Freenet: Distributed Anonymous Information Storage and Retrieval System

Olufemi Odegbile

Abstract

Both client-server and peer-to-peer architectures have been used to provide anonymous communication on internet. Mix-network is an example of client-server anonymous system and Crowds is an example of a peer-to-peer anonymous network. There is also some hybrid systems like onion routing (no batching but proxy servers forwarding request to on another). All these anonymous networks provide anonymous communication between individuals or anonymous access of publicly available information. In this project we will, investigate one way of providing anonymous storage and retrieval of information. This is important because not only author and consumer of information need protection. Storer (server or peer) of information also needs to be afforded deniability about the content of information hosted.

We will discuss a peer-to-peer network that provide anonymous information storage and retrieval: Freenet. Peer-to-peer network is an obvious choice because of scalability and possibilities of using unstructured architecture to increase anonymity. We will fully describe Freenet architecture. Then we will discuss how Freenet achieves stated anonymous goals and consider various threat models. We will conclude by enumerating issues that need to be addressed to improve Freenet performance.

Keyword: Anonymity, mixes, peer-to-peer, client-server, crowds, onion and Freenet

Introduction

There is a growing need for anonymity nowadays. The needs could be for either anonymous online transaction to enhance privacy or anonymous access or distribution of information while living under a repressive government for the fear of persecution among others. Different architectures has been proposed to provide anonymity. Chaum proposed a client-server architecture in form or mix-network that provides message unlinkability to sender. With all the advantages of mix-network, it is not as fault tolerant as desirable because of a single point of failure. It is also not suitable for low latency application because of possibility of high delay, a consequence of batching strategy needed to enhance its degree of anonymity.

Alternative architecture to providing anonymity is a distributed system or peer-to-peer architecture. A peer-to-peer network consist of collaborating end systems that share their resources (storage facilities or computing power) and overlay it special purpose network over internet (Zhang). Crowds is an example of peer-to-peer network that anonymously route internet requests among participating peers called Jondos. The end server is not certain about the identity of the jondo that initiates a request since each jondo randomly decides whether to forward to the destination (end server) or another participating jondo. Crowds is more scalable than mix-network and more applicable for low latency application as jondo forward packet as soon as it is received (Rubin, 1998).

Despite these desirable properties of Crowd, one flaw still remains in its blender that periodically notifies Jondos and distributes keys whenever new peer joins the network. Though blender's failure does not prevent anonymous communication for existing jondos, it however prevents addition of new peers and keys distribution. Other more scalable and decentralized peer-to-peer networks like Tarzan, MorphMix (Rennhard M. and Plattner, 2004) and Salsa (Wright, 2006) have been proposed to address perceived weaknesses in Crowds,. Still, all of these P2P networks only provide anonymous routing of internet requests among peers to a destination severs that host the needed resources.

All these afore mentioned networks do a good of protecting consumer of information but not so much when it comes to producer and consumer of information. Organization hosting unwanted information could be subpoena, harassed to reveal information about the producer of the information or attacked to deny service to legitimate users. There are important features that we desire systems that will provide anonymous information storage and retrieval to possess. In addition to consumer anonymity, we also desire author anonymity and deniability for the storers of the information. We want a system that is resilient to any attempt to deny access to information. We desire a system an anonymous system that will not be at the expense of efficient information storage, routing and retrieval.

Freenet is a peer-to-peer anonymous storage networks that try to achieve all these design goals listed above. Freenet is an unstructured distributed system that depends on location-independent keys to store, route and retrieve information. It consist of nodes/peers that commit a portion of

their discs and serves as proxy servers to anonymously forward requests for information or files to store. Freenet is different from crowds in that both the file obtains as a result of a query and file passed for storage are cached by each node along the query path. Also files are not probabilistically forwarded but rather, using routing table. In the next two sections, we will fully describe architecture of Freenet, how it meets design goals listed above, and threat models. We will conclude by discussing current challenges that may impede Freenet usability.

Architecture

Freenet (Clarke, 2001), designed by Ian Clarke, is a system aimed at providing anonymous information storage and retrieval. Freenet is designed as a distributed file storage system where each peer called *node* contributes a portion of their disc called *datastore* to the system. Anonymity is provided through Freenet proxing of request and answer among nodes. When a node receives a request for file from a user or upstream node, it searches its datastore for the file and then forward the request to the most promising node in its routing table. Once a file is inserted into the network, a user does not know which node(s) will store the file. Consequently the file cannot be remove and legitimate users cannot be prevented from accessing it. An important feature of Freenet is that routing are done using location-independent keys. Routing table is then updated after each successful search. In the next two subsections, we will discuss in details different keys used in Freenet and how queries and insertions are routed

Keys

Keys are very important in Freenet because they facilitate location-independent way of querying and inserting files. There are two main keys used in Freenet. They are Content-Hash Keys (CHK) and Signed-Subspace Keys (SSK). Since Freenet is a (distributed) file system, the two keys are similar to inodes and filenames in a conventional file systems (Clarke I. S., 2002). CHK is obtained from directly hashing the contents of the corresponding file using SHA-1 secure hashes. Since SHA-1 collision is very rear, CHK uniquely identify every file. In addition, each file is encrypted by a randomly generated hash key. Then user publish CHK and decryption key to allow for the file retrieval.

Signed-Subspace Keys (SSK) provide more high-level human use because it incorporates a *descriptive string* that briefly describes each file for the key's derivation. SSK allows creation of a person namespace or directory that only the owner can write into. A user first randomly generates a public and private key pair for his/her namespace. To calculate SSK for a file, but public namespace key and the file descriptive key are independently hashed and then XOR'ed together. The result of XOR is then hashed to yield Signed-Subspace Key (SSK). User signs the file with private subspace key and then publish the descriptive string along with namespace public key. Anybody can read from a namespace but storing into a subspace requires namespace private key. Finally, user encrypt the file with its descriptive string. Note that the decryption key is not stored with the encrypted file.

An advantage of SSK is that it can be used to store indirect files: files containing pointers to CHK(s) of the a instead of encrypted file. SSK is also useful to update file existing in Freenet. To update a file, the user first insert a newer version of the file which will obviously generate a new CHK. Then SSK is now updated to point to the newer version of the file.

Requesting Data

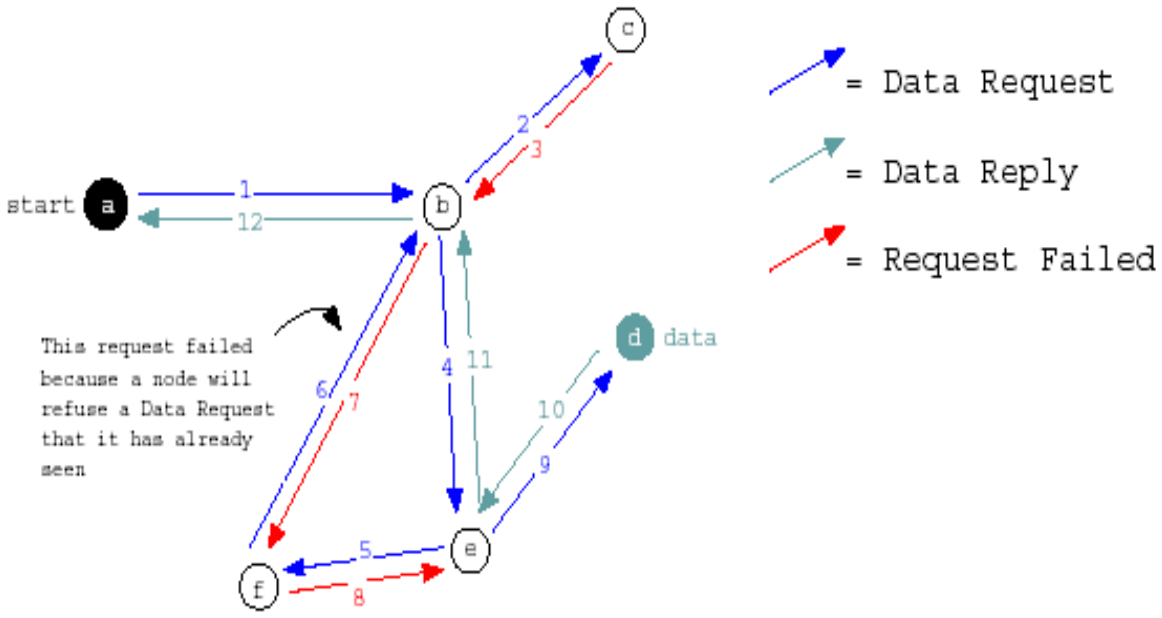
Since Freenet is an unstructured distributed system, natural way of querying data is broadcasting. Freenet avoid broadcasting request because of scalability issue. Instead, it uses steepest-ascent hill-climbing search in which each node forwards queries to the node it thinks closest to the target (Clarke I. S., 2002).

Any node that joins Freenet commit a portion of their disc to host file in the system and routing table that list the addresses of other nodes and the keys of the files they host. Now to query a file, the keys (either SSK or CHK) of the file is first obtained or calculated from the published information. Each query contains key, hops-to-live limit (HTL), and a randomly generated 64 bits transaction ID. HTL prevents indefinite searches and transaction ID prevents loops. When a node receives a query requesting a key, it checks it datastore first for the key. If the key is not found, the node send the query to node hosting a closest key lexicographically to the key requested. Each node also start a timer equal to the expected time to contact as many nodes as HTL.

When the HTL is reached, a failure message is propagated backward to the requesting node. After a node receives a failure message or the timer runs out without receiving any reply or it cannot

forward the request to its preferred choice, the node then forward the request to the next node in its routing table hosting key lexicographically similar to the requested key. A node can also return failure message if it has encountered the request before to prevent looping.

If the search is ultimately successful, the data is pass backward to the original requestor with the address of the node that supply the data. In addition, every node on the route cache the file and dynamically updates its routing table. To protect the identity of the file storer, each node on the route can occasionally change the reply by pointing themselves as the original host of the file. The diagram below shows a typical request sequence. The request moves through the network from node to node, backing out of dead-end (step 3) and loop (step 7) before locating the desire file at node *d* (Clarke, 2001).



Storing Data

Inserting a file follows similar process as in requesting a file. Insert message is sent by the user along with randomly-generated transaction ID, HTL and the keys. The first thing that will be done is to check for key collision. Collision can occur if there is a file with the same key in the system in case of CHK or there is a file with the same descriptive string in case of SSK. HTL represents

the number of copies of the file to be stored. When a node receive an insert message, it first check its datastore for key collision. If there is a file with the same key as the file to be inserted, the file will be returned or propagated backwardly to the each node in the route.

On the other hand, if there is no collision, the node forward the insert message to the node that host the closest key in the its routing table as it would be done for query. The forwarding will continue until HTL limit is reached. At this point, the user sends the file along the already established path. Each node along the path verifies the file against the key and then caches the file. Dead-ends and loops are handled the same way as in request/query by backtracking to the next closest key lexicographically.

Managing Data

Freenet storage capacity depends on the number of the peers and the total disc space committed to the system is determined by each peer. Since the number of peers is finite, the capacity of the network will also be bounded. Consequently, there may not be enough space left to cache a new file or input a new route in the routing table. More so, Freenet has no restriction on the number of files nor the size of file inserted into the system. So how does Freenet handle insufficient space in the datastore for a new file or when there is no entry left for update in routing table? Both situations are managed using Least Recently Used (LRU).

Node storage and routing table sorted using the time files or entries are recently accessed. If a file or entry has never been accessed, we use the time it was inserted. When there is a new file to cache/store and there is no sufficient space in a node's datastore, least recently accessed file is deleted until there is enough space for the new file. LRU is also used to update a routing table when there is no all the space is filled up. A node may still be able to access deleted file because it takes longer time to delete routing information of the deleted file in the routing table. One of the consequence of LRU is that Freenet does not provide permanent story. File copies will be eventually remove from the system if all node never access the file again (Clarke, 2001).

How Freenet Achieve Design Goals

Freenet has an unstructured distributive key space. It was chosen to be unstructured because of robustness and security considerations (Zhang). Irregularity in system increases it entropy which

consequently increases anonymity. In this section we will discuss how Freenet achieves the anonymity design goals.

Consumer Anonymity

Freenet anonymous routing protocol is similar to Crowd. A node forwards each request to the node in its routing table that hosts closest key to the key requested. Since *hop-to-live* (HTL) value are randomly decremented, a node on the route cannot be certain if the preceding node on the route is the initiator of the request or not. Consumer anonymity is however exposed to a local eavesdropper through a traffic analysis even though all traffic are encrypted. A local eavesdropper can link a node to a request forwarded if s/he observes that there is no incoming request. This may be defeated if there is high traffic in the system and each node also act as a mixes. The cost will be delay and extra traffic through padding.

Author Anonymity

Can an author/publisher be linked to a file? To insert a file in Freenet, the network will first be probed for collision and the file is then send if there is no collision. In other words, an author behaves like consumer when probing for collision of keys and sending file. So consumer anonymity extends to author when performing those activities. Other properties of Freenet makes author anonymity stronger. The file inserted is also cached on multiple nodes along the route. Thus the anonymity set of potential authors is large. Another important property is that if two file are related content-wise or belong to the same namespace (inserted by the same author using SSK), hashing ensures that there keys will be scattered throughout the network (Clarke I. S., 2002). Therefore preventing using clustering of keys to identify an author.

Deniability of Storer

There are two ways in which deniability is afforded to the storer of information. The first way is that a node has no control on what can be stored in its datastore. So we cannot link a storer as a consumer or author of the file in its datastore. This is because any file in its datastore could have been cached when another node request the file through the storer. The second way a storer is afforded deniability is through encryption. Users are encouraged to encrypt file with any

encryption techniques they choose before inserting file into the system and not to store decryption key with the file. This allow storer to claim ignorance about the content of the file.

For legal purpose, there is an underlining assumption: storer will not be held liable for the content of the material on their computer. In 2010 however, Germany Supreme court ruled that owner of Wifi could be held liable to what is done through their connection (Schweda, 2010). If this law were to be applied to Freenet, deniability afforded to storer in Freenet may not be sufficient to escape legal trouble. Currently, there is no known legal challenge related to Freenet.

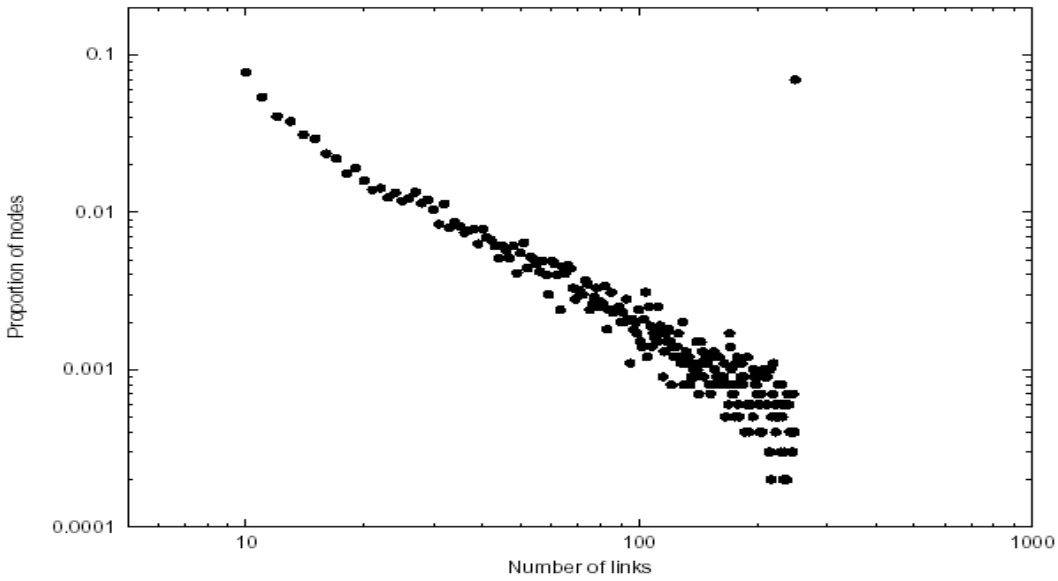
Resilient Against Preventing Legitimate Access

A draw back to client-server architecture is susceptibility to one point of failure attack. Freenet is designed as a distributed system to address this concern. In addition, there are other inherent attributes of Freenet to prevent denial of access to materials on Freenet. First, there is no way for materials to be directly removed or deleted. Even if the file can be deleted, it will be difficult t to identify all nodes that host the file, since materials are replicated on multiple nodes (probably scattered over the internet). Lastly, any attempt to prevent access to a node that store material of interest will lead to creation of new route. This will further propagate the file as nodes on the new route cached the material. This further serves as a deterrent against preventing legitimate access.

Network Convergence

A consequence of Freenet being unstructured distributed system is that it has to route request non-deterministically. This can potentially leads to inefficient information retrieval. In particular, the network may return a fail search even though the material is in the system. In other words, there is high probability of not finding an existing file in the network (Skogh, 2006). This is a huge hit to Freenet's usability. However, Freenet's creator hypothesize in (Clarke, 2001) that the routing quality should improve over time for two reasons. First, after many queries and insertions, nodes will begin to specialize in similar keys. So if a node is listed in a its routing table as hosting some keys, all the request forwarded to it will most likely be for keys similar to key already hosted. Consequently, routing table keeps history of successful searching which helps improve future queries.

Another important observation is that a node will receive insert request of keys that are similar to the keys already in its datastore. So nodes acquire files with similar keys reinforcing the clustering of similar keys. Therefore, these observed phenomenon should lead to better and efficient routing over time. Freenet creator also demonstrated in (Clarke, 2001) that this observation holds by simulating Freenet consisting of 1000 nodes with datastore size of 50, routing table size of 250 addresses. Random keys were inserted in the network. After sufficient keys have be inserted into the network, successive 300 random requests using 500 hops-to-live limit were made over time to evaluate performance of the network. Figure below shows the distribution of links before locating the requested links. We can infer from the distribution that over time hit ratio improves and files are found in few number of hop. This implies that Freenet convergences



Routing in Freenet can be improved if each node has complete information about keys hosted by nodes on its routing table. In traditional Freenet, each node knows of only keys listed on its routing tables. In (Skogh, 2006), a modification to Freenet's routing protocol was proposed that will improve each node information about its neighbors. A File Mesh is used to compactly and anonymously represent all the files that a node has. File Mesh consist of a bit field of length N. Each bit in the field represents one-Nth of the key space. Whenever a storer return a file upstream or accept a file, it also pass file mesh representing its keys space. The hope is that file mesh will provide approximate knowledge about all requests that may be forwarded to a node.

Attack Models

Freenet's attackers are similar to crowds: local eavesdropper, compromised node and collaborating nodes. In previous section, we discuss how Freenet provides anonymity for consumers, authors and storers of information. Nevertheless, there are still possible attacks especially the ones that exploit Freenet's non-deterministic routing strategy and hops-to-live limits included in every message. We will primarily focus on attacks that can be launched by compromised nodes and collaborating nodes.

First a compromised node may want to use HTL value to infer requestor position on the path. However, this will be difficult to do because the true position is at least masked by the random decrement of the HTL. Another important threat from compromised node is passing of a junk file upstream or attempt to steer traffic to itself by pretending to host requested keys. This can lead to poisoning of routing tables if successful. However, these attacks can be foiled as each node can detect inauthentic data by hashing the file and comparing it hash with key requested. The only way these attacks are successful is if the compromised node somehow forged cryptographic signature. Furthermore, a compromised node may decide not to answer a request for keys in its datastore. This may not have significant effect of Freenet hit ratio because the data is probably cached by another node.

Node collaborates to identify author or consumer or deny access to data. For nodes collaborating to compromise anonymity, one of the nodes must have the requested keys and other nodes have similar keys. Non-deterministic routing of Freenet makes this difficult. Nodes can also collaborate to exhaust storage space in the system by inserting large file very frequents. Thus replacing data in the system through LRU. One way to defeat this attack is to divide datastore into a "new files" section and to a "establish file" section. This will ensure that collaboration will not be able to replace established file in the system. Cost of this modification is that new legitimate file may not be able to abide for a long time in the system. Another subtle attacks by collaborating nodes is to try to replace a material by passing a corrupted material under its key. This may end up spreading the legitimate file if collision is detected. This attack can have a limited success if HTL is chosen sufficiently small to avoid collision.

More so, an attack that can be used to probe the content of neighboring nodes (nodes that are in its routing table) is by sending a query with HTL = 1. This attack may not, by itself, be effective since a node can claim deniability about the contents of its datastore. However, success of large requests of similar files (in content) with HTL = 1 attack can raise a red flag.

Conclusion and Future Works

Freenet was conceived as a decentralized system that provides anonymous storage, retrieval and routing of information using location- independent keys and caching of data requested on multiple nodes. We have discussed some inherent properties of Freenet that makes it difficult to prevent access to legitimate users. Though Freenet hit ratio may be undesirable at set-up, its creator expects that hit ratio to improve significantly after many queries and insertions.

There are limitations to Freenet especially when it comes to its usability. The first limitation is how to publish keys. There are two suggestions about how to publish keys on Freenet's websites (https://freenetproject.org/help.html#chk). The fist way is to advertised keys on existing Freenet sites by sending anonymously message containing brief description about information contained on the file corresponds to the key being advertised. The second way is for individual to send keys to people interested in accessing data s/he inserted into Freenet. No way yet to carry out "google search" like or keywords search of all the keys of interest in Freenet.

Another limitation is that no incentives for people not interested in anonymity to commit bandwidth and portion of their discs to Freenet. This is important because the more nodes we have in the system the more capacity and the faster is the download and upload speed. It is also worth noting that Freenet does not provide a permanent facility. Thus, it is also not too practical if the user just wants to store some files for backup purposes and only needs to access them when needed (Qian, n.d.). Free Haven is an example of persistent anonymous storage system (R. Dingledine, 2000).

Lastly, a very important question is that does Freenet really converge over time. If not, is there any better caching policy, or even alternative routing protocols to enhance Freenet especially its usability? These are some of the issues that must be addressed going forward.

References

Clarke, I. O. (2001). Freenet: A Distributed Anonymous Information Storage and Retrieval System. *International workshop on Designing privacy enhancing technologies: design issues in anonymity and unobservability* (pp. 46-66). NYC: Springer-Verlag.

Clarke, I. S. (2002, January). Protecting Free Expression Online with Freenet. *IEEE Internet Computing, 6*, 40 - 49.

Edman, M. a. (2009, December). On anonymity in an electronic society: A survey of anonymous communication. *ACM Computing Surveys, 42*(1), 3.

Grahn, K. J. (2014). Anonymous communication on the internet. *Proceedings of Informing Science and IT Education*, (pp. 103-120). Retrieved from http://Proceedings.InformingScience.org/InSITE2014/InSITE14p103-120Grahn0483.pdf

Greene, T. C. (n.d.). *I know what you downloaded from Freenet*. Retrieved from The Register: http://www.theregister.co.uk/2005/05/13/freener_not_so_anonymous/

Hui, K. Y. (2004). Twelfth IEEE International Workshop on Quality of Service, 2004. IEEE.

Qian, H. D. (n.d.). *Paper review: Freenet: A Distributed Anonymous Information Storage and Retrieval System*. Retrieved from http://zoo.cs.yale.edu/classes/cs633/Reviews/Cswh00.hq9.html

R. Dingledine, M. J. (2000). The Free Haven Project: Distributed Anonymous Storage Services. *Proceedings of the Workshop on Design Issues in Anonymity and Unobservability*.

Rennhard M. and Plattner, B. (2004). Practical Anonymity for the Masses with MorphMix. *In Proceedings of the Financial Cryptography Conference* , (pp. 233-250). Key West, USA.

Rubin, M. K. (1998). Crowds: anonymity for Web transactions. *ACM Transactions on Information and System Security, 1*(1), 66-92.

Schweda, S. (2010, July 1). *IRIS Merlin.* Retrieved from BGH Finds WLAN Operator Liable: http://merlin.obs.coe.int/iris/2010/7/article13.en.html

Skogh, H.-E. H. (2006). Fast Freenet: Improving Freenet Performance by Preferential Partition Routing and File Mesh Propagation. *Sixth IEEE International Symposium on Cluster Computing and the Grid.*

Wright, A. N. (2006). Salsa: A Structured Approach to Large-Scale Anonymity. *Proceedings of the 13th ACM conference on Computer and Communications Security* (pp. 17-26). NYC: ACM.

Zhang, H. G. (n.d.). Using the small-world model to improve freenet performance. *ACM SIGCOMM Computer Communication Review.* NYC.