Table 3: Name Component Statistics of Two Name Sets.

URL set	# of names	# of total components	average component length (Byte)	average # of components per name	original size (MB)	# of components /edges in NPT	NPT size (MB)	ENPT size (MB)	ENPT+Hash Table size (MB)	compression ratio
10M Name Set	9,834,747	24,808,603	7.35	2.52	182.26	12,228,081	236.57	48.91	116.02	63.66%
8M Name Set	7,624,393	26,882,827	15.35	3.53	412.78	4,570,563	125.03	18.28	51.83	12.56%

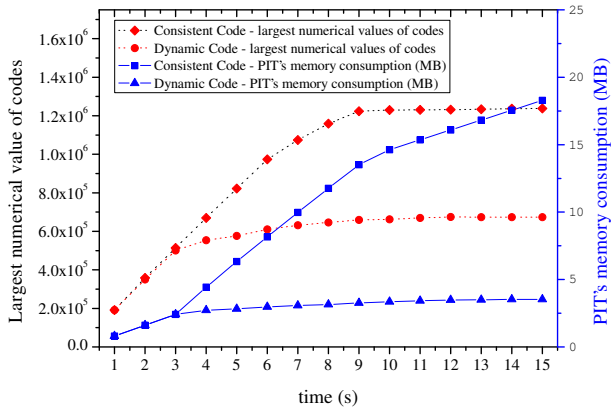

Figure 18: Largest numerical values of codes and PIT's memory consumption.

is 20.67 M components per second and 18.96 M components per second for the 10M Name Set and 8M Name Set, respectively. Divide them by the average number of components per name, we further compute the encoding performance: 8.20 M names per second and 5.37 M names per second for the 10M Name Set and 8M Name Set, respectively. Therefore, the encoding performance is better than the lookup performance and will not be the performance bottleneck.

6.2.4 *Results of dynamic code*

The drawback of assigning consistent codes to components has been discussed in Section 4.5. To demonstrate the effects of dynamic code, we replay the HTTP packets in the captured trace to mimic the packet (name) incoming and outgoing process, which will lead to a PIT of around 300 K (refer to Table 1) entries, and measure how large the numerical value of the codes will be. For comparison, both consistent and dynamic code method will be measured, as well as the PIT's actual memory consumption. The result is shown by Figure 18, the dotted curves represent the largest code of all the CASes, which show that as times goes on, names keep coming and going, the largest code increases. For consistent code, the largest keeps increasing at a high rate after the PIT reaches 300 K valid entries, thus the consumed memory of PIT increases as well. However, for dynamic code, after the PIT reaches 300 K valid entries, the largest code greatly slows down its increasing pace, making PIT's memory consumption remains stable (solid curves). In fact, the largest code by RULE 1 at each snapshot is the number of total components observed by a CAS until this snapshot, while the largest code by RULE 2 is the amount of components a CAS contains at each snapshot. The PIT's memory consumption exhibits similar growth law of the codes. The hash table size (not shown in Figure 18) is almost the same for both consistent and dynamic code methods, since the received names are the same, and thus the name components. The hash table size is 33.55 MB (for the 8M URL Name Set). Therefore, with NCE and dynamic code, PIT exhibits good scalability.

7. RELATED WORK

This section compares our NCE solution to our previous work in [14], and we name it Original NCE. In fact, this paper only continues the encoding idea, but the ways to assign codes, lookup, insert and delete are different. We conclude three major distinctions: 1) The data structure to implement the ENPT in Original NCE involves complicated memory management, such as data movement, fragment management; 2) Original NCE allocates consistent codes to components and does not allow identical components be encoded to different, dynamic codes, which fundamentally contradicts with the code allocation function f in this paper; 3) Original NCE only achieves lookup speedup, but does not exhibit good support for insert and delete operations. However, in this paper, NCE not only significantly accelerates lookup, but also insert and delete operations.

8. CONCLUSION

NDN/CCN propose that PIT caches yet un-responded Interests, when the responding Data packets return, the names are removed from PIT. PIT brings significant features to NDN/CCN. However, none has conducted a measurement study to show the size and access requirements of PIT. Without these knowledge, we have no data to support the design of NDN routers or the actual deployment of NDN. In this paper, we are the first address three problems associated with the PIT: 1) the size and access (lookup, insert, delete) frequency of PIT; 2) how to address the large size and high access frequency problem with a scalable solution; 3) where does PIT reside within a router.

We emulate NDN's application-layer working paradigms by transferring the existing IP applications to the NDN platform. By mapping/translating a captured 20 Gbps gateway trace from IP to NDN scenario at the application perspective, we quantify the size and access frequency of PIT, which demands an efficient and scalable solution. Therefore, NCE is proposed to accelerate the access throughput of PIT, as well as to reduce its size. Moreover, the dynamic code allocation technique makes the NCE solution practical, and further keeps the actual memory consumption of PIT stable. At last, we propose to place PIT on the packets' outgoing line-cards (egress channel) when actually implementing PIT, which meets the PIT design in [15] and eliminates the cumbersome synchronization problem among multiple PITs on line-cards.

9. REFERENCES

[1] HTTP/1.1 RFC. http://www.ietf.org/rfc/rfc2616.txt.

[2] l7-filter. http://l7-filter.clearfoundation.com.

[3] TIE. http://tie.comics.unina.it.

[4] Tstat. http://tstat.tlc.polito.it.

[5] H. Balakrishnan, K. Lakshminarayanan, S. Ratnasamy, S. Shenker, I. Stoica, and M. Walfish. A layered naming architecture for the internet. In *Proc. of SIGCOMM*, 2004.

[6] A. Dainotti, W. de Donato, and A. Pescapé. TIE: A community-oriented traffic classification platform. In *TMA'09*, May 2009.

[7] C. Esteve, F. L. Verdi, and M. F. Magalhaes. Towards a new generation of information-oriented internetworking architectures. In *Proc. of ACM CoNEXT*, 2008.

[8] A. Finamore, M. Mellia, M. Meo, M. M. Munafò, P. di Torino, D. Rossi, and T. ParisTech. Experiences of internet traffic monitoring with tstat. *IEEE Network*, 25(3), May-June 2011.

[9] E. Fredkin. Trie memory. *Communications of the ACM*, 3(9):490–499, Sep 1960.

[10] V. Jacobson, D. K. Smetters, J. D. Thornton, M. Plass, N. Briggs, and R. Braynard. Networking named content. In *Proc. of ACM CoNEXT*, 2009.

[11] H. Jiang and S. Jin. Exploiting dynamic querying like flooding techniques in unstructured peer-to-peer networks. In *Proc. of IEEE ICNP*, 2005.

[12] A. Kumar, J. J. Xu, and E. W. Zegura. Efficient and scalable query routing for unstructured peer-to-peer networks. In *Proc. of IEEE INFOCOM*, 2005.

[13] S. Nilsson and G. Karlsson. IP-Address Lookup Using LC-tries. *IEEE Journal on Selected Areas in Communications*, 17(6):1083–1092, JUNE 1999.

[14] Y. Wang, K. He, H. Dai, W. Meng, J. Jiang, B. Liu, and Y. Chen. Scalable name lookup in ndn using effective name component encoding. In *Proc. of IEEE ICDCS*, 2012.

[15] L. Zhang, D. Estrin, V. Jacobson, and B. Zhang. Named Data Networking (NDN) Project. In *Technical Report, NDN-0001*, 2010.