SocialDNS: A Decentralized Naming Service for Collaborative P2P VPNs

Pierre St Juste, David Wolinsky, Kyungyong Lee, P. Oscar Boykin, Renato J. Figueiredo
Advanced Computing and Information Systems Lab, University of Florida, Gainesville, FL 32611
Email: {pstjuste,davidiw,klee,boykin,renato}@acis.ufl.edu

Abstract

The ability to define domain names for resources in a collaborative virtual organization is usually reserved to network administrators through centralized domain servers. We propose SocialDNS, a decentralized, naming service that gives individual collaborators the power to choose the domain names for their resources. Our approach is based on similar concepts of decentralized naming solutions available in local area networks. We enable short-names for resources by limiting the scope for uniqueness. We also employ a rank-based mechanism for dealing with name conflicts. We evaluate our design through graph level analysis to anticipate scope, bandwidth costs and latency. We also conducted experiments involving Amazon EC2 and PlanetLab to analyze the latencies in a real world environment.

Index Terms

Social Networks, Rank, Naming

1. Introduction

The ability to define domain names for resources in collaborative virtual organizations is usually reserved for network administrators through centralized domain servers. This restriction precludes effective collaboration especially in the P2P VPN enabled virtual organizations with no central authority. Without a domain naming service, collaborators would have to resort to hard-to-remember IP addresses to refer to the location of services. This is further exacerbated in dynamic IP environments such as P2P VPNs and private networks where IP addresses can change frequently; hence making it extremely difficult to keep track of the location of collaborative services.

We propose SocialDNS, a decentralized, naming service for collaborative P2P VPNs. P2P VPNs provide collaborators IP access to each in a decentralized fashion; however, there does not exist a decentralized solution which allows collaborators the freedom to choose domain names for the services that they host. Decentralized solutions such as multicastDNS [1] and WINS [2] exist for private networks and LAN environments; however, these solutions cannot be applied to the P2P VPN environment unmodified. We address this need in SocialDNS by providing an alternative comparable to decentralized solutions such as multicastDNS but better suited for P2P VPNs. SocialDNS uses a simple broadcast mechanism for communication which can be deployed on top of existing P2P VPN solutions such as Hamachi [3], or SocialVPN [4]. Moreover, SocialDNS does not rely on any specific peer-to-peer overlay topology.

With SocialDNS, P2P VPN collaborators are able to select short-names among themselves for their resources through social scope uniqueness instead of the global uniqueness enforced by the normal DNS system. Name conflicts can arise in the SocialDNS system if two peers decide to choose the same domain name for a resource. In such cases, we use a simple rank-based method to select the mapping with the highest popularity in the social circle.

We provide a social graph analysis of our design where we estimate expected bandwidth costs, latency, and outcomes of name conflicts. Our analysis is based on the assumption that P2P VPNs form a social graph with small world characteristics since each VPN link represents a social relationship. Although not all P2P VPNs possess this property, we are only focused on P2P VPNs that only created VPN links based on social relationships. Based on this assumption, we used a 100,000 social networking dataset from Orkut to validate our design choices. Our analysis shows that a typical SocialVPN query can consume about 42 Kbytes of bandwidth. We also observed that by setting a timeout of 100 ms per query, we are able to get responses from 79% of our peers on average. We

also conducted experiments involving Amazon EC2 and PlanetLab to analyze the latencies in a real world environment.

The rest of the paper is outlined as follows. Section 2 provides motivating uses cases for SocialDNS. Section 3 explains some background on P2P VPNs and decentralized DNS solutions for private networks and LANs. In section 4, we present the design of SocialDNS, followed by its analysis in section 5. Section 6 describes a prototype implementation and some experiments conducted with the prototype. Section 7 covers related works in both decentralized naming and peer-to-peer reputation systems. We conclude in Section 8.

2. Motivation

The main motivation for SocialDNS is to provide end-users with the freedom to set their own domain names in P2P VPN environments. Domain names serve an important role in the user-friendliness of the Internet and are used in many phases of TCP/IP connections. Here we provide a few use cases that demonstrate the importance of a decentralized, user-controlled, domain naming service for private networks.

Naming for Self-Hosted Services. P2P VPNs make it possible for end users to host services on their personal resources and provide network level access to peers of interest. For example, an end-user, Alice, can host a blog from her laptop that only her P2P VPN friends can access, or share her desktop through a VNC session to do a PowerPoint presentation. An important requirement for hosting services is user-friendly domain names to these services so that Alice's colleagues can connect to her blog by typing *aliceblog.sdns* or view her presentation by connecting to her laptop by typing *alicepc.sdns* in their remote desktop clients. Domain names also provide location transparency in dynamic IP environments such as private networks, and P2P VPNs; thus end-users are not required to rediscover the dynamic IP address to a service every time there is a change in the host's IP address. In the previous example, without a domain naming service, Alice's friends would have to discover Alice's IP address every time they want to access her blog or view her presentation. Hence, IP connectivity is not the only requirement for enabling users the freedom to host their own content, a domain naming system is also necessary so that end users can select the names used to refer to these services.

Distributed Virtual Environments. A distributed virtual environment, such as OpenSim [5], comprises of a virtual world where different regions are hosted on separate host machines that may be geographically dispersed. A P2P VPN can facilitate the deployment of such an environment by providing a virtual private network of collaborators where each collaborator adds to the virtual world by hosting their own region. Through this private network, collaborators can securely share information and data through the virtual environment without fear of information leakage to unauthorized third parties. Configuring this virtual world requires the use of configuration files containing pointers to the host machine of each region that make up the virtual world. Using IP addresses in the configuration file is not ideal because private networks typically use dynamic IP; hence, changes in IP addresses would require constant updates to configuration files. However, by extending the P2P VPN model to provide a domain naming service, the changes in IP address would be transparent to the application thus eliminating the need to constantly update configuration files.

3. Background

In this section, we provide a brief description on the use of P2P VPNs for collaborative environments, the social aspects of P2P VPNs, and current decentralized solutions for private networks and LANs.

P2P VPNs for Collaboration. Virtual private networks (VPNs) allow remote users secure access to private organizational networks over the Internet. The basic concept is to tunnel IP traffic through encrypted TCP/IP links and therefore provide secure network access to private resources such as a company database, an intranet, a printer, or a computing cluster. In typical VPNs, enabling such a service requires dedicated resources such as a VPN gateway servers to serve as the traffic broker for these external access clients, as well as complex administration to guarantee privacy, access control, and quality of service. P2P VPNs, on the other hand, allows geographically dispersed collaborators to form their own VPNs by tunneling IP packets directly to each other without reliance on centralized relay gateways. P2P VPNs also provide firewall and NAT traversal thus making it feasible for users to communicate without access to public IP addresses. Various P2P VPN solutions currently exist such as Hamachi [3], SocialVPN [4], Wippien [6], just to name a few. P2P VPNs effectively facilitate collaboration by making it possible for individual users to privately share resources over VPN links without the typical infrastructural and administrative overheads of traditional VPNs.

The P2P VPN Social Graph. Various P2P VPN solutions such as SocialVPN [4], or Wippien [6], have

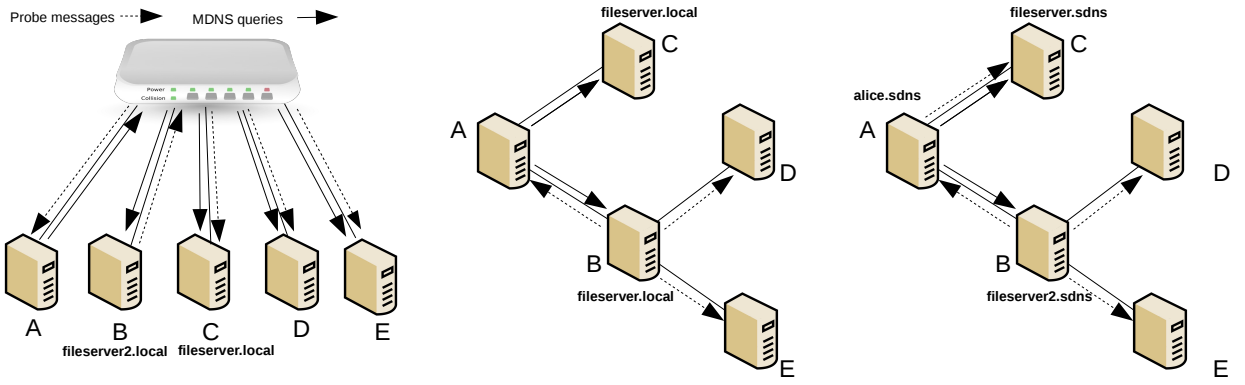


Figure 1. **On left: MulticastDNS in LAN environment.** All-to-all connectivity among nodes allows multicast DNS to successfully detect duplicate names, hence host B chooses domain name *fileserver2.local* because host C's mapping of *fileserver.local* is discovered in through a probing phase. **In middle: MulticastDNS in P2P VPN environment.** Lack of all-to-all connectivity causes host B to be unaware of host C's mapping of *fileserver.local* and thus claims the same mapping. This creates a conflict for host A who now has two peer in her network with the domain name of *fileserver.local*. **On right: SocialDNS in P2P VPN environment.** SocialDNS uses a two-hop broadcast to search for duplicate names in the social circle. With the two-hop broadcast, host B discovers host C's mapping for *fileserver.local* and chooses *fileserver2.local* instead. If host B still decides to pick the same domain name as host C, then the conflict resolution mechanism picks the most popular name in the social circle.

an interesting characteristic not found in traditional VPNs in that the encrypted tunnel links of the P2P VPN network represent the edges of a social graph with small world characteristics. As shown in Figure 1, even though both collaborators B and C are part of collaborator A's VPN, it does not imply that collaborators B and C have a VPN link to each other. This is analogous to a social network where Alice can be friends with both Bob and Carol, but it does not mean that Bob and Carol are friends. For scalability and security, it is important to only have VPN connections with trusted peers; there is no point of having links with peers of no common interest. This is a departure from the common concept of private networks such as local area networks (LANs) and traditional VPNs where there is the expectancy of all-to-all connectivity among nodes in the same network.

Decentralized naming in private networks. Decentralized naming services are extremely useful and common in private networks (e.g. LANs) because they provide a zero-configuration solution to mapping user-friendly names to resources. For example, in a typical home network, a decentralized naming service makes it possible to access a file using the following url smb:\\mom-pc\SharedDocs\familypic.jpg. The two commonly available decentralized naming solutions for private networks are the Windows Internet Name Service (WINS) [2] by Microsoft and Apple's multicastDNS [1] system called Bonjour. One approach would be to run one of these naming solutions on the P2P VPN unmodified. However as shown in Figure 1, these approaches were designed with the assumption of all-to-all connectivity amongst all node within the same network through a common networking backbone (e.g. routers and switches). The lack of such all-to-all connectivity in the P2P VPN setting renders these solutions nonfunctional. SocialDNS aims to address these limitations for the P2P VPN environment.

4. SocialDNS Design

The main design goals of SocialDNS are short-names through social scope, decentralization, simple user management, and name conflict resolution through social popularity. The SocialDNS design is also based on the following assumptions: 1) a P2P VPN creates a social graph where links represent relationships between ends users, 2) each P2P VPN tunnel is an authenticated and encrypted end-to-end link similar to an IPSec connection, and 3) the previous two assumptions makes it non-trivial for a malicious user to mount a Sybil attack. The authentication and encryption of the IP tunnels is discussed in previous work [4]. Previous work [7] have also shown that Sybil attacks are much harder on P2P networks with social links versus anonymous links.

4.1. Enabling Short Domain Names

One of SocialDNS's key roles is to enable short domain names to resources in the P2P VPN, for example, Alice can set *alicepc.sdns* as the domain name for her personal computer. The first step in enabling short domain names is to choose an unallocated root-level domain zone to avoid conflicts with the global DNS. SocialDNS chooses an unassigned root-level domain zone; therefore preventing phishing attacks based on

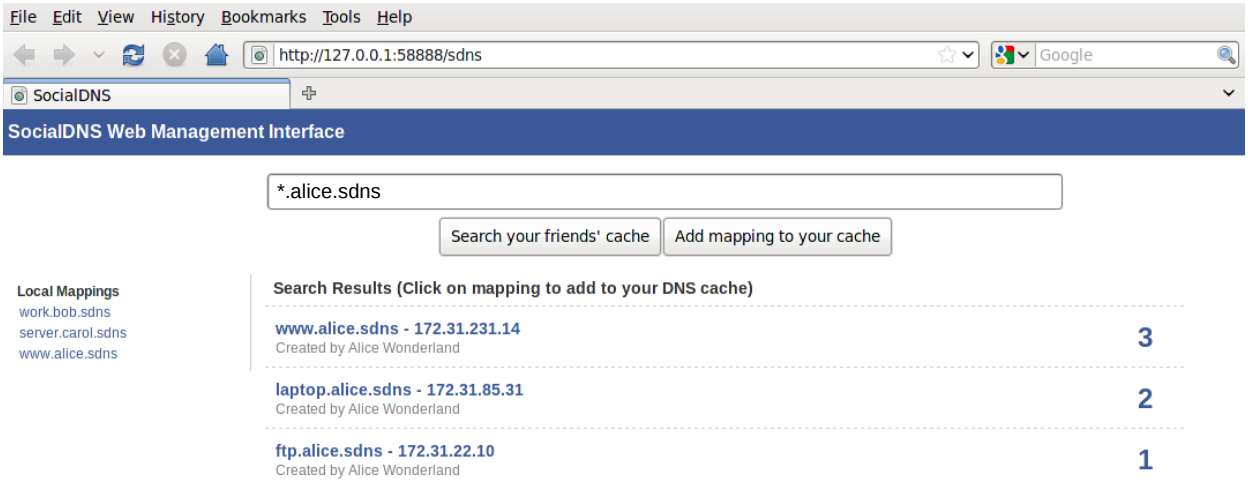


Figure 2. **SocialDNS AJAX Web Interface.** In this screenshot, Bob does a wildcard search for all DNS mappings with the **.alice.sdns* search string. By clicking on the *Search your friends' cache* button, the search is sent to all friends through the P2P VPN. As the responses arrive from friends, the Web interface is updated along with a ranking for each mapping on the right. The ranking information is used to sort the mappings, the end user makes the final selection on the mapping to add to the local cache. Existing local cache mappings are shown on the left hand side.

addresses used on the public Internet i.e. by creating a fake DNS mapping for *www.bankofamerica.com*. A second requirement is to limit the scope for uniqueness. The global DNS system provides naming for the whole world, potentially billions of hosts thus making short-names scare. In the P2P VPN setting, uniqueness is only required within a peer's social circle which is usually composed of a few hundred or at most a few thousand hosts. The reduced scope lessens the probability of collision for short-names such as *alicepc.sdns*.

4.2. Decentralization and Broadcasting

We achieve decentralization in SocialDNS by running a local DNS service on each peer's machine; hence there is no head node or central point of failure. We also leverage the unstructured peer-to-peer social graph created by P2P VPNs as the messaging substrate connecting these local DNS servers. The local DNS service provides users with an interface when they can easily add and remove DNS mappings. The SocialDNS nodes use one or two-hop broadcast messages to communicate amongst each other thus allowing users to perform arbitrary searches on each other's SocialDNS caches. Using broadcast greatly simplifies the design due to its stateless nature and topology independence. As users create new DNS mappings, they are able to freely share these mappings with social peers over the P2P VPNs. Therefore, a typical DNS query can search the local DNS cache as well as the DNS caches of friends in the social circle. This is analogous to the common Gnutella P2P file sharing paradigm where each peer runs a local file server, and other peers are able to broadcast queries to search for files. In the SocialDNS case, each peer runs a DNS server instead of a file server, broadcast queries are sent to social peers instead of random peers, and the results are DNS mappings instead of files.

4.3. Simple User Interface and Management

Managing a typical DNS server such as BIND is a serious undertaking, a task suited mainly for network administrators [8]. Our focus is to provide simple management to the user with minimal configuration and an intuitive Web interface for creating, deleting, searching, and sharing DNS mappings. The user experience for the SocialDNS system involves just a few steps. First, by running the SocialDNS service on the local machine, startup scripts configures the operating system's settings and point to the local server as one of the DNS servers. The local DNS server only resolves requests under the *.sdns* root domain zone to avoid common DNS phishing attacks and to allow coexistence with the global DNS. The user then uses their web browser access the SocialDNS interface (see Figure 2).

4.3.1. Creating User-Defined Mappings. Users can add new mappings to their local SocialDNS cache in one of two ways. The first method is for the user to manually create a SocialDNS mapping through the Web interface: for example Alice user the input box to

map the name *alicepc.sdns* to her local PC. The newly created mapping will only be accepted if it follows the following criteria: 1) the mapping ends with *.sdns*, and 2) it points to a virtual IP address in the P2P VPN address range. SocialDNS gives the user the freedom to pick any DNS mapping under the *.sdns* root zone.

4.3.2. Importing Mappings through Search. The second method is through a SocialDNS search which allows users to query each others' SocialDNS caches for mappings. The searching process resembles a typical Web search, and the results are presented to the user ranked by our ranking method described below. A user can choose to import a mapping to his/her SocialDNS cache by simply clicking on the mapping.

Figure 2 shows a search for **.alice.sdns* and the results of that search and the user can choose to import these mapping locally. Since this search is done as a typical flood-based broadcast in an unstructured P2P network, SocialDNS therefore supports various types of queries such as exact, nearest, and regular expression matching. The search is done by broadcasting the DNS query to all friends and maybe friends of friends depending on user settings. Using broadcast queries with greater hop counts gives the local user a more accurate view of the popularity of a DNS mapping at the cost of generating more traffic and higher latencies.

4.3.3. Freedom to Redefine Domains. SocialDNS also gives users the option of redefining the domain name point to a friend's resource in his/her local DNS cache. This is similar to creating a bookmark to a web page, but instead of using the page title provided by the website, the user can create his/her own reference to that web link. SocialDNS operates in a similar fashion because the user can import the existing mapping defined by a friend or create another mapping at his/her discretion. For example, let's say Bob chooses *bobpc.sdns* as the SocialDNS name for his machine, Alice can either import that domain mapping or define her own mapping such as *bobby-mac.sdns* for Bob's machine. Consequently, SocialDNS allows a user to create map multiple domain names to the same IP address.

Creating a different mapping for a peer's resource only makes that mapping available the local SocialDNS cache, however peer's have the option of importing that mapping in their own SocialDNS cache through search. Extending the previous example, assuming Carol is friends with both Alice and Bob, through a SocialDNS search, Carol noticed that Bob the following mapping *bobpc.sdns = 172.31.231.23* while Alice has *bobby-mac.sdns = 172.31.231.23* and both mappings point to the same IP address, Carol will have the option of choosing either Bob's mapping, or Alice's mapping, or both. Once again, DNS mappings can be thought of as bookmarks to webpages, and SocialDNS makes it easy to create any mappings and share it with friends. As a mapping is replicated across peer's SocialDNS cache, its popularity in the social circle and that creates the basis for our rank system described below.

4.4. Name and Conflict Resolution

SocialDNS performs two types of name resolution: trusted and automatic. Providing an automatic mode gives the end user the option of not having to manually import each mapping created by social peers, but supporting a trusted mode is crucial for some secure services. The mode of operation is a configuration the user is able to set at startup or runtime.

4.4.1. Trusted Name Resolution. In trusted mode, SocialDNS only resolves mappings that have been explicitly created or imported by the user through the web interface. This mode is dubbed trusted because each DNS mapping is approved manually by the user therefore will always point to the consistent IP address. This is important because changing a DNS mapping from $IP1$ to $IP2$, which can happen in automatic mode without user intervention, may cause information leakage due to web browser cookies. Hence, users should use the SocialDNS trusted resolution for information sensitive services.

For a trusted name resolution, the local DNS server utilizes its SocialDNS cache only to resolve incoming DNS queries from the operating system. In other words, when Bob types *www.alice.sdns* in his browser, the browser asks the operating system to resolve the DNS name, the request is forwarded the local SocialDNS service, the mapping is looked up in the local cache, if found, the IP address is returned, if not found an NXDOMAIN response is sent back to the application. In this mode, there are no name conflicts because they are resolved by the user through the web interface. For example, if Alice tries to create or import a mapping such as *www.alice.sdns* which already exists in her local SocialDNS cache, she will be informed of the collision and required to choose a different domain name for the resource. Also, in the web interface, if there can be multiple search results with the same ranking; in this case, the user makes a selection on the mapping they would like to import thus resolving the name conflict themselves.

4.4.2. Automatic Name Resolution. In automatic mode, SocialDNS automatically searches the social circle for mappings that are not present in the local cache and picks the highest ranked mapping. This mode of operation is insecure because it can lead to frequent changes in IP address, because the resolution is based on the most popular mapping at the time of the query. To perform this resolution, the SocialDNS system sends an exact match search to all friends, waits for a predefined time interval to gather results, and the highest ranked mapping is chosen. In the case of a tie, we randomly pick one of the highest ranked SocialDNS mappings.

4.4.3. Ranking Domain Names. We do not enforce uniqueness during the creation of the DNS mappings because we believe in using social context to help with the DNS resolution; therefore, *www.alice.sdns* can map to one IP address in one social circle, and to a totally different IP address in another social circle. In the case where both mappings exist in the same social context, a ranking algorithm is used to provide preference to one mapping over another. So *www.alice.sdns* will map to the IP address with the highest ranking, and the local user will have to specify an alternate name for the conflicting mapping.

Our current ranking algorithm is simple; when a DNS query is sent to all friends, the friends search their DNS cache for matching results and send the responses back to the requester, mappings are ranked based on an aggregation of the responses. For instance, if five friends return responses saying *www.alice.sdns* maps to 172.32.122.31, while two friends says that it maps to 172.15.223.112, then the first mapping will get a ranking of five, and the second a ranking of two. Once ranked, the mappings are presented to the local user for selection (see Figure 2). With this simple scheme, we use the presence of a SocialDNS mapping in a peer's local cache as a vote for that mapping, the more peers that have a mapping in their cache, the more votes that mapping gets. This is similar to the ranking system used by Delicious.com where a bookmark's popularity increases as more people add that bookmark to their accounts.

4.5. Protection against DNS attacks

. Attacks such as phishing, session hijacking, cache poisoning have plagued DNS and it requires careful administration and robust software to protect against the DNS security flaws. Hence, it is important for the SocialDNS design to avoid these same flaws currently plaguing the current DNS system. SocialDNS, however, benefits from inherent network level security provider by P2P VPNs. In a P2P VPN each tunnel is encrypted and authenticated either through the use of IPSec, or a TCP/IP level encryption such as TLS or DTLS. Therefore, every network packet (i.e., SocialDNS request/reply, probes, trust aggregation messages) is can be non-repudiatively bound to a peer's identity making it impossible to spoof an IP packet or a DNS response. This secure primitive hence makes attacks such as phishing and cache poisoning futile because the culprit can be detected and banned for the P2P VPN.

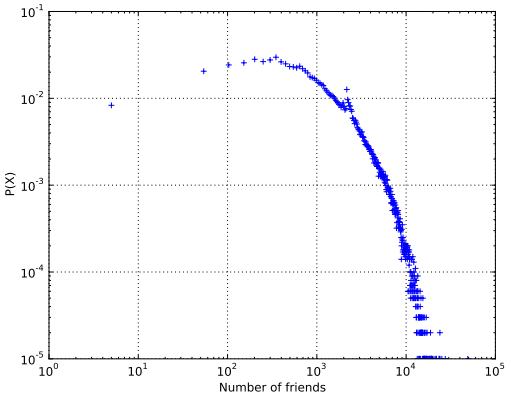


Figure 3. **Distribution of Number of Friends in 2-hop Radius of Social Circle.** As shown in this distribution, a user has about 2000 friends in a two-hop radius on average.

5. Analysis

In our analysis, we explore the following aspects of our design: number of peers in the social scope, bandwidth cost, latency, and conflict resolution. Due to our assumption that the P2P VPN create a social graph, we have used a 100,000 node sample social data captured from Orkut. The dataset was provided by the authors of [9]. We then used NetworkX [10], a Python package for complex network analysis, to study the different aspects of our design through the social graph. Our social graph contains the following small-world characteristics: 1) an average clustering coefficient of 0.27, 2) and a powerlaw degree distribution.

5.1. Reduced Conflicts in the Social Scope

A major strength of SocialDNS is the increased availability of short names through social scoping, meaning because we do not have to guarantee global uniqueness as in the case of regular DNS, uniqueness is

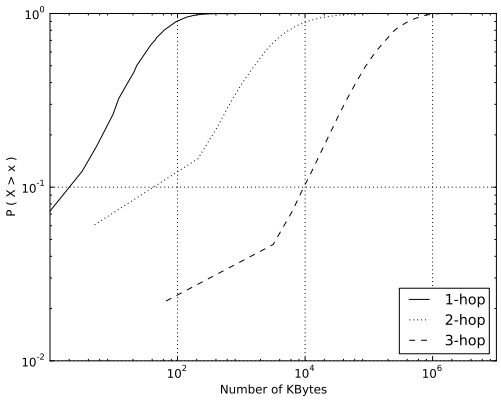


Figure 4. **CDF of One, Two, and Three hop Queries.** As the graph shows, increasing the hop count for SocialDNS Queries increases the bandwidth consumption by two orders of magnitude.

only guaranteed within the social circle. Since the recommended social scope for uniqueness is SocialDNS two hops, we examined the distribution of the number of friends in a two-hop radius to get an idea for the number of peers that may be competing for the same domain name. Through our analysis, we observed that in a two-hop radius of the social circle the average number of friends are 1,832, and a median of 1,150. Therefore Alice would only have to compete with less than 2,000 peers on average to claim the name *alice.sdns* instead of the billion of users on the Internet to guarantee uniqueness in a two-hop radius of the social circle. The reduced number of peers makes finding a short SocialDNS domain name much easier. Figure 3 shows the actual distribution of the number of host within a two-hop social circle.

5.2. Expected Bandwidth Cost

SocialDNS uses broadcasting as the primary method of communication, therefore understanding the expected bandwidth costs helps us predict the traffic generated by SocialDNS queries. Figure 4 shows us the cumulative distribution function (CDF) of the expected number packets generated when a user does a one, two, or three hop search for a SocialDNS mapping. We assume the maximum packet size allowed in the DNS RFCs [11] of 512 bytes. Hence, a one, two, and three hop SocialDNS search generates about 42 Kbytes, 4.5 Mbytes, and 161 Mbytes of traffic on average, with a medians of 25 Kbytes, 1.7 Mbytes, and 90 Mbytes respectively. This data suggests that a good design should carefully consider the bandwidth load of a three hop broadcast.

5.3. Anticipated Latency

Analyzing the latency helps us determine the proper timeout to set per SocialDNS query to gather adequate responses from social peers. The first step in our latency analysis is to examine the relationship between friendships and geography. According to Liben-Nowell et al. [12], friendships in a social network are based on a geographic preference given by formula $P(d) = \frac{1}{d^{1.2}} + 5.0 \times 10^{-6}$ where $P(d)$ is the probability of friendship between two peers that are located at distance d kilometers away. Based on the works of Bassett et al. [13], one can derive latency from distance by approximating latency as $\frac{4}{9}$ the speed of light. Another work by Dischinger et al. [14] also has shown that in residential ISPs, a packet may take up to 2.5 milliseconds from the host machine to the ISP's router for a total round-trip time of 5 milliseconds. Using these previous works, the latency in milliseconds is approximated as:

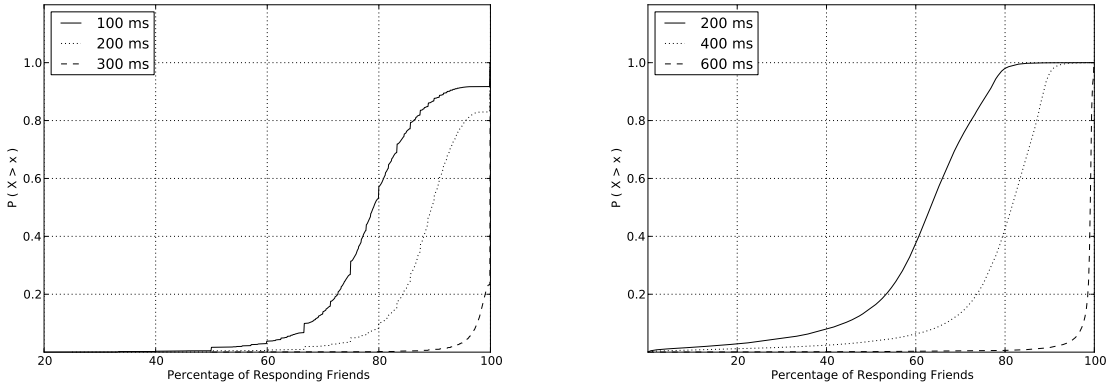
$$latency = \frac{d \times 10^3}{\frac{4}{9}c} + 5 \times 10^{-3} \qquad (1)$$

where c is the speed of light at 299,792,458 m/s. Hence we used the probability function $P(d)$ to assign distances between friends and the equation (1) to approximate the latencies of friendship links in the social graph created by the P2P VPN.

Based on the aforementioned latency distribution, we measured the percentage of received responses based on timeouts of 100 ms, 200 ms, and 300 ms for one-hop queries, and 200 ms, 400 ms, and 600 ms for two-hop queries. For a timeout of 100 ms for a one-hop broadcast, we received responses from 79% of friends on average, 89% for 200 ms, and 99% for 300 ms. In the two-hop broadcast scenario with timeout of 200 ms, 400 ms, and 600 ms and received responses from 61%, 78%, and 98% of friends on average, respectively. Figure5 plots CDF showing the distribution of the percentage of responding friends with the various timeouts for one and two hop queries.

6. Implementation and Experiments

We currently have a preliminary prototype of the SocialDNS system implemented in the SocialVPN [4] software stack. The Web-based interface is implemented as a search engine-like interface written in AJAX (see Figure 2). Through the interface, users can search all of their friends' SocialDNS cache for mappings to import locally, or create new mappings of their own.



(a) Timeout Latencies for One-hop Broadcast

(b) Timeout Latencies for Two-hop Broadcast

Figure 5. **Impact of Query Timeouts. For one-hop queries**, at 100 ms median is at about 75% and we hear back from all friends 10% of the time; the median at 200 ms is about 90% and we hit 100% results about 16% of the time; at 300 ms, our median is at 100%, and we get all results 75% of the time. **For two-hop queries**, at 200 ms, we obtain a median of 65%, but we obtain 100% less than 1% of the time; at we achieve a better median of 80%, with barely any improvement at the 100% mark; at 300 ms, we get a much better median of 99%, and we hit 100% about 3% of the time.

Our prototype uses a P2P library called Brunet also written in C# which supports various types of messaging for P2P overlays. The Brunet library features an RPC system which we used to add the necessary functions to support our SocialDNS service. The Brunet RPC system is a powerful feature because it allows for fast prototype by providing a simple abstraction to route information between peers. So, once we know the addresses of our friends, we can route DNS queries as RPC calls over the peer-to-peer layer of the VPN.

We conducted experiments to assess the functionality and performance of our prototype. The experiments involved both PlanetLab [15] and Amazon Elastic Cloud [16] (EC2) infrastructures. PlanetLab, a global research testbed with nodes located around the world, was used to deploy 600 P2P nodes which formed our bootstrap P2P overlay. The P2P nodes on PlanetLab did not include the SocialDNS software stack. We deployed SocialDNS nodes on Amazon EC2 because our system requires adding the local SocialDNS server to the operating system's DNS configurations. Modifying the DNS settings is not possible on PlanetLab nodes, but, this is possible on Amazon EC2 because we have total root access to the virtual machine. On the Amazon EC2 nodes, we added DNS mappings to the local caches, searched and imported DNS mappings from social peers, and resolved DNS mappings created by the local user and friends. Hence, our design proved feasible and all components integrated successfully.

6.1. Experimental Latency

We also wanted to explore the latencies of the DNS queries through the peer-to-peer overlay running on PlanetLab with nodes located around the globe. Although the magnitude of the network size in our experiment is significantly smaller than the expected size of an entire social network, we assume that techniques exist to create "social overlays" which restrict the scope of queries to a user's social circle. Three SocialDNS nodes were deployed on the various regions of the Amazon EC2 infrastructure located in Virgina, California, and Ireland; a fourth node was located in Florida from a residential ISP. Figure 6 shows the cumulative distribution function (CDF) of 3000 DNS queries conducted at each of the four nodes. The results show that more than 90% of the DNS requests take less than one second irrespective of the geographic locations. This delay is associated with packet loss and the retry mechanism of the Brunet RPC system.

It is also desirable to see how long it takes to broadcast DNS queries simultaneously to all friends through the overlay. Assuming a private peer-to-peer overlay [17] consisting only of our social peers, we measured the time taken to broadcast queries to all peers with network sizes of 200, 400, and 600 nodes. Using only the PlanetLab nodes, we sent 100 broadcast queries to each of the different sized networks. Figure 7 shows the 25th, 50th, 75th, and 95th percentile of time taken to broadcast to the whole network. The 95th percentile are 8, 12, and 16 seconds for network sizes of 200, 400, and 600 nodes, respectively.

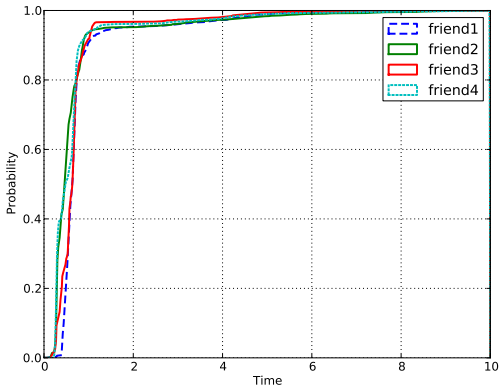


Figure 6. **CDF of User Request Times.** Each of the four user sent 3000 DNS requests (1000 to each friend). The four nodes were located in Virginia, California, Florida, and Ireland. We measured the round-trip latency of each DNS query.

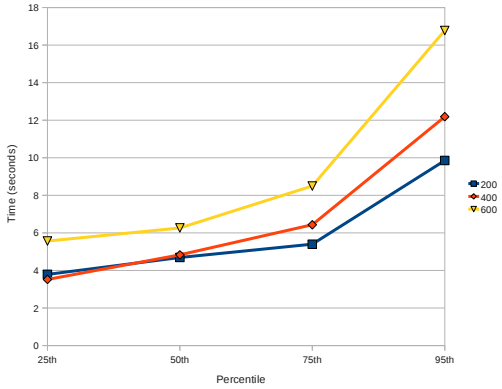


Figure 7. **Time taken to broadcast varying size networks.** 100 broadcast queries were sent to each network size. 95% of the queries took less than 16 seconds for a network size of 600 nodes.

7. Related Works

There have been various previous works aimed at improving the DNS experience. One of the first is by Cox et. al. [18] who discussed the implementation of a DNS system on top of the Chord DHT. Although such an approach was feasible, the latencies of a DHT-based DNS were much higher than conventional DNS. Our approach is not depend on a structured DHT and is deployed on top of an unstructured social peer-to-peer network through the use of P2P VPNs.

Walfish et. al. [19] suggested using semantic free references (SFR) as a replacement for DNS-based URLs on the web. They also suggested using a structured DHT to store self-certifying o-records containing pointers to resources. Their focus was on decoupling the mnemonics names from the actual references which were 160-bit hash tags, and a whole separate mechanism, maybe social, to map names to the complex tags. The SFR design therefore suggested a total redesign of referencing on the Internet. CoDNS [20] is a system which automatically forwards pending DNS requests to another local DNS server to a remote administrative domain. The rationale is that most DNS failures are associated with poorly configured local DNS server, by sending long pending requests to different local DNS server in another domain, it provides some redundancy to local DNS failures. The SocialDNS system reuses the current DNS protocol without requiring any re-architecting and simply supplements current DNS systems.

Allman [21] introduced the concept of personal namespaces (*pnames*) to provide easier references to their email, blogs, links, and so on. Every user is given an NID which is a cryptographic hash of the public key. These NIDs are shared among friends through a one-time exchange and are used to retrieve the mappings of a user's namespace from a DHT. The *pnames* system share many similarities with SocialDNS, but it is proposed as second level of indirection to name resolution and the resolution are not restricted to DNS records. For example, *alice:email* would resolve to *alice@mailserver.com*, a second resolution would then be required to resolve *mailserver.com*. Since NIDs are always unique, and the local user defines the mapping of names to NIDs, name conflicts are not an issue. SocialDNS has to deal with name conflicts and uses a social conflict resolution to help rank various names.

8. Conclusion and Future Work

In this paper, we present a decentralized domain naming solution suited for P2P VPNs. We make the case that provide IP access is not enough to enable collaborative virtual organizations, give users the freedom to create short names to the services that they host is also necessary. We presented SocialDNS which provides that freedom and reused existing concepts from decentralized naming solutions for in local area networks. We also improve upon the current limitations of LAN-based decentralized naming systems that makes them unsuited for a P2P VPN environment.

The SocialDNS design makes it trivial to assign short domain names to resources through a P2P VPN through social scoping, meaning, a user only has to compete with less than 2,000 host on average for a domain name. For simple management, we have designed a minimal configuration, easy-to-use AJAX

Web interface, where a user can search his/her friends's SocialDNS cache and import mappings of interest For design simplicity, broadcast is used as the main method of communication among SocialDNS nodes in the P2P VPN. In case of name collision due to the lack of a central authority, a popularity-based ranking mechanism is employed to pick the mapping that it present in the most caches in the social circle.

We use social graph analysis to predict the bandwidth cost and responsiveness by assuming a P2P VPN will form a social graph. Our current prototype only allow for one-hop broadcast, and we are working on support two-hop broadcasts, but nothing beyond that. For future work, we plan on exploring gossip techniques that can help propagate both SocialDNS mappings and reputation information at a controlled rate. The hope is to minimize both bandwidth consumption and latency through more intensive caching of social information. We also plan on investigating other potential reputation mechanisms that may also consume less bandwidth.

9. Acknowledgements

This research is sponsored by the National Science Foundation under grant IIP-0758596, CCF-0622106, the South East Alliance for Graduate Education and the Professoriate, the Florida Education Fund under the McKnight Doctoral Program. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

References

[1] M. Krochmal, "Multicast dns internet draft," http://files.multicastdns.org/draft-cheshire-dnsext-multicastdns.txt/.

[2] "Multicast dns internet draft," http://technet.microsoft.com/en-us/library/cc784180(WS.10).aspx.

[3] "Hamachi - instant, zero configuration vpn." https://secure.logmein.com/products/hamachi/vpn.asp?lang=en.

[4] "Socialvpn," http://socialvpn.org/, 2010. [Online]. Available: http://socialvpn.org/

[5] "Opensimulator," http://opensimulator.org, 2010. [Online]. Available: http://opensimulator.org

[6] "Wippien p2p vpn," http://wippien.com/, 2010. [Online]. Available: http://wippien.com/

[7] C. L. Laas, "A sybil-proof one-hop dht," in *SocialNets '08: Proceedings of the 1st Workshop on Social Network Systems*. New York, NY, USA: ACM, 2008, pp. 19–24. [Online]. Available: http://dx.doi.org/10.1145/1435497.1435501

[8] M. D. Bauer, "Securing dns and bind," *Linux J.*, p. 2, 2000.

[9] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee, "Measurement and analysis of online social networks," in *Proceedings of the 5th ACM/USENIX Internet Measurement Conference (IMC'07)*, October 2007.

[10] "Networkx - high productivity software for complex networks," http://networkx.lanl.gov/, 2010. [Online]. Available: http://networkx.lanl.gov/

[11] P. Mockapetris, "Domain names - implementation and specification," http://tools.ietf.org/html/rfc1035, 1987.

[12] D. Liben-Nowell, J. Novak, R. Kumar, P. Raghavan, and A. Tomkins, "Geographic routing in social networks." *Proceedings of the National Academy of Sciences of the United States of America*, vol. 102, no. 33, pp. 11 623–11 628, August 2005. [Online]. Available: http://dx.doi.org/10.1073/pnas.0503018102

[13] E. K. Bassett, J. P. John, A. Krishnamurthy, D. Wetherall, T. Anderson, and Y. Chawathe, "Towards ip geolocation using delay and topology measurements," in *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*. New York, NY, USA: ACM, 2006, pp. 71–84. [Online]. Available: http://dx.doi.org/10.1145/1177080.1177090

[14] M. Dischinger, A. Haeberlen, K. P. Gummadi, and S. Saroiu, "Characterizing residential broadband networks," in *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*. New York, NY, USA: ACM, 2007, pp. 43–56. [Online]. Available: http://dx.doi.org/10.1145/1298306.1298313

[15] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "Planetlab: an overlay testbed for broad-coverage services," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 3, pp. 3–12, 2003.

[16] "Amazon elastic compute cloud," http://aws.amazon.com/ec2/, 2008.

[17] D. I. Wolinsky, P. St. Juste, P. O. Boykin, and R. Figueiredo, "Towards social profile based overlays," University of Floriday, Tech. Rep., Feb 2010. [Online]. Available: http://arxiv.org/abs/1002.0865v1

[18] R. Cox, A. Muthitacharoen, and R. T. Morris, "Serving dns using a peer-to-peer lookup service," in *In IPTPS*, 2002. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.12.620

[19] M. Walfish, H. Balakrishnan, and S. Shenker, "Untangling the web from dns," in *NSDI'04: Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation*. Berkeley, CA, USA: USENIX Association, 2004, pp. 17–17.

[20] K. Park, V. S. Pai, L. Peterson, and Z. Wang, "Codns: improving dns performance and reliability via cooperative lookups," in *OSDI'04: Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation*. Berkeley, CA, USA: USENIX Association, 2004, pp. 14–14.

[21] M. Allman, "Personal namespaces." ACM SIGCOMM Hotnets 2007, November 2007.