SIDMAP:A Service-oriented Mapping System for Loc/ID split Internet Naming

Daochao Huang,Dong Yang,Fei Song and Hongke Zhang
National Engineering Lab for Next Generation Internet Interconnection Devices
School of Electronic and Information Engineering,Beijing Jiaotong University,Beijing 100044, China
Email: {08111043,dyang,fsong,hkzhang}@bjtu.edu.cn

Abstract— **While locator/identifier split makes Internet routing more scalable, the promise of a mapping system that resolves a service to corresponding location is yet to be realized. Part of the challenge is due to most current approaches only focusing on how to find the location of a host, not the location of a service, which make these systems rely on extra naming system to provide applications the requested service's location and introduce complex load interactions between the locator/identifier mapping system and naming system. Another challenge is due to the vast majority of current Internet usage changed from host-centric applications to data retrieval and service access, traditional domain name system (DNS) is unable to support today's Internet needs such as service migration and composition. We introduce SIDMAP–a novel,efficient service-oriented naming scheme for incorporating service identifier/identifier mapping system that returns appropriate locators in response to mapping requests for specific service identifiers,service migration and service composition into one system. According to the result of evaluations, our mechanism has good scalability and low resolution latency.**

Index Terms— **naming system,service composition, service migration,identifier,locator**

I. INTRODUCTION

Today's Internet has evolved greatly from its original incarnation. Firstly, the vast majority of current Internet usage is data retrieval and service access. The Internet is increasingly a platform for diverse services,such as social networks, video streaming, Web search etc. These services always distribute across multiple locations for better reliability and performance. And Internet users become more aware of how to obtain the services rather than where to get them. Secondly,the host-center feature of the DNS system makes it failed to fulfill future Internet naming. As interest grows in new approaches to both mobile hosts and mobile networks,and network virtualization promises a high flexibility by decoupling services from the underlying substrate network and allowing the virtual network to adapt to the needs of the service, the dynamics of both the mobile hosts as well as the ability to migrate services closer to the host location need to be supported. However,the DNS was designed as a host-centric application. This results in contradiction between current Internet usage and DNS original design. For example, when a video moves from server A to server B, the link between the client and server A breaks because URL hard-code a domain and path and the DNS is unable to adapt to this change. Thirdly, the Internet evolution needs more powerful name system. For example, in the Internet of Things (IoT), communications are likely to occur between (or with) objects instead of hosts, it means that Object Name Servers (ONS) are needed to map a reference to a destination of a specific object and the related identifier, and vice versa. This is impossible for DNS without significantly change the original design. Therefore, to expand from Internet to IoT, a scalable and service-oriented naming system is needed to unify the two together.

Furthermore, scaling issues such as endpoint renumbering in the Internet motivate many approaches to evolve the Internet Architecture through clean-slate methods.These approaches includes LISP [1],ILNP [2] and so on. The locator/identifier(Loc/ID) split scheme is expected to be a design principle for new routing architectures that make Internet routing more scalable. It works as follows: Full addresses consist of two parts: the ID or endpoint identifier (EID) identifiers an endpoint and the Loc or routing locator(RLOC) describes its location in the Internet. When a packet is sent to an EID, if the communication occurs between nodes inside the same domain, it works like today's Internet.However, the communication occurs between nodes in different domains, it requires tunneling. A mapping system is needed to provide the corresponding Loc for the ID in the communication process. A few mapping system proposals integrate naming together, but many others only consider routing. In the latter case, applications need to rely on naming systems such as DNS to resolve the resources name and great changes are necessary to make the DNS be updated with new Locator values as hosts and/or networks move: new sessions would obtain the correct Locator(s) for a mobile host and/or network from the DNS. This is a special challenge to change the DNS to hold and update the mapping information.

In this work, we present SIDMAP, a distributed mapping system for future internet naming. It supports routing architectures implementing the Loc/ID split and provides name service for these new Internet architectures,and permits applications fast, reliable access to obtain the EID and location of a service. To be effective, the SIDMAP

Manuscript received July 30, 2011;accepted September 30, 2011.
This work was supported by the National Natural Science Foundation of China under Contract No. 60833002

© 2011 ACADEMY PUBLISHER
doi:10.4304/jcm.6.8.601-609

must satisfy several important goals:

Service-oriented:Comparing to existing methods, SIDMAP is a service-oriented solution rather than host-oriented way to provide dynamic resources discovery and location using unique SID of the service.By decoupling a service from its location,SIDMAP is updated with the new SID-to-resolver information when the location of a service changes,which is usually difficult to address in DNS without great changes to make the service reachable at the new location.In other words, service migration is naturedly supported in SIDMAP.

Uniformly named:The Internet is now widely used by applications to gain access to services and data, yet the Internet does not have a mechanism for directly and persistently naming the vast majority of data and services uniformly.And the trend toward IoT network or Loc/ID split future network will only continue and increasingly make current host-centric naming inconvenient to move service instances and data, as well as to replicate them. We define a novel naming scheme in which any resources can be named by a uniform,permanent 160bits long flat identifier,namely service identifier.

Scalable:The system should have sufficient scalability to support large-scale network and should response quickly to changing map-requests without introducing instability.

The rest of this paper is organized as follows: In section II, we present some research closely related to our work. In section III, we make a brief overview on SIDMAP architecture and give a description of registration of mappings and resolution process of mapping requests, in addition, we discuss the functionality primitives provided by SIDMAP. Section IV designs implementation. Section V gives some performance valuations of our naming system. Section VI concludes the paper and describes the future work.

II. Related Work

Our work in naming system necessarily builds upon work in a number of related areas. Perhaps most closely related work to our efforts are various efforts in designing mapping systems for the Loc/ID split architecture, such as FIRMS [3],DHT-MAP [4],and LISP-DHT [5]. Most of these mapping systems interconnects have been organized as overlay networks such as Chord in LISP-DHT [5], CAN in DHT-MAP [4], to provide the corresponding Loc for an ID. While these approaches take full advantage of the DHT architecture to archive an efficient and secured mapping lookup systems,applications in these systems rely on extra naming systems to return appropriate locators in response to requests for specific service. INLP [2], [6] provide approaches using DNS to store the correct Locator(s) for a mobile host and/or network, and support mobility, multi-homing through naming.However, great changes to DNS in these approaches are typically impracticable.

To satisfy requirements of future Internet applications, some clean-slate redesigns of Internet naming and name resolution are introduced. DONS [7] proposes a novel name service that can seamlessly accommodate mobility and security and integrate middle-boxes. Another related research is [8] which presents a Data-Oriented Network Architecture (DONA) to solve the almost same problems in current Internet involving data availability. [9] provides DONAR which can be used to satisfy naming demands and service selection for cloud services. [10] provides a P2P approach to associate descriptions of a specific object to the related RFID tag identifier, and vice versa.

Basically, there are two types of peer-to-peer based naming systems, simply reimplementing DNS using p2p mechanisms with high lookup latency such as [11], [12], [13], and minimizing lookup latency with high maintenance cost tradeoff such as Behive [14]. These approaches provide some extra and efficient mechanisms such as replication in Behive and rendezvous-based communication abstraction in [13], flat identifiers based multi-level distributed name resolution in [12] and so on. All of these new naming systems,to some extent,solved some special problems in DNS, for example, Behive focus on improving data availability, i3 on decoupling the act of sending from the act of receiving to support variety of communication services such as mobility, anycast and multicast, [12] on supporting multi-homing and mobility.Distributed Domain Name System (DDNS) [15] employs the DHash layer of CFS system [16] to store current resource records of DNS. The basic algorithm of the DDNS is Chord which has the advantages such as high fault-tolerant and good load balancing. However, the simple Chord-based storage policy makes its resources retrieval inefficiency.Cooperative Domain Name System (CoDoNS [17] is an another method which combines DHT and DNS. The basic idea is to separate namespace management from domain name resolution by deploying one or several CoDoNS servers in each domain to resolve domain name and using the traditional DNS servers to manage the domain namespace. All of the CoDoNS servers logically are organized as an overlay network in CoDoNS. In this way, both advantages of DHT and DNS are integrated in one system.

Our proposed naming techniques are based on the presented approaches and can meet the future Internet naming requirements.

III. SIDMAP architecture

In this section we present SIDMAP, a new mapping system for future Internet naming. We describe its architecture, specify its operating process, and discuss its service-oriented features.We make use of LISP's nomenclature (EID,RLOC), but SIDMAP is also applicable to other approaches that are based on the locator/identifier split.

A. SIDMAP architecture

Figure 1 illustrates the architecture of the SIDMAP which consists of resolvers and a mapping system overlay

© 2011 ACADEMY PUBLISHER

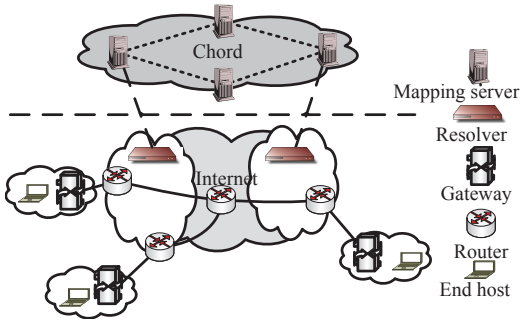

Figure 1. SIDMAP architecture.

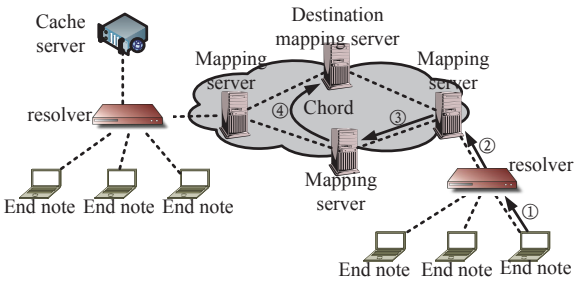

Figure 2. Illustration of the registration process.

network. The mapping system overlay comprises a collection of mapping servers that run SID mapping server software. This server software runs on top of the chord protocol. According to each AS's size, several or many mapping servers are deployed and all of the mapping servers in the whole network are organized as virtual ring. These mapping servers are used to store and retrieve SID-to-resolver mapping entries.

A resolver stores and retrieves SID-to-EID mappings for end hosts and servers in its corresponding zone. When an end host or a server attempts to publicize a resource (service or data) to the Internet, it first generates a corresponding SID by hashing one or more metadata of the resource, such as URLs, key words. Then, the publishing end host or server sends this SID, along with its EID and other information to the resolver for registration. Taking security into consideration, the publishing end host or server should obtain ¡public key, private key¿ pair from the mapping server in the same AS. After that, it can put its public key into the registration message.

B. Registration of Mappings

The registration progress of a SID works as follows (see the Figure 2):

Step 1. When an end host or server (we call it Publisher) attempts to publish service on it, it first generates 160bits long SIDs using SHA-1 consistent hash function for the service. Then, the Publisher sends a registration message which includes its EID to its directly connected resolver to indicate that it will publish service.

Step 2. After receiving this registration message, the resolver first finds in its local mapping database whether there is a SID-to-EID mapping for the SID. If it is the case, the resolver should further check whether the existing SID-to-EID mapping includes the same EID. If so, the resolver simply updates the timer of the mapping and sends an UPDATE message to the directly connected mapping server in the same AS. If there is no same SID-to-EID mapping at the resolver's local mapping database, the resolver creates a new one,stores it in its database and sets a timer for it. Then, the resolver sends a SID-to-resolver mapping registration message into mapping system.

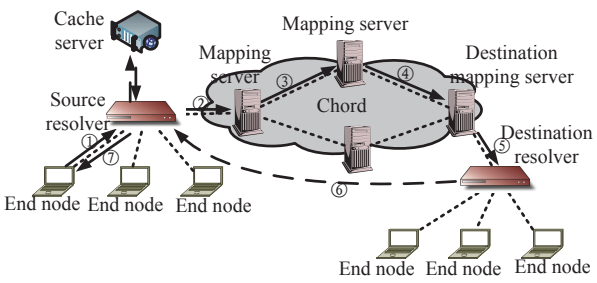

Figure 3. Illustration for the SID resolution process.

Step 3. When a mapping server receives the registration message sent by the resolver mentioned above, it first extracts the SID from the message and then assigns the mapping to its destination mapping server that is responsible for this mapping in a straightforward way. In practice, the SID, namely a key in the mapping system, is finally stored on the first mapping server whose identifier, also 160bits long, equals to or follows the specific SID in its identifier space. Two cases may occur when the process continues. In the first case, the registration message received by mapping server that is just the destination mapping server for it, then the process simply jumps to Step 4 below; otherwise, the mapping server will send the message to an intermediate mapping server which is closer to destination mapping server.

Step 4. When the destination server receives this SID-to-resolver mapping registration message, it simply stores this in its main mapping database and sends copies of this SID-to-resolver information to its backup servers, which will store this mapping in their backup database (in Figure 2, the backup mechanism was not illustrated). To improve the resolution efficiency, each mapping entry should be replicated to more than two mapping servers. The exact number of the backup nodes is determined by the popularity of the mapping entry.

C. Resolving an EID for a SID

Consider the following scenarios involving different types of queries:

Local query. According to our design, the quantity of resolvers in each AS varies with the size of the AS, ranging from several to tens of thousands, while each resolver maintains a mapping database which contains SID-to-EID mappings for all end hosts and servers within its zone. When a mapping request from an end host and a server arrives, the resolver (for ease of presentation, we called this resolver as source resolver) first checks weather

© 2011 ACADEMY PUBLISHER

there is a mapping entry in its local mapping database corresponding to the SID included in the request. If so, the result of the request, namely the SID-to-EID mapping record, is returned directly. After obtaining the EID of server which provides the resource, the end host initiates an end-to-end session using transport protocol under LISP architecture.

Non-local query. If the source resolver finds that the requested SID could not been found in its local mapping database, the resolving process will take steps as follows: (see Figure 3)

Step 1. The end host sends a mapping request to the source resolver in its domain. The mapping request should contain the SID information which can be obtained by search engine like *google*, and some signature used for security.

Step 2. When the resolver receives such a mapping request, it finds in the caching server weather there is cache entry for the requested SID or not. If so, then the process simply jumps to Step 7. Otherwise, the source resolver immediately forwards the request to the mapping server in the same AS.

Step 3. When the mapping server receives the mapping request, it checks weather the SID contained in the request is in its own zone. If it is the case, namely, the mapping server is just the destination mapping server (we name it DMS for short in the following)for the SID, then simply jumps to Step 5. If not, the mapping server will send the mapping request to an intermediate mapping server which is closer to destination mapping server.

Step 4. As an intermediate mapping server, it is one of the neighbors of previous mapping server and it is closer to the DMS of the requested SID. If the DMS is not the next hop of the Chord ring, the forwarding process will continue until the request finally reaches the DMS.

Step 5. When the DMS receives the mapping request, it finds out the SID-to-resolver mappings that matches the request in its local main mapping database. Then, the DMS select the destination resolver from the collection of occurrences according to their priorities in terms of quality of service, bandwidth,etc,before forwarding the request to it.

Step 6. When the destination resolver receives the request, it encapsulates the data packet in the mapping request and sends back the SID-to-EID mapping to the source resolver.

Step 7. When the source receives this SID-to-EID mapping, it stores this SID-to-EID mapping into its caching entries and sends the SID-to-EID mapping to the requesting end host.

D. Functionality primitives provided by SIDMAP

Today's DNS-based names for services and data force application to resolve the service and data names down to an IP address, thereby binding the application request to particular network location, as expressed by an IP address. At the meantime, a serious scaling problem encountered in current Internet is the overloading of IP address semantics. That is, an IP address represents not only the location but also the identity of a host. In sense,the current Internet architecture was built based on the end to end model and, as a result, the architecture is well-suited for communication between pairs of stationary hosts, but it does not fit comfortably service access and data retrieval. Given this discordance between historical design (host-oriented) and current usage (service-oriented), our research suggests that most of the necessary changes reside in how Internet names are structured and resolved. We propose replacing DNS names with flat, self-certifying names, and replacing DNS name resolution with a DHT-based mapping system that runs above the transport layer. We call the resulting design SIDMAP.

We now discuss a few possible ways in which this basic primitive might be used by applications to achieve more general support for local mapping entries update, service composition, service or data migration, and multi-level cache mechanism.

Local mapping entries update. According to [18], there are approximately 15,553,600 name servers running on the Internet; there are as many as 11,920,500 open resolvers on the Internet. To obtain the total capacity of the mapping system, we notate the number of resolvers N_r and the average number of the SIDs per resolver N_{as}, the total entries stored in the mapping system are $N_r \cdot N_{as}$. If all the UPDATE messages are processed by mapping system, it will cause heavy overhead for the system. To adapt to the scaling of the Internet, in SIDMAP the resolver plays a role more than cashing and forwarding entity like stub-resolver in traditional DNS systems. Besides these functionality, the resolver maintains a local mapping database for storing SID-to-EID mapping entries. Given this design, if a SID-to-EID UPDATE message arrive at the resolver, what the resolver needs to do is updating corresponding entry in its local database, rather than forwarding the UPDATE message to the mapping system simply. Furthermore, if a mapping request applies for a SID in the same zone, the resolver can directly answer the request without contacting the mapping system.

Service composition. Service composition is widely used in today's Internet, especially in mobile Internet. Application data cannot be used without transformed by a series of third-party servers before it reaches the destination. Unfortunately, the application needs to know the set of servers' IP addresses through DNS and, as a result, causes unnecessary latency.

With SIDMAP, service composition can be easily implemented. Two types of service composition are considered here, parallel service composition in which composed service is the union set of several sub-services and cascaded service composition where the output of the previous sub-service is the input of the successor service. For types 1, the composed service $SID_c = SID_1 + SID_2 + \ldots + SID_n$, for example, *Sohu* web pages consist of many pictures which have identifiers

© 2011 ACADEMY PUBLISHER

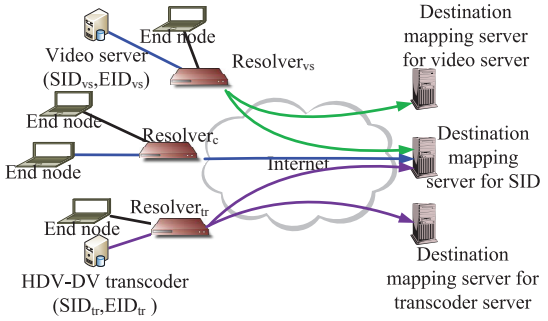

Figure 4. Service composition.

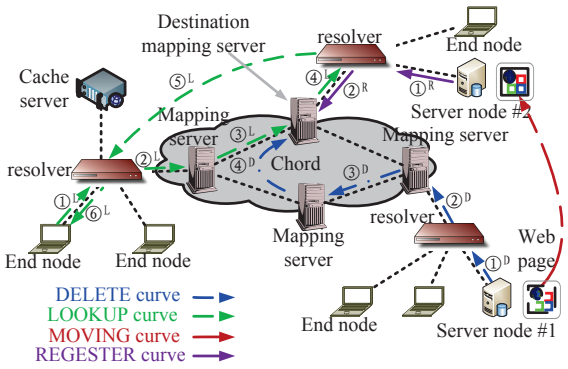

Figure 5. Service or data migration.

$SID_{pic1}, SID_{pic2}, \ldots$,when we get the web pages, in fact we obtain a composed service SID_{web},where SID_{web} is the union set of $SID_{pic1}, SID_{pic2}, \ldots$.

For type 2, in this case, data might need to be processed by a serious of third-party servers, where the output of previous server is the input of the next server. Figure 4 shows how data packets containing high-definition video can be redirected to the transcoder to change video format, and, thus, arrive at the receiver containing appropriate video format. When the original video server with SID_{vs} and translating server with SID_{tr} register in the mapping system respectively, both of the two also send $SID_{vs} - to - resolver_{vs}$ and $SID_{tr} - to - resolver_{tr}$ to the destination mapping server of SID, where SID is the composed service's identifier. The corresponding mapping entry to the SID is $SID - to - < SID_{vs} - to - resolver_{vs}, SID_{tr} - to - resolver_{tr} >$. The DMS of SID answers the mapping requests with this mapping entry.After this process,the data packet from the client is routed first to the video server hosting service SID_{vs}. Next, the video server forwards the request to transcoder server SID_{tr}. The transcoder server changes the original video format and delivers the video to the client.

Service or data migration. A service or data can first unregister from its original location and then re-register at its new location. Subsequent requests will be routed to the new location as soon as the new registrations have taken effect. This process is referred to as service migration. With SIDMAP, when service migration occurs, the most flexible and efficiency approach is provided. In general, the service migration can be classified into two types in terms of their migration distance. If the target destination zone of the service migration is belonging to a different resolver, then, both the original SID-to-resolver mapping entry and SID-to-EID mapping entry need to be updated. If this is not the case, in other words, the target destination zone of the service migration is belonging to the same resolver but with different end host, only SID-to-EID mapping entry needs to be updated. For examples, if the service migration occurs locally, where locally means the service with SID_1 migrates within the scope of original resolver i but emigrate from the original end host h_1 with EID_1 to the destination end host h_2 with EID_2. Then, the corresponding mapping entry $SID_1 - to - resolver_i$ which is stored by Chord-based mapping system remains unchanged. The only thing should do is updating $SID_1 - to - EID_1$ to $SID_1 - to - EID_2$ in the mapping entries database of resolver i. But if the service emigrates from resolver i to resolver j where $i \neq j$, the corresponding mapping entry $SID_1 - to - resolver_i$ needs to be changed to $SID_1 - to - resolver_j$, at the same time, the $SID_1 - to - EID_1$ mapping entry should be deleted from resolver i and a new mapping entry $SID_1 - to - EID_2$ should be added into the mapping entries database of resolver j.To describe the complex service migration more reasonably, we describe this process as follows, with the help of the illustration shown in Figure 5. In this example, some Web content located on server node ♯1 migrates from server node ♯1 to server node ♯2. The entire process consists of three steps: unregistering or deleting the mapping information ($①^D \sim ④^D$ in Figure 5), re-registering the mapping ($①^R \sim ②^R$ in Figure 5) and querying process ($①^Q \sim ⑥^Q$ in Figure 5).As soon as service or data migrates from the server node ♯1 to the server node ♯2, the server node ♯1 immediately sends a DELETE message which includes the SID to its resolver ($①^D$ in Figure 5). Once the resolver receives the DELETE message, it deletes the corresponding SID-to-EID mapping entry and forwards the message to the mapping system ($②^D$ in Figure 5).Then the destination mapping server deletes the SID-to-resolver mapping entry.Re-registration process is similar to the process of registration described in previous Subsection III-B.

Multi-level caching scheme. In SIDMAP, to reduce lookup latency or mapping resolution delay, a novel multi-level caching scheme is proposed to decrease lookup latency and enhance mapping servers' performance. This scheme integrates resolver level caching and mapping server level caching together in one system.

Resolver level caching is similar to the DNS caching, one of the main differences between the two is that SID-to-EID mapping entries are cached for mapping requests rather than URL to IP address resolution results. Another difference is that the destination resolver information corresponding to a request can be cached locally by source resolver. The introduction of resolver level caching is capable of reducing the burdens of mapping servers significantly as illustrated in our simulations.

© 2011 ACADEMY PUBLISHER

Mapping server level caching is implemented at each source mapping server(the mapping server that is directly connected to the source resolver) when the mapping requests are resolved by its destination mapping server. The caching contents include SID-to-resolver mapping entries and their destination mapping servers.

Since SID-to-EID mappings and SID-to-resolver mappings are cached at source resolvers and source mapping servers respectively, an update cannot lead to inconsistency. And our simulation results show that this multi-level scheme can achieve high availability and low latency.

IV. IMPLEMENTATION

In this section, we provide the implementation of SIDMAP prototype.

Chord is one of the famous structured peer-to-peer (P2P) networks which usually use a distributed hash table (DHT) approach to overcome scalability problems. To determine the location of nodes and resources in a Chord overlay graph by the hash values of their identities and keys, Chord maps nodes and resources to identities of bits placed around a modulo identifier circle, each node maintaining links to distances for greedy routing. This logical topology facilitates for Chord easily retrieve the identity of the node which stores the resource, from any other node in the network.

With m machines in a Chord system, it takes $O(\log m)$ space and time for routing, and $O\ \log^2 m$ time for node insertion. Because hash function destroys the semantic meaning of keys, Chord system does not support complex query such as sematic query,range query. Besides, Chord suffers from high lookup latency, $(O(\log m))$ in the average case. Such high latency prevents Chord from being used in naming system without changes. As a result, we make some modifications to the original Chord so that it can be more readily used in SIDMAP.

A. Mapping Server Implication

We make some modifications to the original Chord such that it can be more readily used in SIDMAP.

Modification 1: Set two backup mapping servers for each SID-to-resolver mapping.As reference [19] stated, to cope with node failure, the practical number of backup nodes is three. In order to prevent the SID-to-resolver mappings stored in a mapping server from being unreachable, our first modification is to set two randomly selected neighbors from the mapping server's successor list as its backup servers. With this modification, a SID-to-resolver mapping is not only owned by its destination mapping server, but also is backup to the other two mapping servers. Therefore, if a mapping server fails, the mapping server system can also correctly resolve a value for a given key. However, this backup mechanism is useful to improve system robustness not only, be in the same time it increases the risk of replica consistency. We have taken this into account in our implications.

```
/*An End host with identifier EID_x has service SID_y to be published. Assuming that It directly
connects to resolver resolver_z */
RegisterS2E ( SID_y, EID_x )   /*Registration process on resolver_z . */
1   for ( k = 0; k < t; k++ ) /* t  is the total number of existing mappings in  resolver_z . */
2       if ( SID_k == SID_y )
3           if ( EID_k == EID_x )   /*Mapping already exits. */
4               UpdateTimers;
5               break;
6           else
7               UpdateMapping ( SID_k , EID_x );
8       else
9           continue;
10  CreateNewMappings ( SID_y , EID_x );
    /* Create new SID-to-EID mappings in resolver. */
11  SendMessage (REGISTER,  SID_y , resolver_z );
    /*Forward REGISTER message to closest mapping server. */
```

Figure 6. Pseudo code of SID-to-EID mapping registration.

```
/*Assuming that  resolver_z  directly connects to mapping server  MS_w */
RegisterS2R ( SID_y , resolver_z )   /*Registration process on  MS_w . */
1   if ( predecessorNodeID < SID_y ≤ nodeID )
    /* MS_w is the destination mapping server of  SID_y . */
2       for ( j = 0; j < s; j++ ) /* s  is the total number of existing mappings in MS_w . */
3           if ( SID_j == SID_y )
4               if ( resolver_j == resolver_z )   /*Mapping already exits. */
5                   UpdateTimers;
6                   break;
7               else
8                   UpdateMapping ( SID_j , resolver_z );
9           else
10              continue;
11      CreateNewMappings ( SID_y , resolver_z );
        /* Create new SID-to-resolver mapping in mapping server. */
12  else
13  ForwardMessage (REGISTER,  SID_y , resolver_z );
    /*Forward REGISTER message according to MS_w 's finger table. */
```

Figure 7. Pseudo code of SID-to-resolver mapping registration.

Modification 2: The mapping server level caching Because each resolver stays constantly connected to one or several (to protect against a single point of failure) mapping servers, it will immediately receive most of request answers if the connected mapping server has built-in cache mechanism and provide high cache hit ratio as far as possible. At system first startup time, there is no cache existing. After a mapping request is correctly answered by mapping system, the destination mapping server sends

B. Pseudo code of critical sections

We implement a prototype of SIDMAP system using C++ programming. The basic ideas behind the prototype are exactly following the architecture design described above.Some pseudo codes of critical sections,including SID-to-EID mapping registration,SID-to-resolver mapping registration,SID-to-EID mapping resolution,SID-to-resolver mapping resolution and service migration, are given respectively as follows(see Figure 6,Figure 7, Figure 8,Figure 9,Figure 10):

The local mapping entries update mechanism is included in the SID-to-EID mapping registration process. As shown in FIGURE 6,when a mapping entry already exists,what needs to do is just updating the timers belonging to the entry (to prevent the resolver's SID-to-EID database cluttering with incorrect or invalid entries that may seriously degrade the performance of the resolver) or adding new service providers with different EIDs to

© 2011 ACADEMY PUBLISHER

```
/*A user send mapping request with SID_y to be resolved. Assuming that It directly connects to
resolver resolver_p */
ResolveS2E ( SID_y )   /*Resolution process on resolver_p . */
1   for ( i = 0; i < m; i++ ) /* m  is the total number of cache records in resolver_p . */
2       if ( SID_y == cacheRecord_i )
3           return  cacheRecord_i ; break;
4       else
5           continue;
6   for ( j = 0; j < n; j++ ) /* n  is the total number of existing mappings in resolver_p . */
7       if ( SID_j == SID_y )
8           return  EID_x ; break;
9       else
10          continue;
11  SendMessage (RESOLUTION, SID_y );
    /*Forward RESOLUTION message to closest mapping server. */
12  UpdateCacheRecords( SID_y , EID_x );
     /*Update the cache records when mapping server returns the resolution results. */
```

Figure 8. Pseudo code of SID-to-EID mapping resolution.

```
/*Assuming that  resolver_p  directly connects to mapping server  MS_q */
ResolveS2R ( SID_y )   /*Resolution process on MS_q . */
1   for ( i = 0; i < n; i++ ) /* n  is the total number of cache records in MS_q . */
2       if ( SID_y == cacheRecord_i )
3           return  cacheRecord_i ; break;
4       else
5           continue;
6   if ( predecessorNodeID < SID_y ≤ nodeID )
    /* MS_w is the destination mapping server of SID_y . */
7       for ( k = 0; k < l; k++ ) /* l  is the total number of existing mappings in MS_q . */
8           if ( SID_k == SID_y )
9               return  resolver_z ;
10              ForwardMessage (RESOLUTION, SID_y , resolver_z );
                /*Directly forward the RESOLUTION message to destination resolver_z . */
11              break;
12          else
13              continue;
14  else
15  ForwardMessage (RESOLUTION,  SID_y , MS_next );
    /*Forward RESOLUTION message to next hop mapping server  MS_next  according to MS_q 's
    finger table. */
16  UpdateCacheRecords( SID_y , resolver_z );
     /*Update the cache records when destination mapping server returns the resolution results
*/
```

Figure 9. Pseudo code of SID-to-resolver mapping resolution.

the same SID.The relative functions for updating timers and increasing service providers are *UpdateTimers* and *UpdateMapping*,respectively. This locally updating mechanism is proved enormously useful for today's Internet which has more and more applications depending on naming system to resolve resource names to locations.

Both in the registration and resolution process, mapping servers determine the next hop of the messages (such as REGISTER,RESOLUTION,DELETE etc.) or mapping lookups with finger table which is one of the key components of Chord. When a mapping lookup arrives, the mapping server first lookups its routing table or finger table to see weather the requested SID belongs to its range or not. If so, namely $predecessorNodeID < SID \leq nodeID$, the mapping server happens to be the destination mapping server of the SID. If not, the mapping server forwards the lookup to next hop server. Here, $predecessorNodeID$ and $nodeID$ are 160 bits long identifiers of the predecessor mapping server and this mapping server.

In FIGURE 10, we take four types of service or data migrations into account according to whether the server or data belong to the same resolver zone or not and

```
/*Service migration.*/
ServiceMigration( SID_y , EID_src , EID_dest , resolver_src , resolver_dest )
1   if ( resolver_dest == resolver_src )
2       if ( EID_dest == EID_src )   /*Same end host, same resolver zone, do nothing. */
3       else   /*Same resolver zone, but different end host, locally update. */
4           SendMessage(DELETE,  SID_y , EID_str );
             /*Delete the original SID-to-EID mapping. /
5           RegisterS2E ( SID_y , EID_dest );
6   else
7       if ( EID_dest == EID_src )   /*Same end host, but different resolver zone. */
8           SendMessage(DELETE,  SID_y , EID_str );
            /*Delete the original SID-to-EID mapping. /
9           RegisterS2E ( SID_y , EID_src );
10          SendMessage(DELETE,  SID_y , resolver_str );
            /*Delee the original SID-to-resolver mapping. /
11          RegisterS2R( SID_y , resolver_dest );
12      else   /*Different resolver zone, and different end host. */
13          SendMessage(DELETE,  SID_y , EID_str );
             /*Delete the SID-to-EID original mapping. /
14          RegisterS2E ( SID_y , EID_dest );
15          SendMessage(DELETE,  SID_y , resolver_str );
            /*Delete the original SID-to-resolver mapping. /
16          RegisterS2R( SID_y , resolver_dest );
```

Figure 10. Pseudo code of service migration.

weather it is still hosted by the same host or not after migration: *Type 1:*same resolver zone, same end host. This is the most simplest case, in which the service or data hosted on an end host follows the end host moving around in the same resolver zone;*Type 2:*same resolver zone, different end host. In this case, what needs to do is just updating the corresponding SID-to-EID entry locally on the resolver to correct EID;*Type 3:*different resolver zone,same end host.If it is this case,because resolver changes, service or data should be republished on the new resolver, and SID-to-resolver mapping should be updated to correct resolver;*Type 4:*different resolver zone,different end host. This case also needs to republish the mapping on the resolver and update SID-to-resolver mapping on the destination mapping server.

V. EVALUATIONS

In this section, we evaluate the resolution delay which is the key parameter in the system and simulate the performance of SIDMAP. Two different data sets derived from measurements of a real network are used. The first data set involves traces of the resolution queries to our campus DNS servers during the period from December 2,2010 to December 14,2010. There are over 13,000 active users per day among the total number of 40,000 users in Beijing Jiaotong University network. A traffic filter is defined to obtain a number of query logs which include all the queries answered by our campus DNS servers. As a result, 70,831,353 queries meet the requirements. Each query contains the following six fields: the time of the query, the IP of the requesting client, the name queried, the type of the query ,the protocol and the recursive query tags.We treat each query as an SID-to-EID mapping requests sent into SIDMAP system. Assuming that SID and EID obtained by simply hashing (using SHA-1) the name queried and the corresponding IP addresses respectively. Then, we use the second data set which introduced later to build a Chord topology to simulate the mapping system.

© 2011 ACADEMY PUBLISHER

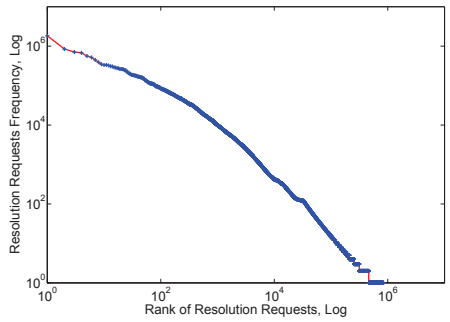

Figure 11. Zipf-like distribution of resolution requests.

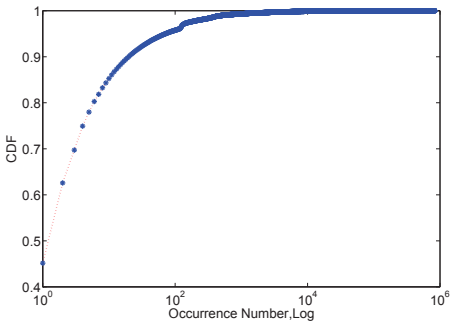

Figure 13. Occurence times of queries.

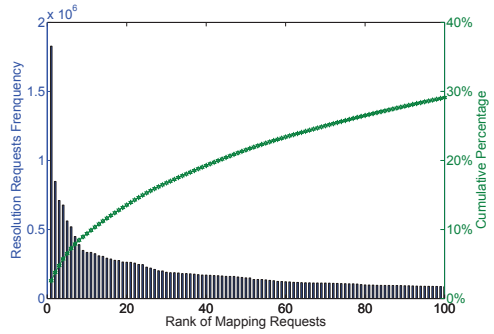

Figure 12. Resolution requests frequency.

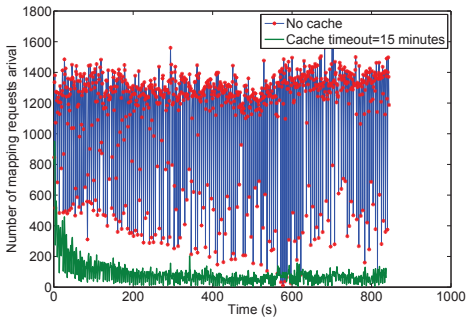

Figure 14. Average number of resolution requests arriving at mapping servers with cache mechanism.

The second data set involves 2051 Internet DNS servers. Based on the kingdata [20], we built several topologies that can be used in P2PSim [21]. In these topologies, the distance between two DNS server nodes is used to simulate the distance of two mapping servers. Therefore, the simulation results greatly reduce the deviation between simulation and real network because it includes a connection of real network RTT (Round-trip Time) values.

We first investigated the resolution requests' frequency of data set one. We found that the resolution requests follow Zipf's law with slope of approximately -0.91 (see Figure 11).

Then, to demonstrate the effects of multi-level caching scheme that has been described above, we conduct an emulation to estimate average frequency of resolution requests arriving at mapping server in practical environment. For this purpose, statistics of the first data set are given here. We count the number of the occurrences of queries with the same required name during the tracing period, and then rank them (there are about 839,795 unique names over 13 days in total) according to their frequencies.The frequency of the kth ($1 \leq k \leq 100$) most popular query is shown in Figure 12. We found that the most 100 popular queries account nearly 30% of the total queries. We also found that most of resolution requests ask for local SID-to-EID mapping resolution (5,831,068 local queries,about 9% of the total). In other words, resolution delay can be deduced significantly by the local mapping databases.

Furthermore,from Figure 13, we found that 45.14% of the requests came out only once during the observation period(here,we selected the unique requests rather than the total requests). Requests up to 90.04% appear less than or equal to 20 times and 95.76% less than or equal to 100 times. From inside the statistic results we reach such conclusion,only less than 10% of the requests really need to be cached because these requests occur frequently.

To estimate the average resolution latency, we use the queries of our campus network as mapping resolution requests input, and take the topologies of 128, 256, 400, 512, 700, 850, 1024, 1200, 1500, 1700, 2048 nodes in kingdata as mapping servers' topologies. Figure 14 shows that the caching scheme has significant impact on the number of resolution requests sent to mapping servers. The cache timeouts is 15 minutes. One can clearly see that the number of resolution requests decreases substantially from about 1,119 requests per second to 89 requests per second, approximately reduced by 92% of the total resolution requests (see Figure 14). Apart from, we also found that,compared with the resolution delay without our multi-level cache mechanism, the average resolution delay with cache is significantly reduced (see Figure 15).

To understand the incremental scalability of the SIDMAP system, we simulated the maximum lookup rate (about 100 lookups/sec/server) with 128, 256, 512, 1024, 2048 mapping servers topologies. The result confirmed that large-scale SIDMAP with 2048 mapping servers can response as quickly as small-scale SIDMAP with 128 mapping servers (see Figure 16). In this simulation, we do not take the caching scheme into account.

© 2011 ACADEMY PUBLISHER

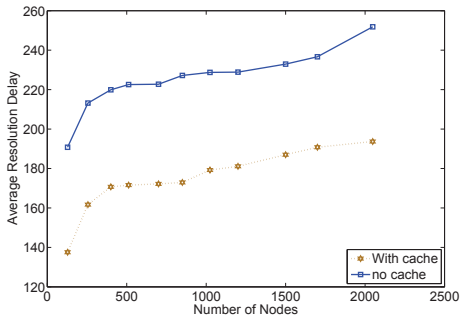

Figure 15. Average resolution delay with cache.

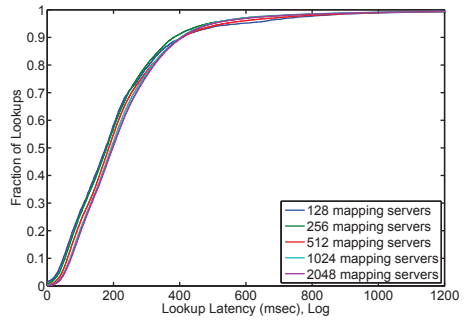

Figure 16. Fast response time for lookups.

VI. Conclusion

We have presented the design and implementation of SIDMAP as a novel naming system for new Internet architectures implementing the identifier/locator split. By decoupling a service from its location, and designing the mapping scheme with service migration and composition supports, SIDMAP forms a service-oriented naming system.When the location of a service changes, the mapping system is updated with the new SID-to-resolver information,neither endpoint renumbering nor changes to the DNS system are needed to the service reachable at the new location. According to the design principle of SIDMAP, each resolver provides a map-base holding the SID-to-EID mappings locally for all its EIDs and each mapping server is configured with a map-base storing the SID-to-resolver mappings for all its responsible SIDs' locations, if a local mapping is required, a one-hop map-reply is returned directly by the resolver,instead of forwarding the map-request to mapping servers, thus providing good scalability.It also includes good worst-case lookup performance and can resolve a service to its location fast with caching and local mapping updating schemes. Our performance evaluations showed that map-requests frequency, update loads, scalability, and resolution delays for SIDMAP are manageable.

Acknowledgment

We thank Shiyong Li for numerous discussions concerning this work,and the reviewers for their detailed comments.

References

[1] D. Farinacci, V. Fuller, D. Oran, and D. Meyer, "Locator/ID Separation Protocol (LISP), IETF Internet draft,draft-farinacci-lisp-12.txt," 2009.

[2] R. J. Atkinson, S. Bhatti, and S. Hailes, "Inlp: mobility, multi-homing, localised addressing and security through naming," *Telecommunication Systems*, vol. 42, no. 3-4, pp. 273–291, 2009.

[3] M. Menth, M. Hartmann, and M. Hofling, "Firms: A mapping system for future internet routing," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 8, pp. 1326–1331, 2010.

[4] H. Luo, Y. Qin, and H. Zhang, "A dht-based identifier-to-locator mapping approach for a scalable internet," *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, no. 12, pp. 1790–1802, 2009.

[5] L. Mathy and L. Iannone, "Lisp-dht: towards a dht to map identifiers onto locators," in *CoNEXT*, 2008, p. 61.

[6] R. J. Atkinson, S. N. Bhatti, and S. Hailes, "Evolving the internet architecture through naming," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 8, pp. 1319–1325, 2010.

[7] H. Wu, D. Gao, D. Yang, and H. Zhang, "A novel data-oriented name service," *Journal of Communications*, vol. 5, no. 9, pp. 684–691, 2010.

[8] T. Koponen, M. Chawla, B. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," in *SIGCOMM*, 2007, pp. 181–192.

[9] P. Wendell, J. Jiang, M. J. Freedman, and J. Rexford, "Donar: decentralized server selection for cloud services," in *SIGCOMM*, 2010, pp. 231–242.

[10] V.Krylov, A.Logvinov, and D.Ponomarev, "Epc object code mapping service software architecture: web approach."

[11] R. Farha and A. Leon-Garcia, "Peer-to-peer naming architecture for integrated wireline/wireless networks," in *GLOBECOM*, 2007, pp. 2019–2024.

[12] L. Loyola, P. Mendes, F. Romero, and M. Jimenez, "Multi-level distributed name resolution system based on flat identifiers," in *GLOBECOM*, 2008, pp. 2253–2258.

[13] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana, "Internet indirection infrastructure," *IEEE/ACM Trans. Netw.*, vol. 12, no. 2, pp. 205–218, 2004.

[14] V. Ramasubramanian and E. G. Sirer, "Beehive: O(1) lookup performance for power-law query distributions in peer-to-peer overlays," in *NSDI*, 2004, pp. 99–112.

[15] R. Cox, A. Muthitacharoen, and R. Morris, "Serving dns using a peer-to-peer lookup service," in *IPTPS*, 2002, pp. 155–165.

[16] F. Dabek, M. Kaashoek, D. R. Karger, R. Morris, and I. Stoica, "Wide-area cooperative storage with cfs," in *SOSP*, 2001, pp. 202–215.

[17] V. Ramasubramanian and E. G. Sirer, "The design and implementation of a next generation name service for the internet," in *SIGCOMM*, 2004, pp. 331–342.

[18] http://dns.measurement-factory.com/surveys/201010/.

[19] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: amazon's highly available key-value store," in *SOSP*, 2007, pp. 205–220.

[20] K.P.Gummadi, S.Saroiu, and S. Gribble, "King: Estimating latency between arbitrary internet end hosts," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*. ACM, 2002, pp. 5–18.

[21] T. Gil, F. Kaashoek, J. Li, R. Morris, and J.Stribling, "p2psim, a simulator for peer-to-peer protocols," 2003.

© 2011 ACADEMY PUBLISHER