Realizing Name-based Routing in the Network Layer

Haesung Hwang

January 2012

List of Publications

Journal Papers

1. H. Hwang, S. Ata, K. Yamamoto, K. Inoue, and M. Murata, “A New TCAM Architecture for Managing ACL in Routers,” *IEICE Transactions on Communications*, vol. E93-B, no. 11, pp. 3004–3012, November 2010.

2. H. Hwang, S. Ata, and M. Murata, “Resource Name-based Routing in the Network Layer,” submitted to *Journal of Network and Systems Management (JNSM)*, June 2011, Revised September 2011.

3. H. Hwang, S. Ata, K. Inoue, and M. Murata, “A New Memory Architecture for Realizing Name Lookup Tables in Content-centric Networks,” submitted to *Computer Networks*, October 2011.

Refereed Conference Papers

1. H. Hwang, K. Yamamoto, S. Ata, K. Inoue, and M. Murata, “Minimization of ACL Storage by Adding Minimal Hardware of Range Matching and Logical Gates to TCAM,” in *Proceedings of the 9th IEEE International Conference on High Performance Switching and Routing (HPSR 2008)*, pp. 116–122, May 2008.

2. H. Hwang, S. Ata, and M. Murata, “A Feasibility Evaluation on Name-based Routing,” in *Proceedings of the 9th IEEE International Workshop on IP Operations and Management*

(IPOM 2009), pp. 130–142, October 2009.

3. H. Hwang, S. Ata, and M. Murata, “The Impact of FQDN Database Updates on Name-based Routing Architecture,” in *Proceedings of the 5th IEEE/IFIP International Workshop on Broadband Convergence Networks (BcN 2010)*, pp. 16–21, April 2010.

4. H. Hwang, S. Ata, and M. Murata, “Frequency-aware Reconstruction of Forwarding Tables in Name-based Routing,” in *Proceedings of the 5th International Conference on Future Internet Technologies (CFI 2010)*, pp. 45–50, June 2010.

5. H. Hwang, S. Ata, and M. Murata, “Realization of Name Lookup Table in Routers Towards Content-centric Networks,” in *Proceedings of the 7th International Conference on Network and Service Management (CNSM 2011)*, October 2011.

Non-Refereed Technical Papers

1. H. Hwang, K. Yamamoto, S. Ata, K. Inoue, and M. Murata, “Efficient Management of Access Control List by Combining Prefix Expansion and Range Matching Devices,” *Technical Report of IEICE (IN2007-105)*, vol. 107, no. 378, pp. 37–42, December 2007. (in Japanese).

2. A. Shingo, H. Hwang, K. Yamamoto, K. Inoue, and M. Murata, “Management of Routing Table in TCAM for Reducing Cost and Power Consumption,” *Technical Report of IEICE (NS2007-120)*, vol. 107, no. 443, pp. 7–12, January 2008. (in Japanese).

3. H. Hwang, S. Ata, and M. Murata, “A Feasibility Analysis of Name-based Routing by Routers,” *Technical Report of IEICE (IN2008-178)*, vol. 108, no. 458, pp. 273–278, March 2009. (in Japanese).

4. H. Hwang, S. Ata, and M. Murata, “Mapping between Logical and Physical Topologies in Name-based Routing,” *Technical Report of IEICE (IN2009-167)*, vol. 109, no. 449, pp. 139–144, March 2010. (in Japanese).

5. H. Hwang, S. Ata, and M. Murata, “Feasibility of Name Lookup Table in Routers for Realizing Content-centric Networks,” *Technical Report of IEICE (IN2010-149)*, vol. 110, no. 449, pp. 31–36, March 2011. (in Japanese).

Preface

New network architectures and routing technologies are currently being proposed to reflect the current trend in Internet communication toward becoming more and more *resource-centric*: the demand for the hierarchically structured resource in the network is emphasized rather than the physical location of the resource. Therefore, routing the packets not only with an explicit destination address but also based on the content of the message is more suitable for the future Internet with some attractive advantages such as reducing the burden of resolving the identifier to location and increasing the scalability by using a provider-independent addressing structure. In order to reflect this paradigm shift in the network architecture from host-centric to resource-centric communication, one of the most fundamental issues is whether the network layer devices, i.e. routers, can support the shift as well. When the routing is performed based on the content of the packet, the information that is stored in the forwarding lookup table changes as well. Specifically, the number of the destinations depend on the type of resource and also on the number of the users who request this resource. Therefore, the information to store in the forwarding table becomes larger than the conventional IP address, triggering the need for a change in the storage location as well. In other words, in order to support the new storage and the new lookup mechanism, the traditional method of storing the IP addresses should be reconsidered and redesigned to suit better for the future usage. In this thesis, *name* is considered as one of the *resource* attributes and is the primary element of the possible keys describing the resource.

This thesis begins by exploring the characteristics of Ternary Content Addressable Memory (TCAM) which is a special type of memory used in routers. The packets are forwarded by referring to the rules in the forwarding table used in the current Internet, whereas the packets are classified by referring to the rules in the access control list. The memory type used for storing these forwarding tables and the access control list to achieve a high-speed packet forwarding and classification is TCAM. However, TCAM uses more transistors than random access memory, resulting in a high power consumption and a high production cost. Therefore, it is necessary to reduce the entries written in the TCAM to minimize the utilized transistors. The proposal presents a new TCAM architecture by using the range matching devices integrated within the TCAM's control logic with an optimized prefix expansion algorithm. The proposed method reduces the number of entries required to express the access control list rules, especially when specifying port ranges. With less than ten range matching devices, the total number of the entries required to store the port ranges in the TCAM can be reduced to approximately 50% with less than 0.3% increase of the manufacturing cost.

The evaluation result from the above shows that storing the data in the TCAM requires a careful consideration of its characteristic. In addition to `0` and `1`, TCAM uses an arbitrary don't care value, '$\star$' which enables a faster searching speed but leads to a higher energy consumption compared to the general purpose memories. In other words, it is crucial to effectively use the $\star$ value to better utilize TCAM in storing the access control list. This confirmation is used in the next part of this thesis to investigate whether the routers are capable of storing the name information of each named node, therefore showing the feasibility of the proposed name-based routing. The main contribution of this part is on proposing how to distribute a large sized database of the name information among the routers. Using approximately 700 million existing fully qualified domain names, the evaluation result shows that resource name-based routing is feasible even when considering the limitations of the TCAM size in the routers. Utilizing a hash function and longest alphabet matching, inspired by the longest prefix matching, the name database is evenly distributed among all routers. The resulting number of routers is approximately 1% of the currently deployed number of all routers. In addition, the effects of dynamic updates of the database are evaluated by investigating whether the storage location of the names is migrated frequently. Furthermore, the proposed name-based routing has 56% shorter path lengths compared to the lookup system based on the overlay end nodes.

The last part of this thesis addresses one of the most challenging issues in realizing a name-based routing in the routers: how to manage the information of the numerous contents and the large-scale

number of the users. A new router lookup table structure is proposed to manage such information where the content itself is named and the interest of the users is related to the name of the content. In order to complete the packet forwarding within the network layer, i.e. the search for content which matches the interest, the routers acting as the rendezvous points of a publish/subscribe system should maintain the information of the content names and the users subscribing to the content. Three lookup table structures for the name lookup tables in the routers are proposed, each with a different combination of DRAM, SRAM, and TCAM depending on the usage purpose. The proposed memory architecture is evaluated with the parameters such as memory cost, latency, and utilization using real-life and synthetic databases that have a Zipf distribution. The lookup table that has the lowest manufacturing cost and the lowest latency for storing the given database within a fixed budget is presented. Depending on the lookup table, the read latency ranges from 75 μsec to 45 msec. In addition, it is shown that the chip cost is exponentially reduced from \$997 to \$17 for distributing the names and the subscriber information by increasing the number of the rendezvous points.

By thoroughly examining the issues regarding both the hardware and the network architecture addressed above, this thesis shows that realizing the name-based routing in the network layer using currently existing routers is a feasible technology for the future Internet.

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 heartily grateful to the members of my thesis committee, Professor Koso Murakami, Professor Makoto Imase, and Professor Teruo Higashino of Graduate School of Information Science and Technology, Osaka University, and Professor Hirotaka Nakano of Cyber Media Center, Osaka University, for their critical reviews and comments from various angles.

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 Professor Naoki Wakamiya, Associate Professor Shin'ichi Arakawa, Associate Professor Go Hasegawa, Assistant Professor Yuichi Ohsita, Assistant Professor Yuki Koizumi, and Assistant Professor Shinsuke Kajioka of Osaka University for their valuable comments and suggestions on this study.

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 Professor Kazunari Inoue of Information Engineering at Nara National College of Technology, Mr. Koji Yamamoto of Renesas Design Corporation, Mr. Hisashi Iwamoto, Mr. Yasuto Kuroda, and Mr. Yuji Yano of Renesas Electronics Corporation for

their technical advices on TCAM. The summer internship in 2007 was a priceless experience.

I gratefully acknowledge the funding sources that made my Ph.D. work possible. I was given scholarship for six years of master's and doctoral course by Japanese Ministry of education, culture, sports, science and technology. My work was also supported by GCOE (global centers of excellence) program of Osaka University and PREDICT (Promotion program for Reducing global Environmental loaD through ICT innovation) of Japanese Ministry of Internal Affairs and Communications.

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 always been there for me, giving me never ending love and mental support. And most of all for my loving, supportive, encouraging, and patient husband whose faithful support during the entire stages of this Ph.D. is so appreciated. Thank you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

After decades of a successful growth of the Internet, networking has become an integral part for the daily communication in the human life. The communication model in the early days of the Internet was a conversation between the one providing the access to a resource and the others wanting to use the resource [1]. Naturally, it was *who* and *where* that mattered the most - *who* being the identifier and *where* being the physical attachment point of the nodes in the Internet. Internet protocol (IP) addresses have been predominantly used for such communication among the heterogeneous devices in the Internet, simultaneously serving two principle functions: host/network interface identification, i.e. who, and location addressing, i.e. where. This function of IP addresses led to the problem of a poor scalability since the number of multi-homed sites that cannot be addressed as part of topologically- or provider-based aggregated prefixes are increasing [2]. In addition, the conventional communication in the Internet was mostly based on the fact that one knows how to reach a node that serves a specific purpose, such as knowing the IP address of the web server. Even if the exact IP address of a web site is not known, it could still be guessed from the name, and the known domain name system (DNS) servers resolve the name to the destination IP address. For example, the address of a company web site can be easily imagined as `company.com` and the DNS servers resolve the name to the associated IP address.

What happens if there is no prior knowledge about the structure nor keyword of a uniform resource locator (URL)? Generally search engines - an application service provided in the Internet - help resolving the location of the information source that the end user requests. However, search engines are for seeking information in major content providers which are mainly static nodes. In other words, they are not suitable in a situation where some content needs to be searched in local (such as a file server in the lab) or mobile nodes.

On the other hand, the current information retrieval and delivery are based on *what* (resource). In other words, the resource itself rather than the location is more important to the end nodes [3]. This trend entails a shift in the current Internet toward a new networking paradigm for the future Internet: a *resource-centric routing*. In highly intelligent network, the routing is possible not only with the numerical identifier (e.g. IP address) but also by the resource. Routing by resource implies that the network working as a 'broker' to intermediate the senders and the receivers of the information knows where to send the messages by looking at the content of that message when no explicit IP address is present [4]. In other words, when a resource source sends data to an end node that requests some piece of information, the source can specify the resource's destination by the resource's metadata instead of allocating the packet with a single IP address. Furthermore, nodes can notify devices in the network such as servers and other peer nodes of the resource's metadata that an end node requested and have the devices utilize the knowledge to find the optimal path for reaching the resource source. In short, the advantage of resource-centric networks is that the information source does not have to worry about the destination IP address of the receiver and send out the message to the network [5].

Resource-centric routing is a flexible communication model that meets the requirements of the content distribution trends of the future Internet. Figure 1.1 shows the direction of resource-centric routing. In the current IPv4, its addressing space can accommodate up to four billion nodes and the core routers have over three hundred thousand aggregated prefixes as shown in Figure 1.2 [6]. In addition, the number of currently active servers is more than one hundred million and the number of fully qualified domain name (FQDN) addresses in DNS is more than seven hundred million [7]. In summary, hundred million to billions is the target number for managing in the IP routing. When it comes to resource-centric routing, the routers need to manage a much more amount of information

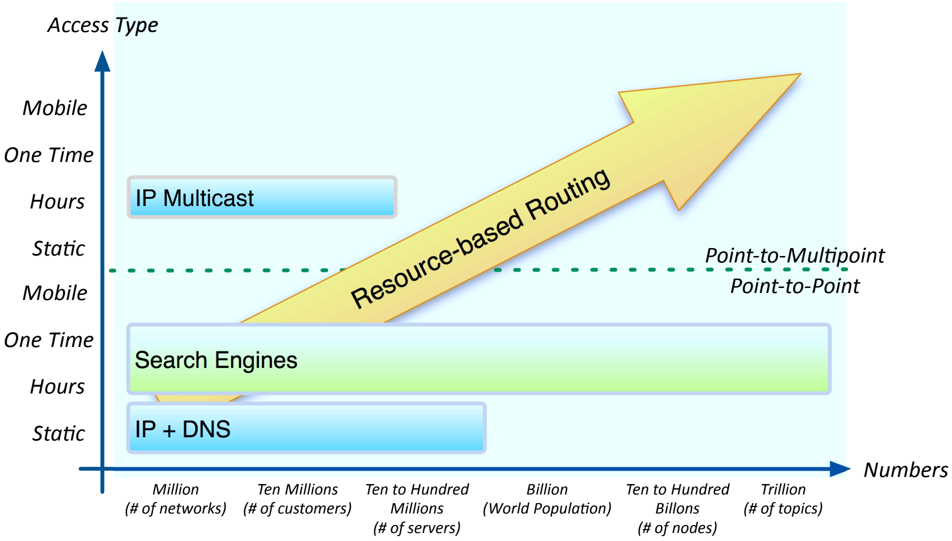


Figure 1.1: Direction of resource-centric routing

since the address of the destination is the content by itself. The information stored in the Internet is growing by a significant rate. Figure 1.3 shows the growth of volume stored in the Internet (in both number of bytes and web pages). The numbers are from different estimation sources and an average forecasting line is given. As shown in the figure, the volume is growing exponentially with 23% in bytes and 35% in pages of annual growth [8–10] and the total volume will exceed 20 zettabytes (2×10^{22}) before 2020. Moreover, Cisco estimated that there will be two networked devices per capita in 2015, up from one per capita in 2010. That is to say, the number of devices will be ten to hundred billions in the future. If each device has ten to hundred connected pages such as in Facebook [11], the total number of destination address would be around trillions. Therefore, the number of information types on destinations will increase more than hundreds of times than that of the location-centric routing, i.e., the network should treat different content as different destination address. In other words, the number of destination addresses will increase from the number of nodes to the number of types of contents and processing a huge volume of content is much more complicated than location-centric routing in resource-centric routing.

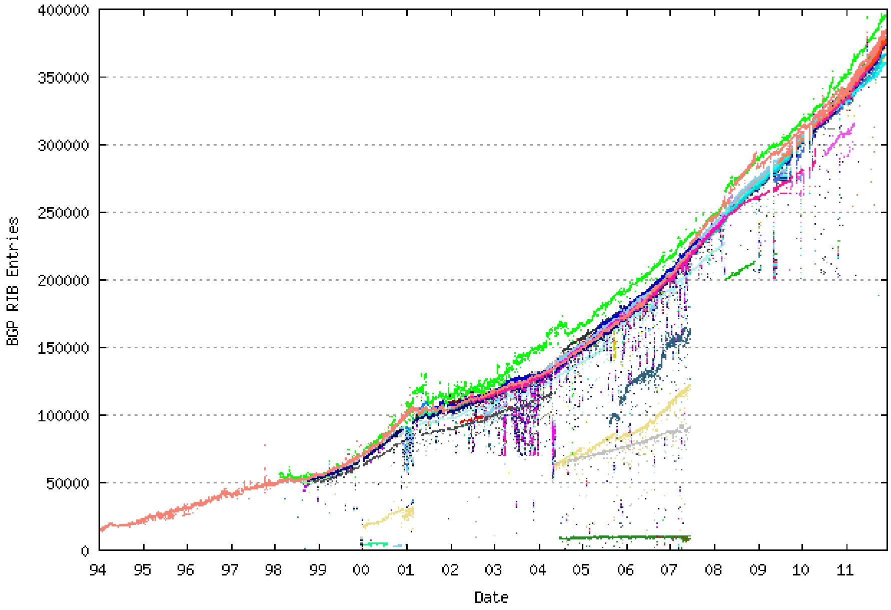


Figure 1.2: Growth of the border gateway protocol (BGP) table - 1994 to present

The recent global promotion of the resource-centric routing reflects the popular trend. For example, national science foundation (NSF) of the United States have launched a program called future Internet architecture (FIA) since 2010. Named data networking (NDN) [12] distinguishes data by its hierarchical names rather than its location, i.e. IP address. Specifically, the data consumers send out packets carrying a name that identifies the desired data and the routers send out the data which matches the given name. 4WARD [13], one of the many EU FP7 [14] projects, shares a similar goal to NDN where the information objects have their own identity, enabling connections to information objects rather than just to specific hosts. These work as well as the software defined networking (SDN) mainly put emphasis on the solutions based on the software. Although some consideration on implementing the hardware is concerned, it is not the core of their research. This thesis is motivated by the increase in the recent demand for routers with a higher specification, i.e. larger storage and higher processing speed. In addition, the data structures and the future large scale integration (LSI) design should also be taken into a consideration. Therefore, not only the architecture of the network but also that of the network layer hardware, i.e. router, should be reconsidered in order to make a considerable contribution to the future Internet since the routers will still play a

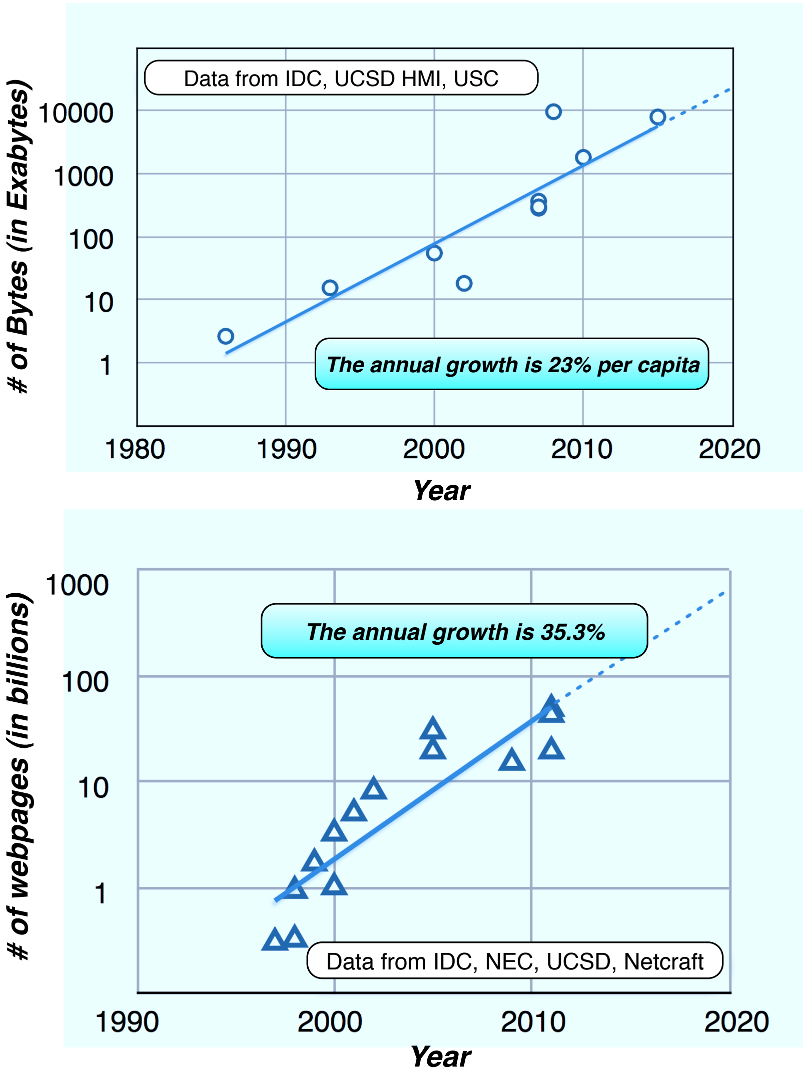


Figure 1.3: Growth of Internet

key role in the future Internet.

Specifically, the routers in the future Internet should be reconsidered with the below issues in mind.

- Sufficient storage for a large-scale information: As the information volume increases, the demand for an additional number of routers or higher storage space will increase. Since neither of the options are easy to apply to the current Internet, it is important to utilize the currently deployed hardware. Understanding the characteristic of the search/storage memory used in routers is one of the crucial steps to support a large-scale information in the routers.
- A new routing architecture using names in addition to IP addresses: In order to route with names as well, a new routing algorithm should be designed to take the advantage of the names. In addition, the network should be scalable upon the updates of the name database and the required number of the routers should not exceed that of the currently deployed.
- A new lookup scheme in the forwarding table: The information of resource in names and the users requesting the resource is stored in the forwarding table. As the names and users have a Zipf distribution, the memory used for the lookup and the storage should have an appropriate structure to minimize the search latency and the memory chip cost.

The following section addresses these issues in detail and states the proposed solution to each topic.

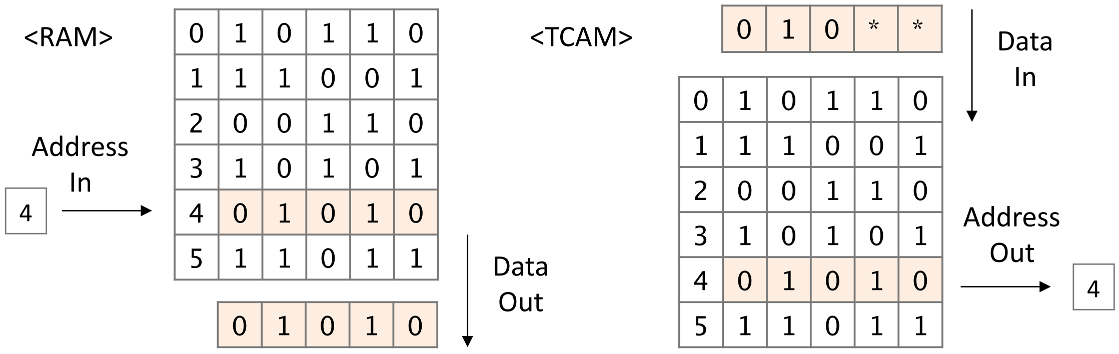
1.2 Issues for Name-based Routing within the Routers

1.2.1 Supporting a Large-scale Information by Understanding and Utilizing the TCAM

The need for a high-speed Internet communication has prompted many researchers to design fast network architecture. One of the crucial components in the architecture is the router. Routers are network devices used for forwarding and classifying packets and usually consist of a specialized operating system and memory. More specifically, when a router forwards the packets, it uses forwarding tables to associate destination networks with output ports. Upon every packet arrival, the table is searched for the appropriate entry specifying which port and the IP address the packet should be forwarded to. In the case of access control, the access control list (ACL) is searched to determine whether the packet should be permitted or denied, causing the packets to be forwarded to the destination or to be dropped. Examining and controlling the flow is a preliminary step to the name-based routing since when the resource is sent, names representing the particular resource should be examined in a similar manner.

With the growth of the Internet and its heterogeneity of new services and protocols, the above processes will become more and more important in the future. The average size of routing tables has shown a rapid increase, approaching 250,000 entries as of December, 2007. This number had doubled over the previous five years, and is expected to accelerate in growth [6]. Also, according to [15], the rules in an ACL of a typical enterprise network gateway consist of approximately 5,000 entries, a number that is also expected to increase due to greater awareness of network security. Access control policy is becoming more restrictive and shifting from ‘reject unnecessary traffic’ to ‘transmit only the minimal amount of legitimate traffic’ leading to a tendency toward more rules defining which specific packets are legitimate rather than just denying a group of packets. Therefore, in both cases, it is necessary to search for a certain object that quickly matches the given criteria from an enormous candidate set.

These routing tables and ACLs are written in high-end routers in a type of memory called content addressable memory (CAM). The difference between random access memory (RAM) and TCAM [16] is shown in Figure 1.4. 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 1.4, RAM takes address `4` as the input and returns `01010` as the result, whereas TCAM takes `010**` as the input and returns address `4` as the output. Unlike binary CAM, 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 diagram for TCAM is shown in Figure 1.5 and the way cell values are decided is indicated in Table 1.1. 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



0	1	0	1	1	0
1	1	1	0	0	1
2	0	0	1	1	0
3	1	0	1	0	1
4	0	1	0	1	0
5	1	1	0	1	1

0	1	0	1	0

0	1	0	*	*

0	1	0	1	1	0
1	1	1	0	0	1
2	0	0	1	1	0
3	1	0	1	0	1
4	0	1	0	1	0
5	1	1	0	1	1

Figure 1.4: Difference between RAM and TCAM

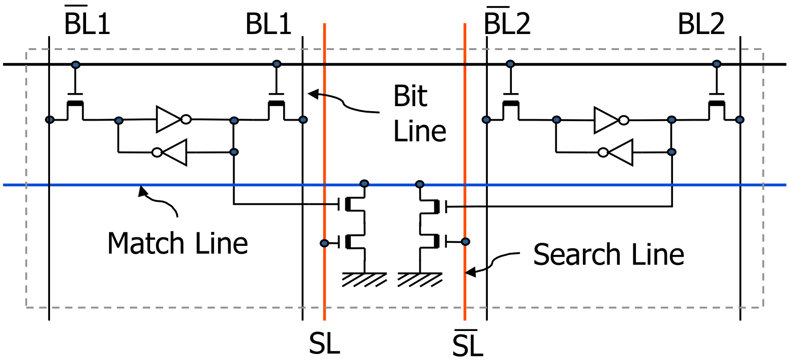


Figure 1.5: Circuit diagram of a TCAM chip

corresponding TCAM cell represents 1.

Having ⋆ 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 110000001010100010000000⋆⋆⋆⋆⋆⋆⋆⋆.

TCAM performs partial matching of the entries excluding the ⋆ 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* value to fully utilize the advantages of TCAM.

One of the other main advantages of the TCAM besides its search speed is its ability to express ranges. A typical type of data in the form of a range is the port number field of an ACL, which is

Table 1.1: 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

Table 1.2: Example of Access Control List

```
access-list 101 permit tcp host 10.1.1.2 host 172.16.1.1 eq telnet
access-list 102 deny tcp any range 1024 65535 any
access-list 103 permit ip 10.1.1.0 0.0.0.255 172.16.1.0 0.0.0.255
access-list 111 deny icmp any 10.1.1.0 0.0.0.255 echo
access-list 191 permit udp any any range 16384 16483
```

used in the form of range to allow/deny various applications requiring several port numbers as a set as in video plus audio chat. Table 1.2 shows an example of an ACL. ACL entries typically consist of five fields: source and destination IP addresses, source and destination port numbers, and protocol type. Packets are classified according to the corresponding rule and a necessary action (permit/deny) is performed only when all fields are matched. For example in Table 1.2, `access-list 101` permits the TCP packets with source IP address as `10.1.1.2` and with destination address as `172.16.1.1` that equals `telnet`. In BCAM, it is impossible to represent the range without writing every corresponding number within the range in the memory. For example, when writing the range `1024-65535` of `access-list 102` in a BCAM entry, the simplest form is to write every single number to exactly match the entire entry. However, it is obvious that 64,512 entries are required to write a single range which ends up consuming a huge number of entries in BCAM. This method is referred as *Full Expansion* in this paper.

Since full expansion is not a practical method, in most cases the range has to be divided into several subranges to be stored in the memory. The process of dividing ranges is called *Prefix Expansion (PE)*. Using TCAM's capability to use $\star$ value, PE expresses several port numbers as a single entry by aggregating the least significant bits as $\star$, which makes it possible to represent subranges in units of 2^i and those aligned on boundaries of powers of 2. The range `1024-65535` would then be represented by the following six lines as in Table 1.3.

Table 1.3: Prefix Expansion of 1024-65535

1***************	32768-65535
01**************	16384-32767
001*************	8192-16383
0001************	4096-8191
00001***********	2048-4095
000001**********	1024-2047

Compared to full expansion, the prefix expansion method can significantly reduce the number of TCAM entries needed. However, the configuration of ACL is usually performed manually and ranges based on decimal values (e.g. 10 – 20, 100 – 200) are commonly used. This might be a convenient setting for a human operator to understand, but it results in superfluous entries in TCAM which causes problems when TCAM exhaustion occurs. Partial application of the ACL occurs via software and the packets that match the rules that are not applied in the TCAM are processed by software [17], making wire speed search unachievable.

Despite the advantages of the TCAM, it also has a major drawback. In order to express a don't care bit, TCAM uses 16 transistors per a memory cell, whereas RAM uses 6. Therefore, it is desirable to write data in the TCAM more compactly, i.e. reduce the number of TCAM entries, using less space to save the number of the transistors needed.

The above characteristics of TCAM is taken into the consideration in this thesis to effectively lookup and store data especially the names of the resource.

1.2.2 Designing a New Routing Architecture: From IP to Name

One inspiring thought of the future Internet is being able to route on *name* in addition to IP address and implement this as a mechanism in the network layer. The binding of desired object to name and name to address is currently performed in services in the upper layer whereas in this thesis, the role of resolving names is substituted by routers so that the routing is done within the routers without consulting any external name servers. In other words, the mechanism for finding, querying, and fetching the desired information is performed within the routers themselves.

The advantages of the name-based routing in the network layer are:

- **Reduces the necessity of resolving names to IP addresses** - Conventional and current method of dealing with names is to first resolve them to IP addresses by DNS servers. Routers then route with IP addresses. When names are identifiers from the beginning, they can be mapped directly to locations by forwarding table in routers without having to be resolved to IP addresses.

- **Reduces the burden of resolving names to locations in devices other than routers** - It is possible to reduce a great number of management nodes, saving hardware and network requirements such as electricity, response time, and propagation delay. For example, it is reported that there are approximately 16 million name servers [18]. DNS is a scalable name resolving system but its resource utilization is low since a large number of name servers have to be operated for twenty-four hours seven days a week.

- **Increases overall network availability** - Named resources are searched using highly available routers instead of the conventional peer-to-peer (P2P) end-nodes or DNS servers. In other words, the routers can resolve names to locations even if the DNS servers go down therefore increasing the network accessibility.

With the limited network resource, the current DNS is the only way to resolve names to locations considering the performance. However, it is not desirable to completely eliminate the DNS server since it has important roles other than translating the domain name to the IP address, such as distributing the load of the web servers and resolving domain names of e-mails to mail servers. Therefore, it is the router that does the main job of resolving the names.

Among the numerous issues when realizing resource name-based routing in the network layer, some primary challenges and questions that have to be answered are listed below.

- **Network scalability** - Is the network topology scalable upon the routing information update, e.g. growth of the number of nodes? Names generally have less scalability than IP addresses especially when the name is flat, making aggregation of names in routing/forwarding tables more difficult.

- **Storage constraints** - Is storing routing information of names in the number of currently deployed routers possible? Names especially those in human-readable formats are typically longer than 32-bit IP addresses and it is undesirable if a large number of additional routers is required to support forwarding tables with name information as mentioned in the work of Shue et. al [19].
- **Performance** - What is the overhead of routing/forwarding by names compared to that of the conventional IP in routers? Substituting IP address with name information might result in low payload to packet size ratio as discussed in [1].
- **Mobility** - How are names managed and routed when the end nodes and the resource itself change their physical position in the network? Routing by names are thought to have better support for mobility since it separates the identifier from location. A mechanism for supporting roaming hosts is required as discussed in [20].
- **Routing path** - How does a router find out the optimal path to reach the desired information of a source node by names? When names of resource are flat, i.e. difficult to aggregate, the number of hops from source to destination might increase since the forwarding table is not organized in network addresses. However, maintaining a lot of name information to find out an optimal routing path causes scalability problem.
- **Security** - Who is eligible for injecting named resource? Or is the resource itself certified by a trusted host? As discussed in [1], the authentication of binding between names and resource is more important than conventional host authentication.

It is emphasized that the goal here is not to replace the DNS nor do the proposal in this thesis shows how to fully achieve name-based routing within the router. Rather, the main focus is on **network scalability** and **storage constraints** since those are considered as crucial issues in realizing resource name-based routing. Specifically, regarding the **network scalability**, it is shown that that the number of required routers increases with scalability rather than diverge exponentially when the fully qualified domain name (FQDN) database is updated. The reason why FQDN is used as an example of names is explained in Section 4.1.1. In addition, for the **storage constraints**, it is presented that the number of the required routers to store routing information of names is less than 1% compared to the number of currently deployed routers. Although the other challenges stated above are well recognized, these are not within the scope of this thesis and are left to be addressed in the future work.

1.2.3 Implementing Publication/Subscription Model in Routers

Determining the content's destination by its metadata indicates that there might be more than one receiver for a message. That is, a message that matches one or more conditions specified by multiple users should be sent to every user who requests the message. This is one of the biggest differences between location-centric network and resource-centric network where one-to-many communication style is demanded more than one-to-one communication. Exchanging duplicate packets in one-to-many communication among the individual nodes makes the complex large-scaled Internet more and more congested.

To realize one-to-many packet forwarding, extended IP multicast technology is promising. In the router, the multicast forwarding table is composed of multicast forwarding information base (FIB), adjacency table (ADJ), and multicast expansion table (MET). Especially for the MET that keeps the information of multiple output interface, current number of output interfaces is at maximum of 64,000 [21]. When the table is full, multicast packets can not be processed in the hardware and the software switching in the CPU is required which drastically degrades the performance due to its slow speed [22]. In order to catch up with the increasing number of topics in resource-centric routing, expanding the capacity of MET is a simple solution. However, it is necessary to consider the additional cost of making a detour around the router whose MET is full to avoid a drastic performance degradation.

Publish/subscribe (pub/sub) paradigm is considered as an effective communication model [23] for the future network architecture that solves many challenges of the current Internet and is appropriate for designing distributed systems due to its loose-coupled and asynchronous communication [24]. In addition, relevant data is delivered to the consumers according to the interest they

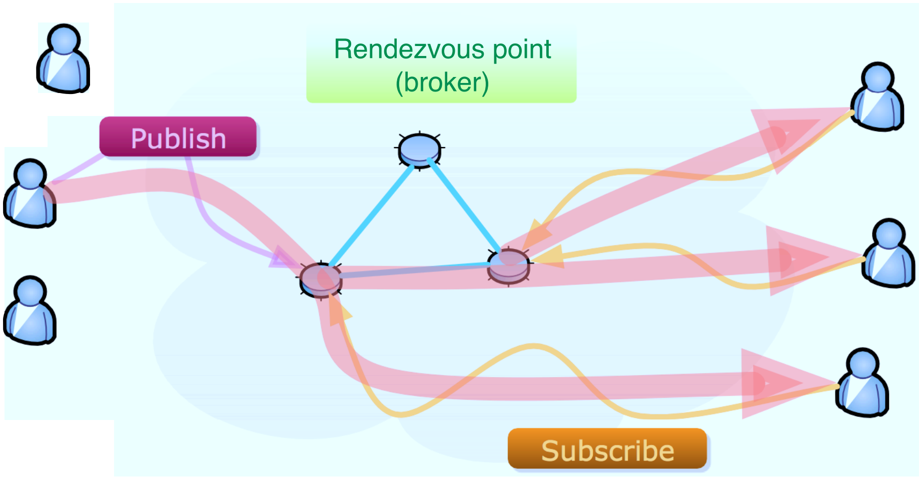


Figure 1.6: Basic concept of publish/subscribe model

have expressed. As shown in Figure 1.6, pub/sub mainly consists of *publisher*, *subscriber*, and *rendezvous point (a.k.a. broker)*. The subscriber sends out a *subscription* message for some information which it has an *interest* for. When the *publication* of the publisher matches this subscription's condition, the information is sent to the subscriber.

The advantage of this interaction based on events is that the publisher and the subscriber do not have to care about their counterparts and do not have to communicate at the same time, since *rendezvous point* connects the paths between the publisher and the subscriber [25]. In addition, when the messages are being published and processed by each side, it does not affect the main flow control [26].

The two most widely used subscription models of pub/sub are content-based and topic-based. In content-based pub/sub, the interest for events are expressed in a combination of multiple meta-data conditions such as, `name = osaka, price < 100, 50 < data size < 100`. Rich expression means higher granularity but it also means higher cost since the rendezvous points have to compare every conditions for each interest and the data. On the other hand, in topic-based pub/sub, events are published and subscribed by topics or keywords where the topics are similar to the notion of groups. Topic-based pub/sub is much simpler to implement than the content-based pub/sub but the subscribers result in receiving more information than they intended since a topic name may indicate a broad object. However, this can be solved by designating a hierarchical name which can give more granularity. This thesis targets topic-based pub/sub to simplify the discussion and furthermore it is assumed that the routers are the rendezvous points which simultaneously caches the published content as the routers forward the content. These routers also become publishers of the content for the future subscriptions.

The communication style of pub/sub also shows the need for paradigm shift in the router architecture. That is, it is only natural that network layer devices have to support the change in the network architecture from the host-centric to resource-centric.

1.3 Outline of the Thesis

This thesis begins by mentioning the related work in Chapter 2. Regarding the main proposals, the characteristic of TCAM is explored first in Chapter 3 by proposing a range matching device with prefix expansion algorithm to store ACL. Chapters 4 and 5 focus on name-based routing method and its evaluation on network and hardware requirements. Finally, this thesis is concluded in Chapter 6.

A New TCAM Architecture for Managing ACL in Routers [27–30]

Chapter 3 presents the characteristics of Ternary Content Addressable Memory (TCAM) which is a special type of memory used in routers. The packets are forwarded by referring to the rules in the forwarding table used in the current Internet, whereas the packets are classified by referring to the rules in the access control list. The memory type used for storing these forwarding tables and the access control list to achieve a high-speed packet forwarding and classification is TCAM. However, TCAM uses more transistors than random access memory, resulting in a high power consumption and a high production cost. Therefore, it is necessary to reduce the entries written in the TCAM to minimize the utilized transistors. The proposal presents a new TCAM architecture by using the range matching devices integrated within the TCAM's control logic with an optimized prefix expansion algorithm. The proposed method reduces the number of entries required to express the access control list rules, especially when specifying port ranges. With less than ten range matching devices, the total number of the entries required to store the port ranges in the TCAM can be reduced to approximately 50% with less than 0.3% increase of the manufacturing cost.

Resource Name-based Routing in the Network Layer [31–36]

The evaluation result from Chapter 3 shows that storing the data in the TCAM requires a careful consideration of its characteristic. In addition to `0` and `1`, TCAM uses an arbitrary don't care value, '$\star$' which enables a faster searching speed but leads to a higher energy consumption compared to the general purpose memories. In other words, it is crucial to effectively use the $\star$ value to better utilize TCAM in storing the access control list. This confirmation is used in Chapter 4 to investigate whether the routers are capable of storing the name information of each named node, therefore showing the feasibility of the proposed name-based routing. The main contribution of this part is on proposing how to distribute a large sized database of the name information among the routers. Using approximately 700 million existing fully qualified domain names, the evaluation result shows that resource name-based routing is feasible even when considering the limitations of the TCAM size in the routers. Utilizing a hash function and longest alphabet matching, inspired by the longest prefix matching, the name database is evenly distributed among all routers. The resulting number of routers is approximately 1% of the currently deployed number of all routers. In addition, the effects of dynamic updates of the database are evaluated by investigating whether the storage location of the names is migrated frequently. Furthermore, the proposed name-based routing has 56% shorter path lengths compared to the lookup system based on the overlay end nodes.

A New Memory Architecture for Realizing Name Lookup Tables in Resource-centric Networks [37–39]

The last part in Chapter 5 of this thesis addresses one of the most challenging issues in realizing a name-based routing in the routers: how to manage the information of the numerous contents and the large-scale number of the users. A new router lookup table structure is proposed to manage such information where the content itself is named and the interest of the users is related to the name of the content. In order to complete the packet forwarding within the network layer, i.e. the search for content which matches the interest, the routers acting as the rendezvous points of a publish/subscribe system should maintain the information of the content names and the users subscribing to the content. Three lookup table structures for the name lookup tables in the routers are proposed, each with a different combination of DRAM, SRAM, and TCAM depending on the usage purpose. The proposed memory architecture is evaluated with the parameters such as memory cost, latency, and utilization using real-life and synthetic databases that have a Zipf distribution. The lookup table that has the lowest manufacturing cost and the lowest latency for storing the given database within a fixed budget is presented. Depending on the lookup table, the read latency ranges from 75 μsec to 45 msec. In addition, it is shown that the chip cost is exponentially reduced from \$997 to \$17 for distributing the names and the subscriber information by increasing the number of the rendezvous points.

Chapter 2

Related Work

This chapter introduces the related work regarding the topics addressed in this thesis.

A New TCAM Architecture for Managing ACL in Routers

It is shown in Section 1.2.1 that TCAMs have high energy consumption and the number of used TCAM entries should be minimized.

Several methods for reducing the number of entries have been suggested. Che et. al [40] proposes *Dynamic Range Encoding Scheme* which encodes a subset of the ranges in the ACL and maps it to unused bits in each entry of the TCAM. Each encoded range affects other ranges which further reduces entries; an entry expansion ratio of 1.23 can be achieved, whereas an average full expansion ratio is approximately 6.20 [15]. However, the proposed algorithm uses TCAM resources when encoding the range and ends up accessing TCAM more frequently when there are more ranges to encode. Also, it needs additional memory for mapping, which can result in an increased search time.

In the work of Dong et. al [41], the subranges in the ACL are processed to be in powers of 2 under the premise that the range ends up being semantically unchanged. The total number of required entries decreases by 50%. However, the total reduction rate depends on the inter-dependent ACL rules whereas the proposal in this thesis focuses on reducing the number of required TCAM entries to store a single port range which is independent of the ranges. Also, the work of Dong et. al [41] approaches the given issue with a software method (trimming, expanding, adding, and merging) whereas the proposal in thesis uses a hardware method (NOT/AND/OR gates) to achieve a faster processing speed.

In the paper of Lakshminarayanan et. al [42], the ranges are encoded using ternary values, but the major difference from the two papers mentioned above is that it places don't care bits at arbitrary places instead of in a prefix form. The result also uses unused bits in each entry. Using 32 bits for this encoding, the suggested algorithm can reduce the necessary entries by up to 50%. However, the authors mention that the optimal encoding method depends on the ACL's content which cannot be known *a priori*.

Cisco's IP routers use a device called Layer-4 Operation Unit (LOU) to write port numbers in ranges in order to make use of the limited TCAM resources [43]. LOU is a hardware device which resides outside of the TCAM and makes it possible for the ranges to be logically compared using the operators *gt (greater than)*, *lt (less than)*, *range*, and *neq (not equal)*, and determines if the input port number matches the specified condition by matching every other field except for the port ranges. Only after a positive result does it begin to search whether logical terms of port ranges also match. By this method, port fields in ranges are only managed under LOU which is independent of the TCAM itself. As a result, the possibility of expressing a rule in a single TCAM entry has the effect of writing more rules to TCAM in the end.

However, the biggest problem with this technique is that LOU is implemented outside of the TCAM circuit. Because the input data format for the LOU is entirely different from TCAM's, the input/output (I/O) band consumption ends up being twice as much, trying to represent the same rate when there is a search from the outside to TCAM. In other words, the amount of I/O pin numbers needed under the same frequency doubles. When the physical size of the device is the same, increasing the I/O pin numbers for external I/O causes the distance between the pins to get smaller and ends up requiring higher-precision wire connections. Also, the signal integrity and the simultaneous signal output (SSO) noise become important factors in the recent high speed hardware devices, which creates restrictions in designing boards in order to reduce the interference noise between the pins. Because of this, it is desirable to have a smaller number of pins if the hardware chip performance should remain the same.

```
Access control list 10       deny     src. = 1,
Access control list 100      permit   src. = 2, tcp port gt. 224 lt. 241
```

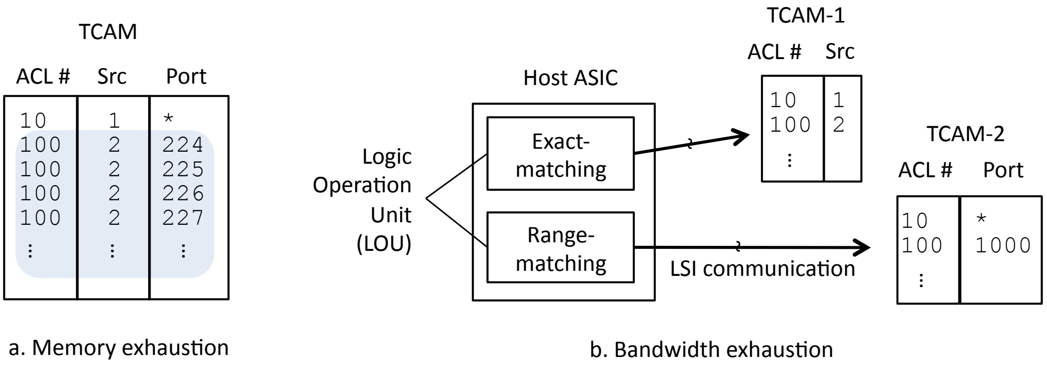


Figure 2.1: Two reasons for TCAM exhaustion

These algorithms consist only of software solutions or the implementation of new devices outside of the TCAM and their actual application is limited. Adding an extension to the TCAM itself and thoroughly considering how the number of entries can be reduced is an unprecedented work.

Another commercial product, NetLogic's range encoding engine (REE) [44], implements range encoding that allows customers to effectively double the efficiency of performing port range inspection. However, it is very difficult to compare this work with NetLogic's REE because (i) no public document is available, therefore it is difficult to find out about the specifications and (ii) it is unclear if 'double the efficiency' means whether the search time halved or the required memory space was saved 50% .

Figure 2.1 shows two reasons for TCAM exhaustion. Figure 2.1(a) is when a port range from `224` to `241` ends up occupying memory space and Figure 2.1(b) is when devices such as LOU exhausts bandwidth by handling exact and range matching process separately, therefore increasing I/O pin numbers.

The proposal in this thesis regarding the ACL is to use a modified TCAM chip with *Range Matching Devices (RMD)* in order to restrain the growth of TCAM entries through prefix expansion. As shown in Figure 2.2, range matching function does not require separate circuits. RMDs are integrated within the TCAM's logic, therefore reducing the cost of extra I/O lines. In addition, it

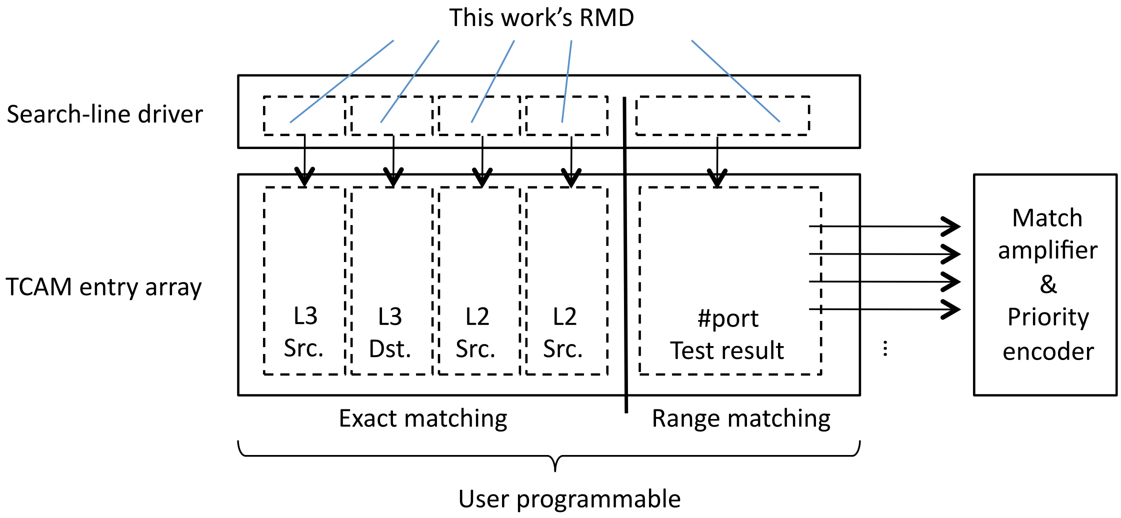


Figure 2.2: Range matching device (RMD)

is possible to maintain the conventional TCAM from the user's point of view and simultaneously permit arbitrary ranges to be stored which reduces hardware costs. In addition, in a situation where the limited number of RMDs is exhausted, the effect of adding logical NOT and AND gates to the TCAM in order to achieve a flexible combination of range expressions is considered. Using this method, it is shown that improving ACLs management is possible compared to the current practice of only using logical OR operations.

Resource Name-based Routing in the Network Layer

This thesis shares a fundamental research background and the motivation with Koponen et al. [20]'s proposal Data Oriented Network Architecture (DONA), which involves a clean-slate redesigning of Internet naming and name resolution. The mismatch between Internet's design for one-to-one communication and the usage for data access is causing problems such as difficulties in persistence, availability, and authenticity [20]. Specifically, the paper proposes replacing DNS names with flat, self-certifying names, and replacing DNS name resolution with a name-based anycast primitive that lives above the IP layer. Content in DONA must first be published, or registered, with a tree of trusted resolution handlers (RHs) to enable retrieval. Each resolution handler must maintain a large forwarding table providing the next hop information for every piece of content in the network.

Once the content is located, packets are exchanged with the original requester using the standard IP routing. The fundamental difference of the work in this thesis from DONA is that the nodes which perform routing by name, RHs, are overlay nodes whereas the proposal in this thesis implements the function within the routers in the network layer. However, the work in this thesis is not in a competitive relationship with DONA but rather in a supportive relationship: the RHs' role of resolving name can be better achieved in the network layer, typically in routers instead of in overlay nodes. The routers utilize the information of the underlying physical topology better and can provide TCAM [16], specialized for longest-prefix matching that RHs use for searching the registration table. In this thesis, the required router memory for storing forwarding information in names is evaluated.

In name-based routing protocol (NBRP) proposed by Gritter et al. [45], clients and users desire connectivity not to a particular server or IP address but to some piece of a content specified by name, typically a URL. *Content routers* are extended to support names which act as both conventional IP routers and name servers to provide name-to-next hop mappings. However, names are used only while the connection is being established and not for routing the actual data packets. Once the 'best' server with the desired information is reached using the name-to-next hop mapping information of the content routers, IP address of the content server is sent back to the client which originated the request. In contrast, the proposal in this thesis assumes that the routers can forward packets with names attached by storing the forwarding/routing information within the routers. Furthermore, the proposal in [45] stays at calculating the routing table size by a single DRAM, lacking an evaluation on the size of the routing tables that each router has. When dealing with names typically longer than 32-bit IP address, storage problem is a major issue that has to be addressed. In this thesis, TCAM is utilized for storing routing information to assume a more realistic hardware circumstance.

Another name-based routing scheme is propose by Shue et al. [19] and the performance of their proposal is compared with IP in terms of the creation, search, and update time of the routing tables, as well as the required memory. A major challenge of 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 the hardware requirements. 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. The work in this thesis focus on evaluating the general feasibility of the routing using names instead of IP addresses, for both network and hardware requirements using TCAM.

In Networking Named Content (NNC) proposed by Jacobson et al. [1], named packets arriving on the interface of CCN nodes are matched with the entries in forwarding information base, content store, and pending interest table. Based on the result of the longest prefix match, actions such as forwarding or discarding is performed. One of the main differences between NNC and the work in this thesis is that NNC provides data caching in the forwarding engine whereas this work presume resource name-based routing in high-speed routers that are mainly devoted to forwarding. In addition, the implementation and the deployment of NNC considers overlay whereas the proposal in this thesis provides a network layer service, not a service of a specific application. However, this is not to say the proposal in this thesis contradicts nor is superior to their work. The authors mention that the content names can be easily hashed for efficient lookups rather than using the fast but expensive TCAMs. The search speed can be improved by showing that the resource name-based routing is feasible even when considering the limitations of TCAM size in routers.

A New Memory Architecture for Realizing Name Lookup Tables in Resource-centric Networks

Among the past work that supports the implementation of one-to-many communication in routers is IP multicast [46]. A multicast address is assigned to a group of multiple users for the content to be sent to that address. IP multicast is implemented in the network nodes, and trees are constructed in order to avoid transmitting the same packets multiple times over the same path. Packets with a multicast IP address are duplicated and forwarded by the routers along the path. However, in a circumstance where there are several tens of thousands destinations for the duplicated packet to be sent, the multicast packets are processed in the software instead of in the hardware which slows down the performance. Moreover, the hardware for IP multicast shows a limitation in storing only the source and the multicast group IP address which makes it difficult to store the information of

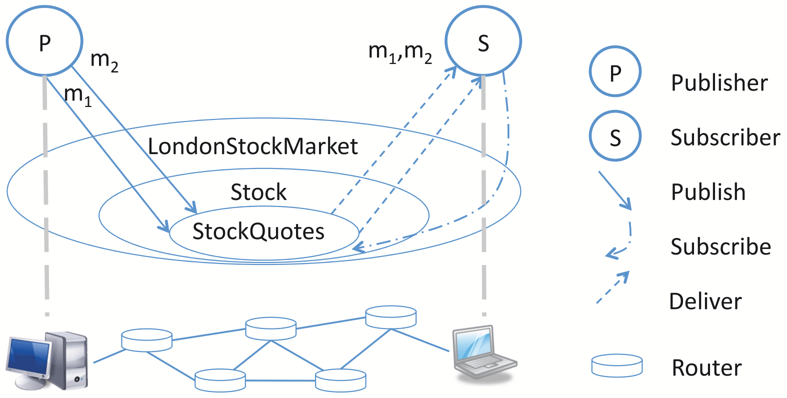


Figure 2.3: Example of topic-based pub/sub interactions

users' specific request for a content.

In this thesis, a hardware architecture is proposed where the network layer routers behave as the brokers in pub/sub that can perform well even when the number of the subscribers increases drastically. Specifically, the proposed router memory architecture for storing tables in topic names and the subscribers is proposed. Topic names are used to express the interest of a content, similar to the subscription model of topic-based pub/sub in Figure 2.3, where the users subscribe to content (messages m_1 and m_2) named 'LondonStockMarket/Stock/StockQuotes'. Meanwhile, the work in this thesis differs from Data-oriented Network Architecture (DONA) [20] since a mechanism for the naming of content and name resolution is not proposed in this thesis. Rather, the fundamental idea is shared with Jacobson et. al [1] which matches the content from the provider and the interest from the consumer by content names. The contents are assumed to be already named and the router memory architecture is evaluated with the parameters such as memory cost, latency, and utilization using the topic name and the subscriber databases that have Zipf distribution.

The objective of this research is to propose a router architecture for the future Internet, not to confront nor compete with application level solutions such as overlays to realize content-centric networks [3] and name-based routing [1, 19, 20, 47]. Implementing resource name-based routing on network layer can replenish the conventional overlays by adding features such as accelerated lookup algorithm and optimized construction of routing path that utilizes the physical topology. Moreover, overlays can be used as a method to aid the migration from the host-centric to resource-centric when the routers have a legacy architecture.

Chapter 3

A New TCAM Architecture for Managing ACL in Routers

3.1 Management of Range Matching Device (RMD)

In this section the logical circuit diagram of the RMD is described, a policy for storing ranges in it is proposed, and the three PE algorithms used in this work is clarified. The mechanism is first described and the proposal is evaluated with updated ACL data.

3.1.1 TCAM Implementing Range Specifications

As previously mentioned in Chapter 2, the number of entries to be stored in TCAM does not decrease much by applying prefix expansion and it is difficult to optimize and update LOU. Numerous studies so far have used an existing TCAM device and implemented additional software or external devices to minimize the number of required entries. The port numbers in ranges is supported by extending the TCAM device. As an extension to the conventional TCAM, the effect of adding a range determining circuit and logical operators between the entries is analyzed using the following methods.

Figure 3.1 shows the interconnection of Figure 3.2 and 3.3. Figure 3.2(a) is a logic circuit diagram of the proposed RMD. It contains *Normal*, *RMD*, *Src/Dst*, *From*, *To*, *Load Enable* and

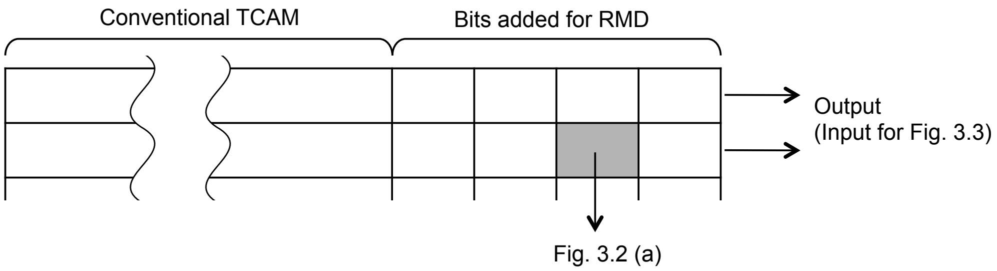
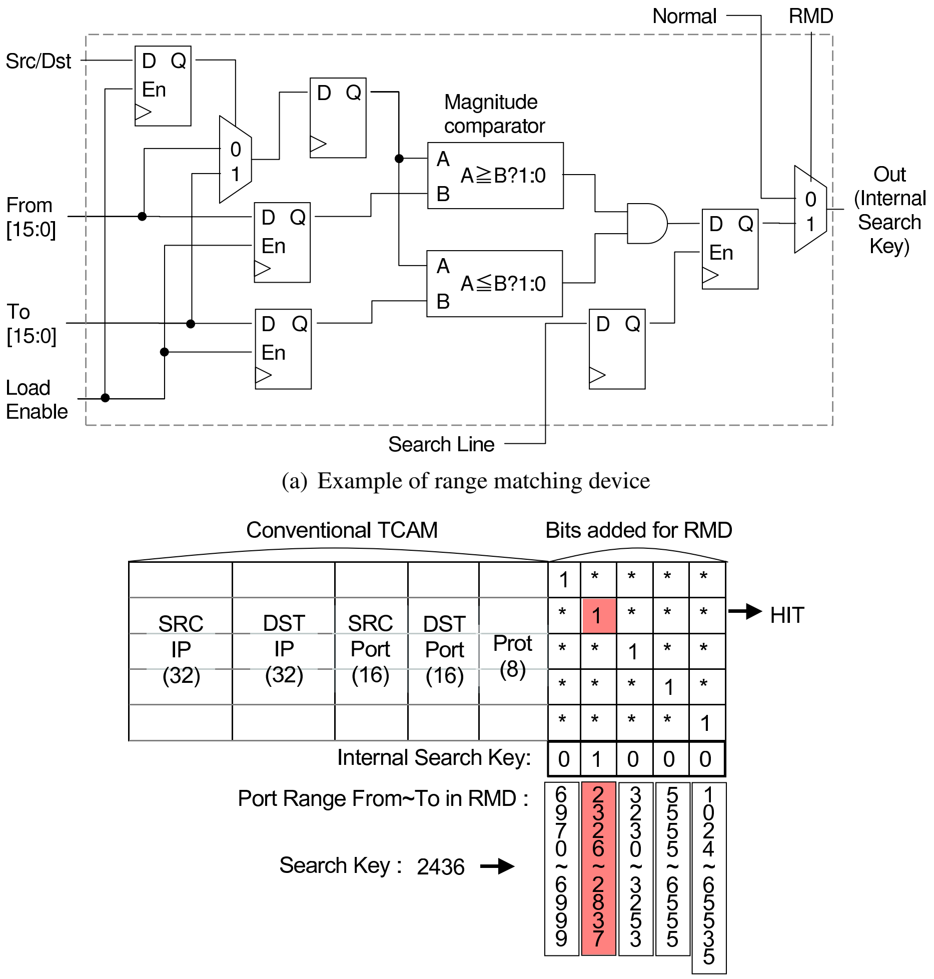


Figure 3.1: Interconnection of Figures 3.2 and 3.3

Search Line as inputs and *Out* as an output. *Normal* and *RMD* are used when an user decides whether to use this device in TCAM as a memory cell that has a data or as a pointer to the RMD. To make an RMD contain a range, there has to be a positive signal to *Load Enable* and *Src/Dst* which determines if this is set for the source or destination port number. At the same time, each *From* and *To* line gets an input of 16 bits that represents the range.

When the RMD is operated as a searching device, either the source or destination port is selected based on the input information above and is compared with the stored range. Only when the input is between *From* and *To* does it return the result of *Out* as 1 (otherwise 0) which becomes the internal search key in Figure 3.2(b). In TCAM, separate bits for the RMD have to be assigned in addition to the other data bits such as source/destination IP address, source/destination port, and the protocol. A single bit is assigned for each RMD. The bit, corresponding to the RMD that contains the desired range, is set to `1` and rest of the bits for other RMDs are set to $\star$. Figure 3.2(b) shows an example of a simple TCAM structure. Let us assume that there are five kinds of ranges already stored in the RMD (`6970-6999`, `2326-2837`, `3230-3253`, `5555-6555`, and `1024-65535`). According to the above RMD bit assigning rule, the RMD part within the TCAM becomes $\star 1 \star\star\star$. When there is a search key port number 2436, it returns the output of `01000` (Internal Search Key) and second entry's result is 1, meaning a match ("HIT" in Figure 3.2(b)) and then this packet will be either denied or permitted, depending on the specified action in the corresponding ACL rule.

RMD is a dedicated LSI with a fixed circuit. The reason why field-programmable gate array (FPGA) was not considered for designing the RMD was because i) FPGA is suited to satisfy multiple functions and may have a useless part, ii) as the demand for TCAM is low for FPGA, it is



(a) Example of range matching device

(b) Bits for range matching device in TCAM

Figure 3.2: Example of range matching device with TCAM

not implemented for FPGA, and iii) when it comes to chip size and speed, a dedicated LSI is more effective than FPGA for a large-scale TCAM. The RMD is designed with the minimum number of gates for the status quo. A possible downsizing is by eliminating the circuits for selecting the src/dst port but the utility may degrade in exchange.

Since TCAM is considered to have performed a successful search when it returns at least one entry that satisfies every field, there can be a situation in which each entry's result performs a logical OR action with the other entries' results. In this thesis, in addition to the conventional PE-OR method, the performance of extending the TCAM capability by adding NOT and AND gates is

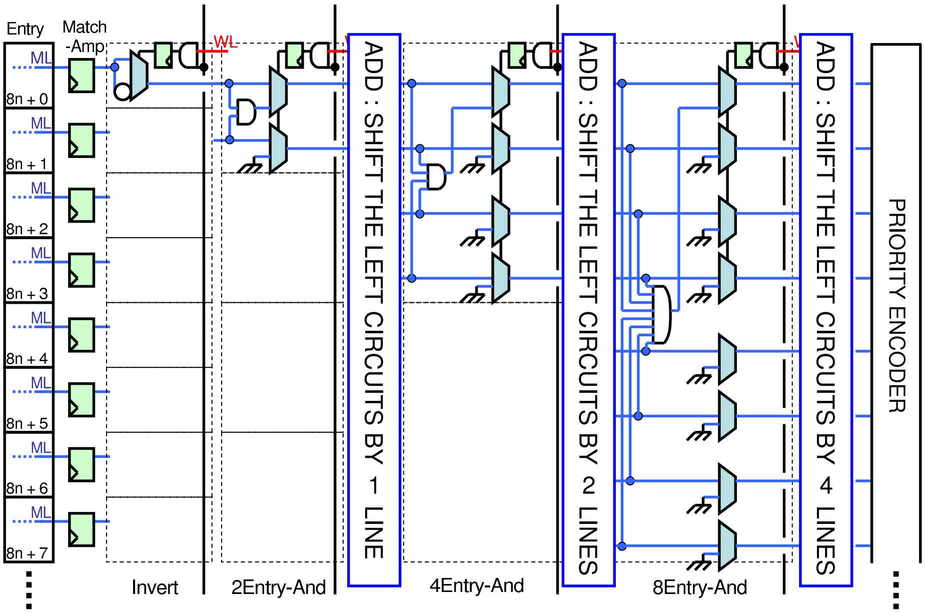

Entry
Match -Amp
WL
ML
8n + 0
8n + 1
8n + 2
8n + 3
8n + 4
8n + 5
8n + 6
8n + 7
Invert
2Entry-And
4Entry-And
8Entry-And
ADD : SHIFT THE LEFT CIRCUITS BY 1 LINE
ADD : SHIFT THE LEFT CIRCUITS BY 2 LINES
ADD : SHIFT THE LEFT CIRCUITS BY 4 LINES
PRIORITY ENCODER

Figure 3.3: Additional NOT/AND operation gates in TCAM array

analyzed. This is because additional gates such as NOT and AND are necessary to fully express PE-MIN. The algorithm of PE-MIN and an example range decomposition is given in Section 3.1.3. Figure 3.3 shows a modified TCAM circuit which has NOT/AND gates. NOT gates are first implemented in the left-most part, followed by AND gates. Since it is impossible to perform a logical AND operation in the hardware only when it is needed, sets of ANDs (2, 4, 8 in this case) are needed, to be already embedded in the hardware. Depending on the number of entries required in representing each subrange that uses AND sets, the smallest AND set needed is selected and any rules exceeding 8 lines can be written in a regular PE-OR form. The benefit of these extra gates is being able to utilize the PE-MIN, which reduces the total number of entries needed to represent ranges.

3.1.2 RMD Policy

RMDs are limited resources and should be used carefully since they can determine the TCAM's performance specification. Writing ranges that are unpopular or ranges that do not consume multiple numbers of entries after the prefix expansion in to RMDs would not decrease the total number of required entries to the maximum degree. Therefore, in order to minimize the TCAM entries, it is desirable to give a higher priority to the range that has the highest entry reduction ratio when being written to the RMD. In this thesis, the weight of each range, which determines the rank to be written in the RMD, is calculated as *(Lines after PE - 1)* multiplied by *(Number of ACLs referring this range)*. In other words, a range has a higher tendency to be written into the RMD when it expands to more lines, or has a higher appearance in the ACL database, than other ranges.

3.1.3 Optimization of the Prefix Expansion (PE)

In this section, an optimized prefix expansion scheme is presented using NOT/AND/OR operations between entries in the TCAM with the proposed hardware structure in Section 3.1.1. Here the three prefix expansion algorithms are explained: PE-OR, PE-Exclusive, and PE-MIN. Figure 3.4 shows a simple representation of the three algorithms with the example range of `5000-6000`.

- PE-OR

  PE-OR is the conventional prefix expansion method that represents the range in units of powers of 2. It shows especially outstanding performance when the range is also expressed as 2^i to $2^{i+1} - 1$ which would consume only a single entry. However, in the worst possible case of the range `16385-65534`, it ends up being 29 lines.

- PE-Exclusive

  PE-Exclusive first finds the minimum range of [2^i, $2^{i+1} - 1$] (x and y in Figure 3.4, respectively) that entirely covers the given range, and expresses the unnecessary parts in combinations of powers of 2. This method is convenient in expressing ranges that are not suitable for PE-OR. For example, the above worst-case range of `16385-65534` can be expressed in only 3 lines: `16384-65535 AND (NOT 16384) AND (NOT 65535).`

- PE-MIN

  PE-MIN is the most optimized entry reduction algorithm of these three. It uses brute force to find the appropriate ranges. Below is the description of the algorithm.

  1. First, do a longest prefix match of $From$ and To in the range $[From, To]$. Select

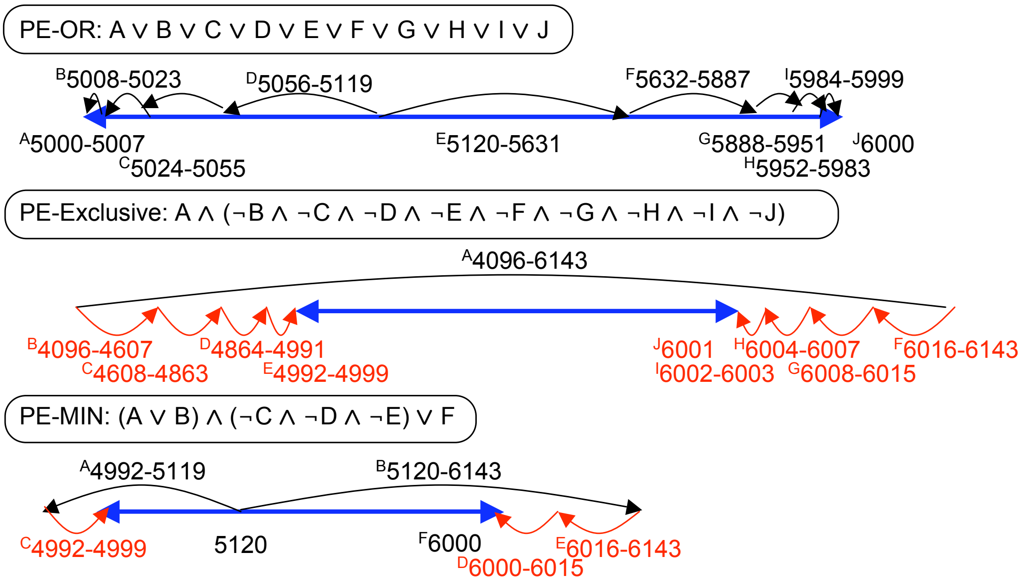


Figure 3.4: Expansion of `5000-6000` using the prefix expansion algorithms

Reference Point b by setting the most significant bit position that differs for the first time. Furthermore, let us divide the range into two parts $[From, b]$ and $[b, To]$

2. Let D be the interval from the b and To. For the interval $[b, To]$, find the largest value of i which satisfies $2^i < D \leq 2^{i+1}$
3. For the two ranges of $[b, To]$ and $[b, b + 2^{i+1}]$, calculate x_l and x_r which are the entries needed for each range, respectively
4. If $x_l < x_r$, add a prefix of $[b, To]$, or else add $[b, b + 2^{i+1}]$
5. For the range $[To, b + 2^{i+1}]$ which exceeds the original given range, repeat steps 2–4 and apply NOT operation
6. Repeat steps 2–4 for the range $[From, b]$

As shown in Figure 3.4, the range `5000-6000` in PE-MIN is expressed as ((`4992-5119`) OR (`5120-6143`)) AND (NOT (`4992-4999`) AND (NOT (`6000-6015`)) AND (NOT (`6016-6143`))) OR (`6000`), where b is 5120 (binary: `1010000000000`) because 5000 is `1001110001000` and 6000 is `1011101110000` in binary, respectively.

Table 3.1: Three ACL sets for evaluation

Date Captured	Apr. 07		Oct. 07		Apr. 08	
# of unique ACL entries	6,440		7,202		7,703	
	Src	Dest	Src	Dest	Src	Dest
# of ranges	3	256	6	325	6	373
# of unique ranges	63		74		82	

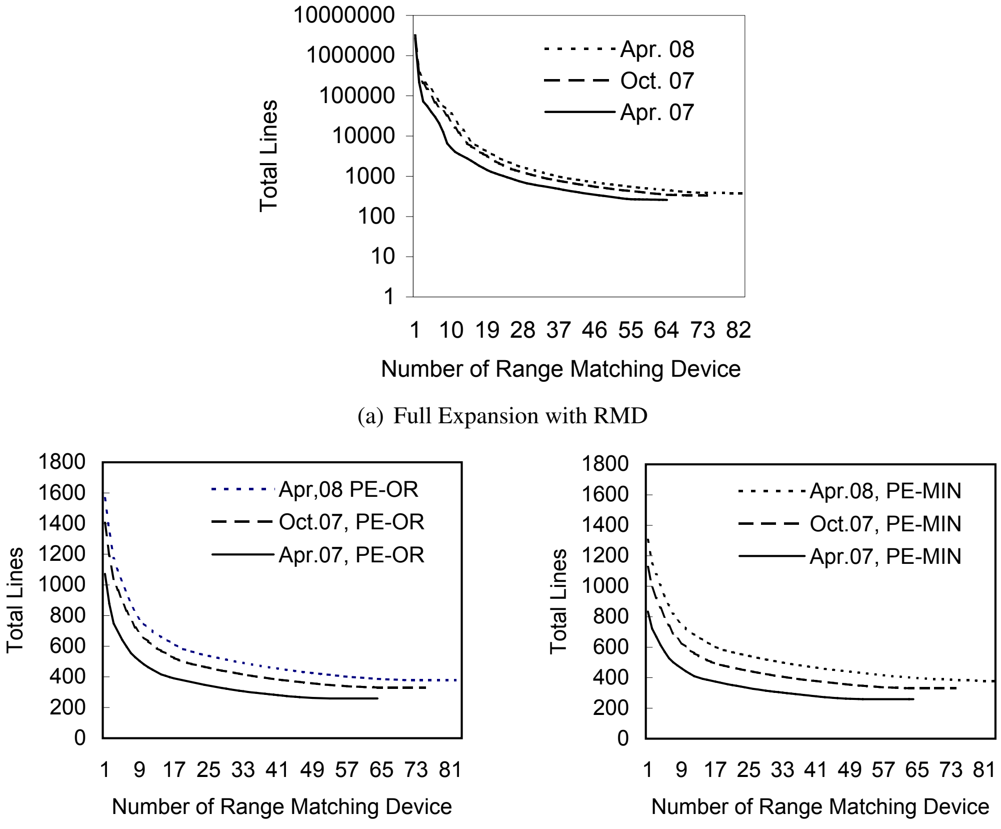
3.2 Simulation Experiments and Discussions

This section analyzes how many entries are required in writing the port numbers expressed in ranges and discusses the result using a real-life database. Table 3.1 shows the information about the three campus-level ACL sets used in this analysis, which were obtained in April and October 2007, and April 2008.

3.2.1 Entry Reduction

Figure 3.5 shows how the required entries can be reduced when RMDs are used. Figure 3.5(a) shows how writing every single number (Full Expansion) that is represented in ranges can consume an enormous amount of TCAM resources. From this figure, it easily seen that an order of a million entries are needed when there are no RMDs. The reason for this phenomenon is that the range `1024-65535` which consisting of registered/dynamic ports is common and it expands to 64,512 lines. By only using one RMD, this range gets written in the memory with the highest priority which results in one tenth of the total required number of lines.

Next, Figure 3.5(b) shows the effect of entry reduction when RMDs are used with prefix expansion. The the weight proposed in Section 3.1.2 is calculated and ranges with larger weight are written first to the RMD. As a result, it is shown that it is possible to drastically reduce the number of entries by putting them into RMDs. When the limited number of RMDs is exhausted, ranges with lower priority are written in the TCAM by using prefix expansion. The reduction rate is especially high when a small number of RMDs is implemented and when PE-OR is used: when there are seven to eight RMDs, the total required ranges to write can be reduced as much as 50% from that of not using any RMDs at all. In addition, five to six RMDs can achieve the same effect as in the PE-MIN case (see Figure 3.5(c)). Also, all three figures in Figure 3.5 show an exponential



(a) Full Expansion with RMD

(b) PE-OR with RMD

(c) PE-MIN with RMD

Figure 3.5: Comparisons of total required entries (w/ or w/o PE combined with RMD)

decrease and after a certain number of RMDs, it is useless to add further RMDs. This is due to the calculation of the weight of each range. Ranges with low priority (i.e., low weight) do not reduce the total number of entries needed even if they are written in the RMD.

Furthermore, when comparing the PE-OR and PE-MIN schemes, the required total entries decrease by about 12% by adding AND/NOT logical gates to the TCAM (614 lines for PE-OR and 695 lines for PE-MIN, both with three RMDs). When PE-OR tries to accomplish the same level as that of PE-MIN, two to three further RMDs are needed.

An algorithmic analysis of the different prefix expansions shows the distribution of expanded lines in each method using the April 2007 database. Figure 3.6 shows how each algorithm expands the range. The total number of lines can be reduced to approximately 10% using the PE-MIN

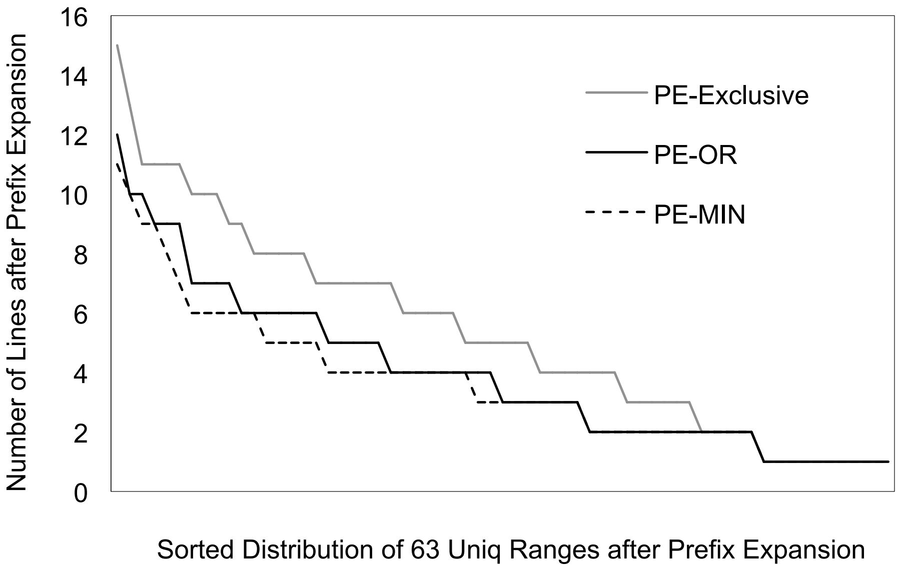
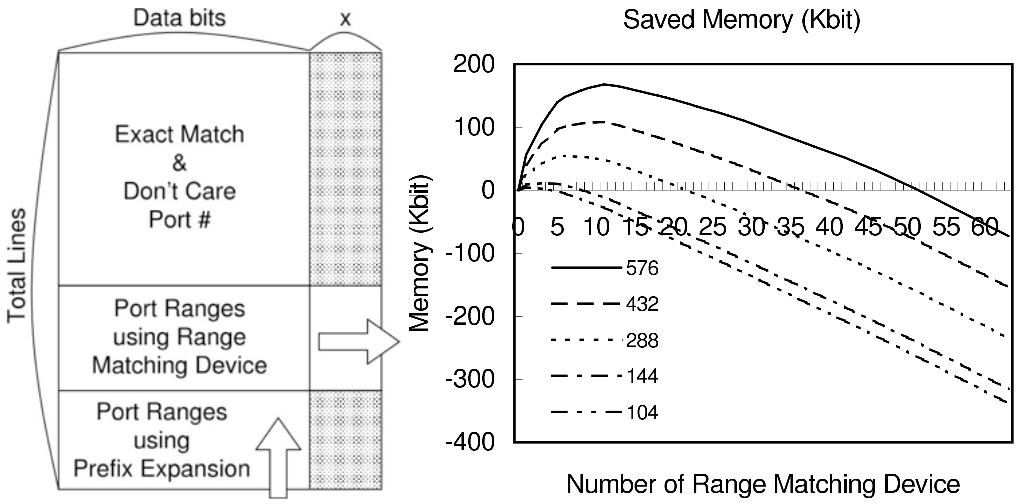


Figure 3.6: Comparisons of prefix expansion algorithms

compared to PE-OR. Comparing the entry reduction itself, PE-MIN uses the least TCAM resources, but it requires built-in logical NOT and AND gates in the TCAM itself.

Then, the question arises: which is a better choice? From the point of view only of the number of required TCAM entries, PE-MIN is the definite choice since it can minimize the number of vertical lines. However, adding NOT/AND gates to fully support PE-MIN might end up increasing the horizontal bits (bits added for RMD, x in Figure 3.7(a)) in the TCAM, which is not a highly desirable choice since determining the optimum AND sets might be different for each ACL. PE-OR might seem like an appealing choice since it does not require any additional logical operation gates. The problem with using only PE-OR, however, is that this method is not well-suited for some intentional worst-case tests, like setting worst-case ranges mentioned in Section 3.1.3 in each source and destination port numbers to create 29 × 29 = 841 lines for a single rule. Unless this is the case, using PE-OR might be adequate.

One of the biggest advantages of adding logical NOT/AND gates in addition to minimizing the total entries required in storing the ACL is that it makes it possible for the ‘except’ rules of the ACL to be stored directly in the TCAM, whereas the conventional methods such as Cisco IOS or Juniper



(a) Trade-off in Memory

(b) Saved Memory Space

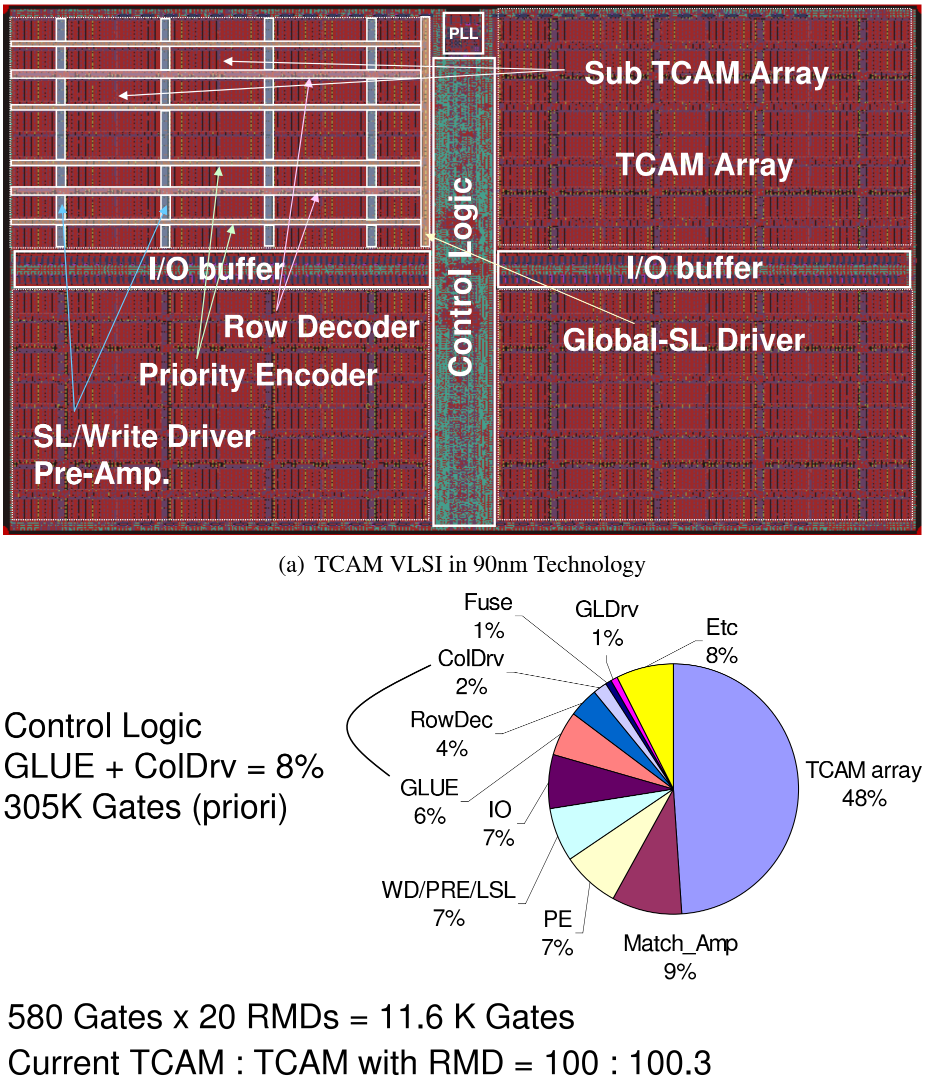
Figure 3.7: Wasted and saved memory when range matching devices are used

Junos prefer to use the software solution.

3.2.2 Overhead Cost Estimation

Finally, the increase in the hardware cost caused by adding RMDs is analyzed. Figure 3.8(a) shows the TCAM hardware structure using 90 nm technology. The ratio of the components in this chip are shown in Figure 3.8(b). Among all of the components, 8% is *Control Logic* which is known to be approximately 305 K gates. Meanwhile, the gate, which is used as a unit for chip area, of a single RMD is 580 gates. When inserting 20 RMDs in TCAM, this results in around 11.6 K gates, and corresponds to 0.3% of the total TCAM gates. Therefore, with the 0.3% increase in manufacturing cost, it is possible to significantly reduce the required number of TCAM lines to write port numbers, which results in more ACL storage space for the limited TCAM resource. Furthermore, the component ratio would stay about roughly the same even if the chip technology advances to 65 nm.

Figure 3.7(a) shows that one additional RMD requires one more TCAM bit to represent the result of the RMD part. Thus RMD reduces the number of vertical lines in TCAM, but at the cost of increasing the horizontal lines. Therefore, it is necessary to consider the trade-off of both factors



(a) TCAM VLSI in 90nm Technology

Control Logic
GLUE + ColDrv = 8%
305K Gates (priori)

580 Gates x 20 RMDs = 11.6 K Gates
Current TCAM : TCAM with RMD = 100 : 100.3

(b) Cost compared to the existing chip

Figure 3.8: Existing TCAM and cost comparison with proposed hardware

in order to decide the optimal number of RMDs for implementation in a TCAM device.

For example, when there is a data with a bit length of x_0 to write in the TCAM without any RMDs, and let y_0 be the required number of entries to write prefix expansion-ed ACL. In this case, the total required bits C_0 in TCAM can be represented as $C_0 = x_0 y_0$. When one RMD is added in a situation where i RMDs are being used, d_{i+1} lines can be reduced. The total necessary bit after adding the RMD in TCAM can be now expressed as the Equation 3.1.

$$C_{i+1} = x_{i+1} y_{i+1} = (x_i + 1)(y_i - d_{i+1}) \tag{3.1}$$

Therefore, the change Δ_{i+1} in total bits by adding an RMD can be expressed as,

$$\Delta_{i+1} = x_{i+1} y_{i+1} - x_i y_i = y_i - (x_i + 1) d_{i+1} \tag{3.2}$$

In order to reduce the total bits after adding the RMD, Δ_{i+1} has to be negative. As a result the following relation is acquired: $d_{i+1} > y_i/(x_i + 1)$.

Figure 3.7(b) shows the total saved memory as RMDs are added. Five kinds of data bits are considered: 104, 144, 288, 432, and 576 (104 bits for the five-tuple and others for a multiplication of 72 which is a common unit of TCAM cells in current technology). This result shows that memory space is saved in the beginning when the RMDs are first added but after a certain number of RMDs, the overhead increases and it is no longer possible to gain any further benefits. Also, when the data bits are shorter, the wasted memory increases faster as the RMDs are added. The ratio of port numbers expressed in ranges in an ACL affects the optimal number of RMDs required in order to make the most of its advantages.

3.3 Summary

For the next generation of TCAM in routers, it will be important to design hardware based on the data that is written in TCAM such as ACLs. The benefit is lower power consumption and better utilization of the hardware resources.

A new TCAM architecture combining optimized prefix expansion and range matching devices is proposed in this chapter to reduce the number of lines required in TCAM to represent port numbers in ranges. By using real-life ACL databases, it is also shown how the proposed architecture can manage the ACL more effectively than conventional methods, reducing the required number of TCAM entries to express the port numbers in ranges by 50%. The significance of reduced entries is large. By 50%, it means the required transistors and the power consumption of the memory array is reduced by half. In addition, less space per a chip means more number of chips in a wafer, lowering the production cost.

Chapter 4

Resource Name-based Routing in the Network Layer

4.1 Name-based Routing

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

4.1.1 Network Architecture and Name Structure

When deciding what *name* should be used to evaluate the feasibility of the name-based routing, a number of standardized names to represent the resource information are first examined.

The examples are the common language equipment identifier (CLEI) [48], the extensible resource identifier (XRI) [49], the life science identifiers (LSID) [50], and the digital object identifier (DOI) [51]. A CLEI identifies a communication device and is globally unique. The ID's ten alphanumeric characters represent the device's technology, type, function, and manufacturer, as well as provide complementary data. The XRI is independent of domain, location, and application (syntax (1) in Table 4.1). The LSID identifies biologically significant resources using syntax (2) in Table 4.1, and the DOI identifies content objects in a digital environment using syntax (3) in Table 4.1 where the prefix and the suffix identify the registrant and the single object, respectively.

Table 4.1: Syntax of standardized names

```
(1) xri://authority/path?query#fragment
(2) urn:lsid:authority.org:namespace:object:revision
(3) prefix/suffix
(4) scheme name:hierarchical part?query#fragment
```

These uniform resource identifiers (URIs) [52] are commonly used to identify abstract and physical resources and are compatible with syntax (4) in Table 4.1. Most of the naming schemes have a hierarchical structure, i.e., a tree structure containing a number of domains each of which having names and/or lower-level domains. Particularly, the authority part in the names in Table 4.1 show the hierarchical structure, ideally suited to a more distributed registration scheme [53].

FQDN is used as an example of names that constructs forwarding/routing tables in routers in this chapter with the following reasons: (i) currently there are approximately 700 million FQDNs which is a sufficient number for evaluating the feasibility and demonstrating the solutions to the questions regarding the **network scalability** and the **storage constraints** in Section 1.2.2 and (ii) FQDN has a hierarchical structure which is a common feature of standardized names in Table 4.1. Therefore, it is believed when the feasibility of FQDN-based routing is confirmed, generalizing it to the resource name-based routing is possible.

The proposed system has a hierarchical network architecture in order to have the local information routed in a closed network instead of being transmitted over a widespread area. In addition, it is possible to take an advantage of one specific characteristic of FQDNs that 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 the extra 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 large number of nodes have to be handled by each node, which makes the routing table of each node very large. In the research on hierarchical routing for P2P applications [54, 55], two-level structures are commonly used. In addition, as the number of hierarchy levels increases the reduction in the routing table size is most effective when the structure has two or three levels [56]. Therefore, the maximum number of levels of the hierarchy in the proposed system here is chosen

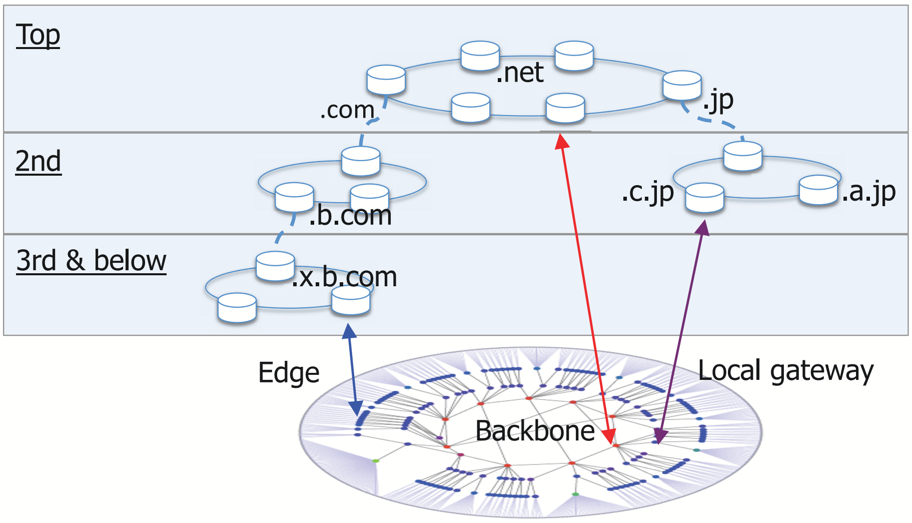


Figure 4.1: Virtual topology for storing hierarchical names and physical network topology

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 router in the highest level can handle all of the routing information within a certain TLD, then a single level is sufficient for that particular TLD. However, this number of hierarchy can be decreased or increased when necessary. The hierarchical topology used in this paper and the network topology inspired by the Abilene network [57] are shown in Figure 4.1.

The Abilene network is known to have similar characteristics to the real Internet topology [58]. It groups routers into three levels from the center outwards to the edge: backbone, local gateway, and edge routers. The architecture in this chapter also groups routers into three levels: the routers managing TLDs, the routers managing 2nd-level domain names, and the routers managing 3rd-level domain names. This hierarchical topology is mapped to the Abilene-inspired topology correspondingly. Since there are fewer than 300 kinds of TLDs, it is possible to statically configure which router to manage what TLD. For the local gateway and the edge routers, dynamic configuration is used to store the routing information, which is explained in detail in the next section. In this chapter, the difference of the hierarchical grouping cases between gTLDs (global TLDs, such as `.com` and `.net`) and ccTLDs (country code TLDs, such as `.us` and `.jp`) is ignored. That is, although the current domain name scheme puts `foo` and `co` of `.foo.com` and `.co.jp` in a different level, these are considered as the same since the exact hierarchical ordering of the FQDNs is not a main object in this chapter.

4.1.2 Routing Protocol

The resource name-based routing differs from the IP routing in terms of the registration because of the ID/locator separation. As IP serves as both an ID and an locator, it simplifies the aggregation of the downstream traffic because the IP assignment is the same in the hierarchical and in the Internet topology. However, names are not always location dependent which makes a separate registration process necessary. A node that intends to register its 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 name will be registered. Each router has a limited amount of storage capacity to handle the routing information. In order to ensure that there is always a enough room for the routing table updates, a threshold value is set. 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 the existing names is 75% of the router's storage capacity. The remaining 25% is reserved for the routing table updates. When a new FQDN is registered, the utilization ratio of the router is checked. If it is below the threshold, the 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 the new 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 created.

A brief description on the type and exchanged time of the messages is mentioned below.

1. Name registration: When registering new FQDNs to the forwarding/routing table of the routers

2. Name update: When there are changes in the routers where the FQDNs are registered

3. Name deletion (unregistration): When deleting FQDNs from the forwarding/routing table of

the routers

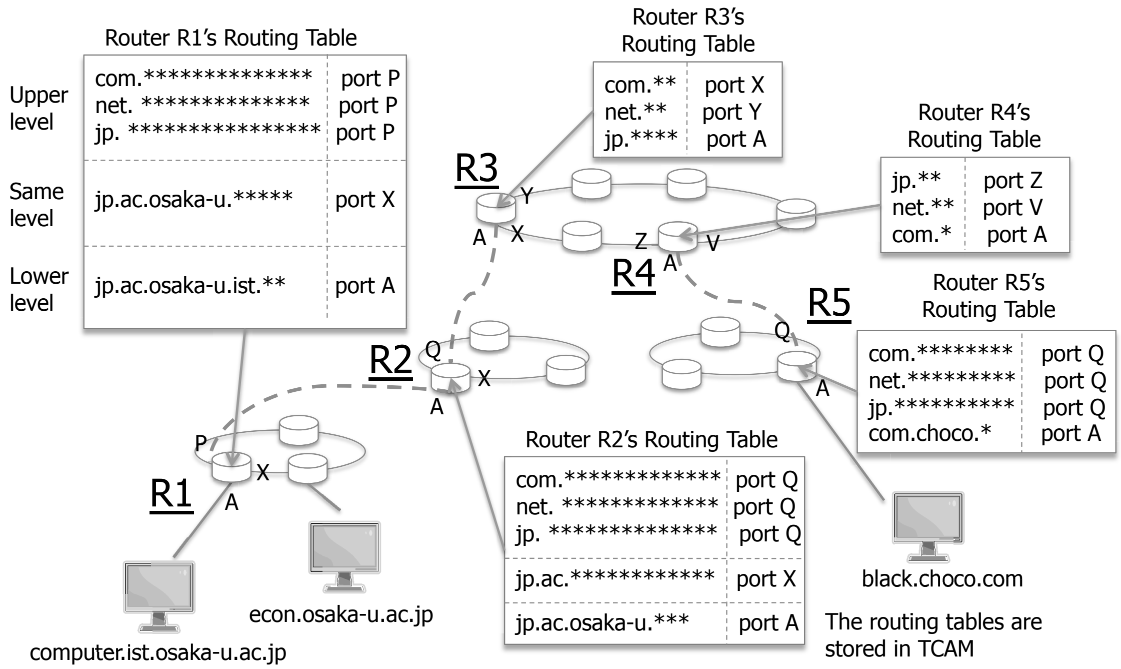
4. Forwarding information base (FIB) bulk transfer: When a router's threshold of number of the entries have been reached. The entries are split and transferred to the new router

5. Router registration: When the new routers are registered. These new routers in the network have a blank list

6. Router deletion (unregistration): When routers are removed from the network. FIB bulk transfer message is required beforehand

Within an autonomous system (AS) [59], one or more interior gateway protocols are used. An exterior gateway protocol such as BGP-4 [60] is used to route packets between the ASs. In the resource name-based 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 name registration. In the case of IP, routers exchange information about the reachable subnets in the network, such as `192.168.0.0/16`, which means hosts from `192.168.0.0` to `192.168.255.255` can be reached. A similar behavior can be achieved in the resource name-based routing by exchanging information such as `*.osaka-u.ac.jp`, which means that all names having `osaka-u.ac.jp` as their suffix can be reached. The exchange of the routing tables is achieved by extending the BGP-4 by adding the two following functions as proposed by Sato et al. [61], (i) support of variable-length addresses with the namespace and (ii) request/response message for the search method.

The 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 Figure 4.2. When the packets are forwarded from `computer.ist.osaka-u.ac.jp` to `black.choco.com`, the former sends packets to the router R1. This router's routing table consists of three parts for names in the upper level, the same level, and the lower level. Since the packets are destined for a higher level than the router R1, they are forwarded to port P. Router R2 goes through a similar process and sends the



Router R1's Routing Table

Upper level	com.**************	port P
	net. **************	port P
	jp. ****************	port P
Same level	jp.ac.osaka-u.*****	port X
Lower level	jp.ac.osaka-u.ist.**	port A

Router R2's Routing Table

com.*************	port Q
net. *************	port Q
jp. **************	port Q
jp.ac.************	port X
jp.ac.osaka-u.***	port A

Router R3's Routing Table

com.**	port X
net.**	port Y
jp.****	port A

Router R4's Routing Table

jp.**	port Z
net.**	port V
com.*	port A

Router R5's Routing Table

com.********	port Q
net.*********	port Q
jp.**********	port Q
com.choco.*	port A

R3
Y
A
X
Z
V
R4
A
R5
Q
A
R2
Q
X
A
P
R1
X
A
computer.ist.osaka-u.ac.jp
econ.osaka-u.ac.jp
black.choco.com
The routing tables are stored in TCAM

Figure 4.2: Packet forwarding

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 a 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 name. The process is also done by extending BGP-4, using the character information instead of the IP address.

4.1.3 Distribution of Names in Routers

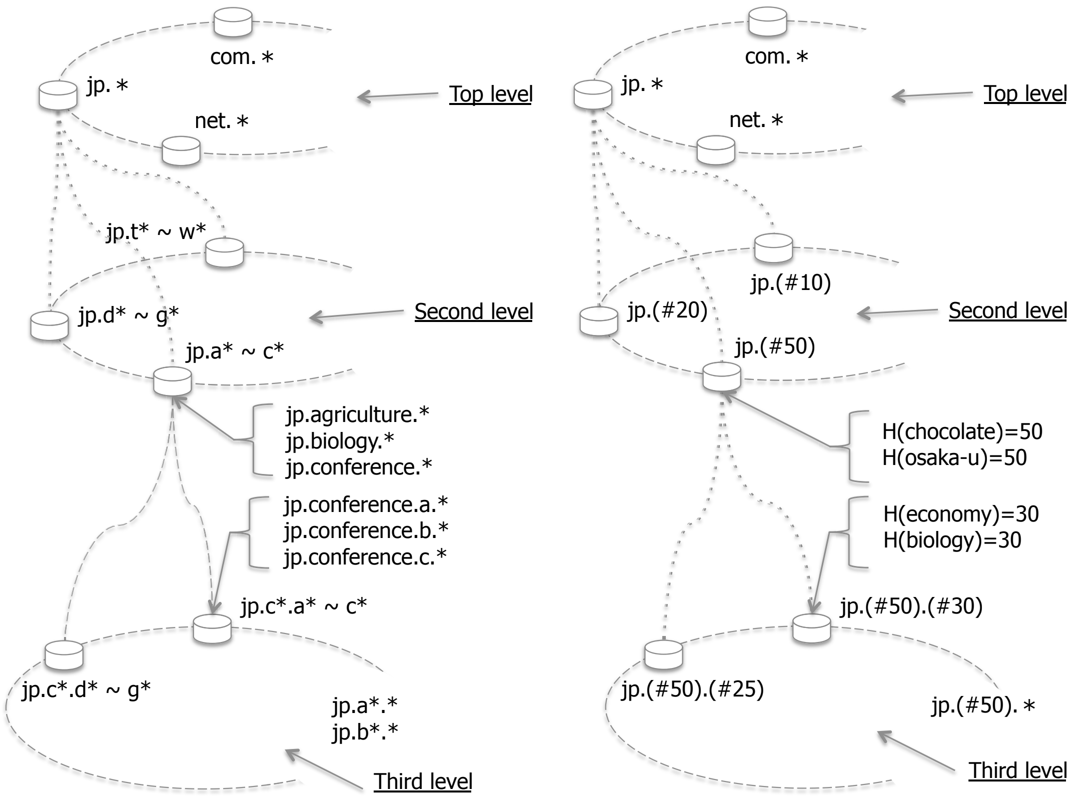
Compared with the IP forwarding/routing tables, tables in names require large storage capacity because of the variable length of names. Therefore, it is important to balance the routing information among the multiple routers. In this section, three algorithms for equally distributing the routing information in FQDN format is proposed which will be used in Section 4.2. FQDNs described in this thesis are sometimes written in the reverse order, i.e. `com.foo.a*` instead of `a*.foo.com` to better express all FQDNs having `a` as the first character of the third level domain name.

Pure Hash

The simplest of the three proposed distribution methods is a 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 an equal status, the total number of the routers required for the domain name routing is obtained by (the number of the domain names) × (the number of bits needed per entry) / (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, the domain names that end with ccTLDs lose the 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 [62] where the number of physical hops taken by the protocol to reach the destination node from the source node is much larger than the shortest distance in terms of the physical hops. Therefore, although it is considered ideal in terms of achieving a nearly equal distribution, it is not realistic for the implementation.

Hierarchical Longest Alphabet Match (HLAM)

The example of the proposed algorithms are shown in Figure 4.3. Hierarchical longest alphabet match (HLAM) is inspired by the longest prefix match and it expresses names with the ASCII code. The characters used in the FQDNs are 26 case insensitive letters of the English alphabet and the ten numbers from 0 to 9 plus hyphens and dots [63]. When alphabets have common bits, they can be stored in the same router. For example, `jp.a*`, `jp.b*`, and `jp.c*` can be stored in the same router since `a`, `b`, and `c`'s ASCII codes are `1100001`, `1100010`, and `1100011`, respectively, which can be expressed as `11000**` in ternary logic. As shown in Figure 4.3(a), the names are grouped using the first character in each level. In the 'second level' of Figure 4.3(a), `jp.a*~c*` represents that all FQDNs with the second level domain names starting with `a`, `b`, and `c` are grouped together. This process is recursively done for each level except for the lowest level in the hierarchy where the grouping is done by the first character of the third-level domain names. However, as explained in Section 4.1.1, this number of levels can be adjusted when necessary and *n*th level



(a) Hierarchical Longest Alphabet Match (HLAM)

(b) Hybrid Distribution (HD)

Figure 4.3: Examples of HLAM and HD

domain name should be used for the grouping in the corresponding level. The advantage of this method is the ability to utilize the don't care bit of the TCAM. In addition, the aggregation of the multiple FQDNs with the same suffix is possible, which reduces the occupied memory space.

The pseudocode for this calculation is shown in Algorithm 1. 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 ($router_c$). An entry is assumed to consume 180 bits, where the 160 bits are the hashed value plus 20 bits of output port plus 4 parity bits. Again a router is defined to have 10 TCAMs and each TCAM has a capacity of 18 Mbits. Threshold t is explained in Section 4.1.2.

If e multiplied by $entry_l$ is less than $router_c$, then the total number of routers required in this

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**
```

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 is ignored for the moment. Here, u domain names are written dynamically by referring to the ASCII table. The basic idea is that the names are grouped in an 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. The names are first divided into two groups: a group with names that start with digits and the one that starts with alphabets. If the size is still too large, the alphabets 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 the 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}.

Hybrid Distribution (HD)

The third algorithm used for distributing the name database is hybrid distribution (HD). The name ‘hybrid’ comes from combining the pure hash-based distribution and the hierarchical structure. HD distinguishes FQDNs by their TLD first and applies a hash function to the 2nd level and below. In other words, even if the second level domain names are different, those FQDNs will be placed in a same router if the hash values of that second level domain names are identical. For example, as shown in Figure 4.3(b) when the hash value of `chocolate` (part of `chocolate.co.jp`) and `osaka-u` (part of `osaka-u.ac.jp`) is the same, the FQDN is stored in the same router. By using hash values, FQDN entries are more equally distributed among the routers, therefore requiring fewer routers than HLAM.

The pseudocode is shown in Algorithm 2.

Algorithm 2 Number of routers for Hybrid distribution

```
(Refer to Algorithm 1 for the variables)
if e × $entry_l \leq router_c$ then
    RETURN $n_{L2} \leftarrow 1, n_{L3} \leftarrow 0$
else if u × $entry_l \leq router_c$ 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 Algorithm 1 up until the list of u does not fit into a single router.

n_{L2} routers required in the 2nd level in the hierarchy is obtained 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 the groups that have a nearly equal entries independent of the alphabets are obtained. 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 the groups.

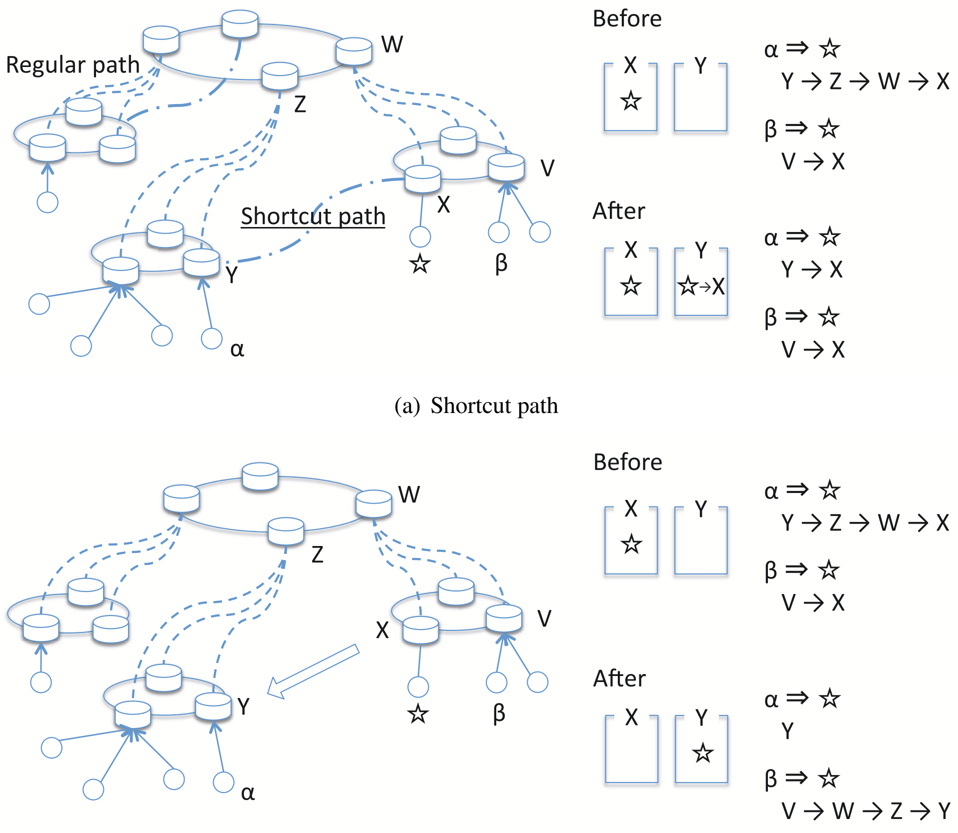
4.1.4 Reconstruction of the Routing Tables

This section discusses the scenarios for quickly responding to the queries regarding the routing information for highly accessed FQDNs. The routing table's *initial state* represents the state of the routing tables immediately after the database distribution using the HLAM or HD algorithm and before routing table reconstruction. Here, the *request frequency* means the degree of requests from the source and the *access frequency* means the degree of accesses that the destination receives. The gateway router of the source or the routers along the path to the object receive 'requests,' and the router with the object receives 'accesses.'

Scenario 1: Shortcut Path

One scenario for quickly answering the queries regarding the routing information for highly accessed FQDNs is to add entries to the routing table. These entries are shortcuts to the routers with high access frequencies (i.e., the routers that have routing information for the destination FQDN) from a router that has a high request frequency. As shown in Figure 4.4(a), router Y adds an entry to reach router X directly depending on the number of requests to the node/resource '$\star$'. After the shortcut is established, other routers in the same ring as router Y can use it to effectively reach router X. Therefore, the other nodes in the ring are notified about Y → X.

Adding entries for shortcuts makes it unnecessary to traverse the upper layers in the hierarchy.



(a) Shortcut path

(b) Entry migration

Figure 4.4: Shortcut path and entry migration

In addition, the routers can receive information about network changes as soft-state updates and reselect neighboring nodes, which increases robustness [64]. However, it is not always possible to create shortcut paths through different Internet service providers.

Scenario 2: Entry Migration

Another scenario for quickly answering the queries regarding routing information for highly accessed FQDNs is to delete the routing information for entries that have a high access frequency from the initial state router and to add the entries that have a high request frequency to the router. As shown in Figure 4.4(b), with this ‘migrating entry’ scenario, the routing information for node/resource ‘⋆’ is moved from router X to router Y when the number of requests exceeds a threshold. If it is moved, notification is sent to the routers in the same ring as X, to upper layer router W, to router Y's upper layer router (Z), and to the routers in the same ring as Y.

Migrating an entry from a router with a high access frequency to one with a high request frequency enables the routing information for the entry to be deleted from the initial router. This results in a better utilization of the router memories. In addition, routing information is stored in a router with a high request frequency or in one close to where there are many requests, which speeds up query response. However, overhead is increased because other routers have to be notified of the change.

Scenario 3: Combination

The third scenario for quickly answering queries regarding routing information for highly accessed FQDNs is a combination of scenarios 1 and 2. The algorithm uses two parameters.

1. $P_a(i)$ is the number of accesses to an item P. It is the sum of the number of accesses from the router i that has the routing information for P and from the routers that have the same TLD as i.
2. $P_r(j)$ is the number of requests to an item P. It is the sum of the number of requests from the router j that is on the path from the source to P and from the routers that have the same TLD of j.

The algorithm has three steps.

1. If $P_r(j) > P_a(i)$, add a shortcut path from j to i
2. If $P_r(j) > P_a(i)$ over a time period t, move item P to j
3. Repeat (1) and (2) depending on the request/access frequency

4.2 Simulation Experiments and Discussions

In this section, the routers' memory size requirement and the effect of dynamic updates of database entries are evaluated using FQDN database from April, July, October 2008, and January 2009

Table 4.2: Example of database used in the evaluation

`207.55.68.184`	`foo.bar.baz`
`59.124.12.148`	`foo1.bar1.baz`
`174.86.13.189`	`foo2.bar2.baz`
`59.124.12.146`	`foo3.bar3.baz`
`59.124.12.147`	`foo4.bar4.baz`

obtained from ISC [7].

4.2.1 Memory Requirement

The database entries are distributed among the routers based on the two algorithms in Section 4.1.3, hierarchical longest alphabet match (HLAM) and hybrid distribution (HD). The ISC database is formatted as shown in Table 4.2, where an IP address and the corresponding FQDN are written in each entry. A router is assumed to have ten 18 Mbit TCAMs and that each entry consumes 180 bits. This is because a 18 Mbit TCAM is a common unit of TCAM and the 180 bits correspond to the popular SHA-1 hash value.

The number of routers required using HLAM is 1,396, assuming that one entry consumes 180 bits. A 7-bit ASCII code is sufficient to distinguish the characters used in FQDNs. Furthermore, since there are fewer than 300 TLDs, they can be differentiated using only 9 bits. Whereas it is easy to adjust the bit length in each entry if each entry is hashed, this is not the case in HLAM. To satisfy the static length of 180 bits which is used throughout this chapter to evaluate the required number of the routers, 155 bits must be free for use, excluding the 9 bits reserved for the TLD and the 16 bits for the output port. Assuming that one character consumes 7 bits, approximately 22 characters can be written in a TCAM entry. FQDNs that are shorter than 22 characters are less than 30 % of the database and in order to store 99 % of the FQDNs, the TCAM must be able to store entries having up to 50 characters [7]. Although the search speed decreases when the lookup size increases [16], designing the TCAM to suit longer bit lengths should not impose much of a technological difficulty.

The number of routers required for HD is 952. HLAM and HD require 2.4 and 1.7 times as many routers compared to the pure hash-based distribution mentioned in Section 4.1.3, respectively.

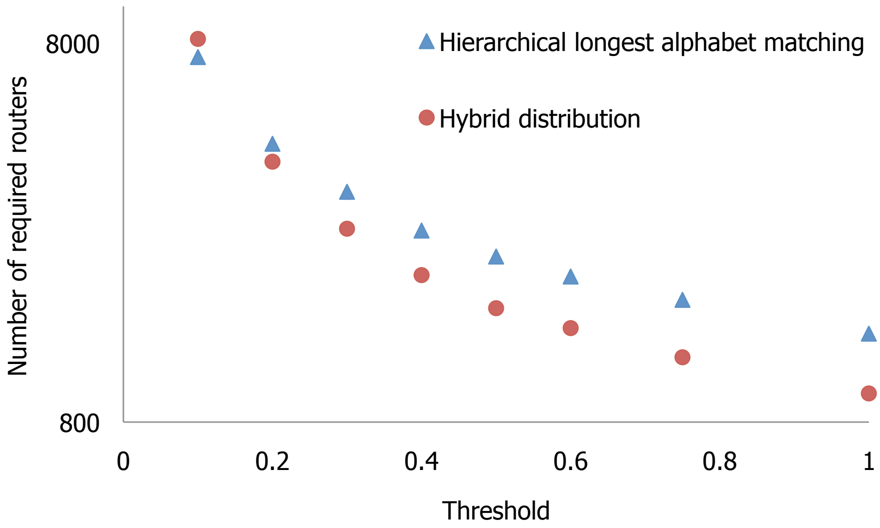
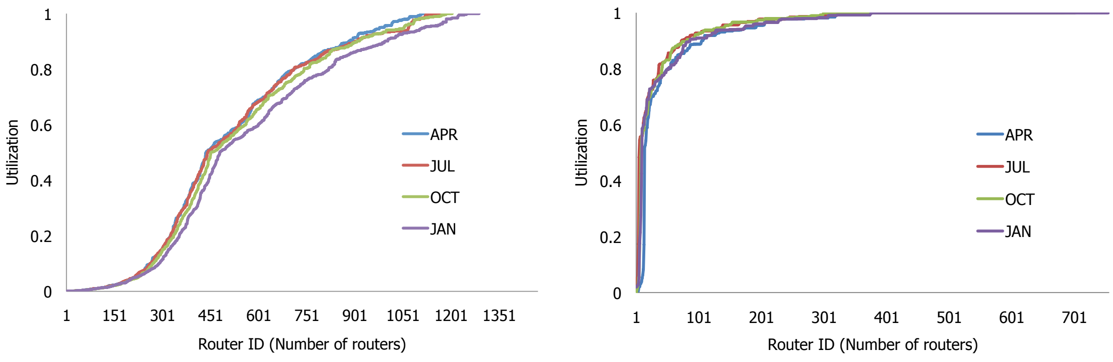


Figure 4.5: Required number of routers for different thresholds

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.

Comparing these two algorithms, it is shown that HD needs less number of routers since it utilizes hashing which leads to more equally distributed database. However, HLAM utilizes the advantage of the TCAM better by effectively using the don't care value and using ranges of the alphabets. If TCAMs can handle distributed hash table (DHT) data, then HD can also be considered to be a feasible method. The number of routers required for the different thresholds in the routers are shown in Figure 4.5. 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 names with a threshold value of 0.2 would require approximately 4,000 routers for both HLAM and HD.

In the work of Yook et. al [65] and Lakhina et. al [66], the work from [67] is used to estimate the number of routers deployed world-wide as approximately 228,260. Therefore, the router numbers required for the methods proposed in this paper are realistic values that indicate that the proposed resource name-based routing architecture is feasible. Achieving the resource name-based



(a) HLAM (b) HD

Figure 4.6: Number of required routers

routing with TCAM has been considered unrealistic in the past because it was presumed that a lot of hardware memories are required. This thesis is the first work known that suggests the feasibility of the resource name-based routing by evaluating and estimating the required TCAM storage capacity.

4.2.2 Effects upon Dynamic Updates of Database

Figure 4.6 shows how the change in the FQDN database affects the total number of required routers. The routers are sorted by the utilization (Equation 4.1) in increasing order on the x axis. Although the graphs for all four databases (April, July, October, and January) are shown, there is not much difference among the curves, implicating the storage requirement of the names in the routers increases in a scalable way.

$$\textit{Utilization} = \frac{\text{Number of entries in a router} \times \text{Bits per entry}}{\text{Memory size}} \tag{4.1}$$

After distributing the routing information to the multiple routers using HLAM or HD, aggregation of the names is used to further save the routers' memory resource. HLAM uses the ASCII code of the TLD's alphabet to aggregate names. For example, ac, ad, ae, aero, af, ag, ai, al, am, an, ao each using a router results in an inefficient memory resource of the eleven routers. When these TLDs are aggregated by the ASCII code, 1100001110$\star\star\star\star$ can represent everything since a is 1100001 and a - o is 110$\star\star\star\star$.

HD classifies FQDNs by the TLDs first and groups the FQDNs by their hash values in the lower level in the hierarchy. Since the algorithm uses hash values, the aggregation method differs from that of the HLAM which shows a limitations of the alphabet. For example, suppose H(`osaka-u.ac.jp`) = 50 (mod x) and H(`chocolate.jp`) = 50 (mod x) (i.e. both FQDNs are in group 50, where x is an arbitrarily value of TLD's size) and H() is a hash function. Although the name is different, both FQDNs can be written in a router which has ID #50.

Aggregating the entries will not increase the number of routing hops when discovering the resource. For HLAM, `1100001110****` in the TCAM as the next hop would match '`ac` (`110001 1100011`)' as well, therefore searching for `ac` does not require extra steps. For HD, if routers can calculate the name's hash value and figure out the next output port, no extra hops is needed. However, the overhead of implementing hashing hardware within the router might be necessary.

In order to claim that using the distribution algorithms shows no drastic change in the topology upon the update of the database, it is necessary to show that the storage point of the entries are not changed so frequently. Tables 4.3 and 4.4 show the change of routers where FQDN entries are written. 'Same' is defined as when the entry exists in both 'before' and 'after' the update and the storage place is the same, where storage place here is the router. 'Different' is defined as when the entry exists in both 'before' and 'after' the update, but the storage place is different.

$$\textit{Transfer ratio} = \frac{\text{Different}}{\text{Same}} \times 100 \tag{4.2}$$

For the three updates, HD's transfer ratio is 4.7 %, 3.3 %, and 4.9 % and HLAM's transfer ratio is 250 %, 222 %, and 218 %, respectively. The reason why HLAM shows a much higher transfer ratio is in the way that HLAM moves on to the next router when distributing the entries. HLAM aligns the FQDN in an alphabetical order and stores the ones starting from `a` in the routers. It refers to the ASCII codes and stops writing when the entries reach a router's threshold and moves on to the next router to store further entries. When there are a lot of new FQDNs in one update, the border line that distinguishes the FQDNs in one router from the next router changes and the transfer ratio is increased. On the contrary for HD, unless there is a sudden increase from the 'before' database, the transfer ratio does not change drastically because the storage place for a FQDN is decided by

Table 4.3: Change of FQDNs' storage location (HD)

	Apr 08 → Jul 08	Jul 08 → Oct 08	Oct 08 → Jan 09
Same	495,106,893	518,627,699	531,823,934
Different	23,253,852	17,010,744	26,056,043

Table 4.4: Change of FQDNs' storage location (HLAM)

	Apr 08 → Jul 08	Jul 08 → Oct 08	Oct 08 → Jan 09
Same	147,836,915	165,916,860	174,961,495
Different	369,471,202	368,211,950	381,208,966

the FQDN's hash value.

4.2.3 Number of Path Length

In this section, the logical and the physical path length of the general overlay and the proposed name-based routing system are compared. Figure 4.7 shows an example of the logical and the physical path that overlay and the proposed name-based routing traverse. In the case of the overlay, the host that wishes to have the destination name resolved to an IP address queries another end node on the overlay ring such as Chord [68]. On the other hand, the proposed name-based routing system looks up the name directly in the routers.

Figure 4.8 shows the path length of the overlay and the proposed name-based routing. The average path length is defined as the required number of hops to traverse the nodes during a lookup operation i.e., finding a node that stores the value associated with the search key. For overlay which utilizes the Chord ring, this value is known to be $\frac{1}{2}\log N$ where N is the number of nodes [68]. In case of the proposed system, path length is the number of hops taken from the source node to the router managing the name of the destination.

The simulation settings are as follows. Forwarding information on names are distributed among the routers for the name-based routing system where the memory capacity of a router is ten 18 Mbit. The number of the end nodes per a router is fixed as ten. For overlay, the names are distributed among the end nodes which are placed on a ring where the memory capacity is assumed as same as the router since the aim is to compare the path length. The number of the routers is fixed as 100.

As the number of the names increase, i.e. the required number of the routers or the end nodes

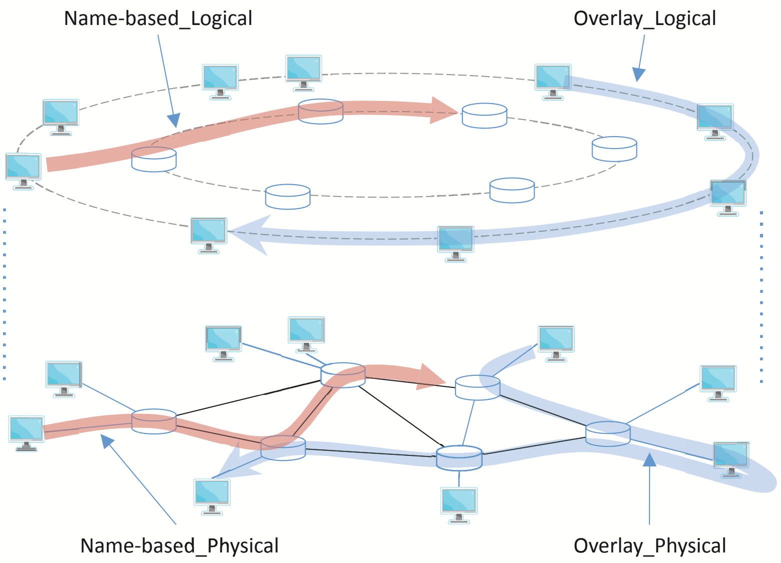


Figure 4.7: Example of traversed path of overlay and proposed name-based routing system

increase, the path length to reach the node having the name information increase with an $O(\log N)$. When approximately 850 million FQDNs (database for July 2011 [7]) are distributed among the routers and the end nodes, the average logical path length for the overlay and the proposed system are 3.3 and 4.3, respectively. However, the underlying physical path length for the overlay and the proposed system are 11.7 and 6.6, respectively which shows the proposed system has 56% lower path length compared to that of the lookup system based on the overlay end nodes.

In addition to the comparison of the logical and the physical path length of the general overlay and the proposed name-based routing system, the static placement of routing tables and the adaptive placement (reconstruction) of routing tables in accordance with the access frequency is also evaluated. The database is distributed using the HLAM and HD algorithms, which are also referred to here as the 'name-based' and 'hash-based' algorithms. To shorten the simulation time, only 1% of the entries (7.3 million) from ISC [7] are used. The size of the TCAM in a router is also reduced, to 1.8 Mbit × 1 TCAM = 1.8 Mbit compared to 18 Mbit × 10 TCAMs = 18 Mbit used in previous parts in the Chapter.

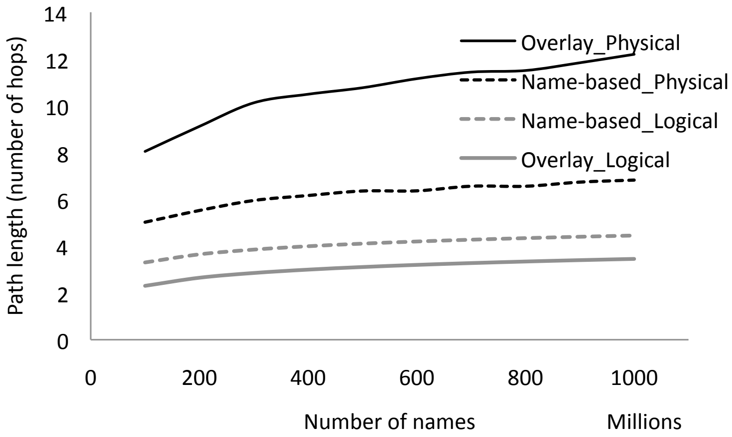


Figure 4.8: Path length of overlay and proposed name-based routing

The static placement, which does not reconstruct the routing tables, is compared with the adaptive placement, which reconstructs the routing tables in accordance with the request/access frequency of the destination FQDN. The evaluation metric is the average number of hops between the source and the destination. Since the average number of hops in a general DHT ring with n nodes is $O(\log n)$, we estimate it as $\log n$. The number of hops between a router in the lower level and one in the upper level is set to 1, on the basis of the single-connection intra-group structure described by Zoels et al. [69]. This is considered to be a desirable structure in a hierarchical DHT system, in which multiple peers are connected to a super peer. Furthermore, it is assumed that the nodes in each lower level make a ring of their own.

There are two rules for selecting the source/destination pairs.

1. Determine the destination first. The destination has a Zipf distribution with the index factor 1 [70]. In other words, of N = 732,740 pairs, the 2nd most popular destination has half the number of accesses as the most popular destination. In addition, it is assumed that gTLDs are accessed more often than ccTLDs. The top three destinations are selected from the pool of gTLDs.

2. The rules for selecting the source depend on whether the destination is a gTLD or a ccTLD. To bias the origin of the source, three popular sources are selected to have a distribution

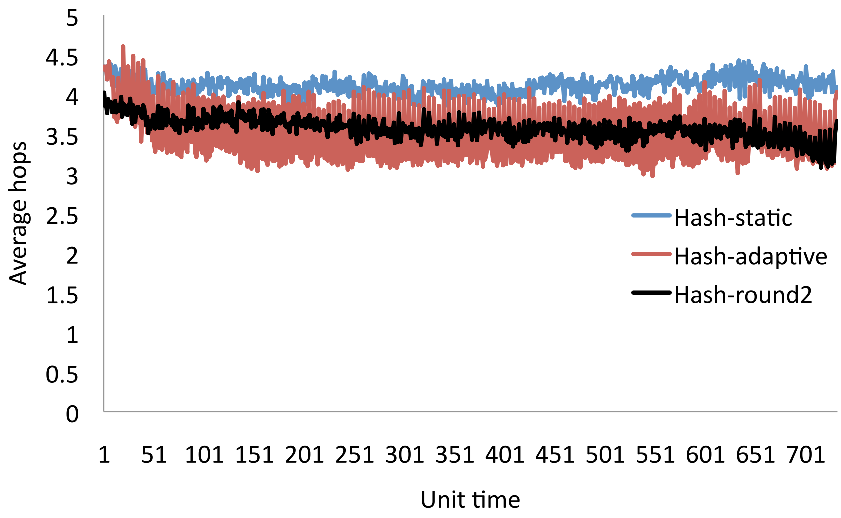


Figure 4.9: Average number of hops (HD)

of 40%, 20%, and 10% when the destination is a gTLD. The remaining 30% are selected randomly. When the destination is a ccTLD, 90% of the sources are selected to have the same TLD to impart a locality characteristic.

The number of hops is calculated for static and adaptive placement using the selected pairs. The condition $P_r(j) > P_a(i)$ is checked after 5 time unit have passed. Here, 1,000 communications (i.e., source requests for destination FQDN) occur in each unit time. Therefore, if $P_r(j) > P_a(i)$ holds after 5,000 communications, a shortcut path is established. Time t is when $P_r(j) > P_a(i)$ holds even after the shortcut path is established. In other words, if there is an attempt to create a shortcut when one already exists, the source and destination pair is considered to be very popular, resulting in entry migration.

Figure 4.9 shows the average number of 1,000 hops in each unit time for hash-based static, adaptive, and adaptive-round 2. 'Adaptive round 2' uses the content of the forwarding tables from the 'adaptive' and the same source-destination pairs are used in the evaluation. An average of approximately 4.1 hops per unit time was maintained with static placement while the average dropped to about 3.4 hops after 730 time units with adaptive placement, reducing the number of hops by approximately 20%. This may seem as a small amount of reduction but shows a potential in developing a mapping algorithm to reach the destination with the shortest path possible.

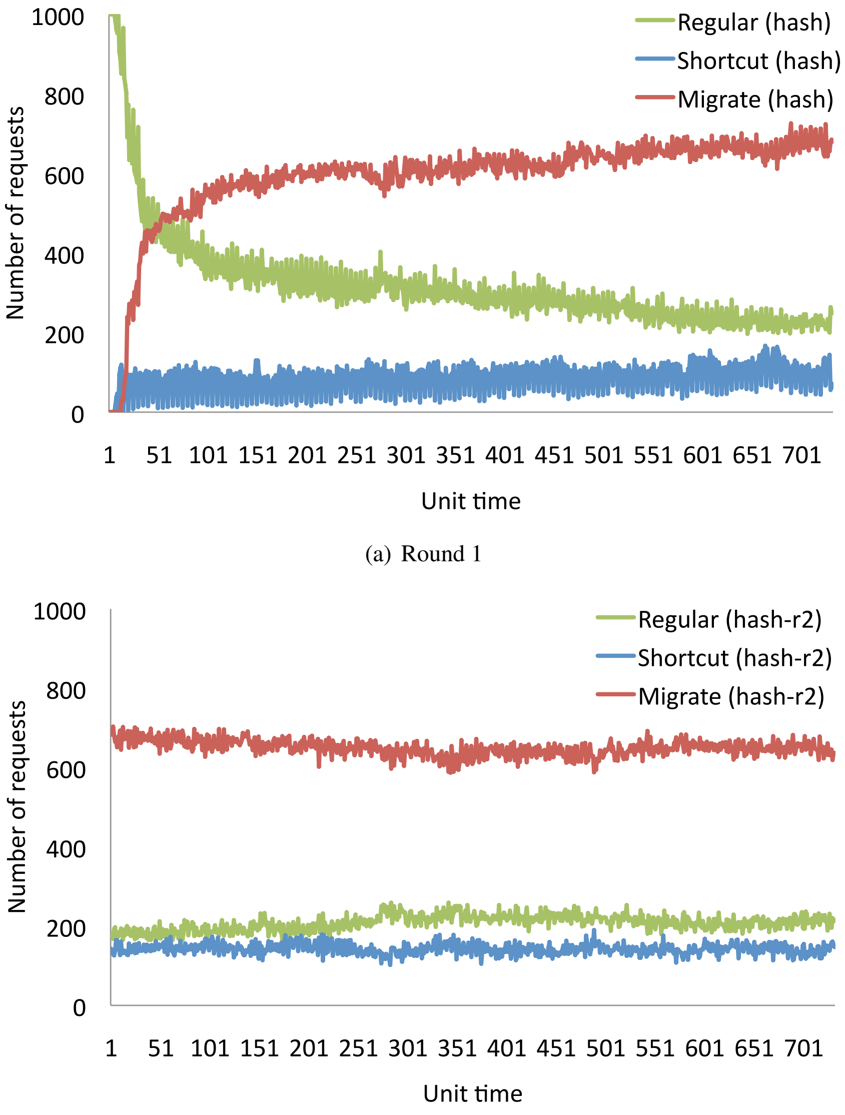
Figure 4.10 shows the number of requests for the three different path types with the hash-based algorithm. ‘Regular path’ means the source reached the destination using the path given in the initial state. ‘Shortcut path’ and ‘Moved path’ are self-explanatory. When the number of communications using the shortcut path increased, the number using the ‘regular path’ decreased. In addition, more source and destination pairs used the ‘shortcut path’ initially, but as these popular destinations migrated, more pairs used the ‘moved path’. In Figure 4.10(b), the number of requests using each path are stable, maintaining the values of Figure 4.10(a). The results shown here are only example cases.

The result of name-based is not shown due to the small difference in the numbers which is thought to be caused by the relatively small database used in the simulation. Since the name-based algorithm distributes the database in accordance with the name’s ASCII, it creates a less balanced distribution of the database, so more routers are needed for storage than with the hash-based algorithm.

4.2.4 Mapping between Virtual Topology and Physical Topology

The two open issues are listed in this section and the next one, which should be considered in realizing the name-based routing: mapping of virtual to physical topology and the deployment. Since these are not the main point in this thesis, these will not be explained in detail but are important questions that have to be answered in the future work.

The three algorithms of distributing the routing entries in the format of names are demonstrated in Section 4.1.3, which can be considered as deciding where to place the large database of names in the virtual routers. In this section, some ideas on how to map these virtually interconnected routers to currently deployed routers in the network are discussed.



(a) Round 1

(b) Round 2

Figure 4.10: Number of requests via regular, shortcut, and migrated paths

- Number of hops: The number of hops between the routers can be used as a way of measuring the distance between the routers. For example, after statically configuring the forwarding/routing tables in backbone routers to contain the information of TLDs, the forwarding/routing information of the second and the third level can be injected to the local gateway and the edge routers that has the smallest number of hops from the corresponding backbone router.

- Routing redundancy and caching: The routing information of the popular named contents that have a high access frequency can be cached in the routers. In addition, by moving the popular forwarding/routing entry to close to the source where there are many requests reduces the total number of hops required to access the required information.

- Data format optimized for TCAM: Since TCAMs excel in longest prefix match, this fact can be utilized by attaching a location information in the packet header. Similar to landmark clustering of algorithm by Xu et al. [64], when there are multiple copies of the content, the one that shares the most prefix with the source can be selected as the destination.

The challenges such as ‘disaggregation of entries’ is also well recognized. That is, the initial forwarding/routing table is an aggregation of names, especially if the list of the names are organized in a hierarchical way such as FQDNs. As these entries get updated, i.e., added, deleted, or changed from the original storage place (router in this case), the list of entries get ‘disaggregated’. A mapping algorithm that can handle these updated information is required. This will also be taken into consideration in addition to the implementation ideas when designing the mapping algorithm.

4.2.5 Deployment

A table comparing the features of the various name-based routing designs can be found in the work of Rajahalme et al. [71]. Considering the deployment method, two main streams exist: universal and partial/overlay. Universal, i.e. clean-slate deployment is proposed by [20,45,72] and partial/overlay deployment by [1,19,71,73]. The proposal in this thesis shares the same idea that the deployment of a new architecture should take place one step at a time. For example, in the initial phase, the name information can be tunneled by encapsulating the packet with the IP address within a small number of overlay tier 1 routers. When a larger number of routers become literate of the names, the nodes can be directly connected using the packet header information in the format of names, eventually replacing the network routers which know how to route with only the names.

4.3 Summary

In this chapter, FQDN is used in routing as a substitute for the conventional IP address. The proposed system is shown feasible in the network layer by estimating the required memory size in routers through statistical evaluations of the currently existing FQDNs. In addition, it is shown that using names for routing is feasible and robust against the dynamic update of the database entries. Furthermore, the path length of the proposed system is compared with the overlay and a possibility of developing a new TCAM architecture suitable for the resource name-based routing is shown when virtual to physical topology mapping is considered. Some of the fundamental technologies required to support the resource name-based routing are sufficient hardware memory size and the robust network architecture. The study in this chapter suggests that it is possible to achieve routing with the FQDN within the routers is feasible by (i) developing TCAMs that have larger size than currently available TCAMs, (ii) achieving robustness of the network topology against dynamically changing database entries, and (iii) having a locality-aware mapping algorithm. These factors are not limited to only name-based routing, but based on the discussions made in this chapter, the routing can be generalized based on a ‘resource’, regardless of being a name, category, or type of a content, as long as the description method of a resource is structured as that of FQDN.

Chapter 5

A New Memory Architecture for Realizing Name Lookup Tables in Resource-centric Networks

5.1 Name Lookup Table in the Routers

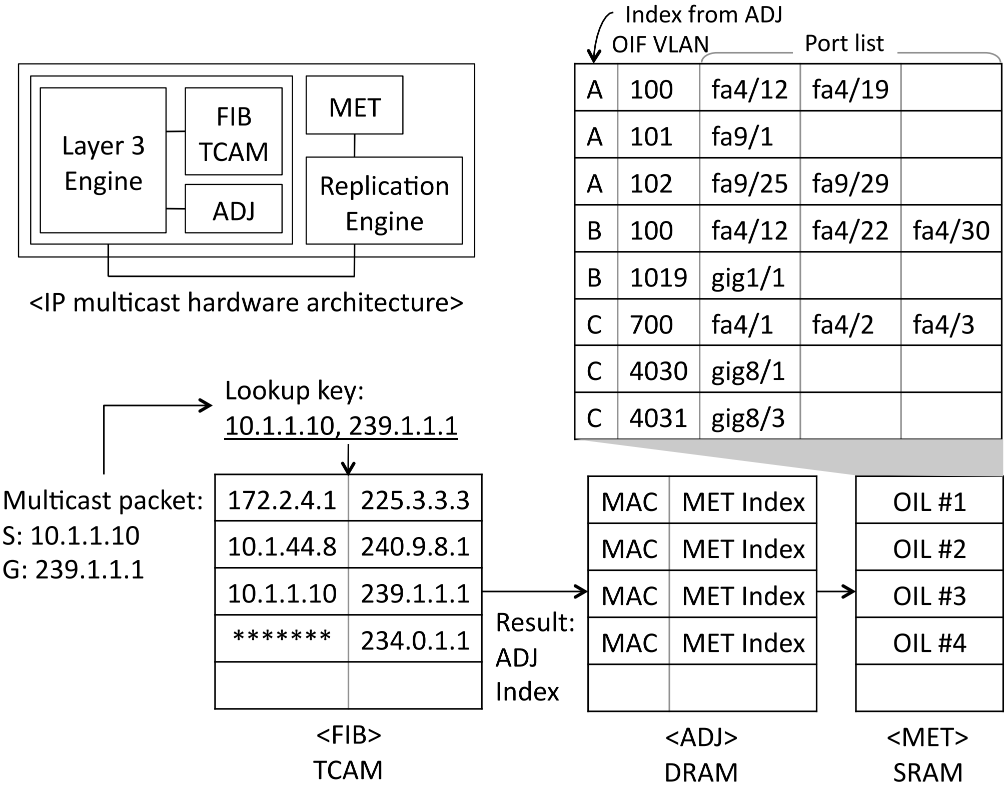
In order to propose a new name lookup table in the routers, it is important to understand the current technology and specify why it is insufficient in supporting the large-scale name and the user information. Based on the speculation, the three lookup tables are proposed in the following part.

5.1.1 Background Technologies

In this section, the background technologies of the lookup table architectures in the conventional routers for performing the multicast and the multimatch using TCAM is introduced.

Multicast

Figure 5.1 shows an example of the router hardware and the memory used for the multicast in Cisco Catalyst 6500 [21]. The multicast forwarding table is composed of multicast forwarding information base (FIB), adjacency table (ADJ), and multicast expansion table (MET). FIB has the



	OIF VLAN	Port list		
A	100	fa4/12	fa4/19	
A	101	fa9/1		
A	102	fa9/25	fa9/29	
B	100	fa4/12	fa4/22	fa4/30
B	1019	gig1/1		
C	700	fa4/1	fa4/2	fa4/3
C	4030	gig8/1		
C	4031	gig8/3		

172.2.4.1	225.3.3.3
10.1.44.8	240.9.8.1
10.1.1.10	239.1.1.1
*******	234.0.1.1

MAC	MET Index
MAC	MET Index
MAC	MET Index
MAC	MET Index

OIL #1
OIL #2
OIL #3
OIL #4

Figure 5.1: Multicast hardware architecture of Cisco catalyst 6500

information of the source and the multicast group, (S, G) or ($\star$, G) and ADJ has *rewrite MAC* (substitutes the source MAC address with the current router's output interface and the destination MAC address with that of the next hop router's or node's input interface) and *MET index* (output interface list (OIL) of duplicated packets to be sent). FIB, ADJ, and MET are all memories with a finite capacity. Especially for the MET that keeps the information of the multiple output interface, the maximum number of output interfaces it can store is up to 64,000 [21]. When the table is full, multicast packets can not be processed in the hardware and the software switching in the CPU is required which drastically degrades the performance due to its slow speed [22].

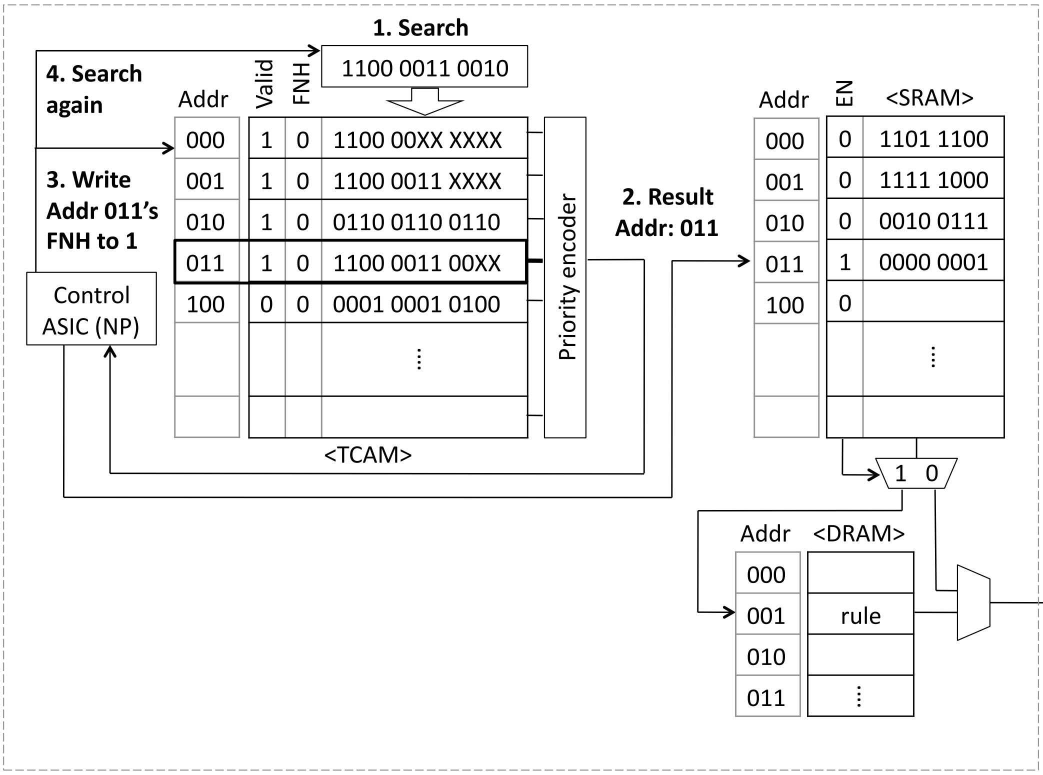


Figure 5.2: Multimatching and parallel process in search unit

Multimatch

The address lookup in the routers compares the input packet's destination IP address and the router's forwarding table's entry. Generally a single result that matches the longest bit sequence is returned which is also known as the longest prefix match. The router's forwarding table is written in the TCAM and the memory address of a single entry that satisfies the search key is returned to the SRAM by the priority encoder (PE). However in a situation where each entry that matches the condition has to be returned (e.g. network intrusion detection system) [74], TCAM is searched multiple times as shown in Figure 5.2.

The first round of the search process is performed with a search key (**1. Search**). The memory address of the entry that matches the key by PE is sent to the network processor and to the SRAM (**2. Result Addr: 011**). Normally the search is terminated here but when a multimatch is performed, the force no hit (FNH) bit — specifically used for the multimatch process in TCAMs — of the matched entry is set to 1 (**3. Write Addr 011's FNH to 1**) to further prevent the entry being matched for the given key. After setting the FNH bit the TCAM is searched again (**4. Search again**) until the 'no hit' signal is returned. After getting the result memory address from the TCAM, the SRAM returns the data of that address to the switch large scale integration (LSI) if the enable (EN) bit is set to 0. When the packet classification or complicated rules such as multicasting is needed, the EN bit is set to 1 and the further processing at the DRAM is required.

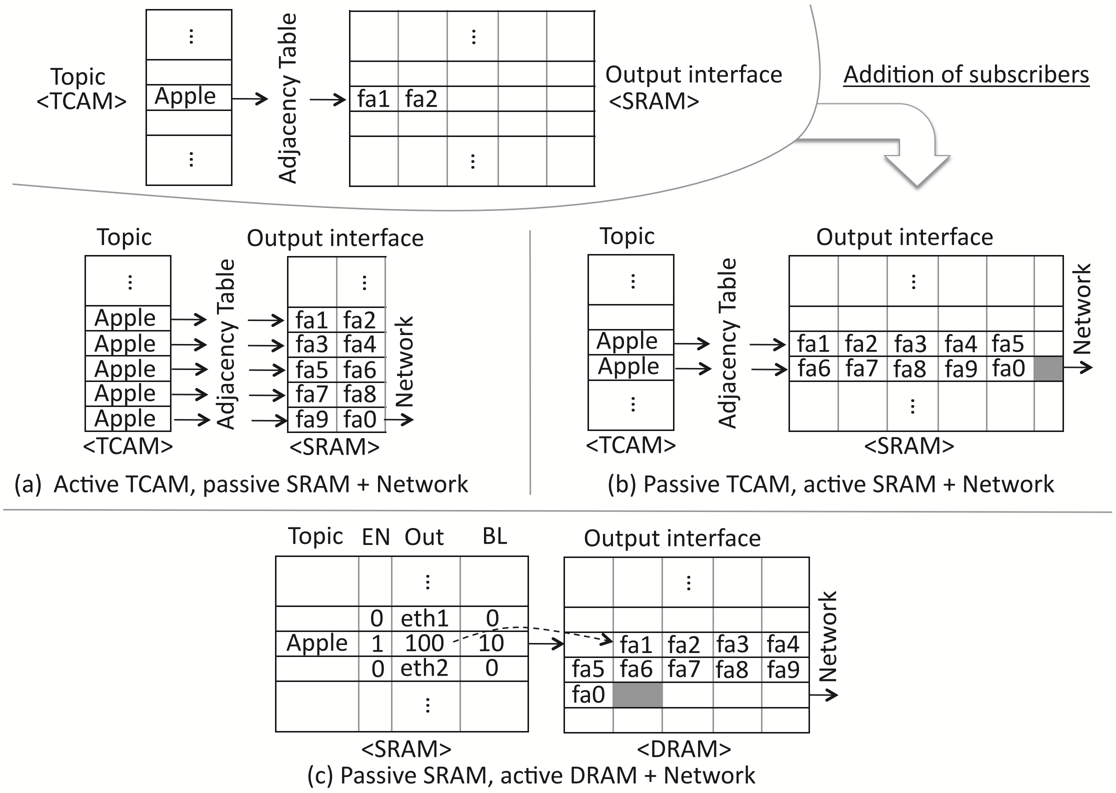
5.1.2 Proposed Scenarios for Storing Name Database

In the conventional routers, multicast is performed by matching a single TCAM entry with a given search key. In addition, the candidate output interfaces are stored in MET as shown in Figure 5.1. However for communication models such as pub/sub, the number of the subscribers of a resource with a topic name can exceed several tens of thousands which can lead to a problem since the current memory structure is not optimized to support the system well for a large number of subscribers. In order to support pub/sub notification service in the routers, three memory scenarios are proposed as shown in Figure 5.3 that make use of the multimatch capability of high-speed search memory such as TCAM and the general purpose memory such as SRAM and DRAM.

Scenario A: Active TCAM and Passive SRAM

Scenario A maximizes the usage (i.e. active usage) of the multimatching in TCAM and minimizes the usage (i.e. passive usage) of SRAM by storing a small number of output interfaces for a topic per an SRAM entry. When the list of output interfaces to be written in the SRAM exceeds the capacity, the process of searching topic names and forwarding contents is delegated to other routers in the network rather than having the slow CPU of the router do the software routing.

- Pros: Ability to utilize TCAM for high-speed searching. The latency between the steps in the multimatch described in Section 5.1.1 can be used to search for different keys in parallel. For example, when the result of the first search key is returned (**2. Result Addr: 011**), the TCAM can be searched with the next key (**1. Search**) in Figure 5.2.



(a) Active TCAM, passive SRAM + Network

(b) Passive TCAM, active SRAM + Network

(c) Passive SRAM, active DRAM + Network

Figure 5.3: Three scenarios to solve problems in current multicast hardware

- Cons: Since multiple entries of TCAM are used to express a single topic, the latency between the search steps shown in Figure 5.2 can be high if the parallel search of the different keys is not utilized well. In addition, if two or more search keys are related to each other, setting the single FNH bit is insufficient. For example, FNH of Addr `011` is set to `1` after being searched with the first key. If the next search key has a same prefix (e.g. `1100 0011 0011`), this also matches the entry addressed `011` but ends up skipping the entry since the FNH bit is set to `1` after the first key. Therefore, the number of different search keys that can be processed in parallel should depend on the number of the FNH bits.

Scenario B: Passive TCAM and Active SRAM

Contrary to the Scenario A, Scenario B makes an active usage of SRAM and minimizes the usage of TCAM and its multimatching process. For the number of users that exceeds the capacity of a single SRAM entry, multiple entries have to be used and the corresponding number of entries are used in TCAM as well, as shown in Figure 5.3(b). In addition, for the topic names that could not be written in a single router, the process of searching topic names and forwarding the contents is delegated to the other routers in the network.

- Pros: Chip cost per unit area of SRAM is 20% compared to TCAM and less FNH bits are required by minimizing the number of multimatches in the TCAM. Minimizing the memory space used in TCAM also implies that the number of searches in TCAM decreases, resulting in a lower power consumption.
- Cons: If a large number of horizontal bits in a row is reserved for the SRAM in order to minimize the number of the required entries, the utilization of the memory can decrease if there are a lot of topics with a small number of subscribers. This can be a big problem if there are a large number of topic names with a small number of subscribers for each topic.

Scenario C: Passive SRAM and Active DRAM

The output interface is written in DRAM. Unlike the Scenarios A and B, only a single SRAM entry is consumed for a topic name. Also, the information whether DRAM referral is necessary is stored in SRAM. EN bit is set to `0` or `1`, where `0` indicates a single output interface is written in the SRAM and `1` indicates the DRAM address that has to be referred is written in the *Out* field with the burst length (BL) field. In addition, when the list of the output interface exceeds the capacity of the DRAM, the process of searching topic names and forwarding the contents is delegated to the other routers in the network.

- Pros: The chip cost per unit area of DRAM is 0.1% compared to SRAM and can handle a large number of subscribers at a low cost. In addition, by using a single SRAM entry for a topic name can reduce the latency for returning the SRAM result multiple times.

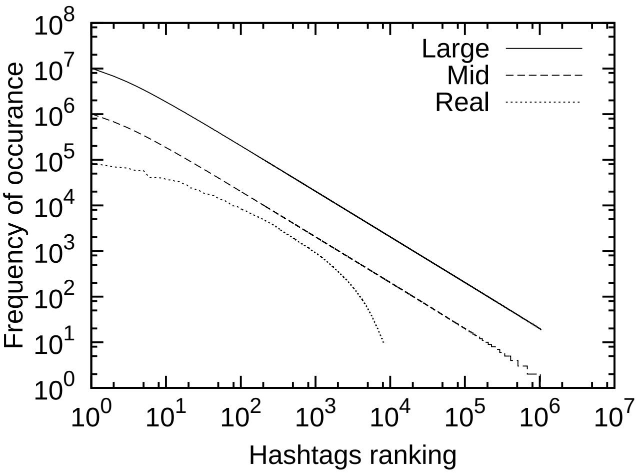


Figure 5.4: Three databases used in the evaluation

- Cons: DRAM has a high latency value compared to TCAM and SRAM, affecting the speed of the overall search process. In addition, increasing DRAM ratio for the memory architecture may decrease the memory utilization due to the unbalanced storage for the names and the subscribers.

5.2 Simulation Experiments and Discussions

In this section, the memory architecture of the three scenarios in Section 5.1.2 with the parameters such as memory cost, latency, and utilization is evaluated.

The database of the topic names and the number of subscribers used in the evaluation is taken from Hashtagsjp [75], where hashtags of Twitter [76] are considered to be the topic names, and the users who twittered using the hashtags are considered to be the subscribers. The database is assumed to have a rough Zipf distribution. In other words, the topic name subscribed by the ith most users has the number of $\frac{1}{i}$ users compared to that of the most popular topic name. This trend is also seen in other social networks that use tagging data [77]. In addition, two synthetic databases that have larger number of topics and subscribers are used. The characteristic of these three database is shown in Figure 5.4 where Real is the database from Hashtagsjp [75] and Mid and Large are the synthetic databases.

Table 5.1: Comparison of memories

	DRAM	Premium DRAM	SRAM
Wire speed	50 ns	20 ns	1 ns
Read latency	50 ns	20 ns	2 ns
Cost per 10 Mbit	$0.01	$1	$10

Table 5.1 shows the wire speed, read latency, and cost per 10 Mbit of DRAM, premium DRAM, and SRAM. Wire speed is the random access interval at which write or read commands can be accepted. Read latency is the consumed time to retrieve the requested data after issuing the read command to the memory. In this evaluation, premium DRAM which has lower latency than regular DRAM is used and DRAM in rest of this paper refers to premium DRAM. TCAM is deliberately left out of the table since its purpose is different from these general-purpose memories. TCAM specializes in searching and uses mega searches per second (Msps) as the unit of the speed which is typically 360 Msps (not utilized in this paper). The TCAM latency estimated in this paper is the search latency for the entire TCAM chip in a clock (consumed time to retrieve the search result after issuing the search command) which is 75 nsec. In addition, TCAM's cost per 10 Mbit is assumed as $50.

As for the unit of TCAM, SRAM, and DRAM in this evaluation, those used currently by Renesas Electronics Corporation [78] is followed, as 20 Mbit, 72 Mbit, and 2 Gbit, respectively. Referring to the cost per 10 Mbit in Table 5.1, the chip cost for TCAM, SRAM, and DRAM is $100, $72, and $200, respectively. For example, if the combination for scenario A is one TCAM and one SRAM, the total chip cost is $172 which is calculated regardless of the used entries. The term actual cost (or utilized cost) is defined as the used entries multiplied by the cost per entry of each memory.

5.2.1 Cost and Latency

The actual cost and the read latency when storing the database of topic names and the subscribers to the memories is first calculated. The number of subscribers that can be stored in a router depends on the horizontal number of bits (row length) of the SRAM for Scenarios A and B and the row length of DRAM for Scenario C. The trade-off between the row length and the necessary entries

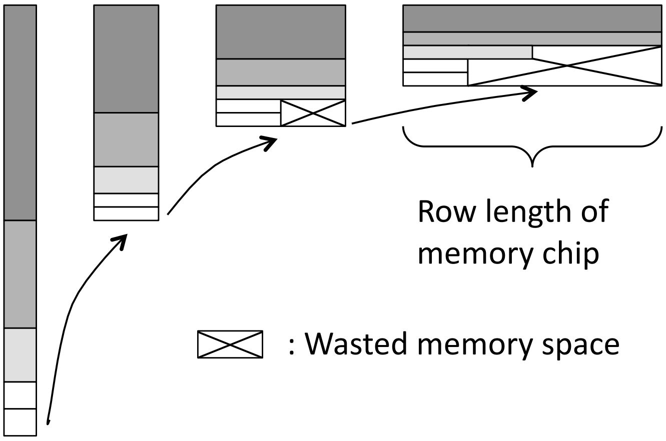


Figure 5.5: Trade-off between the row length and the necessary entries in the memory

in the memory is shown in Figure 5.5. When the row length of the memory is short more entries are needed to store the database. As the row length increases, the number of entries decreases but only up to a certain value since at least one row is needed for topic name. In other words, a long row in the memory is suitable for storing topics with a large number of subscribers but when there are many topics with a small number of subscribers, it can lead to the decreased utilization of the SRAM and DRAM. The actual cost and the utilization of the scenarios are evaluated by setting this row length of the SRAM and DRAM as variables. When an SRAM entry is added, an additional entry in TCAM is required to be able to refer to that new SRAM entry. Therefore, the necessity for the multimatch increases and more FNH bits in the TCAM are required.

Figure 5.6 shows the relationship between the read latency and the actual memory cost of the three scenarios. The x axis of Figures 5.6(a) and 5.6(b) is the row length of SRAM and DRAM, respectively. When the row is short it is closer to Scenario A and otherwise closer to Scenario B in Figure 5.6(a). The extreme example of the Scenario A is when only a single user can be written in an SRAM entry, consuming ‘number of topic names $\times$ number of subscribers in each topic’ entries.

The actual cost is defined as the used entries multiplied by the cost per entry of each memory. For the extreme example of Scenario A, the actual cost ends up being approximately \$8,600. Furthermore, it uses the entry of the TCAM the most among the three scenarios, resulting in high search latency. Extending the row of the SRAM reduces the number of entries to store the subscribers for each topic up to a certain row length, also reducing the actual memory cost. This certain number of SRAM row length, i.e. the value which minimizes the cost can be attained by differentiating Equation (5.1) (required cost to write the database of topic names and the subscribers) by x, where x is the SRAM row length.

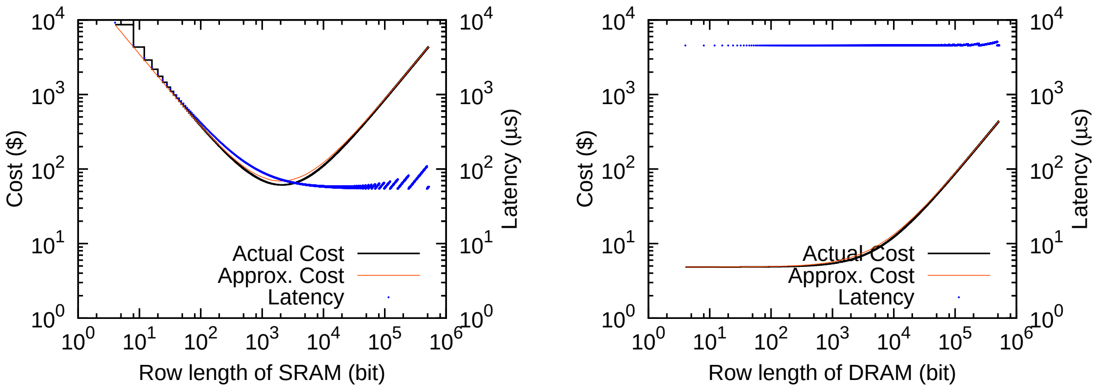
$$Cost = (T + sx) \sum_{i=1}^{R} \left\lceil \frac{b_i P}{x} \right\rceil \tag{5.1}$$

The parameters are as follows:

- Cost per entry of TCAM: $T = 320 \times 0.5 \times 10^{-5}$
  (An assumption: 320 bits per a TCAM entry)
- Cost per bit of SRAM: $s = 0.1 \times 10^{-5}$
- Number of topic names in the database: R
- Number of subscribers for ith most popular topic: b_i
- Number of bits per output interface: P
  (An assumption: P= 4 bits)

The term on the right hand side consists of a product of linear increase and exponential decrease. However, due to the ceiling function, the sum does not approach 0, but is truncated at 1 for each $x \geq b_i P$. Therefore, Equation (5.1) initially decreases then increases. Since the ceiling function is not differentiable, Equation (5.1) is approximated as the following where $B = \sum_{i=1}^{R} b_i$.

$$\begin{aligned} Cost &= (T + sx) \sum_{i=1}^{R} \left(\frac{b_i P}{x} + 1 \right) \\ &= (T + sx) \left(\frac{BP}{x} + R \right) \\ &= \frac{BPT}{x} + sRx + (sBP + RT) \end{aligned}$$



(a) Scenario A and B

(b) Scenario C

Figure 5.6: Evaluation of actual cost and latency using real-life database

Table 5.2: Row length (x bits) of SRAM that minimizes the cost in Scenarios A and B

	Real	Mid	Large
x bits from Eq. (5.1)	1,948	376	1,272
Min. Cost from Eq. (5.1)	\$61	\$2,526	\$5,111
x bits from Eq. (5.2)	2,024	419	1,335
Min. Cost from Eq. (5.2)	\$69	\$2,697	\$5,699

Differentiating *Cost* by x and the x that satisfies $Cost' = 0$ is the value which minimizes the *Cost*.

$$Cost' = -\frac{BPT}{x^2} + sR$$

$$x = \sqrt{\frac{BPT}{sR}} \tag{5.2}$$

Table 5.2 shows the result of the Equations (5.1) and (5.2). For all databases the approximated x and the cost from the Equation (5.2) are slightly larger than the actual x and cost from the Equation (5.1) but generally show a good resemblance as shown in Figure 5.6 (Actual Cost and Approx. Cost).

Figure 5.6(b) shows the result of Scenario C where a topic name consumes an entry in SRAM and the subscribers are written in the DRAM. Since the cost per bit in DRAM is lower than any of the memory types, increasing the row of the DRAM does not have a large effect on the overall actual memory cost even though the required entries decrease. For comparison with Scenarios A and B, a description on the actual cost of Scenario C is given below.

The additional parameters are as follows:

- Cost per entry of SRAM: $S = 320 \times 0.1 \times 10^{-5}$
  (An assumption: 320 bits per an SRAM entry in Scenario C)
- Cost per bit of DRAM: $d = 0.1 \times 10^{-6}$
- Row length of DRAM: z

$$Cost_c = RS + dz \sum_{i=1}^{R} \left\lceil \frac{b_i P}{z} \right\rceil \tag{5.3}$$

The approximation of Equation (5.3) is,

$$\begin{aligned} Cost_c &= RS + dz \sum_{i=1}^{R} \left(\frac{b_i P}{z} + 1 \right) \\ &= RS + dz \left(\frac{BP}{z} + R \right) \\ &= SR + dBP + dzR \end{aligned}$$

Differentiating $Cost_c$ by z and the z that satisfies $Cost_c' = 0$ is the value which minimizes the $Cost_c$.

$$Cost_c' = dR$$

Since the differentiated value is constant, the minimal $Cost_c$ does not exist. However, drastically increasing the horizontal size creates wasted space after all, making the actual memory cost approximately $440 which is still trivial compared that of Scenarios A and B.

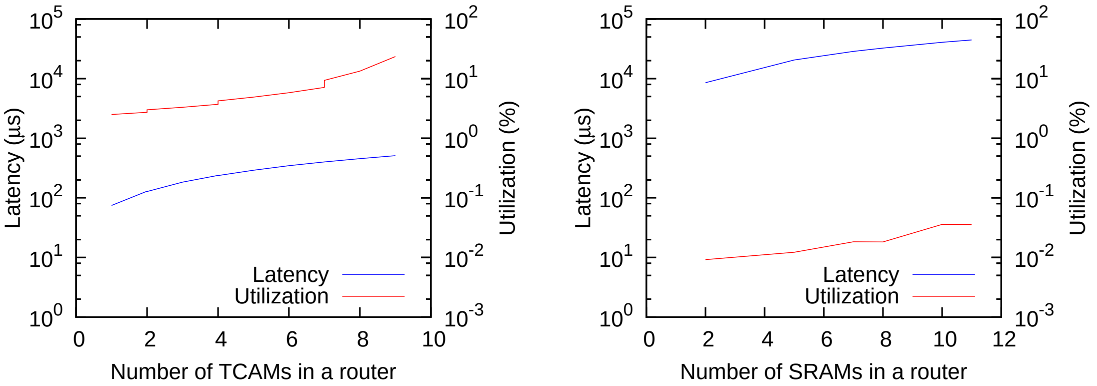
The read latency is the worst-case value which is defined by the time consumed for searching and returning the result (list of subscribers) of the most popular topic name. The worst-case read latency for Scenarios A and B is calculated by summation of the items below.

- Time for searching TCAM to find out whether the search key (topic name) is stored.
- Reading and returning the information of subscribers stored in SRAM.
- When the subscribers are written in multiple entries of SRAM, the time it takes to search TCAM again (multimatch) until the 'no hit' signal is returned.

In Scenario C, SRAM is searched whether the topic name exists in the chip and if the name exists, the corresponding subscriber information stored in the DRAM is returned. The overall latency depends on the number of SRAM entries to search and the word length of SRAM and DRAM since the result (output interface) has to be returned multiple times according to that word length. When only the cost is considered, it seems that the Scenario C is the best solution since the information of subscribers is stored in the most inexpensive DRAM. However, the overall processing speed is affected by the slow SRAM and DRAM as shown in Table 5.1. The latency of Scenario A starts high since SRAM row length is short, resulting in multiple TCAM entries as shown in Figure 5.3 (a). This causes multiple lookup of TCAM chip each with 75 nsec. As the SRAM row length increases, the read latency drops to approximately 60 μsec. This is because the number of used TCAM entries is constant after a certain number of SRAM row length and the only factor affecting the total latency is the (Number of Subscribers)/(Word Length). However, the latency of Scenario C is almost constant with approximately 4.5 msec where the main contributor is from searching the SRAM. Therefore, Scenarios A and B are considered superior to Scenario C and in addition, the latency of Scenario A and B can be further reduced by searching for different keys in TCAM in parallel.

5.2.2 Latency and Utilization

In reality, the row length of SRAM and DRAM can neither be adjusted nor increased to a large value of tens of thousands as one desires as shown in Figure 5.6. Instead more restrictions and realism



(a) Scenario B and A

(b) Scenario C

Figure 5.7: Evaluation of latency and utilization using real-life database

is added to get the results shown in Figure 5.7 by setting the manufacturing chip cost to $1,000 and the row length of the SRAM in Scenario A & B to 32 bits, SRAM in Scenario C to 320 bits, and DRAM in Scenario C to 32 bits. To compare the memory combinations that have a similar manufacturing chip cost, only those between $900 and $1,000 are selected for the evaluation. The x axis of Figure 5.7(a) is the number of TCAM chips and since the manufacturing chip cost is fixed, the number of SRAM chips decreases as TCAMs increase. When the number of TCAM chips is small it is closer to Scenario B and otherwise closer to Scenario A. Two different utilization values exist when there are four and seven TCAM chips because there are two combinations each that satisfy $900 – $1,000 condition. Four TCAM chips with seven and eight SRAMs, and seven TCAM chips with three and four SRAM chips are all in the fixed budget range. The x axis of Figure 5.7(b) is the number of SRAM chips and since Scenario C puts emphasis on the memory capacity, the number of DRAM decrease as the SRAMs increase.

The utilization of Scenarios A and B increase as the number of TCAM chips increase. This is because for a given budget, the number of SRAM decreases as the number of TCAM increase, resulting in lower memory capacity for storing the output interface. The same can be said about the Scenario C since the number of DRAM chips decrease as the number of SRAM chips increase. For larger databases, the utilization can exceed 100% meaning all memory space in a router is used. When this occurs, the searching of topic names and forwarding is delegated to other routers in the network.

From Figure 5.7(a), it is shown that the overall latency is largely affected by the number of TCAMs whereas in Figure 5.7(b), the number of SRAMs affect the overall latency. The latency for Scenario A and B ranges from 75 to 510 μsec whereas for Scenario C the range is from 8.5 to 45 msec. As mentioned in Section 5.2.1, Scenarios A and B are considered superior to Scenario C due to significantly low latency. Among the Scenario A and B, B is considered better as long as the row length of SRAM does not exceed the point where the utilized cost starts to increase again as shown in Figure 5.6.

5.2.3 Cost with Multiple Rendezvous Points

The evaluations in the Sections 5.2.1 and 5.2.2 were based on a rather extreme assumption that there is only a single rendezvous point (RP) managing a large database of topic names and subscribers. This is unrealistic since a single point of failure is created and above all routing is impossible when only one RP exists for all topics. In this section, the effect of placing multiple RPs for a topic name in the network is evaluated which was briefly mentioned as *+Network* in Figure 5.3.

Several research in the past mention of placing multiple RPs. Anycast RP [79] allows two or more RPs such as in Figure 5.8 in order to share the load for source registration and the ability to act as hot backup routers for each other, whereas generally only a single RP exists for a group in IP multicast. However, these RPs have identical database and do not have a mechanism to store a large database divided into small tables. Zhang et al. [80] designates hibernating children RPs as potential RPs in order to reduce the work overload. The premise is that forwarding the content from RP to the subscribers follows the reverse path of interest sent from the subscribers to RP. The nodes on the distribution tree with RP as the root are designated as hibernating children. When the RP's load threshold exceeds a set value it delegates the job of forwarding the content by unloading all subscriptions to one of the hibernating children, typically to the one with the largest forwarding fraction. This child becomes the new RP for that content. Recursively repeating the process leads to balanced distribution of load and storage. However, the scheme only works when the contents follow the reverse path of the interest which may not be accurate if there are failures in

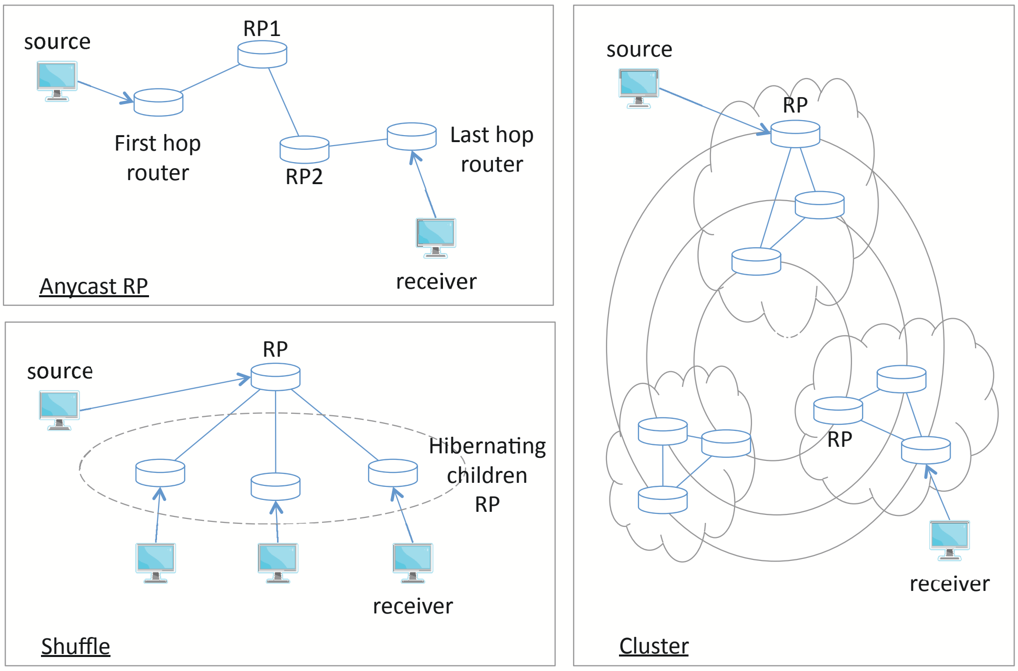


Figure 5.8: Multiple rendezvous points (RPs)

the RP network. In the paper by Jafarpour et al. [81], RPs construct clusters to prevent subscription messages being flooded to the whole network. In other words, the interest from subscribers is disseminated only among the RPs within the same cluster. RPs are connected to every RPs in the same cluster and to at least one RP in other clusters. The published contents are first broadcasted to all clusters through the publisher RP's DHT ring. Next, the content is matched to subscriptions in each cluster and is delivered to the RPs with matching subscription. Load balancing is done by designating more number of clusters for a popular subscription. In addition, overloaded RPs offloads the load to underloaded RPs in the same cluster.

In summary, the goal of placing multiple RPs is to reduce the below three items.

- Work overload: process of analyzing and matching published/forwarded contents in RP with the stored interest information from the subscribers
- Delay: amount of time spent delivering the content satisfying the interest notified from a

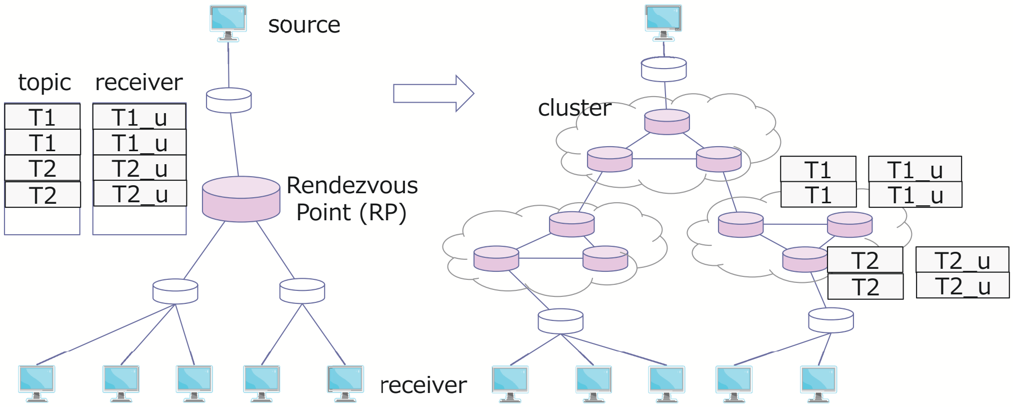
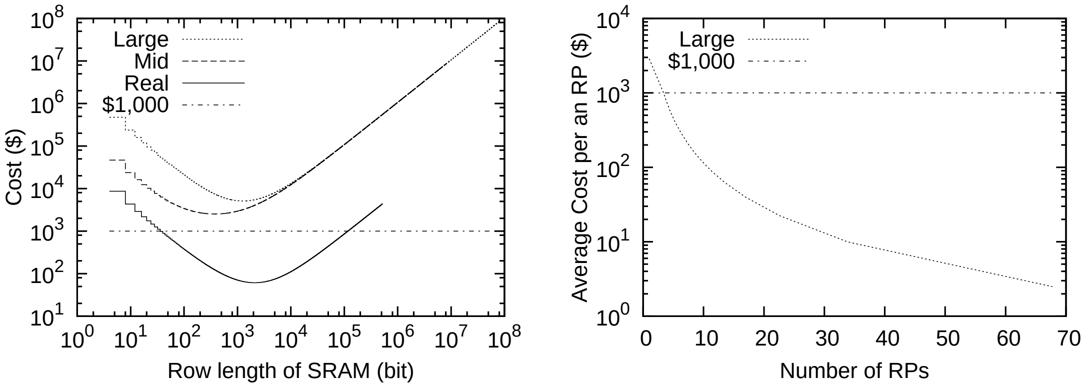


Figure 5.9: Clusters of multiple RPs with a single function

subscriber to its gateway router

- Memory storage: amount of information on content and its subscribers per a RP

Work overload and delay is considered to be roughly proportional to the RP's routing table size [82]. This is because when many types of topic names are stored in the RP, it takes time to compare if the forwarded content matches the topic names. In addition, when a large number of subscribers per a topic name exists, delay increase trying to forward the matching content to every user in the list. Therefore, the focus is on the issue of how to reduce the stored information per a RP by distributing the routing information among the multiple RPs and evaluate the reduced cost. The cost per RP is estimated to decrease exponentially as the number of RPs increase. The reason behind this logic is that the maximum memory capacity is mainly influenced by the semiconductor manufacturing process technology. For example, The optimal onboard DRAM memory capacity is 1 Gbit for 65 nm and 2 Gbit for 40 nm. Regardless of the hardware type, the memory is generally loaded with the maximum possible size. While it is possible to exceed this maximum size, for example 2 Gbit with 65 nm technology, this results in dramatically increased cost ($ per bit) due to the enlarged chip area and the decreased yield. The same can be said about the storage memory in RPs. While it is 'possible' to increase the maximum memory capacity, exceeding the optimal value of the process technology results in an exponential cost increase. The reason for this is, when



(a) Utilized cost with SRAM row length as a variable, RP=1

(b) Utilized cost with number of RPs as a variable, SRAM row length=32 bits

Figure 5.10: Evaluation of utilized cost using real-life and synthetic database

x axis is onboard memory and y axis is the total cost, a linear curve ($ per bit is fixed) changes to an exponential curve ($ per bit increases) at a certain x value and furthermore this 'certain x value' results in a higher y as the process technology generation evolves from 65 nm, 40 nm, and 28 nm. Therefore, when storing a database of a large size such as topic names and a large number of subscribers, dividing the database by n number of RPs results in more than just an $1/n$ cost.

In this paper cluster is defined as a group of RPs which have split from the original RP that has exceeded its storage threshold. Clusters are hierarchically structured as shown in Figure 5.9. In a cluster, RPs are physically multiple routers but they perform the function of a single RP, appearing as a single router when seen from the outside of that cluster. The purpose of having a hierarchical structure is to reduce the amount of traffic and the usage of upper layer's RP resource when subscribers can retrieve contents by only accessing RPs in the lower layer. Here the focus is on the effect, specifically of reducing the cost of placing RPs in hierarchical clusters rather than the routing method itself, which is similar to the IP multicast especially the Protocol Independent Multicast - Sparse Mode (PIM-SM) [79] in theory.

Starting from a single RP, topic names and the user information are distributed among multiple routers when the utilization of the router memory exceeds 100% in Figure 5.7 or the given budget of $1,000 shown in Figure 5.10(a). Name database distributing algorithm 'Hybrid Distribution' proposed in Section 4.1.3 is used to more or less equally distribute the database by hashing the topic names and grouping the names by the same hash value. A fixed number of SRAM row length as 32 bits is used to compare the result with that of Section 5.2.2 and assuming each RP has five TCAM and six SRAM chips to satisfy the $1,000 budget limit.

As the database size increase, the number of required RPs increase as well and the fact that average cost per an RP decreases exponentially is shown in the Figure 5.10(b). Specifically, when only a single RP is used, the cost of memory chips adds up to $3,000. The result shows that at least four RPs are required to store the *Large* database and increasing the number of RP from 4 to 68 decreases the cost per RP from $997 to $17.

5.3 Summary

In this chapter, the advantages of realizing resource-centric routing in the network layer is discussed and the proposed router lookup architecture for storing and searching topic names of contents is evaluated. As a result, it is shown that the memory architecture is affected by the database of topic names and users having Zipf distribution, and the latency of each memory. In addition, it is also shown that distributing the database among the multiple rendezvous points results in an exponentially reduced cost compared to storing the database in a single RP.

Chapter 6

Conclusion and Future Work

The future Internet is expected to be a more intelligent system than the current one, for example being able to route with the resource as well as with the conventional IP address. Using *name* as a one way to express the resource, it is shown in this thesis that the name-based routing in the network layer is a feasible in both hardware and the network environment.

In Chapter 3, the characteristic of TCAM is shown by proposing a range matching device and an optimized longest prefix matching algorithm for storing the ACL's port number in ranges. The evaluation result show that the required number of TCAM entries for storing the ranges is reduced by 50%. The importance of this evaluation is twofold: (i) the confirmation that it is crucial to effectively use the $\star$ value to better utilize TCAM when storing data. This confirmation is used in investigating whether the routers are capable of storing the name information of each named node, therefore showing the feasibility of the proposed name-based routing and (ii) the fact that the particular data used in the evaluation is ACL, one example of examining and controlling the flow, is a preliminary step to the name-based routing. When the resource is sent, names representing the particular resource should be examined in a similar manner.

The above knowledge is used in Chapter 4 when storing the routing information in FQDN format in the routers. Using FQDN as the name representing each node, the proposed network architecture and the name structure is described. In addition, the routing protocol, especially how and what messages are exchanged when registering and forwarding name information. Since names are generally longer than the conventional IP addresses, routing information is distributed among the routers with the proposed algorithm. The resulting number of the routers required to support the new name-based routing is approximately 1% of the currently deployed number of routers. This shows that name-based routing shows a feasibility in hardware even when considering the limitations of the TCAM size in the routers. In addition, it is shown that using names in routing is feasible and robust against the dynamic update of the database entries.

Chapter 5 generalizes the above discussion by assuming the 'resource' is represented by names. When the information of names and the large-scale users of the resource is stored in the router, the lookup table should be adapted accordingly as well. In order to complete the packet forwarding within the network layer, i.e. the search for content which matches the interest, the routers acting as the rendezvous points of a publish/subscribe system should maintain the information of the content names and the users subscribing to the content. Three lookup table structures for the name lookup tables in the routers are proposed. The evaluation show that the lookup table scenario that has the lowest manufacturing cost is not always the optimal solution since it may result in a high latency. In addition, it is shown that the chip cost is exponentially reduced from $997 to $17 for distributing the names and the subscriber information by increasing the number of the rendezvous points. The type of memories used in the evaluation is a typical memory used in the industry. Although there might be an increase in the processing ability and a decrease in the memory cost, the overall relationship between the different types of the memories will remain relatively similar. The combinations of TCAM, SRAM, and DRAM in this thesis are typical examples and can be applied to other superior future memory types.

The evaluation in this thesis utilizes the current hardware resource, FQDN, and twitter database to show the feasibility of the present situation. The actual implementation of name/resource-based routing is expected to happen in approximately 10 years [14]. The estimation of hardware resource holds up for the future as well for the following reasons. The evaluation result in Chapter 4 show that 1% of the current number of routers is needed to store 7×10^8 names. These names are thought as the unique names of the web pages having the growth rate of 35% according to Figure 1.3. On the other hand, the memory capacity is expected to obey Moore's law (doubles in two years in case of RAM [83]). Therefore, the memory in the future is expected to provide sufficient volume in capacity to manage huge name-based routing information. As for the format of the name to describe a resource, it is estimated to be a hierarchical structure in order to be scalable and to avoid collision. For example, when acquiring a name for a device or information under the name *A* (highest in the hierarchy), it is the *A*'s responsibility to assure the new name *B* does not collide. However, *A* does not care of the name *C* that is under *B*. Furthermore, end node users can also name their device or the information they produce as *D* under *C* as long as the authority that named *C* of the name *A/B/C* can guarantee *A/B/C/D* is unique.

The current work in progress in on proposing an algorithm on estimating the optimal number of clusters to further reduce the load and the delay of each rendezvous point. After extending the proposed algorithms with further evaluations, proposing an implementation of resource-centric network in the network layer in the near future is anticipated.

Bibliography

[1] V. Jacobson, D. Smetters, J. Thornton, M. Plass, N. Briggs, and R. Braynard, “Networking Named Content,” in *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies (CoNEXT 2009)*, pp. 1–12, December 2009.

[2] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis, “Locator/ID Separation Protocol (LISP), draft-ietf-lisp-09, IETF Network Working Group,” October 2010.

[3] S. Paul, J. Pan, and R. Jain, “Architectures for the Future Networks and the Next Generation Internet: A Survey,” *Computer Communications*, vol. 34, pp. 2–42, January 2011.

[4] A. Carzaniga, M. J. Rutherford, and A. L. Wolf, “A Routing Scheme for Content-Based Networking,” in *Proceedings of IEEE International Conference on Computer Communications (INFOCOM 2004)*, vol. 2, pp. 918–928, March 2004.

[5] R. Chand and P. Felber, “A Scalable Protocol for Content-based Routing in Overlay Networks,” in *Proceedings of the 2nd IEEE International Symposium on Network Computing and Applications (NCA 2003)*, pp. 123–130, April 2003.

[6] BGP Reports. `http://bgp.potaroo.net/`.

[7] Internet Systems Consortium (ISC). `http://www.isc.org/`.

[8] J. Gantz and D. Reinsel, “Extracting Value from Chaos,” *IDC iView, `http://idcdocserv.com/1142`*, June 2011.

[9] Martin Hilbert and Priscila López, "The Worlds Technological Capacity to Store, Communicate, and Compute Information," *Science*, pp. 60–65, February 2011.

[10] J. E. Short, R. E. Bohn, and C. Baru, "How Much Information? 2010 – Report on Enterprise Server Information," *UCSD Global Information Industry Center*, January 2011.

[11] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi, "On the Evolution of User Interaction in Facebook," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Online Social Networks (WOSN 2009)*, pp. 37–42, August 2009.

[12] Named Data Networking. `http://www.named-data.net/`.

[13] FP7 4WARD. `http://www.4ward-project.eu/`.

[14] FP7 (Seventh Framework Programme). `http://cordis.europa.eu/fp7/home_en.html`.

[15] D. E. Taylor and J. S. Turner, "ClassBench: A Packet Classification Benchmark," *Technical Report (WUCSE2004-28), Department of Computer Science & Engineering, Washington University*, April 2004.

[16] Renesas, "Network Address Search Engine (9M/18M-bit Full Ternary CAM)." `http://documentation.renesas.com/jpn/products/memory/r10sm0001jj0300_memory.pdf`, 2011.

[17] Cisco, "ACL and QoS TCAM Exhaustion Avoidance on Catalyst 4500 Switches." `http://www.cisco.com/en/US/products/hw/switches/ps663/products_tech_note09186a008054a499.shtml`.

[18] The Measurement Factory. `http://dns.measurement-factory.com/surveys/201010/`.

[19] 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.

[20] T. Koponen, M. Chawla, B. Chun, A. Ermolinskiy, K. Kim, S. Shenker, and I. Stoica, “A Data-Oriented (and Beyond) Network Architecture,” *ACM SIGCOMM Computer Communication Review*, vol. 37, pp. 181–192, October 2007.

[21] Cisco Networkers, “Cisco Catalyst 6500 IP Multicast Architecture and Troubleshooting,” 2006.

[22] Cisco Systems, “Understanding and Configuring IP Multicast, Catalyst 4500 Series Switch Cisco IOS Software Configuration Guide, 12.1(12c)EW ,” 2010.

[23] G. Li, V. Muthusamy, and H. Jacobsen, “Adaptive Content-based Routing in General Overlay Topologies,” in *Proceedings of IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2008)*, pp. 1–21, December 2008.

[24] V. Sourlas, L. Gkatzikis, and L. Tassiulas, “On-Line Storage Management with Distributed Decision Making for Content-Centric Networks,” in *Proceedings of 7th Conference on Next Generation Internet (NGI 2011)*, June 2011.

[25] S. Arianfar, P. Nikander, and J. Ott, “On Content-centric Router Design and Implications,” in *Proceedings of the Re-Architecting the Internet Workshop (ReArch 2010)*, pp. 1–6, November 2010.

[26] P. Eugster, P. Felber, R. Guerraoui, and A. Kermarrec, “The Many Faces of Publish/Subscribe,” *ACM Computing Surveys*, vol. 35, pp. 114–131, June 2003.

[27] H. Hwang, S. Ata, K. Yamamoto, K. Inoue, and M. Murata, “A New TCAM Architecture for Managing ACL in Routers,” *IEICE Transactions on Communications*, vol. E93-B, pp. 3004–3012, November 2010.

[28] H. Hwang, K. Yamamoto, S. Ata, K. Inoue, and M. Murata, “Minimization of ACL Storage by Adding Minimal Hardware of Range Matching and Logical Gates to TCAM,” in *Proceedings of the IEEE International Conference on High Performance Switching and Routing (HPSR 2008)*, pp. 116–122, May 2008.

[29] H. Hwang, K. Yamamoto, S. Ata, K. Inoue, and M. Murata, “Efficient Management of Access Control List by Combining Prefix Expansion and Range Matching Devices,” *Technical Report of IEICE (IN2007-105)*, vol. 107, pp. 37–42, December 2007. (in Japanese).

[30] S. Ata, H. Hwang, K. Yamamoto, K. Inoue, and M. Murata, “Management of Routing Table in TCAM for Reducing Cost and Power Consumption,” *Technical Report of IEICE (NS2007-120)*, vol. 107, pp. 7–12, January 2008. (in Japanese).

[31] H. Hwang, S. Ata, and M. Murata, “Resource Name-based Routing in the Network Layer,” *submitted to* Journal of Network and Systems Management (JNSM), June 2011.

[32] H. Hwang, S. Ata, and M. Murata, “A Feasibility Evaluation on Name-based Routing,” in *Proceedings of the 9th IEEE International Workshop on IP Operations and Management (IPOM 2009)*, pp. 130–142, October 2009.

[33] H. Hwang, S. Ata, and M. Murata, “The Impact of FQDN Database Updates on Name-based Routing Architecture,” in *Proceedings of the 5th IEEE/IFIP International Workshop on Broadband Convergence Networks (BcN 2010)*, pp. 16–21, April 2010.

[34] H. Hwang, S. Ata, and M. Murata, “Frequency-aware Reconstruction of Forwarding Tables in Name-based Routing,” in *Proceedings of the 5th International Conference on Future Internet Techonologies (CFI 2010)*, pp. 45–50, June 2010.

[35] H. Hwang, S. Ata, and M. Murata, “A Feasibility Analysis of Name-based Routing by Routers,” *Technical Report of IEICE (IN2008-178)*, vol. 108, pp. 273–278, March 2009. (in Japanese).

[36] H. Hwang, S. Ata, and M. Murata, “Mapping between Logical and Physical Topologies in Name-based Routing,” *Technical Report of IEICE (IN2009-167)*, vol. 109, pp. 139–144, March 2010. (in Japanese).

[37] H. Hwang, S. Ata, K. Inoue, and M. Murata, “A New Memory Architecture for Realizing Name Lookup Tables in Content-centric Networks,” *submitted to* Elsevier Computer Networks, October 2011.

[38] H. Hwang, S. Ata, and M. Murata, “Realization of Name Lookup Table in Routers Towards Content-centric Networks,” in *Proceedings of the 7th International Conference on Network and Service Management (CNSM 2011)*, October 2011.

[39] H. Hwang, S. Ata, and M. Murata, “‘Feasibility of Name Lookup Table in Routers for Realizing Content-centric Networks,” *Technical Report of IEICE (IN2010-149)*, vol. 110, pp. 31–36, March 2011. (in Japanese).

[40] H. Che, Z. Wang, K. Zheng, and B. Liu, “DRES: Dynamic Range Encoding Scheme for TCAM Coprocessors,” *IEEE Transactions on Computers*, vol. 57, pp. 902–915, July 2008.

[41] Q. Dong, S. Banerjee, J. Wang, D. Agrawal, and A. Shukla, “Packet Classifiers in Ternary CAMs Can be Smaller,” in *Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS 2006)*, pp. 311–322, 2006.

[42] K. Lakshminarayanan, , Rangarajan, and S. Venkatachary, “Algorithms for Advanced Packet Classification with Ternary CAMs,” in *Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM 2005)*, pp. 193–204, 2005.

[43] M. A. Ross, S.-D. Chen, and A. V. Bechtolsheim, “Logical Operation Unit for Packet Processing, US Patent 6,658,002,” December 2003.

[44] NetLogic, “Range Encoding Engine (REE).” `http://www.netlogicmicro.com/4-news/pr/2008/08-07-21.htm`.

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

[46] S. Deering and D. Cheriton, “Multicast Routing in Datagram Internetworks and Extended LANs,” *ACM Transactions on Computer Systems (TOCS)*, vol. 8, pp. 85–110, May 1990.

[47] Y. Zhu, M. Chen, and A. Nakao, “CONIC: Content-Oriented Network with Indexed Caching,” in *Proceedings of the 13th IEEE Global Internet Symposium (GI 2010)*, pp. 1–6, March 2010.

[48] K. Tesink and R. Fox, “RFC 4152: A Uniform Resource Name (URN),” August 2005.

[49] D. Reed and D. McAlpin, “Extensible Resource Identifier (XRI) Syntax V2.0.” `http://www.oasis-open.org/committees/download.php/15376#_Toc117301832`, November 2005.

[50] Life Sciences Identifiers (LSID). `http://lsids.sourceforge.net`.

[51] Digital Object Identifier (DOI). `http://www.doi.org`.

[52] T. Berners-Lee, R. Fielding, and L. Masinter, “RFC 3986: Uniform Resource Identifier (URI): Generic Syntax,” January 2005.

[53] C. Kozierok, *The TCP/IP Guide: A Comprehensive, Illustrated Internet Protocols Reference*. No Starch Press, 2005.

[54] L. Garcés-Erice, E. W. Biersack, P. A. Felber, K. W. Ross, and G. Urvoy-keller, “Hierarchical Peer-to-Peer Systems,” in *Proceedings of Euro-Par*, pp. 643–657, August 2003.

[55] 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.

[56] J. Lian, S. Naik, and G. B. Agnew, “Optimal Solution of Total Routing Table Size for Hierarchical Networks,” in *Proceedings of the 9th IEEE Symposium on Computers and Communications (ISCC 2004)*, pp. 834–839, June 2004.

[57] Abilene Network. `http://www.internet2.edu/network`.

[58] 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 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM 2004)*, pp. 3–14, August 2004.

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

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

[61] Y. Sato, Y. Toji, S. Ata, and I. Oka, “Design of Flexible Layer-3 Routing Protocol to Support Variable-Length Address,” in *Proceedings of IADIS International Conference WWW/Internet 2009*, pp. 267–272, November 2009.

[62] 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,” pp. 79–84, August 2008.

[63] P. Mockapetris, “RFC 1035: Domain names - Implementation and Specification,” November 1987.

[64] Z. Xu, C. Tang, and Z. Zhang, “Building Topology-aware Overlays using Global Soft-state,” in *Proceedings of International Conference on Distributed Computing Systems (ICDCS 2003)*, vol. 23, pp. 500–508, May 2003.

[65] S.-H. Yook, H. Jeong, and A.-L. Barabási, “Modeling the Internet’s Large-scale Topology,” in *Proceedings of the National Academy of Sciences of the United States of America (PNAS 2002)*, pp. 13382–13386, October 2002.

[66] 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.

[67] R. Gummadi, N. Kothari, Y. Kim, R. Govindan, B. Karp, and S. Shenker, “Reduced State Routing in the Internet,” November 2004.

[68] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," vol. 31, pp. 149–160, August 2001.

[69] S. Zoels, Z. Despotovic, and W. Kellerer, "On Hierarchical DHT Systems – An Analytical Approach for Optimal Designs," *Computer Communications*, vol. 31, pp. 576–590, February 2008.

[70] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web Caching and Zipf-like Distributions: Evidence and Implications," in *Proceedings of IEEE International Conference on Computer Communications (INFOCOM 1999)*, vol. 1, pp. 126–134, March 1999.

[71] J. Rajahalme, M. Särelä, K. Visala, and J. Riihijärvi, "On Name-based Inter-domain Routing," *Computer Networks*, vol. 55, pp. 975–986, March 2011.

[72] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, and I. Stoica, "ROFL: Routing on Flat Labels," *ACM SIGCOMM Computer Communication Review*, vol. 36, pp. 363–374, September 2006.

[73] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, C. Wells, and B. Zhao, "Oceanstore: An Architecture for Global-scale Persistent Storage," *ACM SIGARCH Computer Architecture News*, vol. 28, pp. 190–201, December 2000.

[74] W. Jiang and V. K. Prasanna, "Field-split Parallel Architecture for High Performance Multi-match Packet Classification using FPGAs," in *Proceedings of the 21st Annual Symposium on Parallelism in Algorithms and Architectures (SPAA 2009)*, pp. 188–196, August 2009.

[75] Hashtagsjp. `http://hashtagsjp.appspot.com`.

[76] Twitter. `http://twitter.com`.

[77] Y. Ding, E. Jacob, M. Fried, I. Toma, E. Yan, S. Foo, and S. Milojević, "Upper Tag Ontology for Integrating Social Tagging Data," *Journal of the American Society for Information Science and Technology*, vol. 61, pp. 505–521, March 2010.

[78] Renesas Electronics Corporation, Memory Products. `http://www.renesas.com/prod/memory/`.

[79] D.Farinacci and Y.Cai, "RFC 4610: Anycast-RP Using Protocol Independent Multicast (PIM)," August 2006.

[80] H. Zhang, S. Ganguly, S. Bhatnagar, R. Izmailov, and A. Sharma, "Optimal Load Balancing in Publish/Subscribe Broker Networks Using Active Workload Management," in *Proceedings of IEEE International Conference on Communications (ICC 2008)*, pp. 5892–5896, May 2008.

[81] H. Jafarpour, S. Mehrotra, and N. Venkatasubramanian, "A Fast and Robust Content-based Publish/subscribe Architecture," in *Proceedings of 7th IEEE International Symposium on Network Computing and Applications (NCA 2008)*, pp. 52–59, July 2008.

[82] A. Gupta, O. Sahin, D. Agrawal, and A. Abbadi, "Meghdoot: Content-based Publish/subscribe over P2P Networks," in *Proceedings of the 5th ACM/IFIP/USENIX International Conference on Middleware*, pp. 254–273, October 2004.

[83] International Technology Roadmap for Semiconductors, "2009 Exective Summary." `http://www.itrs.net/Links/2009ITRS/2009Chapters_2009Tables/2009_ExecSum.pdf`.