SDIS: Distributed Backup Service v2.0

09/04/2018

Bruno Miguel Faustino Moreno, up201504781@fe.up.pt

Francisco Teixeira Lopes, ei11056@fe.up.pt

# Summary of the Application

The application is a continuation of the first application developed for SDIS, a distributed backup service where Peers are responsible for keeping chunks of a given file backed up. The original operations allowed on the files/chunks were: the backup of a file, the restoration of a file, the deletion of a file’s chunks and control over the storage space allowed for each Peer. In this second iteration, the service will be more robust (fault-tolerant) and secure, as well as featuring all the missing enhancements from the first version. The application will continue being a Java standalone desktop application.

# Main features roadmap

These are the main features planned for v2.0 and the ones relevant for the target grade. The features are listed in tentative development order.

1. Implement RESTORE and DELETE enhancement
2. Implement any leftover TODO items from version 1.0
3. Improve the scalability as detailed in the Advanced Features section
4. Improve the consistency as detailed in the Advanced Features section
5. Improve the fault-tolerance as detailed in the Advanced Features section
6. Improve the security of the service as detailed in the Advanced Features section

# Additional features roadmap

The most likely scenario is the main features taking up the entirety of the development time but in case that doesn’t happen, some additional features have been thought about.

1. Analyse the control flow of the service to see if quick upgrades can be applied
2. Upgrade the entire service to online usage using a centralized Tracker Peer that keeps info about the Peers in the system such as their network location and can query them for chunks of a specific file. Seeing as this is a considerable time effort, only one Tracker Peer would exist in the system, not great for fault-tolerance but the main roadmap already features improvements in that regard
3. Either investigate fault-tolerance of the Tracker Peer, since there’s only one, or investigate “trackerless” service (DHT? Chord?)

“trackerless” service would probably be an entire project on its own so it’s the least likely to be implemented.

# Advanced features

## Security

Some security features that should make it into version 2.0:

* Peer authentication using either secret keys or certificates
* Message encryption / Chunk data encryption
* Message validation to prevent tampering

## Scalability

Version 1.0 of the service had a high level of threading but was still missing a few spots, such as:

* PUTCHUNK threading instead of sequential sending
* Service channels using queues to avoid losing messages when thread limit is reached for worker threads
* Asynchronous I/O so threads no longer block on I/O operations

## Consistency

While each Peer keeps a list of what they think is on the service they don’t synchronize this info across the service. To address that issue Peers would synchronize their databases across the service.

## Fault-tolerance

Currently the system is not resilient to file metadata loss, it needs access to the original file to perform any kind of operation. To improve upon this fault, the Peer’s database would store the relevant metadata to allow any operation to be performed without the original file.