CS 856: Paper Summaries
W Anthony Young - 20161423
October 7th, 2004

Clarke, et. al - Protecting Free Expression Online with Freenet

Summary of Contents

Freenet is a peer to peer system built to protect privacy and prevent censorship on the internet. The main idea behind Freenet is to protect the freedom of users to communicate without fear of reprisal from government organizations. Freenet protects the identity of users, creators and holders of information on the network. It also seeks to provide high availability and reliability through decentralization, and efficient and scalable data storage. It has been remarked that a system such as this can be used by criminals to anonymously plan crimes. However, such planning information would be made available to the system's entire user pool, removing its secrecy.

Freenet users each attach a node to the system. That node provides some disk space to the resource pool. Files are inserted by sending an insert message containing a hash key and later the data file. Files are retrieved by sending a request message containing the hash key of the target file. Hash keys are called Globally Unique Identifiers (GUID's). They are made up of the hashed value of the data file. This allows files to be inserted without knowledge of their contents. GUID's can be either content hash keys (CHK) or signed subspace keys (SSK). A CHK is a file hash value. An SSK is a pointer to a directory or alias file. Identical files can be inserted as the CHK's will be the same, and files will coalesce within the system. Identical SSK's cannot be inserted. An SSK consists of a public-private key system where the public key points to the directory or file alias, and the private key allows the author to update the file or key. Alias files can point to real CHK's, thus allowing files to be updated.

Freenet intentionally makes it difficult to direct messages and files to a specific host. This allows the system to be incredibly resilient to attacks from inside and outside the system. This has the unfortunate drawback of making the implementation slightly inefficient. Also, improvements that would increase efficiency are difficult to make without sacrificing some security. For example, messages in Freenet do not travel between sender and receiver directly. Instead, they travel through several intermediate nodes to provide anonymity. Each node in the chain knows only about the next node to receive the message as link information is encrypted in the message header. This arrangement protects producers and users of information, as well as information holders. No site knows where data originates from or is destined for.

Message routing in Freenet is very inefficient. Messages are routed to the next node in the system that is believed to have the data. When a query is received, a node checks its data store to see if it holds the data containing the GUID in the message. If it does, it replies that it is the data holder. If not, it passes the message to the node in its routing table with the closest key to the one requested. If a request is successful, each node in the chain will pass the file data back to the requestor. Intermediate nodes may store copies of the file in their cache to speed up frequent searches or further increase anonymity of the data holder. Request messages have a time to live (TTL) that prevents them from being passed endlessly. Insert messages follow the same path as request

messages. Once an insert request reaches a node with a similarly keyed file, the node replies and the data is transferred from the inserting to the storing node. In this manner, new files are inserted where the system will look for them.

Data at nodes should be encrypted so that data holders are not aware of the information that resides on their system, and thus cannot be held responsible for it. When a node first joins the network, it generates a public-private key pair for itself. After the keys are generated, the node announces its presence to known peers (known through websites or personal communications). It is then assigned a portion of the hash key space so that it might start receiving files to store. The known peers then forward the announcement message to random nodes in their routing tables. Thus, the network knows the new node's presence and location.

Eventually, disk space at nodes will run out and files will have to be removed. So, Freenet maintains a datestamp on files. When a file needs to be deleted, the least recently requested files are removed until there is enough space for the new file. Thus, unused copies of files are deleted, but it is probable that at least one copy of every file will be retained as a cached file in some data store.

Freenet experiences good scalability and fault tolerance as demonstrated by the small-world network model. In this model, the majority of nodes are connected to a few peers while a small number of nodes are connected to many peers. Freenet's scalability was tested with a set of random insertions and queries, as well as adding nodes at specific points in time. Originally, 20 nodes were in the system. Nodes were added and queries were performed until the number of peers grew to 200 000. Median path length for requests is shown to grow sub-linearly with network size to $N^{0.28}$. To test fault tolerance, random nodes were removed from the system. It was found that the network was still relatively well connected until approximately 90% of the nodes were removed. When highly connected nodes were targeted for removal, the network stayed relatively well connected until approximately 60% of the nodes were removed.

The authors outline various improvements that must be made to the system, including the ability to search for files of a given topic.

Comments

In general, this paper is well written and very informative. The ideas presented appear to be quite novel. However, I have three major issues with Freenet and the authors' presentation of it:

-Because files are deleted in a timestamp order, it is possible for malicious users to simply insert huge data files full of random garbage into the system, thereby forcing out older, but still useful, data files. In this manner, it would not take long for a malicious user to completely remove a large portion of the data in the system. The authors claim that this is not possible as files are cached at other sites. However, eventually cached copies must be deleted too. One important point to note is that it is nearly impossible to target specific files for removal as the hashing system makes it extremely difficult.

-Performance evaluations were repeated only ten times before averages were computed. It seems to me that tests should be performed many more times before an accurate sample size has been generated for averages and medians to be computed. Therefore, I do not know how much the statistics and numbers reported can be considered accurate to the sample. As well, the numbers are simulation and not "real world" data.

-The authors do not provide any mechanism to search for files on a related topic in Freenet. It seems to me that no system hosting files would be complete without such a mechanism built in from the start. How can the system be useful if users cannot find the information they are looking for?

Hopefully these issues will be resolved in future releases of Freenet. However, despite these shortcomings, I would be interested to see the software in action as the real novelty of it is the protection of anonymity through enhanced security protocols.

Ripeanu, et. al - Mapping the Gnutella Network

Please see presentation slides