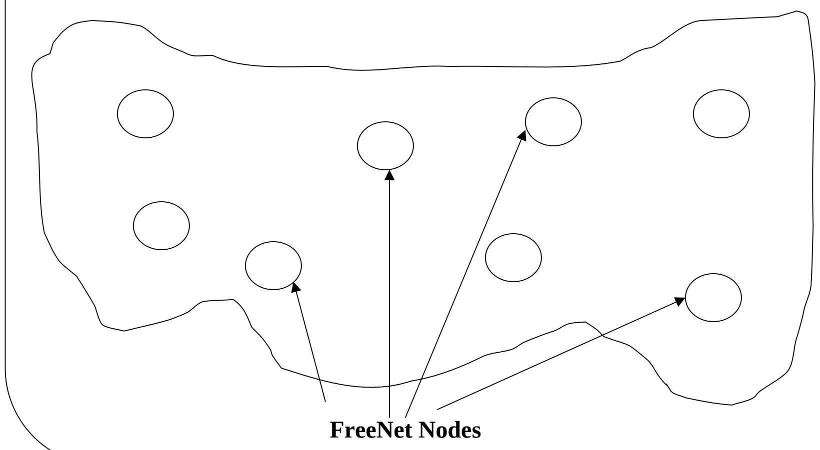
Overview of FreeNet

FreeNet - Serverless, Symmetric, Secure, Parallel Internet File System



Overview of FreeNet

Lecture Goals
- Extract algorithms from FreeNet paper
- Analyze algorithms in FreeNet Implementation

Assumed Knowledge
- Public Key Encryption
- Hashing Algorithms

Content Summary
- Conceptual Elements
- File Insertion
- File Retrieval
- File Update
- Large File Management
- Node Join
- Communication Protocols

Overview of FreeNet

Conceptual Elements of FreeNet

Network Nodes
- Storage
- Files
- Routing Tables

Files
- Byte String
- Path Name
- Search Key
  - Keyword Signed Key
  - Content Key
- File Signature

Directories - Personal Name Spaces
- Signed SubSpace Key

Interactions/Transactions
- Join Network
- Insert File
- Request File

Storage Management Algorithm
- File Storage, Retrieval and Retention
- Routing Tables

Overview of FreeNet

Files

File Content = string of bytes

File Name/ File Description - Unix text string

/text/philosophy/sun-tzu/art-of-war

/peertopeer/books/oram

Keys associated with each file.

{Public Key, Private Key} = PPKP-Generator(File Descriptor)

File Keys

Keyword Signed Key = Secure Hash Algorithm(Public Key)

Content Hash Key = SHA(File Content)

File Encryption Key - Use file descriptor as an encryption key.

File Signature = Private Key

Node Resource Management

1. Each node allocates the amount of storage to be assigned to FreeNet Functions.

2. Storage is partitioned for file data storage and routing table storage.

3. Node data storage is managed as an LRU Cache. Files which are not accessed are eventually deleted from a node.

4. Routing table entries are also managed as an LRU cache but entries of the routing tables persist after eviction of file data.

Personal Name Spaces/Directories

File keys may be coupled to a unique identifier for a personal name space The personal name space is created as follows.

1. Randomly generate a public/private key pair (The signed subspace key.)

{Public Key, Private Key} = PPKP(Random())

2. Hash the public key

HPK = SHA(Public Key)

3. Hash the file path name/file descriptor

HFPN = SHA(Path Name)

4. File Key = XOR(HFPN, HPK)

5. File Signature = Private Key

6. Publish the file descriptor, the public key and the File Encryption Key.

7. This file is a “directory file” to which only the owner of the private key can add files.

Name Space “Directory Structure”

Create a file using the signed-subspace key process with hypertext links to other files, perhaps signed-subspace key files.

The linked files may themselves contain links, etc.

Used to implement file update.

Definitions:

1. Node Locality Set

*** Each node has a locality set of neighbors.**

2. Node Routing Table

*** A Node Routing Table is a set of Search Key - Node pairs.**

3. File Insertion Algorithm

a) Create a path name and a binary file key for the file.

b) The creating node sends a message to itself including a “hops to live” counter for this insertion.

c) The originating node then checks to see if that key is already in use.

d) If a match is found then the file associated with the previously defined key is returned as though a request had been made and the node knows this key has already been used. The node generates a new key for this file and again attempts the insertion process.

e) If no match is found then the originating node finds the node with the nearest key in its routing table and sends it an “insert” message with the key and the “hops to live” counter.

f) If this insert message causes a collision then the file is returned to the upstream inserter and again behave as though a request had been made. (Cache the file locally and create a routing table entry for the data source.

Overview of FreeNet

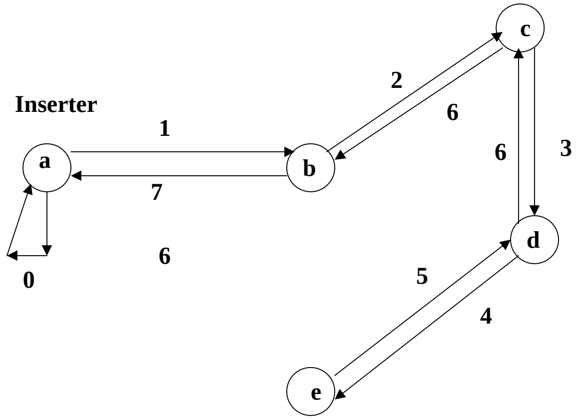
3. File Insertion Algorithm - Continued

g) If the “hops to live” limit is reached without a collision being detected then the insertion is successful and the “all clear” message is sent back to the originator of the file insertion.

h) The originator of the file insertion then sends the file itself which is propagated along the path of the key collision search. Each node will create a local copy of the file and establish a routing table entry matching the inserting node and the key. (A forwarding node may choose to arbitrarily change the supposed source when it forwards the insert message.)

i) The descriptive string (path name) is published by some out of band mechanism or in the case of name space encoded files, the descriptive string name and the subspace public key.

Overview of FreeNet



Successful insertion of file key in five nodes. File data will follow the same path.

Overview of FreeNet

Properties of Insertion Algorithm

1. Key-space locality

Files are cached on nodes with similar keys.

2. New nodes can use inserts to extend their network locality space.

3. Discourages fake file propagation. (A fake file is a file with “junk” or malicious content with a key identical to some file with actual data.)

The original files are propagated upon collision.

Overview of FreeNet

File Search and Retrieval Algorithm

a) Obtain or calculate the binary file search key.

b) Send a message to yourself with the key and a “hops to live” counter.

c) If the data is stored locally the request is satisfied.

d) If the data is not found then the search is continued at the node in the local routing table associated with the key “nearest” to the search key.

e) If the search is successful then the data is returned to the upstream requestor which will cache the data and create a new entry in its routing table associating the actual data source with the key.

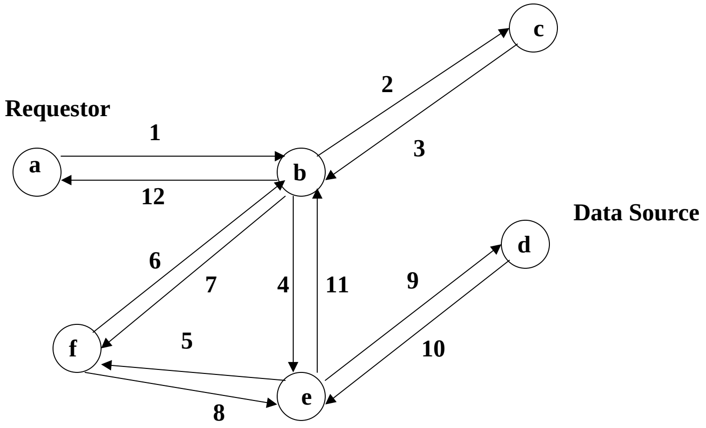
Overview of FreeNet

File Search and Retrieval Algorithm- Continued

f) If a node in the search path cannot forward a request to its preferred downstream node then the node with the second nearest key is selected as the downstream target, etc. If a node runs out of possible downstream paths without finding the file then it reports failure to the immediately upstream requestor. This upstream requestor then chooses its second choice target. This process is recursively followed until the originator runs out of downstream targets at which time failure is reported.

g) If the "hops to live" limit is exceeded at any time in this search process then the failure result is propagated back to the original requestor.

Overview of FreeNet



Overview of FreeNet

Properties of Search Algorithm

1. Nodes will tend to accumulate similar keys and similar files.

2. This should lead to better routing as well as better caching performance.

3. The greater the number of requests the more copies of the file which will exist.

4. The more localized the requests the more localized the number of copies of the file will become.

5. Network connectivity is increased as routing tables are built by requests and inserts.

6. Nodes with popular files will preferentially appear in routing tables of other nodes.

7. While files are clustered by key they are dispersed with respect to subject.

Overview of FreeNet

File Search and Retrieval for Updatable Files

a) An updatable file should be stored under a "content hash" key.

b) A file with the content hash key is inserted in a personal name space (signed-subspace key)

c) The file is encrypted with a random key which is published with the file key.

d) Retrieval is by first retrieving the file containing the content hash key and using this key to search for the file.

File Update Algorithm

a) A new version of an updatable file is inserted under its content hash key.

B) The insert algorithm is executed for the new indirect file under the original signed subspace key.

C) When the insert reaches a node with the old version of the file a key collision will occur.

D) The node will check the signature on the new version, verify that it is correct and replace the old version of the file with the new version. Then the signed subspace key will always lead to the new version while a content hash search will still lead to the old version.

Management of Large Files

1. Partition a large file.
2. Create a content hash key for each partition.
3. Create a file to serve as the indirect access file for the partitions of the large file.
3. Insert the content hash key for each partition separately as an entry in the indirect file.
5. Insert the indirect file.

Publication of Names and Search Mechanisms

1. Create a search engine specific to FreeNet.
2. Create for each actual file a family of “lightweight indirect files” each named by a search keyword relating to the actual file and containing a pointer to the actual file.
3. Encourage users to create directories of “favorites”

Node Join Protocol

1. The joining node must obtain the address of at least one node which is already a member of FreeNet by some out of band means.
2. The joining node chooses a random seed and sends an announcement message containing the hash of that seed, its address and a “hops to live” counter to the existing nodes for which it knows addresses.
3. When a node receives an announcement message it generates a random seed, XORs that with the hash it received and hashes the result again to generated a “commitment.”
4. The node which received the announcement message forwards the new hash to some node chosen randomly from its routing table and decrements the hops to live counter.
5. The last node to receive the announcement message just generates a seed.
6. There all the nodes in the announce chain share their seeds and the key for the new node is assigned as the XOR of all of the seeds.
7. Then each of the nodes in the announce chain add the new node to their routing table under that key.

Overview of FreeNet

FreeNet Communication Protocols

1. Each request, insert or join action is a transaction which has a (probably) unique ID associated with it.

2. Each message originated as a result of a transaction has a transaction ID which is carried in each message resulting from a transaction.

3.Node addresses are {transport identifier, transport specific identifier}, for example {tcp/192.168.1.1:19114}

4. Nodes which change addresses frequently may wish to used address resolution keys which are signed subspace keys updated to contain the current actual address of the node.

5. All transactions begin with a Request.Handshake message specifying the return address of the sending node.

6. The receiver of a Request.Handshake message may (or may not) respond with a Reply.Handshake message specifying the protocol it understands.

Overview of FreeNet

FreeNet Communication Protocols - Continued

7. All messages have a 64 bit randomly generated transaction ID, a hops to live counter, and a depth counter.

8. Hops to live and depth are set by the originator of a transaction is decremented by each receiver.

9. The propagation chain is continued with hops to live = 1 with some finite probability.

10. Depth is incremented at each hop and is used by a replying node to set hops to live high enough to reach a requestor. A depth of 1 is not automatically incremented but may be passed unchanged with some finite probability.

11. A time-out is superimposed on transactions.

12. Nodes in a chain may send back Reply.Restart messages based on knowledge of network delays not accessible to the originator.

Overview of FreeNet

Summary and Conclusions

1. Design achieves goals of symmetry, fully distributed control, parallelism, high security and use of “unused resources.”

2. Performance properties are generally unknown except for simulations.

3. Algorithm domain for fully distributed control is largely unexplored.

4. State of the art is about where client-server systems were 10 years ago.

5. Next step - Integration of distributed computation and distributed file system.

Potential of system with 100,000,000 computers linked by Internet.

6. Problem - Application design