# Up Project Proposal

## Daniel Wilkins

### Project Overview

### Similar Work

The protocol has been implemented by various programs, dedicated to the protocol and not. Most notably [GNU Social](http://gnu.io), [Mediagoblin](http://mediagoblin.org), and [Mastodon](http://mastodon.social). It has also been implemented in programs such as [Friendica](http://friendi.ca). The primary differences between Up and these programs is simplicity: it should be both easy to deploy (ideally you build the program, do something) like ./up init && ./up start and then you tell your web server to proxy it.

### Previous Experience

I have worked on several web apps for my own use, pastebins, image galleries, wikis, and the like. Similarly I've dealt with XML and JSON parsing in, for example, my podcast fetcher. Finally, I've administered GNU Social instances since before it was called GNU Social, probably the past 5 years or so by my estimation.

### Technology

* Go: the only real risk factor in terms of technology, but given that I've been following Go since its announcement and have played around with its predecessor, Limbo, I am fairly confident that I can pick it up quickly.
* SQLite: For the database and anything which won't fit into a config file. Needed to store messages, follows, following, etc. Handled by Mattn's [go-sqlite3](http://github.com/mattn/go-sqlite3) library. By all means it looks like SQLite bindings for any other language, just in Go.
* OStatus: This implies various other (but fairly simple) protocols such as Salmon and PuSH (although the standard does make Salmon optional). The individual standards themselves appear fairly simple, however, with some essentially just defining an xml schema or some other file format (like webfinger). Essentially it's a lot of XML over HTTP, both of which Go supports in the standard library.
* XML: I'm not exactly sure that this qualifies as a technology, but most of the standards in OStatus involve XML in one way or another; I will be reading an awful lot of dumps in that format. XML support is in the Go standard library.
* Webfinger: A simple protocol that defines a special URL and a JSON scchema, essentially; webfinger.net recommends [ant0ine's go-webfinger](http://github.com/ant0ine/go-webfinger) for client work; in this context on the server side the webfinger information would be a static file to be served.

### Risk

* (Small) Go doesn't work for me: I doubt this will be the case but I should be able to move to another language I'm more familiar with without trouble, even if it won't be quite so efficient. My specific backup is Perl and Mojolicious, a web framework I am at least somewhat familiar with.
* (High) Standard Drift: Many of the RFCs listed in the OStatus specification were hosted on Google Code, but are accessible from Wayback Machine. It may be that implementations have drifted considerably from the standard, and so OStatus would now be a de facto standard with a lot of internal knowledge needed. Thankfully it is fairly easy to contact those who are implementing the protocol these days via OStatus itself and via IRC, not to mention email. I know of at least one other person who is creating their own OStatus server right now, for example. To caution against this, I would like to get federation working as quickly as possible, at least in a read-only form; this way I have the maximum amount of time to squish bugs in it.
* (Low) Lightning strikes my server. Up would be running on the server in my guest bedroom and if there were a catastrophic hardware failure I would need to try to move to AWS or similar, which would take time becuase I will mostly be working with real world data. If there *is* a catastrophic failure, I would probably move to AWS or another hosting service which gives students free access and create an account on a public instance to create test data.