Jihočeská univerzita v Českých Budějovicích

Přírodovědecká fakulta

Peer to peer systém pro vzdálené ovládání počítačů

Diplomová práce

Bc. Michal Lejtnar

Školitel: Ing. Jan Fesl

České Budějovice 2017

Jihočeské univerza v Českých Budějovicích
Přírodovědecká fakulta

ZADÁVACÍ PROTOKOL DIPLOMOVÉ PRÁCE

Student: Bc. Michal Lejtnar
(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:

Peer to peer systém pro vzdálené ovládání počítačů

Popis práce:

Diplomová práce bude věnována systému pro vzdálenou administraci počítačů, který bude realizován na bázi decentralizované peer to peer sítě. Systém bude umožňovat současné separátní ovládání více počítačů najednou a zároveň umožní současnou práci několika uživatelů najednou. Vlastní ovládání systému bude realizováno pomocí vhodné terminálové služby (Windows Remote Desktop, Windows Remote Assistance popř. Linux XRDP). Pro vytvoření systému použijte programovací jazyk C++ nebo C#.

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):

doc. RNDr. Dostálková, PhD. podpis: .....

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

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

V Českých Budějovicích dne: .....

Převzal/a dne: ..... podpis: .....

Bibliografické údaje

Lejtnar Michal, 2017: Peer to peer systém pro vzdálené ovládání počítače.

[Peer to peer system to remote control of computer, Mgr. Thesis, in Czech.] – 62 p., Faculty of Science, University of South Bohemia, České Budějovice, Czech Republic.

Anotace

Diplomová práce se zabývá tvorbou decentralizovaného peer to peer systému pro vzdálenou správu počítačů. P2P síť a jednotlivé uzly jsou inspirovány architekturou hybridní peer-to-peer sítě používanou v komunikačním nástroji Skype. Pro vzdálenou správu počítačů jsou použity dostupné terminálové služby operačního systému Windows. Jedná se o vzdálenou plochu (MS Remote Desktop) a vzdálenou pomoc (MS Remote Assistance). Celá aplikace je tvořena v programovacím jazyce C#.

Annotation

This thesis deals with creating of decentralized peer to peer system designed for remote control of computers. P2P network and single nodes in this network is inspired by hybrid peer-to-peer network architecture used by Skype application. The application uses terminal services available in operation system Windows for remote control of computers. Namely, MS Remote Desktop and MS Remote Assistance is used. The entire application is created in programing language C#.

Klíčová slova

Peer to peer síť, RDP, vzdálená plocha, vzdálená pomoc, multithreading, decentralizace

Keywords

Peer to peer network, RDP, Remote Desktop, Remote Assistance, multithreading, decentralization

Prohlašuji, že svoji diplomovou 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 diplomové 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. 4. 2017 ……

Bc. Michal Lejtnar

Poděkování

Na tomto místě bych chtěl poděkovat vedoucímu diplomové práce Ing. Janu Feslovi za odborné vedení práce, ochotu, trpělivost a cenné rady, které mi věnoval v průběhu tvorby této práce. Dále bych rád poděkoval celé své rodině za podporu při psaní diplomové práce i během celého studia.

Obsah

1 Úvod a motivace

Tématem této diplomové práce je tvorba peer-to-peer systému, který slouží pro vzdálené ovládání počítače. Motivací k tvorbě tohoto systému je fakt, že zatím neexistuje žádné podobné řešení vzdálené správy počítačů. Samozřejmě existuje spousta programů, které umožňují ovládat počítač na dálku, ale neexistuje žádný, který by fungoval decentralizovaně, tedy bez komunikace přes nějaký server. Veškerá dostupná řešení fungují na bázi klient–server spojení, kde hrozí nebezpečí výpadku celého systému, pokud by došlo k nějakému problému na serveru.

Aplikace, která je výsledkem této diplomové práce, funguje na bázi virtuální překryvné peer-to-peer sítě vytvořené nad fyzickou sítí např. Internet, kde žádné takové nebezpečí nehrozí, protože jednotlivá spojení jsou vytvářena přímo mezi zúčastněnými uzly. Samotná architektura aplikace je inspirována P2P sítí, kterou používá komunikační nástroj Skype. V této síti existují tři různé typy uzlů, kterými jsou Ordinary node, Super node a Central node. Ordinary node je pouze klientská aplikace pro konečného uživatele. Super node je také klientská aplikace, která zároveň slouží jako propojovací bod pro ostatní klientské aplikace. Central node je jediný centrální prvek aplikace, který je využíván pouze pro prvotní přihlášení, a veškerá další komunikace již probíhá přímo mezi jednotlivými uzly.

Vzdálená správa je zde řešena za pomoci terminálových služeb, které jsou standardně dostupné v operačním systému Windows. Jedná se konkrétně o službu vzdálená plocha a vzdálená pomoc, které jsou volány přímo z aplikace s potřebnými parametry pro připojení ke konkrétnímu uzlu tak, aby se již uživatel nemusel nijak zabývat konfigurací připojení.

Práce je rozdělena do tří hlavních částí. První částí je teoretický úvod do problematiky, popis peer-to-peer sítí a terminálových služeb. V závěru je tato kapitola doplněna o popis některých existujících řešení vzdáleného ovládání počítače. Druhá část se zabývá popisem návrhu celého řešení a výčtem technologií, které jsou použity při implementaci. Dále je zde uveden popis a funkce jednotlivých uzlů, které se vyskytují v peer-to-peer síti. V poslední části je popsána implementace celého řešení a jednotlivých funkcionalit.

2 Teoretický úvod

Tato kapitola slouží pro představení všech technologií použitých pro tvorbu aplikace. Zabývá se popisem a definicí sítě, která slouží pro komunikaci jednotlivých uzlů, a popisem terminálových služeb použitých pro vzdálené ovládání počítače. V závěru této kapitoly jsou popsána existující řešení vzdáleného ovládání počítače.

2.1 Peer-to-peer síť

Označení peer-to-peer nebo P2P se užívá pro označení typu síťové komunikace, kde spolu komunikují jednotliví uživatelé přímo. Podle [1] můžeme definovat peer-to-peer síť jako samoorganizační systém rovnocenných autonomních entit, který se zaměřuje na společné využití distribuovaných zdrojů v síťovém prostředí bez použití centrálních služeb. V takové síti jsou si všichni rovni a nepotřebují žádný centrální prvek v podobě serveru, jako je tomu v případě klient–server architektury. Čistá peer-to-peer architektura vůbec pojem server nezná. Zde každý prvek vykonává funkci klienta i serveru zároveň. To znamená, že může posílat dotaz nějakému klientovi a zároveň může odpovídat na dotaz jiného klienta. Oproti tomu v klasickém klient–server modelu může klient pouze posílat dotaz na server a čekat na jeho odpověď.

Navzdory výše uvedené definici se však v praxi namísto "čistých" P2P sítí vyskytují takové, ve kterých se nějaký centrální prvek v podobě specializovaného serveru vyskytuje. Ten zde většinou slouží pro potřebu prvotního připojení, nebo má funkci tzv. proxy serveru, který umožní vzájemnou komunikaci uzlů, které se z nějakého důvodu nedokážou spolu přímo spojit.

2.1.1 Základní principy peer-to-peer sítí

Základní principy sítí peer-to-peer se dají shrnout do několika bodů [2]:

- *Princip sdílení zdrojů* – každý účastník P2P systému (uzel) poskytuje některé zdroje ostatním uzlům (účastníkům) sítě peer-to-peer. Tyto zdroje mohou být fyzické, jako je např. výpočetní výkon svého počítače nebo diskový prostor (resp. přístup k souborům na svém disku), nebo logické, jako je např. určitý druh služby nebo specifická znalost.

- *Princip decentralizace* – části systému, nebo dokonce celý systém, nejsou řízeny centrálním prvkem. Každý prvek systému funguje jako klient i server. Příkladem plně

decentralizovaného systému je např. Gnutella a Freenet. Způsob adresování klientů v P2P systémech je nezávislý na DNS systému, který je centralizovaný a hierarchický.

- *Princip samoorganizace* – předchozí princip znamená, že v plně decentralizovaném systému P2P neexistuje centrální prvek, který by centrálně koordinoval aktivity nebo který by udržoval centrální databázi o celém P2P systému. Systém vykazuje samoorganizační chování, které je založeno na tom, jakou informaci mají jednotlivé uzly o okolních uzlech systému a jakým způsobem si tyto informace uzly vyměňují.

Kromě toho architektura a funkce systémů P2P někdy zahrnují taková další hlediska, jako je např. požadavek účastníků na to, aby zůstali anonymní, nebo skutečnost, že uzly v P2P systémech jsou často nespolehlivé např. z hlediska jejich stálé dostupnosti apod.

2.1.2 Výhody a nevýhody P2P sítí

Výhody:

- V čisté peer-to-peer architektuře neexistuje žádný bod selhání. Když jeden uzel padne, ostatní mohou bez jakéhokoli omezení pokračovat v komunikaci.
- Schopnost využít nepoužívané zdroje, jako je výpočetní výkon nebo úložná kapacita. Zatímco u klient–server architektury nese veškerou zátěž server, u peer-to-peer sítí je zátěž rozložena mezi uzly.
- Dokážou se vyhnout efektu úzkého hrdla (bottleneck effect) nebo přetížení části sítě, protože zde není žádný centrální bod.

Nevýhody:

- Nedokáže poskytnout vysoký bezpečnostní standard, který dnešní aplikace vyžadují.
- Spojení mezi uzly nejsou navržena pro vysokou propustnost, i když se neustále zvyšuje rychlost připojení.
- Nedostupnost služby poskytované určitým uzlem, který se náhle odpojil od sítě.
- Většina vyhledávačů je přizpůsobena hledání v centrálním katalogu než hledání po jednotlivých uzlech. Tento problém částečně řeší hybridní peer-to-peer architektura.

2.2 Historie peer-to-peer sítí

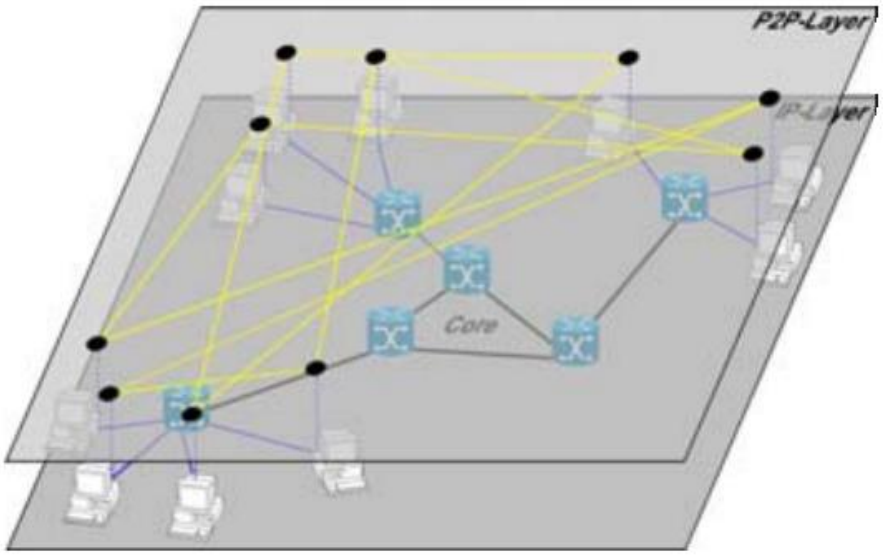
Ačkoliv se může zdát, že peer-to-peer sítě se začaly používat až v posledních letech, tak opak je pravdou.

Peer-to-peer sítě se poprvé objevily v 70. letech spolu se založením ARPANETu. Účelem této sítě bylo sdílení výpočetních zdrojů a dokumentů mezi různými výzkumnými zařízeními ve Spojených státech. V tomto původním systému neexistovalo nic jako typický klient nebo server a veškerá zařízení si byla rovna. Velkým rozdílem oproti dnešním peer-to-peer sítím je skutečnost, že tato síť nebyla vytvořena jako virtuální překryvná síť běžící nad fyzickou sítí, ale jednalo se o fyzické propojení počítačů do peer-to-peer sítě.[1]

Za počátek peer-to-peer sítí, jak je známe dnes, můžeme považovat až příchod aplikace na sdílení dokumentů s názvem Napster, která byla vytvořena v roce 1999. Tato síť již funguje jako samoorganizační virtuální překryvná síť, která je zcela nezávislá na fyzické síti.

2.3 Virtuální peer-to-peer síť

Virtuální překryvná P2P síť, známá jako overlay network, je topologie vytvořená pomocí navázaných TCP spojení mezi jednotlivými uzly. Takto vytvořená síť je kompletně nezávislá na fyzické IP síti. Představíme-li si počítačovou síť jako graf, kde jednotlivé počítače tvoří uzly a spojení mezi nimi jsou hrany grafu, pak můžeme říci, že neexistuje žádný vztah mezi hranami grafu na IP vrstvě a hranami vytvářenými mezi uzly na virtuální P2P vrstvě, což můžeme vidět na obrázku č. 1.



Obrázek 1: Nezávislost spojení vytvořených na IP vrstvě a spojení vytvořených na P2P vrstvě [3]

2.3.1 Rozdělení P2P sítí

Peer-to-peer sítě můžeme dělit podle různých kritérií, jako je účel, ke kterému bude síť použita, nebo míra centralizace. V základu však dělíme peer-to-peer sítě na strukturované a nestrukturované.

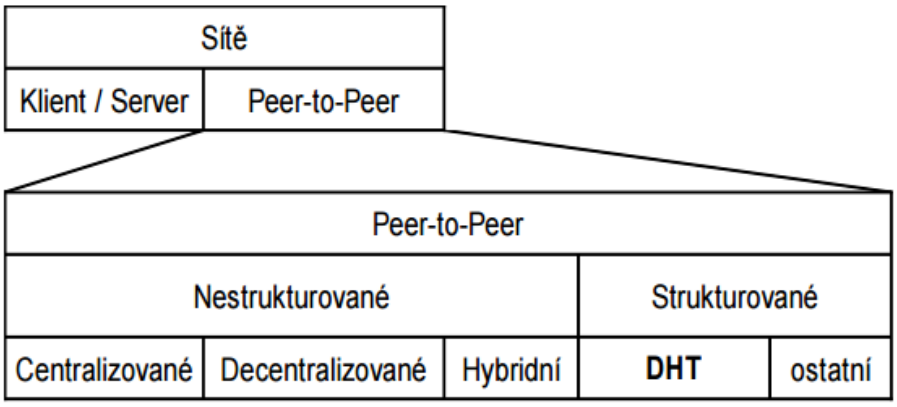
Klasifikace podle účelu použití[4]:

- *sdílení informací*
    - *sdílení souborů* (Napster, Kazaa)
    - *sdílení softwaru a aktualizací* (aktualizace hry World of Warcraft)
    - *sdílení informací o dostupnosti* (dostupnost uživatele v IM programech)
- *sdílení šířky pásma*
    - *load balancing* – každý uživatel, který pošle vlastníkovi dotaz na data, si tato data zkopíruje k sobě. Další žádosti o stejná data už nemusí jít přímo k vlastníkovi, ale také k uživatelům, kteří vlastní jejich kopii.
    - *sdílené užití šířky pásma* – každý uživatel, který pošle vlastníkovi dotaz na data, dostane zpět pouze určitou část z těchto dat. Následně si tito žadatelé vymění obdržené části mezi sebou.

- *sdílení úložného prostoru* – tento typ sítě vytváří cluster z připojených uživatelů, kteří sdílejí svou diskovou kapacitu, např. PAST, Pasta, OceanStore
- *sdílení výpočetního výkonu* – z jednotlivých počítačů seskupených do clusteru vznikne jeden virtuální superpočítač, který svou výpočetní kapacitou předčí i ty nejdražší superpočítače. Nejznámější aplikací je SETI@home

Klasifikace podle architektury:

- *Strukturovaná síť*
- *Nestrukturovaná sít*
    - *Centralizovaná*
    - *Decentralizovaná*
    - *Hybridní*



Obrázek 2: Klasifikace P2P sítí podle architektury [9]

2.3.1.1 Strukturovaná síť

Strukturovaná P2P síť je organizována do specifické topologie. Obsah zde není umisťován nahodile na jednotlivé uzly, ale do přesně definovaných míst. To umožní rychlejší vyhledávání. V těchto systémech jsou využívané distribuované hashovací tabulky (Distributed Hash Table – DHT), které obsahují záznamy ve tvaru klíč–hodnota, kde klíč je identifikátor uzlu a hodnota je hash konkrétních dat. Pokud pak nějaký uživatel vyhledává data, pak pomocí

DHT zjistí, na jakém uzlu jsou uložena. Mezi nejznámější strukturované sítě patří Content Addressable Network (CAN), Chord, Tapestry, Pastry a Kademlia.

2.3.1.2 Nestrukturovaná síť

V nestrukturovaných sítích není žádný prvek, který by se staral o řízení celé topologie nebo rozmístění zdrojů. Když se do sítě připojí nový uzel, pak si může vybrat libovolný uzel, se kterým vytvoří připojení a tím se stane jeho sousedem. Díky tomu, že zde chybí jakákoliv topologická omezení, tak má tento typ sítě lepší výkonnost a je zde méně režijní zátěže, i když dochází k častému odpojování a připojování uzlů. Na druhou stranu je zde větší problém při získávání požadovaných zdrojů. U nestrukturovaných sítí je nižší pravděpodobnost získání požadovaných zdrojů a je možné, že požadovaná data nebudou vůbec nalezena i přes to, že se v síti nacházejí. K nalezení zdroje jsou používány dvě základní směrovací operace, kterými jsou flooding a random walk. Flooding rozešle dotaz na všechny své sousedy a čeká na odpověď, zatímco random walk pošle dotaz vždy jen na jeden sousední uzel.

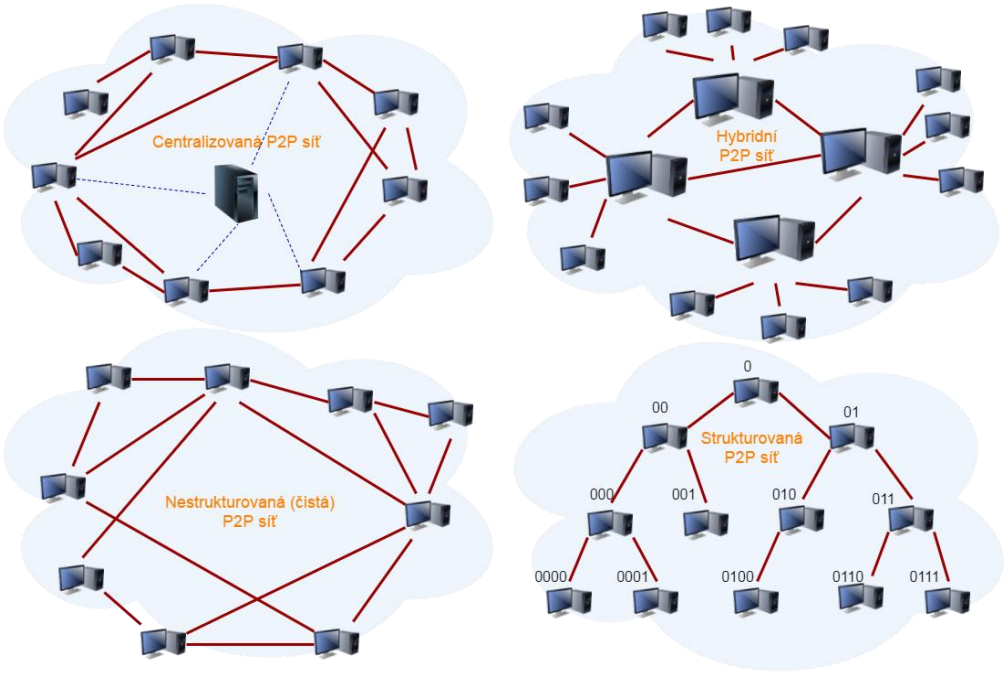
Nestrukturované peer-to-peer sítě se dále dělí podle míry centralizace na centralizované, decentralizované a hybridní.

Centralizované P2P systémy kombinují funkce centralizované (klient–server) a decentralizované architektury. Jako u klient–server systémů i zde existuje jeden nebo více centrálních serverů, které pomáhají uzlům určit, kde se nachází požadované zdroje. Pokud chce uzel získat nějaké zdroje, pošle dotaz na centrální server, který zná adresu uzlu, kde se požadované zdroje nachází. Oproti klient–server systému už komunikace dále neprobíhá přes server, ale naváže se spojení přímo mezi koncovými uzly.

V decentralizovaných P2P systémech mají uzly stejná práva a odpovědnosti. Neexistuje zde žádný centrální server a propojením jednotlivých uzlů vzniká plochá nestrukturovaná síťová topologie. Každý uzel má informace jen o svých sousedních uzlech. To jsou uzly, mezi kterými existuje hrana grafu. Uzel, který hledá konkrétní zdroje, musí tedy poslat dotaz na všechny své sousedy, kteří pokračují v postupném zaplavování sítě až do chvíle, než jsou zdroje nalezeny. Velkou nevýhodou je, že takové vyhledávání obrovskou mírou zahlcuje celou síť a není zaručené, že budou požadované zdroje nalezeny.

Hybridní peer-to-peer sítě jsou kombinací výhod centralizovaných sítí (rychlé a spolehlivé nalezení zdrojů) a decentralizovaných sítí (škálovatelnost). V hybridních P2P systémech se logicky nemůže vyskytovat běžný server, protože by byla znemožněna škálovatelnost takové

sítě. Místo klasického serveru jsou do této sítě umístěny speciální uzly, které mají větší schopnosti a větší odpovědnost než ostatní běžné uzly. Tyto centrální uzly se nazývají super uzly (supernodes). Super uzly tvoří vyšší vrstvu peer-to-peer sítě, která poskytuje ostatním uzlům podobné služby jako servery v centralizovaných P2P sítích. Jedná se především o funkci centrálního adresáře, kde může být připojenému uzlu nebo uloženému obsahu přiřazena konkrétní IP adresa. Další funkcí je řízení provozu mezi jednotlivými uzly. Běžnou praxí je, že běžný uzel se nejprve připojí k super uzlu, aby se zaregistroval do sítě, a další komunikace již probíhá přímo mezi uzly. Pokud však z nějakého důvodu není možné, aby dva uzly komunikovaly přímo, pak super uzel funguje jako proxy server pro tyto uživatele. Výborným přirovnáním k tomuto systému může být běžný kontakt mezi osobami v lidské společnosti. I zde se vyskytují více společenské osoby, které mají více znalostí a kontaktů než běžná osoba, např. profesor, mentor [5].



Obrázek 3: Rozdělení peer-to-peer sítí.

2.3.2 Průchod P2P sítí skrz NAT

NAT (Network Address Translation) je mechanizmus, který provádí modifikaci IP adresy a čísla TCP/UDP portu v paketu. Zpravidla se tak děje při přechodu paketu z privátní sítě do veřejného Internetu. Technika překladu adres funguje tak, že zařízení, které provozuje NAT, dočasně přidělí vnější (veřejný) port jedné privátní IP adrese, ze které přišel dotaz. Toto

mapování po určitou dobu udržuje a veškeré odpovědi, které dorazí na veřejný port, jsou nasměrovány na privátní IP adresu tazatele.

Tento mechanizmus je velmi užitečný, obzvláště pro domácnosti nebo firmy, kde je omezený počet veřejných IP adres a větší množství zařízení. V běžném spojení typu klient–server nečiní žádné potíže, protože dochází k připojení klienta na server, který má veřejně dostupnou IP adresu. Problém nastává, pokud chceme propojit P2P uzly mezi sebou. Jak již bylo uvedeno, každý uzel může být zároveň klientem i serverem. Pokud budeme chtít iniciovat spojení k uzlu, který se nachází za NATem, pak se spojení nikdy nenaváže, protože NAT zařízení neumožní průchod příchozímu spojení na privátní adresu, pokud mu nepředcházelo odchozí spojení, při kterém by došlo k mapování privátní IP adresy na veřejný port.

Řešení tohoto problému vyžaduje od výrobců NAT zařízení, designérů síťových protokolů a vývojářů P2P aplikací, aby poskytovali hladkou, přímou, obousměrnou komunikaci, která bude umožňovat průchod nevyžádaných příchozích spojení. Existují různorodé techniky průchodu NATem, které vytváří virtuální spojení mezi koncovými uzly. Jedná se o vytváření tunelu v NAT bráně (UPnP, ALG) nebo použití nějakého randezvous serveru pro vytvoření spojení (STUN, TCP/UDP hole punching)[6].

2.4 Terminálové služby

Terminálová služba je jedna z technologií vzdáleného přístupu, která umožňuje uživatelům přístup k programům, které jsou instalované na terminálovém serveru, nebo jim umožňuje plnohodnotný přístup k pracovní ploše vzdáleného počítače. V prostředí podniku umožňuje snadnější instalaci a údržbu používaného softwaru. Není nutné instalovat software na jednotlivé PC v podniku. Program se nainstaluje pouze na server, který podporuje terminálovou službu, a všichni uživatelé mohou tento program na serveru spouštět odkudkoli z podnikové sítě nebo Internetu.

Terminálové služby jsou dostupné v operačním systému Windows od verze Windows NT 4.0 Server, Terminal Server Edition, která byla vydána v roce 1998. Pro využívání těchto služeb Windows vytvořil vlastní protokol s názvem Remote Desktop Protocol, který zajišťuje přenos videa a signálu z klávesnice a myši. V roce 2001 se terminálové služby objevily také na klientské platformě operačního systému s názvem Windows XP. Standardní instalace této verze systému používala technologii terminálové služby pro několik úkolů, mezi které patří:

- *Klient terminálového serveru* – jedná se o RDP klienta, který umožňuje přístup uživatele k serverům, na kterých běží terminálové služby.

- *Rychlá změna uživatele* – umožňuje uživateli spustit aplikaci v pozadí, zatímco je ke stejnému počítači přihlášený jiný uživatel.

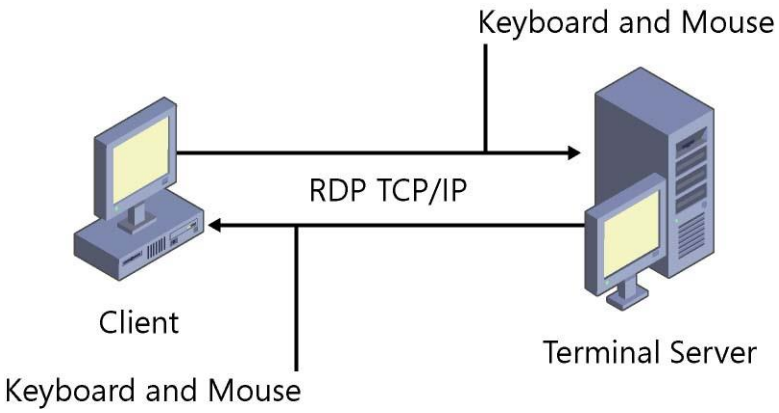
- *Vzdálená pomoc* – uživatel může požádat o pomoc nějakého experta, který pak může ovládat jeho počítač, a uživatel může sledovat, co se děje. Jedná se o sdílené používání uživatelské konzole.

- *Vzdálená plocha* – terminálový server je dostupný také na klientské platformě systému. Díky tomu může uživatel ovládat svou klientskou stanici z jiného počítače. Tato funkce je však dostupná pouze pro uživatele s administrátorskými právy.

2.4.1 Remote Desktop Protocol

V květnu 1997 začal Microsoft vyvíjet protokol pro výměnu dat mezi terminálovým serverem a běžným Windows OS klientem. Tento protokol se nazývá RDP (Remote Desktop Protocol) a je založen na standardech mezinárodní telekomunikační unie (ITU). Zejména je založen na standardech rodiny protokolů T.120, konkrétně se jedná o T.125 Multipoint Communication Service protokol a T.128 Application Sharing. [7]

Vzdálené připojení pomocí RDP protokolu může navázat jakékoliv zařízení, které má nějaké výstupní médium (monitor), klávesnici a myš. Tato zařízení musí mít nainstalovaný potřebný software, který využívá RDP protokol. Ve všech verzích operačního systému Windows je již v základu nainstalovaná klientská aplikace. Tato klientská aplikace se připojuje k RDP serveru. RDP server není součástí každé verze Windows, ale je instalovaný pouze na vyšších verzích Windows. RDP protokol není výsadou operačního systému Windows, ale v průběhu času byl implementován i do jiných operačních systémů (např. Mac OS, unixové systémy).



Obrázek 4: Komunikace mezi klientem a RDP serverem[8].

Remote Desktop Protocol je vytvořen jako vícekanálový protokol, který poskytuje 64 000 oddělených kanálů pro přenos dat. Protokol je navržený tak, že používá vždy jen jeden kanál pro přenos prezentačních dat i vstupů z klávesnice či myši. Další předností RDP je nezávislost na topologii sítě a podpora více síťových protokolů (IPX, NetBios, TCP/IP).

RDP přišlo s množstvím vlastností, které ho odlišují od ostatních protokolů. Mezi tyto vlastnosti patří šifrování, redukce šířky pásma, roaming disconnect, mapování clipboardu, přesměrování tisku a vzdálená kontrola.

- *Šifrování*
  RDP používá RC4 šifry, které zajistí bezpečnou komunikaci přes síť. Administrátoři mohou zvolit mezi 56- nebo 128bitovým klíčem pro šifrování dat.

- *Redukce šířky pásma*
  RDP podporuje různé mechanizmy pro snížení množství dat přenášených po síti. Mezi mechanizmy patří ukládání do mezipaměti, trvalé ukládání do bitmapy nebo komprese dat.

- *Roaming disconnect*
  Pokud se uživatel manuálně odpojí od RDP serveru bez předchozího odhlášení, pak při dalším připojení ze stejného či jiného zařízení pokračuje ze stejného stavu, ve kterém se nacházel před odpojením.

- *Mapování clipboardu*
  RDP umožňuje kopírování textu ze vzdáleného zařízení na lokální počítač a obráceně.

- *Přesměrování tisku*

  Vlastnost, která dovoluje vzdálené aplikaci, aby použila tiskárnu, která je připojena k lokálnímu počítači.

- *Vzdálená kontrola*

  Administrátor systému může díky této vlastnosti jednoduše vzdáleně řídit připojený počítač, provádět diagnózy a řešit problémy bez toho, aby musel být fyzicky přítomný u počítače.

2.4.2 Připojení ke vzdálené ploše

Připojení ke vzdálené ploše, známé jako Remote Desktop Connection, dříve Microsoft Terminal Services Client, je klientská aplikace, pomocí které se uživatel může vzdáleně připojit k serveru vzdálené plochy (Remote Desktop Server). Tento nástroj, který je dostupný ve všech verzích operačního systému Windows od verze Windows XP, umožňuje přistupovat k počítači či serveru ze vzdáleného místa. Jakmile se uživatel připojí ke vzdálenému počítači, získá nad ním plnou kontrolu. To znamená, že může počítač ovládat stejným způsobem, jako kdyby seděl přímo před ním. Může přistupovat k dokumentům, spouštět nejrůznější programy a používat veškerá periferní zařízení, která jsou k němu připojena.

Abychom byli schopni se k hostitelskému zařízení připojit, musíme znát přihlašovací údaje k administrátorskému účtu nebo jednomu z běžných účtů používaných na hostitelském počítači. Po přihlášení k hostitelskému počítači se naše aktivita nebude zobrazovat na displeji. Zobrazí se pouze obrazovka s hláškou o uzamčení počítače a aktuálně připojený uživatel bude odhlášen.

Před prvním použitím vzdálené pomoci musíme povolit používání vzdáleného připojení na hostitelském počítači. Ve výchozím stavu je připojení ke vzdálené ploše zakázáno. Pokud toto nastavení budeme chtít změnit, musíme jít do nastavení systémových vlastností *Systém → Vlastnosti systému* a v části *Vzdálená plocha* změnit nastavení na *Umožnit vzdálené připojení k tomuto počítači.* Pokud se v systémových vlastnostech nezobrazuje část s názvem *Vzdálená plocha*, tak je to z toho důvodu, že ne všechny verze operačního systému mají dostupné služby terminálového serveru. Služba terminálového serveru je dostupná ve vyšších edicích OS Windows (Professional, Enterprise, Ultimate).

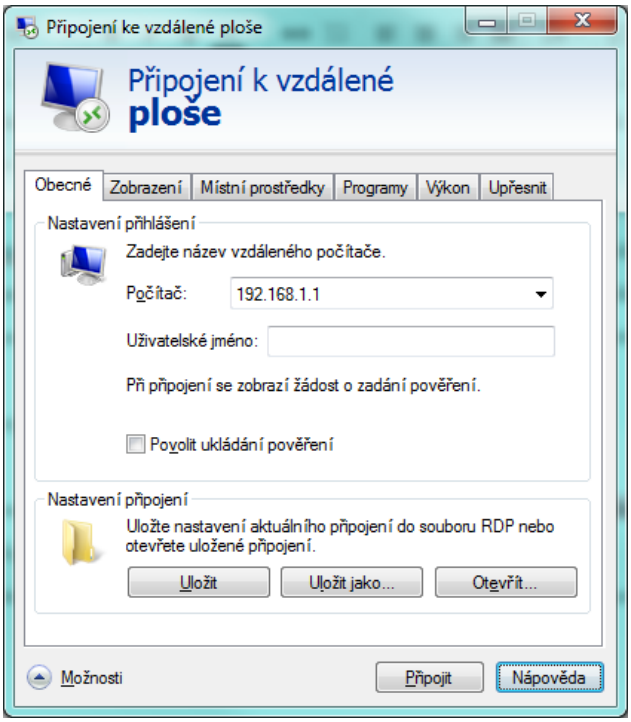
Další možnost, jak změnit nastavení vzdálené plochy, je změna hodnoty klíče v systémových registrech. Nastavení nalezneme pod klíčem *Terminal Server*, který je umístěn

v *HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server*. Pod tímto klíčem změníme hodnotu s názvem *fDenyTSConnections* z hodnoty 1 (zakázáno) na hodnotu 0 (povoleno). Na rozdíl od nastavení ve formuláři *Vlastnosti systému* je natavení v registrech dostupné ve všech edicích OS Windows. Pokud však povolíme vzdálenou plochu v edici, která nenabízí služby terminálového serveru, tak přesto nebude možné se k tomuto zařízení připojit přes vzdálenou plochu.

Pokud uživatel potřebuje zprovoznit *Připojení ke vzdálené ploše* na počítači, který má nainstalovanou nižší edici systému Windows, pak existuje relativně jednoduché řešení. Řešení se nachází v podobě patche (Concurent RDP patcher). Použitím tohoto patche však dochází k porušení licenčního ujednání, které se vztahuje k operačnímu systému Windows.

2.4.2.1 Proces připojení

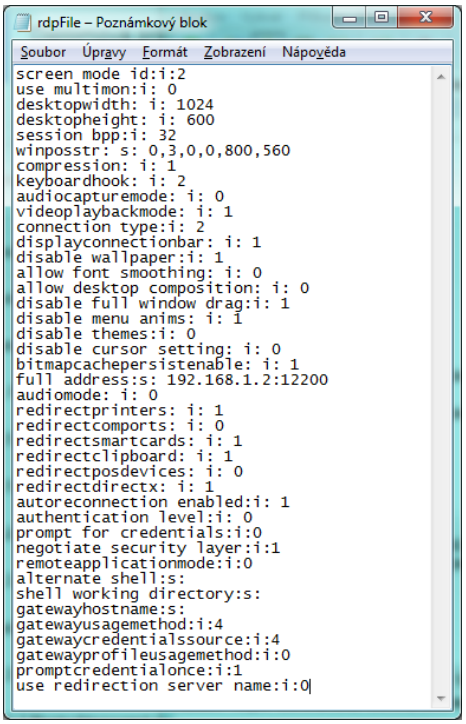
Při pokusu o připojení k hostitelskému počítači klientský počítač pošle signál na IP adresu a port hostitelského počítače prostřednictvím portu 3389 (naslouchací port), ve kterém se ptá na pověření k připojení a přihlášení. V odpovědi hostitelského počítače je žádost o přihlašovací údaje, které následně porovná s listem uživatelů vzdálené plochy. Jakmile dojde k úspěšnému přihlášení, začne mezi připojenými počítači přenos dat obsahujících informace o zobrazené ploše, stisknutých klávesách na klávesnici a pohybech myši. Je nutné mít na paměti, že vzdálená plocha umožňuje pouze jediné současné připojení. To znamená, že pokud se někdo pokusí o vytvoření druhého připojení k hostitelskému počítači, pak dojde k automatickému ukončení prvního spojení.



Obrázek 5: Připojení ke vzdálené ploše - GUI

2.4.2.2 Soubor .RDP

Soubor s příponou .rdp se používá pro uložení hlavního nastavení klientské části vzdálené plochy. Nejjednodušší způsob, jakým lze RDP soubor vytvořit, je pomocí programu *Připojení ke vzdálené ploše*. Uživatel si zvolí požadované nastavení pro připojení a vybere *Uložit jako*. Tím se vytvoří soubor s příponou .rdp, který pak může dále upravovat v textovém editoru. Struktura takto vytvořeného souboru je relativně jednoduchá, a proto je možné RDP soubor upravovat v textovém editoru. Standardní soubor sestává z několika řádek, kde každá řádka obsahuje název parametru, typ a hodnotu. Jednotlivé části v řádku jsou od sebe oddělené pomocí dvojtečky. Jak je možné vidět na obrázku č. 6, každý řádek začíná názvem parametru, za kterým následuje typ hodnoty a nastavená hodnota. Jedním z typů hodnot je *integer* (i), který je použit pro číselné hodnoty, jako je rozlišení obrazovky, ale také pro hodnoty typu zapnuto/vypnuto, protože se počítá s možným rozšířením hodnot, kterých tyto parametry mohou nabývat. Dalším typem je *string* (s), který je použit pro adresářové cesty a názvy serverů. Posledním je typ *binary* (b), který je použit pouze pro uložení hashovaného hesla.



Obrázek 6: Příklad RDP souboru

2.4.3 Vzdálená pomoc

Výše zmíněné *Připojení ke vzdálené ploše* je používáno především ve firemním prostředí, v situacích, kdy je potřeba vzdáleně přistupovat k nějakému serveru, nebo pokud třeba zaměstnanec potřebuje přistupovat z domova ke svému pracovnímu počítači. Na rozdíl od toho se *Vzdálená pomoc systému Windows* (Remote Assistance) používá spíše mezi běžnými uživateli. V dnešní době, kdy jsou počítače dostupné opravdu každému, se stále více stává, že počítač vlastní i nezkušení uživatelé, kteří často potřebují pomoc nějakého odborníka nebo rodinného příslušníka při řešení problému na jejich počítači. Nejjednodušším řešením takových problémů je použití programu *Vzdálená pomoc systému Windows*. Uživatel, který potřebuje pomoci, tzv. nováček nebo začátečník, nasdílí svou pracovní plochu nějakému odborníkovi a může mu přímo ukázat, jaká je podstata problému. Oproti programu *Připojení ke vzdálené ploše* zde nedochází k odhlášení uživatele, který je fyzicky u počítače, ale oba sdílí stejnou obrazovku. Začátečník může také odborníkovi povolit sdílené interaktivní ovládání svého počítače. Pokud tak učiní, budou moci oba dva ovládat myš i klávesnici.

Na každém počítači, který podporuje vzdálenou pomoc, existuje speciální lokální uživatelský účet s názvem HelpAssistant, který má omezená oprávnění a je standardně nedostupný. Jakmile se naváže spojení vzdálené pomoci, je odborník připojen právě pomocí tohoto účtu, který má náhodně generované silné heslo [12].

2.4.3.1 Způsoby vytváření připojení

Vzdálená pomoc má dva základní typy provozu. Prvním z nich je vyžádaná vzdálená pomoc (Solicited RA). Tento typ provozu začíná u uživatele, který pošle odborníkovi žádost o pomoc (pozvánku). Vyžádaná vzdálená pomoc se dále dělí na podskupiny podle způsobu, kterým je pozvánka doručena expertovi.

- *Vyžádaná vzdálená pomoc přes e-mail*
  Vzdálená pomoc použije e-mailového klienta, který je v počítači nainstalovaný a vytvoří e-mail. Do přílohy e-mailu vloží soubor s pozvánkou [10].

- *Vyžádaná vzdálená pomoc přes Windows Messenger*
  Uživatelé Windows Messengeru mohou požádat kontakt o pomoc přímo v této aplikaci bez nutnosti spouštění programu *Vzdálená pomoc systému Windows*. Expertovi se ve Windows Messengeru objeví oznámení o žádosti, kterou může přijmout, nebo odmítnout. Tato metoda pozvání je nejvíce preferována [10].

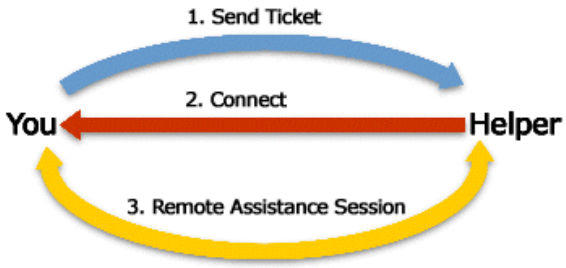
- *Vyžádaná vzdálená pomoc přes Easy Connect*
  Tato metoda je dostupná od Windows 7 a užívá Peer Name Resolution Protocol (PNRP), aby byl dostupný přímý P2P přenos pozvánky přes cloud. Uživatel musí sdělit expertovi heslo přes nějaký komunikační kanál, např. telefon. Expert si stáhne pozvánku z cloudu a vytvoří spojení mezi uživatelem a jeho pomocníkem.[11]

- *Vyžádaná vzdálená pomoc přenosem souboru*
  Tato možnost je použita, pokud začátečník i pomocník mají přístup ke stejné složce nebo chce uživatel využít jiný způsob doručení pozvánky. Uživatel vytvoří soubor pozvánky, který uloží na disk, a dále ho může jakýmkoli způsobem dopravit k odborníkovi.

Další možnost, jak vytvořit spojení vzdálené pomoci, se nazývá nevyžádaná vzdálená pomoc. Tento typ je typicky užíván ve firemním help desku. Podmínkou je, že jsou všichni uživatelé v doméně a odborník je doménový administrátor. Odborník musí zadat doménové jméno nebo IP adresu počítače, ke kterému se chce připojit, a provede připojení. V tomto typu provozu uživatel neposílá žádnou pozvánku.

2.4.3.2 Proces připojení

Vzdálená pomoc používá technologii terminálové služby, aby se mohl vzdálený uživatel připojit k počítači. Jak je vidět na obrázku č. 11, *Vzdálená pomoc systému Windows* používá jednoduchý, bezpečný proces pro navázání spojení mezi žadatelem o pomoc a pomocníkem.



Obrázek 7: Průběh navázání spojení programu Vzdálená pomoc systému Windows [13]

Před samotným zahájením vzdálené pomoci je nutné, aby měl uživatel tuto funkcionalitu v systému povolenou. Pokud není tato funkce v systému povolena a uživatel se i přesto pokusí pozvánku vytvořit, obdrží od aplikace *Vzdálená pomoc systému Windows* pouze oznámení, že vytváření pozvánky není povoleno. Před zahájením vzdálené pomoci je tedy nutné nejprve změnit nastavení ve vlastnostech systému *Systém → Vlastnosti systému* a na kartě *Vzdálený přístup* zatrhnout možnost *Povolit připojení vzdálené pomoci k tomuto počítači*. Toto nastavení je možné změnit také editací hodnoty klíče v systémových registrech, stejně jako tomu je u vzdálené plochy.

Proces připojení vzdálené pomoci začíná téměř vždy vytvořením pozvánky. Jedinou výjimkou je již zmíněná nevyžádaná vzdálená pomoc, kde se žádná pozvánka nevytváří, ale odborník se připojuje přímo. Pozvánku je možné vytvořit v grafickém uživatelském rozhraní aplikace, které se nachází v nabídce *Start → Všechny programy → Údržba* pod názvem *Vzdálená pomoc systému Windows*. Po spuštění si uživatel vybere, jakým způsobem bude pozvánka doručena odborníkovi. Poté se vygeneruje soubor s příponou .msrcincident, který obsahuje pozvánku. Zároveň se na obrazovce zobrazí heslo, které musí uživatel sdělit vzdálenému pomocníkovi, a aplikace začne naslouchat na specifickém portu a vyčkávat připojení. Port použitý pro vzdálenou pomoc se liší podle verze Windows. Pokud je na počítači vytvářejícím pozvánku Windows XP, pak je použit TCP port 3389. Pokud vytváří pozvánku počítač s novějším OS Windows, pak se dynamicky vybírá TCP/UDP port z rozsahu 49152–65535. Při použití nevyžádané vzdálené pomoci se používá TCP port 135.

Zkušenější uživatelé mohou namísto grafického uživatelského rozhraní použít příkazovou řádku. Totožného stavu lze totiž dosáhnout spuštěním spustitelného souboru *msra.exe* s parametry, jejichž kompletní popis se zobrazí provedením příkazu *msra.exe /?*

Na druhé straně, u vzdáleného počítače, musí uživatel spustit soubor s pozvánkou a zadat heslo, které obdržel od odesílatele pozvánky. Tím se naváže spojení a uživateli se zobrazí na obrazovce plocha vzdáleného počítače. Odborník nemusí pouze pasivně sledovat dění na ploše vzdáleného počítače, ale může požádat o řízení vzdáleného počítače. Od této chvíle může ovládat počítač se stejnými právy jako uživatel, který vytvořil pozvánku. Oba uživatelé nyní mohou ovládat kurzor myši a zadávat příkazy na klávesnici.

2.4.3.3 Vytváření logu

Vzdálená pomoc vytváří ke každému svému spuštění soubor s aktivitami, které s tímto spuštěním nějak souvisí. Tato funkce je standardně zapnuta. Každý soubor obsahuje jednotlivé záznamy opatřené časovou značkou. Tyto záznamy obsahují informace, které jsou přímo spjaté se vzdálenou pomocí, např. kdo zahájil spojení, zda bylo povoleno sdílené řízení atd. V žádném případě se zde nezaznamenávají informace o prováděných aktivitách na počítači.

Zaznamenávání aktivit vzdálené pomoci je určené převážně pro firemní účely. Některé instituce mají dokonce povinnost ze zákona zaznamenávat, kdo měl přístup do systému a v jakém čase. Jelikož vzdáleně připojený uživatel, který dostal povolení ke sdílenému řízení počítače, vystupuje ve firemní síti pod účtem žadatele o pomoc, tak je nutné tyto aktivity zaznamenávat.

Jak je vidět na obrázku č. 8, záznamy jsou ukládány do souboru ve formátu XML pod názvem, který odpovídá času vytvoření souboru.

```
<?xml version="1.0"?>
<SESSION>
    <INVITATION_OPENED TIME="13:19" DATE="5. ledna 2017" EVENT="Byla otevřena pozvánka funkce Vzdálená pomoc."/>
    <INCOMING_IP_ADDRESS TIME="13:26" DATE="5. ledna 2017">192.168.1.7</INCOMING_IP_ADDRESS>
    <CONNECTION_ESTABLISHED TIME="13:26" DATE="5. ledna 2017" EVENT="Bylo vytvořeno připojení pomocí funkce Vzdálená pomoc.">
    Michal Lejtnar
    </CONNECTION_ESTABLISHED>
    <CONNECTION_ENDED TIME="13:42" DATE="5. ledna 2017" EVENT="Připojení pomocí funkce Vzdálená pomoc bylo ukončeno."/>
    <INVITATION_CLOSED TIME="13:42" DATE="5. ledna 2017" EVENT="Pozvánka funkce Vzdálená pomoc byla zavřena."/>
</SESSION>
```

Obrázek 8: Příklad logovacího souboru programu Vzdálená pomoc systému Windows

2.4.3.4 Soubor .MsRcIncident

Pozvánka vzdálené pomoci (soubor s příponou .MsRcIncident) je dokument ve formátu XML, který obsahuje informace, které slouží vzdálenému počítači k připojení. Tyto informace jsou uloženy v tagu *UPLOADDATA*, který obsahuje následující atributy[14]:

- *USERNAME* – obsahuje jméno žadatele o pomoc.
- *LHTICKET* – zašifrovaný připojovací řetězec (connection string), kde je každý byte převeden na dvě hexadecimální čísla.
- *RCTICKET* – nezašifrovaný připojovací řetězec.
- *RCTICKETENCRYPTED* – hodnota, která indikuje, zda je pozvánka identifikovaná podle hodnoty RCTICKET zašifrovaná nebo ne. Hodnota 1 – šifrovaná, 0 – nešifrovaná.
- *DtStart* – čas, kdy byl vytvořen nový připojovací řetězec.
- *DtLength* – časový interval v minutách, po který je vygenerovaný připojovací řetězec validní. Pokud se vzdálený počítač nepřipojí v tomto intervalu, vzdálená pomoc se zavře na počítači, který vytvořil pozvánku.
- *PassStub* – šifrované heslo, které vytváří dodatečné zabezpečení.
- *L* – indikátor typu spojení. L=1 znamená modem, 0 znamená vysokorychlostní připojení.

2.4.3.5 Připojovací řetězec

Připojovací řetězec vzdálené pomoci obsahuje parametry RDP protokolu potřebné pro navázání spojení vzdálené pomoci. Jedná se o Unicode řetězec, který obsahuje následující informace oddělené čárkami [15].

<verze protokolu>,<typ protokolu>,<seznam adres>,<heslo>,<ID spojení>,<název spojení>,<heslo spojení>,<specifické parametry protokolu>

Verze protokolu – identifikuje verzi protokolu. Hodnota musí být nastavena na 65538.
Typ protokolu – identifikuje typ protokolu. Hodnota musí být nastavena na 1.

Seznam adres – identifikuje síťové adresy iniciátora vzdálené pomoci. Je to list dvojic IP adresa:port nebo jméno počítače:port.
Heslo – jedná se o heslo k účtu vzdálené pomoci. Musí být nastaveno na "*".
ID spojení – jedná se o jednoznačný identifikátor spojení.
Název spojení – název spojení vzdálené pomoci. Musí být nastaven na "*".
Heslo spojení – heslo spojení vzdálené pomoci. Musí být nastaveno na "*".
Specifické parametry protokolu – specifické parametry RDP protokolu.

Příklad připojovacího řetězce:

"65538,1,192.168.1.15:62284;90.179.81.209:10517,,GDA9Zc6DNJ1GAhgbcYnVGzfLWc3U2mr0H8+eV15dSxjhkZdAjH1k9SNartKht9fD,*,*,LzLxzyP03PcA8LXvMtSXZ9nvw8I"*

2.5 Existující řešení vzdáleného ovládání počítače

V současnosti existuje nepřeberné množství programů třetích stran pro vzdálenou správu počítačů. Některé programy umožňují pouze vzdálené sledování obrazovky, jiné nabízejí také vzdálené ovládání počítače. Většinou mají stejné vlastnosti jako nástroje poskytované přímo operačním systémem Windows. U všech těchto programů však platí, že nejsou standardně dostupné v operačním systému a musí být dodatečně doinstalovány. Instalace musí proběhnout na všech zařízeních, která budou chtít dané připojení používat. Nezřídka jsou tyto programy také placené. Jednodušší aplikace se dají používat zcela zdarma, ale propracovanější software je většinou nabízen za nějaký paušální nebo jednorázový poplatek.

Porovnání téměř všech dostupných programů je dostupné zde [16]. My si zmíníme pouze ty nejpoužívanější, jako např. TeamViewer, LogMeIn, VNC nebo Chrome Remote Desktop.

2.5.1 Chrome Remote Desktop

Chrome Remote Desktop je bezplatný program vytvořený společností Google a běží jako rozšíření webového prohlížeče Chrome. Aplikace užívá vlastní vytvořený protokol s názvem Chromoting a podporuje oba typy připojení, jak vzdálenou pomoc (s možností ovládání PC), tak připojení k vzdálené ploše.

Pokud chceme používat tento software, musíme se nejprve přihlásit k Google účtu. Po přihlášení si můžeme vybrat, zda budeme chtít používat vzdálenou pomoc (Chrome Remote Assistance), nebo vzdálenou plochu (Chrome Remote Desktop). Vybereme-li si vzdálenou pomoc, máme ještě na výběr, zda budeme sdílet svou obrazovku, nebo se k nějakému počítači připojíme. Pokud budeme chtít využít vzdálenou plochu, bude nutné nainstalovat Chrome Remote Desktop Host na cílovém počítači. Tento program umožní připojení pomocí vzdálené plochy.

Výhodou tohoto programu pro vzdálený přístup je jeho snadná instalace a možnost použití na různých operačních systémech. To je možné z toho důvodu, že celý program běží ve webovém prohlížeči. Nevýhodou může být fakt, že se nejedná o plnohodnotný program pro vzdálený přístup, ale o jednoduchou aplikaci na sdílení obrazovky, která neumí používat vzdálenou tiskárnu, nezvládá předávání běžných klávesových zkratek a nemá žádný komunikační nástroj mezi oběma stranami.

2.5.2 Virtual Network Computing

Virtual Network Computing (VNC) je grafický program, který umožňuje vzdálené připojení ke grafickému uživatelskému rozhraní pomocí počítačové sítě. VNC pracuje jako klient–server, kde server vytváří grafickou plochu v operační paměti počítače a komunikuje přes síť s klientem, který plochu zobrazuje uživateli (většinou na jiném počítači). Pro komunikaci se používá protokol RFB (Remote FrameBuffer), jehož cílem je minimalizovat objem přenášených dat mezi klientem a serverem a umožnit tak komunikaci i přes pomalejší datové linky (např. přes Internet)[17].

VNC je open source řešení vzdálené plochy, proto existuje velké množství různých aplikací založených na VNC a používajících protokol RFB. Jelikož mají všechny programy stejný základ, je možné použít jeden program jako server a jiný jako klienta. Navíc je většina programů zcela zdarma. Jedinou výjimkou jsou aplikace, které obsahují nějakou přidanou hodnotu, například v podobě podpory šifrované komunikace.

Ačkoliv má Virtual Network Computing spoustu výhod v podobě velkého výběru různých aplikací, ceny a multiplatformních provedení, jsou zde také podstatné nevýhody, které veškeré klady převyšují. Nejzásadnějším problémem tohoto systému je absence šifrování komunikace a velká hrozba napadení (např. Man In The Middle útok), protože tato slabina VNC je velmi dobře známá. Jediným řešením může být navázání relace skrz zabezpečený

tunel (SSH, VPN). Další nevýhodou je velké množství možností konfigurace a z toho plynoucí složité nastavení, které nemusí nezkušený uživatel zvládnout.

2.5.3 TeamViewer

TeamViewer je v současné době nejrozšířenější a nejvíce používaný software pro vzdálenou správu počítače. Uvádí se, že TeamViewer používá celosvětově asi 100 000 000 uživatelů. Tento program umožňuje ovládání vzdáleného počítače, sdílení plochy, on-line schůzky a webové konference.

Aplikace má jednoduché grafické rozhraní, které zobrazuje heslo a Id pro aktuální sezení. Aby byla zajištěna dostatečná bezpečnost, tak se mění hesla a Id při každém spuštění programu. Použití TeamVieweru je velmi jednoduché. Postačí, když zadáme Id vzdáleného počítače, následně vybereme účel, pro který se připojujeme na vzdálený počítač (např. prezentace, vzdálená podpora, sdílení souborů). Jako poslední zadáme heslo a zahájíme spojení.

TeamViewer je tolik používaný, protože poskytuje velké množství funkcí a výhod. Jednou z výhod je snadné ovládání, které zvládne opravdu každý uživatel bez zkušeností. Mezi další výhody patří kompatibilita s velkým množstvím operačních systémů a přenosných zařízení nebo velmi dobré zabezpečení. Nejpůsobivější vlastností je možnost vzdáleného připojení bez nutnosti instalace TeamVieweru na počítači. Tato možnost je užitečná v případě, kdy nějaký zákazník potřebuje pomoc od technika, ale nechce, nebo si nemůže program instalovat.

Ani tento software se neobejde bez nevýhod. Tou nejzásadnější je, že TeamViewer není zdarma pro komerční účely. Za možnost používat plnohodnotnou verzi v komerční sféře musí firma zaplatit jednorázový poplatek, který je poměrně vysoký. Vývojáři však mysleli i na případ, že by firma chtěla používat TeamViewer bez zakoupené licence. Pokud se program bude připojovat s mnoha různými zařízeními, pak začne být podezíravý a omezí přístup. Neplacená verze navíc neobsahuje takové množství funkcí jako verze placená.

3 Návrh řešení

Tato kapitola se zabývá návrhem aplikace pro vzdálenou správu počítačů prostřednictvím vlastní vytvořené peer-to-peer sítě. V této části nalezneme seznámení s použitým typem peer-to-peer sítě a motivaci k jejímu využití. Dále se zde seznámíme se způsobem, jakým jsou využity terminálové služby vzdálená plocha a vzdálená pomoc.

3.1 Použité technologie

Pro vývoj aplikace byl zvolen programovací jazyk C#. Jedná se o vysokoúrovňový objektově orientovaný jazyk, který byl vytvořen firmou Microsoft, a který vychází z programovacího jazyka Java a C++. Autor má s programovacím jazykem C# největší zkušenosti, proto byl vybrán pro vývoj aplikace. Navíc aplikace bude provozována výhradně na operačním systému Windows, vzhledem k tomu, že bude využívat terminálové služby OS Windows. To je další důvod, proč je vhodné použít programovací jazyk C#.

Jako vývojové prostředí bylo použito Microsoft Visual Studio 2015. Toto IDE má téměř nekonečné možnosti, což značí i symbol nekonečna, který má ve svém znaku. Jelikož je nejjednodušší vyvíjet nástroje pro vlastní produkt. Protože C# je výtvorem Microsoftu a celá aplikace poběží na OS Windows, pak je asi nejlepší volbou pro vývoj právě Visual Studio.

Další použitou technologií je .NET Framework. Je to platforma, která poskytuje nástroje a technologie, které jsou potřeba při vývoji síťových aplikací, distribuovaných webových služeb nebo webových aplikací. .NET Framework poskytuje nezbytný kompilační čas a čas běhu pro vytvoření a běh jakéhokoli jazyka, který je v souladu s Common Language Specification (CLS). Hlavní dvě komponenty .NET Frameworku jsou Common Language Runtime (CLR) a .NET Framework Class Library (FCL). CLR je běhové prostředí .NET Frameworku, které provádí veškerý spuštěný kód jako virtuální stroj. .NET Framework Class Library (FCL) je obrovská sbírka jazykově nezávislých a typově bezpečných opakovaně použitelných tříd. .NET Framework knihovny třídy (FCL) jsou uspořádány do logického seskupení podle jejich funkčnosti a využitelnosti do jednotlivých Namespace [18].

Pro ukládání dat byla vybrána relační databáze Microsoft SQL. Jedná se o databázový systém vytvořený Microsoftem. MS SQL Server je dostupný v několika různých verzích. Rozdíly mezi verzemi nejsou jen v nabízených funkcích, ale také v účelu jejich použití, např. Bussiness Inteligence (podpora pokročilých funkcí pro BI), Compact (určeno pro mobilní aplikace). Pro účely této práce byl zvolen MS SQL Server ve verzi Express. Jedná se o verzi,

která je volně ke stažení. Ačkoliv má určitá omezení, je dostačující pro menší aplikace, které nebudou mít velké množství dat.

Síťová komunikace mezi jednotlivými uzly bude zajištěna za pomoci socketů. Jedná se o sadu funkcí, které jsou určeny pro komunikaci po síti. .NET Framework obsahuje třídu Socket, která reprezentuje sockety a jejich funkce. Další možností je použití TCPClient pro klientskou část a TCPListener pro serverovou část. Jedná se o praktický wrapper nad třídou Socket. Při tvorbě jednoduché klient–server aplikace je jednodušší použít TCPClient/ TCPListener. Pro složitější aplikace se doporučuje použití přímo třídy Socket. Je to třída nižší úrovně než TCPClient nebo TCPListener, tím má blíže k Winsock2 a je rychlejší. Navíc třída Socket má rozsáhlejší dokumentaci a více dostupných příkladů.

3.2 Vymezení aplikace

Aplikace, která je předmětem této práce, nese pracovní název *Remote Helper*. Úkolem této aplikace je vzdálené ovládání počítače v peer-to-peer síti pomocí terminálových služeb operačního systému Windows. Celý systém lze rozdělit na tři části. První část se zabývá tvorbou a údržbou peer-to-peer sítě (připojování/odpojování uzlů, propagace informací v síti). Další je část, která využívá programu *Připojení ke vzdálené ploše*, který je dostupný ve Windows. Tato část se stará o přípravu RDP serveru a klienta, ale také o přípravu celého systému, aby bylo možné vytvořit spojení vzdálené plochy mezi jednotlivými uzly. Poslední část se zabývá přípravou prostředí pro druhou terminálovou službu, kterou je program *Vzdálená pomoc systému Windows*. Tato část má stejný úkol jako část předchozí. I tato část se stará o nastavení systému, aby bylo možné vytvořit spojení vzdálené pomoci.

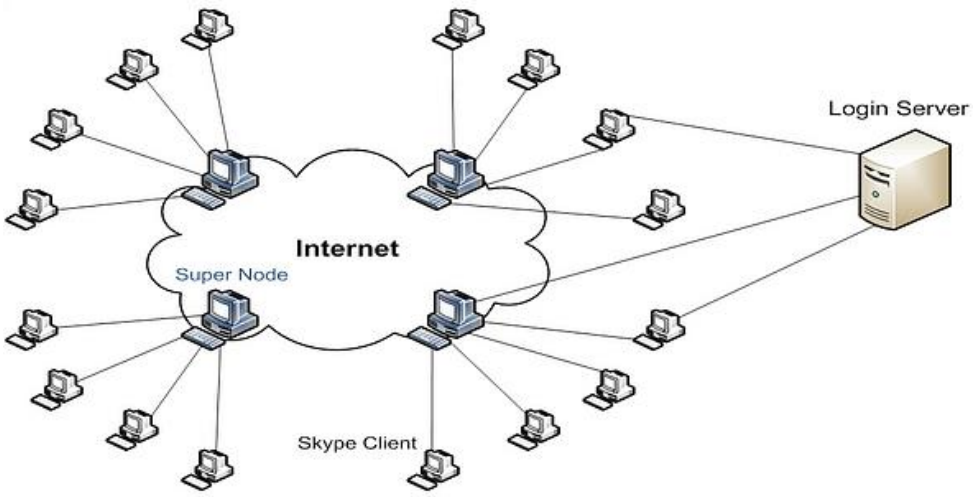
3.3 Návrh peer-to-peer sítě

Jak bylo uvedeno v teoretické části, existuje několik různých architektur peer-to-peer sítí. Po nastudování výhod a nevýhod jednotlivých typů P2P sítí byla zvolena nestrukturovaná hybridní peer-to-peer architektura. Jelikož v této síti nebudou uloženy žádné zdroje, které by bylo potřeba vyhledávat, pak není potřeba, aby byla síť strukturována. Naopak je velmi pravděpodobné, že bude neustále docházet k připojování a odpojování uzlů. To by mělo u strukturovaných sítí za následek velké množství režijních úkonů. Z toho důvodu byla vybrána nestrukturovaná síť. Jelikož hybridní peer-to-peer architektura kombinuje výhody centralizované a decentralizované sítě, tak byla vybrána právě tato architektura.

Dalším důvodem pro výběr hybridní peer-to-peer architektury je komunikační nástroj Skype, který funguje také na principu hybridní peer-to-peer sítě. Na tomto programu je možné vidět, že je námi vybraná architektura vhodná pro takové aplikace a bez problému funguje i s velkým počtem připojených uzlů.

3.3.1 Inspirace peer-to-peer sítí Skype

Skype je nejznámější a nejpoužívanější VoIP klient po celém světě. Celý systém sestává z jednotlivých klientských aplikací (uzlů) nainstalovaných na počítačích uživatelů, které dohromady vytváří jednu velkou peer-to-peer síť. Jako jeho předchůdce na sdílení souborů, Kazaa, i tento systém je tvořen nestrukturovanou, hybridní, překryvnou peer-to-peer sítí. V této síti se objevují dva typy uzlů. Jedná se o běžného klienta (ordinary host, skype client, SC) a super uzel (super node, SN). Super uzlem může být každý klient, který má veřejnou IP adresu, dostatečný výkon CPU, dostatečně velkou paměť a rychlé připojení k internetu. Takový uzel pak slouží běžným klientům jako přípojný bod do sítě. Každý běžný klient se musí připojit k nějakému super uzlu a autentizovat se na přihlašovacím serveru. Přihlašovací server slouží jako uložiště přihlašovacích jmen, hesel a tzv. buddy listů (seznam spřátelených kontaktů). Každý klient posílá své přihlašovací údaje přímo na webové API serveru pomocí HTTP dotazu. Server je jediným centrálním prvkem v této peer-to-peer síti a tím pádem také nejslabším místem [19]. Vztah mezi jednotlivými prvky peer-to-peer sítě můžeme vidět na obrázku číslo 9.



Obrázek 9: Skype peer-to-peer síť [20]

Podle provedené analýzy [19] má každá klientská aplikace tzv. host cache (HC). To je seznam IP adres a portů, které patří super uzlům, ke kterým se může připojit. Každý uzel má navíc přímo v sobě zakódováno 7 adres. Pokud se klient nepřipojí ani k jednomu super uzlu z HC, pokusí se připojit k jedné ze sedmi pevně zakódovaných adres. Po připojení k super uzlu se klient autentizuje na přihlašovacím serveru, který obsahuje jedinečné přihlašovací údaje pro celý systém. Až po autentizaci je možné spojení označit za úspěšné.

Aplikace, která je předmětem této práce, je inspirována právě architekturou peer-to-peer sítě Skype. Architektura P2P sítě použitá v této aplikaci se skládá ze tří typů uzlů. Jsou to Ordinary node, Super node a Central node. Funkci přihlašovacího serveru zde zastává Central node. Také proces přihlašování se liší. Běžné uzly se nepřipojují přímo na Central node, aby bylo ověřeno jejich jméno a heslo. Ověření probíhá přes Super node, ke kterému se uživatel připojil.

3.3.2 Ordinary node

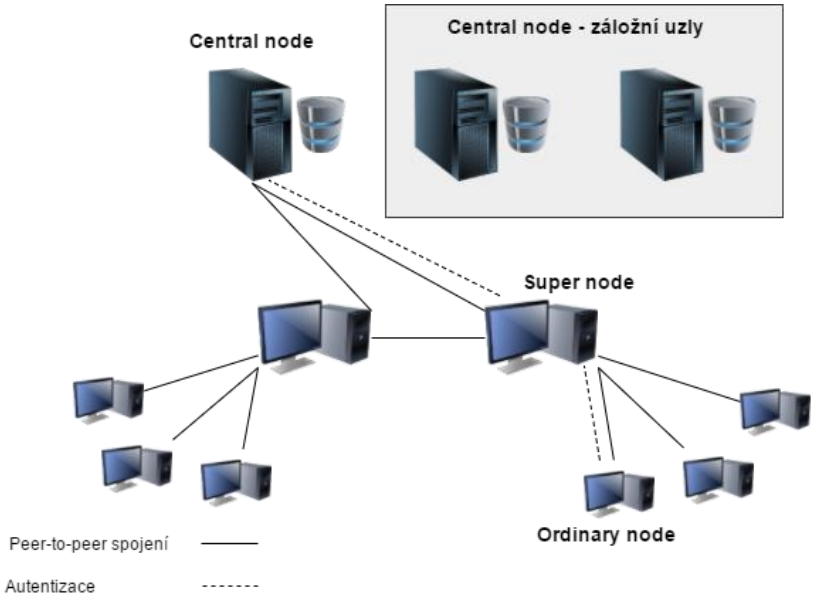
Základním typem uzlu je Ordinary node (ON) neboli běžný uzel. Tento uzel má nejomezenější vlastnosti a funkce. Jedná se o uzel určený pro koncového klienta, jehož počítač bude mít nízký výkon a bude omezen nepříliš rychlým připojením k Internetu. Běžný uzel má několik základních funkcionalit, mezi které patří připojení s přihlášením, připojení s registrací, odpojení, žádost o vzdálenou pomoc, připojení ke vzdálené ploše a možnost psaní textových zpráv s ostatními uživateli připojenými k síti. Každý uzel má u sebe uložený soubor, který obsahuje záznamy o Super node uzlech, jejich IP adresách a portech. Soubor slouží pro výběr uzlu, ke kterému se Ordinary node připojí, a při každém připojení se soubor aktualizuje.

3.3.3 Super Node

Super node (SN) je dalším typem uzlů v peer-to-peer síti. Tento uzel je v podstatě běžný uzel, obohacený o rozšiřující funkce. Super node slouží běžným uzlům jako vstupní bod do celé peer-to-peer sítě. Z toho důvodu je potřeba, aby měl tento typ uzlu vždy veřejně dostupnou IP adresu. Po připojení nového uzlu zajišťuje ověření jeho přihlašovacích údajů u Central nodu, protože jen Super node uzly se k němu mohou připojit. Tento uzel neslouží běžným uzlům jen k připojení, ale následně zprostředkovává komunikaci mezi jednotlivými běžnými uzly, které nemohou být spojené přímo z toho důvodu, že nemají veřejnou IP adresu. Slouží tedy jako takový proxy server pro přeposílání řídicích příkazů, běžných zpráv a komunikace terminálových služeb.

3.3.4 Central Node

Central node (CN) je posledním typem uzlu používaným v tomto systému. Jak můžeme vidět na obrázku číslo 10, jedná se o jediný centrální prvek celé sítě. Tento uzel slouží jako přípojný bod pro všechny Super node uzly, které se připojují do sítě, a může se v celé síti vyskytovat pouze jednou. Pokud je z nějakého důvodu nedostupný, Super node uzly se mohou připojit k jednomu z jeho alternativních náhradníků. Central node má jako jediný veškeré informace o celé síti a všech připojených uzlech. Tyto informace si ukládá do databáze, která slouží hlavně pro uchovávání a ověřování uživatelských jmen a hesel. Autentizace uživatelů při připojení je jeho hlavní funkcí.



Obrázek 10: Schéma peer-to-peer sítě aplikace Remote helper

3.4 Návrh databáze

Databáze obsahuje pouze jedinou tabulku s názvem *dbo.Users*, která je určena pro ukládání záznamů o jednotlivých uživatelích. Tato tabulka obsahuje 9 sloupců:

- Id – slouží jako jedinečný identifikátor.
- Nickname – přezdívka uživatele, která se bude zobrazovat ostatním uživatelům.
- Password – uživatelské heslo.
- E-mail – slouží jako jedinečné přihlašovací jméno.
- NodeId – jedinečný identifikátor uzlu ve formě GUID (Globally Unique Identifier).
- NodeType – informace o typu uzlu při posledním přihlášení (Ordinary, Super).
- Status – informace, zda je aktuálně přihlášený, nebo není.

- Address – IP adresa, se kterou je připojený nebo byl naposledy připojený.
- Port – použitý TCP port.

Na následujícím obrázku s číslem 11 můžeme vidět strukturu tabulky *dbo.Users* a nastavení jednotlivých atributů.

	Column Name	Data Type	Allow Nulls
🔑	Id	int	☐
	Nickname	nvarchar(50)	☐
	Password	nvarchar(50)	☐
	Email	nvarchar(50)	☐
	NodeId	uniqueidentifier	☐
	NodeType	int	☐
	Status	int	☐
	Address	nvarchar(15)	☑
	Port	int	☑

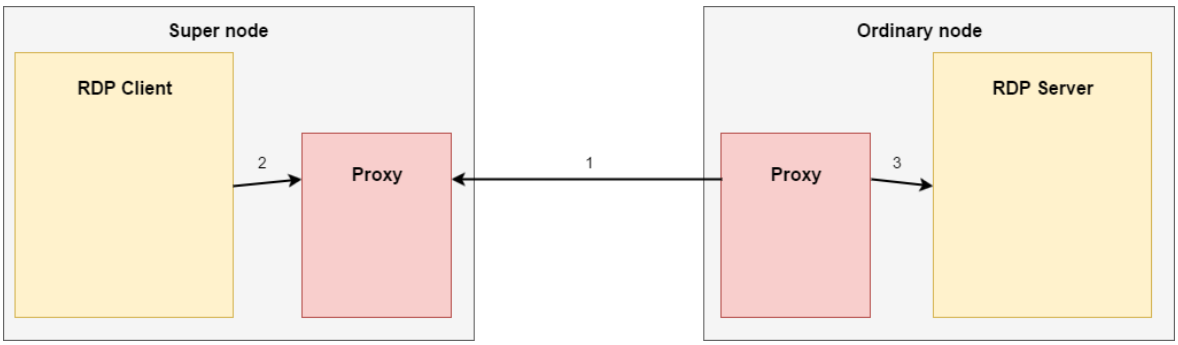
Obrázek 11: Databázová tabulka dbo.Users

3.5 Vzdálená plocha

Pro vzdálené ovládání počítače je vybrán program *Připojení ke vzdálené ploše*. Tento typ terminálové služby se výborně hodí pro případy, kdy jsme my iniciátory spojení.

Připojení ke vzdálenému počítači a jeho ovládání pomocí programu *Připojení ke vzdálené ploše* je velmi jednoduché. Stačí pouze zadat IP adresu vzdáleného počítače nebo jeho doménové jméno, pokud jsme v doméně, po vyžádání zadat přihlašovací údaje a připojení je dokončeno. Problém nastává, pokud zařízení není v lokální síti, ale je dostupné někde v Internetu a zároveň nemá veřejnou IP adresu. V případě námi navržené peer-to-peer sítě je bezproblémové spojení v případě, že se jedná o spojení mezi dvěma Super node uzly nebo mezi běžným uzlem a super uzlem, kdy je běžný uzel původcem spojení a cílem je super uzel. Když se potřebujeme připojit programem *Připojení ke vzdálené ploše* k nějakému Ordinary node uzlu, který nemá veřejnou IP adresu, tak máme problém. V takovém případě se nám nepodaří navázat připojení z venkovní sítě (Internetu) ke koncovému zařízení, které je schované za NATem.

Řešením tohoto problému je vytvořit nejprve spojení z Ordinary node uzlu na super uzel a po takto vytvořeném spojení pak posílat data, která si vyměňují terminálový server a klient v podobě vzdálené plochy. Není však možné, aby se RDP server, který naslouchá na běžném uzlu, připojil na klienta na super uzlu. Z toho důvodu je nutné přidat nějakého prostředníka mezi RDP klienta a server. V našem případě je to proxy, které můžeme vidět na obrázku číslo 12.

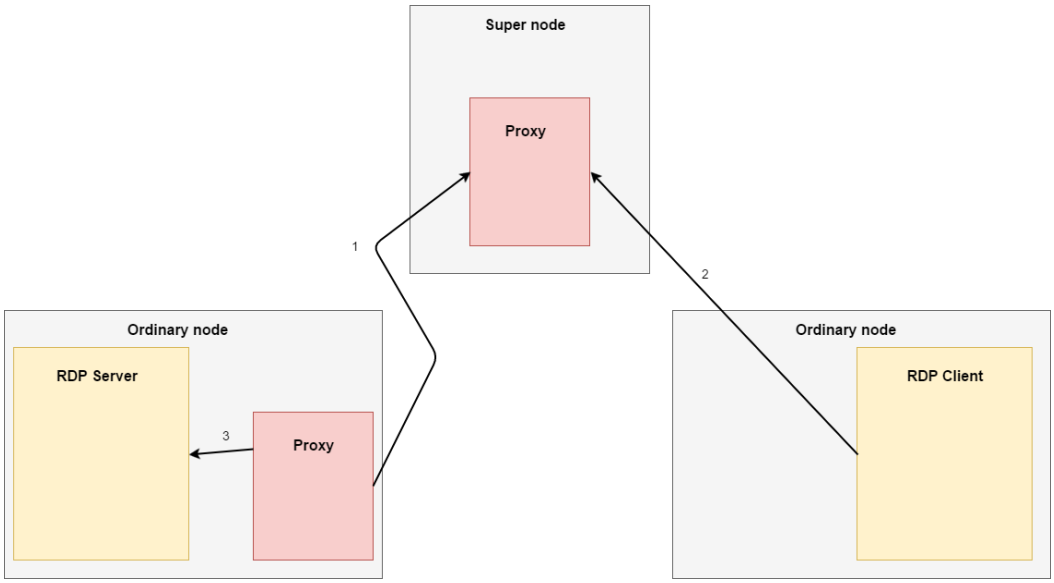


Obrázek 12: Vzdálená plocha - proces připojení z SN na ON

Na obrázku č. 12 můžeme vidět princip připojení programu *Připojení ke vzdálené ploše* k uzlu, který se nachází v nějaké lokální síti za NATem. Nejprve se tedy vytvoří spojení z proxy serveru na běžném uzlu na proxy super uzlu (1). Když je toto spojení vytvořeno, může se RDP Client připojit k lokálnímu proxy serveru (2). Ten přepošle veškerá data přes vytvořené spojení na proxy server běžného uzlu (1), který je předá RDP serveru (3).

O něco složitější je spojení v situaci, kdy je potřeba, aby se jeden Ordinary node připojil vzdálenou plochou k jinému Ordinary node uzlu. Ani v tomto případě není možné navázat spojení přímo mezi uzly, protože ani jeden uzel nemá veřejnou IP adresu. Proto musí k navázání spojení využít nějaký Super node, který jim bude zprostředkovávat komunikaci. Jako prostředník bude vždy vybrán Super node, ke kterému je připojen iniciátor spojení. Jako v předchozí situaci musí mít zúčastněné uzly spuštěný proxy server. Proxy běžící na Super node uzlu je odlišné od toho, co běží na obyčejném uzlu. K jednomu Super node uzlu může být připojeno více Ordinary node uzlů současně, a čím více uzlů je připojeno, tím větší je šance, že bude chtít více uzlů současně zprostředkovat připojení ke vzdálené ploše. Z toho

důvodu musí být proxy server schopný současně přeposílat data vzdálené plochy mezi více páry připojených uzlů.



Obrázek 13: Vzdálená plocha - proces připojení z ON na ON

Schéma tohoto složitějšího případu spojení můžeme vidět na obrázku č. 13. Zde můžeme vidět, že prvním krokem pro navázání spojení je vytvoření spoje od proxy na Ordinary node uzlu směrem k super uzlu (1). Dalším krokem je připojení RDP klienta k proxy serveru na Super node uzlu (2). Posledním krokem je vytvoření spojení z lokálního proxy směrem k RDP serveru na cílovém počítači (3).

3.6 Vzdálená pomoc

Vzdálená pomoc je druhý způsob, kterým můžeme sledovat, nebo dokonce ovládat vzdálený počítač. Tento typ vzdáleného ovládání využívá další terminálovou službu dostupnou na operačním systému Windows. Jedná se o program *Vzdálená pomoc systému Windows*. V této aplikaci bude vzdálená pomoc použita pro situace, kdy nějaký uživatel bude potřebovat, aby se někdo jiný připojil k jeho počítači. Uživatel si bude moci vybrat nějakého uživatele, který je dostupný v síti, a požádat ho o pomoc. Tento vzdálený pomocník může žádost přijmout a připojit se, nebo může žádost zamítnout. V pozadí této žádosti probíhá vytváření pozvánky (soubor s příponou .msrcincident), kterou potřebuje program *Vzdálená*

pomoc systému Windows pro navázání spojení. Vytvořená pozvánka je následně poslána vybranému uživateli, který se s její pomocí dokáže připojit k cílovému počítači.

Jako v případě vzdálené plochy i zde se vyskytuje problém s vytvořením připojení vzdálené pomoci u uzlů, které nemají veřejnou IP adresu. K řešení problému je opět využito proxy serverů, které zajistí, že se uzly úspěšně spojily. V procesu připojení je jeden významný rozdíl mezi vzdálenou plochou a vzdálenou pomocí. U vzdálené plochy je iniciátorem spojení klient (RDP Client), který se připojuje k serveru. U vzdálené pomoci však iniciátor spojení vygeneruje pozvánku a pak čeká na připojení. Tím v podstatě zastává funkci serveru. Příjemce pozvánky pak zastává funkci klienta a připojuje se k serveru.

4 Implementace

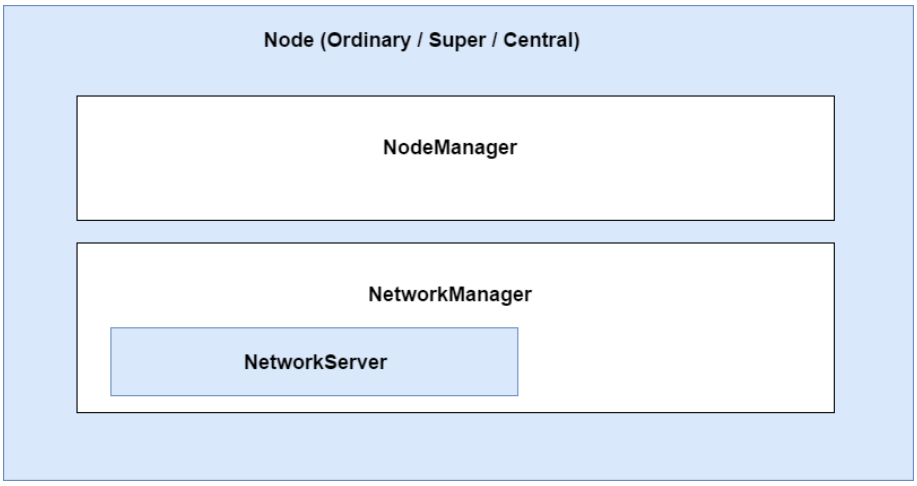
Tato kapitola popisuje implementaci navržené aplikace. Zahrnuje popis struktury celého řešení, popis jednotlivých částí aplikace a jejich funkcionalit. Dále popisuje, jakým způsobem je do praxe uvedena navržená peer-to-peer síť a ostatní části aplikace určené pro vzdálené ovládání počítače.

4.1 Struktura aplikace

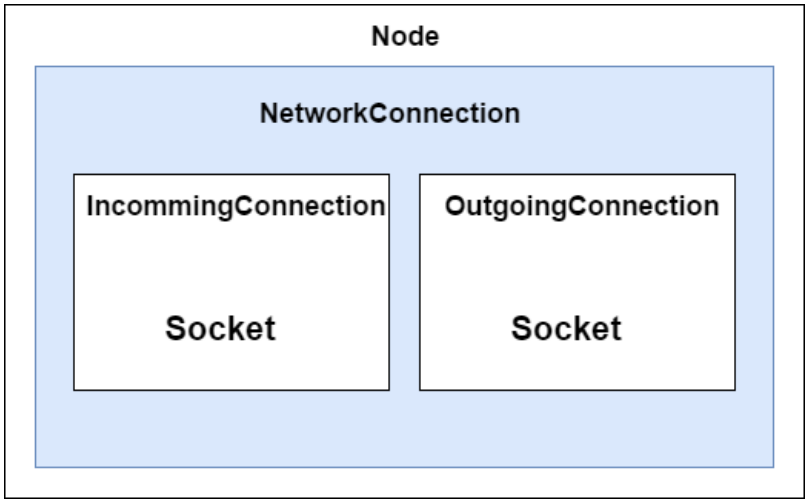
Celá aplikace *Remote helper* je rozdělena do čtyř samostatných projektů, kterými jsou *Network*, *Proxy*, *RemoteAssistance* a *RemoteDesktop*. Každý z projektů řeší určitou část funkcionality aplikace.

4.1.1 Projekt Network

První projekt s názvem *Network* je implementací navržené peer-to-peer sítě. Jádrem celé aplikace je třída *MainNode*. Tato třída zajišťuje vytvoření jednoho uzlu, který se připojí do sítě. Uzel reprezentovaný touto třídou je rozdělen do několika vrstev, jak můžeme vidět na obrázku číslo 14. Nejvyšší vrstva je *NodeManager*, tvořená třídou stejného názvu, která se stará o připojené uzly. Pod ní se nachází vrstva *NetworkManager*, která je taktéž tvořená třídou, která má totožný název. Tato vrstva se stará o jednotlivá spojení daného uzlu. Dále obsahuje *NetworkServer*. Jedná se o třídu, která naslouchá na vybraném portu a vyčkává na připojení nějakého socketu. Na obrázku číslo 15 můžeme vidět, z čeho se skládá objekt připojeného uzlu. Každý objekt typu *Node* reprezentuje konkrétní připojený uzel. Tento objekt má vytvořená dvě spojení *IncommingConnection*, *OutgoingConnection*. Tato dvě spojení jsou vytvořená pomocí socketu. Jedno slouží pro posílání odchozích zpráv a druhé pro příjem zpráv.



Obrázek 14: Zobrazení úrovní uzlu



Obrázek 15: Struktura objektu typu Node

4.1.1.1 Popis tříd projektu Network

MainNode – Třída, která je jádrem celé aplikace. Při spuštění aplikace se vytvoří instance této třídy. Tato instance představuje jeden celý uzel v peer-to-peer síti. Vytvořený uzel může být *Ordinary Node*, *Super Node* nebo *Central Node*, podle výběru uživatele. Třída také slouží pro spuštění dalších částí aplikace, které jsou potřebné při vzdáleném ovládání. Jedná se konkrétně o spuštění vzdálené pomoci, vzdálené plochy nebo proxy serveru, který pomáhá použitým terminálovým službám se vzájemným propojením.

NodeManager – Třída, která spravuje seznam všech připojených uzlů a vše kolem nich. Stará se o dvě kolekce uzlů. První je kolekce s názvem *UncompleteNodes*, kde jsou uloženy uzly, které jsou připojené, ale zatím neautentizované. Po korektní autentizaci je uzel přesunut do

kolekce *AllNodes*, se kterými už může aplikace dále pracovat. Dalším úkolem této třídy je zpracování požadavků, které vyvstaly na základě přijatých zpráv.

NetworkManager – Třída, která spravuje veškerá navázaná spojení ve formě socketu. Reaguje na události vyvolané *NetworkServerem* tím, že páruje vytvořené sockety (*IncommingConnection, OutgoingConnection*), které patří ke stejnému uzlu, a z tohoto uzlu vytváří objekt *NetworkConnection*, který představuje připojení ke konkrétnímu uzlu.

MessageManager – Třída, která spravuje frontu všech doručených zpráv typu *IMessage*. Každá doručená zpráva je zařazena do fronty, ze které jsou zprávy postupně odebírány a zpracovány metodou *DoProcess*, která běží v samostatném vlákně. Každá zpráva je odebrána z fronty a podle typu zprávy je zavolána konkrétní metoda z třídy *NodeManager*, která provede požadované úkony.

NetworkServer – Třída, která slouží k příjmu žádostí o TCP spojení ve formě socketu. Hlavní částí třídy je metoda *WaitForConnection*, která je spuštěna v samostatném vlákně a po celou dobu, co je spuštěna aplikace, naslouchá na vybraném TCP portu. Při příjmu každého nového spojení je vyvolána událost, kterou obslouží *NetworkManager*.

IONetworkHelper – Jedná se o pomocnou třídu, která pomáhá objektům typu *NetworkConnection* odesílat a přijímat zprávy. Každý vytvořený *NetworkConnection* obsahuje vlastní instanci této pomocné třídy, která pak zajišťuje odesílání zpráv přes socket *OutgoingConnection*. Pro příjem slouží metoda *StartReceive*, která běží v samostatném vláknu a po celou dobu přijímá zprávy doručené přes *IncommingConnection* socket.

DatabaseConnector – Třída, jejímž úkolem je komunikace s databází. Zajišťuje připojení aplikace k databázi, provádí kontrolu přihlašovacích údajů, registraci atd.

GuiHandler – Tato třída představuje prostředníka pro ovládání grafického uživatelského rozhraní z jednotlivých vláken. Obsahuje statické delegáty pro volání metod z hlavního formulářového okna.

Globals – Třída, která obsahuje pouze několik statických proměnných, které jsou potřeba skrz celou aplikaci.

IOXmlHelper – Pomocná třída, která vytváří soubor ve formátu XML, který obsahuje seznam Super node uzlů, které jsou dostupné v síti.

Contact – Třída představující objekt *Contact*, který sdružuje přihlašovací údaje uživatele.

INode – Jedná se o rozhraní, které je v aplikaci použito jako zástupný datový typ pro objekty *Node* a *AccessibleNode*.

Node – Třída, která představuje uzel, který je přímo připojený k našemu uzlu. Každá instance této třídy obsahuje informace o daném uzlu, např. IP adresu, port, atd. Tato třída implementuje rozhraní *INode*.

AccessibleNode – Třída reprezentující uzel, který je dostupný přes jiný přímo připojený uzel. Stejně jako třída *Node* i tato třída implementuje rozhraní *INode*. Hlavním rozdílem mezi těmito dvěma třídami je implementace metody *Send*. Metoda *Send* z třídy *Node* posílá zprávy přímo na IP adresu daného uzlu. Metoda *Send* z třídy *AccessibleNode* však posílá zprávy na IP adresu prostředníka, přes kterého je uzel dostupný.

TNode – Pomocná třída, která obsahuje informace o uzlu a dále je rozšířena o přihlašovací údaje uživatele dostupného z konkrétního uzlu.

LoginForm – Třída rozšířená třídou *Form*, slouží pro zobrazení přihlašovacího formuláře.

MainForm – Třída rozšířená třídou *Form*, která slouží pro zobrazení hlavního formuláře aplikace.

IMessage – Jedná se o rozhraní, které v aplikaci slouží jako zástupný datový typ pro všechny objekty představující zprávy.

BaseMessage – Základní třída, obsahující společné vlastnosti všech zpráv. Tato třída je děděna všemi třídami představujícími zprávy.

Dále tento projekt obsahuje několik tříd, které představují jednotlivé typy zpráv posílaných mezi uzly. Příkladem mohou být třídy *RegistrationMessage*, *ConnectMessage*, *BroadcastNodesMessage*, *DisconnectMessage*, atd.

4.1.2 Projekt Proxy

Část aplikace, která je implementována v projektu s názvem *Proxy*, zahrnuje třídy potřebné pro vytvoření proxy serveru na koncových uzlech (Ordinary node) i na uzlech zprostředkovávajících přeposílání informací (Super node), které posílají použité terminálové služby.

4.1.2.1 Popis tříd projektu Proxy

LocalProxy – Třída, jejíž instance vytváří proxy server na jednotlivých koncových uzlech. Každé lokální proxy má dva sockety. Jeden zajišťuje spojení s použitým programem vzdálené správy a druhý zajišťuje spojení s jiným uzlem. Podle toho, na jakém typu uzlu je lokální proxy spuštěné a kdo byl iniciátorem spojení, používáme 4 typy lokálního proxy serveru.

RemoteProxy – Třída, která zajišťuje přeposílání dat mezi programy vzdálené správy, které běží na dvou koncových uzlech. Pro každé spojení dvou koncových uzlů je vytvořen objekt typu *ConnectionBridge*.

ConnectionBridge – Třída, jejíž instance představuje spojení dvou konkrétních koncových uzlů. Tento objekt zajišťuje přenos dat mezi oběma uzly v obou směrech.

4.1.3 Projekt RemoteDesktop

Tento projekt sdružuje vše, co se týká programu pro vzdálenou správu počítače s názvem *Připojení ke vzdálené ploše*. Stará se o aktivity jako spuštění a ukončení programu pro vzdálené ovládání počítače.

4.1.3.1 Popis tříd projektu RemoteDesktop

RemoteDesktopManager – Jako jediná třída tohoto projektu má na starost všechny úkony, které je potřeba provést pro korektní spuštění programu *Připojení ke vzdálené ploše.* Nejprve je nutné vytvořit RDP file s požadovaným natavením IP adresy a portu. Poté může dojít ke spuštění procesu, pod kterým poběží program *Připojení ke vzdálené ploše*.

4.1.4 Projekt RemoteAssistance

Projekt s názvem *RemoteAssistance* se zabývá veškerými úkony, které souvisí s nastavením a spuštěním programu *Vzdálená pomoc systému Windows*. Jednotlivé třídy v projektu mají za úkol vytvořit pozvánku a následně spustit samotný program.

4.1.4.1 Popis tříd projektu RemoteAssistance

RemoteAssistanceManager – Třída zajišťující spuštění programu *Vzdálená pomoc systému Windows*. Jelikož je potřeba spustit program na jednom uzlu jako iniciátor spojení a na druhém jako vzdálený pomocník, tak má tato třída různé metody pro odlišné typy spuštění.

InvitationParser – Jedná se o pomocnou třídu, která má za úkol parsovat soubor pozvánky. Pro naši potřebu je nutné ze souboru určité části odstranit a samozřejmě změnit IP adresu a port v případě, že není možné navázat přímé spojení mezi programem (*Vzdálená pomoc systému Windows*) na obou koncových uzlech.

4.2 Vytvoření uzlu

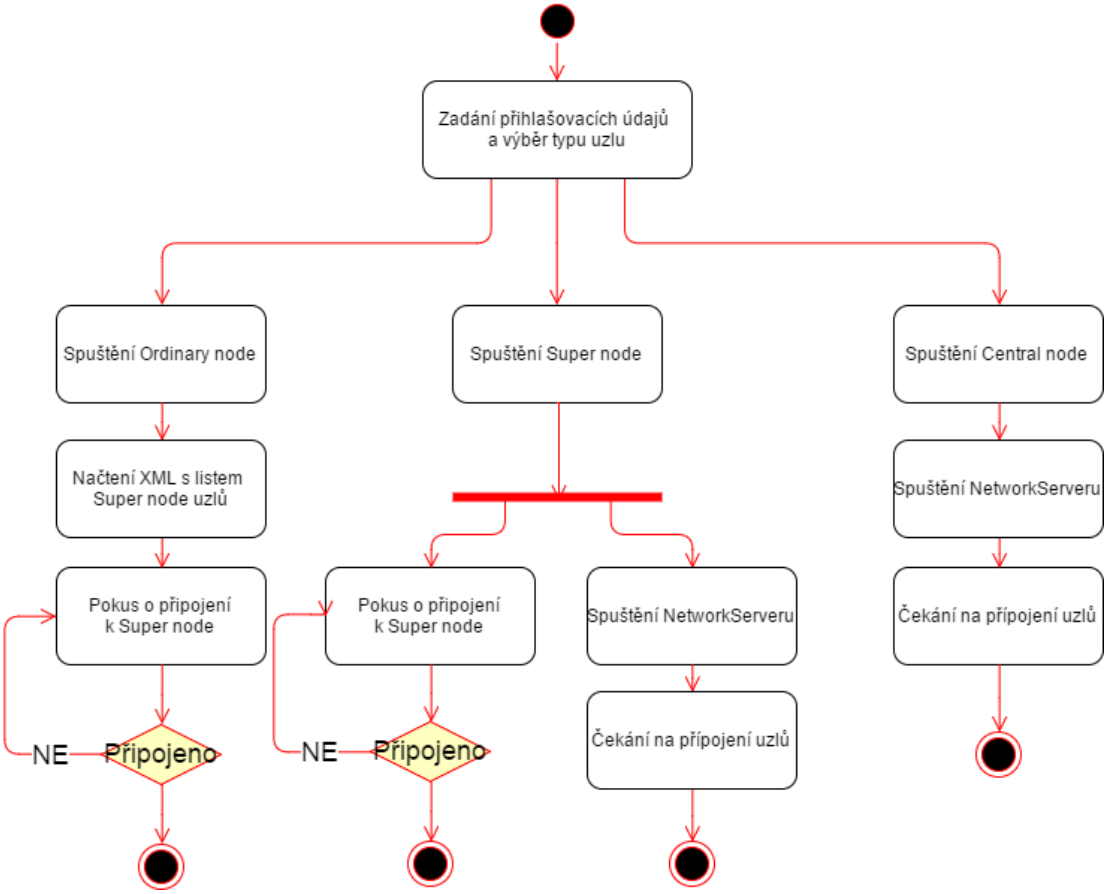
Proces vytvoření nového uzlu začíná spuštěním aplikace a zadáním přihlašovacích údajů v úvodním formuláři (*LoginForm*). Dále se vytvoří instance třídy *MainNode*, kde se program větví podle toho, jaký typ uzlu uživatel vybral.

Zvolil-li si uživatel jako typ uzlu Central node, pak se zavolá metoda *StartListenServer* ze třídy *NetworkManager*. V této metodě je zavolána metoda *StartListening* z třídy *NetworkServer*, která vytvoří nové vlákno, ve kterém bude po celou dobu chodu aplikace poslouchat na zadaném portu a vyčkávat na připojení uzlů. Standardně naslouchá na portu 11000, avšak tento port může být změněn v souboru *RemoteHelper.exe.config*.

Pokud uživatel zvolí typ uzlu Super node, je proces vytvoření uzlu o něco složitější. V první řadě dojde ke spuštění *NetworkServeru*, stejně jako v předchozím případě, a uzel bude vyčkávat na připojení jiných uzlů. Zároveň se aplikace pokusí o připojení k Central node uzlu. Jeho IP adresu a port je možné nastavit v konfiguračním souboru *RemoteHelper.exe.config*.

Poslední možností, kterou může uživatel vybrat, je typ uzlu Ordinary node. U tohoto typu uzlu nedochází ke spuštění *NetworkServeru*. Předpokládáme, že tento typ uzlu bude většinou spouštěn z počítače, který nemá veřejnou IP adresu, a proto je naprosto bezpředmětné, aby uzel naslouchal a očekával připojení jiného uzlu. Místo toho dojde k načtení XML souboru se seznamem dostupných Super node uzlů. Aplikace pak tento seznam cyklicky prochází a pokouší se připojit k některému z nich, dokud nedojde k připojení.

Jak celý proces vytváření nového uzlu probíhá, můžeme názorně vidět v UML diagramu aktivit na obrázku číslo 16.



Obrázek 16: UML diagram - proces vytvoření nového uzlu

4.3 Proces připojení

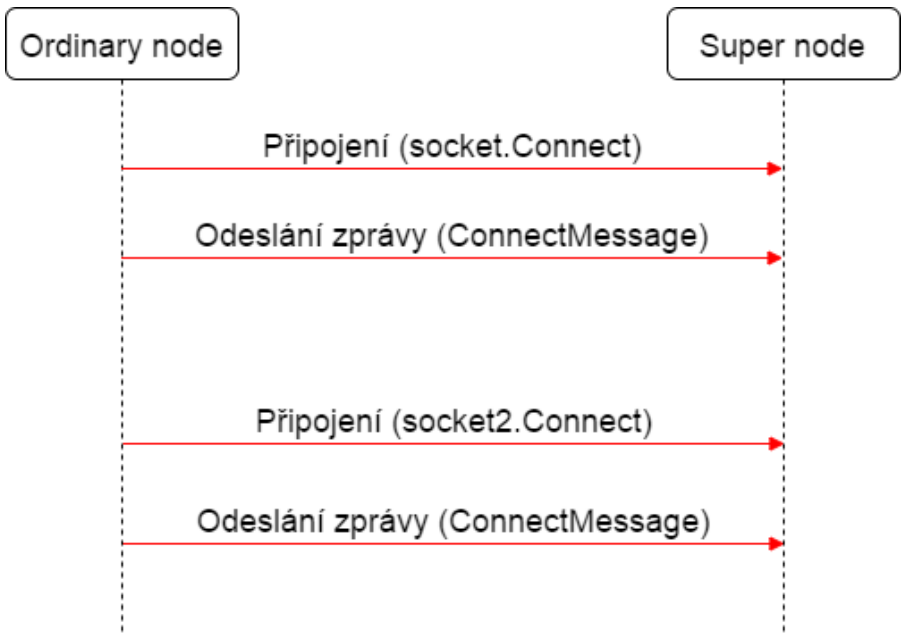
Aby mohl být proces připojení úspěšný, musí v síti existovat minimálně jeden uzel, který má spuštěné naslouchání na konkrétním portu (standardně 11000) a očekává žádost o připojení. Jedná se o metodu třídy *NetworkServer*. Pro připojení Super node uzlu musí být aktivní Central node, pokud připojujeme Ordinary node, musí být aktivní alespoň jeden Super node, ke kterému se připojíme. V následující části si vysvětlíme proces připojení mezi Ordinary node uzlem a Super node uzlem.

Připojení začíná vždy na straně ON uzlu. Ihned po vytvoření nového uzlu se zavolá metoda *ConnectTo* z třídy *NetworkManager*. Tato metoda vytvoří postupně dvě spojení se vzdáleným uzlem pomocí socketu. Pokud byl socket úspěšně připojen, pošle ON uzel zprávu typu *ConnectMessage*, ve které posílá informace o sobě a o typu spojení, které představuje daný socket. Každé spojení dvou uzlů se skládá ze dvou socketů. Vzhledem k tomu, že Ordinary node uzly nemají většinou veřejnou IP adresu (NAT problém), tak musí být oba sockety připojovány ve směru od ON k SN (obr. 17). Informace o typu spojení, která je

posílána ve zprávě *ConnectMessage*, pomáhá SN uzlu, aby věděl, přes který socket má odesílat data a na kterém socketu má očekávat příchozí komunikaci. Jakmile jsou obě spojení úspěšně navázána, je z nich vytvořen objekt *NetworkConnection*, který je ve třídě *NodeManager* přidělen instanci uzlu *Node* a přiřazen do kolekce připojených uzlů.

Na straně Super node uzlu proces začíná tím, že socket ve třídě *NetworkServer* naslouchá na portu, kde zaznamená žádost o připojení od nového uzlu. Tato událost je dále zpracována ve třídě *NetworkManager*. Na socketu se spustí blokovací metoda *Receive*, která čeká na příchod nějakých dat, která v našem případě budou představovat zprávu *ConnectMessage*. Podle informací získaných z doručené zprávy zjistí, jaký typ spojení bude tento socket představovat. Poté dojde k vytvoření instance třídy *NetworkConnection*, kterému je přiřazen socket. Po připojení druhého socketu a rozpoznání typu spojení je socket přiřazen do již vytvořeného objektu typu *NetworkConnection*. Každá zpráva *ConnectMessage* obsahuje také jednoznačný identifikátor spojení, díky kterému jsou oba sockety přiřazeny do stejného objektu. V okamžiku, kdy je objekt třídy *NetworkConnection* kompletní, je předán vyšší vrstvě aplikace (*NodeManager*), kde je vytvořen nový uzel a je mu přiřazeno zmiňované spojení.

Pokud tento proces proběhne bez problémů, tak je tento uzel zařazen do seznamu uzlů s názvem *UncompleteNodes*. V tomto seznamu zůstane, dokud nedojde k úspěšnému přihlášení, nebo k odpojení uzlu.



Obrázek 17: Diagram procesu připojení

4.4 Proces přihlášení

Pokud proběhne připojení uzlu úspěšně, musí se nový uzel autentizovat, aby mohl v síti dále komunikovat s ostatními uzly. Autentizace se provádí na základě dvou údajů, kterými jsou e-mailová adresa a heslo. Pro přenos těchto údajů od Super node uzlu nebo Ordinary node uzlu k Central node uzlu se používá typ zprávy s názvem *IdentificationMessage*. Spolu s přihlašovacími údaji se současně posílají také informace o novém uzlu, tj. IP adresa, port a typ uzlu.

Je-li nově připojeným uzlem Super node, pak se tato zpráva posílá přímo Central node uzlu, který ji dále zpracovává. Zpracování spočívá v kontrole zadaných údajů oproti údajům v databázi. K této kontrole slouží metoda *CheckUser* ze třídy *DatabaseConnector*. Tato metoda provede ověření e-mailové adresy a hesla pomocí metody *Verification*. Pokud jsou zadané údaje správné, provede se aktualizace informací o uzlu v databázi a je vrácen objekt typu *TNode*, který obsahuje aktualizované informace o uzlu a přidělené Id uzlu, což je identifikátor GUID. Připojený uzel je zařazen do seznamu korektně připojených uzlů a informace vrácené metodou *CheckUser* jsou následně odeslány zpět na nově připojený uzel ve zprávě typu *IdentificationReplyMessage.* Na jeho straně se provede aktualizace informací (jméno, identifikátor uzlu).

Jakmile Central node uzel odešle odpověď o úspěšném přihlášení, vytvoří seznam připojených Super node uzlů a odešle je novému uzlu. Ten si tímto seznamem aktualizuje záznamy v XML souboru, kde se uchovávají informace o dostupných Super node uzlech, jejich IP adresách a použitých portech. Může se zdát, že nemá význam posílat tento soubor uzlu Super node, ale může nastat situace, kdy se z toho samého počítače spustí aplikace s volbou Ordinary node uzlu a tyto aktualizované záznamy budou potřeba.

Přihlášení Ordinary node uzlu probíhá stejným způsobem. Rozdíl oproti předchozímu procesu je v tom, že Ordinary node posílá zprávu *IdentificationMessage* Super node uzlu, ke kterému je připojen, a ten posílá zprávu dál na Central node. U odpovědi je princip stejný.

Uzel uživatele, který zadá špatné přihlašovací údaje, není zařazen mezi korektně připojené uzly a informace o jeho připojení není publikována ostatním uzlům v síti. Namísto toho je znovu požádán o zadání přihlašovacích údajů. To se děje opakovaně, dokud nezadá správné údaje nebo neukončí aplikaci.

4.5 Propagace informací v síti

V peer-to-peer síti musí být zajištěno šíření informací o uzlech, které se nově připojily do naší sítě, nebo se z ní naopak odpojily. Z toho důvodu je v programu implementovaná funkcionalita pro propagaci těchto informací mezi uzly.

Po připojení nového uzlu je potřeba, aby se o této události dozvěděly všechny uzly v síti. K tomuto účelu je použita zpráva typu *ConnectedNodeNotification,* která je odeslána ze Super node uzlu, ke kterému se nový uzel připojil. Tato zpráva je odeslána na všechny přímo připojené uzly, tedy na takové uzly z kolekce všech uzlů, které jsou typu *Node*. Každý uzel, který přijme tuto zprávu a zatím nemá ve svém seznamu uzlů záznam o tomto nově připojeném, si přidá do tohoto seznamu objekt typu *AccessibleNode*, který reprezentuje uzly, které jsou dostupné přes nějakého prostředníka. Každý *AccessibleNode* má vlastnost *MiddlemanNode*, ve které je uveden GUID uzlu, přes který je možné s novým uzlem komunikovat. Následně každý z těchto uzlů rozešle zprávu dál na všechny uzly, které má k sobě připojené.

O odpojení některého z účastníků v síti jsou ostatní informováni prostřednictvím zaslané zprávy typu *DisconnectMessage*. Tato zpráva je odeslána vždy, když dojde k odpojení uzlu, ať už je to z důvodu nějaké chyby, nebo cíleného ukončení aplikace uživatelem.

Jelikož nově připojený uzel nemá ponětí o ostatních uzlech připojených v síti, tak je nutné mu tuto informaci předat. K tomuto účelu je použita zpráva *BroadcastNodesMessage*, která obsahuje seznam všech uzlů, které jsou přímo i nepřímo připojeny k Super node uzlu, ke kterému se nový uzel připojuje.

4.6 Navázání spojení vzdálené plochy

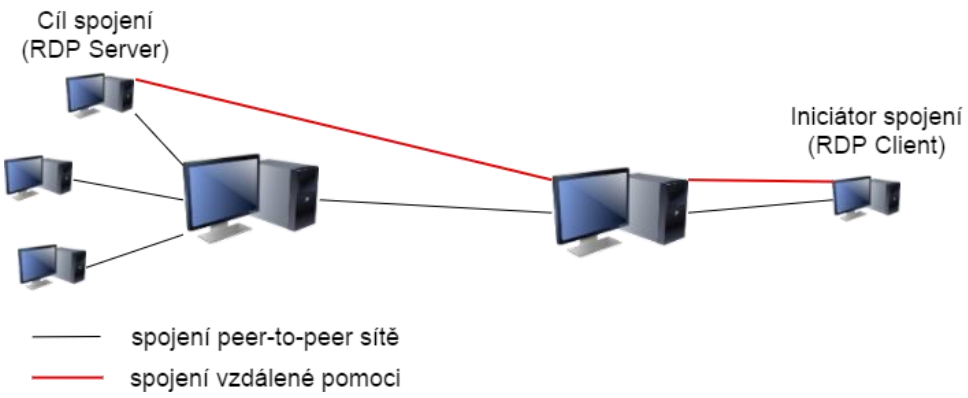
Před samotným spuštěním programu *Připojení ke vzdálené ploše* je potřeba provést několik úkonů, aby bylo vůbec možné spojení úspěšně navázat. Mezi tyto úkony patří ověření, zda je možné se k danému uzlu vzdáleně připojit, nastavení jednotlivých uzlů, aby bylo možné navázat spojení vzdálené plochy, a úprava RDP souboru, který používá program *Připojení ke vzdálené ploše* pro připojení.

4.6.1 Příprava na spojení

Celý proces připojení ke vzdálené ploše začíná tím, že libovolný uzel typu Ordinary node nebo Super node zavolá metodu *BeginRemoteDesktopConnection* s parametrem typu GUID,

který představuje identifikátor konkrétního uzlu ze seznamu připojených uzlů. Nejprve je nutné zjistit, jaký typ uzlu se skrývá pod tímto identifikátorem, a určit, o jaký typ spojení půjde. V naší síti můžou být celkem čtyři různé druhy spojení. Jedná se o tato spojení:

- *Super node → Super node*: Tento typ spojení je nejjednodušší. Provede se přímé připojení RDP klienta z jednoho uzlu na IP adresu druhého uzlu, kde na portu 3389 naslouchá RDP server.
- *Ordinary node → Super node*: Pokud je toto spojení navázáno mezi Ordinary node uzlem a Super node uzlem, ke kterému je v síti přímo připojen, jedná se opět o přímé připojení RDP klienta k RDP serveru, a tedy k nejjednoduššímu typu spojení. I v případě, že se Ordinary node připojuje k Super node uzlu, ke kterému není přímo připojen, tak se stále jedná o přímé připojení RDP klienta k RDP serveru, protože komunikace vzdálené pomoci nebude směrována přes prostředníka. To je možné, protože i koncový Super node má veřejnou IP adresu, ke které je možné se připojit.
- *Super node → Ordinary node*: Tento typ spojení ukrývá dvě možné alternativy spojení. Může se jednat o přímé spojení mezi Super node uzlem a Ordinary node uzlem, který je k němu přímo připojený. Zde je možné provést úspěšné připojení ke vzdálené ploše za pomoci lokálních proxy serverů vytvořených na obou uzlech. Druhou možností je připojení Super node uzlu k Ordinary node uzlu, který je dostupný přes jiný Super node. Zde je potřeba vytvořit lokální proxy servery a zároveň jeden proxy server na uzlu, který je prostředníkem pro tyto dva uzly.
- *Ordinary node → Ordinary node*: U tohoto typu spojení je jisté, že mezi koncovými uzly bude vždy existovat jeden prostředník ve formě Super node uzlu. Také v případě, který můžeme vidět na obrázku číslo 18, bude komunikace vzdálené pomoci probíhat pouze přes jeden Super node uzel, aby nedocházelo ke zbytečnému zatěžování jiných uzlů. Toto spojení je tedy možné vytvořit jen s pomocí lokálních proxy serverů a proxy serveru na prostředníkovi.



Obrázek 18: Vzdálená pomoc mezi dvěma Ordinary node uzly

Jako příklad pro vysvětlení si vezměme situaci, při které je použit typ spojení Ordinary node → Ordinary node. Když už známe typ spojení, odešle iniciátor zprávu typu *RemoteHelpMessage*. Tato zpráva je doručena přes prostředníka až k cílovému uzlu. Prostředník na základě této zprávy vytvoří nový pár z IP adresy a portu iniciátora a cíle spojení, mezi kterými bude následně přeposílat komunikaci mezi RDP klientem a RDP serverem.

Cílový uzel zaznamená přijetí zprávy typu *RemoteHelpMessage*. Pokud zatím neexistuje žádné spojení vzdálené plochy nebo vzdálené pomoci s cílovým uzlem, vytvoří se lokální proxy a zavolá se metoda *StartProxyType1*, která zajistí připojení k proxy serveru na Super node uzlu a k RDP serveru. Jakmile je cílový uzel hotov s přípravami, odešle zpět iniciátorovi spojení zprávu typu *RemoteHelpReplyMessage*, která ho informuje, zda se může připojit ke vzdálené ploše nebo nikoli. Tímto jsou všechny zúčastněné uzly připraveny a může dojít k poslednímu kroku, kterým je vygenerování RDP souboru a připojení ke vzdálené ploše pomocí tohoto souboru. Jakmile je spuštěn program *Připojení ke vzdálené ploše*, je uživatel požádán o zadání přihlašovacích údajů a vše již dále probíhá stejným způsobem jako při běžném připojování ke vzdálené ploše.

4.6.2 Úprava RDP souboru

Dříve než se spustí program *Připojení ke vzdálené ploše*, je nutné vytvořit nový RDP soubor a upravit některé parametry v něm obsažené. Tento proces vykoná metoda *CreateRDPConnectionFile* ze třídy *RemoteDesktopManager*. U vytvořeného souboru je v metodě změněn řádek s názvem *full address*, kde vyplníme IP adresu a port vzdáleného uzlu. Port je potřeba změnit v případech, kdy se připojujeme k nějakému proxy serveru, a ne přímo k RDP serveru, který naslouchá na portu 3389. Další upravený řádek nese název

authentication level. Tento řádek určuje, zda bude při přihlášení použit certifikát. Musíme ho nastavit na hodnotu 0, aby se certifikát nepoužíval. Pokud byl certifikát použit, vždy byla komunikace ukončena při zobrazení hlášky o důvěryhodnosti spojení. Z tohoto důvodu je také použit RDP soubor, protože tento parametr nejde změnit jinak než nastavením v tomto souboru.

4.7 Navázání spojení vzdálené pomoci

Druhou implementaci vzdálené správy počítače představuje program *Vzdálená pomoc systému Windows*. Stejně jako při použití vzdálené plochy i zde je potřeba nejdříve zjistit, zda je vůbec možné požádat vzdáleného účastníka o pomoc, provést potřebná nastavení na všech zúčastněných uzlech, a až poté je možné vytvořit a poslat pozvánku vzdálenému uzlu.

4.7.1 Příprava na spojení

Tento proces přípravy na spojení začíná voláním metody StartRAInvitation. Tato metoda je zavolána s parametrem (Id uzlu), který si uživatel vybral, aby mu poskytl vzdálenou pomoc. Jako v předchozím případě, v této metodě dojde také nejprve k vybrání typu spojení na základě typu uzlu, který žádá o pomoc, a typu uzlu, který je požádán o pomoc. Jakmile je nám známo, o jaký typ spojení se jedná, dojde ke spuštění programu *Vzdálená pomoc systému Windows*. Ten je spuštěn metodou *CreateInvitation,* která před samotným spuštěním programu zjistí, zda již neexistuje pozvánka se jménem *Invitation.msrcincident*. Pokud by totiž tento soubor existoval, program vzdálené pomoci by vyvolal dialogové okno, kde chce potvrdit zachování souboru, nebo jeho nahrazení námi nově vytvořenou pozvánkou. To jsou zbytečné úkony navíc, které by mohly nezkušeného uživatele zmást. Pokud tedy takový soubor existuje, tak je smazán. Dalším krokem této metody je již spuštění programu *Vzdálená pomoc systému Windows* a vytvoření nové pozvánky. O jeho spuštění se stará metoda *StartRemoteAssistance* ze třídy *RemoteAssistanceManager*. Vytvořenou pozvánku je nutné upravit podle toho, o jaký typ spojení se jedná. Tomuto procesu je věnována vlastní podkapitola.

Na rozdíl od vzdálené plochy, kde připojení probíhalo ve směru od iniciátora spojení ke vzdálenému počítači, tady se naopak připojuje vzdálený počítač k iniciátorovi spojení, který vytvořil pozvánku. Z toho důvodu je potřeba, aby se provedlo nastavení v síti ještě před tím, než je pozvánka odeslána vzdálenému uzlu. Toto nastavení zajišťujeme odesláním zprávy typu *RemoteAssistancePrepareMessage*. Každý participující uzel na tuto zprávu zareaguje, a pokud to typ spojení vyžaduje, spustí se na potřebných místech proxy server. Vytvořený

a upravený soubor pozvánky je nyní poslán prostřednictvím zprávy *RemoteHelpMessage* konečnému uzlu.

V okamžiku, kdy cílový uzel obdrží zprávu, vezme zaslanou pozvánku a otevře ji v programu *Vzdálená plocha systému Windows*. Tím se cílový uzel připojí přímo na uzel, který byl iniciátorem spojení, nebo na jiný uzel, který tvoří prostředníka. K tomu slouží metoda *AcceptInvitation*. Od této chvíle již funguje program *Vzdálená pomoc systému Windows* normálním způsobem.

4.7.2 Úprava pozvánky

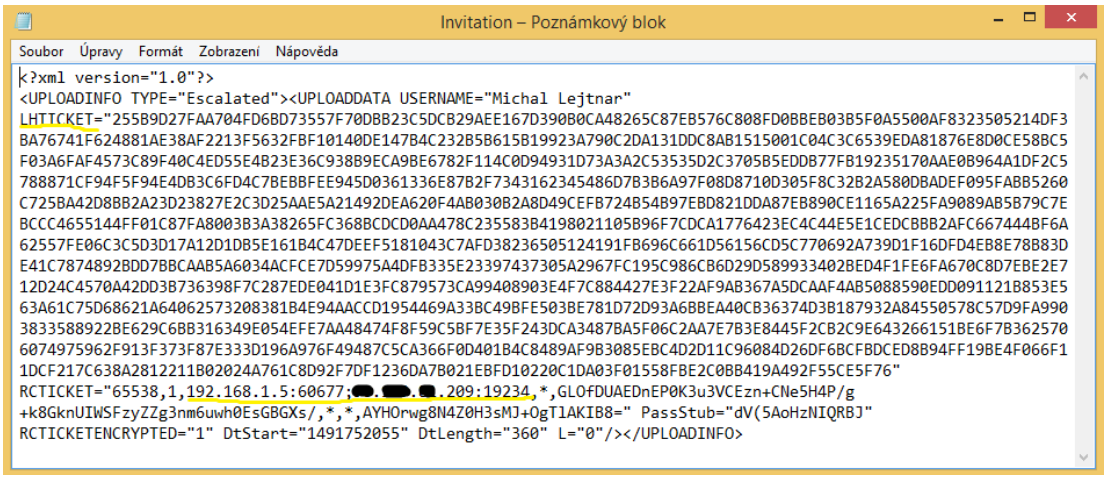
Při standardním vytváření pozvánky programem *Vzdálená pomoc systému Windows* se do části připojovacího řetězce vloží IP adresa uzlu, který pozvánku vytváří, a náhodně vybraný port z rozsahu 49152–65535. Vždy když se nejedná o přímé připojení libovolného uzlu k Super node uzlu (Ordinary node → Ordinary node, Super node → Ordinary node), je potřeba upravit pozvánku.

O úpravu pozvánky se stará metoda *CreateModifiedText*, která se nachází ve třídě *InvitationParser*. Tato metoda provádí rozbor originální pozvánky (parsing) a hledá část s názvem *RCTICKET*, která obsahuje zmíněný připojovací řetězec. Z připojovacího řetězce je vybrána část, která obsahuje informace o kombinacích IP adres a portů, přes které je možné se k počítači vzdáleně připojit. Hlavním úkolem metody *InvitationParser* je záměna originální IP adresy a portu, které byly vygenerovány programem, za jinou IP adresu a port, na které bude naslouchat nějaký proxy server. Originální údaje nezahazujeme. Budeme je dále potřebovat, aby se proxy server mohl připojit k aplikaci spuštěné na počítači, který vytvářel pozvánku a nyní naslouchá na originálním portu a očekává vzdálené připojení.

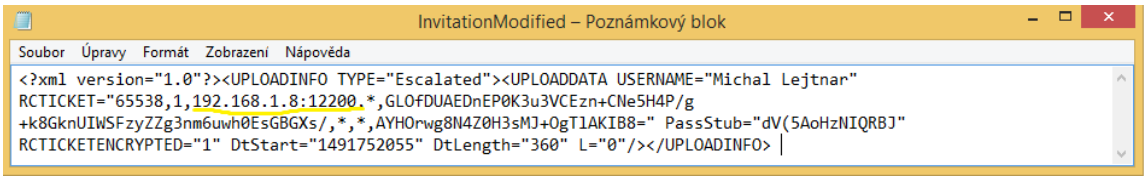
V průběhu tvorby aplikace bylo zjištěno, že poté, co byly změněny údaje v pozvánce, se stále používá originální IP adresa a port. Toto chování je zapříčiněno tím, že všechny operační systémy Windows, které jsou novější než Windows XP, nepoužívají k připojení informace obsažené v části *RCTICKET*. Místo toho tyto informace získávají z části s názvem *LHTICKET*, která obsahuje veškeré informace uvedené v části *RCTICKET* v podobě hashe. Zřejmě v zájmu zpětné kompatibility jsou i novější OS Windows schopné pracovat s údaji v části *RCTICKET*, pokud jim v souboru chybí část s hashem.

Aby bylo možné provést připojení přes prostředníka, je tedy nutné nejprve změnit IP adresu a port a následně odstranit ze souboru celou část s názvem *LHTICKET*. Po takové

úpravě souboru se již můžeme korektně připojit k libovolnému uzlu, jehož adresu a port vložíme do souboru.



Obrázek 19: Originální pozvánka vzdálené pomoci



Obrázek 20: Upravená pozvánka vzdálené pomoci

4.8 Funkce proxy serveru

V naší aplikaci existují dvě implementace proxy serveru. První nese název *LocalProxy* a je spuštěn vždy na koncovém nebo počátečním uzlu, kde je spuštěn jeden z programů pro vzdálenou správu počítače. Druhý typ nese název *RemoteProxy* a běží na všech uzlech typu Super node.

Lokální proxy server je spuštěný vždy až ve chvíli, kdy je potřeba. Každý lokální proxy server obsahuje jednu instanci třídy *ConnectionBridge*, která představuje spojení dvou konkrétních uzlů. Nová instance se může vytvořit celkem čtyřmi různými metodami. To je závislé na typu konečných uzlů, které se budou připojovat. Podle toho vytváříme *ConnectionBridge* s různým nastavením jednotlivých socketů. Možnostmi jsou *Connect-Connect*, *Connect-Listen*, *Listen-Listen* a *Connect-Connect concrete*. Tyto názvy odpovídají tomu, jak budou sockety nastaveny, jestli budou naslouchat na portu, nebo se budou

k nějakému portu připojovat. Jakmile je *ConnectionBridge* vytvořen a oba sockety mají navázané spojení, dochází k asynchronnímu přeposílání dat z jednoho socketu na druhý.

Vzdálený proxy server je vytvořen ihned při startu uzlu. U tohoto typu je možné vytvořit více instancí třídy *ConnectionBridge*, kde každá představuje spojení dvou konkrétních uzlů. *ConnectionBridge* je vytvořen v průběhu nastavení sítě, které probíhá před navázáním spojení vzdálené plochy nebo vzdálené pomoci. Při vytváření se dozví, jaké jsou IP adresy uzlů, které má spojit. Když pak dojde k navazování spojení pomocí socketů, jsou sockety přirazeny do odpovídajícího *ConnectionBridge* objektu. V takto vytvořeném objektu probíhá přeposílání dat mezi sockety stejným způsobem, jako tomu je u lokálního proxy serveru.

5 Testování

Funkčnost vytvořené aplikace byla testována v reálné síti za použití několika počítačů s různými verzemi operačního systému Windows. Vzhledem k omezenému počtu veřejných IP adres, které by mohly být použity při testování (1 veřejná IP adresa), byla funkčnost celé aplikace testována převážně na lokální síti. Při testování na lokální síti se aplikace jevila jako funkční řešení, které běželo bez větších potíží.

Dílčí funkcionality byly samozřejmě testovány také v situaci, kdy se koncový uzel připojoval přes Internet do lokální sítě. Pro tento testovací scénář bylo potřeba nastavit na směrovači přesměrování portů tak, aby komunikace přicházející na veřejnou IP adresu na porty používané aplikací byla přesměrována na počítač, který slouží v lokální síti jako Super node. Při testování v tomto prostředí docházelo k občasným problémům s připojením, u kterých předpokládáme, že jejich zdrojem mohlo být nějaké nastavení na směrovači.

V průběhu vývoje byla aplikace testována celkem na pěti různých verzích OS Windows. Jedná se o Windows Vista Home Premium, Windows 7 Home Premium, Windows 7 Starter, Windows 8.1 Enterprise a Windows 10 Home. Jako nejvíce problémový se zdál být počítač s Windows 7 Starter, kvůli kterému bylo nutné upravit spouštění externích programů (*Připojení ke vzdálené ploše*, *Vzdálená pomoc systému Windows*). Namísto přímého spuštění aplikace v novém procesu bylo nutné spustit příkazový řádek, ze kterého bylo poté možné spustit požadovanou aplikaci.

6 Návrh na zlepšení

Během tvorby aplikace bylo zjištěno několik možností, které by mohly celou aplikaci vylepšit a zjednodušit používání programu.

Nejvýraznějším vylepšením se zdá být automatické rozhodování o tom, jaký typ uzlu bude jedna konkrétní spuštěná aplikace představovat. V současnosti je toto rozhodnutí na uživateli. Zde nastává problém, pokud budou aplikaci užívat nezkušení uživatelé. Ti mohou vybrat typ uzlu Super node, i když nebudou mít veřejnou IP adresu, což je minimální požadavek pro tento typ uzlu. Nezkušený uživatel nemusí tomuto požadavku rozumět, a tak něco náhodně vybere.

Pokud by se tento výběr prováděl automaticky, bylo by možné proces výběru obohatit o další metriky, které používá pro výběr typu uzlu i třeba Skype [19]. Mezi tyto metriky patří výkon procesoru, velikost operační paměti a rychlost připojení. Dalším parametrem by mohla být doba, po kterou je počítač zapnutý a připojený k Internetu.

Dalším vylepšením by mohlo projít samotné připojování uzlů. V současné době dochází k hloupému připojení Ordinary node uzlu k jakémukoli Super node uzlu, jehož adresu má v XML souboru s dostupnými uzly. V tomto případě může dojít k nevyváženému zatížení jednotlivých Super node uzlů a zpomalení části sítě. Nehledě na to, že se nebere v potaz ani vzájemná vzdálenost uzlů nebo odezva na síti. Řešením by bylo sofistikované rozhodování o možnosti připojení nového uzlu podle vytížení jednotlivých Super node uzlů nebo podle dalších síťových metrik (např. počet směrovačů na cestě, odezva).

7 Závěr

Cílem této diplomové práce bylo vytvořit prototyp systému postaveného na bázi peer-to-peer sítě, který by sloužil pro vzdálené ovládání počítačů. Pro řešení vzdáleného ovládání mělo být použito terminálových služeb operačního systému Windows.

Teoretická část této práce se zabývá popisem peer-to-peer sítí a jejich rozdělením podle různých kritérií. Dále tato část obsahuje popis terminálových služeb jako celku, ale také popis jednotlivých zástupců, kterými jsou programy *Připojení ke vzdálené ploše* a *Vzdálená pomoc systému Windows*.

V následující části byl popsán způsob výběru nejvhodnější architektury peer-to-peer sítě, kde se jako nejlepší možnost jeví hybridní typ sítě. Na základě výběru typu sítě bylo nutné popsat jednotlivé účastníky (uzly), kteří se budou v systému vyskytovat. Tato část dále obsahuje popis, jakým způsobem budou v aplikaci využity programy pro vzdálené ovládání počítače a jakým způsobem bude vyřešena jejich komunikace mezi jednotlivými uzly, které se mnohdy mohou vyskytovat za nějakým směrovačem a nemusí mít veřejnou IP adresu.

Poslední část popisuje implementaci tohoto řešení, která byla provedena v programovacím jazyce C#. V této části je popsána implementace vzájemné interakce mezi jednotlivými uzly a způsob, jakým jsou do aplikace zakomponovány programy OS Windows pro vzdálené ovládání počítače.

V závěru je možné říci, že byl úspěšně vytvořen prototyp systému pro vzdálené ovládání počítače a tím bylo dosaženo zadaných cílů diplomové práce.

Seznam použité literatury

[1] *Peer-to-Peer Systems and Applications* [online]. Ilustrované vydání. Berlin: Springer, 2005 [cit. 2017-02-10]. ISBN 0302-9743. Dostupné z: https://books.google.cz/books?id=A8CLZ1FB4qoC&pg=PA353&redir_esc=y#v=onepage&q&f=false

[2] BERÁNEK, Ladislav. Peer-to-peer (P2P) systémy a jejich bezpečnost. *Historie programování a VT u nás* [online]. 2005 [cit. 2017-02-10]. Dostupné z: http://prog-story.technicalmuseum.cz/images/dokumenty/Programovani-TSW-1975-2014/2005/2005-02.pdf

[3] BALKE, Wolf-Tilo; SIBERSKI Wolf. *Unstructured Unstructured Peer-to-Peer Networks* [online prezentace]. Hannover: L3S Research Center, University of Hannover, [cit. 2017-02-12]. Dostupné z WWW: <http://www.l3s.de/~balke/lecture-p2p/Vorlesung_2.pdf>.

[4] SCHODER, Detlef; FISCHBACH, Kai; SCHMITT, Christian. Core Concepts in Peer-to-Peer. *Peer-to-peer computing: The Evolution of a Disruptive Technology*, 2005, 1.

[5] MIHAI, Lupu, Quang Hieu VU a Ooi BENG CHIN. *Peer-to-Peer Computing: Principles and Applications* [online]. Berlín, 2010 [cit. 2017-02-16]. ISBN 978-3-642-03514-2. Dostupné z: http://www.springer.com/cn/book/9783642035135

[6] ZHOU, Hu. *NAT Traversal Techniques and Peer-to-Peer Applications* [online]. Helsinki, 2005 [cit. 2017-02-16]. Dostupné z: http://www.tml.tkk.fi/Studies/T-110.551/2005/papers/hu.pdf

[7] Remote Desktop Protocol (RDP): Chapter 3: Communication Protocols and Thin Clients. *ETutorials.org* [online]. ©2008-2017 [cit. 2017-02-16]. Dostupné z: http://etutorials.org/Microsoft+Products/microsoft+windows+server+2003+terminal+services/Chapter+3+Communication+Protocols+and+Thin+Clients/Remote+Desktop+Protocol+RDP/

[8] Configuring Remote Desktop. *Microsoft TechNet* [online]. 2017 [cit. 2017-02-16]. Dostupné z: https://technet.microsoft.com/en-us/library/bb457106.aspx

[9] BÁRTA, Jaroslav. *Distribuované Hashovací Tabulky* [online prezentace]. 2009 [cit. 2017-02-16]. Dostupné z: http://www.kiv.zcu.cz/~ledvina/Cviceni-PDS-2009/barta.pdf

[10] Overview of Remote Assistance in Windows XP. *ExpertReplies* [online]. 2016 [cit. 2017-02-16]. Dostupné z: https://expertreplies.com/overview-of-remote-assistance-in-windows-xp/

[11] Supporting Windows 7 Users with Remote Assistance. *Microsoft Press Store* [online]. 2016 [cit. 2017-03-11]. Dostupné z: https://www.microsoftpressstore.com/articles/article.aspx?p=2229237

[12] Description of the Remote Assistance Connection Process. *Microsoft* [online]. 2017 [cit. 2017-04-15]. Dostupné z: https://support.microsoft.com/en-us/help/300692/description-of-the-remote-assistance-connection-process

[13] Step-by-Step Guide to Remote Assistance. *Microsoft TechNet* [online]. 2017 [cit. 2017-04-13]. Dostupné z: https://technet.microsoft.com/en-us/library/bb457004.aspx

[14] [MS-RAI]: Appendix A: Remote Assistance Invitation File Format. *Microsoft Developer Network* [online]. 2017 [cit. 2017-03-13]. Dostupné z: https://msdn.microsoft.com/en-us/library/cc240167.aspx

[15] [MS-RAI]: Remote Assistance Connection String 1. *Microsoft Developer Network* [online]. 2017 [cit. 2017-03-13]. Dostupné z: https://msdn.microsoft.com/en-us/library/cc240131.aspx

[16] Comparison of remote desktop software. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2017 [cit. 2017-03-13]. Dostupné z: https://en.wikipedia.org/wiki/Comparison_of_remote_desktop_software

[17] Virtual Network Computing. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2017 [cit. 2017-03-13]. Dostupné z: https://cs.wikipedia.org/wiki/Virtual_Network_Computing

[18] What is Microsoft .Net Framework. *Net-informations.com* [online]. [cit. 2017-03-24]. Dostupné z: http://vb.net-informations.com/framework/what_is_net_framework.htm

[19] BASET, Salman A. a Henning G. SCHULZRINNE. *An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol: Salman A. Baset and Henning G. Schulzrinne* [online]. 2006 [cit. 2017-03-24]. Dostupné z: http://www1.cs.columbia.edu/~salman/publications/skype1_4.pdf

[20] Controlling Skype: Network Overview, Skype Login and FortiGate Configuration to block Skype. *Fortinet Knowledge Base* [online]. [cit. 2017-03-24]. Dostupné z: http://kb.fortinet.com/kb/viewContent.do?externalId=FD30776

Příloha A – Manuál k systému

Pro zprovoznění systému je potřeba, aby měl každý počítač, na kterém budeme chtít spustit aplikaci, nainstalovaný .NET Framework ve verzi 4.5.2. Aplikaci zkopírujeme do adresáře, ve kterém máme právo k zapisování. Dalším krokem je spuštění jednotlivých uzlů. Vždy musí být nejprve spuštěn Central node, ke kterému se připojují Super node uzly. Až posledním krokem je spuštění uzlu typu Ordinary node.

Spuštění Central node uzlu

1. Vytvoříme databázi, která bude obsahovat tabulku *Users*. Pro vytvoření databáze můžeme použít skript uložený na přiloženém CD.
2. V konfiguračním souboru *RemoteHelper.exe.config* změníme hodnotu klíče *DBConnectionString* na připojovací řetězec k vytvořené databázi.
3. Spustíme program, kde nemusíme zadávat žádné přihlašovací údaje. Je nutné změnit typ uzlu na *Central node* a stisknout tlačítko *Přihlášení.*

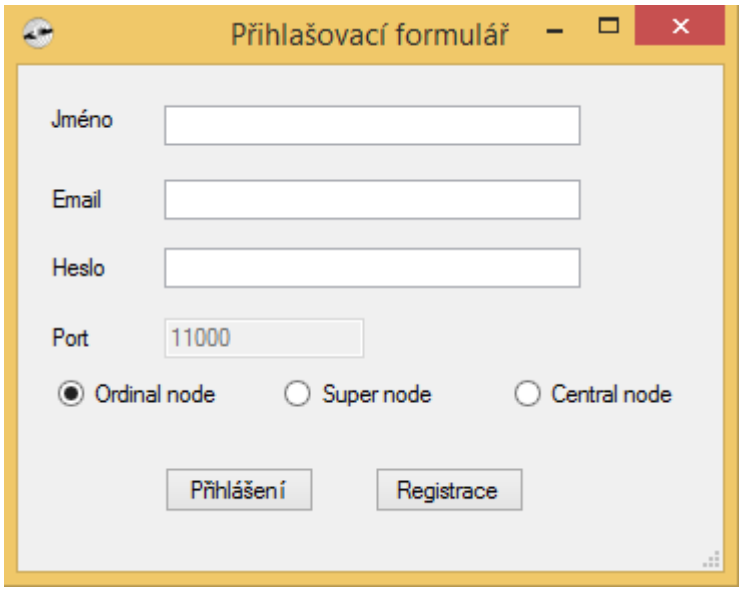
Spuštění Super node uzlu

1. V konfiguračním souboru *RemoteHelper.exe.config* zapíšeme IP adresu vytvořeného Central node uzlu do hodnoty klíče *CentralnodeAddress*. Pokud máme vytvořené také alternativní Central node uzly, můžeme jejich IP adresy zadat jako hodnoty klíče *CentralnodeAddress2* a *CentralnodeAddress3.*
2. Spustíme program. Můžeme využít testovacích dat v databázi, zadat email s heslem, změnit typ uzlu na *Super node* a stisknout tlačítko *Přihlášení.* Pokud nechceme používat testovací data, zadáme jméno, email a stiskneme tlačítko *Registrace.*

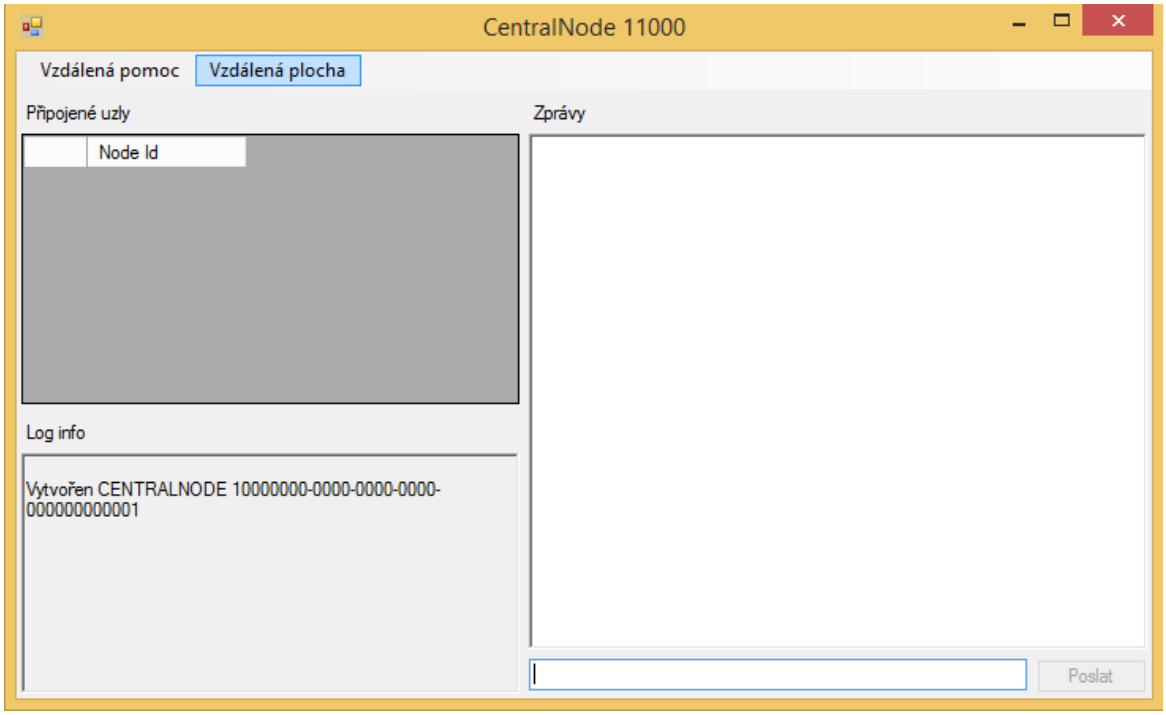
Spuštění Central node uzlu

1. V konfiguračním XML souboru *ConfigLocal*, změníme IP adresu uzlu v atributu s názvem *Address* na IP adresu, kterou má námi vytvořený Super node.
2. Spustíme program. Jako v případě Super node uzlu i zde můžeme využít testovací data v databázi nebo vytvořit vlastní přihlašovací údaje.

Příloha B – Grafické uživatelské rozhraní



Obrázek 21: Přihlašovací formulář aplikace Remote helper



Obrázek 22: Hlavní formulář aplikace Remote helper

Příloha C – Obsah přiloženého CD

- Zdrojové soubory programu Remote helper
- Skript pro tvorbu databáze
- Spustitelná aplikace