Peer-to-Peer Networking

Case Study: Freenet

Lecture content

- Freenet (protocol version 0.5)
  - Distributed storage service
  - Anonymity, security and censorship resistance
- P2P system studied further
  - The original design
  - Later enchantments are not discussed here

Freenet Design Principles

- Freenet is distributed information storage system
  - While Napster, Gnutella, etc. are *sharing* systems, Freenet provides a storage service
- Anonymity, privacy and prevention of censorship
  - Adding and retrieving files cannot be traced back to user
  - Strategies for storage that prevent erasing or anyhow censoring the content (or at least making it quite hard)
- System is fully decentralised
  - Self-organising, providing availability and reliability

Freenet Design Principles

- Assume that participants can operate maliciously or fail without warning
- Pooling of unused disk space among the users
  - Cannot see where a certain file is located physically
  - Redundant replication of data
  - Note: not a guaranteed data storage, strategies to delete junk and least needed data
- Trusting participants to provide resources rather than demanding funding or disk space

Freenet Architecture

- Participants each run a node with some disk space
  - Add new file: send the network an insert message containing the file and its location-independent globally unique identifier (GUID), which causes file to be stored on some set of nodes
  - During a file's lifetime, it might migrate to or be replicated on other nodes. To retrieve a file, the user sends out a request message containing the GUID key
  - When the request reaches one of the nodes where the file is stored, that node passes the data back to the request's originator

GUID Keys

- GUID keys calculated using SHA-1 secure hash
- Content-hash keys (CHK): low-level data storage key
  - Always points to the exact file
  - Calculated from the file content, usable for verification
- Signed-subspace key (SSK): higher-level human use
  - CHK is like inode, while SSK is analogous to filename
  - Generate random public-private key pair to identify file
  - Choose a short description, “beverage/coffee”

SSK Continued

- Signed-subspace key (SSK): higher-level human use
  - Calculate the file SSK by hashing the public part half of the subspace key and the above string independently
  - Concatenate hashes and hash again
  - Sign the with the private key: handling nodes verify that file is not tampered before accepting
  - Retrieve file: need public key and descriptive string to recreate SSK
  - Add and updating requires to have the private key

SSK Usage

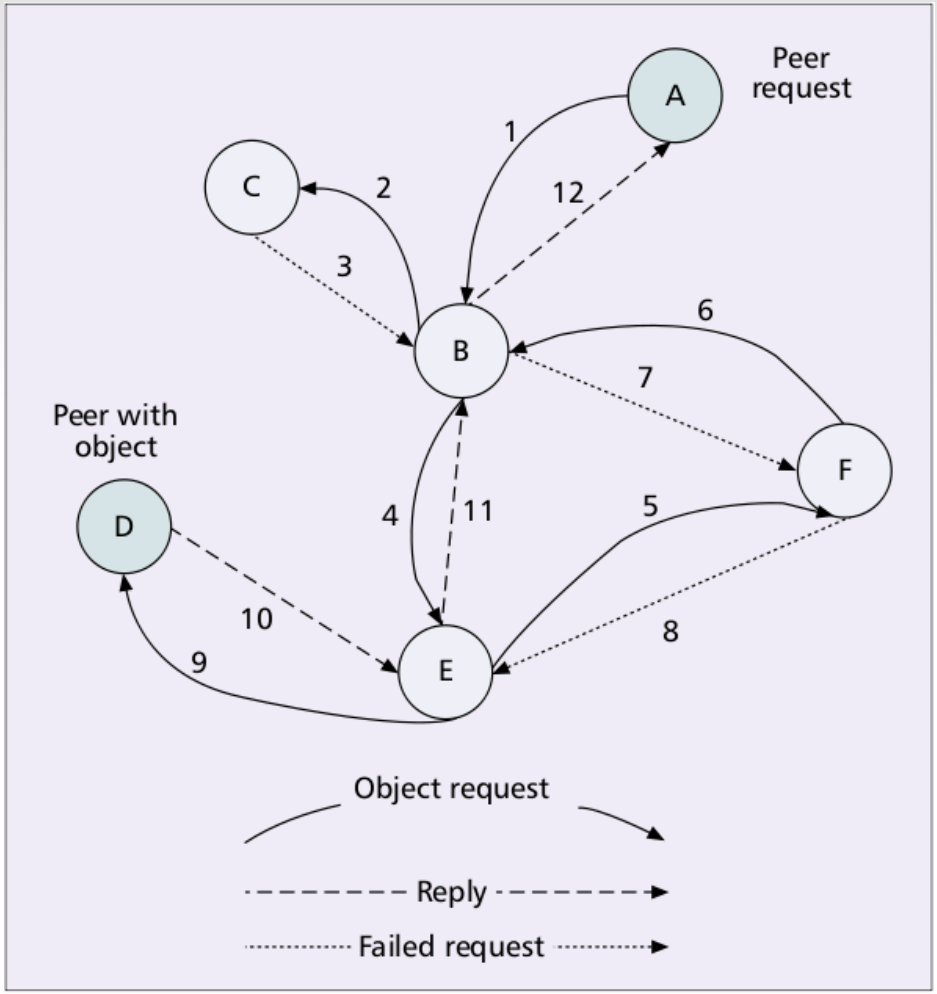
- Facilitate trust by guaranteeing that the same pseudonymous person created all files in subspace
  - Use SSK to send out a newsletter or publish a web page
- SSK used commonly to store indirect files containing pointers to CHKs rather than store data directly
  - Combine human readability and publisher authentication of SSKs with the fast verification of CHKs
- Splitting files in multiple parts

Messaging

- Assume hostile nodes from inside and outside
- Trade-off between data locatability and security (concealing the routes and messaging)
- Messages travel through node-to-node chains
  - Each link is individually encrypted
  - Nodes know only about their immediately neighbours
  - Cannot say who are the original sender and final receiver, just pass the message according to routing rules
  - Protects also the file holder: cannot find it, cannot attack it

Routing

- Typical Freenet request sequence
- The request moves through the network from node to node (1,2)
- Backing out of dead end (3)
- A loop occurs (6,7), and finally the object found in D
- Node selection method called as the steepest-ascent hill climbing search
- Each node forwards queries to the node that it thinks is closest to the target (i.e. who has the closest key in the routing table)



[Lua et al. A survey and comparison of peer-to-peer overlay network schemes]

Requesting Files

- Each node maintains a routing table
  - Addresses of other nodes and GUID keys they're holding
- Successful request is passed back upstream
  - Each node adds the data holder and key to routing table
  - Nodes reliably answering requests are added to more tables
  - Nodes may cache object locally
- Concealing data holder identity: alter reply messages
- Requests have a time-to-live value

Inserting Files

- Insert message follows the same path that a request for the same key would take
- Calculate GUID key and send with a TTL value
  - TTL represents the number of copies to store
- When insert message received, node checks if it's a new key and inserts, otherwise the insert fails
- Again, routing tables are updated during an insert message routing

Data Encryption

- Node operators can remain ignorant of the contests of their data stores, as data is encrypted before storing it to the network
- Data encryption keys not used in routing or included in network messages
  - Inserter distributes them at the same time as the corresponding GUID keys
- Node operator sees only random GUIDs attached to opaque data

Network Evolution

- Local knowledge improves as more queries are being processed
  - Does not require global directories to upkeep routing
- Joining the network
  - A node needs to find one or more existing Freenet node addresses via out-of-band means
  - New node sends its identity (public key) and physical address to known peer, which forwards it to others
  - New node is then assigned a random key space for data

Clustering Keys and Data

- Routing tables of nodes should specialise in handling clusters of similar keys
  - Each node will mostly receive requests for keys that are similar to the keys it is associated with in other nodes routing tables
- Nodes' data stores should also specialise in storing clusters of files of similar keys
- Clustering keys and data should make future queries more effective

Searching

- How users can search the network for relevant keys?
  - Like the Web: spidering or publishing lists of bookmarks?
  - Conflicting to Freenet design goals
- Original Freenet did not specify any keyword search, but required that user provides the appropriate key
- Keyword search is a difficult to implement when data is hashed (applied also to DHT)
  - Newer unofficial version specify partially distributed techniques (At them moment Freenet is on version 0.7)

Caching

- Performance and availability by caching objects
- When an object is returned (forwarded) after a successful retrieval (insertion), the peer caches the object in its datastore, and passes the object to the upstream (downstream) requester
- Causes datastore to exceed the designated size
  - Least Recently Used (LRU) objects are ejected in order until there is space. LRU policy is also applied to the routing table entries when the table is full.

Small-world network model in Freenet

- Characterised by a power-law distribution of graph degree (here, the number of routing table entries)
- Majority of nodes has relatively few local connections to other nodes
- But, a significant small number of nodes have large wide-ranging sets of connections
- Enables efficient short paths because these well-connected nodes provide shortcuts

Performance

- Original simulations by Clarke et al. in 2002
- Fairly good scalability and fault-tolerance
  - Explained by the small-world network (SWN) effect
  - Scalability: median path length $N^{0.28}$ (N nodes)
  - One million nodes would mean median path length of 30
  - SWN is resilient to random node failures (likely to eliminate nodes from the poorly connected majority)
  - But targeted attack against well-connected nodes shatter network into disconnected fragments when 60% removed

Performance

- A Freenet network with 1 000 nodes and 5 000 actions (inserts and requests) to train the routing
  - The median path length is below 10, fair good
  - average pathlength is around 35.4, not so good
  - Only 60.5% of requests are successful with an HTL of 10, hence, 4 out of 10 queries are unfulfilled
- Freenet has good average performance but poor worst-case performance, because a few bad routing choices can throw a request completely off track

Freenet Conclusions

- Anonymity and privacy as main goals by designing a distributed and decentralised P2P data storage system
- Achieving security and data verification not easy without central authority
  - Trade-off between performance (routing and messaging) and providing certain level of anonymity and security
- Besides the Freenet provided materials, you might want to check the “Frequently Ignored Questions”
  - http://pl.atyp.us/content/tech/freenet_fiq.html

Freenet in Practise

- Usage depends if Freenet goals match your goals
  - Performance was not the first design principle
  - Some parties consider it merely as an interesting research
- Have to manually find some peers to connect first and learn the network
  - Efficiency should increase when more links known
- Actually includes various content besides files
  - Web browser can access the pages on Freenet (Freesites)

Lecture Considerations

- For what purposes Freenet is suitable?
- What are the drawbacks?
- Check out the materials for further information, how the search works (does version 0.7 offer improvements)
  - For example, can you do any keyword-based searches?