NEBLO: Anonymity in a Structured Overlay

Giuseppe Ciaccio
DISI, Università di Genova
via Dodecaneso 35
16146 Genova, Italy
ciaccio@disi.unige.it

DISI Technical Report DISI-TR-05-05

11 May 2005

Abstract

NEBLO, a NEarly BLind Overlay, is a structured overlay network in which the use of imprecise routing tables, whose size and extent are severely constrained, yields a better privacy (aka "anonymity") to information requestors as well as providers, while keeping the length of routing paths within reasonable limits. Fingers in routing tables are imprecise, and the amount of approximation for a finger increases proportionally with the finger distance. This keeps routing paths within a logarithmic length, while making it impossible for a "small" coalition of malicious peers to correlate overlay addresses to hosts for censorship or auditing purposes. [1]

Keywords: structured overlay, anonymity, distributed hash table, peer to peer, censorship-resistant

1. Introduction and motivation

Overlay networks have been receiving a lot of attention by the research community, as flexible and scalable low-level infrastructures for distributed applications of many kind: network storage [16, 12, 31], naming [11], content publication [14, 10, 3, 29, 36, 32], multicast [28, 6], and communication security [26]. They have also been proposed as general networking infrastructures [15, 34, 18, 17], because of their potential ability to decouple network addresses from physical placements of cooperating hosts, an important feature for privacy and mobility.

[1] This research is supported by the Italian FIRB project *Webminds*.

The vast population of existing or proposed overlay systems can be broadly divided into two families, namely, unstructured overlays and structured overlays.

Structured overlays [27, 35, 13, 33, 20, 30, 24, 37] are receiving far more attention lately, because of performance guarantees they can in principle provide thanks to their regular topologies. Regular topologies allow routing algorithms to provably converge, and a careful choice of entries in routing tables can reduce the number of routing hops to even a constant quantity, independent of the overlay size [19, 23].

On the other hand, unstructured overlays like [10] and [2] first leveraged techniques to enhance identity privacy or *anonymity* of participant entities. Trivial indirection based on source rewriting and usual cryptographic machinery, or, even better, mixes chains [7, 4], can help hide the identity of a message sender. But there is another face of the coin, namely, recipient anonymity. This has to do with the ability (or, better, the difficulty) to correlate an overlay recipient address to the physical Internet address of the destination host. It must be made difficult for an adversary to build a *map of the overlay*, relating overlay addresses to host addresses.

To build a map of a given part of the overlay, and adversary can only deploy a malicious coalition in the system, then eventually gather and analyze the routing tables of the colluding peers (additional information gathered by routed messages could only be useful if messages carried explicit information about sender or recipient identity). But, the more irregular the overlay, the greater the control the adversary must put over coalition placement, or, alternatively, the larger the coalition itself. The irregularity of unstructured

overlays thus works as a device for recipient anonymity. This is the reason for the success of these networks as a support for censorship-resistant network archives, in which storing and retrieving are recipient roles.

It must be said, however, that recipient anonymity alone is not sufficient for censorship resistance: without the coexistence of sender anonymity, a document could be censored by exerting menace upon that document's creators and/or readers (which play sender roles in a network storage systems, as they send write and read requests to storage hosts). Another point to be stressed is that the irregularity of the overlay shape is paid in terms of efficiency and availability.

That said, there is an obvious interest to find trade offs between the efficiency of structured overlays and the privacy offered by unstructured ones. This indeed is the motivation of the work accounted here.

2. Related work

The problem of ensuring recipient anonymity is hard. As pointed out in Section 1, a common way to hide recipients is to use indirection in an unstructured overlay of unknown shape. This is not perfect, as the peer in the last hop always watches the recipient directly. The solution proposed by [32], in which the public address of a document is decoupled from the overlay placement of the document itself, only succeeds in enforcing so called "active server document anonymity" [14], that is, the inability for any peer to correlate a given (part of a) document to any information kept in its own local store. This is not enough: other peers could make such a correlation (it is necessary in order to retrieve the document), and then menace or attack the storer to censor its content. Actually, we do not believe perfect recipient anonymity be achievable; we prefer to seek "pretty good" anonymity, that is, anonymity that cannot be broken by a "small" coalition of distributed resources.

There are very few attempts to improve anonymity in structured overlays. Achord [21] is a study aimed at enhancing Chord with censorship-resistant features. Aiming at enforcing both sender and recipient anonymity, Achord implements recursive-style routing (because of the indirection), forces responses to travel back to sender in a recursive way (for the very same reason), and restricts each peer's ability to know other peers (so that a small coalition could not easily map the entire overlay). With Achord, any peer is allowed to know the IP addresses of at most $k * \log(N)$ other peers, where k is the finger table size. This is insufficient for recipient anonymity, as the entire ring could be mapped (and the anonymity of any recipient could thus be broken) by a coalition of $N/k * \log(N)$ peers, which is asymptotically infinitesimal w.r.t. N and therefore "small".

An unexplored aspect in Achord is related to the rule for the insertion point of new peers in the ring. Any rule only based on the pair $\langle IPaddress, port \rangle$ of the newcomer is weak, because it gives an adversary some rough control on which positions to hold in the overlay.

Achord is the only work we are aware of concerning recipient anonymity in a structured overlay. Other studies [22, 25] only focus on sender anonymity in Chord, which however seems to be already good in terms of size of anonymity set [25].

The issue of finding a valid metric for anonymity is not yet fully understood and is actively researched upon [2, 5].

3. Routing tables and anonymity

For the sender anonymity, it is clear that any effort devoted to minimizing the routing path length is detrimental. Indeed, too few routing hops decrease the effectiveness of the mixing technique, which work well when more mixes are chained. Moreover, short routing paths imply large routing tables, which means that an adversary controlling a small number of peers can quickly map a large portion of the overlay. Thus, imposing severe constraints on the maximum size of routing tables is necessary to improve anonymity. However this may not be sufficient. For instance, in a Chord-like overlay of N peers, routing tables as small as $O(\log(N))$ would allow an adversarial coalition of $O(N/\log(N))$ peers to map the entire ring of hosts, which means that an asymptotically infinitesimal (w.r.t. N) coalition is potentially able to break anonymity of an arbitrary recipient.

The size of a coalition devoted to map an entire overlay of N peers is upper bounded by N/S_r, where S_r is the size of the routing tables. The ratio between coalition size and overlay size is thus $1/S_r$. If S_r is allowed to increase with N, then the coalition is asymptotically infinitesimal and thus "small". If, however, S_r is fixed, then either the average routing path length becomes $O(N)$ when N is large (loss of scalability), or S_r is determined as a function of the size of the overlay address space, which N approaches to, and thus the coalition becomes asymptotically "small" again. Apparently there is no way out for recipient anonymity.

Our idea is that a routing table should be allowed to grow large enough to allow $O(\log(N))$ routing, but not larger, in order to preserve sender anonymity through a sufficient degree of indirection on communications.

In addition, a routing table should only be allowed to contain a small and fixed amount of *exact information*, whereas most of the information in the table should be *imprecise*. Indeed, only exact information can contribute to map the overlay, so if its amount is kept small and fixed in each routing table, a coalition of peer wishing to map the overlay should be "large". This way, recipient anonymity can be preserved.

Imprecise routing information is at the core of unstructured overlays. With Freenet, for example, a message directed to key A is routed towards a node P if P has previously been able to route back responses from keys "similar" to A [10]. Thus, an entry in the routing table pointing to P does not say anything about the keys actually stored at P, nor does it imply much about the placement of P in the overlay topology. GNUnet and MUTE follows a similar approach, with some more randomness [2, 29].

Having imprecise entries in the routing table raises questions concerning convergence and efficiency of the routing algorithm. The routing paths could become longer, or be unsuccessful. In the following we stem from a ring-shaped overlay in which each peer has a successor and some fingers, as in Chord, and propose a way to manage successors and fingers so that convergence is ensured, efficiency is largely preserved, but the routing tables of peers are of no help for an adversarial coalition to map the overlay.

3.1. Successors

Let us consider a set of peers organized according to a ring topology. Each peer has a *successor* in the topology.

The *overlay address space* is the set of 2^k binary words of k bits, ordered as a circle modulo 2^k. This space is mapped onto the ring of peers in consecutive chunks or *address intervals*. If peer P is responsible for the address interval from A_l to A_u, and peer Q is the successor of P, then all addresses of Q are greater than A_u (modulo 2^k). In the ideal scenario in which the map is complete, that is, no "hole" is left between each peer and its successor, a request for address A can always reach the recipient (the peer whose address interval contains A) by traversing the successor chain starting from the sender. In a realistic scenario, however, a faulty or disconnected peer could break the successor chain and also create a "hole", a discontinuity in the address space. Routing towards backup locations and a suitable degree of redundancy are not enough: the system must also quickly seal the successor chain and restore the address hole, or redundancy would eventually degrade. To this end, an easy solution is to allow each peer to know a *successor list*, rather that just the immediate successor. This allows a peer to talk directly to its successor's successor to seal the ring in case the successor has gone (the extension to the case of multiple adjacent faulty peers is straightforward).

Note that, for the routing to take place along the successor chain, each peer is *not* required to know the address interval of any member in its own successor list, not even the immediate successor. The knowledge of the locally owned address interval is indeed sufficient to decide whether a request can be satisfied locally or has to proceed on. This is good news for recipient anonymity.

On the other hand, a peer P wishing to seal a hole ahead of it is required to explore its own successor list, find the member Q immediately ahead of the hole, and ask Q to reveal the lower bound of its address interval; this way only, could P extend its own interval to span the hole. To ensure recipient anonymity, however, Q should never disclose its own interval to anybody. We clearly need a compromise, and propose that Q only reveals something to P if Q verifies that its own *predecessor* in the ring has gone and that P is found along its own *predecessor chain*.

To sum up, recipient anonimity requires that each peer knows and manages a predecessor list, in addition to the successor list. Moreover, each peer may be asked to reveal the lower bound of its own address interval to one of its predecessors, in case a hole has opened immediately past of it in the ring. The information leak caused by the occasional disclosure of addressing information to a predecessor, however, cannot be exploited by an adversarial coalition to map the overlay: more and more knowledge can only be gained at the price of opening more and more holes in the ring, perhaps killing more and more peers. The topological information found in each successor list is exact (and has to be kept so through periodical checks), but it contributes very little against recipient anonymity: the only threat comes from the fact that each peer indirectly knows the lower bound of its successor's address interval. This cannot be eliminated, but the achieved degree of recipient anonymity is still "pretty good" according to our informal definition given in Section 2.

3.2. Fingers and their anonymity issues

For distant recipients we need an alternate routing algorithm, so as to keep the routing path length below an acceptable size. A common such mechanism is given by the *fingers*. We present here a generalized version of the concept originally introduced by [35].

A fingers is a descriptor pointing to a "distant" peer

in the overlay. The distance is measured between (the lower bound of) the local address interval and (the lower bound of) the remote address interval. Each peer has its own list of fingers, the elements of which are ordered by increasing distance.

Finger distances must match a mathematical requirement that we call the *distance rule*. The distance rule is often geometric on base 2. Given a bottom value C, called *cutoff*, the first finger has the largest possible distance $\leq C$ from local peer, the second finger has the largest possible distance $\leq 2C$, the third finger has distance $\leq 4C$, and so on, up to spanning half of the address space. The finger at distance $C \cdot 2^m$ is said to have *magnitude* m; we will also call it the "finger m" for brevity.

The routing algorithm takes advantage of fingers in the following way: after computing the residual distance D to be travelled by a request, a peer chooses the finger of largest magnitude whose distance does not exceed D, and forwards the request to it. If no such finger is found, the peer forwards to successor.

In an overlay with complete finger lists conforming to a geometric distance rule, a total travel distance of D is covered in two phases, namely, $O(\log_2(D/C))$ finger hops until the residual distance falls shorter than the cutoff C, then an $O(1)$ number of short-range hops through the successor chain.

The quickest way to build and maintain a finger list takes advantage from the recursive nature of the geometric distance rule. To find the first finger, of magnitude 0, P can send a suitable request along its successor chain, perhaps in a recursive style, until the most distant peer still within cutoff C is found. To find a finger of magnitude $m > 0$, P can ask its own current finger $m-1$ to be put in contact with its finger $m-1$. [2] Such an *incremental procedure* minimizes the number of contacted peers, so it should be preferred when anonymity is of concern, because it can minimize the information leak towards potential adversaries.

However, the above (traditional, after Chord) definition of fingers poses two serious issues in terms of recipient anonymity, namely:

1. If peer P has peer Q as its own finger of magnitude m, then P knows that Q's address interval is more or less at distance $C \cdot 2^m$ from itself. Thus, Q's address interval is indirectly exposed to P. In general, in a ring counting N peers, each participant can map the address intervals of other $O(\log(N))$ peers. A malicious coalition counting $O(N/\log(N))$ peers, which is asymptotically infinitesimal, can thus map the entire overlay, as already pointed out in Section 3.

2. When searching the first finger, peer P exposes its own address interval to the whole successor chain up to first finger, which is bad for recipient anonymity, especially if the successor lists are long.

[2] In case the address interval of P spans the entire cutoff distance, the finger of magnitude 0 could not be found. In this case P starts by directly searching its finger of magnitude n along successor chain within distance $C \cdot (n+1)$, where n is such that $C \cdot (n+1)$ is larger than the size of P's address interval.

3.3. Improving anonymity with imprecise fingers

The two anonymity flaws outlined at the end of last Section are impossible to fix, because they are implied by the traditional definition of fingers. To improve recipient anonymity we must shift to a slightly different definition of fingers.

Our goal is to obfuscate part of the topological information conveyed by traditional fingers, and to protect peers against excessive exposure when searching fingers of magnitude 0.

In the following, F_P is a secret random value generated by the generic peer P with uniform probability over $[0, C/4[$. F_P thus cannot exceed $C/4$.

To build its own finger list, let us suppose peer P use the incremental procedure outlined in Section 3.2. The first step in the procedure is to find the finger of magnitude 0. As already pointed out, finding finger 0 potentially exposes the address interval of P to the whole successor chain, up to finger 0, because of the need for all successors to compute their distance from P. To fix such information leak, P acts as follows: rather than sending (the lower bound of) its own address interval to the successor, P alters the information by subtracting F_P, then sends the altered value A.

Let Q be a generic peer in P's successor chain. After receiving an altered value A from its predecessor, it acts as follows:

1. Q evaluates (the lower bound of) its successor's address interval, L; this is trivial, as the successor is adjacent in the address space.

2. Q computes the distance between $A - F_Q$ and L, namely, $L - A + F_Q$. This amounts to computing the distance between the original requestor, namely P, and Q itself, altered by the sum of the two random secrets F_P and F_Q; the computed distance is thus lower than the real one by an unknown quantity in $[0, C/2[$, since each random secret is in $[0, C/4[$.

3. If such distance is greater than C, then Q contacts the original requestor P and claims to be its finger

0.

4. Otherwise Q's successor is a better candidate, so Q forwards the value A to its successor.

Since the request emitted by P is initially altered by the random quantity F_P, P is not disclosing its own addressing information to the successors. When P is eventually contacted by a peer Q claiming to be its finger of magnitude 0, P can conclude that the distance of Q is the largest possible within an upper bound randomly distributed between $C - C/2 = C/2$ and C. This is a substantial departure from the traditional definition of fingers, according to which the distance of finger 0 is known with far greater precision (the largest distance not exceeding C).

The approximation introduced on finger 0 cumulates over fingers of greater magnitude. Using the incremental procedure, a finger of magnitude 1 is imprecise because its distance is close to a value between $2 \cdot (C/2) = C$ and $2C$, a finger of magnitude 2 is imprecise because its distance is close to a value between $4 \cdot (C/2) = 2C$ and $4C$, and so on. The generic finger m is located at a distance which is the largest possible not exceeding an unknown random value between $C \cdot 2^{m-1}$ and $C \cdot 2^m$.

Such an amount of finger impreciseness is a good device for recipient anonymity. The more distant a finger, the less a peer can know about the overlay addresses it actually owns. A lot of indeterminacy can be put on even the closest of fingers, provided the cutoff C be large compared to the average size of address intervals of peers, as in highly populated rings. No matter how large, no adversarial coalition can gather sufficiently exact information from finger lists; on the other hand, the successor lists have already be shown to be poor of useful information for an adversary (see Section 3.1).

One could argue that, since an N-fold convolution of uniform probability distributions exhibits an evident modal value which is also the expectation, a peer P could get a meaningful statistical estimation of the distance of its fingers. Following [5], we could say that in such a situation the anonymity is weaker because of a lower entropy. On the other hand, from the above algorithm we know that finding the finger of magnitude 0 only requires the sum of two uniform random variables, and finding a finger of magnitude $m > 0$ only requires traversing the m fingers of lower magnitude and thus requires the sum of $2m$ uniform random variables. In other words the number of convoluted uniform random variables is always small (typically, the number of fingers is in the order of two or three dozens) and thus the standard deviation of the impreciseness on each finger keeps large enough to ensure a pretty good entropy.

3.4. Imprecise fingers and routing performance

It only remains to assess the price of anonymity in terms of performance. We may expect that the routing algorithm converges (because the successor relation is kept exact), but within a greater number of hops.

Traversing a generic finger of magnitude m reduces the residual distance by no more than $C \cdot 2^m$, as we have seen in Section 3.2. With the imprecise fingers defined as in Section 3.3, however, a hop through a finger of magnitude m is shorter on average. At worst, hopping through an imprecise finger of magnitude m decreases the residual distance by no more than $C \cdot 2^{m-1}$. Therefore, to accomplish a distance decrease of $C \cdot 2^m$, it is necessary to traverse no more than two consecutive fingers of same magnitude m.

We thus conclude that the use of imprecise fingers as defined in Section 3.3 at most doubles the total number of traversed fingers. In a system with complete finger lists, the routing paths thus remains $O(\log_2(D/C))$.

Actually, if the random secret F_P owned by the generic peer P was randomly distributed over a smaller fraction of the cutoff C, the routing paths with imprecise fingers would be shorter. We have chosen to let F_P span half of the cutoff because this way all the formulas were simpler, without sacrificing the generality of the result.

4. Conclusions

Our result can be summarized by saying that anonymous routing can be accomplished even in a structured overlay, and can be done in $O(\log(N))$ hops where N is the number of peers in the overlay. If we liked slogans, we would say that anonymity can be efficient.

This is the theory. In practice, the actual impact of imprecise fingers on average routing paths remains to be evaluated; our simple analysis is just a worst case one.

Of course, a substantial growth of the average routing paths raises performance and availability concerns: latency-sensitive applications might suffer, and longer routing paths are also more sensitive to the failure probability of individual peers. By tuning some parameters (the span of the random secret) it is possible to obtain shorter routing paths, approaching the ones generated by an ideal routing, but with some loss of anonymity. Trade-offs between efficiency, anonymity and availability deserve further investigation, perhaps based on simulations.

Yet, we claim this result is a good step in a proper direction. Anonymity, efficiency and availability can

coexist in the same distributed system, on the ground of a structured overlay.

5. NEBLO: a working prototype

NEBLO, a NEarly BLind Overlay, is a working implementation of the principles accounted in this paper. It is a structured overlay network organized as a ring with fingers, in which the use of imprecise finger lists, whose size and extent are severely constrained, yields a pretty good anonymity to information requestors as well as providers. A description of the system and protocol is out of the scope of this paper; see [8] for more details.

NEBLO implements successor and predecessor lists as described in Section 3.1. A periodic maintenance thread is in charge of probing liveness of the successor, in order to seal the ring again whenever necessary. To this end, an up-to-date successor list is of great importance. In order to keep the successor lists as much updated as possible, each peer periodically propagates its current successor list backward to the predecessor (after having removed the last element, so that the lists cannot grow beyond a fixed limit). This way, a newcomer eventually meets (and is met by) all peers in its predecessor list. Moreover, when a peer joins the ring, all peers in its successor list are immediately notified about the new member, so they can update their predecessor lists in real time. This ensures that the protocol for ring sealing also works in presence of newcomers that are too recent to have been propagated to their respective predecessors, because they are however known as predecessors of some successor of the failed peer.

NEBLO implements the incremental procedure of Section 3.2 for building the finger lists, with additional security features. For the initial request aimed at meeting the finger of magnitude 0, the system explicitly ensures that such a request may only travel along successor chains. To this end, each recipient checks if the request comes from a peer in its own predecessor list, and discard the request if it is not so. For fingers of magnitude greater than 0, the system explicitly limits the scope of those requests. A request for finding finger m only makes sense if sent to current finger $m-1$, and is considered malicious otherwise. To this end, each peer P maintains an *inverse finger list*, that is, a list of all other peer having P as one of their fingers, together with the corresponding magnitude. A request for finding a finger of magnitude m is only accepted if coming from an inverse finger of magnitude $m-1$, and is discarded otherwise. Periodically, the entire finger list is rebuilt from scratch to take account of membership changes in the overlay.

In the current implementation, all communications between couples of peers take place through TCP sockets, and are cyphered by a session key, exchanged in a pretty secure way (RSA) at the time of establishing the connection. Indirection implements sender anonymity through source rewriting. However, a number of traditional anonymity devices are still missing: mixing, message padding to standard size, and noise traffic, will be eventually added.

The join protocol is still weak in terms of security. It is not based on hashing the IP address and port of the newcomer. The position in the ring of a newcomer P is computed as follows:

1. P contacts a peer Q which is already member of the overlay;
2. P and Q negotiate a number I in a such a way that neither P nor Q could predetermine the result [10];
3. Q computes the cryptographic hash $H = RIPEMD160(I)$; and finally
4. Q emits a join request towards overlay address H, containing the IP address, port, and RSA public key of P, together with a random identificator R_i for this request and the originally negotiated number I. The identificator R_i is also made known to P.

The recipient of the join request will welcome the newcomer P by sending it R_i, so that P can check it is not a spoofed welcome; but it will only do so after having checked that the equality $H = RIPEMD160(I)$ also holds on its own side. This check makes it more difficult for an adversary to determine which positions to take in the overlay, as it should invert a cryptographic hash. Nevertheless the method is weak, because an adversary could always use a partial dictionary of the hash function, which might be feasible, and decide its own position in the overlay with an approximation of some bits of address. We are investigating the feasibility of other, more robust join protocols.

NEBLO is presented to distributed applications in the form of a runtime library with a simple API. Details can be found in [8] and [9].

NEBLO is free software, released under the GNU General Public Licence and available for download at [1].

References

[1] The NEBLO homepage (in preparation), http://www.disi.unige.it/project/neblo/.

[2] K. Bennett and C. Grothoff. GAP: Practical Anonymous Networking. In *Proc. of Workshop on Privacy Enhancing Technologies (PET 2003)*, Dresden, Germany, Mar. 2003.

[3] K. Bennett, C. Grothoff, T. Horozov, and I. Patrascu. Efficient Sharing of Encrypted Data. In *Proc. of ACISP 2002*, pages 107–120. Springer-Verlag, July 2002.

[4] O. Berthold, H. Federrath, and S. Köpsell. Web MIXes: A system for anonymous and unobservable Internet access. In H. Federrath, editor, *Proc. of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 115–129. Springer-Verlag, LNCS 2009, July 2000.

[5] N. Borisov and J. Waddle. Anonymity in Structured Peer-to-Peer Networks. gnunet.org/papers/borisov_waddle.pdf, Dec. 2003.

[6] M. Castro, P. Druschel, A. M. Kermarrec, and A. Rowstron. Scribe: A Large-scale and Decentralized Application-level Multicast Infrastructure. *IEEE Journal on Selected Areas in Communications, special issue on Network Support for Multicast Communications*, 20(8), Oct. 2002.

[7] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 4(2), Feb. 1981.

[8] G. Ciaccio. A NEarly BLind Overlay. Technical Report in preparation, DISI, Università di Genova, 2005.

[9] G. Ciaccio. A Pretty Flexible API for Generic Peer-to-Peer Programming. Technical Report DISI-TR-05-06, DISI, Università di Genova, 2005.

[10] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Proc. of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 46–66, July 2000.

[11] R. Cox, A. Muthitacharoen, and R. Morris. Serving DNS using a Peer-to-Peer Lookup Service. In *Proc. of the 1st International Peer To Peer Systems Workshop (IPTPS02)*, Mar. 2002.

[12] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area Cooperative Storage with CFS. In *Proc. of 18th ACM Symp. on Operating Systems Principles*, Oct. 2001.

[13] F. Dabek, J. Li, E. Sit, J. Robertson, M. F. Kaashoek, and R. Morris. Designing a dht for low latency and high throughput. In *Proc. of the 1st USENIX Symposium on Networked Systems Design and Implementation (NSDI '04)*, San Francisco, California, Mar. 2004.

[14] R. Dingledine, M. J. Freedman, and D. Molnar. The free haven project: Distributed anonymous storage service. In H. Federrath, editor, *Proc. of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*. Springer-Verlag, LNCS 2009, July 2000.

[15] J. Eriksson, M. Faloutsos, and S. Krishnamurthy. PeerNet: Pushing Peer-to-Peer Down the Stack. In *Proc. of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS03)*, Berkeley, CA, 2003.

[16] J. K. et al. Oceanstore: An Architecture for Global-scale Persistent Storage. In *Proc. of ACM ASPLOS*, Nov. 2000.

[17] M. J. Freedman and R. Morris. Tarzan: A peer-to-peer anonymizing network layer. In *Proc. of the 9th ACM Conference on Computer and Communications Security (CCS 2002)*, Washington, DC, Nov. 2002.

[18] I. Goldberg. *A Pseudonymous Communications Infrastructure for the Internet*. PhD thesis, UC Berkeley, Dec. 2000.

[19] A. Gupta, B. Liskov, and R. Rodrigues. One hop lookups for peer-to-peer overlays. In *Proc. of the 9th Workshop on Hot Topics in Operating Systems (HotOS-IX)*, Lihue, Hawaii, May 2003.

[20] I. Gupta, K. Birman, P. Linga, A. Demers, and R. van Renesse. Kelips: Building an Efficient and Stable P2P DHT Through Increased Memory and Background Overhead. In *Proc. of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS03)*, Berkeley, CA, 2003.

[21] S. Hazel and B. Wiley. Achord: A Variant of the Chord Lookup Service for Use in Censorship Resistant Peer-to-Peer Publishing Systems. In *Proc. of the 1st International Peer To Peer Systems Workshop (IPTPS02)*, Mar. 2002.

[22] J. Kannan and M. Bansal. Anonymity in Chord. www.cs.berkeley.edu/ kjk/chord-anon.ps, Dec. 2002.

[23] B. Leong and J. Li. Achieving one-hop dht lookup and strong stabilization by passing tokens. In *Proc. of the 12th International Conference on Networks (ICON)*, Nov. 2004.

[24] G. S. Manku, M. Bawa, and P. Raghavan. Symphony: Distributed Hashing in a Small World. In *Proc. of the fourth USENIX Symposium on Internet Technologies and Systems (USITS'03*, Seattle, WA, Mar. 2003.

[25] C. O'Donnell and V. Vaikuntanathan. Information Leak in the Chord Lookup Protocol. In *Proc. of the 4th IEEE Int.l Conf. on Peer-to-Peer Computing (P2P2004)*, Zurich, Switzerland, Aug. 2004.

[26] P. Perlegos. DoS Defense in Structured Peer-to-Peer Networks. Technical Report UCB-CSD-04-1309, U.C. Berkeley, Mar. 2004.

[27] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-Addressable Network. In *Proc. of ACM SIGCOMM*, Aug. 2001.

[28] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Application-level Multicast Using Content-addressable Networks. In *Proc. of 3rd Int.l Workshop on Networked Group Communication*, Nov. 2001.

[29] J. Rohrer. MUTE: Simple, Anonymous File Sharing. http://mute-net.sourceforge.net/.

[30] A. Rowstron and P. Druschel. Pastry: Scalable, Distributed Object Location and Routing for Large-scale

Peer-to-peer Systems. In *Proc. of Int.l Conf. on Distributed System Platforms*, Nov. 2001.

[31] A. Rowstron and P. Druschel. Storage Management and Caching in PAST, a Large-scale, Persistent Peer-to-peer Storage Utility. In *Proc. of 18th ACM Symp. on Operating Systems Principles*, Oct. 2001.

[32] A. Serjantov. Anonymizing censorship resistant systems. In *Proc. of the 1st International Peer To Peer Systems Workshop (IPTPS02)*, Mar. 2002.

[33] A. Serjantov. Kademlia: A Peer-to-peer Information System Based on the XOR Metric. In *Proc. of the 1st International Peer To Peer Systems Workshop (IPTPS02)*, Mar. 2002.

[34] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana. Internet Indirection Infrastructure. In *Proc. of ACM SIGCOMM'02*, Aug. 2002.

[35] 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. of ACM SIGCOMM*, Aug. 2001.

[36] M. Waldman, A. Rubin, and L. Cranor. Publius: A robust, tamper-evident, censorship-resistant and source-anonymous web publishing system. In *Proc. of the 9th USENIX Security Symposium*, pages 59–72, Aug. 2000.

[37] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. Tapestry: An Infrastructure for Fault-resilient Wide-area Location and Routing. Technical Report UCB-CSD-01-1141, U.C. Berkeley, Apr. 2001.