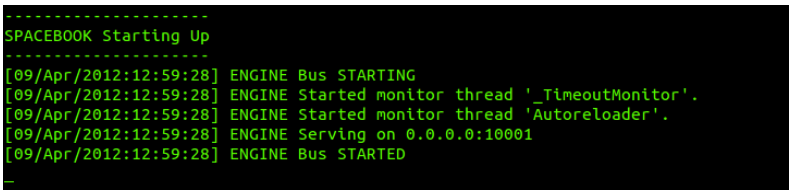
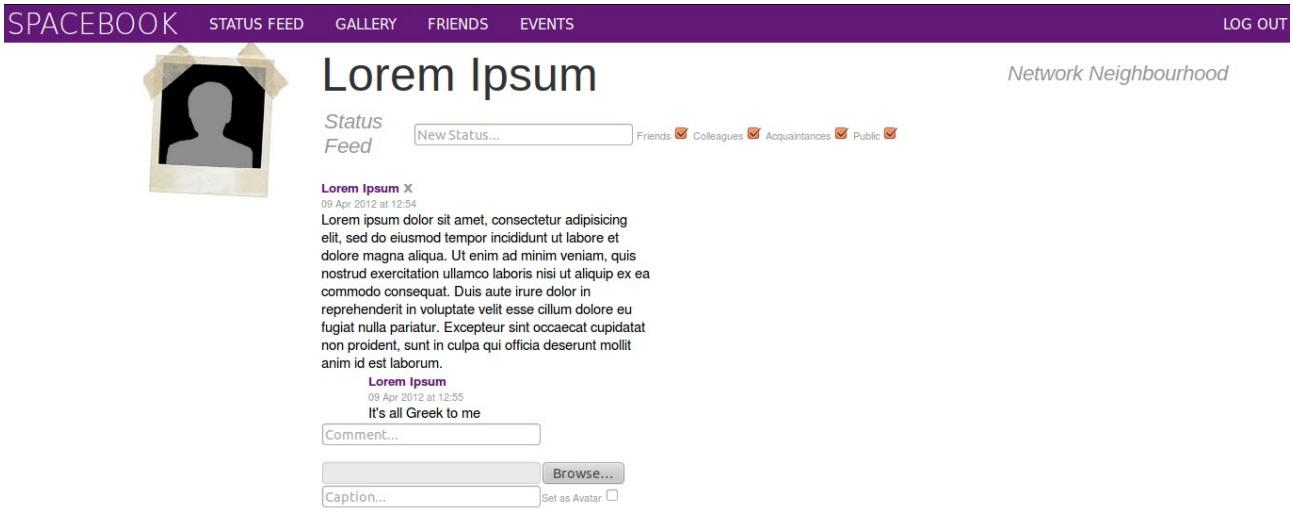
Timothy Bronson O'Brien
tobr017@aucklanduni.ac.nz
COMPSYS.302





SPACEBOOK

Concept Social Networking Server

Monday, April 9th 2012

Abstract

A customer has contracted a developer to create a prototype, peer-to-peer social network (SPACEBOOK). It was to be written using Python and Cherrypy, and was to be accessible everywhere, through a standard web browser. Protocols for intercommunication were defined by both an advisory body, and democratic voting. Protocols used for intercommunication between servers allowed for security compromises. The peer-to-peer implementation used is unable to be used on large scale. Developing the concept has led me to believe that, using these technologies, creating a large social network is not feasible, and not worthy of further investment or development.

I, Timothy Bronson O'Brien of the University of Auckland,
declare that this document and associated computer code,
are in their entirety, of my own work, unless otherwise referenced.

Timothy O'Brien
Monday 9th April 2012

Table of Contents

1. Introduction

The customer has contracted me as a developer to investigate the feasibility of a peer-to-peer social network. All of the users data is to be hosted from their own server (node), which users interact with through a standard web browser.

The SPACEBOOK concept was implemented in Python, using SQLite to save user data. I found these appropriate tools for the implementation of the prototype, and may be appropriate if a final version were to ever be produced.

Intercommunication protocols were largely decided upon using a democratic process. This was time costly, and parameters decided upon by a governing body greatly eased the development of code.

The peer-to-peer nature of the network means that scaling the number of participant nodes quickly degraded page refresh speeds. This was largely due to the non-threaded implementation of each SPACEBOOK node.

Although intended to be peer-to-peer, implementation is heavily reliant on a centralized address server. This slows drastically when user numbers are scaled, and undermines the peer-to-peer goal of the prototype.

The inter-communication protocols that are used are simplified in such a way that passwords are passed unencrypted in URLs. This is a security concern.

In order for a user to set up a server node on their personal computer, administration privileges, and knowledge of port forwarding, is required. This would impede users without relatively advanced technical knowledge, or without administration privileges.

Locally stored data is not encrypted by SPACEBOOK. This leaves the data on each node only as secure as physical access to a SPACEBOOK node. Furthermore, all computers on the network hold keys belonging to other users, which may be stolen and used maliciously. This renders the network only as strong as its least secure node. Changing of this would require redefinition of inter-communication protocols, and re-writing the way data is stored in SPACEBOOK's SQL databases.

Based on these findings, I deem the use of peer-to-peer technologies for social networking impractical on large scale, difficult to implement, impeding to novice users, and unlikely to become accepted for widespread use.

2. SPACEBOOK Design

2.1 Design Process

SPACEBOOK was written in Python, using the Cherrypy server library. SQLite was used for the storage of all user data, which is kept in three files. A statusfeed database holds all statuses, comments, and events of all users. An images database holds all images as Binary large objects, along with their associated metadata and caption. The third database holds all login credentials, first and last names for usernames, with their privacy credentials and relationships. Most functions are handled solely through SPACEBOOK.py with the exception of HTML code for the header bar, the CSS style sheet, gallery page, and new user verification.

If I were contracted to design SPACEBOOK once over, I would make several changes to its development and implementation. Many of these changes mitigate weaknesses with the current implementation.

The use of SQLite from the beginning. The initial use of text files for data storage proved inefficient and time consuming, then was proven to be ultimately pointless. Final implementation required the versatility of a SQLite database.

Separation of HTML from python source code, through the use of string replacement, or a templating library such as Jinja, alongside external .html files. HTML code began obfuscating program functions as the project grew to maturity.

Larger reliance on Javascript. The use of client side script was largely overlooked in the development of SPACEBOOK. Navigation and page updating would have been greatly streamlined through the use of Javascript.

Background threading. Pulling of updates only happened on a page refresh, which slowed page refresh times. The ability to scale SPACEBOOK is almost entirely dependent on implementing background threading.

Coding to transmit and receive push updates. This wasnt written because of time constraints by external factors.

2.2 Inter-communicational Protocol

The selection of most inter-communicational protocol parameters were made by democratic vote. This increased the length of decision times, resulting in stress where program segments had to be re-written, or developed very quickly, at the expense of the developer, quality of the final product, and efficiency of programmer hours. The parameters decided upon by increasingly smaller bodies of people proved to be decided upon faster. This allowed time for further iterative development to remove ambiguity.

Despite the large amount of time spent debating aspects of the communicational protocol, several issues still exist with it.

No encryption. Use of packet sniffing or other interception methods allow possibly sensitive data to be human read.

When authenticating, the user visits a page of a node with their unprotected key and username in the URL. This could be read straight out of a web browsers history log.

Ambiguity surrounding certain functions. After successful authentication, there is no agreed upon default page that users are redirected to.

With these issues, I don’t believe that users will be able to keep their personal information or communications private.

2.3 Implementation

With internet browsers ubiquitous, the use of HTTP and TCP was a natural interface choice for a social networking tool such as this.

SPACEBOOK nodes are interfaced to the internet using Cherrypy, a free open source web server library. Albeit poorly documented, Cherrypy proved to be appropriate for this design. The nature of a social network, being mostly dynamically generated page content with highly individual requests between users called for an implementation heavily reliant on server side computation.

The nature of a user hosted social network allows the use of expensive server side operations; this justified the use of storing all users images as BLOB data in SQLite databases. The use of SQLite for the web server was deemed appropriate because of the small volume of data that a single node has to manage and record.

In order to set up a SPACEBOOK node, the end user is expected to have access and understanding of router port forwarding. This is required to allow web traffic through each nodes' routers' firewall. This set-up process is both mandatory, yet inhibitory to inexperienced computer operators, or users without administration privileges. This will dampen mainstream acceptance of the SPACEBOOK concept.

2.4 Suitability of Peer-to-Peer Technologies

The current implementation is heavily reliant on a central address server. This is used to supply IP addresses of all nodes on the network, to every other node. SPACEBOOK nodes contact this on every login, and friends page refresh. Scaling this implementation bigger would result in large amounts of network activity between nodes and this central server.

Individual SPACEBOOK nodes currently request recent activity from all friends on each Status Feed refresh. With more than ten or so friends, this process will become fatally slow.

The need for most users to host their own server node is an environmental concern. A single centralized computer running a web server would use less data space, network bandwidth, and energy more efficiently.

For the reasons above, I see the use of decentralized, peer-to-peer communication as an impracticality for social networking.

2.5 Security

SPACEBOOK requires significant development before it can be considered even remotely secure.

At the current stage of development, there is a possibility some administration functions could be accessed by guessing URLs.

The image uploading box is not sterilized, and other file types may be uploaded into the SQLite database.

A protected view Server Error 500 page has yet to be developed, this may allow malicious attackers to further understand the source code of a SPACEBOOK node, and develop exploits.

Currently, HTML code may be injected into comments and statuses. This has the potential to mess with page formatting. Worse, this could be used to host malicious phishing content.

The current reliance on a centralized address server, coupled with the lack of control on the issue of usernames, it is possible to create a username the same as somebody else. This could cause unpredictable network behaviour.

It could be possible to create a denial of service attack against the centralized address server by declaring an undefined number of usernames. Using current implementation, failure of the centralized address server cripples all SPACEBOOK login operations.

The intercommunication protocol transmits unencrypted keys and usernames. These are often stored by friends' SPACEBOOK databases, rendering the network only secure as its weakest link.

These security exploits, along with problems with the protocol allow me to conclude that SPACEBOOK's implementation is highly insecure, and certainly not fit for the protection or communication of sensitive data.

3. Conclusions

The use of peer to peer technology to implement a social network is impractical. This is further exacerbated by holes in prototype implementation, and inter communicational protocols.

- Python and SQLite are appropriate tools for implementing the prototype.
- The ubiquity of web browsers make them perfect as an interface to SPACEBOOK
- Having an official body or select group allows more time and resource efficient decision upon inter-communicational protocols.
- Programming SPACEBOOK has improved my skills in using the tools involved.
- When scaled, the use of peer to peer methods for fetching recent activity becomes costly.
- Although intended to be peer-to-peer, implementation is heavily reliant on a centralized address server. This slows drastically when user numbers are scaled, and undermines the peer- to-peer goal of the prototype.
- Protocols currently require unencrypted passwords to be passed in URLs.
- Web serving, along with port forwarding, is not a user friendly process.
- User data is not encrypted at any stage, posing further security threat
- Every node holds login keys of all friends. These may be used maliciously.
- The SPACEBOOK peer-to-peer social network concept is a bad investment, not worthy of further development.