*Ximin Luo*
*Queens' College*
*xl269*

Computer Science Tripos Part II Project Proposal

# Tag-driven routing in a distributed storage network

2009-10-30

**Project Originator:** self

**Resources Required:** none

**Project Supervisor:** *George Danezis*

**Signature:**

**Director of Studies:** *Robin Walker*

**Signature:**

**Overseers:** *Alan Blackwell* and *Ian Wassell*

**Signatures:**

# Introduction

The search capabilities of most current peer-to-peer networks are weak compared to centralised indexes on the world wide web. Popular query-routing algorithms, and their main shortcomings, include:

**Broadcasting:** (eg. old Gnutella) These are generally susceptible to flooding attacks and do not scale well.

**Multicasting, random walks:** (eg. OneSwarm) The reach of the request is limited; there is no systematic way to obtain results beyond this limit.

**Routing hashkey-addresses:** (eg. DHTs, Freenet, GNUnet) This requires prior knowledge of the target object. Some systems emulate a namespace by converting keywords to sets of hashkey addresses (eg. GNUnet `KBlocks`), but this is hard to secure whilst remaining globally open.

The aim of this project is to build a basic high-level search system for a distributed storage network, that addresses the above shortcomings. It will use semantic routing to find objects from anywhere on the network, hopefully with good probability even if the object is rare.

In order to perform semantic routing, the system will make use of the following data structures. (Conceptually, these are only abstract objects; the mechanism in which they are stored and retrieved is, for now, overlooked.)

**index:** an inverted index that pins tags against objects. The objects could be content (ie. search results), but they could also be other indexes.

**tag-graph:** a graph that defines relationships between tags. The design must allow an algorithm to determine whether, and to what extent, a tag is a more "general" or more "specific" form of another tag (in some sense).

Any node will be able to publish such objects, and to give approval to the ones it encounters. The routing algorithm would then involve retrieving trusted indexes and tag-graphs, and efficiently navigating the network of links between them. The structure of the network is likely to greatly affect its performance; this will be explored in the evaluation.

# Design issues

I have chosen to focus on tag-based searches, since the vast majority of global queries are of this level of complexity. Generally, advanced queries only need to operate on a local, rather than global scale.

Indexes could (and ideally would) be automatically generated; however, methods of assigning tags to data is a whole topic in itself and this project will not cover it. Instead, I will assume that tags have already been assigned, and work on designing a system to deliver this information globally.

Allowing anyone to maintain indexes has the advantage of not limiting metadata supply to a small minority (eg. the authors), who could abuse their position, or just be inefficient. Experience indicates that this tends to give better results, even if the "metadata" is as simple as a link.

Hierarchical address spaces are good for routing. With semantic tags, however, research shows that strict hierarchies are hard to maintain, and eventually become inconsistent and inadequate for usage needs. So we take the basic idea behind hierarchical routing, and modify it slightly: instead of having just one parent, any tag can now be semantically related to any number of more "general" tags. This relationship creates a tag-network that contains subgraphs with hierarchy-like structures, to guide routing.

A pattern similar to this is mirrored in society: more "general" ideas are known by a greater subset of the population to a lesser degree of detail, with the gaps often being filled instead by pointers to people who do know the details. As humans, we often exploit this property in our everyday attempts to find information. We can use the index and tag-graph data structures to construct a network with this property, and then derive a routing algorithm that exploits it to perform efficiently.

Extracting search results then becomes a question of which indexes to trust, and how to efficiently traverse the links between indexes to reach the data you want. We can use social network information in order to facilitate this, such as having nodes give (revokable) approval ratings for indexes and tag-graphs.

The project will focus on implementing a prototype of the system as an abstract library, rather than lower-level components needed for real-world deployment, such as the syntax of communications protocols or data formats. Producing the latter is unnecessary, since many networks have such infrastructure already that the library could hook onto (assuming that our assumptions and design principles are compatible).

# Evaluation

I will measure how performance of the algorithm scales as it runs on different network topologies and usage loads, and how it degrades as various parts of the network fail or come under attack.

The prototype library should be light enough for up to perhaps 10,000 nodes to be simulated on a single machine, which should be suitable for our purposes.

Realistic data can be obtained from social-networking sites. For this project we need user

network *and* tag network data. A brief look around the world wide web shows that last.fm, flickr and delicious have such features, although the data needs some further processing - eg. inferring weight and direction for tag relationships, and working out which tags are more "general".

I will take increasing subgraphs of the final processed data set, and use it to generate networks with statistical properties defined by various schemes. Some interesting properties to experiment with might include:

**intra-index properties:** eg. how semantically unified a given index is (ie. to what extent are its tags mutually related), or how specialised a given index is (ie. whether its tags are more "general" or "specific").

**inter-index properties:** eg. the cumulative distribution of indexes with varying levels of a given property, the types of neighbours an index with a given property is likely to link to.

A collection of search requests will be simulated on each network generated, to test the performance of each scheme as it is applied to increasing network sizes. Failures and attacks will also be simulated, eg. by having some indexes deviate from the structure defined by the given scheme, giving false data, etc.

# Success Criteria

The project will have the following goals:

- To implement a versatile and extensible prototype of the system as a library.
- To obtain a representative set of realistic data, and to process it (as described previously) into a form suitable to use for simulations.
- To create a simulation suite that is flexible enough to run all the appropriate tests.
- To obtain a comprehensive set of test results on the performance of the routing algorithm under various conditions.

A successful outcome is one that accomplishes all these goals. The technically hardest part of the project will be to design a robust system, and to estimate optimal performance conditions for it. The most time-consuming part will be to design a comprehensive set of precise test conditions against which to evaluate the system.

## Resources Required

This project is unlikely to require any additional resources other than my personal computer. I will also use github as a backup service.

## Timetable and Milestones

As the project progresses, I will write much of the material required for the dissertation as extended notes. This should reduce the time needed at the end of the project to finalise it and prepare it for submission.

**2009-11-w0 - 2009-12-w0:** Design the specific details of the system, including the routing algorithm. Read up on research into similar systems to get a general idea of the issues that need to be addressed.

**2009-12-w0 - 2009-12-w2:** Implement the library. This will include the data structures and algorithms designed in the previous step, as well as a high-level node class that could be deployed in a real system.

**2010-01-w1 - 2010-01-w3:** Design the test suite, including selecting an underlying storage network to use, and algorithms for populating a test network with data.

**2010-01-w3 - 2010-02-w0:** Implement the test suite. This will be a detachable adapter module between my library and the selected storage network.

**2010-02-w0 - 2010-02-w1:** Implement data extraction code. This will involve writing a crawler and a parser for one of the websites noted previously.

**2010-02-w1 - 2010-02-w3:** Design the set of properties for which to generate test data. During this time, the data extraction code will be running.

**2010-02-w3 - 2010-03-w0:** Implement data processing code. This will take the data extracted previously and convert it into a form appropriate for my system.

**2010-03-w0 - 2010-03-w2:** Implement test data generation code. During this time, the data processing code will be running.

**2010-03-w2 - 2010-04-w0:** Generate test data and run tests. This should be a fairly straightforward and automatic process; during this time I will gather and structure my notes into a format appropriate for my dissertation.

**2010-04-w0 - 2010-04-w2:** Evaluate results and add this to the dissertation.

**2010-04-w2 - 2010-05-w1:** Finalise and proof-read dissertation and prepare it for submission.