A Scalable Name Resolution System for Information Centric Networking

Walid Elbreiki, Suhaidi Hassan and Adib Habbal
InterNetWorks Research Lab, School of Computing, Universiti Utara Malaysia, 06010 UUM Sintok, Kedah, Malaysia.
walid@internetworks.my

***Abstract*—Information Centric Networking (ICN) is a new paradigm, aimed at shifting to the future Internet from host centric to a content centric approach. ICN focuses on retrieval and dissemination of information between pairwise communications of hosts. Information are organized in the form of Information Objects (IO), known as Named Data Objects (NDO). These NDO are location independent. Objects in ICN are stored in the system overlay; popularly known as Name Resolution System (NRS). NDOs are requested by the Subscribers in the network to get the needed information from the Publishers, through NRS. Thus, the NRS is responsible in forwarding the interest packets based on the names of NDOs. This application of ICN depends on the scalability of the NRS. To design NRS, the most significant issue is scalability due to the ever-increasing number of NDOs. This paper aims to present the issues, by proposing balanced binary tree data structure to organize and store the NDOs. The methodology proposed in this work is thus; for every new insertion in the tree, a Balance Factor (BF) is computed to balance the height of left and right sub-tree. According to our investigation, balanced binary tree provides less searching time when compared to the Distributed Hash Table (DHT) approach. Simulation results show that End-to-End delay decreases by increasing the throughput in the network.**

***Index Terms*—Information Centric Networks (ICN); Balanced Binary Tree (BBT); Named Data Objects (NDO); Name Resolution System (NRS).**

I. INTRODUCTION

Information-centric Networking (ICN) is an emerging architectural approach for the future Internet. It has the potentials of solving a variety of issues in the existing Internet. Many ICN approaches have been proposed to handle these problems which lead towards the future Internet actualization such as A Data-oriented Network Architecture (DONA) [1], Content-centric Networking (CCN) [2], Network of Information (NetInf) [3], Publish-Subscribe Internet Routing Paradigm (PSIRP) [4] Mobility First [5] and Named Data Networking (NDN) [6]. Several ICN approaches comprise of name resolution system, which translates object IDs into network addresses. Constructing NRS for approximate 10^{16} IDs is really challenging with the notion of scalability, efficient network utilization and low latency.

ICN is a networking paradigm which is proposed to solve several problems such as, inefficient resource utilization, distributed denial of service attacks, inadequate security and flash crowds of Internet architecture. The ICN approaches have experienced tremendous developments of information on the Internet, with increasing demands for data access. ICN supports multi access and mobility, multicast, broadcast and anycast. Here, connectivity is irregular and in-network storage and end host interactivity is capitalized. Data is independent of application, location and storage which enables replication and caching. This nature of ICN improves the scalability with respect to bandwidth and information demands and efficiency in communication era.

The main aim of ICN is to present as form of retrieving the data based on location, either by name based routing or by name resolution. In ICN, information is identified with the help of location independent identifiers. Generally, ICN targets on infrastructure which enables in-network caching to distribute the content for scalability, security and cost efficiency objectives. This is a receiver-client driven model for getting objects which are of interest. ICN supports transparency, interactivity and node oriented services, thereby providing Peer-to-Peer (P2P) connectivity within the network.

In this paper, a mechanism that adapts the balanced binary tree data structure is proposed in order to store and manage the NDOs in a systematic order [7] [8] [9]. The mechanism is able to minimize the average end-to-end delay thus increasing the total throughput in the network. Since the mechanism is based on a balanced binary tree structure, the searching time is minimized by half on every iteration. The paper reports some simulation results and evaluation graphs.

The remainder of this paper is organized as follows: Section II presents the related work specifically on distributed hash table, in which Chord and Pastry, and bloom filter in NRS are briefed. Section III proposes our mechanism BBT based NDO storage mechanism. This Section also comprises of theoretical description and system model of the proposed mechanism. Section IV highlights performance evaluation of the simulation results with experimental setup and results and discussion. Finally, Section V concludes the paper.

II. RELATED WORK

The basic task of NRS is to map object names to its locators, which enables to reach the information about corresponding object, commonly known as IP addresses. The NRS till now mentioned are either based on DHT and based on BF. In this section three NRS structures with their drawbacks are briefed as follows:

A. Bloom Filter in Name Resolution System (BF-NRS)

BF-NRS maintains and resolves binding between locators and names. It takes input as names and generates locator sets as output. The BF-NRS is based on a flat naming system which is locator independent. The Flat naming system is the simplest name allocation system with high flexibility, most scalable and most advantageous in terms of privacy and persistency [10]. One of the advantages of the BF-NRS is its constant time for insertion and search operation. This is due to the non-dependent on the individual names in the set and efficiently supports for the union of bloom filters and group of hash functions implemented by bitwise OR operation.

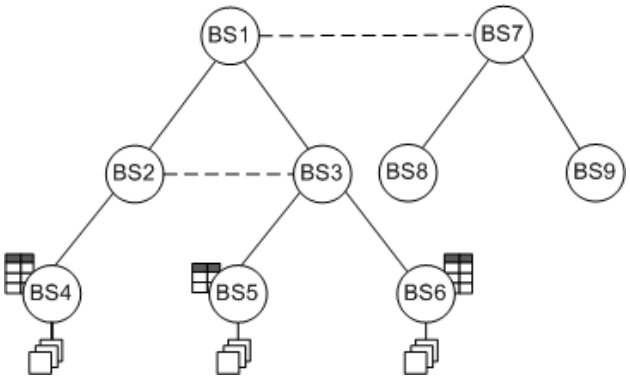

Figure 1: Bloom Filter

BF-NRS is constructed hierarchically by combining a network of BF-NRS servers. This comprises of a forest with several disjoint trees that are defined by parent-child relationship. Example of BF-NRS with name lookup table as a tree is shown in Figure 1. In the figure, there are 8 BF-NRS servers $\{BS1, BS2 \ldots BS7\}$ where between servers $BS1$ and $BS2$ an establishment of parent child relationship exist and $BS2$ and $BS3$ peering relationship. The peering relationship reduces the overhead for the top most servers by serving better performance. The inserted data in the tree are as shown in the example. This is a forest representation of BF-NRS with two trees. There is a peering relationship between two trees, with the root of the tree.

Name lookup table stores the binding of source locators and names for all the names which are already published by the publishers. There may be more than one source locator for information. The input for name lookup is named and the output it produces is locator set. BF-NRS servers announce aggregated form of names among peers. BF-NRS servers store the $n + m + 1$ number of bloom filters, where m is the number of peers and n is the number of child servers.

The main drawback of bloom filter is its quality of false positive and deletion of member is not possible. At the time of a deletion operation, reconstruction, operation is followed up in bloom filters which is expensive. This reconstruction is known as updating the bloom filter because of its inability to handle deletion operation itself.

B. Distributed Hash Table in NRS

The DHT based NRS is hierarchical, distributed and topologically embedded by the underlying network. Like bloom filters, DHT stores binding between object IDs and locators of object copies. The main design issue of DHT is its low latency, scalability, locality, agility, scoping and network utilization [11].

a. Chord Protocol

Mainly there are two types of DHTs viz: Chord and Pastry [12]. The main features of Chord are simple, provable correctness and performance. The main issues handled by Chord are load balancing through implementation of distributed hash function, decentralization of key management by storing the keys in many nodes; availability through automatically updating all the internal tables during node arrivals. It offers Scalability with reduced number of lookups even then the network is extremely large and flexible naming. These features and advanced capabilities make the chord suitable for implementing co-operative mirroring, distributed indexing, large scale combinatorial search and time shared storage.

The implementation of the hash function is by mapping keys to nodes. The assigning of the keys to nodes takes through consistent hashing, which has several properties compared to other techniques [13]. Figure 2 shows the Chord ring with 10 nodes where each node has five keys [14]. When N^{th} node joins or leaves the network, load balancing is managed by distributing the key to all nodes and an $O(1/N)$ fraction of keys are moved to a different location. Chord does not allow every node to know about every other node rather than small amount of routing information about the nodes. The main drawback of DHT is that all servers are linked in the form of circular linked list and the connections are stored in appropriate server other than servers. This causes serious trust problem with respect to authority issue and lookup messages are propagated through long paths.

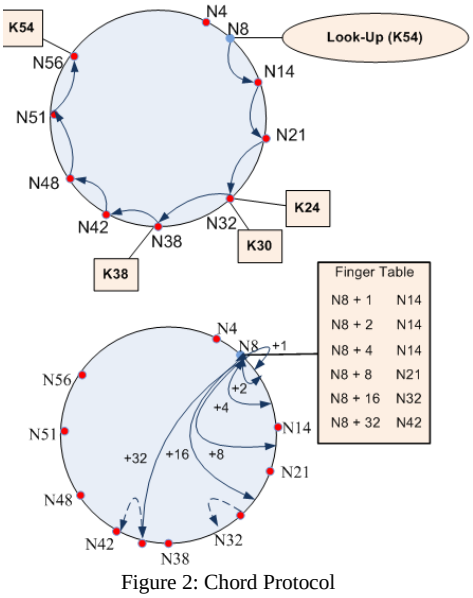

Figure 2: Chord Protocol

b. Pastry protocol

Pastry [15] is decentralized object location, scalable and routing mechanism for large scale peer-to-peer system [3]. This is the overlay network where each node router interacts with local instances of applications. Every node in Pastry is assigned a 128 bit long random unique identifier. This ID includes

position of a node in a circular nodeID space ranging from 0 to 2^{128} -1 and is generated randomly. These nodeID are uniformly distributed over the space. The given message and a key can efficiently help to route the message to the destination. All the pastry nodes are aware of all the adjacent nodes in nodeID space and new arrivals, failures and recoveries are informed of the applications.

Figure 3 presents the Pastry protocol implemented using DHT. Pastry minimizes the message travel distance by scalar proximity metric. Routing tables have the key of closest node and candidate node enabling shortest route. Pastry is completely scalable, decentralized and self-organizing by supporting mobility, caching and route convergence services. The advantage of Pastry is, it automatically adapts to the node arrival changes in the network. The protocol has the same drawback as the Chord.

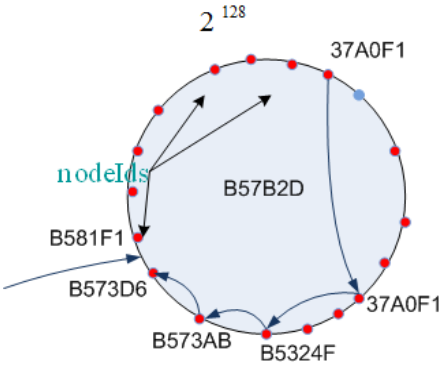

Figure 3: Pastry Protocol

III. BBT-Based NDO Storage Mechanism (BBT-NDO Storage)

In this paper, we constructed a BBT-NDO storage mechanism for NRS which exhibits Balanced Binary Tree data structure. The network of BBT-NDO storage is defined by a balanced binary tree and parent-child relationship with individual NDOs and balancing operation of tree with each insertion of the NDO.

We divided the network into several domains as a prerequisite for this mechanism. Each domain has its own NRS to store the NDOs. All the resolution domains are interconnected. In our work we consider one NRS for better understanding and set of operations on it.

A. Theoretical description of BBT-NDO

Balanced Binary Tree data structure conceived to efficiently perform all data structure, basic operations on large data sets [7]. To ease the comprehension of the notions presented in the system model, a summary of all adopted symbols is reported in Table 1.

In general, BBT with *n* number of NDOs that can be used to map IDs of the NDO, belongs to a tree data set T. On this tree structured network, a basic function is defined for NDO mapping or searching. The aim is to insert NDO into BBT, and to test whether NDOs are members of the tree.

Table 1
Adopted set of symbols for BBT

Symbol	Description
N	Data set represented as NDO set
H	Height of the tree
TL	Left sub tree
TR	Right sub tree
hR	Height of right sub tree
hL	Height of left sub tree
T	Balanced binary tree data set
X	New NDO to be inserted
Y	Parent NDO
Z	Grand parent
K	Searching NDO

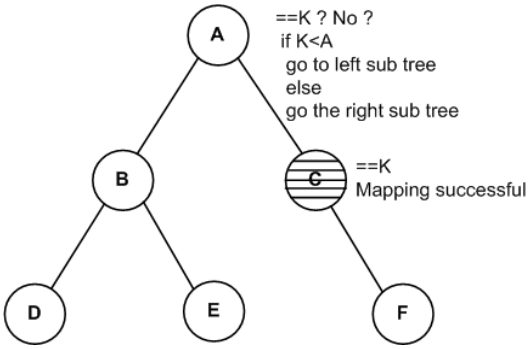

Figure 4: NDO Mapping

NDO mapping operation is handled by these consecutive steps;

1. The mapping or searching NDO *K* is as depicted on Figure 4, mapped with root NDO.
2. If *K* mapped to root.
3. Return.
4. else
   The NDO is checked whether the NDO value greater or lesser than root.
5. If smaller than the root, then
   Maps the element in the left sub tree.
6. else
   Maps the NDO in the right sub tree.
7. Repeat until the map is successful.

B. System model for BBT-NDO storage mechanism

Balanced binary search tree is a self-balancing tree data structure, which is also known an AVL tree. AVL tree is invented by three researchers Georgy Adelson, Velsky and Evgenis Landis. This binary search tree with balanced height, height (level) of the tree is balanced from both sides. That is the height of the two sub tree are equal or may differ by level at most one level. After inserting a new node, if the two levels are differed by more than one level, re-balancing is performed to restore the property of balancing. The average and worst case time complexity is $O(logn)$, where *n* is the number of nodes in the tree.

For example the balanced binary search tree, for its every interval node v of T, the height of the children of v can differ by at most 1.

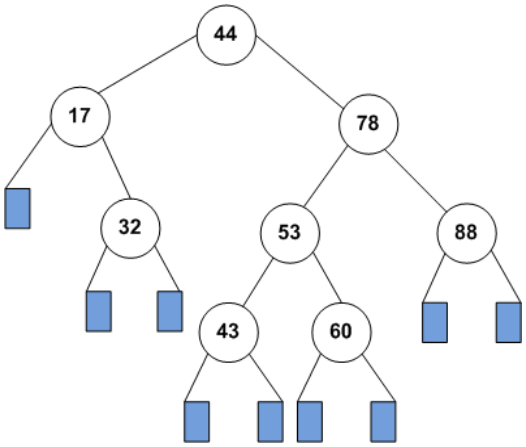

Figure 5: Balanced Binary Tree

The height of a BBT storing n keys and searching time is $O(logn)$.

Assume a setup with minimum number of interval nodes of a BBT of height h: $n(h)$

We know that $n(1) = 1$ and $n(2) = 2$.

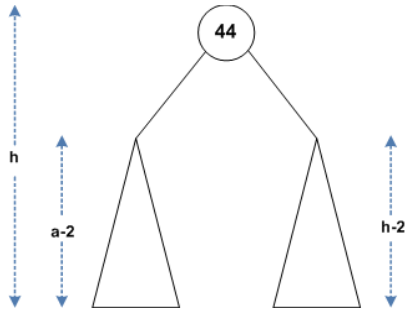

Figure 6: Height of BBT

For $n > 3$ a BBT of height h with n(h) minimal contains the root node, one BBT sub tree of height $h - 1$ and other sub tree of height $h - 2$.

If $$n(h) = 1 + n(h-1) + n(h-2) \quad (1)$$

Such that,

$$n(h-1) > n(h-1) \quad (2)$$

From Equation (2) we assume that,

$$n(h) > 2n(h-2) \quad (3)$$

$$n(h) > 2n(h-2) \quad (4)$$
$$n(h) > 4n(h-4) \Rightarrow n(h) > 2^2 n(h-2*2) \quad (5)$$
$$n(h) > 2^i n(h-2^i) \quad (6)$$

Solving the base case,

$$n(h) > 2^{\frac{h}{2-1}} \quad (7)$$

Taking log both sides of the Equation (7)

$$h < 2logn(h) + 2 \quad (8)$$

So the height of BBT is $O(log\ n)$.

a. Balance Factor

When the new node is inserted into the tree, the tree need to be balanced. The balancing action is carried out by the balance factor.

$$Balance\ factor = height\ of\ left\ subtree - height\ of\ right\ subtree$$

The binary search tree is height balanced, if T is a non-empty binary search tree with TR and TL as right and left sub trees respectively. Then T is height balanced if and only if,

1. TL and TR are height balanced.
2. $|hR - hL| <= 1$where hR and hL are the heights of TR and TL respectviely.

The balancing factor of a binary tree is the difference in the heights of its two sub trees$(hR - hL)$. The balance factor of a height balanced binary search tree may take on one of the values $-1, 0, +1$.

b. Insertion of NDO in BBT

When a new NDO is published by a publisher, the NDO is stored in NRS in BBT structure. The insert operation is followed by tree balance operation. If the balance factor is $\pm 1, 0$.

Inserting NDO into NRS in BBT structure T changes the height of the sub tree in T. If after insertion becomes unbalanced, from the sub tree rooted at its child y.

$$height(y) = height(sibling)(y)) + 2$$

Now we need to re-balance the tree from its sub tree y, either by right rotation or by left rotation.

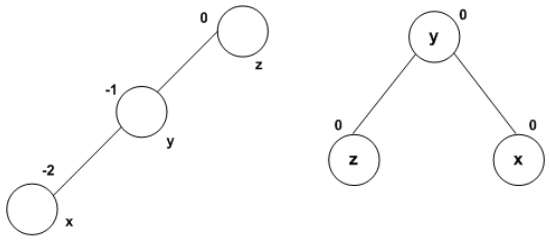

Figure 7: Right Rotation

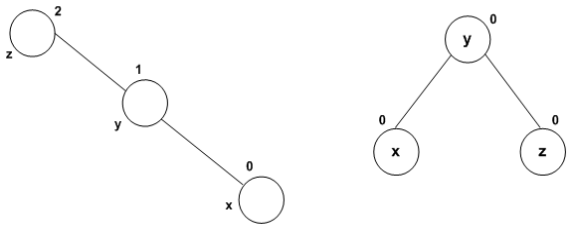

Figure 8: Left Rotation

c. Pseudo code to re-balance

Input: A node x as parent y and grandparent z

Output: Tree is re-balanced by a rotation involving x, y and z

Method: Let T_1, T_2, T_3 and T_4 be an in order form of the four sub tree of x, y, and z not rooted at x, y and z.

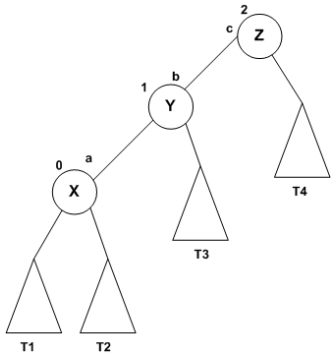

Figure 9: Unbalanced Binary Tree

Replace the sub tree rooted at z with a new sub tree rooted at y.

Let x be the left child of y and T_1, T_2 be the left and right sub trees of x respectively

Let z be the right child of y and T_3, T_4 be the left and right sub trees of z respectively.

Deletion of a node from BBT

Deletion operation makes the BBT unbalanced.

Let x be the unbalanced subtree after deleting a node from the BBT. And let z be the subtree with larger height. Then again re-balance is a function called to balance the tree. Each re-balancing function is followed by a check balance function of the tree.

C. Performance Evaluation

In this paper, OMNeT++ simulation environment with INET framework and ICNSim has been used for building simulation models. The simulation tool is well designed, widely-used, modular system; OMNeT++ is an open source network simulator that has been made available for teaching, research and development purposes [16]. The simulator has been developed using Python and C++ along with scripting capability in a modular fashion as a set of libraries. These libraries can be combined together with other external software libraries for data analysis for better presentations of the outcomes [17].

a. Experimental Setup

Experiment environment has been created in the latest OMNeT++ version 4.6, with INET 2.6 framework along with ICNSim, and the GCC compiler 4.9 running on Ubnutu 14.04 LTS. For clarity purposes, Table 2 shows the simulation parameters used in this research.

b. Results and Discussion

Figure 10 illustrates the total throughput of using distributed hash table and balanced binary tree in bits/sec. The simulation results were collected on power law topology with varying numbers of publishers. Initially the total throughput is equal for both the mechanisms. The distributed hash throughput was fluctuating in between. However, balanced binary tree throughput was gradually increasing as the number of subscribers increased.

Table 2
Simulation Configuration

Parameter	Description
Number of NRS	2
Number of Publishers	10-50
Number of Subscribers	10-50
Number of contents	50.000
Size of the content	The size of the content is 1 GB
Publisher StartTime	0s
Publisher pubInterval	3s
Publisher datarate	3Mbps
Topologies Used	Power Law Topology
Number of Host	Each Router Two Host
Testing Tool	OMNeT++ 4.6
Testing Application	BasePSApp Application
Simulation Time	200 sec

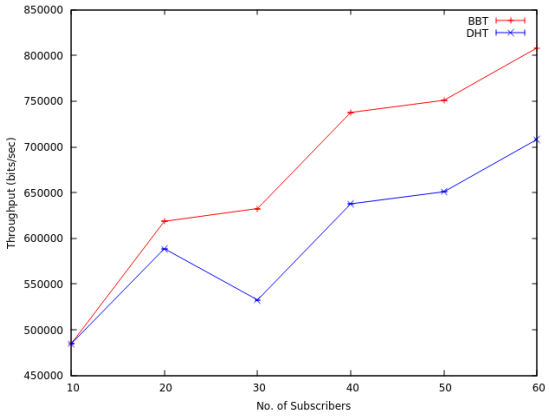

Figure 10: Throughput of distributed hash table and balanced binary tree in Mbps

Additionally, Figure 11 illustrates the average end-to-end delay differences of distributed hash table and balanced binary tree mechanism for different number of subscribers. Although both mechanisms showed a gradual rise in the delay as the numbers of subscribers increased, distributed hash table mechanism seemed to be higher than a balanced binary tree. This is because BBT searching time is less than DHT. In brief, balanced binary tree achieves lower delay than distributed hash tables.

IV. CONCLUSION

NRS in ICN is considered as an adequate approach to store NDOs. NRS is beneficial to all the applications of NDOs such as search, delete and insert by providing the facility to name resolution and name-based routing. In this work, the proposed mechanism adopts Balanced Binary Tree (BBT) for overlay network to store NDOs. Simulation results show that the BBT based NDO mechanism can show better throughput and delay with respect to the number of subscribers' interests. Our simulation results are evaluated with DHT based networks and

the graph is shown in the previous section of this paper. The BBT based NDO mechanism, provides a scalable solution by accommodating 30 levels of the tree while keeping the end-to-end delay low.

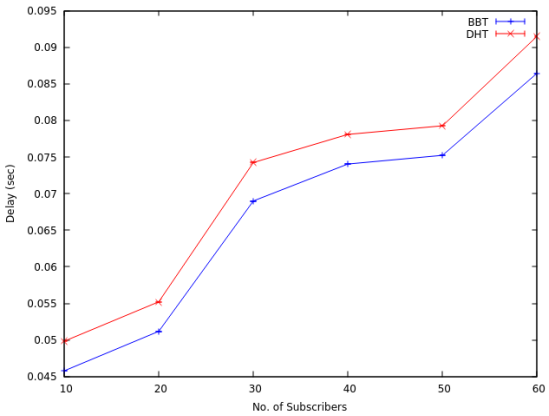

Figure 11: End-to-end delay differences of distributed hash table and balanced binary tree

REFERENCES

[1] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker and I. Stoica, "A Data-oriented (and Beyond) Network Architecture," *ACM*, vol. 37, pp. 181--192, 2007.

[2] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs and R. L. Braynard, "Networking Named Content," *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, pp. 1--12, 2009.

[3] C. Dannewitz, D. Kutscher, B. Ohlman, S. Farrell, B. Ahlgren and H. Karl, "Network of Information (NetInf) - An Information-centric Networking Architecture," *Computer Communications*, vol. 36, pp. 721-735, 2013.

[4] P. Project, "Publish/Subscriber (PURSUIT)," 2010. [Online]. Available: http://www.fp7-pursuit.eu/PursuitWeb/.

[5] I. Seskar, K. Nagaraja, S. Nelson and D. Raychaudhuri, "Mobilityfirst future internet architecture project," *Proceedings of the 7th Asian Internet Engineering Conference*, pp. 1--3, 2011.

[6] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. D. Thornton, D. K. Smetters, B. Zhang, G. Tsudik, D. Massey, C. Papadopoulos and others, "Named data networking (ndn) project," *Relatrio Tcnico NDN-0001, Xerox Palo Alto Research Center-PARC*, 2010.

[7] H. V. Jagadish, B. C. Ooi and Q. H. Vu, "Baton: A balanced tree structure for peer-to-peer networks," 2005.

[8] H. Takamizawa, K. Saji and M. Aritsugi, "A replica management protocol in a binary balanced tree structure-based P2P network," *Journal of Computers*, vol. 4, pp. 631--640, 2009.

[9] Manku and G. Singh, "Balanced Binary Trees for ID Management and Load Balance in Distributed Hash Tables," *Proceedings of the Twenty-third Annual ACM Symposium on Principles of Distributed Computing*, pp. 197--205, 2004.

[10] A. Ghodsi, T. Koponen, J. Rajahalme, P. Sarolahti and S. Shenker, "Naming in content-oriented architectures," in *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*, 2011.

[11] C. Dannewitz, M. DAmbrosio and V. Vercellone, "Hierarchical DHT based name resolution for information-centric networks," Vols. 36, no. 7, p. 736749, 2013.

[12] W. Elbreiki, S. Hassan, A. Habbal, M. Firdhous and M. Elshaikh, "A Comparative Study of Chord and Pastry for the Name Resolution System Implementation in Information Centric Networks," in *Conference: 4th International Conference on Internet Applications, Protocols and Services (NETAPPS2015)*, Cyberjaya, Kuala Lumpur, Malaysia, 2015.

[13] S. T. Visala, M. Ain and Kari, "The Publish/Subscribe Internet Routing Paradigm (PSIRP): Designing the Future Internet Architecture," in *Towards the Future Internet*, IOS Press, 2009, pp. 102-111.

[14] I. Stoica, R. Morris, D. L. Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," *SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pp. 149-160, 2001.

[15] A. Rowstron and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems," *IFIP/ACM International Conference on Distributed Systems Platforms*, pp. 329-350, 2001.

[16] A. Khan, S. Bilal and M. Othman, "A performance comparison of open source network simulators for wireless networks," 2012.

[17] K. Katsaros, G. Xylomenos and G. C. Polyzos, "MultiCache: An Overlay Architecture for Information-centric Networking," *Computer Networks*, vol. 55, pp. 936--947, 2011.

[18] J. Alpert and N. Hajaj, "Google we knew the web was big," 25 July 2008. [Online]. Available: https://googleblog.blogspot.my/2008/07/we-knew-web-was-big.html

[19] K. Katsaros, G. Xylomenos and G. C. Polyzos, "MultiCache: An Overlay Architecture for Information-centric Networking," *Comput. Netw.*, vol. 55, no. 4, pp. 936-947, #mar# 2011.