Freenet: A Distributed Anonymous Information Storage and Retrieval System

Presented by:
Nathaniel Husted
School of Informatics
Indiana University
nhusted@indiana.edu

What is wrong with current systems?

- Central Points of failure
- Little privacy is given
- Certain people desire privacy in authorship and/or readership
- People don’t like central points of failure.

What is FreeNet?

- A distributed information storage and retrieval system.
- designed to address concerns of privacy and availability
- operates as a location-independent distributed file system across many individual computers
- allows files to be inserted, stored, an requested anonymous

There are Five Design Goals

- Anonymity for both producers and consumers of information
- Deniability for storers of information
- Resistance to attempts by third parties to deny access to information
- Efficient Dynamic Storage and routing of information
- Decentralization of all network functions

What are the inspirations for Freenet?

- Chuam's Mix-net scheme
- Anonymizer
- Crowds
- Web Mixes
- Rewebber
- Taz
- Eternity
- Free Haven
- Distributed.net
- Napster
- Gnutella
- Intermemory
- India
- Akamai

Freenet’s Architecture is P2P

- Implemented as a peer-to-peer network of nodes
- They query each other to store and retrieve files
- Use location-independent keys
- Nodes have their own data store
- Nodes have a dynamic routing table

How do I retrieve data with Freenet?

- User’s hash a short descriptive string (e.g. text/philosophy/sun-tzu/art-of-war)

How do I retrieve data with Freenet?

1. User sends a request message to her own node
    - Specifies hops-to-live and key

How do I retrieve data with Freenet?

- Node receiving a request checks it local data store.
- If the data is found, it returns it with a note saying it was the source of the data.
- If the data was not found, it looks up the nearest key in its routing table and forwards the request to that node.
- NOTE: Keys are ordered lexicographically

How do I retrieve data with Freenet?

- If final request is successful, the data is returned by the final node.

- Each node along the way updates its routing table and aches the file in its own local data store.

There can be problems with retrieval

- What happens when a node runs out of candidates?
  - It reports a failure to its upstream neighbor which will try a second choice.
- What if hops-to-live count is exceeded?
  - A failure result propagates back to the original requester
- What if there is a loop?
  - Any node will return a failure if it receives a request that it sent.
- NOTE: Nodes can curtail hops-to-live and drop requests.

Routing improves over time

- Nodes specialize in locating sets of similar keys
- Nodes become specialized in storing clusters of files with similar keys
- Nodes replicate data with each request so data will be closer to requesters.
  - Redundancy is also provided with this mechanism.

Storing Data

- Storing is similar to requesting
- To insert:
  - A node picks appropriate descriptive text string and hashes it.
  - She then send san insert message to her own node.
  - Her node see's if the key is already there, if so it returns a pre-existing file.
  - If the key is not found, it looks up the nearest key in the table and forwards it to that node.

  - Process finishes when hops-to-live is reached and no collision is detected. Data is sent after this point.

Storing has three positive affects

- Newly inserted files are placed on nodes possessing files with similar keys.
- New nodes can tell the rest of the network about their existence by inserting data.
- Attempting to overwrite a file with a collision only spreads it further.

Managing the daata…

- Nodes us a Last Recently Used cache.
- Items sorted in decreasing order by time of most recent request.
- Files are evicted when a new file comes in and there is no more storage space.
    - Least recently used file is chosen.
- The data store is not a cache
- Inserted files should be encrypted because Freenet does not does this itself.
    - Authors recommend using unhashed descriptive strings as keys

Protocol Details

- Protocol Agnostic
- Request.Handshake
- Request.Data
- Reply.Restart
- Send.Data
- Reply.NotFound
- Request.Continue
- Reply.Restart
- Request.Insert
- Send.Insert

Naming, Searching, and Updating

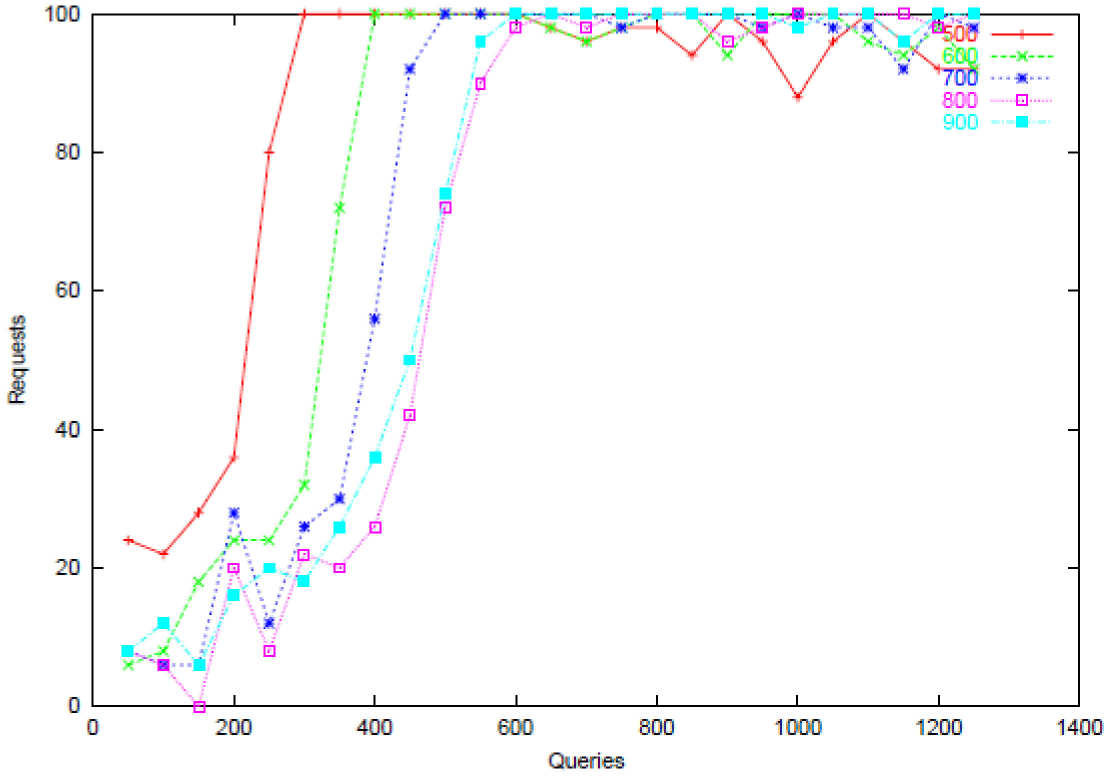
- The name space is very flat so discovering documents and name collisions is difficult.
  - Solutions:
    - Bookmark lists in the form of compilation keys
- Name collisions:
  - Solved by two-level structure.
    - Indirect and Real files.
- Updating: Done with a signature-verifying Key and updated with this key.
  - More indirection can be used to avoid "updating out of existence"

Performance Simulation

- Data stores of size 40
- Table size of 50 addresses
- 10 unique items to store locally

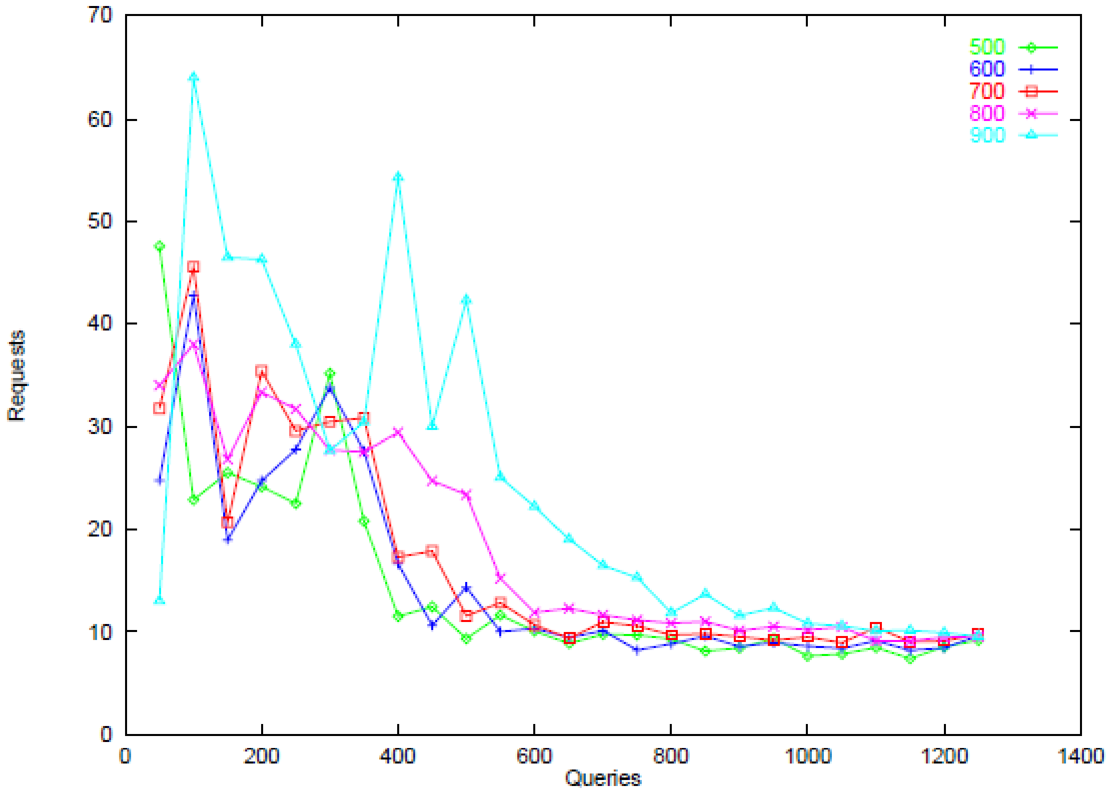
Performance Simulation

- Percentage of successful requests over time

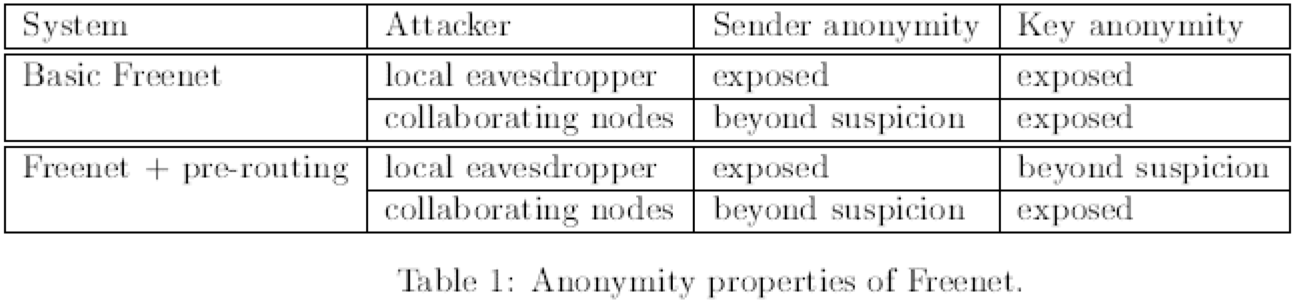


Performance Simulation

- Number of Hops Per Request over time



Security



System	Attacker	Sender anonymity	Key anonymity
Basic Freenet	local eavesdropper	exposed	exposed
	collaborating nodes	beyond suspicion	exposed
Freenet + pre-routing	local eavesdropper	exposed	beyond suspicion
	collaborating nodes	beyond suspicion	exposed

Table 1: Anonymity properties of Freenet.

Conclusion

- Freenet provides an effective means of anonymous information storage and retrieval
- Over 15,000 copies deployed and interesting files in circulation.
- More realistic simulations must be done.

Can I download Freenet?

- Yes!

- Just go to http://freenet.soourceforge.net

Questions?