An Efficient Lookup Service in DHT Based Communication System

Kai Shuang, Peng Zhang, and Sen Su

State Key Laboratory of Networking & Switching Technology
Beijing University of Posts & Telecommunications (BUPT)
Beijing, China
{shuangk,susen}@bupt.edu.cn, zhangppmmeer@163.com

Abstract. Telecom systems utilize the Distributed Hash Tables (DHTs) approach to build the network infrastructure for advantages of even distribution of workload, high scalability and cost-effectiveness. Although DHT is undoubtedly applicative in such architectures, some practical distinctions still should be considered to meet the performance requirements of telecom infrastructures. This paper focuses in two features of the distributed telecom system, so-called the real-time response and geographic partition, proposes a hierarchical DHT lookup service named Comb. Comb's overlay is organized as a two-layered architecture, workload is distributed evenly among nodes and most queries can be routed in no more than two hops. Comb performs effectively with low bandwidth consumption and satisfactory fault tolerance even in a continuously changing environment. Both theoretical analysis and experimental result demonstrate that the two-layered architecture of Comb is feasible and efficient. Comb improves the performances on routing delay and lookup failure rates with high scalability and availability.

Keywords: Distributed Communication System, Peer-to-Peer, Comb, DHT, Two-Hop.

1 Introduction

Distributed Hash Table (DHT) are widely applied in building large-scale self-organizing overlay networks [1], such as file-sharing, search engines and content distribution. Currently, both distributed IP Multimedia Subsystem and P2PSIP are proposed to adopt the structured P2P overlay in communication systems. A fundamental problem for a DHT lookup service is resources locating and routing. Overlay topology, routing path latency and maintenance cost are three elements that impact the efficiency of a DHT algorithm [1]. Designs of DHT algorithms vary largely, so far as we know, most proposed DHT algorithms vary routing tables' size from $O(\log N)$ to $O(N)$, with routing hops ranged from $O(\log N)$ to $O(1)$ [2]. Large routing tables are expensive in maintenance and hard to scale to large systems while long routing hops lead a long time routing delay. Trade-off should be made between routing table's size and routing hops in selection of a DHT algorithm.

J. Su et al. (Eds.): ICoC 2013, CCIS 401, pp. 74–84, 2013.

A communication system is used for real-time intercommunication, path latency will directly impact the quality of connections [3]. For this reason, real-time response is required for a distributed communication system and minimizing routing delay is a primary objective. Additionally, communication systems are deployed in accordance with the geographic session distribution pattern, the whole network shall be organized as a multilayer architecture with several regions. Therefore, a hierarchical overlay is necessary for the distributed communication system. Previous work proposes hierarchical P2P algorithms [4] where each hierarchy consists of super-nodes for upper hierarchy. Super-nodes take more responsibility acting as centralized index, makes it impractical for load-balance. The Comb protocol proposed in this article is purely peer-to-peer and protocols on each node are completely the same. The whole overlay is organized as a two-layered architecture, a Comb node maintains information about $O(\sqrt{N})$ other nodes for routing, resolves a lookup in no more than two hops. Mainly two features distinguish our design from many other P2P lookup protocols.

1. Comb is purely distributed. The Comb overlay is divided into several domains, two-layered architecture satisfies the requirement of geographic distribution pattern. Meanwhile, Comb abandons super-node, nodes in Comb overlay have no distinctions. This makes Comb a load-balanced network, avoids single point of failure and performance bottleneck. On the other hand, a purely peer-to-peer network is much better for scalability.

2. Comb is simple and stable. A Comb node requires about $O(\sqrt{N})$ size routing table for lookups resolve, but routing performance degrades gently when routing information is out of date. Comb guarantees correct routing (though slow) of lookups as long as one piece in routing table is correct. This is important for a distributed system to keep steady.

The rest of this paper is organized as follow: Section 2 compares Comb to relevant DHTs. Section 3 presents the system model. Section 4 is the base Comb protocol, and Section 5 evaluates Comb's performance through simulation and experiments. Finally, we summarize our contributions in Section 6.

2 Related Work

The first generation of DHT algorithms adopt a completely flat structure, the whole overlay is organized as a ring or other plain topology. Consistent hashing [5] is utilized to assign keys to nodes and resources. Generally, a DHT overlay consists of N nodes which share R resources (N << R), the key space is partitioned randomly by participating nodes, and each node is in charge of the resources belongs to its key space section. Designs of flat DHTs include Chord[6], CAN[7], Kademlia[8], Pastry[9] and Tapestry[10].

Flat DHTs have certain advantages, such as structural stability and workload balancing. On the other hand, they are incapable of achieving latency guarantees for queries and offering a hierarchical overlay network. That makes flat DHTs improper in to utilize in a distributed communication system. Hierarchical DHTs (HDHTs) [11] are the last generation of DHT designs, HDHT nodes are distributed into hierarchies;

each next hierarchy consists of super-nodes for upper hierarchy, leading a tree-like architecture. Distinguished from flat DHTs, Hierarchical DHT algorithms are able to guarantee path latency, for the last decade, a group of HDHT designs have been proposed: OneHop [12,13],Sandstone [14], D1HT [15], 1h-Calot, 2h-Calot[16].

However, two virtual nodes in different slices don't take geographic distribution and network connectivity into consideration. For large system deployed over the countrywide, it is difficult to keep nodes in system connected with all the nodes in two different slices at the same time, especially when the two slices have a long geographical distance. Moreover, two virtual nodes may bring twice as much as bandwidth consumption in joining, leaving procedures and routing table maintenance.

3 System Model

Comb is designed for the distributed communication system with two special features affect the design. Geographic Distribution Pattern: The structure of a communication system should be in accordance with the geographic pattern in practice, and interactions and network connectivity between host pairs are also related to this pattern. As a survey on MIIT[17], nearly 80% call sessions are between intra-province host pairs, while 20% call sessions between inter-province host pairs. Therefore, a hierarchical P2P lookup service with regionalism is essential. Real-time Response: Minimizing message delay is an important performance objective for a communication system, it is assumed that 20~50 milliseconds delay between any inter-province and less than 10 milliseconds delay between any intra-province host pairs for an IP based network [14], so less routing hops is critical for the system.

Figure 1 gives a software structure for Comb, it is consisted of four main components: Communication, Topology Routing, Data Storage and Application Layer. Intercommunication between hosts is implemented by the bottom layer communication

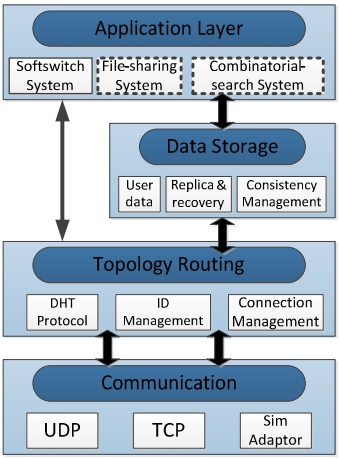

Fig. 1. Structure of an example comb-based distributed system

module, applications choose the transport layer protocol according to the scenarios. The topology routing module provides lookup service, finds the best suitable host for a given key, and also includes ID management and topology maintenance. Subscriber data such as storage, consistency verification are taken charge of by the data storage module. Application layer takes topology routing module and data storage module as substruction, specifies logical function of the system despite the underlying details. Modularized designs simplifies the system implementation.

In addition, a mathematical model is presented for system verification. Previous work parameterized the workload of a P2P system by a tuple $< n, f, l >$ [16], which means n nodes in the system with an average node lifetime l, each node process f lookups per second on average. Corresponding to Comb's topology, our work splits f to fl and fg representing the average lookup times of intra-domain and inter-domain per second. Suppose the whole overlay is divided into k domains, then each domain has n/k nodes, in a system with a node lifetime l, on average n/(kl) nodes join and leave each second in one domain. When a node joins or leaves, at least one message will be sent to all its neighbors for informing, Comb maintenances all nodes within the domain as its neighbors, i.e., at least n/k messages are sent during the joining or leaving procedure. Assuming that all messages get unit size s and each request is acknowledged by a response, then the traffic for nodes arrivals and departures B1 in one domain is:

$$B_1 = (1+1)s \bullet (\frac{n}{k} \times \frac{n}{kl} + \frac{n}{k} \times \frac{n}{kl}) \tag{1}$$

Each node process f_l intra-domain lookups with one hop and f_g inter-domain lookups with two hops per second, resulting in traffic for lookups with one domain B_2 is:

$$B_2 = (1+1)s \bullet (\frac{n}{k} f_l + 2\frac{n}{k} f_g) \tag{2}$$

As a result, without regard to the inter-domain maintenance and keep-alive traffic, the minimum traffic demand in Comb B_3 is:

$$B_{\min} = (B_1 + B_2) \bullet k = (\frac{4sn^2}{k^2 l} + \frac{2snf_l}{k} + \frac{4snf_g}{k}) \bullet k = \frac{4sn^2}{kl} + 2snf_l + 4snf_g \tag{3}$$

4 The Base Protocol

Comb provides protocol for key based resources location, how nodes join and leave the system and how to recover from the failures. This section gives a simplified version of description of the Comb protocol.

4.1 Overview

As in Figure 2, Comb organizes all the nodes in a circular ring like Chord [6].

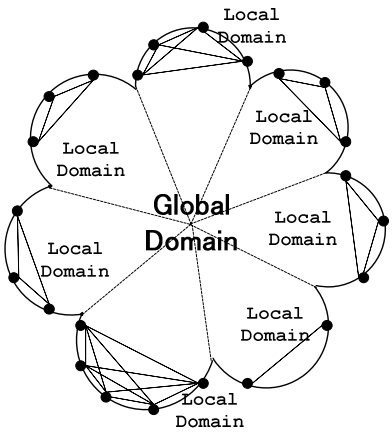

Fig. 2. Comb overlay consists of two domains: global domain and local domain

We refer to a node's counter-clockwise neighboring node as its predecessor and clockwise node as successor. Comb divide the whole overlay into several regions named Domain, and runs a one-hop DHT protocol inside each Domain. Correspondingly, overlay of Comb is deployed as a two-layered structure: global layer (correspond to countrywide) map the whole ring and local layer (correspond to a province or a city) map the domain. Table 1 gives an example of the routing tables: a Local Table contains complete information of nodes within domain and a Global Table maintains at least one node's information for each outer domain. To route a message between two different domains, an origin tries to find a node N in destination domain from global table and forwards the message, N looks up its local table and sends the message to the destination node.

Table 1. an example of local table and global table

Node ID	URI	IP Address
00100001	alice@comb.com	xx.xx.xx.xx
00001010	bob@comb.com	xx.xx.xx.xx
00001000	tom@comb.com	xx.xx.xx.xx

Local Table

Domain	Node ID	URI	IP Address
01	01000011	a@comb.com	xx.xx.xx.xx
10	10000100	b@comb.com	xx.xx.xx.xx
11	11000101	c@comb.com	xx.xx.xx.xx

Global Table

4.2 Identifier

In Comb, the whole overlay is divided into several domains, organized as a two-layered architecture. Accordingly, the identifier space is separated into two parts: domain ID and host ID as depicted in Figure 3.

Fig. 3. Comb identifier consists of two parts: domain id and host id

When a node joins into the Comb system, it chooses the nearest domain and takes the domain ID as the identifier prefix, then applies consistent hashing to its IP address or URI as the identifier postfix. This assignment also makes it possible to determine the geographic domain of any nodes by its ID.

4.3 Node Joins and Leaves

Assume that a new joined node named JP (Joined Peer) knows about at least one node named BP (Bootstrap Peer) already in the system through some out-of-band methods [2], BP is in the nearest domain from JP. JP copies BP's local table and global table in order to build routing tables itself. For the local table, a copy from BP is enough, but for the global table, ID Transformation is necessary, though it will function well with a complete copy from BP logically.

ID Transformation:

Suppose BP get its identifier $ID_{BP} = ID_{domain}^{BP} + ID_{host}^{AP}$, and JP's identifier $ID_{JP} = ID_{domain}^{JP} + ID_{host}^{JP}$, $ID_{domain}^{BP} = ID_{domain}^{JP}$. define $offset(\mathrm{B}P, JP) = ID_{host}^{BP} - ID_{host}^{JP}$, for each node item in BP's global Table ID_N, JP builds a new item $ID_M = ID_{domain}^{N} + | ID_{host}^{N} - offset(BP, JP) |$, whose address is fetched through AP.

ID transformation makes global table varies from node to node within a domain, requests from domain A to domain B are forwarded by different nodes in domain B, resulting in a network with load balance and scalability. For user information security and system robust, communication operators always choose nodes only when they are deployed as BP instead of all nodes in overlay.

Once JP built its routing tables, it need inform other nodes in domain of its arrival. A multicast tree [15] rooted at JP for disseminating membership changes can be applied: For a N nodes domain, update messages are propagated to the 2^{TTL} th (TTL $= 0, ..., \log_2 N$) successors of JP, each message consists of JP's identifier and value of TTL. Notified successors continue with message propagating to their 2^i th (i $= 0, ...,$ TTL-1) successors and minus TTL by 1 until TTL = 0. An example of a 8 nodes domain is shown in Figure 4.

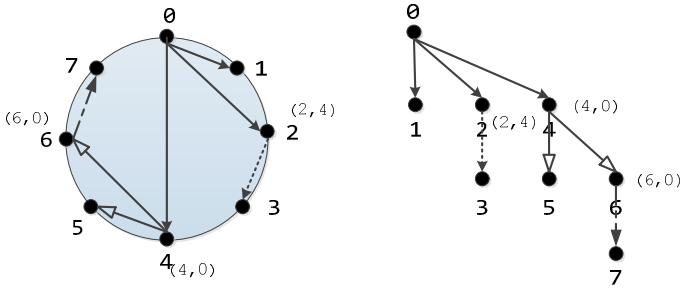

Fig. 4. An example of the multicast tree for disseminating membership changes

The multicast tree disseminates messages all over the domain in collaboration, workload are distributed to nodes. However, the workload in disseminating of each node varies greatly, nodes closer to the tree root get more messages to cast [16], sometimes exceeds nodes' bandwidth capacity. We propose another multicast tree for disseminating called the binary-multicast tree for the structure is a binary tree.

A binary-multicast tree is also rooted at JP, among N nodes in its local table, JP selects its immediate successor and the $2^K th(\mathrm{K} = \log_2 N / 2)$successor as its children (define as C_1 and C_2) in the tree, and asks them to cover the range $(2, 2^K - 1)$ and $(2^K + 1, N)$, respectively. Node C_1 and C_2 use a similar process to expand the tree by adding their immediate successor and the $2^{K-1} th$ successor as children, and so forth. The process stops when there is no node in the range. Figure 5 gives a brief example of binary-multicast tree, Node 0 acts as the root of the tree and selects 1 and 4 as its children, each child of node 0 is responsible for a covering range, and they also build their own tree to expand the binary-multicast tree.

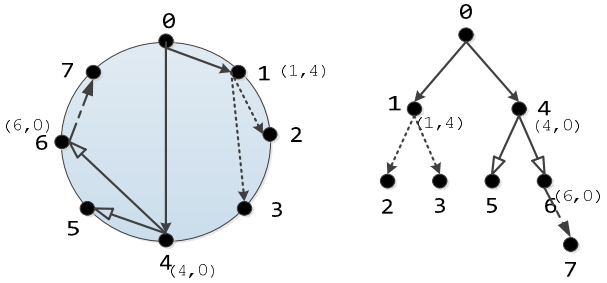

Fig. 5. Binary-multicast tree for disseminating membership changes

The binary-multicast tree disseminates messages with more reasonable and balanced bandwidth demands, nodes in a binary-multicast tree propagate update messages to only 2 other nodes at most (see figure 5), the tree is implicitly embedded in the overlay, there is no message to construct the trees before use and no message to tear down the trees after use. However, inaccurate routing tables are acceptable in

messages dissemination, suppose node N that should be selected as a child of Node P had not been selected, in the worst case, N will receive the notification from its predecessor as the range narrows. We can prove that binary-multicast tree disseminate messages in the same time complexity with a multicast tree of $O(\log_2 N)$.

When a node leaves, it notifies its predecessor and successor, they (or one of them) propagates this change to all nodes over the domain using a process similar to a node's arrival.

4.4 Routing Tables Maintenance

Only nodes in local table are notified when a node leaves, in a purely distributed overlay of Comb, this node is probably also in global tables of other nodes from outer domains. To this end, a heartbeat with nodes in global table to keep alive is necessary. Every t_g seconds afterwards, a heartbeat message is sent to each node item in global table to keep-alive, if a heartbeat is not acknowledged until time fires, we treat it as a departed node, and do ID transformation to find a new substitute item through BP or any other nodes as in node join.

For each node K with an ID of ID_k in Node N's global table, heartbeats can carry the destination identifier of $ID_{dest} = ID^k_{host} + ID^N_{domain}$, when node K receives the heartbeats and find it isn't responsible for ID_{dest}, a redirection should be sent back to node K with the new substitute node responsible for ID_{dest}.

The period t_g is chosen such that 80% nodes lives longer than t_g seconds, assume node lifetime t follows an exponential distribution $f(t) = \lambda e^{-\lambda t} (t > 0)$, then:

$$\int_{t_g}^{+\infty} f(t)dt = 0.8 \quad \Rightarrow \quad t_g = -\frac{\ln(0.8)}{\lambda} \tag{4}$$

The expectation of t is:

$$E[t] = \frac{1}{\lambda} = l \text{ ,so we have } \quad t_g \approx 0.2l \tag{5}$$

Another situation need to be taken into account is that not all nodes leave the system normally; some nodes may fail without any notifications. Hence, a mechanism to handle failed nodes is needed. Periodically, a node sends an announcement to its immediate predecessor to state its existence, a node also sets timer for its successor's announcement.

If an announcement is received from node N which is already in local table by node P , P would reset the timer for the next announcement. If node N is not in the local table of P or P's timer for N is fired, then P adds/deletes N to/from local table, and broadcast the membership change to all the nodes in domain through a binary-multicast tree.

5 Simulation and Experimental Results

In order to evaluate the function and performance of Comb, we implemented a simulator in C++. The simulator can simulate Comb system with up to 8,000 nodes.

Table 2. Parameters and Settings of the simulation

	Parameter	Value
N	Total number of peers in the network	1000, 2000, 4000, 8000
D	Total number of Domains	20, 40, 80
L	Node's lifetime	90s, 120s, 180s
F	Frequency of lookup	1, 2, 3

The simulation is configured by parameters N, D, L and F. When simulator starts up, the Comb system contains N nodes with settled routing tables distributed in D domains averagely, nodes join and leave the system with the lifetime L, and send out lookup messages by every F seconds. The simulator stops when 10N nodes joined in the system accumulatively. Parameters are configured as the table 2.

We evaluate Comb's routing efficiency by figuring out the average routing hops and failed rate per lookup. Figure 6 plots the average number of routing hops per lookup when varying the system size and numbers of domains. The Figure shows that the average routing hops are always below 2 in Comb, and the number increases alone with both the system size and domains number.

Failed lookups should also be considered for routing efficiency. In Comb simulator, messages record every hop it routed through, a lookup (and also other messages) was regarded as a failed message when it is forwarded by more than 10 peers, and would be discarded by the simulator. Figure 7 reports the failed rate of lookups in a variety of conditions, we see that Comb system has a satisfied failed rate, in the experiment environment, failed rates are below 0.5% in all conditions.

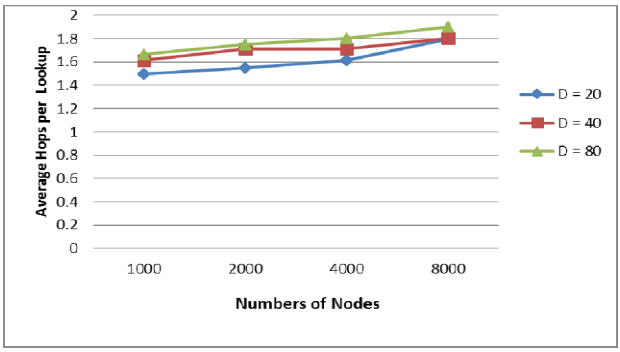

Fig. 6. Routing hops per lookup varying nodes' number and domains' number

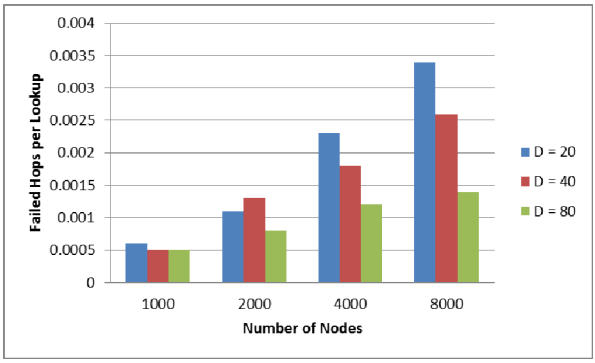

Fig. 7. Failed rates per lookup varying nodes' number and domains' number

As well as average routing hops, failed lookup rate increased along with the system size, this is also because more nodes leaving and joining brings about more time delay to keep routing tables up to date, inaccurate routing tables increase the failed probability of failure. On the other hand, small domains number also results in a high failed rate, this is due to few domains generates a big domain size, Comb runs a one-hop protocol within the domain, failed rate increased along with the domain size because of the time delay for routing tables up to date as mentioned before. Therefore, tradeoffs should be made between average routing hops and failed lookup rate on the selection of domains number, generally, the optimal number for minimum traffic we calculated in section 4 is a good choice. Figure 6 and Figure 7 demonstrate that Comb has an excellent performance in routing efficiency.

6 Conclusions

This paper presents Comb, a two-hop DHT lookup algorithm for P2P overlay and evaluates Comb by analysis and experiments. Comb algorithm aims at the features of the distributed communication system: real-time response and geographic partition, constructs the whole overlay a two-layered architecture. Nodes in Comb system maintain two routing tables: local table and global table, messages are routed intra-domain and inter-domains.

Compared with other DHT lookup algorithms, the main contributions of Comb are: a non-hierarchical two-layer DHT architecture that avoids unbalanced workload and bottlenecks; a two-hop routing mechanism that routes most lookups in no more than two hops to satisfy the real-time response requirement; a domain-partition mechanism that adapts the geographic distribution pattern of distributed communication system; an improved message dissemination mechanism--binary-multicast tree that balances the workload in disseminating of each node and an abstract model for network traffic calculation.

Acknowledgements. This work is partially supported by 973 program of China (No. 2009CB320504), the Fundamental Research Funds for the Central Universities(2013 RC1102), Innovative Research Groups of the National Natural Science Foundation of China (61121061).

References

1. Korzun, D., Gurtov, A.: Survey on hierarchical routing schemes in “flat” distributed hash tables. Peer-to-Peer Networking and Applications 4(4), 346–375 (2011)
2. Tang, C., Buco, M.J., Chang, R.N., et al.: Low traffic overlay networks with large routing tables. ACM SIGMETRICS Performance Evaluation Review 33(1), 14–25 (2005)
3. Lindman, P., Thorsell, L.: Applying distributed power modules in telecom systems. IEEE Transactions on Power Electronics 11(2), 365–373 (1996)
4. Garces-Erice, L., Biersack, E.W., Ross, K.W., et al.: Hierarchical peer-to-peer systems. Parallel Processing Letters 13(4), 643–657 (2003)
5. Karger, D., Lehman, E., Leighton, T., et al.: Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web. In: Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing, pp. 654–663. ACM (1997)
6. Stoica, I., Morris, R., Karger, D., et al.: Chord: A scalable peer-to-peer lookup service for internet applications. ACM SIGCOMM Computer Communication Review 31(4), 149–160 (2001)
7. Ratnasamy, S., Francis, P., Handley, M., et al.: A Scalable Content-Addressable Network (2001)
8. Maymounkov, P., Mazières, D.: Kademlia: A peer-to-peer information system based on the xor metric. In: Druschel, P., Kaashoek, F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, pp. 53–65. Springer, Heidelberg (2002)
9. Rowstron, A., Druschel, P.: Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In: Guerraoui, R. (ed.) Middleware 2001. LNCS, vol. 2218, pp. 329–350. Springer, Heidelberg (2001)
10. Zhao, B.Y., Huang, L., Stribling, J., et al.: Tapestry: A resilient global-scale overlay for service deployment. IEEE Journal on Selected Areas in Communications 22(1), 41–53 (2004)
11. Korzun, D., Gurtov, A.: Hierarchical architectures in structured peer-to-peer overlay networks. Peer-to-Peer Networking and Applications, 1–37 (2013)
12. Fonseca, P., Rodrigues, R., Gupta, A., et al.: Full-information lookups for peer-to-peer overlays. IEEE Transactions on Parallel and Distributed Systems 20(9), 1339–1351 (2009)
13. Gupta, A., Liskov, B., Rodrigues, R.: Efficient routing for peer-to-peer overlays. In: First Symp. on Networked Systems Design and Implementation (NSDI), pp. 113–126 (2004)
14. Shi, G., Chen, J., Gong, H., et al.: Sandstone: A dht based carrier grade distributed storage system. In: IEEE International Conference on Parallel Processing, ICPP 2009, pp. 420–428 (2009)
15. Monnerat, L.R., Amorim, C.L.: D1HT: a distributed one hop hash table. In: IEEE 20th International Parallel and Distributed Processing Symposium, IPDPS 2006, p. 10 (2006)
16. Tang, C., Buco, M.J., Chang, R.N., et al.: Low traffic overlay networks with large routing tables. ACM SIGMETRICS Performance Evaluation Review 33(1), 14–25 (2005)
17. MIIT 2012, `http://www.miit.gov.cn`