A Cloud P2P Environment for Controlled Sharing of Images

CSCE 4411 – Term Project
Fall 2025

Prof. Amr El-Kadi

A cloud P2P environment for controlled sharing of images

CSCE 4411 – Term Project

Fall 2025

Prof. Amr El-Kadi

Introduction

In this project, you will put to use lots of the design concepts we cover in class, with a primary focus on transparency as it applies to load balancing, fault tolerance, and P2P services. You will use a distributed election algorithm(s) of your choice to possibly support these cloud qualities where applicable.

Your cloud will support two main functionalities: encryption of user images through steganography and a user-oriented discovery service.

The testing environment will physically consist of three machines running the cloud for services and three machines each running many user processes.

Users

Users are expected to own images. Upon registering with the cloud's discovery service, they will get their images encrypted before being able to share them with their peers. So, request messages for encryption will be generated, and images will be sent to the cloud for encryption. Upon returning encrypted images (and this operation could be done any number of times), a client may inquire with the discovery service who its peers are and directly contact them seeking lower resolution images of what they are willing to share. The client may specify which images it needs along with the requested number of views. The images' owner would then transfer encrypted copies of the images embedded within them the number of permitted views.

Users can only see their own images, or images that have their usernames hidden in them. Also, each time a user views an image, the number of views needs to be updated inside the image, and the user is denied access to the image when the number of views gets consumed (it would see a default image instead).

The owner of the image has the right to update the image data by adding or removing users from the image or changing their viewing quota. In that regard, we should somehow support offline operation with a best-effort policy to update views, taking into account viewers who are offline or image owners who are offline. You may add such services' support to the discovery service. Such a facility would provide for reconstructing permissions should a failure happen to one of the peers.

During the demo, three physical machines will be used, each to run one or more frontends for clients. Your demos (and project design document) should show test cases demonstrating the system's behavior, covering the cases described herein.

The Cloud

The cloud will provide two main services, as stated earlier, and utilize a P2P architecture for decision-making, especially in load balancing and fault-tolerance. One simple yet effective way to implement load balancing is to have the client's middleware multicast requests to the cloud servers (physically, there will be three). Those server peers would then elect who would carry the incoming workload using a distributed election algorithm of your choice. Several parameters may be involved in the decision-making, including current server load, I/O bandwidth, communication reliability with the originating client, communication delay, and so on.

To simulate failures, the three servers will periodically elect one machine to ignore communication, thus creating the effect of a failure. It would remain in that state for up to 20 seconds, then rejoin the pack. Upon its revival, its state has to be updated accordingly for consistency with that of its peers (the other servers in the cloud).

The discovery of service is a very simple service that users register with when they are online. It is a means by which a user would get to know which peers are online and how to reach them. It would then talk to them P2P without using the cloud as long as both are online. That table should be kept consistent within the cloud.

Grading etc.

Much more detail is and will be communicated to you in class; thus, attendance is of utmost importance, as is the constant consultation with myself. The documentation of your design decisions, alternates considered, and justifications is of utmost importance as well. The project grade is set to 15% for the Distriburted Election Survey, 30% for encryption service, load balancing, and fault tolerance, and 30% for the Discovery Service and the P2P operation of the clients.

Documentation should include results, measurements, and data collection of tens of thousands of runs to be statistically viable. Design decisions made, changed, considered with justifications need to be documented consistently over the course of the semester. Poor documentation, or lack thereof, will only result in a bad grade no matter how well the project works and looks.

The project grade breakdown is as follows:

15% Distributed Election Algorithms Survey

30% for Phase 1

> 10% for Demo
>
> 20% for design, report, and analysis to include encryption, load balancing, fault tolerance, stress testing, analysis of results collected, and client and cloud node parallelization models adopted with justification.

30% for Phase 2

10% for Demo

20% for design, report, and analysis to include directory of service, peer-to-peer operation, and detailed use-cases.

In all cases, the report should clearly show design decisions taken, changed, and provide justification. There should be a separate section on the role of each and every member of the team.

Useful Resources

Steganography

http://en.wikipedia.org/wiki/Steganography
http://easybmp.sourceforge.net/steganography.html
http://manytools.org/hacker-tools/steganography-encode-text-into-image/
http://www.maketecheasier.com/hide-confidential-data-inside-images-in-linux/
http://www.findbestopensource.com/tagged/Steganography