NDNBench: A Benchmark for Named Data Networking Lookup

Ting Zhang, Yi Wang, Tong Yang, Jianyuan Lu and Bin Liu*
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University, Beijing, China

Abstract**—**Content-centric Networking (CCN) and the later proposed Named Data Networking (NDN) have attracted wide attention in both academia and industry, as the clean slate future Internet architecture. Wire speed name lookup for packet forwarding is one of the most challenging tasks in CCN/NDN. As a promising technology, its feasibilities including reachable speed, scalability, and update performance are imperative to be deeply evaluated. However, CCN/NDN is currently on its initial stage and no actual network is deployed, which means no real name routing tables and NDN traffic are available. In order to fulfill performance comparisons among various innovative name lookup solutions and facilitate future name lookup researches, we present **NDNBench**, a publicly available platform for evaluation, comparison and experiments with different name lookup approaches. **NDNBench** can generate various Forwarding Information Bases (FIBs), traces with structure and size diversity to conduct the tests thoroughly by adjusting the parameters. **NDNBench** provides a simulation package tool with flexibility to evaluate various name lookup approaches. Furthermore, in order to verify the effectiveness of **NDNBench**, we benchmark some existing name lookup schemes and the results are very supportive. **NDNBench** has been applied to recent work and is publicly available at the following site: http://s-router.cs.tsinghua.edu.cn/~zhangting/.

I. INTRODUCTION

With the development of technology and popularization of information, Internet, whose original objective is in pursuit of hardware resource sharing by network interconnections, is undergoing drastic changes in its main function. Customers trend to less concern about the location where the content is originated but more about the availability, quality and security of the content. Named Data Networking (NDN) [1], proposed as a clean slate future network architecture, has attracted great attentions [1-3, 13-15]. NDN separates safety, accessibility and location of content from content itself. NDN operates based on the identity of contents and every distinct content/entity is referenced by a unique name. Consequently, this avoids the complex mapping between contents and their locations and will potentially improve the data retrieval efficiency of future Internet.

Name lookup of packets forwarding is taken as the foundation and one of the key technologies in NDN. When a request packet arrives at an NDN router, the name key will be abstracted from this packet header and sent to the search engine to execute longest prefix matching (LPM) against name prefixes in name table[1] to obtain the outgoing port(s). Then this packet will be forwarded to the corresponding port(s).

Name lookup in NDN forwarding plane is much more challenging than current IP lookup due to the following reasons: 1) *Content names in NDN are more complex than IP addresses.* An NDN name employs URL-like structure, and its length is variable (it could be composed of tens or even hundreds of characters) and no externally upper bound; 2) *NDN name table could be much larger than the existing IP lookup table*. It is anticipated that an NDN name table could swell to tens of millions of entries, even more, which will be orders of magnitude larger compared with an IP forwarding table; 3) *NDN name table update frequency is expected to be much higher than today's Internet*. This is because besides the regular network topology/policy changes, content publishing/deletion can also trigger the update of NDN FIB, which could be more frequent. The above changes make NDN name lookup a very tough task and thus it is quite imperative for name lookup to be studied deeply.

However, existing IP-based lookup approaches cannot be directly applied to NDN scenario to achieve high performance due to the following two reasons. First, NDN names are hierarchical and composed of a series of components, while IP addresses can match a prefix at any bit position. Second, IP addresses are fixed length and 32 memory accesses are required in worst case when performing IP lookup, while NDN name lengths are variable. Several NDN name lookup solutions [1-3, 13-15] have been proposed recently to address this issue. Our goal is to evaluate various proposed solutions' speed, memory occupation, scalability and update performance. However, CCN/NDN is in its initial stage of research, so practically no actual CCN/NDN network is deployed yet. Without real name route table and real-world NDN traffic, it is difficult to evaluate the above solutions on a reference basis. Moreover, without the public and fiducial reference FIBs, as well as corresponding traces (including name traces and update messages), the performance comparison among various approaches is far from convincing.

In pursuit of conquering these challenges, in this paper, we originally develop ***NDNBench***, a publicly available platform for evaluation, comparison and experiments with different lookup approaches. NDNBench consists of four tools: **seed FIB analyzer**, **FIB generator**, **name trace generator** and **updates generator**. In particular, our main contributions include the following.

1) According to our extensive experiments, we find that the characteristics of FIB and traces will greatly influence the performance of NDN name lookup. Then we identify and refine these characteristics and form the concrete and quantitive parameters. More importantly, these parameters inspire researchers to develop new name lookup approaches which exploit

**Corresponding author: liub@tsinghua.edu.cn.*
Others: { ting-zhang10, yiwang09, yang-t10, lu-jy11} @mails.tsinghua.edu.cn.
This work is supported by 863 project (2013AA013502), NSFC(61073171, 61202489), Tsinghua University Initiative Scientific Research Program (20121080068), the Specialized Research Fund for the Doctoral Program of Higher Education of China (20100002110051),
1 In this paper, we use two terms --FIB and name table-- interchangeably.

structure of FIB and trace to accelerate lookup or update and reduce memory requirements.

2) We design and implement a system to generate various FIBs, name traces and update messages. Two approaches are proposed to produce FIBs. Further, the system generates synthetic traces which contain a sequence of content names traces and update messages for a given NDN FIB, guided by the corresponding parameter settings. Users can also configure these parameters according to their requirements.

3) Given the complexity of NDN update compared with IP update scenario, we make a comprehensive analysis of the composition and behavior of NDN update, which can guide users to generate rational update messages.

To our best knowledge, this is the first effort to implement a benchmark for NDN name lookup and NDNBench has been applied to the recent work [2, 3, 13, 14]. We are committed to facilitate the name lookup solution and future NDN research agenda by quantifying the performance of lookup. We have released NDNBench in [12].

The remaining parts of this paper are organized as follows. Section II presents the workflow of NDNBench. All the sub-modules are presented in Section III. In Section IV, NDNBench is applied to evaluate the performance of different name lookup solutions. Section V surveys the related work and finally we conclude our work in Section VI.

II. The Working Flowchart of NDNBench

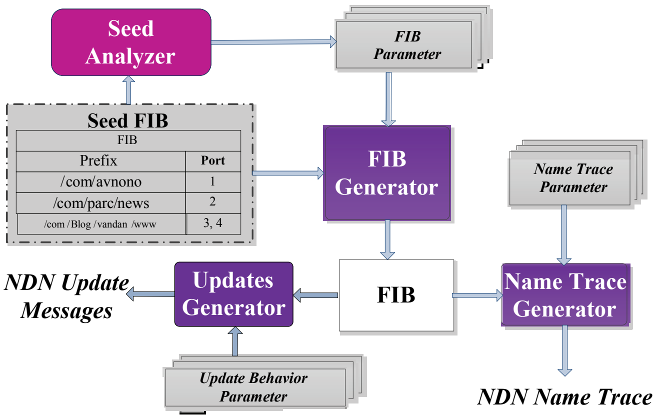

Fig. 1. The framework of NDNBench.

Figure 1 shows the working flowchart of NDNBench. We collected a 10-million name table from the current Internet with the URL-like style as a seed FIB. Based on this seed FIB, flexible FIBs with size and characteristic diversity can be generated. Seed analyzer takes the seed FIB as input, summarizes the characteristics of FIB that influence the performance of name lookup and dumps the information into the parameter set. These parameters are absorbed into FIB generator. Users can also modify these parameters according to their requirements. There are two approaches to generate synthetic FIB and users can choose either of them. Similarly, updates generator and name trace generator take FIB and corresponding parameter set as input. They produce traces composed of a sequence of name traces and update messages, to exercise the corresponding FIB. When conducting performance evaluation on different approaches, users can apply the same FIB and corresponding traces to them, enabling comparisons to verify the feasibility such as throughput, memory occupation and update performance. They can also evaluate the scalability of various approaches by adjusting the relevant parameters of FIB and traces.

Based on the above description, it can be concluded that parameters of FIB, name traces and update messages determine the structure and size diversity and will greatly influence the performance of name lookup, so they are the cornerstone of NDNBench.

III. Design of NDNBench Functional Modules

As described in Figure 1, NDNBench consists of several functional modules, which co-work in a flow-chart manner to finally produce the required FIBs as well as the testing traces. This section will detail the design of each functional module including seed analyzer, FIB generator, updates generator and name trace generator sequentially.

A. Seed Analyzer

This module focuses on identifying and understanding the characteristics of FIB which largely influence the performance of lookup approaches. The input of this module is seed FIB. By analyzing the seed FIB, seed analyzer extracts the relevant statistics and outputs them into the parameter set.

1) *Data Preparation*

Above of all, we introduce the structure of NDN FIB and content name contained in the name trace. An NDN name is hierarchically structured and composed of many components. Between every two adjacent components, there exists a delimiter which is not a part of the name, usually "/". For example, a webpage produced by PARC may have the name /com/parc/newsroom/new.html, where /com/parc is the website and /newsroom/new.html locates the resource on the server. Such hierarchical name structure enables LPM and aggregation, which are essential to the scalability of routing and forwarding mechanism, just the same as IP address. The basic aggregation unit is a component, while name lookup can be performed at character granularity.

Each FIB entry is composed of a name prefix and the corresponding outgoing port(s). Figure 1 shows a FIB which contains 3 entries. The content name extracted from packet header will probe the FIB to get the next hop port(s) by LPM. Name prefixes in FIB are converted from Internet domain names. Seed FIBs are obtained through the following steps:

Step I: Domain name collection. We implemented a web crawler to fetch URL in order to obtain domain names as many as possible. To achieve good geographic coverage, we installed our web crawlers in North America, Europe and Asia, respectively. The program has kept running since October 1st, 2011. Then these fetched URLs are transformed into domain names. Moreover, we downloaded some existing domain names from NameJet [4]. So far, we have obtained about **80,000,000 non-duplicate domain names** in total.

Step II: Name prefix transformation. These non-duplicate domain names are transformed into NDN name prefixes. For instance, www. google. com is transformed into /com/google/www.

Step III: Next hop port mapping. For each name prefix, we can get one or more IP addresses resolved by DNS. Each IP address performs LPM on an IP routing table downloaded from www. ripe.net and output a next hop port.

Step IV: FIB entries generation. NDNBench maps the name prefix to its corresponding next hop port(s). Finally, an FIB entry is synthesized.

In this paper, we generate a seed FIB by randomly selecting 10 million entries from 80 million candidate entries and conduct extensive experiments based on it.

2) *FIB Parameter Extraction*

It is especially important to identify the characteristics of FIB that affect the lookup performance. For instance, the average length of prefix varies among different FIBs, which probably has an impact on lookup. Furthermore, different approaches vary in the degree of sensitivity to specific parameters of FIB. Component-based approaches are sensitive to component number in each prefix, while not influenced by whether it is a letter or a digit in specific position. Therefore, it is necessary to leverage FIB with the same characteristics for the performance comparison among different solutions.

In this paper, NDN name lookup approaches are classified into two categories, namely, component-based and character-based approaches. For each category of approaches, the parameters related to lookup are listed respectively. Meanwhile, we thoroughly conduct data mining on seed FIB based on the following metrics and the results are presented in experimental section.

For component-based approaches, components are the atomic lookup unit. And such approaches are sensitive to the following metrics:

a) The distribution of component number in each prefix.

b) The distribution of component length in each level.

For character-based approaches, characters are the atomic lookup unit. They are sensitive to the following metrics:

a) Name prefix length distribution.

b) The frequency of each character appearing in each position.

We conduct modular design based on the fact that such parameters are independent of each other. When a new parameter that influences lookup performance is needed to be involved, it can be easily integrated into the seed analyzer, parameter set and FIB generator.

B. *FIB Generator*

FIB generator takes a parameter set as input and target FIB size to generate. Users can utilize the parameter set learned from seed FIB, and such parameters can be modified according to their requirements. We propose the following two approaches to generate new FIBs. The FIBs generated by these two approaches inherit the characteristics of parameter set.

a) We select specified number of entries from seed FIB based on the parameters. So all the entries of the new FIB come from seed FIB. In this case, target FIB size cannot be larger than seed FIB.

b) We generate new prefixes according to the parameters. This approach doesn't need to take the seed FIB as the input and will generate more flexible prefixes that don't exist in the seed FIB.

The final step in generating synthetic FIB is filtering redundant FIB entries. The naïve implementation is comparing each entry with others in the FIB, which makes execution time of the FIB generator prohibitively long. In order to accelerate this process, we construct component-trie, in which each node represents a component of name prefix. When generating a new prefix, we insert it into the component-trie to check whether this prefix already exists. In this way, the redundant entries are eliminated naturally.

C. *Updates Generator*

Update performance is one of the key metrics for routers' packet processing. NDN update should be highly addressed due to its complexity and difference with update in IP scenario. In this section, we will analyze the composition and behavior of NDN update. They are the rationale of our updates generator. Then we extract the parameters that cover the characteristics of NDN update behavior and implement the updates generator.

1) *Composition of NDN Update*

Usually, an update message can be of three forms: 1) Modification, 2) Insertion and 3) Deletion. When an update message arrives, routing table needs to be recomputed and packet forwarding will not proceed until the update message has been handled. This will even lead to packet loss when the lookup queue goes full during burst updates. In NDN scenario, this problem becomes aggravated. Besides network topology and policy changes, the following two situations may also incur the update of FIBs: a) the contents are published/deleted; b) the mobility of content provider. These are different from today's routers and make FIB update more frequent than today's Internet. So the update performance of routing lookup rises to be an important concern and should be evaluated.

2) *NDN Update Behavior Analysis*

Given no NDN network is deployed today, we analyze the NDN update behavior based on the current Internet. In order to understand network topology changes and routing policy modifications, we conduct data mining on real routing tables and update messages and get the following observations:

a) The incoming update messages do not always arrive uniformly and sometimes come as a bust. According to our experimental results, received update messages rate can reach nearly 35K/s in the peak while the average situation is only 1.43/s. Similar results can also be found in [16]. Thereby routers should implement name lookup at wire speed while keep the ability to handle the update messages in busts.

b) Most update messages only cause a small fraction of prefixes to be frequently updated. Our statistical results show that 4% of all the prefixes are affected by the updates while the others keep stable.

c) We use binary trie to construct routing table and find the above affected prefixes are mostly leaf nodes. The explanation of this result is that leaf nodes in binary trie usually cover a small range of IP addresses and represent the edge networks, which is unstable and suffer frequent update. Our results show that 86.7% updates occur in the leaf node.

When the content is published or deleted, name prefix will be inserted or deleted in the FIB. This procedure is similar to DNS changes for domain names. We try to quantify the content publish/deletion by analyzing DNS changes. Figures released in [17] show that there are only a few tens of top-level domains updated per second and thus the content publish/deletion contributes to a small proportion of updates.

How mobility causes FIB update can be explained via the following example. Name prefix */com/google/learning /network.pdf* and */com/google/learning/compiler.pdf* can be aggregated to */com/google/learning/*. Hierarchical name structure enables the prefix aggregation and eases the expansion of FIB size. But when content provider moves to the new network whose name prefix is */com/parc/resource/*, the router where this publisher located before needs to delete the stale prefix and */com/google/learning /compiler.pdf* cannot be aggregated any longer. The network where this publisher located now also needs to insert the prefix to the FIB. So mobility in NDN may have a serious effect on update.

3) *Updates Generator Implementation*

Guided by the above analysis, we design an updates generator to simulate update behavior, and generate a sequence of update messages for a given FIB based on the parameters. We list the parameters of update messages in our updates generator.

a) The Arrival Time Interval Distribution

The time interval of arriving update messages could follow different distributions. For example, the arrival of update messages can be consistent with the Poisson distribution. We integrate some frequently-used distribution into updates generator, including Gaussian distribution and Uniform distribution. Users can easily select one of them and input the corresponding parameter(s).

b) The Arrival Rate

The arrival rate refers to the number of update messages arrived per second, and it is an important factor that lookup algorithms should pay attention.

c) The Distribution of Updated Node

Take FIB organized in character-trie as an example, intuitively, updating the leaf node spends more computational cost than the updating intermediate nodes as it increases the memory access times in order to locate the corresponding node. This judgment is in accord with our experimental results.

D. Name Trace Generator

Only a few metrics such as memory requirement can be evaluated when we only use FIB to benchmark a particular name lookup algorithm. In order to conduct a thorough benchmark including throughput and scalability, name traces can be employed to probe the FIB. Name trace generator produces name traces which contain a sequence of name requests with respect to a given FIB. Similar to updates generator, name trace generator also takes FIB and parameter set as input. The key of the generator is to find out the crucial parameters of name trace such as matching ratio, which can heavily affect the performance of lookup algorithm. These parameters are listed as follows.

1) ***Trace Number.*** *It refers to the number of* name requests in the trace.

2) ***Hit Ratio.*** In NDN, some incoming name requests can't find the entry of the FIB to match, which indicates that not all the requests will probe one of the entries in FIB. In order to simulate this scenario, we define hit ratio, as $HR=Num_{used}/Num_{total}$, where HR is hit ratio; Num_{used} is number of requests to probe the FIB; Num_{total} is the total number of incoming name requests.

3) ***Hit Trace Distribution.*** In the process of LPM, different names will match prefixes with various lengths. Hit trace distribution indicates the length distribution exercised by the traces. For example, for the FIB in Figure 1, if all the names exercise the prefix "/com/avnono" which has two components, it will achieve better lookup speed compared with the situation when all the names exercise the prefix "/com/Blog/vandan /www" which has four components.

4) ***Trace Length Distribution.*** The performance of some implementations, for example, solution which batches a certain capacity of name requests and then transfers them to the lookup engine, is influenced by this metric. Given a certain memory, the longer the traces are, the less entries stored in the container will be.

5) ***Locality.*** In IP network, routing cache is employed to store the most frequently used prefixes and traffic to accelerate the lookup speed by reducing average memory access times. They can also be applied to NDN and we name them as FIB locality and name trace locality, respectively. Experiment shows the cache hit rate can easily achieve 96.2% with 256 cache entries by caching prefixes [11].

IV. Experiments on NDNBench

NDNBench has been used in recent work [2, 3, 13, 14]. In this section, we utilize it to evaluate NDN name lookup solutions in terms of lookup throughput, memory occupation, scalability, and update performance.

A. Experimental Settings

We implement and compare the lookup solutions on a commodity PC installed with a 6-core CPU (Xeon E5645 ×2), 2.4GHz clock frequency and DDR3 48GB (1333MHz) memory. The PC runs Linux operating system version 2.6.41.9-1.fc15.x8664.

B. Data Mininig on NDN FIB and Name Traces

We leverage the seed FIB mentioned in Section III. Based on 10M seed FIB, we generate other 9 FIBs which have the same characteristics with the seed FIB and the sizes are ranging from 1M to 9M, respectively. First we conduct data mining on the seed FIB and list the relevant parameters in Table I, II and Figure 2.

TABLE I. The Distribution of Component Number in Seed FIB

Cmp. Num in Per Prefix	2	3	4	5	6	>=7
Frequency	58.98%	34.85%	5.64%	0.46%	0.01%	0.06%

From the above results, we can conclude that nearly 94% of prefixes have two or three levels. In each level, components with three characters account for the majority. In addition more than 90% of the prefixes' length is between 10 and 29.

Name traces contain a sequence of name requests formed by concatenating a name prefix from the FIB with generated suffixes. We generate two types of name traces for each FIB. Parameters of each type are listed as follows.

TABLE II. PREFIX LENGTH DISTRIBUTION[2]

Length interval	1~9	10~15	16~19	20~29	>=30
Frequency	2.18%	30.09%	28.77%	34.00%	4.96%

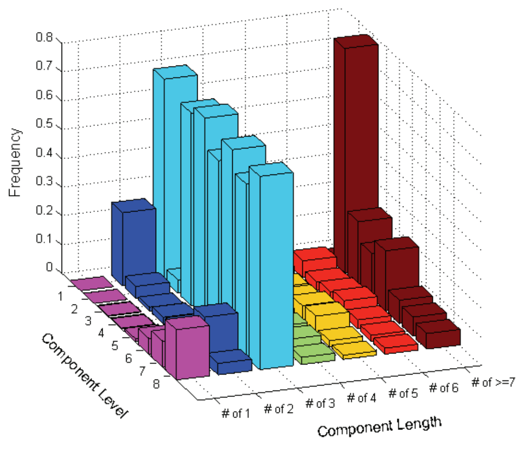

Fig. 2. The distribution of component length in each level in seed FIB.

a) Average Workload. The name requests in this set randomly exercise entries in the FIB. Every entry will have the same probability to be visited. We name this type as average workload.

b) Heavy Workload. For hit trace distribution, the name requests exercise the top 10% longest prefixes in the FIB. We name this type as heavy workload.

These two types both have the following parameters: trace number = 50M, hit ratio = 100%. And the length distribution of these two types is illustrated in Figure 3.

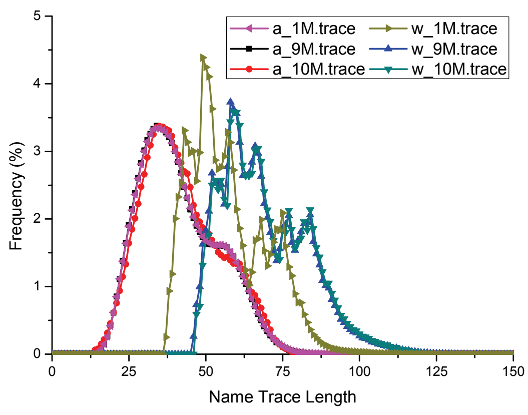

Fig. 3. Trace Length distribution[3].

[2] 1~9 means prefix length ranges from 1 to 9.

C. *Performance Evaluation on Name Lookup Approaches*

We implement character-trie and component-trie approaches and compare them with NCE [3]. It turns out that NCE is an effective name lookup solution since it encodes components to improve the throughput and reduce memory overhead.

1) *Throughput*

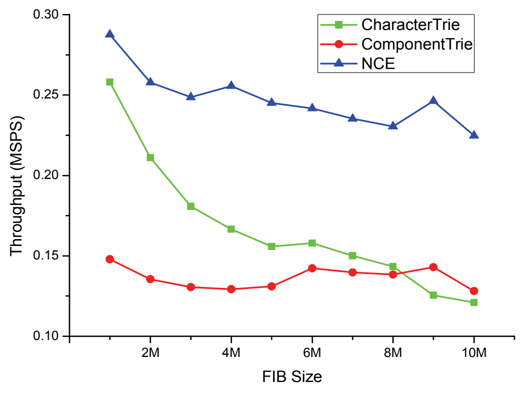

Fig. 4. The throughput of three approaches on different FIB sizes (average workload).

Figure 4 and Figure 5 illustrate the throughput of the three approaches with different sets of FIBs and corresponding name traces. We notice that the throughput of them exhibits downward trend along with the increasing size of FIBs. Especially, when the FIB size is 10M, the lookup speed of character-trie solution degrades dramatically due to the increasing memory access time.

On the other hand, compared with the average workload, the lookup speed of heavy workload degrades since more incoming names match longer prefixes. It indicates that the performance of lookup solution heavily relies on characteristics of the name traces, especially hit trace distribution. In addition, Figure 4 demonstrates NCE is superior in name lookup with better scalability than the other two.

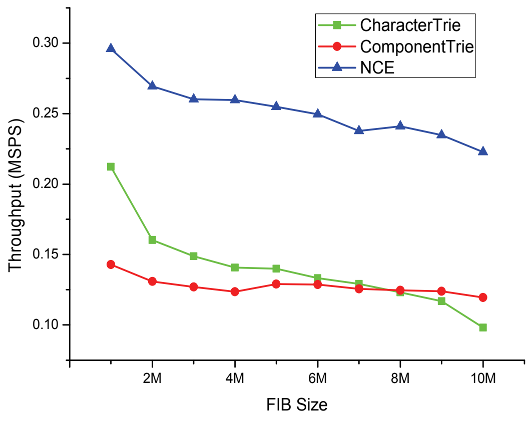

Fig. 5. The throughput of the approaches on different FIB sizes (heavy workload).

[3] Due to the lack of space, we only list six sets' results. "a_1M.trace" means the average workload to exercise 1M FIB and "w_1M.trace" means the heavy workload to exercise 1M FIB.

2) *Update Performance*

In order to evaluate the update performance of name lookup solutions accurately and objectively, we define the metric **Update Period** (UP), which represents time interval between the point when an update message is received and the point when update operation is accomplished. It indicates a router's sensitivity to the changes of network. We measure the performance of these solutions and Figure 6 shows insertion results.

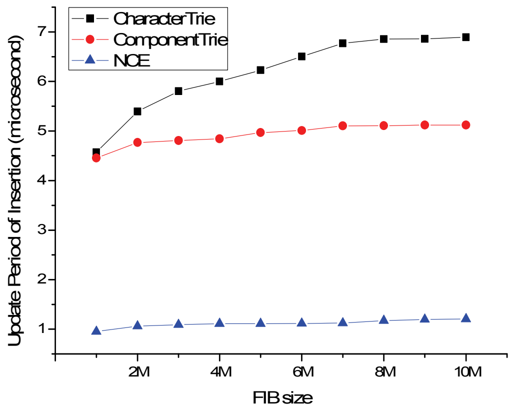

Fig. 6. The update period of insertion of the approaches on different FIB sizes.

3) *Memory Occupation*

The memory occupation of these three solutions is illustrated in Figure 7. To 10M FIB, character-trie needs 936MB memory and component-trie needs 1126 MB while NCE only needs 695MB. Their memory requirement is approximately linear to the FIB size.

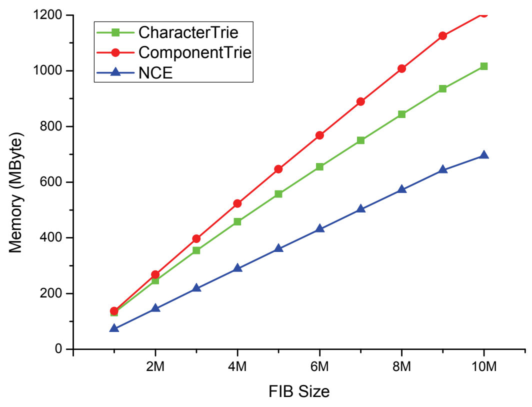

Fig. 7. The memory requirement of the approaches on different FIB sizes.

V. Related Work

Extensive efforts [5-10] have been published in developing benchmarks for IP packet processing. The Benchmarking Methodology Working Group defined a general framework for particular metrics [5]. ClassBench in [6] presented a suite of tools for benchmarking 5-field packet classification algorithms and devices. FRuG was proposed as benchmarking tool for evaluating future packet forwarding algorithms. Moreover, Mei Wang in [9] and Kai Zheng in [10] both initiated benchmarking efforts on IPv6 routing table engine. However, such efforts can't be directly applied to NDN benchmarking due to the different structures and characteristics of FIBs. Through literature investigation, we have found no publication related to NDN FIB by now. Our proposed NDNBench first provides a simulation platform for evaluation and comparison between different NDN name lookup approaches.

VI. Conclusion

Name-based routing lookup is one of the fundamental functions in NDN. However, benchmarking for different lookup solutions has been lacking. In this paper, we develop a scalable platform for evaluation and comparison between different name-based routing lookup approaches. We hope this work can serve for the name lookup solutions by quantifying the performance of lookup and initiate the discussion on NDN benchmarking methodology.

References

[1] L. Zhang, D. Estrin, V. Jacobson, and B. Zhang, Named data networking (ndn) project, in Technical Report, NDN-0001 , 2010.

[2] WANG, Y., DAI, H., JIANG, J., HE, K., MENG, W., AND LIU, B. Parallel name lookup for named data networking, *in IEEE Global Telecommunications Conference (GLOBECOM)* (Dec. 2011), pp. 1 –5.

[3] WANG, Y., HE, K., DAI, H., MENG, W., JIANG, J., LIU, B., AND CHEN, Y. Scalable name lookup in ndn using effective name component encoding, *In IEEE 32nd International Conference on Distributed Computing Systems (ICDCS)* (June 2012), pp.688–697.

[4] NameJet. [Online]. Available: www. namejet.com.

[5] V. Paxson, G. Almes, J. Mahdavi andM. Mathis, Framework for ip performance metrics, RFC2330, May1998.

[6] D.E. Taylor and J.S. Turner, Classbench: A Packet Classification Benchmark, IEEE/ACM Trans. Networking, vol. 15, no. 3, pp. 499-511, June 2007.

[7] T. Ganegedara, W. Jiang, and V. Prasanna. Frug: A benchmark for packet forwarding in future networks, *In IPCCC '10: Proceedings of IEEE IPCCC* 2010, 2010.

[8] M. Castelino, R. Gunturi, V. Filauro, G. Vlantis, M. Campmas, A. Coppola, A. Benchmark for IP forwarding tables. *In Proceedings of the IEEE International Conference on Performance, Computing, and Communications* (2004), IEEE, pp. 123–130.

[9] M. Wang, S. Deering, T. Hain, and L. Dunn, Non-Random Generator for IPv6 Tables, *in 12th Annual IEEE Symposium on High Performance Interconnects*, Stanford University, CA, Aug. 2004.

[10] Kai Zheng and Bin Liu,A Scalable IPv6 Prefix Generator for Route Lookup Algorithm, *Advanced Information Networking and Applications, 2006. AINA*, 18-20 April 2006 Volume: 1.

[11] Mohammad J. Akhbarizadeh and Mehrdad Nourani, Efficient Prefix Cache for Network Processors, *High Performance Interconnects* 2004, pp.41-46, August 2004.

[12] NDNBench Website. [Online]. Available: http://s-router.cs.tsinghua.edu.cn/~zhangting/.

[13] Yi Wang, Yuan Zu, Ting Zhang, KunyangPeng, Qunfeng Dong, Bin Liu, Wei Meng, Huichen Dai, XinTian, ZhonghuXu, Hao Wu, Di Yang, Wire Speed Name Lookup: A GPU-based Approach, Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI '13), April 2-5, 2013, Lombard, IL USA.

[14] Yi Wang, Tian Pan, ZhianMi, Huichen Dai, XiaoyuGuo, Ting Zhang, Bin Liu and Qunfeng Dong, NameFilter: Achieving fast name lookup with lowmemory cost via applying two-stage Bloom filters, Proceedings of INFOCOM2013, Mini-confernce, April 14-19, 2013,Turin, Italy.

[15] CCNx project, www .ccnx.org.

[16] The BGP Instability Report. Available: http://bgpupdates.potaroo.net/instability/bgpupd.html.

[17] Internet Statistics, www.whois.sc/internet-statistics/.