HttpDNS: A Flexible Architecture for Edge Server Exploration and Selection in 5G Network

Yuchao Zhang[1]*, Shuang Wu[1]*, Wendong Wang*
Zhuoyun Zhang†, Yunbo Han†
* *Beijing University of Posts and Telecommunications,* † *Tencent*
Email: {yczhang,wushuangmail,wdwang}@bupt.edu.cn, {zhuoyuzhang,yunbohan}@tencent.com

***Abstract*—With the emerging and development of low-latency applications (such as autonomous driving, SmartCity, telemedicine), low-latency has become one of the key requirements of current network services, especially in edge computing and 5G network. While under the state-of-the-art architecture, the request resolution is provided by Internet Service Provider's localDNS, which assigns a specific edge server for each request, but this method is now suffering from inaccurate scheduling problem, and fails to adapt to network changes due to the relatively fixed allocation strategy. These drawbacks result in the long response latency, and are becoming more severe in nowadays 5G edge networks.**

In this paper, we propose a flexible HttpDNS architecture for 5G edge server exploration and selection, where the domain names of users' requests are resolved by HttpDNS instead of the traditional localDNS. Under such architecture, user's request is directly transmitted to HttpDNS servers, which will then allocate a specific edge server for this request. With a server scoring module, the proposed HttpDNS architecture can finally ensure user requests be guided to the optimal edge server. We implement a prototype of HttpDNS architecture and conduct a series of experiments, the results show that such architecture not only reduces at least 10% latency compared with the traditional localDNS, but also has higher stability when network changes.

***Keywords*-Edge Server; 5G; DNS; HttpDNS;**

I. INTRODUCTION

Recent years have witnessed the prosperity and breakthroughs of 5G network, such as autonomous driving [1], [2], SmartCity [3], telemedicine [4], and so on. There are also three major scenarios [5]] [6] [7] of 5G in the future, widely known as eMBB (enhanced Mobile Broadband), mMTC (massive Machine Type Communications), and uRLLC (Ultra Reliable Low Latency Communication). The key requirements shared by all the application areas are low-latency and high-reliability. 5G research organizations such as the ITU and IMT-2020 Promotion Group have put forward millisecond end-to-end delay requirements [8] [9], raising low-latency as the urgent matter to be solved in 5G network.Edge server is the primary problem for solving low latency [10].

MEC (Mobile Edge Computing) provides the possibility for low latency, which enables servers to be deployed at the edge of the network to provide computing and storage functions for mobile terminals [11] [12]. To be specific, in edge computing, one application deploys it's service among multiple geographically distributed edge servers [13] [14]. These multiple edge application service usually share the same IP address [15] [16], thus when receiving a user request, the system has to assign one of these edge servers to that request [17] [18]. In this process, how to explore and select a particular edge server plays a very important role on the request latency. [19].

Existing network systems usually find an available server through localDNS. such systems are suffering from the following two problems: (1) Inaccurate scheduling. In localDNS, some ISPs directly forward the resolution request to the recursive LocalDNS, making the source IP of the domain name resolution request becomes the IP of other operators. In this case, scheduling made by the load balancing server may possibly fail, and users therefore are scheduled to an inappropriate server. If localDNS continues to be used in the 5G scenario, things become even worse to find a suitable MEC server. (2) Cannot adapt well to network changes. When the user's network status changes, the origin assigned edge server may possibly become invalid, resulting in high access time and severely affecting user experiences.

It is not easy to solve the edge server exploration and selection problem. The most direct way is to modify localDNS, but it would introduce too many modifications to the existing network, and the cost overhead is not acceptable.

Under this system, compared to localDNS, the latency in discovering the server address is probably reduced about 10%.

In this paper, we propose a HttpDNS architecture which solves these problems to a certain extent. HttpDNS is a domain name resolution mechanism that can bypass localDNS. It enables the client to resolve domain name with

The work was supported in part by the National Key R&D Program of China under Grant 2019YFB1802603, the National Natural Science Foundation of China (NSFC) Youth Science Foundation under Grant 61802024, the Fundamental Research Funds for the Central Universities under Grant 2482020RC36, and the CCF-Tencent Rhinoceros Creative Foundation under Grant S2019202.

[1]Yuchao Zhang and Shuang Wu contribute equally to this paper.

the HttpDNS server instead of the traditional localDNS, thereby avoiding problems such as domain name hijacking and inaccurate scheduling. we use HttpDNS to replace the original traditional localDNS in the proposed flexible HttpDNS system, and then further design a domain name testing model and a evaluation model. At last, we implement a prototype of our system and integrate a customized SDK to the terminal end, then conduct a series of evaluations, the results show that compared with localDNS scenarion , user request latency is probably reduced by at least 10% by assigning better edge servers.

The contributions of this paper are summarized as follows:

- We designed a flexible HttpDNS based on IP scores, which successfully solves the inaccurate scheduling and network changing problem.
- We implement a prototype and develop a software SDK to evaluate the proposed architecture, and the results show that this architecture not only reduces at least 10% latency compared with the traditional localDNS, but also enjoys higher stability when network changes.

The contents of this paper are arranged as follows: Section I briefly introduces the importance of server discovery and some existing problems, Section II introduces traditional solutions and some other existing schemes, Section III presents the background and motivation of this work, Section IV provides the detailed design scheme of the system , Section V evaluates the proposed system by comparing the performance with traditional methods, and Section VI concludes this paper.

II. Related work

A. Traditional LocalDNS

The traditional LocalDNS query is shown in Figure 1. The client's domain name resolver checks whether there is a domain name mapping relationship between the local cache and the host file. If there is, the IP address mapping process is called to complete the resolution; if there is no corresponding URL, the local resolution server will initiate a recursive query request to the preferred DNS server set in the TCP/IP parameters (called Local DNS server) [20]. If the domain name is successfully resolved by the LocalDNS, the resolution result will be returned to the client.If the LocalDNS server cannot directly resolve the query but has cached the corresponding URL mapping, then the IP address mapping process is called to complete the domain name resolution, such resolution is not authoritative. If the localDNS resolutions are unfortunately all invalid, the query is performed according to the settings of the localDNS server (whether recursive or not). In other words, if the localDNS server is set to be in the open mode, the localDNS server sends the request to one of the 13 Root DNSs. While when the server is set to be in the recursive mode, the localDNS server will forward the request to the upper-level DNS server

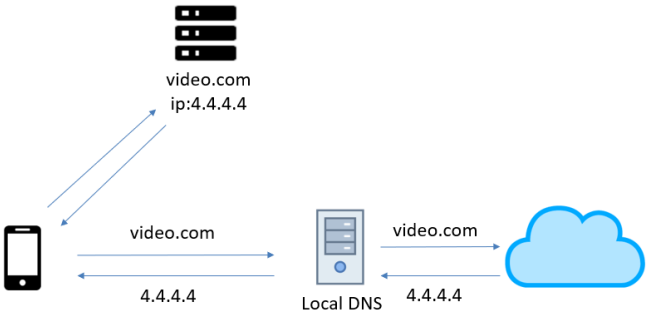


Figure 1: Traditional localDNS

for resolution [21]. When the upper-level server finish the resolution, the corresponding IP address is finally obtained. DNS query is also divided into recursive query and iterative query. Recursive query: if the local domain name server inquired by the host does not know the IP address of the domain name being queried, then the local domain name server, as a DNS client, continues to send query requests to other root domain name servers instead of letting the host do its own the next query. Iterative query: when the root domain name server receives the iterative query request message from the local domain name server, it either gives the queried IP address or tells the local domain name server which domain name server should query next. Then the local name server perform subsequent queries instead of performing subsequent queries for the local name server. It can be seen that from the client to the Local DNS server, the local DNS and the upper-level DNS server are recursive queries. The DNS server and the root DNS server were previously iterative queries. In real cases, because the use of recursive mode will cause a lot of DNS server traffic, most of the DNS is now iterative mode.

Such localDNS has many problems, for example, the whole resolution process is complex and inefficient. Furthermore, the fixed resolution mode cannot work well in time-varying 5G network scenarios.

B. HttpDNS

Recently, new resolution methods are getting more and more attention, A new architecture proposed by Tencent can be shown as Figure 2, such HttpDNS-based methods can be generally described as follows: end users initiate a domain name resolution request which carry the domain name to be resolved and it's IP address; the HttpDNS server make the resolution of this request by analyzing its IP address and the other attributes, then the HttpDNS server returns the resolved domain name to the terminal.Such HttpDNS-based methods are not only simple, but also can improve the efficiency of domain name resolution, because the HttpDNS servers can be flexibly reconfigured. As a result, the terminal can quickly access to the required addresses,which therefore can improve the Quality of Service (QoS) of 5G services.

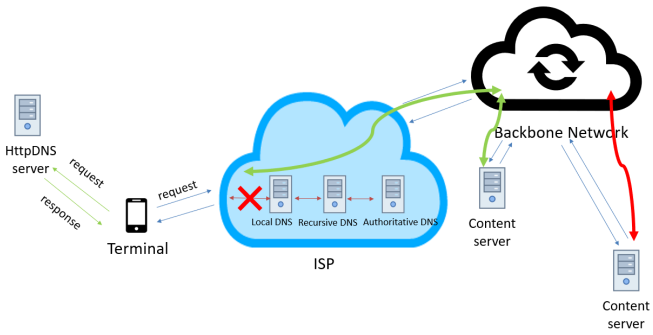


Figure 2: Over view of HttpDNS

The above Http-based DNS methods can address the following two challenges:

Local DNS hijacking: Because HttpDNS directly requests HTTP to obtain the server A record address through IP, there is no process of querying the domain resolution with the local ISP, so the problem of hijacking is avoided at all.

User connection failure rate declines: algorithms are used to reduce server rankings with high failure rates in the past, server rankings are improved through data that has been accessed recently, and server rankings are improved through historical access success records.

C. Other solutions

There are also some other solutions to the abnormal domain name resolution and inaccurate scheduling problem [22], [23].

1) Real-time monitoring business promotion: but this kind of schemes are still haven't been landed because there is no motivation for ISPs to do so. this kind of scheme has a relatively long period. After all, it is time-consuming to push ISP to solve this problem through administrative means.

2) To bypass automatic DNS allocation, use 114dns or Google public DNS: Although both of 114dns and Google are large DNS with seldom traffic scheduling failures, some serious problems are still up-in-the-air:

How to construct a domain name request on the user side: For a PC client, it is not difficult to construct a standard DNS request packet. While for a mobile client, the compatibility cost will be very high to make the various packets generated from different operating systems (such as ISO and Android) compatible to a specific LocalDNS.

Promoting users to modify the configuration is extremely hard: Promoting PC users to update/modify the DNS configuration under WiFi is barely feasible, while asking mobile users to modify the DNS configuration is still infeasible.

3) Abandon the domain name completely, and build a connect-center to schedule traffic: This kind of solutions can only be established with a fully access to an accurate IP address database. A connect center is set up to make scheduling and do the access layer dispatch. This scheme has the same drawback with the second schemes, that is the unacceptable high cost, especially for large-scale deployment.

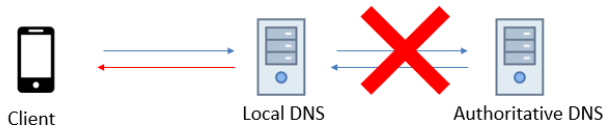


Figure 3: Domain hijacking

III. Background and Motivation

A. Drawbacks of Traditional HttpDNS

Domain hijacking: There are great differences in bandwidth resources, inter network settlement fees, IDC room distribution and intra network ICP resources distribution among domestic Internet access ISP [24]. In order to ensure the access quality of users in the network and reduce cross network settlement, ISP set up a content cache server in the network [25]. By forcibly pointing the domain name to the IP address of the content cache server, the purpose of leaving the local network traffic completely in the local area is realized. When the local DNS caches the resolution results of domain names, it does not recurse to the authoritative DNS, and the local DNS is attacked by a third party, or the cache content is tampered with, or some local DNS will cache the content pointed to by some domain name resolution results, and replace it with the advertisement of the third party advertising alliance, which will result in domain name hijacking as Figure 3, making users access the wrong resources.

Inaccurate scheduling: In addition to the domain name cache, the ISP's local DNS also has the phenomenon of Forwarding domain name resolution [26]. Forward domain name resolution refers to the behavior that ISP do not perform domain name recursive resolution, but forward domain name resolution requests to recursive DNS of other ISP. In order to save resources, some small ISP directly forward the resolution request to the recursive local DNS of other ISP. As a direct result, the source IP of domain name resolution request received by authoritative DNS becomes the IP of other ISP, which eventually leads to user traffic being directed to the wrong IDC and user access becoming slower, as shown in Figure 4.

Poor response to changes in the network environment: When a user moves across domains or changes the network status, the original use of IP will greatly reduce the user's experience. In this case, the optimal IP needs to be replaced in time [27]. As shown in the Figure 5, after the network status changes, most of the current used IP rating will drop.

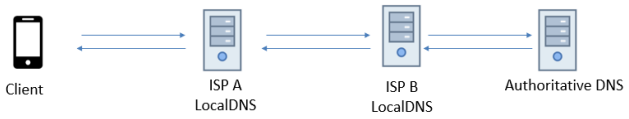


Figure 4: Forward domain name resolution

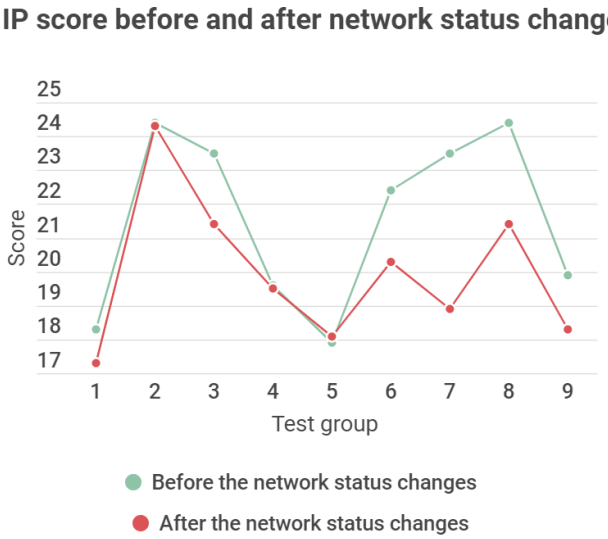


Figure 5: IP score before and after network status change

B. Potential Solutions

1) Motivation: We deeply studied the edge server discovery problem and point out the key procedure.

The essential issue of 5G edge server exploration and selection is to help the UE finds the right edge server.

While traditional DNS interaction that is based on ISP LocalDNS is suffering from inaccurate scheduling and poor response to network change problem, the way of designing an intelligent domain name resolution HttpDNS server seems help, because it can not only obtain the information of 5G backend servers but also be easily re-configured to keep up to date, so that this method can improve the accuracy of 5G edge server allocation. In addition, because the DNS server acquires the real client IP instead of the localDNS IP, it can accurately locate the client's geographic location, thereby effectively improving the accuracy of IP scheduling. According to the load information updated by all edge application servers in the geographic locations and range of the UE, the corresponding load balancing algorithm can be used to resolve the best edge server address. The UE application can use the edge application server address to directly communicate with its edge server. For other non-MEC applications, the DNS server of the edge application server forwards its DNS query to other DNS servers on the Internet.

Consider scenarios (if any) for which the UE needs to be aware that there is an application server in the Edge Hosting Environment and scenarios (if any) for which the UE does not need to be aware that there is an application server in the Edge Hosting Environment. Such scheme can be divided into two cases to adapt to different scenarios.

To make the above HttpDNS scheme practical, we further discussed the information that can be used to assist server discovery problem. Here we adopt the user's geographic location, the amount of data offloaded by the UE task, the amount of calculation, delay requirements, the current network transmission rate, etc.

After that, the remaining problem is how to allocate a specific edge server to a particular request. There are also many influence factors such as the relative geographic location of the MEC where the EAS is located, the clock frequency of the EAS, the data transmission rate of the EAS and the central cloud data, the remaining capacity of the EAS, the load of the EAS, etc.

Along with this solution, the 5G edge server discovery problem can be solved under static network environment, but things are more difficult under dynamic and time-varying network scenarios. how to make edge application server discovery when the last edge application server becomes unoptimized or unavailable? The simplest way is to initiate a second DNS resolution to the edge application server and update the current edge application server status, return the result to the UE to connect to the new edge application server, or select a secondary superior edge application server that can initiate a secondary DNS resolution if the suboptimal is still unavailable.

2) Challenge: To realize the above HttpDNS scheme, there are several challenges.

How to get the real IP of the terminal: Only when the real IP of the requester is obtained, can the dispatcher perform accurate scheduling. In order to save resources, some ISP will directly forward the resolution request to the recursive LocalDNS of other ISP, causing the source IP of the received domain name resolution request to become the IP of other ISP. So it is hard to get the terminal IP is some cases.

How to adapt the network changes:: When the user's network status changes, or the current server load is too high, the original user's IP experience will be greatly reduced, but localDNS will not respond to this, because it can not get the feedback of user's experience.

To solve the above challenges, we propose a flexible HttpDNS scheme which can make accurate edge server selection and quickly adapt to network changes. The details are explained in the next section.

IV. System Design

We propose a flexible HttpDNS system, which consists of two key modules, the HttpDNS local library, IP evaluation

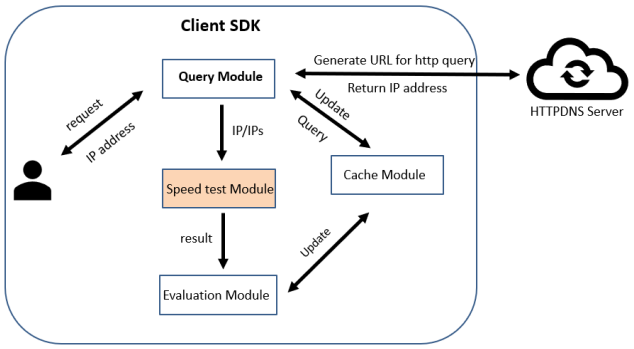


Figure 6: The proposed HttpDNS system

Algorithm 1: Total score

Input: Impact N factors: α, β, γ, δ, ϵ, ε
Output: Total score S

```
S = 0 ;
Initialize each weight coefficient : θ, ϑ, ι, κ, λ, μ ;
for each factor do
    if factor is true then
        Get the coefficient of the factor ;
        S += factor * coefficient ;
    else
        continue ;
    end
end
Return S ;
```

and scoring system, which ensure that the user's tasks are offloaded to the optimal MEC server even in the time-varying network conditions. To cope with the HttpDNS server, we also deploy a client SDK, which can be divided into four modules: domain name query module, domain name cache module, domain name test module, and domain name evaluation module. Each module performs data interaction through a RESTful interface, and configures traditional localDNS as a back up scheme. The framework of this system is shown in Figure 6:

1) Domain name query module. This module intercepts the domain name resolution requests from mobile devices, extracts the domain name to be accessed and combine it into an Http query URL. The query module first query the local cache module, and will initiate a query to the HttpDNS server if it receive no response from the caching module. In a word, this module is responsible for receiving and processing the user data.

2) Domain name caching module. The local cache is updated according to the HttpDNS domain name resolution results. It is responsible for maintaining the existing domain name resolution results, so this module is responsible for updating the local cache by interacting with the domain name evaluation module.

3) Domain testing module. According to the test of the domain name resolution results returns the corresponding report on server quality to the domain name evaluation module , so that the evaluation module can decide whether to update the local caching module.

4) Domain evaluation module. The current domain name is evaluated according to the experience report returned by the domain name testing module, and the cache module is updated according to the evaluation result. After testing a group of IPs, the test module returns the results to the evaluation module. The evaluation module is based on Algorithm 1: This speed test: specific data for each IP speed test of the IP group, represented by α. HttpDNS server default priority: the priority returned in the HttpDNS interface A record.represented by β. History Success: the number of successful visits within 7 days of the IP, represented by γ. Historical errors: the number of access errors in the IP within 7 days, represented by δ. Last success time: the last time the ip last succeeded from the current time, in hours, represented by ϵ. Historical average throughput: the average throughput of the IP within 7 days, represented by ε.

The core workflow of this system can be described as follows:

Step 1: During domain name resolution process, the client SDK intercepts the domain name resolution request of the corresponding application, extracts the domain name, and sends it to the domain name query module.

Step 2: The domain name query module first queries whether there is a domain name resolution result available in the domain name IP caching module. if yes, it directly returns the corresponding domain name resolution result to the user, and simultaneously starts the domain name test module; while if there is no available results, configure the corresponding HttpDNS server address and combine Url to initiate a Http connection to HttpDNS.

Step 3: When receiving a request from user, the HttpDNS server returns the corresponding one IP or a IP group to the user according to the IP and other relevant information in the Http request.

Step 4: After obtaining the domain name resolution result returned by the HttpDNS server, the domain name query module first updates the corresponding local domain name IP cache module, and simultaneously passes the IP or IP group to the testing module. If there is only one IP, it is directly passed to the user. Otherwise, each IP is returned to the user according to the default order of that IP group.

Step 5: When the domain name testing module obtains a new domain name resolution result or when the user uses the cached IP, the domain name evaluation module performs a latency test on the IP or IP group of the corresponding domain name, and returns the result to the evaluation module. The domain name IP or IP group is

comprehensively evaluated and updated according to server quality testing report.

Step 6: The domain name IP cache module determines whether the current IP is optimal after each update, and if it is not the optimal one, the caching module will push the optimal IP to the domain name query module, and the domain name cache module updates the currently used IP after receiving the push.

To make it easy to understand, we take the following scenario as an example: Suppose a company equipped with the proposed flexible HttpDNS system and client SDK has also been deployed in terminals. When the users business need to request network resources on the MEC, the client intercepts the requested Url and calls the SDK to perform domain name resolution:

There is no cache locally during the first query, and the domain name query module in the SDK reorganizes the Url: *http://1.2.3.4/d?host=www.example.com&info=someinfo*, initiates a Get request to the HttpDNS server. After receiving the request, the HttpDNS server returns IP group according to the terminal's information is shown as follows:

```
{
  "host": "www.example.com",
  "ips": [
    "11.22.33.44",
    "11.22.33.45"
  ],
  "ttl": 57
}
```

After receiving the returned result, the domain name query module returns the first IP address 11.22.33.44 to the user, and returns the whole set of IP addresses to the test module. At this time, the application will return the IP address returned by the domain name query module. Application replaces the original server IP address with a new one and therefor forms a new URL, using which to initiate the subsequent HTTP requests.

After the testing module tests this group of IPs, the results are returned to the evaluation module. The evaluation module then sends the final score of each IP to the domain name IP cache module. The domain name caching module updates its cache accordingly. The IP 11.22.33.45 has the highest score, then the domain name is returned to the domain name query module, and the domain name query module updates its domain name IP to the user.

The domain name testing module tests and scores the currently used IP from time to time according to the corresponding rules. When the user's network environment changes, such as regions switches or network fluctuations, the priority of the current domain name in the domain name IP caching module drops immediately. Then the domain name IP caching module will push the optimal IP to the domain name query module, thereby improving the user experience.

In the proposed flexible HttpDNS system, a HttpDNS server needs to be built so as to allocate the corresponding IP address to end users. On the user side, a SDK needs to be embedded. Client's resolution method is changed to HttpDNS resolution, and Retaining the localDNS resolution, so that the company and its users can enjoy the HttpDNS service.

V. Evaluation

A. Prototype

This experiment uses the SDK of this system to build a simple demo app. This app uses two methods to obtain some resources. The underlying network framework of the project uses OkHttp, and the third-party open source chart library MPAndroidChart is used to visualize the data statistics in the app. In server side, we use DNSPod [28] public DNS interface to get the IP address. In client, we use Redmi K30 pro to run the demo application, CPU is Snapdragon865 Octa-core Max 2.83Ghz, 8 GB RAM, Android 10.

B. Performance Evaluation

In the experiment, we try to access a series of pictures using both traditional method and the proposed HttpDNS scheme. These resources are distributed in different websites , so the queries for these resources not in the same content server. In the traditional method, user requests are directed through the domain name,i.e., this method uses the traditional localDNS; the second way is to use the proposed HttpDNS, the server side in this experiment uses a third-party public DNS server, DNSPod Interface, use *Http://119.29.29.29/d?dn=DomainName* to simulate HttpDNS access.The results are shown in Figures 7 and 8.

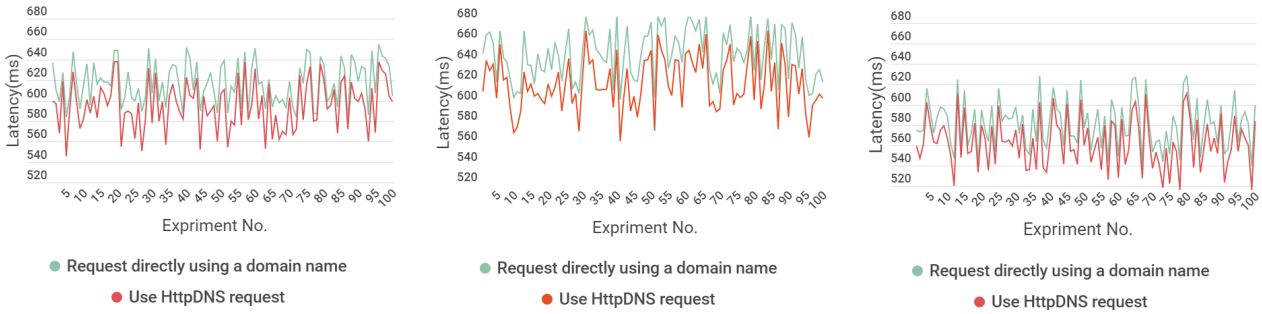
Figure 7 is a comparison of access latency under the two different request methods. We use two different request to get three different resources: image, video and MP3. We can see that using HttpDNS can reduce 18% latency in some cases. The CDF of latency is shown in Fig 10. On the whole, it reduces the overall latency to about 10%.

Figure 8 shows the IP scores of the two task request methods after the network condition changes. It can be seen that when encounter network changes, the allocated edge server IP under HttpDNS scheme enjoys higher scores than that allocated under traditional LocalDNS scheme. The CDF of scores is shown in Fig 11, which shows that the the HttpDNS edge servers enjoy higher scores.

The speedup ratio of tasks compared to localDNS in all simulated tasks is up to 60%. It can be clearly seen that the access method of HttpDNS can improve the task speed.

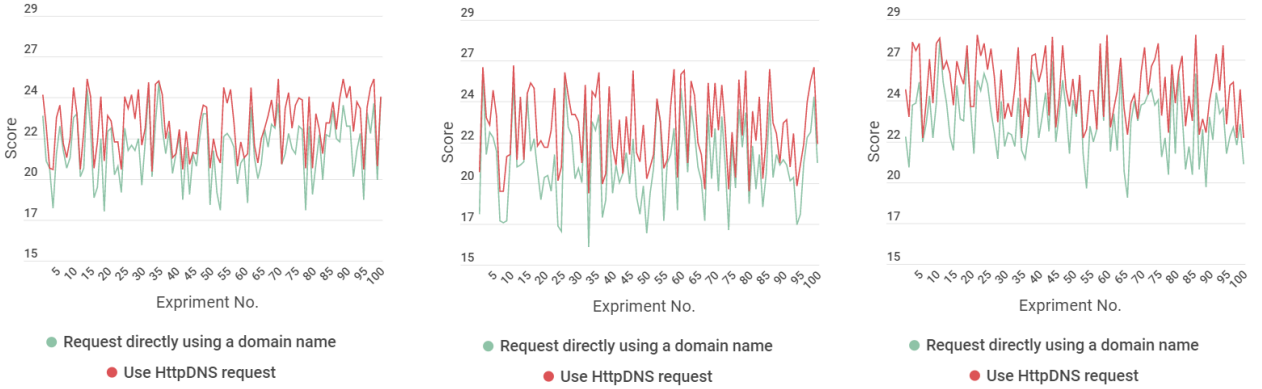
C. System Evaluation

Figure 9 shows the time taken by the client to obtain the domain name resolution results from the server using



(a) Getting image resources (b) Getting video resources (c) Getting MP3 resources

Figure 7: Latency of all task requests (lower is better)



(a) Changing from Wi-Fi to China Mobile (b) Changing from Wi-Fi to China Telecom (c) Changing from China Mobile to China Telecom

Figure 8: IP scores after the network condition changes (higher is better)

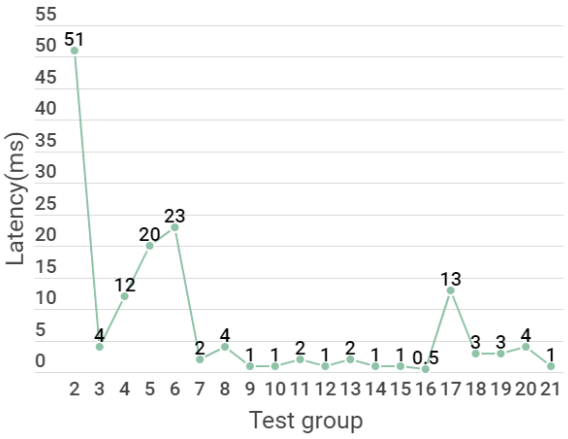


Figure 9: Request latency using HttpDNS

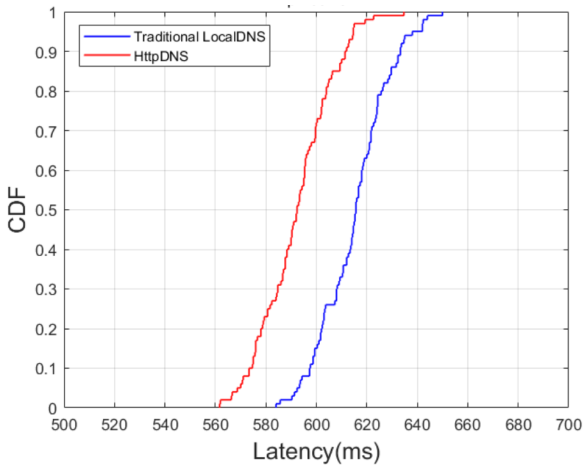


Figure 10: The comparison of CDF latency for request access under different scenarios

HttpDNS. It can be seen that although the time taken in the first packet is relatively long,the latency quickly decreases after several packets. This is because the caching module gradually caches the optimal edge servers. We set some network changes during the experiments and find that the latency will enter some fluctuation to some extend, because the caching module has to update the maintained the edge server IP group. But on the whole, it takes very little time to obtain the domain name resolution results from the HttpDNS server, and even negligible. The fluctuation of this time

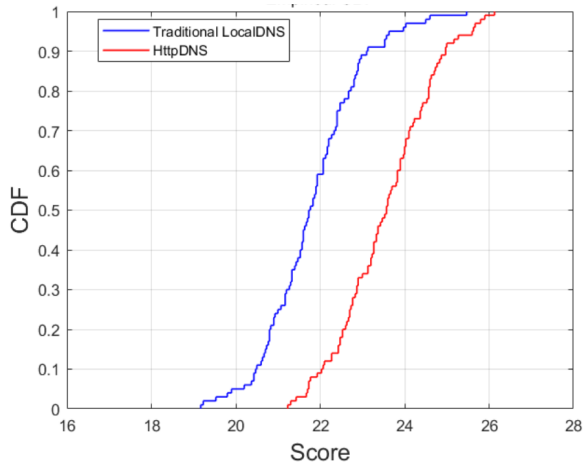


Figure 11: The comparison of average IP score after network condition changes

consumption is relatively stable.

Besides, there are four additional advantages of using the proposed flexible HttpDNS system:

Reduce abnormal domain name resolution: because the local DNS of the ISP is bypassed, user's request to resolve the domain name is transparently transmitted to HttpDNS server that obtains the latest edge server information. So the user's domain name resolution request can get rid of the domain name resolution abnormal troubles;

Accurate scheduling: HttpDNS can directly obtain the user's IP. By combining the IP address database, speed measurement system and scheduling strategy, it ensures that users can be guided to the optimal edge server;

Easy implementation: accessing to HttpDNS services only requires a small amount of modifications to the client access layer, without the need to root or jailbreak the user's mobile phone. Also, the Http protocol request structure is compatible with all versions of mobile operating systems. In addition, the back-end configuration of HttpDNS fully reuses the existing authoritative DNS configuration, so that the management cost is also very low. In short, it solves the problem that the business suffers from domain name resolution anomalies and meets the demand for accurate traffic scheduling of the business with the minimum transformation cost;

Strong scalability: HttpDNS provides a reliable domain name resolution service. The business can combine its own scheduling logic with the results returned by HttpDNS to achieve more refined traffic scheduling. For example, the specified server type that client requests, the specified network type that is required, etc.

On the whole, after implementing the HttpDNS system , the speed of accessing resources will generally increase about 10% compared with the traditional method, and the performance is much more stable when the network environment changes. Although HttpDNS will also bring some additional overhead, such as occupying a certain amount of local storage space and local computing performance, the impact of these overheads is small and negligible.

VI. Conclusion

In this paper, we explores a new method for edge server discovery and selection in 5G environment, called HttpDNS, which bypasses the local DNS of the ISP, and effectively improves the efficiency of domain name resolution even in the dynamic time-varying network conditions. By accurately locating the client's geographic location and obtaining ISP information , HttpDNS scheme can reduce the access latency to about 10% compared with the traditional method. Besides, such flexible HttpDNS method can be accompanied by other information to further improve scheduling accuracy. For the future work, designing reinforcement learning algorithms in the domain name scoring system to dynamically score each domain name, is a promising direction to further improve user experience in the 5G network.

References

[1] Huisheng Ma, Shufang Li, Erqing Zhang, Zheng-nan Lv, Jing Hu, and Xinlei Wei. Cooperative autonomous driving oriented mec-aided 5g-v2x: Prototype system design, field tests and ai-based optimization tools. *IEEE Access*, 8:54288–54302, 2020.

[2] Y. Dai, K. Zhang, S. Maharjan, and Y. Zhang. Edge intelligence for energy-efficient computation offloading and resource allocation in 5g beyond. *IEEE Transactions on Vehicular Technology*, pages 1–1, 2020.

[3] Bo Rong, Shuai Han, Michel Kadoch, Xi Chen, and Antonio Jara. Integration of 5g networks and internet of things for future smart city. *Wirel. Commun. Mob. Comput.*, 2020:2903525:1–2903525:2, 2020.

[4] Sadia Anwar and Ramjee Prasad. Framework for future telemedicine planning and infrastructure using 5g technology. *Wirel. Pers. Commun.*, 100(1):193–208, 2018.

[5] Yajie Li, Yongli Zhao, Jun Li, Jiawei Zhang, Xiaosong Yu, and Jie Zhang. Side channel attack-aware resource allocation for URLLC and embb slices in 5g RAN. *IEEE Access*, 8:2090–2099, 2020.

[6] Jin Cao, Zheng Yan, Ruhui Ma, Yinghui Zhang, Yulong Fu, and Hui Li. LSAA: A lightweight and secure access authentication scheme for both UE and mmtc devices in 5g networks. *IEEE Internet Things J.*, 7(6):5329–5344, 2020.

[7] Ali A. Esswie, Klaus I. Pedersen, and Preben E. Mogensen. Online radio pattern optimization based on dual reinforcement-learning approach for 5g URLLC networks. *IEEE Access*, 8:132922–132936, 2020.

[8] ETSI. *MEC Deployment in 4G and Evolution Towards 5G*, volume No. 24. ETSI White Paper, 2018.

[9] Carlos Renato Storck and Fátima Duarte-Figueiredo. A 5g v2x ecosystem providing internet of vehicles. *Sensors*, 19(3):550, 2019.

[10] ETSI. *Mobile Edge Computing; A Key Technology towards 5G*, volume No. 11. ETSI White Paper, 2016.

[11] Chi-Yu Li, Hsueh-Yang Liu, Po-Hao Huang, Hsu-Tung Chien, Guan-Hua Tu, Pei-Yuan Hong, and Ying-Dar Lin. Mobile edge computing platform deployment in 4g LTE networks: A middlebox approach. In Irfan Ahmad and Swaminathan Sundararaman, editors, *USENIX Workshop on Hot Topics in Edge Computing, HotEdge 2018, Boston, MA, July 10, 2018*. USENIX Association, 2018.

[12] 3GPP. System architecture for the 5g systems. Technical Report 3GPP Standard TS 23.501 V0.4.0, Technical Specification Group Services and System Aspects : Stage 2 (Release 15), 2017.

[13] Xin Chen, Zhiyong Liu, Ying Chen, and Zhuo Li. Mobile edge computing based task offloading and resource allocation in 5g ultra-dense networks. *IEEE Access*, 7:184172–184182, 2019.

[14] Yuchao Zhang, Pengmiao Li, Zhili Zhang, Chaorui Zhang, Wendong Wang, Yishuang Ning, and Bo Lian. Graphinf: A gcn-based popularity prediction system for short video networks. In Wei-Shinn Ku, Yasuhiko Kanemasa, Mohamed Adel Serhani, and Liang-Jie Zhang, editors, *Web Services - ICWS 2020 - 27th International Conference, Held as Part of the Services Conference Federation, SCF 2020, Honolulu, HI, USA, September 18-20, 2020, Proceedings*, volume 12406 of *Lecture Notes in Computer Science*, pages 61–76. Springer, 2020.

[15] Meng Shen, Huisen Liu, Liehuang Zhu, Ke Xu, Hongbo Yu, Xiaojiang Du, and Mohsen Guizani. Blockchain-assisted secure device authentication for cross-domain industrial iot. *IEEE J. Sel. Areas Commun.*, 38(5):942–954, 2020.

[16] Meng Shen, Xiangyun Tang, Liehuang Zhu, Xiaojiang Du, and Mohsen Guizani. Privacy-preserving support vector machine training over blockchain-based encrypted iot data in smart cities. *IEEE Internet Things J.*, 6(5):7702–7712, 2019.

[17] Yuchao Zhang, Pengmiao Li, Zhili Zhang, Bo Bai, Gong Zhang, Wendong Wang, Bo Lian, and Ke Xu. Autosight: Distributed edge caching in short video network. *IEEE Netw.*, 34(3):194–199, 2020.

[18] Liang Lv, Yuchao Zhang, Yusen Li, Ke Xu, Dan Wang, Wendong Wang, Minghui Li, Xuan Cao, and Qingqing Liang. Communication-aware container placement and reassignment in large-scale internet data centers. *IEEE J. Sel. Areas Commun.*, 37(3):540–555, 2019.

[19] Yi Zhu, Kevin Chevalier, Xi Wang, and Nannan Wang. Efficient mobile edge computing for mobile internet of thing in 5g networks. In *53rd Hawaii International Conference on System Sciences, HICSS 2020, Maui, Hawaii, USA, January 7-10, 2020*, pages 1–10. ScholarSpace, 2020.

[20] Mário Almeida, Alessandro Finamore, Diego Perino, Narseo Vallina-Rodriguez, and Matteo Varvello. Dissecting DNS stakeholders in mobile networks. In *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies, CoNEXT 2017, Incheon, Republic of Korea, December 12 - 15, 2017*, pages 28–34. ACM, 2017.

[21] Daniel Moscoviter, Mozhdeh Gholibeigi, Bernd Meijerink, Ruben Kooijman, Paul Krijger, and Geert Heijenk. Improving spatial indexing and searching for location-based DNS queries. In Lefteris Mamatas, Ibrahim Matta, Panagiotis Papadimitriou, and Yevgeni Koucheryavy, editors, *Wired/Wireless Internet Communications - 14th IFIP WG 6.2 International Conference, WWIC 2016, Thessaloniki, Greece, May 25-27, 2016, Proceedings*, volume 9674 of *Lecture Notes in Computer Science*, pages 187–198. Springer, 2016.

[22] Luo Yuchong, Wu Jigang, Wu Yalan, and Chen Long. Task scheduling in mobile edge computing with stochastic requests and m/m/1 servers. In *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 2379–2382. IEEE, 2019.

[23] D. Kaiser and M. Waldvogel. Efficient privacy preserving multicast dns service discovery. In *2014 IEEE Intl Conf on High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Syst (HPCC,CSS,ICESS)*, pages 1229–1236, 2014.

[24] Johann Schlamp, Josef Gustafsson, Matthias Wählisch, Thomas C. Schmidt, and Georg Carle. The abandoned side of the internet: Hijacking internet resources when domain names expire. *CoRR*, abs/1412.5052, 2014.

[25] Jamsheed Manja Ppallan, Karthikeyan Arunachalam, Sweta Jaiswal, Dronamraju Siva Sabareesh, Sungki Seo, and Madhan Raj Kanagarathinam. Flare-dns resolver (FDR) for optimizing DNS lookup overhead in mobile devices. In *16th IEEE Annual Consumer Communications & Networking Conference, CCNC 2019, Las Vegas, NV, USA, January 11-14, 2019*, pages 1–7. IEEE, 2019.

[26] Robert Pettersen, Steffen Viken Valvåg, Åge Kvalnes, and Dag Johansen. Jovaku: Globally distributed caching for cloud database services using DNS. In *2nd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, MobileCloud 2014, Oxford, United Kingdom, April 8-11, 2014*, pages 127–135. IEEE Computer Society, 2014.

[27] H. Wu, S. Deng, W. Li, J. Yin, X. Li, Z. Feng, and A. Y. Zomaya. Mobility-aware service selection in mobile edge computing systems. In *2019 IEEE International Conference on Web Services (ICWS)*, pages 201–208, 2019.

[28] Public dns+:dnspod free intelligent dns service provider https://www.dnspod.cn/products/public.dns, 2020.