Private Communication Through a Network of Trusted Connections: The Dark Freenet

Ian Clarke [*], Oskar Sandberg [**], Matthew Toseland [***], Vilhelm Verendel [†]

Abstract. Freenet is a distributed, Internet-wide peer-to-peer overlay network designed to allow anonymized and censorship resistant publication and distribution of information. The system functions as a distributed hashtable, where participating computers store information which can be retrieved by any other participant.

In this paper we describe a new architecture for the Freenet system, designed to strengthen the privacy of users, and to protect participating nodes from attack in situations where just running a node in the network could be a liability. While in previous incarnations of Freenet the edges of the overlay network were decided algorithmically in a manner intended to optimize routing in the network, in the new version we allow nodes to restrict their connections to nodes that they trust. In this type of network (sometimes called a Darknet, or a friend-to-friend network) every participant communicates directly only with his own trusted peers, and thus, under some assumptions, reveals his identity only to peers he already trusts. While everybody has to trust somebody in the network, there is no central party whom everybody must trust. We offer participants to either stay hidden in a darknet, or to take part of a hybrid model with direct connections also to strangers.

The novel algorithm we use for routing in the fixed mesh of trusted connections has been previously described in [29]. This paper focuses on the practical aspects of deploying a functioning data publication system based on the trusted connection model: we describe the system we have settled on and simulate it under realistic conditions.

1 Introduction

With the proliferation of Internet connectivity, the potential exists to increase the availability of information for everybody, while strengthening free speech and discourse. However, a blind belief that the Internet, in and of itself, cannot be controlled and will necessarily open the doors to free speech everywhere is unwise. In fact, as has been seen in many cases, it is possible to control and censor much of the Internet, and users of online services are considerably less anonymous and more exposed than most believe (see for instance [33]).

The Freenet Project is an ongoing effort to develop a distributed peer-to-peer network for publication and distribution of data. The system, Freenet [7][8], is an Internet wide distributed hashtable (DHT) intended to strengthen the freedom of speech by allowing anyone to publish anything in the network at any time. To this end, the system integrates many levels of redundancy while dispersing data throughout the Internet's topology, which makes it difficult for an adversary to remove data from the network. The network strives to be opaque, so that it is difficult without global knowledge to see which nodes are responsible for storing

[*] FreenetProject Inc. ian@freenetproject.org
[**] The Department of Mathematical Sciences, Chalmers University of Technology and Göteborg University. ossa@math.chalmers.se
[***] FreenetProject Inc. matthew@freenetproject.org
[†] Chalmers University of Technology. vive@chalmers.se

which data at which time, and flexible, so that it can survive both the everyday churn of an uncontrolled peer-to-peer environment and malicious attacks.

An important element of allowing free communication is that the privacy of both those who publish and those who access information should be protected, both from outsiders and from other participants. Earlier generations of Freenet anonymized only via a Crowds [27] type system, which, while offering some disassociation between queries and those who initiate them, is vulnerable to many forms of attack. One of the outstanding questions of Freenet development has been how the network can better anonymize queries in the future, with the use of stronger mixing techniques such as [6] and [16] considered a possible, but complicated, option.

A second concern, that has come to the forefront with the actual deployment of Freenet, is the vulnerability of people operating nodes in the network. While the network strove to dissociate users from the data they accessed and nodes from the data they served, it did not hide that a particular node was part of the network. In order to find Freenet nodes in earlier incarnations of the system, it was sufficient to join the network and start operating as a node oneself: through the continuous process of routing and optimizing the network, one would eventually learn the identities and Internet addresses of more and more nodes in the network. This means that somebody wishing to attack the Freenet network in its entirety would have had no problem finding and identifying participants given sufficient time.

In order to resolve these problems, we have radically shifted the direction of Freenet development. Instead of direct connections between nodes forming dynamically in the course of the networks operation, we now also allow nodes to limit their connections only to other nodes whom they trust. This model of trusted connections radically improves the security of the network, under the assumption that trusted neighbors really are trustworthy, but complicates other issues. Notably, since the we cannot design the overlay graph of connected nodes ourselves, routing from one point to another becomes more difficult. There is also a cost to usability, especially with regard to barrier of entry into the network, since with trusted connections, a new user must know and be authenticated by at least one existing user before they can join, unless they want to reveal their identities to strangers wishing to do the same (like previous versions of Freenet, not necessary but still supported).

We believe, however, that the benefits of the trusted connections model outweigh the complications it implies for nodes seeking improved privacy. We hold that in many cases trusted connections are a better model for protecting the identities of those publishing and accessing data than traditional mixing approaches. They also dramatically increase the difficulty of identifying and attacking participants in the network of trusted connections, which must be done by subjugating one node at a time, and then proceeding to attack its neighbors[1].

1.1 Previous Work

As mentioned, earlier versions of the Freenet system were previously described in [7], [8], as well as [17]. The algorithm which we use to route in the network was described and simulated in [29]. The algorithm draws off the small-world model of Jon Kleinberg [19], which has also inspired previous attempts to route in social networks [1] [22].

In some respects and goals, but not in others, Freenet resembles other designs for distributed secure data storage such as the Eternity service [2] and the Free Haven Project [10]. In

[1] Assuming of course that nodes cannot be identified in another way, such as monitoring the network for specific traffic patterns.

operation, Freenet also shares a lot with common distributed hashtable (DHT) designs, such as [26] [32] [23]. Oceanstore [20], is a fully developed system for the distributed storage of information based on a DHT, but it is not designed with uncensorability or user privacy in mind.

The term “Darknet”, which we have used to describe the trusted connections model on which the new version of Freenet is based, was coined in [3] as a more general term for private, concealed overlay networks. This is also described in [21]. An alternative name for the same thing, “Friend-to-Friend network” was coined by Dan Bricklin in [5]. In a similar spirit, we use the term “Opennet” for Freenet nodes that decide to take part of a hybrid model - allowing connections also to strangers. These nodes use the algorithm in [30] to dynamically optimize links in the overlay while routing in the network, and a bootstrapping mechanism to announce their presence to other nodes that want to peer with strangers.

Despite the idea having been around for some time, few trusted-connection based networks have actually been deployed. A program called *Aimster* (and later *Madster*) was released in 2000 allowing file-sharing between people who were on each others AOL Instant Messenger “buddy lists”, but the company that made it is no longer in operation. *WASTE* is a program released, and then withdrawn, by AOL division Nullsoft in 2003, that is somewhat similar, see [34] for a description. A program known as *OneSwarm* was released in 2009, developed with file-sharing in mind [18]. These systems do not attempt to allow censorship resistant communication between any two participants.

1.2 Contribution

We summarize the contribution of this paper as follows:

- We describe a model for a practicly deployable data storage network where communication is limited to trust peers, but data is made available globally to all.
- We perform simulations to evaluate this model using real social-network data (from the PGP Web-of-Trust) to provide the trusted connections, and in environments with the type of churn which can be expected from true Internet deployments.
- We simulate a hybrid model where some nodes in the network may choose to reveal their identities to strangers to optimize performance. Nodes operating in this manner are said to participate in “Opennet”. We study whether these nodes may co-exist in the same network.

While the method we use for routing has been previously described [29] in a highly idealized setting of a static of graph with points set in an exact grid, using it as a practical routing function in a realistic environment has not previously been explored.

While we are far from having addressed all potential issues and functionality in the system, the goal of this paper is to present an overview and a working implementation of our new model in practice.

2 Freenet’s Design and Operation

Freenet’s goal is to provide a distributed system for the publication and access of information. Documents are stored, or inserted, into the network with an associated address, or key, which consists of a binary string. The key is used both to decide where in the network to store the document, and to authenticate the document when it is transfered. A person wishing to access the document sends out a retrieve query, or request, which performs a routed depth-first search with backtracking. When the request finds a node containing a copy of the document, the entire document is returned through the search path.

Freenet nodes do not make guarantees about the persistence of documents, but keep them using a best-effort algorithm that removes documents using randomness when a node's storage overflows. In fact, nodes aggressively cache many if not all the documents that they transfer, so a system for removing documents from the cache is necessary. The network can be viewed as a large grid of caching proxy hosts, each proxying and caching for one another, but with no source to return to should the document not be in the cache. In practice, we have found that the system is relatively good at maintaining data when a sufficient amount of storage is available (see for instance [17], and [38] for a modification to improve this). Furthermore, Freenet does not guarantee that data in the network can be found by any particular search in a bounded number of steps (or indeed at all) - it only attempts to make it likely that it will.

2.1 Inserting and Requesting data

We assume that to every piece of data, D, there is an associated key K. Each node has a routing table, which with respect to the key K, gives an ordering of nodes neighbors after their desirability as the destination of the query. The route is then constructed by going from node to node and selecting the most desirable neighbor that is available and not already in the route as the next step. When this is not possible, the query backtracks and the route is restarted from the previous node. The routing process is terminated either due to the query achieving its purpose or because of some heuristic based on the number of steps taken. In current implementations, the reason for terminating is either that the sought document has been found, or that nodes have given up looking for it.

It should be noted that, unlike most other DHT systems, a key has no canonical home or destination in the network. While there may be a node that is the most desirable destination for a particular K, that node will most likely not know that it is. The manner in which the routing tables are constructed need only be consistent, in that two routes for the same key should be likely to intersect, and well distributed, meaning that routes should not all tend toward some small subset of the nodes. See Figure 1.

To search for a document in the network, matching a key K, one establishes a route for K. At each step, the node checks its cache to see if a document associated with that key is present. If such a document is found, the route terminates and the document is returned to the previous node in the route, which proxies it back towards the node which initiated the request. If it is not found locally, a node also has a quick method to search the cache of its neighbors for the document. The latter is possible by e.g. the sharing of Bloom filters [4] between neighbors, see Section 2.4.

The inverse process, that of originally inserting a document, is similar. A route is established, and if it terminates without finding a cached document matching K, the document D is proxied up the route. If an existing document matching K is found while routing, it is returned down the path just like if K had been requested.

During the proxying of data in both the insert and request steps, nodes will store the data in their cache. Each node has two seperate caches: one for short term storage meant to satisfy immediate demand for popular documents, and one meant to hold documents "belonging"

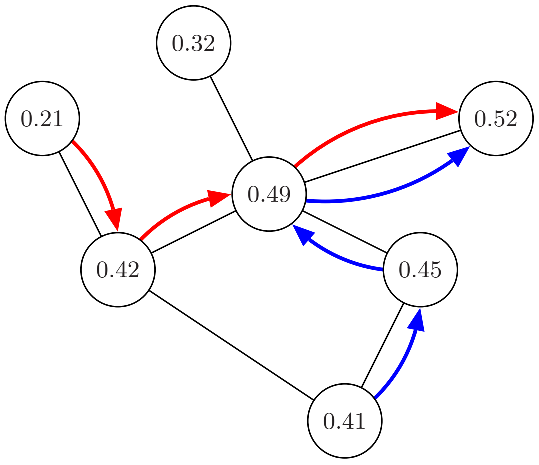

Fig. 1. The routing algorithm does not need to route a query for a particular key to a specific place. It only needs to route queries for the same key, but started at different nodes, to the same place. In picture, the red and blue queries are both for the key $K = 0.50$, but start that different points.

to the node for longer periods of time. All documents that a node transfers are stored in the short term cache, and expunged using a random scheme. Which nodes place a document in the longer term store is described below.

2.2 Keys and Document Authentication

In order to maintain data integrity, keys are related to the documents they denote by cryptographic means. In particular Freenet uses keys that either a cryptographic hash of the document itself, or the bitwise XOR of a hash of a document name and the fingerprint of a public asymmetric cryptography key (PK) which the document is signed.

While proxying and caching a document, each node can thus verify its integrity. In the first case, the node need only calculate the hash of the document, and make sure it matches the key it routed for. In the second case, the PK and name string are provided as metadata with the document. When the node receives the document, it checks that the name and PK hash to the key that it routed for, and then checks that PK verifies the signature on the data.

The first type of key, called a Content Hash Key (CHK), is used for large static pieces of data. Since the key is generated from the data, it can neither be generated beforehand nor predicted by somebody who does not have the data. However, such keys are still useful, for example, when a document wishes to provide a link to a copy of a particular static document.

The second type of key, known as a Signed Subspace Key (SSK), allows several documents to be published by the holder of a single asymmetric key-pair. As the name implies, the holder of the key-pair has a private subspace in which he can publish signed documents, by varying the name part. This is used to implement updating schemes - either by including a version in the name, or having the name include a date of publication.

In both cases, the entire key cannot be derived from a readable address (in the CHK case, the entire key is an unreadable binary string, and in the SSK case at least the subspace wide part is). Thus finding documents in Freenet requires finding at least the publishers PK in some way.[2]No method for direct keyword searching is provided, though some methods are possible[3].

2.3 Data formats and encryption

An important aspect of Freenet, that has partially been carried over from the previous versions to the current, is a complete system for encrypting, splitting, presenting data. This occurs at a higher level than the Freenet nodes, in what has been designated the client (though in the spirit of peer-to-peer, the node and client are often the same program), and is thus unaffected by the recent changes in network architecture. The Freenet nodes which route for, store, and fetch data only ever see atomic, encrypted, documents, which are processed by the client into plaintext reconstructed versions.

Encryption of the data is typically done using symmetric encryption, the key being part of the address (in the case of the CHK), or an encrypted key provided in the headers, to be decrypted using a subspace wide key provided in the address (in the case of an SSK). Either way, the key itself is normally derived from the data's original hash.

The client also handles splitting and padding of data. As a basic measure against profiling and traffic analysis, documents in Freenet are a fixed size - 32kB for CHKs, and 1kB for SSKs. Clients ensure this by padding and by splitting larger documents into many smaller parts. The latter process is also an enhancement to performance and reliability. Because the parts are inserted under different keys, they are spread throughout the network which allows for effective multi-source downloading (in the style of newer peer-to-peer programs such as bittorrent) when downloading. Splitting also provides an additional layer of redundancy, since the clients can use erasure codes to make sure that not all the parts are necessary to recreate the document. Freenet currently makes use of Vandermonde Forward Error Correction codes [28].

2.4 Routing in the Network

Under the trusted connection model, we have to assume that the graph of nodes and links between them is fixed, and cannot be optimized for routing purposes. This is a very different model from what previous version of Freenet, and other DHT systems have operated under: they all construct a graph explicitly so as to make efficient routing possible.

In fact, the possible efficiency of routing depends on the structure of the fixed network. If there is no short path between two nodes, it will not be possible for a document inserted at one to be reachable at another in a short number of steps, regardless of how well we route. We must thus rely on certain observations about the network of trusted connections. In particular, it is a subgraph of the of the world's social network, the dynamics which has been the subject

[3] A system for expressing keys as URIs have been devised for Freenet. The non-readable sections are expressed as strings in a modified BASE64 encoding, and the readable bytes (such as the document name in the SSK) in plaintext.

[3] One method that has been used is to create the public key/private key pair for an SSK using a string as a seed to a known PRNG. Then anybody who knows or guesses the string can derive the public key necessary for querying for the data in the network. The insecurity implied by this method is obvious however: they can also derive the private key needed to insert other data under the same name.

of much study, initially by sociologists and experimental psychologists [25], and increasingly by mathematical means [9] [37]. It is generally accepted that the worlds social network is a so called “small world”, meaning that there are short paths between every two nodes, and that there is local clustering (people who share a common friend are more likely to know each other).

Studies of routing in the world’s social network have already been done, starting in the 1960s with Stanley Migram’s famous small-world experiments [25] [36]. These, and subsequent repetitions [12] seem to indicate that it is possible to find routes from any person to any other, using only a very small number of steps (the idea of “six degrees of separation” for the entire world is derived from Milgram’s data.)

We assume (optimistically) that the network of trusted connections that we can use is a small world. In order to route on it, we firstly must give each node an identity so that they can be routed to. The most direct way of assigning these identities is to choose them from the same space as the keys, and then try to route queries for a particular key to those nodes whose identities most closely match the key value. This technique is used by almost all current DHT systems. In light of the observation described in Figure 1 the identities of each node can be selected quite randomly, though their assignment will need to reflect the network in some respect.

Once each node has an identity, one could imagine calculating the all pairs shortest path for the graph, and keeping a routing table at each node showing where to reach queries intended to ultimately reach every node. This would ensure that the best possible route was always used, but involve a large amount of work and bandwidth when trying to calculate the paths, and require network sized routing tables at each node. More crucially, it would require one to recalculate the entire routing table every time a node joins or leaves the network, which is an impossibility under the intended usage. Other previously known systems schemes for building routing tables with guarantees on the stretch[4] such as [35] suffer from the same problems under our application.

Instead, we use a method which draws off the small world models of Jon Kleinberg [19]. The routing we perform is purely greedy: at each step, the desirability of the neighbors is ordered by the proximity of their identities to the route key K (seen as floating point number between 0 and 1 with periodic boundary[5]). The question then becomes one of trying affect the randomized assignment of identities such that this becomes an efficient way of routing. For routing to be efficient Kleinberg’s results show that certain relation between the frequency of connections of different lengths (with respect to the identities) must be present, so our goal is to, to the degree that it is possible, assign identities so that this holds. The method we have chosen for doing this is a development of that described in [29].

In our implementation nodes start upon joining the network with randomly selected identities seen as numbers between 0 and 1. These identities are then switched between the nodes using the simulated annealing like method first explored in [29], which causes nodes that are in some sense close in the network topology to also have nearby identities. (It is only when this property holds that greedy routing makes sense in anything but the final step.) The use of random positions, rather than fixed points in a grid, makes our model slightly different than the Kleinberg model that is the basis of previous analysis of this method, but similar continuum models have been previously studied in [15] and [14]. Nodes at a constant rate initiate random walks, which terminate after a fixed number of steps (current six, which simulation indicates is enough even in a large network). When the walk terminates, the node

[4] The stretch of a routing scheme is the amount by which the lengths routes found differs from the optimal possible value.

at which it was started and that at which it ended attempt to switch identities, which will happen with a probability specified by the algorithm.

Ideally, one would hope to be able to assign the identities so that every step in a route for K brings us to a node whose identity is closer to K than the last, until we have found the best node on the network. In practice, this may not always be possible, but we still use this as heuristic to show us when to terminate a route. Currently the route continues until it has reached a maximal Hops-To-Live (HTL), which is motivated by attempting to balance a thorough search while limiting resource usage.

Several enhancements to the basic algorithm described in [29] which could lead to better results have been suggested. Knowing the identity of ones neighbors' neighbors is known to improve the performance of routing in Kleinberg type networks [24] by allowing nodes to route to the neighbor whose neighbors identities best match the query. Knowing the identities of neighbor's neighbors reveals something about the surrounding network, but does not tell one who they are, so the basic principle of only revealing oneself to trusted peers remains. Another performance enhancement that we use is for nodes to be aware of which documents are present in their neighbor's cache, by for instance neighbors passing Bloom filters [4] summarizing the contents of their cache to each other. A combination of both these techniques can greatly increase the number of successful searches and decrease the query length.

One artifact of the fact that nodes are constantly (though, when the network is steady, not very frequently) changing identities is that such a switch invalidates much of their cache. A node that has just gained a new identity will not have a cache full of documents closely matching this, since it has seen few if any queries for that identity before. Some method from migrating data between nodes after an identity switch may be necessary, but in the current implementation we rely only redundancy and the re-caching of data to ensure this. Simulation results (see below) seem to indicate that this is enough up to a certain level of churn in the node population.

2.5 Storing of Data

As noted above, each node keeps two seperate caches of data. One is a short term cache where all data that the node transfers is stored temporarily until it pushed out by other data. The other is meant to keep data longer by storing only inserted data that matches the nodes identity.

The strategy used to populate this second cache, which we have dubbed *sinkstore*, is this:

A node will attempt to place the document corresponding to a key K in its store if it is better located at the node, given its identity, than to that of any of its neighbors. For example, in the configuration shown in Figure 1, the node with identity 0.49 is a *sink* for $K = 0.50$: its current identity is closer to that value than any of its neighbors identities - if the query was an insert, the node with current identity 0.49 would store it.

In an ideal network, each query would encounter exactly one sink: the unique node with identity closest to K. In reality, we have not found the configurations to work this well, and data tends to get stored at several places. In the implementation, we have heuristics to address the fact that routing may not reach overloaded nodes who have the best location - or such nodes that have low uptime. A document is stored unless a nodes predecessor or any of its successor(s) is closer to K than the node, and has average uptime over 40%. In a

[5] That is circular distance, so the distance between 0.9 to 0.0 is 0.1.

sense, sinkstore adapts to this by using greater redundancy when routing is less optimal. See the simulation section for the impact of redundancy when using networks derived from real trust data. Finally, only new data is placed in the store; requests do not modify it because for popular data they may complete very quickly before reaching nodes close to the "ideal node", whereas inserts always continue until they run out of HTL. In practice documents will be stored on a few nodes out of the whole chain.

Like the short term cache, the longer term storage is cleared using a random removal scheme.

2.6 Joining and Leaving the Network

The procedure for nodes when joining and leaving the network is very straightforward. When joining, nodes choose an identity at random, and then connect to those peers with whom they have a pre-established trusted relationship. To protect against man in the middle attacks, the connections are initiated by the exchange of certificates between users - the certificates include both contact information (in practice, the nodes current IP address and port number) and cryptographic identifiers for the nodes.

When a user imports a certificate his node will adds the certificate's node as a trusted peer, and attempts to contact it. The other node will only respond if it has previously the certificate of the node attempting to contact it. On failure to connect, nodes try again, letting the time between subsequent attempts increase exponentially.

Once a new node has established its connections to a set of trusted peers, it can start taking part in the routing. It starts taking part in the switching procedure described above so as to get an identity which suits its position in the network and those of its neighbors. Nodes may initially switch between identities at a higher rate, so that they can become useful to the routing as soon as possible.

In practice, it is of course unrealistic to expect nodes to join the network fully formed, with a complete set of trusted peers. One of the most important and difficult aspects of the project, from a usability perspective, is to make it sufficiently easy and profitable for users to create connections that the overlay graph really does reflect the underlying social network. If, for instance, new users only ever connect to one person when joining the network, the graph will become a tree, and the routing algorithm will fail completely. In the current, early, versions of the software, we are depending only on the promise of increased performance to encourage people to authenticate and connect to as many trusted peers as possible, but in the future other ways of encouraging may be necessary (for example allowing the software to be used as an instant messaging client and/or file-sharing client with those one has authenticated).

To make it possible for nodes with few or no connections to connect, they can do so under a weaker threat model for the nodes themselves. This is done by using the Opennet model, where other users provide a small number of publicly known "seednodes" that are already connected to the network. When Opennet nodes connect, the seednode "announces" to the part of the network which is close (in the identity space) to the position that the new node wants to use. This is done like a normal route through the network, allowing the joining node to get replies from other opennet nodes who are interested in more connections. When having gained connections, nodes start operating as normal, as well as using the algorithm detailed in [30] dynamically arrange its connections to others who want to use the Opennet.

Leaving the network requires no special action. Nodes operate under the assumption that neighbors may cease operating at any time, and simply send routes to the next most desirable neighbor when the most desirable one is no longer responding. Attempts to contact a node that is not responding tailor off exponentially in frequency.

3 Simulations

Since the scalability of the routing algoritm has been discussed elsewhere [29], we will concentrate on looking at the effects of various factors in networks of fixed sizes. In particular, the amount of redundancy needed to get good results, as well as the amount of churn in the node population that the network can withstand. Furthermore, we study the scenario where a fraction of nodes take part in a "hybrid network" by Darknet and Opennet nodes, with some nodes announcing to strangers they are part of the network as others keep the privacy of the Darknet.

3.1 Simulation Dataset

To evaluate the our model we perform simulations with real-world social network data. Because of privacy protection we cannot reliably extract the full topology from the real Freenet network. Instead, we run simulations using data from the PGP Web-of-Trust (WoT). This data is publicly available[6], and in the following we only consider mutual signatures made by cryptographic keys, which we assume reflects trust comparable to that between people in a darknet. As of 2009-02-18, the PGP WoT data set contains 41315 persons, with the largest connected component being of size 33133. We further restrict nodes to the requirement of having a degree of minimum 20 (to be in the Darknet), or 10 when we study how nodes get extra links with the Opennet functionality. Thus, we use a connected core from the PGP network to fulfill these criteria. Using the k-core algorithm [31], we extract a set of size 1580 (for $k = 20$) and of size 3791 (for $k = 10$). These are the fixed underlying networks for the following experiments.

3.2 Simulation Background

The simulation is driven by events where people in the social network let their nodes join, leave, request and insert data, and switch numerical identities. These events occur with randomness of different intensity. The simulation is thus meant to describe an overlay network whose structure varies depending on which nodes are online at a time, but where the Darknet links of trust are constrained and picked from the underlying network between people who trust each other. We have chosen simulation intensitities of events (in the resolution of microseconds, due to the large number of events in a network) from studying a small set of typical nodes in the real network. We also vary parameters such as redundancy between different simulations, to study performance under various hypothetical settings. When a person is online, she starts to operate a node which is either new or has a number of stored documents from previous operation. A node of a person connects to the persons friends that are currently online. Nodes that are offline are unavailable to routing but with previous documents remaining in cache, that can later become available again. Nodes are homogenous with respect to the total size of the cache, and parameters guiding nodes stochastic behavior are constant for all nodes over one simulation. When nodes leave they may with a small probability do so permanently, which means wiping out all documents in their cache as well as later joining like a new node in the network.

Furthermore, a fraction of nodes (which we vary) are less worried about being visible in the network and choose to participate in the Opennet - which allows dynamic optimization

[6] See `http://www.lysator.liu.se/~jc/wotsap`

of links in the overlay whenever the person running a node wants to. These nodes do also take part in switching locations, and forward such requests to allow "pure" Darknet nodes improved routing. The Opennet nodes are offered a set of other Opennet nodes when joining the network, by the announcement mechanism described above. This allows us to examine whether the Darknet model can co-exist with a more traditional mechanism to adaptively design a peer-to-peer overlay.

The simulator is written in Java and publicly available at `http://freenet.googlecode.com/svn/trunk/apps/simsalabim/`

3.3 Simulation Details and Objectives

We now describe the objectives of simulations depending on different parameters that affect the operation of the simulation. Some important parameters are examined in the following, but there are many more that are used to simulate a node in the network (in the implementation of the simulator). First, the intensities of the different events rule the stochastic behavior of nodes, by sampling the time (in microseconds) from an exponentially distributed joint variable to the next event that occurs in the network. The nodes that request, insert, join, leave and switch are picked uniformly at random from the online nodes (or from the offline nodes, for the joins) without any order. The simulation thus completes by completing a large number of events, and outcomes are collected by statistics in the program.

We have let 20% of the nodes to be persistent in the network by not performing joins or leaves, in line with different but similar observations and estimates. Furthermore, each node has uniform storage capacity to hold 2000 documents (1000 for the store and 1000 for the cache), that get overwritten according to a random-based scheme. In practice the number of documents limit the time period for which the availability of data can be simulated, but in theory there is no upper bound on time and effort to study the network over longer runs.

We now report the following simulations. The first and second simulations study the feasibility of a pure Darknet. The third simulation study the impact on performance where a fraction of nodes use the Opennet to connect to get friends. The networks were typically simulated for a week (time in the simulation, with a resolution of a microsecond). This time is a tradeoff between the available computational resources and the interest to observe stationary behavior over longer runs. We have found that all simulations to give similar results for obvious modifications of duration.

- First, we study how the amount of redundancy (total size of storage, compared to size required by available documents) affects the availability and the load on the network. This is done by simulating a full Darknet network for a week, and then evaluating the success rate and average number of steps for requests over one simulated day. See Figures 2 and 3.
- Second, we study how the network tolerates the amount of churn of individual nodes that are removed from the network and permanently lose their data. This is done by simulating the network for a week, then studying performance for one day. We vary the intensity at which a node leaves and joins the network, as well as the probability to permanently leave. When a node leaves permanently, all its documents are lost and it can later reconnect but as "fresh" node without any data. Here, we want to study at which rate the background process of nodes requesting and caching the data can compensate for varying rate at which nodes leave the network with data. See Figures 4 and 5.

– Finally, we study opennet functionality by letting some nodes get fixed connections to friends and dynamically optimized connections to strangers as these choose to seek peers when routing in the network [30]. Here, nodes may connect to a small number of seednodes when joining the network. The nodes that use the Opennet functionality are picked as a random fraction of the population, and nodes use a random seednode if they needs help with getting connections in the network.

  To study the effect of how opennet may help nodes and impact the network, we use a larger network with nodes having a degree of at least 10 (as previously, with the k-core algorithm). Nodes with less that 20 connections can use the opennet functionality to obtain the total number of 20 connections (including Darknet connections). Such nodes prefer Darknet connections over Opennet connections, which means the need of Opennet peers varies depending on how many of its Darknet peers are online. Nodes drop Opennet peers if more Darknet peers become available during the simulation, to make as limited use of the Opennet as possible. Performance is measured as in the first experiment. See Figures 6 and 7.

3.4 Simulation Results

Taking a step back, the simulations indicate a number of things under our assumptions.

In the first experiment, the performance of the network depends on the popularity distribution of the documents that are requested - at least when the load of storage in the network grows proportionally large. Since it is common that some documents are more popular than others, this likely translates into a bias for such documents when load gets large. It is currently an open question whether the nodes in the network should compensate for this, e.g. by attempting to re-insert less popular data. Where some documents are more popular, average performance is at least as good as in the uniform case, and likely better.

In the second experiment, the simulations show a quite strong resistance to churn of nodes permanently leaving the network with data. This can perhaps be explained by that the intensity of requests dominate the join-leave process by orders of magnitude. Thus, once a document is stored well in the network, the continuing requests for it from some part of the network will often compensate for the loss of a particular copy. It is really for a very mobile network, with nodes having volatile caches, that the leave-churn seems to be a problem that removes many of the keys. The remaining keys that were requested were also shown to have good availability.

In the final experiment, we find that the network can work well as a hybrid - with Darknet and Opennet nodes acting concurrently under different privacy requirements. By studying the performance, there is no indication that the different dynamics of switching Darknet locations and dynamically optimizing Opennet connections will hurt the load or availability on the network.

There are clearly a very large number of simulations that could be made of the networks we are considering. These results are necessarily limited by the available computational power available to the project, but we strive for simulations to be an on-going effort in network design and development. While the results are limited to making assumptions about parameters that are hard to fully measure in the actual network, this can partially be compensated by varying a number of parameters in simulations.

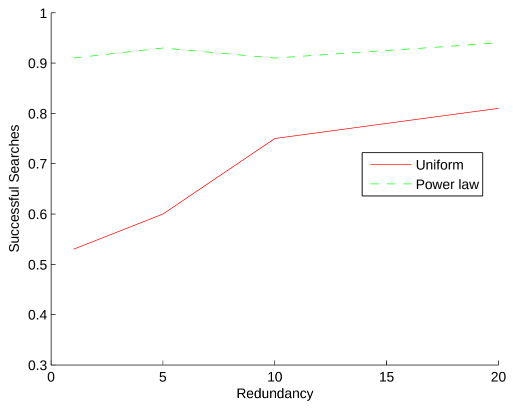

Fig. 2. Successrate as a function of redundancy (total network storage / number of documents inserted) in the PGP graph. In one case with uniform popularity, in another with document popularity following a power-law with exponent one. 1580 nodes, 1 week.

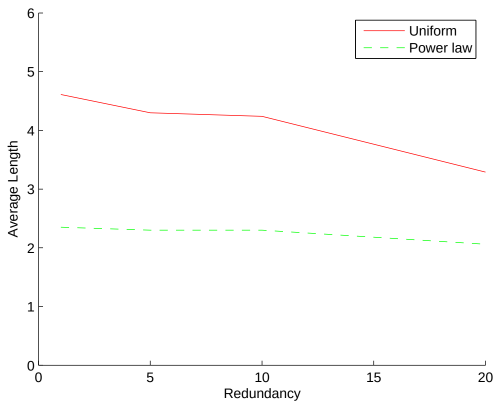

Fig. 3. Average request length as a function of redundancy (total network storage / number of documents inserted) in the PGP graph. In one case with uniform popularity, in another with document popularity following a power-law with exponent one. 1580 nodes, 1 week.

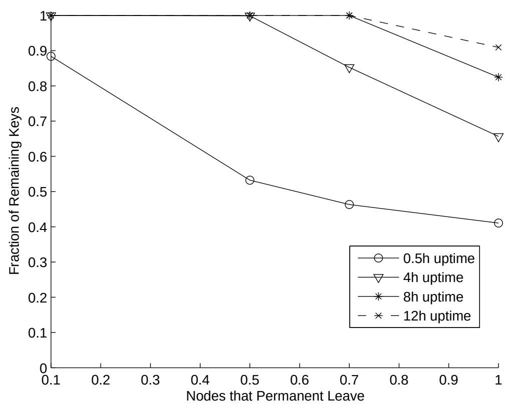

Fig. 4. Remaining documents as a fraction of leave churn (the fraction of nodes that decide to leave the network, or lose their documents between connects). With different rates of average up/downtime. 20% of nodes persistent in network. 1580 nodes, redundancy=10, 1 week.

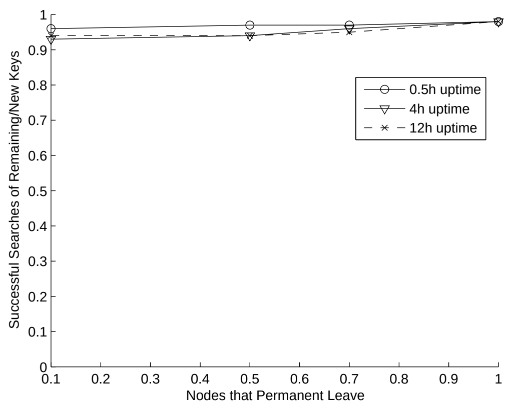

Fig. 5. Availability of remaining documents as a fraction of leave churn (the fraction of nodes that decide to leave the network, or lose their documents between connects). With different rates of average up/downtime. 20% of nodes persistent in network, with document popularity following a power-law with exponent one. 1580 nodes, redundancy=10, 1 week.

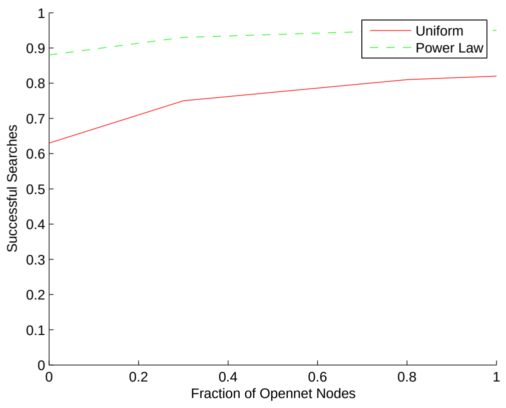

Fig. 6. Successrate as a function of nodes using opennet. 10 opennet seednodes exist among the persistent nodes in the network. 20% of nodes persistent in network, with document popularity following a power-law with exponent one. 3791 nodes, redundancy=10, 1 week.

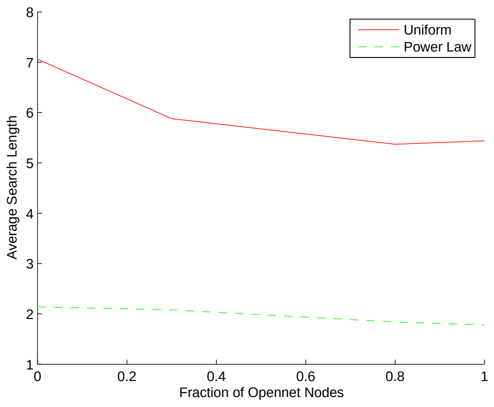

Fig. 7. Average request length as a function of nodes using opennet. 10 opennet seednodes exist among the persistent nodes in the network. 20% of nodes persistent in network. 3791 nodes, redundancy=10, 1 week.

4 Implementation

As part of the Freenet Project[7] we have an early implementation of the new version of the Freenet. The current implementation is in Java, using a UDP/IP messaging layer originally similar to that originally used for the *Dijjer*[8] download accelerator. Other code has been carried over from the previous versions of Freenet. This includes the higher level code for processing data, as well as some of the security and authentication code.

The software is functional, but not yet ready for mainstream deployment. The user interface is still primitive, especially when it comes to the creation and authentication of the trusted connections, which, as noted above, are crucial to the network becoming something we can hope to route in. Test networks have been deployed and seem to function - an example of such a network is shown in Figure 8 - but they are too small and used by too limited a class of people to provide any real data on what the network will look like once truly deployed.

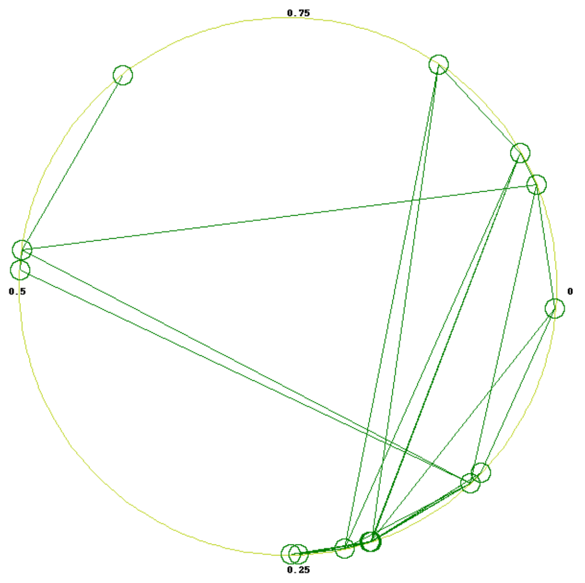

Fig. 8. A (small) realization of the actual Freenet network.

The fact that the technical functionality of the new Freenet depends essentially on the usage patterns and the dynamics of social networks that we can do little to control is an almost unique challenge for software development. Nobody can know for sure that the implementation will function until we have reached a large number of users and continue to see positive results.

Likewise, the goal of building a dark network, which by design is to be opaque, makes a deployed system very difficult to analyze. Indeed, if the system works as well as we have hoped, it will be difficult to even estimate the size of a deployed network, let alone say anything about its graph topology. The current early code provides a minimal amount of data to developers about the network, that could also be obtained using other means, but we cannot expect to continue doing that when Freenet is being used for real purposes (it

[8] `http://www.freenetproject.org/`

[8] `http://dijjer.org/`

would be counter to our goals even if just a portion of our users revealed their neighborhoods publicly).

5 Privacy and Anonymity

It is perhaps a matter of definition how much privacy the trusted connections model provides compared to layered or telescopic encryption schemes such as those used in Mixnets [6] and in Onion Routing [16] [11]. While these schemes generally attempt to ensure that no party in the network can alone determine what the user is accessing, we allow our neighbors in the network to see what we are accessing, but assume that they are trusted friends. In this sense the anonymity provided is simply the classical method of going through a trusted third party, but with the difference that trust is distributed throughout the network, and so single party needs to be trusted by everyone.

The type of privacy required also depends on the application. A user who wants to access information about, for example, a venereal disease, might prefer simply hiding among the masses on the current plaintext World Wide Web than routing his query through people who know him. On the other hand, people wishing to share political information that is being repressed by their government may have much to gain from the trusted-connection model, especially when even accessing open anonymity networks may be blocked and illegal.

The perhaps largest advantage that the trusted-connection model offers over traditional anonymity networks is protection against the Sybil attack [13], where a single attacker infiltrates the network by pretending to have many identities. Since the network depends on out of band trust relationships, an attacker gains strength only through the number of people he can fool into trusting him, and gains nothing by presenting multiple identities online.

Our current implementation does not perform any mixing of queries inside the nodes, and thus cannot, in spite of the fixed size of documents, defend against an adversary with global access. The threat model considered is more along the lines of that used by TOR [11], in that we assume that the adversary cannot perform advanced traffic analysis on the inter-node communication. It is possible to implement mixing within the nodes, at least to some extent, but currently our emphasis is on getting a efficient, usable, functioning system rather than strengthening the threat model.

6 Conclusion

We have described a new version of Freenet based on constructing an overlay using only trusted connections to build a distributed hashtable for use as a censorship resistant information publishing system. Much work still lies remains constructing this system, especially in testing it with a large user base, and seeing if peoples trusted connections in practice sufficiently match the small-world of theory to make communication in the network practical.

Many questions remain as to the viability of operating software like Freenet in the real world. Other open question regard the security and anonymity provided by the model. The difficulty of performing denial of service attacks by attempting to affect the routing in various ways needs to be studied, and similarly attacks by flooding and spamming in order to decrease the availability of legitimate data.

However, we see our design as a proof of concept of an entirely new form of peer-to-peer overlay, which ensures privacy for all participants, while offering a robust alternative for building free communication networks even where they are not desired by all.

References

1. L. Adamic and E. Adar. How to search a social network. *Social Networks*, 27:187–203, 2005.
2. R.J. Anderson. The eternity service. In *Proceedings of the 1st International Conference on the Theory and Applications of Cryptology*, Praque, Czech Republic, 1996.
3. P. Biddle, P. England, M. Peinado, and B. Willman. The darknet and the future of content distribution. In *Proceedings of the 2002 ACM Workshop on Digital Rights Management*, Washington DC, USA, 2002.
4. B. Bloom. Space time tradeoffs in hash coding with allowable errors. *Communications of ACM*, 13:422–426, 1970.
5. D. Bricklin. Friend-to-friend networks. `http://www.bricklin.com/f2f.htm`, 2000.
6. D. L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.
7. I. Clarke, T. Hong, S. Miller, O. Sandberg, and B. Wiley. Protecting free expression online with Freenet. *IEEE Internet Computing*, 6:40–49, 2002.
8. I. Clarke, T. Hong, O. Sandberg, and B. Wiley. Freenet: A distributed anonymous information storage and retrieval system. In *Proceedings of the ICSI Workshop on Design Issues in Anonymity and Unobservability*, pages 311–320, Berlin / Heidelberg, 2000. Springer.
9. I. de S. Pool and M. Kochen. Contacts and influence. *Social Networks*, 1:1–48, 1978.
10. R. Dingledine, M. Freedman, and D. Molnar. The free haven project: distributed anonymous storage service. In *Proceedings of the Workshop on Design Issues in Anonymity and Unobservability*, Berkeley CA, USA, 2001. Springer.
11. R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, 2004.
12. P. S. Dodds, M. Roby, and D. J. Watts. An experimental study of search in global social networks. *Science*, 301:827, 2003.
13. J. Douceur. The sybil attack. In *Proceedings of the IPTPS02 Workshop*, Cambridge MA, USA, 2002.
14. M. Draief and A. Ganesh. Efficient routing in Poisson small-world networks. *Journal of Applied Probability.*, 43:678–686, 2006.
15. M. Franceschetti and R. Meester. Navigation in small world networks, a scale-free continuum model. *Journal of Applied Probability*, 43:1173–1180, 2006.
16. D. Goldschlag, M. Reed, and P. Syverson. Onion routing fo ranonymous and private internet connections. *Communications of the ACM*, 42:39–41, 1999.
17. T. Hong. Performance. In A. Oram, editor, *Peer-to-Peer*. O'Reilly, Sebastopol CA, USA, 2001.
18. Tomas Isdal, Micheal Piatek, Arvind Krishnamurthy, and Thomas Anderson. Privacy-preserving P2P data sharing with OneSwarm. Technical report, 2009.
19. J. Kleinberg. The small-world phenomenon: an algorithmic perspective. In *Proceedings of the 32nd ACM Symposium on Theory of Computing (STOC)*, 2000.
20. J. Kubiatowicz, D. Bindel, Y. Chen, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. Oceanstore: An architecture for global-scale persistent storage. In *Proceedings of ACM ASPLOS*, 2000.
21. J. D. Lassica. *Darknet: Hollywood's War Against the Digital Generation.* John Wiley and Sons, New York NY, USA, 2005.
22. D. Liben-Nowell, J. Novak, R. Kumar, P. Raghavan, and A. Tomkins. Geograph routing in social networks. In *Proceedings of the National Academy of Science*, volume 102, pages 11623–11628, 2005.
23. G. Singh Manku, M. Bawa, and P. Raghavan. Symphony: Distributed hashing in small world. In *Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems*, 2003.
24. G. Singh Manku, M. Naor, and U. Wieder. Know thy neighbor's neighbor: the power of lookahead in randomized p2p networks. In *STOC '04: Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 54–63, New York, NY, USA, 2004. ACM.
25. S. Milgram. The small world problem. *Psychology Today*, 1:61, 1961.
26. C. G. Plaxton, R. Rajaraman, and A. W. Richa. Accessing nearby copies of replicated objects in a distributed environment. In *ACM Symposium on Parallel Algorithms and Architectures*, pages 311–320, 1997.

27. M. K. Reiter and A. D. Rubin. Anonymous web transactions with crowds. *Communications of the ACM*, 42:32–38, 1999.
28. L. Rizzo. Effective erasure codes for reliable computer communication protocols. *ACM Computer Communication Review*, 27(2):24–36, 1997.
29. O. Sandberg. Distributed routing in small-world networks. In *Proceedings of the Eighth Workshop on Algorithm Engineering and Experiments (ALENEX06)*, 2006.
30. O. Sandberg and I. Clarke. The evolution of navigable small-world networks. Technical Report 2007:14, Department of Computer Science and Engineering, Chalmers University of Technology, 2007.
31. Stephen B. Seidman. Network structure and minimum degree. *Social Networks*, 5(3):269 – 287, 1983.
32. I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *Proceedings of the 2001 ACM SIGCOMM Conference*, pages 149–160, 2001.
33. Z. Tan, W. Foster, and S. Goodman. China's state-coordinated internet infrastructure. *Communications of the ACM*, 42(6), 1999.
34. WASTE Team. Nullsoft waste: Network and security. `http://waste.sourceforge.net/index.php?id=information`, 2004.
35. M. Thorup and U. Zwick. Compact routing schemes. In *Proceedings of the 13th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 1–10, 2001.
36. J. Travers and S. Milgram. An experimental study of the small world problem. *Sociometry*, 32:425–443, 1969.
37. D.J. Watts and S. Strogatz. Collective dynamics of small world networks. *Nature*, 393:440, 1998.
38. H. Zhang, A. Goel, and R. Govindan. Using the small-world model to improve freenet performance. *Computer Networks*, 46:555–574, 2004.