# Reduce Chord Routing Latency Issue in the Context of IPv6

Jiping Xiong, Youwei Zhang, Peilin Hong, Jinsheng Li, and Lipeng Guo

***Abstract*— Due to the mismatch between an overlay and its underlying physical network, the routing latency in existing DHT systems such as Chord and CAN is rather high. We note the fact that the IPv6 address hierarchical structure can aggregate route entries in Internet, and thus propose that nodes in an overlay can cluster together by taking advantage of the shared IPv6 prefix of specific length. While utilizing DHT systems themselves to store and retrieve the IPv6 prefix information, we adapt it to Chord and thus construct an improved version dubbed eChord. Analysis and simulation results confirm that eChord can significantly reduce routing latency.**

***Index Terms*— Peer-to-peer, IPv6, routing latency, chord.**

## I. Introduction

DUE to their advantages of being scalable and resilient, Distribute Hash Table (DHT) systems such as Chord [1], CAN [2], Pastry [3] and Tapestry [4] have been identified as promising substrates for future Internet-scale peer-to-peer applications. In general, most DHT systems can achieve path lengths of O (log n) hops, where n is the number of participating nodes. However, these are just application-level hops, not true end-to-end latencies that do matter in networks. Actually, since existing DHT systems like Chord are constructed with negligence of geographic proximity of participating nodes, the lookup process can be rather inefficient, with extra latency introduced by routing into and out of the same area/domain many times. Therefore, we argue that to reduce unnecessary inter-domain hops will be a very effective way to improve DHT performance.

IPv6, the next generation standard of network protocol, defines a hierarchically structured address format to reflect the network clustering property [5] [6]. That means hosts with the same address prefix of specific length will reside in the same autonomous system and vice versa (e.g. a large ISP will be assigned an address space with the prefix length of 32 bits). Having noticed such advantages of IPv6 as its address hierarchical feature and standard address allocation scheme, we devise a smart mechanism to incorporate topology information (made available by peers' IPv6 addresses) into the construction of DHT system.

Our method does not need to assume the existence of landmark hosts to which each participating nodes must measure its round-trip-time. Besides, not like other solutions [1] [7] that require peers to maintain more states or introduce more message exchanges. In our recent work [9], we devise a smart scheme to map physically nearby hosts (made available by peers' hierarchical IPv6 addresses) onto a contiguous space in the overlay, so as to reduce the routing latency. Though the mechanism is simple and effective, it may suffer the problem of load unbalance when the number of domains is few. Here, we also exploit the IPv6 hierarchical feature, but in an alternative way. This solution is also very simple, only modifying the original DHT protocols slightly. In fact, we use DHT systems themselves to store and retrieve the topology information. A node first computes a key by hashing its IPv6 prefix of specific length, and then gets a (key, value) pair by associating the key with its node identifier and IP address. When a node joins the system, it inserts the (key, value) pair into the DHT system and meanwhile retrieves those (key, value) pairs containing the same key from the system. Thus, the group of nodes within the same AS can learn one another.

This paper first presents our topology-aware scheme upon the original Chord system, which we call eChord (embedded Chord), and then demonstrates the performance gains by simulation. At last, we conclude in Section 3.

## II. eChord And Evaluation

In this section, we will modify Chord system, and present a two-level system dubbed eChord. To achieve routing locality, we build many small local Chords composed of nodes within the same AS, and embed these local Chords into a big global Chord comprising all the live nodes in the peer-to-peer network. In eChord, each node maintains two routing tables: One is called localFinger table, which is actually the finger table for the local Chord. The other is globalFinger table, which maintains the fingers in the global Chord, only if the identifiers of the fingers fall between those of the node and its successor in the localFinger table.

The following steps are the node join mechanism:

Step 1: When a new node A wants to join the Chord network, it first query to learn if there are living nodes in the network that share address prefix with it. The result of this query contains a set of nodes with the same address prefix. We call it Same Prefix Node Information Set (SPNIS). If the result set is null, then jump to step 3.

Step 2: Node A selects a node B as its bootstrap node to join a local Chord from the returned SPNIS, so as to join the system.

Step 3: Node A joins the system and update his localFinger and globalFinger table through the bootstrap mechanism of the original Chord mechanism.

Manuscript received July 9, 2005. The associate editor coordinating the review of this letter and approving it for publication was Prof. Iakovos Venieris. This work was supported in part by the National Natural Science Foundation of China under Grant No. 60272043.

The authors are with the University of Science and Technology of China, Hefei, P. R. China (e-mail: xjping@mail.ustc.edu.cn).

Digital Object Identifier 10.1109/LCOMM.2006.01002.

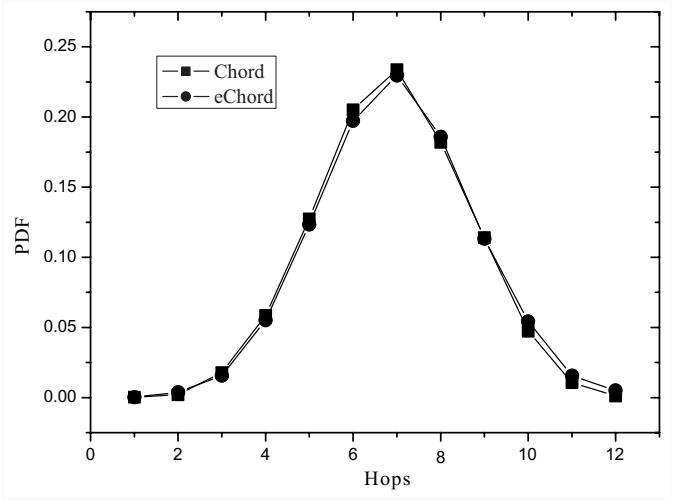


Fig. 1. Distribution of application-level hops.

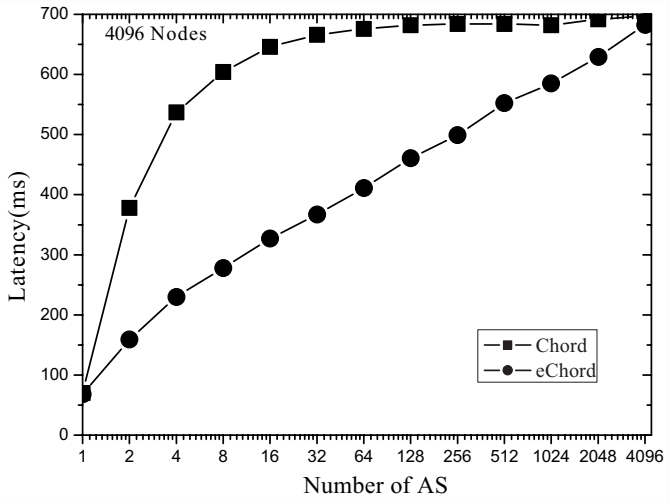


Fig. 2. End-to-end routing latency with the number of domains increased from 1 to 4096.

Step 4: Node A publishes his IPv6 address information to the system through Insert (IPv6Address, LengthOfPrefix) operation. Consequently, node S that is responsible for storing the identifier of hash(IPv6AddreePrefix) needs to update the SPNIS, with the node A's information added. Here the join procedure is finished.

### *A. Design of eChord*

In this section, we evaluate the efficiency of eChord by means of simulation. In our experiments, we use a two-level hierarchical topology generated by BRITE [8], which closely correlates with real network topologies and abides by power law as well.

*1) Average Overlay Message Hops:* To evaluate the average overlay message hops, we generate a topology with 4096 nodes dispersed in 100 domains. The identifier space size is set as $2^{32}$.

We first compare the PDF (probability density function) of application-level hops between eChord and Chord. As shown in Fig. 1, average hops per message in both systems are almost identical. The result is reasonable because the size of a localFinger plus a globalFinger maintained by a node in eChord equals that of a finger table of a node in Chord. Therefore, there is hardly any difference between eChord and Chord from the perspective of the application level hops.

*2) End-to-End Path Latency:* Given that intra-domain latencies are quite small, it is reasonable for us to set the latency of one intra-domain hop as 10 milliseconds, and that of inter-domain hop as 100 milliseconds.

We generate a topology with 4096 nodes and evaluate the true end-to-end path latency with the number of domains increased from 1 to 4096. Fig. 2 shows that only in the worst case where no pare of nodes share the same IPv6 address prefix, the latency in both systems is the same. Though a lookup process in Chord only involves 7 application-level hops on average as shown in Fig. 1, the true end-to-end latency can even reach 700 milliseconds in most cases. This means almost every application-level hop in the lookup path in Chord occurs between two different domains. In contrast, the more nodes

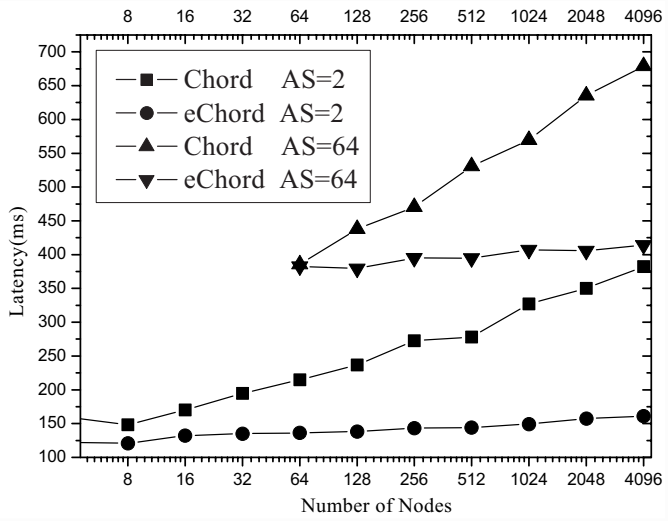


Fig. 3. End-to-end routing latency with the number of nodes increased.

share the same specific address prefix, the more extra inter-domain traffic can be saved in eChord, which confirms our prediction. Consequently, the performance of eChord becomes much better than that of Chord.

In addition, we evaluate the end-to-end latency with the number of nodes increased from 1 to 4096, using two topologies with 2 and 64 domains respectively. As shown in Fig. 3, the latency of Chord goes high as nodes increased, while the latency of eChord remains almost at the same low level. It is an important advantage of eChord that its performance keeps independent of the number of participating nodes if the number of domains rarely changes.

## III. Conclusion

To the best of our knowledge, we are one of the earliest groups who ever attempt to couple IPv6 to P2P systems, so as to reduce the routing latency. In this paper, we try to build physically adjacent IPv6 hosts into many small local Chord using the hierarchical IPv6 address prefix, and embed

them into the global Chord as performance enhancements, thus significantly reducing the end-to-end routing latency. This method can be easily applied to other DHT system [2] [4], and this will be our future work. Recently, the IRTF P2P Research Group also initiates research and development of P2P under IPv6 [10]. We believe that to re-examine the P2P systems in the context of IPv6 will be an important research topic and thus deserves more work on it.

## REFERENCES

[1] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup service for internet applications," in *Proc. IEEE ACM SIGCOMM'01*, vol. 31, pp. 149-160, Aug. 2001.

[2] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, "A scalable content-addressable network," in *Proc. IEEE ACM SIGCOMM'01*, vol. 31, pp. 161-172, Aug. 2001.

[3] A. Rowston and P. Druschel, "Pastry: scalable, distributed object location and routing for large-scale peer-to-peer systems," in *Proc. 18th IFIP/ACM International Conference on Distributed System Platforms (Middleware'01)*, vol. 2218, pp. 329-350, Nov. 2001.

[4] K. Hildrum, J. D. Kubiatowicz, S. Rao, and B. Y. Zhao, "Distributed object location in a dynamic network," in *Proc. 14th ACM Symp. on Parallel Algorithms and Architectures(SPAA)*, pp. 41-52, Aug. 2002.

[5] Y. Rekhter and T. Li, "An architecture for IPv6 unicast address allocation," *RFC 1887*, Dec. 1995.

[6] R. Hinden, S. Deering, and E. Nordmark, "IPv6 global unicast address format," *RFC 3587*, Aug. 2003.

[7] P. Ganesan, K. Gummadi, and H. Garcia Molina, "Canon in G major: designing DHTs with hierarchical structure," in *Proc. IEEE ICDCS'04*, pp. 263-272, Mar. 2004.

[8] Brite, a network topology generator, http://www.cs.bu.edu/brite/

[9] X. Jiping, Z. Youwei, H. Peilin, and L. Jinsheng, "Chord6: IPv6 based topology-aware chord," in *Proc. International Conference on Networking and Services*, Oct. 2005, pp. 19-23.

[10] http://homepage.mac.com/cvgl/p2pv6/p2pv6.htm