

High-Performance Anonymity Proxy Framework

Kefei Dan Zhou - zhouk@seas

Qiushi (Andrew) Mao - maoq@seas

Advisors: Micah Sherr, Boon Thau Loo

Senior Project Final Report

April 22, 2009

Contents

Abstract	2
Introduction	2
Related Work	3
Technical Approach	4
Background	4
Technical Challenges & Goals	5
Implementation	7
Future Work	8
Conclusion	10
References	11

Abstract

We introduce a high-performance, cross-platform framework for accessing the A³ anonymity network. A³(Application-Aware Anonymity) is a decentralized network for accessing web services anonymously. A³ offers significant improvements over existing anonymity systems like Tor by employing a network coordinate system and intelligent path selection algorithm. The current A³ system, however, can only be accessed programmatically and is limited to communication within the immediate network. The focus of our project is creating a portable proxy interface that allows any user application to operate seamlessly over the A³ network with no additional configuration. We designed the anonymity proxy framework utilizing virtual network devices and an anonymity protocol using TCP over UDP. The framework extends the capabilities of A³ and makes the anonymity network available to the masses.

Introduction

Anonymity is demanded by a wide range of users and for many purposes. Both Internet Service Providers (ISPs) and servers keep logs of user activities and transmitted private data that can potentially include names, addresses, and other sensitive information. Malicious hosts can also intercept this traffic and determine its origin. Individuals need to protect their privacy from malicious servers, identity thieves, and organizations that collect and sell network usage data. Military and law enforcement agencies use anonymous networks for gathering intelligence in the presence of eavesdroppers and communicating with field agents securely. Journalists and activists can send controversial information anonymously without revealing the source and rendering themselves a target.

The Internet and its primary data protocols such as TCP and UDP offer a scalable and reliable communication network but completely disregard privacy. While cryptography applications can secure transmitted data, analysis of network traffic still easily reveals the connection endpoints. Efficient sender and receiver anonymity is inherently a difficult problem because network routing protocols require packets to have destination addresses on both the subnet and hardware level. A majority of the applications exist today offer sender or receiver anonymity by routing encapsulated packets through one or more intermediate nodes in an attempt to hide the identities. Modern low-latency anonymous networks offer sufficient consistency for browsing the Internet but not enough for streaming multimedia and other high bandwidth or low latency applications. A³ is designed specifically to provide a customizable anonymous service that meets these demands.

We must first explore some of the theory behind the Application-Aware Anonymity (A³) network Sherr et al. (2007), with research and development led by Micah Sherr. The decentralized system of A³ combines three main components: a distributed directory service, a coordinate embedding system, and a route selection engine. The directory service Balakrishnan et al. (2003) assigns a unique global identifier to each node in the network. For efficient lookups of network nodes, A³ leverages Bamboo bam (2008), a Distributed Hash Table. (Stoica et al. (2003) is another alternative.) The network coordinate system allows

individual nodes to estimate performance measurements over the entire network, providing the necessary data for randomized route selection. The current implementation of A³ uses Vivaldi Dabek et al. (2004), a light-weight distributed coordinate system that maps nodes into n-dimensional Euclidean metric space and uses the Euclidean distance between nodes to represent measurements such as latency. Vivaldi continually recalculates node coordinates using a spring-system algorithm, and convergence to accurate performance measurements occurs minutes after a node joins the network. Each node maintains the coordinates of a set of random peers in its local cache. To construct an anonymous route, we create random routes using the nodes in the local cache and rank the routes based on their performance characteristics calculated from the coordinate system. Finally we select the route using a randomized algorithm that gives higher weights to more desired paths.

Next, we continue with a discussion of the advantages of A³ with respect to other anonymity services, followed by our implementation of an application interface to A³.

Related Work

One of the simplest anonymizing services is a proxy server that replaces the source node's packet headers before forwarding it to its destination. Anonymizer ano (2008) is a single hop proxy that provides basic sender anonymity. This design, while efficient in terms of bandwidth and latency, is vulnerable if the adversary can eavesdrop and analyze the incoming and outgoing traffic. In addition, the sender must trust the proxy and is susceptible to single point of failure. Most modern anonymous services are based on Digital Mixes (Mix-Net) Chaum (1981), which provides a much stronger anonymity by wrapping the message in layers of public-key (RSA) encryption and routing it through multiple relay nodes. This relay-based design provides the foundation for most of the anonymous services existing today.

One class of relay-based designs sacrifices latency to boost anonymity. These high-latency networks, including Babel Gulcu & Tsudik (1996) and Mixmaster, introduces additional random latencies, making the network resilient against strong global adversaries and end-to-end timing correlation attacks. However, the potentially large latency introduced makes such networks undesirable for common network tasks involving bidirectional communication. Low latency systems include P2P designs such as Tarzan Freedman & Morris (2002), MorphMix, and circuit-based systems like Freedom and Tor Dingledine et al. (2004). These networks provide enough interactivity for general tasks such as web browsing. We will focus on Tor, one of the most widely used anonymity networks.

To access the Tor network, the application contacts a directory server for a list of Tor nodes and randomly constructs a path using a subset of these nodes. The Tor application then creates an encrypted virtual circuit by extending the path one hop at a time until it reaches the destination. At each extension, the next node exchanges a set of unique private keys with the source node through the established route. Each node in the path only knows the predecessor and successor node, but the source node will share a unique set of encryption keys with each node in the path. To send a message, we repeatedly encrypt the data with the set of encryption keys and route the packet to the first node. Each node removes the

outer most layer of the encryption and forwards the packet to the next node. This technique, called onion routing Reed et al. (1998), prevents the intermediary nodes from learning the source, destination and the contents of the message. Only the exit node, the last node in the path, will be able to see data as plain text.

We will compare A³ specifically with Tor. A³ inherits the major strengths of Tor including a flexible and simple design, strong and reliable anonymity, and resilience to most adversarial attacks. In addition, A³ addresses many of the weakness of Tor. Tor's low-latency design allows sufficient interactivity for browsing the Internet, but can not keep up with modern applications such as streaming multimedia and VoIP that demand certain levels of latency and bandwidth. A³ uses a distributed coordinate system with embedded performance metrics and select paths that adhere to the performance requirements set by the application. The fully distributed nature of A³ avoids some of the pitfalls with Tor's central servers and further strengthens the network against active and passive attacks.

At the application level, Tor still requires a client application to have SOCKS proxy support, as it is accessed through a proxy server interface on the local host. This is a drawback because applications need to be built with proxy protocol support or need to be recompiled with overridden functions for default socket operations. We aim to overcome these problems with a virtual network interface, which is supported by drivers for almost all operating systems and enables applications to communicate the same way as they would with hardware network interfaces.

Next, we introduce the cross platform application for accessing the A³ network, as well as other results of our project. To open the A³ network to the masses, we designed an anonymity framework with the goal of creating a flexible and portable application that allows any existing network application to connect over the A³ network without any special configuration. The anonymity framework will employ TUN/TAP tun (2008) virtual network devices to allow legacy applications transparent access of the A³ network. We will also discuss in detail the technical approaches to implementation.

Technical Approach

Background

Our project continues and collaborates with the work of Micah Sherr and Boon Thau Loo Sherr et al. (2008a), with the end goals of creating a public anonymity network and releasing an A³ compatible software interface on both Linux, Windows, and other platforms.

A³ depends on Bamboo (an open source distributed hash table) bam (2008) and the Vivaldi network coordinate system Dabek et al. (2004) to implement its path selection algorithm. For more information on peer-to-peer data lookups, see Balakrishnan et al. (2003). In our opinion, A³ has successfully remedied many of the drawbacks of the Tor onion routing system.

Currently, A³ is fully implemented to the specifications in the original paper, and supports anonymous transmission of datagrams on onion routes between in-network nodes. Since

the open-source implementation of Bamboo (including an implementation of Vivaldi) is written in Java, much of A³ has been built around this existing framework. However, to be useful in practice, A³ needed to support forwarding of TCP and UDP connections from other applications and connect them to destinations that are external to the A³ network. Furthermore, it is very useful to have anonymous services reachable within the A³ network. In the next section, we explain our achievements toward these goals as well as potential future work.

Technical Challenges & Goals

The focus of our project was to create a portable interface that allows any application to seamlessly access the A³ network and allow connections to reach destinations outside of the network. Our project specification calls for a lightweight framework that utilizes standard protocols in the kernel and avoids implementing unnecessary features. The following sections aim to divide this design into parts and simplify the discussion of implementation, and finally describe the implementation itself.

Virtual Network Devices

In considering a cross-platform interface to A³, we initially considered the use of a SOCKS5 proxy server running as part of the A³ Java application. This would allow any client supporting connections through the SOCKS protocol to establish anonymous connections to the A³ network. However, there are several drawbacks to this design, most noticeably that we need an additional translation layer to support clients using basic routing. Also, implementing anonymous servers (currently supported by Tor) would require an excessive number of helper applications, or a completely different implementation.

For this reason, we chose instead to use the TUN/TAP tun (2008) virtual network device drivers available for multiple platforms including Linux, Win32, OS X, and Solaris. These drivers enable a userspace program to directly read and write Layer-3 packets to and from a virtual network device, effectively inserting traffic into the standard network stack. The main advantage of this implementation is that the kernel still does the work of translating from a stream to packets and vice versa. Although the drivers are different for each platform, an implementation using virtual network devices will allow any application to use A³ with its standard network protocols. Many VPN applications already use this method to create virtual devices, which compress and encrypt the data in userspace before sending it over the actual hardware interface. In a similar fashion, processes can connect or listen over the virtual device and A³ will carry out the necessary computations to establish anonymous routes for each connection. Our implementation will require TUN devices on every A³ node for forwarding external traffic as well as communicating with local processes.

The advantage of supporting transmission and reception of standard IP frames is that basic bidirectional Layer-3 traffic to clients and servers can be treated the same way, allowing access to all existing applications. (We address the additional requirements for anonymous servers in a later section.) This approach, while very flexible, presented challenges of handling

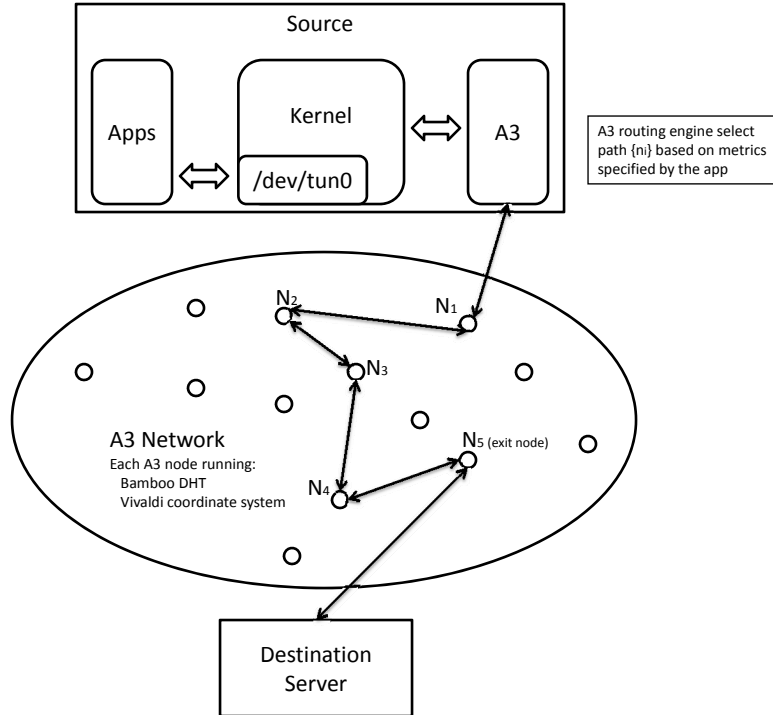


Figure 1: How an A³ client connects to a remote server

the multiplexed connections in A³ originating from the virtual device and keeping each track of each separate route. Also, our proof of concept only supported Linux; in the future we will have to develop different bindings from Java to kernel devices for each operating system. In Linux, the TUN/TAP drivers are part of the kernel, whereas Windows drivers are available only as part of the OpenVPN source package.

Next, we discuss the additional challenges of the application layer.

Anonymous Client Connections

Another basic problem deserving attention is the establishment of anonymous routes originating from within the A³ network to external servers. Since the destination is not part of the A³ network, we need to modify the anonymous routing algorithm for endogenous nodes. For example, Tor selects a completely random exit node (as well as the rest of the path), which result in very poor performance if it is on the other side of the world. To avoid the performance hit from a badly selected exit node, we use geographic coordinate to locate an exit node for the anonymous route that is close to the destination, and create the anonymous route to that exit node.

Another, much more important result is supporting standard Layer 3 protocols over the straightforward anonymized UDP packet transmissions currently implemented in A³. While the TUN driver simplifies this process, there are other possible problems when TCP packets are encapsulated and sent over A³, which will need to determine the correct routes over the

anonymity network. All of these computations take place within the application as it processes data to and from the virtual network device. Although there is much established research on this topic, it becomes more tricky to ensure that the protocol works over segmented links, such as the additional connection from an exit node to an endogenous server. After our implementation, we have gained significantly more knowledge about the TCP protocol and its implementation in the kernel.

Implementation

For the purposes of this discussion, we will refer to the entry node and exit node endpoints of an established connection on two different hosts. It is important to keep in mind that every A^3 node supports originating connections as an entry node as well as bridging connections to external servers as an exit node, and that there can be many different connections each on different established onion routes.

We must preserve the anonymity that is already inherent in A^3 when extending its functionality to virtual network devices. Therefore, the IP addresses of virtual network devices cannot reveal any additional information about a host. For this reason, we have decided to bind unroutable IP addresses to virtual network adapters and identify different connections by a randomly generated identification tag internal to A^3 . To allow the kernel to still operate transparently on TCP connections, we make use of extensive packet rewriting - both port and network address translation.

When a TCP socket connect is performed, the kernel sends out a corresponding packet with the SYN flag set over the appropriate network interface. (The exact interface is determined by the kernel's routing table.) By looking for packets with this flag set, we can detect new TCP connections to remote hosts. At this point, the entry node can assign a new 160-bit randomly generated ID to the connection and establish an onion route to an appropriately chosen exit node. Connections from the local host can be identified uniquely by source port, destination host, and destination port, and tied to this 160-bit ID, which will be as a header to internally tag all future packets for this connection, and makes it mathematically very unlikely that any two connections on the network will have the same ID. This ID is necessary because connections on the network cannot be distinguished simply by examining packets' addresses, which are all identical to preserve anonymity.

When an exit node receives a request to forward a connection, it establishes a separate TCP connection to the external server, and also creates a server socket that listens on a random local port. Packets that are delivered to the virtual network interface of the exit node have their IP address, port, and checksum rewritten to reflect that of the exit node, and vice-versa in the reverse direction. This network and port address translation allows an exit node to handle a number of connections only limited by its available ports, and is similar to the mechanism used by a NAT router.

All TCP packets sent by the entry node are encapsulated in datagrams and sent over the onion route. Since the TCP protocol takes care of reliable, in-order delivery, it is fine for datagrams to be lost. In addition, since most modern networks have very high reliability, the establishment of a single TCP connection tunneled over datagrams has much lower overhead

than bridged end-to-end TCP connections between nodes, used in anonymity networks such as Tor. The use of TCP over UDP is commonly used in situations such as VPN, but it has been rarely applied to anonymity networks before. We believe that it has the potential to greatly increase the performance of low latency networks.

In summary, the external server will only see a connection from the exit node, and the onion route is used to transmit all information across the A³ network. The exit node also has no additional knowledge of the entry node's identity by examining the packet headers, since all internal IP addresses are the same. A client application does not need to be aware of A³ to take advantage of this functionality, since the SYN packets for TCP connections are automatically detected. Finally, this can be easily extended to protocols such as UDP, which in spite of being stateless, can still be identified by destination and port numbers.

Design Advantages

- Applications don't need explicit proxy (i.e. SOCKS) support
- Portable design - most platforms support virtual network devices
- Compact; utilizes kernel implementation of TCP/UDP protocol
- Forward and backward compatible with core protocols
- Preserves anonymity of A³, with packet rewriting so that all virtual network devices can use arbitrary IP addresses
- Secure - no superuser access or use of kernel network stack needed

Our implementation in Python/Java proved to have minimal-overhead and clocked a download speed of over 10Mbps. As it was only a proof of concept, we are confident that further optimization will allow it to fully take advantage of the available resources of A³ network nodes.

Future Work

In this section, we discuss how these results can be extended to support anonymous servers, as well as some drawbacks of the design that can be improved.

Anonymous Servers

In addition to the reliability protocols necessary for normal TCP connections as mentioned above, the introduction of anonymous servers brings a whole slew of new problems to tackle. Ideally, public clients should be able to access anonymous servers by symbolic identification only, with no knowledge of their whereabouts. In the Tor system, this is performed by the client and server establishing anonymous routes to a common destination. However, this limits the usefulness of anonymous servers to public clients, and to leverage the existing A³ network structure, we plan to use a modification of the previously proposed MCAnonymous ? anonymous server idea.

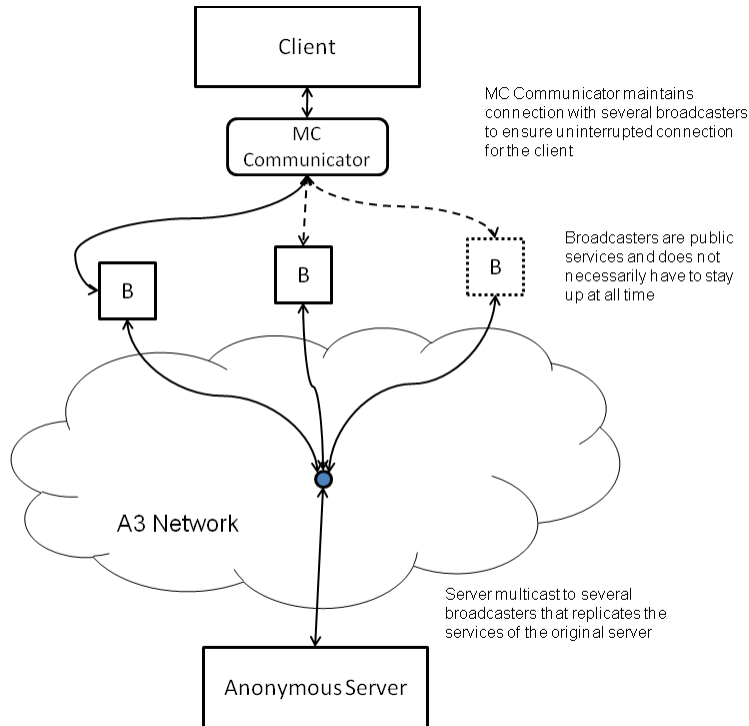


Figure 2: How an A³ client connects to a remote server

For public (external) clients to connect to anonymous servers, we use the idea of broadcasting nodes that replicate services through anonymous connections from the server. The server application need only to listen on the virtual network device, and A³ will establish anonymous routes to the broadcasting nodes, where a helper application will pass through packets to and from the anonymous route. It is important to note that while these broadcaster nodes must be known by the server or A³ beforehand, and that public clients can find their addresses by a DHT lookup on a unique service identifier.

The advantage of having many identical broadcasters or repeater nodes is that they can be replaced in real time with different nodes so that the service is always available. This way, the risk of an anonymous service being disabled by attacks on individual broadcaster nodes is very low, as is the risk to any individual broadcaster itself.

While any public client can connect to an individual broadcaster and use the anonymous service for an indeterminately short period of time, it would be much more useful to provide a transparent interface that presented a normal connection by a multiplexed reliability protocol to each of the broadcaster nodes. This application, called 'MCCommunicator' in the diagram above would keep track of the "alive" broadcasters in real time and ensure that any packets transmitted to and from the anonymous server would reach their destination by way of at least one of the broadcaster nodes. This is a challenging problem that is separate from the A³ system itself, but one that can be addressed by extending our basic implementation.

Theoretical Problems

The previous theoretical problems (exclusive of implementation) that we sought to address have been exceeded by the challenges of implementing a practical A³ system; however, we summarize them here.

Currently, a drawback of A³ is the high embedding error of bandwidth measurements in the network coordinate systems. Although current testing tools such as Pathrate Dovrolis et al. (2004), Spruce Strauss et al. (2003), and Pathchirp Riberio et al. (2003) provide relatively accurate measurements of bandwidth availability and capacity, network coordinate systems (using metric spaces) fail to embed bandwidth measurements well because of their inverse nature leading to frequent triangle inequality violations. The difficulty of effective bandwidth estimates, combined with slower measurement tools challenging the real-time requirements for accuracy in the network coordinate system, means that an effective solution for calculating path bandwidth has not yet been found.

Once A³ has a practical implementation, we will need to determine latency and bandwidth requirements of various applications in order to leverage its advantages. Since legacy applications won't be able to specify the path requirements through a connection to the virtual network device, the A³ library needs to choose some reasonable requirements for the path. One useful idea is to profile applications and determine reasonable latency and bandwidth requirements transparently, perhaps with the use of aggregate data collected from users on the Internet. Since lower constraints on path requirements provides potentially higher anonymity, it is important in practice to make accurate estimates here.

Finally, there is much more potential research in the effects of malicious nodes on the network. Blaze and Sherr Sherr et al. (2008a) demonstrated that A³ worked effectively with as many as 40% of network nodes being malicious, but as the scale of implementation increases to the size of the Internet as a whole, new issues will appear Sherr et al. (2008b). Currently our working knowledge is insufficient to tackle these kinds of problems, but by designing our own system with security and anonymity in mind, we have learned much about solving this type of problem and crypto-analysis.

Conclusion

During the project, we learned the inner workings of the IP Suite protocols, specifically IP, TCP and UDP. We developed software systems to intercept and and decode protocol packets from the network layer to the application layer. In addition, we designed a network layer messaging protocol specialized for establishing session and transporting data between the nodes.

Following the agile development methodology, we went through several iterations of the design process. We approached the problem from several angles and considered a wide range of potential designs (and we eliminated many because of limitations in technologies or overcomplicated implementation). In the end we combined the existing technologies of NAT/PAT and distributed systems to create a prototype that was efficient and platform

independent.

We went through several initial designs and were forced to make several complete revisions before finally discovering an implementation that was workable, efficient, and platform independent. With almost no prior work to draw upon, we finally combined the existing technologies of NAT/PAT and distributed systems after much brainstorming.

The project was a very rewarding experience in conducting our own research and discovering solutions to unsolved problems. It was also an excellent learning experience in network architecture and distributed systems design.

References

(2008). Anonymizer. [Http://www.anonymizer.com](http://www.anonymizer.com).

(2008). The bamboo distributed hash table. [Http://bamboo-dht.org/](http://bamboo-dht.org/).

(2008). Tun/tap.

Balakrishnan, H., Kaashoek, M. F., Karger, D., Morris, R., & Stoica, I. (2003). Looking up data in p2p systems. *Commun. ACM*, 46(2), 43–48.

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

This paper introduces one of the original anonymous service. The author, David Chaum, is an established expert in cryptology and an inventor of many widely used cryptographic protocols. Digital mixes (or Mix Net) communicate through a chain of proxy servers. Each message is hidden within layers of public-key encryption is routed through the path. The concept behind MixNet is critical to more modern anonymity networks such as Tor and our A3. We will conduct our research and development with a focus of improving upon the weaknesses of Mix Net and Tor.

Dabek, F., Cox, R., Kaashoek, F., & Morris, R. (2004). Vivaldi: a decentralized network coordinate system. *SIGCOMM Comput. Commun. Rev.*, 34(4), 15–26.

Vivaldi is a decentralized network coordinate system that uses a spring system algorithm to update latency measurements of a network in real time. Convergence is relatively fast and very accurate for latency measurements. Since it relies on a distributed hash table instead of any centralized server to find peers, it is particularly useful as A3's measurement embedding system.

Dingledine, R., Mathewson, N., & Syverson, P. (2004). Tor: the second-generation onion router. In *SSYM'04: Proceedings of the 13th conference on USENIX Security Symposium*, (pp. 21–21). Berkeley, CA, USA: USENIX Association.

This paper introduces the second-generation Tor and was presented at the 13th USENIX Security Symposium. Tor is originally sponsored by the US Naval Research Laboratory. The original design of Tor is created by authors with academic degrees and expertise in network systems. Today research and development of Tor is sponsored by The Tor Project. Currently Tor is one of the most widely used anonymous network. The design behind Tor is critical to A3 as A3 uses Tor as a blueprint and inherits many of the strength of Tor. In addition we will analyze the weaknesses of Tor and attempt to offer realistic solutions.

- Dovrolis, C., Ramanathan, P., & Moore, D. (2004). Packet-dispersion techniques and a capacity-estimation methodology. *IEEE/ACM Trans. Netw.*, 12(6), 963–977.
- Freedman, M. J., & Morris, R. (2002). Tarzan: a peer-to-peer anonymizing network layer. In *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, (pp. 193–206). New York, NY, USA: ACM.
- Gulcu, C., & Tsudik, G. (1996). Mixing email with babel. In *SNDSS '96: Proceedings of the 1996 Symposium on Network and Distributed System Security (SNDSS '96)*, (p. 2). Washington, DC, USA: IEEE Computer Society.
- Reed, M. G., Syverson, P. F., & Goldschlag, D. M. (1998). Anonymous connections and onion routing. *IEEE Journal of Selected Areas in Communications*, 16(4).
- Riberio, V., Riedi, R., Baraniuk, R., Navratil, J., & Cot, L. (2003). Pathchirp: efficient available bandwidth estimation for network paths. In *Passive and Active Measurement Workshop*.
- Sherr, M., Loo, B. T., & Blaze, M. (2007). Towards application-aware anonymous routing. In *HOTSEC'07: Proceedings of the 2nd USENIX workshop on Hot topics in security*, (pp. 1–5). Berkeley, CA, USA: USENIX Association.

This article is presented in the 2nd USENIX workshop by the USENIX Association. The authors discuss the existing network systems for anonymity and the lack of emphasis on performance metrics such as latency. Most of the existing anonymity networks trade latency and bandwidth for anonymity, and as a result can not handle many of today's multimedia applications that demand certain level of latency or bandwidth. The paper presents a novel solution aimed at delivering better performance through intelligent path selection. This article is particularly relevant as our project is a joint collaboration with the A3 project.

- Sherr, M., Loo, B. T., & Blaze, M. (2008a). A3 : Application-aware anonymity for the masses.

This paper expands upon the idea in a previously published paper, Towards Application-Aware Anonymous Routing, by the same authors. It presents a concrete design for an anonymity network that tailors latency and bandwidth

to the demands specified by individual applications. The ideas in this paper will serve as our core resources as our project aims to help complete the development of the A3 network and to deliver an interface that allows clients to seamlessly connect to this anonymous network.

Sherr, M., Loo, B. T., & Blaze, M. (2008b). Veracity: A fully decentralized service for securing network coordinate systems. In *7th International Workshop on Peer-to-Peer Systems (IPTPS 2008)*.

This conference paper is presented at the 7th International Workshop on Peer-to-Peer Systems by the same team that developed A3. All members of the team have academic degrees and specialize in fields in or related to network systems. Veracity is a decentralized coordinate system critical to the research of A3. We embed performance metrics such as latency and bandwidth into veracity to make A3's intelligent path selections possible.

Stoica, I., Morris, R., Liben-Nowell, D., Karger, D. R., Kaashoek, M. F., Dabek, F., & Balakrishnan, H. (2003). Chord: a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Trans. Netw.*, 11(1), 17–32.

This paper presents one of the original designs for a distributed hash table. Chord is researched and developed by scholars at MIT, funded by DARPA and the Space and Naval Warfare Systems Center (SPAWAR). It is a scalable and robust distributed system that provides efficient storage and lookups in a peer-to-peer network. A3 utilizes a variation of Chord (Bamboo) as the foundation for its decentralized directory service. This brings A3 to a fully-distributed state and makes the network more robust and resilient against active and passive attacks.

Strauss, J., Katabi, D., & Kaashoek, F. (2003). A measurement study of available bandwidth estimation tools. In *IMC '03: Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, (pp. 39–44). New York, NY, USA: ACM.