QHNS: QoS-aware Hierarchical Name System

Fuhong Lin, Changjia Chen, Hengkui Wu, Hongke Zhang
Electronics and Information Engineering
Beijing Jiaotong University, Beijing, China
06111006@bjtu.edu.cn, changjiachen@sina.com.cn, 06111021@bjtu.edu.cn, hkzhang@bjtu.edu.cn
Ke Xiong
Institute of Information Science
Beijing Jiaotong University, Beijing, China
xiongke.bjtu.iis@gmail.com

Abstract—Naming and name resolution mapping are playing extremely important roles in Internet applications. Currently, naming is constructed by the combination of the location of host and the location of resource in the host, and name resolution mapping system is constructed by a tree-like domain name system (DNS). To overcome the shortcomings of DNS, such as not supporting data migration and replication, vulnerable to Denial of Service (DoS) attacks and not supporting quality of service (QoS), researchers have proposed a DHT-based flat structure to achieve naming and name resolution mapping. This system deals with the shortcomings of DNS above very well except supporting QoS, but it introduces a new problem that the resolution time cost is so large that users often can not tolerate this long delay. In this paper, we present an improved structure called QoS-aware Hierarchical Name System (QHNS) by combining the advantages of DNS and DHT. The architecture of QHNS is a two-layer’s structure, namely top-layer which maintains the global information and bottom-layer which maintains local information. Owning to the location information, the resolution delay is greatly reduced while the shortcomings of DNS have been eliminated. And this design can also do well with the shortcomings of the above two approaches that they can not provide QoS. Finally, theoretical analysis and numerical experiments show that our system is feasible in the practical use.

Index Terms—Naming, name resolution system, DNS, DHT, resolution time delay, QoS

I. INTRODUCTION

Among the applications of Internet, naming and naming resolution system play an important role. Owning to naming, resource registration comes true; owning to naming resolution mapping system, resource discovery comes true. Today, Domain name and Domain name system (DNS) are used as naming and naming resolution mapping system respectively. This system works very well in the past years, but it reveals some shortcomings recently, such as domain name can not support data replication and migration [1], DNS is not robust enough for Denial of Service (DoS) attack [2], [9], [11] and [13] and it does not supporting quality of service (QoS). So many researchers are paying attention to the design of next generation naming and name resolution mapping system.

In [3-6], [10] and [12], authors proposed the scheme in which peer-to-peer scheme is applied to name a resource. They hash the meta-message, which is the smallest representative unit of the resource, to get a string to represent it. This method can deal with the problem of data replication and migration. However, it brings a new drawback that the location information of the resource is lost and this goes against effectively locating it. In [1] and [4], a DHT-based (Distributed Hash Table) name resolution system was proposed. In this way, it can deal with the DoS attack to a certain extent. However, this design brings a new drawback that the time delay of one name resolution is too large and it is not suitable for practical application. Also this design dose not support QoS.

Based on the work above, we propose a new naming and name resolution mapping system called QoS-aware Hierarchical Name System (QHNS). The basic idea of QHNS is to combine the advantages of both DNS and DHT to overcome the shortcomings of them, meanwhile, make some improvement to support QoS for services. In our design, a name is divided in to three parts which are location-part, QoS-part and identity-part. They are used to identify resource location information, QoS levels and resource itself respectively. Our name resolution mapping system is a two-layer hierarchical structure. The top layer maintains a global DHT structure, which is used to register all names and to deal with the resolution requests that the bottom layer can not solve; while in the bottom layer, each country maintains a DHT structure which has two functions 1) dealing with the name registration and resolution generated by it own country, 2) solving some resolution requests generated by other countries. Theoretical analysis shows that our system can work well in large scale applications.

The rest of this paper is structured as following: in the next section we provide the background. In section 3, we explain how our designed name is constructed. In section 4, we investigate the new name resolution system that we proposed. In section 5 and 6, we just analyze our
designed system in usability and delay aspects. Section 7, concludes the paper with a summary.

II. BACKGROUND

In this section, we present some background closely related to our work.

2.1 Overview of peer-to-peer networks

Peer-to-peer (p2p) networks are loosely organized networks of autonomous nodes which make their resource available to other nodes. In these networks, search is one of the important areas that have gained a lot of attention. There are mainly three types of p2p networks: client/server model, unstructured and structured which also calls DHT structure. e.g. Napster, Gnutella and Chord. Each of these has its own search strategy.

In client/server model, nodes are organized loosely, and there is a server to register resource information. People who want to share register the file names and connection information (IP and port). If somebody quires a file, he firstly connects to the server to find the file connection information, then use this information connects to the node who owns the file.

In unstructured p2p networks, nodes are also loosely organized. Every node is equal and no one node is more important than another. Resource on a node needs not register on the other nodes. There are two search strategies: random walk and flooding. The key point of these strategies is that each node forwards the query to its neighbor. For random walk, the search query only passes to one of its neighbors; for flooding, the search query is given to all of its neighbors. There is a hop constrain in the flooding search to bound search area.

In structured p2p networks, nodes are organized in a regular graph, such as a tree or a ring, et al. Resource register and search are both based on Hash. People, who want to share his resource, firstly hash the resource meta-data, and then register this Hash ID and connection information on the node that is responsible for this ID. If somebody wants to search a file, he will firstly hash the file meta-data, and then find the node that is responsible for it to fetch to connection information by using this Hash ID, finally using the information connects to the node that has the file.

In the following sections, we mainly use Chord as our DHT structure, so we introduce it in detail. Chord belongs to the completely decentralized and symmetric structured p2p networks; it provides support for just one operation: given a key, it maps the key onto a node. Data location can be easily implemented on top of Chord by associating a key with each data item, and storing the key/data item pair at the node to which the key maps. Node identifiers are obtained by hashing nodes IP and ports into m-bit integers ordered in a ring mod $2^m$. A key $k$ is assigned to the first node whose identifier is equal to or follows it in the ring. This node is called the successor node of key $k$, denoted by successor ($\hat{k}$).

In the original design, to ensure that queries of object keys can be resolved in the ring, every node maintains a link to its successor in the ring. See Fig. 1 for example.

![Figure 1. The original chord routing](image)

So the successor node of key $k$ can be determined by traversing through the successor links until the node successor ($\hat{k}$) is reached. Because of the long delay and to speed up the search process, the original design has changed to this: each node also maintains $m-1$ fingers, where the $i$th finger points to the node $2^i$ away in the identifier ring ($0 < i \leq m - 1$). See Fig. 2 for example.

![Figure 2. Chord routing with figure table](image)

Since every node has a finger pointing at least halfway to any destination key, each query can be returned with a sequence of halfway forwarding in at most $\log N$ steps, where $N$ denotes the network size.

2.2 The current version of naming and name resolution

In today’s application, naming adopts domain form which is a combination of host location and resource location in the host. It belongs to host-oriented applications. For example, there is a software “matlab.exe” which is stored in the file named “tools” of the host at Beijing Jiaotong University. Then, the name should be www.bjtu.edu.cn/eaie/tools/matlab.exe. This design plays an important role in the original Internet, but with the development of network technology, users need more, and data replication and migration frequently appear that the Domain name system can not deal very well. For example, suppose the software “matlab.exe” above migrates to another file called “open-source”. Then the resource will have a new domain name of www.bjtu.edu.cn/eaie/open-source/math.exe. However, users do not know this change, so they still use www.bjtu.edu.cn/eaie/tools/matlab.exe to fetch the
resource. Consequently, this time they will fail to get it, since there has no “matlab.exe” in the file “tools”.

For the current name resolution mapping system, DNS, a tree-like structure, map a domain name to its corresponding IP. When a name resolution request comes, DNS deals with it from the top to the bottom. For example, if a user wants to find the IP of www.bjtu.edu.cn, he or she first sends this request to the DNS server on the top layer, which maps “cn” to China. Then, the DNS server on the top layer gives the request to the lower layer server which maps “edu.cn” to the “education group” of China. After this, the lower layer server gives the request to its lower layer server which maps “bjtu.edu.cn” to Beijing Jiaotong University. This time the corresponding IP can be found and returned back to the user who initializes the name resolution request. This design is the first version of name resolution mapping system and plays an important role in the early Internet. Nevertheless, the shortcoming of this design is that the whole system would be down and could not provide service anymore when DNS servers are under DoS attacks, especially the top one.

There is another shortcoming that DNS can only map a domain name to a single IP which is a one-to-one mapping. Users can only get the service provided by the certain domain name and have no idea if this one satisfies their QoS requirements. For example, there are three service providers providing a certain service respectively with different QoS levels. A user can only choose one of them blindly each time, since he does know which provider can meet his requirements better.

2.3 Recent version of naming and name resolution

The most representative one of these designs is proposed in [4]. We will give a detailed introduction on its main idea.

In this design, naming uses hash function to get the meta-message of resource and then to hash for a string to index the resource. Because of the data-oriented mechanism, the produced name lasts until the resource is removed. Thus, users can use the original name to find the resource whether it is replicated and migrated or not. Consequently, it can deal with the data replication and migration. But it brings a new shortcoming that the location information of resource is lost and this hinders to effectively locate the resource.

Name resolution mapping system adopts DHT structure. DHT provides support for just one operation: given a key, it maps the key onto a node. Because of the smart placement, user can retrieve the key surely as long as it is in the system. In [4] they tried the name produced above as a key, then registered this key and the corresponding IP together on a node which works as a name resolution mapping server. Users who generate the resolution request can use DHT algorithm to fetch the relevant IP. This design can defend DoS attack in a sense, for the structure of DHT is flat, and no server is more important than any other one. If there is a DoS attack on this system, it could break down only a few servers and other servers are not influenced and can provide service normally. This design can be treated as the second version of name resolution mapping system and it is able to overcome some shortcomings of DNS. However, it introduces some other problems. For a DHT-based name resolution mapping system, processing one resolution request needs \( \log(N) \) at most hops (take Chord as an example), where \( N \) is the number of servers in the system. As a global application, the value of \( N \) can be million-level. So the resolution delay tends to be very large and is not suitable for practical deployment.

In this design, if a user wants to fetch a certain resource, he or she can get a series of IPs using the designed name, then chooses one of them blindly. This can be treated as a simple QoS mechanism. We use the example above. There are three service providers provide a certain service respectively with different QoS levels. A user uses the produced name to map to get the connection information. This time he will get three IPs, and then he chooses one to connect to. Because of the service providers have not registered QoS information and the user’s choice is blind. Thus, this system can not provide real QoS and the user can not enjoy real QoS supporting.

III. OUR NAMING

In this section, we give a clear insight of our designed name. In our approach, a network name is composed of the following three parts: location-part, QoS-part and identity-part. The location-part is used to identify location information. The granularity of the location information is at country level and there are about two hundred countries and regions in the world, so we use 9 bits to identify the location. This part can be assigned by an international organization. The QoS-part is used to identify QoS level. In G.1010, ITU-T advised that QoS can roughly be classed into 8 categories according to two factors which are error to lerant or error into lerant and sensitive to delay, so we use 3 bits to identify the QoS. The identity-part is used to identify resource itself. The produce procedure can be referred to [4], in which the hash function is applied to get a string as an identity. We choose SHA-1 as the hash algorithm, so the length of the identity-part is 160 bits. We illustrate the name structure in Fig. 3.

![Figure 3. Structure of the new name](image)

Because the identity-part is data-oriented, the name of resource will not change after the data replication and migration occurs in a country. Thus, users can always find the resource in terms of the unchanged name. Another advantage of our scheme is that there is location information in the name structure, so most of name resolution requests can be solved in its own domain and the resolution time delay could be brought down.

IV. OUR NAME RESOLUTION

In this section, we will mainly discuss the name resolution mapping system that we proposed. It is a two-
layer structure. The top layer maintains a global DHT architecture which is used to register all the names and to deal with resolution requests that the bottom layer can not solve. The bottom layer maintains some local DHT architectures. Each country maintains its own DHT architecture, so the number of these structures is equal to the number of countries and regions. For each local DHT structure, there are two functions, namely registering the names generated by its own country and dealing with the resolution requests generated by its own country or generated by other countries. Fig. 4 gives a brief illustration of our name resolution mapping system. To interpret the function of our system more clearly, we classify all the servers in our architecture into three categories which are the global servers, the ordinary servers and the gateway servers.

In our designed system, service providers should provide QoS information when they want to publish services, so users can choose the one which is the most meeting his QoS requirement to provide service. We still use the example that three service providers provide a certain service respectively with different QoS levels. This time they should register QoS information in the name. When a user want to get this service, he or she signs the QoS level information that he needs in his produced name, then use this name to map to the corresponding IP. In this way, we achieve real QoS supporting.

4.1 Register a name

The process of registering a name is just abstracting a name for the resource and registering it to the name resolution mapping system. The detailed procedure is as following.

Step 1): Abstract meta-message of the resource.
Step 2): Hash the meta-message to get the identity-part.
Step 3): Sign in the QoS code into the QoS-part.
Step 4): Sign in the country code in the location-part, and get the full name.
Step 5): Register this name in its own local DHT architecture together with the connection information (just IP) using the identity-part.

Step 6): Register this name and connection information on the top-layer via a gateway server.

In fig. 5, we give an example and explain how it works. A Chinese bookseller wants to register a book called “Gone with the wind”. Firstly, abstract the meta-message. Here, we just use the head name as meta-message; secondly, hash the meta-message to get a 160-bit string of 101100011...001; thirdly, sign in the QoS code. Here, suppose the QoS code is 101; fourthly, 9-bit country code will be signed in. Here, we just assume the country code of China is 111010001. We combine these two strings and get the full name 11101000110...001; fifthly, register this name and connection information on the local DHT server using identity-part and the last procedure is registering this information on the global server via gateway server.

4.2 Resolution a name

The process of resolving a name is just using a network name to find the corresponding connection information. There are three kinds of cases, i.e., 1) the user knows the name being registered in its own country, 2) the user knows the name being registered in a certain foreign country and 3) the user does not know where the name is registered. In the following paragraphs we will give a detailed explanation on how to resolve a name under the above three cases. Fig. 6 gives an illustration about how these resolutions work. The serial numbers in the graph indicate three cases of resolution methods.

Case 1: When a user knows that the name is registered in its own country, the process steps are as follows:
Step 1): Abstract meta-message of the resource.
Step 2): Hash the meta-message to get the identity-part.
Step 3): Sign in the QoS code that one needs.
Step 4): Sign in the country code in the location-part, and get the full name.
Step 5): Give this full name to the access point server.
Step 6): The access point server using the location-part identifies this request can be solved in its own country.
Step 7): Based on the country’s DHT algorithm, the access point server using the identity-part finds the server on whom the name is registered.
Step 8): Using the QoS-part to fetch the proper connection information.

Step 9): Give back the connection information to the original requester.

Step 8): Based on the global DHT algorithm, the global server using the identity-part finds the server on whom the name is registered.

Step 9): Using the QoS-part to fetch the proper connection information.

Step 10): Give back the connection information to the original requester.

V. Usability Analyses

In this section, we will analyze the usability of our designed system, i.e., show why our system could support for the future Internet.

As is known, there are two hundred of countries and regions in the world, so we can set each country a DHT-based domain. For facilitative analysis, we just give the following parameters. Each country manages ten thousands ordinary servers, fifty global servers and fifty gateway servers. Also, we assume that each country needs to register and solve the same mount of requests. Of course, this can not be true in actual situation, so we just give a brief analysis and all the numbers of various servers can be changed according to the country’s own situation. Among hundreds of Internet applications, the biggest one is Web. Google reported that the number of webpage is ten billions in the worldwide on July 2008 [7], so we assume that the number needed to be registered is nine times bigger, i.e., one hundred billion. The percentage of resolution request is 0.01%~0.1%, which is much bigger than the real situation. In chapter 4.2, we introduced three kinds of resolution method, and here we define proportion is 80%, 10% and 10%, respectively. There are also two other parameters to be addressed. A record is composed of a name and its corresponding IP, so the storage space will be no more than 1 KB. A server’s process ability is bigger than one million records per second in nowadays. In the following, in order to verify the feasible of our system, we will analyze the process ability and storage space needed for each kind of server.

Firstly, we consider the global server. When we are registering, there are about one hundred billion records needed to be registered. Using the parameters that the number of names one hundred billions and the storage space needed for one name is 1KB, we can get each global server has to own a storage space bigger than:

\[
\frac{10^{11} \times 1KB}{10000} = 10^{0.5} \times 1 KB
\]

i.e., 10G and it is no problem in the current situation.

When solving, each global server has to process:

\[
\frac{10^{11} \times 0.1\% \times 10\%}{10000} = 10^5
\]

records per second, i.e., one thousand, and this is much less when compared with the process ability of a server which we mentioned above is one million records per second. So the global server could do well in registration and resolution.

Secondly we analyze the ordinary server. When we are registering, there are about one hundred billion records
needed to be registered. Using the parameters that the number of domains is two hundreds, the number of ordinary servers in a domain is ten thousand and the storage space needed for one name is 1KB, we can get that the storage space needed for each ordinary server is no bigger than

\[ \frac{10^6}{10000 \times 200} \times 1KB \]

i.e., 50MB. When solving, an ordinary server takes two roles which are solving the request generated by own domain and other domains. So the process ability needed of each ordinary server is

\[ \frac{10^6 \times 0.1\% \times 80\% + 10^6 \times 0.1\% \times 10\%}{10000} \]

records per second, i.e., nine thousands, and this is much less than the process ability of a server which is one million records per second mentioned above. So the ordinary server could do well in registration and resolution.

Lastly we study the gateway server. When we are registering, a gateway server works as an ordinary server. So the storage space needed for the gateway server is the same as the ordinary server’s, i.e., no bigger than 50MB. When solving, a gateway server takes four kind of situations, such as 1) working as an ordinary server solving the resolution requests which it is responsible for; 2) transmitting resolution requests pointing to other domains from its own domain; 3) transmitting resolution requests pointing to its own domain for other domains; and 4) transmitting resolution requests pointing to the global servers. In the first situation, a gateway server works as an ordinary server, so it needs process nine thousand per a second. In the second and third situation, the process ability needed of each gateway server is

\[ \frac{10^6 \times 0.1\% \times 10\% \times 2}{200 \times 50} \]

records per second, i.e., twenty thousands. In the fourth situation, the process ability needed of each gateway server is

\[ \frac{10^6 \times 0.1\% \times 10\%}{200 \times 50} \]

records per second, i.e., ten thousands. We sum up all the probabilities above; a gateway server has to process thirty-nine thousands records per a second, and this is much less then the process ability of a server which is one million records per second mentioned above. So the gateway server could do well in registration and resolution.

VI. RESOLUTION TIME COST ANALYSES

In this section, we analyze the resolution time delay in our system. In [8], authors proposed that time delay is mainly caused by hops in a DHT-based name resolution mapping system, so we use hops as our analysis units. We compare the performance between our system and the system of Ref. [4] that the main idea is using Chord as our DHT structure), where \( N_1 \) is the number of ordinary servers in its own domain.

(````
showed in Fig. 8. We can get that this time Hops needed for one hop is decreased more quickly than the simulation above. This is more approach to the real situations.

![Graph showing the relationship between P1 and Hops](image)

Figure 8. the relationship between P1 and Hops while N1, N2, and N3 are varied with P1

In this simulation, we set p1 is 0.8, p2 and p3 are 0.2. We vary N1 from 1000 to 10K, N2 and N3 from 1000 to 10K. We show the relationship among Hops, N1, N2, and N3 in Fig. 9. We can get that Hops increase while N1 or N2 and N3 are increased. The maximum number of hops needed is about 13.7.

![Graph showing the relationship between Hops, N1, N2, and N3](image)

Figure 9. the relationship between Hops, N1, N2, and N3

The above simulations show the Hops needed while the parameters are varied. In the following, we compare the performance between our designed system and system introduced in [4] in resolution time cost aspect.

Using the name resolution mapping system introduced in [4] and using the same mounts of servers, the maximum hops needed is

\[
T_{hop} = \log(200 \times 10000) = 20.93
\]  

(9)

From the analysis results above, we can get that we have greatly improved the performance in the time cost aspect. The reason for this is that we have introduced domain information in the design, so that a lot of name resolution requests are solved in their own domains and need not to visit top layer.

VII. Conclusions

In this paper, we designed a new naming mechanism and the corresponding name resolution mapping system. The basic idea is combining the advantages of DNS based naming and name resolution with DHT based naming and name resolution. The former has the advantages of owning location information, little delay in one name resolution, and so on. The latter has the advantages of supporting for data replication and migration, robust to DoS attacks, et al. Our design also has a novel feature that it can provide QoS support.

When naming, we divided a name into three parts which are location-part, QoS-part and identity-part which are used to identify location information, QoS level and the resource itself, respectively. Then we adopted a two-layer DHT structure to be the name resolution mapping system. The top layer maintains a global DHT architecture which is used to register all the names and to deal with resolution requests that the bottom layer can not solve. The bottom layer maintains some local DHT architectures. Finally we simulated and analyzed this system and verified that it is usable in actual environment.

Acknowledgment

The authors wish to thank Yuxiang Zhao, Shiyong Li, Shu Wang who all read drafts and made many helpful suggestions. This work was supported in part by the China NSF 60672069, NSF 60772043 China 973 2007CB307101, China PCSIRT IRT0707.

References


Fuhong Lin Born in 1981, received his B.S. degree and M.S. degree from Beijing Jiaotong University, Beijing, China, in 2002 and 2006, respectively, in Electronics Engineering. Now he is a Ph.D. candidate in the School of Electronics and Information Engineering at Beijing Jiaotong University, Beijing, China. His research interests include queueing theory, next generation Internet architecture, and social network and peer-to-peer network.

Changjia Chen Received the M.S. degree in the Electronics Institute, Academia Sinica China in 1982 and the Ph.D degree in electrical and electronics engineering from University of Hawaii at Manoa, HI in 1986. He is currently a Professor at The Beijing Jiaotong University. His research interests are in the fields of communication networks and communication protocols. His research now focuses on the measurement and modeling of p2p networks.

Dr Chen is fellow of CIC and CIE..

Hongke Zhang Received his M.S. and Ph.D. degrees in communication and information engineering from the University of Electronic Science and Technology of China, Chengdu, China, in 1989 and 1992, respectively. He is currently a professor and Ph.D. supervisor with the School of Electronics and Information Engineering, Beijing Jiaotong University, Beijing, China. He is also the Chief Scientist of the project ”A universal network architecture for supporting pervasive services”(No.2007CB307100) founded by the 973 Program of China. His research interests include computer communication, next generation Internet architecture and wireless sensor networks.