Davis Social Links: Integrating Social Networks with Internet Routing

Lerone Banks Shaozhi Ye Yue Huang S. Felix Wu
{ldbanks, sye, yuehuang, sfwu}@ucdavis.edu
Department of Computer Science
University of California, Davis
One Shields Ave, Davis, CA 95616

ABSTRACT

Internet based communication systems appear in many forms from email to Instant Messenger and chat services to Voice over IP (VoIP). Common weaknesses of these current communication systems include inherent security vulnerabilities to attacks such as SPAM/SPIT/SPIM and DDoS, a lack of separation between routing address and identity, and a lack of message receiver controllability. Based on these observations, we describe preliminary ideas and analysis of a new communication architecture called DSL (Davis Social Links), which integrates the trust relationships of social networks into the internet communication infrastructure to provide receiver controllability, traceability, and global connectivity without centralized services or globally unique IDs. Although it is still in the early stages of development, we believe that DSL can help thwart many large-scale information-based attacks faced by the Internet community.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems

General Terms

Design, Security

Keywords

Social network, SPAM, P2P, routing

1. INTRODUCTION

The current Internet architecture as well as many application layer communication frameworks, such as email, VoIP, or instant messenger, does not provide sufficient *controllability* for its end users. Each communication party has one or more *routable identities*, e.g., domain names, IP or email addresses. Once an identity becomes public, the owner of that identity, in general, cannot make it "unknown" to the world unless he/she changes to a new identity (and, of course, the same problem repeats). If this identity is an email address, it will be collected and used by spammers. If it is an IP address (maybe being resolved from a domain name), it will become a DDoS target for "bots."

Communication activities in general represent a trust relationship between the communicating parties. When two users communicate, both parties should know how much trust they share. Depending on the trust level, the message might be treated very differently. Today's cyber communication infrastructure such as IP or email, however, does not explicitly utilize the concept of trust in the infrastructure itself. In fact, the receiver (or a middle box such as an intrusion prevention system) has to derive such somewhat implicit information from the header or the payload to determine how to appropriately treat the incoming message/packet. As a result, the solutions we have today, under the current communication infrastructure, are either not scalable or ineffective, or both. Not having *trust* as a first class citizen in the communication infrastructure, as we believe, is a key reason for the Internet community having so many information-based large-scale attacks such as DDoS, SPAM/SPIT/SPIM, Phishing, and Botnet.

Another related issue in the current Internet architecture is *traceability*. Today, we have very limited traceability for our communication activities. Traceroute or similar tells us which network routers are in the middle of an end-to-end communication path. Various probabilistic packet marking or ICMP traceback schemes have been proposed, but it is still unclear how these tracing information will help us to defend our network better. Methods for stepping stone tracing for certain applications (such as VoIP/Skype or SSH/Telnet) have been developed. While we might use such network forensic information (if we can obtain them, in the first place) to hunt down some sources of malicious activities, the victim has usually suffered already. Given the recent development of bots and their future P2P-style command and control, our chance to win, in any way, is really decreasing. We argue that, in developing the architecture presented here, tracing the source along the physical communication path around the Internet is insufficient. The source must be accompanied with the trust/social relationship between the communicating parties. In other words, we need to trace not only which intermediate network routers have been used to route a message from Alice to Bob but also "what trust relationships did Alice use to communicate with Bob?" Without the latter, it would be hard for the

LSAD'07, August 27, 2007, Kyoto, Japan.

receiver/victim or the service provider to effectively defend against unwanted traffic.

Based on our observations of certain security issues in today's Internet, we propose a new communication architecture called DSL (Davis Social Links), where we integrate/collapse the trust relationships, normally being only maintained, in the application layer into the network service infrastructure itself explicitly. The key idea of DSL is to route/forward packets from one network entity to another only via one of the many social network paths between them. A social link path represents a quantitative direct/indirect trust relationship between two parties as well as the intermediate social routers supporting the communication. More importantly, without proper trust relationships, it will be very difficult for an attacker (even an insider) to even obtain a valid social network path to reach a particular destination. Once a social network path is established, the trust ranking for that particular communication path can be utilized by the receiver to properly prioritize incoming packets/messages. Furthermore, upon being attacked, the receiver can either downgrade the trust associated with the communication path or even withdraw from that particular social link relationship.

While the DSL project is still in its infancy stage, in this paper, we describe the architecture as well as a few design principles. Our simulation evaluation is still very preliminary, while some of the technical challenges are still being actively developed.

2. THE DSL (DAVIS SOCIAL LINKS) ARCHITECTURE

Conceptually, DSL is built on top of social networks among individual network entities. These relatively small strongly connected networks are joined together to create a large overlay network based on the small-world phenomenon [12].

In order for a pair of nodes to communicate, they must either share a direct link or find a communication path through the social network consisting of intermediate nodes willing to route messages. Nodes with strong social relationships are connected by direct links. Nodes joined by these links can transmit messages using a pre-established channel without the aid of DSL. Nodes that wish to communicate with other nodes for which they do not have a direct link must attempt to find a routable path [1] through the social network to the desired destination. This route discovery is made possible first by the exchange of keywords between nodes connected by direct links. These keywords represent the interests of the corresponding node and are used by a node attempting to discover a route to a particular node as tokens to "buy" routing service.

Route discovery begins when the node desiring to send a data message, referred to as the sender, creates a route discovery message (RDM) based on information about the intended receiving node, referred to as the recipient. The RDM is a query for a path to the receiving node containing the intended recipient's keywords. At each hop, an intermediate node, which can be thought of as a Social Network Router (SNR), uses local rules to match the keywords included in the RDM to the keywords of its directly connected nodes and forwards the RDM along the appropriate direct links. When the RDM reaches the intended recipient, the receiving node responds with a source-routed (ACK) message that contains the path that the sender will use for data transmission. When the sending node receives the (ACK) message it can use this path to transmit data messages.

DSL's keyword-based route discovery is actually similar to a decentralized version of Google. In Google, we provide a set of keywords and the search engine of Google will provide ranked pointers to a set of webpages matching these input keywords. One difference between Google and DSL is that, in Google, both ranking and searching are being done under a centralized administrative domain. Another difference is that, in Google, the keywords corresponding to some specific content (i.e. search terms that produce particular search results)are being extracted by Google, while, in DSL, the content itself (or the owner of the content) has the authority to decide what keywords to announce under this totally decentralized framework.

Among many issues related to DSL, in this paper, we will focus on two novel architectural designs:

- *Recipient Controllability*: DSL provides a recipient controlled routing protocol. A user/node is able to control the routing path via which other users/nodes send messages to it, and automatically rejects unwanted messages. This enables an effective defense against spamming types of attacks.

- *Global Connectivity without Global Identities*: DSL employs addressless routing, i.e. there is no global ID for users/nodes. DSL accomplishes this by embedding the path information into RDMs and using a set of loop detection methods including bloom filter based duplicate detection, hop count checking and forwarding path analysis. Please note that, in today's Internet, we have global routable identities (such as IP or email addresses). Some recent works such as [2] propose to separate identities from routing addresses (e.g., to support better mobility). However, we still need to administratively assign unique identities and routing addresses within the Internet. In DSL, we do not have any form of global identities. The concept of identity in DSL is completely in the application layer, a particular application will define its own identification scheme for the network entities using that application.

3. NOTATIONS

This section defines the basic notations used in this paper.

Let $G = (V, E)$ be an undirected graph, where V represents a set of vertices and E represents a set of links between the vertices in V. If v and u are two vertices in V, then $e(v, u)$ is a Boolean function representing whether there is a direct link in E connecting v and u (i.e., $e(v, u) = 1$ if v and u are directly connected in G, otherwise $e(v, u) = 0$). By default, $e(v, v) = 1$.

If $e(v, u) = 1$, then u is a *neighbor* of v and vice versa. Each vertex has a set of neighbors. $N(v, i)$ represents the ith neighbor of vertex v. In other words, if $e(v, u) = 1$, then $\exists i$, $N(v, i) = u$ and $\exists j$, $N(u, j) = v$. By default, $N(v, 0) = v$, $\forall v$. Furthermore, we denote $I_N(v, u) = i$ (i.e., the index of u in v's neighbor list is i), and $I_N(u, v) = j$ (i.e., the index of v in u's neighbor list is j). Finally, we define the

[1]Since we only care about a routable path here, without further specifying, a path means a routable path in the rest of this paper.

Table 1: List of Symbols

Symbol	Explanation
$N(v,i)$	a node which is v's ith neighbor
$I_N(v,u)$	the index of u in v's neighbor list
$\#(v)$	the number of v's neighbors, other than v itself
$R(v,u)$	Boolean, whether it is routable from v to u
$K_{Ann}(v,i)$	the set of announced keywords which v sends to its ith neighbor
$K_{Agg}(v)$	the set of keywords aggregated by v
$K_{Inc}(v)$	the set of incoming keywords which v gets from all neighbors
K_{Des}	the set of keywords for destination nodes acceptance
K_{Int}	the set of keywords purely for intermediate nodes acceptance
$K_{RDM}(v)$	$K_{Des}(v) \cup K_{Int}(v)$ in the RDM sent from v
$K_{Ack}(v)$	the set of keywords which v is interested in
$T_{Ack}(v)$	the acknowledgement threshold of v
$T_{Ign}(v)$	the ignore threshold of v
S_F	A stack in RDM to record the forwarding path for ACK delivery
S_D	A stack in ACK to record the forwarding path for data transmission

number of neighbors of a vertex v other than v itself, $\#(v) = \sum_{u \in V, u \neq v} e(v,u)$.

Let $R(v,u)$ be a Boolean function representing whether it is routable to send a message from v to u. While the complete definition of the function R will be given later, we now define that $R(v,u) = 1$ and $R(u,v) = 1$ if $e(v,u) = 1$ (please note that this is *if*, not *iff*). By default, $R(v,v) = 1$ (reflective routability).

The other notations are listed in Table 1, which will be introduced in the following sections.

4. KEYWORD BASED ROUTING CONTROL

Besides the trust from social relationships, DSL provides a recipient controlled routing protocol, which incorporates application layer semantics (represented by keywords), to get a balance between security and routability.

4.1 DSL Trust Model

Some existing work [3] [7] [5] considers a message sent by a trusted party to be trustworthy while there are several fundamental problems with this assumption.

- The trust may be dynamic. A party may be trustworthy at some point in time and then later untrusted. For example, a former employee is not as trusted by a company as s/he was when s/he was a current employee. Another good example is a compromised trusted party, such as bot machines.
- The trust may be non-transitive. Some previous work suffers from the case when transitivity does not hold. For example, A trusts B and B trusts C, while A does not trust C.
- The trust may be non-uniform. For example, A trusts B as a colleague for job-related issues but may not trust B's opinion on personal matters.

The previous work has to employ tedious, and sometimes ad hoc, penalizations to deal with these problems. DSL's trust model, however, requires that the message has to be sent/forwarded by a trusted party and at the same time satisfy required application layer semantics. This model helps DSL avoid the above problems and simplifies the design. We will further explain this model in the following sections.

4.2 Keyword Based Filtering

We first formally define a few primitives in DSL. A *message* is an application layer object and a *keyword* is a specified property for objects. *Matching* is the process of evaluating whether a message is consistent with a keyword, resulting in either *match* or *not match*.

For example, in email systems, an email is a message, and the keyword may be a string, a regular expression, or a finite state machine(FSM). The corresponding matching process is to see whether the email contains the string, qualifies the regular expression, or produces a qualified state of the FSM. For VoIP, a phone call is a message, and the keyword may be a caller ID or a certain speech pattern.

Treating a neighbor as a trusted party and using keywords to represent the application layer semantics, the DSL trust model can be described as follows: Each node in DSL has a set of keywords denoted as K_{Ack} and only acknowledges a *route discovery message* which is sent by a neighbor *and* matches K_{Ack} (Section 5.1 formally describes route discovery and matching). A path between two nodes will be established for data transmission *iff* they pass the route discovery process. The following *data transmission messages* are not subject to keyword filtering.

Besides filtering at local hosts, DSL employs *keyword exchange* so that each DSL node can do filtering for other nodes. More specifically, if $e(v,u) = 1$, then v and u will exchange keywords with each other. $\forall i$, $K_{Ann}(v,i)$ is the set of *announced keywords* being exchanged from the vertex v to its ith neighbor. K_{Ann} may be different from K_{Ack}. In either case, node u's third neighbor v that receives $K_{Ann}(u,3)$ from a node u cannot determine $K_{Ack}(u)$. Thus a compromised neighbor of u cannot leak $K_{Ack}(u)$. Each vertex has user/policy-defined functions to produce K_{Ann}. Also, a vertex may choose, according to local policy, different sets of K_{Ann} for each individual neighbor. For example, either $K_{Ann}(v,i) \neq K_{Ann}(v,j)$ or $K_{Ann}(v,i) = K_{Ann}(v,j)$, when $i \neq j$. We denote $K_{Ann}(v,0) = \cup_{i \in [1,\#(v)]} K_{Ann}(v,i)$.

For instance, a social relationship between Alice and Bob means that they are directly connected by a social link. They exchange their keywords for route discovery messages sent via this link. Suppose Alice's keyword is "security" and Bob's keyword is "network." To exchange route discovery

messages an RDM from Alice (orginated or forwarded) to Bob must include "network" and an RDM from Bob to Alice must include "security" in their respective K_{RDM} fields , otherwise the message will be dropped.

For confidentiality or integrity concerns, the message content, except the keywords, would be encrypted with the recipient's public key or signed with the senders private key respectively, which can be exchanged when they set up their DSL connection, i.e. establish their social relationship. The initial DSL connection setup is out of the scope of DSL, therefore we do not discuss key exchange or related details in this paper. We assume the availability of strong cryptography, which cannot be easily compromised.

4.3 Routability: Keyword Propagation

Keywords are also used to increase the routability of a node as a recipient. A DSL node exchanges keywords, which describe desired/expected messages for itself and possibly some of its neighbors, with its neighbors. Keyword exchange and propagation improve the visibility of each DSL node and each node is able to set its own policy to generate, exchange and propagate keywords such that it can control, to some degree, the routes for incoming messages.

For instance, Bob publishes some keywords, together with his public key if necessary, on his homepage for people without a direct DSL with him to discover message routes to him. Then another user, Carol, may use these keywords to discover a path, by using her own DSLs, to Bob within a DSL network and then sends her data messages to Bob. If Bob's keywords are popular, i.e. many nodes have social links that will match them, Carol's RDM will probably reach Bob. If Bob's keywords are not so well known by other nodes, however, Carol has to include other keywords in her RDM to increase her chances of finding a route to Bob.

With the current Internet and many of its applications, a sender is able to send a message to any other node as long as it has the recipient's identity, hence a compromised machine can be easily used as a source of spam. While in DSL, the requirement for including keywords to find a path between the sender and the recipient prevents a spammer from sending messages to other nodes without appropriate knowledge about the possible keyword and path combinations to reach the recipient. In other words, a sender has to choose the correct keywords and links to establish a routable path, which varies from different starting and ending nodes. The details of route discovery are shown in Section 5.1.

4.4 Scalability: Keyword Aggregation

The number of direct links for an individual node may become large, for example, recent analysis of social networks suggests that the average number of "friends" for a single node is around 150 [1]. Moreover, after propagating the keywords, each node will also receive keywords corresponding to nodes which are not directly connected to it. In order for our system to be scalable for large networks (say, several millions of nodes), keyword aggregation is introduced into DSL, which is an automated local operation completely at the discretion of the aggregating node. In other words, each node uses keyword aggregation primarily as a mechanism for limiting the number of keywords that it must maintain.

The set of incoming keywords for a vertex v from all its neighbors (including v itself) can be computed as follows.

$$K_{Inc}(v) = \bigcup_{i \in [0, \#(v)]} K_{Ann}(u_i, I_N(u_i, v)) \quad u_i = N(v, i)$$

Then we can define the aggregation function:

$$K_{Agg}(v) = \text{Aggregation}(K_{Inc}(v))$$

By default, $\text{Aggregation}(K_{Inc}(v)) = \varnothing$ for a node v, which means v's keywords are not aggregated. After getting a new aggregation function, v checks the current keywords in $K_{Ann}(v, i)$ and replaces the keywords which have been aggregated with their counter parts in $K_{Agg}(v)$, i.e. v updates its announced keywords according to its aggregation rules.

Aggregation functions or rules are decided by applications and out of the scope of this paper, while the aggregation process itself is an important building block in DSL. It is important to note that the aggregation function and associated update mechansim must be chosen carefully in order to limit negative effects on network reachability.

5. ADDRESSLESS ROUTING

In DSL, there is no global ID for any node. Instead, DSL provides routability and traceability with an addressless routing protocol.

5.1 Route Discovery

To set up a connection with another vertex t in G, s has to send an RDM, which consists of two sets of keywords: K_{Des} and K_{Int}. K_{Des} represents the keywords for the intended recipient, while K_{Int} is used purely for passing through the intermediate nodes. Let $K_{RDM}(s) = K_{Des} \cup K_{Int}$ denote the keywords in s's RDM. Besides $K_{RDM}(s)$, an RDM also includes an initially empty stack S_F to record the forwarding path of the RDM, for the future ACK delivery from t to s.

The route discovery process performs as follows at each node which receives the RDM.

1. *Acceptance*: A vertex v accepts an RDM from its ith neighbor *only if* $K_{RDM}(s) \supseteq K_{Ann}(v, i)$. v may simply drop the RDM and does not need to respond to its ith neighbor. [2]

2. *Loop Detection*: DSL considers both message ID and hop count to avoid/detect loops. Any RDM which fails loop detection will be dropped. We discuss the details for loop detection in Section 5.3.

For an RDM that passes loop detection, v has the following three options:

- Forward the RDM to a neighbor
- Acknowledge the RDM as a recipient
- Ignore the RDM by dropping it

v will first try both forwarding and acknowledgement, and if neither of them happens, v drops the RDM by default. v may forward and acknowledge the same RDM for there may exist multiple recipients.

[2] v does not need to respond to its ith neighbor even if v accepts the RDM.

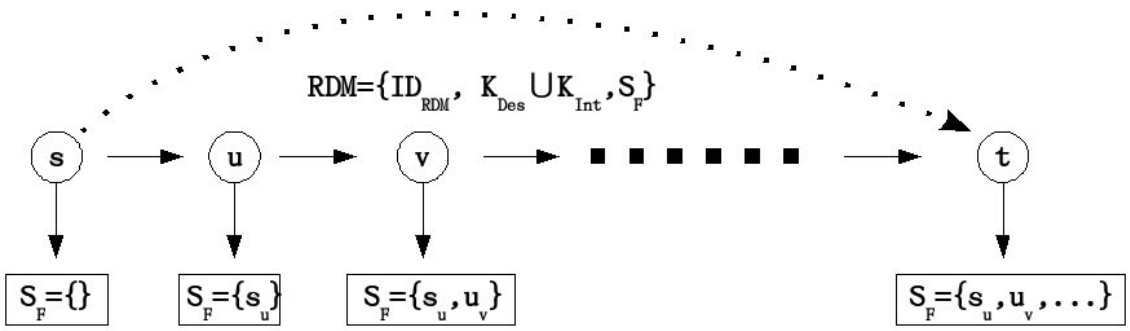

Figure 1: RDM Delivery

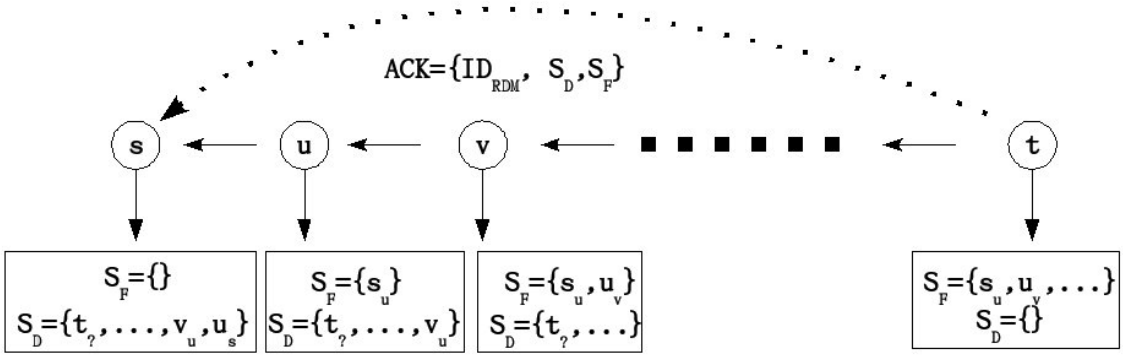

Figure 2: ACK Delivery

3. *Forwarding*: v forwards the RDM to its jth neighbor, u iff

$$K_{RDM}(s) \supseteq K_{Ann}(u_j, I_N(u_j, v))$$

where $u_j = N(v, j)$, $j \in [1, \#(v)]$ and $j \neq i$, where $i =$ the neighbor from which v received the RDM

Before forwarding the RDM, v pushes i into S_F.

4. *Acknowledgement*: v considers itself as the recipient t and acknowledges the RDM with an ACK message iff

$$|K_{Des}(s) \cap K_{Ack}(v)| > T_{Ack}(v)$$

and

$$|K_{Des}(s)| - |K_{Des}(s) \cap K_{Ack}(v)| < T_{Ign}(v)$$

where $K_{Ack}(v)$ is the set of keywords which v is interested in, $T_{Ack}(v)$ and $T_{Ign}(v)$ are locally defined thresholds by v for *acknowledgement* and *ignore* respectively. In our current DSL implementation, $K_{Ack}(v)$ is set to the original announced keywords by v.

An ACK message contains two stacks, S_F and S_D. S_F is the stack that arrives with the RDM being acknowledged. S_D is used to record the forwarding path for data transmission from s to t, initially empty. t sends the ACK to the neighbor which sent t the RDM. For each following node u, which receives the ACK,

- if $S_F \neq \varnothing$, u pops the top entry x from S_F, pushes the neighbor ID, from which u received the ACK, into S_D, and then forwards the updated ACK to $N(u, x)$;
- if $S_F = \varnothing$, the ACK delivery is completed, i.e. $u = s$. s pushes the neighbor ID, from which s received the ACK, into S_D.

With this route discovery protocol, aside from processed (forwarded or accepted) RDM ids, no state information is maintained in any intermediate node for each RDM. Instead, the routing information is carried by the RDM/ACK itself as it is forwarded through the network. Another benefit is that s only knows S_D but does not know S_F, i.e. only partial neighbor information of the intermediate nodes is revealed to s. We will discuss this privacy issue in Section 5.4.

Figure 1 and 2 illustrate the process of RDM and ACK respectively.

5.2 Data Transmission

After ACK delivery, the sender s has established a path between s and t. With S_D, the data transmission from s to t can be conducted in the following way. Starting at s, each intermediate node u along the established path pops the top entry in S_D, v, and then forwards the data message to its neighbor with link index $I_N(u, v)$. Any node receiving an RDM will obtain a path back to the originator of the RDM. To prevent malicious nodes from using these paths to transmit data, a digital signature can be used to ensure integrity and a time stamp can be included for freshness guarantee.

There are other possible options for data transmission after a path is established, which can be considered as extensions to the basic data transmission protocol.

- t may initiate a route discovery message to find an-

other path from t to s, which gives us *asymmetric data transmission*.

- t may include some other keywords in its data messages for s to find a better path.
- s and t may negotiate for a direct link between them, which can be effective for a limited time. For some applications which need real time communications, such as VoIP and streaming media, this option may be preferred.

Before transferring application data, DSL provides the option for the sender to do an *application test* to verify the recipient's identity. For example, with the same keyword, Alice as a sender may get multiple ACKs and she has to use the application test to decide which ACK message is from the recipient to whom she wants to send data. The application test can be done via the established path.

5.3 Loop Detection

For routing protocols, it is hard to detect loops without the use of global IDs. DSL tries to solve this problem by combining the following methods.

- *Duplicated RDMs*: To track the RDM/ACK correspondence, the sender has to assign a message identifier to each RDM it sends out. When a node v gets an RDM, its message ID is checked for duplicate detection, which can be implemented efficiently with a bloom filter. A duplicated RDM will be discarded or ignored.

  Bloom filters generate false positives, i.e. different RDMs may be determined as duplicates. Thus after an RDM is determined to be a duplicate, DSL may check K_{RDM} and S_F to reduce false positives.

  Due to its limited storage and computation cost, the bloom filter or local cache may also generate false negatives, i.e. an RDM that passes the bloom filter test may actually be a duplicate. Therefore after an RDM passes the bloom filter, DSL further checks its hop count and S_F fields.

- *Hop Count*: After a previously seen RDM is detected, v checks the number of nodes this RDM has passed. Since DSL records the forwarding information in S_F, v just needs to check the number of entries in S_F. The RDM is discarded if it exceeds the size limit of S_F or a given hop count threshold.

  A proper hop count threshold has to seek a balance between the timeliness of loop detection and the length of a legitimate path, both of which are decided by applications and the network topology. Because the path length distribution follows a power law in many real applications, the hop count threshold can not be too small. On the other hand, a large hop count threshold results in large performance overhead when many loops can be detected within a few hops.

- *Duplicated forwarding path*: Since each node inserts a neighbor ID into S_F, if a loop exists, a repeated pattern in S_F will be detected.

  The chance for two different nodes to add the same neighbor ID into S_F may be large when the number of neighbors is small. It is less possible, however, for two nodes that pass the keyword matching test to share the same neighbor ID for subsequent nodes. Furthermore, to increase the confidence, we may require a duplication of N successive IDs in S_F. The larger N is, the larger probability that a loop exists.

For loop detection which requires N successive IDs in S_F, distributed loop detection can be conducted as follows.

- Whenever a node v sees a duplicated neighbor ID in S_F, v puts a flag on the entry it pushes into S_F this time, i.e. the one on the top.
- Whenever a node v tries to put a flag, it checks the successive $N-1$ previous IDs in S_F. If they are all flagged, v decides there is a loop and discards the RDM.

In this way, all the nodes on an RDM path help detect loops.

5.4 Intermediate Node Privacy

The bit length for neighbor IDs is limited due to both the scalability concern and the number of neighbors a node can have. Therefore, when S_D is returned to the original RDM sender, although it contains the local neighbor IDs of intermediate nodes, it is still hard for malicious senders to explore the network topology because of the high collision probabilities among neighbor IDs.

An attack with a malicious sender and a malicious recipient together may get both S_F and S_D for the paths between them but requires more sophisticated techniques and more resources, including correct K_{RDM} and necessary social links to obtain additional path information through benign intermediate nodes.

6. A VERY PRELIMINARY EVALUATION

We have begun to develop a prototype implementation to evaluate DSL. At this early stage our goals are to observe overall connectivity of a network of DSL nodes and identify the extent of future challenges. To measure connectivity we build random social networks of varying size based on [8]. We start with this model as it closely represents the geometry of social networks with varying transitivity. After building a social network (no new nodes are added), each node is assigned a single keyword from a dictionary. At each time interval, all connected nodes within the social network exchange keywords. This action is called global propagation because all nodes propagate their keywords during the same time instant. Then, nodes with the lowest values for closeness centrality in the underlying social network attempt to discover routes to all other nodes within the overlay network (created by keyword propagation), using the process described in the route discovery Section 5.1. In our simple case the overlay network is equivalent to the social network but in general this may not be the case. We make the assumption that with uniform keyword propagation and breadth-first route discovery, if the least central nodes can discover routes to all other nodes then any more central node can also discover routes to all other nodes. Closeness centrality is a metric for the average distance from a source node to all other nodes as described in [13]. At each time instant we measure the number of nodes to which the node with the lowest closeness centrality has discovered routes. When this number equals V (the number of nodes in the network), the network is connected.

After a brief analysis of small networks containing thousands of nodes, we observe that under global propagation, the least central node discovers routes to all other nodes in five time steps on average. This observation is not an indicator of the speed of creating connected networks in DSL (particularly since global propagation is not likely in a distributed system) but a demonstration of the ability to form them and some of the inherent protection provided by DSL. By integrating keyword-based filtering into the social network, nodes indirectly connected by social links must rely on keyword propagation to explicitly construct trusted paths in order to route messages.

Also, the impact of clustering within the network on route discovery is unclear. In our tests, increasing the clustering coefficient of the social network sometimes decreases the number of newly discovered routes at each time step. This preliminary result may support the significance of loop avoidance/detection (since more clustering results in more loops) but more extensive evaluation is required.

Our simple experiments gave us the opportunity to observe other features of DSL. A key limiting factor of the protocol is scalability of keyword propagation. In our simple scenario, where each node is assigned a single keyword each node will maintain $O(V)$ keywords. In future tests we will examine more complex cases in order to formally evaluate the protocol's robustness to attack and the community structure formed by nodes with keywords in common.

7. RELATED WORK

In this section, we briefly review the related work in social network based email filtering and peer-to-peer routing.

7.1 Email Filtering

The trust relationship in social networks has been studied extensively [3] [10] [7] and applied to many applications [4] [9] [6] [5]. Among those applications, email filtering is most closely related to our work.

The main idea to apply social networks for email filtering is whitelisting [6] [7] [5] , i.e. allowing emails from the trusted users with a certain trust or reputation score.

TrustMail [6] [7] lets each user (as a recipient) assign reputation scores to other users (as senders) he/she knows directly and then compute the reputation scores for unknown senders by recursively looking for the opinions from users who know these senders. The final reputation score for an unknown sender is determined by the recipient based on the information collected along his/her social path to the sender. Different recipients may get different scores for the same sender, according to their locations in the reputation network.

RE [5] provides an extension to the current email clients and servers for propagating whitelists such that a recipient R can check whether a sender is on the whitelist of R's friends while at the same time keeps the privacy of each user's contact list. Although this *friend-of-friend* relationship can be further explored to longer social paths for a better coverage, the authors note that it increases the risk of false negatives (i.e. whitelisting untrusted senders) because the trust decreases along the long path.

7.2 Peer-to-Peer Routing

Numerous routing protocols have been proposed for peer-to-peer networks, such as Chord [16], CAN [14], Pastry [15], ROFL [2], and Tapestry [17].

CAN [14] uses hash table functionality to provide an Internet scale distributed infrastructure for locating resources within the network. CAN nodes own individual zones within a virtual coordinate space. A public uniform hash function maps the key of a key-value tuple to a point in the coordinate space. By storing the zones owned by local neighbors, CAN nodes use greedy forwarding to find straight line paths to destination coordinates. CANs provide scalable routing and indexing using a set of globally-known hash functions that do not allow control over resource location or every other node's ability to locate the resources of a particular node. Also, CAN construction relies on established infrastructure to create its overlay network. In DSL, we hope to build on the ideas of CAN by providing node-controlled keyword based routing.

A similar class of hash table based routing schemes are presented in [16], [15], and [17]. In each of these networks, the emphasis is on efficiency with some limitations on resource owner control and resilience. A DHT-based approach utilizing social links is presented in [11]. In these networks, social links are used primarily to prevent misrouting and as an optimizing feature. Although it relies on some hierarchical structure and DHTs, the idea of using semantic-free labels for routing as described in [2] is similar to DSL routing. One of our goals, is to eliminate the need for names as unique identifiers for routing while including application semantics or authentication information in the labels to ensure correct routing.

8. REMARKS

In this paper, we present Davis Social Links, a new communication architecture considering both network routing and social/trust relationships. We show that, by introducing a keyword-based route discovery mechanism, not only is it much harder to spam but also a message receiver has sufficient controllability to defend itself against traffic/messages that it does not want to receive. One very interesting design choice of DSL is to remove the need for any form of globally unique network-layer identifier. Identification of a peer under DSL is purely based on keyword discovery, social/trust relationship, and the application-layer semantics. We have shown that global connectivity can be accomplished without globally unique identifiers. Furthermore, in our preliminary study (not being discussed in any details in this paper), DSL provides *traceability* for both network and social/trust relationships, while the privacy of its users is reasonably protected.

We must confess that DSL is still very immature in many ways. Therefore, one important purpose of this paper is to report to the network security community regarding our initial ideas about the security of our communication infrastructure. We are certainly looking forward to feedback, criticism, and discussion leading to collaboration and further development.

9. ACKNOWLEDGEMENTS

The authors would like to thank Dimitri DeFigueiredo for the discussion of this project.

10. REFERENCES

[1] Adjie-Winoto, W., Schwartz, E., Balakrishnan, H., and Lilley, J. The design and implementation of an intentional naming system. In *Proceedings of 17th ACM Symposium on Operating Systems Principles (SOSP)* (1999), pp. 186–201.

[2] Caesar, M., Condie, T., Kannan, J., Lakshminarayanan, K., and Stoica, I. Rofl: routing on flat labels. In *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM)* (2006), pp. 363–374.

[3] Clifford, M. Networking in the solar trust model: Determining optimal trust paths in a decentralized trust network. In *Proceedings of the 18th Annual Computer Security Applications Conference(ACSAC)* (2002), pp. 271–281.

[4] Fiat, A., and Saia, J. Censorship resistant peer-to-peer content addressable networks. In *Proceedings of the 13th annual ACM-SIAM symposium on Discrete algorithms (SODA)* (2002), pp. 94–103.

[5] Garriss, S., Kaminsky, M., Freedman, M. J., Karp, B., Mazieres, D., and Yu, H. Re: Reliable email. In *Proceedings of the 3rd Symposium on Networked Systems Design and Implementation (NSDI)* (2006), pp. 297–310.

[6] Golbeck, J., and Hendler, J. Reputation network analysis for email filtering. In *Proceedings of the 1st Conference on Email and Anti-Spam (CEAS)* (2004).

[7] Golbeck, J., and Hendler, J. Inferring binary trust relationships in web-based social networks. *ACM Trans. Inter. Tech. 6*, 4 (2006), 497–529.

[8] Holme, P., and Kim, B. J. Growing scale-free networks with tunable clustering. *Physical Review E 65*, 2 (2002), 026107:1–4.

[9] Kangasharju, J., Ross, K. W., and Turner, D. A. Secure and resilient peer-to-peer e-mail: Design and implementation. In *Proceedings of the 3rd Conference on Peer-to-Peer Computing (P2P)* (2003), pp. 184–191.

[10] Li, J., Yarvis, M., and Reiher, P. Securing distributed adaptation. *Comput. Networks 38*, 3 (2002), 347–371.

[11] Marti, S., Ganesan, P., and Garcia-Molina, H. Sprout: P2p routing with social networks. In *Proceedings of the 1st International Workshop on Peer-to-Peer Computing and Databases* (2004), pp. 425–435.

[12] Milgram, S. The small world problem. *Psychology Today 1* (May 1967), 61–67.

[13] Newman, M. E. A measure of betweenness centrality based on random walks. *Social Networks 27*, 1 (Jan. 2005), 39–54.

[14] Ratnasamy, S., Francis, P., Handley, M., Karp, R., and Schenker, S. A scalable content-addressable network. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM)* (2001), pp. 161–172.

[15] Rowstron, A. I. T., and Druschel, P. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)* (2001), pp. 329–350.

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

[17] Zhao, B., Huang, L., Stribling, J., Rhea, S., Joseph, A., and Kubiatowicz, J. Tapestry: A resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications 22*, 1 (2004), 41–53.