Separating Identifier from Locator with Extended DNS

Jessie Hui Wang, Yang Wang, Mingwei Xu, Jiahai Yang
hwang@cernet.edu.cn wayang84@gmail.com xmw,yang@cernet.edu.cn
Tsinghua National Laboratory of Information Science and Technology
Network Research Center, Tsinghua University

Abstract—**Although most researchers have agreed that the locator/identifier separation is beneficial for the Internet, there is no consensus on how to define the "identifier". In this paper, we propose a scheme in which identifiers are distributed by authorities to endpoints, and the authorities are responsible for maintaining real-time locators of endpoints with identifiers they distributed. This scheme can be helpful to accounting, security and other network management tasks. We also present in details how to implement this scheme with extended DNS and a new infrastructure, *i.e.*, ID Mapping System (IDMS).**

I. INTRODUCTION

The architecture of current Internet faces several important challenges. In IAB Routing and Addressing Workshop 2006, researchers pointed out clearly that some of current challenges are caused by the overloading of IP address semantics - both "who" (endpoint identifier, as used by transport layer) and "where" (locators for the routing system) [1]. Following this logic, researchers proposed to architect and implement the so-called locator/identifier separation.

With locator/identifier separation, endpoints can communicate with each other through their identifiers instead of IP addresses. When an endpoint moves from one topological location to other locations, its identifier does not change although its addresses must change. This makes it possible that the ongoing communication can be continuous during moving.

The locator/identifier separation is also helpful to routing scalability. Currently the usage of CIDR, multi-homing, prefix splitting or more specific prefix for traffic engineering *etc.* makes topology based address aggregation less efficient. A solution for this issue is to require endpoints in multihomed edge networks to adopt multiple provider-assigned (PA) addresses from their providers to enable provider-based address aggregation, *e.g.*, Shim6 [2] and HIP [3]. With locator/identifier separation, an endpoint can identify its remote peer even if the remote peer is using multiple addresses during the communication.

Although most researchers have agreed that the separation is beneficial for the Internet, there is no consensus on how to define the "identifier". In Shim6, the endpoint identifiers are the initial addresses used between the two hosts. Current Shim6 assumes all available addresses are pre-defined Hash-Based Addresses (HBA) [4], which cannot be true for mobile hosts. In HIP, a Host Identifier is the public key of an asymmetric key-pair, so that all HIP implementations must support the RSA/SHA1 public key algorithm, and should support the DSA algorithm. In LISP [5], EIDs should be usable for routing within sites although they are not usable for global routing, so EIDs are in fact locators of end systems in edge networks.

In this paper, we propose a scheme where identifiers should be distributed by authorities to endpoints, which helps accounting, security and other network management tasks. The identifier takes the format of *hostname@orgnization*, where organization is the name of the authority that distributes this identifier. The authority that distributes this ID is responsible for maintaining real-time locator of this endpoint, *i.e.*, one authority should maintain mapping entries of all identifiers with its name as suffix in its identifer-locator mapping server.

The rest of paper is organized as follows. In Section II, we will describe the framework of our scheme. We also present a detailed example to illustrate how our scheme works. In Section III, we focus on how we extend current DNS infrastructure to serve our purpose of identifier/locator separation. We also briefly introduce our considerations in the development of ID Mapping Syetem (IDMS). In Section IV, we present some well-known schemes on identifier/locator separation, and compare our scheme with them. Section V concludes the paper.

II. FRAMEWORK OF THE SCHEME

In order to decouple identifiers of nodes from their locations, we introduce the new name space of ID. ID represents the identity of a node, regardless of its location. In our scheme, endpoints on the Internet can apply unique identifiers from authorities and the authorities then keep track of these endpoints' locations, which may be dynamic and changing. IP addresses are no longer to identify hosts, and are only used to determine locations of endpoints for routing.

A. Definition of Identifiers

ID takes a format of $hostname@orgnization$, where $organization$ is the name of the authority that distributes this ID. The authority that distributes this ID to the endpoint is responsible for maintaining real-time IP address of this endpoint, *i.e.*, one authority should maintain mapping entries of all IDs with its name as suffix in the ID-IP mapping system.

The endpoint identifier in this format is human friendly and it is convenient for network users to remember and use. This feature is especially important in IPv6 networks since IPv6 addresses are too long to be used directly by network users. More importantly, it would be easier to arrange hierarchical structure to manage global identifiers with this identifier format.

To facilitate software developer of new applications, host protocol stack should be modified to provide ID-based socket functions. The new host protocol stack should be designed in a way fully backwards compatible to endpoints with the current IP stack.

B. *Mapping of Identifier to Location*

The authorities are responsible for tracking locations of those endpoints with IDs they distribute, *i.e.*, maintaining mapping entries between IDs and locators. Let us define these mapping servers of authorities as ID Mapping Server (IDMS).

There are a lot of IDs in the Internet, and they are distributed by different IDMSs. These IDMSs constitute a global mapping system and store mapping entries for all IDs in the Internet. However, how to find the proper IDMS that is responsible for a particular ID when an endpoint initiates a request? In other words, we need an index for these IDMSs. We extend current DNS infrastructure to serve as this index. To distinguish current DNS and our extended DNS, let us define the extended DNS system as eDNS.

Therefore, there are two steps in our scheme to accomplish an identifier-locator lookup. First, the endpoint looks up eDNS to retrieve IP address of the IDMS responsible for the target ID. Second, the endpoint sends a lookup request to the IDMS and gets the IP address of the target ID. Figure 1 illustrates this procedure.

This two-step procedure can be simplified into one step for some IDs. eDNS can store location information for some special IDs, *e.g.*, servers that are visited frequently. These hosts can apply to eDNS and ask eDNS to save their addresses directly. These hosts should be with relatively constant addresses, so that eDNS does not need to update its entries frequently.

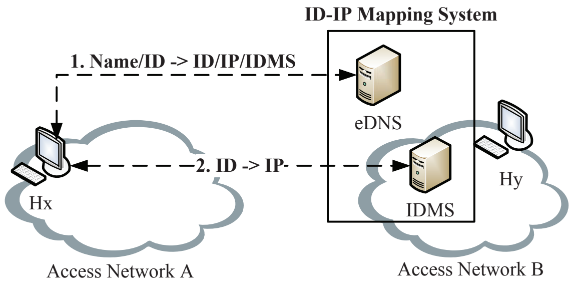

Fig. 1. Mapping of Identifier to Location.

We can see that eDNS should be able to answer two kinds of queries: queries using name and queries using IDs. The response message to a query using name may include ID of the target host and IP of the corresponding IDMS, or include ID and IP of the target host directly. For a query using ID, eDNS only needs to lookup the IP of the target host or the corresponding IDMS. To make it clear, we illustrate the types of query and response in Figure 2.

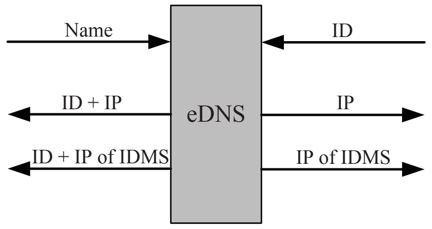

Fig. 2. Functions of Extended DNS.

C. *An Example of Name-ID-IP Lookup*

The previous subsection only considers mappings between identifier and locator, the procedure would be more complicated after it works with functions of traditional standard DNS, *i.e.*, mappings related to domain names. To illustrate the working procedure of eDNS, let us take one of the most complicated scenario, *i.e.*, a Name-ID-IP lookup, as an example. Assume there is a server whose name is *www.cernet.edu.cn*. This server applies an ID *y@tsinghua.edu.cn* from the IDMS *tsinghua.edu.cn*. Now there is a host who wants to visit this server and it only knows the server's name. So this host sends a query request with the server's name to local eDNS. To explain the whole procedure, let us assume the local eDNS does not cache any information of this server. As shown in Figure 3, the following steps are performed:

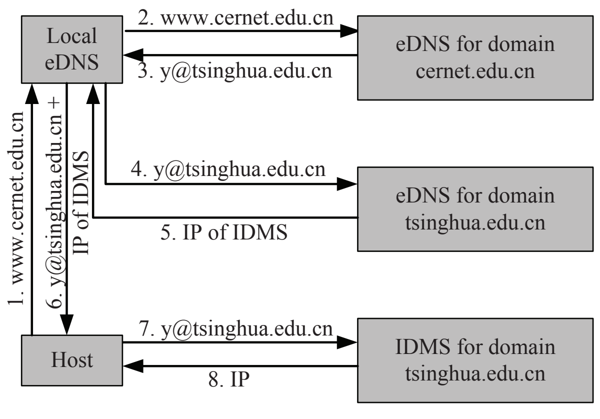

Fig. 3. An Example of Name-ID-IP Lookup.

1) the host requests local eDNS for a lookup using the name.
2) the local eDNS server receives the query, lookups recursively and finally finds the eDNS responsible for the domain *cernet.edu.cn*.
3) the eDNS of *cernet.edu.cn* responds the query request with the ID *y@tsinghua.edu.cn*. This is in fact a eDNS lookup from name to ID.
4) the local eDNS receives ID_y, retrieves the domain name of the responsible IDMS, and lookups recursively for

the IP address of the eDNS responsible for the domain *tsinghua.edu.cn*.

5) the eDNS of *tsinghua.edu.cn* lookups its database and responds the IP address of IDMS of *tsinghua.edu.cn*. This is an eDNS lookup from IDMS name to IP of IDMS.
6) the local eDNS summarizes the lookup results and sends response message to the host.
7) the host contacts the IDMS of *tisnghua.edu.cn*.
8) the IDMS returns the target IP address back to the host.

In the above example, the host only knows the server's name, so it has to initiate request using the name. If the host knows the server's ID, it can initiate request using ID instead of name, so the first two steps can be ignored. If the server has applied the eDNS of domain *tsinghua.edu.cn* to save its IP address directly, this eDNS can respond to the host with the server's address in the step 5, and the last two steps (step 7 and 8) are not needed.

III. Implementation

In Section II, we described the framework of our scheme. In this section, we will further introduce how we implement the two important elements, *e.g.*, eDNS and IDMS. We will focus more on the implementation of eDNS, since eDNS should be compatible with current DNS design, while IDMS is simply a database with interfaces to manage it and can be implemented in any way you like.

A. Implementation of eDNS

In the current DNS implementation, information for name resolving is formatted into zone files, which are loaded into memory to create a tree when DNS server is started. Current DNS implementation can be easily modified and extended to support the functions discussed in Section II. Our implementation is based on BIND v9.2.4.

1) Zone File and Resource Record: Mapping information are formatted into resource records and written into the zone file. Current DNS architecture defines several resource record types. For example, type "NS" represents authoritative name servers, type "A" represents an IPv4 address, which is used in entries mapping names to IPv4 addresses, type "PTR" represents a domain name pointer, which is used in entries mapping IPv4 addresses to names, *etc*.

In order to implement new functions of mapping name to ID, mapping ID to IP, or mapping ID to the IP address of responsible IDMS, in eDNS we define following new resource record types and illustrate their usages in Figure 4.

- IDMS: It defines the IP address of an authoritative IDMS server. For example, the first row of Figure 4 means the IP address of the IDMS server responsible for the domain *cernet.edu.cn* is 192.168.100.192.
- ID: It suggests that the right part of the record is a host ID. For example, the second row of Figure 4 means that host ID of the web server *www.cernet.edu.cn* is *y@tsinghua.edu.cn*, wherein "1" shows that the *first* dot in *y.tsinghua.edu.cn* should be replaced by @.
- A: This has been defined in current DNS to represent mapping entries from names to IP addresses. We define a new reserved label and extend the usage of this type for mappings from IDs to IPs. If the left part of the record ends with the new defined reserved label "id-ip", it means this is a mapping entry from ID to IP. For example, the fifth entry represents the IP address of ID *ftp@cernet.edu.cn* is 192.168.100.195. Later we will show why we define the reserved label "id-ip" for resource records from IDs to IP addresses.

Let us consider the example shown in Figure 3 again. The zone file for *cernet.edu.cn* should contain a record (*www ID 1 y.tsinghua.edu.cn.*). The zone file for *tsinghua.edu.cn* should contain a record (*IDMS 192.168.1.1*), which gives address of the responsive IDMS of the domain *tsinghua.edu.cn*. Then the host can contact this IDMS to lookup the location of the server *www.cernet.edu.cn*.

2) Red-black Tree for Lookups: When DNS server is started, all resource records in the zone file are loaded into an in-memory database, which is in fact a red-black-tree data structure.

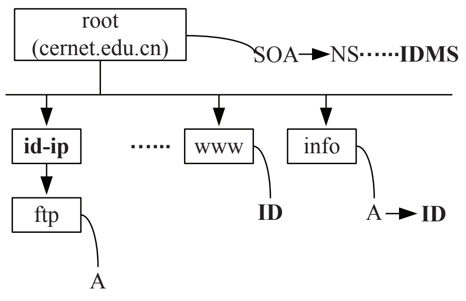

Fig. 5. Example: In-memory Red-black Tree Database for the Domain *cernet.edu.cn*.

Figure 5 presents the in-memory red-black tree for the zone file in Figure 4. Note that with the usage of "id-ip", all records related to identifier/locator separation, *i.e.*, mappings from ID to IP, would form a branch under the node "id-ip" in the red-black tree. When we search the tree to answer queries, we can treat IDs in a same way as names - IDs have been saved as domain names with a particular reserved label. Furthermore, traditional queries and new queries are in different branches, which makes the structure very clear.

3) Protocol Message Formats: Obviously, protocol message formats should also be extended to specify new query and response types. As suggested in [6], reserved *OPCODE* and *QTYPE* values can be exploited for new types of lookups.

OPCODE is a four bit field that specifies kind of query in query messages. The most well-known OPCODE are 0 for standard queries and 1 for inverse queries. In our eDNS, we further define 6 for queries using name and 7 for queries using ID.

QTYPE is a two octet code that further specifies the type of expected answer to the query. It is a superset of all data types defined in DNS, since some queries are asking for multiple

```
-----------------------------------------------------------------
                    IDMS        192.168.100.192       // IP of IDMS server of the domain

  www               ID    1     y.tsinghua.edu.cn.    // name -> ID (y@tsinghua.edu.cn)
  info              A           192.168.100.193       // name -> IP (for traditional query)
                    ID    1     info.cernet.edu.cn.   // name -> ID (info@cernet.edu.cn)
  ftp.id-ip         A           192.168.100.195       // ID (ftp@cernet.edu.cn)-> IP
-----------------------------------------------------------------
```

Fig. 4. Example: Zone File for the Domain *cernet.edu.cn*.

resource records, *e.g.*, AXFR is used in a request for a transfer of an entire zone. We define three new QTYPE, wherein two are new data types, *i.e.*, dns_rdatatype_id = 0xF001, dns_rdatatype_idms = 0xF002, and one is for queries for both ID and IP, *i.e.*, dns_rdatatype_idip = 256.

4) Processing Workflow: Roughly speaking, we do not need to modify the workflow of query processing. The only exception occurs when eDNS receives queries from name to IP address or both ID and IP address, *i.e.*, OPCODE=NAME and (QTYPE=IP or QTYPE=IDIP). As illustrated in Figure 3, eDNS needs to lookup two times, maybe even from two servers, to complete this kind of queries. The eDNS should be able to compose query message to initiate the second query according to the answer of the first query, and compose response message which includes two resource records after it receives the answer to the second query. This is a little similar to recursive DNS query, but local DNS sends the same query to different authoritative DNS servers in a recursive query, while eDNS needs to compose new query message in an eDNS IDIP query.

5) Discussion: Now let us evaluate the additional cost we incur on DNS infrastructure from two aspects, *i.e.*, the number of resource records brought in by identifier/locator separation, and the number of queries that eDNS should answer.

There are three kinds of new resource records in eDNS. The first kind is to state IP address of IDMS for each authority. For each authority, there is only one IDMS resource record in eDNS. The second kind of resource records is mapping from name to ID. Intuitively, mapping between a name and an ID (usually for mobile hosts) and mapping between the same name and an IP (usually for static hosts) would not appear in eDNS simultaneously. In other words, the number of resource records related to name is equal to the number of names, which means the second kind of resource record does not incur additional complexity. The third kind of resource record is mapping from ID to IP, which is in fact the responsibility of IDMS and eDNS is not forced to maintain them. Therefore we can control the number of these mappings in eDNS, and only use them for some relatively static hosts to accelerate eDNS lookup.

It is true that eDNS needs to answer much more queries than DNS since there would be a query once a host initiates a communication session using ID. However, this is solved at local eDNS by caching mechanism. After the local eDNS caches IDMS mappings, most queries from ID to IP can be answered by local eDNS and would not affect global eDNS system.

Therefore, our scheme of extending DNS for identifier/locator separation is feasible, and the additional cost on DNS is acceptable.

B. Implementation of IDMS

IDMS is responsible for distributing IDs upon endpoints' requests, and tracking these IDs' locations after distribution. It is simply a database of dynamic bindings between ID and IP address with two interfaces, *i.e.*, a management interface and a network interface. The management interface is operated by system administrators to deal with ID applications and ID revocations from network users; and the network interface receives and processes three kinds of messages from endpoints, *i.e.*, registration message which is to request IDMS to bind an ID and an IP, unregistration message which is to request IDMS to unbind an IP and an ID, and lookup message which is to request IDMS to lookup the IP of one ID.

The ID distribution of IDMS can be integrated with PKI mechanism to improve the communication security. When IDMS receives an ID application request from a network user, IDMS will generate a pair of public key and private key. IDMS reserves the public key, and the private key is given to the endpoint. Registration messages and unregistration messages are encrypted with the private key by the endpoint and decrypted by IDMS using the public key to prevent attacks.

IV. Related Work

In recent years, researchers have proposed many proposals to separate identifiers from locators. Among these schemes, Shim6 [2] [7], HIP [3] [8] and LISP [5] [9] received the most attentions and discussions.

Shim6 does not introduce any new name space of identifier. The endpoint identities in Shim6 are the initial addresses used between the two hosts. Although it is possible to extend Shim6 to support host mobility, current Shim6 assumes all available addresses are pre-defined HBAs, which cannot be true for mobile hosts.

The Host Identity Protocol introduces a new name space, *i.e.*, Host Identifier, which is the public key of an asymmetric key-pair. HIP is based on a Sigma-compliant Diffie-Hellman key exchange, using host identifier (*i.e.* public key) for mutual peer authentication. As a result, all HIP implementations must support the RSA/SHA1 public key algorithm, and should support the DSA algorithm.

To some extent, HIP facilitates host mobility. A HIP host can notify its peer of the new address by sending a HIP UPDATE packet, thereby enabling continuity of communications across IP address changes. However, in order to start the HIP exchange, the initiator node has to know how to reach the responder. HIP defines a rendezvous infrastructure to solve this issue [10]. The mobile node keeps the rendezvous server (RVS) continuously updated with its current IP address, so that other nodes can initiate a HIP base exchange with the IP address of the RVS, which will relay this initial communication. When a HIP node has registered with an RVS, it should record the IP address of its RVS in its DNS record [11].

Obviously, a HIP node without DNS name cannot record the IP address of its RVS in DNS, therefore it cannot be reached by other nodes during moving. Furthermore, there would be one more resource record for each HIP host, which incurs a lot of extra cost on DNS. Our scheme solves these issues by designing the format of identifier carefully. In eDNS, identifiers are aggregated according to their responsible authorities, therefore, only one resource record is necessary for all identifiers from one authority, which heavily reduces the number of new resource records. One the other hand, all host IDs are assigned by IDMS authorities, which can be used by network operators to facilitate network management functions such as accounting, security *etc*. IDMS in our scheme is similar to RVS in keeping IP addresses of mobile hosts. But IDMS does not relay any data packets for mobile hosts. IDMS is also an authority to assign and manage host IDs.

LISP is primarily designed to reduce the size of global routing table without considerations for endpoint mobility. When an endpoint moves, it requires changes to its EID, therefore other mechanisms are needed to maintain session continuity. Although LISP-MN enables endpoint mobility, it puts extra functions and costs on end systems. End users must apply for reserved mobile EIDs before it becomes a mobile node, which means mobility cannot be available for most end users. Each mobile node is viewed as a LISP site, and injects at least one dynamically changing mapping entry to the mapping database, which is also negative for the promotion of mobility service.

In [12], the authors also proposed that endpoints should have the freedom to choose where to store their mappings, *i.e.*, which mapping service provider (MSP) to use. However, they further require that endpoints should be able to change their MSPs without changing their identifiers. As a result, in their scheme, an endpoint must announce its MSP with its identifier so that other endpoints can initiate sessions with it. In our scheme, the identifier itself specifies its responsible MSP in its suffix. As a result, endpoints can find their MSP through eDNS lookups and do not need to announce addresses of their MSPs in data packets.

V. Conclusion

After it is pointed out that some challenges faced by the Internet are caused by the overloading of IP address semantics in IAB Routing and Addressing Workshop 2006, most researchers have agreed that the locator/identifer separation is beneficial for the Internet. However, till now there is no consensus on how to define the "identifier". Shim6 uses the initial addresses of the communication between the two hosts as identifiers. In HIP, Host Identifier is the public key of an asymmetric key-pair. In LISP, EIDs are in fact locators in edge networks.

In this paper, we propose a locator/identifier separation scheme in which identifiers are distributed by authorities to endpoints. The authorities are responsible for maintaining real-time locations of hosts with identifiers they distribute. We have implemented prototypes of elements in this scheme, and also deployed the system in an experimental network with tens of nodes. Our further work will be to try to deploy this scheme in real-world networks, conduct more experiments, and improve its performance according to feedbacks from experiment experiences. We believe this scheme can facilitate accounting, security and other network management tasks.

Acknowledgment

The authors would like to thank 863 Program (Project No. 2009AA01Z251, 2008AA01A303), 973 Program (Project No. 2009CB320502) and National Key Technology R&D Program (Project No. 2008BAH37B05).

References

[1] D. Meyer, L. Zhang, and K. Fall, "Report from the iab workshop on routing and addressing," RFC4984, September, 2007.

[2] E. Nordmark and M. Bagnulo, "Shim6: Level 3 multihoming shim protocol for ipv6," RFC5533, June 2009.

[3] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson, "Host identity protocol," RFC5201, April 2008.

[4] M. Bagnulo, "Hash-based addresses (hba)," RFC5535, June 2009.

[5] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis, "Locator/id separation protocol (lisp)," draft-ietf-lisp-12, http://tools.ietf.org/html/draft-ietf-lisp-12, April, 2011.

[6] P. Mockapetris, "Domain names - implementation and specification," RFC 1035, 1987.

[7] A. García-Martínez, M. Bagnulo, and I. Van Beijnum, "The shim6 architecture for ipv6 multihoming," *Communication Magazine*, vol. 48, pp. 152–157, September 2010.

[8] P. Nikander, A. Gurtov, and T. Henderson, "Host identity protocol (hip): Connectivity, mobility, multi-homing, security, and privacy over ipv4 and ipv6 networks," *Communications Surveys Tutorials, IEEE*, vol. 12, no. 2, pp. 186 –204, quarter 2010.

[9] L. Iannone, D. Saucez, and O. Bonaventure, "Implementing the locator/id separation protocol: Design and experience," *Computer Networks*, vol. 55, no. 4, pp. 948 – 958, 2011, special Issue on Architectures and Protocols for the Future Internet.

[10] J. Laganier and L. Eggert, "Host identity protocol (hip) rendezvous extension," RFC5204, April 2008.

[11] P. Nikander and J. Laganier, "Host identity protocol (hip) domain name system (dns) extensions," RFC5205, April 2008.

[12] H. Luo, H. Zhang, and M. Zukerman, "Decoupling the design of identifier-to-locator mapping services from identifiers," *Computer Networks*, vol. 55, no. 4, pp. 959 – 974, 2011, special Issue on Architectures and Protocols for the Future Internet.