Towards Application-Aware Anonymous Routing

Micah Sherr Boon Thau Loo Matt Blaze
University of Pennsylvania

Abstract

This paper investigates the problem of designing anonymity networks that meet application-specific performance and security constraints. We argue that existing anonymity networks take a narrow view of performance by considering only the strength of the offered anonymity. However, real-world applications impose a myriad of communication requirements, including end-to-end bandwidth and latency, trustworthiness of intermediary routers, and network jitter.

We pose a grand challenge for anonymity: the development of a network architecture that enables applications to customize routes that tradeoff between anonymity and performance. Towards this challenge, we present the *Application-Aware Anonymity (A^3)* routing service. We envision that A^3 will serve as a powerful and flexible anonymous communications layer that will spur the future development of anonymity services.

1 Introduction

Today's Internet routing protocols, while arguably robust and efficient, are not designed to support anonymous communication. To be routable, packets must include source and destination addresses, and any forgeries or omissions hinders deliverability and reliability. While there have been several attempts at providing anonymity with the use of application-level overlay networks, these solutions typically come at the expense of network performance, and do not consider the diverse requirements of potential client applications.

For example, in the case of decentralized anonymity techniques in which next-hops are selected by routers ("jondos") in the network (e.g., Crowds [20] and Hordes [21]), there is no intrinsic support for favoring paths that meet certain application constraints. While source-routing techniques such as Tor [11] and onion routing [19] allow the application to select a path of anonymous routers, they do not support mechanisms for choosing nodes based on metrics such as latency, bandwidth, network location, AS membership, trust, etc.

On the other hand, to address the rigidity of the current Internet infrastructure, several proposed routing infrastructures such as NIRA [22], i3-ROSE [16] and P2 [17] argue for providing end-host control over routing in the Internet. These proposals generally focus solely on performance-driven metrics such as latency, bandwidth, and resilience, and do not address anonymity.

We assert that the fundamental reason why the above proposals optimize for either *performance* or *anonymity*, but not *both*, is the inherent tension between the two: network path diversity leads to increased anonymity at the expense of network performance. To illustrate this tension in practice, we conducted measurements on PlanetLab [2] that demonstrate the relationship between the number of overlay nodes in a path and its round-trip-time (RTT). Figure 1 shows that an increase in anonymity (as indicated by the number of nodes in the path) corresponds to an increase in path RTT. We observe similar tradeoffs when comparing the number of distinct ASes per path against other performance metrics such as path throughput and jitter.

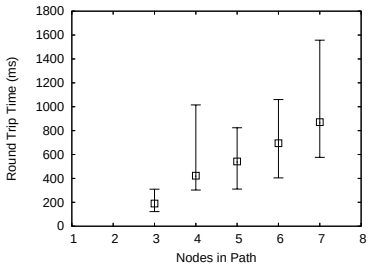

Figure 1: A tradeoff between anonymity and performance. Error bars indicate the first and third quartiles computed over 100 runs.

We argue that there exist several applications that desire not only customizable routing, but also the ability to fine-tune performance and anonymity to meet their specific needs. For example, an anonymous video conferencing system may require high bandwidth and low latency, but be willing to tolerate anonymity that provides only probable innocence[1] [20]. In such a case, the anonymous path from source to sink should include only intermediary nodes that provide high bandwidth and low latency. Anonymity goals such as traversing multiple autonomous systems (ASes) are secondary. In contrast, an anonymous email system may require very strong anonymity (i.e., beyond suspicion) while imposing no constraints on bandwidth or latency. Here, maximizing network diversity to reduce the feasibility of omniscient eavesdropping is paramount.

With this in mind, we introduce a grand challenge for anonymity: *The development of a network architecture that allows applications to tradeoff between anonymity and performance, enforce application-specific constraints, and efficiently customize routes, without sacrificing either existing network scalability or security.*

As a first step towards our challenge, we make the following contributions in this paper. First, we outline the design goals of our grand challenge: *scalability*, *robustness*, *measurable anonymity and performance*, and *security*. Second, we propose an initial design and implementation of the *Application-Aware Anonymity (A^3)* routing service that attempts to meet our design goals. Third, we propose path selection algorithms that can be tuned to meet different anonymity and performance requirements of applications. These path selection algorithms can in turn be used to support anonymous unicast, multicast, and anycast, and may be optimized for a variety of metrics such as latency, jitter, bandwidth, etc. We then conclude with a discussion of future research directions.

We focus the discussion of A^3 at the application layer as an overlay network, with an eye towards feasible deployment over the existing Internet infrastructure. Ultimately, we hope

[1] We adopt the terminology proposed by Reiter and Rubin [20] in our qualitative descriptions of the various strengths of anonymity.

that the lessons learned will provide insights and spark debate into more fundamental security/performance tradeoffs that are inherent in the Internet.

2 Related Work

Anonymous communication has been the subject of a substantial volume of research. Chaum proposed the use of networks in which intermediary routers called *mixes* shuffle encrypted messages from hop-to-hop until they are eventually delivered to their intended recipients [8]. In such mix-based techniques, layered encryption is used to prevent eavesdroppers from discerning the endpoints of the communication. Chaum's work is the basis for several deployed anonymity systems including onion routing [19], its popular implementation, Tor [11], and the Mixmaster anonymous emailer [1]. In their current implementations, these mix-based approaches do not allow client applications to choose routes that adhere to application-specific criteria. Instead, routes are chosen using pre-defined immutable heuristics. Interestingly, recent work has shown that the Tor anonymity network is vulnerable to an attack in which eavesdroppers exploit this homogeneous routing policy by falsely advertising low latency and high bandwidth links, drawing traffic towards mixes under its control [4].

In contrast to mix-based approaches, other techniques such as Crowds [20] and Hordes [21] rely on the network to provide anonymity. Here, packets are forwarded to anonymizing routers called *jondos*. Upon receiving an encrypted message, a jondo flips a weighted coin to determine whether the message should be forwarded to another jondo or to the final recipient. Clearly, such a system provides no policy enforcement mechanism. Since jondo selection occurs beyond the reach of the sender, including application-specified constraints with the initial request offers no guarantees that the requirements will be enforced.

In more recent work, there have been proposals on designing distributed anonymous networks [14, 23, 18]. Apart from scalability, these decentralized networks have the advantage that any participatory node may function as an anonymizing router, requiring eavesdroppers to observe more disparate network locations in order to break anonymity. These networks typically leverage peer-to-peer overlays such as distributed hash tables (DHT) [3] for scalable routing. We build upon these proposals by providing support for application-tunable routes.

3 Design Goals

Before proceeding to describe our own proposal, we sketch the design goals that we think are important for the practical realization of an Internet-scale application-aware anonymous service.

Scalability: The service should not depend on any centralized authority (e.g. Tor's directory), hence providing graceful scaling in terms of the number of participating nodes.

Robustness: Given that the Internet is dynamic and routes may be transient, anonymized path selection algorithms must be highly adaptable to changing network conditions.

Measurable anonymity and performance: In order to decide the "most desired" path, the initiator of communication must be able to measure the anonymity and performance provided by a proposed path of anonymizing routers. For example, anonymity can be quantified by the number of AS crossings and performance in terms of latency, loss rate, bandwidth, etc. The measurements themselves must preserve anonymity of the initiator and all participating nodes.

Security: Malicious nodes should not be able to conduct denial-of-service (DoS) attacks against the system, nor should they be able to break anonymity (for example, by conducting Sybil [12] attacks). At a minimum, the anonymity service should be "as secure" as the current Internet – i.e., it should not introduce any additional security holes, nor should it ease an eavesdropper's task of breaking anonymity.

In the next section, we introduce the A^3 routing service that attempts to meet the above goals.

4 Application-Aware Anonymity (A^3)

The A^3 routing service allows applications to establish anonymized routes that adhere to application-specified criteria. We begin our description of A^3 by describing the various path selection algorithms available to the application, and then describe how anonymous connections can be established using the selected paths.

4.1 Path Selection Algorithms

In all path selection algorithms, A^3 takes as input the initial sender node `n0`, a path constraint `N` defined in terms of the desired number of nodes, and a few application-defined functions described below. The output of each algorithm is a path `P`, which is used to forward packets anonymously from the sender to any destination. Our path selection algorithms utilize the following four functions:

- **`C` ← `f_randomNode()`**, where `C` is the next randomly chosen candidate node selected to construct a path.
- **`M` ← `f_metric(S,D)`** returns a metric `M`, measured between nodes `S` to `D`. The metric can be in terms of available bandwidth, latency, trust, etc.
- **`D` ← `f_agg(`**$\delta_0, \delta_1, ..., \delta_l$**`)`** is the path aggregation function (e.g., summation, minimum, etc.) that computes a combined metric `D` via a path that consists of links with measured metrics δ_0, δ_1, ..., δ_l. For example, sum is used to compute total end-to-end latency, while min is used to determine the bottleneck bandwidth. As an enhancement, `f_agg` can be composed from different standard arithmetic functions, and customized as a user-defined function.

- $\{0,1\} \leftarrow$ **`f_accept(R)`** is a Boolean function that returns true if the input aggregation value `R` is tolerable according to some application-provided constraint.

The functions `f_agg(`$\delta_0, \delta_1, ..., \delta_l$`)` and `f_accept(R)` are application-defined functions that are used to decide which nodes to select as intermediate hops in the path. In addition, the functions `f_randomNode()` and `f_metric(S,D)` require calls to external services which we describe in Section 4.2.

In all algorithms, n_0 denotes the sender's address, and N denotes the number of desired nodes in the path.

`Random(`n_0`,N)` **Algorithm**

The `Random` algorithm represents the Strawman Solution in which constituent path nodes are chosen uniformly at random. The algorithm continues until the path conforms to the application's requirements (i.e. `f_accept` returns true).

1: **repeat**
2: **for** $i = 1$ to $(N-1)$ **do**
3: $n_i \leftarrow f\_randomNode()$
4: $\delta_i \leftarrow f\_metric(n_{i-1}, n_i)$
5: **end for**
6: **until** $f\_accept(f\_agg(\delta_1, \delta_2, ..., \delta_i)) = \text{true}$
7: **return** path $\{n_0, n_1, ..., n_i\}$

Intuitively, `Random` offers a high degree of anonymity since nodes are chosen uniformly at random. However, anonymity comes at a cost – if `f_accept` is conservative in its acceptance of routes, then the algorithm will take a long time to terminate, resulting in high communication overhead (since, as we show below, `f_randomNode` and `f_metric` require anonymous network queries).

`Adaptive(`n_0`,N)` **Algorithm**

Unlike `Random`, `Adaptive` enforces constraints when picking each node along the path. The algorithm is as follows:

1: $i \leftarrow 0$
2: **while** $i < (N-1)$ **do**
3: $z \leftarrow f\_randomNode()$
4: **if** $f\_accept(f\_agg(\delta_1, \delta_2, ..., \delta_i, f\_metric(n_i, z)))$ **then**
5: $i \leftarrow i + 1$
6: $n_i \leftarrow z$
7: $\delta_i \leftarrow f\_metric(n_{i-1}, n_i)$
8: **end if**
9: **end while**
10: **return** path $\{n_0, n_1, ..., n_i\}$

`Adaptive` ensures that during path construction, only hops that do not violate the application constraint will be added. We consider two special cases of `Adaptive`. The first, `EnforcedAvg(`n_0`,N)`, restricts each hop in the path to consume up to an equal share of some total tolerable metric. That is, if P_{max} is the maximum tolerable measurement (e.g., latency) for the anonymous path, then `EnforcedAvg` ensures that no hop will have a cost greater than $P_{max}/(N-1)$. For example, if `f_agg` is the summation of non-negative costs, $f\_accept(R)$ returns true if current aggregate (R) divided by the current path length is less than or equal to $P_{max}/(N-1)$. Note that in general, we can use `EnforcedAvg` only if the aggregation function `f_agg` is monotonic.

Intuitively, `EnforcedAvg` sacrifices some anonymity in favor of lowering communication cost by limiting per-hop cost. If an eavesdropper knows P_{max} and N, she has more information to infer potential next-hops along the path. Furthermore, depending upon the metric of interest, `EnforcedAvg` may lead to a selection of nodes that reside in a particular area of the Internet. This is especially relevant when the metric is latency. In their analysis of location diversity in the Tor network [13], Feamster and Dingledine show that the proximity of nodes in the anonymous path may increase vulnerability to traceforward attacks, particularly when the attacker can observe all traffic within an autonomous system. `EnforcedAvg` is therefore most appropriate when only limited anonymity is necessary and fast path selection is required.

To reduce the likelihood of traceforward attacks, we propose a variant of the `Adaptive` algorithm called `PlusMinus(`n_0`,N,`ϵ`)`. This algorithm allows some leeway when selecting nodes via the *tolerance parameter* $\epsilon \in R^+$, as long as the expected per-hop cost between any two hops in that path remains $P_{max}/(N-1)$.

`PlusMinus` is implemented by modifying `EnforcedAvg` as follows. With probability $\frac{1}{2}$, the $f\_accept(R)$ function in `PlusMinus` returns true if the next potential link (i.e., n_i to z) contributes no more than $(1+\epsilon)(P_{max}/(N-1))$ to the maximum tolerable cost (P_{max}). To compensate for links with potentially greater-than average costs, with probability $\frac{1}{2}$, $f\_accept(R)$ returns true iff the candidate link has a cost less than or equal to $(1-\epsilon)(P_{max}/(N-1))$. As with `EnforcedAvg`, the technique is applicable when the function `f_agg` is monotonic.

While the above path selection algorithms are not exhaustive, they serve to illustrate practical approaches in which applications can tradeoff anonymity and performance. In addition, we believe that these algorithms are sufficiently general to support a wide array of application-specified constraints (e.g., bandwidth, trust, loss).

4.2 Node Selection and Measurements

We next describe the supporting functionalities of A^3 that enable these algorithms. The functions `f_randomNode()` and `f_metric(S,D)` require the implementation of a directory service to retrieve nodes randomly, and a measurement service to measure pair-wise metrics of any two candidate nodes.

To achieve our goal of decentralization, we avoid using any centralized directory services for selecting candidate nodes. Instead, we utilize DHTs as a distributed node location service where nodes are selected randomly from the identifier space. In addition, to conceal the identity of any candidate node, an anonymized version of the basic DHT lookup [5] primitive must be used.

Once candidate nodes are selected, the `f_metric(S,D)` function computes the metric between any two nodes `S` and `D`. The naïve approach in which the sender asks each candidate node `S` to perform measurements to another node `D` in-

troduces a significant communication overhead, requiring the establishment of expensive bidirectional onion routes during the path selection process. Remote measurement, on the other hand, requires only anonymous queries using the DHT.

An interesting direction that we are currently pursuing is the use of *Vivaldi* [10], a coordinate-based measurement system. In Vivaldi, each node is assigned a set of coordinates which it periodically adjusts such that the Cartesian distance between any two nodes corresponds to the distance of their chosen metric. The metric of interest is usually latency, but the same approach can be applied using any measurable quantity such as bandwidth, trust, loss rate, reachability, etc. The resulting coordinates are published into the DHT using the node's unique identifier, and retrieved anonymously[2] using the same anonymized lookup primitive described above. Each metric (e.g., bandwidth, latency, trust, etc.) requires its own logical coordinate system.

4.3 Path Establishment and Maintenance

Having presented path selection algorithms, we demonstrate how A^3 can utilize these algorithms to provide supports multiple communication primitives including anonymous unicast, multicast, and anycast. In all cases, connections are established in A^3 through the following steps:

1. The source (initiator) of communication chooses an anonymous path using the algorithms and supporting functionalities described in Section 4.1. In the case of unicast communication, the last node in the path is connected to the destination node. For multicast communication, the last node serves as the root of a multicast tree. Anycast communication can be similarly supported by using the last node as a point of indirection to multiple possible network endpoints.
2. Once a path has been selected, the initiator forms a bidirectional onion-route [19] through the path. All participating nodes are assumed to have published their public keys to the DHT. To generate onions, the initiator must issue anonymous queries to obtain these keys.
3. Data is anonymously transmitted by the two endpoints of the path via the onion route.

In the case of multicast communication, the initiator may publish into the DHT additional meta-information concerning the multicast stream along with the IP address of the root of the multicast tree. Interested parties may then query the DHT to obtain the information required to join to the multicast tree[3].

To ensure the robustness of constructed paths in a dynamic network, the initiator sends periodic *pings* (heartbeats) to the last node in the path via the onion route. If there is no response to pings, a new anonymous path is re-established by repeating steps 1-2. In addition, the initiator can monitor the performance of the path over time, and construct a new path if the measurement performance of a path falls below an application-specified threshold. For example, if the roundtrip time (the time between sending the ping and receiving the echo reply) exceeds a threshold, a new path must be selected. Similar measurements can be carried out for other performance metrics such as network jitter and throughput. For added resilience, A^3 allows the creation of redundant anonymized routes. When a path is disrupted, a secondary path can be relied upon while a new path is constructed.

[2]Anonymity is not the only method for hiding coordinate requests. *Confusion* techniques in which true requests are masked by artificially injected noise are also applicable [9].

[3]Although the establishment and maintenance of multicast trees are orthogonal to our work, we note that DHT-based multicast techniques (e.g. SplitStream [6]) may be well-suited for A^3 .

4.4 Preliminary Evaluation

We present a preliminary comparison of the path selection algorithms via simulation. The simulator takes as input transit-stub topologies generated using the GT-ITM topology generator [15]. The transit-stub topology consists of 585-590 stub ASes per transit domain, and 3 nodes per stub AS. We increase the number of nodes in the network by varying the number of transit domains from 5 to 10. The latency between transit nodes is set to 50 ms, the latency between a transit and a stub node is 10 ms, and the latency between any two nodes in the same stub is 2 ms. For simplicity, we assume that each node knows the round-trip time between itself and all other peers.

In each iteration of the simulator, a sender is selected at random, and a path selection algorithm is executed to construct a path. In all cases, we enforce an application constraint that all end-to-end path latency have to be less than 700 ms. Fifty iterations are conducted for three different network sizes, and the results are averaged over the 150 total iterations.

Figure 2 shows the performance of the `Random`, `EnforcedAvg`, and `PlusMinus` path selection strategies as measured by their anonymity (left) and network performance (right). `Random` is the top performer in terms of anonymity, but results in the worst end-to-end latency. In contrast, `EnforcedAvg` offers the best network performance and the least anonymity. `PlusMinus`, which uses a tolerance parameter ϵ of 0.5, constitutes a middle-ground between the two. Although we only evaluated path anonymity and latency, the graphs clearly illustrate that the effectiveness of each algorithm depends on the metric being optimized. Our results suggest that applications can utilize different path selection algorithms (or vary ϵ) to best meet their anonymity and performance requirements.

4.5 Revisiting our Design Goals

A^3 is our initial attempt at designing an anonymous routing service that is decentralized, scalable, and adaptable in environments with a high rate of node churn. We emphasize that while we propose using DHTs and the Vivaldi coordinate system as building blocks for anonymous node selection and measurements, other services that satisfy our design goals may also be applicable and worth exploring. For example, Vivaldi is one among several synthetic coordinate systems, and there exists several third-party measurement services. The choice of Vivaldi is particularly attractive because it is non-intrusive, lightweight, and it leverages the same DHT infrastructure that we already use to locate nodes. On the downside, the use of DHTs exposes a number of known security vulnerabilities [7].

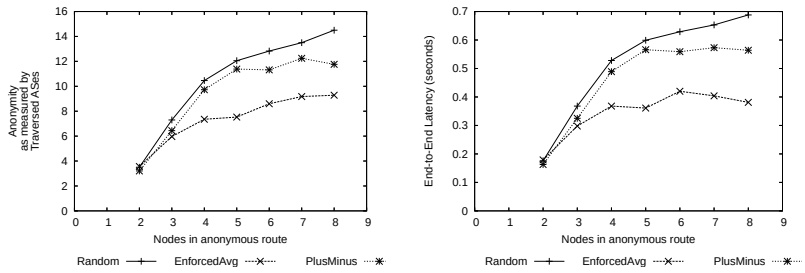

Figure 2: The performance of the `Random`, `EnforcedAvg`, and `PlusMinus` strategies. *Left:* the anonymity of the three algorithms as measured by the number of AS crossings in the path. *Right:* the resultant latency of the selected paths generated by the respective algorithms.

While there are well-known, orthogonal fixes to these vulnerabilities, exploring alternative technologies is an interesting avenue of future work.

5 Conclusions

In this paper, we present the case for application-aware anonymous routing, and lay out a grand challenge of developing network architectures that can better adapt to application-specific anonymity and performance constraints. Towards this challenge, we present the Application-Aware Anonymity A^3 routing service. We envision that A^3 will serve as a powerful and flexible anonymous communications layer that will spur the future development of anonymity services.

Our future work is proceeding along several fronts. First, we are implementing A^3, with the eventual goal of deploying on PlanetLab as an anonymous routing service. Second, we plan to explore declarative approaches [17] for applications to specify their anonymity and performance constraints. Third, we intend to formally analyze the anonymity properties of A^3. Finally, we hope that our experiences will lead to fundamental insights into security and performance tradeoffs inherent in the Internet.

Acknowledgments

This work was supported in part by the NSF under contracts CNS-0524047, CNS-0627579, and NeTS-0721845.

References

[1] Mixmaster. `http://mixmaster.sourceforge.net/`.

[2] PlanetLab. `http://www.planet-lab.org`.

[3] H. Balakrishnan, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Looking Up Data in P2P Systems. *Communications of the ACM, Vol. 46, No. 2*, Feb. 2003.

[4] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker. Low-resource routing attacks against anonymous systems. Technical Report CU-CS-1025-07, University of Colorado at Boulder, Feb 2007.

[5] N. Borisov. *Anonymous Routing in Structured Peer-to-Peer Overlays*. PhD thesis, University of California, Berkeley, 2005.

[6] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. Splitstream: high-bandwidth multicast in cooperative environments. *SIGOPS Oper. Syst. Rev.*, 37(5):298–313, 2003.

[7] M. Castro, P. Drushel, A. Ganesh, A. Rowstron, and D. Wallach. Secure Routing for Structured Peer-to-peer Overlay Networks. In *OSDI 2002*, 2002.

[8] D. L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–90, 1981.

[9] E. Cronin, M. Sherr, and M. Blaze. On the reliability of current generation network eavesdropping tools. In *Second Annual IFIP WG 11.9 International Conference on Digital Forensics*, Jan 2006.

[10] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: a decentralized network coordinate system. *SIGCOMM*, 34(4):15–26, 2004.

[11] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. In *Proc. of the 13th Usenix Security Symposium*, pages 303–320, 2004.

[12] J. R. Douceur. The Sybil Attack. In *First International Workshop on Peer-to-Peer Systems*, March 2002.

[13] N. Feamster and R. Dingledine. Location diversity in anonymity networks. In *WPES '04: Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society*, 2004.

[14] M. J. Freedman and R. Morris. Tarzan: A peer-to-peer anonymizing network layer. In *CCS*, Washington, D.C., November 2002.

[15] GT-ITM. Modelling topology of large networks. http://www.cc.gatech.edu/projects/gtitm/.

[16] K. Lakshminarayanan, I. Stoica, and S. Shenker. Routing as a Service. Technical Report UCB-CS-04-1327, UC Berkeley, 2004.

[17] B. T. Loo, J. M. Hellerstein, I. Stoica, and R. Ramakrishnan. Declarative Routing: Extensible Routing with Declarative Queries. In *ACM SIGCOMM*, 2005.

[18] A. Mislove, G. Oberoi, A. Post, C. Reis, P. Druschel, and D. S. Wallach. AP3: Cooperative, Decentralized Anonymous Communication. In *Proceedings of the 11th Workshop on ACM SIGOPS European Workshop: Beyond the PC*, 2004.

[19] M. G. Reed, P. F. Syverson, and D. M. Goldschlag. Anonymous connections and onion routing. *IEEE Journal of Selected Areas in Communications*, 16(4), May 1998.

[20] M. K. Reiter and A. D. Rubin. Crowds: anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.

[21] C. Shields and B. N. Levine. Hordes: a multicast based protocol for anonymity. *Journal of Computer Security*, 10(3):213–240, 2002.

[22] X. Yang. NIRA: A New Internet Routing Architecture. In *Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture*, 2003.

[23] L. Zhuang, F. Zhou, B. Zhao, and A. Rowstron. Cashmere: Resilient anonymous routing. In *Proc. of Networked Systems Design and Implementation (NSDI)*, 2005.