AutoSetDNS - Resolving High DNS latency
(**Authors: Vellanki Mahitadivya, M. Nitya Jahnavi, D. Harihar)**
Team: VBIT

PROJECT OVERVIEW REPORT

ABSTRACT:

The Domain Name System (DNS) is a naming database in which internet domain names are located and translated into Internet Protocol (IP) addresses. This DNS service is provided by the ISP (Internet Service Provider) by default which is inefficient in many aspects. The performance of third-party DNS servers is superior to that of ISPs, but the user may not be aware of which server is optimal for his system and may move between them frequently to achieve the best browsing experience. We introduce **AutoSetDNS** to automate this process, which assigns the user an optimal DNS server from the Top Global Public DNS servers based on its geolocation, ping rates and query performance.

PROBLEM STATEMENT:

- ➔ Depending on many factors, our ISP's default DNS server may be slow and inefficient, which certainly will have an impact on how fast we are able to gain access to the internet for a better browsing experience.
- ➔ There are multiple Global Public DNS servers which outperform the traditional ISPs DNS such as GooglePublic DNS, cloudflare, CiscoUmbrella, Quad9, NuSEC, SafeDNS.
- ➔ Third-party DNS servers are publicly accessible DNS servers that are run by a variety of operators worldwide as an alternative to the DNS servers offered by our ISP (Internet Service Providers).
- ➔ The current advantage of local DNS resolvers is their ability to represent the end-user in terms of geographic location and its vicinity to content.
- ➔ Although DNS latency doesn't increase browsing speed, it does contribute to network latency when DNS queries are resolved.
- ➔ High DNS Latency equals high loading times. High DNS latency can be as a result of the DNS name servers not being in close geographic proximity to a large percentage of users who visit your site.
- ➔ Users may approach a third-party DNS service to enhance their browsing experience, but if many users do so, the user will once more experience the same latency issue.

INITIAL STAGE (Research and Analysis):

- At first, we considered including AutoSetDNS within the router itself.
- After receiving input from the jury members, we had to gather fresh data in order to implement AutoSetDNS at the system level of the user.
- Therefore, the data that was previously gathered for the project does not align with the new one.
- To learn more and develop an effective architecture for the service, we did a lot of research in different aspects.
- The main resource for this research study is the DNSPerf tool.
- We thoroughly analysed the top six DNS resolvers (Cloudflare, GoogleDNS, Cisco Umbrella, NuSEC, SafeDNS, and Quad9) throughout the world's top six continents and India. (Link for the detailed analysis report on DNS Public Resolvers:ANALYSIS ON GLOBAL DNS SERVERS).
- We have also done research on the DNS softwares currently available, but we came to the conclusion that none of them offer the automated allocation of DNS for free. (Link for the analysis report on Existing softwares: Research on Existing Softwares).

AutoSetDNS:

- As we all know Automation plays a key role in the industrial workspace. Repetitive tasks can be completed faster. Automating processes ensures high quality results as each task is performed identically, without human error.
- Users need not be concerned about the large webpage loads if the same automation approach is used for changing DNS servers. Users frequently use third party technologies to enhance their browsing experiences.
- Users may need some time to learn about third-party DNS servers and analyse which one's offer the best performance or the lowest ping. Additionally, they must be able to update DNS setup, which could take several minutes. But what if the customer wishes to use this conventional approach frequently to switch their DNS server? It might not be a good solution.
- Hence we introduce **AutoSetDNS** which automates the process of changing DNS servers.
- AutoSetDNS is a daemon service which helps to assign a DNS server with least ping automatically to the user's system

ARCHITECTURE:

AutoSetDNS is designed as a client-server architecture. Designing architecture is a challenging task. We made a lot of decisions to decide one particular architecture which would be the best for AutoSetDNS..

Our primary challenges were:

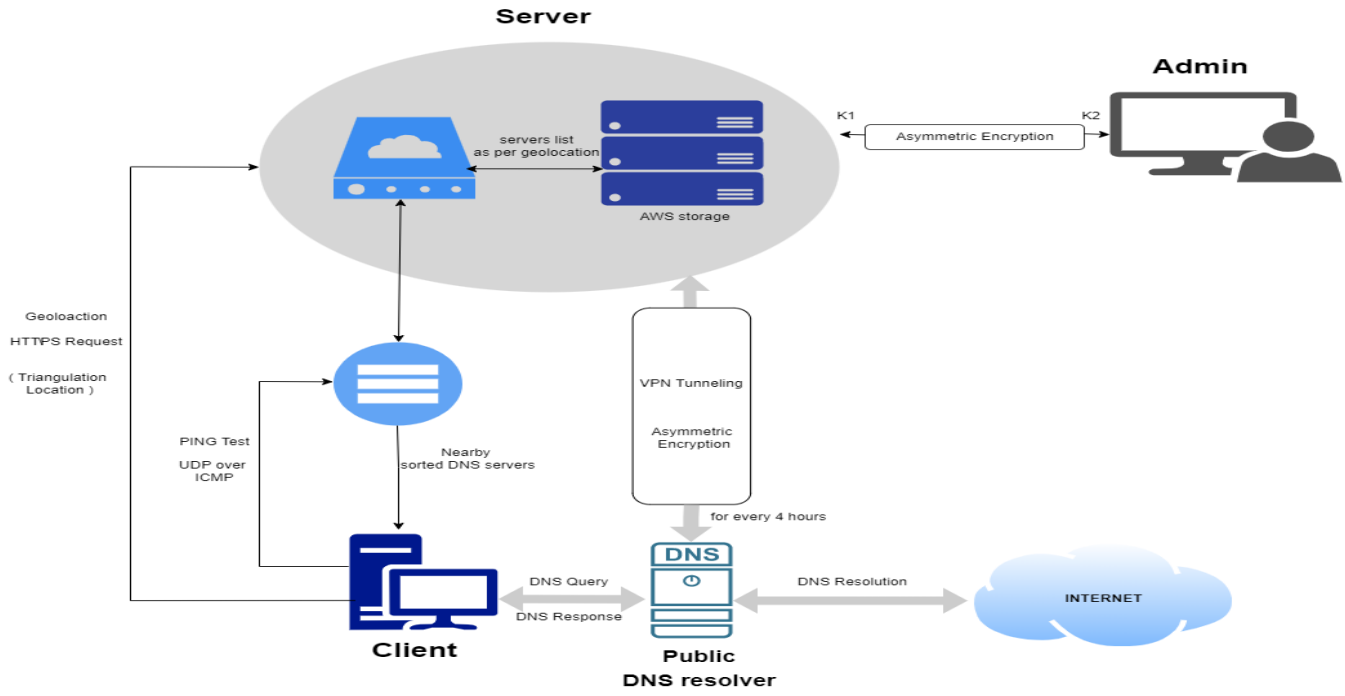
1.) Getting access to user's admin privileges
2.) Making the service cross platform
3.) Establishing a secure service between the server and the user.
4.) Setting up the environment on the user's system to run AutoSetDNS smoothly.
5.) Effective communication between service and CLI

Initially we came up with an overly complicated design which was later modified to a simpler version

➢ **Old Architecture:**

In the start, we developed a complicated model which had overly designed components like:

1. VPN Tunnelling: to encrypt the communication channel between the client and the cloud service
2. Triangulation Location: to prevent ip-spoofing
3. AWS S3 Storage: Cloud service integration with AutoSetDNS which would cost money.



These components overcomplicate the design and increase the overhead on AutoSetDNS.

Hence we moved to a simpler and an efficient architecture.

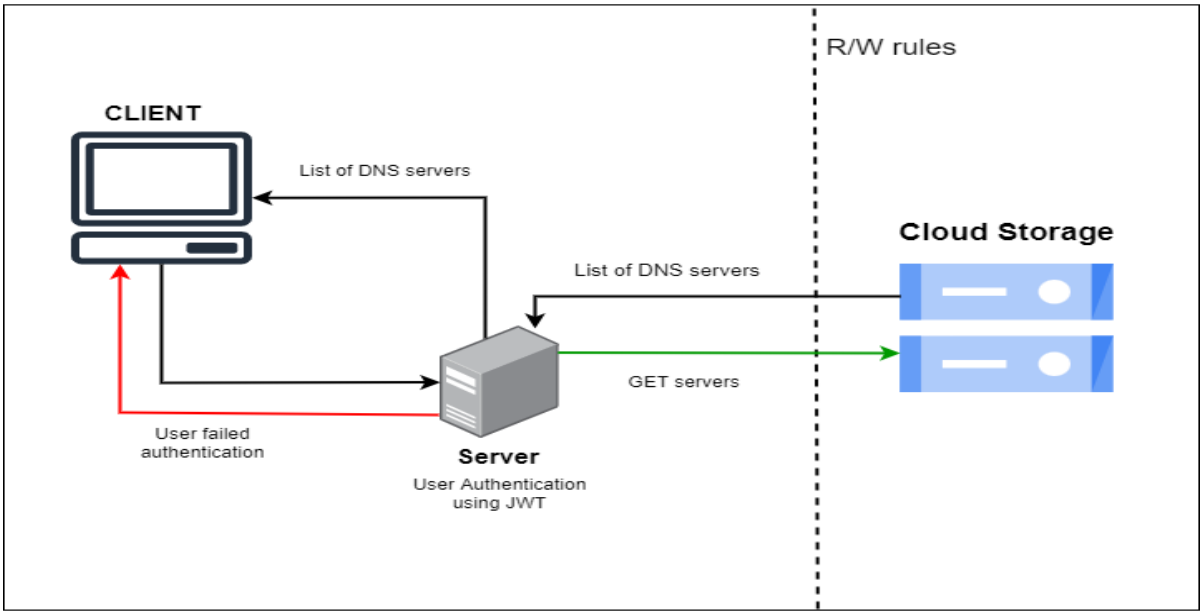
➢ **New Architecture:**

We developed a simpler client-server model which contains three main components:

1. **Client:** Client contains all the necessary logic which deals with fetching the DNS servers from the Server component to allocating the best server.

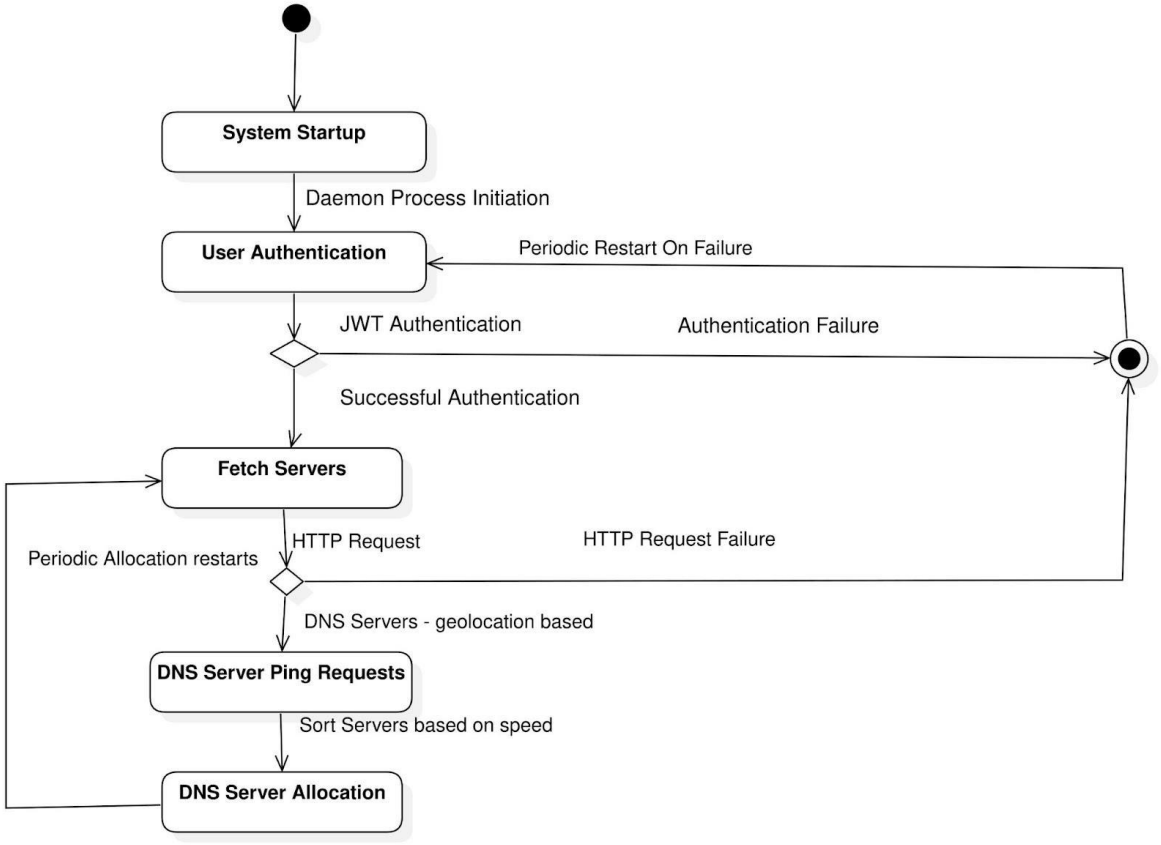
2. **Server**: Server is responsible for two main functions:
   i.)Authenticating the user with a token.
   ii.)Fetching the DNS servers list from the Firestore and passing it to the client.

3. **Cloud Storage**: Firebase is an efficient cloud service provider. AutoSetDNS uses firebase to store DNS servers location wise. This cloud storage is well-protected with read-write rules in terms of security.



FLOW DIAGRAM:

The following UML diagram represents the workflow of AutoSetDNS.



1. **System Startup :**
   When a user logs into the system , AutoSetDNS runs as a daemon process at startup of the system.
2. **User Authentication:**
   User is then authenticated using the JWT with the server to gain access to the list of DNS servers.
   If the user is able to authenticate successfully, access to the server is given.
   If the user is not able to authenticate successfully, then access to the server is denied.
3. **Fetch Servers :**
   After the user gets authenticated, AutoSetDNS fetches a list of DNS servers from the server (node application) using HTTP request.
4. **DNS Servers Ping Requests:**
   The DNS servers list fetched from backend are tested for performance using various ping tests and sorted according to their respective scores.
5. **DNS Server Allocation :**
   The best DNS server is picked from the sorted list and allocated in the user's system.

CODE BUILDING:

AutoSetDNS

- AutoSetDNS is a python script. Python is a dynamic programming language commonly used and comes preinstalled on linux and mac based systems. It makes cross platform scripts easy to implement.
- Changing DNS servers on any system needs admin privileges. So, one downside of it is whenever a user starts the AutoSetDNS, admin privileges have to be provided in order for the script to work.
- On windows, a dialog prompt saying YES or NO is displayed to the user.
- On linux, the terminal asks the user for the password to give access.
- So, We chose python as our platform to build AutoSetDNS.
- AutoSetDNS consists of two main scripts and other helper scripts.

Main scripts

1. AutoSetDNS script:

Which contains the logic for fetching , benchmarking and allocating the DNS servers.

Working :
We mainly focus on two important parameters to resolve high dns latency:

1. How far is the DNS server from the user
2. Query speed of DNS server.

First the user is authenticated, then using a third party geolocation API, the client fetches the current location of the user. This location is then used in the HTTP request query to get the respective servers from the backend. The servers are then benchmarked and the best one is allocated for every 30 minutes.

Working of AutoSetDNS

- ➢ AutoSetDNS is cross-platform and can run on windows 7/8/10/11 and linux.
- ➢ When the client installs **AutoSetDNS ,** a token is assigned to the client to authenticate requests being made to fetch the list of DNS servers.
- ➢ We can configure the program to run at the startup of the OS or we can start it from the command line with command `startASD`
- ➢ AutoSetDNS requests the backend for DNS servers; if the client has a valid token, a list of DNS servers are sent in HTTP request in return. If the token is invalid, the user can not access the list. Once the user gets the list, the following benchmark happens.

NOTE : List can vary depending upon the location of the client.

Benchmarking the DNS servers.

- We have the list of DNS servers right now. Pings of all the servers are calculated first.
- This is done by the command called **ping.** It is available in both windows and linux.
- It is used to calculate the 1st performance metric of the DNS server which is what is the nearest DNS server to the user.
- The nearest one gives the best ping. The lower the score the better.
- After calculating , we pick the top 2 servers. Here comes the second performance metric.
- Query time of pinging different domains. This can vary for different domains.
- We allocate the top 2 servers and ping different domains. Average Query time of different domains is calculated and the one with less query time and ping value is chosen and allocated.
- This way, we get the best possible DNS server.
- AutoSetDNS also calculates the pings every 3 mins and notes the averages down in a file.
- These averages are used in choosing the best DNS server, when allocating next time.
- This way we not only get the best DNS server, but also get the most stable one.
- Let's jump a little more into technical details.

Windows :

We discussed the assignment of the DNS server above. But how do we do that exactly?

- ➢ Windows provides a command line tool called netsh. This tool can be used to set the DNS server of our choice. (But there is a catch ,more on this in the Root privileges section).
- ➢ AutoSetDNS uses this tool and sets the custom DNS server.
  An example of the netsh command would be :

"**netsh interface ipv4 dns Wifi static 8.8.8.8**"

The above line when entered in the terminal and run as an administrator , would set google DNS as DNS server.

So, we call the netsh tool from AutoSetDNS and set the best server.

Linux :

- ➢ So far we have seen the implementation details in windows. Things work a bit differently in linux.
- ➢ Linux has a configuration file called resolv.conf for maintaining DNS servers. This file is in the root directory (exactly in **/etc** directory) of the system.
- ➢ Which means we need root permissions to edit that file.
- ➢ The file contains the following information about the DNS servers.
- ➢ The file resolv.conf typically contains search directives that specify the default search domains used for completing a given query name to a fully qualified domain name when no domain suffix is supplied. For instance, search example.com local.test configures the resolver to try additionally somehost.example.com and somehost.local.test.
- ➢ It also contains a list of IP addresses of nameservers for resolution. For instance, nameserver 1.1.1.1 configures the resolver to query for the name server with IP 1.1.1.1. Additional nameserver directives after the first are only used when the first or last used server is unavailable

Example :

/etc/resolv.conf

namespace 1.1.1.1
namespace 1.0.0.1

Here 1.1.1.1 (cloudflare) is the primary DNS server and 1.0.0.1 is the secondary DNS server.
So, to add a custom DNS server, we need to edit this file.

ROOT PRIVILEGES :

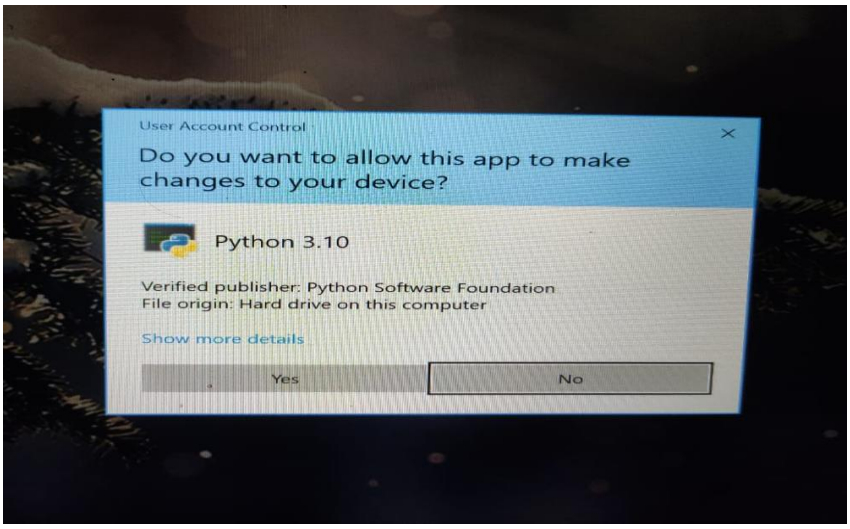
- ★ To edit resolv.conf, as explained above we need root permissions from the user. Editing the file is harmless, because linux generates a new file , if the
- ★ the existing one gets corrupted. So, it is totally safe to edit this file.
- ★ In linux , one can get root permissions using the command called sudo. When you want to run a command which needs root permissions, we run it with sudo command prefixed.

Example :

sudo #SOMECOMMANDHERE#

When we type the above command in the terminal, we will be asked for our password and then the next command is executed.
So, the AutoSetDNS needs the root permission to allocate the third party DNS servers.

In the case of Windows Operating System, the DNS servers can be configured only with the access of Admin-privileges. User has to allow the autoSetDNS prompt to execute.



WORKING in linux :

- As we saw in windows implementation, all the DNS servers, fetched from the backend are checked for speed and reliability.
- The best ones are selected and assigned using netsh in windows. But how do we assign a DNS server in linux? We have to edit the resolv.conf file. We run AutoSetDNSin root mode, so we will have permissions to write the file. So after checking for the best DNS server, we open the file in write mode and write the DNS server (primary and secondary) and close it in the python script.

2. AutoSetDNSCLI script:

AutoSetDNS comes with a command line utility. Through this CLI, we can get essential info from the AutoSetDNS.

Note : To start the AutoSetDNS, we can use the command startASD.
CLI provides following commands to ease the user experience.

1. **autoSetDNS showServers**
   This command gets the list of servers ,calculates the best servers and displays the output in the terminal/command prompt.

2. **autoSetDNS stop**
   This command is responsible for stopping the autoSetDNS running in the background.

3. **autoSetDNS allocated**
   This command shows the current allocated DNS servers.

4. **autoSetDNS set $arg1 $arg2**
   This command is quite different from the others. It can set a custom DNS server provided by the user. arg1 here is the primary DNS server and arg2 is secondary. arg2 is optional.
   Example : autoSetDNS set 1.1.1.1 1.0.0.1

(Note: Link for Detailed report on AutoSetDNSCLI: AutoSetDNS CLI in Windows10)

3. Setup script

To get started with ASD, we need it to be installed. For that , we have built a script to install all necessary packages , setting the paths, and install the ASD.

Setup File behaves differently on different operating systems, which is covered in the later part of this section. Let's dive into the setup file for more details.

What exactly does the setup file do ?
The job of the setup file is to install all necessary packages ASD uses , setting the path and installing ASD itself.
ASD uses the following packages.

1. elevate (for getting admin privileges).

2. requests (for making HTTP requests).

All necessary files are downloaded and moved to a secure location. On windows the location is C:\AutoSetDNS and on linux the location is /home/{user}/.AutoSetDNS

Now, setting up the path is for making ASD available in the command prompt (CMD), which is also called a terminal on linux and powershell on windows. This is done by adding the above locations to the environmental variables on respective operating systems.

Format of the setup file is different on different operating systems.

Windows :
We provide a setup.exe file to install the ASD.
Users will need to download this file first. Then double clicking on it will take care of everything.

Linux :
On linux , we provide a setup.py file. As linux comes with python by default, users will need to run that setup script to install ASD.

Logging the data

Whenever a DNS server is allocated, AutoSetDNS logs necessary information to a log file. Log file is maintained to study the DNS servers allocation and perform analysis on how frequently the DNS servers are changing to conclude with an appropriate time constraint to allocate DNS server.

So, we logged only the timestamp and the connected DNS server to the log file initially ,but then later log data includes details like ping results and all DNS server's results.

Following is the link to sheets containing logged data of past three months:

ASD Server Logs

- There are three sheets in total.

- **ASD log1 :** It consists of only top server's data which have been allocated.
  This log has an interval of 10 sec. We have taken 10 sec into consideration to check how frequently the top server is changing.

- **ASD log2:** It consists of top 5 DNS server's data along with their respective ping results.

- **ASD log3:** The logged data here contains the same details as log2 and as well as **top count (explained below).** This log is taken when core functionality like calculating RTT(Round trip time) is added.

TOP COUNT : Every 30 minutes , a DNS server is allocated. But internally , we calculate ping results of the top 5 DNS servers for every 5 minutes.

- So, we maintain a state called top_count , which is nothing but the number of times a server showed the best result in this 30 minutes of time period.
- This top_count state is considered while allocating the DNS server.

Assume that a DNS server **A** has a top_count of 3 and another DNS server **B** has a top_count of 2.
When ping results are calculated again , **A** server is given more priority nonetheless the score of server **B.**

Testing

We created the AutoSetDNS, but does it function properly? We must conduct extensive testing for that. Pinging a few top domains will not provide a reliable indication of the DNS server's performance, or how quickly it can resolve the query (low latency).

- So, we have built a test script which takes 10,000 domains and tests them with ISP and AutoSetDNS.
- The 10,000 domains includes all the TLDs such as .com, .net, .org, .edu etc.
- It calculates performance of ISP DNS server and AutoSetDNS allocated server based on its RTT value.
  It is done by calculating the time taken to lookup a domain. This time factor tells us about which DNS server is performing well.
- The test script has been ran on few devices (i.e. Windows 10/ Linux) and following are the results.
- The tests were performed simultaneously on two devices i.e. one which has AutoSetDNS and the other one using ISP.
- When these tests were performed, AutoSetDNS was running in the background and we observed the change of DNS servers in between.

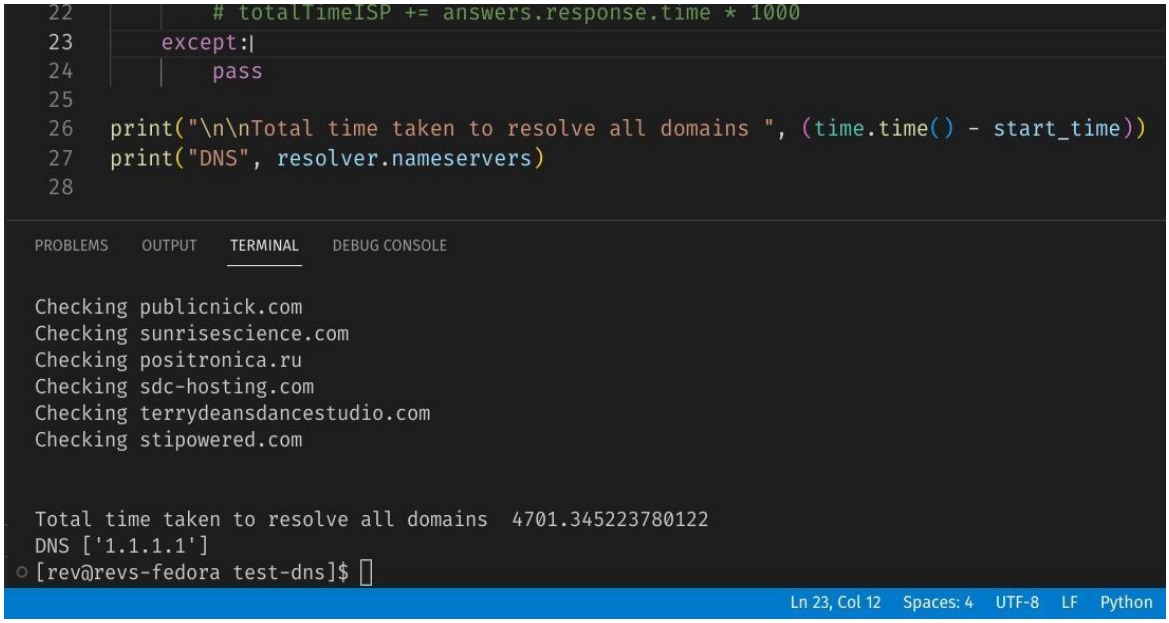
Following are the attachments containing the results performed on **Windows 10.**

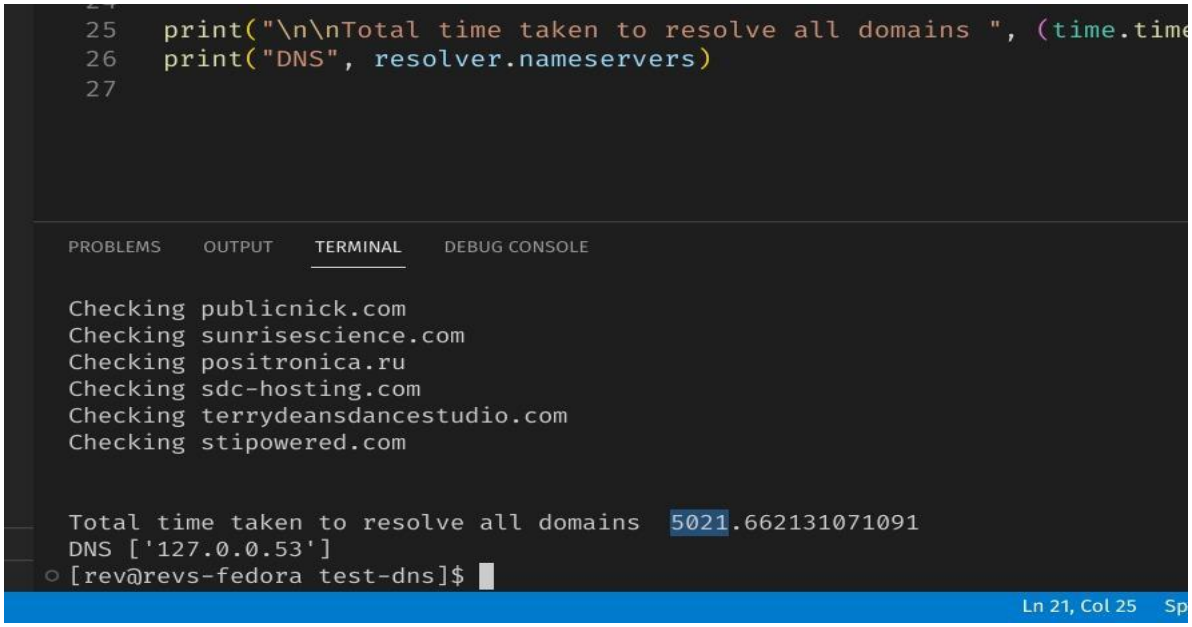
ASD and ISP Test Results

- According to the aforementioned log data, some queries are resolved more quickly by ISPs, whereas others are resolved more quickly by AutoSetDNS (ASD).

- The minimal value recorded by ISP and ASD to resolve that specific domain was taken into account in the result column.

The following results were obtained when the same 10,000 domains were tested on a Linux system:





Results:

With the help of the above mentioned testing log data, we came to the conclusion that while AutoSetDNS does not always outperform ISP, when the overall resolution time was computed, AutoSetDNS was able to complete all queries faster than ISP.

On Windows:

When the result was calculated from the obtained RTTs/ pings by AutoSetDNS and respective ISP, there are cases where ISP was able to resolve the dns queries faster than ASD.
The difference was calculated to check the performance of AutoSetDNS:

- If the minimum value(Result) is obtained from ASD log, it means ISP took higher resolution time to resolve that particular domain than ASD. Hence the difference will result in **-ve value.** Hence it indicates better performance of ASD.

- If the minimum value(Result) is obtained from an ISP log, it means ASD took a higher resolution time to resolve that particular domain than ISP. Hence the difference will result in **+ve value.** Hence it indicates better performance of ISP.

Hence for the above test log we calculated the overall time difference between ASD and ISP.

Time difference when ASD performed better than ISP = **-94860 ms**
Time difference when ISP performed better than ASD = **91387 ms**

% of better performance of ASD = 3473ms i.e 3.6 ~ 4% better than ISP

On Linux:

When the same test was performed on the linux system , the result obtained shown in the images above. We obtained the direct result of total resolution time.

For **ISP** it took **5021.66** secs of total time to resolve 10000 domains i.e **83.69 min**.
For **ASD** it took **4701.34** secs of total time to resolve 10000 domains i.e. **78.35 min**.

% of better performance of ASD= 5.34 min i.e. **5.6%** better than ISP

CONCLUSION:

We designed AutoSetDNS to enhance the DNS performance of the user's system. With little performance boost, a user also experiences a better browsing and gaming experience. Third party DNS servers also provide additional user and security benefits like malware filtering and protecting user's privacy. Our testing has led us to the conclusion that AutoSetDNS responds to DNS queries **3-5%** faster than ISP DNS, somewhat reducing DNS latency. In some rare scenarios, an ISP DNS server might perform a little better than ASD. This situation is uncommon, but possible. The small drawback is that every time a user logs in, admin capabilities must be granted by the user. The operating system's built-in default security mechanism is unavoidable. Hence the drawback. Aside from that, AutoSetDNS functions smoothly without disrupting other tasks and improves the DNS performance on the user’s system.