Demonstrating a Scalable Name Resolution System for Information-Centric Networking

Jungha Hong
ETRI
Daejeon, Korea
jhong@etri.re.kr

Woojik Chun
Hankuk University of Foreign Strudies
Yongin-si, Koreae
woojikchun@gmail.com

Heeyoung Jung
ETRI
Daejeon, Korea
hyjung@etri.re.kr

ABSTRACT

In implementing Information-Centric Networking (ICN), an efficient Name Resolution System (NRS) is required since names assigned directly to the named data objects (NDOs) are location independent. On designing NRS, the most important challenge is scalability on the ever increasing number of NDO, which becomes even more critical when flat names are used in ICN. In this demonstration, we present a scalable NRS with flat names for ICN, where bloom filters are utilized to aggregate flat names. Specifically, we show the feasibility of our NRS with flat names and highlight the benefits of utilizing bloom filters.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design

General Terms

Design, Implementation, Experiments

Keywords

Information-Centric Networking; name resolution system; flat name; bloom filter

1. INTRODUCTION

In contrast to the host-centric networking of the current Internet, the primary communication object in information-centric networking (ICN) is named data object (NDO). In implementing ICN, an efficient Name Resolution System (NRS) is required since names assigned directly to the NDOs are location independent. This shift raises scalability issues to a new level. The current Internet is addressing on the order of 10^9 nodes, whereas the number of addressable ICN objects is expected to be several orders of magnitude higher. Thus, scalability on the ever-increasing number of NDO becomes one of the most important challenges on designing NRS.

In this demonstration, we propose a scalable NRS for ICN, where flat names are used. Flat names provide some advantages compared to hierarchical ones, such as higher flexibility, simpler name allocation and benefits in terms of persistency and privacy [1-2]. On the other hand, scalability becomes more critical challenge on designing NRS with flat names.

ICN'15, September 30 - October 02, 2015, San Francisco, CA, USA
ACM 978-1-4503-3855-4/15/09.
http://dx.doi.org/10.1145/2810156.2812617.

In order to address the scalability issue in designing the NRS with flat name, we need to aggregate names in any form. One popular technique for flat name is Distributed Hashing Table (DHT) based approach [3-5], where multiple servers form circular linked list and the bindings are stored in the appropriate server. However, the DHT technique has some drawbacks; the binding between name and locator must be stored in a particular server other than the owner's server, which causes a serious trust problem related to the authority issue and lookup request may be propagated through the long paths.

In this demonstration, we construct NRS hierarchically and exploit bloom filters (BFs) to aggregate flat names. We will demonstrate the benefits of utilizing BFs by our prototype. We are also expecting to show how to achieve the design goals of our NRS such as scalability, locality, and low latency.

2. SYSTEM DESIGN

We construct NRS hierarchically by defining a network of NRS servers, which consists of a forest by several disjoint trees as shown in Figure 1. The network of NRS servers is defined by both parent-child and peering relationships. Each NRS server consists of a name lookup table and BFs for itself, from children, and from peers as depicted in Figure 1. A certain name can have more than one locator.

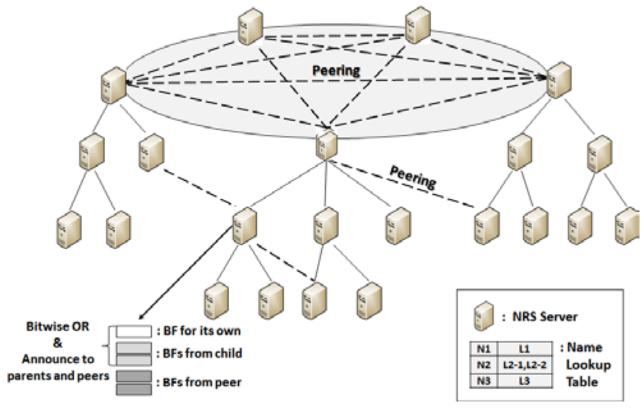

Figure 1. NRS structure.

Instead of announcing the whole list of names, each NRS server announces only one BF to parent and peer servers, which is formed as the union of BFs from all children and itself by bitwise 'OR' operation.

One of the major benefits of BF is a fixed constant time of insertion and search which is completely independent of the number of names already in a BF.

2.1 Name Registration

In this demonstration, it is allowed that a communication entity can be registered in any arbitrary NRS server since names have no structure. Because of this property, locality can be easily supported. When a communication entity is registered in a NRS server, the registration information is extracted from its name using the hash functions and inserted into its own BF first and then the NRS server updates BFs for its parents and peers, where this recursion holds until BFs at the top of trees are completely updated. When names are deleted from the lookup table, we use periodic refresh technique to reflect the deletion on BF since BF cannot handle the deletion by itself.

2.2 Locator Lookup

Requestor sends locator (LOC) lookup to its default NRS server first. On receiving LOC lookup to a NRS server, it first searches the corresponding name on its own BF. If the search fails, then it searches the name on BFs for its child and peer NRS servers. If none of the BFs return a positive answer, the LOC lookup is forwarded to its parent NRS server. On the other hand, if any BF returns a positive answer, the LOC lookup is forwarded to every NRS server that corresponds to BFs with positive answers. We note that because of the false positives of the BF, multiple BFs may return positive answers. This search is done recursively and LOC information can be eventually found. In this demonstration, LOC lookup carries the requestor's information so LOC lookup reply can be sent to the requestor directly.

2.3 Locator Update

In this demonstration, we divide LOC update into tree types: add, delete, and replace. LOC update is forwarded into the server where the name is actually stored by the name search such as processed in LOC lookup. If a name has multiple LOCs, then they are stored as a set of LOCs for the name. Because of this function, mobility and multi-homing are inherently supported.

3. DEMONSTRATION

In this demonstration, we present a scalable NRS with flat names for ICN, where BFs are utilized to aggregate the flat names. We have created prototypes for our NRS: NRS server, top server, and client. Although all NRS servers perform the same functions, we separate top server from the others for convenient implementation. We have utilized the parallel process of a graphics processor unit (GPU) to accelerate the performance of BF check at each NRS server resulting in low latency.

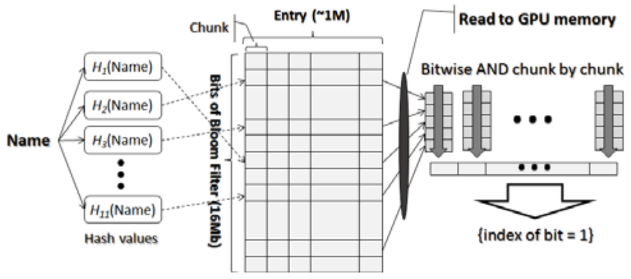

Figure 2. Overview of the algorithm for GPU usage.

Figure 2 shows the overview of the algorithm for the GPU usage. The main idea of the algorithm is to enable to extract only the corresponding bits for the given name check from all BFs at each server to GPU memory and check the extracted bits in parallel to see if any chunk gives 1 by bitwise 'AND' operation. In this demonstration, we use 16Mb BF size and 11 hash functions to keep the false positive probability less than or equal to $4.586 * 10^{-4}$ assuming that each BF can have information at a maximum of 10^6 names. We have used the static tree structure of NRS which is managed by configuration files of each server. We have also implemented the NRS without using GPUs to see the effect of the GPU usage on performance.

We demonstrate the feasibility of our NRS with flat names and highlight the benefits of utilizing BFs by our prototype. We show how scalable our NRS for ICN is even with flat names. We are also expecting to show how to achieve the design goals of our NRS such as scalability, locality, and low latency.

4. REFERENCES

[1] A. Ghodsi, T. Koponen, J. Rajahalme, P. Sarolahti, and Shenker, "Naming in Content-Oriented Architectures," In Proceedings of the SIGCOMM ICN'11, August 19, 2011, Toronto, Ontario, Canada.

[2] International Telecommunication Union (ITU), "ITU-T Recommendation Y.3031 - Identification framework in future networks," available at: http://www.itu.int/rec/T-REC-Y.3031-201205-P/en, 2012.

[3] O. Hanka, C. Spleiss, G. Kunzmann, and J. Eberspacher, "A novel DHT-based network architecture for the next generation internet," Eighth International Conference on Networks, Cancun, Mexico, March 2009.

[4] H. Luo, Y. Qin, and H. Zhang, "A DHT-Based Identifier-to-Locator Mapping Scheme for a Scalable Internet," IEEE Transactions on Parallel and Distributed Systems, October 2009.

[5] L. Mathy and L. Iannone, "LISP-DHT: Towards a DHT to map identifiers onto locators," in ReArch'08. Madrid, Spain: ACM, December 2008.