3^{rd} Generation P2P Networks

Freenet

Smruti R. Sarangi

Department of Computer Science
Indian Institute of Technology
New Delhi, India

Outline

Design Goals

- Freenet is a 3^{rd} generation peer2peer network.
- Features: Publication, replication , and retrieval of data.
- Main feature: It is a **dark net** $\Rightarrow$ Not possible to find the true origin of a file.
- Wonder why Freenet has additional security !!!
- Design goals: Anonymity, Deniability of storers, Efficient storage, Reliability
- The storage is meant to be temporary (not necessarily permanent)

Available at: http://freenet.sourceforge.net

Outline

A Freenet Node

A Freenet Node

- Each node maintains its local data store
- Dynamic routing table: address of other nodes and the keys that they (might) hold
- A node knows only about its immediate neighbours (not others)

Freenet Query

- Queries are sent to a node that can pass it to its neighbours.
- Each query has a TTL (time-to-live field) that is decremented at every hop
- A query has a pseudo-random identifier . This ensures that there are no cycles introduced while forwarding queries.

Steps in a Query

1. The *node* hashes the name of the file. This is the key
2. In its routing table, it looks up the key that is closest to the key , and passes the request to its owner .
3. If a node finds the file, it returns the contents, along with its address (saying that it is the owner of the data).
4. Otherwise, it finds the nearest key in its routing table, and forwards the request to that node.
5. If the request is ultimately successful, then the nodes on the way will:
   1. Cache the file
   2. Create an entry in their routing tables, and record the original source

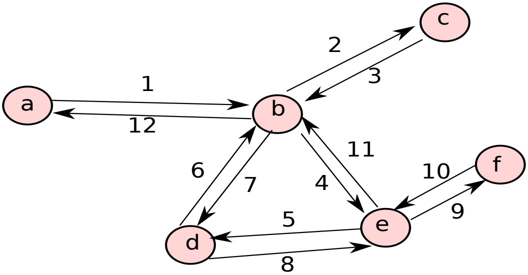
Steps in a Query - II

1. If a node cannot forward the request to another node:
   - Creates a cycle
   - Failure
2. Try the key with the second closest distance.
3. At every hop decrease the TTL till it reaches 0.
   - To reduce the network load, the TTL can be dynamically decreased.
   - Nodes can decided to process which request next based on the TTL

Operation of the System

(a) Routing Table

key	address	content (?)
key	address	content (?)



(b) Example: Multiple messages being sent

Search Quality

- Gradually over time, information disseminates.
- Nodes start aggregating files with similar keys (numerically close)
- Popular data gets replicated at a large number of nodes.
- Routing tables gradually get bigger.
  - New entries get created all the time.
  - Nodes can discover more of the network without disclosing their identity.

Outline

Storing Data

- User first creates a file key .
- She then sends an insert message to its own node (file, key, TTL)
- If the node finds the key in its routing table, it returns the contents of the file.
- Otherwise, finds the closest key in its routing table and forwards the insert message to it.
- If that insert causes a hash collision , the node passes data back to the upstream requester.
    - Cache the file locally, and create a routing table entry (in response to insert).

No Hash Collisions

- If the TTL field becomes 0, without any collisions , then it means that the insert is successful .
  - Let the original requester know.
  - Each node on the path adds the file and key to its routing table. It also caches the file.
  - The original node also adds the file and key to its store, and routing table.
  - How do you beat security ???

- Nodes along the way lie about the data source
- Add their own address or some other node's address

Advantage of this Mechanism

Advantages

- Newly inserted files are placed on nodes with similar keys.
- Information about newly inserted files can quickly be disseminated.
- An intruder will find it difficult to deliberately introduce hash collisions.

Data Management

- The routing table and data stores are finite structures. They follow a LRU (least recently used) replacement policy
- This ensures that old files get deleted from the system.
- Legal issues :
  - A data store can deny the knowledge of the files it has.
  - Take the hash key, and encrypt the contents of the files with it.
  - Any node can always decrypt a file, if it knows the key
    - However, it will not be able to find out what the original key was.
    - Example: A data store can always say that it didn't know that it had that many pop songs. It didn't know which file was a pop song.

Outline

Freenet Message

- Packet oriented, self-contained messages: TCP or UDP
- Every transaction (search or insert) has a unique ID
- All messages contain: 64-bit transaction ID, TTL counter, and a depth field.
  - An attacker can guess the identities of nodes by scanning the TTL value.
  - To thwart this: With a finite probability when the TTL field reaches 1, keep propagating the request to other nodes.
  - Have another field called depth that is incremented at each hop. It starts with a (> 0) random value.
  - Before the destination sends the message back to the source , set the TTL = depth. This ensures that the message will not die before reaching the requester.

Timers

- For every request, the requester starts a timer.
  - If a timer times out, then we infer a failure.
  - Sometimes downstream nodes may send Reply.Restart messages to the requester. The requester extends its timer.

Successful Request

- If the request is successful, the remote node will send the Send.Data message.
- It can send the id of the data source (or possibly fake it).
- It TTL reaches 0, it sends a Reply.NotFound message.
- If there are no more paths left and ($TTL \neq 0$), then a Request.Continue message is sent.
- The requester can send the request to other nodes in its routing table.

Outline

1. Overview
   - Queries
   - Data Storage
2. Details of the Protocol
   - Message Format
   - Naming and Searching
3. Evaluation
   - Setup
   - Results

Naming, Searching, and Security

Organizing Files

- Might be a good idea to have a directory of keys (files containing similar data).
- Read: legal issues
- If there are no such issues, we can introduce directories , and bookmark lists
- Search capabilities:
  - Should we have a search engine like Google. Goes against our design goals: anonymity
  - Solution: use a lot of indirect files containing meta-data, all over the network: list of keywords $\rightarrow$ keys of the files
- To ensure that files have not been tampered, have a content-hash

Security

- Sender anonymity is preserved because:
  - A node can never say if the node that is requesting a file is the original requester, or is merely forwarding a request.
  - Messages between pairs of nodes can also be encrypted (against eavesdroppers)
- Pre-routing
  - The requester decides the routing path to a destination if it has detailed routing tables.
  - Encrypt a message with a succession of public keys (for all the nodes on the path)
  - A node will have no idea regarding the sender. It will only know the id of the next hop.
  - No idea of the requested key .

Outline

Evaluation Setup

Setup

- Network has between 500 to 900 nodes.
- 40 items in each node.
- Routing table size: 50 addresses
- Network topology: Chain

Outline

1. Overview
   - Queries
   - Data Storage
2. Details of the Protocol
   - Message Format
   - Naming and Searching
3. Evaluation
   - Setup
   - Results

Successful Requests(%) vs #Queries

- The number of queries were varied from 50 to 1200
- For 500 nodes, the percentage of successful requests rose quickly from 20% (at 50 queries) to 100% (at 300 queries).
- With 600 to 900 nodes, the percentages started at roughly 10%.
- They reached close to 100% for more than 400-500 queries.
- More are the nodes, lesser is the percentage of successful requests.

#Hops vs #Queries

- Experiments were conducted with 500, 600, 700, 800 and 900 nodes.
- The #hops reduced quadratically from roughly 50 (20 queries) to 10 ($>$ 600 queries).
- More are the nodes, more are the hops.

Clarke, Ian, et al. "Freenet: A distributed anonymous information storage and retrieval system." Designing Privacy Enhancing Technologies. Springer Berlin Heidelberg, 2001.