Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Aplikace pro podporu výuky NAT a NAT Traversal

2023

Jakub Mazur

Vedoucí práce:
doc. Mgr. Jan Outrata, Ph.D.

Studijní program: Aplikovaná informatika,
prezenční forma

Bibliografické údaje

Autor:	Jakub Mazur
Název práce:	Aplikace pro podporu výuky NAT a NAT Traversal
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2023
Studijní program:	Aplikovaná informatika, prezenční forma
Vedoucí práce:	doc. Mgr. Jan Outrata, Ph.D.
Počet stran:	41
Přílohy:	elektronická data v úložišti katedry informatiky
Jazyk práce:	český

Bibliographic info

Author:	Jakub Mazur
Title:	Application to support teaching of NAT and NAT Traversal
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2023
Study program:	Applied Computer Science, full-time form
Supervisor:	doc. Mgr. Jan Outrata, Ph.D.
Page count:	41
Supplements:	electronic data in the storage of department of computer science
Thesis language:	Czech

Anotace

Bakalářská práce se zaměřuje na přehled aktuálních metod obcházení překladu síťových adres, tzv. NAT Traversal a demonstrace vybraných metod v praktické části. V práci je vysvětlen základní princip překládání síťových adres, problémy, které přináší a jejich následné řešení pomocí NAT Traversal. Výsledná aplikace umožňuje vybraná řešení pomocí metod NAT Traversal otestovat v reálném síťovém prostředí.

Synopsis

The thesis focuses on overview of current network address translation traversal methods and demonstration of selected methods in practical part. The thesis explains the principle of network address translation, the problems it brings and their subsequent solution using NAT Traversal. The resulting application makes the selected solutions to be tested using NAT Traversal methods possible in a real network environment.

Klíčová slova: překlad síťových adres; obcházení NAT; Směrování portů; Děrování NAT; Windows Forms

Keywords: NAT; NAT Traversal; Port Mapping; NAT Hole Punching; Windows Forms

Děkuji panu doc. Janu Outratovi, Ph.D. za vedení, čas strávený u konzultací a objektivní rady při tvorbě práce. Dále děkuji panu kpt. Ing. Martinu Frýdlovi za podporu a cenné rady.

Obsah

Seznam obrázků

Seznam zdrojových kódů

1 Úvod

Každé zařízení, které se chce připojit k internetu, potřebuje svou veřejnou IP adresu. IP adresa může nabývat pouze omezených hodnot, a proto se pomalu vyčerpávají. Tento problém nebyl patrný na začátku Internetu, ale projevil se až později.

Řešením je překlad síťových adres (Network Address Translation, dále pouze jako NAT). NAT umožňuje pro více zařízení v privátní síti přístup k internetu pomocí jedné veřejné IP adresy. Jelikož jsou zařízení za NATem, tak nemají přímý přístup k Internetu a je složité navázat přímé spojení s jiným zařízením za NATem. Přímé spojení vyžadují např. VoIP služby jako je Skype nebo aplikace vyžadující peer-to-peer spojení. Pro navázání přímého spojení dvou zařízení na NATem je potřeba NAT obejít. K tomu slouží metody NAT Traversalu, které si představíme v průběhu práce.

Metody NAT Traversalu v dnešní době využívá mnoho služeb na internetu. Problém je v tom, že technologie NAT nemá standard a kvůli tomu jsou metody NAT Traversal mnohdy špatně zdokumentované a closed source (software s uzavřeným kódem). Proto se v této práci pokusím čtenáři problematiku NAT Traversal přiblížit a srozumitelně vysvětlit, jak jednotlivé metody fungují. Vybrané metody je možné demonstrovat v praktické části, která k demonstraci využívá reálné síťové prostředí.

2 NAT

NAT je v dnešní době důležitou vlastností směrovačů a firewallů, jelikož pomáhá zachovat omezené množství veřejných IPv4 adres a zajišťuje částečnou bezpečnost komunikace mezi privátní sítí a Internetem.

2.1 Využití

Když byla vytvořena IPv4 adresa a zavedena jako standard v roce 1982, nikdo si neuvědomoval, jak velký internet jednou bude. Přestože jsou k dispozici přes 4 miliardy (2^{32}) IPv4 adres, tak posledních 30 let je velké téma jejich nedostatek a náhrada. Aby se nedostatku zabránilo, byly vytvořeny soukromé IP adresy a jejich překlad (NAT). Existují 2 druhy IPv4 adres:

1. **Veřejné** - veřejně zaregistrované na internetu, jsou potřeba k připojení k internetu

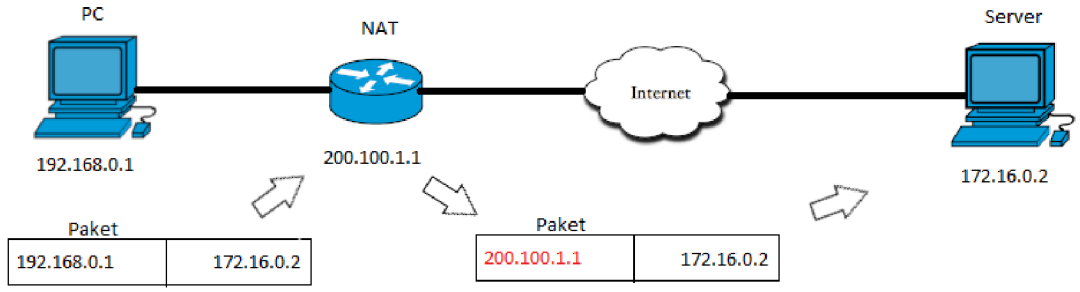
2. **Privátní** - nejsou veřejně zaregistrované, jsou využívány pouze interně (např. v domácnosti nebo ve firmě), nelze se pomocí ní připojit k internetu

Privátní IP adresu přiřazuje jednotlivým zařízením směrovač. Většina domácností nemá pouze jedno zařízení, které potřebuje přístup k internetu, ale několik. Tato zařízení pořebují veřejnou IP adresu, aby se mohla připojit k internetu. Samozřejmě je možnost požádat poskytovatele internetového připojení, aby přiřadil každému zařízení svou vlastní veřejnou IP adresu. To by bylo zbytečné, drahé a vedlo by to k rychlému vyčerpání IPv4 adres.

Místo toho necháme směrovač, aby přiřadil jednotlivému zařízení v domácnosti privátní IP adresu. Pokud budou zařízení potřebovat přístup k internetu, jejich privátní IP adresy budou pomocí NAT přeloženy na jednu veřejnou IP adresu. To zajistí, že se nevyčerpají všechny IPv4 adresy a zařízení z veřejné sítě nemůže zahájit přímé spojení se zařízením v privátní síti. V budoucnu bychom NAT ani privátní IP adresy neměli potřebovat, protože IPv4 adresy by měly být nahrazeny novou generací IPv6 adres.

2.2 Překlad adres

Základní mechanismus NAT je popsán v dokumentu RFC 1631. Mechanismus překladu NAT převádí IP adresu v paketu během jeho průchodu směrovačem. Směrovač s podporou NAT má dvě síťová rozhraní. Jedno je připojeno k privátní síti a druhé k veřejné síti. Na obrázku 1 je znázorněn překlad adres u statického NATu.



Obrázek 1: Překlad adres ve statickém NATu

Host připojený do privátní sítě odesílá packety na server. V tomto případě musí směrovač s NAT převést zdrojovou privátní IP adresu 192.168.0.1 na veřejnou IP adresu 200.100.1.1. Host sice odeslal paket se svou zdrojovou adresou, ta je ale privátní a je potřeba ji převést na registrovanou veřejnou IP adresu. Server příjme paket a myslí si, že komunikuje s hostem 200.100.1.1. Odešle zpět paket s cílovou adresou 200.100.1.1. Tu pak směrovač s NAT převede zpátky na vnitřní IP adresu 192.168.0.1 a odešle hostovi.

Na předchozím obrázku 1 lze popsat 3 typy adres v NAT [1]:

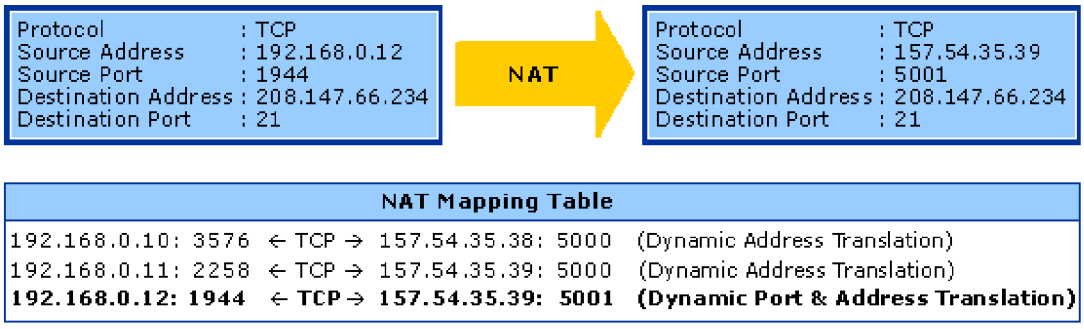
1. **Vnitřní privátní** - IP adresa, která se nachází se uvnitř vnitřní sítě (192.168.0.1)
2. **Vnitřní globální** - adresa, pod kterou jsou zařízení ve vnitřní síti vidět z vnější sítě (200.100.1.1)
3. **Vnější privátní** - privátní IP adresy zařízení ve vnější síti, jak jsou vidět ve vnitřní síti (172.16.0.2)

2.3 NAT tabulka

NAT tabulka je základním prvkem překladu adres, které se odehrává na směrovači, když pakety přicházejí a opouštějí jeho rozhraní. Každé připojení z vnitřní sítě do vnější a naopak je sledováno a je vytvořena speciální tabulka, která má směrovači pomoci určit, co má dělat se všemi příchozími pakety.

2.3.1 Obsah NAT tabulky

Tabulka NAT obsahuje seznam aktivních připojení a jejich přiřazených překladů mezi privátními IP adresami a veřejnými IP adresami. Každá položka v tabulce obvykle obsahuje zdrojovou a cílovou IP adresu, čísla portů a překladové pravidlo používané k mapování privátní IP adresy na veřejnou IP adresu. Překladové pravidlo může také obsahovat další informace, jako je používaný protokol (např. TCP nebo UDP), čas zbývající do platnosti překladu a počet paketů, které byly dosud přeloženy. Tabulka NAT může také obsahovat další informace, jako je čas poslední aktivity pro dané připojení, které lze použít k odstranění neaktivních připojení z tabulky.



Obrázek 2: Příklad NAT Tabulky a překladu [2]

2.3.2 Bezpečnostní rizika

Při nesprávném nastavení směrovače a jeho NAT tabulky může dojít k několika rizikům.

Jedno z nich může nastat, pokud je NAT tabulka příliš malá nebo neodstraňuje včas neaktivní záznamy. To může způsobit chybu v překladu adresy, což by znemožnilo některým zařízením v privátní síti komunikovat s vnější sítí a naopak. Tato chyba by nastala, pokud směrovači (který kromě adresy překládá i porty) dojdou volné porty nebo volné místo v NAT tabulce.

Další riziko nastává, pokud je NAT tabulka naopak příliš velká. V tomto případě může útočník zahltit směrovač nechtěnými pakety, čímž by donutil směrovač využít většinu své paměti na nepotřebné záznamy v NAT tabulce. Problém by ovšem nebyl v paměti směrovače, jelikož jeden záznam zabírá zhruba 160 bytů, ale v procesoru směrovače, který by při takovém útoku nestíhal spravovat všechny záznamy.[3]

2.4 Typy NATu

2.4.1 Statický NAT (one-to-one)

Na směrovači, kde probíhá statický NAT, se překládá jedna IP adresa vždy na stejnou IP adresu. Každá vnitřní privátní adresa je vždy pevně namapovaná na stejnou vnější globální IP adresu. Adresy se mapují 1:1, a tedy nedochází k úsporám veřejných IPv4 adres. Statický NAT je výhodný, pokud chceme dát nějaký server veřejně k dispozici na Internetu, jelikož tento server bude mít stále stejnou veřejnou IP adresu. Protože je NAT tabulka statická, při každé změně je potřeba ji ručně upravit.[4] Statický NAT může být i one-to-many, ale zde nastává problém se zpětným překladem a nepoužívá se, proto je Statický NAT často nazýván one-to-one.

2.4.2 Dynamický NAT (one-to-many)

Většina směrovačů s NAT mapuje více zařízení v privátní síti na menší množinu veřejných IP adres. K mapování se využívá množina vnitřních globálních a vnitřních privátních adres a mapování dvojic probíhá dynamicky podle potřeby. [1] Když zařízení zahájí spojení se zařízením ve vnější síti, je mu přiřazena dostupná veřejná IP adresa z množiny. Mapování mezi vnitřní globální a vnitřní privátní adresou se po ukončení spojení zruší. Výhodou je zpřístupnění vnější sítě mnoha zařízením přes několik globálních adres. Princip dynamického přiřazování adres:

- Zařízení **A** ve vnitřní síti se chce spojit se serverem ve vnější síti.
- Směrovač s NAT dostane paket.
- Směrovač pak nahradí IP adresu v paketu podle NAT tabulky z vnitřní privátní adresy na vnitřní globální adresu. Pokud v tabulce neexistuje záznam, přidělí mu dostupnou vniřní globální adresu. Paket pošle na cílovou adresu.
- Jakmile dostane směrovač odpověď, zkontroluje cílovou adresu. Podívá se do NAT tabulky a zjistí, kterému zařízení cílová adresa patří.
- Tato vnitřní globální adresa v paketu se vymění za vnitřní privátní a paket se pošle do vnitřní sítě cílovému zařízení **A**.
- Tento proces se opakuje, dokud zařízení **A** komunikuje se serverem. Po určité době se záznam v NAT tabulce smaže.
- Pokud by se nyní chtělo připojit k serveru zařízení **B**, tak dostane stejnou vnitřní globální adresu, jako zařízení **A**.

Díky dynamickému přiřazování adres dosáhneme částečné úspory IPv4 adres.

2.4.3 Přetížený NAT (NAT overloading, Port Address Translation)

Přetížený NAT mapuje více vnitřních privátních adres na jednu vnitřní globální adresu na různých portech. To umožňuje více zařízením využití jedné veřejné IP adresy. Přetížený NAT rozšiřuje NAT tabulku o **vnitřní privátní port** a **vnitřní globální port**. Princip přetíženého NATu:

- Zařízení **A** ve vnitřní síti se chce spojit se serverem ve vnější síti.
- Směrovač s NAT dostane paket.
- Směrovač s NAT nahradí IP adresu a port v paketu podle NAT tabulky z vnitřní privátní na vnitřní globální. Paket potom odešlě na cílovou adresu.
- Jakmile dostane směrovač odpověď, zkontroluje cílovou adresu a port. Podívá se do NAT tabulky a zjistí, kterému zařízení cílová adresa s portem patří.
- Vnitřní globální adresa a port v paketu se vymění za vnitřní privátní a paket se pošle do vnitřní sítě cílovému zařízení **A**.
- Tento proces se opakuje, dokud zařízení **A** komunikuje se serverem. Po určité době se záznam v NAT tabulce smaže.
- Pokud by se kdykoliv během předchozího procesu chtělo připojit k serveru zařízení **B**, tak dostane stejnou vnitřní globální adresu, jako zařízení **A**, ale jiný vnitřní globální port.

Přetížený NAT přináší výše zmiňované velké úspory IPv4 adres.

2.4.4 Source NAT (SNAT)

Source NAT se nejčastěji používá k překladu privátní IP adresy na veřejnou směrovatelnou adresu pro komunikaci se zařízením ve vnější síti. Source NAT mění zdrojovou IP adresu paketů, které odcházejí směrovačem ven a cílovou IP adresu paketů odpovědi, které přicházejí směrovačem. Dále může měnit číslo UDP nebo TCP portu v paketu. [5] Source NAT je využit v kapitolách 2.4.2 a 2.4.3 u popisu principů jednotlivých NATů. Source NATu se za určitých podmínek říká maškaráda.

Maškaráda

Maškaráda je specifická implementace nebo konfigurace Source NATu, kdy veřejná IP adresa v okamžiku přidání pravidla do NAT tabulky není známá nebo se veřejná IP adresa dynamicky mění. Maškaráda obvykle zahrnuje dynamické mapování několika privátních IP adres na jednu primární IP adresu odchozího rozhraní. Tato primární IP adresa musí být předem definována rozhraním v systému

směrovače. V tomto případě směrovač maskuje soukromou IP adresu zařízení IP adresou rozhraní, přes které je paket odesílán.

Výhoda maškarády je v tom, že zařízení ve vnější síti nemohou zahájit spojení se zařízením ve vnitřní síti, které je za maškarádou. Díky tomu tato vnitřní síť získá dodatečnou ochranu před útokem z vnější sítě.[6]

2.4.5 Destination NAT (DNAT)

Destination NAT se používá především k přesměrování příchozích paketů s vnější IP adresou nebo cílovým portem na vnitřní IP adresu nebo port uvnitř sítě. Jedná se tedy o opačný princip jak u Source NATu. Může také změnit cílový UDP nebo TCP port. Princip Destination NATu je podobný jako u Port Forwardingu, který je popsaný v kapitole 3.4, ale zařízení ve vnější síti musí navíc od přesměrovaného portu znát i přesměrovanou IP adresu. [7]

Hlavním účelem Destination NATu je umožnit zařízením ve vnější síti přístup k určitým službám nebo serverům umístěným ve vnitřní síti. Konfigurací pravidel lze provoz přesměrovat na příslušnou privátní IP adresu a port, což umožňuje, aby služby, jako jsou webové servery nebo servery FTP, byly přístupné z internetu pomocí veřejné IP adresy.

2.5 Implementace NATu

Překlad síťových adres a portů lze implementovat několika způsoby a to například tradičně, obousměrně nebo dvojitě. Některé aplikace, které používají informace o IP adrese, mohou potřebovat určit vnitřní globální IP adresu.

2.5.1 Tradiční NAT

Tradiční NAT se především používá pro umožnění komunikace v privátních sítích. Existují dvě varianty tradičního NATu, a to Basic NAT a NAPT (Network Address Port Translation). [4]

Základní NAT

Základní NAT je implementace, která překládá nebo mapuje adresy z jednoho bloku IP adres do druhého. Při použití základního NATu je blok externích adres vyhrazen pro překlad adres hostitelů ve vnitřní síti, kteří vytvářejí relace do vnější sítě.

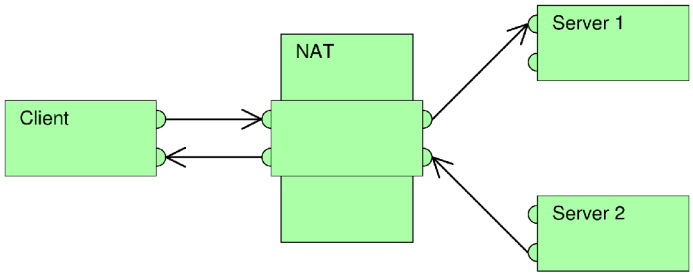
NAPT (Překlad síťových adres portů)

NAPT je technika, při které se čísla TCP a UDP portů a privátní IP adresy mapují z více vnitřních hostitelů na jednu veřejnou IP adresu. Jedná se o typ implementace, která rozšiřuje možnosti tím, že při komunikaci s vnější sítí překládá a mapuje kromě IP adresy také čísla TCP a UDP portů.

Tradiční NAT můžeme dále rozdělit podle typu překládání a mapování vnitřní globální adresy. Dokument RFC 3489 tyto typy dělí na 4 základní implementace [8]:

Full cone NAT

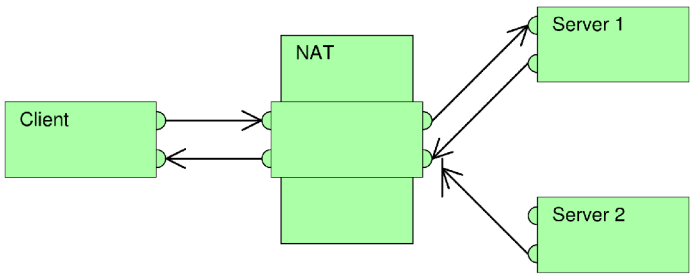
Full cone NAT je implementace, kde veškerá komunikace ze stejné vnitřní privátní IP adresy a portu je namapována na právě jednu stejnou vnitřní globální IP adresu a port (independent filtering). Jakékoliv zařízení ve vnější síti může kontaktovat zařízení ve vnitřní sítí odesláním paketu na mapovanou vnitnřní globální IP adresu a port, jak je znázorněno na obrázku 3. **Server 2** zná vnitnřní globální IP adresu a port, tudíž mu nic nebrání v navázání spojení s **klientem**.



Obrázek 3: Full cone NAT[9]

Restricted cone NAT

Restricted cone NAT je implementace, kde veškerá komunikace ze stejné vnitřní privátní IP adresy a portu je namapována na právě jednu stejnou vnitřní globální IP adresu a port (dependent filtering). Na obrázku 4 je rozdíl od Full-cone NATu na první pohled jasný. **Klienta** ve vnitřní síti mohou kontaktovat pouze taková vnější zařízení, která někdy v minulosti byla kontaktována vnitřní globální IP adresou zařízení. Tedy v případě obrázku 4 může **klienta** kontaktovat pouze **server 1** a příchozí pakety ze **serveru 2** jsou blokovány směrovačem s NAT.



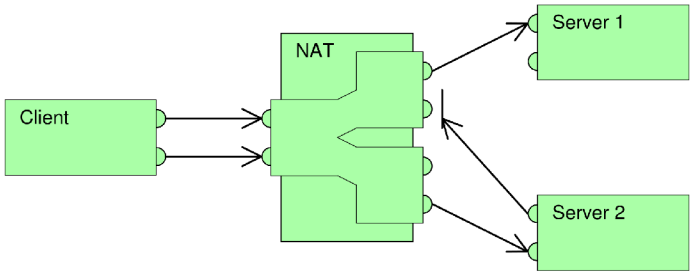
Obrázek 4: Restricted cone NAT[9]

Port restricted cone NAT

Port restricted cone NAT je podobný jako Restricted cone NAT, ale do restrikce se nyní přidá vnitřní globální port. Vnější zařízení může odeslat paket na vnitřní globální IP adresu a port pouze tehdy, pokud byl tou samou adresou a portem kontaktován v minulosti.

Symetrický NAT

Symetrický NAT je implementace, kde jsou všechny pakety ze stejné vnitřní privátní IP adresy a portu odeslané na konkrétní cílovou IP adresu a port (dependent mapping). Každá z těchto kombinací je mapována na vnitřní globální IP adresu a port. Paket může být poslán ze stejné vnitřní privátní IP adresy a portu, ale pokud se změní cílová IP adresa, tak je vytvořeno nové mapování a je mu přidělena nová vnitřní globální IP adresa a port. Tento princip je znázorněn na obrázku 5.



Obrázek 5: Symetrický NAT[9]

Mnoho implementací NATu používá kombinace těchto druhů, proto je lepší se v praxi řídit chováním jednolivých NATů. Tato chování se snaží definovat pomocí základní terminologie dokument RFC 4787, který byl vytvořen v roce 2007 firmou Cisco Systems. [10]

2.5.2 Obousměrný NAT

V Tradičním NATu jsou překládány zdrojové IP adresy pouze v odchozích paketech (Source NAT). Obousměrný NAT rozšiřuje tuto implementaci tím, že umožňuje překládat i IP adresy v příchozích paketech (Destination NAT). Tím umožňuje zařízením ve vnější síti navazovat spojení se zařízeními za Obousměrným NATem pomocí překladu cílové IP adresy v příchozích paketech. Obousměrný NAT tedy rozšiřuje Source NAT o Destination NAT.

Obousměrný NAT umožňuje vytvořit spojení jak z vnitřní sítě, tak i z vnější sítě. Privátní IP adresy jsou vázány na jedinečné veřejné IP adresy staticky nebo dynamicky během navazování spojení v obou směrech. Tuto funkcionalitu zajišťují unikátní jmenné prostory mezi zařízeními ve vnitřní a vnější síti. [8]

2.6 Problémy při využívání NATu

NAT představuje pro síťovou komunikaci několik problémů, včetně ztráty konektivity end-to-end (konec-konec) a obtížné komunikace peer-to-peer (rovný s rovným).

2.6.1 End-to-end konektivita

End-to-end je základním principem návrhu sítě, který říká, že funkce sítě by měly být v co největší míře distribuovány do koncových bodů nebo co nejblíže zařízení, které je ovládáno, nikoli soustředěny v samotné síti. To znamená, že zařízení by měla být schopna komunikovat přímo mezi sebou bez rušení ze strany sítě. V případě použití NAT se však síť aktivně zapojuje do komunikace mezi zařízeními tím, že mapuje soukromé adresy IP používané v rámci privátní sítě na jedinou veřejnou adresu IP, která se používá pro komunikaci se zařízeními mimo síť. To znamená, že zařízení v privátní síti nemohou mít přímý přístup z vnějšího světa a naopak bez použití metod NAT Traversal nebo přesměrování portů.

Tato ztráta end-to-end konektivity může způsobit několik problémů. Za prvé omezuje schopnost zařízení komunikovat přímo mezi sebou, což může mít vliv na výkon a spolehlivost některých aplikací. Například aplikace, které jsou závislé na komunikaci v reálném čase s nízkou latencí, jako jsou online hry nebo videohovory, mohou trpět zpožděním a ztrátou paketů, pokud jsou nuceny komunikovat přes NAT.

Za druhé může ztížit nasazení některých typů síťových protokolů, které se spoléhají na konektivitu end-to-end, jako je například IPsec, který ve svých paketech používá IP adresy. Protože NAT modifikuje IP adresy v paketech, mohou být přijímacím zařízením zahozeny nebo odmítnuty, což způsobí selhání komunikace.

2.6.2 Peer-to-peer komunikace

Komunikace peer-to-peer je přímá komunikace mezi dvěma zařízeními bez účasti serveru nebo prostředníka. V tradičním síťovém prostředí bez NAT je tato komunikace relativně jednoduchá, protože každé zařízení má jedinečnou IP adresu, kterou lze použít k vytvoření přímého komunikačního kanálu.

V síti s NAT je však každému zařízení obvykle přidělena privátní IP adresa, která není viditelná pro zařízení mimo privátní síť. Když spolu potřebují komunikovat dvě zařízení s privátními IP adresami za různými NAT, čelí několika problémům:

1. **Překlad IP adres a portů** - Soukromé IP adresy zařízení za NAT nejsou směrovatelné ve veřejném internetu, takže brány NAT překládají tyto adresy na veřejnou IP adresu a číslo portu, které lze použít pro komunikaci se zařízeními mimo privátní síť. Tento proces překladu však může způsobit problémy, když se dvě zařízení za různými NAT pokusí komunikovat přímo

mezi sebou, protože jejich soukromé IP adresy a čísla portů se nemusí shodovat.

2. **Omezení brány firewall a zabezpečení** - Brány NAT často obsahují vestavěné brány firewall a další bezpečnostní opatření, která mohou zabránit tomu, aby se příchozí data dostala k zařízením za NAT. To může dvěma zařízením za různými NAT ztížit navázání přímého komunikačního kanálu.

3 Metody NAT Traversal

NAT sice přináší spoustu výhod, ale má i svá negativa. Problém nastává, když se dva klienti za rozdílnými NATy snaží komunikovat. Řešením je použití některé z metod NAT Traversal (metody obcházení NATu), které jsou popsány v této sekci.

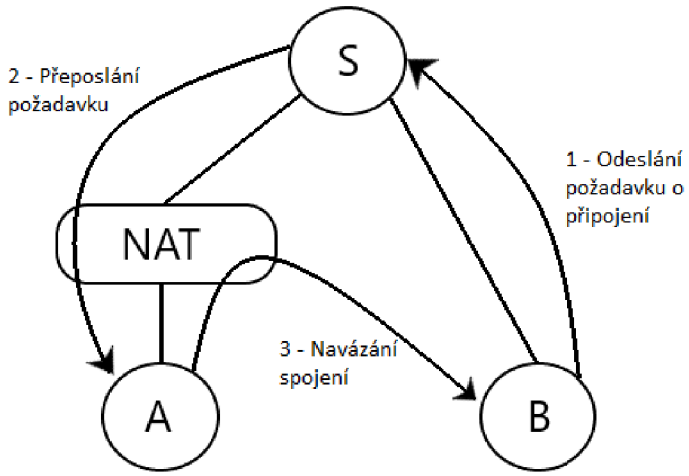
3.1 Traversal Using Relays around NAT (TURN)

TURN poskytuje řešení pro dva klienty za NATem, kteří chtějí navzájem komunikovat a posílat média. Klíčem řešení je server, který slouží jako most mezi dvěma klienty. Oba klienti posílají pakety na server a ten je navzájem klientům přeposílá. Protokol TURN se nejvíce využívá k navázání peer-to-peer připojení mezi dvěma klienty, kteří jsou za symetrickým NATem. Přestože TURN poskytuje spolehlivé připojení, není výhodné ho používat jako výchozí metodu NAT Traversal kvůli náročnému udržování TURN serveru, přes který běží veškerá komunikace klientů. Neexistuje efektivnější řešení se stejnou spolehlivostí jakou poskytuje TURN, proto se využívá v momentě, kdy klienti vyžadují maximální spolehlivost připojení. Protokoly využívané protokolem TURN pro přenos dat:

- UDP - User Datagram Protocol, preferovaný protokol pro přenos dat v reálném čase
- TCP - Transmission Control Protocol, v základu by měl být vypnutý. Využívá se pouze, pokud je UDP zakázáno kvůli firemní politice nebo pokud není UDP dosažitelné z klienta na server.

3.2 Reverse connection

Reverse connection je metoda, která se využívá, pokud jsou oba klienti připojeni k serveru **S** a jeden klient je za NATem, jak je ukázáno na obrázku 6. Klient **A** při pokusu o přímé spojení s klientem **B** nemá problém, protože **B** není za NATem. Pokud by se chtěl klient **B** připojit ke klientovi **A**, tak odešle přes server **S** požadavek na klienta **A**, aby zahájil přímé spojení s klientem **B**. Tato metoda je velmi limitovaná, ale využívá myšlenku přeposlání požadavku přes server, kterou využívají hole punching metody.



Obrázek 6: Obcházení NATu pomocí metody reverse connection

3.3 NAT hole punching

Hole punching je obecně často používaná metoda. Využívá toho, jak NAT zpracovává protokoly UDP, TCP nebo ICMP pro navázání přímého spojení mezi dvěma klienty za pomoci serveru.

3.3.1 UDP Hole Punching

Tato metoda umožňuje klientům vytvořit přímé UDP spojení. Využívá se nejčastěji u akčních online her a VoIP produktů (např. Skype) kvůli snížené latenci, kterou UDP Hole Punching nabízí. UDP hole punching se spoléhá na to, že oba klienti za NATem, kteří chtějí navázat spojení, jsou připojeni ke společnému serveru. Jakmile se nový klient připojí k serveru, jsou jeho vnitřní globální IP adresa a port a vnitřní privátní IP adresa a port zaznamenány na serveru (pokud se klient nenachází za NATem, tak jsou tyto údaje totožné). Princip metody, který je na obrázku 7, funguje následovně:

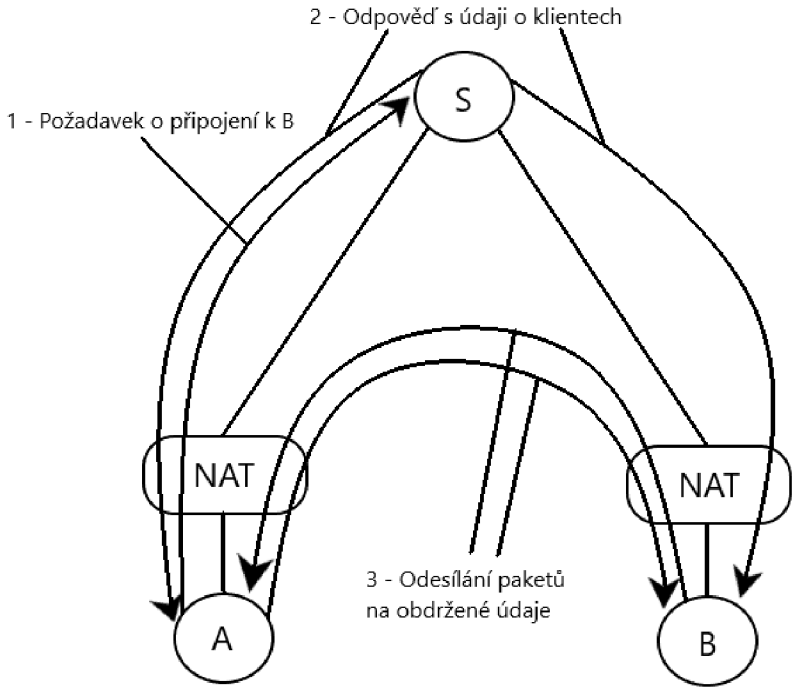
1. Klient **A** chce navázat přímé UDP spojení s klientem **B**, ale oba klienti jsou za směrovači podporující NAT. Proto pošle serveru **S** požadavek o připojení.

2. Server odpoví zprávou, která obsahuje vnitřní globální a vnitřní privátní údaje o klientovi **B**. Zároveň server pošle klientovi **B** údaje o klientovi **A**, aby mohli oba navázat přímé UDP spojení.

3. Po obdržení zprávy ze serveru oba klienti odešlou UDP pakety na obdržené údaje. Tímto se v NAT tabulce směrovače, za kterým je klient **A**, vytvoří

záznam o klientovi **B** a v NAT tabulce směrovače, za kterým je klient **B**, se vytvoří záznam o klientovi **A**.

4. Jakmile si oba klienti navzájem odešlou UDP pakety a vytvoří záznamy v NAT tabulkách, tak vzniká takzvaná "UDP díra". V tento moment oba klienti navázali přímé spojení.



Obrázek 7: Obcházení NATu pomocí metody UDP Hole Punching

3.3.2 TCP Hole Punching

TCP Hole Punching je metoda podobná UDP Hole Punchingu. Jelikož je protokol složitější a těžší na pochopení než UDP, tak není moc používaný směrovači s NATem. Pokud ho ovšem směrovač podporuje, je TCP Hole Punching stejně rychlý jako UDP Hole Punching a současně spolehlivější, protože u protokolu TCP umožňuje určit dobu trvání jednotlivých TCP spojení. Hlavním problémem pro aplikace využívající TCP Hole Punching není složitost protokolu, ale to, že standardní socket API neumožňuje současně TCP socketu odesílat a přijímat data. Takže po připojení socketu na konkrétní port pokusy o připojení druhého socketu na ten samý port selžou. Aby TCP Hole Punching fungoval, musí klient použít jeden privátní port TCP, jak na poslouchání příchozích TCP spojení, tak na navazování TCP spojení. Většina operačních systémů v dnešní době podporuje

speciální nastavení TCP socketu, pojmenované `SO_REUSEADDR`, které umožňuje aplikacím navázat několik socketů na jeden port. [11]

Předpokládejme stejnou situaci jako u UDP Hole Punchingu. Tedy 2 klienti za NATem, kteří mají navázané TCP spojení se serverem. Potom by TCP Hole Punching vypadal následovně:

1. Klient **A** chce navázat přímé TCP spojení s klientem **B**, ale oba klienti jsou za směrovači podporující NAT. Proto pošle serveru **S** požadavek o připojení.

2. Server odpoví zprávou, která obsahuje vnitřní globální a vnitřní privátní údaje o klientovi **B**. Zároveň server pošle klientovi **B** údaje o klientovi **A**, aby mohli oba navázat TCP spojení.

3. Oba klienti využijí ten samý vnitřní port, kterým se připojili k serveru, a začnou na něm poslouchat pro příchozí pakety. Zároveň z tohoto portu budou odesílat TCP pakety na obdržené údaje ze serveru.

4. Odesíláním paketů na obdržené údaje o druhém klientovi se v NAT tabulce směrovače, za kterým se klient nachází, vytvoří záznam, což umožní průchod paketů odeslaných druhým klientem. V tento moment vznikla takzvaná "TCP díra", přes kterou prochází pakety odesílané druhým klientem.

5. Klienti čekají, jestli se jim podaří navázat TCP spojení. Pokud nějaký z pokusů o vytvoření přímého TCP spojení selže, klient ho znovu odešle. Tento proces se opakuje, dokud se nepodaří navázat TCP spojení.

6. Po úspěšném navázání TCP spojení klienti ověří, zda se připojili ke správnemu klientovi. Pokud se jim správnost ověřit nepodaří, klient navázané spojení uzavře a čeká na další. Klienti využívají první úspěšně ověřené spojení, které mezi sebou navázali.

Na rozdíl od UDP Hole Punchingu, kterému k navázání spojení stačí pouze jeden socket, TCP Hole Punching ke spojení potřebuje jeden socket na každou činnost (poslouchání, připojení k serveru, připojení ke klientovi).

3.3.3 ICMP Hole Punching

Tato metoda se využívá pro udržování proudu ICMP (Internet Control Message Protocol) paketů, které procházejí NATem. ICMP je protokol, který používají směrovače pro zjištění chyby během síťové komunikace. ICMP se nejčastěji používá pro zjištění, zda odeslané pakety dorazily do cíle.

Metoda ICMP Hole Punching využívá ICMP paketů, které mají obvykle povolený průchod skrz většinu NAT zařízení a firewallů, pro přeposlání veřejné IP adresy a portu mezi dvěma zařízeními za NATem. ICMP Hole Punching se často využívá u modelu klient-server, kdy chceme, aby libovolný počet klientů za různými NATy mohl komunikovat se serverem, který je také za NATem. [12]

Proces navázání přímého spojení mezi klientem a serverem pomocí ICMP Hole Punching metody funguje následovně:

1. Klient se chce připojit k serveru, přičemž oba jsou za různými NATy a klient zná veřejnou IP adresu serveru.

2. Server pravidelně odesílá ICPM Echo Request paket (ping) na IP adresu, od které neočekává žádnou odpověď.

3. Klient odešle na server ICMP Time Exceeded paket, do kterého vloží svou veřejnou IP adresu. Tento paket obsahuje "původní" paket, který odesílá server. Klient tedy předstírá, že přes něj paket odeslaný serverem procházel a nebylo ho možno doručit.

4. NAT zařízení, za kterým je server, obdrží ICMP Time Exceeded paket od klienta. Jelikož tento paket obsahuje původní paket odesílaný serverem, tak ho přepošle serveru.

5. Jakmile server obdrží tento paket, tak zná veřejnou IP adresu klienta, který se k němu chce připojit, a oba mohou navázat přímé spojení.

Tato metoda na rozdíl od UDP nebo TCP Hole Punchingu nevyužívá žádný externí server pro přeposlání údajů k vytvoření přímého spojení. Předpokladem ovšem je, že server pravidelně odesílá ICPM Echo Request pakety a klient tento paket má, aby ho mohl využít k vytvoření ICMP Time Exceeded paketu. Dále je důležité poznamenat, že úspěšné navázání přímého spojení touto metodou může záviset na několika faktorech, jako jsou konfigurace sítě, typ NAT implementace nebo pravidla na firewallu. [13]

3.4 Session Traversal Utilities for NAT (STUN)

STUN je sada metod, které slouží ke zjištění veřejné IP adresy a portu zařízení za NATem. STUN využívá model klient-server. Pro fungování metody je potřeba STUN server, který je umístěný v jiné síti, než je zařízení za NATem. Po odeslání požadavku zařízením za NATem na STUN server, server odpoví zprávou, která obsahuje veřejnou IP adresu a port, ze kterých přišel požadavek.

STUN sám o sobě nenavazuje přímé spojení mezi klienty, ale pouze poskytuje informace k jeho uskutečnění. STUN se často využívá ve spojení s metodou Hole Punching nebo Interactive Connectivity Establishment (ICE) pro vytvoření samotného spojení.

3.4.1 Simple Traversal of UDP Through NATs (STUN)

Simple Traversal of UDP Through NATs rozšiřuje funkci základního STUNu o možnost navázání přímého spojení použitím UDP Hole Punchingu. Tuto metodu lze ještě rozšířit, aby mohla využívat i TCP Hole Punchingu k navázání přímého spojení. Toto rozšíření se nazývá STUNT (Simple Traversal of UDP Through NATs and TCP too). [14]

3.5 Interactive Connectivity Establishment (ICE)

ICE je metoda, která uskutečňuje přímé spojení mezi dvěma klienty, kteří neznají topologii sítě, ve které se druhý klient nachází. ICE umožňuje klientům zjistit dostatek informací o jejich topologiích, aby našli jednu nebo více cest, kterými by mohli navázat přímé spojení.

Klíčové prvky metody ICE jsou:

- **Sbírání kandidátů**: Za účelem provedení ICE, každý klient identifikuje a shromáždí jednoho nebo více více kandidátů na adresu. Tito kandidáti jsou kombinace IP adres a portů, které může klient využít k odesílání a přijímání dat pomocí UDP/TCP protokolu. Existuje několik typů kandidátů. Někteří jsou odvození od fyzického nebo síťového rozhraní sítě a jiné jsou zjistitelné pomocí STUN serveru.

- **Kontrola připojení**: Proces začíná tím, že oba klienti shromáždí kandidáty a pošlou je druhému klientovi prostřednictvím STUN serveru. Oba klienti mají nyní k dispozici kompletní seznam kandidátů. Kandidáty spárují a vytvoří dvojice kandidátů. Aby určili, které dvojice jsou úspěšné, provede každý klient sérii kontrol připojení. Tyto kontroly zahrnují odeslání požadavků STUN (Session Traversal Utilities for NAT) na STUN server a obdržení odpovědí.

- **Nominování dvojic**: ICE pověří jednoho z klientů rolí kontrolora a druhého rolí kontrolovaného klienta. Kontrolující klient pokračuje v kontrolách připojení, dokud nenajde alespoň jednu dvojici, která úspěšně projde

kontrolou. Po nalezení ji vybere a odešle požadavek STUN na STUN server, čímž kontrolovanému klientovi oznámí, že byla nominována. Po přijetí požadavku STUN s atributem nominace kontrolovaný klient zkontroluje stejnou dvojici. Pokud jsou kontroly úspěšné, klienti označí příznak nominované dvojice a ukončí veškeré budoucí kontroly této složky datového toku. Jakmile je pro každou složku nastaven nominovaný příznak, stávají se tyto dvojice vybranými dvojicemi.

- **Vytvoření přímého spojení**: Pokud dvojice projde kontrolou připojení a je nominována, tak ICE naváže mezi klienty přímé spojení použitím NAT Traversal metody (například UDP Hole Punching).

Využitím těchto prvků umožňuje ICE klientům navazovat přímé spojení i ve složitých síťových prostředích s NAT, firewally a různými síťovými podmínkami. Jeho cílem je najít co nejefektivnější a nejpřímější komunikační cestu. [15]

Hlavním problémem ICE je čas potřebný k zahájení kontroly připojení, která vyžaduje shromáždění všech kandidátů předem. Toto řeší optimalizace Trickle ICE, která umožňuje shromažďovat a vyměňovat kandidáty postupně.

3.6 Port Forwarding (Port Mapping)

Port Forwarding umožňuje přesměrovat komunikaci z jedné IP adresy a portu na druhou IP adresu a port, tedy obejití směrovače s NATem. Vnější zařízení, která se budou chtít připojit ke klientovi ve vnitřní síti, musí pro navázání komunikace znát přesměrovaný port. Port Forwarding může sloužit i jako filtr příchozích paketů, díky vytvoření pravidel přesměrování. Na rozdíl od Hole Punchingu, Port Forwarding redukuje provoz v síti vytvořený udržováním kontaktu se serverem, který Hole Punching vyžaduje. Tradiční Port Forwarding vyžaduje manuální konfiguraci směrovače.

UPnP (Universal Plug and Play) je soubor protokolů, který poskytuje funkci pro automatické přesměrování vnitřních globálních portů do vnitřních privátních portů. Aplikace mohou využít UPnP k rezervaci portu a přesměrovat příchozí pakety na socket, na kterém aplikace poslouchá. Využití UPnP představuje riziko pro zabezpečení sítě, protože UPnP nepodporuje žádné ověření příchozích paketů.

Jednou z hlavních komponent UPnP je protokol IGDP (Internet Gateway Device Protocol). IGDP umožňuje zařízením ve vnitřní síti komunikovat se směrovačem a vyžádat si nebo nastavit přesměrování portu automaticky.

3.7 Port Control Protocol (PCP)

PCP je síťový protokol, určený ke správě a řízení přesměrovaných portů na směrovači s NATem nebo firewallu. Hlavním účelem protokolu PCP je zjednodušit proces vytváření a správy pravidel pro přesměrování portů. PCP tento proces automatizuje tím, že umožňuje zařízením ve vnitřní síti požádat směrovač o otevření konkrétních portů a předání příchozí komunikace na ně, podobně jako funguje UPnP.

PCP umožňuje aplikacím vytvořit mapování z veřejné IP adresy, protokolu a portu na privátní IP adresu, protokol a port. O tomto přesměrování je potřeba informovat zařízení ve vnější síti, od kterých očekáváme data. Tuto informaci si předává aplikace sama, například pomocí serveru.

I když se PCP zdá podobný jako UPnP, tak je PCP považován za bezpečnější. PCP podporuje autorizaci, což znamená, že mapování mohou vytvářet pouze autorizovaná zařízení v síti. PCP dále využívá protokoly, jako například TLS (Transport Layer Security), který šifruje komunikaci mezi zařízeními ve vnitřní síti a směrovačem.

4 Aplikace

Cílem této práce je také demonstrovat vybrané metody NAT Traversal v reálném síťovém prostředí. Tato část se zabývá využitými technologiemi, technickým provedením a popisem používání aplikace.

4.1 Využité technologie

Pro vytvoření aplikace jsem použil následující seznam softwaru. Konkrétní využití v aplikaci je popsáno v kapitole 4.2.

`.NET Framework`
: .NET Framework je open source framework vytvořený firmou Microsoft. Jedná se o původní implementaci .NET. Slouží primárně pro vývoj aplikací pro operační systém Windows. Při vývoji v .NET Framework se využívají programovací jazyky `C#`, `F#` a Visual Basic. [16]

`Windows Forms`
: Windows Forms je prvním frameworkem z .NETu, který umožňuje jednoduchou tvorbu formulářových aplikací pomocí grafického designeru. Je to architektura grafického rozhraní pro vytváření aplikací na Windows. Windows Forms umožňují jednoduchý vývoj a aktualizace uživatelského rozhraní.

`Open.NAT`
: Open.NAT je knihovna tříd pro přesměrování portů na NAT zařízeních, které podporují protokoly UPnP a PMP. Je vyvinuta v `C#` a použitelná v .NET. [17]

`Amazon Elastic Compute Cloud (EC2)`
: EC2 je součástí Amazon platformy cloud computingu, Amazon Web Services. Umožňuje vytvoření vlastního vitruálního počítače pro běh vlastních aplikací, který může sloužit jako server.

4.2 Programátorská příručka

V této kapitole je popsána funkčnost grafického rozhraní, implementace metod NAT Traversal a konfigurace serveru pro Hole Punching metodu. Většina kódu v programu je okomentována, aby bylo vše srozumitelné pro uživatele i bez této příručky.

4.2.1 Grafické rozhraní

Grafické rozhraní jsem navrhl tak, že jsem vytvořil hlavní formulář, ve kterém jsou všechny pevné prvky. Tedy navigace, hlavní lišta s návem aktuální stránky a název aplikace v levém horním rohu. Hlavní formulář obsahuje také panel, do

kterého se zobrazí sekundární forma. Sekundární formulář se mění po zmáčnutí k němu přiděleného tlačítka. Například po zmáčknutí tlačítka Hole Punching se zvýrazní zmáčknuté tlačítko a do panelu pro sekundární formulář se zobrazí forma HolePunching. Kód 1.

```
1       private void OpenSecondaryForm(Form secondaryForm, object
            btnSender)
2       {
3           if (activeForm != null) activeForm.Close();
4
5           ActivateButton(btnSender);
6           activeForm = secondaryForm;
7           secondaryForm.TopLevel = false;
8           secondaryForm.FormBorderStyle = FormBorderStyle.None;
9           secondaryForm.Dock = DockStyle.Fill;
10
11          this.panelObrazovka.Controls.Add(secondaryForm);
12          this.panelObrazovka.Tag = secondaryForm;
13
14          secondaryForm.BringToFront();
15          secondaryForm.Show();
16          lblNadpis.Text = secondaryForm.Text;
17      }
```

Zdrojový kód 1: Změna sekundárního formuláře podle zmáčnutého tlačítka (btnSender).

4.2.2 Architektura aplikace

Pro vývoj aplikace jsem zvolil architekturu MVC (Model-View-Controller). View reprezentuji jednotlivými formuláři, které obsahují všechny grafické prvky a metody, které může volat controller např. pro zobrazení nějaké zprávy. Pro komunikaci mezi view a controllerem jsem vytvořil interface, jehož instanci si controller naváže do proměnné po vytvoření instance controlleru, aby mohl view zasílat zprávy. Po spuštění programu se vytvoří instance jednotlivých view a controllerů a zavolají je metody nutné pro správné fungování aplikace (např. u Hole Punchingu se vytvoří spojení se serverem nebo u Port Mappingu se vypíší všechna existující mapování).

4.2.3 Port Mapping

Pro funkčnost této metody je potřeba, aby uživatel měl spuštěný UPnP na svém NAT zařízení. Většina z funkcí, které využívá tato metoda, jsou jednoduché na implementaci, a to díky použití Open.NAT knihovny.

Po načtení formy se spustí kód 2, který vypíše do check boxu všechna aktuální mapování. Metoda se pokusí zjistit, zda je počítač za NATem. Pokud ano, tak

začne hledat UPnP zařízení, které tento NAT zprostředkovává. Po nalezení NAT zařízení se pro každé existující směrování vypíší jeho informace do check boxu ve tvaru, jako je v kódu 2 na řádku 10. Pokud se nepodaří najít NAT s UPnP, vyskočí okno s chybovou hláškou (Kód 2 na řádku 22).

```
1       private async Task ListMappings()
2       {
3          try
4          {
5             int addedMappings = 0;
6             var nat = new NatDiscoverer();
7             var cts = new CancellationTokenSource(5000);
8             var device = await nat.DiscoverDeviceAsync(PortMapper.
                  Upnp, cts);
9             checkBoxMappings.Items.Clear();
10            // "mapping_name: NAT_device_IP:external_port ->
                  host_machine_IP:internal_port"
11            foreach (var mapping in await device.GetAllMappingsAsync
                  ())
12            {
13               string item = mapping.Description + ": " + await
                     device.GetExternalIPAsync() + ":" + mapping.
                     PublicPort + " -> "
14                   + mapping.PrivateIP + ":" + mapping.PrivatePort +
                        "\n";
15               checkBoxMappings.Items.Add(item, false);
16               addedMappings++;
17            }
18            if(addedMappings == 0) checkBoxMappings.Items.Add("
                  Nebyla nalezena žádná mapování", false);
19         }
20         catch (NatDeviceNotFoundException)
21         {
22            MessageBox.Show("Nebylo nalezeno UPnP zařízení.");
23         }
24      }
```

Zdrojový kód 2: Výpis směrování

Metody vytvoření a smazání směrování jsou podobné, jako metoda ListMappings (Kód 2 řádky 6-8). U metody vytvoření se po stejné části přidá pouze vytvoření směrování se zadanými údaji z text boxů (Kód 3).

```
1             var mapping = new Mapping(Protocol.Tcp, intPort, extPort
                  , mappingName);
2             await device.CreatePortMapAsync(mapping);
```

Zdrojový kód 3: Vytvoření směrování

U metody smazání se přidá vyhledání jmen směrování, která jsou zakliknutá v check boxu, a podle nich se dané směrování vymaže.

4.2.4 Hole Punching

Tato metoda vyžaduje dva rozdílné kódy a nastavení. Jeden pro klienta a jeden pro server. K tvorbě jsem využil protokol UDP, protože je jeho implementace jednodušší, než u protokolu TCP, a mé zkušenosti s programováním síťových aplikací v C# nejsou pro tento komplexní úkol dostačující. Rozdíl mezi TCP a UDP Hole Punchingem je vysvětlený v sekci 3.3.

Server

Před spuštěním kódu na serveru bylo nutné ho správně nakonfigurovat. Tedy otevřít port, přes který ho budou klienti kontaktovat. Dále bylo potřeba povolit v zabezpečení přijímání a odesílání UDP paketů na tom samém portu. Funkce serveru je jinak velmi triviální:

1. Server obdží na port zprávu, která obsahuje klíč.
2. Podívá se do svých záznamů, zda od tohoto klienta už klíč dostal.
3. Pokud ano a klíč je 0, záznam ze serveru smaže. Pokud se jedná o jiné číslo než 0, server klientovi odpoví, že už o něm záznam má.
4. Pokud klíč doposud nedostal a obdržený klíč se neshoduje s jiným v záznamu, tak si do záznamu uloží IP adresu, port, obdržený klíč a čas obdržení zprávy od nového klienta.
5. V případě, že se klíč shoduje s dříve uloženým klíčem v záznamech serveru, tak server přepošle údaje k připojení klientům se stejným klíčem a záznamy o nich smaže.

Server automaticky maže záznamy delší, jak 10 minut, aby nedošlo k přeplnění. Jelikož data lze posílat pouze pomocí byte struktury, tak je před odesláním a po přijetí nutný převod do požadované struktury. Server po každé zprávě hned odpoví, aby nebylo potřeba implemetovat oddělená vlákna pro příjem a odesílání dat.

Klient

Po otevření formy Hole Punching se vytvoří IPEndPoint obsahující IP adresu a port serveru pro připojení a UDP klient, který se na IPEndPoint naváže. Aby klient fungoval správně, bylo na něm potřeba povolit NAT Traversal, vypnout jeho restrikce (aby aplikace mohla plně využít možnosti NAT Traversal) a povolit jeho znovupoužití (Kód 4). Po nastavení klienta se připojí k serveru a udržuje s ním komunikaci. Protože klient může kdykoliv obdržet více zpráv po sobě, tak poslouchání pro příchozí zprávy běží na odděleném vlákně.

```
1                client.AllowNatTraversal(true);
2                client.ExclusiveAddressUse = false;
3                client.Client.SetIPProtectionLevel(IPProtectionLevel.
                     Unrestricted);
4                client.Client.SetSocketOption(SocketOptionLevel.
                     Socket, SocketOptionName.ReuseAddress, true);
5                client.Client.Connect(serverEndPoint);
```

Zdrojový kód 4: Nastavení socketu, na který je připojen server.

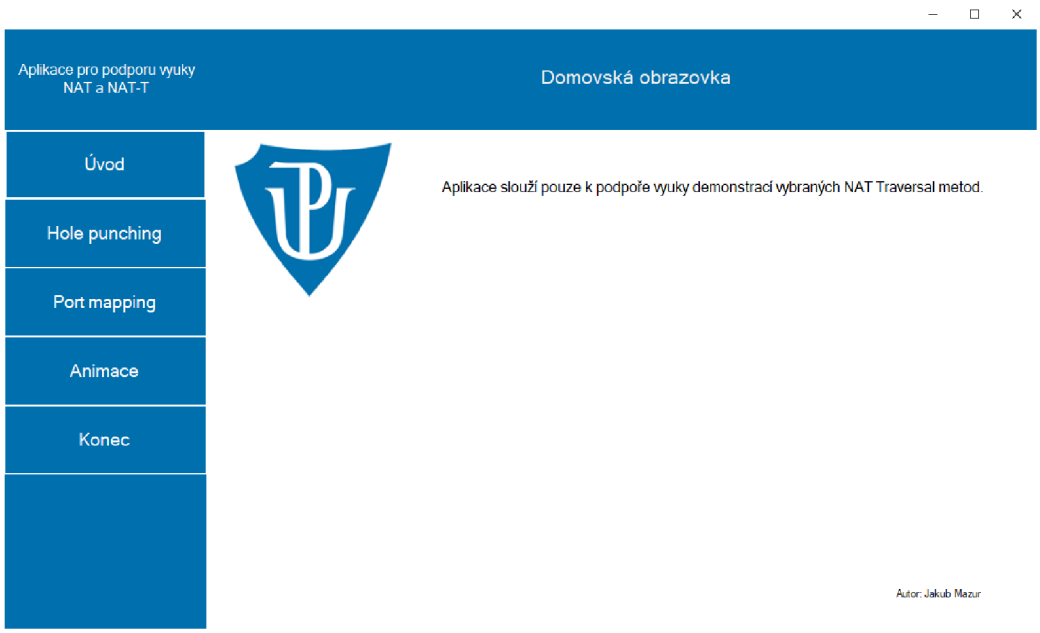
Při každém obdržení zprávy ze serveru klient zkontroluje, zda se nejedná o IP adresu. Pokud ano, tak se na server připojil druhý klient, který zadal stejný unikátní klíč. Tedy obdržená IP adresa a port, který následuje v další zprávě, slouží k navázání přímého spojení s druhým klientem. Klient naváže spojení pomocí stejného UDP klienta, který použil na komunikaci se serverem, a odešle krátkou zprávu, aby se v NAT tabulce vytvořil záznam a klient tak mohl dostávat zprávy od druhého klienta. Jelikož aplikace slouží pouze k demonstraci metod, tak je funkcionalita omezená na jednoduché posílání zpráv mezi klienty a slouží pouze k porozumnění dané problematiky. Celou komunikaci mezi serverem a klienty můžete odchytnout pomocí libovolné aplikace pro analýzu provozu v počítačových sítích.

4.3 Uživatelská příručka

Následující kapitola popisuje základní používání aplikace pro demonstraci metod a ověření jejich funkčnosti.

4.3.1 Používání aplikace

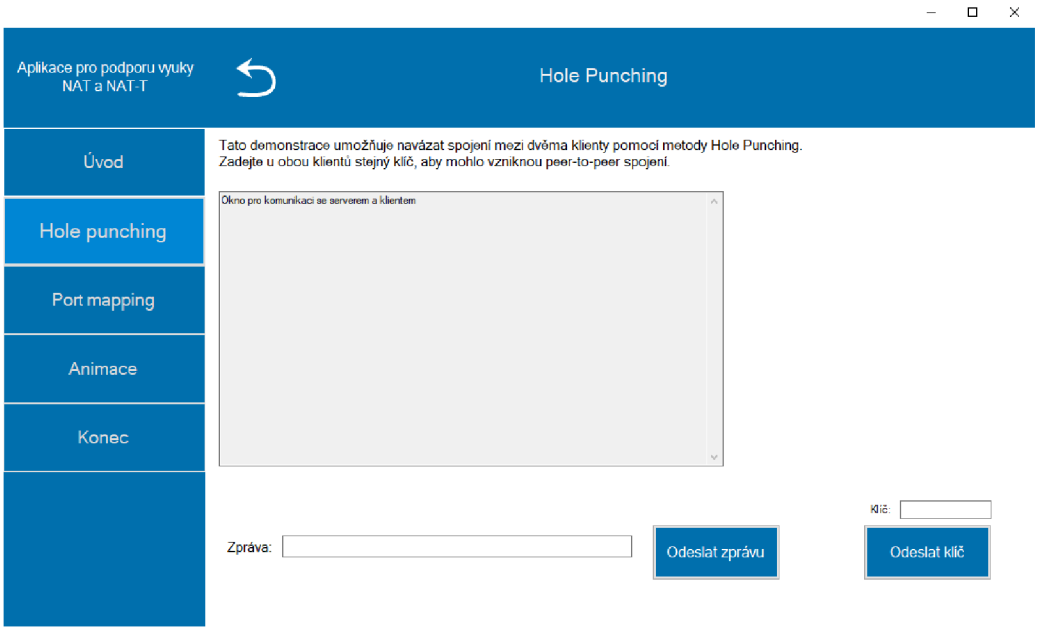
Aplikace se spouští pomocí **exe** souboru obsaženého ve složce `Mazur_NAT-T/bin/Release` nebo je možné spustit instalační soubor ve stejné složce s příponou **application**. Po spuštění se zobrazí domovská obrazovka, viz obrázek 8.



Obrázek 8: Úvodní obrazovka aplikace

Tlačíto *Úvod* v navigační liště zobrazí texty obsahující základní informace o tom, jak vybrané metody fungují. Tlačítka *Hole punching* a *Port mapping* odkazují na jednotlivé metody. Obrazovka Hole punchingu je velmi intuitivní.

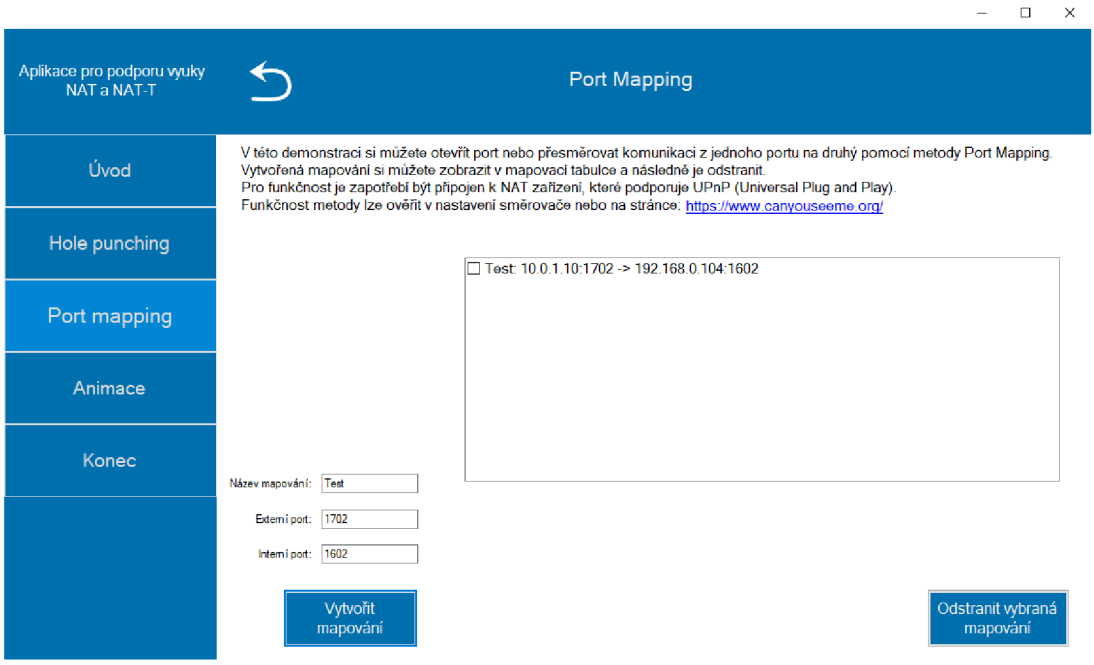
Komunikace se serverem je zahájena od načtení aplikace, tudíž jediné, co je potřeba, je odeslat unikátní klíč. Pro vytvoření spojení mezi dvěma klienty za NATem metodou Hole punchingu musí na obou počítačích běžet tato aplikace. Poté musí oba klienti odeslat na server stejný klíč a aplikace vytvoří spojení (viz obrázek 9). Nyní si mohou oba klienti vyměňovat textové zprávy.



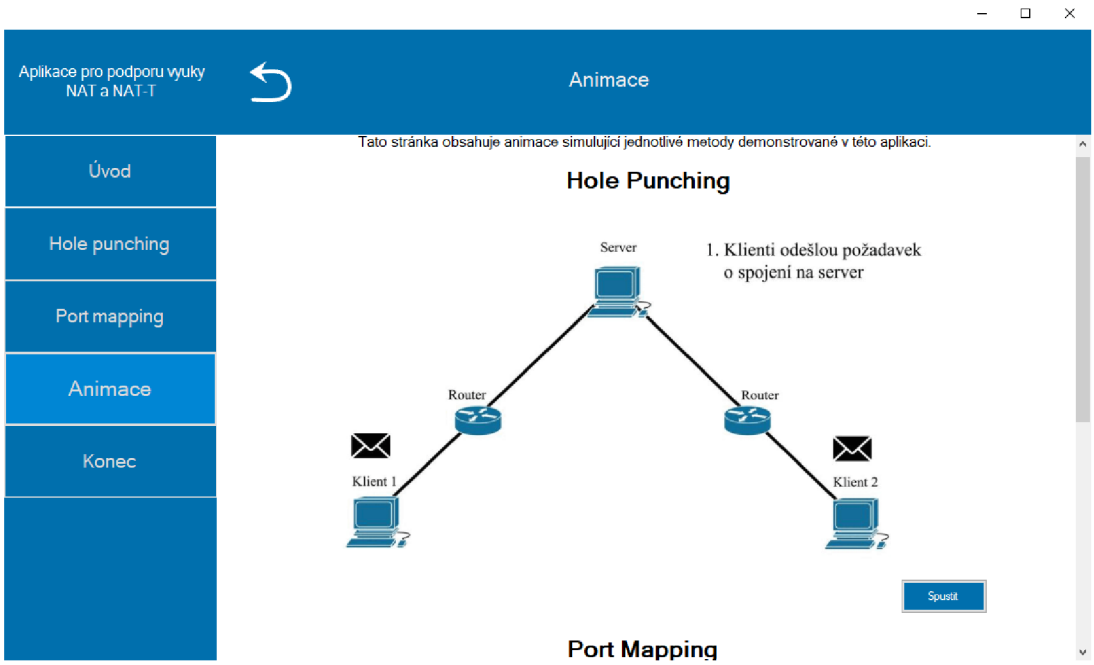
Obrázek 9: Obrazovka Hole punching

Port mapping sekce umožňuje dvě funkce. A to vytvoření nebo odstranění mapování (výpis existujících mapování je automatický). Pro vytvoření mapování je potřeba zadat tři údaje. *Název mapování*, což může být libovolný string. *Externí port*, který se otevře na NAT zařízení, za kterým aplikace běží. *Interní port*, což je port, který se otevře na zařízení, na kterém běží aplikace. Zmáčnutí tlačítka *Vytvořit mapování* vezme hodnoty v textových polích a vytvoří pomocí nich nové mapování (viz obrázek 10). Funkčnost mapování muže uživatel ověřit pomocí stránky uvedené v aplikaci nebo v nastavení směrovače. Odstranění mapování je možné pomocí výběru mapování, které chceme odstranit a následném zmáčknutí tlačítka *Odstranit vybraná mapování*.

Tlačítko *Animace* otevře obrazovku s jednoduchými animacemi simulujícími jednotlivé metody demonstrované v aplikaci. Animace obsahují i popisky vysvětlující celý proces metod. Pro spuštění animací je potřeba kliknout na tlačítko *Spustit* pod jednotlivými obrázky (viz obrázek 11).



Obrázek 10: Obrazovka Port mapping s vytvořenými mapováními



Obrázek 11: Obrazovka s animacemi

4.3.2 Konfigurace vlastního serveru

Ve složce `UDP_Server/bin/Debug` je **exe** soubor, který je potřeba spustit na serveru. Před spuštěním aplikace na serveru je nutné otevřít UDP port, na kterém mohou server kontaktovat klienti. Na Windows 10 serveru je možné toto nastavit

ve Firewallu, kde musíme přidat nové příchozí pravidlo pro UDP protokoly na konkrétním portu.

Protože klientská aplikace má v základu připojovací údaje k jinému serveru, než je tento nový, je potřeba v souboru `Mazur_NAT-T/Resources/config.txt` změnit IP adresu a port na ty údaje, které má nový server.

Závěr

Výsledná práce splňuje mé předem stanovené cíle a požadavky. Těmito je přiblížení NAT problematiky a její řešení pomocí NAT Traversal metod. Práce dle mého názoru stručně a věcně popisuje, co se děje při překládání adres nebo při použití jednotlivých NAT Traversal metod. Vybíral jsem takové metody, se kterými by se mohl čtenář setkat ve světě počítačových sítí. Ty nejčastěji používané jsem převedl do aplikace, která je jednoduše, ale výstižně demonstruje. I když to vypadá jednoduše, vymyslet řešení vybraných metod bylo to nejobtížnější. Většina z existujících řešení je totiž closed source, tudíž jsem musel nastudovat práci s UDP protokolem a vytvořit si vlastní řešení. Díky tomu jsem byl schopen porozumět počítačovým sítím a práci s UDP protokolem v programování. Každá z vybraných metod je v aplikaci doplněna textem, který jednoduše popisuje vykonávání metod a animacemi, které zjednodušeně ukazují, jak jednotlivé metody probíhají.

Aplikace má i své nedostatky. Například řešení uživatelského rozhraní nebo zabezpečení, které je omezeno kvůli použití UPnP. Snažil jsem se tento nedostatek negovat tím, že každý port nebo spojení je otevřeno pouze na nezbytně dlouhou dobu, ale ani to by v nezabezpečené veřejné síti nestačilo.

Conclusions

The resulting work meets my predetermined goals and requirements. Description of NAT problem and its solution using NAT Traversal methods. In my opinion, the thesis concisely and factually describes what happens when translating addresses or when using individual NAT Traversal methods. I have chosen methods that the reader might encounter in the world of computer networking. I have translated the most commonly used ones into an application that demonstrates them simply but concisely. Although it looks simple, coming up with solutions to the selected methods was the most difficult part. This is because most of the existing solutions are closed source, so I had to study working with the UDP protocol and create my own solution. Thanks to this, I was able to understand computer networks and working with the UDP protocol in programming. Each of the selected methods in the application is accompanied by text that simply describes the execution of the methods and animations that simply show how each method works.

The application also has its shortcomings. For example, the user interface solution or the security, which is limited, due to the use of UPnP. I tried to negate this deficiency by making sure that each port or connection is only open for the necessary amount of time, but even that would not be enough in an unsecured public network.

A Obsah elektronických dat

`app/`
: Složka aplikace obsahující zdrojový kód a spustitelné a instalační soubory.

`server/`
: Obsahem složky je zdrojový kód serveru a spustitelný soubor v podobě konzolové aplikace.

`text/`
: Složka obsahující text práce a její zdrojový kód.

`README.txt`
: Instrukce pro instalaci programu a nastavení a spuštění vlastního serveru.

Literatura

[1] Odom, Wendell; Healy, Rus; Mehta, Naren. *Směrování a přepínání sítí: autorizovaný výukový průvodce: Samostudium* [online]. First. Brno): Computer Press, 2009 [cit. 2022-7-20]. ISBN 978-802-5125-205.

[2] Microsoft. *How NAT Works* [online]. 2009 [cit. 2023-6-16]. Dostupný z: ⟨https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc756722(v=ws.10)?redirectedfrom=MSDN⟩.

[3] Firewall.cx. *Network address translation table* [online]. 2022 [cit. 2023-2-26]. Dostupný z: ⟨https://www.firewall.cx/networking-topics/network-address-translation-nat/228-nat-table.html⟩.

[4] Srisuresh, P.; Holdrege, M. IP Network Address Translator (NAT) Terminology and Considerations: RFC 2663. *Network Working Group* [online]. 1999, [cit. 2023-6-15]. Dostupný z: ⟨https://dl.acm.org/doi/book/10.17487/RFC2663⟩.

[5] Networks, Juniper. *Source NAT* [online]. 2023 [cit. 2023-6-15]. Dostupný z: ⟨https://www.juniper.net/documentation/us/en/software/junos/nat/topics/topic-map/nat-security-source-and-source-pool.html⟩.

[6] IBM. *Masquerade (hide) NAT* [online]. 2021 [cit. 2023-6-15]. Dostupný z: ⟨https://www.ibm.com/docs/en/i/7.2?topic=translation-masquerade-hide-nat⟩.

[7] Networks, Juniper. *Destination NAT* [online]. 2023 [cit. 2023-6-16]. Dostupný z: ⟨https://www.juniper.net/documentation/us/en/software/junos/nat/topics/topic-map/security-nat-destination.html⟩.

[8] Rosenberg, J.; Weinberger, J.; Huitema, C.; Mahy, R. STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs): RFC 3489. *Network Working Group* [online]. 2003, [cit. 2022-7-20]. Dostupný z: ⟨https://datatracker.ietf.org/doc/html/rfc3489⟩.

[9] Wikipedia. *Network address translation* [online]. 2022 [cit. 2022-7-21]. Dostupný z: ⟨https://en.wikipedia.org/wiki/Network_address_translation⟩.

[10] Audet, F.; Jennings, C. Network Address Translation (NAT) Behavioral Requirements for Unicast UDP: RFC 4787. *Network Working Group* [online]. 2007, [cit. 2022-7-22]. Dostupný z: ⟨https://datatracker.ietf.org/doc/html/rfc4787⟩.

[11] Ford, Bryan; Srisuresh, Pyda; Kegel, Dan. *Peer-to-Peer Communication Across Network Address Translators* [online]. [cit. 2022-7-23]. Dostupný z: ⟨https://bford.info/pub/net/p2pnat/⟩.

[12] Pemare, Pooja. *Overview of Hole Punching: ICMP Hole Punching, TCP Hole Punching, UDP Hole Punching* [online]. 2020 [cit. 2023-6-18]. Dostupný z: ⟨https://www.irjet.net/archives/V7/i4/IRJET-V7I408.pdf⟩.

[13] Kamkar, Samy. *Autonomous NAT Traversal* [online]. 2010 [cit. 2023-6-18]. Dostupný z: ⟨http://samy.pl/pwnat/pwnat.pdf⟩.

[14] Guha, Saikat. *Cornell University, Department of Computer Science - STUNT* [online]. 2007 [cit. 2023-6-18]. Dostupný z: ⟨https://web.archive.org/web/20170911122644/http://nutss.gforge.cis.cornell.edu/stunt.php⟩.

[15] Keranen, A.; Rosenberg, J. Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal: RFC 8445. *Internet Engineering Task Force* [online]. 2018, [cit. 2023-6-20]. Dostupný z: ⟨https://datatracker.ietf.org/doc/html/rfc8445⟩.

[16] Microsoft. *What is .NET?* [online]. 2022 [cit. 2022-7-23]. Dostupný z: ⟨https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet⟩.

[17] Ontivero, Lucas. *Open.NAT: A NAT Traversal library for .NET and Mono* [online]. 2014 [cit. 2022-7-23]. Dostupný z: ⟨https://www.codeproject.com/Articles/807861/Open-NAT-A-NAT-Traversal-library-for-NET-and-Mono⟩.