A Comparative Performance Evaluation of DNS implementations over different protocols

Group members -
Ayush Anand 19110074 | Bhoomika Mandloi 19110076 | Shreyshi Singh 19110032

Abstract:

The Domain Name Service (DNS) is a vital service on the Internet. DNS turns domain names into IP addresses, which browsers use to load internet pages. DNS servers make it possible for people to input names of websites into their browsers, such as Fortinet.com, without having to keep track of the IP address for every website. There are different DNS resolvers hosted at different servers, supporting different protocols. DNS is foreseen as an elastic and robust service, supporting failover mechanisms, decentralised configuration and multi-tenant data isolation.
In this project, we do a comprehensive analysis of DNS lookups over different protocols using different tools. We will compare their performance in terms of time taken and bandwidth used. We will do this by running tests for over 50 top sites as provided by Amazon's Alexa on their website.
By default, DNS uses the udp protocol for query lookups. As we move on to TCP, TLS and https, overheads are added in order to accommodate 3 way handshakes and security. QUIC is a relatively recent protocol that implements a new encryption mechanism which replaces SSL/TLS, supports the reduction of round trips between client and server during connection establishment, while its safety remains comparable to TLS. We will look at the performance of DNS over QUIC as well.
We end with the conclusions we draw from this project about DNS lookups over different protocols and mentioning the limitations and future works possible

Keywords: DNS, QUIC, TLS, HTTPS, UDP, DNS resolver

Introduction:

Introduced in 1983, the Domain Name Service(DNS) has become a critical component of the internet. DNS is a core service that plays an essential role in network communication. Originally it was designed as an unencrypted protocol, making user security a concern.The domain name system (DNS) serves as the internet's phone book. The DNS finds the correct IP address whenever users input domain names like Fortinet.com or Yahoo.com into the address bar of the web browsers. The IP address of the site is what directs the device to go to the right location to access the data of the site. Browsers utilise the address to transfer data to content delivery network (CDN) edge servers or origin servers after the DNS server has found the correct IP address. After that, the user will be able to access the website's information. The DNS server begins the process by determining the corresponding IP address for the uniform resource location (URL) of a website.A recursive resolver is another name for a DNS resolver. It's designed to handle DNS queries sent by web browsers and applications. After receiving the website URL, the resolver looks for the IP address associated with that URL.

DNS is an old protocol lacking all forms of security. Despite this, it is one of the Internet's most fundamental protocols. DoT and DoH are enhancements to the DNS protocol that use the same security layers as HTTPS: TLS to provide transport security. TLS is used by both DoT and DoH. More recently, QUIC was added to the mix. One of the most significant aims of QUIC is to minimise web traffic latency. QUIC (Quick UDP Internet Connections) is a new generation Internet protocol that speeds online web applications that are susceptible to delay by reducing round-trip time(RTT) needed to connect to a server.
This report presents the comprehensive measurement, comparison and analysis of DoH(DNS over HTTPS), Dot(DNS over TLS) and DoQ(DNS over QUIC) traffic. We start by comparing various protocols bandwidth such as UDP, DNSSEC, HTTPS, and TLS using different tools .We then analyse different server comparisons.

Related Work:

1. **Observing the Evolution of QUIC Implementations:** The QUIC protocol combines features initially found inside the TCP, TLS and HTTP/2 protocols. The paper proposes and implements a QUIC test suite that interacts with public QUIC servers to verify their conformance with critical features of the IETF specification[5].
2. **An Empirical Study of the Cost of DNS-over-HTTPS:** In this paper, the author shows the current DNS-over-HTTPS ecosystem, in terms of the additional security. Then they compare different transports for secure DNS to highlight the improvements DoH makes over its predecessor, DNS-over-TLS[6].
3. **Modified QUIC protocol for improved network performance and comparison with QUIC and TCP:** To achieve maximum throughput and minimum delay, the paper shows modifications in the existing handshaking mechanism of QUIC protocol. This paper, like our project, does research on QUIC protocol performance and introduces a remodeled QUIC (ModQUIC). [7]

Design and Implementation:

In this section, we explain our implementation of different tests and provide details about the methodology we used to evaluate the performance of DNS over different protocols and servers, for different websites.

Our implementation consists of three parts, namely comparison of performances of different DNS servers/resolvers, different protocols used by different DNS servers and different tools used for DNS lookups.

We start with the **dnslookup** tool, implemented by Ameshkov et al[1] in this repository. This tool provides support for DNS lookups over TLS, HTTPS, QUIC, UDP and so on. Our target is to use this tool to perform a DNS lookup for different sites and then compare the bandwidth used and time taken for different protocols. For this, we need DNS resolvers/servers supporting different protocols like TLS and QUIC. There are different servers available with implementations of DNS over different protocols. We look at a few of them in the next part. For this test, we take the DNS servers hosted at Adguard, which support all protocols of our interest. The standard lookup commands over different protocols are as follows:

```
dnslookup example.org 94.140.15.15(Plain DNS(over UDP))

dnslookup example.org tls://94.140.15.15(TLS)

dnslookup example.org https://94.140.15.15/dns-query(HTTPS)

dnslookup example.org quic://94.140.15.15(QUIC)
```

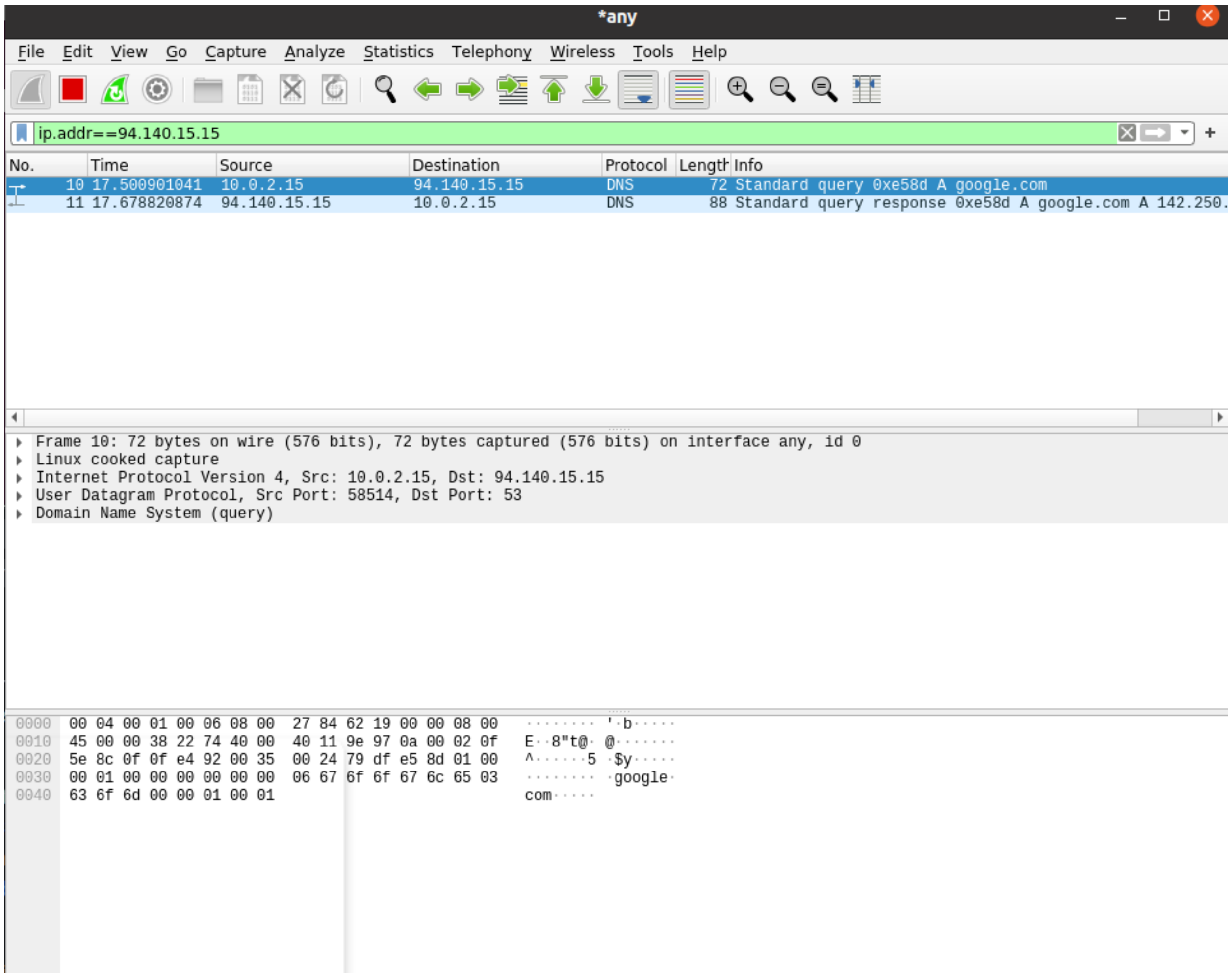
The monitoring of time taken by the query is relatively simple, and we do this using the time command in Linux:

```
time dnslookup example.org 94.140.15.15
```

The next task is to calculate the bandwidth taken up by this query. For this, we revert to a network monitoring tool like netstat or nethogs.

However, there is an issue here. Nethogs monitors bandwidth used by different processes running on the system, but it is not able to detect the queries made by **dnslookup** tool, the reason behind which might stem from the different ways these two tools are implemented.

So, to monitor the bandwidth usage, we switch to Wireshark, and try to capture the packets exchanged between the local ip address and the DNS server we are querying. We add up the size of these packets and divide it by the system time reported by the time command to get the bandwidth used:


*any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
ip.addr==94.140.15.15
```

```
No.   Time          Source       Destination    Protocol Length Info
   10 17.500901041  10.0.2.15    94.140.15.15   DNS        72 Standard query 0xe58d A google.com
   11 17.678820874  94.140.15.15 10.0.2.15      DNS        88 Standard query response 0xe58d A google.com A 142.250.
```

```
Frame 10: 72 bytes on wire (576 bits), 72 bytes captured (576 bits) on interface any, id 0
Linux cooked capture
Internet Protocol Version 4, Src: 10.0.2.15, Dst: 94.140.15.15
User Datagram Protocol, Src Port: 58514, Dst Port: 53
Domain Name System (query)
```

```
0000  00 04 00 01 00 06 08 00  27 84 62 19 00 00 08 00   ········ '·b·····
0010  45 00 00 38 22 74 40 00  40 11 9e 97 0a 00 02 0f   E··8"t@· @·······
0020  5e 8c 0f 0f e4 92 00 35  00 24 79 df e5 8d 01 00   ^······5 ·$y·····
0030  00 01 00 00 00 00 00 00  06 67 6f 6f 67 6c 65 03   ········ ·google·
0040  63 6f 6d 00 00 01 00 01                            com·····
```

We perform this experiment for over 30 sites, taken from Amazon's Alexa's top sites, and observe the trends.

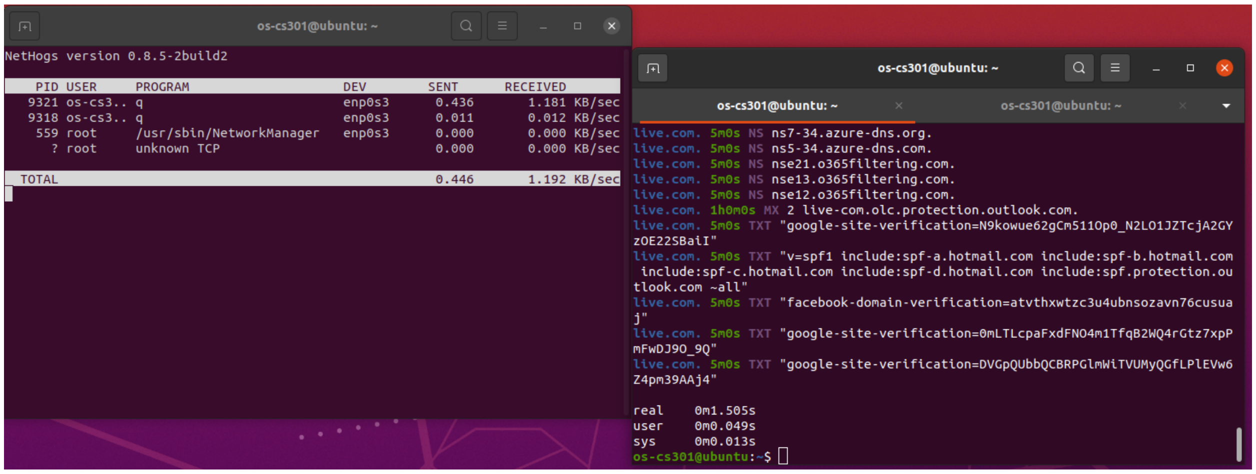
Next, we move on to a different tool, namely q, implemented by Nate Sales in [2]. *Q* is a "tiny CLI DNS client library with support for UDP. DoT, DoH and DoQ" as the official description reads. We reproduce the tests implemented above for different websites using the q tool, using the following commands:

```
q example.org 94.140.15.15(Plain DNS(over UDP))

q --dnssec example.org tls://94.140.15.15(DNSSEC)

q example.org https://94.140.15.15/dns-query(HTTPS)

q example.org quic://94.140.15.15(QUIC)
```

The monitoring of time taken by the query is done as before, using the time command before the query:

```
time q example.org quic://94.140.15.15(QUIC)
```

Next, we have to measure the bandwidth used by this tool for different protocols. We do this using the nethogs network monitoring tool. We do not need to use Wireshark as nethogs is able to detect the q tool, unlike dnslookup:



```
NetHogs version 0.8.5-2build2

  PID USER     PROGRAM                      DEV        SENT      RECEIVED
 9321 os-cs3.. q                            enp0s3     0.436      1.181 KB/sec
 9318 os-cs3.. q                            enp0s3     0.011      0.012 KB/sec
  559 root     /usr/sbin/NetworkManager     enp0s3     0.000      0.000 KB/sec
    ? root     unknown TCP                             0.000      0.000 KB/sec

 TOTAL                                                 0.446      1.192 KB/sec
```

```
live.com. 5m0s NS ns7-34.azure-dns.org.
live.com. 5m0s NS ns5-34.azure-dns.com.
live.com. 5m0s NS nse21.o365filtering.com.
live.com. 5m0s NS nse13.o365filtering.com.
live.com. 5m0s NS nse12.o365filtering.com.
live.com. 1h0m0s MX 2 live-com.olc.protection.outlook.com.
live.com. 5m0s TXT "google-site-verification=N9kowue62gCm511Op0_N2LO1JZTcjA2GY
zOE22SBaiI"
live.com. 5m0s TXT "v=spf1 include:spf-a.hotmail.com include:spf-b.hotmail.com
 include:spf-c.hotmail.com include:spf-d.hotmail.com include:spf.protection.ou
tlook.com ~all"
live.com. 5m0s TXT "facebook-domain-verification=atvthxwtzc3u4ubnsozavn76cusua
j"
live.com. 5m0s TXT "google-site-verification=0mLTLcpaFxdFNO4m1TfqB2WQ4rGtz7xpP
mFwDJ9O_9Q"
live.com. 5m0s TXT "google-site-verification=DVGpQUbbQCBRPGlmWiTVUMyQGfLPlEVw6
Z4pm39AAj4"

real    0m1.505s
user    0m0.049s
sys     0m0.013s
os-cs301@ubuntu:~$ 
```

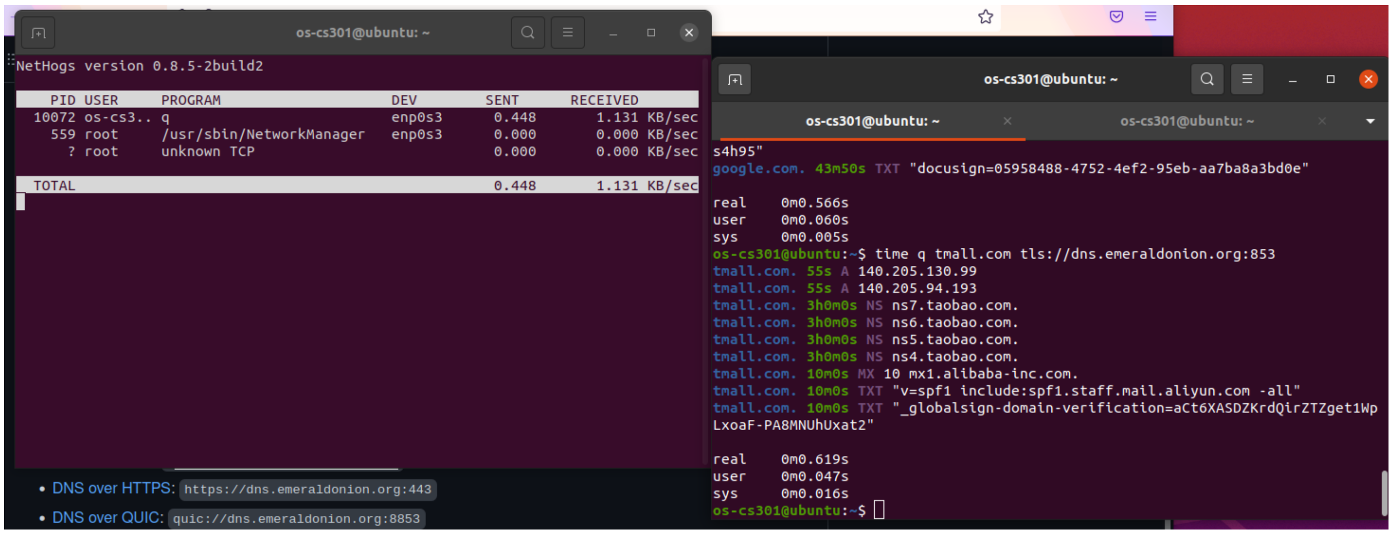
This concludes the tests for the q tool.

Now, we move on to the next phase of our project, which is comparison of performances of different DNS resolvers/ servers. For this, we needed to find public servers supporting different protocols like TLS and QUIC. Since QUIC is still relatively new, not many publicly deployed servers support it officially and are still in the Beta testing phase.

However, we did manage to find two servers other than Adguard which do support QUIC, namely EmeraldOnion[3] and Arapurayil[4]. We repeat the same tests for these two servers as done for Adguard. We use the q tool and nethogs to monitor the bandwidth. The commands for EmeraldOnion queries are as follows:

```
time q example.org tls://dns.emeraldonion.org:853(TLS)

time q example.org https://dns.emeraldonion.org:443(HTTPS)

time q example.org quic://dns.emeraldonion.org:8853(QUIC)
```

The below figure shows the bandwidth measurement for the EmeraldOnion server using nethogs tool:



```
NetHogs version 0.8.5-2build2

  PID USER     PROGRAM                      DEV        SENT      RECEIVED
10072 os-cs3.. q                            enp0s3     0.448       1.131 KB/sec
  559 root     /usr/sbin/NetworkManager     enp0s3     0.000       0.000 KB/sec
    ? root     unknown TCP                             0.000       0.000 KB/sec

  TOTAL                                                0.448       1.131 KB/sec
```

- DNS over HTTPS: `https://dns.emeraldonion.org:443`
- DNS over QUIC: `quic://dns.emeraldonion.org:8853`

```
s4h95"
google.com. 43m50s TXT "docusign=05958488-4752-4ef2-95eb-aa7ba8a3bd0e"

real    0m0.566s
user    0m0.060s
sys     0m0.005s
os-cs301@ubuntu:~$ time q tmall.com tls://dns.emeraldonion.org:853
tmall.com. 55s A 140.205.130.99
tmall.com. 55s A 140.205.94.193
tmall.com. 3h0m0s NS ns7.taobao.com.
tmall.com. 3h0m0s NS ns6.taobao.com.
tmall.com. 3h0m0s NS ns5.taobao.com.
tmall.com. 3h0m0s NS ns4.taobao.com.
tmall.com. 10m0s MX 10 mx1.alibaba-inc.com.
tmall.com. 10m0s TXT "v=spf1 include:spf1.staff.mail.aliyun.com -all"
tmall.com. 10m0s TXT "_globalsign-domain-verification=aCt6XASDZKrdQirZTZget1Wp
LxoaF-PA8MNUhUxat2"

real    0m0.619s
user    0m0.047s
sys     0m0.016s
os-cs301@ubuntu:~$
```

We carry out the tests for all the websites we considered for the Adguard server tests.

This concludes the implementation part of our project. Next, we move on to the Evaluation section and observe the results obtained from the above tests.

Evaluation:

The evaluations carried out by us are three fold. In the following sections, we describe the performance results obtained for three different aspects of our project: Protocols, tools and servers

1. **Comparison of Different Protocols:**
   The first evaluation phase compares the performance of different protocols used by the DNS resolvers at the different servers, namely TLS, HTTPS, QUIC and so on. We do this by comparing the bandwidth used by the q tool(and the dnslookup tool) when querying the DNS resolver implementing a specific protocol. Figure 1 below shows the results obtained for the Adguard Server **using the dnslookup tool**:

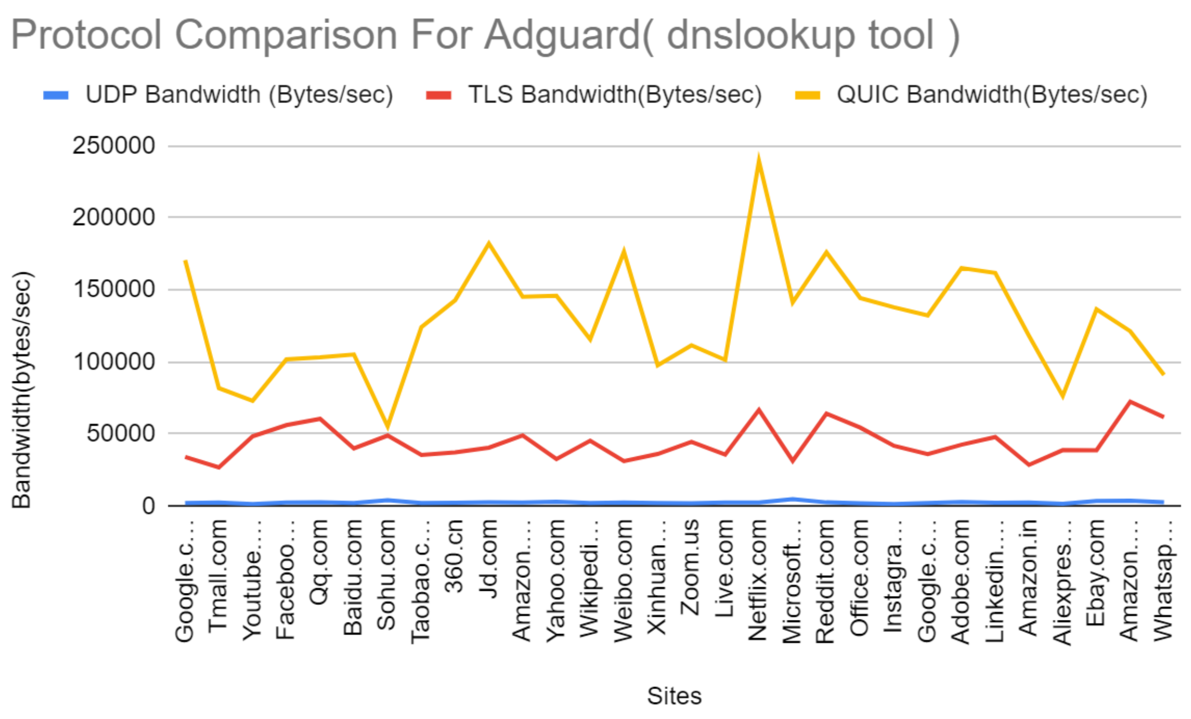


Figure 1:Protocol Comparison(Adguard)

We can see that the bandwidth for DNS over udp is the lowest, which is also expected because there are minimal overheads involved when exchanging information over udp. The bandwidth increases for TLS, because it is implemented over TCP and is aimed to be more secure, so there are overheads like handshaking and secure information exchange involved. The largest bandwidth observed is for QUIC, which seems to be an attribute related to the implementation of the **dnslookup** tool itself, (and maybe a little overhead added on Wireshark), because the same comparison using the q tool(as we will see below) gives different results, wherein QUIC's graph lies close to the TLS one.

Next, we do the same test for the same server, but **using the q tool**. Figure 2 shows the results obtained:

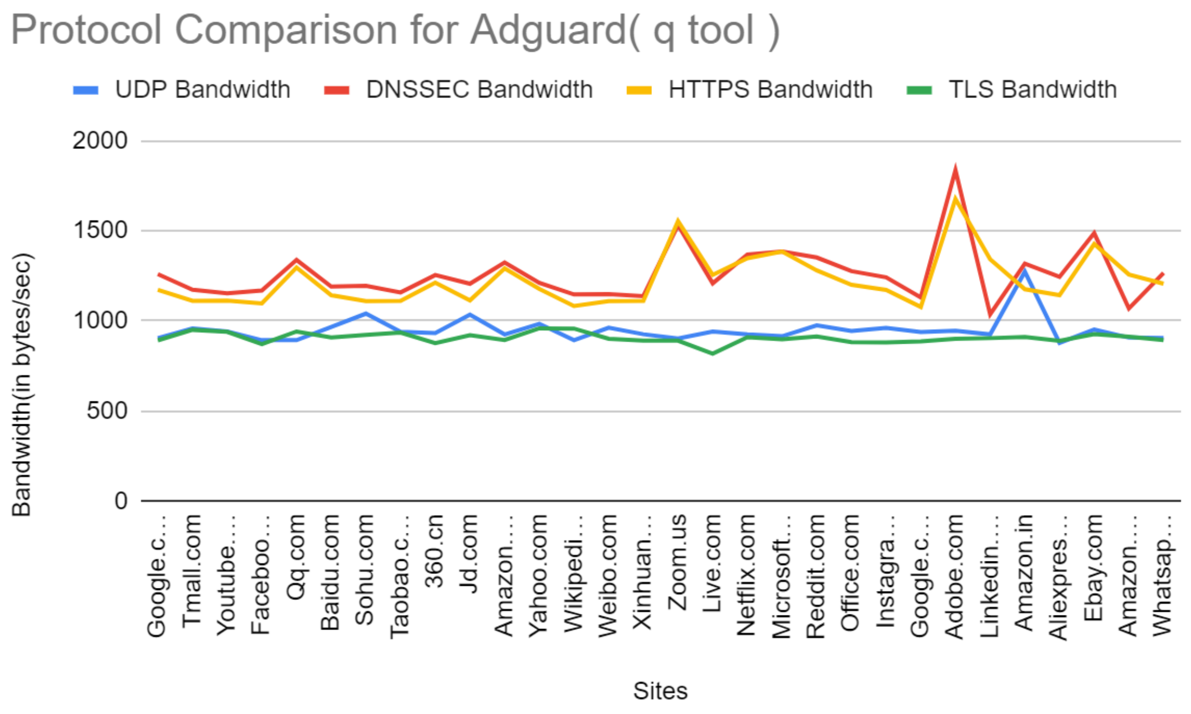


Figure 2: Protocol Comparison for Adguard(using q tool)

Here, we look at the bandwidth corresponding to different protocols for the Adguard Server as measured by the q tool. We observe that the lowest bandwidth is achieved for the udp lookup, and it increases as we move to HTTPS and DNSSEC due to overheads incurred by increased security.

2. **Server Comparison:**
Our next evaluation consists of comparison of performance of different DNS resolvers/servers for different protocols. We carried out our tests on three different servers, namely the Adguard server[1], the EmeraldOnion server[3] and the Arapurayil server[4]. Figure 3 below compares the performance in terms of bandwidth for the TLS protocol using the q tool:

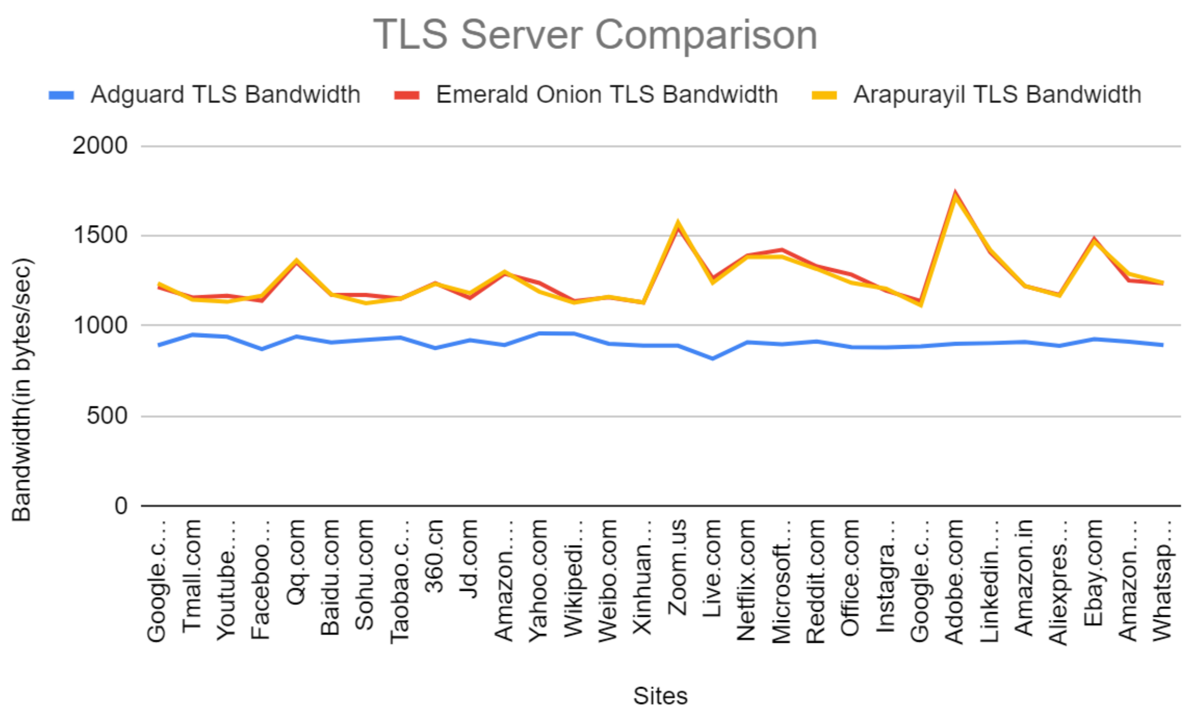


Figure 3: TLS Server Comparison

We note that the lowest bandwidth used is for the Adguard server, while that for Emerald Onion and Arapurayil is almost the same.

Figure 4 shows the performance comparison in terms of bandwidth for the HTTPS protocol using the q tool:

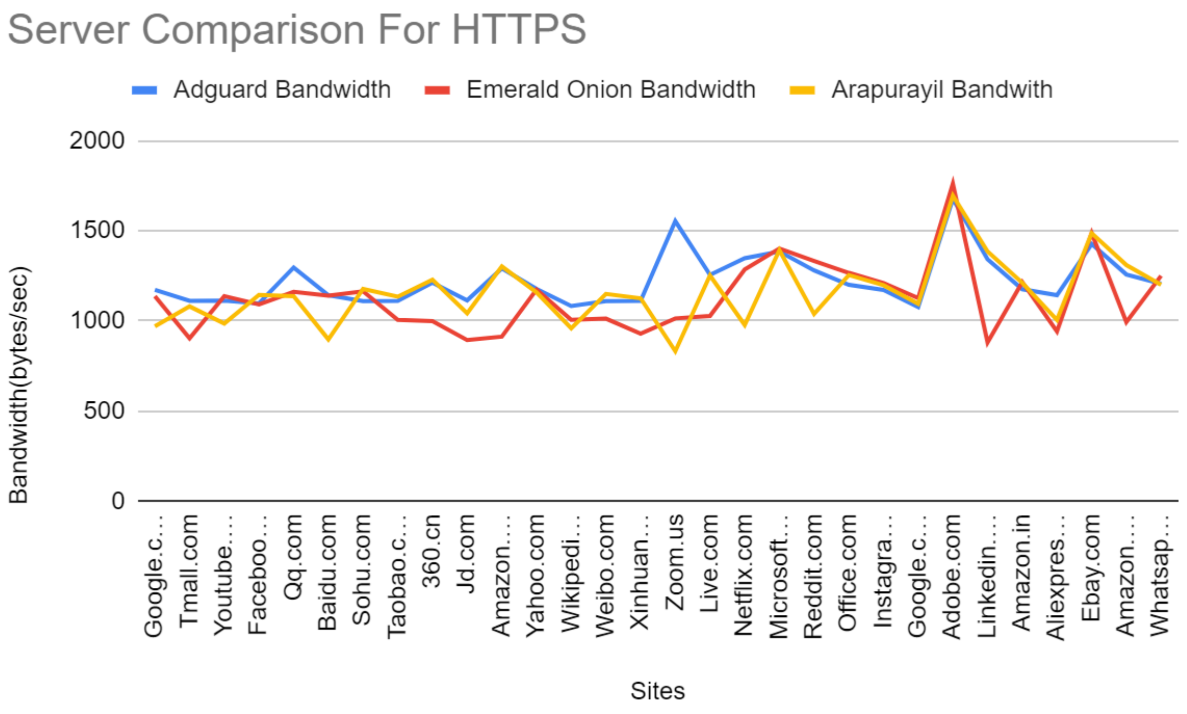


Figure 4: Server Comparison For HTTPS

Here, all the three servers give almost the same results.

Finally, figure 5 shows the performance comparison in terms of bandwidth for the quic protocol for different servers using the q tool:

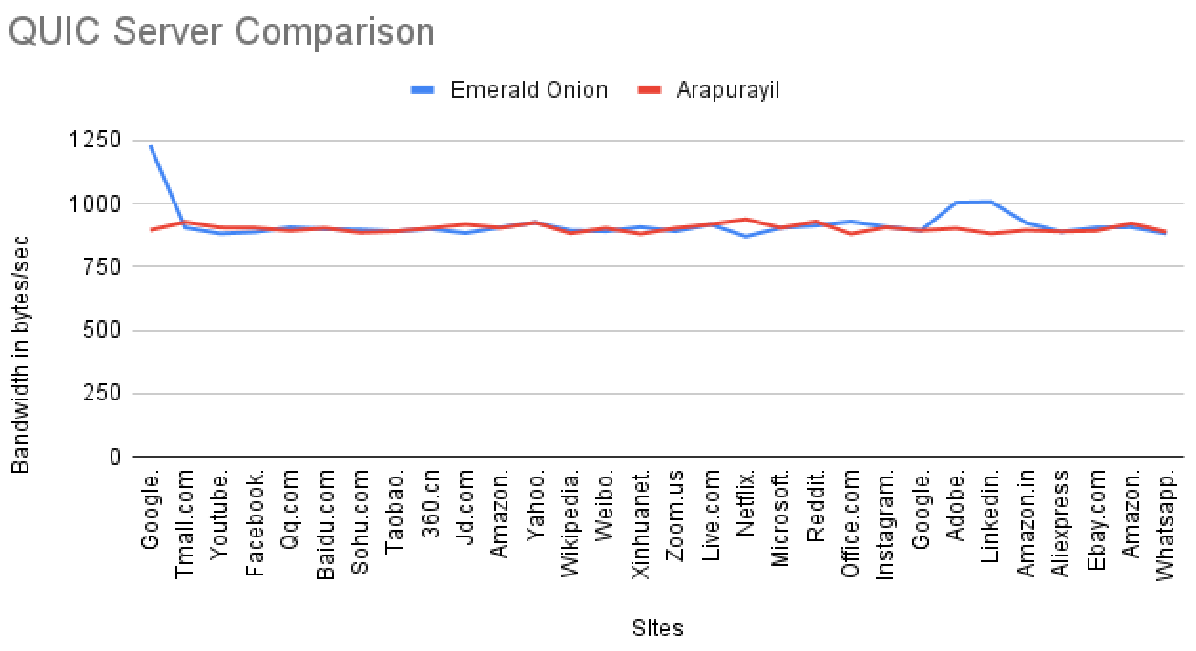


Figure 5: QUIC Server Comparison

We can see the bandwidth used for querying over QUIC is almost the same for these two servers.

3. **Tool Comparison:**
   As a final step of our evaluation, we do a tool comparison between the dnslookup tool and the q tool. We select the Adguard server and the UDP protocol for comparison. Figure 6 below shows the results obtained:

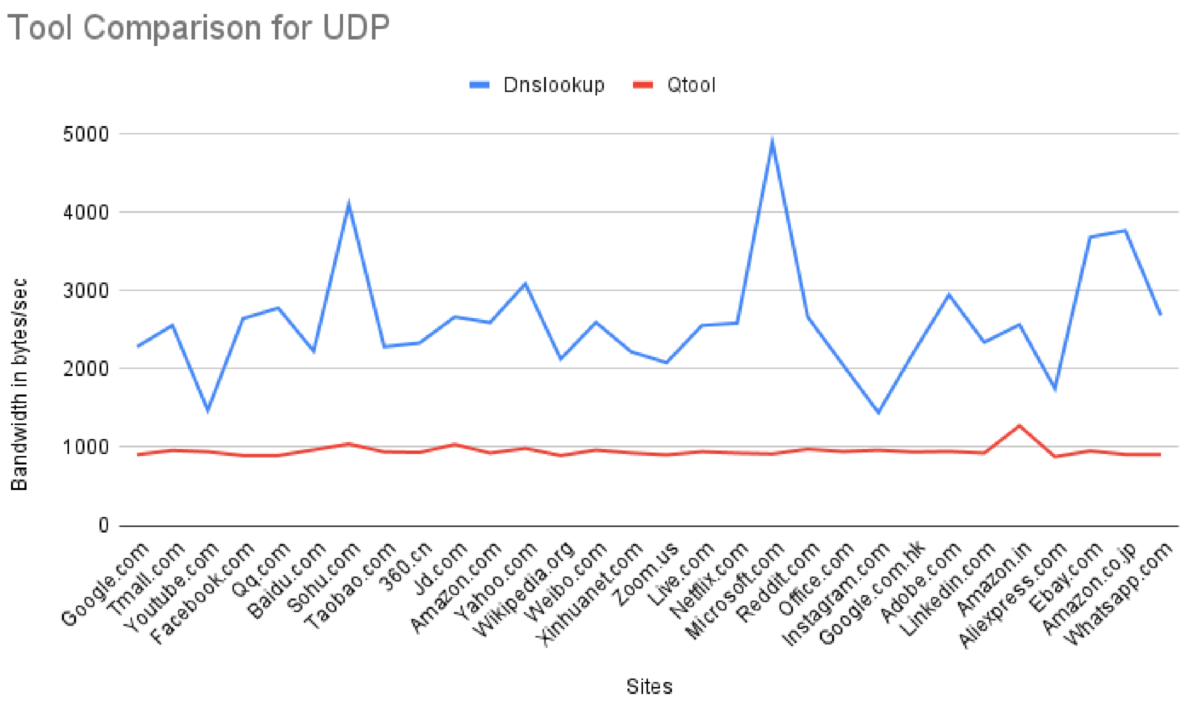


Figure 6: Tool Comparison for UDP

We see that the q tool uses much less bandwidth as compared to the dnslookup tool.

Conclusions:

DNS was developed as an unencrypted protocol and is now one of the most significant protocols for many networked applications. Concerns regarding user privacy have prompted more secure ways to be proposed. In this paper, we have surveyed the different protocols over which DNS servers are implemented and compared their performance in terms of bandwidth. We did a comprehensive study for over 30 sites and compared the results. We then moved on to different DNS servers and compared their performance. Finally, we also did a comparison of two different DNS lookup tools.
On an overall basis, the q tool provides better results as compared to the dnslookup tool. Overall, we conclude that TLS, DNSSECand HTTPS take up more bandwidth as compared to simple UDP, due to security and handshaking overheads involved. The behaviour of QUIC is somewhat irregular, and depends highly on the tool and the hosting server.

Limitations:

- We were not able to run a larger dataset via code for a better result.
- The tools used by us did not support all the protocols.
- The tools used were not very popular and hence had very few alternatives.

Future work:

QUIC is a new network protocol that resides in the application layer over UDP. QUIC was initially developed as an alternative for TCP. We were able to obtain the analysis of DNS over quic in different servers and protocols.

- With the collected information, the security analysis of quic should be the way forward.
- We can design new tests to measure fairness when sharing bandwidth with other QUIC/TCP flows.
- Evaluate advantages over loading HTTP pages.

References:

1. https://github.com/ameshkov/dnslookup
2. https://github.com/natesales/q
3. https://github.com/emeraldonion/DNS
4. https://github.com/arapurayil/dns
5. Piraux, Maxime, Quentin De Coninck, and Olivier Bonaventure. "Observing the evolution of QUIC implementations." In Proceedings of the Workshop on the Evolution, Performance, and Interoperability of QUIC, pp. 8-14. 2018.
6. Böttger, Timm, Felix Cuadrado, Gianni Antichi, Eder Leão Fernandes, Gareth Tyson, Ignacio Castro, and Steve Uhlig. "An Empirical Study of the Cost of DNS-over-HTTPS." In Proceedings of the Internet Measurement Conference, pp. 15-21. 2019.
7. https://www.inderscienceonline.com/doi/abs/10.1504/IJIPT.2019.098489

Contributions:

- Report and Presentation Slides - All members
- Equally divided all implementation part
    - q tool(Adguard)
        - UDP and TLS(Shreyshi), DNSSEC and HTTPS(Ayush)
    - Dnslookup :
        - UDP and TLS(Bhoomika), Quic(Shreyshi)
    - Server: Emerald Onion
        - TLS(Ayush), HTTPS(Bhoomika), Quic(Shreyshi)
    - Sever: Arapurayil
        - TLS(Ayush), HTTPS(Bhoomika), Quic(Shreyshi)