IN-2194 Project: Uncensoring Twitter

1 General Information

Twitter recently announced that they were going to give governments the ability to censor tweets on a country-by-country basis.[1]

The goal of this project is to implement a Peer-to-Peer Twitter Censorship Detection Tool. The basic idea is that users worldwide would use the tool to monitor twitter feeds and compare. If a user fails to receive a tweet due to censorship, the network should find and obtain the missing message from peers in countries that do not censor it and publish the censored messages (on screen, in a special twitter feed using ROT13 encryption (to avoid the censor), on a website, ...).

This is a P2P project because of course your system should have no central server or services. A centralized design would likely neither scale nor offer good censorship-resistance. The basic idea is that if Alice in Germany follows 'Bob', Alice wants to be notified if 'Bob' is censored in Germany. So the P2P network should **find** users in the US, in North Korea, in China and in Denmark (in as many countries as possible) that are also monitoring 'Bob'. Then, if those users do see a tweet from him that Alice does not see, they tell Alice (who will then learn to be greatful to the German government for protecting her from Bob's evil messages).

You must submit your solutions using Subversion by **August 15th 2012**. Late submissions will receive a score of zero unless you notify the instructor about a valid reason (such as the entire group having been abducted by pirates) **prior** to the deadline.

1.1 Subversion Access

In order to receive Subversion access, you must submit a request at

`https://projects.net.in.tum.de/projects/tum2194/register`.

In this request you must include the login name and desired password. Your login name should be your lastname followed by the last two digits of your student identification number.

Once your account(s) have been created and teams have been assigned, you can access the repository at

`https://projects.net.in.tum.de/svn-ext/tum2194/TEAMNAME/`.

You must submit a request for account and team creation by **Mai 15th 2012**. We want you to use Subversion for development, not just submission! Note that only the version of your code that is HEAD in the repository on August 15th 2012 at 23:59 will be graded.

[1]`http://blog.twitter.com/2012/01/tweets-still-must-flow.html`

1.2 Coding Style

You should read the GNU coding standards for a reasonable guideline for writing good C code. Additional coding style guidelines are available in the GNUnet developer handbook. In general, a shorter and simpler implementation will receive higher marks than a complex and long implementation. Naturally, comments and testcases will not be counted against you when we evalute the size of the code (in fact, good testcases and comments may earn you style points).

Your implementation must not perform busy-waiting and must not use pthreads (!). You should also try to avoid (unnecessary) copying of data. You should stick to the GNUnet coding style as much as possible (if your implementation would be good enough to be merged without significant changes into the GNUnet code base itself, you will get full points).

2 The GNUnet Framework

Your implementation should work in the context of the GNUnet P2P framework. Specifically, you will be using the 0.9.x development branch which can be found at `https://gnunet.org/svn/gnunet/`. Some documentation is available online at `https://gnunet.org/`. The framework already contains a DHT, including a C API `src/include/gnunet_dht_service.h` that is used by other components to access the DHT and command line tools (`gnunet-dht-put` and `gnunet-dht-get`).

In GNUnet, each component is executed as a separate process. Your implementation will likely use the *core* or *mesh* service and/or the DHT to communicate with other peers. You can use the *peerinfo* service to bootstrap (discover peers initially).

Each of the services listed above provides a service access library (`libgnunetSERVICE.so`) to communicate with the service process that actually performs the operations. For this, the service process listens on a socket for requests from the service access library. Service processes are managed by the *arm* service.

For your own services, you need to write a service access library. For your implementation, you can choose to write the service process in C/C++ or Java. However, the Java API for some of the access libraries (to *core*, *dht*, *statistics* and others) is still under heavy development; thus the recommended development language is C/C++. If you choose to write your code in Java, you may have to deal with (more) bugs or other limitations in the existing access libraries.

Finally, you can ask questions not answered in the FAQ on the webpage in the `#gnunet` IRC channel on `irc.freenode.net`. We will answer non-FAQ questions in the public channel quickly. Developers in the channel speak English and German (and other languages); however, English is the preferred language.

3 Design

The design of the P2P protocol and your overall architecture is up to your team. However, you should include:

- Configuration options for users to specify the twitter account details (including country code) to use for monitoring
- A storage mechanism (text file, database) for accounts and topics your instance is monitoring
- Command-line tools to add/remove twitter accounts and topics that should be monitored
- A command-line tool to display censored tweets (continuously)
- Documentation on your P2P protocol (and why your design is efficient and how good it is at achieving its objective)
- A command-line tool to "fake" a missing tweet by injecting an additional "censored" message at an individual peer (so that this peer's code acts as if it got the message from twitter and thus all other peers should display the message as "censored")

4 Grading

You are expected to form teams of two to three students, but you can also work alone. Teams of four and more students are not permitted. Within each team, you are free to determine which team member is best for which task. Your overall grade will be determined primarily by the overall quality of the deliverable, but we will also discuss your contributions with you during the final demonstration and make sure that individual students are not penalized if some team members failed to contribute.

Grading will be based on the following main points:

8 Twitter interaction (receiving feed) correct

2 Fake-tweet injection by command-line implemented

4 Persistent storage for monitored accounts and topics is working (including command-line tools to manipualte)

20 P2P protocol is working and detects censorship (including command-line tool to monitor for censorship events)

4 P2P protocol documentation

4 Technical soundness of the design of the P2P protocol

6 Correctness of the implementation of the designed P2P protocol

2 Automated integration test

The project is thus worth a total 50 points.