COMP7370 Advanced Computer and Network Security

Freenet: A Distributed Anonymous Information Storage and Retrieval System (2)

Topics:

1. Architecture

Topic 1: Architecture

- Big picture
  - Cooperative distributed file system
  - Each node – local datastore
  - Location independence
  - Lazy replication
  - Share unused disk space (for users)
  - Routing (adaptive): Requests -> nodes (local decision about where to send requests next)
  - Hops-to-live: prevent infinite chains

- Keys and searching
  - File IDs: binary file keys
  - 160-bit SHA-1
  - Keyword-signed key (KSK)
    - Keywords = string -> [generate key pairs] -> public/private key
    - Public key -> [hash function] -> file key
    - Private key, file -> [sign the file] (Why? integrity check)
    - Encryption: file, string -> [encryption]
  - How to share? Use the string (keywords)
  - Flat global namespace: problem? Solution? Use personal namespace
  - Store/Publish
    - Store: file/private_key
    - Publish: [file/private_key]/public_key

  - Content_hash_key

    File -> [Hash] -> content_hash_key(published)
    |
    ----------> [encryption] -> cipher
    Rand_key (published)----------^

  - Update:
    - lazy
    - update content_hash_key
    - update signed_subspace_key -> [point to] -> new file
    - When old file's content_hash_key differs from new content_hash_key, find new version

- Retrieve data
  - Need binary file keys to retrieve files
  - Send binary file keys as requests
  - Each node check its local store
    - Return if found, return the file
    - Otherwise, What?
      a) Look keys in its routing table;
      b) Forward the request to other nodes
      c) Found: pass data back to upstream requestor
      d) caching: update routing table (file key and source)

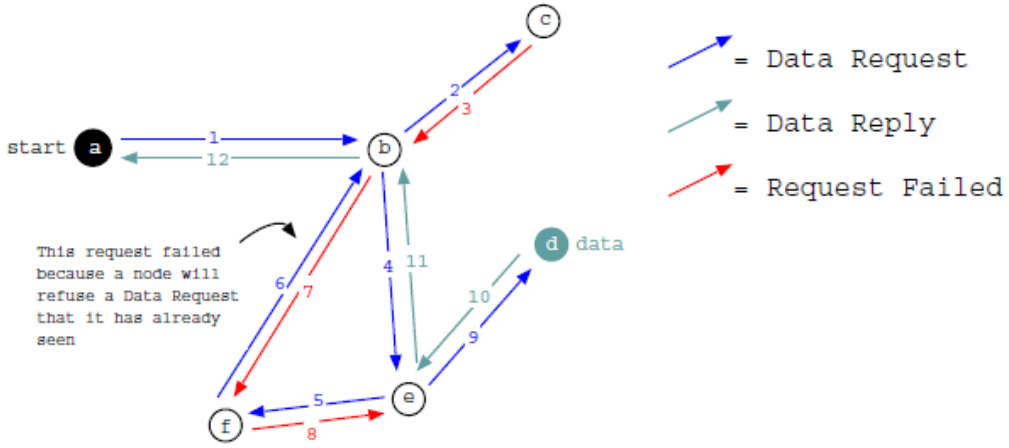

Fig. 1. A typical request sequence.

- Store file - input: f and hops_to_live
  - Compute binary file key, i.e., bfk(f)
  - Check existing keys in own store
  - If pre-existing key found, re-compute binary key
  - Look up the routing table, find the nearest key bfk' to bfk
  - Send f and its bfk to node(bfk')
  - Do above until hops_to_live limit is reached

- Manage data
  - Each node: configure datastore size
  - LRU cache
  - Routing table: LRU.
  - No guarantees of file lifetimes: outdated files can fade away