ICNRG
Internet Draft
Intended status: Informational
Expires: January 2016

J. Hong
ETRI
W. Chun
HUFS
H. Jung
ETRI

July 06, 2015

Bloom Filter-based Flat Name Resolution System for ICN
draft-hong-icnrg-bloomfilterbased-name-resolution-04.txt

Status of this Memo

Copyright Notice

Abstract

In information-centric networking (ICN), uniquely identifiable and location independent names are assigned directly to the named data which raises scalability issues and they get even worse with flat names. Accordingly, name resolution system required for lookup-by-name routing in ICN has to be designed to scale, also considering mobility support. In this draft, a bloom filter-based flat name resolution system (B-NRS) is proposed where the bloom filter as an aggregated form of names and hierarchical structure of the B-NRS are exploited to address the scalability issues.

Table of Contents

1. Introduction

In contrast to the host-centric networking in the current Internet, the primary communication object in information-centric networking (ICN) is named data, where uniquely identifiable and location independent name is assigned directly to the named data. 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 [ICNRG charter]. Accordingly, name resolution system required both for lookup-by-name routing in ICN [ICN Challenges] and for ICN-IoT architecture [ICN-IoT] has to be designed to scale, also considering mobility support.

In this draft, we propose a bloom filter-based flat name resolution system (B-NRS) which maintains and resolves the binding between names and locators, i.e. B-NRS takes a name as its input and produces the locator sets that the name is currently associated with. We assume that the locator independent names are flat since the flat names provide some advantages compared to hierarchical ones, such as higher flexibility, simpler name allocation and benefits in terms of persistency and privacy [Ghodsi, ITU]. On the other hand, scalability becomes the most important challenge on designing the NRS supporting flat names. It is because of the ever increasing number of names in the network and no possible way to compactly represent the flat names such as the aggregation in IP addresses.

In order to address the scalability issue in designing the NRS for flat name, we need to aggregate names in any shape of type. One popular technique for flat name is Distributed Hashing Table (DHT) based approach [Hanka, Luo, Ahlgren, Mathy], 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 must be stored in a server other than the owner’s, which causes a serious trust problem related to the authority issue and lookup message may be propagated through the long paths.

In this draft, to overcome the drawbacks of DHT, we exploit the bloom filter as an aggregated form of names and hierarchically construct the B-NRS. One of the major benefits of the bloom filter is a fixed constant time of insertion and search which is completely independent of the number of names already in the set. Another important and powerful property of bloom filter is the efficient support for union of bloom filters with the same size and set of hash functions which can be implemented with bitwise OR. However, bloom filter also has some drawbacks; false positive and no member deletion. Although there is no way to get rid of the false positive, it can be minimized by choosing the right parameters. The deletion problem is also taken care by periodic reconstruct of the bloom filters or by using variants of the bloom filter such as the counting bloom filter.

We note that the B-NRS in this draft does not require any specific mechanism for registering names, since names have no structure and can be registered to any B-NRS server with no constraint. Thus, the B-NRS needs only lookup mechanism. Whereas in the DHT-based system, the lookup message for a name is forwarded by the same way how to register the name.

2. NRS Requirements

Name resolution system (NRS) may become the bottleneck of the network when the signaling overhead of the location update and lookup becomes very large. Thus, the NRS must provide fast update and lookup for good performance since its basic functionality is to return the current locator for a given name. The NRS also must be secure and resilient because there is no way to respond to the querying message if the NRS is attacked. Obviously, the NRS must be scalable to the number of the ever-increasing ICN objects, i.e. names. Therefore, in this section, we discuss such requirements of the NRS.

2.1. Scalability

In ICN, the primary communication object is named data, where uniquely identifiable and location independent name is assigned directly to the named data. This raises scalability issues to a new level. The current Internet is addressing even on the order of 10^9 nodes, whereas the number of addressable ICN objects is expected to be several orders of magnitude higher considering sensor data, vehicular, Internet of things, etc. Accordingly, the NRS should be able to fully cover the ever-increasing number of ICN objects.

2.2. Fast resolution

A fundamental problem with any global query server network is that the requestor who sends the name resolving request may significantly delay or drop the initial packet of a new session if the resolution time gets too long. Thus, the resolution time should be sufficiently low so it does not affect much the overall system performance.

2.3. Fast update

When a named date moves and changes its point of attachment to Internet or a multi-homed device shuts down one of its physical interface, it needs to update the old information with the new one or delete the deprecated information in NRS. Thus, the NRS should adapt quickly with such changes.

2.4. Resilience

If the NRS fails, there is mostly no way for the requestor to reach other end information since the requester knows only its names. Therefore, the NRS must not fail.

2.5. Security

The NRS can be a potential target for attacks such as denial-of-service attacks. These types of attacks are difficult to prevent. Thus, updates to the NRS or responses from NRS server should be authenticated.

3. Bloom Filter-based Flat Name Resolution System (B-NRS)

We propose a bloom filter-based name resolution system (B-NRS) for supporting flat name which maintains and resolves the binding between names and locators.

3.1. System structure

We construct the B-NRS hierarchically by defining a network of B-NRS servers, which consists of a forest by several disjoint trees. The network of B-NRS servers is defined by both parent-child and peering relationships.

Figure 1 is an example of the B-NRS structure which consists of 8 B-NRS servers forming a tree, where there exists the peering relationship between S2 and S3. The peering relationship is allowed for better performance by reducing the overhead for the B-NRS at the top of the tree. A leaf B-NRS server knows every single name/locator pair that it manages but nothing else. The intermediate B-NRS servers know the name/locator pair for all names that are directly registered to them and also possess only information about the names that their descendant and peer B-NRS servers manage. Although there is a single tree in figure 1, if we assume there are several trees forming a forest, then the B-NRS servers are fully peered at the top of the trees. This means that each server shares its knowledge of all names that it manages with its peers.

We note that we have been very careful in distinguishing between the name/locator pair information and the name information. This distinction is necessary to provide a different level of information abstraction, which is naturally achieved through the hierarchical B-NRS structure and the use of bloom filters.

```
                       +----+
                       | S1 |
                       +----+
                      /      \
                     /        \
                    /          \
                   /            \
                  /              \
                 /                \
                /                  \
           +----+                  +----+
           | S2 |******************| S3 |
           +----+                  +----+
           / | \                      /\
          /  |  \                    /  \
         /   |   \                  /    \
        /    |    \                /      \
       /     |     \              /        \
      /      |      \            /          \
     /       |       \          /            \
   +----+  +----+  +----+    +----+        +----+
   | S4 |  | S5 |  | S6 |    | S7 |        | S8 |
   +----+  +----+  +----+    +----+        +----+

   Legend:

   +---+
   | S |  B-NRS Server
   +---+
   -----  Parent-child relationship

   *****  Peering relationship
```

Figure 1. An example of B-NRS structure

3.2. B-NRS Server Components

A B-NRS server consists of a name lookup table and multiple bloom filters.

3.2.1. Name Lookup Table

Name lookup table stores the binding between names and locators for all names which are directly registered to the BRS server. The associated locator for a certain name can be more than one. So, the locator information is stored as a set shown in table 1. Name lookup table takes a name as the input and produces its associated locator sets as the output.

Table 1. Lookup table

Name	Locators
N1	LOC1
N2	LOC2-1, LOC2-2
N3	-
N4	LOC4-1, LOC4-2, LOC4-3

3.2.2. Bloom Filter

We utilize bloom filters as an aggregated form of names at each B-NRS server. B-NRS servers announce their name set to the other B-NRS servers. Instead of announcing the whole list of names, bloom filter as an aggregated form of names is announced. When announcing its name set to its peers or parents, the B-NRS server announces the union of name sets of all child B-NRS servers. Union of child name sets can be built by using the characteristic of bloom filer that bloom filter for union of sets can be built merely by bitwise ‘OR’ operation on all the sets.

Thus, each B-NRS server stores bloom filters for itself, from children, and from peers depicted in figure 2. The B-NRS server stores n+m+1 bloom filters in figure 2, where n is the number of child B-NRS servers and m is the number of peer B-NRS servers.

We note that the forest of B-NRS servers retains the loop-free property for the use of bloom filter.

```
                     /  -------------------------   \
                    /   | BF for its own        |    \
                   /   --------------------------     \ Bitwise OR
  +----------------+ /    --------------------------     / To Parents and Peers
  | B-NRS Server   |      | BFs from Child 1 to n   |   /
  +----------------+ \    --------------------------   /
                    \   --------------------------
                     \  | BFs from Peer 1 to m    |
                      \ ---------------------------
```

Figure 2. B-NRS server components

3.3. Key Operations

3.3.1. Name Registration

When a communication entity attempts to join the network, it must register itself in at least one B-NRS server. In this draft, it is allowed that the communication entity can be registered in any arbitrary B-NRS server since names have no structure.

Upon receiving the registration request from the communication entity, the B-NRS server registers the name to its lookup table. The locators for the name are stored in the table when the communication entity for the name is actually present into the network. We separate this as the operation of locator update from the name registration.

The name registration is along with bloom filter update. When a communication entity is registered in a B-NRS server, the registration information is extracted from its name using the hash functions for its bloom filter and inserted into its own bloom filter first and then the B-NRS server updates bloom filters for its parents and peers, where this recursion holds until bloom filters at the top of trees are completely updated.

Figure 3 shows an example of the name registration and bloom filter updates, where a new name is registered at the B-NRS server, S4. It inserts information of the new name first into its own bloom filter and updates its parent, S2. Then, S2 updates its parent, S1 and its peer, S3.

When names are deleted from the lookup table, we need to adopt a certain mechanism to update the bloom filters for the deletion since bloom filter cannot handle the deletion by itself. Thus, we use the periodic refresh technique that bloom filters with registered names are rebuilt periodically and followed by bloom filter updates.

```
             (3)BF
             Update    +----+
              -------->| S1 |
             |         +----+
             |        /      \
             |       /        \
             |      /          \
             |     /            \
             |    /              \
             |   /                \
   (2)BF     |  /                  \
    Update +----+  (3)BF Update     +----+
     ----> | S2 |------------------>| S3 |
    |      +----+*******************+----+
    |      / | \                      /\
    |     /  |  \                    /  \
    |    /   |   \                  /    \
    |   /    |    \                /      \
    |  /     |     \              /        \
    | /      |      \            /          \
    |/       |       \          /            \
   +----+  +----+  +----+    +----+        +----+
   | S4 |  | S5 |  | S6 |    | S7 |        | S8 |
   +----+  +----+  +----+    +----+        +----+
     ^^
     ||
     || (1)Name registration
     ||
```

Figure 3. Name registration and BF update

3.3.2. Locator Update

When a communication entity actually presents in the network, the locator update is occurred, where the gateway sends the locator update message to the correspondent B-NRS server and the locator associated with the name is stored in the lookup table. If the name has multiple locators, then they are stored as a set of locators for the name. Through the bloom filter test of the name, the locator update messages are forwarded into the lookup table where the name is actually stored.

When the communication entity depresents from the network, the locators for the name is deleted from the lookup table by the locator update message as well. Table 1 shows the depresence of entity for the name, N3. We note that changing locators has no effect on the structure of the B-NRS and mobility is easily supported.

3.3.3. Locator Lookup

The lookup operation is to find the locator information for a given name. The simplest case is when the source object tries to communicate with the destination object registered in the same B-NRS server. B-NRS server always searches for the destination name in its own lookup table first so the locator information is acquired at the first lookup in such a case.

A harder, but more interesting, case is when the destination object is registered in the other B-NRS server with the source object. In this case, the B-NRS server would quickly learn that the destination object is not registered in the same B-NRS server by a simple search of its lookup table. Then, it searches bloom filters for its child and peer B-NRS servers. If none of the bloom filters return a positive answer, the lookup request message is forwarded to its parent B-NRS server. On the other hand, if any of bloom filters return a positive answer, the lookup request message is forwarded to every B-NRS server that corresponds to the bloom filters with positive answers. We note that because of the false positives of the bloom filter, multiple bloom filters may return positive answers.

This search is done recursively, and the locator information for the destination name can eventually be found. Once the locator information is found, it is delivered to the source object by the lookup reply message which takes the reverse path of the lookup request message.

Figure 4 is an example of lookup and registration processes where the lookup message for a name which is registered at S8 is received by S4. Then, the lookup message is forwarded to S2. Since S2 is peered with S3, S2 forwards it to S3 not to S1. S3 forwards it to S8.

The reply message takes the reverse path of the lookup request message, i.e., S8->S3->S2->S4.

```
                                 +----+
                                 | S1 |
                                 +----+
                                /      \
                               /        \
                              /          \
                             /            \
                            /              \
                           /                \
                          /                  \
           (2)Lookup +----+  (3)Lookup        +----+ (4)Lookup
               ----> | S2 |<----------------->| S3 |<------
              |      +----+*******************+----+      |
              |      / | \   (6)Reply           /\        |
              |     /  |  \                    /  \       |
    (7)Reply  |    /   |   \                  /    \      |(5)Reply
              |   /    |    \                /      \     |
              |  /     |     \              /        \    |
              | /      |      \            /          \   |
              v/       |       \          /            \  v
   (1)Lookup +----+  +----+  +----+    +----+       +----+
   <-------->| S4 |  | S5 |  | S6 |    | S7 |       | S8 |
   (8)Reply  +----+  +----+  +----+    +----+       +----+
```

Figure 4. Lookup and reply

4. Comparison of B-NRS with Other NRSs

One of the critical challenges in designing NRS is scalability due to the ever increasing number of names. In order to overcome this issue, names need to be distributed and also aggregated in any shape of type especially for flat names. One popular technique to distribute and aggregate names is to use DHT (Distributed Hash Table). However, DHT has several drawbacks such as ownership, deployment, locality, etc. Thus, we exploit the bloom filter as an aggregated form of names and hierarchically construct the NRS.

As illustrated in figure 5, NRS can be roughly divided into two types: centralized vs. distributed. Then, the distributed type can be divided again into two approaches: DHT-based vs. all else. DMap (Direct Mapping) [DMap] and MDHT (Multiple DHT) [MDHT] are examples of DHT-based approach. DMap is proposed by MF (MobilityFirst) which is one of the Future Internet architecture projects funded by NSF in US and MDHT is by SAIL (Scalable and Adaptive Internet Solutions) which is an EU-funded project. B-NRS belongs to the distributed type but not DHT-based approach.

```
  *==================== NRS ===================*
 |                                             |
 |    *=========================*              |
 |   *       Distributed        *  Centralized |
 |  *                            *             |
 | *    *==========*              *             |
 | *  *  DHT-based *              *             |
 | *  * o MF-DMap   *  o B-NRS   *             |
 | *  * o SAIL-MDHT *            *             |
 |  *   *==========*              *             |
 |   *                         *               |
 |    *=======================*                |
 |                                             |
 *=============================================*
```

Figure 5. A simple Venn diagram categorizing NRS

Table 2 presents the comparison of B-NRS with DMap and MDHT in respect of scalability, lookup latency and locator update.

For scalability, we compare how many names can be scalable for each NRS. DMap assumes that the number of names is 5*10^9, whereas MDHT and B-NRS assume that it is 10^15.

We define the lookup latency as the multiple of the number of hops, H, and the processing time per hop, T(N), which is proportional to the number of table entries, N. The lookup latencies for both DMap and MDHT are increasing proportionally to the number of hops and the number of table entries at each hop since the table lookup is processed at each hop. However, the lookup latency for B-NRS is dependent only to the number of hops since BF takes a fixed constant time, C, for searching. Even though each B-NRS server has several bloom filters, they are independent to each other and can be parallelized in a hardware implementation.

For locator update, we look at the staleness. Both DMap and MDHT do the location update periodically so the staleness occurs during it is not updated. However, the staleness for B-NRS occurs with probability 0 since it does the location update in real time.

Table 2. Comparison of B-NRS with DMap and MDHT (N and H are the number of table entries and hops, respectively and C is a constant.)

Design goal	Scalability	Lookup latency	Locator update
Metric	number of names	number of hops * processing time per hop	Staleness
MF-DMap	~5*10^9	H*T(N)	periodic update: occur
SAIL-MDHT	~10^15	H*T(N)	periodic update: occur
B-NRS	~10^15	H*C	real time update: occur with probability 0

5. Implementation Issues

Bloom filter has the well-known drawbacks such as false positive and no membership deletion. However, the false positive can be minimized by choosing the right parameters and the deletion problem can also be taken care by adopting a certain mechanism to update the bloom filters for the deletion such as the counting bloom filter, periodic reconstruct of bloom filter, etc.

5.1. False Positive

The width of a bloom filter is directly related to the false positive rate for fixed number of hash functions, the length of the bloom filter is inversely proportional to the false positive rate. Although a lengthier bloom filter is ideal for minimizing the false positive rate but increasing the B-NRS search efficiency, it creates a burden when filter information are exchanged among B-NRS servers.

In addition, since a leaf B-NRS server has a smaller number of names that it needs to manage, it makes sense to use a smaller bloom filter than the B-NRS servers at the higher level of the B-NRS hierarchy. However, the variable bloom filter length approach must be done with care since the key property, union of bloom filter via bit-wise AND operation, may be lost when variable length bloom filters are used.

5.2. Membership Deletion

One of the main advantages of the bloom filter is that data insertion and search can be done in a constant time. However, its major drawback is that a bloom filter does not have an efficient method of supporting data deletion. Of course, there are variants of the bloom filter to overcome the deletion issue. For example, the counting bloom filter supports the deletion by associating a counter to every bit of the bloom filter, where data insertion corresponds to incrementing the counters associated with the bits; data deletion to decrementing the counters; and query to checking whether the counters are positive. However, since each counter needs to have sufficient number of bits to prevent overflow; thus, it is a less space efficient than the traditional bloom filter. The space efficiency is critical to our B-NRS since bloom filters are exchanged among B-NRS servers and it is directly proportional to the size of exchanged control messages.

Because of this drawback of deletion of bloom filter, IMS needs to be carefully designed to support dynamic registration and deregistration of communicating entity.

In one extreme case, even if the de-registration were to be completely ignored by the B-NRS, the B-NRS would eventually be able to find the locator for a given name. This method will generate the fewest number of control messages (bloom filter updates) but the query would become inefficient since this would significantly increase the false positive rates.

The other extreme case would be to update the entire B-NRS whenever there is a single de-registration. Although this method would have the lowest false positive rates, and thus, would have the lowest average number of queries to find the name/locator pair, it would have a very high control message load since there would be a lot of bloom filter exchanges among B-NRS servers.

Certainly, the B-NRS will operate within these two extreme bounds, and the optimal rate is a design parameter in building the B-NRS system.

B-NRS overcome the deletion issue by periodically rebuilding bloom filters using the shadow memory, so called periodic refresh. The refresh frequency can be a day, a week, a month, etc. When B-NRS is refreshed, names in a name lookup table are inserted into the new bloom filter at a time and the merged bloom filters by bitwise OR are announced to parent and peer B-NRS servers. For better performance, the lossless compressed bloom filter can be used to announce the merged bloom filter. We note that the false positive probability certainly increases until all bloom filters are replaced by new bloom filters.

6. Implementation of B-NRS

We have created prototypes for our NRS: NRS server, top server, and client. Although all B-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 B-NRS server resulting in low latency.

We have used an 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 implementation, we use 16Mb BF size and 11 hash functions to keep the false positive probability less 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 NRG without using GPUs to see the effect of the GPU usage on performance.

6.1. Protocol Message

We keep the flat name size as 24 bytes and use the UDP communication with port number, 7979 in the implementation. Prot in protocol messages is the protocol type of 1 byte size. Locator is defined as a variable length string.

O Name registration

```
+-----------------------------------------+

| Prot |            Name                  |

+-----------------------------------------+
```

O Locator update

Locator update message is divided into three types: Add, Delete, and Replace.

```
+----------------------------------------------------------+

| Prot | Mode |  Name  |  Locator length  |    Locator    |

+----------------------------------------------------------+
```

Mode is the type of locator update.

O Locator lookup

```
+-----------------------------------------+

| Prot |             Name                 |

+-----------------------------------------+
```

O Name deregistration

It deletes the name and the corresponding locators from name lookup table.

```
+-----------------------------------------+

| Prot |             Name                 |

+-----------------------------------------+
```

O BF update

```
+-----------------------------------------+

| Prot |             Name                 |

+-----------------------------------------+
```

O CMD_Lookup

It is the locator lookup message between B-NRS servers.

```
+-------------------------------------------------+
| Prot |  Name  |  Client IP  | Up/Down | Depth |
+-------------------------------------------------+
```

It keeps the IP address of the client who creates the locator lookup message so the locator information could be delivered directly to the client once it is found. Up denotes that the lookup message is to parent server and Down is to child servers. We increase the Depth by 1 whenever the message is forwarded to child. We keep the depth information because of the false positive of BF.

O CMD_Lookup NACK

When BF check fails, it is sent to parent server.

```
+-------------------------------------------------+
| Prot |  Name  |  Client IP  | Up/Down | Depth |
+-------------------------------------------------+
```

7. Security Considerations

False positive error is one of the well-known drawbacks of bloom filter and there is no way to get rid of it. Thus, it can be an attack point. For example, if an attacker puts wrong information into bloom filters of B-NRS in order to increase the false positive error rate resulting in getting traffics to go far away and consuming resource, then the performance degradation may occur until the B-NRS is refreshed. Once B-NRS is rebuilt, there will be only probabilistic false positive error rate not the deterministic one.

8. IANA Considerations

TBD

9. References

9.1. Normative References

9.2. Informative References

[ICNRG charter] http://irtf.org/icnrg

[ICN Challenges] D.Kutscher, S. Eum, K. Pentikousis, I. Psaras, D. Corujo, D. Saucez, T. Schmidt, and M. Waehlisch, "ICN Research Challenges ", draft-kutscher-icnrg-challenges-02, February 2014.

[ICN-IoT] Y. Zhang, D. Raychadhuri, L. Grieco, E. Baccelli, J. Burke, R. Ravindran, and G. Wang, "ICN based Architecture for IoT –Requirements and challenges", draft-zhang-iot-icn-challenges-01, December 2014.

[Ghodsi] 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.

[ITU] 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.

[Hanka] O. Hanka, C. Spleiss, G. Kunzmann, and J. Eberspacher, “A novel DHTbased network architecture for the next generation internet,” Eighth International Conference on Networks, Cancun, Mexico, March 2009.

[Luo] 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.

[Ahlgren] B. Ahlgren, J. Arkko, L. Eggert, and J. Rajahalme, “A node identity internetworking architecture,” in INFOCOM 2006. 25th IEEE International Conference on Computer Communications Proceedings. Washington, DC, USA: IEEE Computer Society, April 2006, pp. 1–6.

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

[Fab1999] Faber, T., Touch, J. and W. Yue, "The TIME-WAIT state in TCP and Its Effect on Busy Servers", Proc. Infocom 1999 pp. 1573-1583.

[DMap] T. Vu, A. Baid, Y. Zhang, T. Nguyeny, J. Fukuyamaz, R. Martin, and D. Raychaudhuri, “DMap: A Shared Hosting Scheme for Dynamic Identifier to Locator Mappings in the Global Internet,” Proceedings of the IEEE International Conference on Distributed Computing Systems, pp. 698-707, 2012.

[MDHT] M. D’Ambrosio, C. Dannewitz, H. Karl, and V. Vercellone, “MDHT: A Hierarchical Name Resolution Service for Information-centric Networks,” ICN’11, August 19, 2011, Toronto, Ontario, Canada.

A.1. Authors' Addresses

Jungha Hong
ETRI
218 Gajeong-ro, Yuseong-gu, Daejeon, Korea

Email: jhong@etri.re.kr

Woojik Chun
Hankuk University of Foreign Strudies
81, Oedae-ro, Mohyeon-myeon, Cheoin-gu, Yongin-si, Gyeonggi-do, Korea

Email: woojikchun@gmail.com

Heeyoung Jung
ETRI
218 Gajeong-ro, Yuseong-gu, Daejeon, Korea

Email: hyjung@etri.re.kr