A DHTs-Based Mapping System for Identifier and Locator Separation Network*

Shi Liu[1,2], Jun Bi[2], Yangyang Wang[2,3]
Tsinghua National Laboratory for Information Science and Technology（TNList）
[1]School of Software, [2]Network Research Center, [3]Dept. of Computer Science
Tsinghua University
Beijing 100084, China
liushi06@mails.tsinghua.edu.cn, junbi@tsinghua.edu.cn, wangyy-06@mails.tsinghua.edu.cn

***Abstract*—Today's Internet is facing routing scalability issues, which could degrade network performance seriously. To solve that, several solutions are currently being discussed. Among them, one promising approach is to set up a new architecture which separates the locator and the identifier roles of current IP addresses. A key question for the solution is how to provide an efficient and reliable service for mapping the identifiers to the locators in the new architecture. We propose a mapping system design based on Distributed Hash Tables (DHTs).**

Keywords-routing scalability; Internet; locator and identifier separation; DHTs

I. INTRODUCTION

Today's Internet is facing several important problems, and the routing scalability issue could be one of the most serious. Its symptom includes, but not limited to, rapid growth of DFZ (Default Free Zone) RIB (Routing Information Base) and FIB (Forwarding Information Base), longer routing convergence time and more cost for the requirements on power and core router hardware. In addition, the wide deployment of some new technologies, like IPv6, VPN and multi-homing, is making bad things worse. All the new technologies will inevitably accelerate the growth of routing table and exacerbate the situation.

The IRTF (Internet Research Task Force) and its Routing Research Group (RRG) are focusing on the internet scalability problem related to the inter-domain routing in their recent activities, and have got a conclusion that current architecture of Internet cannot provide support for a scalable routing system. A new architecture is needed, which should bring better scalability as well as providing support to existed equipments and applications. To achieve that target, several mechanisms are proposed. Most of them rely on the separation between the routing locators and the identifiers roles of IP addresses to slow down the growth of routing table size and alleviate the routing burden.

For the separation solutions, a key problem is how to set up an efficient, reliable and scalable system to map endpoint hosts' identifiers to their routable locators. Several designs have been proposed, most of which are designed as complements for the Locator/Identifier Separation Protocol (LISP) described in [1], like LISP-NERD [2], LISP-CONS [3], LISP-DHT [4], LISP-EMACS [5] and LISP-ALT [6]. Among them, LISP-DHT gives an assumption using Chord ring infrastructure to build a mapping system, but there has not been completed implementation and detailed design yet. So based on their research, we propose an improved infrastructure to build up a mapping system based on Distributed Hash Tables, which puts more emphasis on the scalability issue. It could provide reliable and robust service for the new architecture after the separation.

This paper is organized as follows. We first describe the new architecture after separating the identifier and locator roles of IP addresses, and the new routing procedure in section 2. Then we give a brief introduction about DHTs and the Chord infrastructure in section 3. In section 4 we will describe our design in detail. Finally several issues are exposed as future work.

II. THE NEW ARCHITECTURE

Currently, IP addresses are used as both identifiers to identify a host, and locators to assist routing. All endpoints and routers share one namespace. As a consequence, the global routing table grows with Internet user population. As more and more hosts join the Internet, the size of routing table grows unprecedentedly. That will cause longer routing convergence time and heavier burden on core routers. Besides the roles of IP addresses issues, the flat structure in use currently also means any single unstable network can flood the entire Internet with frequent updates.

To solve the problem, a new architecture was proposed to isolate the influence of new hosts' participations from global routing table and core router by splitting current Internet namespace into two independent but connected parts, a customer network and a transit network. The endpoint hosts stay in the customer network, while the routers move into the transit network. In the customer network, every endpoint host has one or more connections with ingress/egress routers on the edge of the transit network. All communication packets between two hosts should be transmitted through the transit network from egress routers to ingress routers.

The separation enables the isolation, but brings another problem. To support the new architecture, an efficient,

*This work was supported by the Specialized Research Fund for the Doctoral Program of Higher Education of China under Grant No.200800030034; the National Science and Technology Support Program of China under Grant No. 2008BAH37B02.

978-0-7695-3664-4/09 $25.00 © 2009 IEEE
DOI 10.1109/AFIN.2009.19

reliable and scalable mapping service should be provided, which can respond to a query including a given host identifier, and reply with a routable locator of some edge router. In other words, the problem is that we need a high-performance and scalable infrastructure to organize and store information, handle inquiries and distribute updates.

III. DISTRIBUTED HASH TABLES

A. Introduction to DHTs

From [7], DHTs (Distributed Hash Tables) are a class of decentralized distributed systems that could provide a lookup service similar to a hash table.

Many name-value pairs are stored in the DHTs, and any participating node can retrieve the value associated with a given name efficiently. Responsibility for maintaining the mapping from names to values is distributed among the nodes, in such a way that a change in the set of participants causes a minimal amount of disruption. This allows DHTs to scale to extremely large numbers of nodes and to handle continual node arrivals, departures, and failures. A key technique used to achieve these goals is that any node needs to coordinate with only a few other nodes in the system, so that only a limited amount of work need to be done for each change in membership.

As a conclusion, because of DHTs' inherent properties, such as self-management, self-configuration and good scalability, using DHTs to build the mapping system is a considerable and promising method.

B. Chord

Chord was developed at MIT [8]. It is one of the original Distributed Hash Tables protocols, and also a very classical and well-accepted one.

In the Chord protocol, node keys are arranged in a circle. Suppose the key space is the set of *m*-bit strings. The circle cannot have more than *2m* nodes.

Every node is assigned an ID, and has a successor and a predecessor in the ring. The successor to a node is the next node in the ring clockwise, while the predecessor of a node is the next node in the ring counter-clockwise. If there is a node for each possible ID, the successor of node 2 is node 3, and the predecessor of node 1 is node 0; however, normally there are not always continuous in the sequence. The nodes not existed will be skipped.

Since the successor (or predecessor) node may leave from the network (because of failure or departure), each node records a whole segment of the circle adjacent to itself, i.e. the *K* nodes preceding it and the *K* nodes following it.

A typical use of the Chord ring infrastructure for storage and retrieval might proceed as follows. Suppose the key space is the set of *m*-bit strings. To store a file with given filename in the DHT, the SHA1 hash of the filename is recorded as a *m*-bit key *k*, and a message put(*k*, data) is sent along the ring to find the node, whose id is the first number not smaller (or bigger) than *k*, where the pair (*k*, data) will be stored. Any other client can then retrieve the contents of the file by hashing filename to produce the key and looking up the associated node to find the data with a message get(*k*).

IV. MAPPING SYSTEM BASED ON DHTS

A. How to Store Mapping Information

The very first question for the mapping system design is to choose a suitable ID to be used in Chord ring. Here are some options:

1) Endpoint IP address: If we would like to store host A's mapping information, the simplest way is to hash A's IP address to produce a key, and then find the specified host where A's mapping information could be store. The problem is that, Chord circle cannot contain more than 2m nodes, but there are thousands of millions hosts in the Internet. To hold all these, based on our calculation, the m should be very huge, which will complicate the hash computing. For the same reason, there may also be a serious scalability problem with the Internet grows.

2) Provider Assigned ID: ISP will feel free to assign an ID to its clients. For the endpoint side, a subnet or a set of hosts could share the same ID, and store their mapping information to the same node. It is clearly more efficient than using endpoint IP addresses as the ID, but when the users change their ISP, they will have to depart from the circle first, renumber all their hosts, and join the Chord circle again. The cost of renumbering could be unacceptable.

3) Top IP address prefix: The top prefixes, which means the prefixes not included by other prefixes. We can hash a top prefix to produce a key, and store mapping information of all hosts sharing the prefix to the node specified by the key. To avoid the renumbering issue, ISP-dependent IP addresses will be preferred. Using Top IP address prefix as the ID for Chord ring can give attention to both efficiency and scalability.

For the consideration of future topology change, the mapping information could be organized in a tree with more specific IP address prefixes, so that when the topology of the network changes, it will be easier to insert the new top prefixes to the Chord ring, as shown in Figure 1.

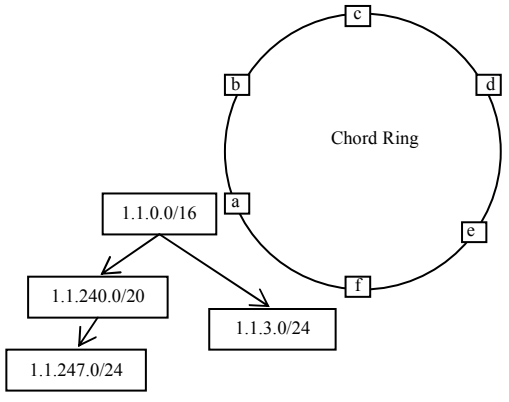

Figure 1. Store mapping information with Chord ring

A mapping server, which could be seen as one node at the Chord ring of the mapping system, stores mapping

information necessary for the communications. So if there is not a valid redundancy mechanism, the system cannot be reliable. By making a reference to Chord's mechanism for nodes' joining and departing, we set up a backup mechanism that every node should keep a copy of the mapping information stored in its successor and predecessor in case of sudden node failure. Nearby nodes should synchronize each other's information in a specific interval

B. Basic Operations

1) Initiate mapping system: At very first, we need certain device which can get routing information from core routers, process them and retrieve the mapping items, such as a tuple of (IP Prefix, Tunnel Points' IP, other information). And the devices will inject these mapping items into the mapping systems. The mapping system should be formed by a lot of mapping servers (which are called entities in [4]) owned by the domains with top IP address prefixes, just like the domain name service systems.

2) Query and answer: As shown in Figure 1, when node "e" query an IP address prefix, such as "1.1.247.0/24". Firstly, it should locate node "a" by hashing and looking up. And then the query request will be forward to node "a". Node "a" can go through the tree down and find the mapping item associated with the specific prefix "1.1.247.0/24". For the last step, node "a" should reply the query with the mapping result to node "e".

3) Update mapping items: The update operation includes Announcement (announcing new mapping items) and Withdrawal (withdrawing old mapping items). Most of the procedure is just like query and answer operation. The difference is that the identity of the update initiator must be verified in case of malicious attacks or wrong notifications.

4) Node participation and departure: When a new node, which could be a mapping server belonging to some domain, join the mapping system, its top prefix will be calculated to produce a value. By tracing the value clockwise along the ring, we can find the expected predessor and successor. The new node will be inserted into the ring between them. Then the three nodes will synchronise their mapping information. Similarly, if a node departs from the ring suddenly, its predessor and successor node should rebuild the connection between them, and synchronise mapping information.

C. Mobility Support

As mentioned before, all mapping information is organized with the top IP address prefix. But if one host is mobile, its mapping information could also be dynamic. To support the mobility situation, we propose a mechanism called "move-and-announce". If a host would like to move, which will cause a prefix change, it should record the previous IP and node information. When the host move to another place, and get an address with a different prefix, the node storing its mapping information should inform the previous node to mark the mapping item as "moved". If there is a query for that item sent to the previous node, it will be forwarded to the node which stores the required information currently.

D. Security

In our mechanism, an outstanding feature is that all information which is necessary for A's routing is stored in B's servers. That will bring a potential danger that if the domain has an absolutely control on its mapping server, it will be free to modify the mapping items and hijack the mapping. So it is necessary to introduce a light-weight security mechanism.

To make sure that only authenticated user can modify the mapping information, we can limit the access rights. Only authenticated user can add, delete or update mapping items. Other users, even the local administrator, only have the right to read mapping information.

E. Other Issues

1) Cache: Frequent inquiries and lookup will bring cost and latency, so a better way is to store the mapping information in local cache to achieve higher efficiency. We can weigh each cached entry according to its hit frequency.

2) Traffice engineering and flexible routing support: Our mechanism can provide support to traffic engineering and flexible routing by improving the storage structure and lookup mechanism. More entries could be added for one key, and a weight could be introduced to assist query.

V. Conclusion and Future Work

We propose a robust, scalable and extensible design for a mapping system supporting identifier/locator separation. Comparing with other existed proposals, it use decentralized structure to avoid the weak link, and more accurate distribution mechanism to reduce the overhead brought by updates.

For the future work, we will make an implementation to test our mechanism and improve the design. Some issues, including a light-weight security mechanism, the implement method of cache, and a better algorithm for splitting the key space and distributing the mapping data, will also be explored. There are some methods presented in other mechanisms, like the update distribution method in LISP-NERD, which we can use to improve our design.

Besides that, an evaluation based on the implementation about the cost and benefits brought by our mechanism is also planned.

References

[1] D. Farinacci, V. Fuller, D. Oran and D. Meyer, "Locator/ID Separation Protocol," draft-farinacci-lisp-09.txt, IETF Network Working Group, October 2008.

[2] E. Lear, "NERD: A Not-so-novel EID to RLOC Database," draft-lear-lisp-nerd-04.txt, IETF Network Working Group, April 2008.

[3] S. Brim, N. Chiappa, D. Farinacci, V. Fuller, D. Lewis and D. Meyer. "A Content distribution Overlay Network Service for LISP," draft-meyer-lisp-cons-04.txt, IETF Network Working Group, April 2008.

[4] L. Mathy and L. Lannone, "LISP-DHT: Towards a DHT to map identifiers onto locators," ACM CoNEXT 2008 ReArch Workshop, Madrid Spain, December 2008.

[5] S. Brim, D. Farinacci, D. Meyer and J. Curran. "EID Mappings Multicast Across Cooperating Systems for LISP," draft-curran-lisp-emacs-00.txt, IETF Network Working Group, November 2007.

[6] D. Farinacci, V. Fuller and D. Meyer. "LISP Alternative Topology (LISP-ALT)," draft-fuller-lisp-alt-03.txt, IETF Network Working Group, October 2008.

[7] Wiki page, http://en.wikipedia.org/wiki/Distributed_hash_table.

[8] Chord. http://pdos.csail.mit.edu/chord/