Master's Thesis

Title

Name-based Routing for Future Networks

Supervisor

Professor Masayuki Murata

Author

Haesung Hwang

February 16th, 2009

Department of Information Networking

Graduate School of Information Science and Technology

Osaka University

Master's Thesis

Name-based Routing for Future Networks

Haesung Hwang

Abstract

The IPv4 (Internet Protocol version 4) addressing scheme has been the standard for Internet communication since it was established in the 1960s. However, the advances in router technology and the enormous increase in Internet traffic usage mean that it can no longer cope with issues such as the increased complexity of routing protocols, explosion in routing table entries, provider-dependent addressing, and security problems. Although some proposals for partially solving these problems have been presented, they 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 Ternary Content Addressable Memory (TCAM), which is a type of memory capable of performing high-speed lookups that is already implemented in high-end routers.

In this thesis, we show that routing based on domain names, which has been proposed as a substitute for IP routing in the future, is already a feasible technology in 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. Simulation results show that this routing scheme is scalable with respect to domain name registration and that the required number of routers is two orders of magnitude smaller than the number of currently existing routers.

Keywords

Domain name routing

Future-generation network

Network/hardware resources

Routing protocol

Ternary Content Addressable Memory (TCAM)

Contents

1 Introduction **6**

2 Background Work **8**

2.1 Name-based Packet Forwarding 8

2.2 Ternary Content Addressable Memory (TCAM) 9

3 Routing by Domain Names **12**

3.1 Hierarchical Architecture 12

3.2 Name Registration 15

3.3 Routing Table Exchange 17

3.4 Packet Forwarding 18

3.5 Mirroring and Failure Handling 21

4 Balanced Distribution of Domain Names in Routers **23**

4.1 Hash-based Distribution 23

4.2 Hierarchical Longest Alphabet Matching 24

4.3 Hybrid Distribution 27

5 Evaluation and Discussion **30**

5.1 Network Resources 30

5.2 Hardware Resources 34

6 Conclusion and Future Work **38**

Acknowledgments **39**

References **42**

Appendix **43**

List of Figures

1 Difference between RAM and TCAM 9
2 TCAM circuit 10
3 Hierarchical structure with maximum of three levels 12
4 Distribution of number of domain names sorted into corresponding domain levels 13
5 Mapping of Abilene-inspired topology and hierarchical structure 14
6 Method of writing ccTLD and gTLD in appropriate places 15
7 Name registration 17
8 Exchange of routing tables 18
9 Packet forwarding 19
10 Search key and hierarchical structure 20
11 Packet header of BGP update message 20
12 Mirroring in domain name routing 21
13 Hash-based distribution 23
14 Example of distributing domain names to routers in hash-based distribution . . . 23
15 Example of longest alphabet matching 24
16 Hierarchical longest alphabet matching 26
17 Structure of search key 26
18 Hybrid distribution 29
19 Distribution of domain name entries in routers for various different thresholds . . 31
20 Distribution of domain name entries in routers for various different thresholds (improved) 32
21 Approximation of number of necessary routers required for various different thresholds 33
22 Percentage of duplication in each TLD compared with com and net 34
23 Balanced distribution of entries among routers in hash-based distribution 35
24 Distribution of domain name length, excluding TLD length 36
25 Comparison of three storage methods 37

List of Tables

1 Dependence of TCAM content on bitline inputs 10
2 Example of group combination 27
3 ASCII table in binary and prefix groups 43

1 Introduction

The Internet Protocol (IP) was established in the 1960s and has been used for communication among heterogeneous devices in the Internet. Despite its popular usage today, problems arise because of differences in the traffic carried at the time it was developed and current trends toward high-volume multimedia communication. The following are some of the major issues.

- Routing protocols are increasing in complexity and the routing table size is exploding [1] as a result of the addressing scheme called Classless Inter-Domain Routing (CIDR).
- IP acts simultaneously as an identification (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). Therefore, the overhead for changing a node's address increases every time it moves.
- Sophisticated routing schemes such as those based on names or content cannot be used. It is necessary to use an overlay or the Domain Name System (DNS), which results in extra resource consumption and propagation delay.
- There are security problems such as Distributed Denial-of-Service (DDoS) attacks.

To solve these problems, a lot of research has been done around the world on designing a post-IP network architecture. The GENI (Global Environment for Network Innovations) [2] Initiative in the USA supports the research, design, and development of new networking and distributed systems capabilities. FIND (Future InterNet Design) [3] is a GENI project that focuses on devising a comprehensive network starting with a clean slate. In addition, the FP7 (Seventh Framework Programme) [4] in Europe established a unit called FET (Future and Emerging Technologies) that is investigating next-generation communications supporting information and communications technology such as SAC (Situated and Autonomic Communication). In Japan, the AKARI Project [5] is specifying the requirements of fundamental rules for designing new generation networks, the basic framework of the new architecture, and a testbed environment.

One of the trends for next-generation networks such as those mentioned above is efforts to design a network that can perform routing using names and semantic information instead of simply numbers. Related studies include Semantic Routing [6], CAN (Content Addressable Net-

work) [7], and LISP (Locator/ID Separation Protocol) [8]. However, most of those studies use overlay networks or routing based on agent nodes using IP addresses as before.

The final goal of our research is to propose a routing scheme that uses resources such as port numbers, applications, and types of information. The mechanism for finding information, making queries, and fetching the desired information can be performed within the routers, which eliminates the process of forwarding packets to specific data servers and domain name servers.

Therefore, as a first step, we propose in this thesis a network architecture that achieves routing by domain names utilizing the advantage of router's TCAM (Ternary Content Addressable Memory) [9]. Specifically, we suggest three methods of equally distributing routing tables composed of domain names among multiple routers. We estimated the necessary hardware and network resources through simulations and the results prove that domain name routing is a feasible technology.

The rest of this thesis is organized as follows. Section 2 surveys existing studies that proposed routing with names instead of the conventional IP addresses and also briefly describes the characteristics of TCAM. Section 3 presents the proposed system structure and routing scheme based on domain names, focusing on registering domain names, exchanging routing tables, forwarding packets, mirroring, and handling failures. Section 4 presents three methods using a hash function and longest alphabet matching to store the domain names in the TCAM of routers. Section 5 evaluates the necessary network and hardware resources for the methods presented in Sections 3 and 4. Section 6 concludes this thesis by briefly summarizing the main points and mentioning future work.

2 Background Work

This section introduces related work on non-IP address routing and also describes TCAM, which is advantageous in longest prefix matching.

2.1 Name-based Packet Forwarding

The recent usage trend of the Internet is closely tied to the emergence of the World Wide Web (WWW) and Peer-to-peer (P2P) applications. More than half of the Internet traffic is attributed to P2P applications [10], where the traffic consists of requests from users for some content and the returning reply containing either the content's location or the content itself. The limitations on performing sophisticated information searches of this kind using only the conventional IP addresses makes it necessary to devise new addressing schemes. Among the various studies that have been done, [11] and [12] focuses on the domain name routing.

NBRP (Name-Based Routing Protocol) [11] is an addressing scheme of the TRIAD [13] Project. The IP address acts only as a temporary routing tag and the URL (Uniform Resource Locator) is used to identify end nodes and perform end-to-end routing. When a user requests a content item specified by its URL, a contents router (CR) plays an important role. A CR acts as a conventional IP router as well as a name server, and it has the name to its next hop sets. When a user requests some content, the CR returns the address of the server that has it. However, although that proposal can eliminate the multiple levels of redirection that the DNS has, the memory size of the CR is evaluated by DRAM (Dynamic Random Access Memory), which is not an optimal memory environment for high-speed packet forwarding because it performs only exact binary matches with `0` and `1`. Partial matches with a prefix, which is explained in Section 2.2, should be able to match the network level of the packets in order to forward the packets rapidly to the next hop router. Furthermore, names are used only while the connection is being established and not for routing the actual data packets.

In [12], the authors propose name-based routing to solve some problems of IPv4 such as exhaustion of IP addresses, security problems, and provider-dependent nodes. They compare their proposal with IP in terms of performance such as the creation, searching, and update time of routing tables and the required memory. The biggest problem with the proposal mentioned in that paper is the storage requirements, which is improved through caching and aggregating domain

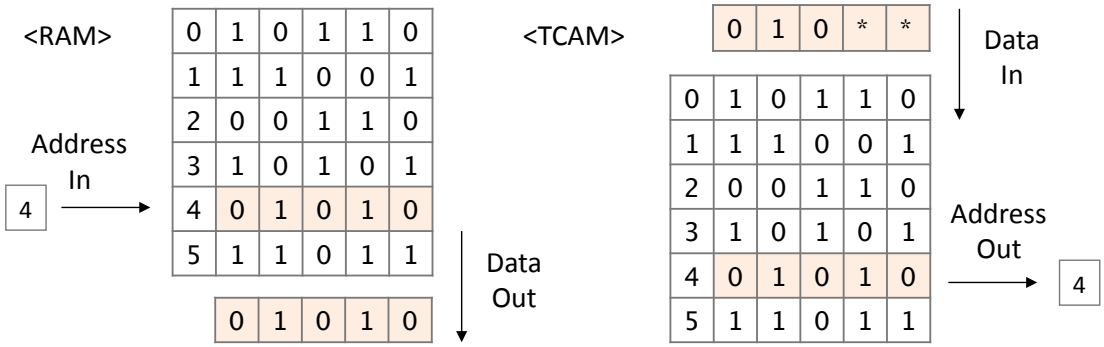


Figure 1: Difference between RAM and TCAM

names. Lookups and updates are first conducted on the cached trie where the popular domain names are stored and then proceed to the regular trie only if no match is found in the cached trie. This reduces the lookup time in large routing tables. In addition, groups of domains can be aggregated together into a single identifier using the fact that multiple domains are often co-located in the same network. This can reduce the number of entries in the core routing tables. 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 was evaluated using only a single router and the paper does not state how the overall system should be designed nor elaborate on the routing method.

In this thesis, we focus on evaluating the feasibility of the domain name routing. The evaluation is done not only for the network resources, but also for the hardware resources.

2.2 Ternary Content Addressable Memory (TCAM)

In this section, we describe on the structure of the TCAM [9] and clarify the reason why TCAM is used in this thesis.

The difference between RAM (Random Access Memory) and TCAM is shown in Figure 1. RAM is searched using a memory address as a search key, and the content of the memory at that address is returned as the result. On the other hand, TCAM is searched using the memory content, and the memory address where the supplied data was found is returned as the result. In the example shown in Figure 1, RAM takes "4" as the input and returns "`01010`" as the result, whereas TCAM takes "`010**`" as the input and returns "4" as the output. Unlike binary CAM,

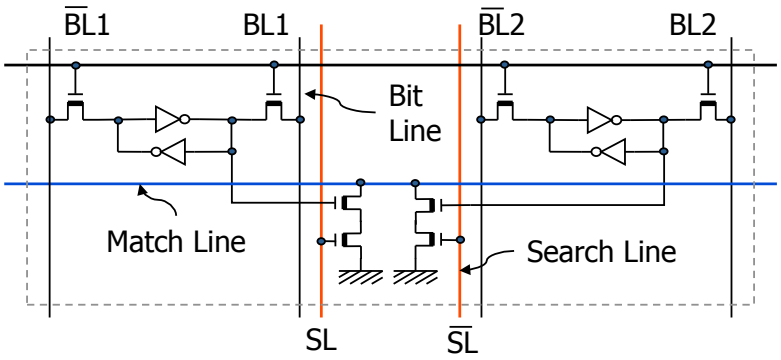


Figure 2: TCAM circuit

Table 1: Dependence of TCAM content on bitline inputs

TCAM's content	Bitline 1	Bitline 2
1	0	1
0	1	0
don't care ($\star$)	0	0
always miss	1	1

which can represent only `0` and `1`, TCAM can also represent a third value in each cell: an arbitrary *don't care* value, "$\star$". A circuit for TCAM is shown in Figure 2 and the way cell values are decided is indicated in Table 1. Bitline 1 (BL1) and bitline 2 (BL2) take either `0` or `1` as input where $\overline{BL1}$ is the inverse of BL1. The value of a particular TCAM cell depends on the values of BL1 and BL2. For example if BL1 is `0` and BL2 is `1`, the corresponding TCAM cell represents `1`.

Having $\star$ as a possible cell value is a big advantage of TCAM. When BL1 and BL2 both take `0` as their inputs, the cell represents a don't care bit. Schemes for storing IP addresses of routing tables in TCAM make use of this don't care bit to represent a network level address. For example, 192.168.128.0/24 can be stored in TCAM (the common notation "/" is used to describe the subnetwork addresses) as follows:

`11000000101010001000000********`

TCAM performs partial matching of the entries excluding the $\star$ part. This characteristic enables high-speed longest prefix matching. Therefore, the data format of the entries in this thesis take into consideration the don't care bit to fully utilize the advantages of TCAM.

During the course of this study on TCAM, we first investigated how Access Control Lists

(ACL) can be stored efficiently TCAM. The results were published in [14, 15] and the study on storing IP routing tables in TCAM was reported in [16].

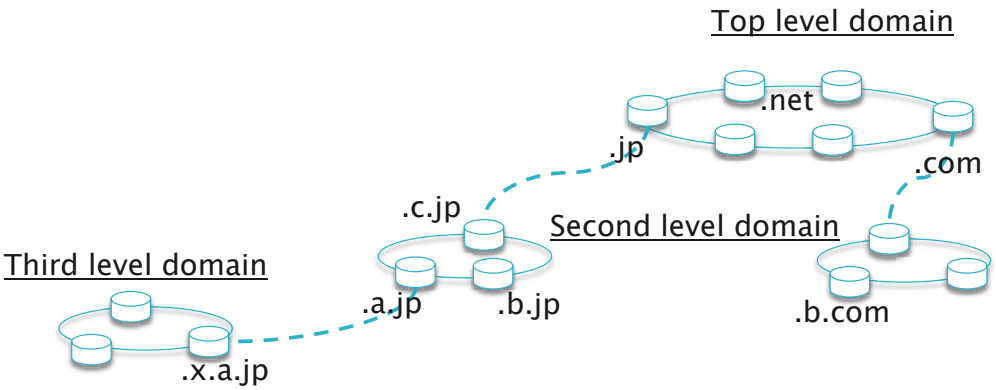


Figure 3: Hierarchical structure with maximum of three levels

3 Routing by Domain Names

This section describes the overall structure of the proposed system and explains the routing scheme using domain names. Its purpose is to list some of the preliminaries and requirements for the design of a new routing scheme.

3.1 Hierarchical Architecture

The proposed system has a hierarchical structure so that local information is routed in a closed network instead of being transmitted over a widespread area. For example, if one wants to search for a domain name that ends with `.jp` (Japan), it is unnecessary to send the query to routers in the USA. In addition, it is possible to take advantage of a characteristic of domain names: they are already divided into different levels by periods. Figure 3 shows a simplified hierarchical structure consisting of three layers: the top-level, second-level, and third-level domains. Routers in the highest level have entries with routing information about the TLD (Top Level Domain) such as `*.jp`, `*.com`, and `*.net`. Here, the "`*`" in domain names means 'anything', which should not be confused with the 'don't care' cell value of TCAM in Section 2.2. The routing information gets more specific in the lower levels, storing entries such as `*.a.jp`, `*.b.com`, and `*.c.net`. The routing between `*.a.jp`, `*.b.jp`, and `*.c.jp` can be performed entirely within domains of the same level, which reduces the number of hops that packets must take.

In a hierarchical structure, the maximum number of levels must be defined beforehand. Using

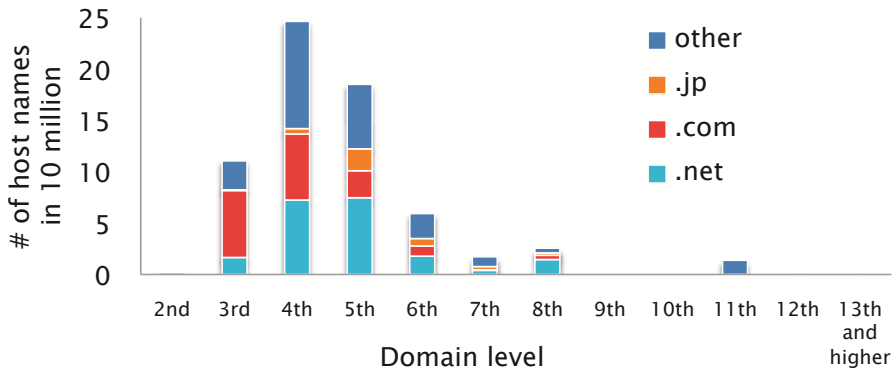


Figure 4: Distribution of number of domain names sorted into corresponding domain levels

the database of the ISC (Internet Systems Consortium) [17], 663,449,326 domain names were sorted according to the number of their corresponding levels. The results are shown in Figure 4. Each entry in the data is a unique IP address and its domain name, where duplicate domain names for an IP address may exist. On the x axis, "2nd" and "3rd" mean domain names such as `X.Y` and `X.Y.Z`, respectively. The figure also shows the share for the three top TLDs, `.net`, `.com`, and `.jp`, which account for 62% of the entire database.

If 2nd and 3rd-level domain names are stored in the same level of the hierarchy and those below 6th-level domain names are grouped into one hierarchy, the percentages for the hierarchy levels are: 17% for 2nd+3rd, 37% for 4th, 28% for 5th, and 18% for 6th and lower. If we considered the highest level where the routing information of TLDs is stored, then the total number of levels required would be five. There are some advantages to having many levels in a hierarchical structure: there are fewer nodes in each level and the routing table of each node may be small. However, there are also disadvantages such as a higher overhead for crossing multiple levels and nodes in end-to-end node communication. On the other hand, if there is only one level, a lot of nodes have to be handled, which makes the routing table of each node very large.

In research on hierarchical routing for P2P applications [18, 19], two-level structures are common: applications use the concept of peer and super-peer. In other words, having two logical layers that do intra-cluster and inter-cluster routing is popular. In addition, [20] shows that as the number of hierarchy levels increases the reduction in routing table size is most effective when the structure has two or three levels. Considering the cost of maintaining the clusters, it is inefficient to have more than three levels.

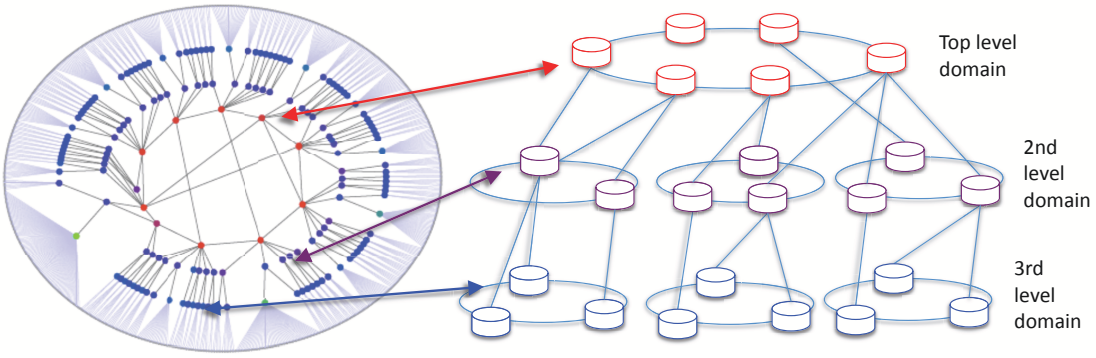


Figure 5: Mapping of Abilene-inspired topology and hierarchical structure

Therefore, we chose the maximum number of levels of hierarchy in the system to be three. Since the number of FQDN (Fully Qualified Domain Name) in different TLDs is not the same, not all TLDs have three levels of routers. If the router in the highest level can handle all of the routing information within a TLD, then a single level is sufficient for that particular TLD. On the other hand, TLDs with larger databases can have up to three levels of routers, which results in efficient routing not only when domain names are followed by a ccTLD (country-code TLD) or a gTLD (generic TLD) such as `google.com`, but also when a gTLD is followed by a ccTLD (e.g., `chinadaily.com.cn`).

The network topology inspired by the Abilene Network (currently also known as the Internet2 Network) [21, 22] and the hierarchical topology used in this thesis are shown in Figure 5. The Abilene Network is a high-performance backbone network created by the Internet2 community. It mainly connects universities, some corporations, and affiliated institutions. The Abilene Network is known as a Heuristically Optimal Topology (HOT), which is said to have similar characteristics to the real topology.

The Abilene-inspired topology groups routers into three levels from the center outwards to the edge: backbone routers, local gateway routers, and edge routers. The architecture in this thesis 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 and found that the TLD routing information is stored at the interconnected backbone routers at the center of the Abilene-inspired topology. Since there are fewer than 300 kinds of TLDs, it is possible to statically configure which router manages a TLD.

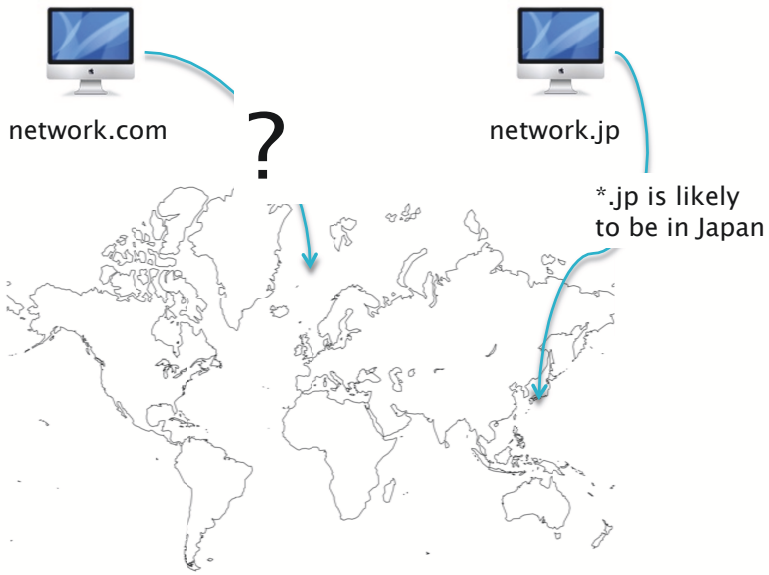


Figure 6: Method of writing ccTLD and gTLD in appropriate places

For local gateway and edge routers, we use dynamic configuration to write the routing information. The dynamic configuration is explained in detail in Section 4.

3.2 Name Registration

IANA (Internet Assigned Numbers Authority) groups TLDs as infrastructure TLD , ccTLD, sTLD (sponsored TLD), gTLD, and generic-restricted TLD. On the other hand, TLDs are commonly grouped into ccTLD and gTLD, which have different characteristics related to where domain names should be stored in the network. An example is shown in Figure 6. The routing information of ccTLDs like `.jp` is likely to be stored in routers residing in Japan, but where should the routing information of domain names that end with `.com` be stored? One method is to keep them in routers located in the USA which not only has abundant resources [23] but also high-capacity links. Although there may be some propagation delay, the RTT (Round-Trip delay Time) is low because most countries have their primary paths to the USA. However, domain names such as `someJapaneseCompany.com` end with a gTLD even though the traffic is mainly within Japan. Optimizing where to write domain names is outside the scope of this thesis since our purpose is to fully utilize the allowed router resources and determine the minimum required number of routers.

In IP routing, each node is assigned either a static IP address from the network administrator

or a dynamic IP address from the server running DHCP (Dynamic Host Configuration Protocol). It is not necessary to separately register the nodes' individual IP addresses with the gateway router in order to receive packets because both addresses belong to the same network as the gateway router.

In domain name routing, when a node's domain name has the same suffix (for domain name `computer.ist.osaka-u.ac.jp`, the suffix might be `ist.osaka-u.ac.jp`, `osaka-u.ac.jp`, `ac.jp`, or `jp`) as the gateway router, it is treated as a similar case to the above IP addressing scheme and does not require a separate registration process. However, the gateway router does not always have the same suffix as the domain name, especially if the node has migrated with a different TLD. In this case, it is necessary to register the node's domain name with the gateway router. An example is shown in Figure 7. When a node with domain name `computer.ist.osaka-u.ac.jp` is connected to a gateway router having the same suffix `osaka-u.ac.jp`, the router already knows the paths to the other nodes that have this suffix, so no registration is needed. On the other hand if the node migrates and when the new gateway router does not have any routing information for the node's suffix, the node must register its domain name.

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 downstream domain names to be aggregated because IP assignment has the same behavior in the hierarchical topology and in the Internet topology. 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 a place where the domain name will be registered.

Each router has several TCAMs for storing the routing table. The size of a TCAM is predetermined and after a certain threshold value is reached, the routing information must be written in a new router. The threshold value is set mainly to ensure that there is room for routing table updates. If a router's threshold is 0.75, it means that the initial storage available for existing domain names is up to 75% of the router's memory capacity. The remaining 25% is reserved for routing table updates. The number of routers required for various threshold values is evaluated in Section 5.

When a new FQDN is registered, the utilization ratio of the router is checked. If the utilization

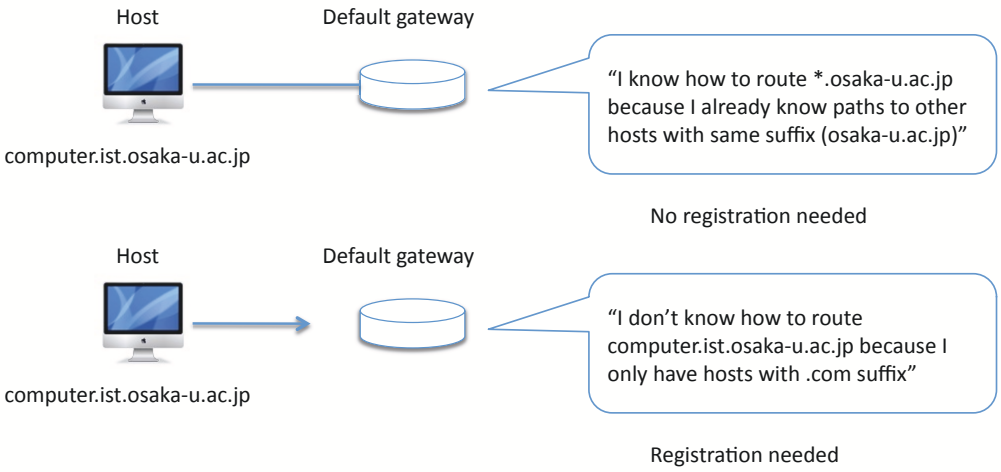


Figure 7: Name registration

ratio is below the threshold, the domain name is registered; otherwise, there are two choices: 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 ones that were designated as candidates from the beginning and they are used when a split table needs to be stored.

3.3 Routing Table Exchange

According to RFC 1930 [24], an Autonomous System (AS) is a connected group of one or more IP prefixes managed by one or more network operators that has a single and clearly defined routing policy. Within the AS, one or more interior gateway protocols are used. An exterior gateway protocol such as BGP-4 (Border Gateway Protocol) [25] is used to route packets between ASs.

In domain name routing, a single 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 to exchange network reachability information, it is necessary to exchange the initial routing table that was created at the time of the domain name registration described in Section 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

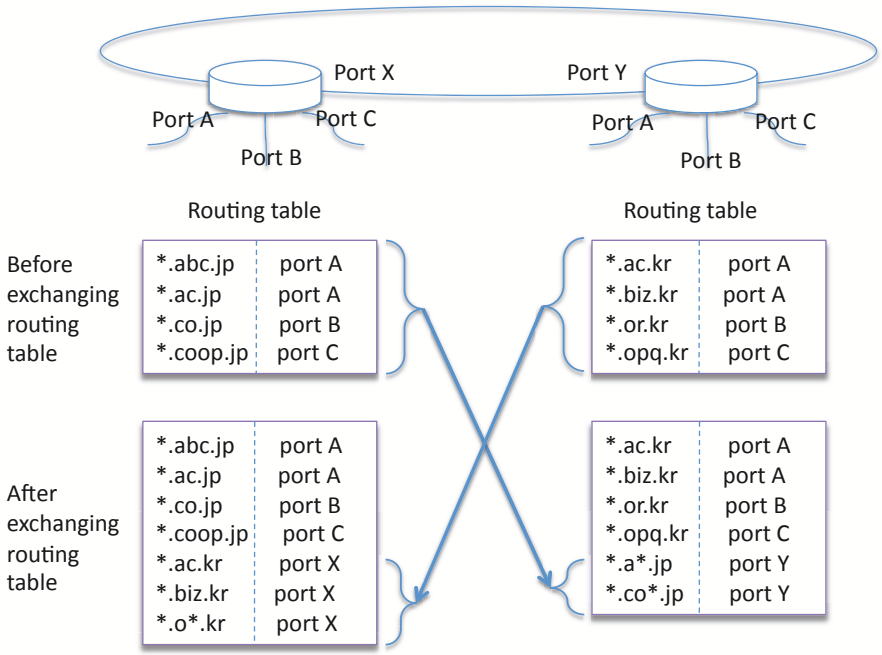


Figure 8: Exchange of routing tables

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.

An example of exchanging routing tables between two routers is shown in Figure 8. Ports A, B, and C are for connecting to the lower level of the hierarchy, which connects to domains such as `*.abc.jp` and `*.ac.kr`. Ports X and Y are for connecting to the same level of the hierarchy. The addresses are aggregated first and then exchanged.

The exchange of routing tables is achieved by extending the BGP-4. The specific mechanism is outside the scope of this thesis and remains to be investigated.

3.4 Packet Forwarding

Routers forward packets on the basis of the information contained in the routing table stored in TCAM. An example of packet forwarding is shown in Figure 9. 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

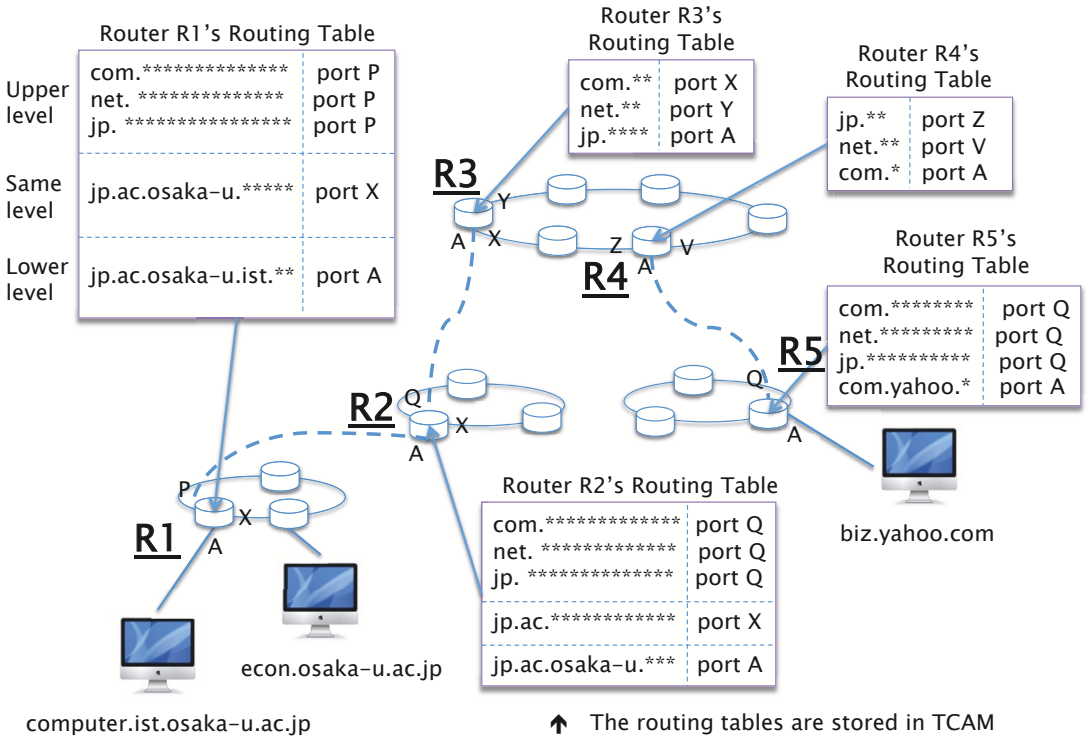


Figure 9: Packet forwarding

level, the same level, and the lower level. Since the packets from `computer.ist.osaka-u.ac.jp` are destined for a higher level of 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 destination domain name, `biz.yahoo.com`.

Routing information for each level is searched for using the corresponding bit positions of the search key. As shown in Figure 10, the TLD name is searched for in the highest level, and the 2nd and 3rd-level domain names are searched for in the lower levels.

Similar to in Section 3.3, packet forwarding in domain name routing is done by extending BGP-4, using character information instead of the IP address. BGP has four different types of messages: open, update, keepalive, and notification. The packet header of the BGP update message is shown in Figure 11. The Network Layer reachability information field defines the network

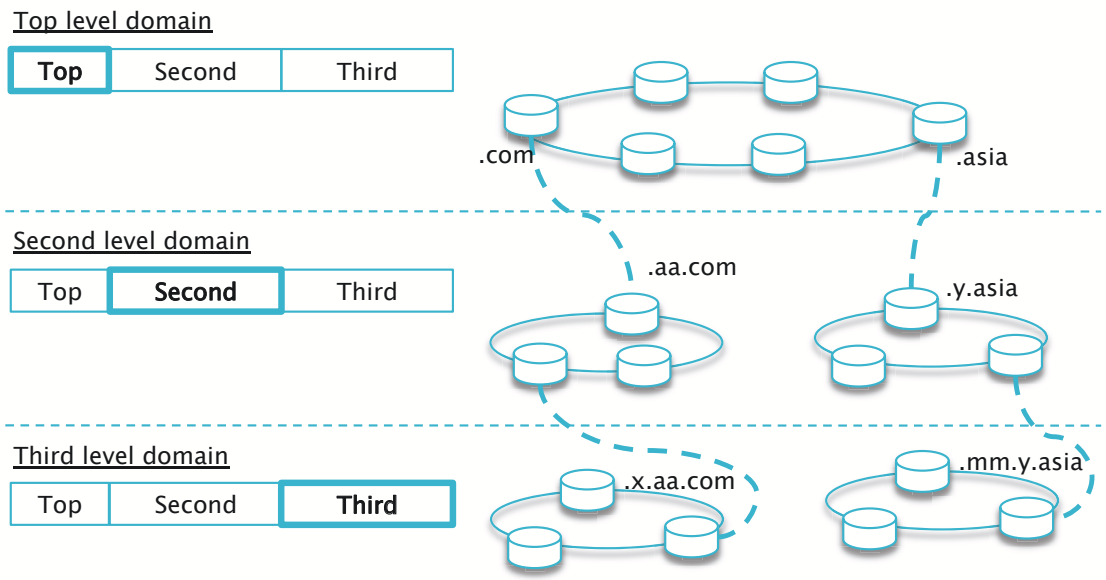


Figure 10: Search key and hierarchical structure

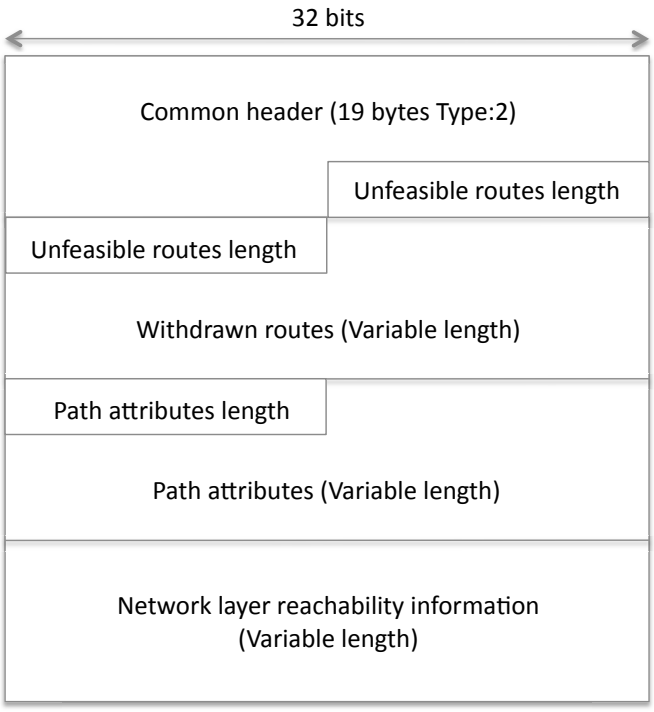


Figure 11: Packet header of BGP update message

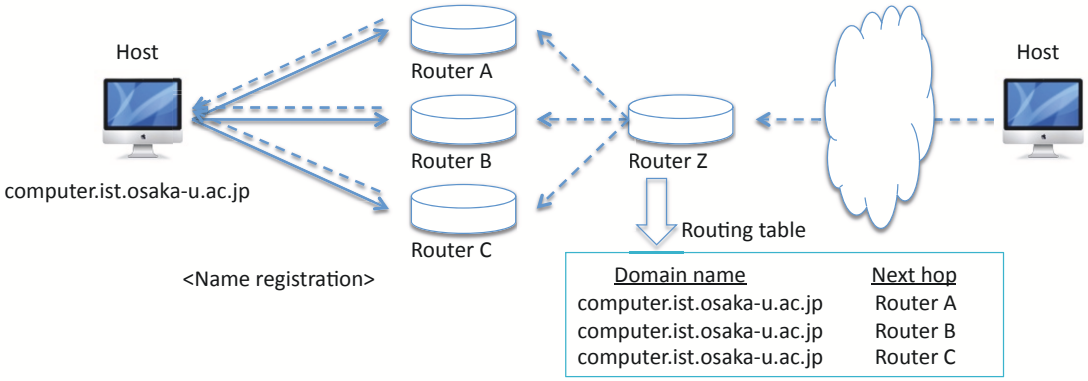


Figure 12: Mirroring in domain name routing

that is actually advertised by this message. The length field defines the number of bits in the prefix, and the IP address prefix in the "Network Layer reachability information" field defines the common part of the network address [26]. In domain name routing, this field can be used for the network character length and the network part by converting the character into ASCII (American Standard Code for Information Interchange) code in binary. The packet header processing performance could be degraded if the field were of variable length. To solve this problem, the domain name could be divided into two parts: (i) TLD plus the 2nd-level domain name to compose the static length and (ii) the rest of the domain name to compose the variable length. When the packets are forwarded, the static field is used to do most of the routing and the variable length is used when only one or two hops remain.

3.5 Mirroring and Failure Handling

Implementing mirroring in domain name routing makes it possible to choose from among different paths when packets are forwarded from the source to the destination, which reduces the burden of using a single path. It can also decrease the search time by ensuring that the routing information is obtained from a router that is physically closer.

Mirroring in domain name routing can be done by initially registering the domain names to more than one router. This increases the total number of routers required but will not cause much of a problem because the required number of routers is two orders of magnitude smaller than the number of currently existing routers, as shown in Section 5.2.

The example in Figure 12 shows a domain name being registered in routers A, B, and C. When router Z forwards packets that came from a node that wants to establish a connection with `computer.ist.osaka-u.ac.jp`, it can use any one of the paths to A, B, and C. However, when a domain name is registered in multiple routers in different 'rings', it is difficult to aggregate domain name suffixes in this case, which makes the router's routing table very large. For example, storing `jp.a`, `jp.b`, ..., `jp.z` in a single router can be represented by one entry: `jp.*`, but storing `jp.a`, `kr.b`, ..., `net.z` in a single router needs 26 entries. Therefore, it is necessary to mirror the 'rings' in the same level. It is possible to mirror 'rings' in Figure 10 if the 2nd-level domain names are identical even when the TLDs are different. For example, when the `jp` part of entry `jp.aa.*` is eliminated, the ring can be mirrored by the ring that has `com.aa.*`. The degree of duplication (as a percentage) of 2nd-level domain names of TLD compared with those of `.com` and `.net` is evaluated in Section 5.

The function for handling router failures is also shown in Figure 12. Even when one router out of routers A, B, and C fail, packets can still be sent to `computer.ist.osaka-u.ac.jp` because router Z still has other paths that are alive. Failures are detected by checking the periodic keepalive messages sent from the neighboring routers.

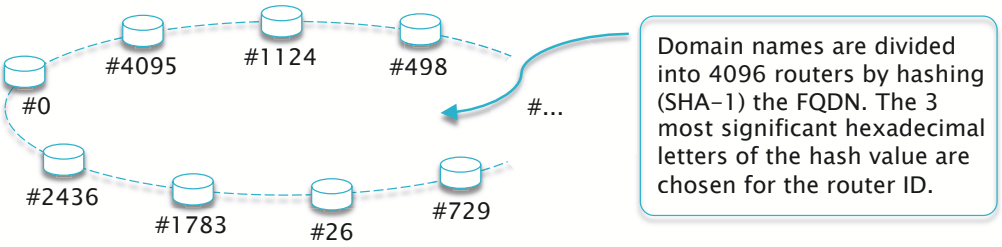


Figure 13: Hash-based distribution

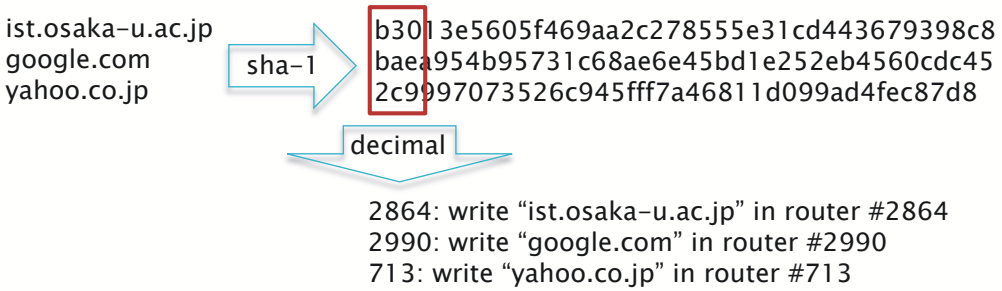


Figure 14: Example of distributing domain names to routers in hash-based distribution

4 Balanced Distribution of Domain Names in Routers

Compared with the IP routing tables, domain name routing tables are larger because the entry length depends on the variable length of the domain names. Therefore, it is important to distribute the routing information among multiple routers.

In this section, we propose three methods for equally distributing the routing information in the TCAM of multiple routers. These methods are evaluated in Section 5, where the required number of routers and their utilization ratio are discussed.

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 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

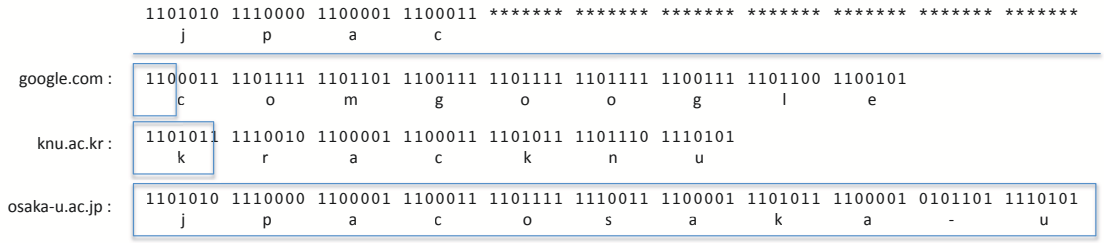


Figure 15: Example of longest alphabet matching

entry and then dividing the result by the TCAM size in a router. The advantage of hash-based distribution 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 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 [27] as the ratio of the number of physical hops taken by the protocol to reach the destination node from the source node and the shortest distance between the source and destination in terms of physical hops. Therefore, although it is considered ideal in terms of nearly equal distribution, it is not realistic for implementation.

The concept of this hash-based distribution is shown in Figure 13 and the algorithm is shown in Figure 14. First, SHA-1 is applied to each FQDN and the three most significant characters are taken. Since the output of SHA-1 is 160 bits represented by 40 hexadecimal characters, the three characters are router IDs ranging from 0 to 2^{12}. Each entry is stored in routers that have the corresponding hash value.

4.2 Hierarchical Longest Alphabet Matching

The hash-based distribution introduced in Section 4.1 is independent of the TLD. The other two methods take TLD into consideration.

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 defined as the 26 case insensitive letters of the English alphabet and the 10 numbers from 0 to 9 plus hyphens and periods in [28]. Inspired by the longest prefix matching scheme, longest alphabet matching is applicable when the domain names are represented in binary. As shown in Figure 15, when `jp.ac.*` is searched for, the entry that most closely matches the search key is returned as the output, so

the search is faster compared with finding the exact entry. In addition to the good searching speed, aggregation of multiple domain names with the same suffix is possible, which reduced the occupied memory space. These features show that hierarchical longest alphabet matching takes full advantage of TCAM, which can represent a don't care value '$\star$' 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 each TLD is shown in Algorithm 1.

Algorithm 1 Pseudocode 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
$entry\_length \leftarrow$ 180 (bits per one entry)
$router\_capacity \leftarrow 18 \times 10^6 \times 10 \times t$
$u \leftarrow$ entries with unique 2nd-level domain names
$e \leftarrow$ entries in the TLD
if e $\times\ entry\_length \leq router\_capacity$ **then**
  return $n_{L2} \leftarrow 1, n_{L3} \leftarrow 0$
else if u $\times\ entry\_length > router\_capacity$ **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

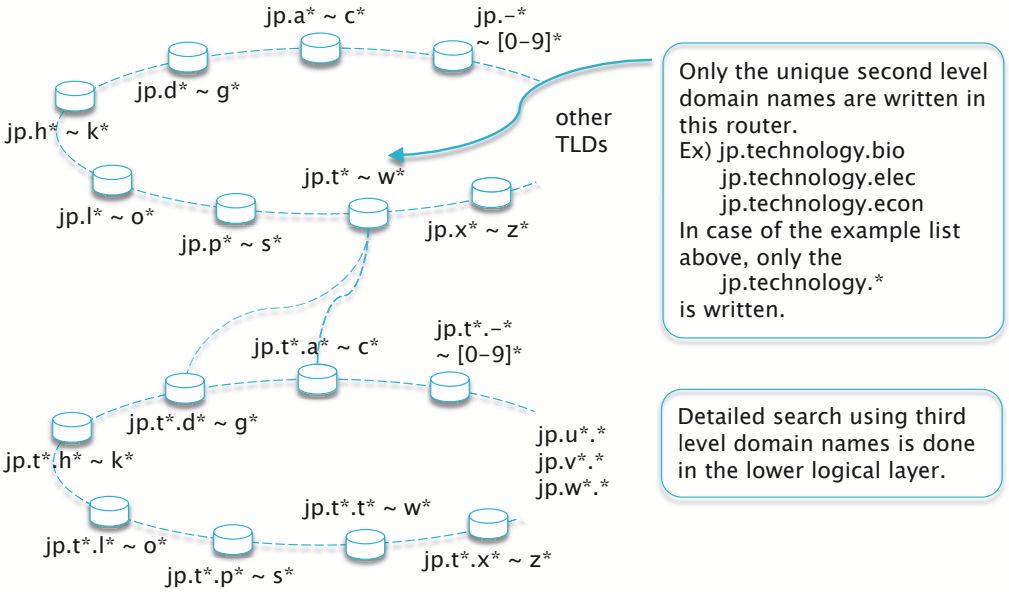


Figure 16: Hierarchical longest alphabet matching

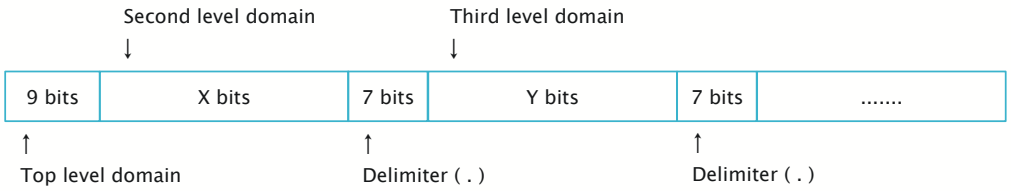


Figure 17: Structure of search key

For each TLD, the first step is to determine whether all domain name entries (e) would fit into a router. Each entry is multiplied by 180 bits ($entry\_length$) and divided by the router's available TCAM size ($router\_capacity$). We assume that one entry consumes 180 bits in order to make comparisons with the result in Section 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 mentioned in Section 3.2.

If e multiplied by $entry\_length$ is less than $router\_capacity$, then the total number of routers required in this TLD is set to 1, and the process ends. Otherwise, unique 2nd-level domain names (u) 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

Table 2: Example of group combination

Content	Group #	TCAM cell representation
hyphen and digits	Group 0	01*****
a to c	Group 3	11000**
d to g	Group 3	11001**
h to o	Group 2	1101***
p to q	Group 4	111000*
r to s	Group 4	111001*
t to w	Group 3	11101**
x to z	Group 2	1111***

are written dynamically by referring to Table 3 in the Appendix. The evaluation of this dynamic configuration is given in Section 5. 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 (Group 0). If the size is still too large, the letters are divided again into smaller groups, a group of `110****` and a group of `111****` (Group 1). This process is repeated until every router is well within the $router\_capacity$. One example combination of the grouping is given in Table 2.

When the partitioning of the 2nd-level domain names is over, the grouping is done with 3rd-level domain names, which are also shown in Table 3. 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}.

As shown in Figure 16, `bio.technology.jp`, `elec.technology.jp`, and `econ.technology.jp` are written as `jp.technology.*` in the upper level of the hierarchy. Throughout this thesis, domain names are written in reverse order (TLD → 2nd-level domain → 3rd-level domain) as shown in Figure 17, to show clearly how the hierarchical longest alphabet matching is done.

An evaluation of hierarchical longest alphabet matching is given in Section 5.

4.3 Hybrid Distribution

In Section 4.1, hash function (SHA-1) was applied to FQDNs and every domain name was considered to be 'flat'. In this section, TLDs are distinguished by 9 bits because there are fewer than

300 kinds of TLDs. The hash function is applied to 2nd and 3rd-level domain names separately. To calculate the total number of routers required, we first compute the number of routers required in each TLD. The pseudocode for each TLD is shown in Algorithm 2.

Algorithm 2 Pseudocode for hybrid distribution

$n_{L2} \leftarrow$ number of routers in 2nd level
$n_{L3} \leftarrow$ number of routers in 3rd level
$t \leftarrow$ threshold of TCAM utilization
$entry\_length \leftarrow$ 180 (bits per entry)
$router\_capacity \leftarrow 18 \times 10^6 \times 10 \times t$
$u \leftarrow$ entries with unique 2nd-level domain names
$e \leftarrow$ entries in a TLD
if e $\times$ $entry\_length \leq router\_capacity$ **then**
  return $n_{L2} \leftarrow 1, n_{L3} \leftarrow 0$
else if u $\times$ $entry\_length \leq router\_capacity$ **then**
  return $n_{L2} \leftarrow 1$
  return $n_{L3} \leftarrow (e \times entry\_length)$ / $router\_capacity$
else
  $n_{L2} \leftarrow (u \times entry\_length)$ / $router\_capacity$
  hash(2nd level domain name) % n_{L2} (divide routers in 2nd-level into groups)
  for each group
  calculate n_{L3}
  return $n_{L3} \leftarrow (e_{group} \times entry\_length)$ / $router\_capacity$
end if
END

For each TLD, the first step is to determine whether all domain name entries (e) would fit into a router. Each entry is multiplied by 180 bits ($entry\_length$) and divided by the router's available TCAM size ($router\_capacity$). Detailed explanations of the variables can be found in Section 4.2. If e multiplied by $entry\_length$ is less than $router\_capacity$, then the total number of routers required in this TLD is set to 1, and the process ends. Otherwise, unique 2nd-level domain names (u) are written in the 2nd-level of the hierarchy in the form of `TLD.uniq.*`. If this list of

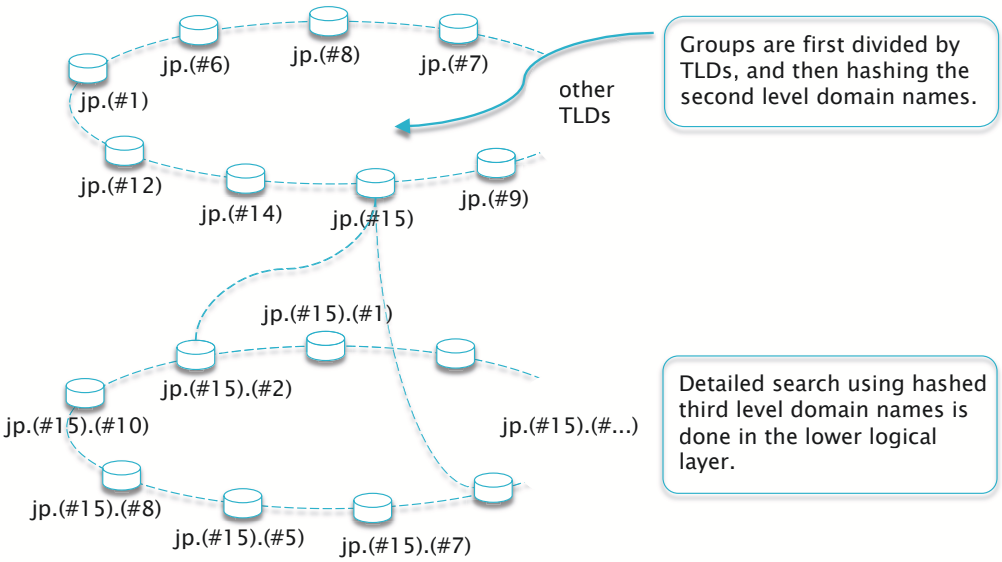


Figure 18: Hybrid distribution

u does not fit into a single router, we obtain the number of routers required in the 2nd level (n_{L2}) in the hierarchy by multiplying u by $entry\_length$ and dividing it into $router\_capacity$. 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 the number of routers required in the 3rd-level (n_{L3}); 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 $entry\_length$ and divided by $router\_capacity$. This was done assuming that hashing has the characteristic of equally distributing a large amount of data among groups. An example of the hybrid distribution scheme is shown in Figure 18.

5 Evaluation and Discussion

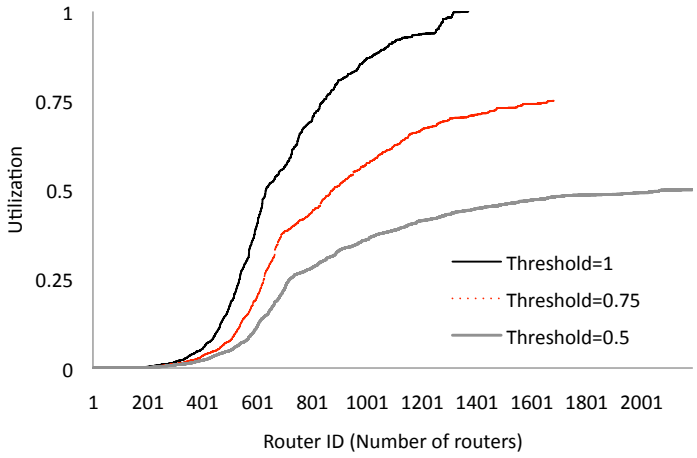
We evaluate the network and hardware resources required for domain name routing based on the routing methods proposed in Section 3 and domain name distribution methods proposed in Section 4. The evaluation is done with entries obtained from the ISC database in July 2008 [17].

5.1 Network Resources

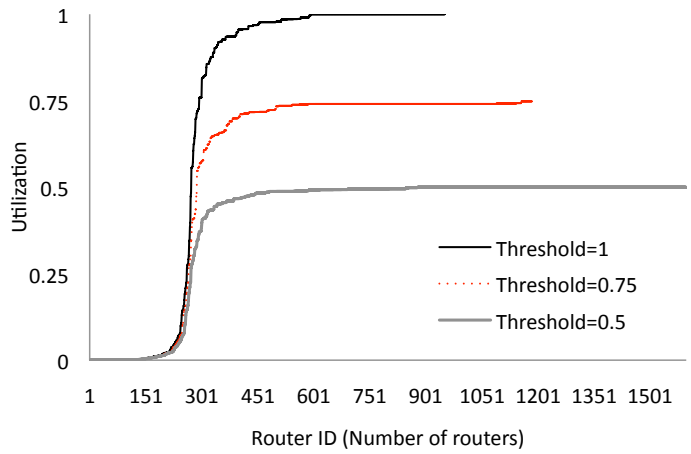
Figure 19 shows the distribution of domain name entries in routers, arranged in ascending order of the routers' utilization ratio, where the utilization is defined as the number of entries in a router multiplied by 180 bits divided by the router's memory size. The graph shows the results for hierarchical longest alphabet matching and hybrid distribution (proposed in Section 4) for threshold values of 1, 0.75, and 0.5, corresponding to the maximum percentage of router capacity allowed to be used being 100%, 75%, and 50%. For the allowed sizes of 75% and 50%, the remaining 25% and 50% of the memory is reserved for routing table updates. The total number of routers required is smallest when the threshold is 1: hierarchical longest alphabet matching requires 1396 routers and hybrid distribution requires 952 routers. Moreover, since the number of deployed routers is small, the average number of routing hops is expected 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 performances of both hierarchical longest alphabet matching and hybrid distribution can be improved by aggregating in the routers those entries that have a low utilization ratio. Figure 20 shows the improvement this achieves over the result in Figure 19. Since hierarchical longest alphabet matching and hybrid distribution 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 reduced 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 Section 3.1, needs careful consideration of the growth rate of domain names in each country.



(a) Hierarchical longest alphabet matching

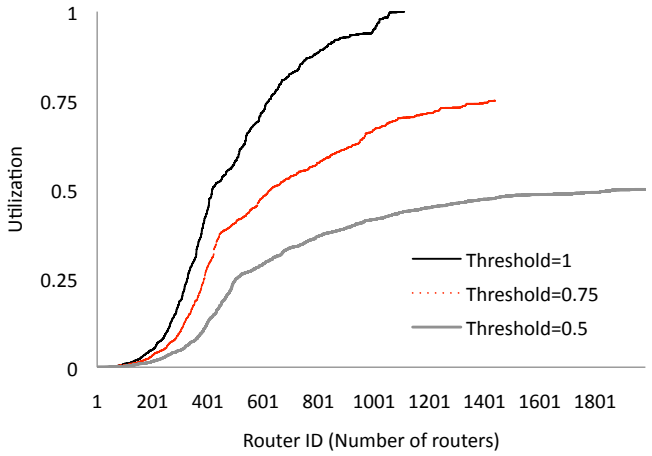


(b) Hybrid distribution

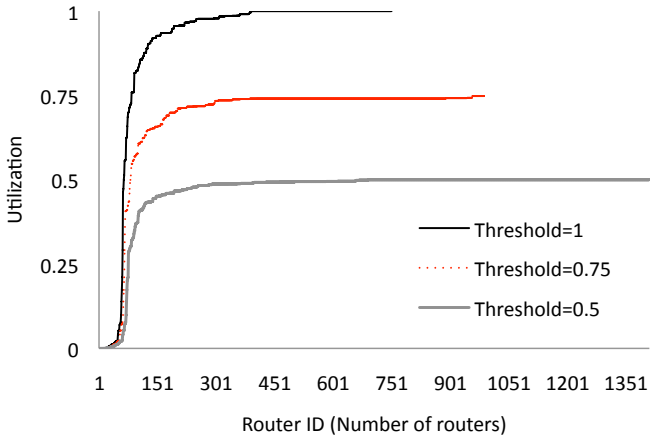
Figure 19: Distribution of domain name entries in routers for various different thresholds

Comparing these two algorithms, we can see that hybrid distribution does not need as many 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.

Trendlines for the number of routers required for various different thresholds in routers are shown in Figure 21. We obtained the number of routers required for thresholds of 0.4, 0.5, 0.6, 0.75, and 1 and performed curve fitting with cubic spline interpolation to obtain the number required for other threshold values. 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



(a) Hierarchical longest alphabet matching



(b) Hybrid distribution

Figure 20: Distribution of domain name entries in routers for various different thresholds (improved)

storage of existing domain names with a threshold value of 0.2 would require approximately 4000 routers for both hierarchical longest alphabet matching and hybrid distribution. It is interesting to see that the two trendlines cross over. We would expect to get a more accurate plot by obtaining the calculation results for both algorithms with a threshold value below 0.3. If the results in Figure 21 hold, we can say that hybrid distribution does not always give a better result than hierarchical longest alphabet matching.

Figure 22 evaluates the duplication rate of the TLD's 2nd-level domain names in comparison with `.com` and `.net`, as mentioned in Section 3.5. Figure 22(a) shows the duplication percentage

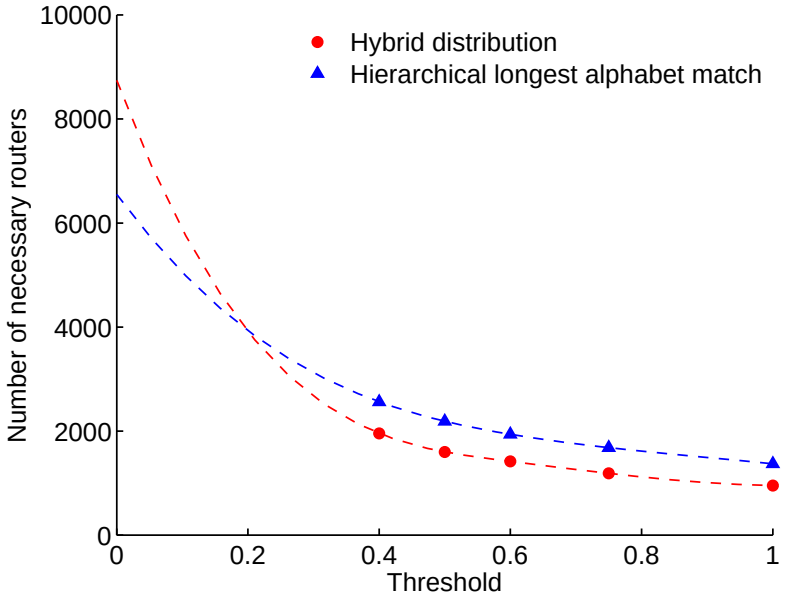
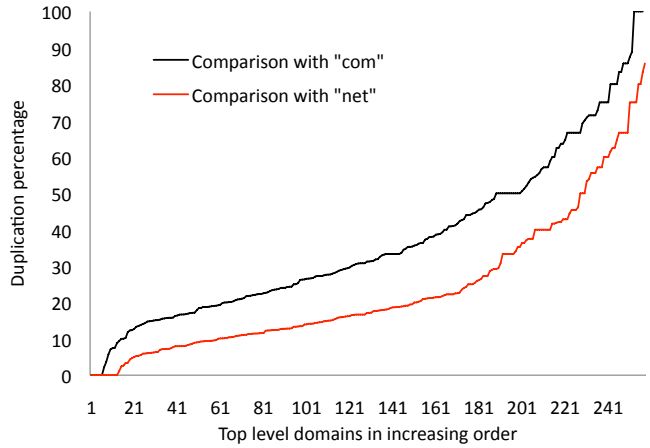
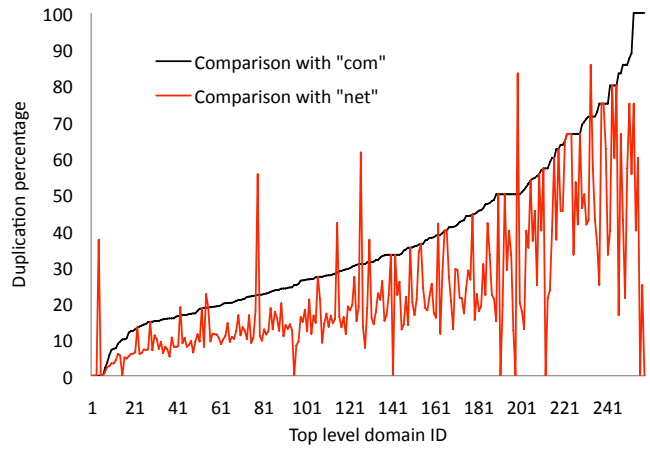


Figure 21: Approximation of number of necessary routers required for various different thresholds

arranged in ascending order. The average values are 36.9% for `.com` and 23.3% for `.net`. In addition, Figure 22(b) compares the duplication for the same router ID, where `.com` is arranged in ascending order and `.net` is arranged to correspond to the same TLD. Most of the TLDs have a lower duplication percentage with `.net` than with `.com`, with the exception of `.sl`, `.asia`, `.cd`, `.af`, `.ne`, `.et`, `.ye`, `.tm`, `.yu`, and `.gh`. The above duplication ratios show that the mirroring levels of rings proposed in Section 3.5 can be considered to be feasible.



(a) Comparison with “com” and “net”, increasing order



(b) Comparison with “com” and “net”, increasing order of com and same ID of net

Figure 22: Percentage of duplication in each TLD compared with com and net

5.2 Hardware Resources

The balanced distribution of entries among 4096 routers in hash-based distribution is shown in Figure 23. The average number of entries in a router is 145,976. As mentioned in Section 4.2, we obtained the memory size of 26 Mbits by multiplying 180 bits (160 bits of SHA-1 output plus 16 bits of output port plus 4 bits of parity bits) by the number of entries. Thus, having two 18-Mbit TCAMs is sufficient when there are 4096 routers. In other words, if there are ten 18-Mbit TCAMs in a router, a total of 597 routers ((145,796 entries $\times$ 180 bits $\times$ 4096 routers) $\div$ (18 $\times$ 10^6 bits $\times$ 10 TCAMs) = 597) are required to store the existing domain names.

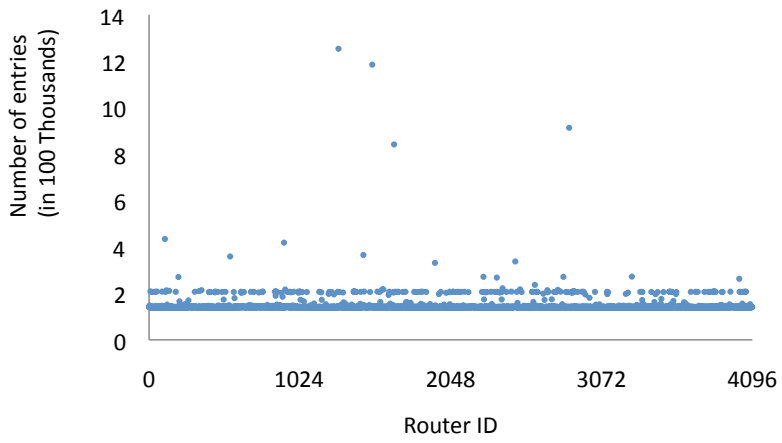


Figure 23: Balanced distribution of entries among routers in hash-based distribution

Figure 23 does not show the data for three FQDNs: `n003-000-000-000.static.ge.com`, `nothing.attdns.com`, and `reserved-multicast-range-not-delegated.example.com` because their duplication rates are two orders of magnitude larger than those of the other entries. Large values around router #2500 are also duplicated entries such as `walmart.com` and `unknown.net.reach.com`.

The number of routers required in hierarchical longest alphabet matching is the sum of routers in the 2nd and 3rd levels of the hierarchy. Assuming that one entry consumes 180 bits, the number of routers required to store the existing domain names is 1396. In hierarchical longest string matching, domain names are written in the form of ASCII code. To distinguish the characters used in the domain names (letters, digits, hyphens, and periods), 7 bits are sufficient. 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. Each character is 7 bits and we must evaluate whether TCAM can cope with longer entries.

The length distribution of the FQDN in ccTLD and gTLD is shown in Figure 24. From the ISC database, there are 662,703,300 legal FQDNs, where legal means the FQDN consists only of letters, digits, hyphens, and periods. 99% of the FQDNs have fewer than 50 characters and the average lengths are 32.95 and 34.89 for ccTLD and gTLD, respectively.

To satisfy the static length of 180 bits that we used to evaluate the required number of routers

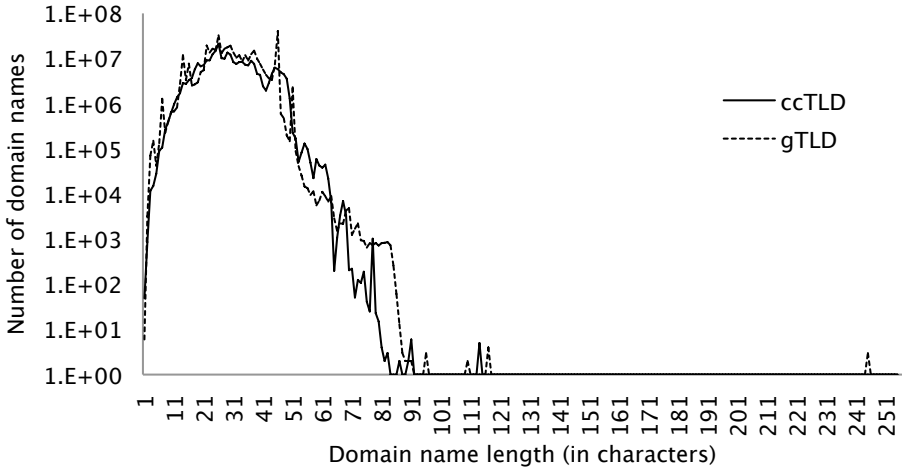


Figure 24: Distribution of domain name length, excluding TLD length

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. From Figure 24, we can see that only 30% of FQDNs have fewer than 25 characters. Therefore, limiting an entry to 180 bits does not qualify for 70% of the FQDNs.

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 [9], 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 the sum of the routers in the 2nd and 3rd level of the hierarchy. Assuming that one entry consumes 180 bits, the number of routers needed to store the existing domain names is 952. To store the FQDN database, 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, as mentioned in Section 3.1. 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.

The three distribution methods hash-based distribution, hierarchical longest alphabet matching, and hybrid distribution are compared in Figure 25. Of the 266 TLDs in the database, only the

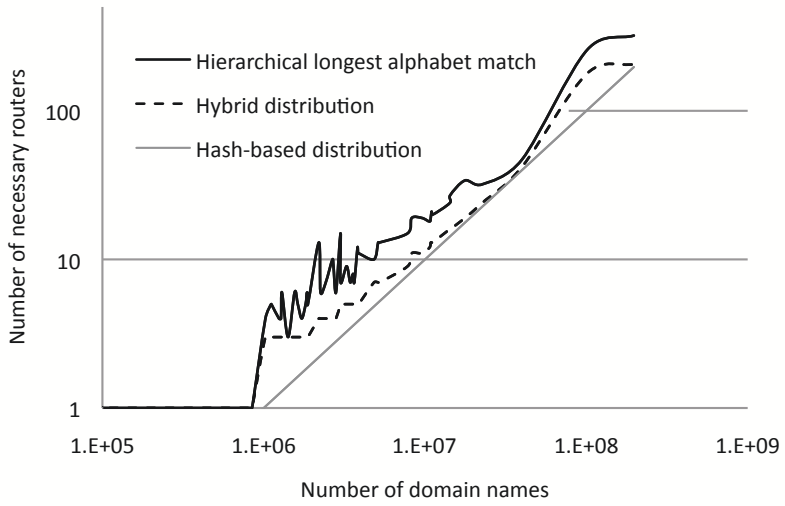


Figure 25: Comparison of three storage methods

76 that have more than 10^5 domain names were considered. On the x axis, the TLDs are arranged in ascending order according to the number of entries they have; the y axis shows the required number of routers. The number of routers required for hash-based distribution is proportional to the entries in each TLD. The values on the y-axis are discrete so the spikes do not have any significant meaning.

In [23, 29], the authors use the work from [30] 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 thesis are realistic values that indicate that a name-based routing architecture is feasible.

6 Conclusion and Future Work

Technologies evolve with the users' need for a better quality of service. To establish the foundation of future-generation networks, we need sophisticated routing mechanisms such as those based on the content or the name of the information.

In this thesis, 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. The proposal is scalable for name registration, routing table exchange, and packet forwarding. In addition, mirroring and failure handling functionality can be implemented to alleviate the burden on the system. The evaluation of hardware resources required for the three algorithms for distributing domain names among the TCAM of multiple routers by using a hash function, longest alphabet matching, or both methods at the same time showed that the total number of routers required is two orders of magnitude lower than the number of currently existing routers.

Future work involves the 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, since routers forward packets by using the domain names, the effect of eliminating the current domain name servers should be investigated.

Acknowledgments

This thesis would not have been possible without the contributions of several people. First and foremost, I would like to express my deepest gratitude to Professor Masayuki Murata for giving me the chance to study under his supervision and for offering me his invaluable comments throughout the study, and his strong encouragement on my path to pursuing the doctoral course. I am also grateful to Associate Professor Shingo Ata of Osaka City University for his kind advice and guidance throughout my research. His suggestion of this highly interesting and timely research topic as well as introductions to collaborators from industry are deeply appreciated.

Furthermore, I must acknowledge Associate Professor Naoki Wakamiya, Assistant Professor Shin'ichi Arakawa, Associate Professor Go Hasegawa, and Assistant Professor Yuichi Ohsita of Osaka University for their valuable comments and suggestions on this study. I am very grateful to Professor Koso Murakami, Professor Makoto Imase, Professor Teruo Higashino, and Professor Hirotaka Nakano for their critical reviews and comments from various angles.

Very special thanks go to Specially Appointed Associate Professor Kenji Leibnitz for the countless number of times he proofread the papers I submitted to conferences and journals and for his proofreading of this thesis as well.

In addition, I would also like to thank Dr. Kazunari Inoue and Koji Yamamoto of Renesas Technology Corporation for their technical advice on TCAM. The summer internship in 2007 was a priceless experience.

I thank all the members of the Advanced Network Architecture Laboratory at the Graduate School of Information Science and Technology, Osaka University, for their support and encouragement and for providing a pleasant and collegial atmosphere.

Last but certainly not least, I would like to thank my parents and sister who have been always been there for me, giving me never ending love and mental support.

References

[1] "BGP Reports." available at http://bgp.potaroo.net/.

[2] "GENI (Global Environment for Network Innovations)." available at http://www.geni.net/.

[3] "FIND (Future InterNet Design)." available at http://www.nets-find.net/.

[4] "Seventh Framework Programme." available at http://cordis.europa.eu/fp7/home_en.html.

[5] "AKARI: Architecture Design Project that Illuminates the Path to the New Generation Network." available at http://akari-project.nict.go.jp.

[6] K. Inoue, D. Akashi, M. Koibuchi, H. Kawashima, and H. Nishi, "Semantic Router using Data Stream to Enrich Services," in *Proceedings of the International Conference on Future Internet Technologies (CFI 2008)*, pp. 20–23, June 2008.

[7] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, "A Scalable Content-Addressable Network," in *Proceedings of the Special Interest Group on Data Communications (SIGCOMM 2001)*, vol. 31, pp. 161–172, August 2001.

[8] D. Farinacci, V. Fuller, and D. Oran, "Locator/ID Separation Protocol (LISP)," *"draft-farinacci-lisp-10.txt", IETF Network Working Group, work in progress*, November 2008.

[9] 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.

[10] R. Steinmetz and K. Wehrle, "Peer-to-peer Systems and Applications," *Lecture Notes in Computer Science*, 2005.

[11] M. Gritter and D. R. Cheriton, "An Architecture for Content Routing Support in the Internet," in *Proceedings of the 3rd Usenix Symposium on Internet Technologies and Systems (USITS 2001)*, pp. 37–48, March 2001.

[12] C. A. Shue and M. Gupta, "Packet Forwarding: Name-based Vs. Prefix-based," in *Proceedings of the 10th IEEE Global Internet Symposium (GI 2007)*, pp. 73–78, May 2007.

[13] D. R. Cheriton and M. Gritter, "TRIAD: A New Next-Generation Internet Architecture," tech. rep., Computer Science Department, Stanford University, July 2000. available at `http://www.dsg.stanford.edu/triad`.

[14] Haesung Hwang, Koji Yamamoto, Shingo Ata, Kazunari Inoue, Masayuki Murata, "Efficient Management of Access Control List by Combining Prefix Expansion and Range Matching Devices," *Technical Report of IEICE (IN2007-99–118)*, pp. 37–42, December 2007. (in Japanese).

[15] Haesung Hwang, Koji Yamamoto, Shingo Ata, Kazunari Inoue and Masayuki Murata, "Minimization of ACL Storage by Adding Minimal Hardware of Range Matching and Logical Gates to TCAM," *International Conference on High Performance Switching and Routing (HPSR 2008)*, pp. 116–122, May 2008.

[16] Shingo Ata, Haesung Hwang, Koji Yamamoto, Kazunari Inoue, Masayuki Murata, "Management of Routing Table in TCAM for Reducing Cost and Power Consumption," *Technical Report of IEICE (NS2007-119–129)*, pp. 7–12, January 2008. (in Japanese).

[17] Internet Systems Consortium, "Internet Domain Survey." available at `https://www.isc.org/solutions/survey`.

[18] L. Garcés-Erice, E. W. Biersack, P. A. Felber, K. W. Ross, and G. Urvoy-keller, "Hierarchical Peer-to-Peer Systems," in *Proceedings of the ACM/IFIP International Conference on Parallel and Distributed Computing (Euro-Par 2003)*, pp. 643–657, August 2003.

[19] P. Kersch, R. Szabo, Z. L. Kis, M. Erdei, and B. Kovács, "Self Organizing Ambient Control Space: An Ambient Network Architecture for Dynamic Network Interconnection," in *Proceedings of the 1st ACM Workshop on Dynamic Interconnection of Networks (DIN 2005)*, pp. 17–21, September 2005.

[20] J. Lian, S. Naik, and G. B. Agnew, "Optimal Solution of Total Routing Table Size for Hierarchical Networks," in *Proceedings of the IEEE Symposium on Computers and Communications (ISCC 2004)*, pp. 834–839, June 2004.

[21] "Abilene Network." available at `http://www.internet2.edu/network/`.

[22] L. Li, D. Alderson, W. Willinger, and J. Doyle, "A First-Principles Approach to Understanding the Internet's Router-level Topology," in *Proceedings of the Special Interest Group on Data Communications (SIGCOMM 2004)*, vol. 34, pp. 3–14, August 2004.

[23] S.-H. Yook, H. Jeong, and A.-L. Barabási, "Modeling the Internet's Large-scale Topology," *Proceedings of the National Academy of Sciences*, vol. 99, pp. 13382–13386, October 2002.

[24] J. Hawkinson and T. Bates, "RFC 1930: Guidelines for Creation, Selection, and Registration of an Autonomous System (AS)," March 1996.

[25] Y. Rekhter and T. Li, "RFC 1771: A Border Gateway Protocol 4 (BGP-4)," March 1995.

[26] Behrouz A. Forouzan, *TCP/IP Protocol Suite*. McGraw-Hill Science/Engineering/Math, 2002.

[27] G.-H. Lu, S. Jain, S. Chen, and Z.-L. Zhang, "Virtual Id Routing: A Scalable Routing Framework with Support for Mobility and Routing Efficiency," *Proceedings of the 3rd international workshop on Mobility in the evolving internet architecture*, pp. 79–84, 2008.

[28] P. Mockapetris, "RFC 1035: Domain names - Implementation and Specification," November 1987.

[29] A. Lakhina, J. W. Byers, M. Crovella, and I. Matta, "On the Geographic Location of Internet Resources," *IEEE Journal on Selected Areas in Communications*, vol. 21, pp. 934–948, August 2003.

[30] R. Govindan and H. Tangmunarunkit, "Heuristics for Internet Map Discovery," in *Proceedings of 19th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2000)*, vol. 3, March 2000.

Appendix

ASCII Table

Table 3: ASCII table in binary and prefix groups

Glyph	ASCII (binary)	Group 0	Group 1	Group 2	Group 3	Group 4	Group 5
-	0101101	01⋆⋆⋆⋆⋆	01⋆⋆⋆⋆⋆	01⋆⋆⋆⋆⋆	01⋆⋆⋆⋆⋆	01⋆⋆⋆⋆⋆	01⋆⋆⋆⋆⋆
0	0110000						
1	0110001						
2	0110010						
3	0110011						
4	0110100						
5	0110101						
6	0110110						
7	0110111						
8	0111000						
9	0111001						
a	1100001	11⋆⋆⋆⋆⋆	110⋆⋆⋆⋆	1100⋆⋆⋆	11000⋆⋆	1100001	1100001
b	1100010					110001⋆	1100010
c	1100011						1100011
d	1100100				11001⋆⋆	110010⋆	1100100
e	1100101						1100101
f	1100110					110011⋆	1100110
g	1100111						1100111
h	1101000			1101⋆⋆⋆	11010⋆⋆	110100⋆	1101000
i	1101001						1101001
j	1101010					110101⋆	1101010
k	1101011						1101011
l	1101100				11011⋆⋆	110110⋆	1101100
m	1101101						1101101
n	1101110					110111⋆	1101110
o	1101111						1101111
p	1110000		111⋆⋆⋆⋆	1110⋆⋆⋆	11100⋆⋆	111000⋆	1110000
q	1110001						1110001
r	1110010					111001⋆	1110010
s	1110011						1110011
t	1110100				11101⋆⋆	111010⋆	1110100
u	1110101						1110101
v	1110110					111011⋆	1110110
w	1110111						1110111
x	1111000			1111⋆⋆⋆	11110⋆⋆	111100⋆	1111000
y	1111001						1111001
z	1111010					1111011	1111010