dvoma počítačmi. Funguje na princípe peer-to-peer (povolené je spojenie len jeden zdieľajúci ku jednému klientovi), takže nepotrebuje žiaden centrálny server. Jeho limitácia je v použiteľnosti cez internet, keď sa pripájané zariadenie nachádza za routrom s NAT (Network address translation). Vtedy je potrebná ručná konfigurácia routra, alebo použitie tunelovania pomocou SSH (Secure Shell), čím sa zabezpečí takisto lepšia bezpečnosť prenosu medzi oboma stanicami.

RealVNC ponúka ako aj open source verziu, tak aj verzie platené. Platené verzie obsahujú viacero pridaných funkcií, ako napríklad prenos súborov, chat, vzdialená tlač, autentizácia serveru, zvýšená bezpečnosť.

Klient ako aj server sú implementované v jazyku C++. Pre jednoduchosť použitia existuje takisto klient implementovaný v jazyku Java, takže akékoľvek zariadenie môže byť ovládané vzdialene pomocou webového prehliadača bez potreby inštalácie prídavného software.

3.3.2 TightVNC

TightVNC je rozšírená verzia VNC, ktorá obsahuje viacero nových vylepšení, funkcií, optimalizácií, pričom si zachováva kompatibilitu so štandardným VNC. Aplikácia je multiplatformná a použiteľná v rôznych sieťových prostrediach.

Aplikácia je udržovaná Constantinom Kaplinskym, pričom na vývoji, testovaní a podpore sa podieľajú aj ďalšie firmy a jednotlivci. Oproti ostatným VNC aplikáciám ponúka ďalšie vylepšenia ako zlepšenie kompresie, zlepšenie bezpečnosti či multiplatformnosť.

TightVNC je dostupná zdarma, pod licenciou GPL (General Public License). Aplikácia je implementovaná v jazyku C++ alebo jazyku C, no dostupná je takisto klientská verzia v jazyku Java. Viac informácií je možné získať na ich hlavnej stránke [13].

3.3.3 UltraVNC

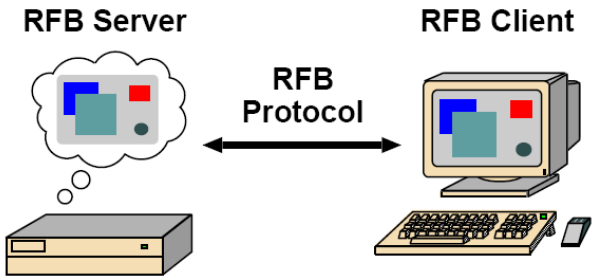
UltraVNC je ďalšou z aplikácií využívajúcich VNC. Ponúka mnoho vylepšení vo forme zásuvných modulov (pluginov). Hlavnými výhodami tejto aplikácie je prenos súborov, prídné kódovanie a zabezpečenie spojenia, chat, podpora viacerých monitorov a rôzne formy autentizácie.

Nevýhodou je problematické spojenie pri zariadení za routrom s NAT (Network address translation), no aj pre toto obmedzenie existuje zásuvný modul.

Aplikácia je určená len pre systém Windows a to od verzie 95. Implementovaná je v jazyku C++ a jazyku C, no takisto existuje aj implementácia v jazyku Java pre jednoduchý prístup k serveru pomocou webového prehliadača. Ďalšie informácie ako aj produkty je možné nájsť na [14].

Kapitola 4

RFB



Obr. 4.1: RFB protokol

Remote Framebuffer [10, 11] je protokol pre vzdialený prístup ku grafickému používateľskému prostrediu. Je používaný systémom VNC. Zámerom tohto protokolu je jednoduchosť a prenositeľnosť. Keďže pracuje na úrovni framebufferu, nemusí rozumieť sémantike grafických operácií, či vykonávať zložité renderovanie. To znamená, že je nezávislý od operačného systému.

Pre transport dát medzi serverom a klientom je použitý model TCP/IP. Používa jeden socket pre každé spojenie, pričom štandardný port, na ktorom server načúva je 5900. Klient nadväzuje spojenie. Po nadviazaní spojenia môže začať komunikácia medzi klientom a serverom. Protokol definuje aj tzv. načúvací mód, kde môže server nadväzovať spojenie s klientom. V tomto prípade klient načúva na porte 5500.

Komunikácia medzi RFB klientom a serverom sa skladá z dvoch hlavných fáz:

1. Počiatočné podanie rúk (handshake) – tabuľka 4.1
2. Prenos správ – tabuľka 4.2

Po inicializácií spojenia so serverom klient môže posielať a prijímať správy zo servera. Správy začínajú bajtom značiacim typ správy.

Interakcia medzi RFB klientom a serverom zahŕňa vyjednanie formátu a kódovania dát. Klient sa môže rozhodnúť medzi 32-, 16-, alebo 8-bitovými formátmi pixelov a 5 formátmi kódovania (Raw, Copy Rectangle, CoRRE, RRE, Hextile).

Protokol je tiež rozšíriteľný. Server môže podporovať napríklad prídavné kódovanie správ a ak klient podporuje toto isté kódovanie, môže byť použité. Takisto je možné použiť pseudo-

	Smer správy		
Poradie	**Z**	**Do**	**Názov správy**
1	Server	Klient	ProtocolVersion
2	Klient	Server	ProtocolVersion
3	Server	Klient	SecurityTypes
4	Klient	Server	ChosenSecurityType
5	Server	Klient	SecurityResult
6	Klient	Server	KlientInit
7	Server	Klient	ServerInit

Tabulka 4.1: Správy pri nadviazaní spojenia

	Smer správy		
Poradie	**Z**	**Do**	**Názov správy**
1	Klient	Server	SetPixelFormat
2	Klient	Server	SetEncodings
3	Klient	Server	FramebufferUpdateRequest
4	Server	Klient	FramebufferUpdate
5	Server	Klient	SetColourMapEntries
6	Server	Klient	Bell

Tabulka 4.2: Správy pri normálnej komunikácií

kódovanie, ktoré už môže byť použité na akúkoľvek prídavnú funkcionalitu, od zmeny veľkosti obrazu po prídavnú komunikáciu medzi serverom a klientom.

RFB je protokol, ktorý má málo požiadaviek na klienta. Vďaka tomu môže RFB klient bežať na systémoch s nízkymi nárokmi na hardware, vyžadujúc čo najmenšiu potrebnú výpočetnú silu stroja.

Zobrazovanie zdieľanej obrazovky je založené na jednoduchom princípe vloženia pixelov na danú pozíciu. To predpokladá udržovanie predchádzajúcej kópie obrazovky u klienta a je tým tiež docielené zmenšenie počtu prenášaných dát sieťou. Každý prenos obrazových dát by mal byť kódovaný jednou z dostupných metód, ktoré majú za následok ďalšiu kompresiu dát posielaných dát. Posielanie dát aktualizácie obrazu nie je automatické, ale vysielané vždy na podnet klienta, ktorý žiada o obnovenie obrazu. Tým je docielená možnosť adaptácie prenosu na siete s nižšou rýchlosťou prenosu, prípadne zmenšenie nárokov na výpočetný systém zariadenia.

Existuje mnoho implementácií VNC. Dostupné implementácie RFB serveru sú napríklad:

- *libvncserver*[3]: je knižnica pre implementáciu VNC serveru a klienta v jazyku C.
- *x0rfbserver*: je to X11 klient, ktorý pracuje ako RFB server
- *RealVNC*[9]: je jednoduchá aplikácia, ako klient, tak aj server. Ponúka aj dodatočné AES kódovanie.
- *KDE Desktop Sharing*: dávnejšie známe ako krfb, no od KDE verzie 3.1 je jeho súčasťou.
- *x11vnc*: podporuje ovládanie systému X Windows vychádza z LibVNCServer projektu

Takisto je dostupných niekoľko desiatok klientov, bežiacich na rôznych operačných systémoch, obsahujúcich rôzne výhody a nevýhody ich použitia[16].

RFB protokol má teda mnoho výhod:

- Jednoduchý a otvorený protokol
- Nízka náročnosť na šírku pásma, oneskorenie
- Rozšíriteľnosť
- Dostupnosť niekoľkých open source implementácií
- Mnoho existujúcich klientov pre rôzne platformy

Kapitola 5

Návrh

V tejto časti práce bude podrobne rozpísaný návrh aplikácie, a to presnejšie výber protokolu pre komunikáciu medzi serverom a klientom, voľba a návrh architektúry aplikácie, obmedzenia v routovanej sieti (internet), návrh riešenia pre odstránenie týchto obmedzení a výber programovacieho jazyka pre aplikáciu.

5.1 Protokol pre prenos dát obrazu

Ako jednu z dôležitých vecí je potrebné vybrať protokol pre zasielanie zdieľanej obrazovky a komunikáciu. Existujú dve rôzne riešenia tohto problému, a to návrh vlastného protokolu alebo použitie existujúceho protokolu.

Návrh vlastného protokolu a jeho implementácia nesie mnoho výhod a nevýhod. Výhodami sú hlavne neobmedzená škála rozšíriteľnosti a novátorský prístup k problémom prenosu. No nevýhody prevažujú toto riešenie. Vlastný protokol by v konečnom dôsledku nebol vôbec kompatibilný s aktuálnymi aplikáciami, čo prináša nevýhodu v uvedení novej aplikácie na trh. Ďalšou nevýhodou je náročnosť implementácie, pretože by bolo potrebné riešiť problémy, ako napríklad procesy autentizácie, zasielania dát obrazovky, formát správ, použitie kódovania, či prenos dát. Zároveň treba dodať, že všetky spomínané problémy sú už vyriešené a otestované v existujúcich protokoloch.

Použitie existujúceho protokolu je v tomto prípade vhodnou alternatívou. Pre súčasne najpoužívanejšie protokoly existujú špecifikácie, či prípadne aj samotné implementácie funkcií protokolov. Dva hlavné protokoly, ktoré sú v súčasnosti používané sú:

- RDP (Remote Desktop Protocol),
- RFB (Remote FrameBuffer).

Výber správneho protokolu pre implementáciu je teda zložitejší. Tvorba vlastného protokolu by bola veľmi náročná na implementáciu a takisto aj na dĺžku času, preto bola táto možnosť zavrhnutá. Dlhý čas by trval návrh a hlavne testovanie nového protokolu, pričom by stále nebolo zaručené odstránenie problémov nového protokolu a overená správna funkčnosť. Zostala možnosť použitia existujúceho protokolu. Protokol RDP je, ako už bolo spomenuté, protokolom zložitým po implementačnej stránke. Keďže obsahuje mnoho prídavných funkcií je jeho implementácia časovo náročná. Protokol RFB je na rozdiel od RDP jednoduchý protokol, ktorý obsahuje všetko potrebné pre prenos obrazu a preto bol zvolený tento variant protokolu. Voľba tohto protokolu nám takisto zaručí kompatibilitu s inými, už existujúcimi aplikáciami.

5.2 Programovací jazyk

V dnešnej dobe existuje rada jazykov, od nižších cez interpretované až po kompilované. Použitie jazykov nižších a interpretovaných môžeme rovno vylúčiť a to hlavne kvôli porovnaniu výkonu s jazykmi kompilovanými. Z kompilovaných jazykov je na výber znova viacero variant. Sú to napríklad jazyky ako Java, C, C++, C#.

Keďže cieľovým systémom pre vyvíjanú aplikáciu je systém Windows od firmy Microsoft, je najrozumnejšie využiť jazyk C# spolu s frameworkom .NET. Framework .NET je vyvíjaný spoločnosťou Microsoft a obsahuje obrovskú knižnicu tried a metód, čím sa vývoj aplikácie urýchli. Nevýhodou voľby tohto programovacieho jazyka je chýbajúca multiplatformnosť výslednej aplikácie, no využitím C# a .NET je dosiahnutá rýchlejšia spolupráca so systémom Windows, vďaka mnohým použitým optimalizáciám. Napriek tomu existujú riešenia aj pre tento problém multiplatformnosti (projekt Mono [8]).

Pre implementáciu grafického používateľského prostredia bolo zvolené rozhranie Windows Presentation Foundation (WPF) a to preto, že je to novšia technológia spoločnosti Windows ako technológia WinForms. Poskytuje podobné, v niektorých smeroch dokonca lepšie riešenia ako WinForms.

5.2.1 WPF

WPF (Windows Presentation Foundation) je programovacie rozhranie pre vytvorenie grafického používateľského prostredia pre aplikácie využívajúce framework .NET. Fyzicky je to sada knižníc a podporných nástrojov. Je určený pre vytvorenie bohatých a sofistikovaných používateľských rozhraní.

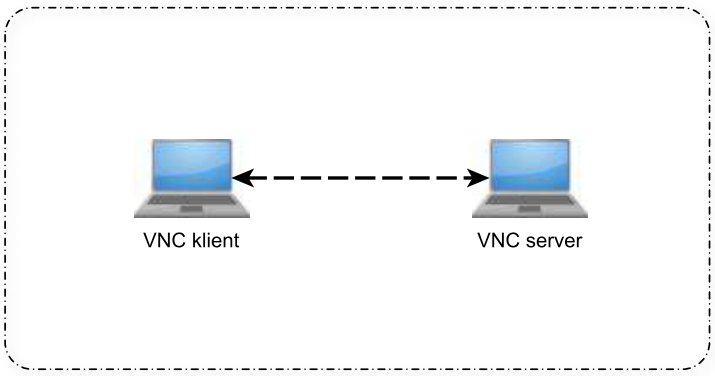
Výhodou oproti WinForms je, že WPF podporuje vektorovú grafiku a animácie. Využíva silnú integráciu s operačným systémom a tzv. „data binding“. Ten podporuje oddelenie dát a funčnosti aplikácie od prezentačnej vrstvy. Obsahuje tiež podporu pre 3D vykresľovanie a vylepšenú typografiu. Ďalšou výhodou oproti jeho predchodcovi, Windows Forms, je, že WPF je postavené priamo na DirectX. DirectX bolo spočiatku určené pre multimédiá a herné programovanie. Vo WPF je použitím tohto rozhrania možné vytvárať rôzne vizuálne triky, ktoré vo Windows Forms nie je takmer vôbec možné dosiahnuť. Taktiež to znamená, že WPF využíva výhody hardwarovej akcelerácie vždy, kedy je to možné.

WPF má stále podobnosti s Windows Forms. Microsoft poskytuje knižnicu základných ovládacích prvkov ako napríklad textové pole alebo tlačidlá. Taktiež dodržiava koncepty ako napríklad „data binding“ alebo „code-behind“ súbory. Všetky tieto koncepty boli ale zdokonalené pre WPF.

Pre reprezentáciu ovládacích prvkov a definovanie vzhľadu prostredia je využitý jazyk XAML (Extensible Application Markup Language). Je to nový programovací jazyk vytvorený Microsoftom pre jednoduché vytváranie používateľských prostredí.

5.3 Dostupné knižnice

Pri implementácií existujúceho protokolu je vhodné vykonať prieskum dostupných knižníc, ktoré by daný problém implementovali. Je to dôležité hlavne preto, že existujúce knižnice sú väčšinou preverené veľkým množstvom používateľov, a preto môžeme predpokladať, že hlavné a časté problémy sú odladené a kód je optimalizovaný. Programátor teda nemusí znovu implementovať rovnaké základné úlohy a môže sa venovať podstate problému na



Obr. 5.1: Komunikácia v spoločnej sieti

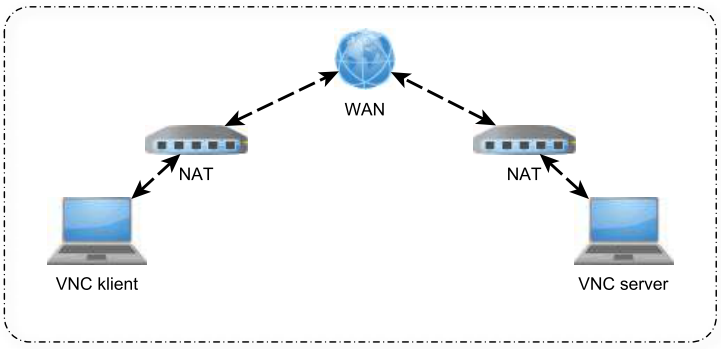
vyššej, abstraktnej vrstve. Základnými zvolenými parametrami aplikácie sú teda protokol RFB, programovací jazyk C#, a framework .NET. Po zvolení týchto parametrov je prehľad knižníc značne zúžený. Pre klientskú časť existuje mnoho knižníc implementovaných v jazyku C#, napríklad VncSharp[15]. VncSharp je open source implementácia VNC RFB protokolu pre .NET framework. Je dostupná pod licenciou GPL[1]. Pre zobrazenie používateľského prostredia ale využíva WinForms, čo tomuto projektu nevyhovuje, keďže bude využité rozhranie WPF. Napriek tomu budú zdrojové kódy použité ako inšpirácia pre implementáciu. Domnievam sa, že na rozdiel od klienta, pre server neexistujú žiadne knižnice napísané v jazyku C#. Preto bude serverová časť implementovaná od základov, s prihliadnutím na knižnice a voľne dostupné riešenia implementované v iných programovacích jazykoch.

5.4 Komunikácia

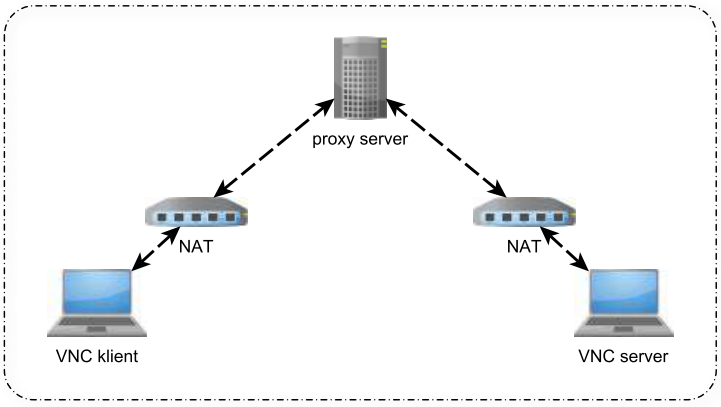
Prenos obrazu medzi VNC klientom a VNC serverom je zabezpečovaný protokolom RFB nad TCP/IP. Jedná sa o jednoduchý klient-server model komunikácie, kde server načúva na vopred známom porte a klient sa k nemu pripája, čo je zobrazené na obrázku 5.1. Toto riešenie pripojenia bude funkčné v rámci spoločnej siete alebo pokiaľ server disponuje verejnou IP adresou a my poznáme číslo portu, na ktorom naslúcha. Avšak pokiaľ sa v ceste nachádza router s NAT, komunikácia bude problematická.

5.4.1 Komunikácia za NATom

Ak sa v ceste komunikácie medzi klientom a serverom nachádza NAT smerovač, komunikácia nebude funkčná. Takéto zapojenie je zobrazené na obrázku 5.2. NAT je technika, ktorá sa používa pre zvýšenie počtu zariadení pripojených k internetu pomocou jednej IPv4 (Internet Protocol version 4) adresy a to tak, že za uzlom siete s jednou vonkajšou IPv4 adresou sa ukrýva niekoľko zariadení. Rozlíšenie zariadení sa uskutočňuje na základe portu. NAT teda mení internú IP adresu zariadenia na vonkajšiu IP adresu uzla (smerovača) s NAT a zároveň používa zmenu portu pre priradenie komunikácie k určitému zariadeniu. Pre každú odchádzajúcu komunikáciu sa v NAT vytvorí pravidlo pre prípad, že zariadenie prijímajúce dotaz bude chcieť odpovedať. Pri správach prichádzajúcich z internetu sa vyhľadáva pravidlo v NAT. Ak pravidlo existuje, správa sa dostáva ďalej do internej siete ku



Obr. 5.2: Zapojenie s NAT



Obr. 5.3: Zapojenie s NAT a proxy serverom

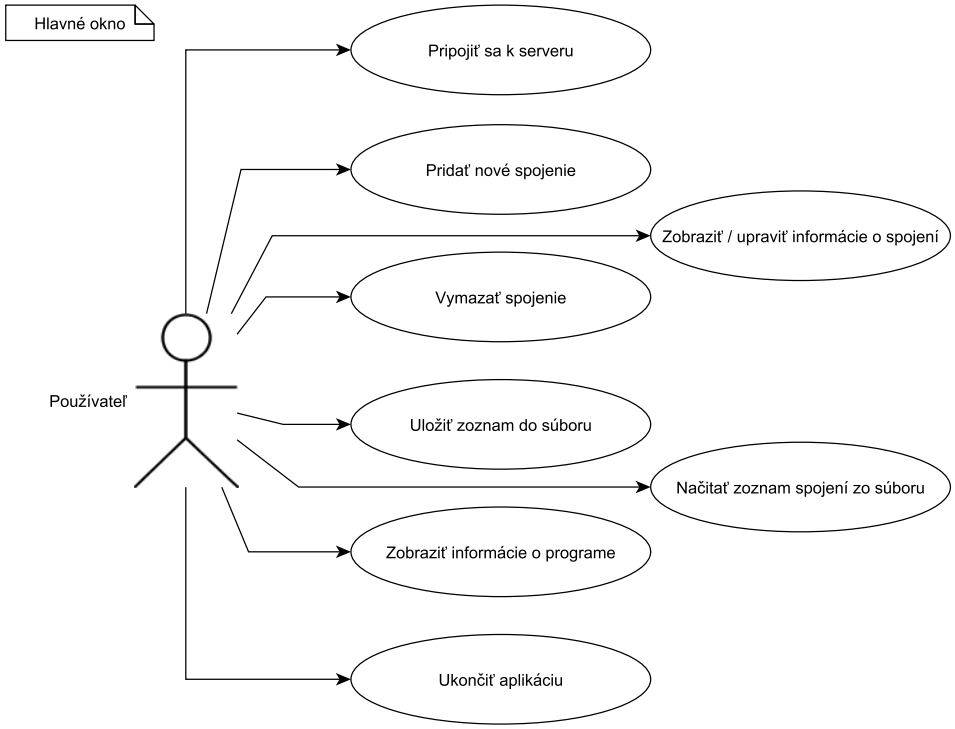
odpovedajúcemu zariadeniu. Ak pravidlo neexistuje, správa sa zahadzuje.

Riešenia komunikácie za NAT je niekoľko. Najjednoduchším riešením je manuálne nastavenie smerovača, aby komunikáciu pre určitý port smeroval na určité zariadenie v internej sieti. V anglickej terminológií sa táto technika nazýva „port forwarding“. Táto technika bude použitá aj v prípade tejto práce.

Ďalšou možnosťou je použitie tzv. proxy serveru, čo je zobrazené na obrázku 5.3. Proxy server musí disponovať verejnou IP adresou, aby sa na neho mohli pripájať zariadenia za NAT. V prípade použitia tejto metódy sa najskôr VNC server pripojí k proxy serveru, pričom je medzi nimi neustále udržovaný komunikačný kanál. Proxy server si udržuje databázu pripojených serverov. VNC klienti sa takisto pripájajú k proxy serveru a posielajú mu identifikáciu, s ktorým serverom by radi nadviazali sedenie. Proxy server teda slúži ako prostredník v komunikácií, teda smeruje správy medzi pripojenými zariadeniami.

5.5 Klient

Klient je časť aplikácie, ktorá je spustená na počítači, z ktorého sa používateľ pripája k vzdialenému počítaču – serveru. Návrh klientskej časti je veľmi dôležitý hlavne preto, že používateľ s ňou príde najviac do styku. Jej ovládanie by malo byť čo najjednoduchšie, no zároveň by malo poskytovať dostatočné možnosti nastavení parametrov prenosu. Klientská



Obr. 5.4: Hlavné okno klienta

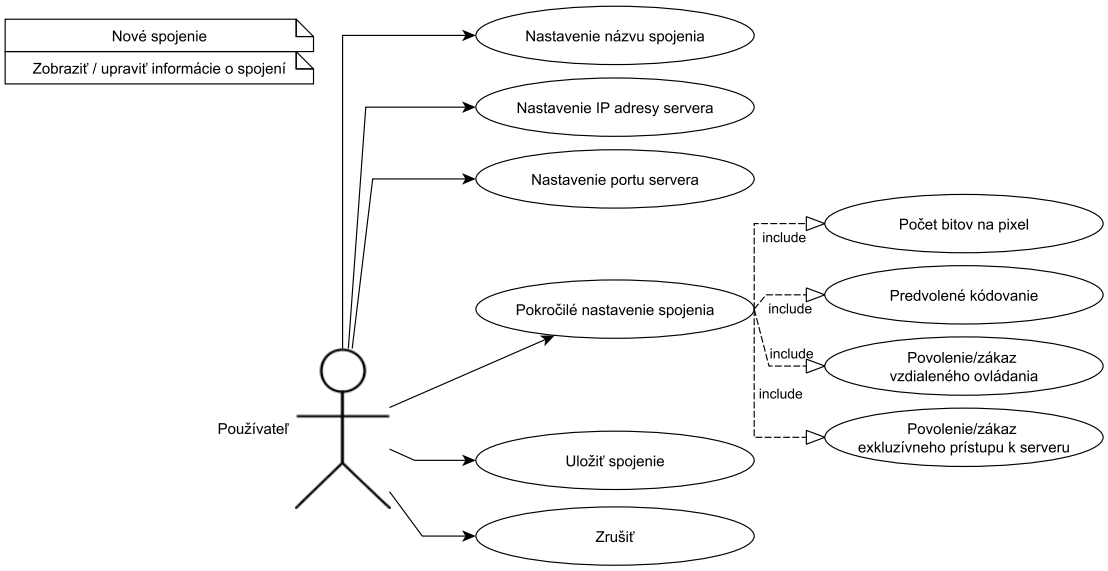
časť bola vytvorená v programovacom jazyku C# a v prostredí Visual Studio 2012. Pre vývoj prototypu a konečného dizajnu bol použitý nástroj Microsoft Expression Blend 4. Je to WYSIWYG (What You See Is What You Get) aplikácia, ktorá umožňuje definovať štýly, farby, umiestnenie objektov grafického používateľského prostredia. Jej ovládanie je dostatočne intuitívne a svojou jednoduchosťou pripomína ovládanie nástroja Adobe Photoshop. Pre ovládanie a definovanie štýlov nie je potrebné poznať jazyk XAML. Blend tak isto podporuje WPF. Na zoznámenie sa s týmto nástrojom boli využité rôzne články a návody[4], ako aj rôzne videá, ktoré zoznamovali používateľov, začiatočníkov, s fundamentálnymi princípmi práce s aplikáciou.

5.5.1 Diagram prípadov použitia klientskej časti

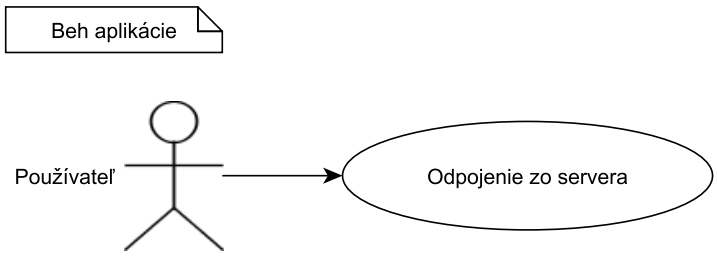
Diagram na obrázku 5.4 zobrazuje voľby použitia aplikácie po spustení z jej hlavného okna. Je možné sa priamo pripojiť k serveru, prípadne pridať spojenie do zoznamu spojení aplikácie a následne sa k nemu pripojiť. Je možné tak isto upravovať nastavenia spojenia zo zoznamu spojení, prípadne spojenie odstrániť. Ďalej je tu možnosť uloženia zoznamu spojení do súboru, prípadne nahranie zo súboru. Nakoniec je možné aplikáciu ukončiť.

Obrázok 5.5 znázorňuje možnosti výberu nastavení spojenia pri vytváraní položky spojenia do zoznamu spojení, prípadne možnosti pri otvorení úpravy nastavení existujúceho spojenia v zozname.

Obrázok 5.6 zobrazuje akcie, ktoré je možné vykonať pri aktívnom spojení so serverom. Počas aktívneho spojenia je možné zrušiť spojenie, teda odpojiť sa od servera.



Obr. 5.5: Nové spojenie klienta



Obr. 5.6: Beh aplikácie klienta

5.5.2 Objektový návrh klientskej časti

Pri objektovom návrhu klientskej časti aplikácie bolo čiastočne vychádzané z existujúcich riešení, ako napríklad VncSharp[15], ktorý je dostupný pod licenciou GPL[1]. Táto implementácia, no aj mnoho ďalších open source implementácií v rôznych programovacích jazykoch nepodporovali kódovania dát prenosu, ktoré sú kľúčové pre maximálne využitie možností serveru. Na tieto aplikácie bolo hladené skôr ako na všeobecné vzory implementácie klienta, pričom viacero častí bolo či už pozmenených, alebo od základov vybudovaných.

Nasleduje stručný popis vytvorených tried, ich funkcia, implementácia a integrácia s ostatnými triedami klientskej časti. Popísaný bude taktiež tok dát aplikáciou od prijatia od serveru až po konečné spracovanie prijatých dát.

GUI (Graphic user interface)

Grafické používateľské prostredie je pre klientskú časť veľmi dôležité, a to hlavne preto, že je to prvá časť projektu, s ktorou používateľ prichádza do styku. Preto by GUI malo byť jednoduché, prehľadné a malo by zaujať používateľa. Zároveň by malo poskytovať dostatočnú ponuku zmien a nastavení vzhľadom na možnosti a funkcie aplikácie. Na vytvorenie vzhľadu aplikácie bol použitý nástroj Microsoft Expression Blend 4 a počet a voľby nastavení vyplývajú z použitej metódy prenosu obrazu.

Používateľské prostredie je zložené z piatich okien, ktoré sa v priebehu práce s aplikáciou môžu zobraziť. Diagram tried je možné nájsť v prílohách na obrázku A.2. Hlavné okno tvorí trieda *MainWindow*, ktorá sa zobrazí používateľovi hneď po spustení aplikácie. Väčšiu časť okna zaberá panel so zoznamom spojení. Tento zoznam je implementovaný ako inštancia triedy *TrulyObservableCollection*, v ktorej sú použité mechanizmy pre identifikáciu zmeny nastavení spojenia v zozname a automatická aktualizácia zobrazenia. Pre pridávanie spojenia do zoznamu a pre zmenu nastavení spojenia v zozname slúži okno z triedy *AddEditConnectionWindow*, v ktorom sú nastavenia rozdelené na základné a rozšírené.

Pre pripojenie k serveru slúži metóda *Connect()* triedy *VNCClient*, ktorá má dve možnosti volania, a to s parametrami značiacimi IP adresu a port, alebo s parametrom typu *ConnectionItem*. Trieda *ConnectionItem* obsahuje ako aj všetky potrebné, tak aj voliteľné možnosti pri ustanovení spojenia.

Spojenia v zozname spojení je možné tiež exportovať, prípadne importovať do/zo súboru. Uloží sa tým celý zoznam spojení a to vo formáte XML. Pre uloženie týchto možností spojenia sa využíva serializácia do XML a pre načítanie zo súboru zase deserializácia, teda operácia opačná.

Uložený súbor má nasledujúci formát:

```
<?xml version="1.0" encoding="utf-8"?>
<ArrayOfConnectionItem
 xmlns:xsd="ttp://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <ConnectionItem>
      <Name>New_Connection</Name>
      <Address>127.0.0.1</Address>
      <Port>5900</Port>
      <Encoding>5</Encoding>
      <BitsPerPixel>32</BitsPerPixel>
```

```
    <ExclusiveAccess>false</ExclusiveAccess>
    <RemoteControl>false</RemoteControl>
  </ConnectionItem>
</ArrayOfConnectionItem>
```

Za zmienku stojí značka *Encoding*, ktorá udáva preferované kódovanie. Ďalšou značkou je aj *ExclusiveAccess*, ktorá značí, či bude mať používateľ výlučný prístup k serveru (server odpojí ostatných klientov).

Trieda VNCClient

Táto trieda slúži ako medzičlánok, ktorý upravuje parametre a vytvára inštanciu triedy *RFBClient*. Do konštruktora predáva ako parameter inštanciu triedy *ConnectionItem* (obrázok A.1), ktorá obsahuje potrebné parametre a tým inicializuje triedu. *VNCClient* následne volá metódy triedy *RFBClient*, a to metódu *ConnectAndHandshake()*, ktorá uskutoční pripojenie a handshake. Následne je volaná metóda *StartRemoteScreen()*, ktorou sa spustí spracovanie prijímaných aktualizácií zdieľanej obrazovky.

Trieda RFBClient

Trieda je hlavnou časťou celého klienta. Jej diagram je na obrázku A.3. Ako vidno na diagrame, obsahuje len dve verejné metódy (ak nerátame konštruktor). Prvá pre pripojenie k serveru a handshake a druhá pre spustenie prijímania a spracovania aktualizácií vzdialenej plochy.

Po úspešnom pripojení k serveru je pre komunikáciu vytvorený stream, ktorý je prispôsobený tak, aby pre čítanie dát používal *BinaryReader* a pri odosielaní správ používal *BinaryWriter*. Všetky tieto zmeny sú vytvorené v triede *VNCNetworkStream*. Trieda tak isto obsahuje metódy pre jednoduchšie odosielanie a prijímanie dát a prevod na rôzne dátové typy.

Následne je zahájená fáza nazývaná handshake. Pri nej si klient a server vymenia informácie špecifické pre daný protokol. Najskôr si server a klient porovnajú podporované verzie protokolu, pričom sa dohodnú na jednom, ktorý budú používať. Nasleduje bezpečnostná fáza, pri ktorej server odošle informácie o možnosti prihlásenia sa. Implementované možnosti sú dve, a to prihlásenie bez potreby autentizácie alebo prihlásenie pomocou VNC autentizácie. Pri použití VNC autentizácie server odosiela klientovi 16-bajtové náhodné číslo, ktoré sa spolu s heslom od klienta použije na DES autentizáciu. DES autentizáciu vykonáva trieda *DESAuthenticator* (obrázok A.1), ktorej metóda vracia 16-bajtové číslo, ktoré je následne odoslané naspäť serveru. Server ďalej informuje o úspechu, prípadne neúspechu prihlásenia. Nasleduje fáza inicializácie, kde klient zasiela správu o požiadavku na exkluzívny prístup k serveru (server odpojí ostatných klientov). Server zasiela správu s prednastaveným formátom posielaných dát u servera. Tým končí fáza handshake.

Pokračuje fáza normálnej komunikácie, kde si už server a klient vymieňajú správy potrebné pre správne zobrazenie zdieľanej plochy. Klient zasiela správu *SetEncodings*, ktorou serveru oznamuje podporované druhy kódovania. Ak túto správu nepošle, server posiela obraz v kódovaní *Raw*, čo je vyžadované protokolom RFB u všetkých klientov. Klient ďalej zasiela aj správu *SetPixelFormat*, ktorou môže zmeniť formát dát obrazu a tým aj potrebnú

šírku pásma siete. Klient v tejto fáze zasiela aj správy *FramebufferUpdateRequest*, ktorými od serveru požaduje zaslanie dát obrazu zdieľanej plochy.

Po zavolaní verejnej metódy *StartRemoteScreen()* začne klient spracovávať správy prichádzajúce od servera. V tejto fáze sa vytvorí okno pre zobrazenie prijatého obrazu a tak isto sa spustí samostatné vlákno, ktoré vykonáva samotné spracovanie obrazu. Samotné spracovanie správ je vykonávané v triede *ProcessingMessages*.

Trieda ProcessingMessages

V tejto triede sa nachádza metóda *ProcessIncomingMessages()*, ktorá je spúšťaná v samostatnom vlákne. Táto metóda spracováva správy zo servera, a to správy *FramebufferUpdate*, *SetColourMapEntries* a *Bell*. V prípade, že klient potrebuje obnoviť svoju plochu, zasiela serveru správu *FramebufferUpdateRequest*. To všetko sa vykonáva práve v tejto triede. Diagram je možné nájsť na obrázku A.3.

Zo správy *FramebufferUpdate* je najskôr zistené kódovanie, podľa ktorého sa rozhodne o čítaní a spracovávaní ďalších dát. Funkcia volá statické metódy konkrétneho dekodéru a ten po spracovaní vracia dáta vo formáte, ktorý je jednoduché zobraziť.

Dekodéry

Tieto triedy slúžia na dekódovanie snímku plochy z prichádzajúcich paketov. Je tu využitý spoločný predok, a to trieda *VNCDecoderPixel*, ktorá je definovaná ako abstraktná trieda. To zaistí, že všetci potomkovia budú musieť implementovať nasledujúce metódy:

public virtual int[] Decode(PixelFormat pixelFormat, Int32Rect rect);

public abstract int[] DecodeIndexed(PixelFormat pixelFormat, Int32Rect rect, int[] colourMap);

Obe metódy slúžia na prijatie dát od servera, spracovanie dát do formy pixelov a zreťazenie do pola čísiel typu integer. Toto pole už je jednoduché nakopírovať na potrebné miesto v pamäti pre zobrazenie.

Rozdiel v týchto metódach je v tom, že *Decode()* je určená pre dekódovanie obrazu, kde sú pixely reprezentované pomocou „true color“ a *DecodeIndexed()* dekóduje obraz, pri ktorom sú pixely reprezentované pomocou mapy farieb, pričom prijímaný je iba index do tabuľky farieb. „True color“ znamená, že z prijatých dát je možné pomocou operácií ako logický súčet, logický súčin, či bitový posuv získať priamo tri zložky farby – červenú, zelenú a modrú. Bez použitia „true color“ je potrebné, aby server zaslal mapu farieb a to pomocou správy *SetColourMapEntries()*. Tým sa vytvorí tabuľka farieb, z ktorej sa pri spracovaní samotných farieb pixelov čerpá.

Výsledkom tohto návrhu je, že aj napriek rozdielnemu kódovaniu a spracovaniu dát je navrátený vždy rovnaký formát. To uľahčuje prácu s potomkami, pretože je možné s nimi pracovať jednotne. Protokol RFB definuje celú radu algoritmov pre kódovanie snímok plochy. V tejto práci boli implementované tieto:

- *RAW* kódovanie
- *RRE* (rise-and-run-length encoding) kódovanie
- *CopyRect* (copy rectangle) kódovanie
- *Hextile* kódovanie

Všetky kódovania okrem *CopyRect* majú vlastné triedy, ktoré dedia od triedy *VNCDecoderPixel*, čo je možné vidieť na diagrame A.1.

Pseudo-kódovanie

Pseudo-kódovanie je využité iba jedno, a to *DesktopSize*. To umožňuje meniť veľkosť zdieľanej plochy počas behu programu. Implementácia je nie príliž triviálna, pretože prijatím správy s týmto pseudo-kódovaním je nutné zmeniť taktiež veľkosť framebuffera. To znamená, že je potrebné zrušiť existujúci framebuffer a nahradiť ho novým. Pre správne fungovanie je nutné synchronizovať spracovanie správ servera so zmenou veľkosti framebuffera a následným odosielaním žiadostí o obnovu obrazu. Zároveň je potrebné zažiadať o obnovu celého obrazu zdieľanej plochy.

5.6 Server

Server je časť systému, ktorá je spustená na vzdialenom stroji, ktorého plochu chceme zdieľať. Preto je jeho rola pasívnejšia čo sa týka používateľovho zásahu do aplikácie. Po spustení server naslúcha na dopredu známom, používateľom nastavenom porte a čaká na prichádzajúce spojenia. Čo sa týka používateľského rozhrania, je minimalistické a väčšinou si vystačí s predvolenými nastaveniami. Navzdory tomu, že server má iba pasívnu rolu, je jeho úloha oveľa zložitejšia. Server prijíma spojenia od viacerých klientov a sa stará o komunikáciu s nimi. Po úspešnom spojení a po inicializačnej fáze komunikácie sa server musí taktiež starať o odosielanie, správu a spracovanie pracovnej plochy a jej framebufferu.

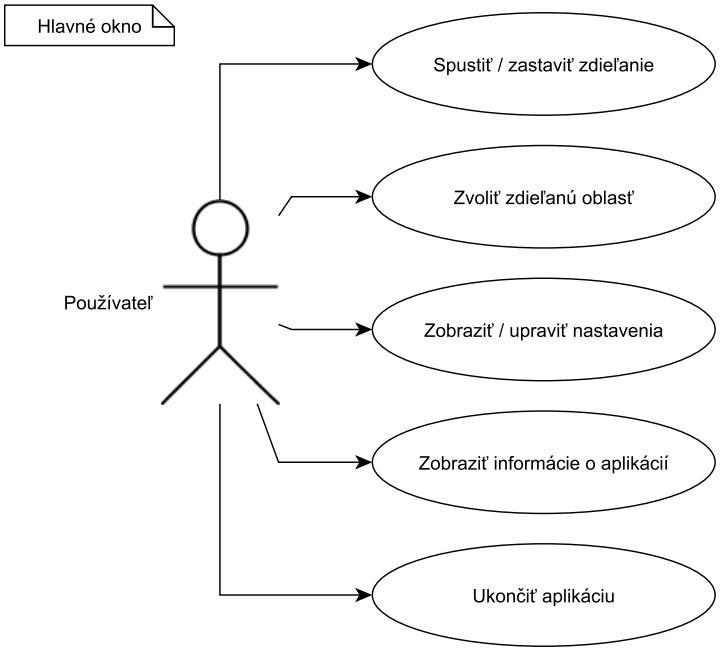
5.6.1 Diagram prípadov použitia serverovskej časti

Používateľské prostredie pre server je jednoduchšie, no funkcia serveru je o to zložitejšia. Diagram na obrázku 5.7 zobrazuje možnosti používateľa po spustení aplikácie. Používateľ má možnosti ako spustenie načúvania servera, výber zdieľanej oblasti, zmena nastavení servera a na koniec ukončenie aplikácie.

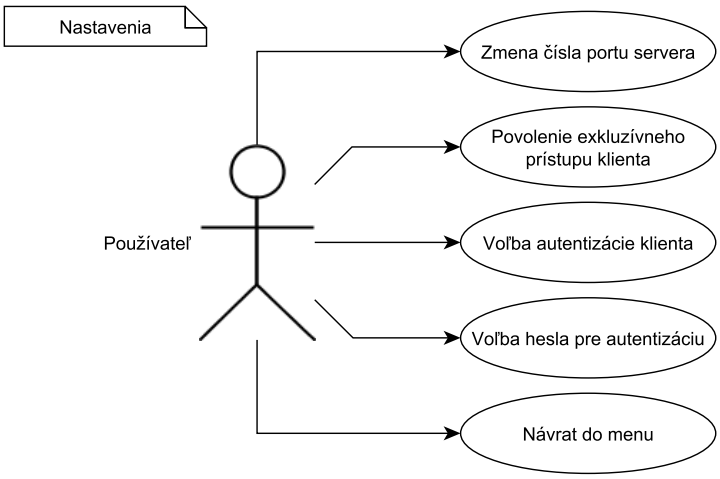
Po zvolení výberu zdieľanej oblasti sa používateľovi objaví transparentné okno, ktorým nastavuje požadovanú veľkosť zdieľanej plochy. Na tomto okne má používateľ možnosť zmeniť súradnice a rozmery posielaného okna.

Pri voľbe zmeny nastavení sa tak isto otvorí okno, ktoré umožňuje zmeniť hlavné nastavenia serveru. Možnosti tohto okna zobrazuje diagram na obrázku 5.8.

Po spustení načúvania začne server prijímať spojenia od klientov. V tejto fáze je možné už len zastaviť zdieľanie, prípadne presunúť či zmeniť veľkosť zdieľanej oblasti.



Obr. 5.7: Hlavné okno serveru



Obr. 5.8: Zmena nastavení

5.6.2 Objektový návrh serverovskej časti

Projekt bol rozdelený do rôznych menných priestorov, keďže logické časti a funkcie servera boli ľahko identifikovateľné a rozdeliteľné. Výhodou rozdelenia do viacerých menných priestorov spočíva hlavne v možnosti neskoršej náhrady časti projektu ako celku, ako aj jednoduchšie porozumenie častiam servera a správe kódu. Menné priestory, do ktorých bol projekt rozdelený sú:

- *ServerPart*,
- *RfbModule*,
- *VncServer*,
- *VncScreen*,
- *VncEncodings*,
- *MyImaging*.

Grafické používateľské prostredie - ServerPart

Tento menný priestor v sebe obsahuje celé grafické používateľské prostredie a obsahuje definície všetkých okien, s ktorými príde používateľ do styku. Nastavenia serveru sa ukladajú do objektu typu *UserOptions*, ktorým sa jednoducho predávajú nastavenia do dátovej časti aplikácie. Serverová časť aplikácie bola vyvíjaná s ohľadom na návrhový vzor Model View ViewModel (MVVM)[12], čo je návrhový vzor vytvorený firmou Microsoft. Základ vzoru spočíva v snahe o jasné oddelenie vývoja grafického používateľského rozhrania od vývoja biznis logiky alebo logiky známej ako model dát. Grafické používateľské prostredie (View) je implementované pomocou framework-u využívajúceho značkový jazyk XAML. V ňom vytvára spojenia (tzv. bindings) na ViewModel. ViewModel obsahuje všetku logiku, potrebnú pre chod aplikácie a obsahuje taktiež všetky premenné a dáta potrebné pre binding. ViewModel slúži ako prostredník medzi View a Model, no má charakter viac ako model. Model obsahuje celú logiku programu a do ViewModel posiela iba dáta potrebné pre zobrazenie používateľovi.

Hlavným vstupným bodom aplikácie je trieda *App*, v ktorom je instanciovaná trieda *WindowViewModel*, ktorá už podľa názvu slúži ako ViewModel pre všetky okná aplikácie. Ďalej nasleduje vytvorenie objektov všetkých okien, ktorým je ViewModel objekt posielaný ako parameter. Na koniec začína aplikácia spustením hlavného okna, a to je okno reprezentujúce triedu *MenuWindow*.

Za zmienku stojí tak isto aj okno triedy *SnapWindow*, ktoré slúži ako okno pre výber časti pracovnej plochy, ktorá bude zdieľaná. Toto okno je priehľadné a jeho veľkosť je modifikovateľná, čím sa vďaka data bindingu menia aj súradnice v modeli. Diagram tried je možné nájsť na obrázku A.4.

Základné funkcie a údaje protokolu RFB - RfbModule

Tento menný priestor obsahuje triedy a štruktúry zobrazujúce základné vlastnosti protokolu RFB. Nachádza sa tu taktiež statická trieda *DESAuthenticator*, ktorá slúži na vytvorenie kľúča pre prihlásenie pomocou metódy DES. Ďalej sa tu nachádza napríklad trieda *RfbMsgCreator*, ktorá je takisto statická. Obsahuje statické funkcie pre tvorbu správ pre komunikáciu servera s klientom a statické funkcie pre tvorbu a inicializáciu štruktúr ako *ServerInit*, či *PixelFormat*. Diagram tried tohto modulu je na obrázku A.5.

Sieťová komunikácia - VncServer

Menný priestor *VncServer* obsahuje triedy pre komunikáciu cez sieť, spracovanie správ a nastavenie ďalších častí programu podľa prijatých dát. Trieda *TcpServer* obsahuje implementáciu konkurentného TCP IPv4 serveru. K tomuto serveru je možné pripojiť až 100 klientov. Server využíva vlákna pre rýchlejšie spracovanie a obsluhu všetkých klientov rovnomerne. Pri pripojení klienta sa vytvorí nový objekt triedy *RfbServiceProvider*. Táto trieda dostáva ako vstup nastavenia od používateľa. Udržuje nižšiu vrstvu a volá jej metódy pre získanie obrazu. Pri nadviazaní spojenia s klientom sa musia server a klient dohodnúť na verzii protokolu, ktorý bude použitý. Ďalej nastáva handshake fáza, kde na začiatku server rozhodne o forme autentizácie. Po úspešnej autentizácií si klient a server medzi sebou vymenia informácie o veľkosti zdieľanej plochy a o formáte prenosu obrazu a podporovaných kódovaniach obrazu. Po úspešnej fáze dohodnutia sa na prenose sa prechádza do fáze normálnej komunikácie. V tejto fáze si klient a server vymieňajú informácie o samotnom obraze. Klient zasiela správy s požiadavkou na obnovu obrazu a trieda *RfbServiceProvider* predáva túto požiadavku na nižšiu vrstvu, ktorá jej obratom vracia správu, ktorú je potrebné zaslať klientovi ako odpoveď. Diagram tried je na obrázku A.6.

Spracovanie obrazu pracovnej plochy - VncScreen

Tento menný priestor obsahuje jednu triedu, a to triedu *MyScreen*. Táto trieda udržiava informácie z inicializačnej fáze komunikácie, potrebné pre prenos obrazu, ako aj zoznam kódovaní podporovaný pripojeným klientom a aj lokálny framebuffer. Ten obsahuje duplikát obrazu, ktorý je aktuálne zobrazený u klienta. Takto môže klient posielať požiadavky len na obnovu obrazu, ktorý sa zmenil od poslednej aktualizácie, namiesto neustáleho zasielania celého zdieľaného obrazu. Pre získanie obrazu požíva trieda nižšiu úroveň spracovania obrazu, ktorá vracia potrebný formát. Diagram tried je v prílohe na obrázku A.7.

Kódovanie obrazu – VncEncodings

Obsahuje triedu pre kódovanie obrazu, a to triedu *VncEncoderRaw*. Implementované bolo len toto základné kódovanie, no je tu možnosť pridať ďalšie, čo môže byť námetom na pokračovanie v tomto projekte. Táto trieda je statická a obsahuje statické funkcie pre kódovanie obrazu vo formáte *Raw* podľa špecifikácie RFB. Jedna metóda slúži pre zostavenie správy z obrazu celej plochy, druhá pre zostavenie správy z rozdielu medzi aktuálnou plochou a framebufferom. Táto úloha nie je vôbec triviálna a potrebuje ďalšie revízie. Problémom je

vyhľadanie zmenenej oblasti obrazu a samotný fakt, že táto operácia trvá príliš veľa času. Pri pripojení viacerých klientov je optimalizovanie tejto časti kódu veľmi žiadané, pretože v aktuálnej verzií nie je použité spracovanie rozdielu obrazov v samostatnom vlákne a tak isto použitie nového vlákna pre tento výpočet. Využitím a zmenou kódu a tejto časti programu by bolo možné dosiahnuť ďalšie zrýchlenie aplikácie. To môže slúžiť ako ďalšia rada a inšpirácia pre pokračovanie vo vývoji tejto aplikácie a jej podpore konkurencieschopnosti medzi existujúcimi aplikáciami tohto druhu. Pre spracovanie obrazu a prevod do správneho formátu je obraz posúvaný do nižšej vrstvy na to určenej. Tá vracia žiadaný obraz v správnom formáte.

Získanie a spracovanie obrazu – MyImaging

Priestor obsahuje triedu *ImagingClass*, ktorej metódy sú všetky statické. Z pomedzi nich je možné spomenúť metódy, ako napríklad metóda na získanie aktuálneho obrazu pracovnej plochy, prípadne jej časti, či kopírovanie obrazu do iného obrazu, alebo zmenšenie obrazu. Ďalším členom je napríklad metóda pre porovnanie dvoch obrazov s návratom súradníc s plochou, v ktorej sa odlišujú. Diagram tried je možné nájsť v prílohách na obrázku A.7.

Hlavnými metódami sú ale tie na spracovanie obrazu do formátu vyžadovaného klientom. Formát výstupných dát je predávaný do metód ako parameter, preto môžu byť všetky metódy statické. Pre urýchlenie tejto časti kódu bol použitý tzv. unsafe kód, ktorý umožňuje pracovať s ukazovateľmi na hodnoty a tým urýchľuje kopírovanie a porovnávania a prácu s dátami. Tento prístup sa osvedčil ako najrýchlejší. Ďalšou možnosťou bolo využitie externých knižníc, no vytvorenie týchto funkcií bolo triviálnou záležitosťou a pre chod aplikácie je ich výkonnosť dostačujúca. Využitie ďalších knižníc a zrýchlenie procesu tvorby dát je dobrou možnosťou pokračovania tohto projektu.

Kapitola 6

Overenie funkčnosti

Po dokončení vývoja aplikácie je vždy potrebné overiť, že vyvíjaná aplikácia funguje správne a podľa očakávaní. Obe časti aplikácie – klient aj server – boli testované na rôznych systémoch. Najjednoduchší test funkčnosti, ktorý existuje pre túto aplikáciu je spustenie servera a pripojenie sa k nemu klientom. Ak u klienta uvidíme snímok vybranej oblasti plochy zo servera a pozorujeme aj zmenu tohto snímku v závislosti od zmeny plochy počítača, kde server beží, tak môžeme predpokladať, že aplikácia funguje správne.

6.1 Testovanie na platforme Windows

Aplikácia bola vyvíjaná na platforme Windows (Windows 8 Pro), kde bolo tak isto vykonané testovanie aplikácie. Aplikácia bola testovaná tak isto aj na operačnom systéme Windows 7. V priebehu vývoja aplikácie boli testované všetky jej súčasti. Spojenie je správne nadviazané, komunikácia medzi klientom aj serverom je plne funkčná. Správy sú zasielané správne, v správnom formáte, podľa špecifikácie protokolu RFB. Snímanie, zasielanie plochy a vykresľovanie funguje správne. Zobrazovanie dialógových okien ako aj prekresľovanie snímok funguje takisto správne.

6.2 Testovanie na platforme Linux

Na tejto platforme aplikácia testovaná nebola aj napriek tomu, že bolo spomenuté, že knižnica .NET a aplikácie založené na nej je možné spustiť aj na platforme Linux a to vďaka projektu Mono. Hlavnou úlohou aplikácie nie je jej prenositeľnosť ale jej testovanie a porovnanie funkčnosti na systéme Windows oproti ďalším aplikáciám, ktoré ponúkajú prenos obrazu pomocou protokolu RFB.

6.3 Náročnosť na zdroje

S vývojom súvisí tak isto testovanie vplyvu aplikácie na systémové zdroje, prípadne na sieťové prostriedky. Ako už bolo spomenuté, serverová časť aplikácie je veľmi náročná na systémové zdroje. Server musí neustále snímať obraz plochy, spracovávať, správne ich kódovať a zasielať klientovi. Tento neustály výpočet zaťažuje procesor počítača. Na rozdiel od servera je klientska časť menej náročná na výpočetný výkon. To vďaka tomu, že klient len prijíma správy, dekóduje a vykresľuje ich, čo sú operácie, ktoré natoľko nezaťažujú procesor.

Pre nasledujúce výpočty šírky pásma je predpokladaná obrazovka s rozlíšením 1366x768. Veľkosť posielaných dát závisí od zvoleného formátu pixelov. Prvý výpočet ukazuje veľkosť dát, ktoré sú potrebné preniesť na obnovu celej plochy obrazovky (rozlíšenie 1366x768) v kódovaní *RAW* a s formátom pixelov nastaveným na základnej hodnote serveru (jeden pixel je vyjadrený pomocou 32 bitov, pričom farebná hĺbka je nastavená na 24 bitov na pixel). Pri danom nastavení kódovania a formáte pixlov je veľkosť dát potrebných pre prenos:

$$1366 * 768 * 32b = 33570816b = 4196352B = 4098kB \doteq 4,002MB$$

Pre docielenie plynulého obrazu je nutné posielať snímok plochy aspoň 15 krát za sekundu. To by činilo dokopy šírku prenosu:

$$4,002MB * 15/1s \doteq 60,03MB/s$$

Takto rýchla linka nemusí byť k dispozícii v lokálnej sieti, nieto ešte na vzdialenej sieti. Preto je možné využiť niektoré z úspornejších kódovaní, prípadne je možné zmeniť formát reprezentácie pixelov. To by zmenšilo počet zasielaných bitov na pixel a teda aj zmenšilo veľkosť prenášaných dát. Ak by bol pre prenos použitý formát pixlov, kde by bol jeden pixel reprezentovaný len 8 bitmi, veľkosť jedného snímku plochy by bol:

$$1366 * 768 * 8b = 8392704b = 1049088B = 1024,5kB \doteq 1,0005MB$$

Použitie tohto formátu reprezentácie pixelov by teda zmenšilo veľkosť prenášaných dát takmer na štvrtinu. Opäť pre plynulý prenos by to činilo:

$$1,0005MB * 15/1s \doteq 15,007MB/s$$

Výpočet plynulého prenosu je ale za predpokladu, že každý snímok je posielaný s maximálnou veľkosťou, teda 1366x768. V reálnom systéme sú ale zasielané len plochy, na ktorých prebiehala nejaká zmena, čo výrazne zmenšuje veľkosť dát.

Na veľkosť posielaných dát vplýva taktiež použité kódovanie. Porovnanie rôznych kódovaní (pre tento projekt zaujímavé porovnanie *RAW* a *Hextile*) je možné nájsť na stránke [18]. Vykonaná bola rada testov s použitím rôznych formátov reprezentácie pixlov a viacerých podmienok a úkonov počas zdieľania plochy.

Kapitola 7

Možnosti rozšírenia

Obe časti projektu sú plne funkčné a splňujú svoju úlohu. Vždy je ale možné nájsť oblasti, v ktorých je možné projekt rozšíriť a tým zlepšiť. V tomto projekte existuje taktiež rada možností rozšírenia aplikácií, prípadne ich zefektívnenia.

7.1 Zníženie zaťaženia zdrojov

Ako bolo spomínané, práca serveru je veľmi náročná na zaťaženie procesoru, a to kvôli neustálym výpočtom. Po pozorovaní aplikácie za chodu je možné postrehnúť toto obmedzenie veľmi jasne, pričom podobné aplikácie pre zdieľanie plochy nemajú podobné zaťaženia. Príčinou je rozdiel v algoritme pre výpočet inkrementálneho snímku zdieľanej plochy.

V aktuálnej implementácií servera sa zaťaženie zdrojov prejaví najviac pri viacnásobnom zdieľaní plochy. Pre výpočet zmeny plochy totižto nie je použité samostatné vlákno, ktoré by počítalo inkrementálny snímok centrálne a posielalo informácie o zmenách všetkým klientom. Namiesto toho sa pre každého klienta vytvorí samostatné vlákno a každé počíta inkrementálny snímok pre svojho pripojeného klienta.

Ďalšie zrýchlenie a obmedzenie použitia zdrojov by bolo možné nahradením samotného algoritmu pre výpočet. Ten je totižto vytvorený na základe porovnávania dát obrazu pixel po pixely, pričom na tento účel je možné nájsť funkcie aj vo Windows API. To ale znamená nutnosť dynamického linkovania knižníc DLL. Pre budúcnosť aplikácie a jej využitie v praxi je toto nutnosťou.

7.2 Ďalšie kódovania

So znížením výpočetnej náročnosti súvisí taktiež použité kódovanie obrazu. Ako bolo spomínané, server implementuje iba jedno kódovanie, a to kódovanie *RAW*. Toto kódovanie je z tohto hľadiska nevýhodné, pretože prenáša veľké objemy dát. Z testov[18], ktoré boli spomenuté skôr v tomto dokumente je zrejmé, že *RAW* kódovanie je nevýhodné.

Lepšie je na tom kódovanie *Hextile*, ktoré ponúka prenos menšieho objemu dát. V špecifikácií protokolu RFB sú definované ešte ďalšie kódovania. Jedná sa o *TRLE* (Tiled Run-Length Encoding) a *ZRLE* (Zlib Run-Length Encoding). Tieto kódovania sú odporúčané aj v samotnej špecifikácií protokolu RFB ako preferované kódovania, pretože ponúkajú lepšiu kompresiu dát.

7.3 Bezpečnosť

Bezpečnosť nie je silnou stránkou protokolu RFB. Špecifikácia tohto protokolu sa ňou moc nezaoberá. Ponúka len autentizáciu klienta voči serveru a aj to len s použitím šifrovania algoritmu DES. Prenos správ je nešifrovaný a je preto veľmi náchylný na útoky typu spoofing, prípadne sniffing.

Keďže tvorcovia protokolu RFB mysleli na rozšíriteľnosť, tak vyriešenie jednoduchosti šifrovania DES je veľmi jednoduché. Do protokolu je možné pridať ďalšie typy autentizácie klienta a to jednoduchou zmenou zaslaného čísla značiaceho typ použitej autentizácie. Naproti tomu je zlepšenie bezpečnosti prenosu správ takmer nemožné docieliť. Protokol RFB s týmto nerátal a preto je nutné siahnuť po ďalších riešeniach. Jednou z možností je tunelovanie cez SSH (Secure Shell). Výhodou tohto riešenia je, že nevyžaduje žiaden zásah do zdrojového kódu programu a taktiež ani žiaden zásah do protokolu RFB.

Kapitola 8

Záver

V tejto práci bol popísaný koncept zdieľania plochy, ktorý ponúka zobrazenie plochy vzdialeného počítača. Tiež boli uvedené možnosti použitia tohto konceptu. Časť práce sa venovala popisu existujúcich aplikácií ponúkajúcich prenos a zdieľanie obrazu vzdialeného počítača. Vysvetlené boli aj rozdiely medzi nimi a ich základné princípy, ku ktorím patrí aj protokol pre prenos samotného obrazu. Tých bolo spomenutých niekoľko, no detailnejšie boli popísané dva z nich – RDP a RFB protokol. Uvedené boli možnosti, ktoré tieto dva protokoly ponúkajú, a popísané bolo zároveň aj ich porovnanie.

Časť práce tvorí popis návrhu a realizácie vlastného riešenia v programovacom jazyku C#. Samozrejmosťou je rozdelenie a popis návrhu v modelovacom jazyku UML pre popis diagramov. Vytvorené riešenie využíva protokol RFB, ktorého základ je v práci uvedený a vysvetlený. Tento protokol umožňuje tiež vzdialené ovládanie, no táto funkcionalita nebola do implementácie zahrnutá.

Pozornosť bola venovaná aj popisu implementácie častí aplikácie – klienta aj servera. Vysvetlená bola štruktúra oboch častí aplikácie, rozdelenie na triedy a podúlohy. Časť práce bola venovaná aj správnej komunikácií v prípade výskytu NATu v sieti, pričom boli popísané možnosti riešenia tohoto problému.

Funkčnosť oboch aplikácií bola tiež overená a výsledky v tejto práci uvedené. Testovanie bolo vykonané na počítači s platformou Windows, na ktorej prebiehal aj vývoj aplikácie. Na platforme Linux aplikácia testovaná nebola, napriek tomu boli popísané možnosti prenesenia a spustenia aplikácie aj na tejto platforme. Vykonané tiež boli výpočty pre potrebnú šírku pásma pre prenos obrazu.

Práca je zakončená popisom možností pokračovania projektu. Aplikácia je porovnaná s existujúcimi komerčnými riešeniami a naznačené sú aj zmeny potrebné pre reálne využitie aplikácie. Bez týchto zmien nie je možné, aby mohla vytvorená aplikácia konkurovať iným riešeniam.

Voľba protokolu RFB sa ukázala ako správna, pretože v priebehu vývoja nedošlo k žiadnym neriešiteľným problémom. Jedná sa síce o protokol jednoduchý, no spĺňuje všetky potreby vytváranej aplikácie a ponúka možnosti rozšírenia. Výber programovacieho jazyka sa ukázal byť správny (minimálne pre klientskú časť aplikácie), a to vďaka jednoduchosti vytvorenia vhodného používateľského prostredia. Pre server bol využitý tento programovací jazyk ako experiment, či je v ňom možná implementácia serveru. Implementácia možná je, ale je veľmi diskutabilné, či je aplikácia v programovacom jazyku C# účinnejšia ako v ostatných jazykoch, keďže by server bolo nutné silne optimalizovať.

Literatura

[1] *GNU General Public License Versions* [online]. [cit. 8. května 2013]. Dostupné na: <http://opensource.org/licenses/gpl-license.php>.

[2] ITU-T. *Recommendation ITU-T T.128: Multipoint application sharing* [online]. 2008-06-13 [cit. 8. května 2013]. Dostupné na: <http://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-T.128-200806-I!!PDF-E&type=items>.

[3] *LibVNCServer/LibVNCClient* [online]. [cit. 8. května 2013]. Dostupné na: <http://libvncserver.sourceforge.net/index.html>.

[4] Microsoft Corporation. *Microsoft Expression Learning Resources* [online]. 2009-02 [cit. 8. května 2013]. Dostupné na: <http://msdn.microsoft.com/en-us/expression/cc136522.aspx>.

[5] Microsoft Corporation. *Remote Desktop Protocol* [online]. 2012-10-26 [cit. 8. května 2013]. Dostupné na: <http://msdn.microsoft.com/en-us/library/aa383015(v=vs.85).aspx>.

[6] Microsoft Corporation. *Understanding the Remote Desktop Protocol (RDP)* [online]. 2007-03-27 [cit. 8. května 2013]. Dostupné na: <http://support.microsoft.com/default.aspx?scid=kb;EN-US;q186607>.

[7] *Mikogo* [online]. [cit. 8. května 2013]. Dostupné na: <http://www.mikogo.com/>.

[8] *Mono* [online]. [cit. 8. května 2013]. Dostupné na: <http://www.mono-project.com/Main_Page>.

[9] *RealVNC* [online]. [cit. 8. května 2013]. Dostupné na: <http://www.realvnc.com/>.

[10] Richardson, T. a Levine, J. *The Remote Framebuffer Protocol* [online]. RFC 6143. IETF. 2011-03 [cit. 8. května 2013]. ISSN: 2070-1721. Dostupné na: <https://tools.ietf.org/html/rfc6143>.

[11] Richardson, T. *The RFB Protocol* [online]. Version 3.8. RealVNC Ltd. 2010-11-26 [cit. 8. května 2013]. Dostupné na: <http://www.realvnc.com/docs/rfbproto.pdf>.

[12] Smith, J. *WPF Apps With The Model-View-ViewModel Design Pattern* [online]. 2009-02 [cit. 8. května 2013]. Dostupné na: <http://msdn.microsoft.com/en-us/magazine/dd419663.aspx>.

[13] *TightVNC Software* [online]. [cit. 8. května 2013]. Dostupné na: <http://tightvnc.com/>.

[14] *UltraVNC Remote PC support* [online]. [cit. 8. května 2013]. Dostupné na: <http://www.uvnc.com/>.

[15] *VncSharp: A .NET VNC Client Library* [online]. [cit. 8. května 2013]. Dostupné na: <http://cdot.senecac.on.ca/projects/vncsharp/index.html>.

[16] Wikipedia, The Free Encyclopedia. *Comparison of remote desktop software* [online]. [cit. 8. května 2013]. Dostupné na: <http://en.wikipedia.org/wiki/Comparison_of_remote_desktop_software>.

Seznam zkratek

GPL – GNU General Public License

GUI – Graphical user interface

IANA – Internet Assigned Numbers Authority

IP – Internet Protocol

IPv4 – Internet Protocol version 4

ITU-T – International Telecommunication Union - Telecommunication Standardization Sector

LAN – Local area network

MVVM – Model View ViewModel

NAT – Network address translation

RDC – Remote Desktop Connection

RDP – Remote Desktop Protocol

RFB – Remote framebuffer

RRE – rise-and-run-length encoding

SSH – Secure Shell

TCP – Transmission Control Protocol

TRLE – Tiled Run-Length Encoding

VNC – Virtual Network Computing

VoIP – Voice over Internet Protocol

WPF – Windows Presentation Foundation

WYSIWYG – what you see is what you get

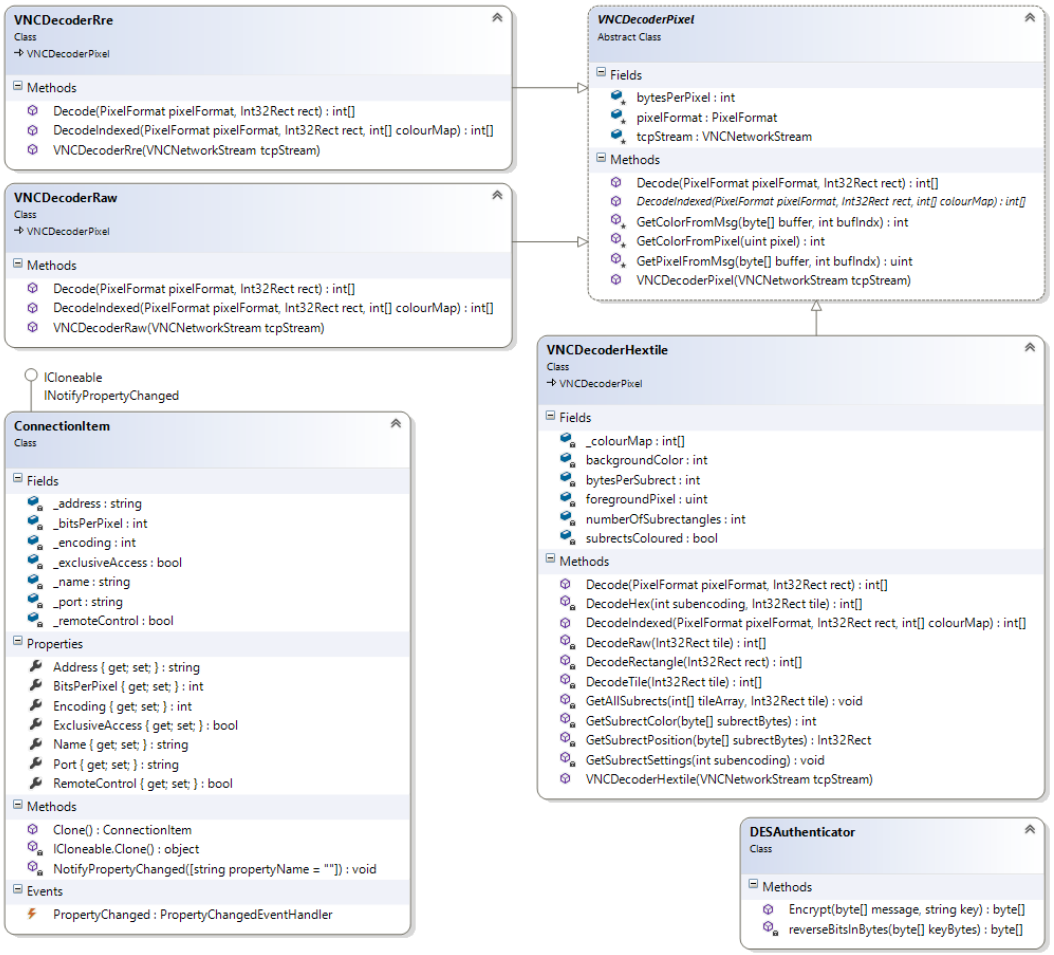
XAML – Extensible Application Markup Language

ZRLE – Zlib Run-Length Encoding

Příloha A

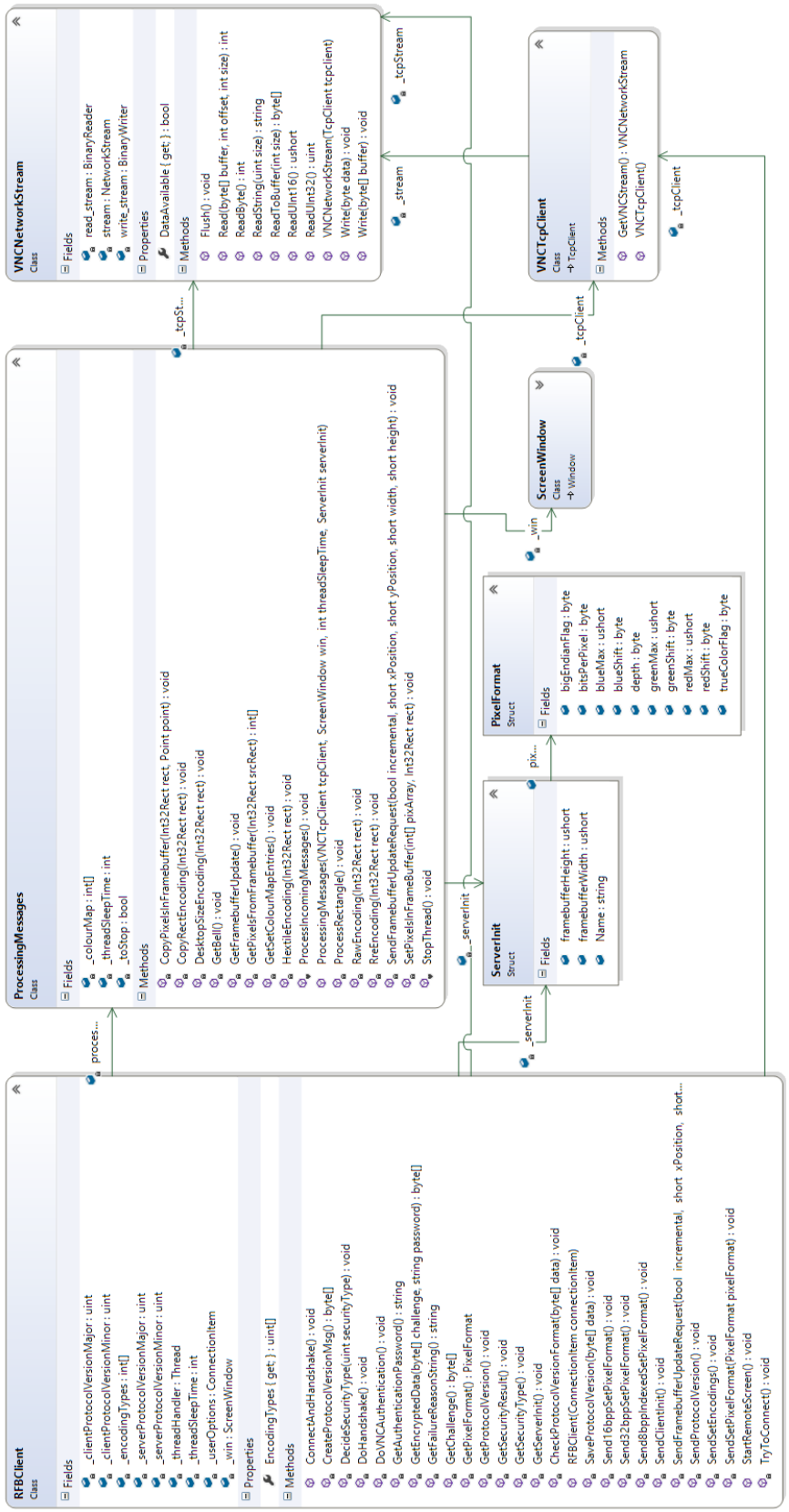
Diagramy tried

Klient



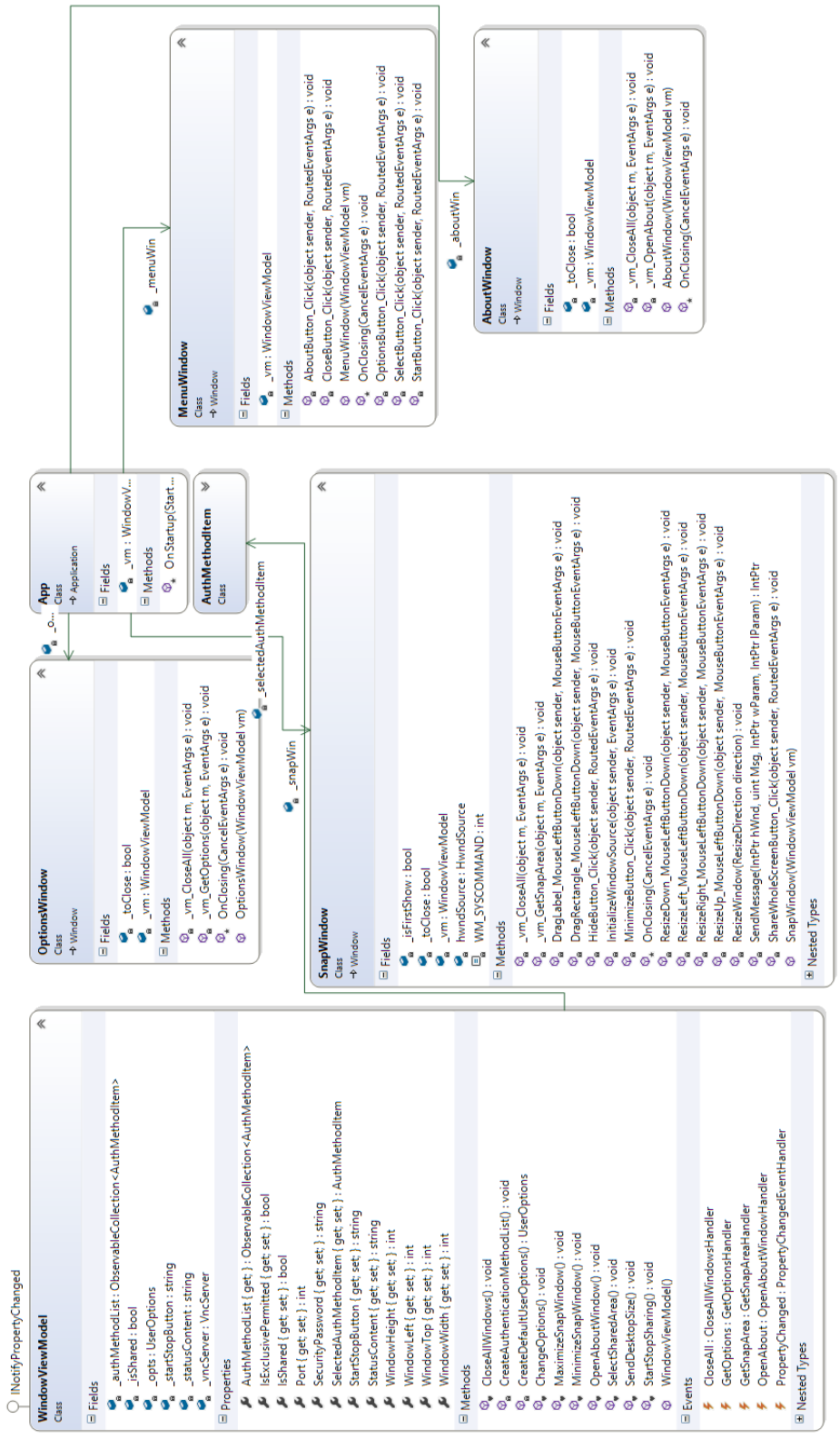
Obr. A.1: Triedy dekóderov, trieda ConnectionItem a DESAuthentication

Obr. A.2: Hlavné okno klienta

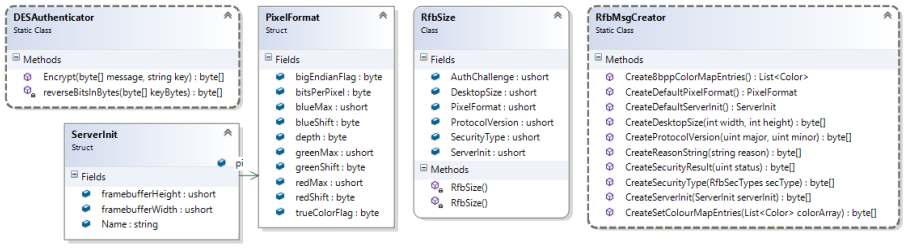


Obr. A.3: Trieda ProcessingMessages

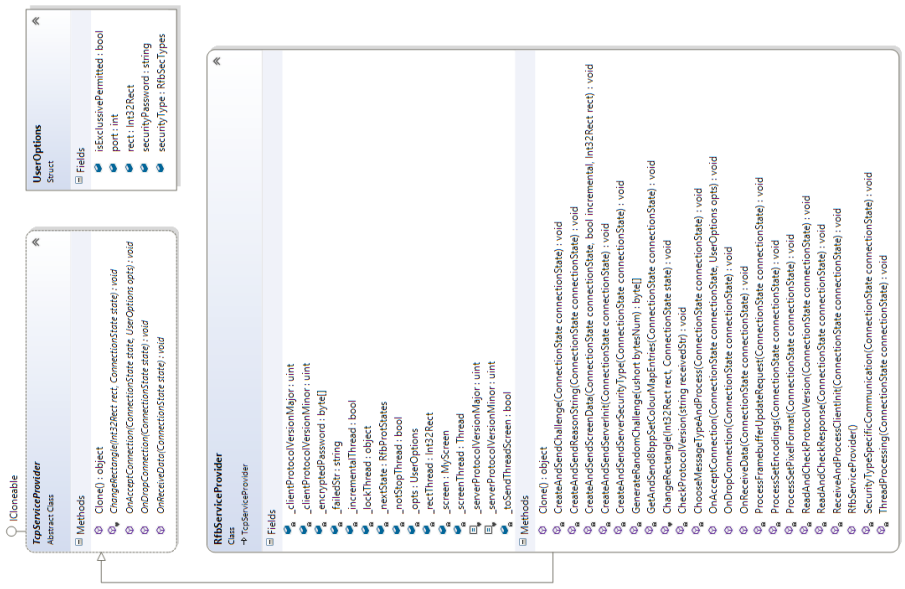
Server



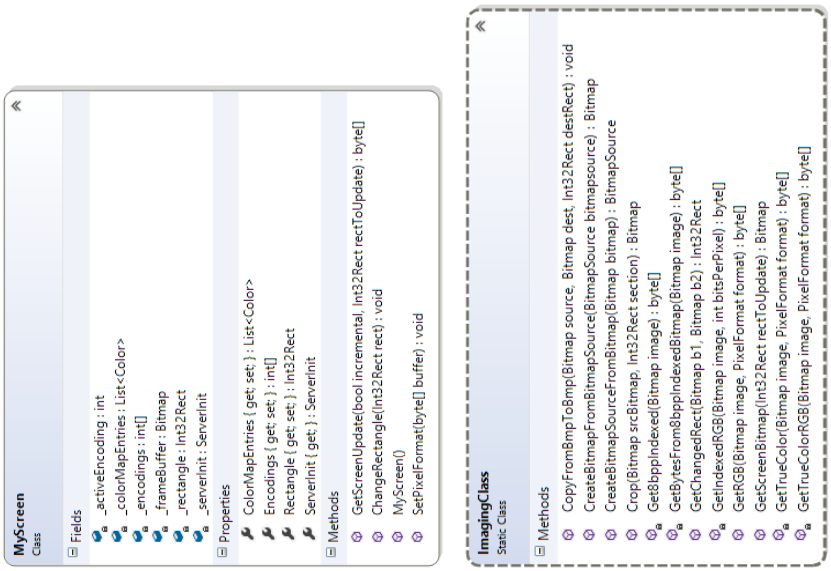
Obr. A.4: ViewModel a okná servera



Obr. A.5: Balík RfbModule



Obr. A.6: Balík VncServer

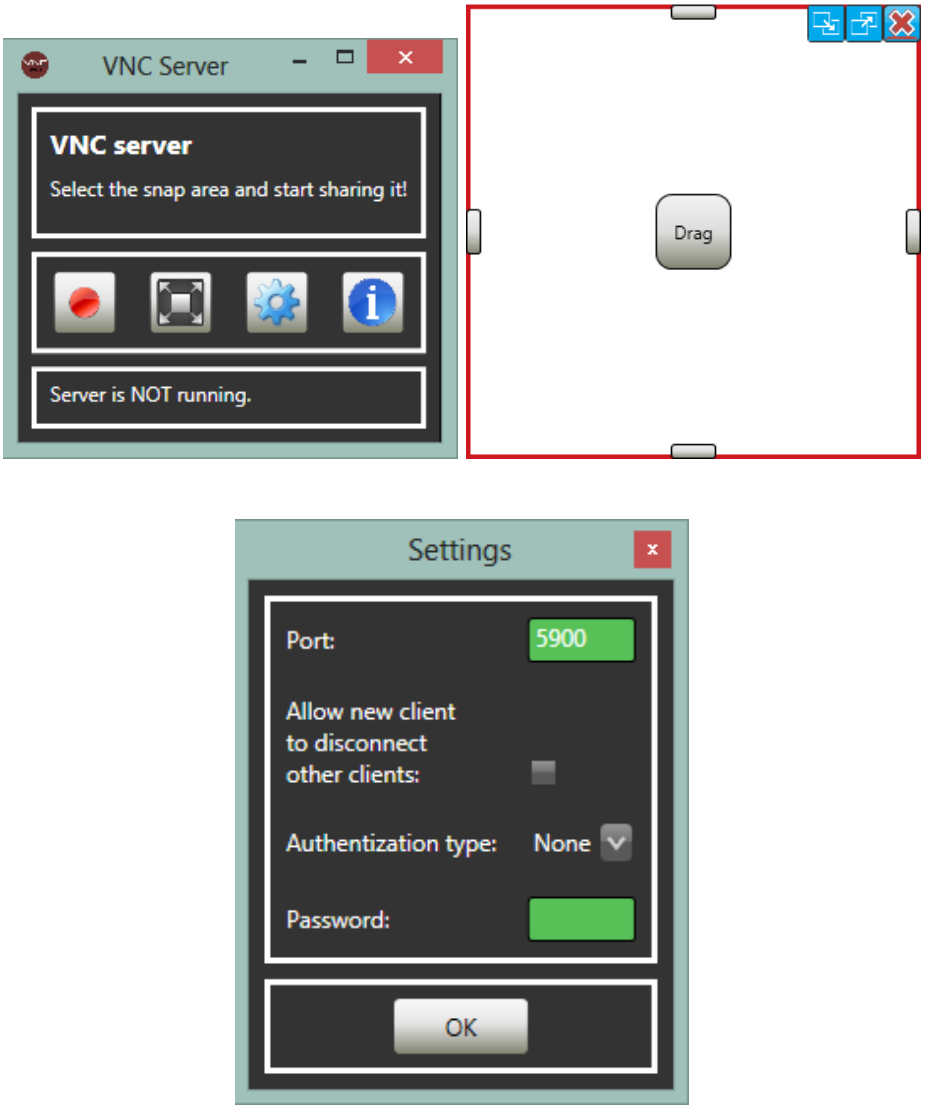


Obr. A.7: Balík MyScreen a MyImaging

Príloha B

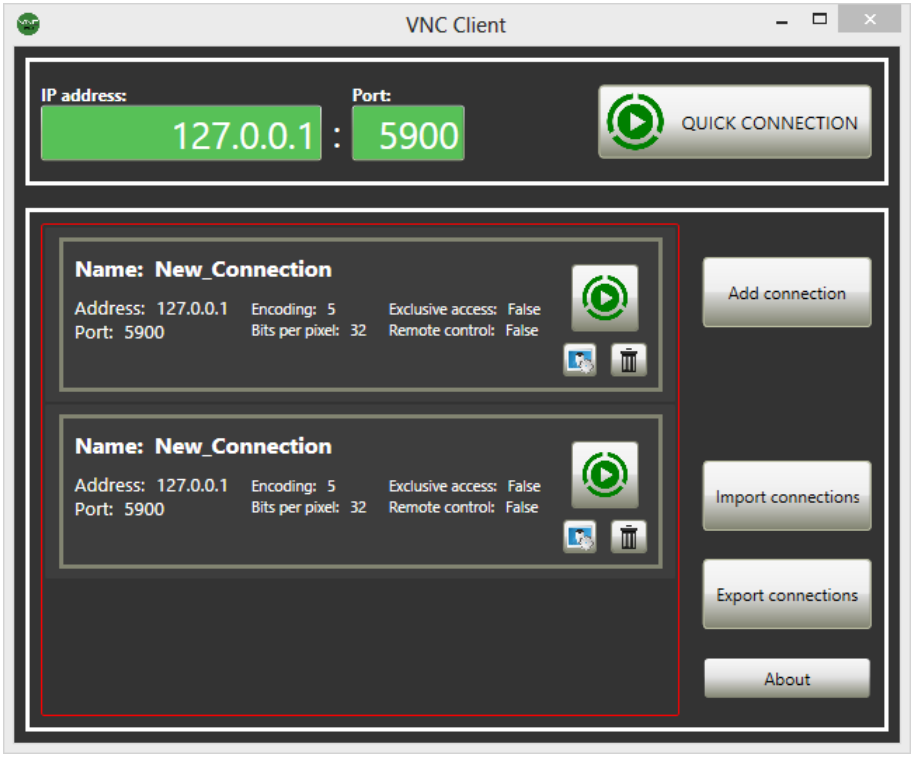
Výsledná aplikácia

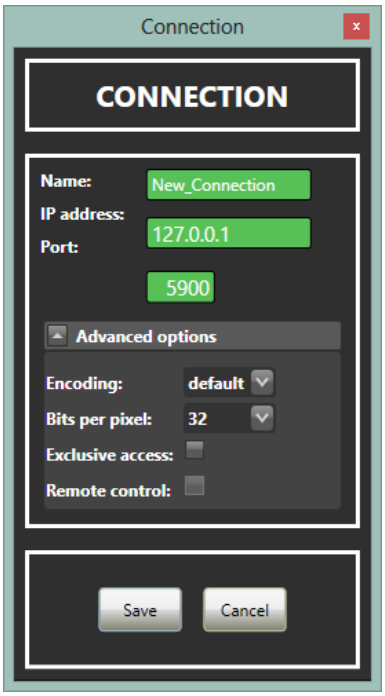
Server



Obr. B.1: GUI servera

Klient





Obr. B.2: GUI klienta

Příloha C

Obsah CD

- **bin** adresár so spustiteľnými súbormi aplikácie.
- **bin/saved** adresár s príkladom uloženia zoznamu spojení.
- **install** adresár s inštalačnými súbormi aplikácie.
- **src** adresár so zdrojovými súbormi aplikácie.
- **doc** adresár s dokumentáciou aplikácie.
- **thesis_src** adresár obsahujúci prácu v latexu.
- **thesis_pdf** adresár obsahujúci prácu vo formáte pdf.
- **README.txt** súbor s popisom CD, návod na zavedenie a spustenie aplikácie.

Příloha D

Používateľská príručka

Klient

Rýchle pripojenie

Ako vidno na obrázku B.2, používateľské prostredie klienta sa skladá z niekoľkých okien. Po spustení aplikácie je užívateľovi zobrazené hlavné okno. V hornej časti je umiestnený panel, ktorý umožňuje rýchle pripojenie k bežiacemu serveru. Do ľavého textového poľa je potrebné zadať IPv4 adresu servera, do druhého textového poľa port, na ktorom je aktivované načúvanie servera. Potom je potrebné stlačiť tlačidlo s názvom *Quick connection* a klient sa automaticky pripojí k serveru, ak je server aktivovaný.

Zoznam spojení

Ďalšou možnosťou, ako sa pripojiť k serveru, je z panelu so zoznamom spojení. Spojenie do zoznamu je možné pridať tlačidlom *Add connection*. Pri pridávaní nového spojenia, či úprave existujúceho spojenia v zozname (tlačidlo s vyskakujúcim popisom *Edit connection* pri prvku v zozname spojení) je možné meniť viaceré nastavenia, medzi ktorými napríklad aj kvalitu prenosu. Odstránenie položky zo zoznamu spojení je možné stlačením tlačidla s vyskakujúcim popiskom *Delete connection* pri prvku zo zoznamu. Pripojenie ku spojeniu sa vykoná stlačením tlačidla s vyskakujúcim popiskom *Connect to server* pri prvku v zozname spojení.

Prvky do zoznamu spojení je možné pridať aj načítaním zo súboru (tlačidlo *Import connections*) a uloženie zoznamu do súboru je vykonané stlačením tlačidla *Export connections*. Pre príklad je k aplikácií priložený súbor so spojeniami, ktorý je možné načítať (príloha C).

Ukončenie prenosu

Po úspešnom pripojení k serveru sa otvorí okno s obrazom prenášaným zo servera. Zatvorením tohto okna sa zruší spojenie so serverom a prenos je prerušený. Oknu sa môže počas spojenia so serverom zmeniť geometria, pretože klient podporuje zmenu veľkosti prenášaného obrazu.

Server

Nastavenia

Používateľské prostredie servera je možné nájsť na obrázku B.1. Po stlačení tlačidla s vyskakujúcim popiskom *Settings* sa zobrazí okno s možnými nastaveniami aplikácie. Je tu možné napríklad zmeniť metódu autentizácie, či heslo potrebné pre pripojenie k serveru. Po spustení servera už tieto nastavenia nie je možné meniť.

Spustenie načúvania

Spustenie servera, teda povolenie načúvania na vybranom porte je možné aktivovať tlačidlom s vyskakujúcim popiskom *Start sharing*. Po spustení načúvania je ho možné deaktivovať stlačením toho istého tlačidla (vyskakujúci popisok *Stop sharing*).

Výber zdieľanej plochy

Implicitne je nastavené zdieľanie celej dostupnej obrazovky, no to je možné zmeniť. Zmenu veľkosti zdieľanej plochy je možné vykonať aj počas aktívneho pripojenia s klientmi. Tá je vykonaná pomocou tlačidla s vyskakujúcim popisom *Select shared area*. Po stlačení sa otvorí obrysové okno, ktoré je aktuálne zdieľané.

Po otvorení okna je možné ľubovoľne meniť jeho veľkosť. Posun okna je zabezpečený stlačením a držaním tlačidla myši na mieste s popisom *Drag*. Okno je možné aj maximalizovať, minimalizovať či skovať. To všetko je dostupné aj počas aktívneho pripojenia ku klientom, pričom sa tieto zmeny odrazia aj u klienta (ak klient podporuje túto funkcionalitu).