

Proposal:

Drupal is an amazing, flexible Content Management System that, on its own and through modules can present and store a large variety of information. This data can be in the form of a blog entry about one's weekend or it could be a high resolution image of our solar system. With Drupal, a vast array of information and data can be shared within a community. With the advent of broadband, the sizes of files available for download on the internet have increased dramatically. Files that are multiple gigabytes in size can be downloaded in under an hour. The increased bandwidth offered by broadband comes with a price to some website owners - the bandwidth that their site uses. For example, suppose someone has a gallery of high resolution photos of the planets in our solar system and each one is about 200MB in size. Each time someone attempts to download one of these images, they use 200MB of the website's bandwidth. This assumes the download completes without error, if an error occurs the file would have to be re-downloaded, using another 200MB of bandwidth. Should his or her site have a high volume of traffic, the bandwidth used by the file downloads alone will exceed the bandwidth used to simply display the galleries of images. BitTorrent is a solution that will decrease the amount of download bandwidth used, increase the download speed for his clients, and ensure download quality.

Through the BitTorrent protocol multiple *peers* downloading the same file request different pieces of that file from either peers who already have the piece they need, *seeders* (peers who have all of the pieces of the file), or the server (acting as a seeder). In this method, as the peers download pieces of the file they are uploading pieces that they already have to other peers. Peers and seeders determine who to contact by contacting a *tracker* for their download. The tracker does what its name implies, it tracks. Clients contact the tracker to inform the tracker of its progress and receive a list of current peers to contact and from whom to attempt to download pieces. Once a client receives a piece of the file, it is checked against a known hash of that piece to ensure download quality. The URL of the tracker, information about the file or files, and hashes of the file pieces are contained within a *.torrent* file. The user opens the *.torrent* file with their BitTorrent client and the client contacts the tracker to receive an initial list of peers to contact and from whom to request pieces. The advantages of BitTorrent are best summarized by Wikipedia (<http://www.wikipedia.org>), "BitTorrent is a method of distributing large amounts of data widely without the original distributor incurring the whole of the corresponding costs of hardware, hosting and bandwidth resources." Wikipedia also has an excellent visual example of the BitTorrent protocol located at http://en.wikipedia.org/wiki/Image:Torrentcomp_small.gif.

The tracker portion of the BitTorrent protocol is really simple. First, a request is sent to the tracker (usually to an "announce" file), this request contains info hash (SHA1 of the are of the torrent meta file), peer id (unique id of the peer), IP Address (optional, contains the IP Address of the peer or DNS name), port (the port on which the client is listening), uploaded (the total amount uploaded in base 10 ASCII), downloaded (the total amount downloaded in base 10 ASCII), left (The number of bytes the peer still has to download in base 10 ASCII), and event (optional, set to 'started', 'completed', 'stopped', or '[empty string]'). Once the tracker receives this data, it verifies that it is indeed tracking the torrent with the same info hash (usually this information is within a table). After that, the data for uploaded, downloaded, and left are placed into a statistics table.

Depending on the event provided by the client, the tracker responds accordingly. If the client sends an empty event or no event, the tracker responds as though this is a regular announce. The response contains two keys- the interval (time to wait until next announce) and peers, a list containing dictionaries (the dictionaries contain peer id, IP Address, and port). Should a 'started' event be sent, the tracker adds the peer to its tables and responds with the standard tracker response. If the event is 'stopped,' the tracker removes the peer from the peers table by their peer id. Finally, if the event is 'completed,' the tracker sets the peer as having the entire download in its peers table. Additionally most of the trackers used today keep statistics, such as how many people are seeding and leeching, and this data is usually stored in another table separate from the peers table. The whole process can be summarized as follows:

1. The client sends its information to the tracker.
2. The tracker receives data from the client.
3. The tracker verifies the received data.
4. The tracker performs appropriate database queries based on the event value (or lack of value).
5. The tracker returns a list of peers who are currently active.
6. The client continues contacting peers and downloading from them.

The portion of the protocol that this project is concerned with is parts 2-5. Another convention being used is the "scrape" convention. This page accepts one or more info hash parameters. The scrape page then returns a list of dictionaries (one for each file) with the following information:

- Files: a dictionary where the key is the 20-byte info hash and the value is a nested dictionary:
 - Complete: The number of peers with the entire download (seeders)
 - Downloaded: The number of times the tracker has received the "completed" event
 - Incomplete: The number of peers who do not have the entire file downloaded (leechers)
 - Name: The name of the torrent as specified within the torrent file (optional)

Through the announce and scrape conventions, the client can request a list of peers to contact or just the statistics for the current torrent.

Drupal currently has two BitTorrent Tracker modules, one of which is not publicly available for download due to piracy concerns, and the other I have reviewed and determined areas of improvement. Portions of the code are completely reusable, such as the bencoding and decoding functions. The areas in which the module could be expanded and improved upon cover everything from SQL server interaction to the logging system. The focus for this proposal is to adjust the current tracking solution to be more user and server friendly, as well as provide additional functionality. My proposal focuses on minimal database interaction and user (Drupal administrator) intervention. The current module offered to Drupal users is heavy in database interaction and logging. I am not against logging and heavy SQL usage, but should the tracker be used on any sort of large scale environment, the log file alone has the potential to grow exponentially, along with the large number of database queries to process a single client request of the tracker. Also, the module does not support the scrape convention appearing in the current

BitTorrent clients. As far as Drupal integration is concerned, there is no interface for Drupal, the module is completely independent of the Drupal CMS. Below, I outline my plan for improvements and additions to the BitTorrent tracker module for Drupal.

To begin improving, I will group all of the queries used during a single request to the tracker. Whether the user is requesting a piece or reporting a completed download, there are currently too many calls to the SQL server. There are many calls to the server requesting the same information *or* requesting different pieces of information from the same row in the table. In order to reduce the amount of queries, as soon as the tracker receives the request, the “type” of the request will be determined. Whether it is an actual “request” or a “completed” message, the tracker will split into a specific bracket which contains focused SQL calls. The bracket’s SQL calls will be tailored specifically to the “type” of request the client used. This generic SQL statement will request most of the information required to process the client information. Rather than making a call and determining what to request from there, all the information that is used will be requested at once and manipulated/stored within the PHP file. This method should reduce the number of SELECT and UPDATE statements required to process a single client request.

The BitTorrent module also makes use of heavy logging to store **everything** that the tracker does. I propose to replace the log file with a SQL table. This seems to contradict the premise of light SQL usage, but with the use of an SQL table, the logging for the request/response may be performed with a single statement at the end of the request. Currently, the log file has a write after each interaction with the SQL server. This will be replaced with a single update at the end of the request (after the response). This presents two distinct advantages. The only true need for a log file is for SQL errors and the ability to keep limited logs. Suppose the user wants to keep logs for a 48 hour period and remove the rest. By using a timestamp field in the SQL table, a single DELETE command can remove all of the logs that do not fall within the 48 hour period. Along with the advanced control over the logs, the Drupal Administrator may search the logs through a web interface rather than dealing with having to access a log file on the server.

Security is another issue with the current tracking solution. While the current solution escapes some strings, a large portion of the strings sent to the SQL server go unchecked or even un-escaped. Escaping and typecasting all data being sent to the SQL server would greatly increase security. A feature present in phpMyBitTorrent checks the user agent to block web browsers from accessing the announce and scrape files. Implementing a similar solution that checks the user agent would block malicious users trying to issue database commands from their web browser and provide valuable statistical data. The IP Address of a user using a web browser could be placed in the logs allowing the administrator to block the IP Address of a malicious user as well as track which BitTorrent clients are used on the tracker. Through proper escaping techniques and data verification, the Drupal environment is kept safe from malicious users and beneficial statistical data is collected.

Besides security the documentation for the BitTorrent module in its current state is a little lacking. That is not to say it does not exist, but functions are not fully documented and what *is* there doesn’t really explain what is happening in the code. I would like to expand the documentation to be standardized above each function, defining the input and output clearly along with a clear division of the function areas and main execution portion. Implementing a solution similar to JavaDoc would help make the code

more readable and aid in helping developers extend, repair, and remove code. Improved documentation methods and a change log at the start of each file will drastically improve future code modification and understanding.

In an effort to make things easier on the Drupal user, the install process should be simplified and an administrator interface will be created. A single install.php file should be created which prompts the Drupal user for all of the configuration settings along with defaults for a basic installation. This file will create the database tables, write the configuration files, and adjust the .htaccess file, provided the user allows it (adding the appropriate Rewrite rules in the appropriate locations). This “one-click” install will allow Drupal users the ability to integrate the module into their environment easily. The administrator interface within Drupal will make changing settings easy for anyone, such as adjusting the table prefixes and log files settings. This interface should have an area to display statistics and possibly export them into an XML format to display to guests on other pages. Also, the administrator should be able to upload a torrent file (through a file field) and have the tracker detect and begin tracking it.

Updating and optimizing the tracker module for Drupal will alleviate bandwidth usage on websites with large file downloads and high traffic, as well as increase the security of the file being downloaded (through hash checking). The goal of this project is to implement an advanced BitTorrent Tracker module for Drupal. It will consist of announce and scrape files for the “invisible tracker” portion of a BitTorrent tracker, a Drupal module that manages current torrents and IP Addresses as well as live tracker statistics such as the number of seeders and peers. Provided there is additional time, the module should also have an option for *Server Side Seeding* should a torrent not have any seeders. Judging by the requests from the community, a BitTorrent tracker module would definitely have a niche among other Drupal modules.

Drupal benefits-

- A simple interface to create and track torrents
- A module that can easily be implemented in a variety of Drupal environments
 - Focus on ease of installation and minimal database interaction
- Adding distributed downloading technology to the Drupal CMS

Requirements to Succeed-

- The module must be cross platform and browser compliant.
 - The tracker should work on IIS and Apache with PHP 5 and MySQL 5. (I will also check for compatibility with PHP 4 and MySQL 4)
 - The module interface must work in all major browsers.
 - IE 6 & 7
 - Gecko based browsers
 - Opera 8 and 9
 - Safari
 - The tracker must interface with most major BitTorrent Clients (hopefully all that implement the protocol).
 - Azuereus
 - BitTorrent

- μ Torrent
 - BitComet
 - BitLord
 - BitTyrant
- Ease of installation
 - It must be easy to install the module and set up the tracker.
- Ease of use
 - The user must be able to ban IP Addresses (or ranges).
 - The user must be able to add and delete torrents.
 - The user must be able to view and reset statistics.
 - Provided there is additional time, the server should automatically seed a torrent if there are no seeders available (possibly only the pieces that are not available).
- Security
 - The tracker must not have the ability to cause malicious damage to the web server / SQL server.
- Quality Coding
 - There must be flexible code design to aid in future development work.
 - There must be quality documentation to help others new to the module understand its inner workings.
- Low-impact implementation
 - The tracker must not cause significant load on the servers it uses.
 - The tracker must occupy a small hard disk footprint.
 - The tracker interfaces must comply with current web standards.

Timeline March 25th – August 20th

This project will take most of the time allotted to maximize quality code and minimize errors. I will use the interim period to begin community involvement and generate optimizations to the current solution. I have the cooperation of the current author of the BitTorrent Drupal module (<http://drupal.org/node/31435>). The module meets some of the requirements to succeed, but there is major work to be done as far as SQL interaction and Drupal integration.

March 25th – May 28th

This section will focus on gaining an understanding of the current solution and identifying its weaknesses. I will also become involved in the community and set up my test environment. The current solution needs some major reformatting to make it the current code readable and aid in extending it.

- **COMPLETED:** Create Drupal account
- **COMPLETED:** Contact the current owner of the BitTorrent module for Drupal 4.6 to secure co-operation
- **COMPLETED:** Research of the current BitTorrent module
- **COMPLETED:** Create test environments
 - <http://www.soc.deviantolutions.net/> (Drupal 5, PHP 5)
 - <http://www.soc-4.deviantolutions.net/> (Drupal 4, PHP 4)
 - Both have the current module installed.

- **ONGOING:** Become active in the Drupal Community
 - Join IRC and forum communities to both introduce myself and learn more about the Drupal project and environment.
- **IN PROGRESS:** Research other PHP based BitTorrent trackers
 - phpMyBitTorrent
- **IN PROGRESS:** Research the BitTorrent Protocol
 - Includes, but is not limited to:
 - Client – Tracker Requests: Announce and Scrape data, what the tracker actually receives from the client and what that information translates to in relation to the torrents.
 - Tracker – How the tracker responds and what data is sent back to the client.
 - Torrent File Semantics – The current specification for the content of Torrent files.
 - Server Seeding or WebSeeding
- **IN PROGRESS:** Research and evaluate current tracking solutions:
 - **IN PROGRESS:** Become familiar with the current Drupal BitTorrent tracker module and perform testing to determine where optimization and work could be done.
- **IN PROGRESS:** Extend current documentation and separate current code segments to facilitate future development and optimizations
- **IN PROGRESS:** Begin optimizations to the current module
- **IN PROGRESS:** Generate database schema
 - Determine number of total queries needed by the tracker to minimize database interaction
 - Concentrate on minimal database interaction
- **May 28th – July 9th**

During this time I will focus on the continuing optimization and re-factoring of the current module. The tracker should be nearing completion at the end of this period, the database schema will be finalized with the logging portion of the tracker up and running (includes searching logs).

 - Optimize current announce.php file and create scrape.php
 - Rework the logging system to utilize a SQL table instead of a log-file
 - Allow browsing and searching of the logs
 - Allow removal of items from the logs
 - Allow limits to be placed on the logs (limit on the lifetime)
 - Continue Tracker optimization.
 - Finish splits for request types
 - Implement generic SQL calls
- **July 9th – August 20th**

The final portion of time will focus primarily on finishing the tracker portion, testing it with multiple clients, and creating the user-interface. A variety of clients on different platforms will test both the tracker and the interface. This final period of time will devote major attention to quality assurance, not just when displaying information to the user or tracker, but also in coding standards, verifying that the code

is easy to follow and understand, allowing for future updates and changes to be made with ease.

- Finish any work on the tracker
- Test/verify the “invisible tracker” portion of the project between multiple client solutions
- Develop Drupal module interface
 - Create the input and display pages for the project
 - Installation files
 - Management files
 - Statistics Display files
 - Interface to browse torrents
- Develop management / statistics gathering portion of the Drupal module and test interaction with the tracker tables
 - Test adding and removing torrents
 - Allow the user to upload a torrent and have the tracker verify it, and begin tracking the torrent. If the tracker is not listed in the torrent, update the torrent to include the trackers information.
 - Gather statistics (number of torrents, seeders, lechers, swarm, etc.)
- Test the interface within Drupal
 - Verify that the web based portion of the project works in all major browsers without error
- (Provided there is time) Work on “extras” after core portions of the project are finished and tested
 - “Torrentizer” – Create torrents for files that meet certain criteria.
 - Server Seeding – The web server provides the initial seed or seeds pieces of the torrent which are not currently available (this may be moved to a core part of the project, but only after other work is completed)

Biography:

My name is Christopher Bradford and I am twenty years old. I am an Information Systems major at Virginia Commonwealth University located in Richmond, Virginia, USA. I am currently employed by Virginia Commonwealth University Libraries as a Web Application Developer for the Information Systems department. Presently I am pursuing a Bachelor of Science in Information Systems and a minor in Computer Science. I have been programming for six years in a variety of programming languages including Java, JavaScript, HTML, XHTML, CSS, PHP, Perl, and ColdFusion. My most recent projects include the implementation of the redesign of the VCU Libraries website (<http://www.library.vcu.edu>) and their staff directory (http://www.library.vcu.edu/cfapps/nts/ulmdir_action.cfm?dept=all). I maintain a regular blog at <http://blog.vcu.edu/bradfordcp/> which contains snippets of code I am currently using. I am familiar with many operating systems and environments from a UNIX shell to OS X. I have a passion for coding and creating usable web applications. Last year, I created a conference management system for registration and processing of conference attendees that was written in PHP with a MySQL backend. If I am creating an application

and I come to a problem, I will ask for help from others and perform further research until I can fix the issue or come up with another solution. I am proud that I have never given up - I keep working and looking until I find an answer to my problem and come up with a solution that fixes my problem.

BitTorrent greatly interests me on many levels- I have always been interested in network applications and the sharing of data between two people through a network. I have used BitTorrent for a few years now, mostly to download Linux ISO files which can range from 600MB to 4GB. One thing that intrigued me was not only how the pieces were sent between peers, but how the peers “discovered” each other. After some research, I thought trackers were neat in the way they connect peers. Following the advice of my supervisor, I started to look at Google’s Summer of Code. I started randomly clicking through projects’ idea web pages until I came to Drupal’s idea page and a suggestion had been made to implement a BitTorrent tracker module within Drupal. I jumped at the chance to work in Summer of Code as well as develop a tracker for the BitTorrent protocol. I have experience with PHP and MySQL, both of which are used by Drupal, and I believe I have the drive and determination for this project. I am floored by the idea of working on something of this scale. I have the opportunity to work on a project that genuinely interests me and on top of it all; my work will benefit an open source project. I have the skills necessary for this project along with the passion and drive to implement such a solution.