VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2021 Bc. Martin Biolek

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

KLIENTSKÁ APLIKACE PROTOKOLU DNS S GRAFICKÝM ROZHRANÍM PRO ÚČELY VÝUKY
DNS CLIENT APPLICATION WITH A GRAPHICAL INTERFACE FOR TEACHING PURPOSES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Martin Biolek**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Jan Jeřábek, Ph.D.**
SUPERVISOR

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Bc. Martin Biolek ***ID:*** 191382
Ročník: 2 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Klientská aplikace protokolu DNS s grafickým rozhraním pro účely výuky

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte z literatury problematiku protokolů DNS, mDNS a LLMNR a také chování programů nslookup, DNS resolver, DNS benchmark a podobných nástrojů. V rámci diplomové práce proveďte návrh vlastního klienta protokolu DNS, který bude mít obdobnou funkcionalitu jako DNS resolver, z jehož zdrojových kódů můžete vyjít. Ve vaší aplikaci bude konfigurace DNS dotazu a i výstup možný v přehledném grafickém rozhraní pro operační systém Windows. Vytipujte vhodný programovací jazyk a vývojové prostředí, proveďte návrh funkčnosti budoucího programu včetně popisu plánovaných funkcí či modulů. Vytvořené dílo musí splňovat nejnovější specifikace DNS, mDNS a LLMNR protokolu a podporovat konkrétní funkcionality, jako je např. i DNSsec. Kompletní specifikace aplikace musí být konzultována a schválena vedoucím práce. Proveďte implementaci jádra programu a taktéž grafického rozhraní. Výslednou aplikaci důkladně otestujte a zdokumentujte její vlastnosti, stejně tak jako zdrojové kódy. Vytvořte ukázky možného použití ve výuce předmětů zaměřených na komunikační technologie.

DOPORUČENÁ LITERATURA:

[1] Kurose, J. F., Ross, K. W., Computer networking: a top-down approach. 7th global ed. Essex: Pearson, 2017, 852 s. ISBN 978-1-292-15359-9.

[2] JEŘÁBEK, J. Pokročilé komunikační techniky. Skriptum FEKT Vysoké učení technické v Brně, 2020. s. 1-180.

Termín zadání: 1.2.2021 ***Termín odevzdání:*** 24.5.2021

Vedoucí práce: doc. Ing. Jan Jeřábek, Ph.D.

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

ABSTRAKT

Cílem diplomové práce na téma Klientská aplikace protokolu DNS s grafickým rozhraním pro účely výuky je vytvoření programu s možností zasílání, přijímání zpráv protokolu DNS, mDNS a LLMNR s možnostmi volby volitelných parametrů protokolu. Dále porovnat vytvořenou aplikaci s dostupnými nástroji jako jsou Nslookup, Dig a vytvoření příkladů využití aplikace pro výuku.

KLÍČOVÁ SLOVA

Domain Name System, Multicast Domain Name System, Link-Local Multicast Name Resolution, Domain Name System over HTTPS, Domain Name System over TLS, Domain Name System over Quic, resolver, klientská aplikace pro Domain Name System

ABSTRACT

The goal of the Master thesis on the topic of the Client application of DNS protocol with graphical interface for teaching purposes is to create a program with the features of sending, receiving DNS, MDNS and LLMNR protocols with optional parameters. Additionally, compare the created application with available tools such as Nslookup, Dig and create examples of application for teaching.

KEYWORDS

Domain Name System, Multicast Domain Name System, Link-Local Multicast Name Resolution, Domain Name System over HTTPS, Domain Name System over TLS, Domain Name System over Quic, resolver, client application for Domain Name System

BIOLEK, Martin. *Klientská aplikace protokolu DNS s grafickým rozhraním pro účely výuky*. Brno, 2021, 108 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: doc. Ing. Jan Jeřábek, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Klientská aplikace protokolu DNS s grafickým rozhraním pro účely výuky“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora

Tato práce vznikla jako součást klíčové aktivity KA6 - Individuální výuka a zapojení studentů bakalářských a magisterských studijních programů do výzkumu v rámci projektu OP VVV Vytvoření double-degree doktorského studijního programu Elektronika a informační technologie a vytvoření doktorského studijního programu Informační bezpečnost, reg. č. CZ.02.2.69/0.0/0.0/16_018/0002575.

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

Projekt je spolufinancován Evropskou unií.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu doc. Ing. Janu Jeřábkovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Úvod

Diplomová práce má za cíl seznámit čtenáře s problematikou doménových jmen a jejich překladu pomocí různých protokolů.

Teoretická část pojednává o hierarchii doménových jmen a systému překladu jmen na Internet Protocol adresu. Dále je probrán formát zpráv Domain Name System protokolu a rozšíření protokolu Domain Name System Security Extension, které reaguje na možnost původní nedostatky původního protokolu. Pro překlad jmen na lokální síti jsou zmíněny protokoly Multicast Domain Name System protokol a Link Local Multicast Resolution Name protokol. Dále jsou rozebrány šifrované protokoly pro překlad doménových jmen. Mezi popsané protokoly patří DNS over Transport Layer Security, DNS over Hypertext Transfer Protocol Secure a DNS over Quic.

Poslední menší kapitola teoretické části pojednává o možné budoucnosti DNS. Zaměřuje se primárně na skloubení DNS a blockchain technologie s reálným příkladem jako řešení přenesení překladu doménových jmen na blockchain Etherea.

Praktická část se skládá ze tří témat. První téma rozebírá nástroj DNS benchmark, který slouží k zvolení nejlepšího veřejně dostupného resolveru pro aktuální stanici. Měření probíhalo ze dvou rozdílných sítí: z domácí sítě a sítě VUT.

Dále se největší oblast praktické části skládá z popisu implementace klientské aplikace pro zasílání a zobrazení dotazů pro protokoly DNS, mDNS a DoH. Aplikace je implementována pomocí programovacího jazyka Java a vyžaduje pro svůj běh Javu v minimální verzi 11 a knihovnu JavaFX, která je součástí složky, kde je i výsledný spustitelný soubor aplikace. Vytvořená grafická klientská aplikace je porovnávaná s již existujícími nástroji Nslookup a Dig.

Vývoj je veřejně dostupný v repozitáři na Githubu, kde lze najít i jednotlivé vydání aplikace. V části DNS jsou rozebrány rozdíly v resolverech, které byly pozorovány při testování klientské aplikace. Dopodrobna je zaznamenán rozdíl mezi resolvery při stejném typu dotazu. Současně všechny tři protokoly obsahují výstup z testování jednotlivých vlastností programu, aby bylo patrné, co bylo předmětem vývoje a s jakými uživatelskými vstupy si aplikace umí poradit. V poslední kapitole je popsán proces importu projektu do vývojového prostředí Eclipse a export až do spustitelné .exe aplikace.

1 Teoretická část

1.1 DNS

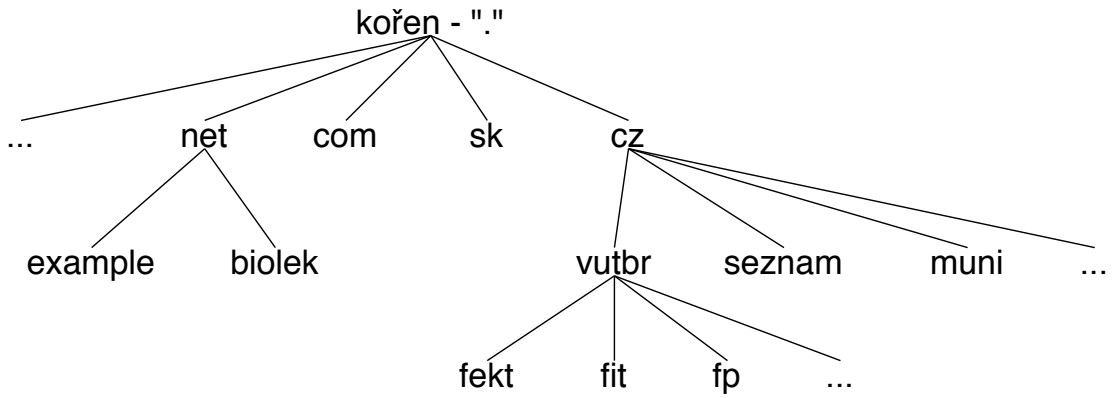
DNS je zkratka pro protokol Domain Name System. Jedná se o protokol pracující na 7. vrstvě ISO/OSI (International organization of Standardization/Open System Interconnection) modelu. Z pohledu dnes častěji využívaného síťového modelu TCP/IP (Transmission Control Protocol/Internet Protocol) se jedná o 4. vrstvu. Oba modely protokol zasazují do aplikační své vrstvy. Primární funkce je přeložení doménového jména na IP (Internet Protocol) adresu. Uživatel si nemusí pamatovat číselnou IP adresu, ale pouze srozumitelné řetězce znaků a protokol DNS se postará o překlad. V dnešní době má protokol další funkce, avšak tato základní je využívána nejvíce. [1]

1.1.1 Vymezení pojmů v rámci DNS

Resolver je stanice nebo software starající se o překlad doménového jména a to napříč stromovou strukturou doménových jmen.
Zóna je část domény, kterou organizace/osoba spravuje.
Autoritativní DNS server je DNS server, který zná veškeré informace o spravované zóně.
Rekurzivní dotaz je typ dotazu, kdy klient požaduje od serveru konečný překlad i za cenu, že server bude muset vykonat další dotazy, než překlad bude moci provést.
Iterativní dotaz je druhý typ dotazu, kdy si klient vystačí s odpovědí bez překladu doménového jména, ale s odkazem na adresu DNS serveru níže v hierarchii.

1.1.2 Historie

První podoba doménových jmen se objevila v síti ARPANET (v 70. letech 20. století), kde Stanford Research Institute udržoval soubor jménem HOSTS.TXT. V souboru bylo mapováno jméno počítače v síti na číselnou adresu. Jednalo se tedy o silně centralizovaný seznam, který bylo velmi nepružné měnit. Soubor si uživatelé stahovali pomocí protokolu FTP (File Transfer Protocol) a po stažení využívali překlady na stanici. Tento koncept využíváme i dnes při definování doménových jmen na lokální stanici. U Unixových operačních systémů v rámci souboru */etc/hosts*. U Windows operačních systémů v souboru *C:\windows\system32\drivers\etc\hosts*.



Obr. 1.1: Struktura hierarchie doménových jmen

Následovala specifikace protokolu DNS, vytvořena Paulem Mackapetrisem. Specifikace se nachází v dokumentech RFC (Request for Comments) 1034 a 1035. Samotný standart byl uznán v roce 1986, poté prošel dalšími změnami v novějších RFC. Asi nejvýznamnější úprava je z roku 2005, kdy byl představen DNSSEC (Domain Name System Security Extensions). Rozšíření reagující na možnost podvrhnutí doménového jména za jinou IP adresu. Přidává ověření pomocí asymetrické kryptografie.[2]

1.1.3 Vlastnosti protokolu

Protokol je typu klient–server. Jedná se o aplikační protokol využívající jak UDP (User Datagram Protocol) i TCP (Transmission Control Protocol) na transportní vrstvě. Oba využívají port 53 (na straně serveru). V případě překladu doménového jména se nejčastěji využije protokol UDP, protože je zde důležitá rychlost překladu. Pokud protokol UDP nemůže přenést informace o překladu, protože například data jsou větší než povolená velikost datagramu, naváže klient spojení a informaci získá pomocí spolehlivé služby TCP. Protokol obsahuje dva druhy zpráv: dotaz a odpověď.

Hierarchie DNS serverů

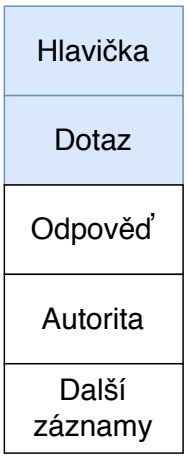
Hierarchii tvoří decentralizovaný systém serverů. Byla koncipována tak, že každá instituce si chce obhospodařovat vlastní část doménových jmen, aby nedošlo k problému stejnému jako u projektu ARPANETu. Struktura serverů je stromová, kdy kořen je tečka a pod ní se větví jednotlivé TLD (Top Level Domain). Mezi takové domény patří například doména .cz nebo .com. Pod TLD se nachází další domény. Struktura je zobrazena na obrázku 1.1. Ve stromu jsou domény odděleny tečkou (tečka reprezentuje cestu mezi listy). Maximální počet znaků

na doménové jméno je 255. Jednotlivé části (subdomény), mohou mít maximálně 63 znaků. Maximální počet úrovní je 127 (nejdelší možná cesta ve stromu).

V každé doméně existuje alespoň jeden autoritativní server, který zná veškeré doménové záznamy, nebo odkáže na server níže v hierarchii, který překlad může vykonat.

1.1.4 Formát zpráv

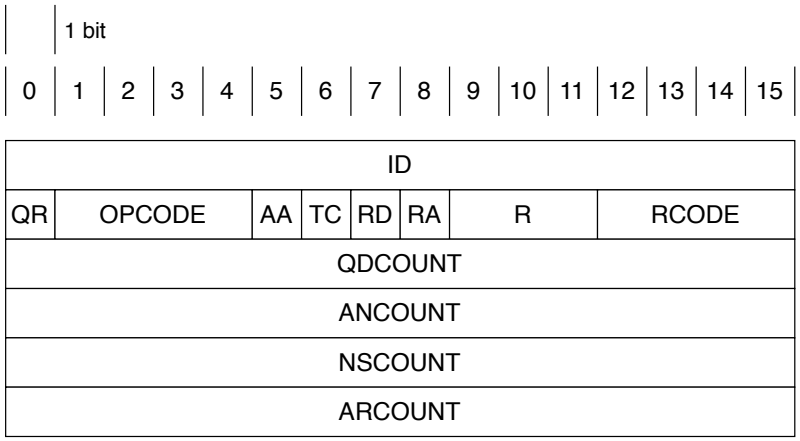
Zprávy protokolu jsou popsány ve více doporučení RFC. Nejzásadnější je RFC 1035, které navazuje na RFC 883 a 973. Jednotlivé části jsou povinné nebo volitelné. Povinné musí být ve zprávě vždy. Jedná se o části hlavička a dotaz. Další části již jsou volitelné: odpověď, autorita a další záznamy. Obecná struktura zprávy je zobrazena na obrázku 1.2. [3]



Obr. 1.2: Obecná struktura zprávy protokolu DNS

Hlavička

Hlavička obsahuje 6 polí s různou bitovou délkou. Znázorněna je na obrázku 1.3.
ID – Je 16 bitové číslo. Je generováno náhodně pro každou relaci dotaz/odpověď. Pro odpověď musí být použité stejné číslo jako v dotazu. Slouží k identifikaci k jakému dotazu se odpověď vztahuje.
QR – Jeden bit rozlišující, zda zpráva je dotaz (0) nebo odpověď (1).
OPCODE – Identifikace druhu dotazu. Stejně jako ID je i OPCODE kopírován z dotazu do odpovědi. Pole může nabývat hodnot 0 až 15. Využívané jsou aktuálně 3 hodnoty. Jedná se o obyčejný (0), reverzní (1) a stav serveru (3). Zbytek hodnot je rezervován pro další využití.
AA – Bit označuje autoritativní odpověď (1). V případě, že v odpovědích je více



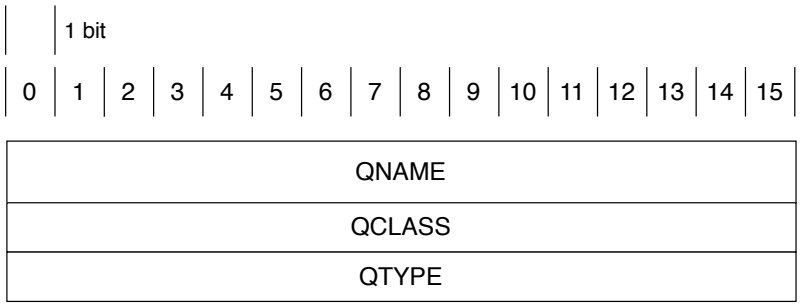
Obr. 1.3: Struktura hlavičky protokolu DNS

záznamů, stahuje se hodnota pole pouze na dotaz, který byl první v dotazu.
TC – Bit významově reprezentuje fragmentaci zprávy kvůli délce. Využito při přenosu přes UDP.
RD – Rekurze je vyžadována. Bit je nastaven v dotazu, když klient chce, aby server vyhledal odpověď a poslal až výsledný překlad. Podpora rekurze není povinná. Kořenové servery těmto požadavkům nemusí vyhovět.
RA – Nastavení bitu v odpovědi. Server dává informaci, že rekurzi podporuje.
R – 3 bitová rezerva, které je volná pro další využití. Ve všech zprávách musí být nastaveno na hodnotu 0.
RCODE – Reprezentuje výsledek překladu. Nabývá hodnot 0 až 15, kdy je specifikováno aktuálně 6 hodnot: bez chyby (0), chyba formátu dotazu (1), chyba serveru (2), server není autoritativní pro danou zónu (3), není implementováno (4), zamítnutí požadavku (5).
QDCOUNT – Zkratka pro Query Domain Count. Číslo o velikosti 16 bitů reprezentuje počet dotazů ve zprávě.
ANCOUNT – Reprezentuje počet záznamů v odpovědi. Jedná se o zkratku Answer Count. Číslo vyjadřuje počet přeložených odpovědí.
NSCOUNT – Hodnota 16 bitového čísla slouží k zjištění počtu záznamů v části autorita, neboli kolik záznamů pochází přímo z DNS serveru pro danou zónu.
ARCOUNT – Zkratka pro Additional Information Count, počet záznamů v další záznamy. Hodnotou lze zjistit, zda byl využit DNSSEC (0 znamená, že ne).[4]

Dotaz

Obsahuje tři pole. Schéma dotazu je zobrazen na obrázku 1.4.
QNAME – Reprezentace doménového jména. První je číslo, které reprezentuje

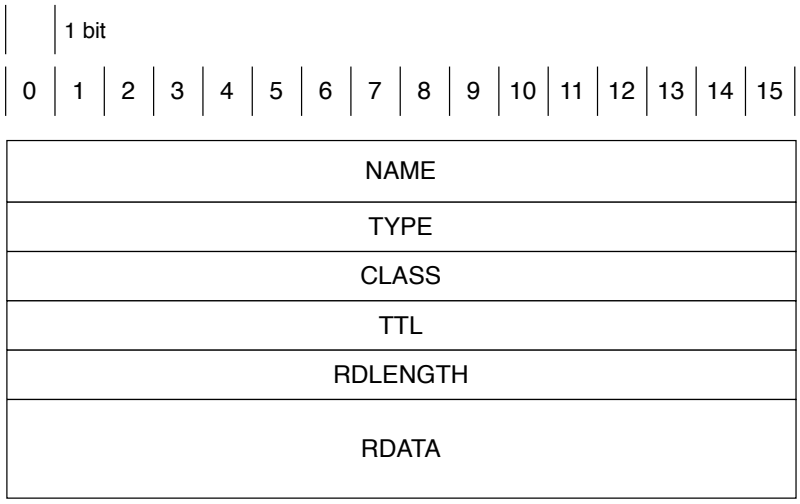
počet znaků po první tečku. Následuje část subdomény po tečku. Poté se postup opakuje. Poslední je 0 reprezentující kořen (.).
QCOUNT – Požadavek na typ dotazu. Dotaz na záznam A je reprezentován hodnotou 1.
QTYPE – Pole pro třídu dotazu. Aktuálně se používá pouze hodnota 1 (IN) pro internet.[5]



Obr. 1.4: Schéma bloku dotazu protokolu DNS.

Odpověď

Část odpovědi je zobrazeno na obrázku 1.5. Jedna zpráva může obsahovat více odpovědí. Počet lze vyčíst z hlavičky z pole QDCOUNT. Stejný typ pole je využit v částech autoria a dalších záznamech. V dalších záznamech může být přenášen záznam OPT, který má stejnou strukturu jako odpověď, ale položky mají jiný význam.
NAME – Doménové jméno, ke kterému se výsledek vztahuje.
TYPE – 16 bitové číslo určující typ záznamu. Záznam typu A odpovídá číslu 1.
CLASS – Třída, kde daný záznam je validní. Měl by odpovídat třídě z dotazu. Aktuálně využívána pouze hodnota 1 pro internet.
TTL – Doba platnosti záznamu neboli, jak dlouho si mohou zařízení po síti uložit odpověď do paměti cache. Číslo odpovídá počtu sekund.
RDLENGTH – Číselná reprezentace počtu oktetů v části RDATA, která je závislá na typu záznamu.
RDATA – Záznam, který odpovídá typu a třídě z předešlých polí. Délka odpovídá poli RDLENGTH.



Obr. 1.5: Struktura části DNS odpovědi.

1.1.5 Typy záznamů

Záznamy slouží pro uložení informací o zóně. Pokud chce správce sítě změnit konfiguraci zóny, provádí ji cestou změny záznamů. V protokolu je definováno velké množství různých typů záznamů. Ne všechny se často využívají, proto následující seznam je výběr s největší četností.

A – Záznam mapuje doménové jméno na IPv4 adresu. Nejedná se o základní záznam, kvůli kterému protokol vznikl.

AAAA – Záznam mapuje doménové jméno na IPv6 adresu.

CNAME – Mapuje doménové jméno na jiné doménové jméno. Nejčastěji se využívá při konfigurace více služeb na jedné stanici. Existuje pouze jeden záznam typu A (nebo AAAA) a několik záznamů typu CNAME, které odkazují právě na tento záznam. Je-li stanice přesunuta v rámci sítě, stačí změnit pouze záznam typu A (nebo AAAA). Pokud by každá služba musela být mapována zvlášť, bylo by potřeba vícero změn.

MX – Záznam odkazuje na mailový server pro danou doménu (server příchozí mailové pošty).

NS – Odkaz na jiný DNS server, který je odpovědný za danou část zóny.

SOA – Povinný záznam, který je vždy jeden v databázi. Jedná se o zkratku Start of Authority record. Záznam obsahuje: jméno zóny, jména DNS serverů pro danou zónu, sériové číslo a parametry časovačů. Využívá se při synchronizaci záznamů mezi primárním a sekundárními servery. Po každé změně na primárním serveru je hodnota sériového čísla inkrementována. V pravidelných intervalech si sekundární server záznam vyžádá a porovná sériová čísla. V případě, že hodnota není stejná, začne transfer zóny (synchronizace záznamů).

PTR – Záznam slouží k reverznímu mapování. A/AAAA záznam mapuje doménové jméno na IP adresu. PTR záznam mapuje IP adresu na doménové jméno. Záznam je vytvořen IPv4 adresou zapsanou oktetově pozpátku a k ní přidána doména.in-addr.arpa. Pro příklad doménové jméno www.vutbr.cz má IPv4 adresu 147.229.2.90. Záznam PTR bude: 90.2.229.147.in-addr.arpa www.vubr.cz. U IPv6 adres je postup obdobný. Adresa je zapsaná celá pozpátku a každý symbol oddělen tečkou, poté je přidáno .in-addr.arpa. Záznamy slouží pro zjistění, jaká doménová jména jsou mapována na danou IP adresu. Lze využít k detekci phishingu.
RRSIG – Přidán při implementaci DNSSECu. Slouží k podpisu všech stejných typů záznamů. Obsahuje data podpisu, typ záznamů a algoritmus pro podpis.
DNSKEY – Jedná se o záznam veřejného klíče. Záznamy RRSIG jsou podepsány privátním klíčem, který se nesmí odtajnit. Veřejný klíč je uložen v záznamu typu DNSKEY.
DS – Slouží k uložení otisku klíče. Je vkládán do doménového serveru vyšší úrovně k vytvoření řetězu důvěry. Otisky klíčů pro kořenovou zónu musí být vloženy manuálně do resolveru při konfiguraci.
CAA – Certification Authority Authorization je záznam, kde se definují certifikační autority pro danou zónu, neboli jaké certifikační autority jsou validní u domén ze zóny.
CERT – Záznam pro uložení certifikátů. Jeden z možných způsobů bezpečné distribuce certifikátů. Zejména, pokud pro oveření je využit DNSSEC.
TXT – Uložení textového řetězce. Často se používá k zapsání kontaktních údajů o správci domény.
OPENPGPKEY – Certifikátu typu X.509 pro šifrování a ověření podpisu emailové korespondence.
OPT – Samotný záznam není uložen v databázi zóny. Ve zprávě je vložen mezi dalšími záznamy. Záznam má stejnou strukturu jako odpověď. Jednotlivé položky odpovídají jiným atributům. V části RDATA jsou uloženy páry $< atribut,\ hodnota >$. Své využití má zejména v rámci DNSSEC zpráv.
NSEC – Při dotazu na neexistující doménu zavádí DNSSEC odpověď se záznamem NSEC. Útočník nemůže zasílat tedy odpovědi, že doména neexistuje. Pokročilejší záznam, který vychází z NSEC záznamu je NSEC3. Oba jsou více popsány v kapitole 1.2.4.
NSEC3 – Plní stejnou funkci jako záznam NSEC včetně vlastnosti, že skrývá doménové jméno za hash doménového jména. Uživatel si musí ověřit, že zpráva je validní pomocí DNSSECu, ale již nemůže vidět samotné doménové jméno. Zajišťuje nemožnost získat veškeré informace zóně jako je tomu u NSEC záznamu.[6]

1.1.6 Kořenové DNS servery

Pro zajištění decentralizace je potřeba mít více kořenových serverů nezávislých na sobě v různých geolokacích. Kořenové servery nedělají překlad domény na IP adresu, ale mají záznamy, kde překlad bude úspěšný. Znají adresy DNS serverů TLD. Lokace a adresy všech kořenových serverů je možné získat

Tab. 1.1: Seznam kořenových doménových serverů

Jméno	IPv4 adresa	IPv6 adresa
a.root-servers.net	198.41.0.4	2001:503:ba3e::2:30
b.root-servers.net	199.9.14.201	2001:500:200::b
c.root-servers.net	192.33.4.12	2001:500:2::c
d.root-servers.net	199.7.91.13	2001:500:2d::d
e.root-servers.net	192.203.230.10	2001:500:a8::e
f.root-servers.net	192.5.5.241	2001:500:2f::f
g.root-servers.net	192.112.36.4	2001:500:12::d0d
h.root-servers.net	198.97.190.53	2001:500:1::53
i.root-servers.net	192.36.148.17	2001:7fe::53
j.root-servers.net	192.58.128.30	2001:503:c27::2:30
k.root-servers.net	193.0.14.129	2001:7fd::1
l.root-servers.net	199.7.83.42	2001:500:9f::42
m.root-servers.net	202.12.27.33	2001:dc3::35

z `https://root-servers.org/`. Klienti mají tento seznam definovaný v rámci softwaru. Seznam je zobrazen v tabulce 1.1.

Servery jsou pojmenované písmeny A až M. Jedno písmeno představuje jeden stejný typ serveru, který vystupuje pod veřejně známou IP adresou. Pro směrování se využívá anycast, kdy v síti je vícero zařízení se stejnou IP adresou, ale pouze jeden komunikuje s klientem. Nejčastěji to bývá ten, který je nejblíže od klienta. Tímto způsobem je možné službu decentralizovat a nespoléhat se pouze na centrální prvek.

Kořenové servery nereagují na žádosti o rekurzivní dotaz. Daný požadavek zahodí, jelikož vyhledávání v rámci struktury by mohlo vést k odepření služby jiným uživatelům. Pro využití rekurzivního dotazu je proto nutné využít DNS server nižší úrovně.[7]

1.1.7 Resolver

Resolver se stará o kompletní překlad od kořene až po koncové doménové jméno. O překlad v rámci hierarchie se nestará koncová stanice, která překlad požaduje,

ale DNS server, který je její primárním DNS serverem. Na něj dotaz posílá. V lokálních sítích tuto úlohu plní DNS server pro danou síť. Sám tedy vystupuje jako server i klient zároveň, protože posílá požadavky za stanice na další servery.

Primární DNS server je definován v nastavení síťového rozhraní. Lze konfigurovat ručně nebo využít protokol DHCP (Dynamic Host Configuration Protocol), který nastaví jak adresu sítě, stanice, masku, ale může nastavovat i primární a sekundární DNS server.

V případě, že se resolver v lokální síti nenachází, často lze využít resolver poskytovatele nebo veřejné resolvery. Veřejné resolvery jsou zdarma, ale stejně jako u jiných neplacených služeb jsou data, která jsou vyhledávána, využita pro komerční užití. Seznam nejznámějších veřejných resolverů je zobrazen v tabulce 1.2.[8]

1.1.8 Průběh dotazování

Průběh překladu doménového jména v případě, že jméno není cachováno na stanici ani na resolveru. Nejprve se stanice podívá na statické lokální záznamy. Pokud záznam neexistuje, může vyzkoušet protokol LLMNR (Link-Local Multicast Name Resolution) nebo mDNS (Multicast DNS) pro překlad v lokální síti. V případě, že překlad nebyl úspěšný, přichází na řadu protokol DNS.

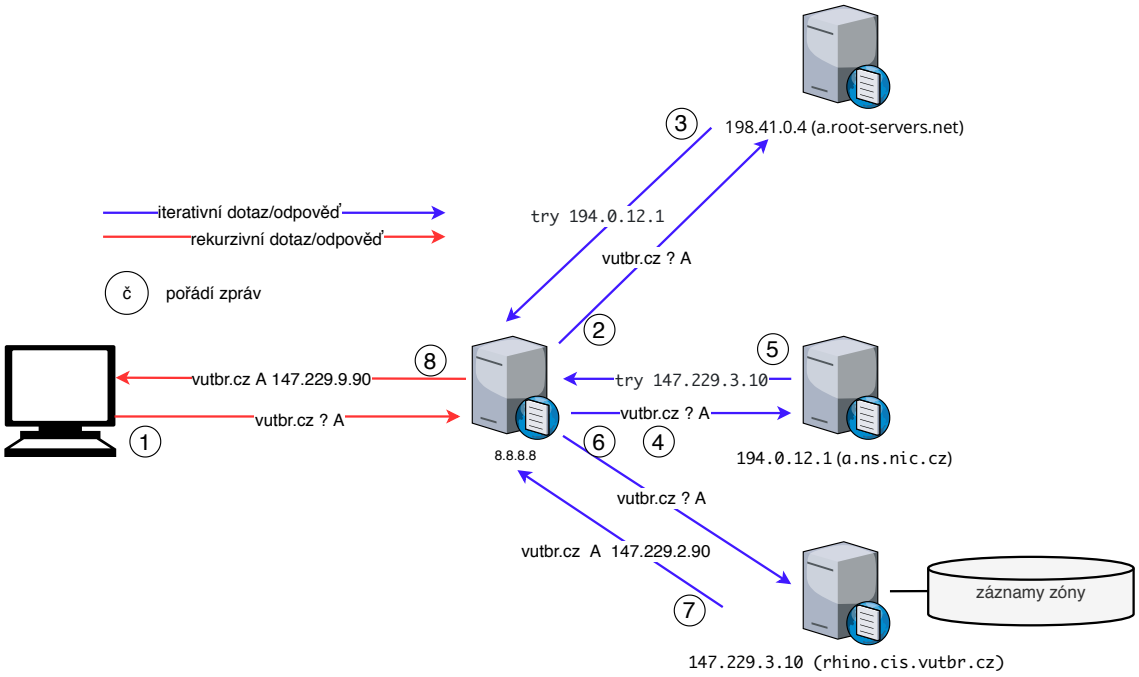
Stanice kontaktuje svůj primární DNS server s rekurzivním dotazem. Resolver nejpravděpodobněji nebude zároveň DNS server spravující zónu. Zašle iterativní dotaz na kořenový server. Kořenový server nemá daný záznam v cache paměti, ani nemá záznam ve své databázi. Odkáže jej na server DNS pro zónu .cz, pro který má v databázi NS záznam. Průběh dotazování je zobrazeno na obrázku 1.6.

Resolver v dalším kroku kontaktuje server zodpovědný za zónu .cz. Ten již může mít záznam v cache paměti a rovnou tedy odpovědět. Pokud ho ale v paměti nemá, následuje odpověď stejného typu jako od kořenového serveru, ale se záznamem NS vutbr.cz.

V dalším kroku znovu posílá iterativní dotaz tentokrát na server spravující zónu vutbr.cz. Ten již záznam A má v databázi a posílá odpověď. Resolver odpověď zpracuje (uloží do cache paměti podle TTL položky v odpovědi, validuje DNSSEC) a pošle rekurzivní odpověď stanici.[9]

1.1.9 Nástroje Whois, Nslookup a Dig

Služba Whois slouží k získání informací o provozovateli domény. Jednotliví registrátoři shromažďují informace při registraci domény. Vzniklé databáze jsou přístupné pomocí služby Whois. Lze ji využít pro získání jmen (emailové



Obr. 1.6: Příklad překladu doménového jména pomocí protokolu DNS

adresy) osob zodpovědných za správu domény. Služba je dostupná na všech platformách pomocí příkazového řádku. Aktuálně existuje možnost osobní údaje nezobrazovat. Velké množství registrátorů funkci podporuje a nezobrazuje osobní údaje.[10]

Výpis 1.1: Příklad využití nástroje Nslookup

```
1 $ nslookup www.vutbr.cz
2 Server:    192.168.0.1
3 Address:   192.168.0.1#53
4
5 Non-authoritative answer:
6 Name: www.vutbr.cz
7 Address: 147.229.2.90
```

Nslookup slouží k přeložení doménového jména (popřípadě reverznímu překladu) pomocí příkazové řádky. V operačním systému nelze zjistit na jakou adresu je doménové jméno přeloženo. Použití nástroje Nslookup dává uživateli možnost informaci získat. Pro zaslání dotazu je využit primární DNS server nakonfigurovaný v síťovém rozhraní. Výsledek obsahuje nejen informace o překladu, ale také, jaký server informaci zaslal a zda se jedná o autoritativní server pro danou zónu. Příklad dotazu na doménové jméno www.vutbr.cz lze vidět ve výpisu 1.1.

Tab. 1.2: Příklad několika veřejných resolverů

	Primární resolver	Sekundární resolver
Provozovatel	IPv4 IPv6	
Cloudflare	1.1.1.1 2606:4700:4700::1111	1.0.0.1 2606:4700:4700::1001
OpenDNS (Cisco)	208.67.222.222 2620:119:35::35	208.67.220.220 2620:119:53::53
Google	8.8.8.8 2001:4860:4860::8888	8.8.4.4 2001:4860:4860::8844
Comodo	8.26.56.26	8.20.247.20
Quad9	9.9.9.9 2620:fe::fe	149.112.112.112 2620:fe::9
Verisign DNS	64.6.64.6 2620:74:1b::1:1	64.6.65.6 2620:74:1c::2:2
CZ.NIC	193.17.47.1 2001:148f:ffff::1	185.43.135.1 2001:148f:fffe::1

Podobný nástroj jako nslookup je i nástroj Dig. Na rozdíl od Nslookup poskytuje možnost nastavit velké množství parametrů v dotazu. Mezi parametry patří dotaz na specifický typ záznamu, zaslání požadavku na konkrétní resolver DNS a mnoho dalších parametrů. Lze si nástrojem ověřit správné nakonfigurování domény popřípadě získat veškeré záznamy zóny. Příklad dotazu je zobrazen ve výpisu 1.2. Dotaz specifikuje parametry: doména vutbr.cz., záznam typu CAA a je zaslán na veřejný resolver společnosti Google (8.8.8.8). [11]

Výpis 1.2: Příklad využití nástroje Dig

```
; <<>> DiG 9.10.6 <<>> vutbr.cz CAA @8.8.8.8
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 50292
;; flags: qr rd ra ad;
QUERY: 1, ANSWER: 6, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;vutbr.cz.      IN  CAA

;; ANSWER SECTION:
vutbr.cz.   14399 IN  CAA 0 issuewild "digicert.com"
vutbr.cz.   14399 IN  CAA 0 issue "sectigo.com"
vutbr.cz.   14399 IN  CAA 0 issuewild "sectigo.com"
vutbr.cz.   14399 IN  CAA 0 issue "digicert.com"
vutbr.cz.   14399 IN  CAA 0 issue "letsencrypt.org"
vutbr.cz.   14399 IN  CAA 0 issuewild "letsencrypt.org"

;; Query time: 42 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Mon Oct 26 19:34:39 CET 2020
;; MSG SIZE  rcvd: 239
```

1.1.10 DNS Flag day

Flag day je den, kdy se zavádí reakce na problém v komunitě DNS. Událost je podporována velkými společnostmi, jako například: Google, Facebook, Cz.nic, Alibaba a další. Každý rok je Flag day vyhlašován na jiný den. Letos (2020) připadá na 1. října. Veškeré pokyny jsou dostupné na adrese `https://dnsflagday.net/2020`.

Aktuálně se řeší problém, kdy je navrhována maximální velikost zpráv posílaná přes protokol UDP a to z důvodu velikosti zprávy. Pokud je zpráva větší než 1232 bajtů, bývá s velkou pravděpodobností po cestě fragmentována a nedojde celá, což vede k neúspěšnému dotazu. DNS servery mají posílat přes protokol UDP pouze zprávy do této velikosti. Větší zprávy se mají poslat pomocí protokolu TCP.

Ne všechny resolvery tento stav podporují. Flag day problém rozebírá a doporučuje řešení pro administrátory i klienty.[12]

1.2 DNSSEC

DNSSEC reaguje na bezpečnostní nedostatek protokolu DNS, což je možnost změny záznamu za účelem útoku. Aby útočník mohl po cestě záznam změnit, musí pouze vědět Qeury ID. To není problém, protože QID je obyčejné číslo, které pouze identifikuje zprávu v rámci relace dotaz/odpověď.

Rozšíření bylo představeno v RFC 4033 v roce 2005. Aktuálně česká doména patří mezi jedny z neprogresivnější v nasazování DNSSECu ve své zóně. Pokud si chce uživatel zjistit, zda doména je chráněna protokolem DNSSEC, může využít služby CZ.NIC a ověřit na stránce `https://www.dnssec.cz/`. Služba je funkční pro doménu .cz.[13]

1.2.1 Zavedení DNSSECu

Pro ověření zpráv se využívá asymetrické kryptografie. Na začátku správce domény vytvoří dva páry veřejných a soukromých klíčů: PK_1, VK_1 a PK_2, VK_2. Veškeré záznamy zóny podepíše pomocí privátního klíče PK_1. Podpis má následující postup $SIGN(H(zaznam), PK_1)$, kde H je hashovací funkce, SIGN je podpis pro daný kryptografický mechanismus. Výsledek podpisu je zakódován pomocí base64. Zde záznam je myšlen souhrn všech záznamů stejného typu. Pro ověření je nutné zaslat veškeré záznamy stejného typu.

Výstup je vložen do zóny jako nový záznam typu RRSIG. Záznam obsahuje doménové jméno, typ podepsaných záznamů a zakódovaný podpis. Tímto způsobem jsou podepsány veškeré původní záznamy zóny. Poté je VK_1 zakódován base64 a vložen do záznamu typu DNSKEY. Samotný klíč VK_1 je označován jako Zone Signing Key (ZSK). Dále je do záznamu typu DNSSKEY vložen i druhý veřejný klíč VK_2, označován Key Signing Key (KSK)

V dalším kroku jsou záznamy DNSKEY podepsány pomocí PK_2 (privátního klíče KSK). V posledním kroku je $base64(H(VK_2))$ vložen do záznamu typu DS o doménu výše ve stromové struktuře. Pro příklad vutbr.cz je záznam vložen do DNS serveru zpracující doménu .cz. Privátní klíče od obou veřejných klíčů správce domény bezpečně uloží na hardwarovém zařízení bez připojení do sítě. Ačkoliv není mechanismus dvou klíčů povinný, je nejčastěji využívaný.[14]

Vytváří se řetězec důvěry napříč strukturou. Pro ověření musí mít resolver uložen hash veřejného klíče kořenového serveru (KSK). Popřípadě může mít uložen i klíče nižších úrovní. Hashe klíčů se musí manuálně vložit do resolveru

při prvotní konfiguraci [1]. Poslední podepisování kořenové zóny proběhlo v roce 2017.[15]

Bezpečné algoritmy využívané pro podpis a hashování jsou vyjmenované v RFC 8624. Aktuálně se doporučuje využívat hashovací funkci SHA256 a pro podpis algoritmy RSA (Rivest Shamir Adleman) a ECDSA (The Elliptic Curve Digital Signature Algorithm). Pro doménu .cz jsou definovány požadavky na podpis pomocí algoritmu RSA[2]. Dokument od CZ.NIC není již několik let aktualizován. Nezavádí podmínky pro ECDSA. Doporučené velikosti klíčů je vhodné hledat na stránkách NÚKIB (Národní úřad pro kybernetickou a informační bezpečnost). Zde zjistíme vlastnosti klíče pro RSA, minimální velikost klíče je 3072 bitů. Pro ECDSA je doporučená minimální velikost klíče 256 bitů. Algoritmus ECDSA je využit u více než 50 % podepsaných domén v české subdoméně. Výhodou je menší velikost podpisu a 10× rychlejší ověření než u RSA.[17][18]

1.2.2 Ověření DNSSECu

Ověření překladu probíhá na resolveru (průběh popsán na www.vutbr.cz). Celý postup je zobrazen na obrázku 1.7. Resolver má ručně vložen hash klíče (KSK) kořenové zóny v konfiguraci. Při odpovědi se odešle více záznamů, tak aby resolver mohl odpovědi validovat. Prvně ověří, že hash DNSKEY (KSK) odpovídá ručně vloženému záznamu do resolveru. Pokračuje ověření podpisu ZSK pomocí KSK, který je uveden v záznamu DNSKEY. Dále ověří podpis u záznamu RRSIG pro záznamy DS a uloží si hash klíče pro .cz doménu. Validuje podpis pro záznamy NS .cz z důvodu dalšího dotazu.

Následuje dotaz na server spravující doménu .cz. V prvním kroku porovná hash uloženého DS záznamu s DNSKEY (KSK). Dále ověří podpis záznamu RRSIG skupiny DNSKEY pomocí KSK a následně podpis záznamu RRSIG DS. Validní podpis záznamu znamená uložení DS pro doménu vutbr.cz do paměti pro další odpovědi.

Po posledním dotazu se proces opakuje. Porovná se hash DS záznamu s klíčem (KSK). Ověří se ZSK pomocí KSK. Poslední záznam pro ověření již není RRSIG pro záznamy DS, ale pro typ A. Resolver oveří podpis záznamu pomocí ZSK a pošle stanici překlad záznamu typu A pro www.vutbr.cz. Tímto je dokončen řetěz důvěry, záznam odpovídá tomu, co je uložen v DNS serveru. Útočník sice může změnit záznam typu A, který nese pouze „zabezpečení“ v podobě Query ID, ale již nemůže podvrhnout záznam RRSIG, jelikož nemá privátní klíče ZSK ani KSK pro danou doménu.[19]

[1] `https://data.iana.org/root-anchors/root-anchors.xml`

[2] `https://nic.cz/files/nic/doc/Provozni_manual_DNSSEC_201001_final.pdf`

Schéma na obrázku 1.7 zobrazuje také provázanost jednotlivých záznamů napříč doménovou strukturou. Modrá barva vytváří vztah v rámci odkazování pomocí NS záznamů. Fialová barva zobrazuje vztahy klíčů.

Dále však platí, že mezi stanicí, žádající překlad, a resolverem musí být důvěrný kanál, protože zde je záznam přenášen bez DNSSECu. Stanice si nemůže ověřit, že záznam nebyl pozměněn. Řešení nabízí IPsec mezi stanicí a resolverem, nebo využití protokolů DoH (DNS over HTTPS), DoT (DNS over TLS). Obě řešení vytváří důvěrný kanál pomocí protokolu TLS (Transport Layer Security). Poslední možností je spuštění resolveru přímo na stanici a nastavit primární DNS server na 127.0.0.1:53. DNSSEC je poté validován na samotné stanici.

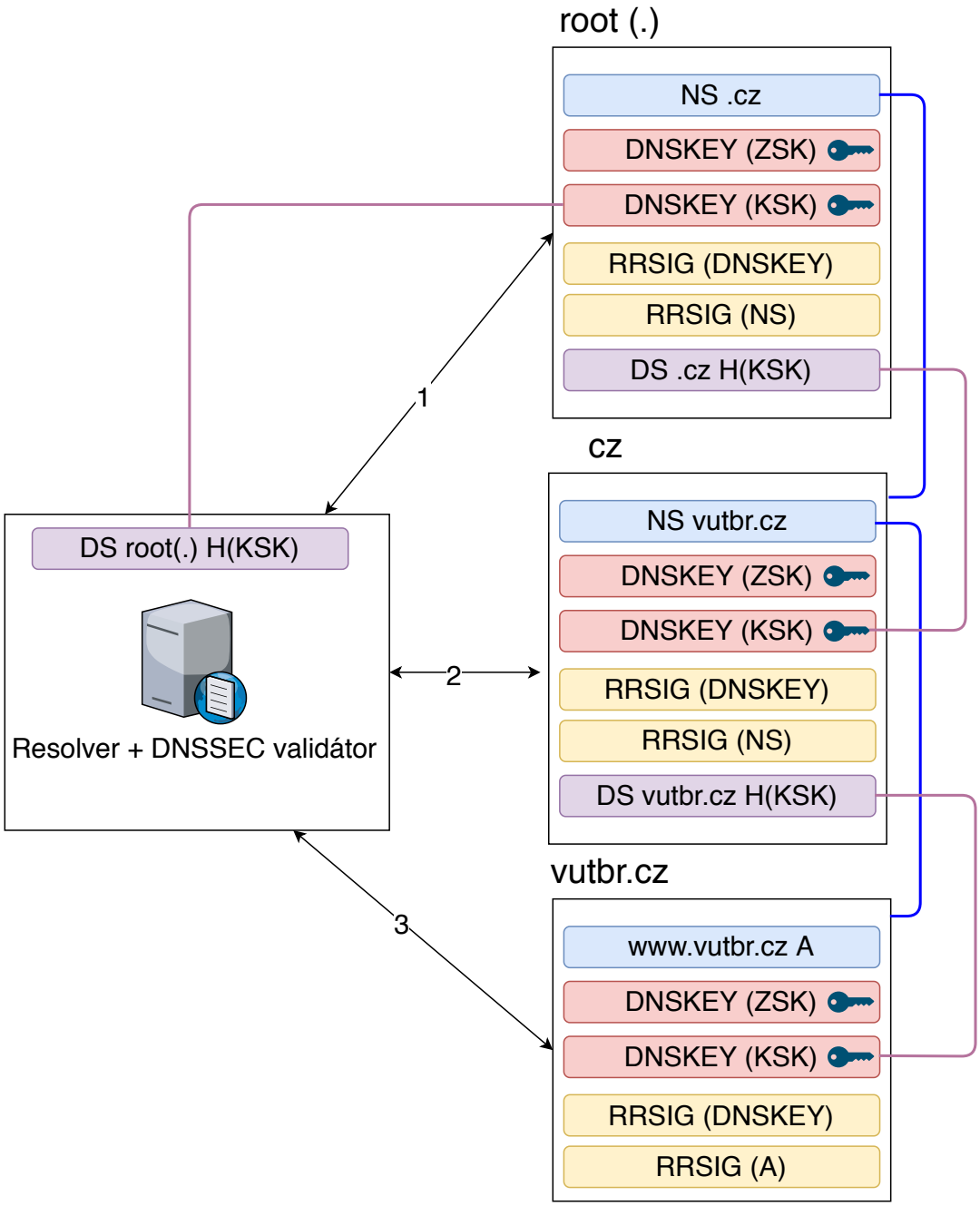
1.2.3 Formát zpráv

Zprávy využívající DNSSEC se výrazně neliší od zpráv původního protokolu DNS. Původní návrh protokolu s rozšířením počítal. V hlavičce byla vytvořena rezerva 3 bitů pro další využití. První bit rezervy zůstal zachován jako rezerva a musí být nastaven na hodnotu 0. Další bit v pořadí již je využit. Nastaven je ve zprávě od serveru. Má označení AD. Server, který ověřil podpisy zpráv a prohlásil je za validní, nastaví tento bit na hodnotu 1. Pokud nejsou ověřeny, je bit nastaven na hodnotu 0. Další bit je využit v rámci dotazu, kde hodnota 1 znamená, že server může poslat i neověřené záznamy. Bitem nastaveným na nulu dává klient najevo, že server může poslat záznamy, které nebyly ověřeny. Poslední atribut je v rámci OPT záznamu v části další záznamy. Zde klient nastavuje, zda si přeje využít ověření samotného DNSSECu. Hodnotou bitu DO nastavuje zaslání RRSIG záznamů pro ověření. Bit 1 dává klient příkaz k poslání i záznamu RRSIG, který se váže k dotazu samotného. Nastavení bitu na 0 znamená, že klient není schopný provést ověření podpisu, a proto si nepřeje záznamy RRSIGu zasílat. Záznam OPT je kopírován do odpovědi.[20]

1.2.4 Záznamy NSEC a NSEC3

Při zaslání odpovědi doménové jméno neexistuje, nelze v DNS ověřit, že odpověď nezaslal útočník. V rámci rozšíření DNSSEC je ověření možné. Proto vznikl nejprve záznam NSEC, který vytváří řetěz mezi existujícími doménovými jmény.

Pro příklad jsou v doméně tři záznamy: a.local, b.local, d.local. Vytvoří se záznamy typu NSEC, kde a.local odkazuje na b.lokal, b.local odkazuje na d.local a d.local odkazuje na a.local. Při dotazu na c.local je vyhodnoceno, že doménové jméno neexistuje a je odeslán zpráva typu NSEC, kde bude uvedeno, že c.local neexistuje, ale následující doménové jméno, které je uloženo v databázi



Obr. 1.7: Schéma záznamů a přenosů při ověření DNSSECu.

je d.local. Takto nastavená doména dává možnost útočníkovi zjistit všechny validní záznamy domény, protože na základě neexistujících doménových jmen získá jména validní. Možností, jak doménová jména skrýt, je využití záznamu NSEC3.

U záznamu NSEC3 není uložen řetez domén v čitelné podobě jako v předchozím případě, ale každé doménové jméno je uloženo pomocí jednocestné hashovací funkce. Takže vzniká řetězec $H(a).local, H(b).local, H(c).local$, kde H je právě hashovací funkce. Pro útočníka, který odešle dotaz na neexistující doménu je odpovězeno, že dotazované jméno neexistuje. Odesláno je následující validní hashované doménové jméno. Z této zprávy nemůže získat existující doménové jméno, kvůli jednocestnosti hashovací funkce.[21]

1.3 Multicast DNS (mDNS)

mDNS vznikl po protokolu DNS. Jeho hlavním cílem je možnost vyhledání doménových jmen v lokální síti. Protokol je původně od společnosti Apple, která jej označovala jako službu Bonjour. Využíván byl pro nacházení služeb v lokální sítí jako například služby AirPrint pro tisk dokumentů na sdílené tiskárně. Protokol je poprvé definován v RFC 6762 a 6763.

Struktura protokolu je stejná jako u DNS. Jednou ze změn je kódování doménových jmen. U DNS je využito kódování Ascii. Převod UTF-8 doménového jména je zajištěn pomocí Punycodu. U mDNS jsou jména přenášena v kódování UTF-8, které zachovává znaky Ascii a přidává další. Je tak zpětně kompatibilní. Jedním z funkce je bezstavová konfigurace síťového rozhraní. Pokud se v síti nenachází DHCP server ani DNS server, měly by zařízení být i tak schopné si nastavit síťové rozhraní. Na základě unikátní fyzické adresy si můžou vygenerovat IPv4 adresu pomocí APIPA (Automatic Private IP Addressing) z rozsahu 169.254.0.0/16. K tomu odpovídá doména .local, ve které může mít stanice jedinečné jméno. Pro zjištění ostatních zařízení využije protokol mDNS, který nepotřebuje nastavení primárního DNS serveru.[22]

1.3.1 Vlastnosti protokolu mDNS

Veškeré dotazy jsou odeslány na multicastovou adresu 224.0.0.251 u IPv4 nebo ff02::fb u IPv6. Ačkoliv síť využívá IPv4 protokol, zprávy mDNS jsou často zasílány pomocí IPv6. Zařízení od společnosti Apple se takto chovají ve výchozím nastavení. Pro multicast lze využít pouze transportní protokol UDP. Port protokolu mDNS je 5353. Struktura zpráv je stejná jako u DNS i s podporou DNSSECu. Pro ukázku, že protokoly DNS a mDNS jsou kompatibilní, je výpis 1.3. Využit je nástroj Dig, který zasílá dotaz na multicastovou adresu s portem 5353.

Samotné doménové jméno v lokální síti může být jiné než v protokolu DNS. Například v lokální síti stanice macMartin.local může mít i globální doménové jméno macMartin.biolek.net.

Výpis 1.3: Příklad kompatibility protokolu mDNS a DNS.

```
$ dig macMartin.local -p 5353 @224.0.0.251

; <<>> DiG 9.10.6 <<>> macMartin.local
;; global options: +cmd
;; Got answer:
;; WARNING: .local is reserved for Multicast DNS
;; You are currently testing what happens when an mDNS
query is leaked to DNS
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 65198
;; flags: qr aa; QUERY: 1, ANSWER: 1,
AUTHORITY: 0, ADDITIONAL: 1

;; QUESTION SECTION:
;macMartin.local.   IN  A

;; ANSWER SECTION:
macMartin.local.  10  IN  A 192.168.0.166

;; ADDITIONAL SECTION:
macMartin.local.  10  IN  AAAA  fe80::1c10:ce52:4a40:9e81

;; Query time: 71 msec
;; SERVER: 192.168.0.166#5353(224.0.0.251)
;; WHEN: Sun Dec 06 20:37:19 CET 2020
;; MSG SIZE  rcvd: 77
```

Hlavní výhodou zasílání multicastových zpráv je šetření šířky pásma, jelikož dotazy i odpovědi si můžou uložit všechny zařízení v síti. Zmenšuje se tak i zatížení DNS serverů, protože dotazy, které mohou být přeloženy v lokální síti, nejsou zaslány mimo interní síť. Jednou z nevýhod je pasivní scan sítě útočníkem. Díky naslouchání na multicastové adrese může získat podrobný popis o síti, a to včetně služeb.[23]

1.4 Link-Local Multicast Name Resolution (LLMNR)

LLMNR je stejně jako mDNS protokol pro překlad doménového jména v lokální síti. Byl vyvinut společností Microsoft. Také zachovává strukturu protokolu DNS, avšak některé položky mají jiný význam. Primárním cílem byla podpora překladu doménového jména nezávisle na protokolu DNS. Protokol je definován pouze informačním dokumentem a nemá tedy podobu platného standardu RFC. Protokol mDNS by měl být preferován jako záložní protokol, avšak v sítích se stanicemi s operačním systémem Windows je často využíván LLMNR.[24]

1.4.1 Vlastnosti protokolu LLMNR

Hlavní rozdíl oproti mDNS je ve stylu komunikace stanic. Dotaz je vždy zaslán přes multicast. Odpověď není poslána multicastem zpátky, jak u mDNS, ale unicastově pouze stanici, která dotaz pokládala. Tato vlastnost vede k mitigaci pasivního skenování sítě, které je u mDNS možné.

Pro zasílání dotazu je využit transportní protokol UDP s portem 5355 na multicastovou adresu IPv4 adresu 224.0.0.252 a IPv6 adresu ff02:0:0:0:0:0:1:3. Dotazy jsou zaslány pomocí obou protokolů zároveň. Překládány jsou pouze domény .local. Lokální jméno se může lišit od globálního. Nejčastěji se využívá jméno stanice s doménou .local. Na rozdíl od DNS můžou zprávy mít větší velikost než 512 bajtů, jelikož v lokální síti je nízká pravděpodobnost fragmentace paketů. Protokol podporuje možnost zasílaní zpráv přes protokol TCP, a to v případě, že jedna ze stanic odpoví s nastaveným bitem pro dělení zprávy.[25]

1.5 Šifrovaný přenos zpráv DNS

Rozšířením DNSSECu se zavedla možnost ověření původu a integrity zpráv, avšak jednotlivé uzly po cestě mohou zprávy přečíst a získávat citlivé informace v podobě využívaných služeb, navštěvovaných webových stránek a další. Pro získání důvěry je třeba zavést šifrování end-to-end, kdy je vytvořen důvěrný kanál mezi klientem a serverem (resolverem). Porovnání otevřeného a šifrovaného formátu je na obrázku 1.8. Nejcitlivější na profilování dat je komunikace mezi stanicí a jejím resolverem, kde lze jednoznačně spojit dotaz s IP adresou, v lokální síti i s fyzickou adresou zařízení. Zprávy od resolveru jsou již mixované i se zprávami od jiných uživatelů. Nelze jednoznačně říci, kdo dotaz zaslal. I v tomto případě ale existuje varianta, že dotazy jsou ukládány pro vytváření uživatelských profilů, a to na samotném resolveru. V případě veřejných resolverů je to i velice pravděpodobné, jelikož poskytují službu zdarma. Pokud za produkt

není placeno, stává se produktem sám klient, respektive jeho data. Ideální variantou je tedy využívat neveřejný resolver s důvěrným kanálem mezi klientem a resolverem s podporou DNSSECu. Mezi protokoly, které podporují šifrování ve výchozím nastavení jsou DNS over HTTPS (DoH), DNS over TLS (DoT) a DNS over Quic (DoQ).[26]

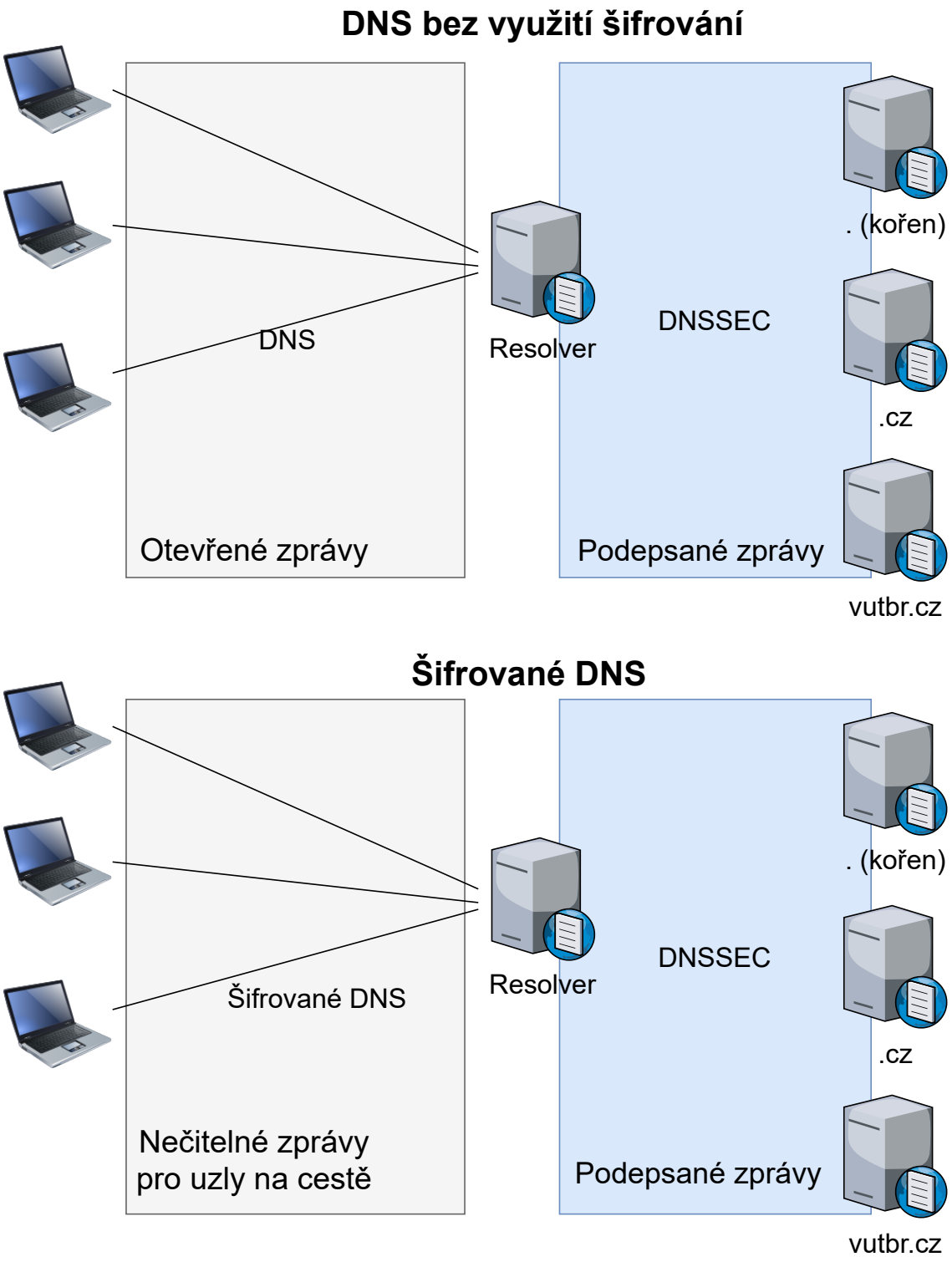
1.5.1 DNS over TLS

Protokol byl definován v RFC 7858 z roku 2016. Mezi klient-serverem se vytváří kanál TCP s TLS, nelze využít zprávy přes protokol UDP. Aplikační protokol je DNS, tak jak byl popsán v prvních kapitolách. Podobně byly vytvořeny protokoly HTTPS, SMTPS a další. Ponechávají aplikační protokol nezměněn a přidávají šifrování na nižších vrstvách komunikace.

Spojení TLS využívá symetrické šifrování, avšak k dohodnutí klíčů je využita asymetrická kryptografie. K ověření slouží certifikáty, které nesou informace o majiteli certifikátu, veřejný klíč a další. Jsou podepsány certifikační autoritou. Ověřeny mohou být obě strany komunikace, nebo pouze jedna z pravidla jen server. Klient chce mít jistotu, že spojení navazuje s validním serverem a ne útočníkem, jelikož útočník nemá privátní klíč od certifikátu, nemůže se za server vydávat.

Má vlastní TCP port 853, protože kdyby naslouchal na stejném portu jako DNS (53), umožňovalo by útok a degradoval spojení na nešifrované. Resolvery, které naslouchají na portu 853 DoT podporují. Mezi takové patří resolvery Cloudflare, Google, CZ.NIC a další.

Vyzkoušet překlad lze pomocí nástroje kdig. Pro správnou funkci je třeba dodat doménové jméno, na které je vydán certifikát pro ověření resolveru. Všichni provozovatelé tento parametr mají vystavený na svém webu. Pro CZ.NIC je to doménové jméno odvr.nic.cz, které musí resolver předložit v TLS certifikátu. Z ukázky 1.4 lze vidět, že před překladem je vytvořeno TLS spojení, které lze vyčíst ve výpisech DEBUG.[27]

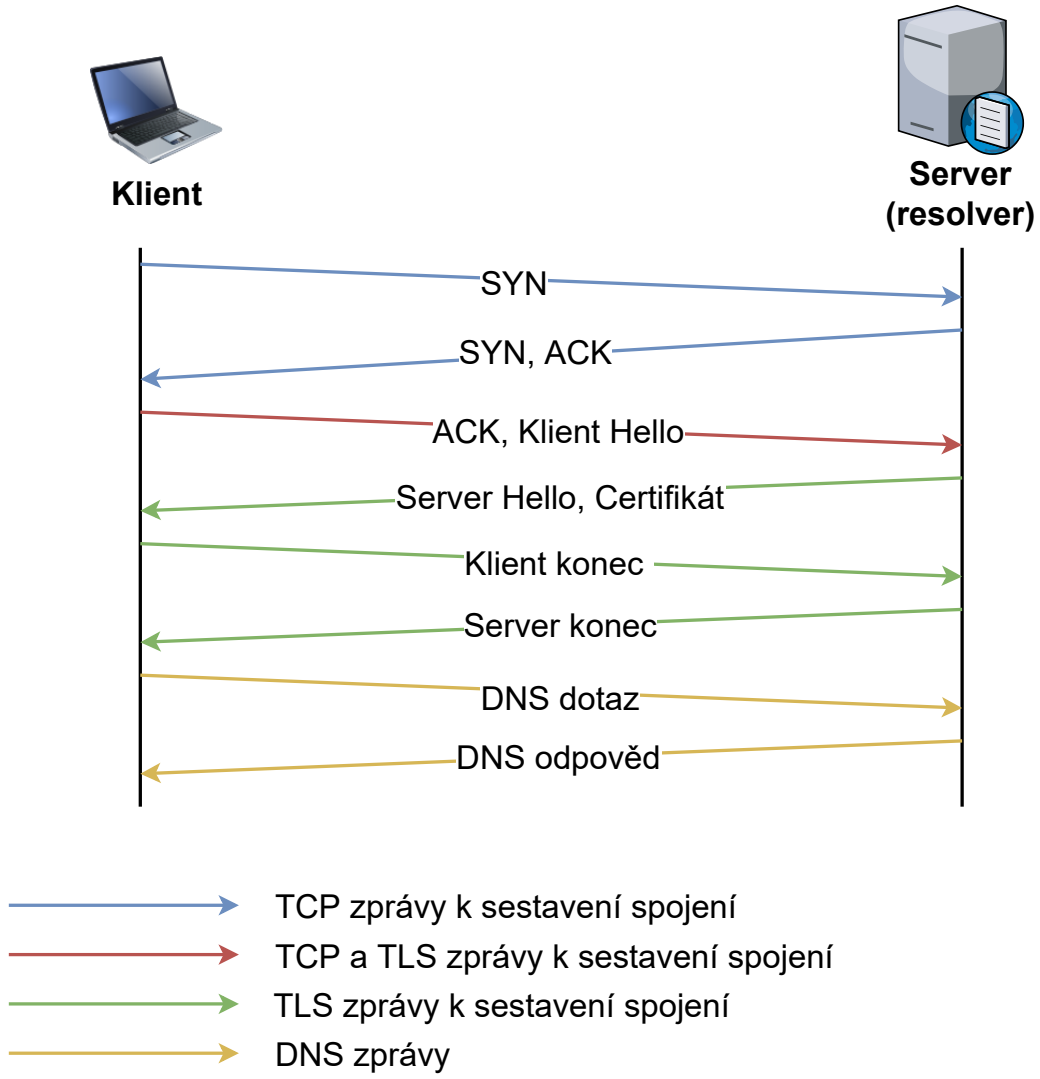


Obr. 1.8: Porovnání otevřeného formátu zpráv se šifrovaným formátem

Výpis 1.4: Příklad překladu doménového jména pomocí protokolu DNS over TLS

```
$ kdig -d @193.17.47.1 +tls-ca +tls-host=odvr.nic.cz
vutbr.cz


;; DEBUG: Querying for owner(vutbr.cz.), class(1),
type(1),server(193.17.47.1),
port(853), protocol(TCP)
;; DEBUG: TLS, imported 164 system certificates
;; DEBUG: TLS, received certificate hierarchy:
;; DEBUG:  #1, CN=odvr.nic.cz
;; DEBUG:
SHA-256 PIN: kFlI7YYB7oUTRXnCXzKfzVpVUaQ+b6HCpf4+DXnbFVc=
;; DEBUG:  #2, C=US,O=Let's Encrypt,CN=R3
;; DEBUG:
SHA-256 PIN: jQJTbIh0grw0/1TkHSumWb+Fs0Ggogr621gT3PvPKG0=
;; DEBUG: TLS, skipping certificate PIN check
;; DEBUG: TLS, The certificate is trusted.
;; TLS session (TLS1.2)-(ECDHE-SECP256R1)
-(RSA-SHA256)-(AES-256-GCM)
;; ->>HEADER<<- opcode: QUERY; status: NOERROR; id: 29939
;; Flags: qr rd ra; QUERY: 1; ANSWER: 1;
AUTHORITY: 0; ADDITIONAL: 1

;; EDNS PSEUDOSECTION:
;; Version: 0; flags: ; UDP size: 1232 B;
ext-rcode: NOERROR
;; PADDING: 411 B

;; QUESTION SECTION:
;; vutbr.cz.                IN   A

;; ANSWER SECTION:
vutbr.cz.             300 IN   A 147.229.2.90

;; Received 468 B
;; Time 2021-01-17 16:00:51 CET
;; From 193.17.47.1@853(TCP) in 92.3 ms
```

Hlavní nevýhodou protokolu je potřeba vytvářet TCP spojení, které pro své ustanovení vyžaduje 3 zprávy, nad tímto spojením se vytváří TLS spojení a poté následují DNS zprávy. Pro TLS 1.2 je minimální počet zpráv pro sestavení spojení 6. Samotný průběh je zobrazen na obrázku 1.9. Při využití nejnovější verze protokolu TLS 1.3 lze počet zpráv snížit na 4 zprávy, ale oproti nešifrované variantě narůstá potřebná doba pro překlad. Z příkladu 1.4 lze vyčíst, že byl využit TLS ve verzi 1.2 a celková doba překladu byla 92.3 ms. Stejně zaslaný dotaz otevřeným formátem DNS byl přeložen za 25 ms. Nárůst je skoro 4–násobný. Zjištěné hodnoty podporuje také článek od poskytovatele Apnic, kde žádná odpověď se nedostala pod 10 ms. U protokolu DNS za stejných podmínek šlo získat překlad i pod touto časovou hranicí.[29]



Obr. 1.9: Průběh zpráv protokolu DNS over TLS

1.5.2 DNS over HTTPS

Druhým šifrovaným protokolem je DNS over HTTPS (definován v RFC 8484 z roku 2018), kde již není DNS protokol využit na aplikační vrstvě, ale je nahrazen aplikačním protokolem HTTPS, který také vytváří spojení TCP s TLS. Hlavní rozdíl je ve formátu zpráv. Existují dvě varianty. První formát dává DNS protokol do těla zprávy HTTPS. Druhý využívá rest api formátu, kdy dotaz je součástí Uniform Resource Locator (URL a odpověď je ve formátu JavaScript Object Notation (JSON) v tělu odpovědi.

Výhodou také je, že webové servery jsou vázané na doménové jméno a s nimi i TLS certifikát. Není tedy nutné zadávat doménové jméno ručně pro ověření certifikátu jako tomu bylo u DoT. Oproti DoT má velkou výhodu protokol HTTPS, využívá port 443, který nebývá zakazován na firewallech. U protokolu DoT je časté blokování této služby nějakým firewallem po cestě. Odpovědi jsou přenášeny v části těla HTTPS protokolu v JSON formátu, který je v hlavičce protokolu označen jako kódování dns-json. Protokol lze využívat i v kombinaci s obyčejným DNS, kdy DoH lze zapnout pouze pro prohlížeč. Prohlížeče Chrome, Brave, Firefox atd. tuto funkci umožňují. Implementace je rychlá a nenáročná.[28]

Pokud budu chtít záznam A doménového jména www.vutbr.cz, můžu využit službu CloudFlare. Samotný překlad je zobrazen ve výpisu 1.5. Struktura odpovědi zůstala stejná, jen má podobu asociativního pole, které je pro formát JSON typické.[30]

Stejně jako u DoT je také u DoH zvýšen počet režijních zpráv pro sestavení spojení, které se projevují větší prodlevou k úspěšnému překladu doménového jména. Podle doporučení NSA (National Security Agency) by společnosti měli preferovat překlad doménových jmen pomocí právě tohoto protokolu, a to na svém resolveru. Hlavní motivací by mělo být vyhnout se útoku muže uprostřed a standardizované logování jak webového serveru, tak i resolveru. Dále doporučují při využívání veřejných resolverů metodu nazvanou Oblivious DNS over HTTPS, která přidává možnost anonymizovat dotazy i při užívaní veřejných resolverů. Technika spočívá ve využití proxy serveru, který tak zná původce zprávy, nezná obsah (kvůli šifrování TLS) a resolver zase po rozšifrování zná dotaz, ale nemůže identifikovat původce zprávy. Takto lze předejít vytváření uživatelských profilů podle vyhledávaných dotazů.[31]

Výpis 1.5: Příklad překladu doménového jména pomocí protokolu DNS over HTTPS

```
$ curl -sH 'accept:application/dns-json'\
'https://cloudflare-dns.com/dns-query?\
name=www.vutbr.cz&type=A'
{
    "Status": 0,
    "TC": false,
    "RD": true,
    "RA": true,
    "AD": true,
    "CD": false,
    "Question": [
        {
            "name": "www.vutbr.cz",
            "type": 1
        }
    ],
    "Answer": [
        {
            "name": "www.vutbr.cz",
            "type": 1,
            "TTL": 117,
            "data": "147.229.2.90"
        }
    ]
}
```

1.5.3 DNS over Quic

Poslední možností ze šifrovaného DNS je DoQ neboli DNS over Quic. Jedná se o nejpokročilejší protokol, ale doposud nebyla dokončena specifikace protokolu. Samotný mechanismus sází na transportní protokol Quic, který byl vytvořen společností Google pro odlehčení zátěže webovým serverům, jelikož stavěl na protokolu UDP a zaváděl bezpečnost v podobě TLS 1.3. Hlavní výhodou je nízký počet zpráv pro navázání šifrovaného spojení, a to maximálně 2. V případě, že klient již se serverem komunikoval dříve, otevírá se mu možnost takzvaného 0-RTT (Zero Round Trip Time). Klient posílá již šifrovanou zprávu bez dřívějšího ustálení spojení, jelikož dokáže odvodit klíče pro novou relaci

z relace předešlé. Dotaz a odpověď zabírají celkem dvě zprávy jako u klasického DNS protokolu, ale s výrazným benefitem v podobě důvěrnosti a integritě dat.[32]

Standardizace protokolu začala v roce 2017. Aktuálně ještě není dokončena. Pro možnosti otestování existuje první veřejný resolver, který DoQ podporuje v experimentálním módu – AdGuard. Díky nástroji Dnslookup je možné překlad protokolem provést. Příklad je zobrazen ve výpisu 1.6. [33]

Dnes je doporučováno využívat DoH. Po dokončení standardu DNS over Quic a implementaci na resolverech, lze předpokládat, že DoQ bude primárně využívaný protokol, protože spojuje rychlost klasického DNS s bezpečností TLS verze 1.3.

Výpis 1.6: Příklad překladu doménového jména pomocí experimentálního protokolu DoQ

```
$ ./dnslookup vutbr.cz quic://dns.adguard.com

dnslookup result:
;; opcode: QUERY, status: NOERROR, id: 57830
;; flags: qr rd ra; QUERY: 1, ANSWER: 1,
AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;;vutbr.cz. IN   A

;; ANSWER SECTION:
vutbr.cz. 300 IN  A 147.229.2.90
```

2 DNS a blockchain

Dnes jsou DNS a blockchain dvě různé technologie, které nejsou navzájem propojeny. Obě technologie staví na decentralizované síti. DNS využívá více úrovňový systém. Blockchainové kryptoměny zase jedné vrstvy, kdy uzly jsou spojeny pomocí TCP spojení. Všechny plné uzly sítě uchovávají celý blockchain a mají tak veškeré informace o síti.

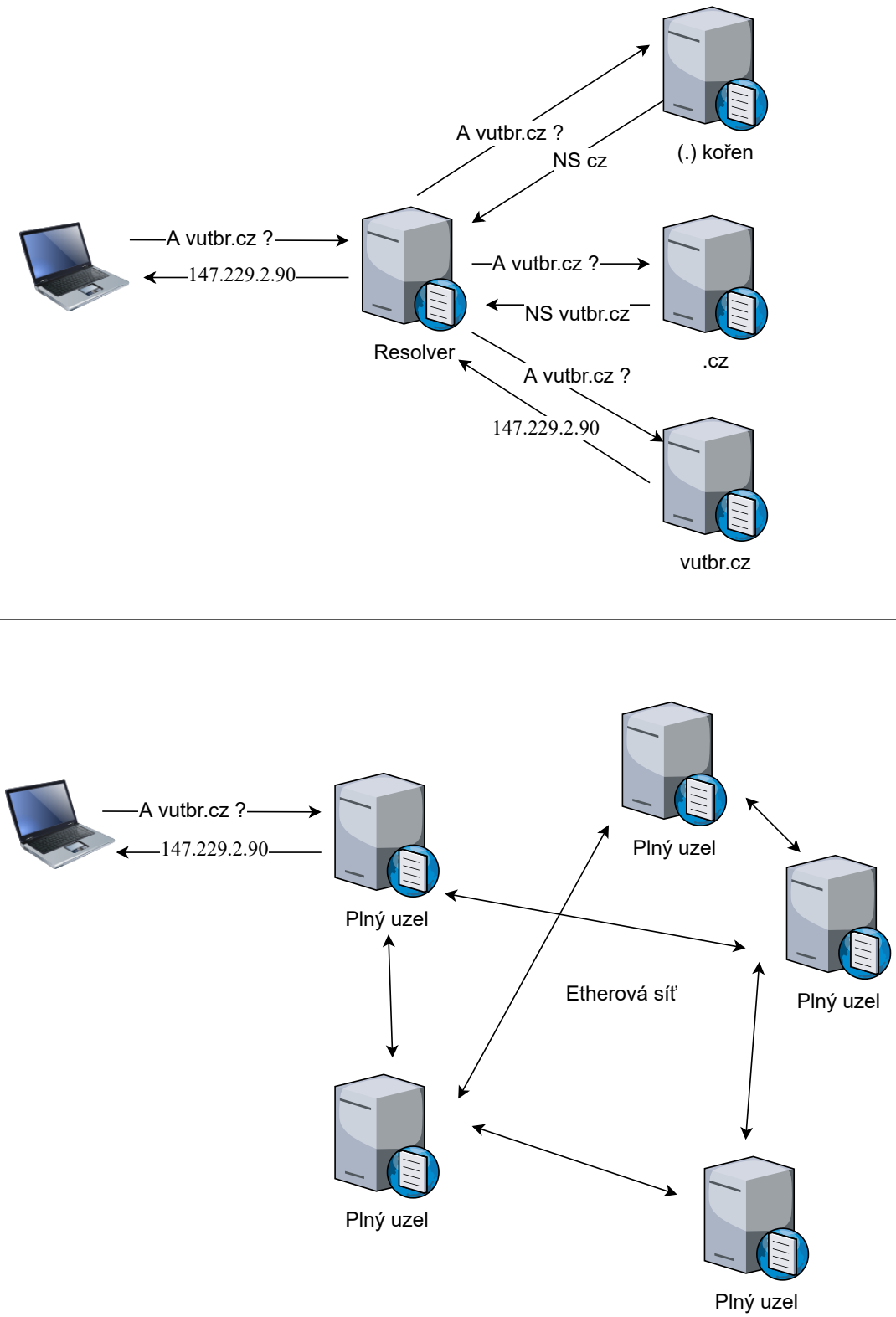
Ačkoliv je autor Bitcoin maximalista, musí uznat možné skloubení funkce DNS a Etherea, které zatím jedno z mála má plně nasazené smart kontrakty ve své síti. Postupně se smart kontrakty dostanou také na Cardano, Polkadot a Solanu síť, ale v současnosti jsou plně dostupné pouze na této síti. Smart kontrakt je bajtový kód zapsaný přímo do blockchainu pomocí transakce. Volat metody a zapisovat data lze pomocí dalších transakcí. Kód vykonávají přímo uzly. Díky transakčnímu poplatku mají ekonomický zájem kód vykonat správně a uzavřít i díky tomu aktuální blok. Systém DNS zapsaný do smart kontraktu přináší výhody i nevýhody popsané v následujících kapitolách. Rozdíly lze pozorovat ze obrázku 2.1.

2.1 Výhody DNS v blockchainu

Blockchain Etherea tvoří uzly postavené všechny na stejnou úroveň, kdy není žádný nadřazen jinému. Lze tedy předpokládat, že dotaz na jakékoliv doménové jméno, které je uloženo ve smart kontraktu bude moct být vyčteno z jakéhokoliv uzlu. Odpadá tak nutnost hierarchického dotazování, což vede k menšímu počtu dotazů a rychlejšímu překladu.

Doménové jméno se bude moct registrovat na Etherovou adresu. Veškeré změny provádět pouze na základě transakcí z této adresy. Jelikož transakce jsou podepsány pomocí privátního klíče, ze kterého je odvozen veřejný klíč a z něj adresa, lze tak vypustit nutnost DNSSECu. Záznamy budou moct být měněny na základě validní transakce, která musí vždy být podepsaná privátním klíčem.

Mizí registrátoři a kořenové servery. Registrace domény by mohla probíhat pomocí transakce, kde smart kontrakt by si uložil veřejnou adresu a doménové jméno. Takto by pak mohl vyhledávat a ověřovat, že změny provádí pouze držitel privátního klíče k dané adrese. Pramení z toho i jedna nevýhoda, nutnost mít časovač, který po uplynutí data uvolní doménové jméno z držby nebo minimálně jednou za uvedený čas volat metodu smart kontraktu, kterou se časovač obnoví a prodlouží se doba registrace domény.



Obr. 2.1: Rozdíly v dotazování. Horní obrázek reprezentuje DNS, dolní pak dotazování na smart kontrakt.

2.2 Nevýhody DNS v blockchainu

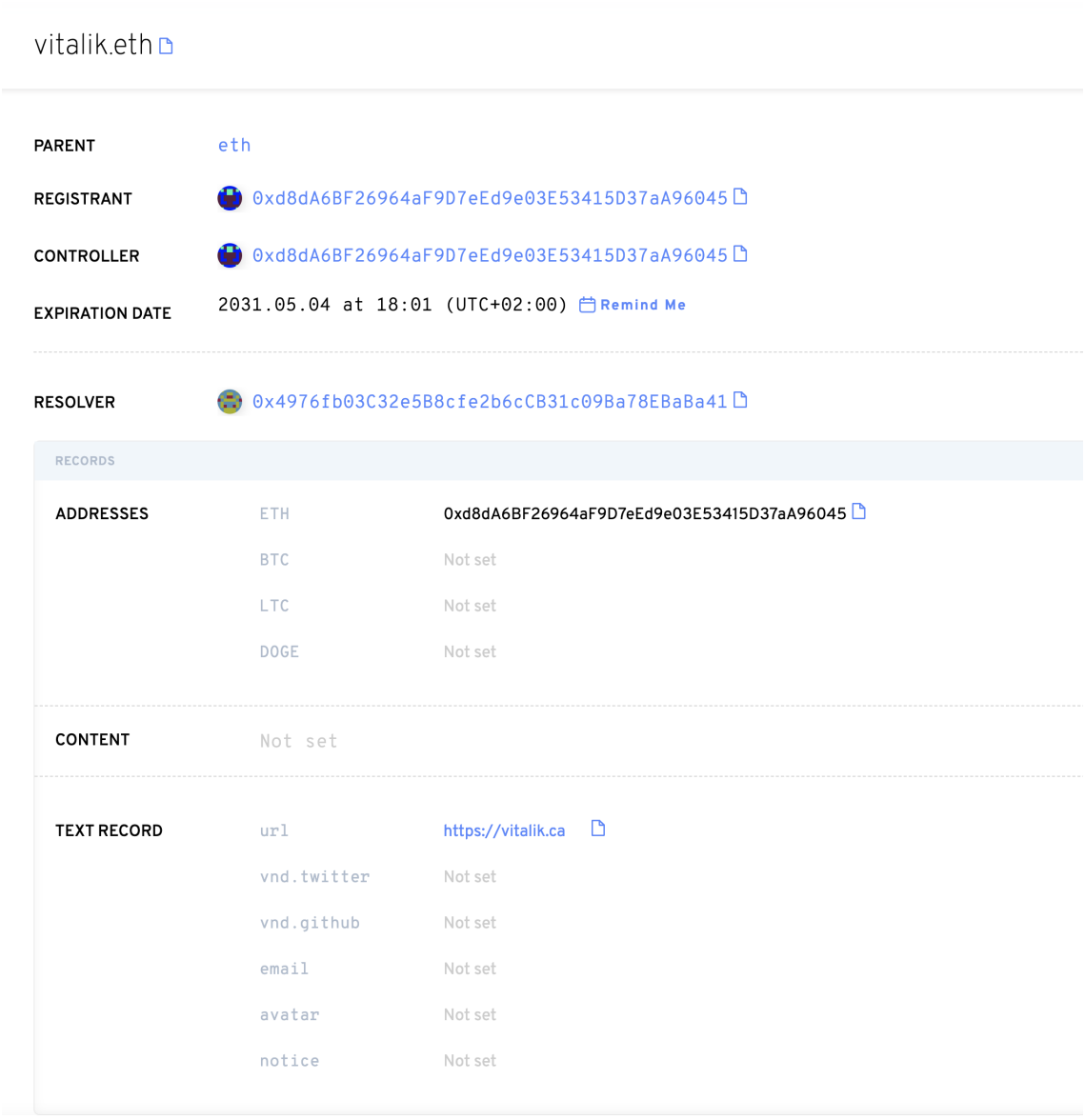
Navázání doménového jména na privátní klíč přináší nutnost mít zálohu privátního klíče v reálném světě, nejlépe ve formě seedu, což je slovníková reprezentace. Pro významné domény by bylo vhodné privátní klíč uložit pomocí Shamirova algoritmu, kdy je tajný klíč rozložen mezi více účastníků. Problémy spojeny s privátním klíčem v původním DNS nejsou, ale pokud doména je zabezpečena pomocí DNSSECu jedná se o podobný případ a administrátor je nucen privátní klíč uchovávat.

Možnou nevýhodu je nutnost transakčního poplatku při změně záznamu. Poplatek u dnešního DNS je hrazen pouze při registraci a prodloužení domény. Nutnost transakčního poplatku je ale benefit, který zajišťuje, že těžaři a plné uzly mají ekonomický důvod službu provozovat.

2.3 Dnešní využití DNS na Ethereu

Dnes se využívá mechanismus smart kontraktu k ukládání krypto adres, protože stejně jako IP adresy jsou i krypto adresy pro člověka těžko zapamatovatelné. Na Ethereu funguje služba `https://app.ens.domains`, kde si pod doménové jméno lze uložit do smart kontraktu veřejné informace. Příklad domény vitalik.eth lze vidět na obrázku 2.2.

Otázka zůstává zda takovéto řešení přináší tak rozsáhlé výhody, aby ekosystém DNS jednoho dne přešel celý na blockchain. Dnes aktuálně převažují výhody aktuálního protokolu, je vyzkoušen a kolem jsou postavené všechny služby. Přechod dnes nedává smysl, ale vývoj na blockchainových projektech postupuje dopředu a jednoho dne se může stát skutečností, že DNS bude právě na blockchainu.



Obr. 2.2: Příklad uložení informací ve smart kontraktu pro doménu vitalik.eth

3 Praktická část

3.1 Srovnání resolverů

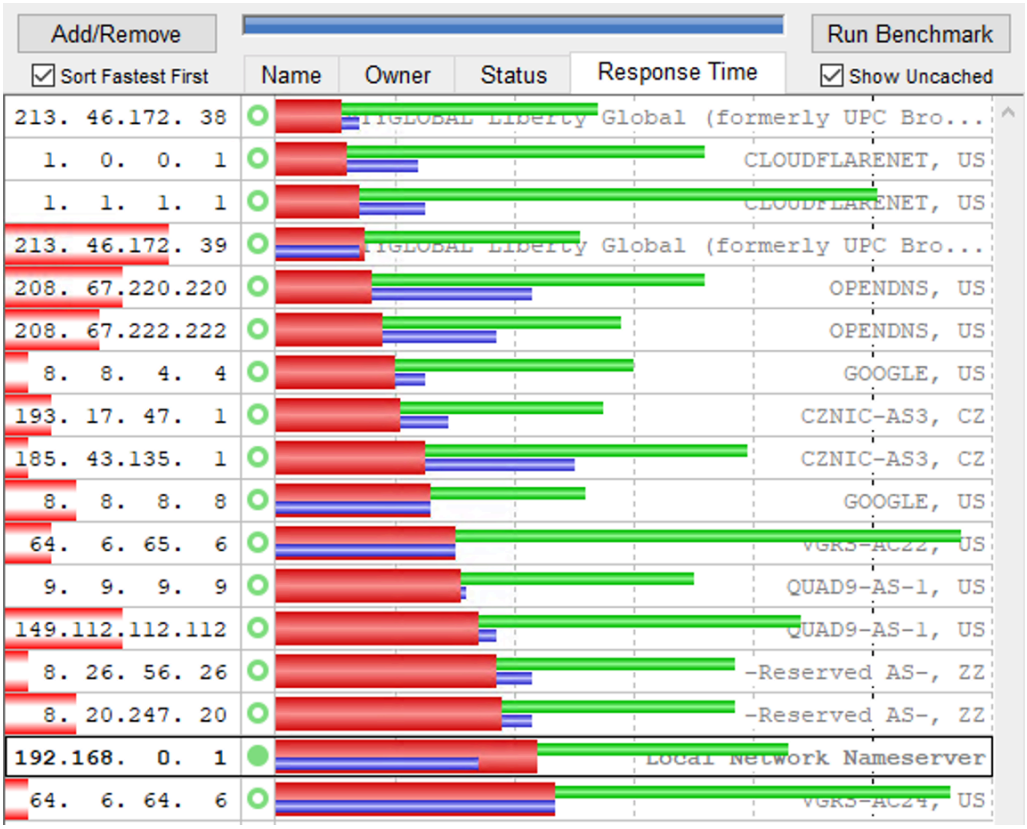
Pro srovnání resolverů je využito metriky celkového času odezvy. Pokud chceme snížit zpoždění při procházení například webových služeb, je dobré mít nastaven jako primární DNS server ten nejrychlejší pro aktuální lokaci. Pro zjištění odezvy různých resolverů existuje program DNS Bechmark, který resolvery může srovnat právě z aktuální lokace. [34]

3.1.1 Srovnání resolverů z domácí sítě

V rámci srovnání resolverů bylo použito veřejných resolverů z tabulky 1.2 k nim byly přidány další tři. Resolver výchozí brány (192.168.0.1), primární a sekundární DNS resolver poskytovatele (213.46.172.38, 213.46.172.39). Poskytovatel nepodporuje IPv6, proto srovnání je pouze pro protokol IPv4.

Metrika je čas odpovědi na dotaz v milisekundách. Červená část grafu znázorňuje dobu odpovědi na dotazy, které byly cachovány v resolveru a nemusel tedy provádět další vyhledávání. Zde jako u ostatních parametrů platí: čím menší hodnota, tím lepší výsledek. Modrá část grafu představuje dobu na odpověď při dotazu na DOT COM doménu. Jedná se o TLD .com. V praxi nejvíce provozu míří právě na tyto doménové jména. Poslední část má zelenou barvu. Jedná se o odpovědi na dotazy obecného charakteru mířící mimo DOT COM doménu, tedy i na část .cz a další. V grafech se zobrazuje průměrná doba odpovědi. Měřítko je nastaveno na 20 ms za dílek. Nejlépe hodnocený je pak ten, který součet těchto dílčích částí má nejmenší. Červený graf v místě *Internet Protocol* (IP) adresy znázorňuje relativní počet odpovědí, na které server neodpověděl.

Výsledky srovnání lze vidět na obrázku 3.1. Jak je z obrázku patrné, resolvery jsou již srovnány od nejlepšího po nejhorší. Nejlépe vyšel resolver od poskytovatele (Vodafone Czech Republic a.s. - dříve UPC). Zajímavé je srovnání sekundárního resolveru od poskytovatele, který vyšel výsledkově třetí, avšak má velké množství nezodpovězených dotazů. Pro domácí síť je nejlepší kombinace tedy v primárním resolveru poskytovatele a sekundárního resolveru od CloudFlare. Resolver na výchozí brány vyšel jako druhý nejhorší, co se týče doby odpovědí. V případě, že by stačila pouze rychlost odpovědí, tento resolver nedává smysl využívat. Výhoda spočívá v ověření DNSSECu, kdy ověření záznamu je na rozhraní domácí sítě. Kombinace útoku MITM (Man In the Middle) a DNS Spoofing je možná až rámci lokální sítě, kde lze zapnout protokol IPsec (IP security) a vyhnout se tomuto typu útoku. Otevřený resolver dává možnost útoku i mimo lokální síť.



Obr. 3.1: Výsledky srovnání veřejných resolverů na základě rychlosti odpovědí z domácí sítě

3.1.2 Srovnání resolverů ze sítě VUT

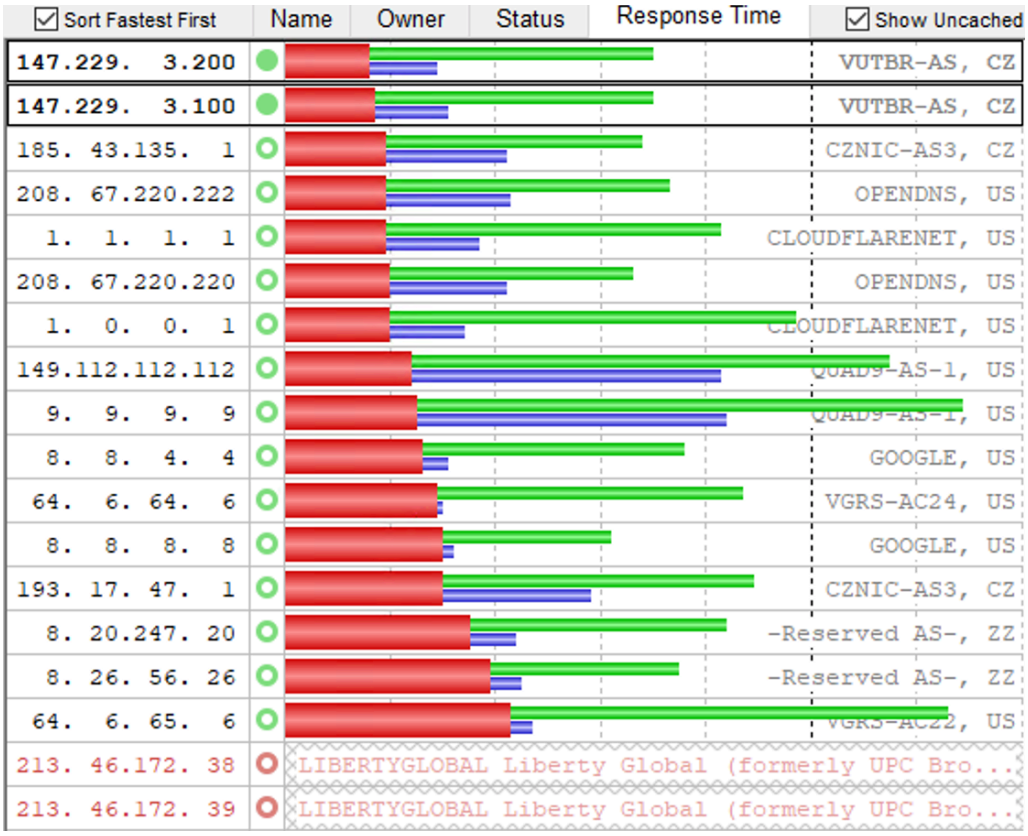
Pro srovnání resolverů z druhé lokace byla použita síť VUT. Pro připojení bylo vytvořeno spojení VPN (Virtual Private Network) L2TP/IPSEC - před-sdílený klíč. Z seznamu resolverů byla smazána výchozí brána a přidány dva resolvery ze sítě VUT. Výsledky lze vidět na obrázku 3.2.

Z výsledků vyplývá, že resolvery VUT jsou nejlepší volbou pro nastavení primárního, sekundárního resolveru. Využití nejlepšího resolveru z domácí sítě by aktuálně nebylo vhodné, protože resolvery mého poskytovatele jsou nastaveny pouze pro síť poskytovatele. Dotaz nelze dopravit na tento resolver a funkci primárního resolveru by nemohl plnit.

3.2 Implementace programu

3.2.1 Cíle a výsledný formát programu

Program má za cíl dát uživateli nástroj k parametrizování DNS dotazů a zobrazení výsledků překladu. Mezi parametry, které by měl mít možnost uživatel ovlivnit,



Obr. 3.2: Výsledky srovnání veřejných resolverů na základě rychlosti odpovědí ze sítě VUT

patří druh protokolu na aplikační vrstvě, transportní a síťové vrstvě z TCP/IP modelu. Program musí obsahovat grafické rozhraní pro zobrazení výsledků překladu a pro nastavení parametrů dotazu. Za cíl si také klade zobrazit co nejvíce informací spojené s překladem: aplikační data, velikost zpráv, čas transakce a popřípadě další. Program musí být vytvořen pro primární platformu Microsoft Windows, aby mohl být využit pro výukové předměty na fakultě.

Výstupem je program DNS klient.exe, který musí být ve stejné složce jako knihovna JavaFX pro bezproblémové spuštění programu. Ke spuštění programu je potřeba nainstalovaná Java verze 11 a výše. Java 11 je již nainstalována na stanicích fakulty a má dlouholetou podporu. Oproti Java 8 není již knihovna JavaFX součástí výchozích knihoven Javy, a proto je knihovna ve stejné složce se spustitelným .exe souborem.

3.2.2 Programovací jazyk a knihovny

Pro implementaci byl zvolen jazyk Java. K výběru jazyka přispěla výuka na fakultě a možnost spouštění kódu na různých platformách. Pro kompilaci

a spouštění bylo využito Java Runtime Enviroment ve verzi 11. Pro grafické rozhraní byla vybrána knihovna JavaFX verze 15. JavaFX je nejmladší knihovna vyvinuta pro práci s grafickým rozhraním. Na rozdíl od knihoven Swing a AWT má aktuální podporu.

Projekt je typu Maven z důvodu velice rychlého importování knihoven a to na základě vkládaní xml tagů do souboru pom.xml. Vývoj probíhal na operačním systému Windows 10 Home edice, z důvodu kontinuálního testování aplikace na primární platformu. Pro vývoj byl využit Eclipse ve verzi 2020-09. Grafické rozhraní bylo vytvořeno v Scene Builder verze 11 od společnosti Gluon. Scene builder je určení k vytváření uživatelských oken bez nutnosti znalosti fxml notace, která se pojí s grafickou knihovnou JavaFX.

Software je verzován pomocí nástroje Git verze 2.29.2 a ukládán na vzdálený repozitář na Githubu. Samotný repozitář je otevřený a lze průběžně sledovat vývoj. Popřípadě hlásit možné chyby softwaru a to i ze samotného softwaru, který nabízí tlačítko pro hlášení chyby. Po stisku tlačítka dojde k otevření prohlížeče na stránce projektu v záložce nahlašování chyb. Stránky využitých knihoven a softwaru jsou uvedeny v tabulce 3.1.

Pro ukládání nastavení do souboru byl využit formát Json, který lze spravovat pomocí knihovny Json-simple verze 1.1.1. Zpracování IP adres a převod na formát záznamu PTR byla využita knihovna IPAdrress ve verzi 5.3. Pro výpis Json objektu do řetězce je využita knihovna Google Gson ve verzi 2.8.6. zajišťující formátování řetězce. Pro DNS over HTTPS je využita knihovna Apache HTTPClient ve verzi 4.5.13. Projekt je typu Maven, kdy knihovny lze importovat pomocí pom.xml souboru. Knihovny stáhne na pozadí samotné vývojové prostředí.

Tab. 3.1: Seznam použitých knihoven a softwaru

Jméno softwaru	Odkaz
Java JRE 14	`https://jdk.java.net/14/`
JavaFX 15	`https://openjfx.io/`
Eclipse 2020-09	`https://www.eclipse.org/`
Scene Builder 11	`https://gluonhq.com/products/scene-builder/`
Git 2.29.2	`https://git-scm.com/`
Repozitár projektu	`https://github.com/mbio16/clientDNS`
Json-simple 1.1.1	`https://code.google.com/archive/p/json-simple/`
IPAddress 5.3.3	`https://seancfoley.github.io/IPAddress/`
Google Gson 2.8.6	`https://github.com/google/gson`
Apache HttpClient 4.5.13	`https://hc.apache.org/httpcomponents-client-5.0.x/`

3.2.3 Struktura programu

Program je vytvořen pomocí návrhového vzoru Model-view-controler, jednotlivé třídy jsou členěny do 10 balíčků (application, models, ui, records, exceptions, test, enums, fxml, images, locale). Balíček application obsahuje pouze hlavní metodu Main a soubor s kaskádovými styly pro aplikaci. Třídou Main se spouští samotná aplikace.

V balíčku models jsou třídy starající se o abstrakci práce s daty a síťovými prostředky. Jde říci, že všechny třídy typu model se nachází v tomto balíčku. Samostatný balíček mají DNS záznamy, kdy každý typ záznamu má jinou datovou strukturu popsanou v doporučení RFC. Datové typy záznamů jsou uloženy v balíčku records, aby jejich vývoj byl přehlednější a nebyl součástí obecných modelů.

V balíčku ui lze najít kontroléry pro jednotlivé okna, která jsou definována v souborech .fxml v balíčku fxml. Jméno okna koresponduje s jménem kontroléru napříč těmito balíčky. Uživatelsky definované výjimky jsou reprezentovány třídami v balíčku exceptions, kde veškeré třídy jsou potomky třídy Exception. Číselníky či jiné seznamy se nachází v balíčku enums. Jedná se zejména o třídy, kde jsou definované číselníky pro příznaky například do hlavičky zprávy. Obrázky lze najít v balíčku images, kde se nachází pouze obrázky .png pro zobrazení obrazu bez blokových artefaktů (které by hrozily při použití formátu .jpg). Jazykové lokalizace jsou umístěny v balíčku locale v souborech typu .properties. Aktuálně aplikace podporuje dvě jazykové sady: češtinu a angličtinu.

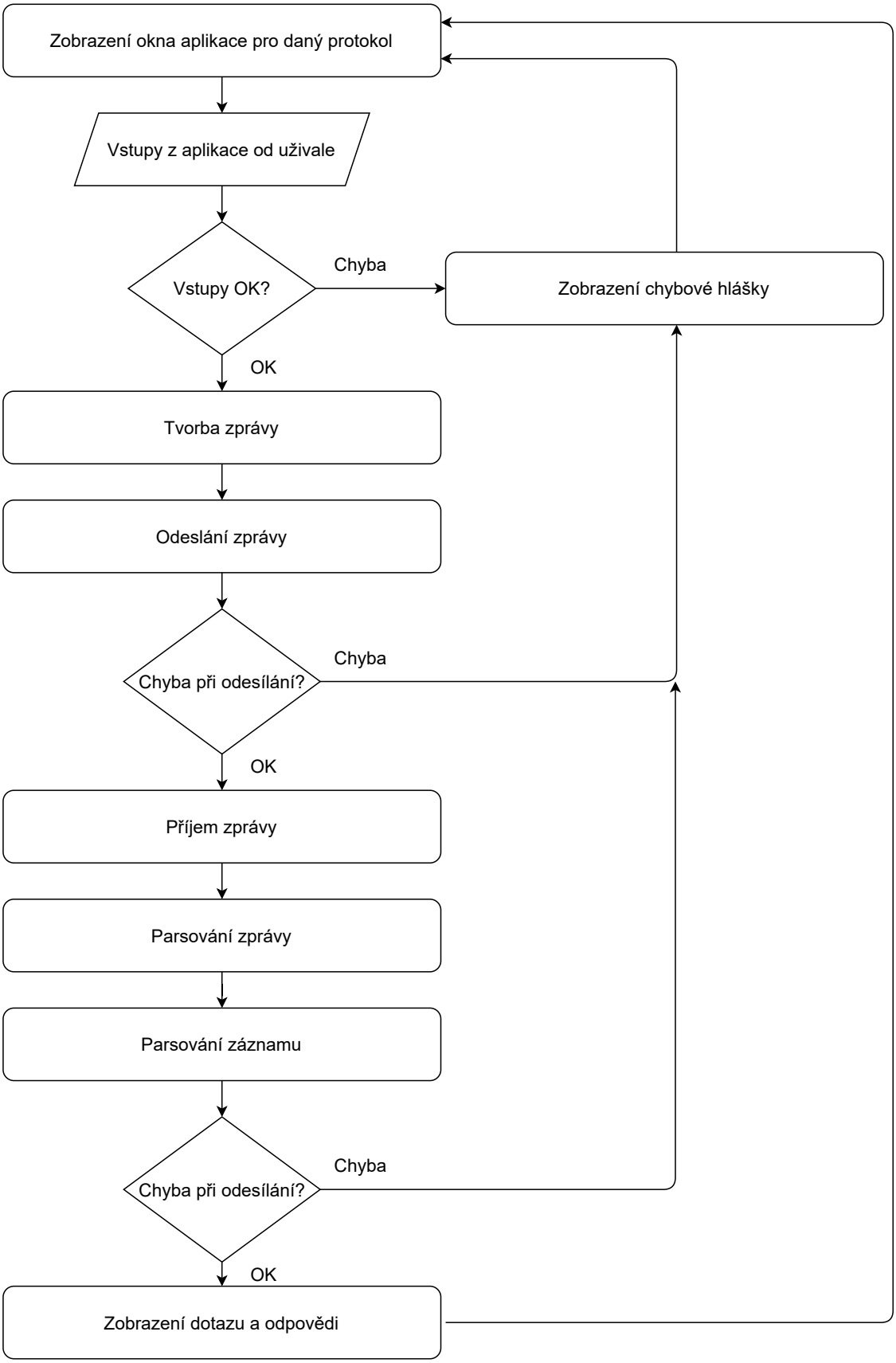
Balíček test obsahuje testy typu JUnit a třídy TestMain, která byla využita při testování odesílání a přijímání zpráv, jelikož pro testování změn programu nebylo praktické spouštět samotné grafické rozhraní.

Třídy obsahují statickou konstantu typu Logger, která uchovává logovací nástroj do konzole. Výpisy slouží pouze k vývoji, protože samotný program nebude mít konzoli připojenou. Názvy tříd, proměnných a metod jsou psány v anglickém jazyce, z důvodu editovatelnosti kódu nezávislou osobou. Kompletní znázornění stromové struktury programu lze nalézt v příloze B. Trojice tříd pro ukázku zdrojového kódu je vložena v příloze C.

Znázornění průběhu překladu doménového jména je zobrazeno na obrázku3.3.

3.2.4 Třídy Settings a Language

Třída Settings slouží k ukládání historie do souboru. Při ukončení aplikace jsou data uložena do souboru ve složce Klient DNS do souboru settings.json. Adresář je vytvořen při prvním spuštění aplikace ve složce dokumenty. Při startu aplikace jsou data vyčtena ze souboru a předána ostatním objektům.



Obr. 3.3: Vývojové schéma při překladu doménového jména programem.

Výpis 3.1: Příprava a uložení souboru settings.json

```
1  private void setupJsonFile() throws IOException {
2      Map<String, ArrayList<String>> jsonMap =
3      new HashMap<String, ArrayList<String>>();
4      jsonMap.put(DNS_SERVERS, dnsServers);
5      jsonMap.put(DOMAIN_NAMES_DNS, domainNamesDNS);
6      jsonMap.put(DOMAIN_NAMES_mDNS, domainNamesMDNS);
7      Map<String,String> jsonMap2 =
8      new HashMap<String,String>();
9      if(netInterface == null) {
10       netInterface = NetworkInterface.getByInetAddress(
11       InetAddress.getLocalHost());
12     }
13     jsonMap2.put(LAST_USED_INTERFACE,netInterface.getName());
14     JSONObject json = new JSONObject(jsonMap);
15     json.putAll(jsonMap2);
16     try (FileWriter fw = new FileWriter(
17         file,
18         StandardCharsets.UTF_8);
19         BufferedWriter writer = new BufferedWriter(fw)) {
20       writer.append(json.toString());
21     }
22     jsonMap.clear();
23 }
```

Druhým objektem vytvářeným ihned po startu je objekt typu Language. Objekt slouží ke změně jazyka. Pro ostatní objekty je získání překladu transparentní. Jazykové lokalizace lze nalézt v balíčku locale v souborech Lang_cz.properties a Lang_en.properties. Pro zaručení správného načtení atributu z lokalizačních souborů, jsou atributy pojmenované stejně jako ID JavaFX komponenty. V případě změny jazyka je pak velice efektivní vytvořit pole stejných komponent a při iterováním polem měnit popisky pro zvolený jazyk. Příklad využití techniky je zobrazen ve výpisu kódu 3.2. Ukázka obsahuje pouze nastavení pro jeden druh komponent. Stejný postup je využit u dalších JavaFX komponent.

Při přidání nové komponenty lze rychle přidat její lokalizaci. Stačí přidat komponentu do jednoho z polí a doplnit samotný překlad v obou lokalizačních souborech. Metoda má nastavenou viditelnost public, protože je volána při načítání okna před zobrazením, aby se nastavila lokalizace před zobrazením.

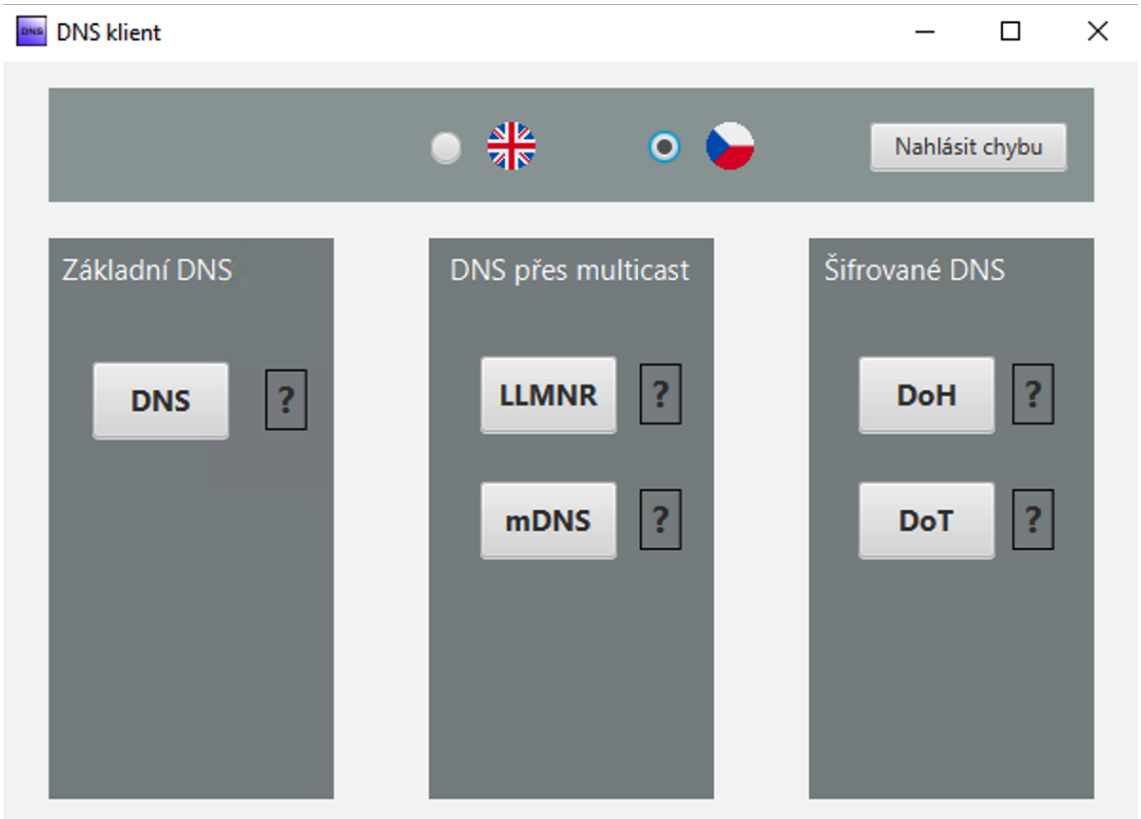
Výpis 3.2: Přeložení popisků při změně lokalizace

```
1  public void setLabels() {
2      TitledPane titlePaneArray [] = new TitledPane []{
3              domainNameTitledPane,
4          ipTitledPane,
5          dnssecTitledPane,
6          recordTypeTitledPane,
7          responseTitledPane,
8          queryTitledPane};
9
10     for (TitledPane titledPane : titlePaneArray) {
11       titledPane.setText(
12       language.getLanguageBundle().getString(
13       titledPane.getId())
14       );}
15       ... Stejný postup u dalších komponent
16     }
```

3.2.5 Grafické rozhraní

Při startu aplikace je zobrazeno okno aplikace, které je definováno v souboru Main.fxml. Jako kontrolér okna využívá třídu MainControler. Okno je zobrazeno na obrázku 3.4. Uživatel má možnost zvolit jazykovou lokalizaci a typ protokolu pro překlad. Aktuálně je podporován pouze DNS a mDNS. Při výběru jiného protokolu než zmíněných dvou, je zobrazeno chybové okno s informací, že protokol doposud nebyl implementován. Tlačítka jsou již nachystána pro další rozvoj aplikace. Vedoucí práce může v dalších letech vypsat nová témata právě na dopsaní funkcionality LLMNR, DoH, DoT a tím i rozšířit laboratorní úlohu ve výuce.

Po kliknutí na tlačítko mDNS je zobrazeno okno definované v souboru mDNS.fxml. Pro usnadnění práce bylo vytvořeno jako první, protože kontrolér DNS dědí právě z okna mDNS. Samotné rozhraní DNS má více volitelných parametrů než mDNS. Lze ušetřit duplicitní kód a využít kontrolér DNS jako potomka mDNS. Veškeré objekty typu JavaFX musí mít nastavenou viditelnost na protected. Okno DNS je zobrazeno na obrázku 3.5. Při zobrazení jsou dynamicky načteny IP adresy DNS serverů. Načtení je provedeno pomocí Powershell skriptu. Samotný proces je popsán v samostatné kapitole. Dále jsou načteny hodnoty ze souboru nastavení, pokud již program byl v minulosti spuštěn. Zároveň při zobrazení okna je dynamicky načteny všechny dostupné síťové rozhraní ze stanice, aby uživatel mohl vybírat odchozí rozhraní.

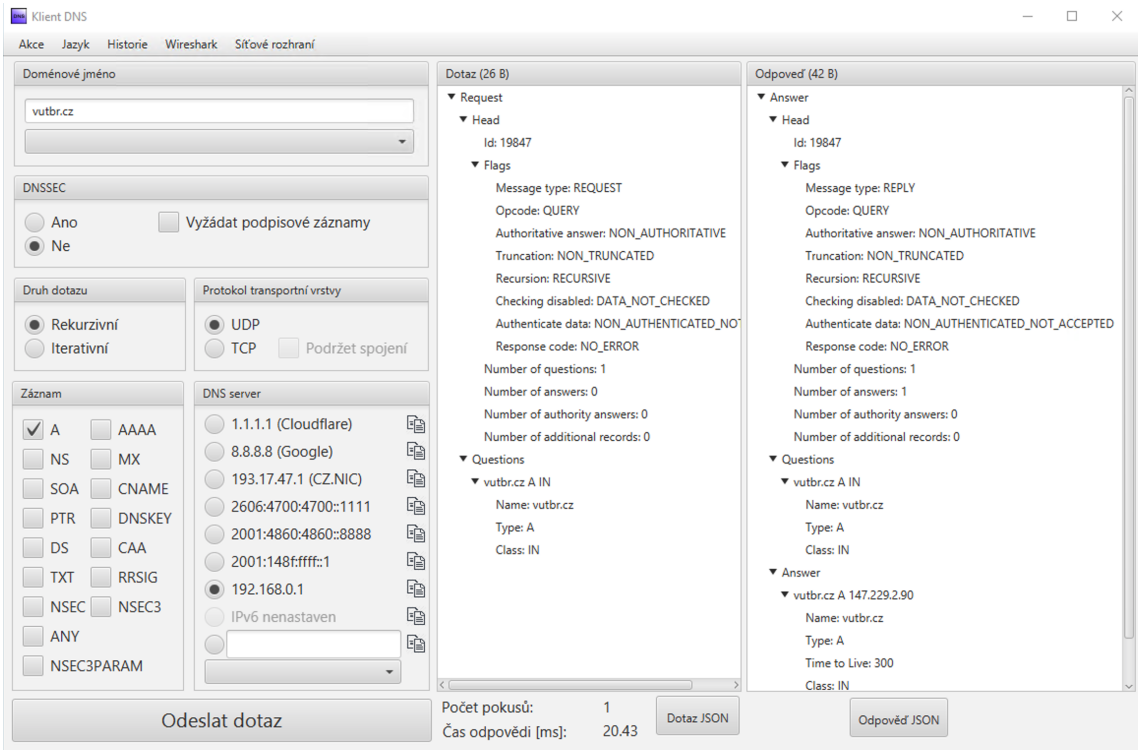


Obr. 3.4: První zobrazené okno vytvořené aplikace po spuštění programu.

Každé okno aplikace sloužící k odesílání přijímáni zprávy má vlastní kontrolér a ten se stará o vykonávání příkazů. Obsahuje vždy instance tříd MessageSender a MessageParser. MessageSender se stará o tvorbu, odeslání a přijetí zprávy v bajtové podobě. MessageParser má za úkol zprávu z bajtové podoby převést do čitelné podoby. Výsledkem překladu je objekt typu Json nebo stromová struktura, kde uživatel může jednotlivé uzly procházet a podívat se na hodnotu po překladu.

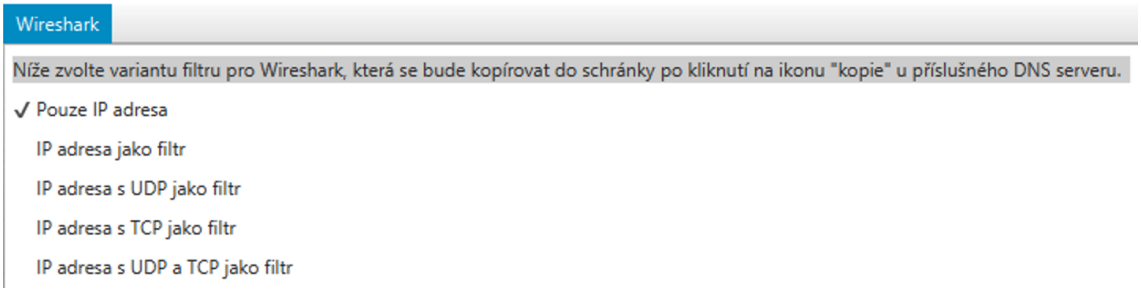
3.2.6 Grafické rozhraní pro zprávy DNS

Okno programu při zasílání zpráv je zobrazeno na obrázku 3.5. V menu liště se nachází 4 možnosti volby. Pod možností Akce je jediná položka: návratu do okna pro výběr protokolu. Dále je zde volba jazyka a historie, kde lze smazat historii vyhledávaných doménových jmen, tak i uživatelsky definovaných resolverů. Poslední menu je Wireshark. Tato položka slouží k ovlivnění filtru při kopírování do schránky. Vedle položek DNS server se nachází ikona kopírování, kdy po kliknutí na ikonu ve výchozím nastavení je do schránky zkopírována IP adresa serveru na daném řádku. V menu Wireshark lze měnit formát řetězce,



Obr. 3.5: Rozhraní DNS při přeloženém dotazu doménové jména vutbr.cz.

který je do schránky zkopírován. Položky menu Wireshark lze vidět na obrázku 3.6. Při práci s aplikací je velmi často využíván program na zachytávání provozu Wireshark a lze předpokládat, že při výuce bude využíván také. Pro ulehčení filtrování provozu si lze zkopírovat filtr pro tento program přímo z aplikace. Pokud uživatel bude chtít filtrovat provoz pro adresu 1.1.1.1 s transportním protokolem TCP. Může si vybrat položku: IP adresa s TCP jako filtr a po kliknutí na ikonu vedle DNS serveru 1.1.1.1. Je do schránky systému zkopírován řetězec: „ip.addr == 1.1.1.1 && tcp.port == 53“, který lze vložit do programu Wireshark pro filtrování právě zvoleného provozu.



Obr. 3.6: Položky menu Wireshark při lokalizaci do českého jazyka.

Další vlastností je nápověda z historie. Při zadávání znaků do pole domény

(i uživatelsky definovaného DNS serveru) jsou při každém zadaném znaku filtrovány názvy doménových jmen, kde se řetězec zadávaný v textovém poli vyskytuje. V případě, že se najde alespoň jeden záznam, automaticky vyjíždí rolovací panel a zobrazuje tyto možnosti. Pokud uživatel klikne přímo na rolovací panel myší, vyjede mu výpis všech doménových jmen, která jsou uložena v historii. Stejně popsaná funkcionalita je i u manuálně zadávaných DNS serverů.

Při dvojkliku na položku v odpovědi, nebo v dotazu, je hodnota položky automaticky kopírovaná do schránky paměti. Většina hodnot je zapsaná ve formátu parametr: hodnota. Do paměti je kopírovaná pouze hodnota. V momentu, kdy uživatel udělá dvojklik na položku, která není ve zmíněném tvaru, není do schránky kopírování nic. Typicky by se jednalo o dvojklik na Head, Flags a další názvy. Vlastnost má podporovat rychlé kopírování nalezených hodnot jak pro filtrování pro program Wireshark, tak i pro možné dotazy na doménové jména vrácená v odpovědi.

V rámečku, kde je označeno, zda zpráva je typu dotaz nebo odpověď se v závorce po úspěšném překladu objevuje počet bajtů zprávy. Při dotazu pomocí transportního protokolu UDP na dotaz, který nebyl rozdělen, lze pozorovat, že stejný dotaz zaslaný pomocí protokolu TCP má o 2 bajty více. Důvod je ten, že první dva bajty u přenosu TCP reprezentují velikost samotné DNS zprávy.

Do textového pole DNS serveru lze zadat i doménové jméno. Typické využití je u kořenových serverů, které jsou označovány písmenem A až M následuje doména root-servers.net. Samotné doménové jméno je přeloženo pomocí primárního resolveru systému a přeložená adresa využita pro překlad uživatelského dotazu. O překladu jména DNS serveru je uživatel informován skrze informační okno, kde se mu zobrazí hláška o překladu na pozadí. Do textového pole je vložena přeložená adresa a uživatel může získat Wireshark filtr pro danou adresu pomocí kopírovací ikony.

3.2.7 Výběr síťového rozhraní

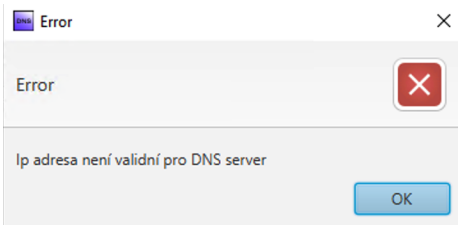
Další položkou v menu, kterou lze vybrat je síťové rozhraní. Po rozkliknutí se zobrazí seznam, který je dynamicky načítán a obsahuje veškeré síťové rozhraní na stanici. Při úvodním spuštění je automaticky vybráno rozhraní, které stanice udává jako hlavní. Ukončení programu vede k uložení aktuálního rozhraní do souboru s historii doménových jmen. Při dalším spuštění je toto rozhraní nastaveno jako výchozí.

Možnosti nastavení rozhraní si uživatel určuje zdrojovou adresu pro překlad. Význam má v situaci, kdy stanice má více rozhraní, které mají validně nastavenou adresu. Typickým příkladem může být stanice s ethernetovým a wifi rozhraním. Obě mohou být připojeny do jiné sítě, ale bez možnosti výběru v aplikaci by rozhraní wifi nemohlo být využito.

Význam má také u mDNS, kde lze zaslat dotaz do multicastové skupiny z různého rozhraní. Při testování aplikace tak šlo komunikovat s virtuální stanicí na virtuální síti, protože virtuální síť měla své rozhraní v hostitelském operačním systému.

3.2.8 Chybová hlášení programu

Při běhu programu se mohou vyskytnout chyby, které neumožňují programu pokračovat. Mezi takové patří například špatně zadané doménové jméno, nevalidní IP adresa nebo vypršení času při sestavování spojení. Pro informování uživatele u chybě je využito vyskakovací okno zobrazeno na obrázku 3.7. Každá chyba



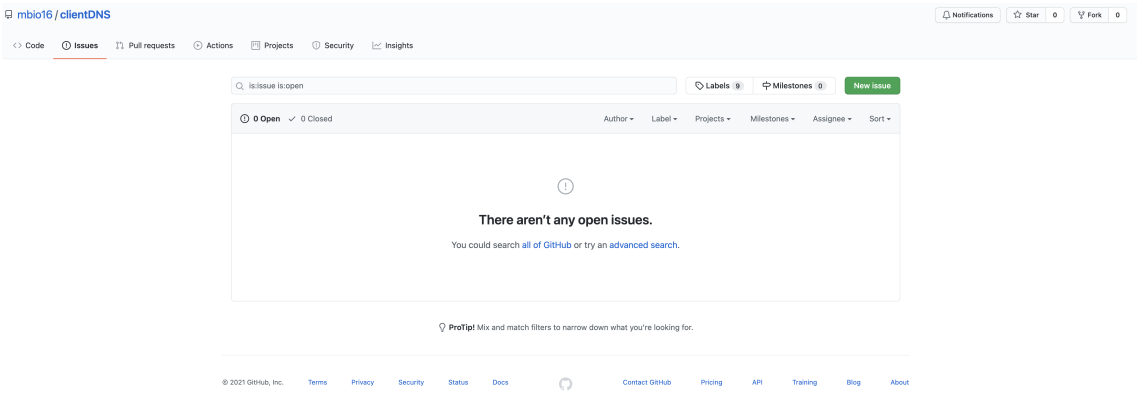
Obr. 3.7: Příklad chybového okna při zadání nevalidní IP adresy do pole pro DNS server.

má definovanou vlastní hlášku pro všechny lokalizační jazyky. Existuje jedna obecná chyba. Veškeré chyby nemající vlastní hlášku jsou zachyceny a zobrazena obecná hláška. V případě, že uživatel objeví chybu, která není samostatně ošetřena, může tuto skutečnost nahlásit. Pomocí tlačítka v úvodním okně. Po stisknutí tlačítka se otevře prohlížeč na webové stránce repozitáře projektu (`https://github.com/mbio16/clientDNS`) v záložce sloužící k nahlášení chyb. Chybu lze reportovat pomocí vytvoření tiketu, kde uživatel chybu popíše. Automaticky po vytvoření tiketu dojde email vývojáři o nově nahlášené chybě a může ji začít řešit. Stránka odkazu, kde se otevře webový prohlížeč je zobrazena na obrázku 3.8.

3.2.9 Kódování doménových jmen

Pro přenos doménového jména je nutné kódovat do požadovaného formátu. Pro příklad test.example.cz je nutné poslat jako řetězec 4test7example2cz0. Struktura převodu se skládá z číslic určující počet znaků subdomény a řetězců subdomény. Na konci řetězce je 0, která značí kořen (.). Inverzní proces je aplikován při přijmutí odpovědi a zobrazení doménového jména.

Původní znaková sada tvoří pouze americká abeceda, pomlčka, čísla a tečka (Ascii znaky). Aktuálně si lze zaregistrovat doménu se znakovou sadou i jiných

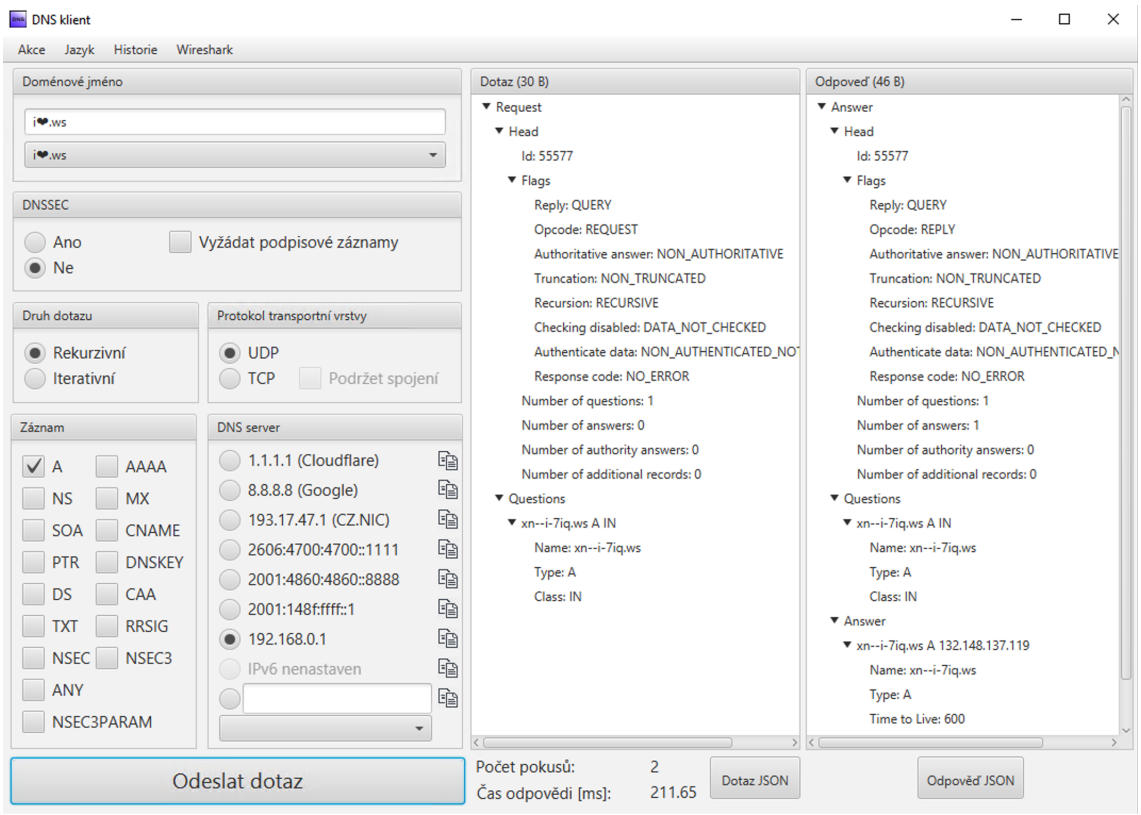


Obr. 3.8: Obrázek stránky pro nahlášení doposud neznámé chyby (aktuálně s 0 nahlášenými chybami).

abeced. Převod do Ascii znakové sady je pomocí Punycode, což je reprezentace Unikódu v základní znakové sadě. Pro příklad: jedna ze dvou funkčních domén v adresním prostoru .cz je `https://háčkyčárky.cz`. Po převodu do Punycodu je dotaz reprezentován řetězcem xn--hkyrky-ptac70bc.cz. Dnes často využívané jsou emotikony, které mohou být součástí domény. Příklad lze vidět na obrázku 3.9. Výsledný dotaz je reprezentován řetězcem xn--i-iq.ws. Stránka samotná je registrátor emoji domén. Následný postup je pak stejný jako u běžného dotazu. Oba převody implementuje třída DomainConvert. Pracuje jak s Ascii, tak i s Unicodovým doménovým jménem. Výsledkem je řetězec připravený k odeslání v rámci dotazu. Dekódování pracuje stejně jen jako inverzní operace.[35]

3.2.10 Komprese doménového jména

Kromě kódování doménového jména, popsaného v předešlé kapitole, lze využít i komprese. Příklad je zobrazen na obrázku 3.10. Celé doménové jméno je reprezentováno dvojicí bajtů. Kompresi lze odhalit pomocí prvního bajtu, kdy pokud nejvíce důležité dva bity jsou 1, jedná se o kompresi. Zbytek bitů (14) reprezentuje index první bajtu doménového jména ve zprávě. Pokud se podíváme na obrázek 3.10, jedná se o 12. bajt v právě došlé zprávě. Tímto lze ušetřit velké množství bajtů, které by museli být přeneseny pro každý záznam. Kompresi lze i kombinovat se subdoménami. Pokud v části dotazu je vutbr.cz a v části odpovědi je www.vutbr.cz je možné zakódovat pouze www a následující dva bajty pak být kompresní odkazující na vutbr.cz jiné části zprávy.

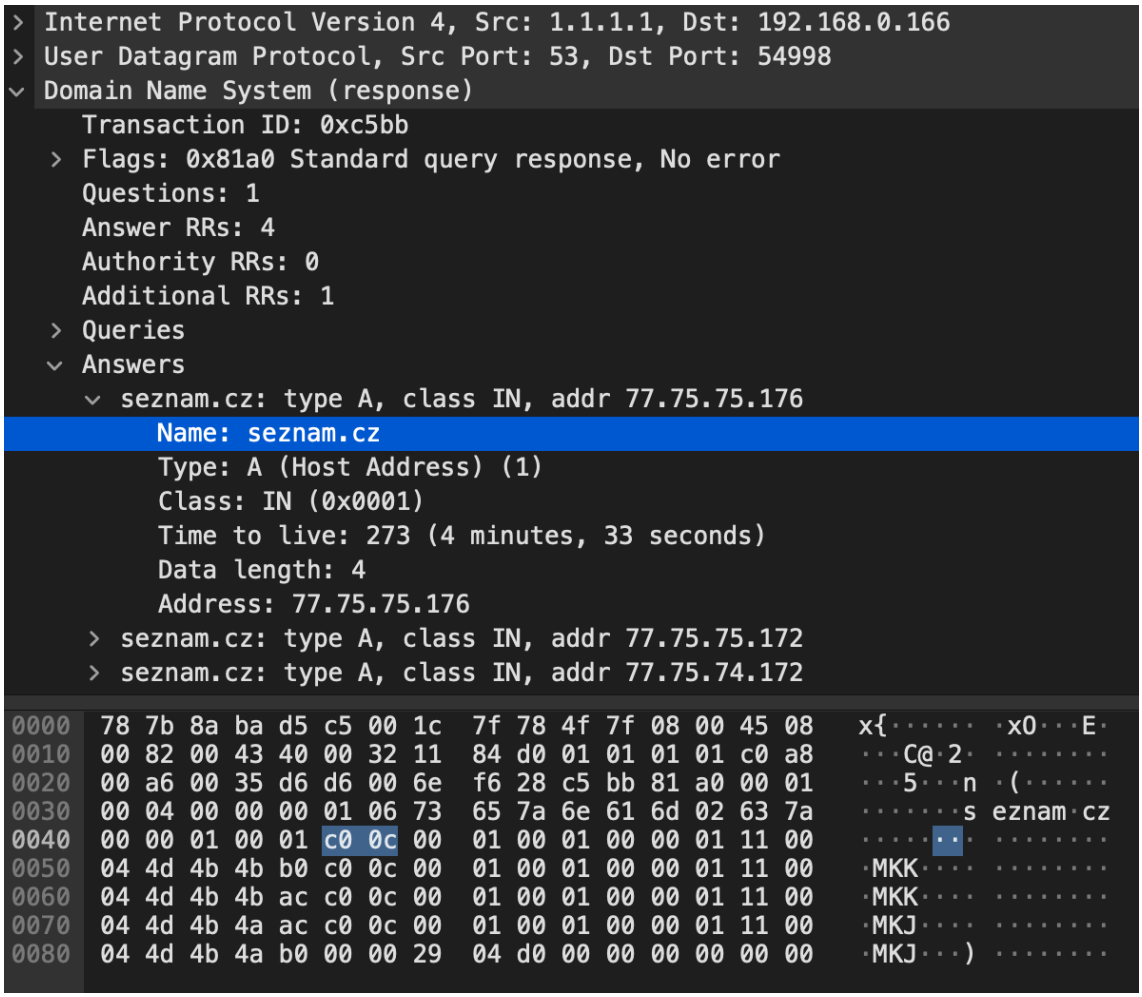


Obr. 3.9: Příklad přeložení doménového jména se znakem emotikonu.

3.2.11 Získání IP adres resolverů stanice

Vývojové prostředí Java 11 neobsahuje možnost systémového volání pro získání resolverů, které jsou nastaveny přímo na stanici. Je to zejména z důvodu, že ostatní aplikace si vystačí s možností dotazu na lokální síť a možnost překladu doménového jména přímo systémem.

Aplikace DNS klienta pro svůj chod potřebuje informace o IPv4, IPv6 resolveru, aby je mohla zařadit mezi možné resolvery, které lze pro překlad využít. Veškeré informace o systému lze na platformě Windows získat pomocí skriptovacího jazyka Powershell. Powershellové dotazy lze vykonat pomocí Javy. Aplikace volá systémový terminál, předává příkaz tvořen požadavkem na Powershell a samotným Powershell skriptem. Naslouchá na vrácený textový řetěz. Každý řádek reprezentuje jeden resolver získaný ze všech rozhraních (i virtuálních). Příkaz pro získání resolverů síťových rozhraní je zobrazen ve výpisu 3.3. V případě, že uživatel nemá Powershell nainstalován (je součástí výchozí instalace) nebo není Powershell povolen, neznamená to konec aplikace. Uživatel pouze neuvidí lokálně využívaný resolver v seznamu resolverů. Příklad, kdy IPv4 adresa resolveru na stanici je nastavena, ale IPv6 nikoliv je zobrazen na obrázku 3.11.



Obr. 3.10: Příklad komprese doménového jména, kdy se v části odpovědi odkazuje do dotazu, kde je zbytek doménového jména.

Získání adres resolverů pomocí Powershell skriptu je prováděno při startu aplikace, jelikož spuštění Powershell konzole je časově náročné. Od chvíle spuštění aplikace do zobrazení úvodního okna může uplynout poněkud delší doba než je u jiných nativních aplikacích pro platformu Microsoft Windows. Výsledné adresy jsou předávány napříč kontroléry ve formě objektu, aby chod aplikace nemusel být zpomalován opětovným spouštěním Powershell konzole.

Výpis 3.3: Powershell skript pro získání lokálně nastavených resolverů

```
1 powershell.exe $ip=Get-NetIPConfiguration;
2 $ip.'DNSServer' | ForEach-Object -Process {$_.ServerAddresses}
```

Obr. 3.11: Příklad načtení programu na stanici, která má nastavený IPv4 resolver, nikoli však IPv6 resolver.

3.2.12 Dotaz na reverzní záznam

Speciální typ dotazu je reverzní (PTR), kdy program očekává místo doménového jména IPv4 popřípadě IPv6 adresu. Program provede vytvoření dotazu včetně správně zapsané IP adresy s doménovým jménem. Pro příklad dotaz na 8.8.4.4 bude programem převeden na 4.4.8.8.in-addr.arpa a následně zaslán.

Limitace dotazu PTR je pouze, že nelze již kombinovat s dalšími druhy záznamů, jelikož v ostatních záznamech se očekává validní doménové jméno. Při volbě záznamu PTR ještě s dalšími záznamy se automaticky aktivuje chyba, která vyvolá zobrazení chybové hlášky tak, aby uživatel pochopil, že dotaz musí být samostatně.

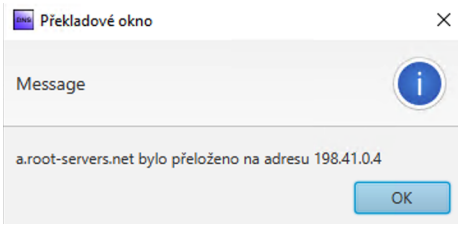
Stejná funkce je i pro IPv6, kdy dotaz na validní IPv6 je transformován a odeslán. Uživatel má ale také možnost zadat přímo 4.4.8.8.in-addr.arpa s možností reverzního záznamu. Vytvořený program rozpozná doménu .arpa a zašle dotaz přímo bez transformace adresy.

3.2.13 Uživatelsky volený DNS server

Uživatel si má možnost vybrat z předvolených resolverů společnostech Cloudflare, Google, cz.nic a resolverů načtených ze stanice. Pokud chce zvolit jiný resolver, může zadat IP adresu resolveru do textového pole, který je předposlední položka. Po zadaní validní IP adresy dojde k zaslaní dotazu právě na tuto destinaci.

DNS servery mají i své doménové jména a aplikace umožňuje do pole pro resolver zadat i doménové jméno. Typickým příkladem můžou být kořenové servery, které jsou označovány písmeny A až M a následuje doména root-servers.net. Při zadání doménového jména je na pozadí přeloženo pomocí resolveru stanice. Získaná IP adresa je využita pro překlad již zadaného doménového jméno z aplikace. Zároveň je uživatel informován pomocí informačního okna, že doménové jméno bylo přeloženo na IP adresu a IP adresa nahrazuje zadané doméno jméno v textovém poli, kde se doposud nacházelo doménové jméno.

Informování o překladu na pozadí má uživateli připomenout, že nelze se dotazovat přímo na doménové jméno pomocí doménového jména DNS serveru, ale je nutný mezikrok v podobě překladu DNS serveru, až poté je možné zaslat uživatelský dotaz. Informační okno po zadání DNS serveru a.root-servers.net je zobrazeno na obrázku 3.12.



Obr. 3.12: Informační okno při zadání DNS serveru a.root-servers.net.

3.2.14 Měření času

Aplikace zobrazuje uplynulý čas od odeslání zprávy po přijetí odpovědi (pokud odpověď vůbec přijde). Samotné měření počítá uplynulý čas v systémových nanosekundách a výsledek převádí do milisekund. Hlavní výhodou měření času řádově nižších jednotkách je přesnost, protože při využívání systémových milisekund dochází k zaokrouhlování a při testování programu to vedlo ke stejným výsledkům, i když časy pozorované ve Wiresharku byly rozdílné.

Pro měření u zpráv zasílaných přes protokol UDP je zahájeno měření příkazem před odesláním datagramu na server a zastaveno při přijetí odpovědi. U TCP mohou nastat dvě situace. Je potřeba vytvořit nové spojení, nebo lze využít podržené spojení. Při vytváření nového spojení je měření spuštěno před vytvořením spojení pomocí 3 cestného handshaku a zastaveno při přijetí odpovědi. V případě, že spojení je již navázané, není ve spojení prodleva způsobená navazováním spojení a měření je spuštěno před odesláním dotazu. Díky velice přesnému měření lze pozorovat rychlost překladu v závislosti na: geolokační pozici DNS serveru, využití transportního protokolu, cachování výsledků resolvery po cestě a dalšími faktory.

3.2.15 Datový typ UInt16

V balíčku models se nachází i třída UInt16. Slouží k uložení datového typu unsigned integer o velikosti 16 bitů. Programovací jazyk Java pracuje s základním datovým typem integer, který má 32 bitů. Důvodem vlastního datového typu byly dvě vlastnosti 32 bitového integeru, které nelze kombinovat s protokolem DNS. První omezení je na minimální velikost 0 a maximum $2^{16} - 1 = 65535$. Při vytvoření proměnné je kontrolováno, že zadané číslo je v daném rozsahu. Druhým důvodem je převod do bajtů. Kde původní integer je tvořen 4 bajty, což při vytváření například hlavičky dotazu způsobuje chybu. Zde musí mít číslo přesně 2 bajty. Využití datového typu short není možné, protože maximální velikost typu short je $2^{15} - 1$. V důsledku by to znamenalo, že určité informace uložené v posledním bitu by byly ztraceny a výsledná zpráva deformována.

Nový datový typ UInt16 se stará o kontrolu rozsahu a zavádí metody pro konverzi hodnoty na 2 bajty a zpět. Zároveň obsahuje možnost generovat náhodné číslo v 16 bitovém rozsahu. Funkce je využita při vytváření hlavičky pro pole QID. Třídu UInt16 lze vidět v příloze C.3.

3.2.16 Třídy typu Enum

Samostatný balíček mají pouze třídy, které dědí z třídy Enum. Slouží k ukládání konstant a reprezentaci jejich hodnot. Příkladem může být třída DNSSEC_ALGORITHM_TYPE. Využita je v záznamech, které jsou spojeny s asymetrickou kryptografií tedy v záznamu RRSIG, DNSKEY. V obou těchto záznamech existuje pole, které má velikost jednoho bajtu a určitá hodnota reprezentuje jiný kryptogramický algoritmus. Například hodnota 5 je algoritmus RSA s hashovací funkcí SHA1. Pro jednoduchou práci s hodnotami jsou reprezentovány konstantou. Zároveň lze získat tento typ z bajtu díky metodě getTypeByCode. Konvence u všech tříd typu Enum je psát název i konstanty velkými písmeny a slova oddělovat dolním podtržítkem. Třída DNSSEC_ALGORITHM_TYPE je zobrazena ve výpisu 3.4. [36]

3.2.17 Zprávy ve formátu Json

Při přijaté odpovědi od resolveru se automaticky žádost a odpověď zobrazí v grafickém okně jako TreeView, kde si uživatel může zprávy procházet a seznámit se s jednotlivými částmi. Při ladění odesílání a přijímaní zpráv se pro zobrazení výsledků využívala konzole, kde formát TreeView není kompatibilní. Pro konzolové zobrazení se využívá formát Json, který je snadno čitelný pro uživatele a zároveň se jedná také o strojově čitelný formát. V současné době preferovaný před formátem Xml.

Vedle popisků s časem dotazu a počtem zaslaných zpráv se nacházejí dvě tlačítka pro získání zprávy ve formátu Json. Obě tlačítka jsou neaktivní před zasláním dotazu aktivují se automaticky v situaci, kdy se zobrazí obě zprávy v objektech typu TreeView. Po kliknutí je Json zpráva kopírována do schránky a lze si ji otevřít v jakémkoliv textovém editoru. Samotný formát je upraven pomocí knihovny Gson, která se stará o odřádkování, aby výsledná podoba byla, co nejvíce čitelná pro uživatele. Příklad odpovědi na žádost o záznam typu A na doménu www.vutbr.cz je zobrazen ve výpisu 3.5.

Výpis 3.4: Příklad konkrétní třídy enum

```
1  public enum DNSSEC_ALGORITHM_TYPE {
2    RESERVED((byte)0x00),
3    RSA_MD5((byte) 0x01),
4    DH((byte) 0x02),
5    DSA((byte) 0x03),
6    RESERVED2((byte) 0x04),
7    RSA_SHA1((byte)0x05),
8    DSA_NSEC3_SHA1((byte) 0x06),
9    RSASHA1_NSEC3_SHA1((byte) 0x07),
10   RSA_SHA256((byte) 0x08),
11   RSA_SHA512((byte) 0x09),
12   RESERVED3((byte) 0x0a),
13   ECC_GOST((byte) 0x0b),
14   ECDSA_P256_SHA256((byte) 0x0c),
15   ECDSA_P384_SHA384((byte) 0x0d),
16   ED25519((byte)0x0e),
17   ED448((byte) 0x0f);
18
19   private byte code;
20
21   private DNSSEC_ALGORITHM_TYPE(byte code) {
22     this.code= code;
23   }
24
25     public static DNSSEC_ALGORITHM_TYPE
26     getTypeByCode(byte code){
27         for(DNSSEC_ALGORITHM_TYPE e :
28         DNSSEC_ALGORITHM_TYPE.values()){
29             if(e.code == code) return e;
30         }
31         return null;
32     }
33 }
```

Výpis 3.5: Příklad odpovědi ve formátu Json

```
{
  "Head": {
    "Number of answers": 1,
    "Number of authority answers": 0,
    "Opcode": "QUERY",
    "Fragmented": false,
    "Authenticate data": true,
    "Recursion": true,
    "Authoritative answer": false,
    "Checking disabled": false,
    "Number of additional records": 0,
    "Number of questions": 1,
    "Reply": true,
    "Id": 61608,
    "Response code": "NO_ERROR"
  },
    "Questions": [
    {
      "Type": "A",
      "Class": "IN",
      "Name": "www.vutbr.cz"
    }
   ],
 "Answer": [
    {
      "Type": "A",
      "Time to Live": 278,
      "Class": "IN",
      "Data": {
        "Ipv4": "147.229.2.90"
      },
      "Name": "www.vutbr.cz"
    }
   ],
  "Authority": [],
  "Aditional records": [],
}
```

3.3 Testování aplikace

3.3.1 Uživatelské testování aplikace

Pro zajištění správné funkcionality programu bylo potřeba vyzkoušet program s různými kombinacemi nastavení. Pokud není v testu definováno jinak, jsou výchozí parametry nastaveny na doménové jméno vutbr.cz, transportní protokol UDP, rekurzivní typ dotazu, vypnut DNSSEC bez vyžádání podpisových záznamů, dotaz na záznam typu A a DNS server je 1.1.1.1 (Cloudflare). Výsledky testování jednotlivých vstupů jsou zaznamenány v této kapitole.

Testovaná doménová jména

- **vutbr.cz** – výsledek: OK
- **vutbr.cz.** – výsledek: OK
- **cz** – výsledek: OK
- **cz.** – výsledek: OK
- **8.8.4.4** (PTR záznam) – výsledek: OK
- **4.4.8.8.in-addr.arpa** (PTR záznam) – výsledek: OK
- **2606:4700:4700::1111** (PTR záznam) – výsledek: OK
- **1.1.1.1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.7.4.0.0.7.4.6.0.6.2.ip6.arpa** (PTR záznam) – výsledek: OK
- **háčkyčárky.cz** – výsledek: OK
- **háčkyčárky.cz.** – výsledek: OK
- **xn–hkyrky-ptac70bc.cz** – výsledek: OK
- **Nevalidní doména (a.)** – výsledek: OK (Chybová hláška)

Testování druhu záznamu

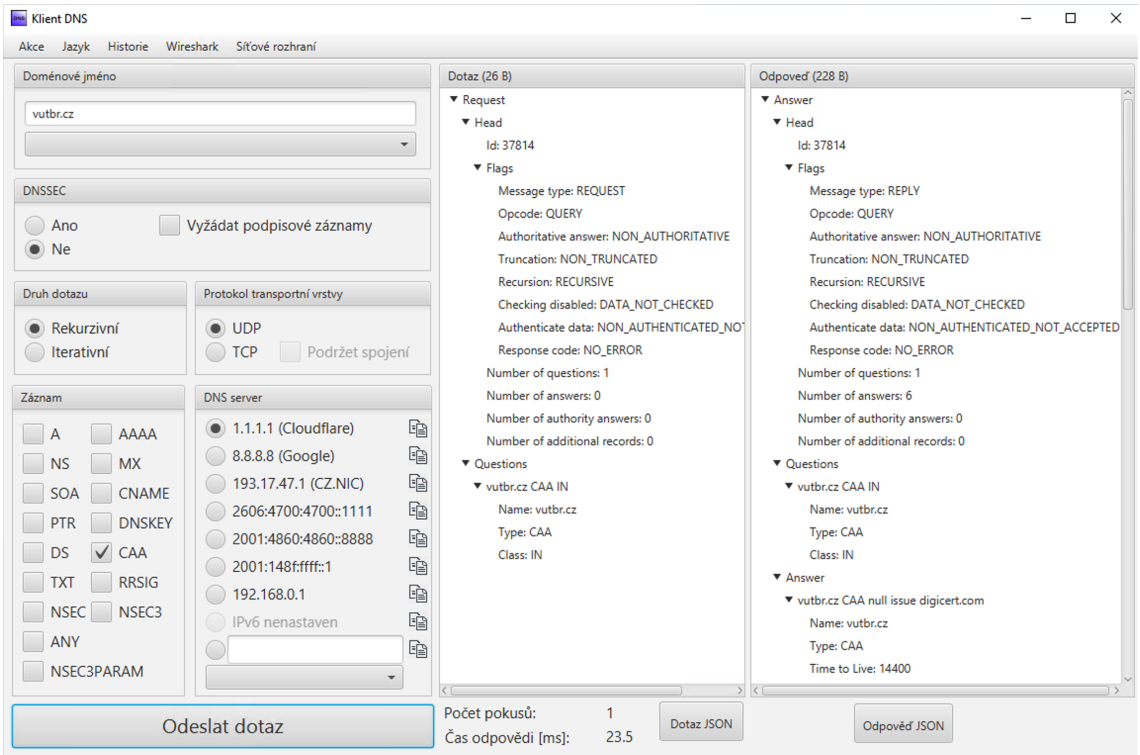
- **A** (doména vutbr.cz) – výsledek: OK
- **AAAA** (doména nic.cz) – výsledek: OK
- **NS** (doména vutbr.cz) – výsledek: OK
- **MX** (doména vutbr.cz) – výsledek: OK
- **SOA** (doména vutbr.cz) – výsledek: OK
- **CNAME** (doména btcpay.biolek.net) – výsledek: OK
- **PTR** (doména 8.8.4.4) – výsledek: OK
- **DNSKEY** (doména nic.cz) – výsledek: OK
- **DS** (doména vutbr.cz) – výsledek: OK
- **CAA** (doména vutbr.cz) – výsledek: OK
- **TXT** (doména vutbr.cz) – výsledek: OK
- **RRSIG** (doména nic.cz) – výsledek: OK

- **NSEC** (doména vutbr.cz) – výsledek: OK
- **NSEC3** (doména cz) – výsledek: OK
- **NSEC3PARAM** (doména cz) – výsledek: OK
- **ANY** (doména cz) – výsledek: OK
- **A + AAAA** (vutbr.cz) – výsledek: OK (přeložen pouze první dotaz)
- **A + AAAA + MX** (doména vutbr.cz)– výsledek: OK (přeložen pouze první dotaz)
- **A + PTR** (doména vutbr.cz) – výsledek: OK (Chybová hláška, PTR nelze kombinovat s jinými dotazy)

Příklad dotazu na záznam CAA je zobrazen na obrázku 3.13.

Testování DNSSEC

- **Vypnutý DNSSEC** – výsledek: OK
- **Vyžadovat ověření resolverem** – výsledek: OK
- **Vypnutý DNSSEC + vyžádat si podpisové záznamy** – výsledek: OK
- **Vyžádat si ověření resolverem + vyžádat si podpisové záznamy** – výsledek: OK



Obr. 3.13: Příklad dotazu na doménové jméno vutbr.cz a záznam CAA.

Testování druhu dotazu

- **Rekurzivní dotaz** – výsledek: OK
- **Iterativní dotaz** – výsledek: OK

Testování transportního protokolu

- **UDP** – výsledek: OK
- **TCP** – výsledek: OK
- **TCP s podrženým spojení + zasláním více dotazů (CloudFlare resolver)** – výsledek: OK
- **TCP s podrženým spojení + zasláním více dotazů (Google resolver)** – výsledek: OK (Chybová hláška, resolver uzavírá spojení)
- **TCP s podrženým spojením + další dotaz přes UDP** – výsledek: OK

Testování síťového rozhraní

- **Ethernetové rozhraní s IP adresou** – výsledek: OK
- **Wifi rozhraní s IP adresou** – výsledek: OK
- **Virtuální rozhraní s IP adresou** – výsledek: OK
- **Ethernetové rozhraní bez IP adresy** – výsledek: OK (chybová hláška)
- **Wifi rozhraní bez IP adresy** – výsledek: OK (chybová hláška)
- **Virtuální rozhraní bez IP adresy** – výsledek: OK (chybová hláška)

Testování DNS serveru

- **1.1.1.1** – výsledek: OK
- **8.8.8.8** – výsledek: OK
- **193.17.47.1** – výsledek: OK
- **2606:4700:4700::1111** – výsledek: OK
- **2001:4860:4860::8888** – výsledek: OK
- **2001:148f:ffff::1** – výsledek: OK
- **IPv4 resolver nastaven na stanici** – výsledek: OK
- **IPv6 resolver nastaven na stanici** – výsledek: OK
- **Vlastní IPv4 DNS server** – výsledek: OK
- **Vlastní IPv6 DNS server** – výsledek: OK
- **Doménové jméno DNS serveru** – výsledek: OK (zobrazena informace o překladu DNS serveru)
- **Nevalidní IPv4 adresa** – výsledek: OK (Chybová hláška)
- **Nevalidní IPv6 adresa** – výsledek: OK (Chybová hláška)

3.3.2 Odlišnosti chování různých testovaných DNS serverů

Při testování grafického programu bylo pozorováno, že resolvery se chovají odlišně při stejném dotazu. Příklad plného otestování resolverů je na doméně cz. Dotazováno byly resolvery s různými parametry a výsledky byly zaznamenány do tabulek 3.2 a 3.3. Z výstupů lze pozorovat, že jednotlivé resolvery reagují odlišně při iterativní zprávě. Ačkoliv význam odpovědi je stejný. Daný záznam neznají a iterativní typ dotazu jim neumožňuje dotaz vyhledávat. Cloudflare a Google odpovídají s chybou dotazu nastavenou v hlavičce DNS zprávy. Zároveň Google není ve své odpovědi konstantní a odpověděl s validní odpovědí. Situaci lze vysvětlit, že se na stejný parametr ptala jiná stanice pomocí rekurzivního dotazu a po určitou dobu byl záznam cachován v paměti. V momentu, kdy se záznam vyskytoval v paměti, byl zaslán iterativní dotaz z klientské aplikace. Server tak mohl odeslat validní odpověď bez nutnosti vyhledávání na jiných serverech. Velice podobně se chová i u IPv6. První zprávy jsou chybové pro protokol UDP, ale informace jsou cachovány a při dotazu TCP, lze vidět, že resolver odpověděl validně i na iterativní dotazy.

Poslední sloupec tabulky má název poměr zesílení. Jedná se o poměr bajtové velikosti zpráv přijaté ku odeslané. Lze pozorovat, že pomocí velice malé zprávy lze získat odpověď, která je několikanásobně větší. V aktuálním případě bylo největší zesílení více než 12 násobné. V běžném provozu se lze setkat se zesílením i 100 násobným. Pomocí veřejně známé domény seznam.cz a dotazu na záznamy RRSIG s žádostí o podpisové záznamy pomocí protokolu UDP lze vygenerovat zesílení 23.1 násobné. Výrazně vyšší zesílení je možné vytvořit v rámci transportního protokolu TCP, avšak zajímavější je více pro protokol UDP, protože pomocí UDP lze i na pomalé lince vytvořit útok DDOS (Distributed Denial-of-service attack). Útočník podvrhne zdrojovou adresu, kde vloží adresu oběti a zasílá dotazy na veřejné resolvery, které odpovídají na adresu oběti. Oběť je zahlcena datagramy, na které se neptala, ale nemůže využít své připojení, protože celkovou kapacitu linky zabírají nevyžádané překlady.[37]

Domácí resolver (192.168.0.1) pro svůj překlad využívá dále DNS server 213.46.172.36. Podle záznamu PTR se jedná o server s doménovým jménem cz-prg01a-dns01.chello.cz. Na základě geolokace IP adresy lze předpokládat umístění v Amsterdamu. Nastaven je pomocí DHCP, kdy domácí resolver v podobě výchozí brány je DHCP klientem a nastavení DNS serveru přebírá právě z protokolu DHCP. Poskytovatel pro domácí síť je původně UPC dnes již po akvizici společnost Vodafone.

Dotazování pomocí IPv6 bylo prováděno ze stanice ze sítě VUT. Při využití resolveru 2001:67c:1220:9847::93e5:470e lze vidět, že neexistuje zpráva s odpovědí,

Tab. 3.2: Příklad výstupu testování klientské aplikace pro doménové jméno cz a typ záznamu A. při využití IPv4 protokolu.

IP protokol	Transportní protokol	Rekurzivní (R) / Iterativní (I)	DNSSEC podpisové záznamy	Resolver	Velikost dotazu [B]	Kód odpovědi	Typ záznamu	Velikost odpovědi [B]	Rychlost odpovědi [ms]	Poměr zesílení
IPv4	UDP	I	NE	1.1.1.1	20	Chyba serveru		20	21.76	1.00
IPv4	UDP	R	NE	1.1.1.1	20	OK	SOA	76	27.33	3.80
IPv4	UDP	I	ANO	1.1.1.1	31	Chyba serveru		31	16.41	1.00
IPv4	UDP	R	ANO	1.1.1.1	31	OK	SOA, RRSIG (2), NSEC3	373	15.51	**12.03**
IPv4	TCP	I	NE	1.1.1.1	22	Chyba serveru		22	58.91	1.00
IPv4	TCP	R	NE	1.1.1.1	22	OK	SOA	78	181.94	3.55
IPv4	TCP	I	ANO	1.1.1.1	33	Chyba serveru		33	32.05	1.00
IPv4	TCP	R	ANO	1.1.1.1	33	OK	SOA, RRSIG (2), NSEC3	373	35.13	11.30
IPv4	UDP	I	NE	8.8.8.8	20	OK	SOA	76	24.69	3.80
IPv4	UDP	R	NE	8.8.8.8	20	OK	SOA	76	33.60	3.80
IPv4	UDP	I	ANO	8.8.8.8	31	OK	SOA, RRSIG (2), NSEC3	373	22.69	**12.03**
IPv4	UDP	R	ANO	8.8.8.8	31	OK	SOA, RRSIG (2), NSEC3	373	26.90	**12.03**
IPv4	TCP	I	NE	8.8.8.8	22	Chyba serveru		22	117.60	1.00
IPv4	TCP	R	NE	8.8.8.8	22	OK		78	86.99	3.55
IPv4	TCP	I	ANO	8.8.8.8	33	Chyba serveru		33	67.02	1.00
IPv4	TCP	R	ANO	8.8.8.8	33	OK	SOA, RRSIG (2), NSEC3	373	259.01	11.30
IPv4	UDP	I	NE	193.17.47.1	20	Zamítnutí dotazu		20	18.00	1.00
IPv4	UDP	R	NE	193.17.47.1	20	OK	SOA	76	49.00	3.80
IPv4	UDP	I	ANO	193.17.47.1	31	Zamítnutí dotazu		31	32.15	1.00
IPv4	UDP	R	ANO	193.17.47.1	31	OK	SOA, RRSIG (2), NSEC3	373	25.42	**12.03**
IPv4	TCP	I	NE	193.17.47.1	22	Zamítnutí dotazu		22	182.72	1.00
IPv4	TCP	R	NE	193.17.47.1	22	OK	SOA	78	37.35	3.55
IPv4	TCP	I	ANO	193.17.47.1	33	Zamítnutí dotazu		33	54.02	1.00
IPv4	TCP	R	ANO	193.17.47.1	33	OK	SOA, RRSIG (2), NSEC3	373	33.09	11.30
IPv4	UDP	I	NE	192.168.0.1	20	OK		20	15.49	1.00
IPv4	UDP	R	NE	192.168.0.1	20	OK	SOA	76	34.61	3.80
IPv4	UDP	I	ANO	192.168.0.1	31	OK		31	30.60	1.00
IPv4	UDP	R	ANO	192.168.0.1	31	OK	SOA, RRSIG (2), NSEC3	373	25.42	**12.03**
IPv4	TCP	I	NE	192.168.0.1	22	OK		22	52.30	1.00
IPv4	TCP	R	NE	192.168.0.1	22	OK	SOA	78	33.63	3.55
IPv4	TCP	I	ANO	192.168.0.1	33	OK		33	209.69	1.00
IPv4	TCP	R	ANO	192.168.0.1	33	OK	SOA, RRSIG (2), NSEC3	373	36.00	11.30

kde není ani jeden záznam. U iterativního dotazu protokolem UDP a bez DNSSEC podpisů je odpovězeno celkem 12 záznamy, které jsou typu A, AAAA a NS pro doménu nic.cz. Lze předpokládat, že resolver má statické záznamy uložené přímo ve své zóně pro doménu cz, protože je velmi často využívaná a NS záznamy

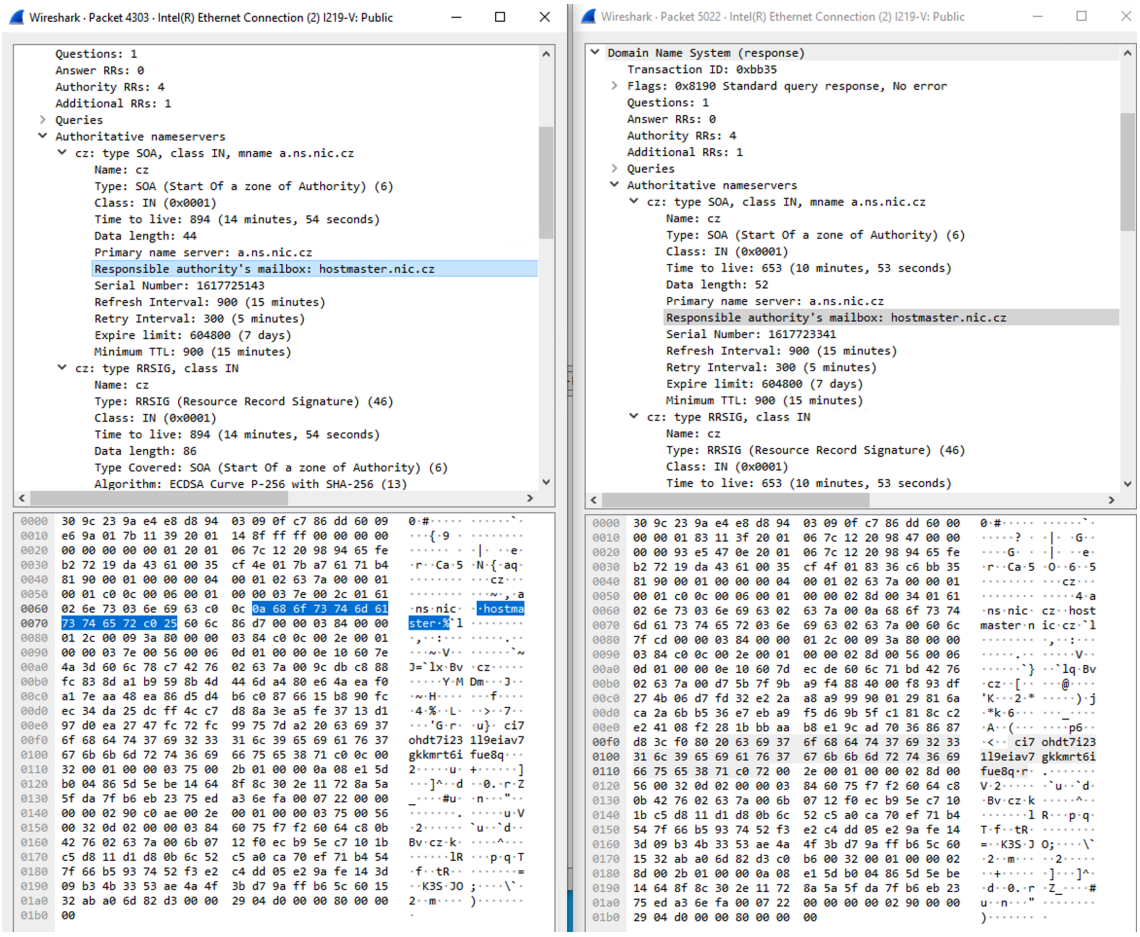
Tab. 3.3: Příklad výstupu testování klientské aplikace pro doménové jméno cz a typ záznamu A. při využití IPv6 protokolu.

IP protokol	Transportní protokol	Rekurzivní (R) / Iterativní (I)	DNSSEC podpisové záznamy	Resolver	Velikost dotazu [B]	Kód odpovědi	Typ záznamu	Velikost odpovědi [B]	Rychlost odpovědi [ms]	Poměr zesílení
IPv6	UDP	I	NE	2606:4700:4700::1111	20	Chyba serveru		20	4.66	1.00
IPv6	UDP	R	NE	2606:4700:4700::1111	20	OK	SOA	76	5.09	3.80
IPv6	UDP	I	ANO	2606:4700:4700::1111	31	Chyba serveru		31	4.49	1.00
IPv6	UDP	R	ANO	2606:4700:4700::1111	31	OK	SOA, RRSIG (2), NSEC3	371	5.28	11.97
IPv6	TCP	I	NE	2606:4700:4700::1111	22	Chyba serveru		22	9.23	1.00
IPv6	TCP	R	NE	2606:4700:4700::1111	22	OK	SOA	78	15.76	3.55
IPv6	TCP	I	ANO	2606:4700:4700::1111	33	Chyba serveru		33	9.59	1.00
IPv6	TCP	R	ANO	2606:4700:4700::1111	33	OK	SOA, RRSIG (2), NSEC3	373	11.06	11.30
IPv6	UDP	I	NE	2001:4860:4860::8888	20	Chyba serveru		20	12.36	1.00
IPv6	UDP	R	NE	2001:4860:4860::8888	20	OK	SOA	76	12.44	3.80
IPv6	UDP	I	ANO	2001:4860:4860::8888	31	Chyba serveru		31	12.33	1.00
IPv6	UDP	R	ANO	2001:4860:4860::8888	31	OK	SOA, RRSIG (2), NSEC3	371	14.28	11.97
IPv6	TCP	I	NE	2001:4860:4860::8888	22	OK	SOA	78	16.36	3.55
IPv6	TCP	R	NE	2001:4860:4860::8888	22	OK	SOA	78	7.89	3.55
IPv6	TCP	I	ANO	2001:4860:4860::8888	33	OK	SOA, RRSIG (2), NSEC3	373	16.09	11.30
IPv6	TCP	R	ANO	2001:4860:4860::8888	33	OK	SOA, RRSIG (2), NSEC3	373	16.37	11.30
IPv6	UDP	I	NE	2001:148f:ffff::1	20	Zamítnutí dotazu		20	4.71	1.00
IPv6	UDP	R	NE	2001:148f:ffff::1	20	OK	SOA	76	4.87	3.80
IPv6	UDP	I	ANO	2001:148f:ffff::1	31	Zamítnutí dotazu		31	4.63	1.00
IPv6	UDP	R	ANO	2001:148f:ffff::1	31	OK	SOA, RRSIG (2), NSEC3	371	15.37	11.97
IPv6	TCP	I	NE	2001:148f:ffff::1	22	Zamítnutí dotazu		22	31.75	1.00
IPv6	TCP	R	NE	2001:148f:ffff::1	22	OK	SOA	78	31.81	3.55
IPv6	TCP	I	ANO	2001:148f:ffff::1	33	Zamítnutí dotazu		33	31.68	1.00
IPv6	TCP	R	ANO	2001:148f:ffff::1	33	OK	SOA, RRSIG (2), NSEC3	373	34.56	11.30
IPv6	UDP	I	NE	2001:67c:1220:9847::93e5:470e	20	OK	NS(4),AAAA(4), A(4)	269	0.62	**13.45**
IPv6	UDP	R	NE	2001:67c:1220:9847::93e5:470e	20	OK	SOA	84	0.53	4.20
IPv6	UDP	I	ANO	2001:67c:1220:9847::93e5:470e	31	OK	SOA, RRSIG (2), NSEC3	379	0.67	**12.23**
IPv6	UDP	R	ANO	2001:67c:1220:9847::93e5:470e	31	OK	SOA, RRSIG (2), NSEC3	379	0.66	**12.23**
IPv6	TCP	I	NE	2001:67c:1220:9847::93e5:470e	22	OK	SOA	86	1.44	3.91
IPv6	TCP	R	NE	2001:67c:1220:9847::93e5:470e	22	OK	SOA	86	1.46	3.91
IPv6	TCP	I	ANO	2001:67c:1220:9847::93e5:470e	33	OK	SOA, RRSIG (2), NSEC3	381	1.35	11.55
IPv6	TCP	R	ANO	2001:67c:1220:9847::93e5:470e	33	OK	SOA, RRSIG (2), NSEC3	381	1.46	11.55

se nemění, lze tak velice snížit čas transakce. Nejrychlejší odpověď byla právě u toho resolveru, a to pod 1 milisekundu. U toho resolveru lze pozorovat i efektivní práce s cachovanými údaji, které jsou využity při překladu u UDP i TCP zpráv. Jedná se o nejlepší výsledek ze všech testovaných resolverů.

Dotazování je pro resolver VUT sítě velice rychlé, avšak si lze povšimnout, že zprávy jsou větší než u jiných resolverů. Poměr zesílení je tak i vyšší. Při pohledu přímo vytvořené aplikace lze vidět pouze změnu velikosti zpráv. Informace v odpovědích jsou stejné. Při zachycení průběhů dotazů programem

Wireshark lze vidět, že resolver VUT neprovádí kompresi doménového jména. Ostatní resolvery jsou schopni ušetřit 8 bajtů k přenesení stejné informace. Rozdíl je patrný na obrázku 3.14.



Obr. 3.14: Rozdílné délky zpráv pro stejný požadavek pro dva různé resolvery, kde jeden podporuje kompresi.

Vícero druhů dotazu v jedné zprávě

Specifikace protokolu DNS umožňuje se dotazovat na vícero záznamu pro dané doménové jméno. Například požadovat záznam typu A a AAAA zároveň pro doménu seznam.cz. Na základě odpovědi zvolit protokol síťové vrstvy v případě, že koncová stanice podporuje dual stack (IPv4 a IPv6 protokol naráz). Díky dotazování různých resolverů bylo vypozorováno, že resolvery odpovídají pouze na první požadavek v části dotazu nebo dotaz úplně ignorují a neodpoví vůbec.

Mezi nejpoužívanější DNS servery se řadí implementace Unbound, Bind a Knot. Knot resolver je implementace DNS serveru od společnosti CZ.NIC.

Lze předpokládat, že jej využívají pro vlastní veřejný resolver (193.17.47.1). Pokud je zaslán dotaz na vícero typů záznamů pomocí protokolu UDP, není odpovězeno vůbec a dochází k vypršení časového limitu zprávy. Při využití protokolu TCP je odpověď pouze na první dotaz a druhý je ignorován.

Při dotazování veřejných resolverů Googlu, Cloudflare je chování jiné, ale pro oba stejné. Odpovídají vždy pouze na první dotaz a to nezávisle na transportním protokolu. Na stránkách Cloudflare se lze dočíst, že využívají také Knot resolver. Jedná se tedy o nastavení konfigurace, které ovlivňuje výsledek překladu. Stejně jako resolver od společnosti CZ.NIC se chová i security gateway od společnosti Checkpoint, která podporuje funkci resolveru v lokální síti.[38]

Důvod proč překládat doménové jméno lze vidět v ochraně klientů. Zejména v případě útoku na zahlcení, kdy DNS zprávy mohou být využity pro amplifikační útok, kdy útočník generuje krátké dotazy na resolvery s podvrhnutou adresou oběti jakožto odesílatele. Vhodným nastavením lze předejít několikanásobným zesílením, když právě resolvery budou odpovídat pouze na první.

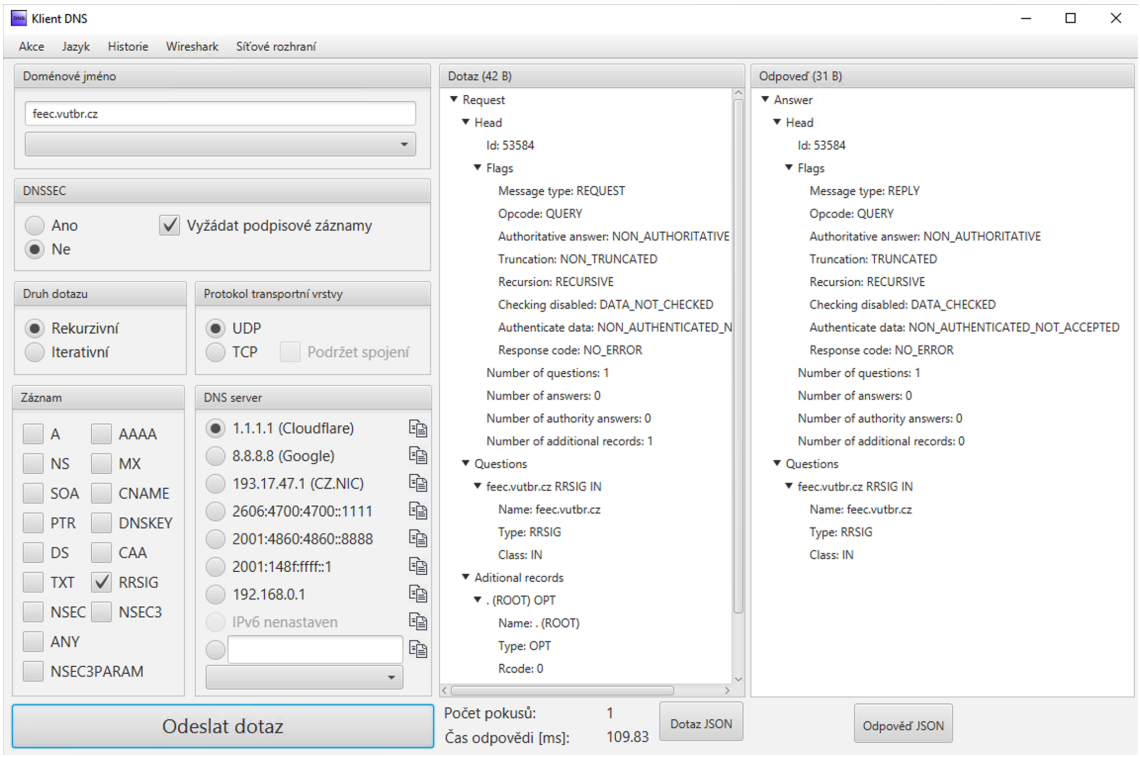
Odlišnost chování při překročení maximální délky zprávy

Další odlišnosti lze pozorovat při odpovědi, která je větší než povolená velikost zprávy pro transportní protokol UDP. Dle doporučení může maximální velikost zprávy být 1232 bajtů. Větší odpovědi mohou být po cestě fragmentovány na úrovni IP paketů. Fragmentace IPv4 paketů je problematická, ale možná a protokol ji podporuje. U IPv6 paketů při přenosu paketů není fragmentace možná, proto se DNS protokol snaží omezit velikost zpráv, aby situace vůbec nenastala.[39]

Při dotazu na doménové jméno feec.vutbr.cz pomocí rekurzivního dotazu s žádostí na záznamy typu RRSIG se zapnutou možností podpory DNSSECu nastává právě výše zmíněna situace, kdy odpověď je větší než doporučených 1232 bajtů. Resolver rozdělení zprávy signalizuje v hlavičce pomocí bitu Truncation nastavený na hodnotu 1. Klient by se měl dotázat pomocí protokolu TCP, který má teoretické omezení velikosti až 65536 bajtů na zprávu. Tato hodnota je dána maximální hodnotou, kterou lze vyjádřit prvními 2 bajty před zprávou DNS. První dva bajty vyjadřující velikost zprávy jsou pouze při využití transportního protokolu TCP. U UDP jsou první dva bajty již QID zprávy DNS. Sledované chování bylo testováno na doméně feec.vutbr.cz a záznamu typu RRSIG s vyžádáním podpisových záznamů.

Odlišnost chování při signalizaci zprávy lze pozorovat u protokolu UDP. Resolver Cloudflare odpovídá pouze zprávou s částmi z dotazu. Úplně ale chybí záznam OPT, který signalizuje, že klient požaduje DNSSEC. V doporučení definuje, že záznam

OPT by měl být součástí odpovědi, když je součástí žádosti. Chování lze chápat jako snahu o co nejmenší odpověď, tak aby resolver nemohl být zneužit pro útok zesílením. Velikost dotazu má pro příklad s doménou feec.vutbr.cz 42 bajtů, odpověď poté 31 bajtů. Situace je zobrazena na obrázku 3.15



Obr. 3.15: Netradiční chování resolveru Cloudflare při dotazu nad doporučenou velikost 1232 bajtů.

DNS server Google se chová podobně jen s rozdílem, že kopíruje záznam OPT do odpovědi. Výsledná odpověď má stejně jako dotaz 42 bajtů. Odlišné chování představuje DNS server CZ.NIC, který součástí zprávy zahrnuje i záznamy tak, aby velikost zprávy byla vždy v souladu s doporučením. V odpovědi se nachází 3 záznamy typu RRSIG, což vede k výsledné velikosti 1201 bajtů.

Při dotazu protokolem TCP je možné získat veškeré záznamy typu RRSIG pro doménu feec.vutbr.cz. Jedná se o 9 záznamů a výsledná zpráva má 3009 bajtů. Jelikož první dva bajty identifikují velikost zprávy vychází pro přenos protokolem UDP 3007 bajtů, což není dle doporučení možné. Každý resolver přistupuje k rozdělení odpovědi rozdílně.

Odpověď na dotaz ANY

Záznam ANY není uložen v databázi zóny. Využití má při cachování dat na resolveru. Pomocí dotazu lze získat veškeré cahované záznamy

pro dotazovanou doménu. Ačkoliv se zdá, že dotaz typu ANY je vhodným nástrojem pro získaní dat o doméně, představuje možné zneužití při amplifikačním útoku. Útočník zjistí, které kombinace dotazů generují dostatečně velké zprávy a nedochází k rozdělení zpráv tak, aby výsledná zpráva měla co možná největší velikost. Výslednou kombinaci si poznamená a počká pár minut na vyprázdnění cache resolverů. Následně se dotazuje na záznamy, které pro útok potřebuje. Ty se uloží v cache paměti resolverů. Útočník má veškeré prerekvizity k provedení DDOS útoku. Stačí pouze aktuálně zasílat dotazy typu ANY se zdrojovou IP adresou oběti, která je pak zahlcena nevyžádanými pakety. Ačkoliv doména, která nemá mnoho záznamů stejného typu, i tak může posloužit právě k útoku, protože útočník si velikost zpráv (i zesílení) může určit sám.

Resolvery by neměly na dotaz ANY přes transportní protokol UDP vůbec odpovídat z důvodu mitigace výše zmíněného scénáře. Rozdíl mezi resolvery je ale reakce na daný dotaz. Testovány byly resolvery pomocí doménového jména vutbr.cz a nejdříve bylo proveden dotaz pomocí transportního protokolu UDP na záznamy A a AAAA, aby resolver měl možnost odpověď cachovat a poté odeslat dotaz typu ANY na stejný resolver nejdříve pomocí protokolu UDP následně TCP.

Resolver Cloudflare odpovídá s nastaveným atributem v hlavičce na hodnotu, která reprezentuje, že funkce není implementována. Stejná odpověď je i pro protokol TCP. Lze vyvozovat, že funkce není vůbec povolena na tomto resolveru. Resolver Google odpovídá s nastaveným bitem v hlavičce na hodnotu rozdělení zprávy a to nezávisle na velikosti zprávy. Test byl proveden i na doméně s jedním záznamem, i tak záznam nebyl součástí odpovědi. Dotaz protokolem TCP je přeložen správně a vrací veškeré cachované záznamy. Resolver CZ.NIC v obou případech odpovídá nastavenou hodnotou chyba serveru. Domácí resolver se chová stejně jako resolver Googlu a odpovídá při dotazu protokolem TCP.

3.4 Porovnání Dig, Nslookup a vytvořené aplikace

Nástroj Nslookup

Základním nástrojem ze všech tří je Nslookup. Jedná se nástroj do příkazové řádky pro nenáročné uživatele. Umožňuje nastavit resolver, doménu, transportní protokol, port resolveru a podporuje základní DNS záznamy mezi než patří například záznam A, AAAA, MX, PTR, NS. Pro dotaz na záznam PTR je potřeba využít příznak -x a poté zadat IP adresu. Nástroj se postará o správně formátování dotazu. Příklad využití nástroje je zobrazen ve výpisu 3.6.

Výhody Nslookup

- Multiplatformní nástroj.
- Nenáročný na konfiguraci parametrů.
- Možnost využití nástroje při skriptování.
- Lze měnit cílový port.

Slabé stránky nástroje Nslookup

- Neumí DNSSEC.
- Nelze zaslat požadavek s OPT záznamem.
- Omezený počet záznamů.
- Nerozpozná IDN domény.
- Nelze získat odpověď jako Json objekt.
- Nelze vyčíst čas transakce.
- Nelze získat velikost zpráv.
- Nelze získat informace o dotazu.
- Nelze zaslat v dotazu více druhů záznamů.
- Nelze získat Wireshark filtr pro daný DNS server.
- Nelze podržet TCP spojení pro více dotazů v jedné relaci.
- Nelze zvolit zdrojovou adresu potažmo zdrojové rozhraní.

Výpis 3.6: Příklad dotazu nástrojem Nslookup na doménu feec.vutbr.cz.

```
C:\> nslookup -type=A feec.vutbr.cz.
Server:  my.firewall
Address:  192.168.0.1

Non-authoritative answer:
Name:    feec.vutbr.cz
Address:  147.229.71.30
```

3.4.1 Nástroj Dig

Pokročilým nástrojem je Dig, který je stejně jako Nslookup program do příkazového řádku. Umožňuje všechny možnosti nastavení jako Nslookup a přidává i další rozšíření. Jedná se o nástroj využívaný hlavně správcemi zóny, aby ověřili správné nastavení DNS serveru. Příklad využití nástroje Dig je zobrazen ve výpisu 3.7.

Výpis 3.7: Příklad dotazu nástrojem Dig na doménu feec.vutbr.cz.

```
$ martinbiolek$ dig A feec.vutbr.cz

; <<>> DiG 9.10.6 <<>> A feec.vutbr.cz
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 34525
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0,
ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;feec.vutbr.cz.      IN   A

;; ANSWER SECTION:
feec.vutbr.cz.     14395 IN   A 147.229.71.30

;; Query time: 13 msec
;; SERVER: 192.168.0.1#53(192.168.0.1)
;; WHEN: Tue Mar 30 23:58:39 CEST 2021
;; MSG SIZE  rcvd: 58
```

Výhody nástroje Dig

- Podporuje DNSSEC.
- Multiplatformní nástroj.
- Velké množství přepínačů a parametrů, čím lze dotaz a odpověď ovlivnit.
- Podporuje veškeré známé záznamy.
- Možnost využít nástroj při skriptování.
- Zobrazuje čas transakce.
- Lze změnit cílový port.
- Lze zaslat dotaz s OPT záznamem.

Slabé stránky nástroje Dig

- Nelze zaslat v dotazu více druhů záznamů.
- Nerozpozná IDN domény.
- Nelze získat velikost zpráv.
- Nelze získat informace o dotazu.
- Nelze získat odpověď jako Json objekt.

- U dotazu PTR neprovede reverzní zápis IP adresy za uživatele.
- Nelze získat Wireshark filtr pro daný DNS server.
- Nelze podržet TCP spojení pro více dotazů v jedné relaci.
- Nelze zvolit zdrojovou adresu potažmo zdrojové rozhraní.

3.4.2 Vytvořená grafická aplikace

Vytvořená grafická aplikace přináší další vlastnosti, které nástroje Dig a Nslookup postrádají. Jako jediná má grafické rozhraní. Pro svůj běh potřebuje prostředí s Java 11+ a knihovnou JavaFX verze 15, která je součástí složky stejně jako i spustitelný .exe soubor. Výstup aplikace pro stejný záznam je zobrazen na obrázku 3.16.

Výhody vytvořené grafické aplikace

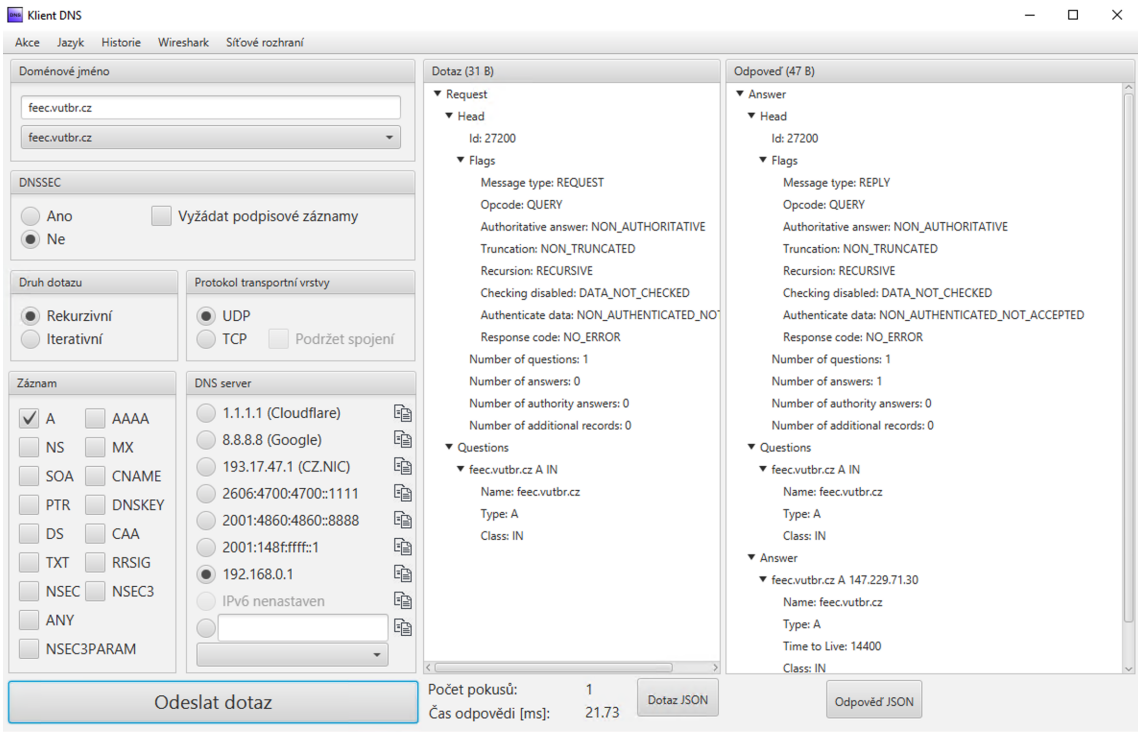
- Grafické rozhraní.
- Zobrazuje čas transakce.
- Možnost podržet TCP spojení a v jedné relaci zaslat více dotazů.
- Zobrazuje počet bajtů zpráv.
- Zobrazuje informace jak o odpovědi, tak i o dotazu.
- Podporuje DNSSEC.
- Lze zaslat dotaz s OPT záznamem.
- Rozpozná IDN domény a provede konverzi do Punycodu za uživatele.
- Pro PTR záznam lze zadat jak validní reverzní doménu, tak pouze IP adresu.
- Lze získat Wireshark filtr pro daný DNS server.
- Lze zaslat v dotazu více druhů záznamů.
- Umožňuje volbu odchozího rozhraní potažmo zdrojovou adresu.

Slabé stránky vytvořené grafické aplikace

- Aktuálně pouze na Microsoft Windows platformu.
- Nelze nástroj využít pro skriptování.
- Nepodporuje veškeré známé záznamy. Aktuálně 16 nejčastějších záznamů.
- Nelze volit port DNS severu.
- Nutnost JAVA 11+.
- Nutnost knihovny JavaFX verze 15.

3.5 Implementace Multicast DNS

Aplikace podporuje protokol mDNS. Jedná se o multicastovou variantu DNS, která zachovává stejný formát zpráv a je tedy s DNS zpětně kompatibilní. Rozdíl



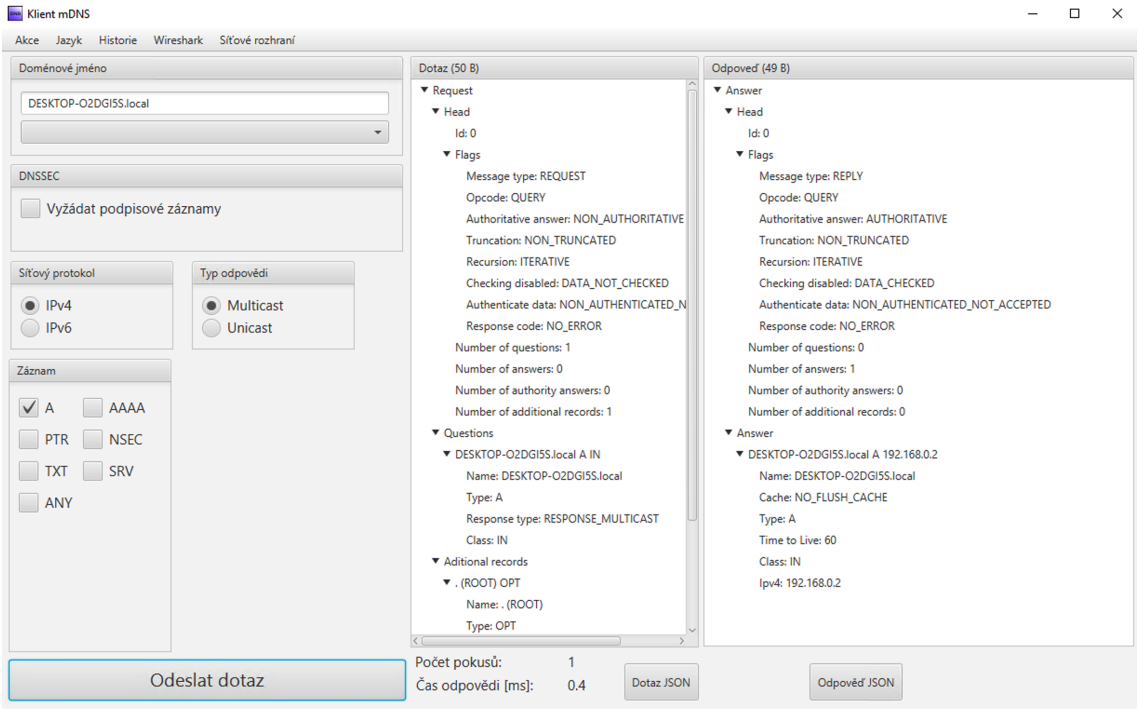
Obr. 3.16: Příklad dotazu vytvořenou aplikací na doménu feec.vutbr.cz.

lze najít v kódování doménových jmen, kde podporuje UTF-8 a není omezen na ASCII znaky. Zprávy jsou zasílány na multicastovou adresu 224.0.0.251 pro IPv4 a ff02::fb adresu pro IPv6. Pro multicast lze využit pouze UDP transportní protokol, mDNS má rezervovaný port 5353.

Při testování bylo pozorováno, že stanice nerozlišují kapitálky v doménovém jménu, ačkoliv pro UTF-8 kódování změna velikosti písmene vede k jiné bajtové reprezentaci doménového jména. Příkladem může být dotaz na doménové jméno Pc.local, kde záměrně bylo C zapsáno malým písmenem. Stanice neodpoví na původní dotaz Pc.local, ale na doménu PC.local. Situace, kdy takto bylo odpovězeno na doménu, která má rozlišený bajtový zápis, je uživatel informován o nestandardním překladu pomocí informačního okna, že domény z dotazu a v odpovědi se neshodují.

Stanice mDNS do zpráv přidávají nejen záznam, který byl zaslán v požadavku, ale veškeré své záznamy, které se váží s danou stanicí. Typickým příkladem může být dotaz na záznam A. V odpovědi lze nalézt záznam A, AAAA (popřípadě NSEC) a to i v případě, že lokální síť je pouze IPv4, protože stanice na pozadí využívají IPv6 pro link-local komunikaci. Adresu pro link-local si každá stanice může vytvořit sama na základě své fyzické adresy. Lze předpokládat, že fyzická adresa je v síti jedinečná a proto i link-local adresa bude jedinečná také. Příklad překladu doménového jména v lokální síti lze vidět na obrázku 3.17.

Grafické rozhraní má kontrolér v podobě třídy MDNSCOntroller.java v balíčku ui a je definováno v souboru MDNS.fmxl v balíčku fxml. Využívá pro zasílání stejné třídy jako DNS jen má jinak definované konstruktory, popřípadě metody pro parsování zprávy. Záznamy, které obsahují doménové jméno umožňují dekódování UTF-8 doménových jmen, což je zpětně kompatibilní s ASCII a na protokol DNS nemá vliv.



Obr. 3.17: Příklad překladu pomocí protokolu mDNS v lokální síti pro doménové jméno DESKTOP-O2DGI5S.local.

3.5.1 Rozdíly ve významu zpráv mDNS a DNS

Formát zpráv DNS dává omezenou možnost měnit význam jednotlivých položek ve zprávě. Dva atributy mDNS jsou uloženy v přenášených hodnotách, které u DNS nejsou.

U zprávy dotazu má každá položka 3 atributy: doménové jméno, typ dotazu (požadavek na záznam) a třídu dotazu. Aktuálně využívaná třída dotazu je pouze třída IN, která má hodnotu 1, ačkoliv je reprezentována 16 bity. MDNS zavádí vyjmutí nejvýše významného bitu z reprezentace třídy a nastavením hodnoty true dává stanice najevo, že očekává unicastovou odpověď. Hodnota bitu false značí odpověď multicastovou. Všechny stanice na síti si danou odpověď mohou uložit.

Stejně jako je třída v dotazu, tak je i jako atribut v odpovědi. Zde také nejvýznamnější bit reprezentuje jiný atribut, který v DNS není. Bitem nastaveným

na true dává odpovídající stanice najevo, že záznamy doposud cachované pro danou doménu mají být smazány a nahrazeny právě přeloženými. Bit se označuje jako Cache flush a díky tomu může stanice na síti velice rychle měnit své záznamy, protože není potřeba čekat na dotaz, ale posílat odpovědi průběžně do sítě. Tím se zaručí, že síť si vede přehled o aktuálně zapojených stanicích a jejich adresách, jmen a službách.

3.5.2 mDNS a unicastové zprávy

Ačkoliv klient umožňuje zasílat dotaz, kdy očekává unicastovou odpověď, zobrazí se ve všech případech chyba. Chyba má hlášku: „Nelze se připojit na lokální UDP port 5353, protože jej již využívá systémová služba.“ Na portu UDP 5353 naslouchá služba operačního systému, která zpracovává přijaté zprávy tak, aby s výsledky operační systém mohl pracovat. Nelze se tedy klientskou aplikaci připojit na tento port a získat odpověď pro zobrazení v grafickém rozhraní. Ačkoliv funkce není kompletní, lze v programu Wireshark sledovat průběh dotazování, a tak i částeční plní svůj účel, protože operační systém si může aktualizovat nově proložené údaje. Příklad zachycené přeložené unicastové odpovědi programem Wireshark je zobrazen na obrázku

Source	Destination	Protocol	Length	Info
192.168.0.2	224.0.0.251	MDNS	86	Standard query 0x0000 A macMartin.local, "QU" question OPT
192.168.0.166	192.168.0.2	MDNS	113	Standard query response 0x0000 A, cache flush 192.168.0.166 AAAA, cache flush fe80::1cd0:cab4:2a52:3975

Obr. 3.18: Zachycená komunikace na unicastový UDP port 5353 na základě požadavku z klientské aplikace.

3.5.3 Testování aplikace pro protokol mDNS

Testování doménových jmen

- **macMartin.local** – výsledek: OK
- **DESKTOP-O2DGI5S.local** – výsledek: OK
- **vutbr.cz** – výsledek: OK (Chybová hláška)
- **192.168.0.2** (PTR záznam) – výsledek: OK
- **2.0.168.192.in-addr.arpa** (PTR záznam) – výsledek: OK

Testování typu záznamů

- **A** (doména macMartin.local) – výsledek: OK
- **AAAA** (doména macMartin.local) – výsledek: OK
- **PTR** (192.168.0.166) – výsledek: OK

- **PTR** (192.168.0.166.in-addr.arpa) – výsledek: OK
- **NSEC** (doména macMartin.local) – výsledek: OK
- **TXT** (doména miracast.local) – výsledek: OK
- **SRV** (doména miracast.local) – výsledek: OK
- **ANY** (doména macMartin.local) – výsledek: OK
- **A a AAAA** (doména macMartin.local) – výsledek: OK
- **ANY a AAAA** (doména macMartin.local) – výsledek: OK

Testování typu odpovědi

- **Multicast** – výsledek: OK
- **Unicast** – výsledek: OK (Chybová hláška – nelze přijmout zprávu)

Testování DNSSEC podpisových záznamů

- **Vypnuté vyžádání podpisových záznamů pro DNSSEC** – výsledek: OK
- **Zapnuté vyžádání podpisových záznamů pro DNSSEC** – výsledek: OK (Stanice nemají podpisové záznamy.)

Testování typu IP protokolu

- **IPv4 protokol** – výsledek: OK
- **IPv6 protokol** – výsledek: OK

Testování odchozího rozhraní

- **Ethernetové rozhraní s IP adresou** – výsledek: OK
- **Wifi rozhraní s IP adresou** – výsledek: OK
- **Virtuální rozhraní s IP adresou** – výsledek: OK
- **Ethernetové rozhraní bez IP adresy** – výsledek: OK (chybová hláška)
- **Wifi rozhraní bez IP adresy** – výsledek: OK (chybová hláška)
- **Virtuální rozhraní bez IP adresy** – výsledek: OK (chybová hláška)

3.5.4 Spoofování doménového jména pomocí mDNS protokolu

Protokol mDNS přináší možnost se vydávat za stanice na lokální síti (spoofing). Do multicastové skupiny může přispívat každá stanice a většina zpráv je bez DNSSEC podpisů. Je tak velice jednoduché pro útočníka zasílat do multicastové skupiny v pravidelných intervalech zprávy o své stanici, která se vydává za stanici s určitým doménovým jménem. Záleží pak jen na operačním systému, jaký mechanismus pro překlad využije. Zda přečte cachovanou hodnotu, nebo zašle nový požadavek do sítě. Pro druhý případ existuje

i zde možnost, jak útočník může změnit výsledný překlad. Stačí, aby jeho odpověď byla rychlejší než odpověď oběti a po odpovědi oběti do sítě vyšle znovu novou odpověď s bitem označující k vymazání dosavadních cachovaných záznamů s tímto doménovým jménem. Útočník může nejen spoofovat doménová jména, ale i služby. Příkladem může být vydávání se za tiskárnu na lokální síti a dokumenty k tisku číst a ukládat. Útočník nemusí využívat nikterak náročné nástroje, stačí mu pouze Python skript, který je veřejně dostupný[1].[40]

Efektivní řešení spočívá v přidání DNSSEC podpisových záznamů do každé odpovědi, což ale vede k nutnosti stanic udržovat DS záznam pro lokální doménu. Toto řešení navíc vůbec nepředpokládá změny IP adres v lokální síti a to i nutnost změny podpisu záznamu, což v lokálních sítích s DHCP serverem je časté a při cestování mobilních stanic mezi sítěmi i neřešitelné. Dnes navíc většina operačních systému validaci podpisů ani neumožňuje, protože tu provádí resolver. Doporučení pro mitigaci útoku je mDNS ve velkých sítích úplně vypnout a nevyužívat jej. V domácích sítích převažují benefity využívání protokolu nad rizikem. Ekosystém produktů společnosti Apple na mDNS sází a pro uživatele přináší velké výhody.

3.6 Implementace DNS over HTTPS

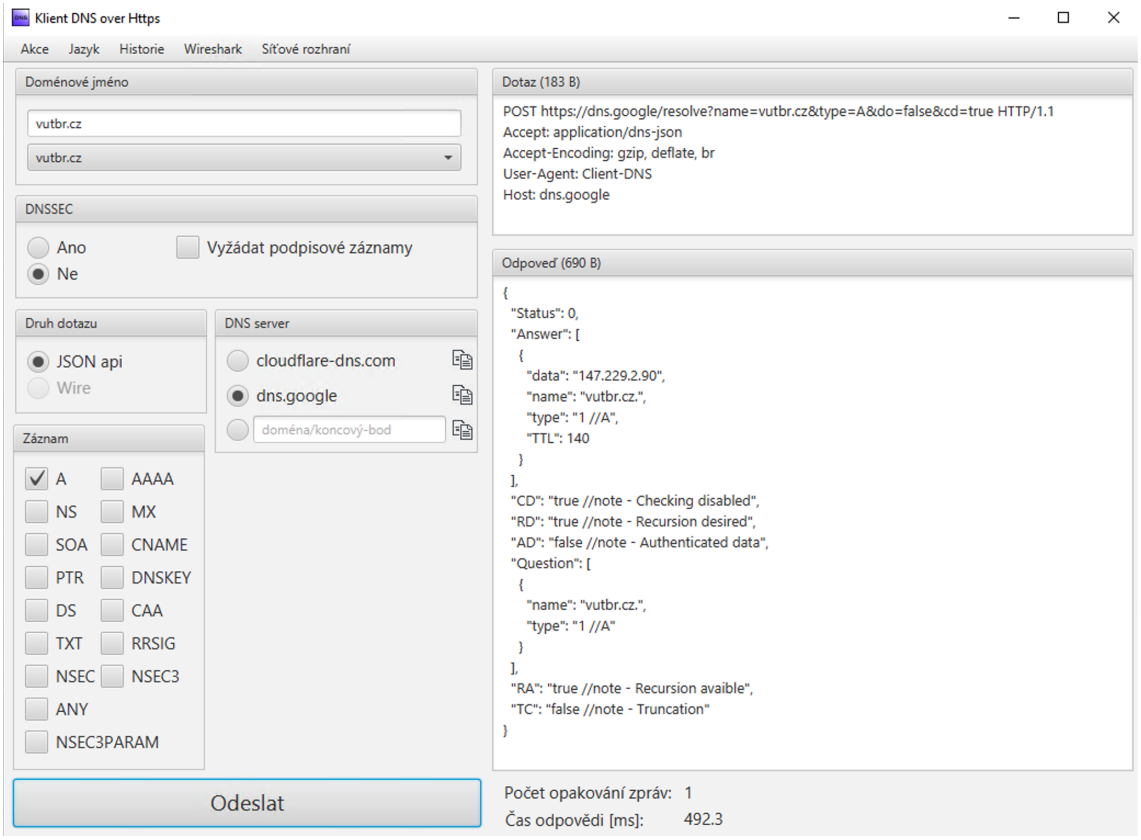
Po dohodě s vedoucím práce jako třetí protokol byl implementován DNS over HTTPS (DoH) namísto Link-Local Multicast Name Resolution protokolu, který byl specifikován v zadání práce. Pro implementaci byla využita knihovna Apache HTTPClient ve verzi 4.5.13. Knihovna slouží k vytvoření a zpracování HTTPS požadavku.

DoH je jeden z šifrovaných DNS protokolů. Staví již na zaběhlé infrastruktuře webových služeb a distribuci certifikátů na doménové jméno. Stanice tak může porovnat, zda certifikát serveru byl vydán důvěrnou kořenovou autoritou definovanou v operačním systému. DoH má dvě varianty Json Application Programming Interface (API) a Wire formát. Wire formát je velice podobný klasickému DNS, kdy bajty DNS zprávy jsou transformovány do hexadecimálních znaků, které jsou pak v tělu HTTPS zprávy. Nevýhodou tohoto formátu je velmi složité hledání chyb při implementaci, jelikož celá komunikace je šifrovaná a nelze nástrojem Wireshark hledat chyby ve zprávě.

Aplikace podporuje formát Json API. Aktuálně jej plně podporují resolvery Cloudflare a Google. Jedná se o transformování DNS požadavku do HTTPS žádosti. Parametry jsou součástí Uniform Resource Locator (URL). Odpověď

[1] `https://www.gnucitizen.org/files/2008/01/mdns.py`

na dotaz je pak ve formátu Json v tělu HTTPS zprávy. Příklad dotazu pomocí aplikace je obrázku 3.19.



Obr. 3.19: Příklad překladu doménového jména vutbr.cz pomocí vytvořené aplikace

3.6.1 Formát odpovědi

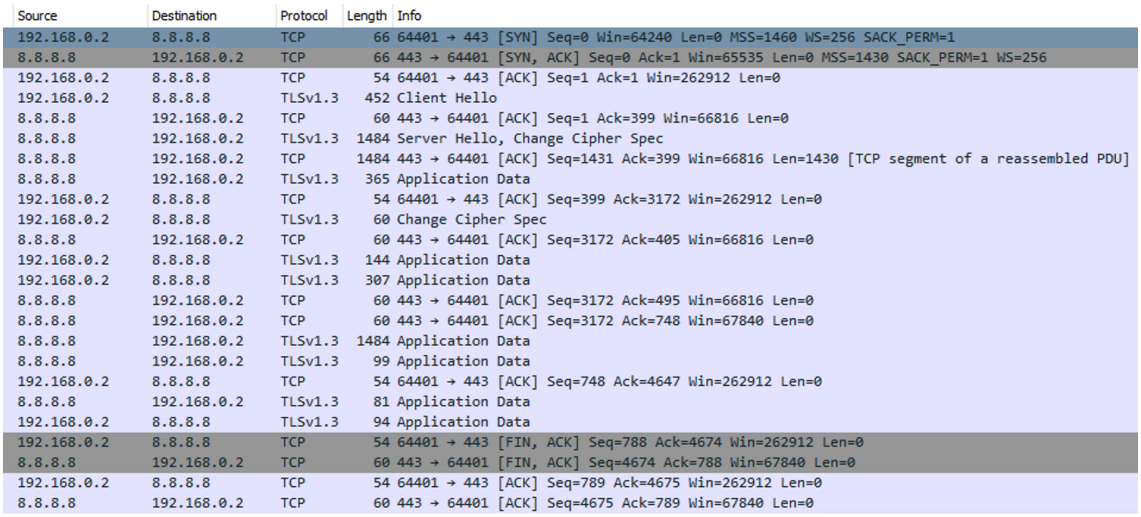
Při překladu požadavku jsou v Json objektu položky, které potřebují klíč, aby bylo zřejmé, co reprezentují. Příkladem může být AD. Reprezentuje klíč pro hodnotu „Authenticated data“ (zda data odpovědi resolver ověřil pomocí DNSSECu). Pro uživatelsky přívětivější dojem program zobrazuje za hodnotou i poznámku, kde klíč rozepisuje.

Podobně i záznam má položku type, kde číselná hodnota reprezentuje typ záznamu. Program do poznámky vkládá název záznamu, aby uživateli bylo hned jasné, o jaký typ záznamu se jedná.

3.6.2 Průběh komunikace

Z obrázku 3.19 lze vidět, že čas pro získání odpovědi je výrazně vyšší než u DNS. Je to dáno nutností sestavení spojení, ustanovení šifrovacích klíčů a samotným

šifrováním. Průběh komunikace je zachycen programem Wireshark jak jde vidět na obrázku 3.20. Oproti DNS využívající protokol UDP je aktuálně zasláno 24 paketů. Řešením, které má vlastnosti DNS a zachovává šifrování, by mohl být protokol DNS over QUIC. Ve své variantě s 0-round time tripem bude pro překlad nutné zaslat pouze 2 pakety. Aktuálně si uživatel musí vybrat, zda preferuje soukromí při překladu zpráv, nebo rychlý překlad.



Source	Destination	Protocol	Length	Info
192.168.0.2	8.8.8.8	TCP	66	64401 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
8.8.8.8	192.168.0.2	TCP	66	443 → 64401 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1430 SACK_PERM=1 WS=256
192.168.0.2	8.8.8.8	TCP	54	64401 → 443 [ACK] Seq=1 Ack=1 Win=262912 Len=0
192.168.0.2	8.8.8.8	TLSv1.3	452	Client Hello
8.8.8.8	192.168.0.2	TCP	60	443 → 64401 [ACK] Seq=1 Ack=399 Win=66816 Len=0
8.8.8.8	192.168.0.2	TLSv1.3	1484	Server Hello, Change Cipher Spec
8.8.8.8	192.168.0.2	TCP	1484	443 → 64401 [ACK] Seq=1431 Ack=399 Win=66816 Len=1430 [TCP segment of a reassembled PDU]
8.8.8.8	192.168.0.2	TLSv1.3	365	Application Data
192.168.0.2	8.8.8.8	TCP	54	64401 → 443 [ACK] Seq=399 Ack=3172 Win=262912 Len=0
192.168.0.2	8.8.8.8	TLSv1.3	60	Change Cipher Spec
8.8.8.8	192.168.0.2	TCP	60	443 → 64401 [ACK] Seq=3172 Ack=405 Win=66816 Len=0
192.168.0.2	8.8.8.8	TLSv1.3	144	Application Data
192.168.0.2	8.8.8.8	TLSv1.3	307	Application Data
8.8.8.8	192.168.0.2	TCP	60	443 → 64401 [ACK] Seq=3172 Ack=495 Win=66816 Len=0
8.8.8.8	192.168.0.2	TCP	60	443 → 64401 [ACK] Seq=3172 Ack=748 Win=67840 Len=0
8.8.8.8	192.168.0.2	TLSv1.3	1484	Application Data
8.8.8.8	192.168.0.2	TLSv1.3	99	Application Data
192.168.0.2	8.8.8.8	TCP	54	64401 → 443 [ACK] Seq=748 Ack=4647 Win=262912 Len=0
8.8.8.8	192.168.0.2	TLSv1.3	81	Application Data
192.168.0.2	8.8.8.8	TLSv1.3	94	Application Data
192.168.0.2	8.8.8.8	TCP	54	64401 → 443 [FIN, ACK] Seq=788 Ack=4674 Win=262912 Len=0
8.8.8.8	192.168.0.2	TCP	60	443 → 64401 [FIN, ACK] Seq=4674 Ack=788 Win=67840 Len=0
192.168.0.2	8.8.8.8	TCP	54	64401 → 443 [ACK] Seq=789 Ack=4675 Win=262912 Len=0
8.8.8.8	192.168.0.2	TCP	60	443 → 64401 [ACK] Seq=4675 Ack=789 Win=67840 Len=0

Obr. 3.20: Průběh komunikace pro překlad doménového jména vutbr.cz využitím protokolu DoH varianta Json API.

3.6.3 Vlastnosti aplikace u protokolu DoH

Aplikace nabízí historii stejnou jako u DNS. Pokud uživatel zadá požadavek na doménu v části DoH, uvidí doménu i v nabídce historie u DNS okna. Aplikace zobrazuje pouze odpovědi, kde kód HTTP odpovědi je 200 (OK). Ostatní kódy automaticky vytvoří chybu, která je ohlášena pomocí chybové hlášky i s příslušným kódem.

Stejně jako u DNS i u DoH je možnost získat Wireshark filtr. Při využívání DoH vznikají problémy, že pro doménu Cloudflare-dns.com existují dva A záznamy. Oba jsou vráceny v různém pořadí. V jednom případě je první záznam s adresou 104.16.248.249 a ve druhém s adresou 104.16.249.249. To vede k možnosti, že Wireshark filtr pro daný překlad nebude mít správnou adresu, na kterou bude požadavek odeslán. Pro co nejlepší možný výsledek je hodnota filtru aktualizována při kliknutí na ikonu pro kopírování i při odeslání. Existuje ale možnost, že ve Wireshark filtru bude adresa, která pro překlad nebyla využita.

3.6.4 Testování aplikace pro protokol DoH

Testování doménových jmen

- **vutbr.cz** – výsledek: OK
- **vutbr.cz.** – výsledek: OK
- **cz** – výsledek: OK
- **cz.** – výsledek: OK
- **8.8.4.4** (PTR záznam) – výsledek: OK
- **4.4.8.8.in-addr.arpa** (PTR záznam) – výsledek: OK
- **2606:4700:4700::1111** (PTR záznam) – výsledek: OK
- **1.1.1.1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.7.4.0.0.7.4.6.0.6.2.ip6.arpa** (PTR záznam) – výsledek: OK
- **háčkyčárky.cz** – výsledek: OK
- **háčkyčárky.cz.** – výsledek: OK
- **xn–hkyrky-ptac70bc.cz** – výsledek: OK
- **Nevalidní doména (a.)** – výsledek: OK (Chybová hláška)

Testování druhu záznamu

- **A** (doména vutbr.cz) – výsledek: OK
- **AAAA** (doména nic.cz) – výsledek: OK
- **NS** (doména vutbr.cz) – výsledek: OK
- **MX** (doména vutbr.cz) – výsledek: OK
- **SOA** (doména vutbr.cz) – výsledek: OK
- **CNAME** (doména btcpay.biolek.net) – výsledek: OK
- **PTR** (doména 8.8.4.4) – výsledek: OK
- **DNSKEY** (doména nic.cz) – výsledek: OK
- **DS** (doména vutbr.cz) – výsledek: OK
- **CAA** (doména vutbr.cz) – výsledek: OK
- **TXT** (doména vutbr.cz) – výsledek: OK
- **RRSIG** (doména nic.cz) – výsledek: OK
- **NSEC** (doména vutbr.cz) – výsledek: OK
- **NSEC3** (doména cz) – výsledek: OK
- **NSEC3PARAM** (doména cz) – výsledek: OK
- **ANY** (doména cz) – výsledek: OK
- **A + AAAA** (vutbr.cz) – výsledek: OK (přeložen pouze první dotaz u Cloudflare, u Google resolveru odpověď typu 400)
- **A + AAAA + MX** (doména vutbr.cz)– výsledek: OK (přeložen pouze první dotaz pro Cloudflare, u Google resolveru odpověď typu 400)

- **A + PTR** (doména vutbr.cz) – výsledek: OK (Chybová hláška, PTR nelze kombinovat s jinými dotazy)

Testování DNSSEC

- **Vypnutý DNSSEC** – výsledek: OK
- **Vyžádaní ověření resolverem** – výsledek: OK
- **Vyžádání si podpisových záznamů** – výsledek: OK

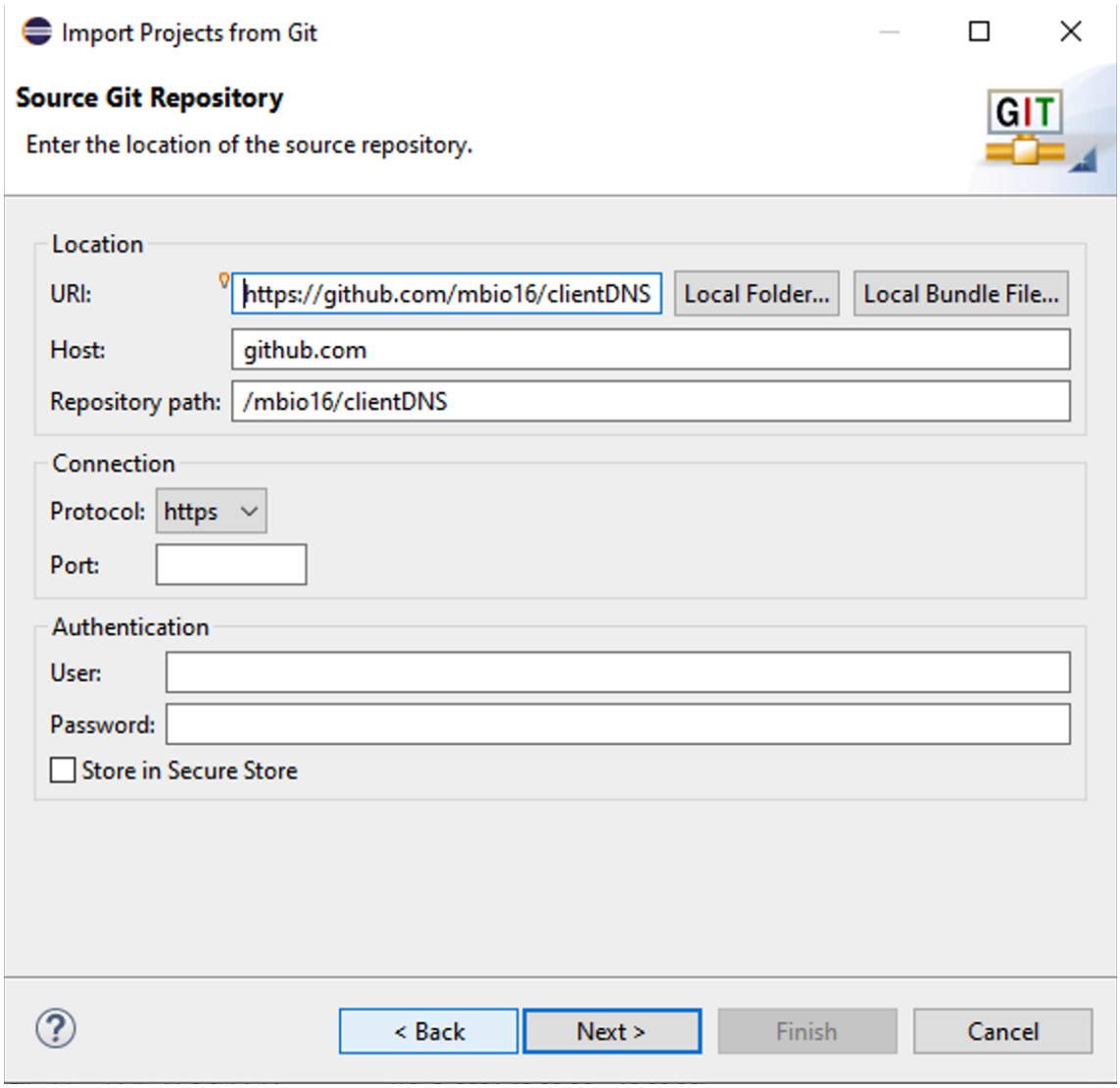
Testování DNS serveru

- **cloudflare-dns.com** – výsledek: OK
- **dns.google** – výsledek: OK
- **8.8.8.8/resolve** – výsledek: OK
- **8.8.4.4/resolve** – výsledek: OK
- **104.16.248.249/resolve** – výsledek: OK (Chybová hláška)

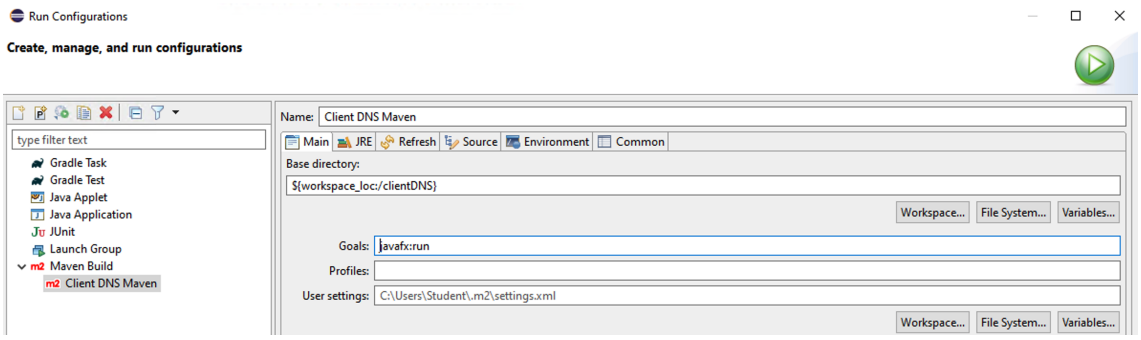
4 Import projektu do vývojového prostředí Eclipse

4.1 Maven projekt aplikace

Projekt z repozitáře na Githubu lze importovat pomocí Git implementace přímo z Eclipse. Není potřeba projekt stahovat před spuštěním Eclipsu samostatně. První co uživatel musí udělat, je zvolit Import projects -> Git -> Projects from Git(with smart import) -> Clone URI. Otevře se menu, kde je potřeba do položky URI vložit `https://github.com/mbio16/clientDNS`, pak kliknout na tlačítko Next jako je zobrazeno na obrázku 4.1. V dalším kroku stačí vybrat



Obr. 4.1: Okno pro importování projektu z Github repozitáře.



Obr. 4.2: Maven konfigurace pro spuštění aplikace.

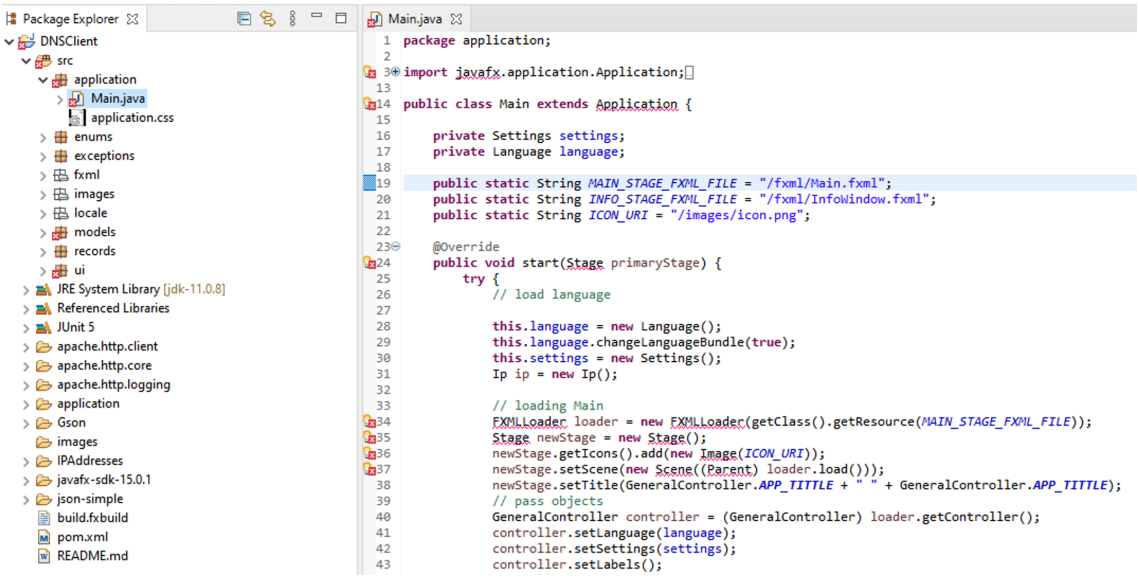
pouze main branch, která je aktuální. Následuje vybrání domovského adresáře v souborovém systému. Lze vybrat vlastní složku, nebo pokračovat s výchozí. Po kliknutí na Next stačí počkat na stažení projektu a dokončení akce tlačítkem Finish.

Malou chvilku bude projekt hlásit chyby, ale ty po pár vteřinách zmizí, protože se stáhnou knihovny definované v pom.xml souboru. Pro spuštění projektu je možné kliknout na položku Run, která se nachází v horní menu Eclips a zvolit Run Configuration. Dvojklikem na Maven Build se vytvoří nová konfigurace, kde je potřeba vyplnit Base directory (lze vybrat aktuální pomocí tlačítka Workspace), a do Goals vyplnit „javafx:run“. V horní části konfigurace je dobré vyplnit položku Name, aby konfiguraci bylo lehké rozpoznat od jiných projektů. Příklad je zobrazen na obrázku 4.2. Po kliknutí na Apply a Run se spustí aplikace. V konzoli, v dolní části Eclipse, lze sledovat logy z aplikace, které ve finální aplikaci .exe nejsou viditelné.

4.2 Import projektu s knihovnami pro export do spustitelného souboru

Od Java 8 není knihovna JavaFX součástí jádra knihoven, které se instalují současně při instalaci Javy na stanici. Pro export do spustitelného souboru je nutné místo Maven projektu, který je popsaný výše využít projekt s knihovnami jakožto součástí projektu. V elektronických přílohách je archiv ClientDNS-s-knihovnami.zip, který je možný importovat do Eclipse pomocí File -> Import -> General -> Existing Project into Workspace. Otevře se okno s nastavením. Je potřeba vybrat Select archive file a pomocí tlačítka Browse najít soubor ClientDNS-s-knihovnami. V poli Projects se objeví DNSClient projekt, který je potřeba zaškrtnout a potvrdit nastavení pomocí Finish. Vytvořený projekt hlásí chyby spojené s knihovnou JavaFX. Ostatní knihovny nutné pro běh aplikace

jsou v pořádku a není nutné je nikterak upravovat. Situaci po importu lze vidět na obrázku 4.3.



Obr. 4.3: Importovaný projekt s chybami spojenými s knihovnou JavaFX před nastavením uživatelsky definované knihovny.

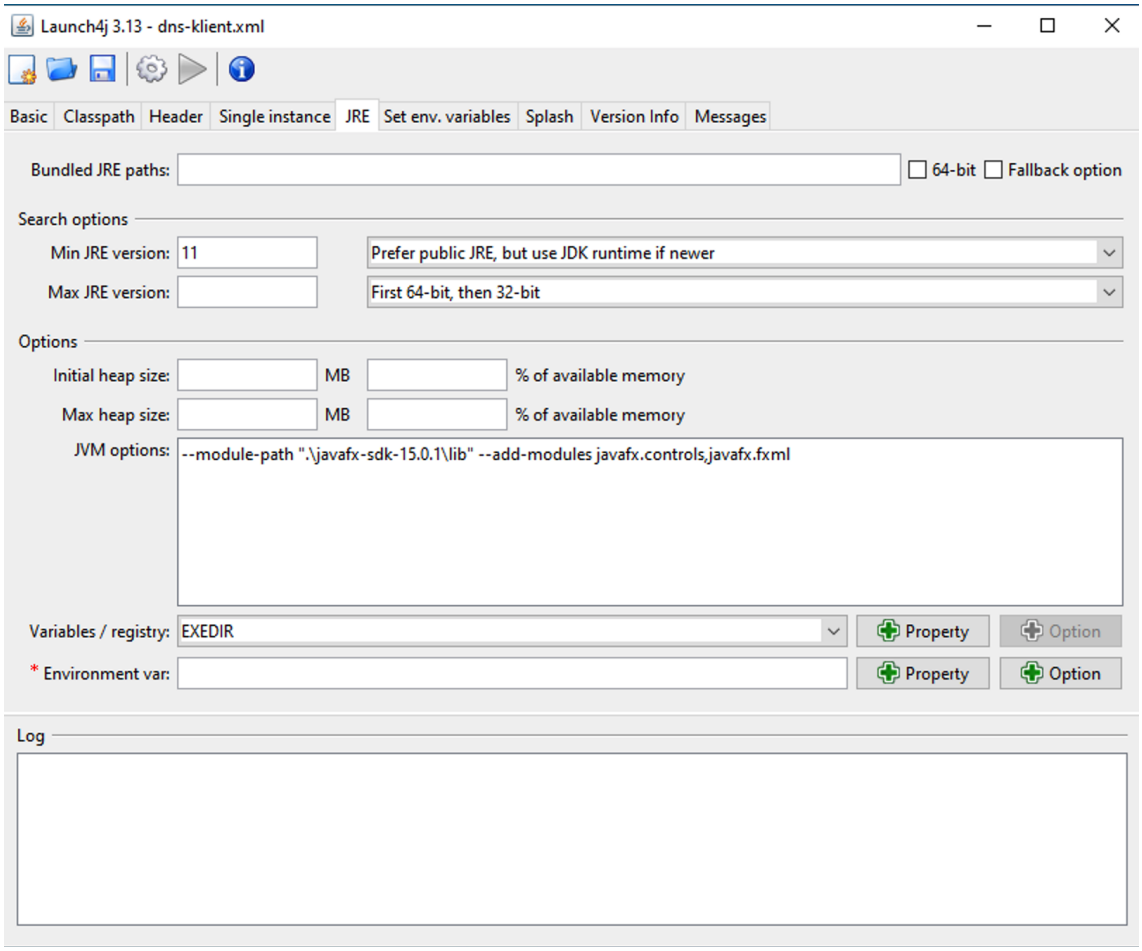
Je nutné definovat uživatelskou knihovnu pomocí Window -> Preferences -> Java -> Build Path -> User Libraries -> New. Objeví se okno s polem pro zadání jména knihovny: JavaFX15. Kliknutím na položku JavaFX15 se povolí tlačítka v pravé části a je třeba zvolit Add JARS. Zobrazí se roleta se složkami v projektu. Zde je potřeba vybrat službu javafx-sdk-15.0.1 -> lib. Ve složce lib zvolit veškeré soubory končící na .jar, poté potvrdit výběr a kliknout na Apply and Close. Projekt nyní nehlásí žádné chyby a je možné jej spustit pomocí Run -> Run Configuration. Dvojklikem na Java Application se vytvoří nová konfigurace. Konfiguraci je dobré si pojmenovat pomocí pole Name, zvolit Projekt pomocí Browse a vybrat Main Class. Aplikace má svou spustitelnou třídu v balíčku application a jedná se o třídu Main. V posledním kroku je nutné jít do záložky Arguments a do pole VM arguments vložit řetězec: „--module-path ".\javafx-sdk-15.0.1\lib" --add-modules javafx.controls,javafx.fxml“. Poté je možné projekt spustit pomocí Run.

4.2.1 Vytvoření spustitelného .exe souboru

Po vytvoření konfigurace pro spuštění programu je možné projekt exportovat do spustitelného .jar souboru pomocí File -> Export -> Java -> Runnable JAR file. V následujícím okně pod Launch configuration je nutné vybrat námi

vytvořenou konfiguraci z předešlé kapitoly a zvolit cestu, kde bude soubor uložen po exportu. Varování při exportu jsou obvyklá.

Pro tvorbu .exe souboru je nutné stáhnout Launch4j pro vytváření spustitelných .exe souborů z .jar souborů. Aplikaci lze stáhnout z `http://launch4j.sourceforge.net/`. Po stažení a instalaci jsou povinné dva parametry: vstupní .jar soubor a výstupní cesta pro .exe soubor. Aplikace DNS klienta potřebuje ještě vložit i parametry do záložky JRE. Potřebné nastavení je zobrazeno na obrázku 4.4. Stisknutím ikony ozubeného kolečka se vytvoří spustitelný .exe soubor.



Obr. 4.4: Nastavení Launch4j pro vytvoření spustitelného .exe souboru

Pro správný běh programu je nutné do stejné složky jako je spustitelný .exe soubor zkopírovat i složky JavaFX knihovny a služku pojmenovat javafx-sdk-15.0.1. Takto vytvořená složka je již součástí Eclipse projektu. Je možné ji zkopírovat odtud. Poté je možné spustit .exe soubor.

Závěr

Výstupem teoretické části této diplomové práce je zmapování aktuálních trendů v rozvoji protokolu DNS a jeho nových variant. Zároveň práce popisuje strukturu zpráv, transportní protokoly pro přenos protokolu DNS, aktualizaci DNSSEC pro možnost ověření původce a integritu zprávy.

Výsledkem praktické části je implementace klientské aplikace pro překlad doménových jmen za účelem výuky. Aplikace podporuje tři standardizované protokoly pro překlad doménového jména. Jedná se o protokoly DNS, mDNS a DoH. Po dohodě s vedoucím práce byl protokol LLMNR nahrazen DoH a zachován počet protokolů definovaný v zadání práce. Program je nástrojem pro pochopení fungování DNS protokolu, se kterým nemá uživatel možnost se jinak setkat, protože překlady na pozadí provádí operační systém.

Při vývoji a testování aplikace autor narazil na rozdíly mezi resolvery a jednotlivé odchylky zaznamenal tak, aby mohly být představeny při výuce nebo vyzkoušeny studenty přímo v klientské aplikaci. Kompletně vyzkoušel a zaznamenal výsledky pro volitelné položky v dotazu pro doménu .cz s typem dotazu a na všechny definované resolvery ve vytvořené aplikaci.

Klientská aplikace je porovnána s již existujícími nástroji, které ovšem jsou bez grafického rozhraní. Popsány jsou jak silné, tak i slabé stránky aplikací Nslookup, Dig a vytvořené aplikace. Samotná aplikace přináší možnosti, které terminálové nástroje nemají. Mezi nové vlastnosti patří volba zdrojového rozhraní, podržení TCP spojení pro zaslání vícero dotazů a protokol DoH, který ani jeden původní nástroj neumí.

Hlavním problémem, který se při vývoji aplikace objevil byla možnost unicastových odpovědí v protokolu mDNS. Bohužel na UDP portu již naslouchá systémová služba a problém nemá vhodné řešení, aby zprávy mohla přijímat jak klientská grafická aplikace, tak i systémová služba.

Při vývoji aplikace v důsledku volání pokročilých metod byl projekt změněn z Java 8 na aktuální Java 11 a ve složce spustitelného souboru je i složka knihovny JavaFX verze 15, která není součástí Java 11.

Úvodní obrazovka aplikace nabízí volbu více protokolů, než aktuální verze podporuje. Nabízí se možnost dotvoření zbylých protokolů v rámci dalšího vývoje popřípadě vypsání nového tématu pro závěrečnou práci.

V rámci dalšího rozvoje aplikace bude repozitář projektu na stránce `https://github.com/mbio16/clientDNS` dále využíván i pro následující studenty, které autor práce přidá jako přispěvatele do výše zmíněného repozitáře. Lze tak předpokládat, že projekt je aktuálně rozpracován a implementací dalších protokolů bude postupně vylepšován.

Literatura

[1] *Internet a Jeho Služby: Srovnání RM ISO/OSI a TCP/IP* [online]. Jablonec nad Nisou: STABLECZ, 2009 [cit. 2020-10-26]. Dostupné z URL: <http://ijs.8u.cz/index.php/standardizace-v-pocitacovych-sitich/srovnani-rm-iso-osi-a-tcp-ip>.

[2] PRAMATAROV, Martin. *DNS history. When and why was DNS created? CloudsDNS* [online]. Bulharsko, 2018 [cit. 2020-10-11]. Dostupné z URL: <https://www.cloudns.net/blog/dns-history-creation-first/>.

[3] Domain Name System (DNS) History. *BroadbandNow* [online]. [cit. 2020-11-05]. Dostupné z URL: <https://broadbandnow.com/internet/i/iw_dns_history.html>.

[4] BENSON, Theophilus a Bryan PROSSER. *Computer Networks - Lab 4: DSN Primer Notes* [online]. 2016 [cit. 2020-11-05]. Dostupné z URL: <https://www2.cs.duke.edu/courses/fall16/compsci356/DNS/DNS-primer.pdf>.

[5] P. MOCKAPETRIS. *Request for Comments: 883: DOMAIN NAMES - IMPLEMENTATION and SPECIFICATION*. The Internet Engineering Task Force, 1983. Dostupné také z URL: <https://tools.ietf.org/html/rfc883>.

[6] JITHIN. *Common DNS Resource Records with examples.* Interserver.net [online]. 2016 [cit. 2020-10-26]. Dostupné z URL: <https://www.interserver.net/tips/kb/common-dns-resource-records-examples/>.

[7] IANA. *Root Servers* [online]. Virginia, USA, 1995 [cit. 2020-10-13]. Dostupné z URL: <https://www.iana.org/domains/root/servers>.

[8] *Computer Hope: DNS resolver* [online]. 2020 [cit. 2020-12-01]. Dostupné z URL: <https://www.computerhope.com/jargon/d/dns-resolver.html>

[9] KUROSE, James F. a Keith W. ROSS. *Počítačové sítě.* Brno: Computer Press, 2014. ISBN 9788025138250.

[10] CARLETTI, Simone. *Understanding the WHOIS protocol. Simone 's Blog* [online]. 2012 [cit. 2020-10-26]. Dostupné z URL: <https://www.citacepro.com/dok/RR7m8G2MVSJRvRtj>.

[11] MCKAY, Dave. *How to Use the dig Command on Linux.* How-To Geek [online]. 2020 [cit. 2020-10-26]. Dostupné z URL: <https://www.howtogeek.com/663056/how-to-use-the-dig-command-on-linux/>

[12] ŠPAČEK, Petr. DNS flag day 2020. *DNS Flag Day* [online]. 2020 [cit. 2020-10-26]. Dostupné z URL: <https://dnsflagday.net/2020/>.

[13] CZ.NIC, z. s. p. o.*DNSSEC: Bezpečení domény* [online]. 2018 [cit. 2020-12-01]. Dostupné z URL: <https://www.nic.cz/dnssec/>.

[14] *DNSSEC Keys and Signing Explained.* BytesArena [online]. USA, 2019 [cit. 2020-10-15]. Dostupné z URL:<https://bytesarena.com/dns/dnssec/2019/02/19/dnssec-keys-and-signing-explained.html>.

[15] R. Arends a S. Rose. *IETF RFC 4033: DNS Security Introduction and Requirements*. The RFC Archive [online]. Colorado: Telematica Instituut, 2005 [cit. 2020-10-15]. Dostupné z URL: <https://www.rfc-archive.org/getrfc.php?rfc=4033#gsc.tab=0>.

[16] P. Wouters. *Algorithm Implementation Requirements and Usage Guidance for DNSSEC* [online]. IETF, 2019 [cit. 2020-10-15]. Dostupné z URL: <https://tools.ietf.org/html/rfc8624>.

[17] CZ.NIC.*Provozní manuál DNSSec pro registr .cz a 0.2.4.e164.arpa* [online]. 2010(1.9), 7 [cit. 2020-10-15]. Dostupné z URL: <https://www.nic.cz/files/nic/doc/Provozni_manual_DNSSEC_201001_final.pdf>.

[18] Národního úřadu pro kybernetickou a informační bezpečnost. *MINIMÁLNÍ POŽADAVKY NA KRYPTOGRAFICKÉ ALGORITMY: doporučení v oblasti kryptografických prostředků* [online]. Brno, 2018, 28.11.2018, (verze 1.0), 8 [cit. 2020-10-15]. Dostupné z URL: <https://www.nukib.cz/download/uredni_deska/Kryptograficke_prostredky_doporuceni_v1.0.pdf>.

[19] BURDA, Karel. *Kryptografie okolo nás.* Praha: CZ.NIC, z.s.p.o., 2019. CZ.NIC. ISBN 978-80-88168-52-2.

[20] BALCI, Mete. *A Minimum Complete Tutorial of DNSSEC* [online]. 2020 [cit. 2020-11-19]. Dostupné z URL: <https://metebalci.com/blog/a-minimum-complete-tutorial-of-dnssec/>.

[21] *DNSSEC Guide : Chapter 6. Advanced Discussions: Proof of Non-Existence (NSEC and NSEC3)* [online]. [cit. 2020-11-23]. Dostupné z URL: <https://dnsinstitute.com/documentation/dnssec-guide/ch06s02.html>.

[22] BHARDWAJ, Rashmi. *What is mDNS(Multicast DNS)?* [online]. [cit. 2020-12-06]. Dostupné z URL: <https://networkinterview.com/what-is-mdnsmulticast-dns/>

[23] CHESHIRE S. a KROCHMAL M. *Request for Comments: 6762: Multicast DNS* [online]. 2013 [cit. 2020-12-06]. Dostupné z URL: <https://tools.ietf.org/html/rfc6762>

[24] JEŘÁBEK, J. *Pokročilé komunikační techniky. Skriptum FEKT Vysoké učení technické v Brně*, 2020. s. 1-180.

[25] B. ABOBA, D. THALER a L. ESIBOV. *Request for Comments: 4795: Link-Local Multicast Name Resolution (LLMNR)* [online]. 2007 [cit. 2020-12-06]. Dostupné z URL: <https://tools.ietf.org/html/rfc4795>.

[26] WU, Peter. *DNS Encryption Explained. The Cloudflare Blog* [online]. 2019 [cit. 2021-01-16]. Dostupné z URL:<https://blog.cloudflare.com/dns-encryption-explained/>.

[27] Z. HU, L. ZHU, J. HEIDEMANN, et al. *Request for Comments: 7858: Specification for DNS over Transport Layer Security (TLS)* [online]. [cit. 2021-01-17]. Dostupné z URL: <https://tools.ietf.org/html/rfc7858>.

[28] P. HOFFMAN, ICANN, P. MCMANUS a MOZILLA. *Request for Comments: 8484: DNS Queries over HTTPS (DoH)* [online]. 2018 [cit. 2021-01-18]. Dostupné z URL: <https://tools.ietf.org/html/rfc8484>

[29] VIET DOAN, Trinh.*Measuring DNS over TLS from the edge.* Apnic [online]. 2021 [cit. 2021-04-20]. Dostupné z URL: <https://blog.apnic.net/2021/04/13/measuring-dns-over-tls-from-the-edge/>

[30] *Cloudflare Docs: DNS over HTTPS - Using JSON* [online]. 13.11.2020 [cit. 2021-01-18]. Dostupné z URL: <https://developers.cloudflare.com/1.1.1.1/dns-over-https/json-format>

[31] LAKSHMANAN, Ravie. *NSA Suggests Enterprises Use 'Designated' DNS-over-HTTPS' Resolvers. The Hacker News* [online]. 15.1.2021 [cit. 2021-01-18]. Dostupné z URL: <https://thehackernews.com/2021/01/nsa-suggests-enterprises-use-designated.html>

[32] C. HUITEMA, PRIVATE OCTOPUS INC., M. SHORE, et al. *Specification of DNS over Dedicated QUIC Connections: draft-huitema-quic-dnsoquic-06* [online]. 2019 [cit. 2021-01-20]. Dostupné z URL: <https://tools.ietf.org/id/draft-huitema-quic-dnsoquic-06.html>

[33] BAGIROV, Vasily. *AdGuard becomes the world's first public DNS-over-QUIC resolver! AdGuard DNS* [online]. 2020, 15.12.2020 [cit. 2021-01-20]. Dostupné z URL: ttps://adguard.com/da/blog/dns-over-quic.html>

[34] KASSNER, Michael. *Use DNS benchmarking tools to optimize performance* [online]. USA, 2010, 6.12.2010 [cit. 2020-10-15]. Dostupné z URL: <https://www.techrepublic.com/blog/data-center/use-dns-benchmarking-tools-to-optimize-performance/>.

[35] A. Costello. *IETF RFC 3492: Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA)* [online]. 2003 [cit. 2020-11-26]. Dostupné z URL: <https://tools.ietf.org/html/rfc3492>.

[36] Internet Assigned Numbers Authority (IANA).*Domain Name System Security (DNSSEC) Algorithm Numbers: DNS Security Algorithm Numbers* [online]. 2020 [cit. 2021-02-17]. Dostupné z URL: <https://www.iana.org/assignments/dns-sec-alg-numbers/dns-sec-alg-numbers.xhtml>

[37] CLOUDFLARE, INC. *DNS Amplification Attack: DNS amplification is a DDoS attack that leverages DNS resolvers to overwhelm a victim with traffic.* [online]. 2021 [cit. 2021-02-17]. Dostupné z URL: <https://www.cloudflare.com/learning/ddos/dns-amplification-ddos-attack/>

[38] KRČMÁŘ, Petr. *Knot Resolver slaví páté narozeniny, používá ho třeba Cloudflare na 1.1.1.1.* [online]. www.root.cz, 2019, 27. 6. 2019 [cit. 2021-5-20]. Dostupné z URL: <https://www.root.cz/zpravicky/knot-resolver-slavi-pate-narozeniny-pouziva-ho-treba-//cloudflare-na-1-1-1-1/>

[39] HUSTON, Geoff. *DNS at IETF 110. Apnic* [online]. 1.4.2021 [cit. 2021-04-19]. Dostupné z URL: <https://blog.apnic.net/2021/04/01/dns-at-ietf-110/>

[40] Petko D. Petkov. *Name (mDNS) Poisoning Attacks Inside The LAN*[online]. 23.1.2008 [cit. 2021-5-13]. Dostupné z URL: <https://www.gnucitizen.org/blog/name-mdns-poisoning-attacks-inside-the-lan/>

Seznam symbolů, veličin a zkratek

DNS Domain Name System

IP Internet Protocol

ISO/OSI International organization of Standardization/Open System Interconnection

TCP/IP Transmission Control Protocol/Internet Protocol

IPv4 Internet Protocol version 4

IPv6 Internet Protocol version 6

FTP File Transfer Protocol

RFC Request for Comments

DNSSEC Domain Name System Security Extensions

UDP User Datagram Protocol

TCP Transmission Control Protocol

TLD Top Level Domain

DHCP Dynamic Host Configuration Protocol

LLMNR Link-Local Multicast Name Resolution

mDNS Multicast DNS

MITM Man In the Middle

IPsec IP security

RSA Rivest Shamir Adleman

ECDSA The Elliptic Curve Digital Signature Algorithm

NÚKIB Národní úřad pro kybernetickou a informační bezpečnost

DoH DNS over HTTPS

DoT DNS over TLS

DoQ DNS over Quic

TLS Transport Layer Security

VPN Virtual Private Network

APIPA Automatic Private IP Addressing

URL Uniform Resource Locator

JSON JavaScript Object Notation

NSA National Security Agency

0-RTT Zero Round Trip Time

DDOS Distributed Denial-of-service attack

IDN Internationalized Domain Name

API Application Programming Interface

URL Uniform Resource Locator

Seznam příloh

A Obsah elektronické přílohy

- `/` ..... kořenový adresář přiloženého archivu
  - `Klient DNS v1.1.0` ..... složka se spustitelným souborem aplikace
    - `Klient DNS v1.1.0..exe` ..... spustitelný soubor aplikace
    - `java-fx-sdk-15.0.1` ..... složka s knihovou JavaFX
      - `bin`
      - `legal`
      - `lib`
  - `ClientDNS-s-knihovnami.zip` ... Eclipse projekt aplikace s knihovnami pro export do spustitelného jar souboru

B Stromová struktura projektu

Výpis B.1: Výpis stromové struktury Java projektu

```
+---src
   +---main
   |   +---java
   |   |   +---application
   |   |   |       application.css
   |   |   |       Main.java
   |   |   |
   |   |   +---enums
   |   |   |       AA.java
   |   |   |       APPLICATION_PROTOCOL.java
   |   |   |       AUTHENTICATE_DATA.java
   |   |   |       CACHE.java
   |   |   |       CERTIFICATE_FLAG.java
   |   |   |       CHECKING_DISABLED.java
   |   |   |       DOH_FORMAT.java
   |   |   |       DIGEST_TYPE.java
   |   |   |       DNSSEC_ALGORITHM_TYPE.java
   |   |   |       KEY_PROTOCOL.java
   |   |   |       IP_PROTOCOL.java
   |   |   |       KEY_TYPE.java
   |   |   |       KEY_USAGE.java
   |   |   |       OP_CODE.java
   |   |   |       QR.java
   |   |   |       Q_COUNT.java
   |   |   |       Q_TYPE.java
   |   |   |       RA.java
   |   |   |       RD.java
   |   |   |       R_CODE.java
   |   |   |       TC.java
   |   |   |       TRANSPORT_PROTOCOL.java
   |   |   |       RESPONSE_MDNS_TYPE.java
   |   |   |       WIRESHARK_FILTER.java
   |   |   +---exceptions
   |   |   |       CouldNotUseHoldConnectionException.java
   |   |   |       DnsServerIpIsNotValidException.java
   |   |   |       MessageTooBigForUDPException.java
   |   |   |       MoreRecordsTypesWithPTRException.java
   |   |   |       NonRecordSelectedException.java
```

```
39  |   |   |       NotValidDomainNameException.java
40  |   |   |       NotValidIPException.java
41  |   |   |       QueryIdNotMatchException.java
42  |   |   |       TimeoutException.java
43  |   |   |       CustomEndPointException.java
44  |   |   |       HttpCodeException.java
45  |   |   |       MessageTooBigForUDPException.java
46  |   |   |
47  InterfaceDoesnNotHaveIpAddressExcepion.java
48  |   |   |
49  ResponseDoesnNotContainRequestDomainNameException.java
50  |   |   +---models
51  |   |   |       DataTypesConverter.java
52  |   |   |       DomainConvert.java
53  |   |   |       Header.java
54  |   |   |       Ip.java
55  |   |   |       Language.java
56  |   |   |       MessageParser.java
57  |   |   |       MessageSender.java
58  |   |   |       Punycode.java
59  |   |   |       Request.java
60  |   |   |       Response.java
61  |   |   |       Settings.java
62  |   |   |       TCPConnection.java
63  |   |   |       UInt16.java
64  |   |   |
65  |   |   +---records
66  |   |   |       Record.java
67  |   |   |       RecordA.java
68  |   |   |       RecordAAAA.java
69  |   |   |       RecordCAA.java
70  |   |   |       RecordCNAME.java
71  |   |   |       RecordDNSKEY.java
72  |   |   |       RecordDS.java
73  |   |   |       RecordMX.java
74  |   |   |       RecordNS.java
75  |   |   |       RecordNSEC.java
76  |   |   |       RecordNSEC3.java
77  |   |   |       RecordNSEC3PARAM.java
78  |   |   |       RecordOPT.java
79  |   |   |       RecordPTR.java
```

```
|   |   |       RecordRRSIG.java
|   |   |       RecordSOA.java
|   |   |       RecordSRV.java
|   |   |       RecordTXT.java
|   |   |
|   |   \---ui
|   |           DNSController.java
|   |           GeneralController.java
|   |           MainController.java
|   |           MDNSController.java
|   |           DoHController.java
|   |
|   \---resources
|       +---fxml
|       |       DNS.fxml
|       |       Main.fxml
|       |       MDNS.fxml
|       |       DoH.fxml
|       |
|       +---images
|       |       copy-clipboard.png
|       |       czech-republic.png
|       |       icon.png
|       |       question-mark.png
|       |       united-kingdom.png
|       |
|       \---locale
|               Lang_cz.properties
|               Lang_en.properties
|
\---test
    +---java
       \---allTests
               AllTests.java
               byteToBitArray.java
               Domain.java
               IpTest.java
               PunycodeTest.java
               TestMain.java
```

C Příklady zdrojových kódů

Veškeré zdrojové kódy programu jsou dostupné v repozitáři projektu, který je veřejně dostupný na `https://github.com/mbio16/clientDNS`. Níže je pak pouze dvojice vybraných tříd.

C.1 Třída RecordA

Výpis C.1: Třída reprezentuje datový model záznamu typu A. Lze ji najít v balíčku records.

```
1  package records;
2
3  import java.net.InetAddress;
4  import java.net.UnknownHostException;
5  import org.json.simple.JSONObject;
6
7  public class RecordA extends Record {
8
9    protected InetAddress ipAddress;
10   protected String ipAddressAsString;
11   private static final String KEY_ADDRESS = "Ipv4";
12
13   public RecordA(
14       byte[] rawMessage,
15       int lenght,
16       int startIndex) throws UnknownHostException {
17     super(rawMessage, lenght, startIndex);
18     parseRecord();
19   }
20
21   private void parseRecord() throws UnknownHostException {
22     byte data[] = new byte[lenght];
23     int j = 0;
24     for (int i = startIndex; i < startIndex + lenght; i++) {
25       data[j] = rawMessage[i];
26       j++;
27     }
28     ipAddress = InetAddress.getByAddress(data);
29     ipAddressAsString = ipAddress.getHostAddress();
30   }
```

```
31
32    @Override
33    public String toString() {
34      return KEY_ADDRESS + ": " + ipAddressAsString;
35    }
36
37    @Override
38    public String getDataAsString() {
39
40      return ipAddressAsString;
41    }
42
43    @SuppressWarnings("unchecked")
44    @Override
45    public JSONObject getAsJson() {
46      JSONObject object = new JSONObject();
47      object.put(KEY_ADDRESS, ipAddressAsString);
48      return object;
49    }
50
51    @Override
52    public String[] getValesForTreeItem() {
53      String[] pole = { toString() };
54      return pole;
55    }
56
57    @Override
58    public String getDataForTreeViewName() {
59      return ipAddressAsString;
60    }
61
62  }
```

C.2 Třída TCPConnection

Výpis C.2: Objekt třídy TCPConnection slouží k abstrakci přístupu k síťovým prostředkům pro protokol TCP. Třídy lze najít v balíčku models.

```
1
2  package models;
3
4  import java.io.IOException;
5  import java.io.InputStream;
6  import java.io.OutputStream;
7  import java.net.InetAddress;
8  import java.net.InetSocketAddress;
9  import java.net.Socket;
10
11 import exceptions.CouldNotUseHoldConnectionException;
12 import exceptions.TimeoutException;
13
14 public class TCPConnection {
15   private InetAddress destinationIp;
16   private Socket socket;
17   private OutputStream outputStream;
18   private InputStream inputStream;
19   private static final int DNS_PORT = 53;
20   private static final int SOCKET_TIME_OUT_SEC = 3;
21   private byte[] responseMessage;
22   public TCPConnection(InetAddress ip) {
23     this.destinationIp = ip;
24     responseMessage = null;
25   }
26
27   public byte[] send(
28       byte messagesAsBytes[],
29       InetAddress ip,
30       boolean closeConnection)
31       throws
32     TimeoutException,
33     IOException,
34     CouldNotUseHoldConnectionException {
35     if (socket == null) {
36       connect();
37     }
```

```
38      if (!socket.getInetAddress().equals(ip)) {
39        closeAll();
40        this.destinationIp = ip;
41        connect();
42      }
43      if (socket.isClosed() || !socket.isConnected()) {
44        socket = null;
45        connect();
46      }
47      sendAndRecieve(messagesAsBytes);
48      if (closeConnection)
49        closeAll();
50      return responseMessage;
51    }
52
53    private void connect() throws
54            IOException,
55            InterfaceDoesNotHaveIPAddressException {
56        try {
57      socket = new Socket(destinationIp, DNS_PORT,
58            Ip.getIpAddressFromInterface(
59                netIntreface,
60                destinationIp.getHostAddress()),
61                0);
62        outputStream = socket.getOutputStream();
63        inputStream = socket.getInputStream();
64      }
65      catch (IndexOutOfBoundsException e) {
66        throw new InterfaceDoesNotHaveIPAddressException();
67      }
68    }
69
70    public void closeAll() throws IOException {
71      if (socket.isConnected() || !socket.isClosed()) {
72        inputStream.close();
73        outputStream.close();
74        socket.close();
75      }
76    }
77
78    private void sendAndRecieve(byte[] messagesAsBytes)
```

```
79              throws
80        CouldNotUseHoldConnectionException,
81        TimeoutException,
82        IOException {
83      try {
84        outputStream.write(messagesAsBytes);
85        // dns message has first two bytes which is the lenght
86        // of the rest of the message
87        byte[] sizeRicieve = inputStream.readNBytes(2);
88
89        UInt16 messageSize =
90        new UInt16().loadFromBytes(
91            sizeRicieve[0],
92            sizeRicieve[1]);
93        // based on size get the dns message it self
94        responseMessage =
95        inputStream.readNBytes(messageSize.getValue());
96      } catch (ArrayIndexOutOfBoundsException e) {
97        System.out.println(e.toString());
98        closeAll();
99        throw new CouldNotUseHoldConnectionException();
100     } catch (IOException e) {
101       throw new TimeoutException();
102     }
103   }
104
105 }
```

C.3 Třída UInt16

Výpis C.3: Objekt třídy UInt16 slouží k reprezentaci celého kladného čísla o velikosti 16 bitů. Lze nalézt v balíčku models.

```
1  package models;
2
3  import java.nio.BufferOverflowException;
4  import java.nio.ByteBuffer;
5  import java.util.Random;
6
7  public class UInt16 {
8    public static final int MAX_VALUE = 65535;
9    public static final int MIN_VALUE = 0;
10
11   private int value;
12
13   public UInt16() {
14
15   }
16
17   public UInt16(int value) {
18     if (value > MAX_VALUE || value < MIN_VALUE)
19       throw new BufferOverflowException();
20     this.value = value;
21   }
22
23   public int getValue() {
24     return value;
25   }
26
27   public int compareTo(UInt16 other) {
28     return (value - other.value);
29   }
30
31   public double doubleValue() {
32     return value;
33   }
34
35   public boolean equals(Object o) {
36     return (o instanceof UInt16) &&
37     (((UInt16) o).value == value);
```

```
38    }
39
40    public int hashCode() {
41      return value;
42    }
43
44    public int intValue() {
45      return value;
46    }
47
48    public String toString() {
49      return "" + value;
50    }
51
52    public byte[] getAsBytes() {
53      byte reverse[] = ByteBuffer.allocate(2).
54      putShort((short) value).array();
55      return new byte[] { reverse[1], reverse[0] };
56    }
57
58    public String getAsHex() {
59      String res = "";
60      for (byte b : getAsBytes()) {
61        res = String.format("%02x", b) + res;
62      }
63      return res;
64    }
65
66    public UInt16 loadFromBytes(byte[] bytes) {
67      byte[] a = new byte[]
68      {
69          (byte) 0x00,
70          (byte) 0x00,
71          bytes[1],
72          bytes[0]
73      };
74      ByteBuffer wrapper = java.nio.ByteBuffer.wrap(a);
75      return new UInt16(wrapper.getInt());
76    }
77
78    public UInt16 loadFromBytes(byte byte1, byte byte0) {
```

```
79      byte[] a = new byte[] {
80          (byte) 0x00,
81          (byte) 0x00,
82          byte1,
83          byte0
84      };
85      ByteBuffer wrapper = java.nio.ByteBuffer.wrap(a);
86      return new UInt16(wrapper.getInt());
87    }
88
89    public UInt16 generateRandom() {
90      Random random = new Random();
91      return new UInt16(random.nextInt(MAX_VALUE));
92    }
93  }
```