Int. J. Communications, Network and System Sciences, 2012, 5, 802-809
http://dx.doi.org/10.4236/ijcns.2012.512084 Published Online December 2012 (http://www.SciRP.org/journal/ijcns)

A Scalable and Robust DHT Protocol for Structured P2P Network

Xiao Shu, Xining Li
Computing and Information Science, University of Guelph, Guelph, Canada
Email: xshu@alumni.uoguelph.ca, xli@cis.uoguelph.ca

Received October 1, 2012; revised October 26, 2012; accepted November 5, 2012

ABSTRACT

Distributed Hash Tables (DHTs) were originated from the design of structured peer-to-peer (P2P) systems. A DHT provides a key-based lookup service similar to a hash table. In this paper, we present the detailed design of a new DHT protocol, Tambour. The novelty of the protocol is that it uses parallel lookup to reduce retrive latency and bounds communication overhead to a dynamically adjusted routing table. Tambour estimates the probabilities of routing entries' liveness based on statistics of node lifetime history and evicts dead entries after lookup failures. When the network is unstable, more routing entries will be evicted in a given period of time, and the routing tables will be getting smaller which minimize the number of timeouts for later lookup requests. An exprimental prototype of Tambour has been simulated and compared against two popular DHT protocols. Results show that Tambour outperforms the compared systems in terms of bandwith cost, lookup latency and the overall efficiency.

Keywords: P2P Network; Distributed Hash Table; Small-World Distribution; Parallel Lookups

1. Introduction

Unlike the majority of current file sharing P2P systems, DHTs organize the P2P network in a structured manner and provide a simple lookup interface which similar to hash table. Logically, each host in DHTs stores and serves resources named by keys like a bucket in classic hash tables, and it employs a distributed lookup function collaboratively with other hosts to locate the hosts being responsible for assigned keys. This simple and elegant lookup interface makes DHTs a potential universal building block for many distributed system applications.

One of the challenge every P2P system has to cope with is churn: Nodes continuously join and leave the system. Studies of file sharing networks observe that the median time a node stays in the system ranges from tens of minutes to an hour depending on variant applications [1-3]. It is not a good assumption that departing nodes will be able to notify their neighbours before leaving, and in many DHTs, nodes do not even know who have them as neighbours. Stale entries result in expensive lookup timeouts, since it takes multiple round-trip time for a node to determine that a lookup packet has been lost and and re-route it through another neighbour. There are various methods to reduce lookup latency and increase the accuracy of routing tables under churn. In general, these methods generate extra communication to get more information about the liveness of existing neighbours and new nodes in the network. In order to be robust in scenarios when the network becomes unstable and churn rate increases rapidly, DHT networks should keep the extra cost in control to avoid flooding the network.

This paper introduces a new DHT protocol, Tambour, which reduces average lookup latency by using parallel lookup and bounds the induced communication overhead automatically. While many popular DHTs organize neighbours in structured manners with a fixed routing table size. Tambour dynamically tunes its table size to get the best lookup performance. Unless there is abundant spare bandwidth, it does not probe the availabilities of its neighbours periodically, but maintains a flexible routing table and selects next hop node in an opportunist way. Most existing DHTs more or less rely on users to set up various parameters in order to tune the performance of the networks under different environments. Unfortunately, studies show that many of these parameters are in practice either left unspecified or deliberately misreported [4]. Instead of asking user to do the optimization, Tambour automatically provides optimal lookup performance in a robust way and fits itself as the deployment environment changes. Performance evaluations show that Tambour could keep its communication overhead within a range over different operating conditions, and has similar or better lookup performance than some existing DHT protocols even when they were tuned for specific workload.

The rest of this paper is structured as follows. Section 2 presents the basic system model and design of the Tambour protocol, and Section 3 shows the techniques employed in the implementation of Tambour. Section 4 demonstrates Tambour's performance through simulation and experiments on a prototype implementation. Section 5 compares Tambour to related work. Finally, we summarize our contributions and outline the items for future work in Section 6.

2. System Design

Similar to Chord [5], Tambour maps nodes to a ring-like circular address space by their randomly selected 128-bit identifiers. By sending probe packet periodically, each node maintains a list of successor nodes, whose IDs most closely follow the node on the ring, and a list of predecessors accordingly. With this simple routing structure, a lookup request is forwarded among adjacent nodes clock- wise on the ring until it reaches the node for which the request is looking. Since in each step, the lookup request reduces the clockwise ID distance to the destination by at least one, this procedure will finish in $O(\log n)$ hops eventually.

Obviously, the simple ring routing structure is not efficient and scalable enough for practical use. Chord solves this problem by maintaining a "finger-table" with $O(\log n)$ entries to fold the ID distance by half at each hop, so that, it guarantees $O(\log n)$-hop lookup performance. Besides Chord, Pastry [6], Tapestry [7], and Kademlia [8] employ similar mechanisms to achieve good scalability. However, in these DHTs, only a few routing entries are eligible to forward a given lookup. This inflexibility in choosing next hop not only reduces the efficiency of proximity neighbour selection (PNS) [9], but also does pose a robustness problem.

For a Tambour node, the ID distance from the node to each of its neighbours follows a $\frac{1}{x}$ distribution. In other words, the further two nodes are apart from each other on the ring, the less likely that one chooses the other as a neighbour. In **Figure 1**, which illustrates this distribution, each dot represents a node in the system and the darker ones with edges connecting to the bottom are the neighbours of the bottom node. This $\frac{1}{x}$ distribution, which is based on the "small-world" model [10], creates a simpler and more flexible routing structure. It has been used by a few previous DHTs such as Symphony [11] and Accordion [12]. Interestingly, Chord follows $\frac{1}{x}$ distribution as well unintentionally, since its "finger-table" maintains a constant number of routing entries for every ID-range $\left[2^{i-1}, 2^{i}\right)$, and this matches the fact that the number of entries required by $\frac{1}{x}$ distribution, $\int_{2^{i-1}}^{2^{i}} \frac{1}{x} \mathrm{d}x = \log 2$, is a constant as well.

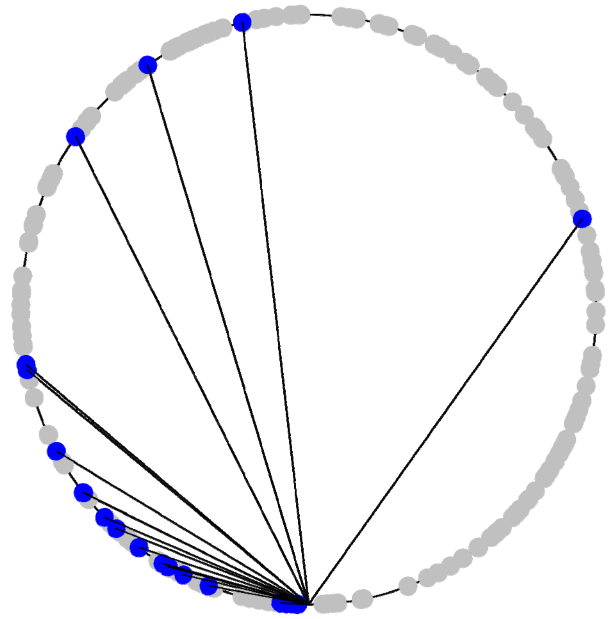
Figure 1. A $\frac{1}{x}$ distribution of neighbours in Tambour, where the further two nodes are from each other on the ring, the less likely that one chooses the other as a neighbour.

With this $\frac{1}{x}$ distribution, Tambour gives a node the flexibility to utilize a routing table of any size. When the network has limited bandwidth or is unstable, Tambour maintains a small routing table to achieve Chord-like $O(\log n)$-hop lookup performance; when the network environment permits, Tambour can scaling its performance up to constant hops like $O(1)$-hop protocols [9,13-16]. Another advantage of $\frac{1}{x}$ distribution is its flexibility to select routes. Since each routing entry near the destination in the address space is a potential route, a node has a large set of routing entries to choose from as next hop to avoid nodes with low availability or high latency.

3. Implementation

The $\frac{1}{x}$ distribution provides a flexible and scalable routing structure for DHTs. Based on this model, Tambour employs several optimization techniques to fulfill its advantages in different operating environments.

3.1. Node State

As the underlying routing structure gives the flexibility to pick next hop from a large pool, Tambour enhances the lookup performance by prioritizing routing candidates that are more responsive and stable.

In order to assure the liveness of neighbour nodes and calculate their latencies, most exiting DHTs send probing message to each routing entry periodically [5-7]. One of the weaknesses of this approach is that it limits the size of routing table by the available bandwidth and reduces the effect of proximity routing [9]. Tambour employs a more efficient technique, parallel lookup, to avoid the probing overhead and mask timeouts caused by stale entries at the same time [17].

Without periodical contact, Tambour cannot be certain whether a neighbour is still in the network before relaying a lookup request to the node. Thus, Tambour assumes that the node lifetimes follow a heavy-tailed Pareto distribution as suggested by empirical studies [3,18], and predicts the probability of a neighbour being alive with the knowledge about when the node joined the network and when it was seen last time [12]. Another problem caused by the lack of ping is that a node has no idea about the latency of a new neighbour right after learning it from other nodes. Tambour compensates this by bringing in the Vivaldi coordinate system, which predicts latencies among nodes at a small bandwidth cost [19]. After the first lookup request between two nodes, Tambour will record the real latency value and keep the estimated one for future reference.

While many DHTs would consider both the latency and liveness of a neighbour in routing selection, they usually give priority to one trait over the other. Tambour characterizes node state in a more balanced way with the mathematical expectation of lookup latency. For example, if a neighbour with 80% chance of being alive could receive the lookup request in 150 milliseconds, it would also possibly lost the request in 20% of the cases and waste 1 second (5 times of the average Internet host latency in a typical setting) for the sender to recover, therefore, in average, the latency of routing through that neighbour is $150 \times 80\% + 1000 \times 20\% = 320$ milliseconds. In general, if a node relays a lookup to d neighbours in parallel, assume that the latencies and live probabilities are $t_1, t_2, \cdots, t_d$ and $p_1, p_2, \cdots, p_d$ respectively, where $t_1 < t_2 < \cdots < t_d$, then the mathematical expectation of the latency of this d degree parallel lookup is:

$$\mathrm{E}[t] = \sum_{i=1}^{d} \left[t_i p_i \cdot \prod_{j=1}^{i-1} \left(1 - p_j\right) \right] + t_{\text{fail}} \cdot \prod_{j=1}^{d} \left(1 - p_j\right) \qquad (1)$$

where t_{fail} is the timeout value. With this mathematical expectation model which gives an all-around understanding of node state, a Tambour node could control the average latency more accurately.

3.2. Parallel Lookups

As illustrated in **Figure 2**, instead of pinging neighbours periodically, Tambour forwards each lookup request to several neighbours simultaneously to reduces the possibility of having to recover from stale routing entries. Parallel lookups is more efficient than active neighbour exploration in term of bandwidth overhead. The reason is that the larger degrees of parallelism, the more opportunities of learning new neighbours and refreshing the status of current neighbours, and at the same time, it reduces the risk of waiting for timeouts. Therefore, parallel lookup not only reduces the negative effect of timeouts on the overall latency, but also does maintain the freshness of routing tables.

Tambour employs a lottery algorithm for neighbour selection, where each entry near the lookup destination on the ID ring is given a number of lottery tickets that is inversely proportional to the expected latency of that entry. This algorithm biases towards nodes with high availability and low latency, but nodes that are not so stable or responsive get some opportunity to be selected as well. With this randomized method, Tambour keeps picking neighbours one by one until the probability of the lookup being successfully received by at least one neighbour reaches a threshold q. In other words, if the average liveness probability of a neighbour is p, Tambour will create a d-way parallel lookup to insure that $1-(1-p)^d \geq q$.

It is nice to be able to guarantee the lookup delivery rate, since by doing so, Tambour would waste little time on waiting for the expensive timeout. However, this feature comes with a potential problem—if each node adopts a large degree of lookup parallelism arbitrarily to meet the delivery rate threshold, one lookup could trigger a flood of parallel messages in the whole system. As this problem affects the efficiency and robustness of Tambour, it is necessary to investigate whether it could trigger off a positive feedback or happen on a regular basis.

The reason a node choosing a number of neighbours to do parallel lookup is because of the poor stability of these neighbours. In order to maintain the delivery guarantee, it has to use the degree of parallelism to compen-

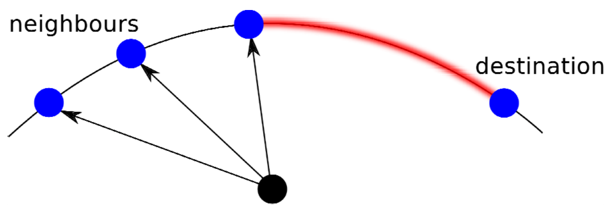

Figure 2. A host forwards multiple lookups to neighbours near the destination key simultaneously.

sate the probability of liveness. Once the node gets acknowledgements from some of the neighbours, it will increase the estimations of their likelihoods of being alive and remove those neighbours that never reply from the routing table. Therefore, the average liveness probability of the routing table is increased by the process, which also reduces the requirement of parallel degree for future lookups. Intuitively, this "refresh" effect neutralizes the risk of continuous whole-system flooding.

In fact, when the average liveness probability reaches an equilibrium between the "refresh" effect, which increases the liveness of neighbours with lookup feedback, and the "aging" effect, which decreases the liveness of neighbours with no recent contact, the liveness probability p will have limited impact on overall cost of parallel lookups. If $p > q$, this statement is obviously true since no parallel lookup is required; if $p \le q$, the equilibrium of p implies:

$$p = \left(\frac{s - dk}{s} \cdot p + \frac{dk}{s} \cdot 100\% \right) \cdot (1 - r) \tag{2}$$

where s is the size of routing table, k is the number of lookup requests in a unit of time, r is node failure rate and d is the degree of parallelism. Then, the cost parallel lookups is:

$$dk = \frac{sr}{1-r} \cdot \frac{p}{1-p} \le \frac{sr}{1-r} \cdot \frac{1}{1-q} \tag{3}$$

This equation shows that, no matter how much the probability p is, the cost dk is always bounded by other factors, such as, the size of routing table and churn rate. Therefore, even with a low level of lifeness probability, the guarantee of lookup delivery rate in Tambour will not overload the system in long term.

Equation (3) also implies that a Tambour node has to maintain a smaller routing table under heavy churn to constrain the control overhead. Since churn leads to evictions of neighbours and decreases the size of the routing table, there is a natural tendency which limits the overhead of parallel lookups. To make this process go smoother and avoid massive amount of lookups under churn, Tambour gives higher priority to nodes with low expected latency when node failure rate increases. It does not do so by adding more lottery tickets to more reliable nodes but by increasing the size of the candidate pool. For example, when the churn is low, Tambour might only pick next hop from 8 nodes near the destination; when the network becomes more unstable, Tambour would pick from 16 candidates. With a larger candidate pool, stabler neighbours get higher probability to be picked, and neighbours with low probability of being alive have little chance to be "refreshed" and will be removed from the routing table through the evicting process eventually. This method has the similar effect as removing unstable nodes from routing table directly. Its advantage is that, if the high level of churn ends in a short period of time, it will not lose much information about those unstable nodes, which helps the system recover from the churn faster.

3.3. Routing Table Maintenance

A Tambour node collects the majority of its routing entries from lookup traffic passing by. However, the senders' keys of normal lookup traffic do not follow small-world distribution as Tambour need. Hence, it adopts two methods to correct the routing entry distribution.

First, a Tambour node piggybacks several routing entries, which follow a small-world distribution from the recipient's point of view, in each lookup and acknowledgement. And, similar to the lottery algorithm used in parallel lookups, the extra routing entries are randomly picked with priority to stable and low-latency neighbours. Second, Tambour explores actively for new neighbours according to the small-world distribution. For every time interval, a Tambour node asks a neighbour for routing entries in the ID range between that neighbour and the very next neighbour in the circular ID space. Since the neighbour is closer to the range than the current node, it knows more routing entries in the range. With this extra knowledge, the neighbour answers with the nodes which have lowest latency to the sender according to Vivaldi coordinate system.

The selection of which ID range to explore is based on whether it needs new neighbours in that range to match the small-world distribution or how much the latency could be improved with new neighbours. According to the small-world distribution, the number of nodes between neighbours i and $i+1$ should be proportional to,

$$\int_{a_i}^{a_{i+1}} \frac{1}{x} \mathrm{d}x = \log \frac{a_{i+1}}{a_i} \tag{4}$$

where a_i, a_{i+1} are the ID distances from current node to neighbour i and neighbour $i+1$ respectively. On the other hand, the level of latency improvement with new neighbours is characterized by the ratio of the current expected lookup latency $\mathrm{E}[t_i]$ for the range to the expected latency $\mathrm{E}[t_i']$ of new neighbours. By the nature of Tambour's parallel lookup algorithm, $\mathrm{E}[t_i]$ is the expected latency of the several neighbours near the ID range forwarding lookup in parallel, and its value can be calculated from Equation (1). Since nodes reply exploration requests with the low-latency entries based on their best knowledge, $\mathrm{E}[t_i']$ is approximately equal to the lowest latency of nodes in that range. Generally, the more nodes are in a set, the shorter latency can be found.

In this case, where the system is supposed to be deployed on the surface area of Earth, the lowest latency is approximately proportional to $\left(a_{i+1}-a_i\right)^{-\frac{1}{2}}$ [20], and so is $\mathrm{E}\left[t_i'\right]$.

Combining above analysis, Tambour selects an ID range to explore randomly with a lottery algorithm which assigns the *i*th range a number of lottery tickets with following formula:

$$\mathrm{E}\left[t_i\right]\cdot\sqrt{a_{i+1}-a_i}\cdot\log\frac{a_{i+1}}{a_i} \tag{5}$$

This exploration process approximates a $\frac{1}{x}$ distribution with the consideration of locality. It keeps hop-count low while providing good latency level for each hop to any part of the ID space.

In a comparatively stable and lookup-request intensive network, a Tambour node could learn enough information about its neighbours through monitoring bypassing lookup traffic and need no active exploration. In this case, further searching for new neighbours is unnecessary and a waste of bandwidth, thus, it is important to know under what condition active exploration is worth the cost and when it should stop. Intuitively, the more neighbours a node have, the more probable a node could find low latency neighbours near the destination in the ID space; And if a node has already learnt many low latency neighbours in a small ID range, it is unlikely to find better neighbours. These observations can be quantified by the following theorems.

Theorem 1 *Suppose nodes are uniformly distributed on Earth and the median latency between each pair of nodes is* t_0. *For two random nodes, the probability of the latency between them being shorter than* x *is* $\frac{1}{2}\left(1-\cos\frac{x\pi}{2t_0}\right)$.

Proof The latency between a pair of nodes is positively correlated with the geographical distance between them [1]. The reason behind this phenomenon is that latency is limited by the speed of electronic signal transmission, meanwhile the data packets on the Internet is usually routed with shortest path. Therefore, if the latency between a pair of nodes is x, then,

$$\frac{\frac{\pi}{2}r}{t_0}=\frac{d}{x}, \tag{6}$$

where r is the radius of Earth and d is the distance between the two nodes.

Suppose the angle between the two nodes is α, then Equation (6) can be written as,

$$\alpha=\frac{d}{r}=\frac{x\pi}{2t_0} \tag{7}$$

Thus, the probability of the latency shorter than x is,

$$\begin{aligned}\Pr\left(\text{latency}<x\right)&=\frac{2\pi rh}{4\pi r^2}=\frac{r-r\cos\alpha}{2r}\\&=\frac{1}{2}\left(1-\cos\frac{x\pi}{2t_0}\right)\end{aligned} \tag{8}$$

A node should check whether it is worth to search for new neighbours with Theorem 1 before doing any probe. If current neighbours are good enough and the probability of getting better neighbours is low, the node does not have to waste bandwidth on active discovery. This indicator is useful for deciding when to update neighbour globally, however, the more important application is to find out which ID regions of neighbours are worth to update. With a limited bandwidth budget, Theorem 1 can be used to improve the cost effective of the bandwidth.

However, sometimes the information, which Theorem 1 relies on, *i.e.*, the median latency of the network and the latency to neighbours, is not available or unreliable. For example, before the first contact with a new neighbour, a node can only estimate its latency with geolocation information which is not always accurate. The following theorem is suitable for such scenario.

Theorem 2 *Suppose nodes are uniformly distributed on Earth. The expected minimal distance between a node and* k *other random nodes is approximately equal to* $\frac{r\sqrt{\pi}}{\sqrt{k+1}}$, *where* r *is the radius of Earth.*

Proof From the proof of Theorem 1, we know that,

$$F\left(x\right)=\Pr\left(d<x\right)=\frac{1}{2}\left(1-\cos\frac{x}{r}\right) \tag{9}$$

So the probability of having at least one node in k nodes whose distance is smaller than d is,

$$F_k\left(x\right)=1-\left(1-F\left(x\right)\right)^k \tag{10}$$

By the definition, the mathematical expectation of the minimal distance is,

$$\begin{aligned}\mathrm{E}\left[\min\left(d\right)\right]&=\int_0^{+\infty}x\cdot f_k\left(x\right)\mathrm{d}x\\&=\int_0^{\pi r}x\cdot F_k'\left(x\right)\mathrm{d}x\\&=x\cdot F_k\left(x\right)\Big|_0^{\pi r}-\int_0^{\pi r}F_k\left(x\right)\mathrm{d}x\\&=\pi r-\int_0^{\pi r}1-\left(1-F\left(x\right)\right)^k\mathrm{d}x\\&=\int_0^{\pi r}\left(1-F\left(x\right)\right)^k\mathrm{d}x\\&=\int_0^{\pi r}\frac{1}{2^k}\left(1+\cos\frac{x}{r}\right)^k\mathrm{d}x\end{aligned} \tag{11}$$

Let $\alpha = \frac{x}{r}$, then

$$\mathrm{E}\left[\min(d)\right] = r\cdot\int_0^{\pi}\left(\frac{1+\cos\alpha}{2}\right)^k \mathrm{d}\alpha$$

$$= r\cdot\int_0^{\pi}\left(\frac{1+2\left(\cos\frac{\alpha}{2}\right)^2-1}{2}\right)^k \mathrm{d}\alpha$$

$$= r\cdot 2\int_0^{\pi}\left(\cos\frac{\alpha}{2}\right)^k \mathrm{d}\frac{\alpha}{2}$$

$$= r\cdot 2\int_0^{\pi}\left(\sin\frac{\alpha}{2}\right)^{2\times\frac{1}{2}-1}\left(\cos\frac{\alpha}{2}\right)^{2\left(\frac{1}{2}+k\right)-1} \mathrm{d}\frac{\alpha}{2}$$

Based on the property of beta function:

$$\mathrm{B}(m+1,n+1) \equiv 2\int_0^{\frac{\pi}{2}}(\sin\theta)^{2m+1}(\cos\theta)^{2n+1}\,\mathrm{d}\theta \qquad (12)$$

We have

$$\mathrm{E}\left[\min(d)\right] = r\cdot B\left(\frac{1}{2},\frac{1}{2}+k\right) = r\sqrt{\pi}\,\frac{\Gamma\left(\frac{1}{2}+k\right)}{\Gamma(1+k)} \approx \frac{r\sqrt{\pi}}{\sqrt{k+1}} \qquad (13)$$

Tambour uses Theorem 2 to predict how many nodes a node has to probe before getting a better node as neighbour if latency information is unreliable. With the consideration of bandwidth limit, a Tambour node keeps in mind that if it knows large number of neighbours in a certain ID region, the probability of discover a new neighbour with lower latency is limited, thus it is more cost-effective to probe an ID region with relatively fewer known neighbours.

4 . Performance Evaluation

This section evaluates the performance of a prototype Tambour implementation in a generic P2P protocol simulator, p2psim [21], and compares it with some other DHTs implemented in p2psim, such as, Chord [5] and Accordion [12]. The experiments run on a network of 10,000 nodes. Each node is assigned a coordinator on a 2-dimensional plain, and the round-trip time between each node pair is proportional to their distance on the plain. The average round-trip time is about 170 ms. The node lifetimes follow a Pareto distribution with a mean of 60 minutes ($\alpha = 0.83$ and $\beta = 1560$ sec). Nodes that depart will rejoin under different addresses after an exponentially-distributed interval with a mean of 6 minutes.

Figure 3 plots the trade-offs between average lookup latency and bandwidth overhead in Chord, Accordion and Tambour. In this experiment, Tambour gets the varieties of bandwidth overhead levels from different exploration rate settings. Both Tambour and Accordion could achieve better average lookup latency than Chord no matter how much bandwidth they use. Since Tambour does not "refresh" routing table actively and employs a cost-effective neighbour exploration algorithm, it is not as sensitive to the bandwidth as the other algorithm do, and it still performs well with limited bandwidth.

Figure 4 shows the performance results of the three protocols under various churn rate. To be a fair comparison, these protocols are all tuned to spend about 20 byte per second on communication. All of them display resilience under churns, but, Tambour outperforms the other two in term of efficiency again.

Figure 5 plots the bandwidth costs of Chord, Accordion and Tambour as the lookup load increases. Since Chord does not use parallel lookup, its bandwidth consumption increases linearly with the lookup load. Accor-

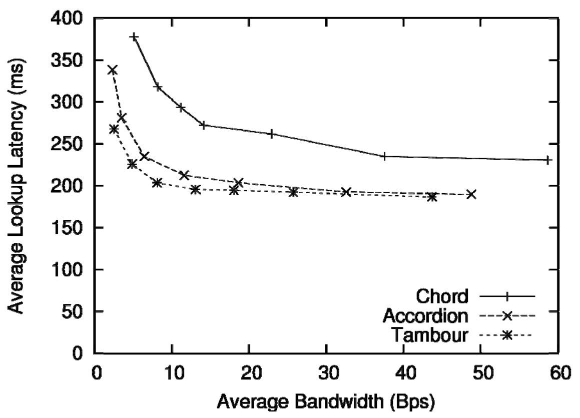

Figure 3. The average lookup latency as a function of the bandwidth overhead for Chord, Accordion and Tambour.

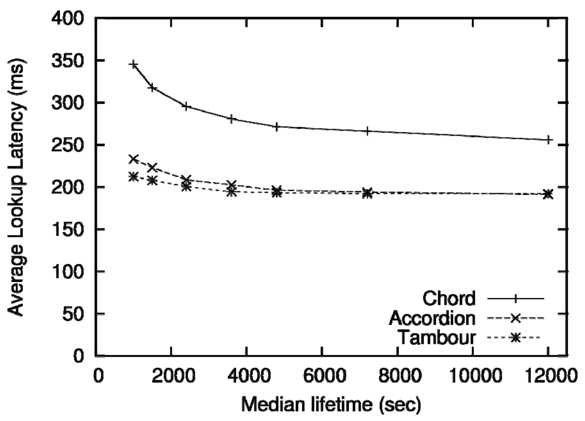

Figure 4. The average lookup latency of Chord, Accordion and Tambour under various levels of churn rate. Their bandwidth overhead are tuned to the same level at about 20 byte/sec.

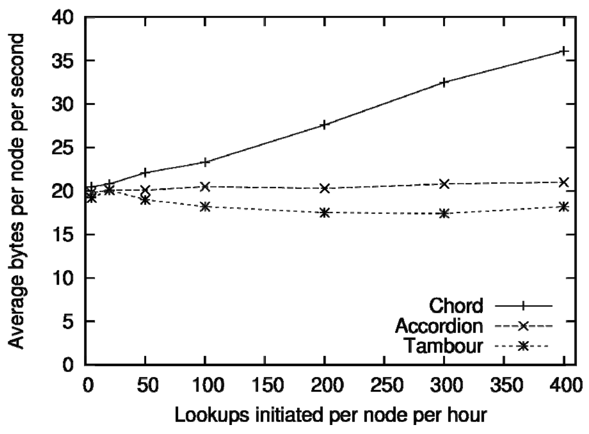

Figure 5. The bandwidth overhead as a function of the lookup load for Chord, Accordion and Tambour.

dion holds its bandwidth consumption at a steady level due to its sophisticated bandwidth budget system. Although Tambour has no direct control over its bandwidth, its parallel lookup algorithm scales well with increasing lookup load. Actually, the extra lookup load brings more routing entries and refreshes neighbour states, as a result, Tambour needs fewer hops, smaller degrees of parallelism and smaller extra overhead.

5. Related Work

As one of the first DHTs, Chord [5] is designed to be scalable over a wide range of system size. Although a latter variation gets a boost from proximity neighbour selection (PNS) [9], the performance is still limited due to its rigid routing tables, especially when bandwidth is comparatively adequate. Tambour has borrowed the ID ring structure and successor pointer stabilization technique from Chord because of their simplicity.

EpiChord [22], a DHT inspired by Chord and various $O(1)$ -hop protocols [9,13-16], demonstrates that a large and flexible routing table combining with parallel lookup technique could be beneficial in a range of lookup workloads. Unlike Tambour, EpiChord only allows a fixed degree of parallelism and use iterative parallel lookup instead of recursive parallel lookup. Iterative parallel lookup avoids the flooding problem raised by multicast, however, it introduces higher latency and reduces the efficiency of proximity neighbour selection.

Symphony [11] and Accordion [12] employ the same small-world distribution [10] as Tambour for populating their routing tables. Symphony has a fixed-sized routing table and acquires its routing entries only from active exploration. Like Tambour, Accordion is more flexible in term of routing maintenance and uses parallel lookup to avoid timeout. In order to restrain the side-effect of parallel lookup, Accordion tunes the degree of parallelism automatically with a complex bandwidth budget system. On the other hand, Tambour achieves a similar result with a simple routing entry picking algorithm without any user-specified parameters.

6. Conclusions

This paper has presented a new scalable and robust DHT protocol, Tambour, which uses parallel lookups to reduce lookup latency and bounds the induced communication overhead automatically as a result of "refresh" effect. Tambour estimates the probabilities of routing entries' liveness based on statistics of node lifetime history and evicts dead entries after lookup failures. When the network is unstable, more routing entries will be evicted in a given period of time, and the routing tables will be getting smaller which minimize the number of timeouts for later lookup requests. Other than the efficiency advantage, Tambour shifts the burden of tuning from the user to the system by automatically adapting itself to the observed churn and workload to provide the best performance.

For future study, it is interesting to see the feasibility of fusing Tambour into other existing DHT systems. Because of the its flexibility and simplicity, a Tambour node should be able to work with DHTs like Chord and Accordion after a little modification. Another possible improvement on Tambour is to utilize the information revealed by the IP address of a Internet host, such as, the timezone and location. These data could help Tambour to make a better predication about a neighbour's availability and latency.

REFERENCES

[1] K. Gummadi, R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker and I. Stoica, "The Impact of DHT Routing Geometry on Resilience and Proximity," *Proceedings of the* 2003 *Conference on Applications*, *Technologies*, *Architectures, and Protocols for Computer Communications*, ACM, New York, 2003, pp. 381-394.

[2] Z. Yang, J. Tian and Y. Dai, "Towards a More Accurate Availability Evaluation in Peer-to-Peer Storage Systems," *Proceedings of the* 2006 *International Workshop on Networking*, *Architecture*, *and Storages*, Shenyang, 2006, pp. 73-80. doi:10.1109/IWNAS.2006.45

[3] M. Zhou, Y. Dai and X. Li, "A Measurement Study of the Structured Overlay Network in P2P File-Sharing Systems," *Advanced Multimedia*, Vol. 2007, No. 1, 2007, pp. 10-18.

[4] S. Saroiu, P. K. Gummadi and S. D. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems," *Multimedia Computing and Networking Conference* (*MMCN*), San Jose, January 2002, pp. 156-170.

[5] I. Stoica, R. Morris, D. Karger, F. F. Kaashoek and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Look up Service for Internet Applications," *Computer Communi-*

cation Review, Vol. 31, No. 4, 2001, pp. 149-160. doi:10.1145/964723.383071

[6] A. Rowstron and P. Druschel, “Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems,” *Lecture Notes in Computer Science*, Vol. 2218, 2001, pp. 329-350. doi:10.1007/3-540-45518-3_18

[7] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph and J. D. Kubiatowicz, “Tapestry: A Resilient Global-Scale Overlay for Service Deployment,” *IEEE Journal on Selected Areas in Communications*, Vol. 22, No. 1, 2004, pp. 41-53. doi:10.1109/JSAC.2003.818784

[8] P. Maymounkov and D. Mazières, “Kademlia: A Peer-To-Peer Information System Based on the XOR Metric,” *IPTPS*, 2002, pp. 53-65.

[9] R. Bhagwan, S. Savage and G. M. Voelker, “Understanding Availability,” *IPTPS*, 2003, pp. 256-267.

[10] J. Kleinberg, “The Small-World Phenomenon: An Algorithmic Perspective,” *Proceedings of the* 32*nd ACM Symposium on Theory of Computing*, ACM, New York, 2000, pp. 163-170.

[11] G. S. Manku, M. Bawa, P. Raghavan and V. Inc, “Symphony: Distributed Hashing in a Small World,” *Proceedings of the* 4*th USENIX Symposium on Internet Technologies and Systems*, 2003, pp. 127-140.

[12] J. Li, J. Stribling, R. Morris and M. F. Kaashoek, “Bandwidth-Efficient Management of DHT Routing Tables,” *Proceedings of the* 2*nd Conference on Symposium on Networked Systems Design & Implementation*, USENIX Association, Berkeley, 2005, pp. 99-114.

[13] I. Gupta, K. Birman, P. Linga, A. Demers and R. V. Renesse, “Kelips: Building an Efficient and Stable P2P DHT through Increased Memory and Background Overhead,” *Proceedings of the* 2*nd International Workshop on Peer-to-Peer Systems*, Berkeley, 2003, pp. 160-169.

[14] R. Rodrigues and C. Blake, “When Multi-Hop Peer-To-Peer Routing Matters,” *The* 3*rd International Workshop on Peer-to-Peer Systems*, La Jolla, 26-27 February, 2004, pp. 112-122.

[15] A. Gupta, B. Liskov and R. Rodrigues, “Efficient Routing for Peer-to-Peer Overlays,” *Proceedings of the* 1*st Conference on Symposium on Networked Systems Design and Implementation*, USENIX Association, Berkeley, 2004, pp. 113-126.

[16] B. Leong and J. Li, “Achieving One-Hop DHT Look up and Strong Stabilization by Passing Tokens,” *Proceedings of the* 12*th International Conference on Networks* (*ICON*), Singapore, November 2004, pp. 344-350.

[17] J. Li, J. Stribling, R. Morris, M. F. Kaashoek and T. M. Gil, “A Performance vs. Cost Framework for Evaluating DHT Design Trade offs under Churn,” *Proceedings of the* 24*th Infocom*, Miami, 13-17 March 2005, pp. 225-236.

[18] S. Saroiu, K. P. Gummadi and S. D. Gribble, “Measuring and Analyzing the Characteristics of Napster and Gnutella Hosts,” *Multimedia Systems*, Vol. 9, No. 2, 2003, pp. 170-184. doi:10.1007/s00530-003-0088-1

[19] F. Dabek, R. Cox, F. Kaashoek and R. Morris, “Vivaldi: A Decentralized Network Coordinate System,” *SIGCOMM*, 2004, pp. 15-26.

[20] H. Zhang, A. Goel and R. Govindan, “Incrementally Improving Lookup Latency in Distributed Hash Table Systems,” *ACM Sigmetrics*, 2003, pp. 114-125.

[21] “A Simulator for Peer-to-Peer (P2P) Protocols,” 2012. http://pdos.csail.mit.edu/p2psim

[22] B. Leong, B. Liskov and E. D. Demaine, “EpiChord: Parallelizing the Chord Lookup Algorithm with Reactive Routing State Management,” *The* 12*th International Conference on Networks* (*ICON*), Singapore, November 2004, pp. 1243-1259.