VYSOKÁ ŠKOLA POLYTECHNICKÁ JIHLAVA

Aplikovaná informatika

TUNELOVACÍ SOFTWARE

Bakalářská práce

Autor práce: Michal Mátl

Vedoucí práce: Mgr. Antonín Přibyl

Jihlava 2024

Vysoká škola polytechnická Jihlava

Tolstého 16, 586 01 Jihlava

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Autor práce:	**Michal Mátl**
Studijní program:	Aplikovaná informatika
Obor:	Aplikovaná informatika
Garant studijního programu:	Ing. Lenka Kuklišová Pavelková, Ph.D.

Název práce:	**Tunelovací software**
Vedoucí práce:	Mgr. Antonín Přibyl
Cíl práce:	Cílem práce je návrh a vytvoření aplikace, která bude sloužit k zpřístupnění lokálního portu (TCP nebo UDP) do internetu, přičemž uživatel nemusí mít veřejnou IP adresu. Uživatel si po spuštění aplikace zvolí protokol a číslo portu, který chce zpřístupnit. Aplikace bude obsahovat klientskou aplikaci a serverovou část, která bude řešit přiřazování portů jednotlivým uživatelům aplikace. Serverová část bude zprovozněna na virtuálním stroji s veřejnou IP adresou a doménou. Aplikace bude vytvořena v programovacím jazyku Java. Obě části aplikace budou spustitelné na operačních systémech Windows a Linux.

Abstrakt

Cílem práce je návrh a vytvoření aplikace, která slouží k zpřístupnění lokálního portu (TCP nebo UDP) do internetu, přičemž uživatel nemusí mít veřejnou IP adresu. Aplikace je vytvořena v programovacím jazyku Java, pro grafické rozhraní je použita knihovna Swing. Výsledkem jsou dvě aplikace: klientská a serverová. Pomocí klientské aplikace může uživatel zpřístupnit port do internetu. Serverová aplikace musí být spuštěná na serveru s veřejnou IP adresou. Obě části aplikace jsou spustitelné na operačních systémech Windows a Linux.

Klíčová slova

port; tunel; TCP; UDP; Java; Swing

Abstract

Work aims to design and create an application that can be used to expose local port (TCP or UDP) to the Internet, whereby the user does not have to own public IP address. Application is created using Java programming language and the Swing library is used for graphical interface. The result is two applications: client and server. User can expose port to the Internet using the client application. The server application has to run on a server with public IP address. Both parts of the application are executable on Windows and Linux operating systems.

Keywords

port; tunnel; TCP; UDP; Java; Swing

Prohlašuji, že předložená bakalářská práce je původní a zpracoval/a jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem v práci neporušil/a autorská práva (ve smyslu zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů, v platném znění, dále též „**AZ**“).

Byl/a jsem seznámen/a s tím, že na mou bakalářskou práci se plně vztahuje **AZ**, zejména § 60 (školní dílo).

Podle § 47b zákona o vysokých školách souhlasím se zveřejněním své práce podle Směrnice pro vedení, vypracování a zveřejňování závěrečných prací na VŠPJ, a to bez ohledu na výsledek obhajoby.

Beru na vědomí, že VŠPJ má právo na uzavření licenční smlouvy o užití mé bakalářské práce a prohlašuji, že **s o u h l a s í m** s případným užitím mé bakalářské práce (prodej, zapůjčení apod.).

Jsem si vědom/a toho, že užít své bakalářské práce či poskytnout licenci k jejímu využití mohu jen se souhlasem VŠPJ, která má právo ode mě požadovat přiměřený příspěvek na úhradu nákladů, vynaložených vysokou školou na vytvoření díla (až do jejich skutečné výše), z výdělku dosaženého v souvislosti s užitím díla či poskytnutím licence.

V Jihlavě dne 8. dubna 2024

…….

Podpis studenta/ky

Poděkování

Tímto bych chtěl poděkovat Mgr. Antonínovi Přibylovi za odbornou pomoc při vypracování mé bakalářské práce.

Obsah

Seznam obrázků

Seznam zkratek

DHCP	Dynamic Host Configuration Protocol
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IDE	Integrated Development Environment
JDK	Java Development Kit
JRE	Java Runtime Environment
JVM	Java Virtual Machine
NAT	Network Address Translator
RAM	Random Access Memory
SSH	Secure Shell
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VCPU	Virtual Central Processing Unit
VPS	Virtual Private Server

Úvod

Téma bakalářské práce jsem si zvolil, protože jsem potřeboval jednoduše zpřístupnit lokální port do internetu, a ačkoliv existuje více podobných aplikací jako je tato, tak jsem nenašel takovou, která by splňovala všechny moje požadavky. Mezi tyto požadavky patří především cena, odezva připojení a podpora jak protokolu TCP, tak i UDP.

Cílem práce je vytvořit jednoduchou aplikaci, kterou uživatelé mohou použít např. pro dočasné zveřejnění lokálního webového serveru či herního serveru bez nutnosti mít veřejnou IP adresu. Aplikace bude obsahovat klientskou aplikaci a serverovou část, která bude řešit přiřazování portů jednotlivým uživatelům aplikace. Serverová část bude zprovozněna na virtuálním stroji s veřejnou IP adresou a doménou. Aplikace bude vytvořena v programovacím jazyku Java. Obě části aplikace budou spustitelné na operačních systémech Windows a Linux. Aplikace bude obsahovat jak grafické rozhraní, tak i konzolové rozhraní.

1 Způsoby zpřístupnění lokálního portu do internetu

Je zde několik různých způsobů, jak lze zpřístupnit lokální port do internetu. V následujících podkapitolách se zaměříme na jejich využití a jakým způsobem fungují.

1.1 Přesměrování portů

Před nastavením přesměrování portů je vhodné nastavit si statickou IP adresu na počítači, z kterého chceme přesměrovávat porty. Poté stačí jít administrace domácího routeru a v něm najít sekci přesměrování portů nebo port forwarding. Zde se nastaví IP adresa a port lokálního počítače a externí port. Dále lze zvolit protokol TCP nebo UDP. (Fisher, 2022)

Aby toto fungovalo je nutné mít veřejnou IP adresu routeru, která většinou není zdarma.

1.2 Reverzní proxy

Proxy funguje jako prostředník mezi klientem (prohlížeč či jiný program) a jinou stránkou. Když klient pošle požadavek na proxy, tak proxy požadavek přepošle na stránku a vrátí odpověď zpět klientovi. Klient musí být v tomto případě nakonfigurován tak, aby využíval proxy k přístupu na internet. (Apache, 2013)

Reverzní proxy je z pohledu klienta stejná jako jakýkoliv jiný server. Klient pošle požadavek na adresu reverzní proxy a ta požadavek přesměruje na jiný server a vrátí odpověď, jako kdyby jej poslala proxy. Tohoto se dá využít například k zpřístupnění serveru za firewallem či serveru bez veřejné IP adresy. (Apache, 2013)

1.3 Reverzní SSH tunel

Pro zprovoznění SSH tunelu je potřeba server s veřejnou IP adresou, na kterém musí být spuštěný SSH server. SSH server také musí být správně nakonfigurovaný, aby dovoloval přesměrování TCP portů. To znamená, že v nastavení SSH serveru musí být povoleny možnosti GatewayPorts a AllowTcpForwarding. Poté se stačí připojit k SSH serveru s parametry, který port z lokálního počítače se zpřístupní, na jaký port serveru. Tato metoda však podporuje pouze protokol TCP. (DSL.cz, 2023)

Při připojení klienta na zadaný port serveru se pak komunikace přesměrovává přes SSH tunel na lokální port. (Tracket, 2014)

2 Protokoly TCP a UDP

V kapitole se zaměříme na rozdíly mezi protokoly TCP a UDP.

Protokol TCP se používá ve všech aplikacích, kde potřebujeme garantovat, že se všechna data dostanou k příjemci, např. u webových stránek. Na rozdíl od UDP zajišťuje správné pořadí doručení dat a také při ztrátě paketu jej znovu přepošle. UDP je na druhou stranu rychlejší, protože nekontroluje, zda byla data doručena, a tím je vhodný například pro streamování audia, videa či hraní her. (Profiber, 2023)

2.1 Protokol TCP

Protokol TCP pro své fungování potřebuje nejprve navázat spojení pomocí třístupňového ověření. První počítač pošle paket s příznakem SYN = 1 a druhý počítač odešle zpět paket s příznakem ACK = 1 a příznakem SYN = 1. První počítač pak odpoví příznakem ACK. Tím je navázáno spojení. Pokaždé když se pošle TCP paket s daty, tak příjemce musí potvrdit co obdržel. První počítač odešle paket s daty a pořadovým číslem. Druhý počítač to potvrdí odesláním příznaku ACK a zvýšením čísla od délku dat. Když chce jeden počítač ukončit spojení, tak pošle příznak FIN. Druhý počítač odpoví příznaky ACK a FIN. Spojení je ukončeno po tom, co odpoví první počítač příznakem ACK. (Khan Academy, 2024)

2.2 Protokol UDP

Protokol UDP neřeší ztrátu paketů nebo špatné pořadí paketů. UDP paket se skládá z 8 bytové hlavičky a dat. V hlavičce se nachází čísla portů pro zdroj a cíl, dále pak délka celého paketu, a nakonec kontrolní součet. Pomocí kontrolního součtu lze zjistit, jestli data byla při přenosu poškozena. Před odesláním paketu odesílatel vypočítá kontrolní součet dat. Příjemce také vypočítá kontrolní součet a porovná ho s přijatým součtem. Pokud se kontrolní součty nerovnají, data byla poškozena. (Khan Academy, 2024)

2.3 UDP hole punching

UDP hole punching je technika používaná k navázání UDP spojení mezi zařízeními schovanými za NATem. Aby tato technika fungovala je potřeba server, na který se obě propojovaná zařízení dokáží připojit. První i druhé zařízení pošle UDP paket na server. Server si zaznamená IP adresy a porty obou zařízení. Poté server pošle adresu a port prvního zařízení tomu druhému a naopak. Obě zařízení musí tomu druhému poslat alespoň 1 paket, aby mohli poté spolu komunikovat. Prvním poslaným paketem se port druhého zařízení uloží do NATu a tímto se vytvoří díra ve firewallu, která povoluje příchozí pakety z druhého zařízení. (Gianchandani, 2012)

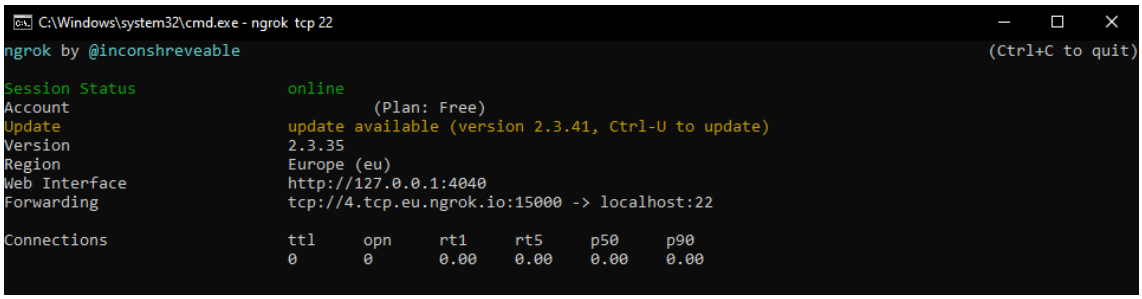
3 Podobné aplikace

Existuje mnoho podobných aplikaci určených pro zpřístupnění lokálního portu do internetu, ale některé z nich mají určité nevýhody nebo jsou jejich funkce zpoplatněné. V následujících podkapitolách se zaměříme na porovnání bezplatných verzí těchto programů.

3.1 Ngrok

Pomocí ngroku lze dočasně zpřístupnit lokální webovou aplikaci či jinou službu na veřejnou IP adresu. Po registraci a stažení programu stačí v příkazovém řádku zadat protokol a číslo portu jako parametry a zobrazí se vygenerovaná veřejná IP adresa, na kterou se pak lze připojit. (Kutáč, 2018)

Na následujícím obrázku můžeme vidět, jak vypadá program po spuštění. Zde například přesměrovává lokální TCP port 22 na veřejnou adresu.



Obr. 1: Program ngrok

zdroj: vlastní zpracování

Přestože podporuje protokoly HTTP, HTTPS a TCP, tak protokol TCP je v nejnovější verzi programu již zpoplatněn. (ngrok, 2023)

Výhody

- podporuje protokoly HTTP, HTTPS a TCP.

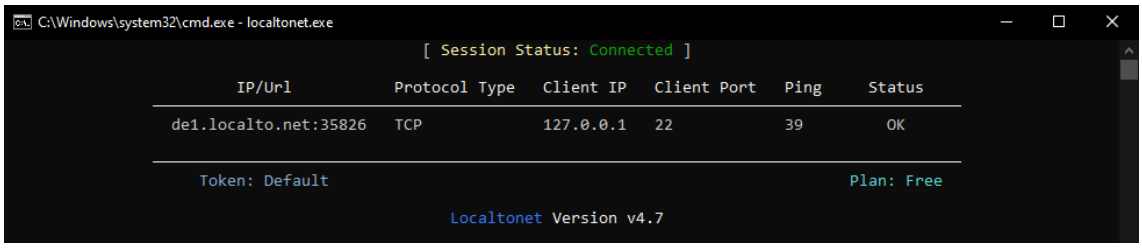
Nevýhody

- nepodporuje protokol UDP,
- protokol TCP je zpoplatněn,
- omezený bandwidth na 1 GB na měsíc na 1 spuštění programu,
- omezený počet přesměrovaných portů na 2,
- náhodná adresa při každém spuštění,
- nutnost registrace.

3.2 LocaltoNet

Tato aplikace funguje jako reverzní proxy, která zpřístupňuje lokální port do internetu. Program se nastavuje pomocí webového rozhraní, které je dostupné po registraci a spuštění aplikace. Po konfiguraci ve webovém rozhraní se zobrazí vygenerovaná veřejná IP adresa. (LocaltoNet, 2023)

Následující obrázek zobrazuje spuštěný program, který přesměrovává lokální TCP port na zobrazenou adresu.



Obr. 2: Program LocaltoNet

zdroj: vlastní zpracování

Zde můžeme vidět webové rozhraní programu LocaltoNet, které zobrazuje všechny aktivní tunely.

Obr. 3: Webové rozhraní programu LocaltoNet

zdroj: LocaltoNet (2023), vlastní zpracování

Výhody

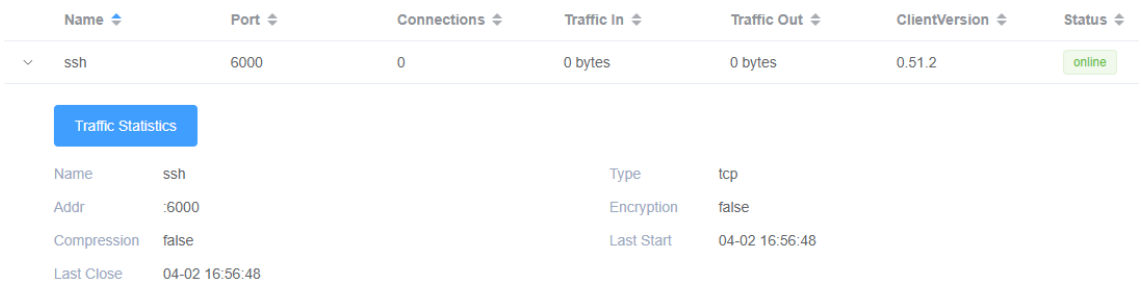
- webové grafické rozhraní,
- podporuje protokoly TCP, UDP a HTTP.

Nevýhody

- omezený bandwidth na 1 GB na měsíc,
- pouze 1 přesměrovaný port,
- náhodná adresa při každém spuštění,
- nutnost registrace.

3.3 Frp

Fast reverse proxy (frp) je jednoduchá reverzní proxy, která se dá použít na zpřístupnění lokálního portu na veřejný server. Podporuje protokoly TCP, UDP, HTTP a HTTPS. K zprovoznění programu je zapotřebí mít server s veřejnou IP adresou, na kterém bude spuštěna aplikace pro server. Uživatel si pak stáhne klientskou aplikaci, která se nastavuje pomocí konfiguračního souboru. Aplikace se pak spouští pomocí příkazového řádku. (Tanner, 2020)

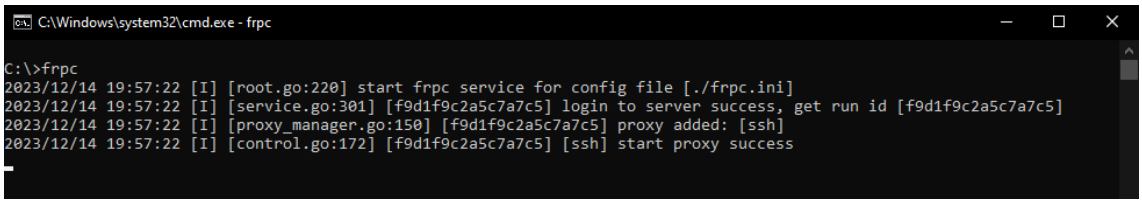


Obr. 4: Grafické administrátorské rozhraní programu frp

zdroj: vlastní zpracování

Dále aplikace obsahuje grafické administrátorské rozhraní v prohlížeči, které zobrazuje přesměrované porty a statistiky přenesených dat. Program nemá omezení na počet přesměrovaných portů. (Fatedier, 2023)

Na následujícím obrázku je spuštěná klientská část programu, která po spuštění načetla konfigurační soubor a začala přesměrovávat nastavený port



Obr. 5: Klientská část programu frp

zdroj: vlastní zpracování

V konfiguračním souboru je nastaveno, aby se přesměroval lokální TCP port 22 na veřejný port 6000 zadaného serveru.

```
1  [common]
2  server_addr = 1.1.1.1
3  server_port = 7000
4
5  [ssh]
6  type = tcp
7  local_ip = 127.0.0.1
8  local_port = 22
9  remote_port = 6000
```

Obr. 6: Konfigurační soubor programu frp

zdroj: vlastní zpracování

Výhody

- zdarma,
- open-source,
- podporuje protokoly TCP, UDP, HTTP a HTTPS,
- není omezen počet přesměrovaných portů,
- grafické rozhraní v prohlížeči.

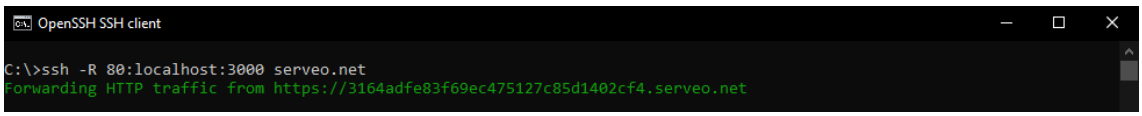
Nevýhody

- složitější konfigurace,
- nutnost mít vlastní server s veřejnou IP adresou.

3.4 Serveo

Serveo je veřejný SSH server, fungující na principu reverzního SSH tunelu, na který se stačí připojit jakýmkoliv SSH klientem či přímo z příkazové řádky, např. pomocí integrovaného SSH klienta. Po připojení se zobrazí veřejná IP adresa, na kterou je přesměrován zadaný lokální port. (Serveo, 2023)

Pokud se zadá port 80 nebo 443, tak se jako protokol použije HTTPS, jak můžeme vidět na následujícím obrázku.



Obr. 7: Program Serveo
zdroj: vlastní zpracování

Výhody

- zdarma,
- nevyžaduje instalaci,
- vlastní subdoména zdarma,
- podporuje protokoly TCP, HTTP, HTTPS,
- není omezen počet přesměrovaných portů.

Nevýhody

- nepodporuje protokol UDP.

4 Použité technologie

V kapitole si představíme technologie, které budou použité při vývoji aplikace.

4.1 Java

Java je objektově orientovaný programovací jazyk. Javu vyvinula v roce 1995 společnost Sun Microsystems. Je multiplatformní, což znamená že program napsaný v Javě může běžet na různých operačních systémech a architekturách. Kód napsaný v Javě se nejprve pomocí kompilátoru přeloží do bajt kódu (byte code) a ten pak lze spustit na jakémkoliv zařízení, které podporuje virtuální stroj Javy (JVM). Syntaxe jazyka byla navržena tak, aby se podobala programovacímu jazyku C++. (Microsoft, 2024)

Fakt, že je Java multiplatformní, byl také hlavní důvod, proč byla vybrána k vývoji aplikace.

Podle firmy Oracle Java běží na 56 miliardách zařízení po celém světě.

4.1.1 Socket

K připojení na TCP server je použita třída Socket v Javě.

Třída Socket v Javě umožňuje klientům vytvářet spojení s TCP serverem a komunikovat s ním. V konstruktoru lze specifikovat na jakou adresu a jaký port se má Socket připojit, nebo lze Socket připojit později pomocí metody connect(). (Majer, 2006)

4.1.2 DatagramSocket

Třída DatagramSocket se používá k připojení na UDP server.

Třída DatagramSocket v Javě umožňuje odesílání a přijímání UDP paketů. Každý odeslaný nebo přijatý paket je individuálně adresován a směrován. Třída DatagramSocket poskytuje metody pro navázání spojení, odesílání a přijímání paketů. (Oracle, 2024)

4.1.3 ServerSocket

Třída ServerSocket se používá na straně serveru k čekání na požadavky od klientů. V konstruktoru lze nastavit, na jakém portu bude server naslouchat. Pomocí metody accept() pak server čeká na připojení, a po připojení klienta vrátí připojený Socket. (Oracle, 2024)

4.1.4 Thread

Třída Thread v Javě se používá k vykonávání více úloh současně. Vlákno lze vytvořit buď poděděním ze třídy Thread nebo implementací rozhraní Runnable. V obou případech se musí přepsat metody run(), která obsahuje kód, který má být vykonáván ve vlákně. (Kripner, 2024)

4.2 Swing

Swing je knihovna určená pro tvorbu grafického uživatelského rozhraní. Obsahuje mnoho widgetů jako jsou např. tlačítka, textová pole, tabulky a panely. Umožňuje také změnit vzhled widgetů, tak aby vypadaly stejně jako nativní komponenty (např. na Linuxu budou vypadat jinak než na Windows) či aby vypadaly na všech platformách stejně. Knihovna je kompletně napsána v Javě a tím pádem je také nezávislá na platformě. Je také součástí standartní edice Javy, což znamená že ji není nutno instalovat. (Rouse, 2011)

4.3 IntelliJ IDEA

K vývoji aplikace jsem použil vývojové prostředí IntelliJ IDEA Community Edition 2023.

IntelliJ IDEA je vývojové prostředí (IDE) od firmy JetBrains určené pro vývoj softwaru v programovací jazycích Java, Kotlin a dalších. IntelliJ IDEA také obsahuje GUI Designer, který umožňuje vytvářet grafické uživatelské rozhraní (GUI) pro aplikace pomocí komponent knihovny Swing. (JetBrains, 2024)

5 Návrh architektury aplikace

V kapitole je popsán způsob, jakým bude aplikace fungovat.

5.1 Klientská aplikace

Klientská aplikace bude mít konzolové i grafické rozhraní. Konzolová verze aplikace se bude spouštět zadáním názvu programu a parametrů k nimž patří protokol a číslo portu. Po spuštění se uživateli zobrazí, na jakou veřejnou IP adresu a port byl jím zadaný port přesměrován. Grafické rozhraní bude nástavba nad konzolovým rozhraním a funkce budou totožné. V grafickém rozhraní bude uživatel moci zvolit z nabídky protokol, zadat port a pomocí tlačítka tento port zpřístupnit.

5.2 Serverová část

Serverová část bude řešit přiřazování portů uživatelům a samotné přesměrování portu do internetu. Přiřazování portů v serverové části bude fungovat podobně jako např. DHCP server. Klient pošle požadavek na volný port, a pokud již někdy předtím poslal požadavek na port a tento port je volný, bude mu znovu přidělen. Již přidělené porty se budou ukládat v operační paměti programu, takže po vypnutí zmizí.

Komunikace bude probíhat buď pomocí protokolu TCP nebo UDP, podle toho, co si uživatel zvolí v klientské aplikaci. Program bude naslouchat na 2 portech: 7050 pro komunikaci s klientem a 7051 pro TCP. Tyto porty lze změnit pomocí přepínačů konzolové verze nebo v nastavení grafické verze.

Při zvolení portu v klientské aplikaci se program připojí pomocí zvoleného protokolu na server, a ten komunikaci přesměruje na veřejnou IP a doménu se zvoleným portem.

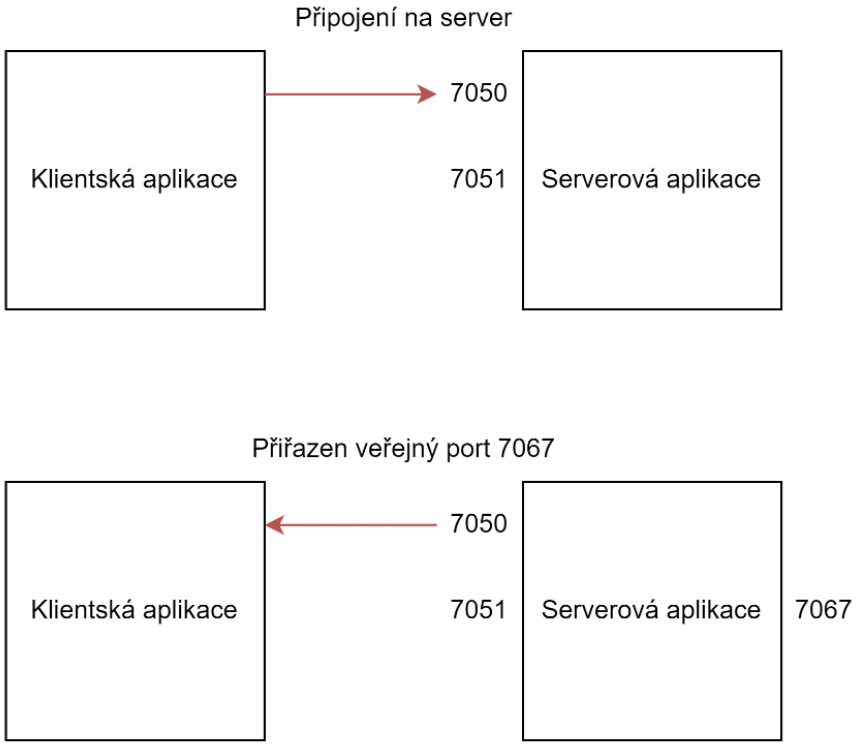
Serverová část bude zprovozněna na virtuálním stroji. Aby však bylo možné se na stroj připojit, bude muset mít přiřazenou veřejnou IP adresu a pro jednoduchost by bylo vhodné, aby také měl doménové jméno.

6 Realizace

V následující kapitole se zaměříme na realizaci samotné aplikace.

6.1 Architektura

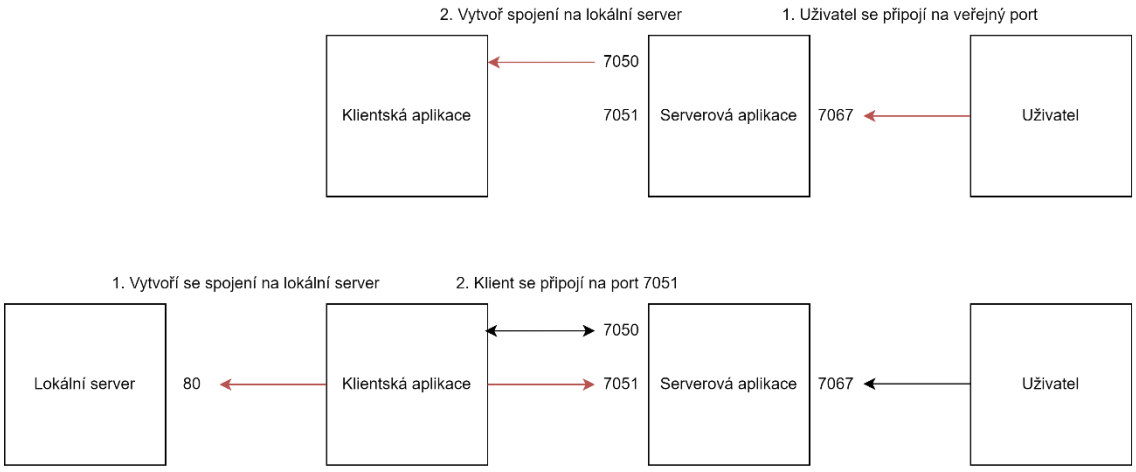
Po spuštění serverové části začne program ve výchozím nastavení naslouchat na portu 7050 pro komunikaci s klientskou aplikací a na portu 7051 pro TCP komunikaci.



Obr. 8: Připojení klienta na server

zdroj: vlastní zpracování

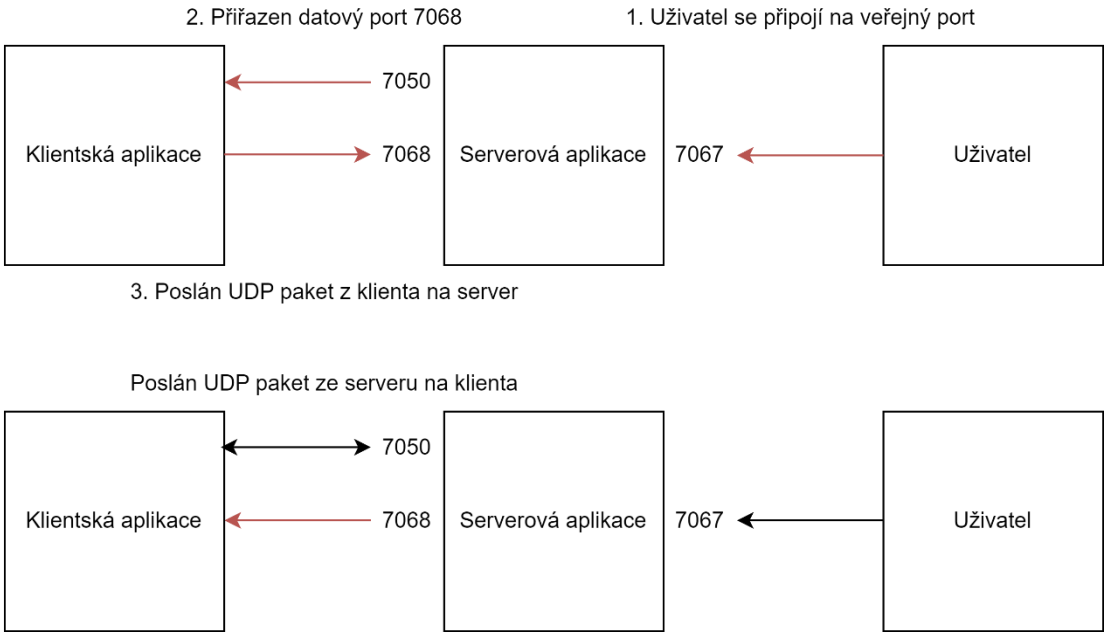
Po zvolení protokolu a portu v klientské aplikaci se program připojí na port 7050 serveru. Poté klient na server pošle název zvoleného protokolu. Server vygeneruje náhodné číslo portu a pošle jej zpět do klientského programu, kde se číslo zobrazí. Poté server začne naslouchat na tomto portu.



Obr. 9: Připojení uživatele na veřejný port protokolem TCP

zdroj: vlastní zpracování

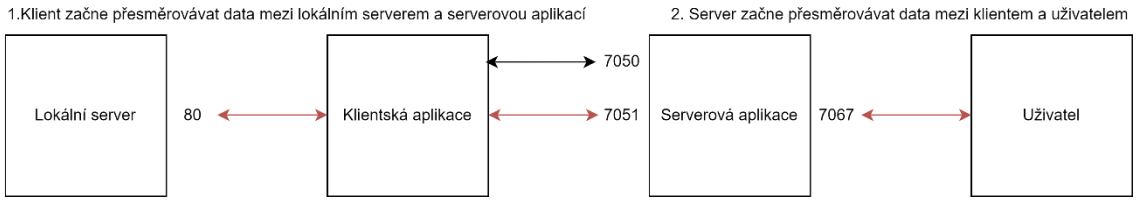
Když si uživatel zvolí protokol TCP a připojí se na veřejný port, server pošle zprávu klientskému programu, aby vytvořil spojení na lokální server. Server poté začne přijímat připojení na portu 7051. Klientský program se připojí na lokální server a poté se připojí na port 7051.



Obr. 10: Připojení uživatele na veřejný port protokolem UDP

zdroj: vlastní zpracování

V případě protokolu UDP se po připojení uživatele na veřejný port vytvoří nový DatagramSocket, který má přiřazený další veřejný náhodný port. Připojením uživatele také server zjistí UDP port uživatele. Poté přes komunikační port pošle číslo veřejného portu klientovi. Klient poté pošle na tento port UDP paket a server ho příjme. Tímto server zjistí UDP port klienta. Server pak pošle UDP paket klientovi, aby bylo možné připojení oběma směry.



Obr. 11: Přesměrování dat mezi uživatelem a lokálním serverem

zdroj: vlastní zpracování

Klient začne poté přesměrovávat data mezi lokálním serverem a serverovou aplikací. Server začne přesměrovávat data mezi připojením od klienta a uživatelem. Takto se uživatel může připojit na lokální server, který nemá veřejnou IP adresu.

6.2 Klient

Klientská aplikace při spuštění nejprve zpracuje argumenty zadané v konzoli. Pokud uživatel zadá špatný název protokolu či port, který není v rozsahu <1-65535>, aplikace ho na tuto skutečnost upozorní a zobrazí nápovědu. Pokud jsou zadané argumenty v pořádku, aplikace se připojí na adresu serveru a odešle na server informaci o tom, jaký protokol si uživatel vybral pomocí zprávy typu SET_PROTOCOL. Poté aplikace čeká na zprávu od serveru. Od serveru můžou přijít následující zprávy, které jsou definované v enumu (číselníku) CommunicationMessage:

- PUBLIC_PORT – Server poslal klientovi veřejný port a klient zobrazí adresu a port uživateli.
- NEW_CONNECTION – Klient po obdržení této zprávy vytvoří nový Socket, který se připojí na lokální server se zadaným portem. Poté vytvoří Socket, kterým se připojí na datový port serveru, který slouží na TCP komunikaci. Nakonec tyto spojení propojí vytvořením instance třídy SocketConnector. V případě protokolu UDP se nejprve vytvoří DatagramSocket, ze kterého se odešle UDP paket na datový port serveru, který přišel ve zprávě. Klient pak čeká na paket od serveru. Poté se propojí lokální server a datový port serveru pomocí třídy SocketConnector.
- NO_PORT_AVAILABLE – Server tímto oznamuje, že již není dostupný žádný volný port. Klientská aplikace tuto zprávu vypíše.

Klient pak znovu čeká na zprávu od serveru do té doby, než se server neodpojí nebo uživatel nevypne klienta.

6.3 Server

Serverová aplikace při spuštění zpracuje argumenty zadané v konzoli. V případě že žádné argumenty nejsou zadané, tak použije výchozí nastavení. Poté vytvoří ServerSocket pro komunikaci s klientem na portu 7050. Dále vytvoří ServerSocket pro datovou komunikaci na portu 7051. Poté v cyklu čeká na připojení klientského programu. Při připojení klienta se vytvoří nová instance třídy ClientProgramSession, která slouží pro obsluhu jednoho připojeni klientské aplikace. Třída Server také obsahuje List typu ClientProgramRecord, který je sdílen mezi instancemi ClientProgramSession.

6.4 ClientProgramRecord

Tato třída slouží k evidenci připojených klientských programů. Eviduje se zde adresa, ze které se klientský program připojil, vybraný protokol, přiřazený veřejný port a jestli je port momentálně aktivní.

6.5 ClientProgramSession

Každá instance třídy ClientProgramSession řeší jeden připojený klientský program a je spuštěna v samostatném vlákně. Při spuštění server čeká na zprávu typu SET_PROTOCOL od klienta, kterou se nastaví protokol. Poté se vygeneruje náhodný veřejný port. Podle zvoleného protokolu se pak začnou přesměrovávat data na veřejný port.

V případě TCP se nejprve vytvoří ServerSocket na vygenerovaném portu. Dále se pak vytvoří instance třídy ServerSocketTerminator, která má za úkol zavřít ServerSocket po odpojení klienta. Server pak pošle klientovi veřejný port pomocí zprávy typu PUBLIC_PORT. Server pak v cyklu čeká na připojení od uživatele na veřejný port. Při připojení uživatele se klientovi pošle zpráva typu NEW_CONNECTION. Server poté čeká na připojení klienta na port určený pro TCP komunikaci. Po připojení klienta se vytvoří instance třídy SocketConnector, která propojí klienta a uživatele připojeného přes veřejný port.

V případě protokolu UDP se vytvoří DatagramSocket na vygenerovaném portu. Server pak pošle klientovi veřejný port pomocí zprávy typu PUBLIC_PORT. Při připojení uživatele na veřejný port se první přijatý paket použije k získání IP adresy a portu uživatele. Poté se vytvoří DatagramSocket na dalším vygenerovaném portu, který bude použit pro přesměrování dat ke klientovi. Klientovi se následně pošle zpráva typu NEW_CONNECTION s tímto portem. Server poté čeká na tomto portu na paket od klienta. Po přijetí paketu server získá IP adresu a port klienta. Server následně odešle paket zpět ke klientovi. Poté se vytvoří instance třídy SocketConnector, která pomocí získaných IP adres a portů propojí klienta a uživatele připojeného přes veřejný port.

6.6 SocketConnector

SocketConnector je třída určená k propojení dvou Socketů či DatagramSocketů oběma směry. To znamená že vstup prvního Socketu je kopírován na výstup druhého Socketu a naopak. Při vytvoření instance této třídy se vytvoří dvě instance třídy ForwardStreamThread. Jedna, která kopíruje data ze vstupu prvního Socketu na výstup druhého Socketu. Druhá instance kopíruje naopak data ze vstupu druhého Socketu na výstup toho prvního. V případě použití protokolu UDP je nutné při vytváření instance této třídy zadat i IP adresy a porty, kam se budou data odesílat.

6.7 ForwardStreamThread

Třída ForwardStreamThread slouží k přesměrování dat ze vstupu jednoho Socketu na výstup druhého Socketu. Třída dědí vlastnosti od třídy Thread. Každá instance této třídy je spuštěna v samostatném vlákně, čímž je umožněno spuštění více instancí zároveň. Po spuštění pomocí metody start() se začnou přesměrovávat data v cyklu.

V případě použití protokolu TCP se nejprve přečtou data ze vstupu prvního Socketu a následně se zapíší na výstup druhého Socketu. Pokud se vyvolá výjimka IOException při zapisování dat, tak to znamená, že se druhý Socket uzavřel. Pokud se naopak tato výjimka vyvolá při čtení dat, tak se uzavřel první Socket. Při zachycení těchto výjimek se nastaví proměnná outputSocketClosed nebo inputSocketClosed na true.

Když je vybrán protokol UDP tak se příjme paket pomocí prvního DatagramSocketu. Tomuto paketu se nastaví cílová IP adresa a port. Paket se následně odešle pomocí druhého DatagramSocketu.

Třída obsahuje metody isInputSocketClosed a isOutputSocketClosed, které vracejí, zda jsou příslušné Sockety uzavřeny.

6.8 Přiřazování veřejných portů

Pokaždé když se klientský program připojí na server, dostane přidělený náhodný port. Všechny aktuálně přidělené porty i ty již neaktivní se ukládají do Listu typu ClientProgramRecord. Každý záznam v Listu eviduje IP adresu klienta, protokol, veřejný port a jestli je port aktivní. Při generování portu se nejprve cyklem projde List portů, a pokud je zde záznam se stejnou IP adresou a zároveň je neaktivní, tak se znovu přiřadí klientovi. Pokud ne, tak se vygeneruje náhodný port v rozmezí nastaveném parametry serverové aplikace, ve výchozím nastavení <7052-7102). Při generování se kontroluje, zda není port již použit, a případně pokud je nalezen neaktivní port tak je záznam smazán a port je použit pro nový záznam. Při odpojení klientské aplikace se port v Listu pouze nastaví jako neaktivní. Pokud by došlo k obsazení všech portů na serveru, aplikace zobrazí upozornění na serveru i klientovi.

6.9 Virtuální stroj

Serverová aplikace je zprovozněna na virtuálním stroji (VPS) s veřejnou IP adresou a doménou, která na tuto IP adresu ukazuje. Jako poskytovatel VPS byl zvolen Oracle Cloud, protože nabízí VPS zdarma. Specifikace použitého VPS jsou následující:

- 0,25 VCPU (architektura x86),
- 1 GB RAM,
- 0,5 Gbit/s rychlost připojení.

Tyto specifikace jsou více než dostačující pro provoz této aplikace. Výhodou je také umístění VPS v Německu, díky čemuž se odezva (ping) pohybuje okolo 40ms.

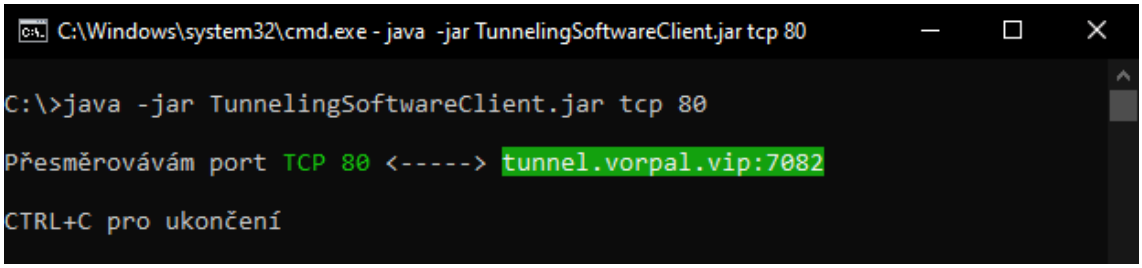
Na tomto virtuálním stroji běží operační systém Ubuntu Server. Aby aplikace fungovala, bylo ještě nutné povolit porty, které aplikace používá, v administraci Oracle Cloudu.

Jako doménu jsem použil již vlastněnou doménu. U mého registrátora domény jsem přidal DNS záznam, který ukazuje na IP adresu vytvořeného virtuálního stroje. Výsledná adresa je: tunnel.vorpal.vip.

6.10 Návod k použití aplikace

V kapitole je popsán způsob, jakým se aplikace používá. Zkompilovaná aplikace je dostupná ve složce „aplikace“ v Příloze A. V této složce je aplikace zkompilovaná pro Windows s architekturou x64. V podsložce „app“ je verze, kterou lze spustit na jiných architekturách a na Linuxu. Pro spuštění multiplatformní verze aplikace je nutné mít nainstalován Java Runtime Environment (JRE) či Java Development Kit (JDK) minimálně verze 17.

6.10.1 Klientská aplikace (konzole)



Obr. 12: Klientská aplikace
zdroj: vlastní zpracování

Po spuštění klientské aplikace se začne přesměrovávat vybraný protokol a port na veřejnou adresu, která se zobrazí vpravo. Pomocí parametrů lze nastavit adresu i port serveru na který se program připojuje viz dále. Ve výchozím nastavení je adresa přednastavena na adresu virtuální stroje. Pro přesměrování na lokálně spuštěnou serverovou část je nutné nastavit jako adresu serveru localhost a port 7050.

Program lze spustit zadáním následujícího příkazu do příkazového řádku do Windows či do terminálu v Linuxu za předpokladu, že se nacházíme ve stejné složce jako je program.

Windows:

TunnelingSoftwareClient protokol port [volitelné přepínače]

Linux:

java -jar TunnelingSoftwareClient.jar protokol port [volitelné přepínače]

protokol – TCP nebo UDP

port – port lokálního serveru

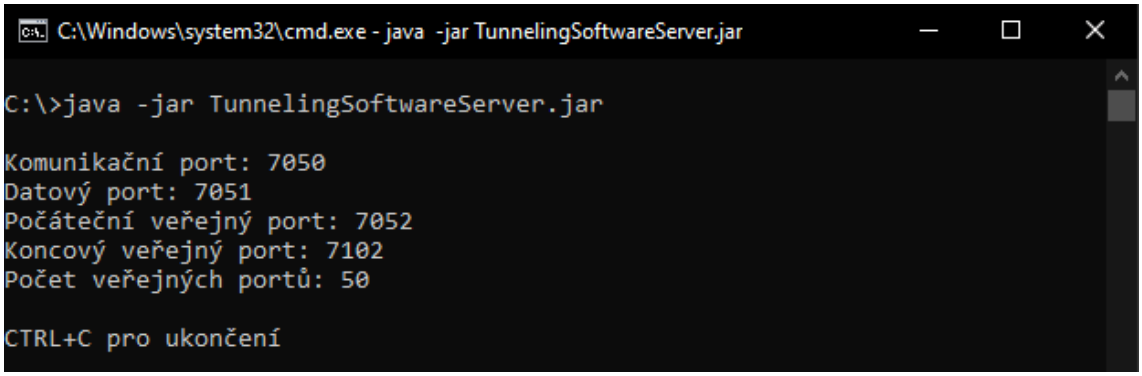
Volitelné přepínače:

-s adresa – adresa serveru

-p port – port serveru

-h – zobrazí nápovědu

6.10.2 Serverová aplikace (konzole)



Obr. 13: Serverová aplikace

zdroj: vlastní zpracování

Serverová aplikace po spuštění načte výchozí porty a čeká na připojení od klientské aplikace. Porty lze změnit pomocí parametrů viz níže.

Program lze spustit zadání následujícího příkazu do příkazového řádku do Windows či do terminálu v Linuxu.

Windows:

TunnelingSoftwareServer [volitelné přepínače]

Linux:

java -jar TunnelingSoftwareServer.jar [volitelné přepínače]

Volitelné přepínače:

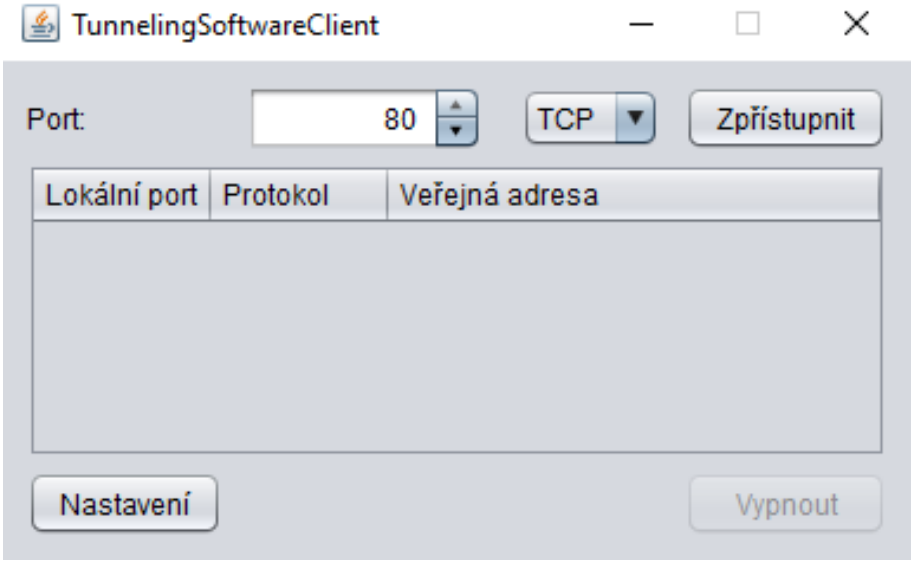
-p port – port na kterém bude server naslouchat

-pp port – počáteční veřejný port

-c počet – počet veřejných portů

-h – zobrazí nápovědu

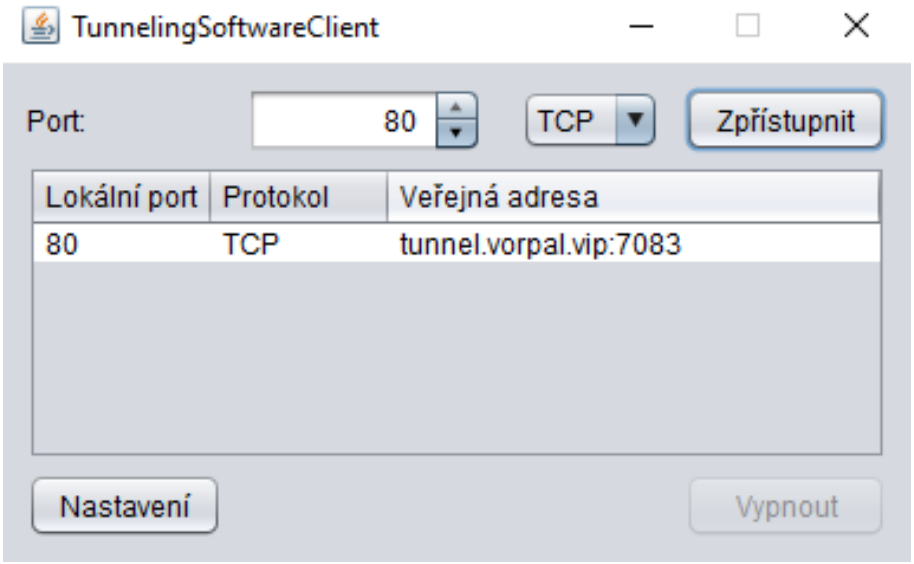
6.10.3 Klientská aplikace (grafické rozhraní)



Obr. 14: Grafické rozhraní klientské aplikace

zdroj: vlastní zpracování

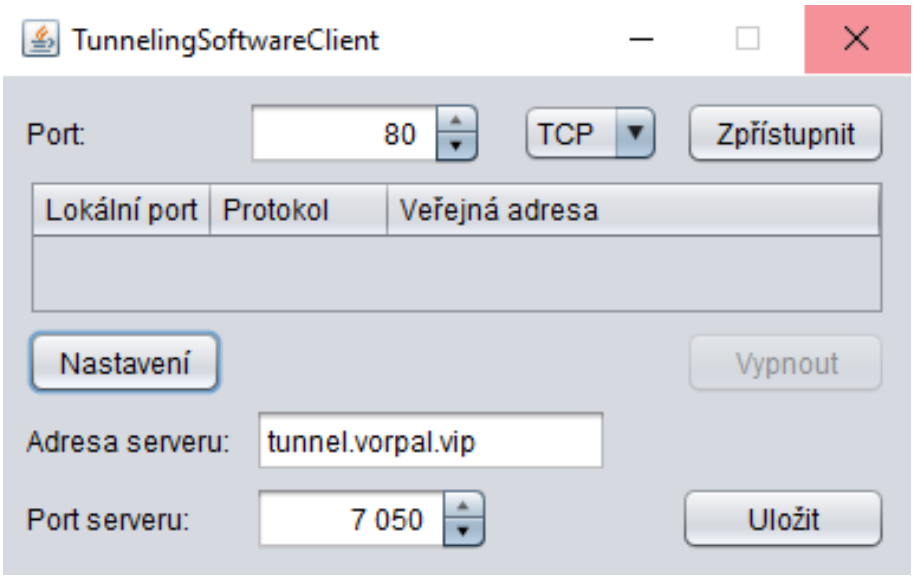
Po spuštění aplikace si lze zvolit port a protokol, který bude přesměrován.



Obr. 15: Zpřístupnění portu v klientské aplikaci

zdroj: vlastní zpracování

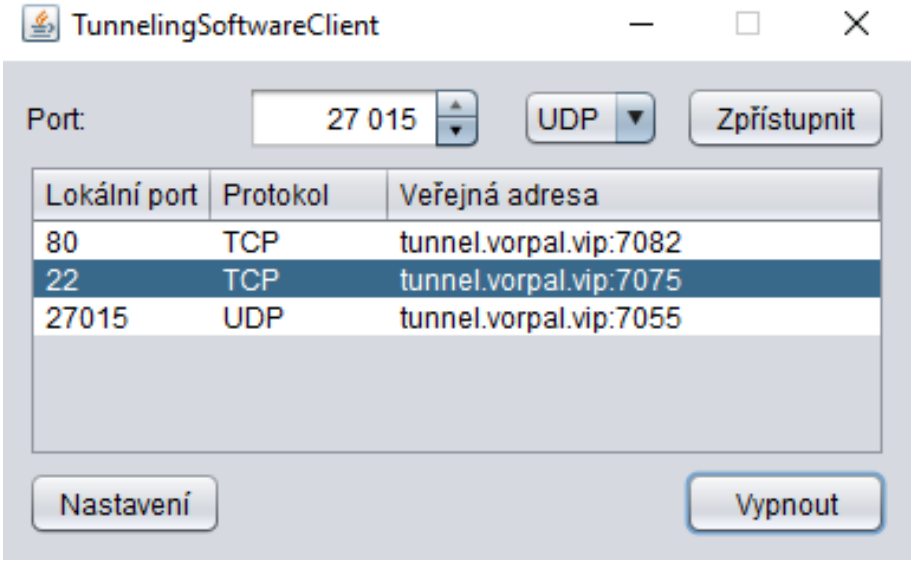
Po kliknutí na tlačítko Zpřístupnit se pak v tabulce zobrazí veřejná adresa zpřístupněného portu. Adresu je možné zkopírovat do schránky dvojitým kliknutím myši nebo kliknutím pravým tlačítkem myši a zvolením možnosti Kopírovat.



Obr. 16: Nastavení klientské aplikace

zdroj: vlastní zpracování

Do nastavení aplikace se lze dostat kliknutím na tlačítko Nastavení. Zde je možné nastavit adresu serveru a port. Kliknutím na tlačítko Uložit se nastavení uloží do souboru a nastavení se schová.

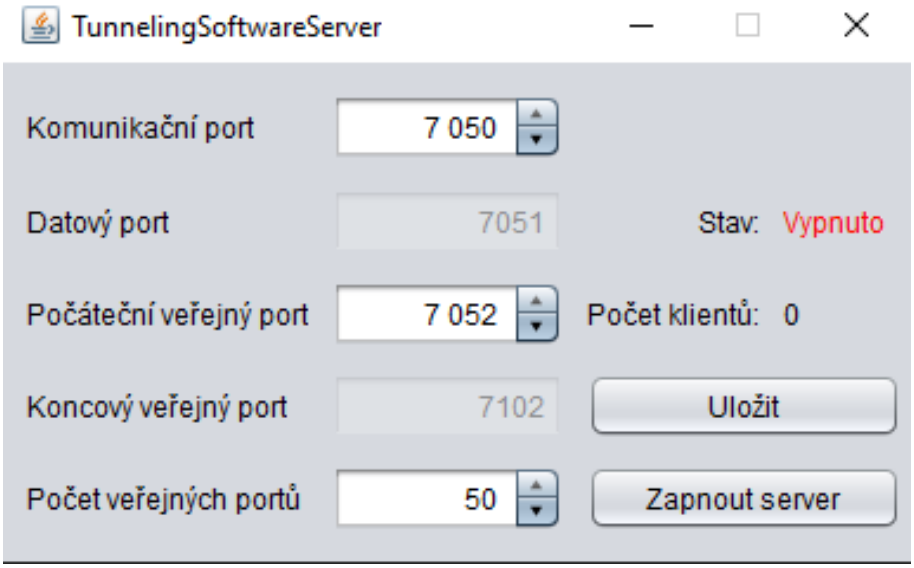


Obr. 17: Zpřístupnění více portů v klientské aplikaci

zdroj: vlastní zpracování

V grafickém rozhraní klienta je také možné zpřístupnit více portů souběžně. Zpřístupnění portu lze vypnout výběrem v tabulce a kliknutím na tlačítko Vypnout.

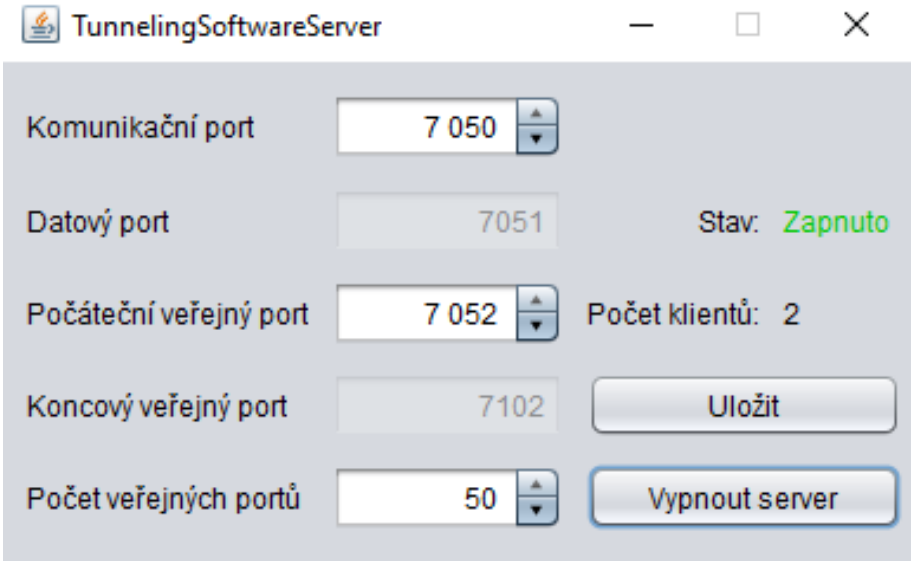
6.10.4 Serverová aplikace (grafické rozhraní)



Obr. 18: Grafické rozhraní serverové aplikace

zdroj: vlastní zpracování

V serverové aplikaci je možné změnit komunikační port, počáteční veřejný port a počet veřejných portů. Datový port a koncový port se automaticky dopočítá podle dalších zadaných hodnot. Kliknutím na tlačítko Uložit se nastavení uloží do souboru.



Obr. 19: Spuštění serveru v serverové aplikaci

zdroj: vlastní zpracování

Kliknutím na tlačítko Zapnout server se server zapne. Stejným tlačítkem pak lze server i vypnout. Aplikace také zobrazuje počet připojených klientů na server.

6.11 Porovnání s podobnými aplikacemi

Programy ngrok a LocaltoNet vyžadují k funkčnosti registraci a některé jejich funkce jsou zpoplatněné. Nejblíže k mé aplikaci má program frp, protože je zdarma a má klientskou a serverovou část.

V programu frp lze nastavit na jaký veřejný port se má daný lokální port přesměrovat. V mé aplikaci se přiřadí náhodný veřejný port, ale pokud se klient v minulosti připojil, tak se použije stejný port. Dále frp umožňuje přesměrovat více portů zároveň, což v mé aplikaci lze pouze v grafickém rozhraní. Frp oproti mé aplikaci podporuje i protokol HTTPS.

Program frp nabízí velkou spoustu konfiguračních možností, ale je trochu složitější na nastavení, protože se všechno nastavuje v konfiguračních souborech. Což může být na druhou stranu výhodou, pokud chceme program spustit znovu se stejným nastavením. V mé aplikaci se vše dá nastavit v grafickém rozhraní či pomocí parametrů v konzoli, ale pokaždé se musí zadávat jaký port chceme zpřístupnit. Má aplikace se tudíž spíše hodí pro rychlé jednorázové zpřístupnění portu. Program frp se zase spíše hodí na opakované použití.

Závěr

Cílem práce bylo vytvoření aplikace pro zpřístupnění lokálního portu do internetu. V teoretické části se zabýváme různými způsoby zpřístupnění lokálního portu do internetu, jakým způsobem fungují a jaké je jejich využití. Také je zde popsáno, jak fungují protokoly TCP a UDP. Následně jsou zde zmíněny podobné aplikace. Jsou zde porovnány jejich výhody, nevýhody a je zde popsána jejich funkčnost. Dále jsou zde vypsány technologie použité při vývoji aplikace. Poslední část teoretické části se zabývá popisem toho, jakým způsobem by měli fungovat jednotlivé části aplikace. V praktické části je popsána architektura aplikace a jednotlivé třídy v kódu. Dále je zde zmíněn virtuální stroj, na kterém běží serverová část. V další kapitole je popsáno, jak se aplikace používá. Poslední část se zabývá porovnáním podobných aplikací s mým řešením.

Výsledem jsou dvě aplikace: klientská a serverová. Klientská aplikace umožňuje uživateli přesměrovat lokální port do internetu bez nutnosti vlastnit veřejnou IP adresu. Serverová aplikace, která běží na virtuálním stroji, zprostředkovává přiřazování veřejných portů klientské aplikaci. Aplikace lze spustit na operačních systémech Windows a Linux. Nad rámec zadání se mi podařilo vytvořit i grafické rozhraní pro obě aplikace.

Jedním z problémů při řešení bylo samotné propojení lokálního serveru s uživatelem připojeným na veřejný port. Nejprve jsem zkoušel použít pouze jedno připojení mezi klientem a serverem, ale ukázalo se, že lepším řešením je pro každého uživatele, který se připojí na veřejný port, vytvořit nové připojení z klienta na server.

Největším problémem však bylo přesměrování UDP protokolu. Dlouho se mi nedařilo implementovat techniku UDP hole punching, než jsem přišel na to, že je nutné, aby se nejprve poslal alespoň jeden paket z klienta na server a jeden paket ze serveru na klienta. Bez tohoto kroku by nefungovala obousměrná komunikace, pokud je klient schovaný za NATem.

Na aplikaci by se dalo vylepšit grafické rozhraní, stávající je velmi jednoduché. Také by se dalo přidat ukládání nastavení pro konzolovou verzi aplikace, momentálně je to možné pouze v té grafické. Co se týče přesměrování protokolu UDP, tak by bylo vhodné implementovat připojení více uživatelů na jeden veřejný port. Momentálně se na jeden veřejný UDP port může připojit pouze 1 uživatel.

Seznam použité literatury

APACHE. Apache Module mod_proxy. *The Apache HTTP Server Project* [online]. 2013 [cit. 2023-12-12]. Dostupné z: https://httpd.apache.org/docs/2.0/mod/mod_proxy.html#forwardreverse

DIXON, Trevor. Serveo: expose local servers to the internet using SSH. *Serveo* [online]. 2023 [cit. 2023-12-09]. Dostupné z: https://serveo.net/

DSL.CZ. SSH připojení k počítači za NATem aneb Stavíme reverzní tunel. *DSL.cz* [online]. 2023 [cit. 2023-11-25]. Dostupné z: https://www.dsl.cz/jak-na-to/jak-na-reverzni-ssh-tunel

FATEDIER. frp. *GitHub* [online]. 2023 [cit. 2023-11-23]. Dostupné z: https://github.com/fatedier/frp

FISHER, Tim. How to Set Up Port Forwarding. *Lifewire* [online]. 2022 [cit. 2023-11-30]. Dostupné z: https://www.lifewire.com/how-to-port-forward-4163829

GIANCHANDANI, Prateek. Circumventing NAT with UDP hole punching. *Infosec* [online]. 2012 [cit. 2024-04-07]. Dostupné z: https://www.infosecinstitute.com/resources/hacking/udp-hole-punching/

JETBRAINS. GUI Designer Basics. *IntelliJ IDEA Documentation* [online]. 2024 [cit. 2024-03-12]. Dostupné z: https://www.jetbrains.com/help/idea/gui-designer-basics.html

KHAN ACADEMY. Transmission Control Protocol (TCP). *Khan Academy* [online]. 2024 [cit. 2024-01-05]. Dostupné z: https://en.khanacademy.org/computing/computers-and-internet/xcae6f4a7ff015e7d:the-internet/xcae6f4a7ff015e7d:transporting-packets/a/transmission-control-protocol--tcp

KHAN ACADEMY. User Datagram Protocol (UDP). *Khan Academy* [online]. 2024 [cit. 2024-01-05]. Dostupné z: https://en.khanacademy.org/computing/computers-and-internet/xcae6f4a7ff015e7d:the-internet/xcae6f4a7ff015e7d:transporting-packets/a/user-datagram-protocol-udp

KRIPNER, Matěj. Lekce 1 - Multithreading v Javě. *Itnetwork.cz* [online]. 2024 [cit. 2024-03-12]. Dostupné z: https://www.itnetwork.cz/java/vlakna/multithreading-v-jave

KUTÁČ, Pavel. Z internetu na lokální server pomocí ngrok. *Kutáč.cz* [online]. 2018 [cit. 2023-11-23]. Dostupné z: https://www.kutac.cz/weby-a-vse-okolo/z-internetu-na-lokalni-server-pomoci-ngrok

LOCALTONET. LocaltoNet | Localhost to Internet. *LocaltoNet* [online]. 2023 [cit. 2023-11-26]. Dostupné z: https://localtonet.com/#pricing

MAJER, Martin. Síťování v Javě: Úvod. *Root.cz* [online]. 2006 [cit. 2024-03-12]. Dostupné z: https://www.root.cz/clanky/sitovani-v-jave-uvod/

MICROSOFT. Co je Java? Příručka pro začátečníky pro jazyk Java. *Microsoft Azure* [online]. 2024 [cit. 2024-01-05]. Dostupné z: https://azure.microsoft.com/cs-cz/resources/cloud-computing-dictionary/what-is-java-programming-language/

NGROK. ngrok pricing. *ngrok* [online]. 2023 [cit. 2023-11-23]. Dostupné z: https://ngrok.com/pricing

ORACLE. DatagramSocket (Java Platform SE 8). *Java™ Platform, Standard Edition 8 API Specification* [online]. 2024 [cit. 2024-03-12]. Dostupné z: https://docs.oracle.com/javase/8/docs/api/java/net/DatagramSocket.html

ORACLE. ServerSocket (Java Platform SE 8). *Java™ Platform, Standard Edition 8 API Specification* [online]. 2024 [cit. 2024-03-12]. Dostupné z: https://docs.oracle.com/javase/8/docs/api/java/net/ServerSocket.html

PROFIBER NETWORKING. TCP VS UDP. *PROFIBER Networking* [online]. 2023 [cit. 2023-11-25]. Dostupné z: https://www.profiber.eu/cz/aplikace/tcp-vs-udp/

ROUSE, Margaret. What is Java Swing? - Definition from Techopedia. *Techopedia* [online]. 2011 [cit. 2024-01-04]. Dostupné z: https://www.techopedia.com/definition/26102/java-swing

TANNER, Gabriel. Exposing your local server to the internet over NAT using FRP. *Gabriel Tanner* [online]. 2020 [cit. 2023-11-23]. Dostupné z: https://gabrieltanner.org/blog/port-forwarding-frp/

TRACKET. SSH Tunnel - Local and Remote Port Forwarding Explained With Examples. *Trackets Blog* [online]. 2014 [cit. 2023-11-25]. Dostupné z: https://blog.trackets.com/2014/05/17/ssh-tunnel-local-and-remote-port-forwarding-explained-with-examples.html

Přílohy

Příloha A

Příloha A obsahuje zkompilovanou aplikaci a zdrojový kód aplikace.