A Feasibility Evaluation on Name-based Routing

Haesung Hwang[1], Shingo Ata[2], and Masayuki Murata[1]

[1] Graduate School of Information Science and Technology, Osaka University
1–5 Yamadaoka, Suita, Osaka 565–0871, Japan
{h-hwang, murata}@ist.osaka-u.ac.jp
[2] Graduate School of Engineering, Osaka City University
3–3–138 Sugimoto, Sumiyoshi-ku, Osaka 558–8585, Japan
ata@info.eng.osaka-cu.ac.jp

Abstract. The IPv4 addressing scheme has been the standard for Internet communication since it was established in the 1960s. However, the enormous increase in Internet traffic usage has been leading in the past to issues such as increased complexity of routing protocols, explosion in routing table entries, provider-dependent addressing, and security problem, demonstrating the need for a redesign in advanced router technologies. The past proposals have limitations when it comes to establishing the foundations of future-generation networks, which require more sophisticated routing protocols, like content-based routing. Furthermore, those previous approaches were not conceived to fully utilize the advantages of TCAM, which is a type of memory capable of performing high-speed lookups that is already implemented in high-end routers. In this paper, we show that routing based on domain names is already a feasible technology on the Network Layer and we evaluate the necessary network and hardware resources needed to implement name-based routing strategies. We present a routing scheme and propose three methods for equally balancing the routing information in the TCAM of multiple routers. The results show that this routing scheme is scalable and that the required number of routers is two orders of magnitude smaller than the number of currently existing routers.

1 Introduction

The Internet Protocol (IP) has been predominantly used for communication among heterogeneous devices in the Internet. Despite its popular usage today, problems arise because of significant differences in the traffic carried at the time it was developed and current trends toward high-volume multimedia communication. In the following, some of the major issues are listed: (i) Routing protocols are increasing in complexity and the routing table size is exploding (BGP Reports, `http://bgp.potaroo.net`), (ii) The IP address acts simultaneously as an 'ID' that distinguishes different nodes and a 'locator' that specifies the location of a node in the Internet, which makes the end node addresses depend on the ISP (Internet Service Provider), (iii) sophisticated routing schemes such as those based on names or content cannot be used. In this case, it is necessary to use

an overlay or the Domain Name System (DNS), which results in extra resource consumption and propagation delay, and (iv) there are security problems such as Distributed Denial-of-Service (DDoS) attacks.

Semantic Routing [1], Content Addressable Network (CAN) [2], and Locator/ID Separation Protocol (LISP) [3] partially resolve the problems mentioned above, but most of those studies use overlay networks or routing based on agent nodes, in both cases using IP addresses as before. On the other hand, next-generation network projects, such as FIND, FP7, and AKARI aim at designing a post-IP network architecture with one of the main trends being the effort to design a network that can perform routing using names and semantic information instead of simply numbered addresses. One of the motivations for name-based routing is to eliminate the need for DNS servers. Resolving Fully Qualified Domain Names (FQDNs) to IP addresses may require queries to multiple servers which depends on the cache updating policy. Locating a specific host under frequent changes of an IP address is difficult since DNS does not premise dynamic updates. In addition, name-based routing can separate ID and locator without using an IP address and provide an unlimited address space.

The final goal of our research is to propose a network architecture that uses existing resources such as port numbers, applications, and types of information. The mechanism for finding, querying, and fetching the desired information can be performed within the routers themselves, which eliminates the process of forwarding packets to specific data servers or domain name servers. As a first step, we propose in this paper a routing scheme that achieves routing by domain names utilizing the advantage of the router's Ternary Content Addressable Memory (TCAM) [4]. Achieving name-based routing with TCAM has been considered unrealistic in the past because it was presumed that a lot of hardware resources are required. This paper is the first work to our knowledge that suggests the feasibility of name-based routing by evaluating and estimating the required TCAM resources.

The rest of this paper is organized as follows. Section 2 surveys existing studies that propose routing with names instead of the conventional IP addresses. Section 3 presents the proposed system structure and routing scheme. Section 4 presents three methods to store the domain names in the TCAM of routers. Section 5 evaluates the necessary network and hardware resources for the methods presented in Sect. 3 and Sect. 4. Section 6 concludes this paper by briefly summarizing the main points and mentioning future work.

2 Related Work

In the Name-Based Routing Protocol (NBRP) [5], the IP address acts only as a temporary routing tag and the Uniform Resource Locator (URL) is used for identifying end nodes. When a user requests a certain content, a content router (CR), which simultaneously acts as a conventional IP router as well as a name server, returns the address of the server that has the content. However, although that proposal can eliminate the multiple levels of redirection that is inherent to

DNS, the necessary memory size of the CR is evaluated by DRAM, which is not optimal for high-speed packet forwarding because it can only perform exact binary matches. Partial matches of the prefix should be able to correspond to the network level of the packets in order to forward the packets rapidly to the next hop. Furthermore, names are used only while the connection is being established and not for routing the actual data packets. In [6], another name-based routing scheme is proposed. They compare the performance of their proposal with IP in terms of the creation, search, and update time of routing tables, as well as the required memory. The biggest problem with the proposal mentioned in that paper is the storage requirement, which is improved through caching and aggregating domain names. However, the work is still in its initial phase and the paper does not fully explain whether this is a feasible technology in terms of hardware resources. Moreover, the algorithm is evaluated using only a single router and does not state how the overall system should be designed nor do the authors elaborate on details of the routing method.

In this paper, we focus on evaluating the general feasibility of routing using domain names instead of IP addresses. This evaluation is done not only for the network resources, but also for the hardware resources using TCAM.

3 Routing by Domain Names

This section describes the overall structure of the proposed system and the routing scheme with a purpose of listing the preliminaries and requirements for the design of a new routing scheme.

3.1 Hierarchical Architecture

Our proposed system has a hierarchical structure in order that local information is routed in a closed network instead of being transmitted over a widespread area. In addition, it is possible to take advantage of one specific characteristic of domain names: they are already separated into different levels by dots. There are some advantages to having many levels in a hierarchical structure: each level consist of only few nodes, making the routing table of each node small. However, there are also some disadvantages such as overhead for traversing multiple levels and nodes in end-to-end communication. On the other hand, if there is only a single level, routing information of a lot of nodes have to be handled by each node, which makes the routing table of each node very large. In research on hierarchical routing for P2P applications [7, 8], two-level structures are commonly considered. In addition, as the number of hierarchy levels increases the reduction in routing table size is most effective when the structure has two or three levels [9]. Considering the maintenance cost, it is inefficient to have more than three levels. Therefore, we chose the maximum number of levels of the hierarchy in the system to be three. Since the number of FQDNs in different Top Level Domains (TLDs) is not equal, not all TLDs have three underlying levels of routers. If the

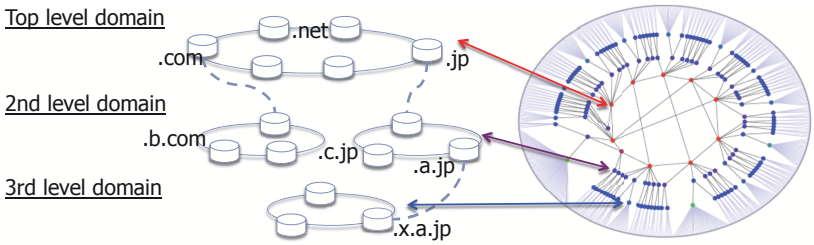


Fig. 1. Hierarchical to real topology

router in the highest level can handle all of the routing information within this TLD, then a single level is sufficient for that particular TLD.

The network topology inspired by the Abilene Network (`http://internet2.edu/network/`) and the hierarchical topology used in this paper are shown in Fig. 1. The Abilene Network is known to have similar characteristics to the real Internet topology [10]. It groups routers into three levels from the center outwards to the edge: backbone, local gateway, and edge routers. The architecture in this paper also groups routers into three levels: routers managing TLDs, routers managing 2nd-level domain names, and routers managing 3rd-level domain names. We map this hierarchical topology to the Abilene-inspired topology correspondingly. Since there are fewer than 300 kinds of TLDs, it is possible to statically configure which router manages a TLD. For local gateway and edge routers, we use dynamic configuration to write the routing information, which is explained in detail in Sect. 4.

3.2 Name Registration

Domain name routing differs from IP routing in terms of the registration because of the ID/locator separation. As IP serves as both an ID and locator, it simplifies the aggregation of the downstream traffic because IP assignment is the same in the hierarchical and in the Internet topology. However, domain names are not always location dependent which makes a separate registration process necessary. A node that intends to register its domain name injects a 'registration message' into the network and this message will eventually reach a router. The router that returns a 'destination unreachable' message because it does not have the path information is the place where the new domain name will be registered.

Each router has several TCAMs of predetermined size. In order to ensure that there is always enough room for routing table updates, a threshold value is used. After a certain threshold value is reached, the routing information must be written in a new router. A router's threshold of 0.75 means that the initial storage available for existing domain names is 75% of the router's memory capacity. The remaining 25% is reserved for routing table updates. When a new FQDN is registered, the utilization ratio of the router is checked. If it is below the

threshold, the domain name is registered in that router; otherwise, there are two possibilities: split the original router's routing table in half and divide it between two routers or leave the original router in its current status and start storing new domain names in a new router. These 'new routers' are designated as candidates from the beginning and they are only used when a split table needs to be stored.

3.3 Routing Table Exchange

Within an Autonomous System (AS) [11], one or more interior gateway protocols are used. An exterior gateway protocol such as BGP-4 [12] is used to route packets between ASs. In domain name routing, an AS can be regarded as a set of routers that manage the routing information of one or more TLDs. For example, a set of routers that reside in Japan and mainly have `.jp` as the TLD can be regarded as an AS. In order for the routers to have the network reachability information, it is necessary for the routers to exchange the initial routing table that was created at the time of the domain name registration described in Sect. 3.2. In the case of IP, routers exchange information about which subnets the network can reach, such as 192.168.0.0/16, which means hosts from 192.168.0.0 to 192.168.255.255 can be reached. Similar behavior can be achieved in domain name routing by exchanging information such as `*.osaka-u.ac.jp`, which means that all domains having `osaka-u.ac.jp` as their suffix can be reached. The exchange of routing tables is achieved by extending BGP-4.

3.4 Packet Forwarding

Routers forward packets on the basis of the information contained in the routing table stored in its TCAM. An example of packet forwarding is shown in Fig. 2. When the packets are forwarded from `computer.ist.osaka-u.ac.jp` to `biz.yahoo.com`, the former sends packets to the router R1. This router's routing table consists of three parts for domain names in the upper level, the same level, and the lower level. Since the packets are destined for a higher level than router R1, they are forwarded to port P. Router R2 goes through a similar process and sends the packets to port Q. Since router R3 is in the highest level of the hierarchy, the packets are sent to port X to reach the router that has the more specific address of the `.com` suffix. Router R4 refers to its routing table to send the packets to port A and the packets finally reach R5, which has the direct routing information for the destined domain name. The process is also done by extending BGP-4, using character information instead of the IP address.

4 Balanced Distribution of Domain Names in Routers

Compared with the IP routing tables, domain name routing tables are larger because of the variable length of domain names. Therefore, it is important to balance the routing information among multiple routers. In this section, we propose three methods for equally distributing the routing information.

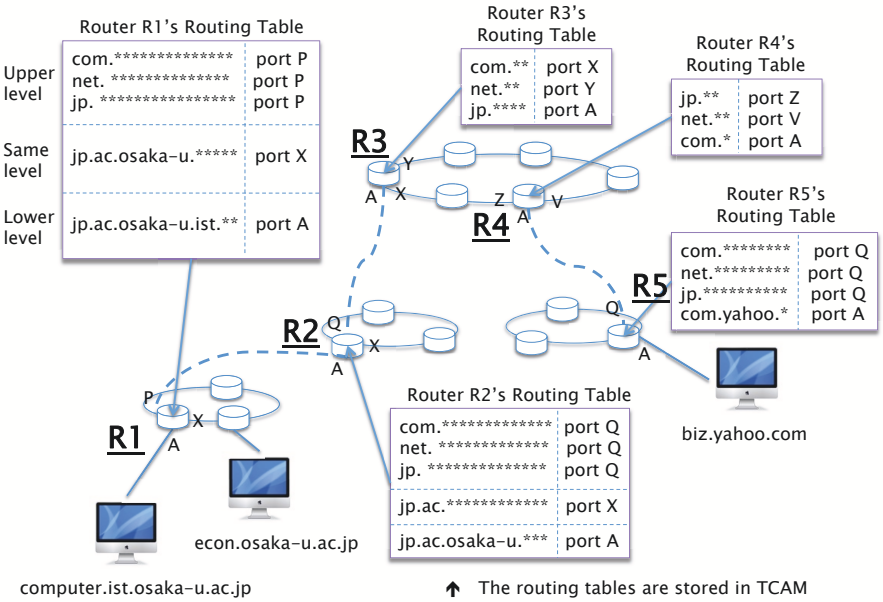


Fig. 2. Packet forwarding

4.1 Hash-based Distribution

The simplest of the three proposed distribution methods is hash-based distribution. This has the best performance in terms of balanced distribution of the routing table. An FQDN is made 'flat' by hashing it with SHA-1 where 'flat' means that it is no longer hierarchical. Since every FQDN is considered to have equal status, we can obtain the total number of routers required for domain name routing by multiplying the number of domain names by the number of bits needed per entry and then dividing the result by the TCAM size in a router. The advantage of this method is that the characteristics of SHA-1 allow a large database to be equally distributed into a given number of groups. However, domain names that end with ccTLDs (country code TLDs) lose locality information after hashing. In addition, when the nodes' physical locations have no relation to their IDs, it may lead to the problem of *long stretch* which is defined in [13] as the ratio of the number of physical hops taken by the protocol to reach the destination node from the source node and their shortest distance in terms of physical hops. Therefore, although it is considered ideal in terms of achieving a nearly equal distribution, but it is not realistic for implementation.

4.2 Hierarchical Longest Alphabet Matching

In order to store domain names in memory, hierarchical longest alphabet matching represents domain names in ASCII code. The characters used in domain

names are 26 case insensitive letters of the English alphabet and the 10 numbers from 0 to 9 plus hyphens and dots [14]. Inspired by the longest prefix matching scheme, longest alphabet matching is applicable when the domain names are represented in binary format. In addition to the good searching speed, aggregation of multiple domain names with the same suffix is possible, which reduces the occupied memory space. Hierarchical longest alphabet matching takes full advantage of TCAM, which can represent a don't care value '`*`' in each cell in addition to `0` or `1`. To calculate the total number of routers required, we first compute the number of routers required in each TLD. The pseudocode for this calculation is shown in Alg. 1.

Algorithm 1 Numbers of routers for Hierarchical longest alphabet matching

$n_{L2} \leftarrow$ number of routers in 2nd level
$n_{L3} \leftarrow$ number of routers in 3rd level
$t \leftarrow$ threshold of TCAM utilization
$entry_l \leftarrow 180$ (bits per one entry)
$router_c \leftarrow 18 \times 10^6 \times 10 \times t$
$u \leftarrow$ entries with unique 2nd-level domain names
$e \leftarrow$ entries in the TLD
if e $\times$ $entry_l \leq router_c$ **then**
  RETURN $n_{L2} \leftarrow 1$, $n_{L3} \leftarrow 0$
else if u $\times$ $entry_l > router_c$ **then**
  divide u into groups (dynamic configuration using 2nd-level domain name)
  RETURN $n_{L2} \leftarrow$ number of groups
  for each group **Func_Calculate** n_{L3}
else
  $n_{L2} \leftarrow 1$, **Func_Calculate** n_{L3}
end if
Func_Calculate n_{L3}
divide e into groups (dynamic configuration using 3rd-level domain name)
RETURN $n_{L3} \leftarrow$ number of groups
END

For each TLD, the first step is to determine whether all e domain name entries would fit into a router. Each entry is multiplied by 180 bits ($entry_l$) and divided by the router's available TCAM size ($router_c$). We assume that one entry consumes 180 bits in order to make comparisons with the result in Sect. 4.1, where 160 bits were the hashed value plus 20 bits of output port plus 4 parity bits. Again we define a router as having 10 TCAMs, each TCAM having a capacity of 18 Mbits. Threshold t is explained in Sect. 3.2.

If e multiplied by $entry_l$ is less than $router_c$, then the total number of routers required in this TLD is set to 1, and the process ends. Otherwise, u unique 2nd-level domain names are written in the 2nd level of the hierarchy in the form of `TLD.uniq.*`. The routers in the highest level are written with TLD information,

which we ignore for the moment. Here, u domain names are written dynamically by referring to ASCII table. The basic idea is that names are grouped in alphabetical order, and when a router overflows, it starts storing the domain names in a new router. An additional idea is to take advantage of the prefix values. Names are first divided into two groups: digits and letters. If the size is still too large, the letters are divided again into smaller groups, a group of `110****` and a group of `111****`. This process is repeated until every router is well within the $router_c$. One example combination of the grouping is, hyphen and digits (`01*****`), a to c (`11000**`), d to g (`11001**`), h to o (`1101***`), p to q (`111000*`), r to s (`111001*`), t to w (`11101**`), and x to z (`1111***`).

When the partitioning of the 2nd-level domain names is over, the grouping is done with 3rd-level domain names. However, since the maximum number of levels in the hierarchy considered in this work is three, routers are written with FQDNs instead of with the unique 3rd-level domain names. The number of groups in each level corresponds to the number of routers. Therefore, the total number of routers required is $n_{L2} + n_{L3}$.

4.3 Hybrid Distribution

In this section, TLDs are distinguished by 9 bits because there are fewer than 300 TLDs. The hash function is applied to 2nd and 3rd-level domain names separately. To calculate the total number of required routers, we first compute the number of routers required in each TLD. The pseudocode is shown in Alg. 2.

Algorithm 2 Number of routers for Hybrid distribution

(Refer to Alg. 1 for the variables)
if e $\times$ $entry_l \leq router_c$ **then**
  RETURN $n_{L2} \leftarrow 1$, $n_{L3} \leftarrow 0$
else if u $\times$ $entry_l \leq router_c$ **then**
  RETURN $n_{L2} \leftarrow 1$
  RETURN $n_{L3} \leftarrow (e \times entry_l)$ / $router_c$
else
  $n_{L2} \leftarrow (u \times entry_l)$ / $router_c$
  hash(2nd level domain name) % n_{L2} (divide routers in 2nd-level into groups)
  for each group **calculate** n_{L3}
  RETURN $n_{L3} \leftarrow (e_{group} \times entry_l)$ / $router_c$
end if
END

The process is the same as Alg. 1 until the list of u does not fit into a single router. We obtain n_{L2} routers required in the 2nd level in the hierarchy by multiplying u by $entry_l$ and dividing it into $router_c$. In other words, the number of groups based on the 2nd-level domain names is equal to hashing 2nd-level domain names modulo n_{L2} to the most significant character. Then we obtain

groups that have nearly equal entries independent of the alphabets. The next step is to calculate n_{L3} routers required in the 3rd-level; these routers are located under the corresponding routers in the 2nd level of the hierarchy. The router in which each FQDN is stored depends on the hash value of the FQDN's 2nd-level domain names. To simplify the calculation of n_{L3}, each entry is multiplied by $entry_l$ and divided by $router_c$. This was done assuming that hashing has the characteristic of equally distributing a large amount of data among groups.

5 Evaluation and Discussion

We evaluate the network and hardware resources required for domain name routing with entries obtained from the ISC database in July 2008 [15].

5.1 Network Resources

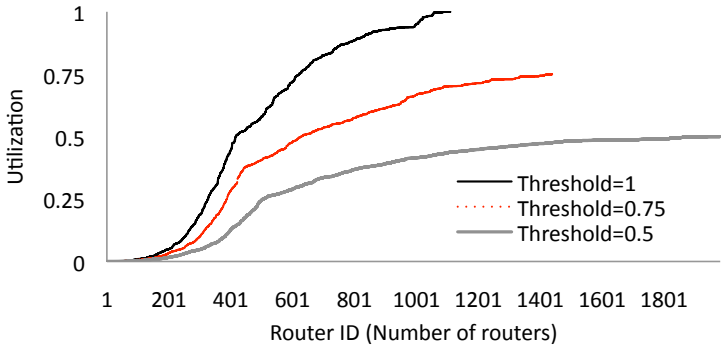
Figure 3 shows the distribution of domain name entries in routers, arranged in ascending order of the routers' utilization ratio. The utilization is defined as the number of entries in a router multiplied by 180 bits divided by the router's memory size. The graph shows the results for hierarchical longest alphabet matching and hybrid distribution for threshold values of 1, 0.75, and 0.5. The total number of required routers is smallest when the threshold is 1, letting the expected number of routing hops to be minimum. However, since there is no space left for updating routing tables, setting the threshold to 1 is not a realistic choice.

For each threshold, the required number of routers is 29.1% smaller on average for hybrid distribution than for hierarchical longest alphabet matching. In addition, hybrid distribution is likely to make a better use of the memory space because there are more routers with a utilization ratio close to the threshold.

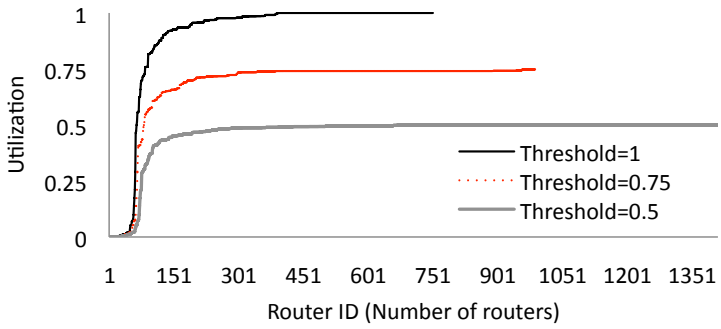
The performance of both methods can be improved by aggregating those entries in the routers that have a low utilization ratio. Since both algorithms assign at least one router for a single TLD, approximately 150 TLDs out of 270 TLDs have a utilization ratio of less than 0.1%. Those entries are collected and stored in only a small number of routers. That reduces the total number of routers required to an average of 14.2% and 16.5% for hierarchical longest alphabet matching and hybrid distribution, respectively. Therefore, statically assigning routers in backbone routers, as proposed in Sect. 3.1, needs careful consideration of the growth rate of domain names in each country.

Comparing these two algorithms, we can see that hybrid distribution needs less routers, but hierarchical longest alphabet matching utilizes the advantage of TCAM better. If TCAMs can handle DHT (Distributed Hash Table) data, then hybrid distribution can also be considered to be a feasible method.

The number of routers required for different thresholds in routers are shown in Fig. 4. Depending on the update rate of the routing table, it is possible to estimate how the required number of routers will increase. The result shows that the initial storage of existing domain names with a threshold value of 0.2 would require approximately 4,000 routers for both hierarchical longest alphabet matching and hybrid distribution.



(a) Hierarchical longest alphabet matching



(b) Hybrid distribution

Fig. 3. Distribution of domain name entries in routers for different thresholds

5.2 Hardware Resources

When hash-based distribution is used, the average number of entries among 4,096 routers is 145,976. We obtain the memory size of 26 Mbits by 180 bits (SHA-1 output (160 bits) + output port (16 bits) + parity bits (4 bits)) × 145,976. Thus, having two 18-Mbit TCAMs is sufficient when there are 4,096 routers. In other words, if there are ten 18-Mbit TCAMs in a router, a total of 597 routers ((145,796 entries × 180 bits × 4,096 routers) / (18×10^6 bits × 10 TCAMs)) are required to store the existing domain names.

The number of routers required in hierarchical longest alphabet matching is 1,396, assuming that one entry consumes 180 bits. 7-bit ASCII code is sufficient to distinguish the characters used in domain names. Furthermore, since there are fewer than 300 TLDs, they can be differentiated using 9 bits. Whereas it is easy to adjust the bit length in each entry in hash-based distribution because each entry is hashed, this is not the case in hierarchical longest alphabet matching. To satisfy the static length (180 bits) that we used to evaluate the required number

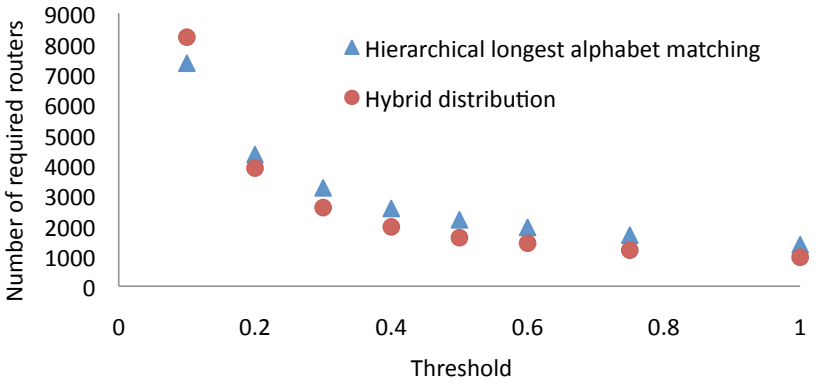


Fig. 4. Number of necessary routers required for different thresholds

of routers throughout this paper, 171 bits must be free for use, excluding the 9 bits reserved for the TLD. When one character uses 7 bits, approximately 25 characters can be written which only consists of 30% of the FQDNs [15]. To store 99% of the FQDNs, the TCAM must be able to store entries having up to 50 characters. Although the search speed decreases when the lookup size increases [4], designing the TCAM to suit longer bit lengths does not impose much of a technological difficulty.

The number of routers required for hybrid distribution is 952. Hierarchical longest alphabet matching and hybrid distribution require 2.4 and 1.7 times as many routers as hash-based distribution, respectively. However, they both use hierarchical structures, which have the advantage of restricting the local information to be routed in a closed network instead of causing it to be forwarded over a widespread area. Although hierarchical longest alphabet matching cannot achieve a nearly equal distribution of domain names among the routers, because it does not use a hashing function, it should reduce the burden of applying hashing to every entry.

In [16, 17], the authors use the work from [18] and estimate that the number of routers deployed world-wide is approximately 228,260. Therefore, the router numbers required for the methods proposed in this paper are realistic values that indicate that a name-based routing architecture is feasible.

6 Conclusion and Future Work

In this paper, we used the domain name in routing as a substitute for the conventional IP address. Through statistical evaluations of currently existing domain names we showed that the proposed method is feasible in the Network Layer by estimating the required network and hardware resources. Future work involves evaluating the routing table updates by observing the evolution of the registered

domain name entries over time. This includes estimating the rate of increase of entries as well as that of the number of newly added names. In addition, the effect of eliminating the current domain name servers should be investigated.

Acknowledgement This research was supported by National Institute of Information and Communications Technology (NICT) of Japan.

References

1. Inoue, K., Akashi, D., Koibuchi, M., Kawashima, H., Nishi, H.: Semantic Router using Data Stream to Enrich Services. In: Proc. of CFI. (Jun 2008) 20–23
2. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Schenker, S.: A Scalable Content-Addressable Network. In: Proc. of SIGCOMM. (Aug 2001) 161–172
3. Farinacci, D., Fuller, V., Oran, D.: Locator/ID Separation Protocol (LISP). "draft-farinacci-lisp-10.txt", IETF Network Working Group, work in progress (Nov 2008)
4. Renesas Technology Corporation: Network Address Search Engine (9 M/18 M-bit Full Ternary CAM) available at `http://documentation.renesas.com/eng/products/others/rej03h0001_r8a20211bg.pdf`.
5. Gritter, M., Cheriton, D.R.: An Architecture for Content Routing Support in the Internet. In: Proc. of USITS. (Mar 2001) 37–48
6. Shue, C.A., Gupta, M.: Packet Forwarding: Name-based Vs. Prefix-based. In: Proc. of GI. (May 2007) 73–78
7. Garcés-Erice, L., Biersack, E.W., Felber, P.A., Ross, K.W., Urvoy-keller, G.: Hierarchical Peer-to-Peer Systems. In: Proc. of Euro-Par. (Aug 2003) 643–657
8. Kersch, P., Szabo, R., Kis, Z.L., Erdei, M., Kovács, B.: Self Organizing Ambient Control Space: An Ambient Network Architecture for Dynamic Network Interconnection. In: Proc. of DIN. (Sep 2005) 17–21
9. Lian, J., Naik, S., Agnew, G.B.: Optimal Solution of Total Routing Table Size for Hierarchical Networks. In: Proc. of ISCC. (Jun 2004) 834–839
10. Li, L., Alderson, D., Willinger, W., Doyle, J.: A First-Principles Approach to Understanding the Internet's Router-level Topology. In: Proc. of SIGCOMM. (Aug 2004) 3–14
11. Hawkinson, J., Bates, T.: RFC 1930: Guidelines for Creation, Selection, and Registration of an Autonomous System (AS) (Mar 1996)
12. Rekhter, Y., Li, T.: RFC 1771: A Border Gateway Protocol 4 (BGP-4) (Mar 1995)
13. Lu, G.H., Jain, S., Chen, S., Zhang, Z.L.: Virtual Id Routing: A Scalable Routing Framework with Support for Mobility and Routing Efficiency. In: Proc. of ACM MobiArch. (2008) 79–84
14. Mockapetris, P.: RFC 1035: Domain names - Implementation and Specification (Nov 1987)
15. Internet Systems Consortium: available at `https://www.isc.org`
16. Yook, S.H., Jeong, H., Barabási, A.L.: Modeling the Internet's Large-scale Topology. In: PNAS. (Oct 2002) 13382–13386
17. Lakhina, A., Byers, J.W., Crovella, M., Matta, I.: On the Geographic Location of Internet Resources. IEEE JSAC **21**(6) (Aug 2003) 934–948
18. Govindan, R., Tangmunarunkit, H.: Heuristics for Internet Map Discovery. In: Proc. of INFOCOM. (Mar 2000)