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:	33
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:	33
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.

Odevzdáním tohoto textu jeho autor/ka místopřísežně prohlašuje, že celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

Obsah

1 Úvod **8**

2 NAT **9**
2.1 Využití . . . 9
2.2 Překlad adres . . . 10
2.3 NAT tabulka . . . 11
2.3.1 Obsah NAT tabulky . . . 11
2.3.2 Bezpečnostní rizika . . . 11
2.4 Druhy přiřazování adres . . . 12
2.4.1 Statický NAT . . . 12
2.4.2 Dynamický NAT (tzv. Maškaráda) . . . 12
2.4.3 Přetížený NAT (NAT overloading, Port Address Translation) 13
2.5 Druhy implementací NATu . . . 14
2.5.1 Full cone NAT (1:1 NAT) . . . 14
2.5.2 Restricted cone NAT . . . 14
2.5.3 Port restricted cone NAT . . . 15
2.5.4 Symetrický NAT . . . 15
2.6 Problémy při využívání NATu . . . 16
2.6.1 End-to-end konektivita . . . 16
2.6.2 Peer-to-peer komunikace . . . 16

3 Metody NAT Traversal **18**
3.1 Traversal Using Relays around NAT (TURN) . . . 18
3.2 Reverse connection . . . 18
3.3 NAT hole punching . . . 19
3.3.1 UDP Hole Punching . . . 19
3.3.2 TCP Hole Punching . . . 20
3.4 Port Forwarding (Port Mapping) . . . 21

4 Aplikace **22**
4.1 Využité technologie . . . 22
4.2 Programátorská příručka . . . 22
4.2.1 Grafické rozhraní . . . 22
4.2.2 Port Mapping . . . 23
4.2.3 Hole Punching . . . 25
4.3 Uživatelská příručka . . . 26
4.3.1 Používání aplikace . . . 26
4.3.2 Konfigurace vlastního serveru . . . 29

Závěr **30**

Conclusions **31**

A Obsah elektronických dat **32**

Literatura **33**

Seznam obrázků

1 Překlad adres ve statickém NATu 10
2 Full cone NAT[4] 14
3 Restricted cone NAT[4] 15
4 Symetrický NAT[4] 15
5 Obcházení NATu pomocí metody reverse connection 19
6 Obcházení NATu pomocí metody UDP Hole Punching 20
7 Úvodní obrazovka aplikace 27
8 Obrazovka Hole punching 27
9 Obrazovka Port mapping s vytvořenými mapováními 28
10 Obrazovka s animacemi 29

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

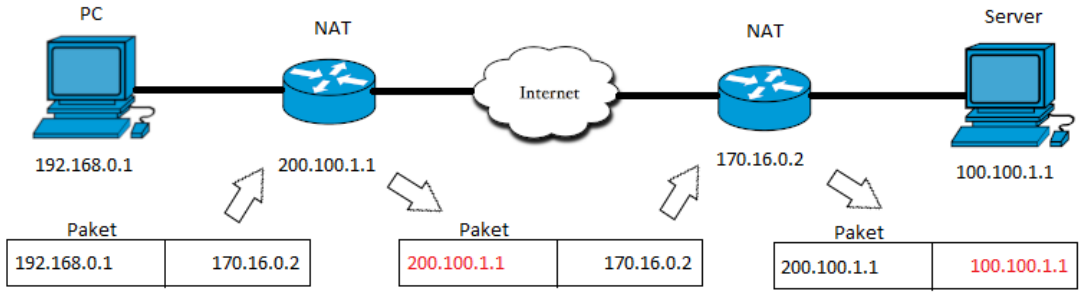
2. **Privátní** - nejsou veřejně zaregistrované, jsou využívány pouze interně (např. v domácnosti nebo ve firmě), nelze se pomocí ní připojit k internetu

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 4 typy adres v NAT [1]:

1. **Vnitřní lokální** - privátní IP adresa, která se nachází se uvnitř lokální sítě (192.168.0.1)
2. **Vnitřní globální** - adresa, pod kterou jsou zařízení v lokální síti vidět z vnější sítě (200.100.1.1)
3. **Vnější lokální** - privátní IP adresy zařízení ve vnější síti, jak jsou vidět ve vnitřní síti (170.16.0.2)
4. **Vnější globální** - veřejná IP adresa přiřazena koncovému zařízení ve vnější síti (100.100.1.1)

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.

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. Zahlcení tabulky by způsobilo větší odezvu a možnou ztrátu paketů. [2]

2.4 Druhy přiřazování adres

2.4.1 Statický NAT

Na směrovači, kde probíhá statický NAT, se překládá jedna IP adresa vždy na stejnou IP adresu. Každá vnitřní lokální adresa je vždy pevně namapovaná na stejnou vnější globální IP adresu. Statický NAT mapuje adresy 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.

2.4.2 Dynamický NAT (tzv. Maškaráda)

Většina směrovačů s NAT mapuje více zařízení v lokální síti na jednu veřejnou IP adresu. K mapování využívá dynamický NAT množinu vnitřních globálních a vnitřních lokálních adres a mapování dvoji probíhá dynamicky podle potřeby. [1] Výhodou je zpřístupnění vnější sítě mnoha zařízením přes několik globálních adres. Princip dynamického NATu:

- 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 lokální 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 lokální 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 ve NAT tabulce smaže.
- Pokud by se nyní chtělo připojit k serveru zařízení **B**, tak dostane stejnou vnitřní globální adresu, jako zařízení **A**.

Díky dynamickému NATu dosáhneme částečné úspory IPv4 adres.

2.4.3 Přetížený NAT (NAT overloading, Port Address Translation)

Přetížený NAT mapuje více vnitřních lokálních 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 lokální 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 lokální 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 lokální 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.

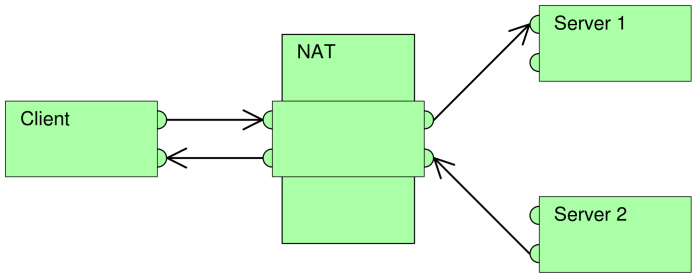
Přetížený NAT přináší výše zmiňované velké úspory IPv4 adres.

2.5 Druhy implementací NATu

Překlad síťových adres a portů lze implementovat několika způsoby. Některé aplikace, které používají informace o IP adrese, mohou potřebovat určit vnitřní globální IP adresu. Někdy může být potřeba určit typ mapování, pro navázání přímého spojení mezi dvěma klienty za NATem. Pro tento účel vznikl protokol Simple Traversal of UDP over NATs (STUN), který rozděluje jednotlivé NAT implementace. [3]

2.5.1 Full cone NAT (1:1 NAT)

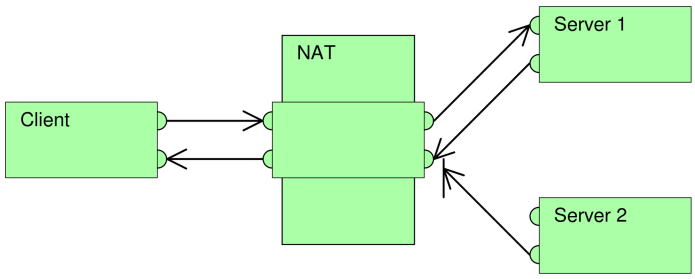
Full cone NAT je implementace, kde veškerá komunikace ze stejné vnitřní lokální IP adresy a portu je namapována na právě jednu stejnou vnitřní globální IP adresu a port. Jakékoliv zařízení ve vnější síti může kontaktovat zařízení ve vnitřní sítí odesláním paketu na mapovanou vnitnřní globální IP adresu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



Obrázek 2: Full cone NAT[4]

2.5.2 Restricted cone NAT

Restricted cone NAT je implementace, kde veškerá komunikace ze stejné vnitřní lokální IP adresy a portu je namapována na právě jednu stejnou vnitřní globální IP adresu a port. Na obrázku 3 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 3 může **klienta** kontaktovat pouze **server 1** a příchozí pakety ze **serveru 2** jsou blokovány směrovačem s NAT.



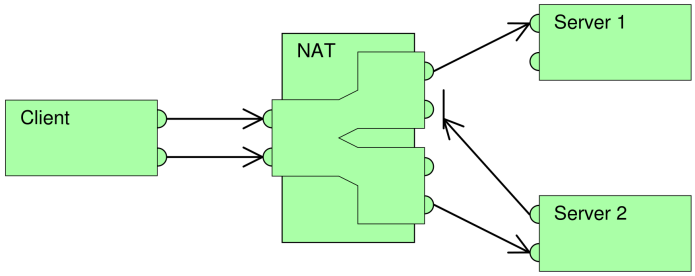
Obrázek 3: Restricted cone NAT[4]

2.5.3 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, pokud by tou samou adresou a portem kontaktován v minulosti.

2.5.4 Symetrický NAT

Symetrický NAT je implementace, kde jsou všechny pakety ze stejné vnitřní lokální IP adresy a portu odeslané na konkrétní cílovou IP adresu a port. Každá z těchto kombinací je mapována na vnitřní globální IP adresu a port. Paket může být poslán ze stejné vnitřní lokální 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 4.



Obrázek 4: Symetrický NAT[4]

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. [5]

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 místní sítě na jedinou veřejnou adresu IP, která se používá pro komunikaci se zařízeními mimo síť. To znamená, že zařízení v místní 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 soukromá IP adresa, která není viditelná pro zařízení mimo místní 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 místní 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 provoz dostal 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.

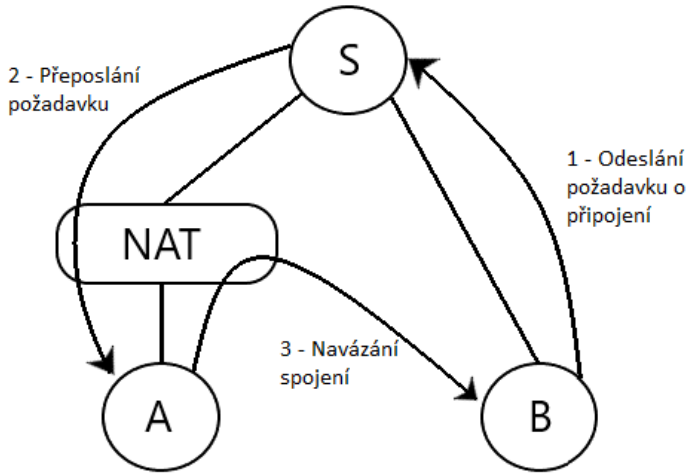
3.1 Traversal Using Relays around NAT (TURN)

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 moment, 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 za NATem, jak je ukázáno na obrázku 5. 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 5: 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 a TCP 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 lokální IP adresa a port zaznamenány na serveru (pokud se klient nenachází za NATem, tak jsou tyto údaje totožné). Princip metody, který je na obrázku 6, funguje následovně:

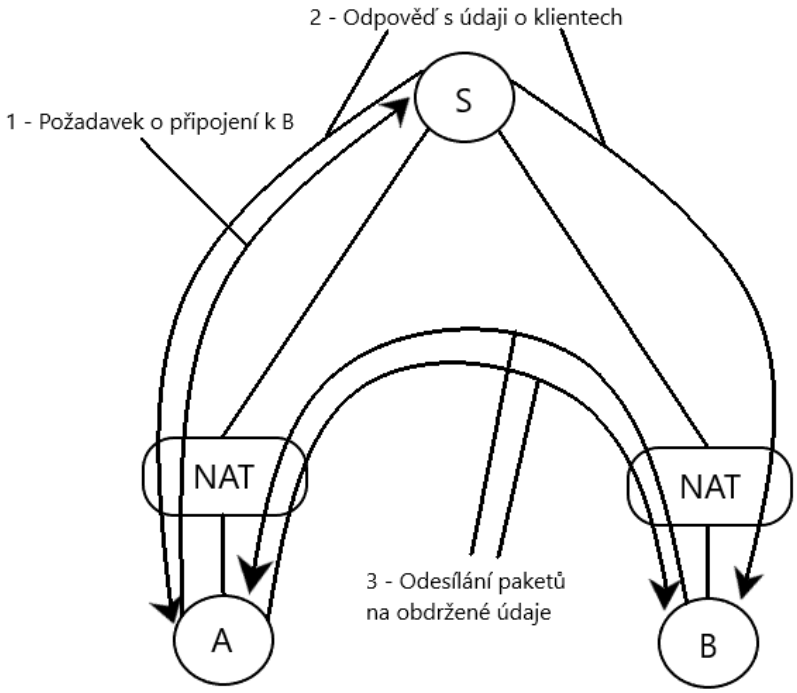
1. Klient **A** chce navázat UDP spojení s klientem **B**, ale ten je za NATem. Proto pošle žádost o připojení na server **S**.

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

3. Po obdržení zprávy ze serveru, začne klient **A** odesílat UDP pakety na vnitřní globální a vnitřní lokální údaje zároveň a čeká, ze kterých mu příjde

odpověď od klienta **B**. Jakmile obdrží klient **A** odpověď, naváže pomocí těchto údajů spojení s klientem **B**.

4. Klient **B** také začne odesílat UDP pakety na obdržené údaje a navazuje spojení po obdržení odpovědi od klienta **A**.



Obrázek 6: 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 spolehlivější, protože u protokol 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 místní port TCP, jak na poslouchání příchozích TCP spojení, tak na navazování TCP spojení. Většina operačních systémů v dnešní době podporují speciální nastavení

TCP socketu, pojmenované `SO_REUSEADDR`, které umožňuje aplikacím navázat několik socketů na jeden port. [6]

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 TCP spojení s klientem **B**, ale ten je za NATem. Proto pošle žádost o připojení na server **S**.

2. Server odpoví zprávou, která obsahuje vnitřní globální a vnitřní lokální 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, který odeslali na server a začnou na něm poslouchat pro příchozí pakety. Zároveň se s tímto portem budou pokoušet navázat spojení s druhým klientem pomocí obdržených údajů ze serveru.

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

5. 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. Po spojení a ověření se všechny ostatní pokusy o spojení zruší.

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).

3.4 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.

Protokol UPnP (Universal Plug and Play) poskytuje funkci pro automatické přesměrování vnitřních globálních portů do vnitřních lokálních portů. Aplikace mohou využít protokol UPnP k rezervaci portu a přesměrovat příchozí pakety na socket, na kterém aplikace poslouchá. Využití protokolu UPnP může být riziko zabezpečení, protože UPnP nepodporuje žádné ověření příchozích paketů.

4 Aplikace

Cílem práce je také demonstrovat vybrané metody NAT Traversal v reálném síťovém prostředí. Tato část se zabývá využitým technologiím, 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 na operační systém Windows. Při vývoji v .NET Framework se využívají programovací jazyky `C#`, `F#` a Visual Basic. [7]

`Windows Forms`
: Windows Forms je knihovna tříd součástí .NET Frameworku. 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. [8]

`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 mám hlavní formu, ve které jsou všechny pevné prvky. Tedy navigace, hlavní lišta s návem aktuální stránky a název aplikace v levém horním rohu. Hlavní forma obsahuje také panel, do kterého se zobrazí sekundární forma. Sekundární forma se mění po zmáčnutí k ní přiděleného tlačítka. Například po zmáčknutí tlačítka Hole Punching se zvýrazní zmáčknuté

tlačítko a do panelu pro sekundární formu 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 sekundární formy podle zmáčnutého tlačítka (btnSender).

Jelikož se sekundární formy po zavření resetují, bylo nutné přemístit globální proměnné nebo metody do hlavní formy. To bylo důležité obzvlášť u Hole Punching metody, protože je potřeba, aby se spojení se serverem nepřerušilo. Hlavní forma také obsahuje hodnoty některých textů, které bylo potřeba zachovat i po zavření formy. V Hole Punching formě to je komunikace serveru. V Port Mapping formě to je tabulka existujících přesměrování, aby uživatel nemusel po znovuotevření formy opětovně nechat vyhledávat existující směrování.

4.2.2 Port Mapping

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

Po načtení formy se spustí kód 2, který vypíše do check boxu všechná 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 vypíšou 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 časti přidá pouze vytvoření směrování se zadanými údaji z text boxů (Kód 3).

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

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

Zdrojový kód 3: Vytvoření směrování

4.2.3 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 zpráva obsahuje číslo 0, záznam ze serveru smaže. Pokud se jedná o jiné číslo než 0, server odpoví, že už o klientovi záznam má.
4. Pokud zprávu 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 nějakým uloženým klíčem v databázi serveru, tak server přepošle údaje k připojení jednotlivým klientům 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 dát příjem a odesílání dat na oddělená vlákna.

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. 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.

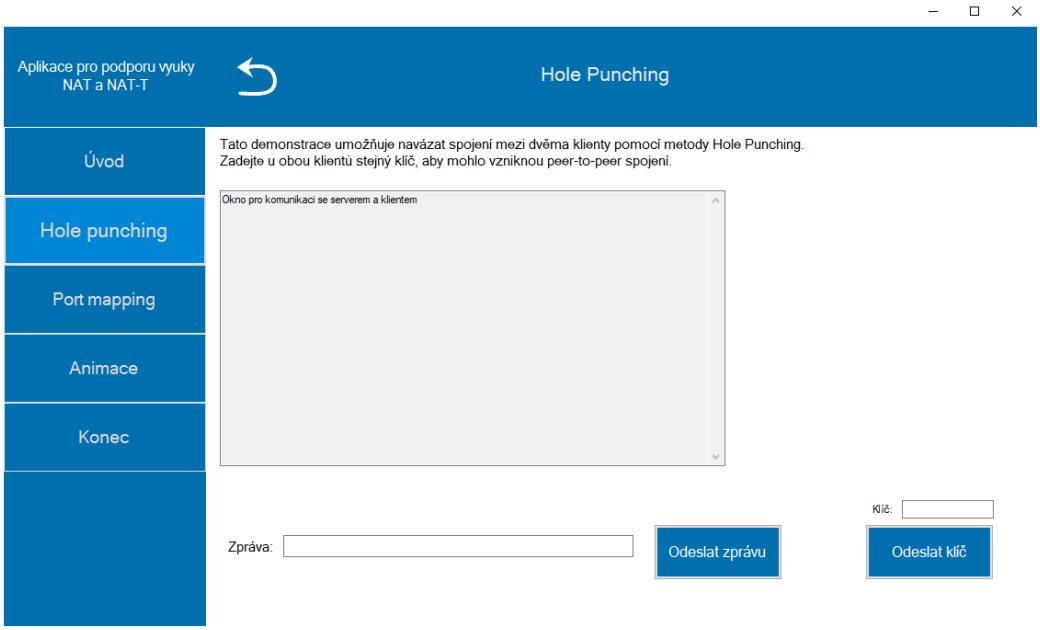
4.3.1 Používání aplikace

Aplikace se spouští pomocí **exe** souboru obsažený ve složce `Mazur_NAT-T/bin/Release` nebo můžete spustit instalační soubor ve stejné složce s příponou **application**. Po spuštění se zobrazí domovská obrazovka, jako na obrázku 7.

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.

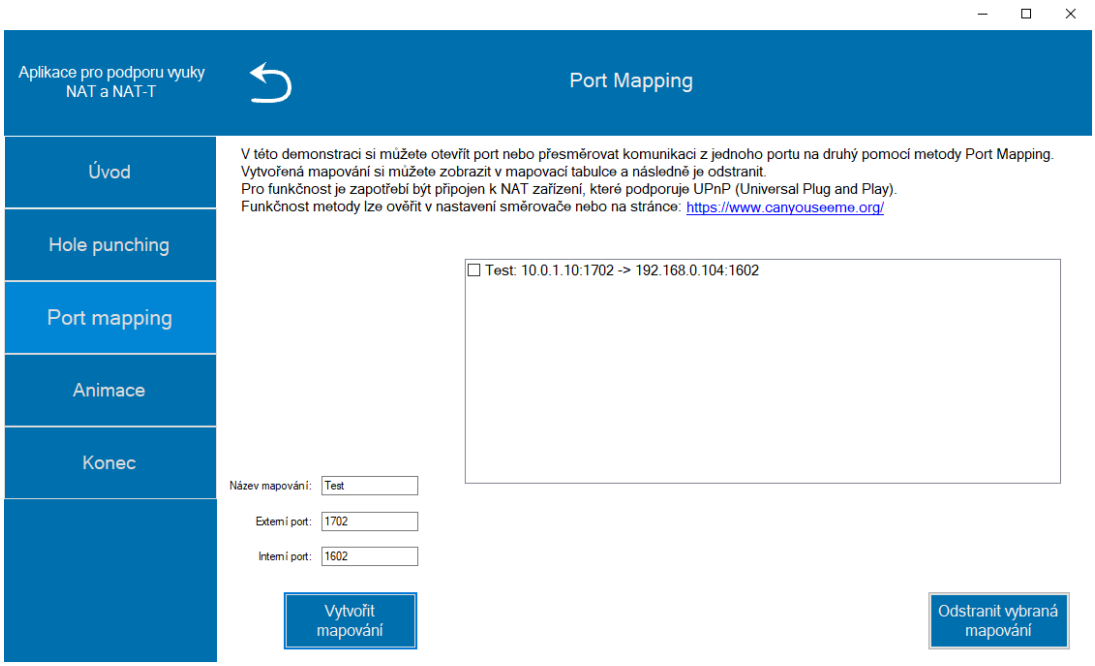
Obrázek 7: Úvodní obrazovka aplikace

Komunikace se serverem je zahájena od načtení aplikace, tudíž jediné co je potřeba, tak je odeslat unikátní klíč. Pro vytvoření spojení mezi dvěma klienty za NATem metodou Hole punchingu musí na obou počítačích běžet tato aplikace. Poté musí odeslat na server stejný klíč a aplikace vytvoří spojení (viz obrázek 8). Nyní si mohou oba klienti vyměňovat textové zprávy.



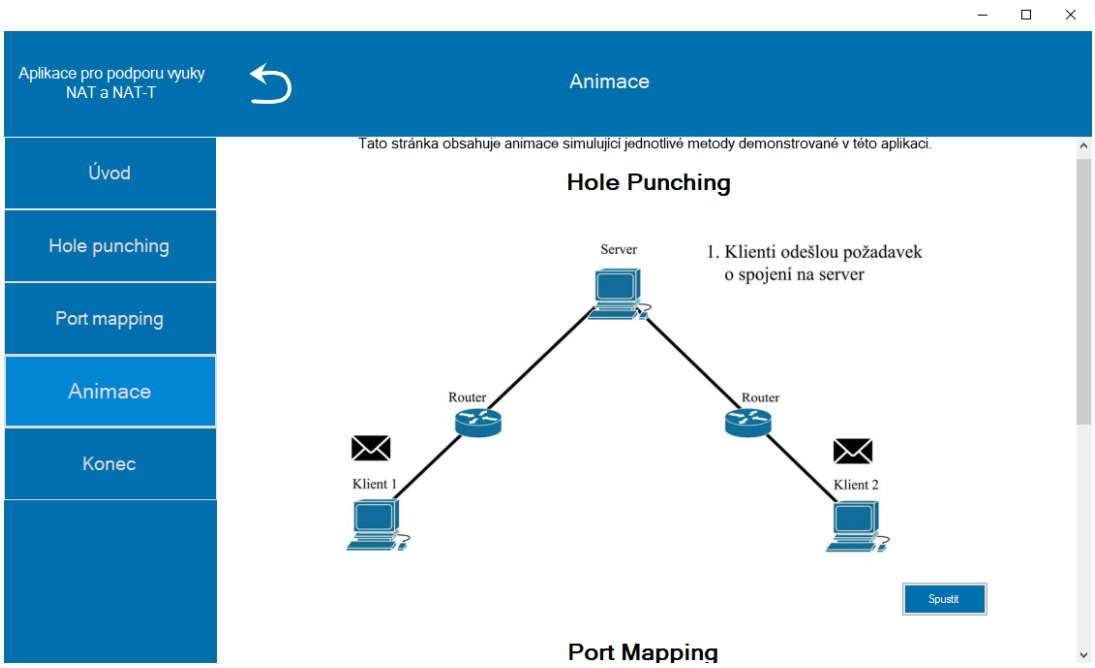
Obrázek 8: Obrazovka Hole punching

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 9). Funkčnost mapování muže uživatel ověřit pomocí stránky uvedené v aplikaci nebo v nastavení routeru. Odstranění mapování je možné pomocí výběru mapování, které chceme odstranit a následném zmáčknutí tlačítka *Odstranit vybraná mapování*.



Obrázek 9: Obrazovka Port mapping s vytvořenými mapováními

Tlačítko *Animace* otevře obrazovku s jednoduchými animacemi simulující 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 10).



Obrázek 10: 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. 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.

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.

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] 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⟩.

[3] 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⟩.

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

[5] 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⟩.

[6] 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/⟩.

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

[8] 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⟩.