Jihočeská univerzita v Českých Budějovicích

Přírodovědecká fakulta

Systém pro vzdálené ovládání virtuálních počítačů

Bakalářská práce

Autor: Tomáš Mika

Vedoucí práce: Ing. Jan Fesl

České Budějovice 2017

Jihočeské univerza v Českých Budějovicích
Přírodovědecká fakulta

ZADÁVACÍ PROTOKOL BAKALÁŘSKÉ PRÁCE

Student: Tomáš Mika
(jméno, příjmení, tituly)

Obor - zaměření studia: Aplikovaná Informatika

Katedra: Ústav aplikované informatiky

Školitel: Fesl Jan, Ing.
(jméno, příjmení, tituly, u externího š. název a adresa pracoviště, telefon, fax, email)

Garant z Přírodovědecké fakulty: .....
(jméno, příjmení, tituly, katedra - jen v případě externího školitele)

Školitel - specialista, konzultant: .....
(jméno, příjmení, tituly, u externího š. název a adresa pracoviště, telefon, fax, email)

Téma bakalářské práce:

Systém pro vzdálené ovládání virtuálních počítačů

Popis práce:

Bakalářská práce bude věnována systému pro vzdálenou administraci virtuálních počítačů. Systém bude realizován na bázi klient-server, bude umožňovat současné separátní ovládání více počítačů najednou, umožní současnou práci několika uživatelů najednou. Vlastní ovládání systému bude realizováno pomocí vhodné terminálové služby napojené na jádro systému (Windows popř. Linux). Pro vytvoření systému použijte programovací jazyk C++ nebo C# a implementujte v něm i intuitivní grafické rozhraní pro správu celého systému. Proveďte srovnání s existujícími řešeními a zdůrazněte hlavní výhody Vašeho řešení.

Cíl práce:

Prototyp funkčního systému pro správu a ovládání virtuálních počítačů.

Základní doporučená literatura:

[1] Jan Fesl, Přednášky z předmětu Distribuované a paralelní algoritmy, JCU PRF, 2015

[2] Miroslav Virius, Od C++ k C#, KOPP, 2002

Financování práce: .....

Vedoucí práce: Fesl Jan podpis: .....

U externích vedoucích fakultní garant práce: podpis: .....

Garant oboru bak. studia (nepožaduje se u zaměření, příprava na mag. stud. biologie):

..... podpis: .....

Vedoucí katedry: Libor Dostálek (v.z. Jan Fesl) podpis: .....

Případný souhlas vedoucí ústavu AV: ..... podpis: .....

V Českých Budějovicích dne: 5. 12. 2016

Převzal/a dne: ..... podpis: .....

BIBLIOGRAFICKÉ ÚDAJE

Mika Tomáš, 2017: Systém pro vzdálené ovládání virtuálních počítačů. [Remote control system for virtual computers. Bc. Thesis, in Czech.] – 44 p., Faculty of Science, The University of South Bohemia, České Budějovice, Czech Republic.

ANOTACE

Tématem bakalářské práce je vytvoření systému pro vzdálenou správu virtualizovaných operačních systémů, zejména operačního systému Windows. Práce popisuje nástroje a technologie nezbytné pro vytvoření takovéhoto systému, zároveň uvádí přímé srovnání s již existujícími a dostupnými řešeními. Práce se zabývá problematikou vzdálené správy virtualizovaných operačních systémů, zejména služby Vzdálené plochy Windows, která je integrována v operačních systémech Windows.

ABSTRACT

The thesis theme is creation of a system for remote management of virtualized operating systems, especially Windows. The thesis describes the technology tools necessary to create such a system. It also presents a direct comparison with existing solutions already available. Thesis deals with the remote management of virtualized operating systems, especially Windows Remote Desktop Services, which is integrated with the Windows operating systems.

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb., v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne 18.dubna 2017

Podpis: ……

PODĚKOVÁNÍ

Na tomto místě bych rád poděkoval panu Ing. Janu Feslovi za cenné připomínky a odborné rady, kterými přispěl k vypracování této bakalářské práce.

OBSAH

1. ÚVOD

Virtualizace operačních systémů je trendem poslední doby, jelikož poskytuje velké množství výhod oproti klasickému dříve využívanému systému jeden počítač, jeden operační systém. Hlavní výhodou tohoto řešení je absence potřeby plnohodnotného počítače, zde stačí slabší a levnější řešení v podobě terminálu pro koncového uživatele. Všechny výpočty a práce pak probíhají na serveru, kde je systém virtualizován. Dalšími výhodami jsou například umístění v sledované síti datacentra, nezávislost na lokálním fyzickém hardwaru a zálohované zdroje. Při využití tohoto řešení ve větším měřítku pak nastává problém se správou a podporou jednotlivých instalací operačního systému na virtualizovaných počítačích, kdy se informace o daném virtuálním stroji, tj. informace nezbytné k připojení k danému stroji a způsob připojení samotný nachází v různých částech systému. Pro snazší správu a podporu takového systému je pak nezbytné udržovat, mnohdy manuálně, databázi informací o všech provozovaných virtuálních strojích, jejich uživatelích, spolu s informacemi nezbytnými pro připojení k jejich virtuálnímu stroji.

Systém vzdálené administrace operačních systémů s využitím „Vzdálené plochy“ Windows je většinou omezen pouze na lokální síť daného operačního systému. Tato omezení jsou řešena programem, který bezpečně umožní identifikaci virtuálních počítačů a jejich vzdálenou správu nejen z lokální sítě, což usnadní práci správci systému virtuálních počítačů danou instalaci systému vzdáleně spravovat, a to i za předpokladu, že správce bude připojen v jiné síti než spravovaný počítač. Pro bezproblémové využití systému byla klientská část naprogramována jako interní služba Windows, spouštěná automaticky po startu operačního systému. Systém byl vytvořen na bázi klient – server – klient (aplikace).

Při tvorbě bakalářské práce byla použita metoda deskripce systémů pro vzdálenou administraci operačního systému, a vzhledem k potřebě správy většího množství virtuálních strojů, i metoda komparace výkonu a vhodnosti použité aplikace pro vzdálenou správu a porovnání s již existujícími řešeními.

V rámci této bakalářské práce vznikl funkční systém pro automatizovanou identifikaci jednotlivých virtuálních operačních systémů, jejich uživatelů a informací potřebných pro připojení k nim.

2. CÍL BAKALÁŘSKÉ PRÁCE

Cílem bakalářské práce bylo vytvoření funkčního prototypu řešení pro správu a ovládání většího množství virtuálních počítačů s operačním systémem Windows, popř. Linux. Systém byl realizován stylem klient – server a umožňuje současné separátní ovládání více počítačů najednou. Vlastní ovládání systému bylo realizováno pomocí vhodné terminálové služby napojené na jádro operačního systému.

V praktické části bakalářské práce bylo využito programovacího jazyku C#. Řešení obsahuje tři části, programy, klientskou službu systému Windows (spustitelnou i pod OS Linux), terminálovou aplikaci pro server (spustitelnou i pod OS Linux) a program s grafickým rozhraním pro správu celého systému. Výsledné řešení bylo porovnáno s již existujícími a dostupnými řešeními.

3. VIRTUALIZACE

Pod pojmem virtualizace se v IT prostředí označují postupy a techniky, které umožňují k dostupným zdrojům přistupovat jiným způsobem, než jakým fyzicky existují či jsou propojeny. Existuje několik metod virtualizace a několik různých úrovní, na nichž lze hardware virtualizovat. Ačkoliv se může zdát, že je pojem skloňován teprve pár let, reálné základy byly položeny již poměrně dávno. V současnosti, kdy virtualizační technologie značně pokročily, nemusí vždy jít o virtualizaci celých počítačů, ale virtualizovat lze i jednotlivé aplikace. [1]

3.1 Historie virtualizace

Pojem virtualizace směřuje do 60. let 20. století, kdy došlo k vytvoření celých virtuálních strojů postavených na kombinaci hardwarových a softwarových technik. Poprvé s touto koncepcí přišla společnost IBM. Pojem virtuální stroj pochází od pokusného stránkovacího mechanismu systému IBM M44/44X. Zakládání a správa virtuálních strojů byla v počátcích CP-40 také označována jako zakládání a správa pseudostrojů a později jako virtualizace serverů. [2] Výhoda tohoto řešení spočívala, stejně jako dnes, ve vytvoření několika zdánlivých počítačů v rámci jednoho fyzického.

Virtualizace ve svých počátcích znamenala především hledání možností, jak ze systémů, které byly schopné zpracovávat jen jeden úkol v jeden okamžik, vytvořit více vláknová zařízení, která by jednotlivé operace prokládala, a tím lépe využívala možností tehdejšího hardwaru. Jedním z prvních strojů, které využívaly virtualizaci, byl počítač IBM 704 s technologií Compatible Time Sharing System. Další vývoj v této oblasti nastavil víceuživatelská prostředí jako jakýsi standard a virtualizace se v těchto letech začala podobat tomu, jak je známa dnes. [1]

Metoda kombinuje myšlenku sdílení systémových prostředků a izolovanosti jednotlivých virtuálních prostředí. Výhodou virtuálního stroje je možnost sestavení celého operačního systému na míru uživateli, který používá počítač, bez toho, aby došlo k jejich vzájemnému ovlivňování, při nízké systémové režii. Jednotlivý operační systém je pak plně funkční, bez dalších omezení. Vše bylo vyvinuto na základě požadavků správců a uživatelů na víceúlohové a víceuživatelské operační systémy, které vedou k vývoji a zdokonalování metod souběžné obsluhy procesů a uživatelů, kteří využívají virtualizaci. Jedná se tedy o mechanismus, pomocí něhož dosáhneme určitého stupně abstraktního stroje s abstraktním

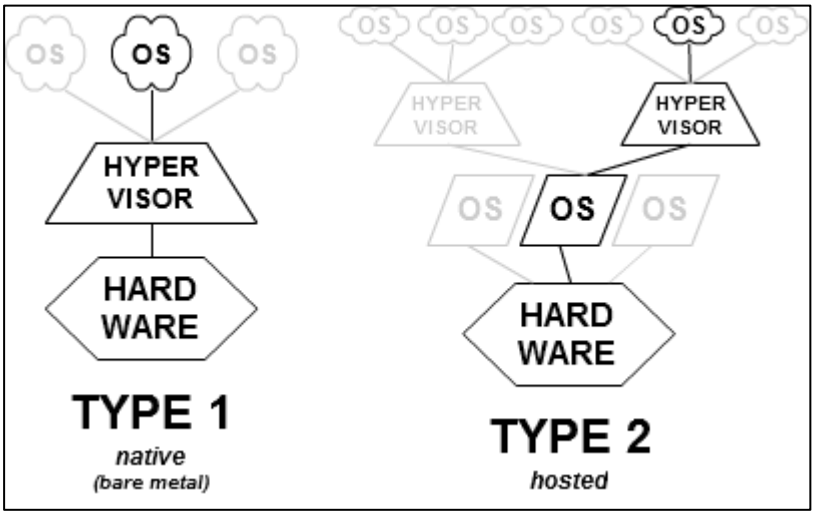
hardwarem. Pro hardwarový zdroj, který je v počítači obsažen pouze jako jedna fyzická instance, je možné vytvořit více virtuálních instancí, stejně jako zprostředkovat hardwarový zdroj, který se vyskytuje ve více fyzických instancích jako jedno virtuální zařízení. [1]

3.2 Klasifikace virtualizace

Virtualizaci lze klasifikovat podle několika kritérií, a to dle vazby na fyzický hardware, režimu a způsobu nasazení. Pod základní klasifikací lze uvést členění dle typu hypervisoru a dle vazby na fyzický hardware.

3.2.1 Virtualizace podle typu hypervizoru

Pod pojmem hypervisor se obecně označuje vrstva, software, firmware, nebo hardware, který umožňuje běh virtuálních počítačů. Počítač, na kterém běží hypervisor a nad ním jeden nebo více virtuálních počítačů nazýváme hostitelský počítač/server. Těmto virtuálním počítačům pak hypervizor poskytuje buď přímý přístup k hardwaru nebo jej emuluje. To je v kontrastu s virtualizací na úrovni operačního systému, kdy je hypervizor podřízen správě systémových prostředků hostitelského operačního systému. Běžně se v literatuře hostující zařízení označuje jako „Host“ a virtualizované zařízení jako „Guest“. [3]



Obrázek 1: Klasifikace podle hypervizoru. [3]

Typ 1 – Bare-metal

Tento typ se také jinak nazývá „native“ nebo „bare-metal“. V tomto případě běží hypervisor přímo na hardwaru hostitele, přímo řídí přidělování prostředků hostujícím

virtuálním počítačům/operačním systémům a spravuje je. Hostované operační systémy běží přímo nad hypervizorem a mají přímý přístup k hardwaru. Systémy jsou naprosto oddělené, nezávislé a nemohou se navzájem ovlivňovat. [3] Na principu Typu 1 funguje například virtualizační software VMware ESX, nebo MS Hyper-V.

Typ 2 – Host-Based

Tento typ se také jinak nazývá „Host-Based“. U tohoto typu hypervizor běží na hostitelském operačním systému, stejně jako ostatní programy a jsou na něm závislé. Takto hostovaný virtuální operační systém není možno provozovat bez plně spuštěného a načteného hostitelského operačního systému. Hostovaný operační systém nemá přímý přístup k hardwaru, jelikož veškeré prostředky spravuje hostitelský operační systém, který komunikaci zprostředkovává. Toto uspořádání způsobuje snížení výkonu hostovaného operačního systému, oproti Typu 1, který má přímý přístup k hardwaru. [3] Na principu Typu 2 funguje například virtualizační software VMware Workstation.

3.2.2 Virtualizace podle vazby na hardware

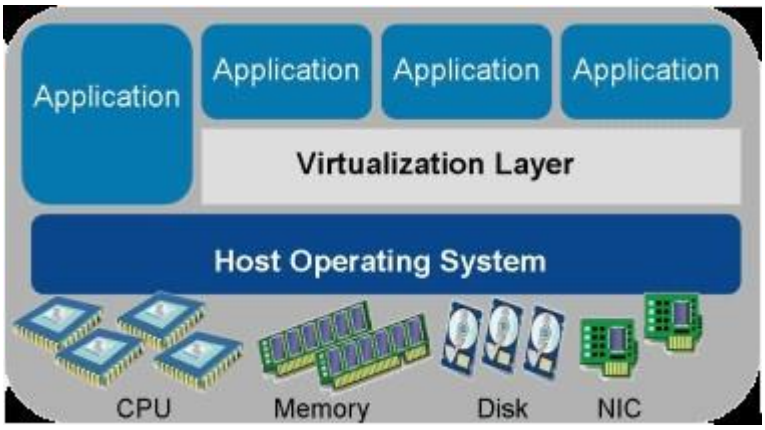
Emulace

Virtualizace hardwarových komponent za účelem simulace jiné hardwarové platformy se nazývá „emulace“. Hostované operační systémy a aplikace není nutné modifikovat. Je potřebné zachovat jejich plnou kompatibilitu. Vzhledem k tomu, že emulace virtualizuje odlišnou hardwarovou platformu, nevyužívá hardwarovou podporu virtualizace a je pouze softwarová. Dochází proto ke značnému snížení výkonu emulovaného operačního systému. Emulace umožňuje programům běžet na jiné platformě, než pro jakou jsou naprogramovány. Příkladem takového emulačního softwaru je DOSBox. [4]

Nativní (plná) virtualizace

Tento druh virtualizace simuluje dostatečné množství hardwarových komponent pro nemodifikovaný běh hostovaného operačního systému izolovaně od hostitelského operačního systému. Pro nativní virtualizaci je vyžadována podpora procesoru. V současnosti jsou na trhu nejrozšířenější technologie Intel-VT a AMD-V. Takto provozované hostované operační systémy nemají přímý přístup k hardwaru. Přístup je zprostředkováván hostitelským operačním systémem, hypervizorem, ale hostovaný operační systém funguje jako by měl

přímý přístup k hardwaru. Na tomto principu pracuje například VMware Server, VMware Workstation nebo KVM.



Obrázek 2: Nativní virtualizace. [4]

Paravirtualizace

U tohoto druhu není vytvářen kompletní virtuální hardware, ale místo toho je poskytováno aplikační rozhraní (API), které vyžaduje modifikace na straně hostovaného operačního systému tak, aby mohlo dojít ke spuštění operačního systému nad virtuálním strojem. Virtuální prostředí je podobné fyzickému, ale není shodné. Výhodou tohoto řešení je vysoký výkon virtuálního systému, kterým se blíží nevirtualizovaným systémům. Virtualizovanému operačnímu systému může být známo, že je provozován ve virtuálním prostředí. Příkladem softwaru využívajícího paravirtualizace jsou například Xen, nebo VMware ESX Server.

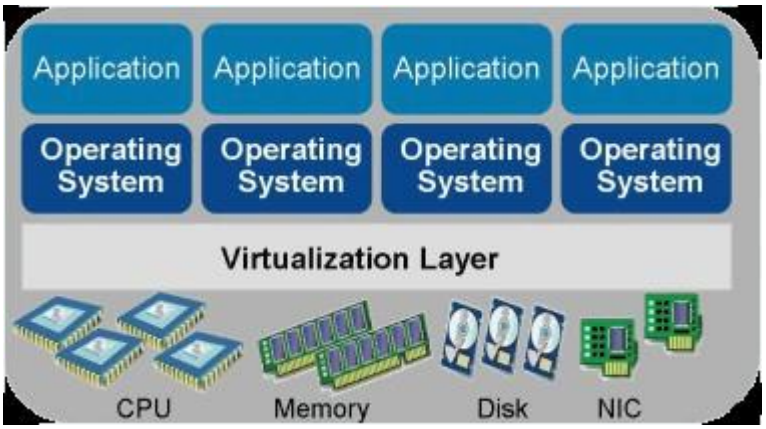
Aplikační virtualizace

Jedná se o metodu umožňující na platformě nezávislé spouštění aplikací, nebo aplikací vyvinutých pro odlišnou platformu. Nedochází zde k plné virtualizaci běhového prostředí, ale pouze k virtualizaci nezbytných komponent pro spuštění takového softwaru. Jedná se o software, který umožnuje spuštění aplikace vyvinuté pro jiný operační systém nebo platformu, jako je například Wine, který umožňuje na operačním systému Linux spouštět aplikace napsané pro operační systém Windows.

OS-level virtualizace – virtualizace závislá na operačním systému

Tato virtualizace využívá hostitelský operační systém a umožňuje tak virtuálním strojům sdílet fyzické prostředky hostitele. Virtualizovaná prostředí jsou spuštěna nad společným jádrem, které má fyzický přístup k hardwaru. Nad jádrem hostitelského operačního systému

jsou vytvořeny izolované kontejnery, které replikují server fyzický. Toto řešení nevyžaduje modifikaci operačního systému, ale pouze instalaci speciálního softwaru pro virtualizaci. Příkladem je Linux-VServer, VMware ESX Server, Solaris Containers, nebo Free VPS. [4]



Obrázek 3: Nezávislá virtualizace. [4]

3.3 Produkty pro virtualizaci operačních systémů

Virtuální počítač je sbírkou virtuálních hardwarových komponent, podobně jako fyzický počítač, obsahuje minimálně jeden virtuální procesor, operační paměť, grafickou kartu, zařízení SCSi, IDE, disketové jednotky, paralelní a sériové porty a síťové karty. Virtuální počítač potřebuje ke svému fungování operační systém úplně stejně jako fyzický počítač. [5]

Mezi nástroje, které nám umožní zprovoznění celého virtuálního stroje s dalšími operačními systémy v rámci hostitelského operačního systému patří VirtualBox, Hyper-V a VMware Workstation.

3.3.1 Oracle VM VirtualBox

Virtualbox je multiplatformní software dostupný zdarma pro nekomerční použití, pod licencí PUEL/GPL. Zastává roli hypervizora na hostitelském systému. Dále je nabízena komerční verze produktu, která má širší možnosti a větší spektrum uplatnění. Běžně je dostupná i Open Source edice, která je nabízena v podobě volně dostupných zdrojových kódů.

I v odlehčené verzi pro nekomerční použití nabízí Virtuabox vše nezbytné pro běžné provozování virtuálního operačního systému, počínaje podrobným nastavením virtuálního

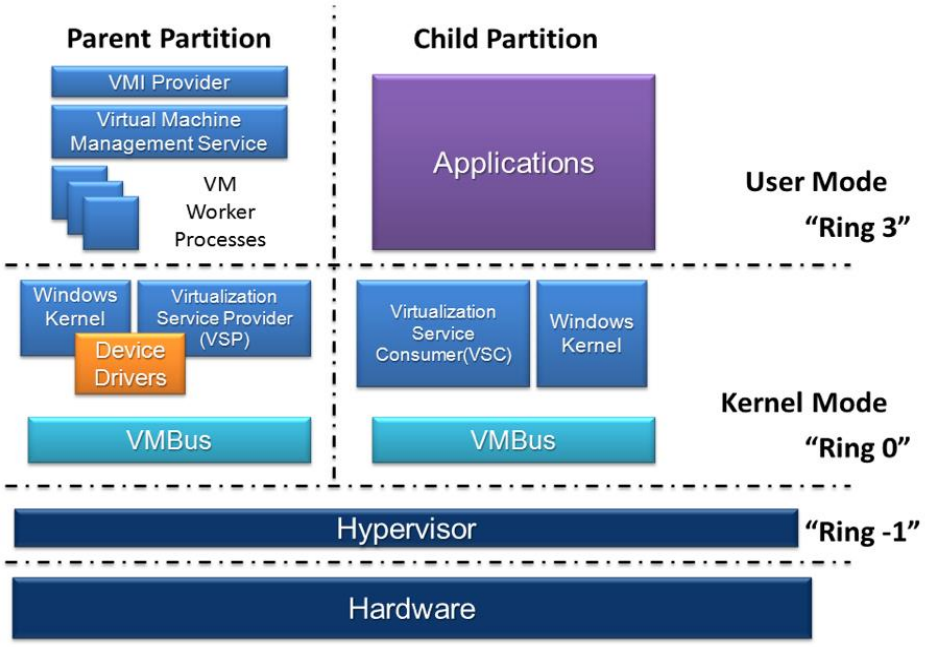
hardwaru, až po Snapshoty (obrazy virtuálního systému - záloha) v čase a částečnou integraci s hostitelským operačním systémem (sdílená schránka, Drag&Drop, atd.). Software také podporuje možnost hardwarové virtualizace od společností Intel a AMD, ale nevyžaduje ji.

3.3.2 Hyper-V

Jedná se o nativní virtualizační software na principu hypervisoru, který vyžaduje virtualizaci podporující hardware, procesor podporující technologii Intel-VT, nebo AMD-V. Bez této hardwarové podpory neumožní systém spuštění Hyper-V.

Role Hyper-V je integrována do operačního systému Windows v různých podobách napříč verzemi tohoto operačního systému. Profesionální řešení nabízí edice Windows Server, ať už v podobě komplexního serverového operačního systému nebo čistě virtualizačního serveru Windows Hyper-V Server. Běžné klientské verze operačního systému Windows obsahují verzi tohoto virtualizačního softwaru a umožňují takřka profesionální použití.

Pro klientský operační systém nabízí Hyper-V vše nezbytné od Snapshotů, stavu systému v čase, až po integraci, propojení virtuálního operačního systému s hostitelským operačním systémem. Zajímavější funkce nabízí až profesionální serverová edice, kde Hyper-V umožňuje konsolidovat servery a vytvořit tak virtuální server přes větší množství fyzických serverů a na tom dále provozovat virtuální operační systémy. Hyper-V je profesionální komerční řešení pro firmy a serverové farmy, které je v slabší a méně vybavené verzi dostupné i v klientských Windows, a to zdarma.



Obrázek 4: Hyper-V schéma. [9]

3.3.3 VMware

Společnost VMware, Inc. zabývající se vývojem virtualizačního softwaru a virtualizačních platforem napříč spektrem druhů a typů virtualizace, od serverových profesionálních řešení v podobě VMware vSphare až po klientský VMware player, tedy od virtualizace celých serverových farem až po virtualizaci jednotlivých klientských operačních systémů.

Prvním a nejznámější produktem této společnosti je VMware Workstation, který slouží k vytváření virtuálních operačních systémů na hostitelském systému, ať už s hardwarovou podporou nebo bez ní. Jedná se o komplexní profesionální řešení, ale uplatnění nalezne i pro běžné uživatele, pokud jsou ochotni nést náklady na pořízení tohoto produktu. Jediný VMware Player, jakožto přehrávač/spouštěč před vytvořených virtuálních strojů je nabízen zdarma pod Open Source licencí. Lze jej označit za multiplatformní software, jelikož je nabízen ve 2 verzích, a to pro operační systém Windows a Linux, jak v 32 bitové, tak i 64 bitové verzi.

4. VZDÁLENÁ SPRÁVA OPERAČNÍCH SYSTÉMŮ

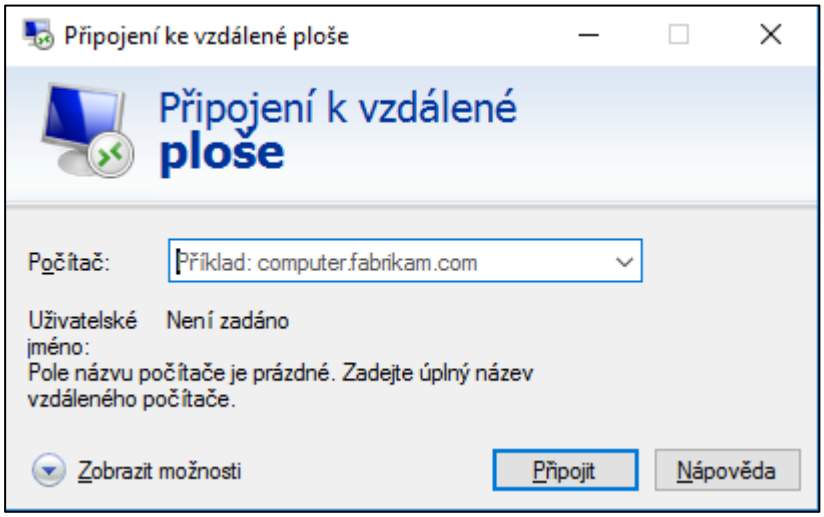
Operační systémy lze na dálku spravovat mnoha způsoby počínaje terminálovými službami (telnet, SSH, powershell), až po služby typu vzdálené plochy (RDP, VNC, Teamviewer). Každý síťový operační systém lze dnes nakonfigurovat pro možnost vzdáleného připojení nebo za tímto účelem nainstalovat software.

Hlavní problematika administrace většího množství virtuálních operačních systémů není v dnešní době problematická adresace (nedostatek zapamatovatelných IPv4 adres), ale jejich množství. Různé virtualizační služby nabízejí mnoho pomůcek a souhrnných výpisů všech hostovaných virtuálních strojů. Ve většině případů tyto seznamy obsahují pouze data nezbytná pro obsluhu/správu celého systému jako celku, ale nikoli pro snadný a rychlý support virtualizovaných operačních systémů.

Nástrojů, technologií a programů umožňující správu operačního systému je velké množství, v dalších odstavcích byly vybrány nejznámější, nepoužívanější a dostupné zdarma.

4.1 Remote Desktop a Remote Desktop Protocol

Výraz vzdálená plocha se vztahuje k softwaru a protokolu připojení, který umožňuje vzdálenému systému zobrazit grafické uživatelské rozhraní systému, ke kterému je tento vzdálený systém připojen, v klientském okně. [6]



Obrázek 5: Okno klienta vzdálené plochy.

Vzdálená plocha pracuje na protokolu zvaném RDP (Remote Desctop Protocol). Používá se výhradně pro vzdálené připojení ke vzdálenému operačnímu systému. Dnes je tento protokol i služba vzdálené plochy podporovány napříč operačními systémy, od domovského Windows až po různé distribuce Linuxu či mobilních operačních systémů jako je operační systém Android. Spojení mezi RDP serverem a klientem je zabezpečeno pomocí systému certifikátů.

RDP je proprietární protokol společnosti Microsoft, pracující na registrovaném portu 3389, používaný softwarem pro připojení ke vzdálené ploše počítačů. Je integrován do všech operačních systému Microsoft počínaje Windows 95. Od verze Windows NT v podobě, kterou známe dnes, tedy že systém obsahuje v základu jak klientskou část, tak i server.

Vzdálená plocha pracuje se vzdálenou obrazovkou jako s objektem, a tak je i přenáší. Tento princip je výhodný z hlediska přenesených dat. Díky integraci do operačního systému umožňuje lepší sdílení prostředků jako jsou například tiskárny, uložiště nebo přehrávání zvuku. Až na výjimky jsou tyto funkcionality u ostatních řešení pro vzdálená připojení omezená nebo naprosto nedostupná. Oproti tomu ve starších verzích Windows vytvářela vzdálená plocha novou relaci a chovala se v podstatě jako grafický terminál, od Windows Vista byl implementován Windows Desktop Sharing, který funguje na podobném principu jako konkurenční VNC, tedy přímé zobrazení pracovní plochy.

Jelikož je vzdálená plocha integrována do Windows, její použití nevyžaduje instalaci dalšího softwaru ani složitá nastavování a je tak snadné i pro laiky.

Existují také alternativní klienti pro připojení k vzdálené ploše vzdáleného systému jako například Rdesktop.

Dále existuje několik méně známých alternativ, počínaje proprietárním ICA (Independant Computing Architecture), což je protokol společnosti Citrix, která se zabývá zejména virtualizací. Tento protokol je využívaný pouze produkty této společnosti, například aplikací pro správu virtuálních systémů XenApp. Další alternativou, pouze pro svět operačních systémů založených na Linuxovém jádře, je X Window Systém v X11 nebo jeho alternativa NX. Ani jeden z těchto protokolů není použitelný v kombinaci s operačním systémem Windows. Nejznámější alternativou k RDP je VNC.

4.2 VNC

Virtual Network Computing je grafický software používaný pro vzdálenou správu operačních systémů, nejčastěji prostřednictvím počítačové sítě. Pracuje na protokolu zvaném RFB (Remote Buffered Protocol), který zprvu umožnoval pouze vzdálený přístup ke grafickému rozhraní vzdáleného systému. Dnes již umožňuje pokročilejší funkce jako je sdílená schránka, přenos souborů, či komprese a šifrování datového toku.

VNC bylo vyvinuto firmou Olivetti Research Laboratory (ORL) v Anglickém Cambridge a jeho zdrojové kódy byly publikovány jako Open Source pod licencí GPL. V roce 2002 došlo k prodeji firmy ORL a následnému ukončení vývoje původního VNC. Důsledkem této události vzniklo několik produktů založených na veřejně dostupných zdrojových kódech původního VNC, jako jsou například programy RealVNC, nebo TinyVNC.

Programy založené na kódech VNC jsou na platformě nezávislé a je možné je provozovat napříč operačními systémy. Jedná se o klient-server systém, tedy pro úspěšné připojení k cílovému systému je zapotřebí mít na něm nainstalovaný VNC server. Ve výchozím nastavení používá VNC registrovaný port 5900.

VNC funguje na principu bitmapy, ne však v podobě nekomprimovaného BMP souboru. Každý bod na zobrazovaném obraze má své souřadnice. Zobrazovaná plocha se přenáší jako jeden velký obrázek, na který se dále vykreslují jednotlivá okna, ikony a další prvky plochy. Mezi spojenými systémy se pak přenáší pouze změny v obraze, což výrazně zvyšuje rychlost a snižuje nároky na dostupné připojení. Tento princip je výhodný pro přenos složitější grafiky, kdy se nevyplatí obrázek přenášet jiným způsobem než jako obrázek.

RealVNC

Program vyvíjený původním týmem, který pracoval na projektu VNC. Tento program je dostupný v několika edicích, počínaje verzí zdarma, pro osobní použití až po placenou platformu pro kompletní správu operačních systémů. Kompletní balík obsahuje několik aplikací, počínaje grafickou aplikací pro koncové stanice, přes serverovou aplikaci až po mobilní klienty. RealVNC Enterprise tak nabízí plnohodnotné komerční řešení pro správu vzdálených operačních systémů se všemi k tomu nezbytnými technologiemi, jako je zabezpečení spojení, statistiky využívání spojení a vzdáleného systému. RealVNC je dnes považováno za Industry Standard.

TinyVNC

Odlehčená varianta VNC, obsahující pouze klientskou aplikaci pro připojení k VNC serveru. Tuto verzi lze nazvat absolutně multiplatformní, jelikož existují verze pro všechny známější dnes využívané operační systémy, včetně těch mobilních (například již nepodporovaný Windows Phone 7).

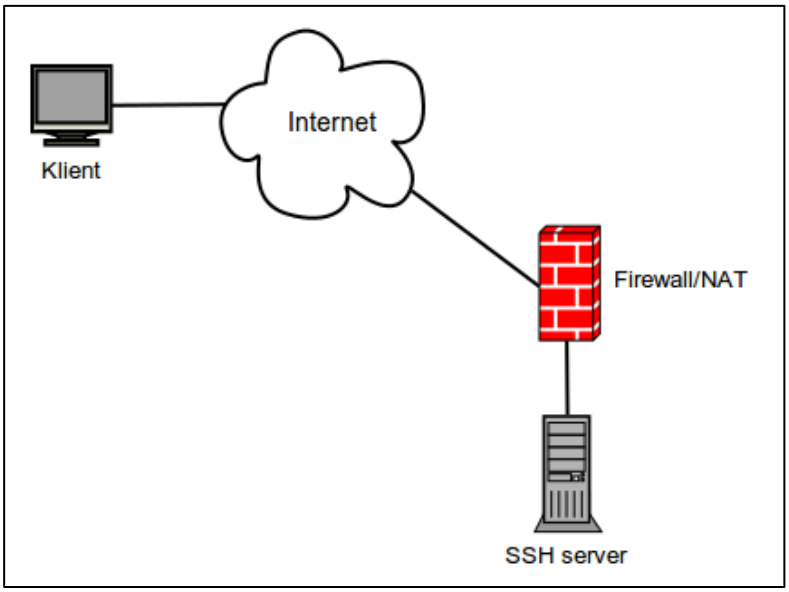
Tight VNC

Projekt zaměřený na minimalizování objemu přenesených dat v rámci využívání vzdáleného připojení. Toto řešení není tak propracované jako RealVNC, ale nabízí zajímavou volně dostupnou alternativu. Také je nabízeno komerční řešení, a to za spíše symbolické ceny, ale ty nenabízí vyšší komfort, či větší množství funkcí. Komerční licence umožňuje přístup ke zdrojovým kódům a jejich upravování s podporou vývojového týmu TightVNC.

4.3 SSH

SSH neboli Secure Shell je zabezpečený protokol pro terminálovou emulaci, který je založen na principu šifrování veřejným klíčem. Uživatel si musí vygenerovat svůj privátní (private key) a veřejný klíč (public key). Standardně užívá port 22. [7]

Vznikl jako bezpečná alternativa k naprosto nezabezpečenému protokolu telnet, který posílá veškerou komunikaci jako prostý text. V základní podobě je SSH terminálová služba a vzdálená správa tak probíhá pouze v textové podobě konzole/příkazové řádky. Tento protokol ale umožňuje vytvoření SSH tunelu, skrze který je možné ovládat



Obrázek 6: SSH spojení diagram. [13]

i grafický režim. Nejčastěji se tento způsob připojení ke vzdálenému systému používá v systémech založený na operačním systému Unix a Linux. Ve většině linuxových distribucí je přítomen SSH klient v základní instalaci.

SSH využívá mnoho protokolů pro zabezpečení spojení, například SFPT nebo SCP. Oba zmíněné protokoly se používají pro přenos souborů a díky SSH bezpečně.

Na Windows lze SSH provozovat pouze prostřednictvím softwaru třetích stran. Windows tento protokol nemá naimplementovaný. Existuje pouze varianta SSH, kterou lze označit za integrovanou ve Windows, a to je povolení a doinstalování Ubuntu System SubShell jako rozšíření/modul pro nejnovější verzi operačního systému Windows 10. Tato varianta přidá do Windows linuxový terminál, konkrétně z distribuce Ubuntu. Instalace a použití Ubuntu terminálu ve Windows je podmíněna přepnutím systém do vývojářského režimu. Jednoduší variantou je proto využití softwaru třetích stran, jako jsou například FreeSSHd, Putty, nebo WinSCP.

4.4 TeamViewer

Jedná se o proprietární software dostupný zdarma pro nekomerční použití. Aplikace pro vzdálený přístup a podporu přes počítačovou síť. Umožňuje vzdáleně ovládat počítač, přenos souborů i zvuku. TeamViewer je obdoba Windows Desktop Sharing (sdílení plochy systému Windows, nebo také vzdálená pomoc) a má i podobnou funkcionalitu, avšak nevyužívá v systému integrované služby.

Po úspěšné připojení je vyžadována instalace klientského programu TeamViewer na obou stranách spojení. Komunikaci poté zprostředkovávají servery společnosti TeamViewer GmbH, která tento produkt vyvinula. Po spuštění vygeneruje TeamViewer uživatelům na obou stranách unikátní ID a heslo pro připojení, které může uživatel změnit. Strana spojení, která se chce připojit k druhé straně, poté zadá ID cílového PC a jeho heslo. Následně dochází k vytvoření spojení, které je šifrováno za pomocí RSA a samotná relace mezi klienty pak ještě pomocí algoritmu AES.

5. POROVNÁNÍ SOFTWARU PRO HROMADNOU VZDÁLENOU SPRÁVU

Problematiku hromadné správy velkého množství instalací operačních systémů, ať už virtualizovaných či nevirtualizovaných, je tématem následujících odstavců. Existuje velké množství programů, které mají pomoci administrátorů ve správě provozovaných operačních systémů, zde vybrané jsou nejzajímavějšími řešeními.

Většina z níže zmíněných programů podporuje připojení mnoha protokoly, nikoliv pouze RDP, ale i například VNC, Citrix ICA, SSH atd.

5.1 Remote Desktor Manager

Moderní aplikace od společnosti Devolutions je prvním zástupcem v seznamu. Toto řešení je možné označit jako jedno z pokročilejších řešení, které nabízí široké možnosti propojení se systémem, na kterém je nainstalován. Tuto aplikaci lze označit jako kompletní platformu pro správu a podporu velkého množství instalací operačních systémů různých typů a velkého množství uživatelských profilů. Podporuje synchronizaci s ActiveDirectory a skenování sítě za účelem hledání dostupných klientů. Klienty lze přidávat i manuálně nebo lze importovat předpřipravené soubory vzdáleného připojení .rdp, nebo .vnc.

Software je dostupný ve dvou variantách. První z nich je volně dostupná, avšak silně omezená verze pro osobní použití, druhá varianta je podnikový balíček pro profesionální použití.

Výhody:	**Nevýhody:**
integrace s Windows zejména pak Server,	cena,
množství podporovaných protokolů,	složitost uživatelského rozhraní.
možnost instalace rozšíření,	
velké množství doplňkových funkcí.	

Tabulka 1: Výhody a nevýhody Remote Desktop Manager

5.2 Terminals

Bezpečný terminálový program pro správu velkého množství operačních systému s podporou vzdáleného připojení k nim. Podporuje širokou škálu protokolů a tím pokrývá většinu dnes provozovaných operačních systémů. Umožňuje současné spuštění více vzdálených připojení. Tento software je volně dostupný pod Open Source licencí.

Uživatelské rozhraní je jednoduché, okno rozdělené na dvě části, pracuje se záložkami, i spouštěná vzdálená připojení se otvírají jako záložka v pravé části okna, nezvolíte-li jinak. Aplikace je intuitivní, vyžaduje manuální konfiguraci a manuální zadávání jednotlivých vzdálených spojení i všech detailních konfigurací.

Výhody:	**Nevýhody:**
jednoduché rozhraní,	pouze manuální vkládání záznamů,
Open Source projekt = zdarma,	spartánský design,
možnosti práce se záznamy,	využívá ActiveX – pouze pro Windows.
možnosti síťové diagnostiky.	

Tabulka 2: Výhody a nevýhody Terminals.

5.3 Remote Desktop Connection Manager – RDCMan

Jednoduchý nástroj pro hromadnou správu vzdálených připojení. Umožňuje v jednom okně snadno uspořádat všechna potřebná vzdálená připojení. Volně dostupný nástroj vyvinutý společností Microsoft, za účelem usnadnění práce administrátorům. Zprostředkovává spojení vzdálené plochy, podporuje zabezpečení pomocí certifikátů.

Rozhraní programu je strohé, velice jednoduché. Okno je rozděleno do dvou částí, kdy v levém panelu je stromová hierarchie uložených spojení a v pravé části buďto vypsaný seznam uložených spojení s náhledem, podle zvolené části v hierarchii vlevo nebo zobrazení

plochy otevřeného vzdáleného spojení. RDCMan je omezen na použití s protokolem RDP a programem vzdálená plocha, který je integrován ve Windows.

Výhody:	**Nevýhody:**
jednoduché rozhraní,	pouze manuální zadávání záznamů,
snadné ovládání.	minimální rozšířená funkcionalita.

Tabulka 3: Výhody a nevýhody RDCMan.

5.4 MultiDesk

Velice jednoduchý do panelů rozdělený správce vzdálených připojení, který je volně dostupný. Toto řešení se skládá ze dvou částí, grafického programu, který je v roli serveru a klientských aplikací, které umožňují na síti a RDP nezávislou identifikaci klienta k vytvoření šifrovaného kanálu pro přenos režie a pak samotného tunelování RDP.

Výhody:	**Nevýhody:**
extrémně jednoduché řešení,	sdílený klíč pro šifrování přenosu.
snadné na použití.	

Tabulka 4: Výhody a nevýhody MultiDesk

5.5 Souhrnné zhodnocení

Vybrané programy pro hromadnou správu operačních systémů jsou vzájemně jen velmi těžce porovnatelné, jelikož každý vznikl za jiným účelem. První uvedený Remote Desktop Manager je příklad profesionálního řešení pro nasazení ve firmě, kde se řeší tato problematika. Druhý zmíněný program Terminals je komunitní řešení, které za určitých okolností může

soupeřit i s profesionálními nástroji, zejména svým jednoduchým rozhraním, i když designově tato aplikace odpovídá Windows 95. Poslední dva zmíněné programy, RDCMan a MultiDesk lze porovnat poměrně snadno, jelikož oba jsou velice jednoduché, rychlé a plní pouze účel, za kterým byly vytvořeny, bez doplňujících funkcí, jejich přítomnost může být mnohdy kontraproduktivní.

Následující tabulka zobrazuje bodové hodnocení jednotlivých parametrů pro každý produkt, kdy 0 znamená minimální hodnotu a 5 maximální hodnotu.

	RD Manager	**Terminals**	**RDCMan**	**MultiDesk**	**Vlastní řešení**
Ovládání	**5**	**5**	3	3	**5**
Přehlednost	3	4	4	3	**5**
Podpora	**5**	2	3	1	1
Požadavky	3	4	**5**	**5**	**5**
Rozšiřitelnost	**5**	2	0	0	0
Cena	2	**5**	**5**	**5**	**5**
Souhrn	**23**	**22**	**20**	**17**	**21**

Tabulka 5: Programy pro hromadnou správu-shrnutí.

Nejlépe vyšlo komerční řešení, a to i přesto, že působilo nepřehledným dojmem, ale propracovaný systém dokumentace pomáhal se v programu zorientovat. Bodový rozestup není příliš velký navzdory diametrálním rozdílům mezi testovanými programy. Hodnocena byla pouze požadovaná základní funkčnost programu, nikoli množství doplňujících služeb a jejich kvalita.

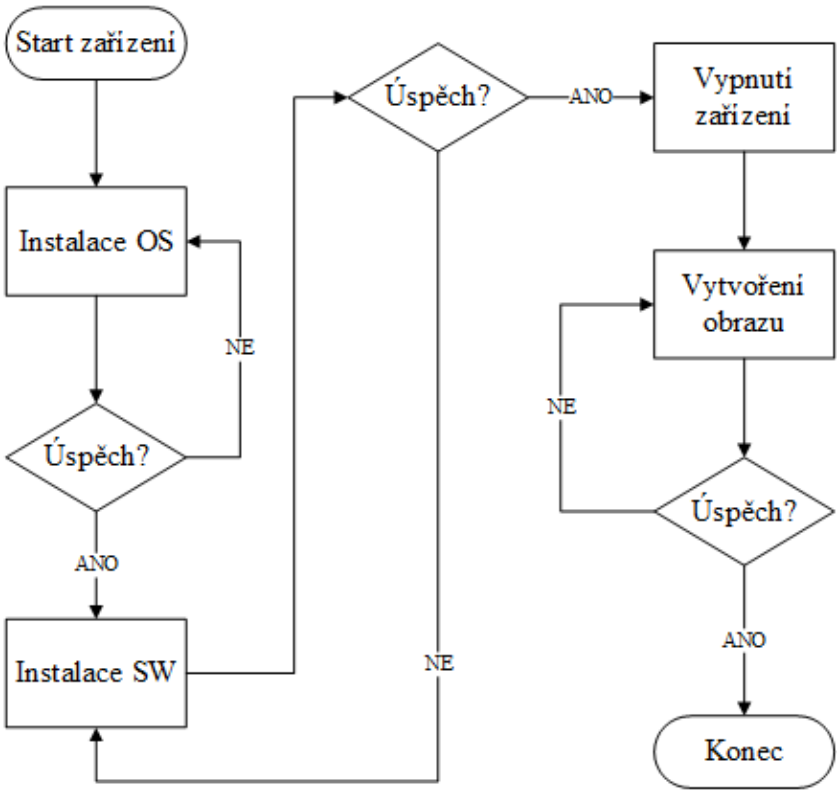
6. POSTUP VYTVOŘENÍ OBRAZU OPERAČNÍHO SYSTÉMU S KLIENTSKOU ČÁSTÍ

Pro vytvoření obrazu operačního systému, popřípadě celého disku, existuje mnoho programů a několik postupů, jak pro jeho vytvoření postupovat. Nejsnazší variantou je nainstalovat čisté Windows do virtuálního stroje, a následné nainstalování potřebného softwaru, včetně klientské části programu a korektně systém ukončit.

Vytvořit obraz disku nejsnazší metodou není příliš vhodné, jelikož tato metoda zahrnuje instalaci potřebného programu do onoho virtuálního systému.

Za vhodnou metodu lze považovat použití softwaru StarWind V2V Converter, který je schopen z virtuálního disku libovolné virtualizační platformy vytvořit obraz disku, či libovolný disk pro jinou virtualizační platformu. Tento software je dostupný zdarma.

Další možnost se nabízí v podobě spustitelného (bootovatelného) systému CloneZilla, který je schopen snadno naklonovat, či převést do souboru obrazu, nejen virtuální systém, ale i ten hostitelský.



Vývojový diagram 1: Schéma vytvoření image operačního systému.

7. ANALÝZA A NÁVRH SYSTÉMU

V praktické části se práce zabývá vytvořením systému tří aplikací pro umožnění a automatizaci vzdálené správy operačních systémů. Tato kapitola se zabývá specifikací výsledného programu, za použití nejvýhodnějších prostředků zjištěných v teoretické části práce. Systém bude vytvořen za použití programovacího jazyka C#.

Hlavním účelem systému je umožnit uživateli grafické aplikace vzdáleně se připojit na libovolný počet vzdálených operačních systémů ze zobrazeného seznamu. V okně grafické aplikace bude zobrazen seznam dostupných operačních systémů, jejich název, seznam dostupných uživatelů v dané instalaci operačního systému pro snadný výběr cíle vzdálené administrace. Funkčnost systému se neomezuje na lokální síť či podsíť, ale je možné jej použít na operační systémy umístěné ve vzdálených sítích.

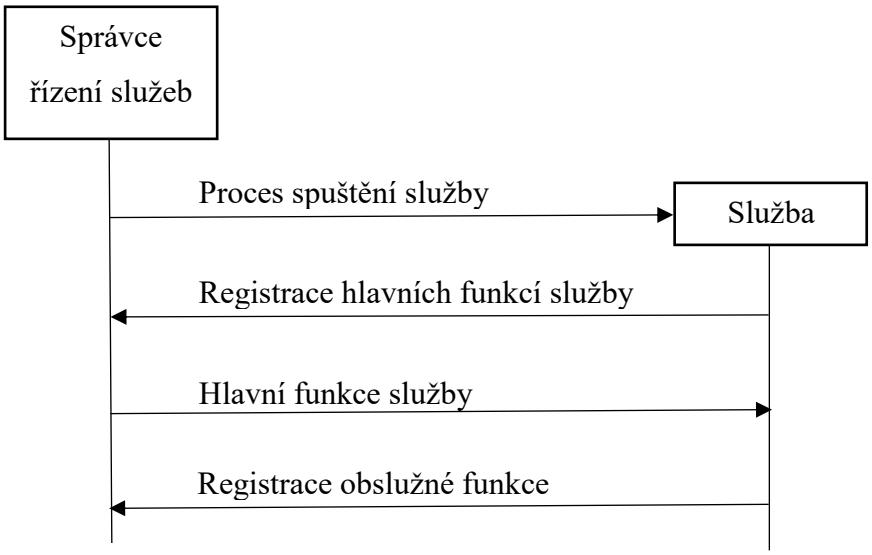
Celý systém nebude vyžadovat, aby klientské stanice, jako cíle připojení, tak uživatel grafické aplikace, měli stálou veřejnou IP adresu. Systém bude vytvářet trvalé spojení mezi klienty a serverem a tím umožňovat „tunelování“ spojení služby Vzdálená plocha (neboli Remote Desktop, dále RD). Systém bude mít svou vlastní ovládací komunikaci nezávislou na službě vzdálené plochy a RD protokolu. Trvalé spojení bude vytvořeno za použití potvrzovaného protokolu TCP. Klient, na straně operačního systému, ke kterému bude realizováno připojení vzdálenou plochou, bude realizován ve formě služby systému.

Ovládací komunikace mezi klienty a serverem bude šifrována za použití sdíleného klíče. K šifrování komunikace bude použita knihovna Cryptography, která je součástí .NET souboru knihoven. Konkrétně bude využito algoritmu Rijndael, který je součástí standardu Advance Encryption Standart (AES).

7.1 Klientská část – služba systému

Tuto část systému bude mít nainstalovaný cílový operační systém, ke kterému se chceme vzdáleně připojovat. Bude realizována formou služby systému Windows, kterou bude možné spouštět bez modifikací, pouze zkompilováním pro danou platformu s využitím softwaru třetí strany, a to i pod operačním systémem Linux. Pro případ nedostupnosti pevné IP adresy k definování cílového serveru, bude možné použít pro jeho definici i doménový název, který se následně přeloží na aktuální IP adresu serveru.

Služba se spustí automaticky po startu operačního systému pod lokálním systémovým účtem s názvem LocalSystem, který má omezená práva a k tomuto účelu je naprosto dostačující. Funkcionalita této části systému bude naprosto nezávislá na uživateli a konfiguraci cílového operačního systému. Tato část systému, služba, bude vyžadovat pouze aktivní síťové připojení a povolenou serverovou část služby Vzdálená plocha.

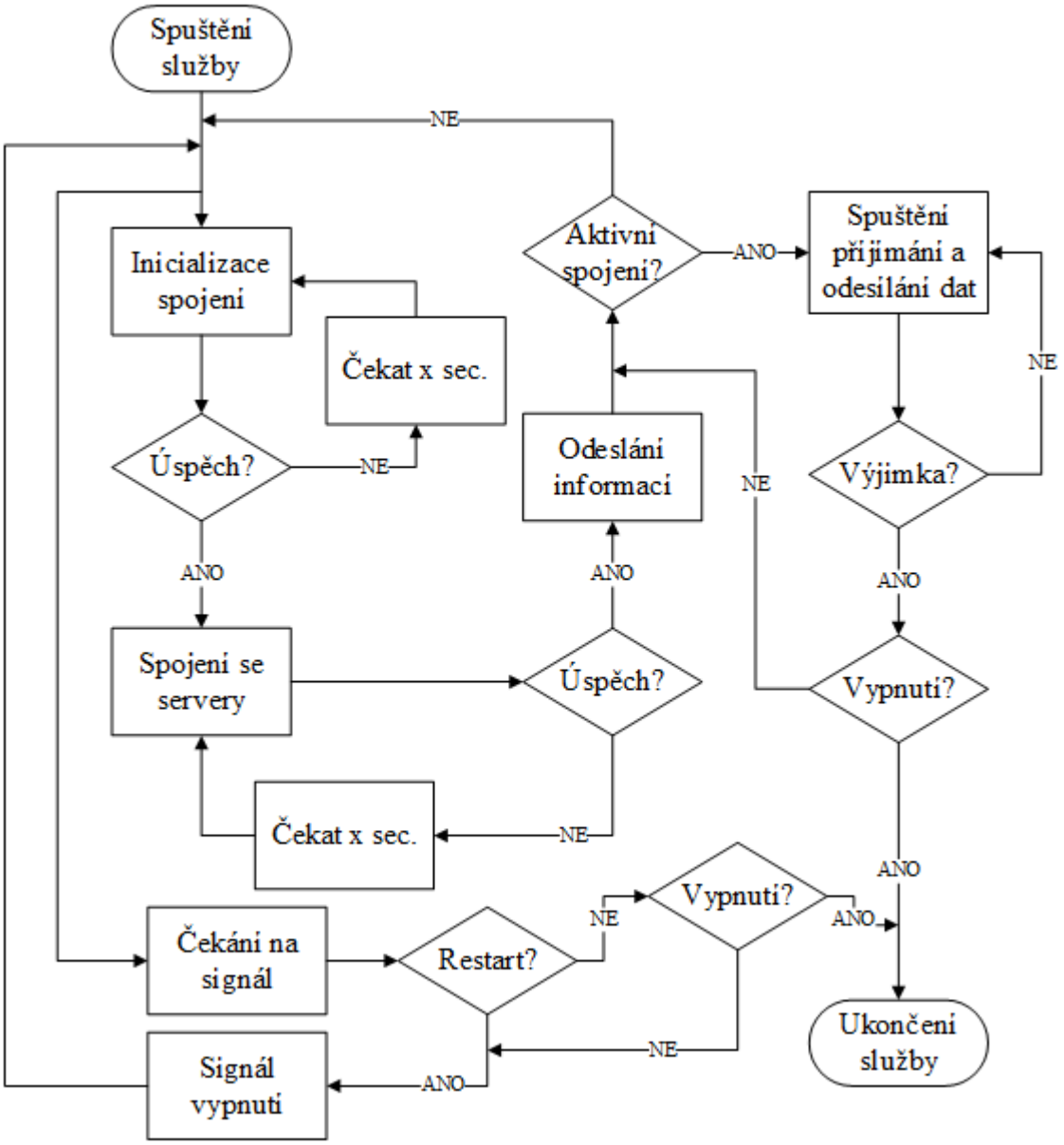


Vývojový diagram 2: Proces spuštění služby systému.

Po spuštění služba inicializuje spuštění navázání komunikace se serverem, a pokud vše proběhne korektně, což je nutná podmínka pokračování, tak dojde ke spuštění procesu navazování spojení se serverem. V případě neúspěchu inicializace se služba uspí, v pravidelných intervalech se probouzí a pokouší inicializovat vše nezbytné. Dokud není úspěšně navázané spojení se serverem, služba vyčkává a v pravidelném intervalu se pokouší o úspěšné připojení k serveru. Stejný proces inicializace a navázání spojení probíhá i vůči lokálnímu serveru služby vzdálená plocha (RD server). Neúspěšné navázání spojení s lokálním serverem vzdálené plochy má za následek ukončení i úspěšného připojení vůči serveru, který je součástí systému, a to z důvodu nemožnosti využití takového připojení pro vzdálené spojení s využitím Vzdálené plochy.

Úspěšným připojením vůči oběma serverům započne služba spouštět obslužná vlákna pro příjem a odesílání dat. Následně odešle serveru systému pro umožnění vzdálené správy

informace o systému, na němž je spuštěn. Zpráva obsahuje název počítače, seznam dostupných uživatelských účtů, s výjimkou systémových. Odesílání těchto informací probíhá v pravidelném cyklu s výjimkou stavu, kdy je spojení využíváno pro „tunelování“ Vzdálené plochy.



Vývojový diagram 3: Schéma funkčnosti služby.

Ze zobrazení vývojového diagramu 3 lze dovodit více vláknový návrh aplikace, kdy hlavní vlákno aplikace bude pracovat v roli dohlížitele, přijímajícího a zpracovávajícího podměty od operačního systému, popřípadě uživatele v podobě signálů pro restart, či ukončení služby.

7.2 Serverová část – konzolová aplikace

Nejdůležitější součást celého systému je server. Tato část systému bude vyžadovat aktivní síťové připojení a stálou IP adresu. Pro funkčnost celého systému mimo lokální síť je nezbytné, aby IP adresa přidělená serveru byla veřejná. Server bude realizován formou konzolové aplikace, tedy aplikace bez grafického rozhraní, pouze s textovým vstupem/výstupem. Server bude možné provozovat pod operačním systémem Linux.

Server v podobě konzolové aplikace, bude možné s minimem modifikací převést na službu systému, bude vyžadovat manuální spuštění nebo naplánování spuštění v plánovači operačního systému. Po spuštění binárního souboru se server automaticky spustí, ale nabídne v konzoli nabídku se vstupem, pro možnost restartování, ukončení a opětovného spuštění.

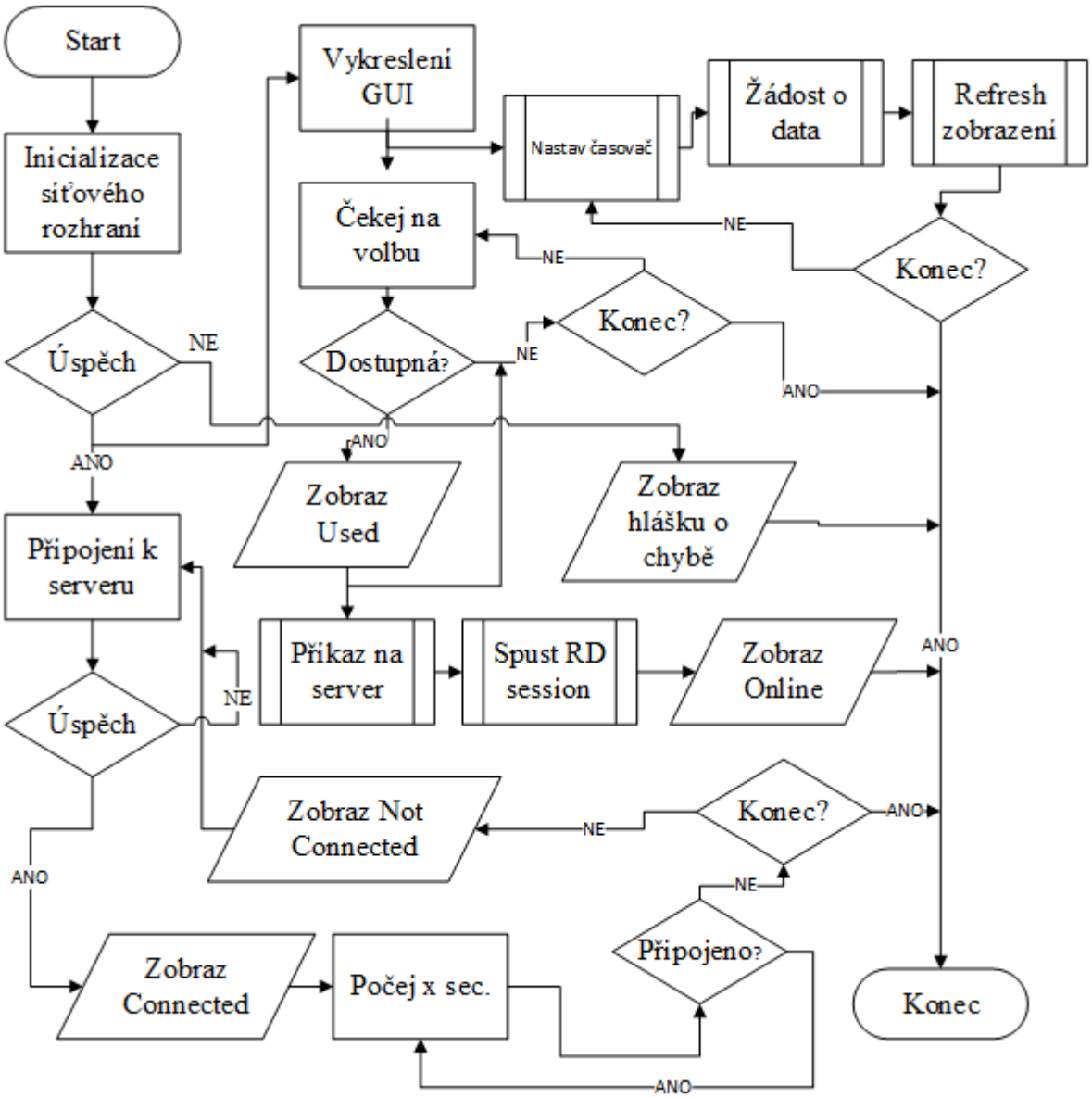
Základní funkcí této části systému bude naslouchat na dvou portech a přijímat žádosti o spojení na ně přijaté. Jeden port bude určen pro připojení klientů – služby systému a druhý pro připojení grafických aplikací a instancí Vzdálené plochy prostřednictvím nich spuštěné.

Server bude uchovávat tabulku s informacemi o připojených klientech na obou portech odděleně. Každý připojený klient bez ohledu na port připojení dostane na serveru unikátní ID (číselný kód) ve struktuře, kam bude i s informacemi uložen. Data poskytnutá klientem o vzdáleném systému, budou uchovávána ve struktuře spolu s informacemi nezbytnými pro komunikaci s klientem a průběžně budou přepisována, jak bude klient v stálém časovém intervalu zasílat aktualizace.

Ze strany grafické aplikace bude server očekávat řídící pokyny, pokyn k odeslání informací o klientech a pokyn k připojení a odpojení od daného klienta. Sever bude mít implementovaný postup pro identifikaci a mazání zastaralých záznamů, tj. odpojených klientů.

7.3 Grafická aplikace pro správu

Tato část systému pro vzdálenou správu bude implementována v podobě grafické aplikace s využitím Windows Forms a programovacího jazyku C#. Aplikace bude pro svůj chod vyžadovat aktivní síťové připojení a funkčního klienta vzdálené plochy, který je integrován ve Windows. Ke svému běhu aplikace nebude vyžadovat veřejnou IP adresu. Pro případ nedostupnosti pevné IP adresy pro definici serveru, bude možné použít pro jeho definici i doménový název, který se následně přeloží na aktuální IP adresu serveru.



Vývojový diagram 4: Schéma fungování grafické aplikace.

Aplikace se jako první pokusí o inicializaci síťového spojení. V případě úspěchu spustí paralelně vykreslování grafického rozhraní a vlákno pro obsluhu spojení se serverem. V případě neúspěchu inicializace síťového připojení aplikace vypíše chybovou hlášku a ukončí se.

V grafickém rozhraní bude zobrazen seznam dostupných klientů s informacemi nezbytnými pro rozlišení, jasnou identifikaci jednotlivých systémů a na nich dostupných uživatelů. Na závěr bude zobrazen stav daného záznamu, zdali je používán jinou aplikací, či již otevřenou instancí Vzdálené plochy. Zobrazena bude pouze informace „dostupný“, „nedostupný“ nebo „používaný“. Nedostupný je záznam od systému odpojeného klienta, který ještě nebyl vymazán z úložné struktury na serveru.

Dále bude aplikace umožňovat, s využitím kontextových nabídek, spuštění nové instance aplikace samotné a její ukončení. Další kontextová nabídka bude umožňovat manuální obnovení zobrazované tabulky (aktualizace dat), která bude mimo tuto funkci probíhat nezávisle na chodu programu v určitém časovém intervalu. Ve spodní části grafického zobrazení programu, tzv. patičce, bude zobrazen celkový počet klientů a stav aplikace, tedy zdali je aplikace připojena k serveru, či nikoliv.

8. IMPLEMENTACE

Tato kapitola práce je věnována popisu struktury jednotlivých prvků systému, které jsou z části společné pro celý systém, tedy všechny tři části systému mají společnou část zdrojového kódu, a z části mají odlišný zdrojový kód. K vytvoření implementace bylo využito prostředí Microsoft Visual Studio 2015.

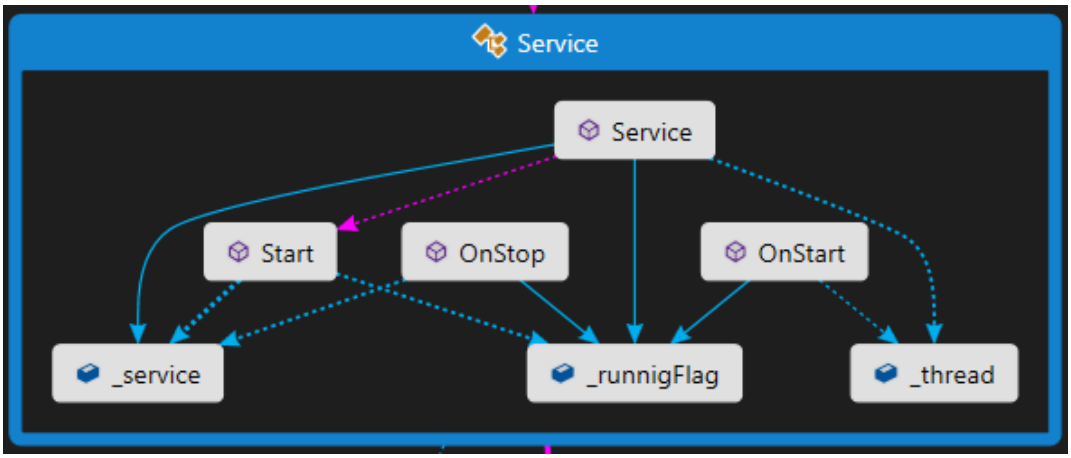
8.1 Klientská část – služba systému Windows

Třída Program

Jedná se o základní třídu aplikace, v níž byla vytvořena, inicializována a následně spuštěna instance třídy *Service*, která vychází ze stejnojmenné šablony pro tvorbu služeb systému Windows.

Třída Service

Tato třída implementuje základní ovládací rozhraní definované pro použití programu jako služby systému Windows. Třída je potomkem předdefinované třídy *ServiceBase*, do jejíž implementace nebylo zasahováno.



Obrázek 7: Mapa kódu třídy Service.

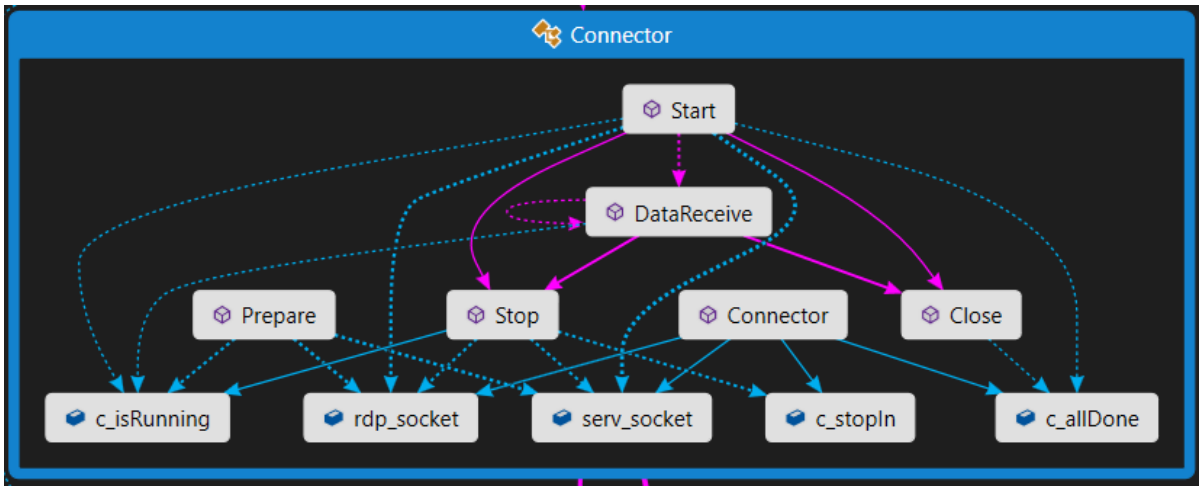
Ve třídě je formou soukromé proměnné definována instance třídy *Connector*, která implementuje samotnou funkčnost klientské části. Další soukromou proměnnou této třídy je typu bool, platformě nezávislý odkaz na datový typ *Boolean*, který může nabývat

dvou hodnot, pravda (true) a nepravda (false). Slouží zde jako Flag (vlajka), pro detekci ukončení programu. Poslední soukromou proměnnou je *Thread* (vlákno), které zde slouží ke spuštění instance třídy *Connector* s implementací nezávisle na vlákně spravující služby systému, pod nímž jsou spouštěny řídící signály.

Implementace obsahuje tři předdefinované metody, které jsou nezbytné pro spuštění a zastavení služby systému Windows. První z nich je metoda *OnStart*, která nastavuje vlajku a následně spouští vlákno s metodou *Start*, která inicializuje a spouští metody třídy *Connector*.

Třída Connector

Tato třída obsahuje hlavní část implementace klientské části systému. Obsahuje několik soukromých proměnných různých typů. První dvě proměnné, *serv_socket* a *rdp_socket*, jsou typu *Socket*, tedy struktura pro uložení informací potřebných pro úspěšné síťové spojení. Následuje soukromá proměnná, *c_allDone*, typu *ManualResetEvent*, která slouží k pasivnímu (nevytěžuje procesor) parkování procesu, který tak čeká na signál povolení dalšího běhu. Poslední dvě soukromé proměnné, *c_isRunnign* a *c_stopIn*, jsou typu bool a slouží k detekci ukončení a korektnímu uzavření síťových spojení.



Obrázek 8: Mapa kódu třídy Connector.

V konstruktoru dochází pouze k inicializaci jednotlivých soukromých proměnných. Pro přípravu spuštění byla vytvořena třída *Prepare*, v níž dochází k pokusu o vytvoření

dvou spojení. Jedno, vzhledem k serveru systému pro vzdálenou komunikaci, a druhé na lokální port vzdálené plochy (3389) na klientském operačním systému. Třída provádí v cyklu pokusy o připojení, dokud nejsou oba úspěšné, poté v cyklu kontroluje, zdali jsou spojení stále aktivní. V případě neúspěšného pokusu o připojení dochází k uspání vlákna na předem definovanou hodnotu, která je uložena ve statické třídě *Globals*. Ta slouží jako náhrada za neexistující globální proměnné v programovacím jazyku C#.

Metoda *Start* implementuje spuštění přeposílání komunikace mezi serverem systému a lokálním serverem vzdálené plochy, který je integrován v operačním systému. Samotné přeposílání probíhá asynchronně s využitím třídy objektu *Socket*, s názvem *BeginReceive*, která zpětně volá poslední implementovanou metodu této třídy s názvem *DataReceive*. Metoda *BeginReceive* vytváří pro přijetí dat nové vlákno, to původní je pak využito pro cyklické odesílání informací o klientské stanici.

Zpětně volaná metoda (tzv. callback) *DataReceive* obsahuje přijetí zprávy a následné odeslání po kontrole, zdali neobsahuje značky pro detekci řídících zpráv ze serveru. Na konci této metody následuje opětovné volání *BeginReceive*. Takto jsou vždy volány souběžně dvě vlákna obsluhující odesílání a přijímaní dat.

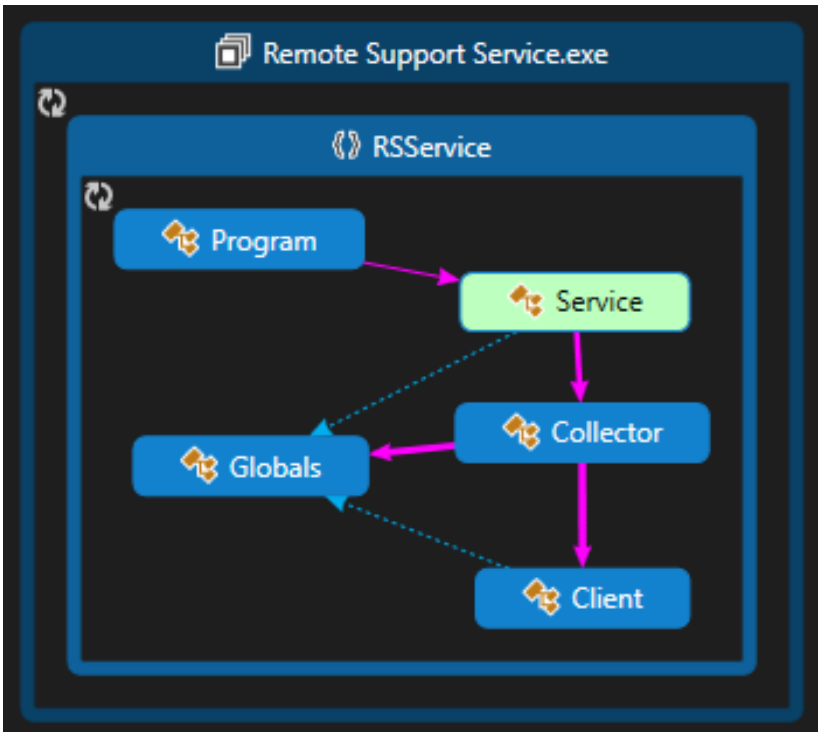
Třída Globals

Tato třída je implementována jako statická, tzn. že data v ní uložená patří třídě, nikoliv instanci z ní vytvořené. Pro tuto třídu nelze vytvořit instanci a její proměnné jsou přístupné v celém jmenném prostoru této služby systému. Obsahuje data jako je IP adresa a port serveru, proměnné, které určují, jak dlouho mají být uspávána jednotlivá obslužná vlákna, a nakonec několik statických metod. Tyto metody slouží k šifrování a dešifrování řídící komunikace mezi klientem/službou a serverem systému. Využit je šifrovací algoritmus RSA v módu CBC.

Poslední metodou této třídy je generátor náhodných bitů, který slouží ke generování inicializačního vektoru pro šifrování komunikace a generování soli (salt), což je pole znaků přidávajících se k heslu, kterým je komunikace šifrována, aby byla zvýšena bezpečnost hesla.

Třída ServiceInstaller

Projekt služby systému Windows také obsahuje třídu *ServiceInstaller*, která slouží k definici informací o službě. Tyto informace se předávají operačnímu systému při zavádění služby.



Obrázek 9: Map kódu služby systému Windows.

8.2 Server

Třída Program

Základní třída projektu serveru, kde dochází pouze k vytvoření instance typu *Service* a její inicializaci. Záměrně je struktura projektu serveru vytvořena tak, aby ji bylo možné s minimem úprav převést na službu systému Windows, popř. Linux. Pro účely konzolové aplikace bylo přidáno textové menu se vstupem dle výběru, které je cyklicky zobrazováno v terminálu. Toto menu je obsluhováno hlavním vláknem aplikace a při ukončení aplikace čeká na dokončení ostatních vláken.

Třída Service

Třída implementuje základní rozhraní podle šablony *Service*, která je potomkem třídy *ServiceBase*. Rozhraní je naprosto totožné s klientem, odlišné ve spouštěných instancích.

Konstruktor nastavuje hodnoty pro soukromé proměnné třídy. Třída obsahuje soukromou proměnnou typu *Server*, instance třídy implementující hlavní funkcionalitu. Dále obsahuje dvě proměnné typu bool pro správu běhu programu a detekci odpojení posluchače serveru. Poslední soukromou proměnnou je *Thread*, tedy vládkno, ve kterém se spouští metody instance třídy *Server*.

Stejně jako v případě služby klienta, i tato třída obsahuje metody *OnStart, OnStop, Start,* a jednu navíc, *ConsoleInput*, která implementuje konzolové ovládání.

Třída Server

Implementuje hlavní funkcionalitu této konzolové aplikace. Obsahuje dvě soukromé proměnné typu *Socket*, pro definici posluchačů na zvolených portech. Jeden posluchač pro klientské aplikace a druhý pro grafické aplikace. Dále obsahuje dvě indexovaná pole typu *Dictonary* (slovník). Jako index byla zvolena celočíselná hodnota. Takto indexované hodnoty jsou uložištěm informací o připojených klientech a aplikacích. Hodnoty o klientech jsou uchovávány v instanci třídy *ClientData* a *AppData*.

Konstruktor této třídy inicializuje všechny své soukromé proměnné a nastavuje jim výchozí hodnoty. Následuje metoda *Prepare*, která připravuje a nastavuje kontext k proměnným třídy. Metoda *Start* pak spouští posluchače pro nově se připojující klienty a aplikace. Implementována je také metoda Stop, která při ukončení programu odpojí všechny klienty a vynuluje všechny proměnné. Následují metody pro obsluhu plně asynchronního přenosu dat mezi grafickou aplikací a klientem na koncové stanici. Jedná se o metody zpětného volání, *AcceptCallBack* a *AppAcceptCallBack*, které zajišťují přijetí nově připojených klientů a spouštějí asynchronní metody *BeginReceive* pro zajištění přenosu dat. Příjem dat poté prování zpětně volané metody *DataReceive* a *AppDataReceive*. Ty obsahují synchronní volání dalších metod pro analýzu obsahu zprávy a rozlišení provozu od řídících zpráv systému.

Metody pro analýzu a provedení řídících příkazů jsou *AnalyzeData* a *AppAnalyzeData*, které následně synchronně volají metody *RunCommand* a *AppRunCommand*. První dvě uvedené metody detekují v bufferu výskyt znaků, které uvozují řídící příkazy. Napříč celým systémem to jsou znaky „#R>#“ pro začátek příkazu a „#R<# pro konec příkazu.

Analyzující metody po detekci výše zmíněný soubor znaků odstraní a zachovají pouze pole znaků mezi nimi. Následně se pokračuje k analýze dvěma posledními zmíněnými metodami.

Na serveru jsou implementovány tyto řídící příkazy přijímané od grafické aplikace:

- S1002 – příkaz pro získání seznamu připojených klientů,
- S1003 – příkaz pro určení klienta, ke kterému se bude připojovat,
- S1004 – zrušení volby klienta pro připojení.

Příkazy směřující na server od klientské služby jsou tyto:

- A1002 – klient zasílá seznam dostupných uživatelů,
- A1003 – klient zasílá název počítače na němž je spuštěn.

Příkazy směřující od serveru pro službu na klientském počítači:

- X1002 – zastavení zasílání informací o koncové stanici,
- X1003 – povolené zasílání informací o koncové stanici.

Vůči grafické aplikaci ze strany serveru probíhá jen strohá řídící komunikace v podobě potvrzení úspěšnosti volby klienta k připojení, či neúspěchu.

Ve třídě Server je implementována metoda *startDataController*, která je spuštěna v novém vlákně a cyklicky kontroluje databáze připojených klientů a aplikací. Pokud nalezne neplatné záznamy (odpojené klienty), odstraní je.

V poslední metodě, *startForwardingRdpSession*, je implementováno rozhraní pro přeposílání dat z aplikace na cílový operační systém, klienta. Toto rozhraní je voláno z metody aplikačního provaděče příkazů *AppRunCommand* a to ve dvou asynchronních vláknech.

Třída ClientData

Tato třída implementuje uložiště dat o připojených klientech. Obsahuje identifikační data přijatá od klienta a další prováděcí proměnné. Všechny proměnné této třídy jsou veřejné. Pro výpis dat za účelem jejich zaslání na žádost grafické aplikace je zde metoda *getString*, která vrací řádkový výpis informací o klientovi. V třídě je vyjma identifikačních a síťových informací uložen také status klienta, zámek obslužné relace (pro přijímání aktualizací identifikačních dat).

Klient může nabývat tří statusů:

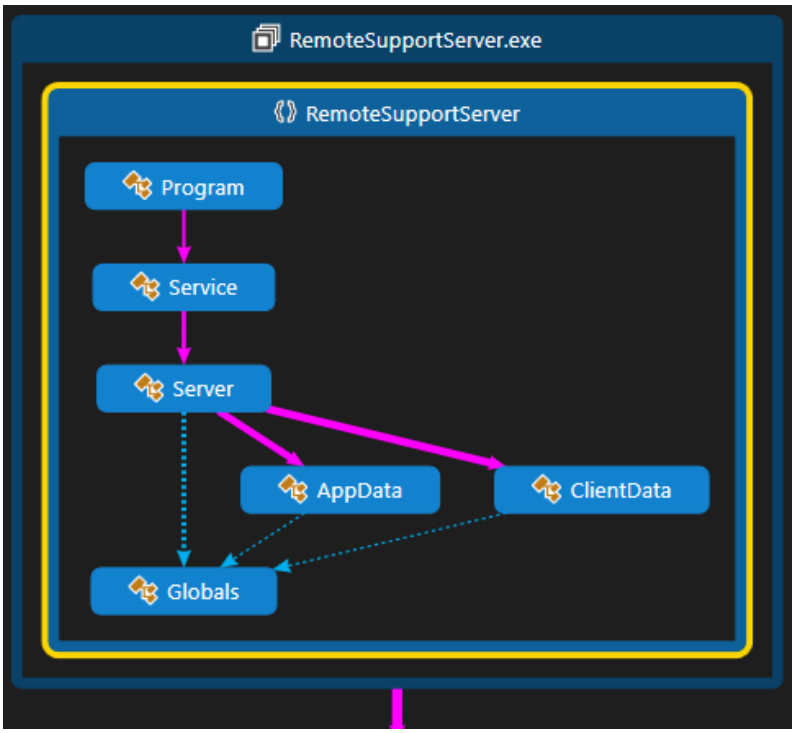
- Online – připojený a dostupný klient,
- Offline – odpojený klient, kdy jeho záznam bude vymazán,
- Used – grafickou aplikací používán, uzamčen pro jiné použití.

Třída AppData

Uložiště dat o připojených grafických aplikacích a jejich spojeních pro přeposílání vzdálené plochy. Ukládá se pouze *Socket* pro obsluhu síťového rozhraní, zámek záznamu a fronta zaznamenaných příchozích řídících příkazů v podobě pole znaků.

Třída Globals

Tato statická třída je téměř shodná s implementací v klientské aplikace v podobě služby systému Windows. Neobsahuje IP adresy, ale pouze porty, na nichž má server poslouchat a následné metody pro šifrování a dešifrování řídící komunikace.



Obrázek 10: Mapa kódu serveru.

8.3 Grafická aplikace

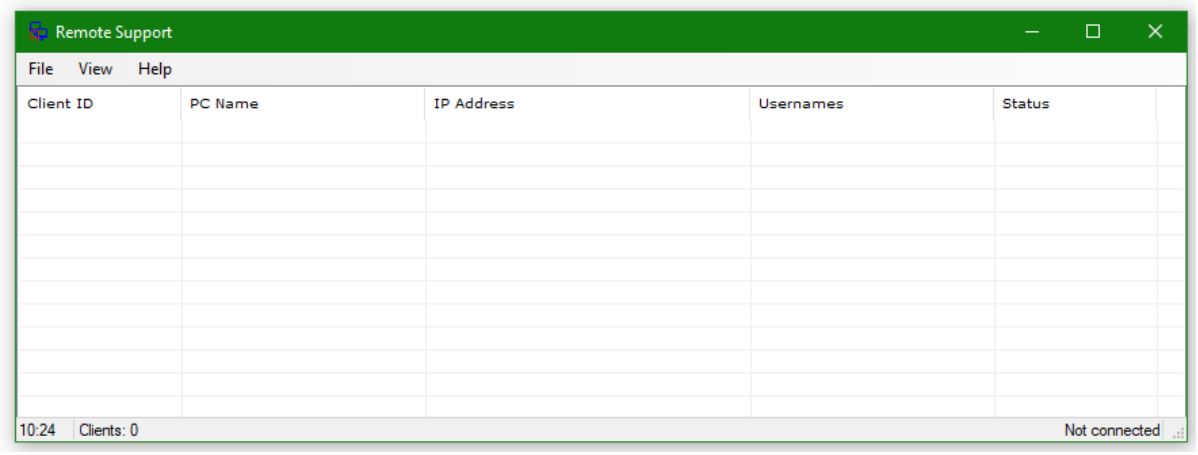
Třída Program

V případě grafické aplikace tato třída spouští instanci aplikace a její vizuální stránku, která je definována třídou *Window*.

Třída Window

Tato třída implementuje kompletní grafické rozhraní programu, včetně jednotlivých akcí na podměty vyvolané uživatelem v grafickém prostředí. Důležitou metodou je zde *timerTick*, která zajišťuje, vyjma zobrazování správného času, i pravidelné obnovené zobrazeného seznamu připojených klientů. Toto obnovení je možné vyvolat i za použití kontextového menu „View“, položky „Refresh view“, která zavolá metodu *refreshView_Click*. V obou zmíněných případech obnovení zobrazení se následně asynchronně volá metoda *refreshViewCallback*, která zajišťuje obnovování seznamu ze soukromé třídní proměnné typu *List<ListViewItem>*, do níž nahrává data metoda *getData* třídy *Client*.

Velmi důležitou součástí grafické aplikace je detekce volných portů, pro vytvoření posluchače klienta vzdálené plochy v metodě *findIPandPort*. Tato metoda zjistí volný port na volné lokální IP adrese a následně na ní vytvoří posluchače. V případě, že by nebyla volná žádná kombinace lokální adresy a portu, vypíše tato metoda chybovou hlášku a ukončí pokusy o vytvoření spojení na zvoleného klienta (vybraného v seznamu grafické aplikace). V případě úspěšného nalezení kombinace IP adresy a portu následuje vytvoření instance služby vzdálené plochy a spuštění klienta vůči posluchači. Služba vzdálené plochy je spuštěna jako nezávislý proces na procesu grafické aplikace a cílové parametry jí jsou předány pomocí RDP souboru (textový soubor s koncovkou .rdp), který je vytvořen v uložišti dočasných souborů systému. V průběhu připojení klienta vzdálené plochy na lokální server aplikace jsou vytvořena dvě vlákna, voláním metody *rdpForward*, pro asynchronní zprostředkování komunikace mezi lokálním serverem a serverem systému pro vzdálenou správu, který zprostředkovává komunikaci s koncovým klientem. Zvolení klienta se provádí dvojklikem na daný řádek v tabulce zobrazení. Tato akce vyvolá metodu *listView_MouseDoubleClick*, která prvotně otestuje, zdali je klient dostupný na základě údajů zobrazených v tabulce, a v případě, že je zvolený záznam ve stavu „Used“ nebo „Offline“, zobrazí chybovou hlášku a ukončí pokus o spojení. V případě, že prvotní test prošel, voláním metody *newRdpSession* započne příprava pro vytvoření spojení. V této metodě se testuje opětovně stav klienta, ale na serveru systému, nikoliv lokálně.



Obrázek 11: Zobrazení grafického rozhraní.

Třída Client

Instance této třídy vytváří permanentní spojení se serverem a zprostředkovává třídě *Window* komunikaci.

Konstruktor třídy inicializuje *Socket* pro připojení k serveru a nastaví výchozí status do stavu „Not connected“. Metoda *ConnectToServer* se pokusí o připojení, v případě úspěchu se status změnění na „Connected“. V případě neúspěchu je vyvolána výjimka a pokus se ukončí. Metoda *getData* zprostředkovává hlavnímu vláknu data pro vykreslení seznamu připojených klientů.

Z důvodu nepřístupnosti grafického rozhraní pro vedlejší vlákna byla implementována metoda *getStatus*, jejímž účelem je poskytnutí hodnoty soukromé proměnné *c_sockStatus*. Dále byly z tohoto důvodu implementovány metody *setStatus*, *getSocket*, *setRdpSocket*, *getRdpSocket*, jejichž názvy odpovídají účelu. Poslední metodou je *defineTarget*, která zasílá serveru volbu pro připojení a vrací „false“ v případě, kdy je volání úspěšné a k zvolenému klientovi je možné se připojit nebo „true“, kdy je volání neúspěšné a následující procesy spojování budou ukončeny. Hodnota „true/false“ je informace sdělující, zdali došlo k chybě, či nikoliv.

Třída Globals

Tato statická třída obsahuje definice základních běhových proměnných jako například g_bufferSize, která definuje velikost bufferu napříč programem. Jsou zde definovány časové

prodlevy mezi pokusy o spojení a mezi odsíláním požadavků na obnovení dat. Také je zde definována IP adresa a port serveru, ke kterému se aplikace po spuštění bude připojovat.

Součástí této třídy je část pro šifrování řídící komunikace, kterou mají všechny tři části systému pro vzdálenou správu společnou. Jedná se o tři metody, *Encrypt*, *Decrypt* a *GenerateRandomBits*.

9. TESTOVÁNÍ

Finální systém byl testován dvěma způsoby. První fáze testování probíhala v rámci jednoho počítače, kdy v operačním systému Windows 10 Pro byl povolen Hyper-V server a na něm nainstalováno pět nezávislých instalací operačních systémů s nainstalovanou službou pro umožnění vzdálené správy. Server byl spuštěn na hostitelském operačním systému. Jak hostitelskému, tak i virtuálním operačním systémům byly přidělovány IP adresy nezávisle, lokálním DHCP serverem umístěným na routeru Turris 1.1.

První testování bylo prováděno na notebooku Lenovo ThinkPad T430 2344-BUG, který disponuje procesorem i7-3520 (2x 2,9GHz – 3,2GHz), 16GB operační paměti, 180GB Intel SSD 525. Dvoujádrový procesor vytváří v systému čtyři vlákna, a tak bylo možné přidělit každému virtuálnímu stroji 60 % výkonu jednoho vlákna, což plně dostačovalo pro plynulý běh základní instalace Windows 10 i s nainstalovanou službou ve všech pěti virtuálních strojích a to bez omezení funkčnosti hostitelského operačního systému.

Každý virtuální stroj disponoval 60 % výkonu jednoho vlákna, 30 GB diskového prostoru na SSD disku, jedním síťovým rozhraním, 1 GB operační paměti. Instalován byl operační systém Windows 10 Pro.

Testováno bylo vytížení dostupných prostředků službou operačního systému umožňující vzdálené ovládání. Zjištěno bylo, že služba systému pro vzdálenou správu nevytěžuje zdroje tak, aby to bylo měřitelné s jedinou výjimkou, kterou je síťové rozhraní, na němž bylo detekováno odesílání informačních dat na server. Vyšší zatížení na síťové rozhraní měla až samotná vzdálená plocha, jejíž využití síťové karty již bylo znatelné. Mimo dobu používání spojení byly požadavky na systém malé, až zanedbatelné.

Měření dopadla se stejným výsledkem i v případě serveru, kdy byla měřitelná zátěž na cpu pod 5 % ve chvíli, kdy bylo inicializováno všech pět spojení současně. V klidovém stavu serverová část systému nevytěžovala hardware tak, aby to bylo měřitelné s jednou výjimkou, kterou bylo opět síťové rozhraní, kde byl znatelný přenos mezi klienty (aplikace i služby) a serverem.

Samotná grafická aplikace pak spotřebovávala od 5 MB operační paměti, kdy 5 MB bylo ve stavu bez připojení a načtených klientů. Během testování aplikace nepřekročila hranici 10 MB ani s padesáti připojenými klienty a osmi aktivními vzdálenými plochami. Vytížení

cpu bylo zanedbatelné, jelikož se pohybovalo mezi 1-5 %, což lze označit za rozsah chyby měření.

Druhou fázi testování lze označit jako „stress test“, kdy byla serverová část systému pro vzdálenou správu nainstalována na serveru s veřejnou IP adresou. Jednalo se o server Dell PowerEdge R210 II, který disponuje jedním procesorem Xeon E3-1220 v2 poskytující čtyři jádra o frekvenci 3,1GHz, 16 GB ECC operační paměti, 500 GB SSD disk Samsung Evo 850 Pro, 3TB WD RED. Na serveru byl nainstalován operační systém Windows Server 2016 Datacenter.

Z různých veřejný i neveřejných IP adres bylo navázáno padesát unikátních a dalších více než sto duplicitních spojení (na jednom operačním systému byla skriptem v cyklu spouštěna konzolová aplikace fungující naprosto shodně jako služba systému). V této fázi bylo navázáno všech padesát unikátních spojení vzdálené plochy. Ve fázi inicializace spojení bylo maximální vytížení procesoru na serveru od serverové aplikace 16 %, což bylo způsobeno z větší části generováním náhodných ID pro nově připojované klienty.

Ve třech případech se ukázalo problematické připojení na lokální službu vzdálené plochy z pohledu služby vzdáleného systému. V těchto případech cílový operační systém odmítal spojení (připojení vzdálenou plochou bylo povolené a funkční, bez využití localhost). Tento problém nebyl vyřešen, jelikož se jednalo o stabilně používané stroje třetích osob, a tak nebyla možná jejich podrobná analýza. Problém byl pravděpodobně způsoben nastavením operačního systému, popřípadě zásahem do DLL knihoven systému vzdálené plochy. Ten je nutný v případě, kdy u neserverových operačních systémů chceme umožnit připojení více vzdálených ploch k jednomu operačnímu systému a různým uživatelům.

Také sporadicky docházelo k resetu a následnému ukončení spojení při pokusu o vytvoření spojení vzdálené plochy prostřednictvím tohoto systému. Tato chyba byla způsobena pravděpodobným poškozením dat při přenosu. V takovém případě oba koncové uzly vzdálené plochy vyresetují spojení, což má za následek ukončení spojení vytvořeného grafickou aplikací a vytvoření nového, které již nemá možnost aplikace kontrolovat, a proto v takovém případě dojde k ukončení spojení a vzdálená plocha se nespustí.

10. ZÁVĚR

V rámci této bakalářské práce byl vyvinut systém pro vzdálenou správu virtuálních i nevirtuálních operačních systému, který byl úspěšně otestován jak za použití virtuálních systémů, tak i fyzických počítačů s jedním instalovaným operačním systémem. Výsledné navržené řešení je plně funkční i při použití klientské části na operačním systému, který je skrytý za překladem IP adres NAT (tedy operační systém nemá veřejnou IP adresu, ale pouze lokální). Plná funkčnost a stabilita programu byla otestována a shledána jako dostatečná pro bezproblémové užití, čímž zároveň došlo k naplnění cíle bakalářské práce.

Výsledné navržené řešení obsahuje tři nezávislé instalátory služby systému Windows, konzolové aplikace a grafické aplikace pro operační systém Windows. Řešení je možné s minimem úprav přenést na operační systém Linux, ať už samostatně nebo s použitím aplikací třetích stran pro převod Windows příkazů na Linuxové (služba systému).

Celý systém byl vytvořen v programovacím jazyce C# a technologie .NET s využitím vývojového prostředí Visual Studio 2015. Řešení nevyžaduje ani neobsahuje žádné aplikace třetích stran potřebné pro použití na operačním systému Windows.

Do budoucna by bylo možné aplikaci rozšířit o rozkládání zátěže mezi více serverů, změnit způsob šifrování a využít tak certifikáty namísto sdíleného klíče. Také by bylo možné rozšířit funkčnost grafické aplikace o další funkce, které nejsou nezbytné pro základní provoz systému.

11. BIBLIOGRAFIE

[1] J. Pomazal, „Virtualizace v kostce,“ 2010. [Online]. Available: https://www.systemonline.cz/clanky/virtualizace-v-kostce-htm. [Přístup získán 12 03 2017].

[2] „Virtualizace,“ 07 12 2016. [Online]. Available: https://cs.wikipedia.org/w/index.php?title=Virtualizace&oidid=14426264. [Přístup získán 12 03 2017].

[3] „Hypervisor,“ 25 03 2017. [Online]. Available: https://en.wikipedia.org/wiki/Hypervisor. [Přístup získán 26 03 2017].

[4] R. Beran, „Virtualizace operačních systémů,“ 01 11 2006. [Online]. Available: http://www.beranr.webzdarma.cz/virtualizace.html. [Přístup získán 01 04 2017].

[5] S. Lowe, Kompletní průvodce profesionální virtualizací 1, Brno: Computer Press, 2013, p. 509.

[6] S. Barrie, Mistrovství - počítačové síťe, Brno: Computer Press, 2010, pp. 795-800.

[7] A. Vitovský, Moderní slovník softwaru, 1. vydání editor, Praha: AV software, 2016, p. 428.

[8] C. Russel a S. Crawford, Microsoft WINDOWS SERVER 2008, 1. vydání editor, Brno: Computer Press, 2008, p. 950.

[9] C. Mellor, „Tintri adds Hyper-V to its virtualised server support,“ 15 12 2014. [Online]. Available: https://www.theregister.co.uk/2014/12/15/tintri_adds_third_virtualisation_tint/. [Přístup získán 18 03 2017].

[10] J. Vašek, „VNC a Vzdálená plocha - kouzlo vzdáleného přístupu,“ tyden.cz, 11 02 2009. [Online]. Available: http://pctuning.tyden.cz/software/jak-zkrotit-internet/12639-vnc_a_vzdalena_plocha-kouzlo_vzdaleneho_pristupu. [Přístup získán 18 03 2017].

[11] „VirtualBox Manual,“ [Online]. Available: https://www.virtualbox.org/manual/ch01.html. [Přístup získán 04 04 2017].

[12] R. P. Goldberk, Architectural Principles for Virtual Computer Systems, Cambridge: Harvard University, 1993.

[13] „SSH – bezpečné používání vzdáleného počítače a kopírování dat,“ [Online]. Available: http://www.dsl.cz/jak-na-to/jak-na-ssh. [Přístup získán 05 04 2017].

[14] „TeamViewer,“ TeamViewer, 2017. [Online]. Available: https://www.teamviewer.com/cs/features/. [Přístup získán 02 03 2017].

[15] J. Fesl, „Přednášky z předmětu distribuované a paralelní algoritmy,“ 2015.

[16] M. Virius, Od C++ k C#, České Budějovice: Kopp, 2002.

[17] M. Tulloch, Introducing Windows Server 2012 R2., Redmond, Washington: Microsoft, 2013.

[18] M. Virius, C#: hotová řešení, Brno: Computer Press, 2006.

[19] J. Hanák, Praktické objektové programování v jazyce C# 4.0, Brno: Artax, 2009.

[20] „Průvodce programováním v C#,“ Microsoft, 2016. [Online]. Available: https://msdn.microsoft.com/cs-cz/library/67ef8sbd.aspx. [Přístup získán 11 12 2016].

12. SEZNAM OBRÁZKŮ

13. SEZNAM TABULEK

14. SEZNAM VÝVOJOVÝCH DIAGRAMŮ

15. PŘÍLOHY

Příloha 1: CD s PDF verzí práce, zdrojovými kódy systému ve formě projektů pro Visual Studio.