U.P.B. Sci. Bull., Series C, Vol. 87, Iss. 4, 2025 ISSN 2286 – 3540

MULTITHREADING VPN APPLICATION TO MONITOR WEB TRAFFIC AND UNBLOCK GEOBLOCKING WEBSITES

Marius-Valentin DRĂGOI[1], Adrian Nicușor MAIER[2], Alaa Abdul Al Muhsen Hussain ALZUBAIDI[3], Cosmin Petru SUCIU[4], Haider Abdullah ALI[5], Tiberiu Gabriel DOBRESCU[6], Anton HADĂR[7,8,9], Roxana-Adriana PUIU[10,*], Gabriel PETREA[11,*], Andra Paula AVĂSILOAIE[12,*].

This paper presents the development of a system that enables the end user to have easy and transparent access to geoblocked websites by using a multithread approach of VPN. These websites are sites that have been pre-loaded in the database contained inside the system. The first funda-mental component of the system is the DNS resolver, which can listen on both UDP/53 and TCP/53 for data transmissions. The TCP Proxy with Server Name Indication (SNI) support is the second key component of the system. It can listen on both TCP/80 (HTTP) and TCP/443 (HTTPS) ports while simultaneously supporting SNI. Both DNS and HTTP/S make use of a shared database, and user authentication is determined by the user's IP address, which can be either version 4 or 6 for both protocols. Through the utilization of the program that has been built, geoblocked websites can be made accessible by locating the actual IP addresses and establishing a tunnel that enables access to those sites.

Keywords: VPN; geoblocked websites; DNS resolver; TCP Proxy; SNI;

[1] Lecturer, Faculty of Industrial Engineering and Robotics, NUST POLITEHNICA Bucharest, Romania, e-mail: marius.dragoi@upb.ro
[2] Engineer, Faculty of Informatics, University *Titu Maiorescu*, Bucharest, Romania; adrian.p.maier@gmail.com
[3] PhD Student, Faculty of Mathematics and Computer Science, University of Bucharest, Romania; alaa-abdulalmuhsen.alzubaidi@s.unibuc.ro
[4] Scientific Researcher III, National Research and Development Institute for Gas Turbines COMOTI, Bucharest, Romania; cosmin.suciu@comoti.ro
[5] PhD Engineer, Ministry of Youth and Sport, Baghdad, Iraq, e-mail: h_haider26@yahoo.com
[6] Professor, Faculty of Industrial Engineering and Robotics, NUST POLITEHNICA Bucharest, Romania, e-mail: tiberiu.dobrescu@upb.ro
[7] Professor, Faculty of Industrial Engineering and Robotics, NUST POLITEHNICA Bucharest, Romania, e-mail: anton.hadar@upb.ro
[8] Vice-President, Academy of Romanian Scientists, Bucharest, Romania
[9] Corespondent Member, Technical Sciences Academy of Romania, Bucharest, Romania
[10] Lecturer, Faculty of Industrial Engineering and Robotics, NUST POLITEHNICA Bucharest, Romania, e-mail: mechnoroxana@yahoo.com
[11] PhD Engineer, Faculty of Industrial Engineering and Robotics, National University of Science and Technology POLITEHNICA Bucharest, Romania, e-mail: gabriel.petrea@gmail.com
[12] PhD Student, Faculty of Industrial Engineering and Robotics, NUST POLITEHNICA Bucharest, Romania, e-mail: andra.avasiloaie@stud.etti.upb.ro

1. **Introduction**

The Internet traffic keeps growing by several factors, including business globalization, high speed network access, the proliferation of Internet of Things (IoT), content delivery, popularization of wearable devices, and new machine-to-machine communication technologies [1].

Currently, with the widespread use of computers and the internet, a Domain Name System (DNS) resolver is an essential component for the efficient operation of the internet. The DNS resolver specifically accomplishes this by transforming easily remembered domain names into Internet Protocol (IP) addresses, which are essential for identifying web servers and other network resources. This process enables users to access websites, email services, video games [2], and a wide variety of other applications that are accessible via internet in a rapid and secure manner.

Additionally, the DNS resolver helps reduce latency by utilizing caching and load sharing on DNS servers. This process ensures consumers a browsing experience that is not only reliable but also uninterrupted.

For the last forty years, the main protocol for transporting data on the Internet is Transmission Control Protocol (TCP) [3]. TCP Proxy is a server application that intermediates TCP connections, routing traffic between clients and servers. It can be used for load balance, traffic inspection, and traffic security. Operating at the TCP connection level can provide various functionalities without need to interpret or manipulate protocols at a higher level than TCP. For this reason, TCP Proxy can work with any type of protocol running over TCP, such as Hypertext Transfer Protocol Secure (HTTPS), File Transfer Protocol (FTP), etc. Notable TCP proxy applications include Socat, Squid, HAProxy, and Nginx.

Real-time monitoring including the selection of optimal service still representing a serious challenge [4]. This paper presents the software implementation of an application that unblocks geoblocked websites without altering the rest of the internet traffic.

Using the implemented software solution, any site for which an IP tunnel is configured, related to the site's region, becomes accessible. The solution presented in this paper does not affect the access time of sites that do not need to be unblocked.

This is compared to a regular Virtual Private Network (VPN) solution, where the computer is configured to filter all internet traffic, which slows down access to sites that do not require a VPN [5].

Another feature of the software solution presented in this paper is that all operations carried out on the network occur asynchronously, unlike the synchronous way they are typically conducted.

This work also offers the advantage of simultaneously configuring all devices connected to the server that routes internet traffic for multiple regions. One device can access the same site in one region's version, while another device can access it in a different region's version.

The present paper builds upon prior research conducted in the field of secure web access, VPN infrastructures, and DNS-based content routing. Below are five relevant studies that reflect similar goals, technologies, or conceptual approaches.

In 2018, Velusamy and Lent [1] presents a paper that introduces a mechanism for dynamically routing web requests based on network cost and server location. The routing logic adapts to traffic conditions and optimizes content delivery from geo-distributed resources. Its applicability is especially relevant to systems that prioritize speed and regional efficiency.

In 2019, Kohana et al. [2] presents a study that explores the application of Barabási–Albert (BA) scale-free network models to simulate browser-level interactions within a distributed virtual environment. While primarily designed for virtual worlds, the proposed architecture offers potential use cases for scalable peer-to-peer communication in browser-based applications.

In 2021, Marques et al. [6] propose a DNS-based firewall system that uses machine learning techniques to detect and prevent malicious DNS queries. The system employs classification algorithms to identify suspicious patterns in DNS traffic, aiming to mitigate threats such as DNS tunneling and poisoning.

In 2024, Rajendran and Ramasamy [4] show a research that presents a real-time performance evaluation framework for IoT networks using an improved „Eagle Strategy" model. It focuses on adaptive routing and communication efficiency in distributed systems, which can be analogously applied to large-scale VPN or tunnel-based routing systems.

In 2024, Gentile et al. [5] presents a study that evaluates the performance and security of overlay and virtual private networks (VPNs) deployed with open-source infrastructure. It focuses on comparative analysis between different VPN protocols and topologies, providing insights into throughput, latency, and encryption overhead in practical deployment environments.

A comparative analysis of the current paper with the presented papers from the literature can be seen in Table 1.

Table 1

Comparative analysis

Criteria / Work	**This paper**	**Velusamy and Lent [1]**	**Kohana et al. [2]**	**Marques et al. [6]**	**Rajendran and Ramasamy [4]**	**Gentile et al. [5]**
Main Objective	Unblocking geoblocked websites using an asynchronous	Cost-based web routing optimization.	Scalable network model for virtual browsers.	Malicious DNS traffic detection via ML.	Routing efficiency in IoT systems.	Security and performance of open-source VPNs.

	DNS and TCP Proxy system.					
Technologies used	C++, ASIO, Boost, Wireguard, SNI, SQLite, NAT64, DoH.	Cost-aware dynamic routing	BA model, distributed web architecture	DNS classifiers, supervised ML models.	Heuristic evolution-based routing.	OpenVPN, IPSec, Squid.
Unique contribution	Integrates DNS resolution and TCP SNI proxy into one asynchronous multithreaded application.	Region-aware content routing.	Peer-to-peer scaling in browser architecture.	ML-powered threat prevention at DNS level.	Evolutionar y strategy for real-time traffic.	Empirical benchmark of open-source VPN setups.
Advantages	High performance, region-specific tunnel mapping, platform-agnostic, non-intrusive for non-blocked traffic.	Adaptive and cost-efficient.	Scalable and redundant connection models.	Intelligent DNS filtering.	Real-time optimization in distributed setups.	Mature infrastructure, real-world deployment.
Disadvantages	Requires manual tunnel configuration at the router level.	Lacks security mechanis ms.	Experiment al in nature, limited web focus.	Limited to DNS-layer protection.	Not tailored to web traffic or VPNs.	May require advanced setup.
Novelty introduced	First integration of DNS recursion and SNI proxy with asynchronous, per-region routing logic.	Geogra-phical routing logic.	Scale-free P2P browser connectivity .	ML-based DNS security layer.	Intelligent monitoring of networked nodes.	Comparative performance metrics for VPNs.

2. Used technology

2.1 Recursive DNS Resolver

The Domain Name System (DNS) is a fundamental Internet service that maps human-readable domain names to IPv4/IPv6 addresses. This mapping is maintained by name servers, and most application-level communications therefore begin with a DNS lookup [6].

The domain name space is a rooted hierarchy (at "."), where names consist of labels separated by dots, each label denoting one level in the hierarchy.

At the top, Top-Level Domains (TLDs) delegate authority for their subdomains; TLDs include generic domains (e.g., .com, .net, .org, .gov, .edu) and country-code domains (e.g., .ro, .uk, .md, .iq).

A recursive resolver traverses this hierarchy to obtain records from the authoritative server for the queried name; if the authoritative answer returns a Canonical Name (CNAME), the resolver repeats the process for the canonical target until an address record is obtained.

2.2 TCP Proxy with SNI support

A TCP Proxy with Server Name Indication (SNI) support is a specialized type of proxy that routes client requests to external servers by analyzing the SNI field, an extension of the Client Hello, which is part of the Transport Layer Security (TLS) security protocol negotiation. This type of proxy operates at the transport layer protocol level. This proxy functions by intercepting TCP connections, extracting the site name from the SNI field, and then forwarding the request to the relevant external server. The SNI extension allows the specification of the domain name before negotiating the secure connection.

Thus, the proxy can:

- make connection routing decisions without decrypting the traffic.
- allowing the hosting of multiple Secure Socket Layer (SSL)/TLS services on the same IP.
- provide an increased level of security by maintaining end-to-end encrypted traffic.

3. Method

3.1 Description of the application

For the implementation of the application, several functional requirements were defined. The system integrates a recursive DNS resolver, capable of handling queries over UDP and TCP, with optional support for DNS over HTTPS (DoH). It supports resolution of A, AAAA, and CNAME records starting from the root servers, including positive and negative caching, redirection of selected domains to external resolvers, and tunnel-based resolution for geoblocked domains.

The second core component is the TCP Proxy with SNI support, which extracts the target domain name from the ClientHello message and forwards traffic through preconfigured tunnels (IPv6, NAT64, Wireguard, etc.) on a per-client or per-site basis.

Both modules rely on a shared SQLite database storing clients, domains, tunnels, and configuration data, complemented by a configuration file for initialization and a web interface for tunnel management. Access control is enforced at the resolver and proxy level based on registered client IP addresses.

The configuration of tunnels will be done on the router and is not within the scope of the application.

The implemented system can be employed in multiple scenarios, depending on whether the requested resource is geoblocked or not. These main use cases are summarized in Fig. 1.

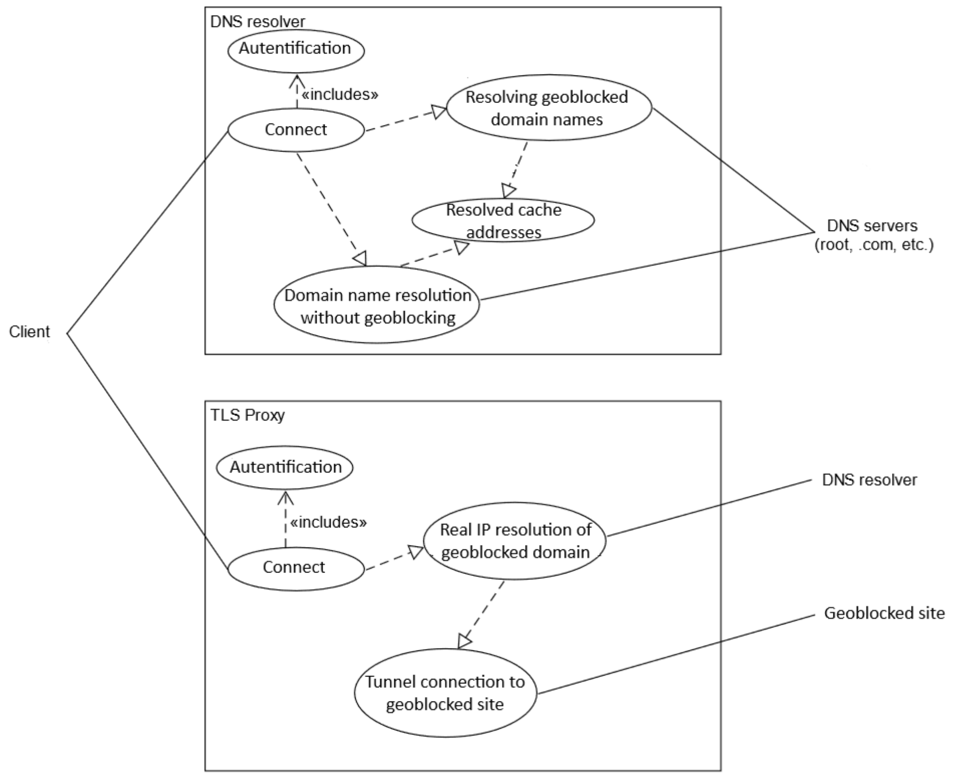

DNS resolver
Autentification
«includes»
Connect
Resolving geoblocked domain names
Resolved cache addresses
Domain name resolution without geoblocking
DNS servers (root, .com, etc.)
Client
TLS Proxy
Autentification
«includes»
Connect
Real IP resolution of geoblocked domain
Tunnel connection to geoblocked site
DNS resolver
Geoblocked site

Fig. 1. Use cases of the implemented system

The implemented system contains two major components (Fig. 2):

- The recursive DNS Resolver that listens to both UDP/53 and TCP/53;
- TCP Proxy with SNI support, which listens on both TCP/80 (HTTP) and TCP/443 (HTTPS).

Both components use a common database, and user authentication is based on the IP address (v4 or v6), for both DNS and TCP Proxy.

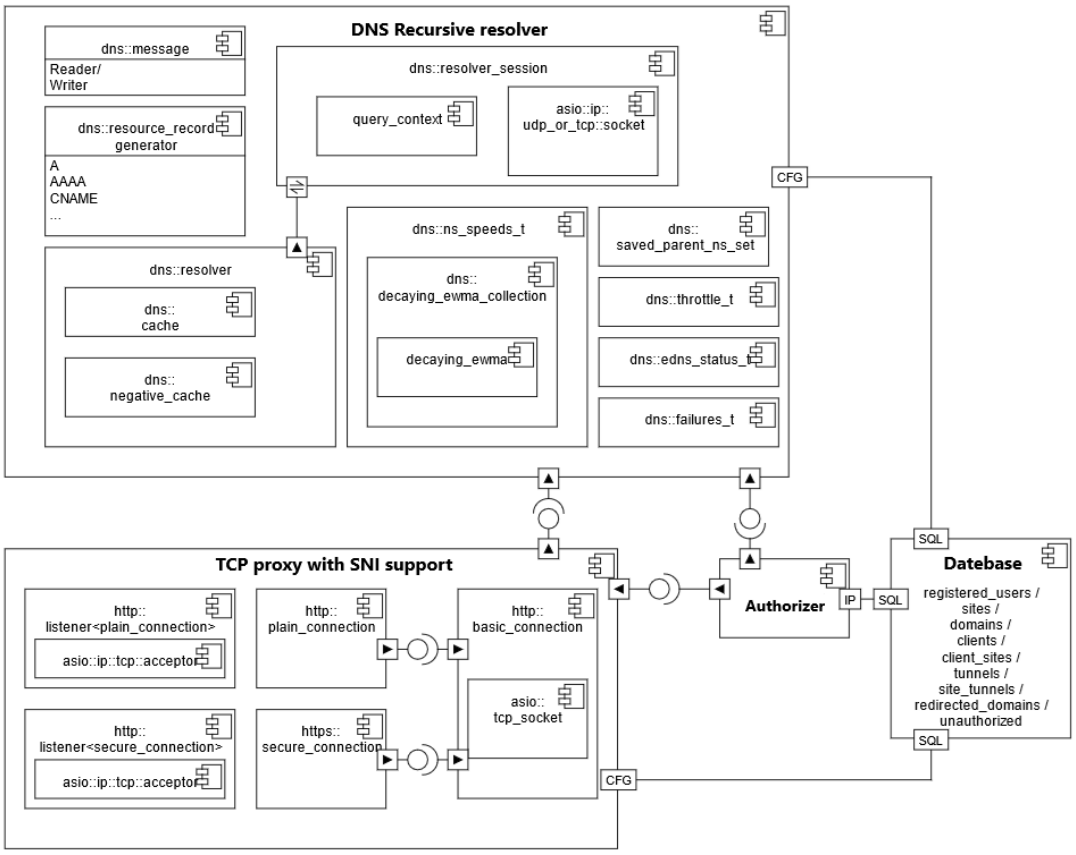


Fig. 2. Component diagram

The resolver first checks the cache for the queried domain; if no match is found, it recursively queries parent zones up to the root until authoritative name servers are located. If the authoritative response contains only referrals, the process continues iteratively until an answer is obtained or a negative response is confirmed. For geoblocked domains listed in the database, the resolver returns the system's own proxy address instead of the real IP, ensuring that the subsequent browser connection is redirected through the TCP Proxy.

The TCP Proxy authenticates clients by IP and handles both HTTP and HTTPS traffic: in the first case, the requested host is extracted from the HTTP header, while in the second case it is obtained from the TLS ClientHello SNI extension. Once the actual IP of the target site is resolved, the proxy establishes the external connection and transparently relays traffic in both directions. For geoblocked domains, connections are routed through regional tunnels to bypass restrictions, whereas normal domains are resolved natively without affecting performance.

In Fig. 3, all the information described in this subsection can be observed as sequence diagram.

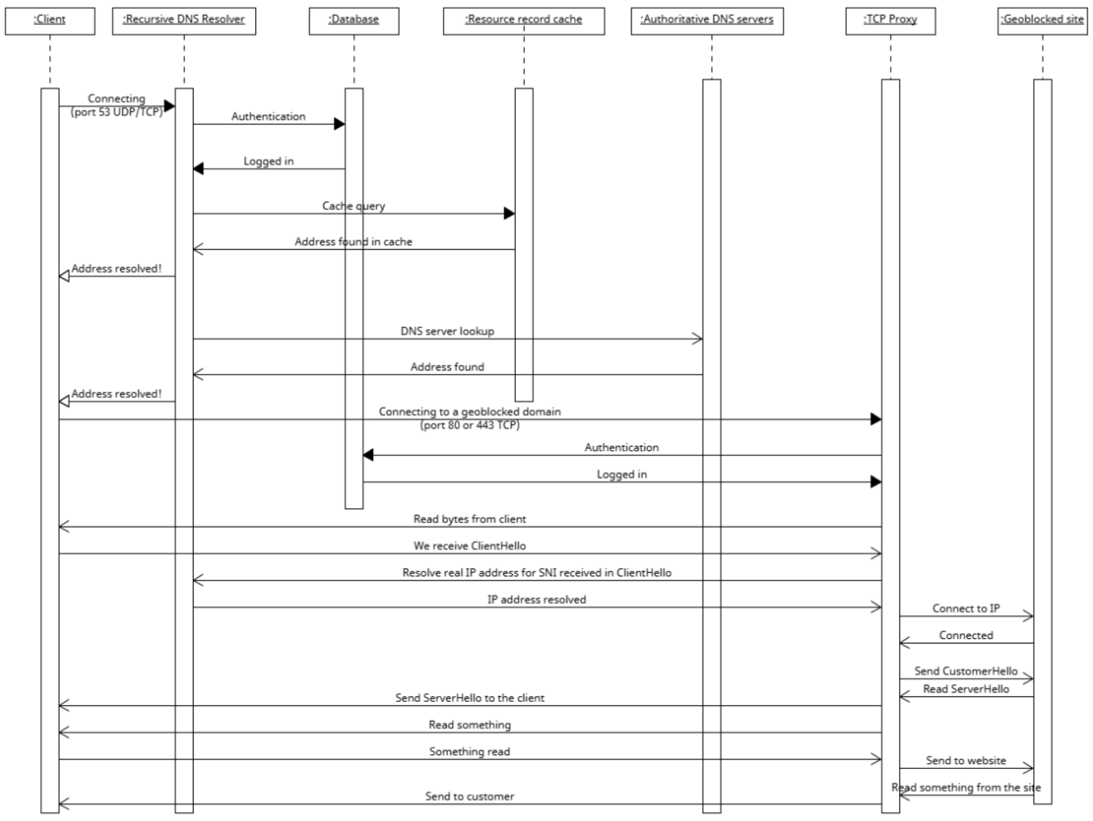


Fig. 3. Sequence diagram

A group of geoblocked sites for Romania has been listed, and for each site, a list of some of the regions where these sites are available has been created.

Implementing a recursive DNS Resolver, a TCP Proxy with SNI support, a mini-HTTP server for the interface, and configuring IPv6 GIF (Generic tunnel interface) tunnels, WireGuard, NAT64, OpenVPN, etc., a geoblocked site for a region is unblocked for the region where it is accessible.

3.2 Application implementation

The application was implemented in the C++ programming language, using VisualStudio2022 as the IDE (Integrated Development Environment). The application's interface was created using dynamic HTML, CSS, and JavaScript.

Among the open-source libraries used in the implementation of the application is ASIO, which is utilized for all asynchronous operations performed over the network and for managing timeouts, providing a higher level of handling TCP and UDP sockets.

Boost.MultiIndex was also used in the implementation of the DNS Resolver's recursive cache, particularly for managing resource records (NS, CNAME, A, AAAA, etc.), both positive and negative responses, allowing the implementation of multiple indices for the same collection of elements.

For logging operations, spdlog was used, which allows simultaneous logging to the console, file, network, and other destinations.

Library SQLite3 was used to implement the database that stores all the application's configurations, except for those necessary for application initialization, which are implemented in a TOML file. Access to the database is achieved through a custom wrapper, implemented in C++, to facilitate interaction in the written code.

The personal library stl_utils, implemented from scratch to address the shortcomings of the standard C++ library, was used for base64 conversions, md5 and sha1 hashes, conversions between network-byte-order and host-byte-order, bitwise operations for "enum class," MIME Media Type, and other functionalities.

Finally, fmtlib was used for operating systems where the compiler does not support the std::format class, thus ensuring the necessary compatibility and flexibility in various development environments.

3.2.1. Implemented classes

The application includes a set of classes dedicated to the recursive DNS Resolver and its integration with the TCP Proxy. At the I/O level, dns::udp_listener and dns::tcp_listener manage passive sockets for UDP and TCP connections (the latter supporting TCP Fast Open), while the corresponding udp_session and tcp_session classes parse queries, validate access based on client IP, and forward requests to the resolver.

The central component, dns::resolver, maintains runtime configuration and implements resolution logic using resource record caches for both forwarding and authoritative zones. Supporting structures such as dns::label, dns::name, and dns::main_cache (Boost.MultiIndex, LRU-based) enable efficient management of domain names, with dns::negative_cache handling negative responses.

Tunnel-specific addressing is managed by dns::connection, while dns::authoritative_domain stores zone data before delegation. Additional utilities include synchronization wrappers (dns::lock_guard<T>), message handling (dns::header), and type-generic resource record definitions (dns::gen_base), consistent with IANA RR TYPEs [7].

The recursive resolution algorithm itself is implemented in dns::resolver_session [8]–[14], which applies an asynchronous model for both internal queries (used by the TCP Proxy) and external DNS lookups.

Interaction with the SQLite database is handled through sqlite3::environment and sqlite3::statement, ensuring efficient query execution and secure configuration management. Together, these classes provide the structural basis for a performant and multithreaded recursive resolver tightly coupled with the proxy module.

3.2.2. DNS query resolution algorithm

The recursive resolution process begins with a cache lookup. If a valid record is found, the response is returned immediately; otherwise, CNAME redirections are followed, or NS records are searched by traversing the hierarchy from child domains to the root. Root NS records are always preloaded at startup to guarantee the ability to initiate resolution. For DS-type queries, the same logic is applied, with cache checks adapted to DS and CNAME types.

Once NS names are obtained, their IP addresses are either resolved from cache or queried directly from authoritative servers. Queries are performed asynchronously, first via UDP, with an automatic switch to TCP if the TC (truncated) bit is set in the DNS header, as required by RFC 1035 and RFC 2181 [3], [4]. Timeouts are enforced (1.5 s), and in case of failure the resolver retries using another IP of the same NS or another NS in the set. During this process, server performance is measured and stored for future decisions.

After a complete response is received, the system validates it by checking the Authoritative Answer flag and filtering relevant records. If the answer is authoritative and matches the original query, it is returned to the client. If the response contains a CNAME record, the resolution process restarts for the canonical target, while NS referrals are followed iteratively until the final answer is obtained or a negative response is confirmed.

The overall algorithm is consistent with the logic of PowerDNS Recursor [15], but in this implementation it has been designed fully asynchronously, with results also stored according to the regional tunnel used for resolution, thereby enabling efficient operation within the proposed VPN-based system.

4. **Performance evaluation**

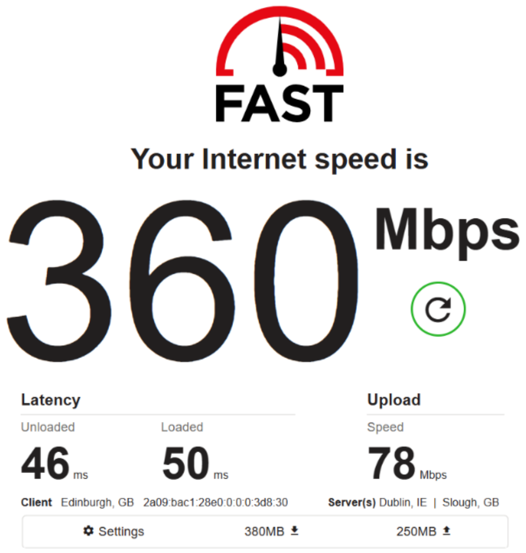
The implemented application presented in this paper:

- Achieved a latency of 46 ms, with the IP geolocation identified as Edinburgh, UK, and recorded network performance of 360 Mbps download and 78 Mbps upload (see Fig. 4).
- A traceroute performed via our application with VPN enabled resulted in a latency of 17 ms, as the native connection was used due to the absence of site-specific unblocking (see Fig. 5).

Using Windscribe VPN application:

- The connection achieved a latency of 45 ms with an IP location in London, UK, and recorded speeds of 98 Mbps for download and 43 Mbps for upload (see Fig. 6).
- The traceroute recorded a latency of 52 ms, with routing still occurring through London, UK, despite no unblocking being intended for the accessed site (see Fig. 7).

The performance evaluation was conducted using Fast.com.



FAST

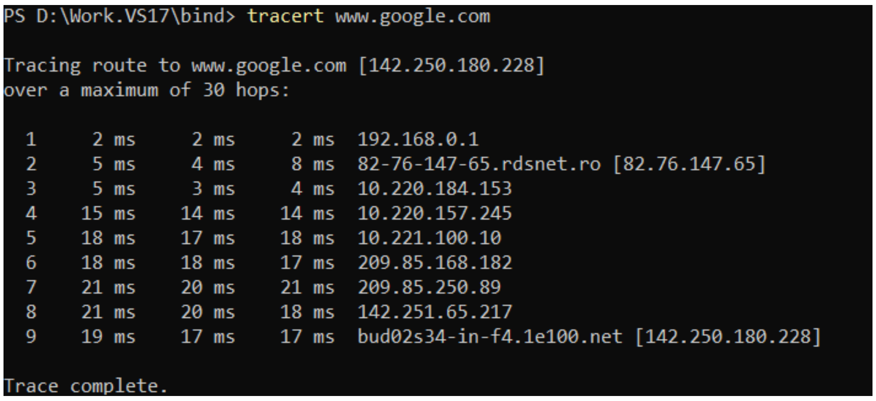
Your Internet speed is

360 Mbps

Latency		Upload
Unloaded	Loaded	Speed
46 ms	50 ms	78 Mbps

Client Edinburgh, GB 2a09:bac1:28e0:0:0:0:3d8:30 Server(s) Dublin, IE | Slough, GB

Fig. 4. Performance evaluation when the implemented application is used



```
PS D:\Work.VS17\bind> tracert www.google.com

Tracing route to www.google.com [142.250.180.228]
over a maximum of 30 hops:

  1     2 ms     2 ms     2 ms  192.168.0.1
  2     5 ms     4 ms     8 ms  82-76-147-65.rdsnet.ro [82.76.147.65]
  3     5 ms     3 ms     4 ms  10.220.184.153
  4    15 ms    14 ms    14 ms  10.220.157.245
  5    18 ms    17 ms    18 ms  10.221.100.10
  6    18 ms    18 ms    17 ms  209.85.168.182
  7    21 ms    20 ms    21 ms  209.85.250.89
  8    21 ms    20 ms    18 ms  142.251.65.217
  9    19 ms    17 ms    17 ms  bud02s34-in-f4.1e100.net [142.250.180.228]

Trace complete.
```

Fig. 5. Tracert via the implemented application

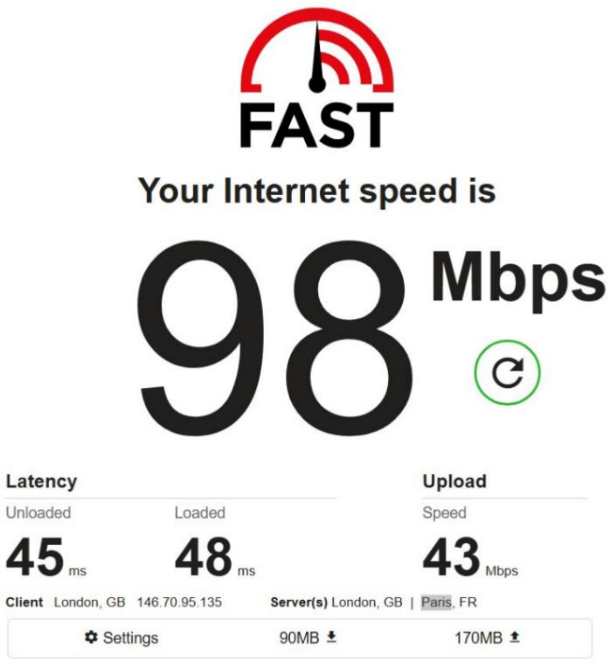
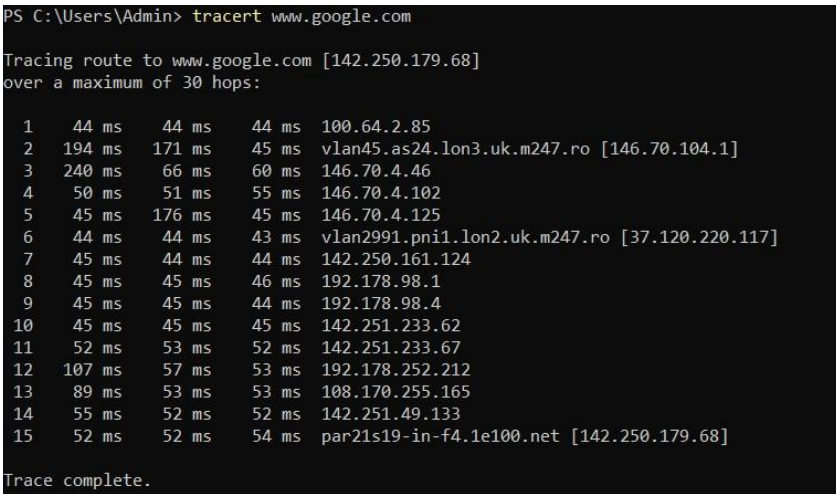


Fig. 6. Performance evaluation when Windscribe VPN is used



```
PS C:\Users\Admin> tracert www.google.com

Tracing route to www.google.com [142.250.179.68]
over a maximum of 30 hops:

  1    44 ms    44 ms    44 ms  100.64.2.85
  2   194 ms   171 ms    45 ms  vlan45.as24.lon3.uk.m247.ro [146.70.104.1]
  3   240 ms    66 ms    60 ms  146.70.4.46
  4    50 ms    51 ms    55 ms  146.70.4.102
  5    45 ms   176 ms    45 ms  146.70.4.125
  6    44 ms    44 ms    43 ms  vlan2991.pni1.lon2.uk.m247.ro [37.120.220.117]
  7    45 ms    44 ms    44 ms  142.250.161.124
  8    45 ms    45 ms    46 ms  192.178.98.1
  9    45 ms    45 ms    44 ms  192.178.98.4
 10    45 ms    45 ms    45 ms  142.251.233.62
 11    52 ms    53 ms    52 ms  142.251.233.67
 12   107 ms    57 ms    53 ms  192.178.252.212
 13    89 ms    53 ms    53 ms  108.170.255.165
 14    55 ms    52 ms    52 ms  142.251.49.133
 15    52 ms    52 ms    54 ms  par21s19-in-f4.1e100.net [142.250.179.68]

Trace complete.
```

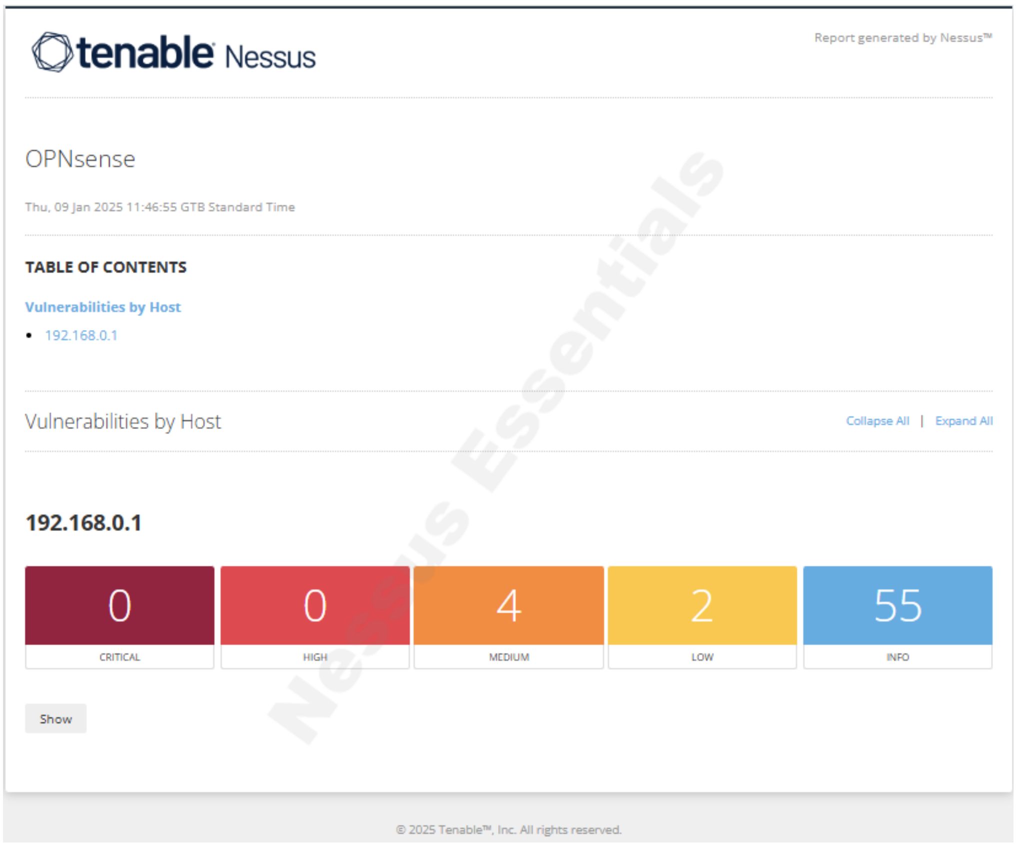
Fig. 7. Tracert via Windscribe VPN

5. Application security

It is important to check the software security [16]. For the vulnerability assessment of the application implemented and presented in this paper, the Tenable Nessus software solution [17] was used.

Tenable Nessus is a comprehensive vulnerability assessment solution designed to help organizations identify, evaluate, and manage security vulnerabilities in their network infrastructure.

In Fig. 8, the vulnerabilities of the implemented application and the system on which it runs can be observed. According to the customized report generated by the Nessus software solution, the application that is the subject of this thesis does not present any vulnerabilities. The only medium-level vulnerability warnings that appear in the generated report pertain to the operating system of the OPNsense firewall, based on FreeBSD 13.2.


Report generated by Nessus™

OPNsense

Thu, 09 Jan 2025 11:46:55 GTB Standard Time

TABLE OF CONTENTS

Vulnerabilities by Host

- 192.168.0.1

Vulnerabilities by Host

Collapse All | Expand All

192.168.0.1

0	0	4	2	55
CRITICAL	HIGH	MEDIUM	LOW	INFO

Fig. 8. Vulnerability scan report of the application and the operating system (FreeBSD 13.2) on which it runs

6. Conclusion

This paper describes the implementation of an application that serves both as a recursive DNS Resolver and as a TCP Proxy with SNI support. The main objective of this work was to implement a solution for accessing geo-blocked websites in an easy manner for the end user.

The solution offered combines two functionalities, DNS Resolver and TCP Proxy, to allow users to resolve DNS queries and establish secure connections to the desired online resources.

The recursive DNS Resolver module was designed and implemented, capable of efficiently interpreting and responding to DNS queries. This resolver interacts optimally with authoritative DNS servers to provide precise and fast query results. By fully implementing the DNS query resolution process, they can be routed through tunnels, thus preventing potential censorship at the level of Internet Service Providers (ISP). Additionally, this implementation represents the only solution that applies the DNS query resolution algorithm in an asynchronous manner for network data transfer, operating efficiently using a single thread of execution.

A TCP proxy with support for SNI was developed and integrated within the same application as the DNS Resolver module. This integration allowed for the management of HTTPS traffic in an efficient and secure manner, facilitating the optimal handling of encrypted connections. The TCP Proxy module with SNI support is essential for accessing geo-blocked sites, as it allows the establishment of SSL/TLS connections by correctly obtaining the hostname from the SNI extension. Thus, it ensures that the traffic is properly directed, avoiding potential blocks imposed by ISPs and contributing to maintaining free access to geographically restricted web resources.

In this paper, the integration of the DNS Resolver module and the TCP proxy into a single application was achieved, uniquely combining DNS query resolution and proxy functionality into a single solution. This integration enhances the end-user experience, as all geoblocked traffic is managed within the same system.

The intuitive interface of the application was designed with the aim of achieving an easy-to-use GUI, allowing the user to set up, without difficulty, the preconfigured sites in the application that need to be unlocked.

Multi-platform compatibility has been ensured by designing the application to support multiple operating systems, such as Windows, Linux, FreeBSD, etc. Additionally, certain features such as TCP FastOpen and TCP DelayAck have been implemented to the extent that the operating system allows their use.

The number of clients accessing websites that require unblocking is limited by the bandwidth of the server hosting the application. For example, streaming services such as Netflix typically require approximately 5 Mbps per user. Given a total bandwidth of 360 Mbps, the connection can be split to support up to 72 clients simultaneously accessing that specific service. All other websites are accessed via the native connection, and therefore, no practical limitation applies to clients in those cases.

REFERENCES

[1] G. Velusamy and R. Lent, "Dynamic cost-aware routing of web requests," Futur. internet, vol. 10, no. 7, p. 57, 2018.

[2] M. Kohana, S. Sakamoto, and S. Okamoto, "Web browser network based on a BA model for a web-based virtual world," Futur. Internet, vol. 11, no. 7, p. 147, 2019.

[3] S. K. Shrestha, S. R. Pokhrel, and J. Kua, "On the Fairness of Internet Congestion Control over WiFi with Deep Reinforcement Learning," Futur. Internet, vol. 16, no. 9, p. 330, 2024.

[4] V. Rajendran and R. K. Ramasamy, "Real-Time Evaluation of the Improved Eagle Strategy Model in the Internet of Things," Futur. Internet, vol. 16, no. 11, p. 409, 2024.

[5] A. F. Gentile, D. Macrì, E. Greco, and P. Fazio, "Overlay and Virtual Private Networks Security Performances Analysis with Open Source Infrastructure Deployment," Futur. Internet, vol. 16, no. 8, p. 283, 2024.

[6] C. Marques, S. Malta, and J. Magalhães, "DNS firewall based on machine learning," Futur. Internet, vol. 13, no. 12, p. 309, 2021.

[7] R. B. Roy Arends, Frederico AC Neves, Ólafur Guðmundsson, "Resource Record (RR) TYPEs," Internet Assigned Numbers Authority. https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml#dns-parameters-4 (accessed Jan. 14, 2025).

[8] P. V Mockapetris, "Rfc1035: Domain names-implementation and specification." RFC Editor, 1987, [Online]. Available: https://dl.acm.org/doi/pdf/10.17487/RFC1035.

[9] R. Elz and R. Bush, "RFC2181: Clarifications to the DNS Specification." RFC Editor, 1997, [Online]. Available: https://www.rfc-editor.org/rfc/rfc2181.html.

[10] R. Arends, R. Austein, M. Larson, D. Massey, and S. W. Rose, "Protocol modifications for the DNS security extensions RFC 4035," 2005, [Online]. Available: https://www.rfc-editor.org/rfc/rfc4035.html.

[11] S. Rose and W. Wijngaards, "RFC 6672: DNAME Redirection in the DNS." RFC Editor, 2012.

[12] S. Weiler and D. Blacka, "Rfc 6840: Clarifications and implementation notes for dns security (dnssec)." RFC Editor, 2013, [Online]. Available: https://www.rfc-editor.org/rfc/rfc6840.html.

[13] S. Bortzmeyer and S. Huque, "RFC 8020: NXDOMAIN: There Really Is Nothing Underneath." RFC Editor, 2016, [Online]. Available: https://www.rfc-editor.org/rfc/rfc8020.

[14] K. Fujiwara, A. Kato, and W. Kumari, "RFC 8198: Aggressive Use of DNSSEC-Validated Cache." RFC Editor, 2017, [Online]. Available: https://www.rfc-editor.org/rfc/rfc8198.html.

[15] PowerDNS, "PowerDNS Recursor." https://www.powerdns.com/powerdns-recursor.

[16] A.-I. Concea-prisăcaru, T.-A. Nițescu, V. Sgârciu, “SDLC and the importance of software security”, UPB Sci. Bull. Ser. C, no. 1, pp. 1–14, 2023, [Online]. Available: https://www.scientificbulletin.upb.ro/rev_docs_arhiva/full9e9_223740.pdf

[17] “Tenable Nessus.” https://www.tenable.com/products/nessus