The lobber architecture

Lobber is a data distribution service based on BitTorrent technology and federated authentication and authorization. Lobber consists of a few collaborating parts:

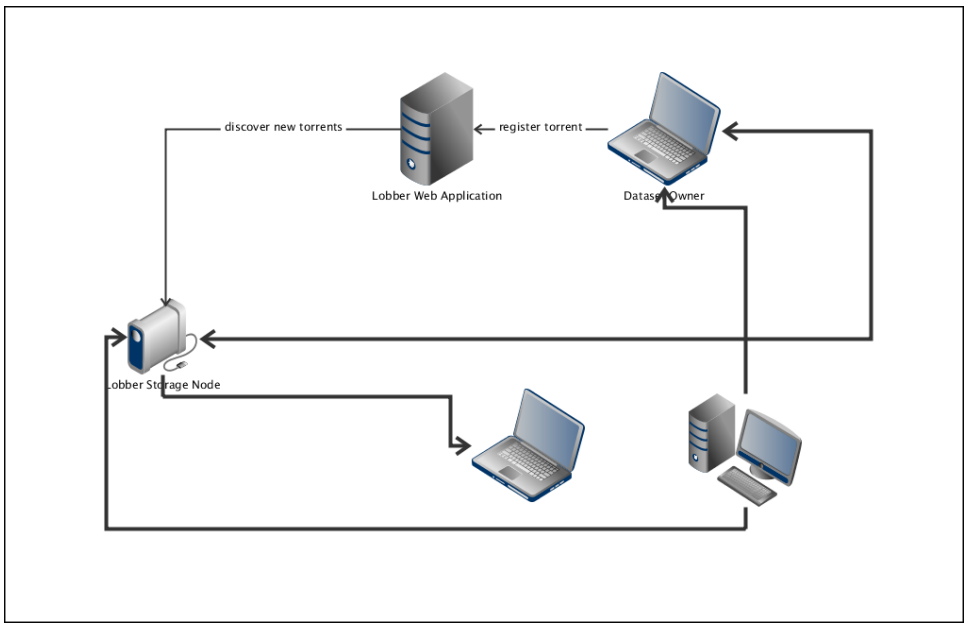
- The Lobber web application
- A messaging and notification mechanism
- Lobber clients (aka "storage nodes")
- A java web-start Lobber Client for easy upload of torrents
- Regular BitTorrent clients

One of the design goals of Lobber has been to do as little *damage* to the BitTorrent protocol as possible, thereby enabling as many regular BitTorrent clients to work unmodified. Instead Lobber implements a modified tracker and torrent registry in order to achieve the following goals:

- Enable a user to securely share a dataset with other users.
- It must be possible to limit access to datasets based on user-defined groups.
- It must be possible to use a normal BitTorrent client both for sharing and for downloading datasets.
- An API is needed for integration with other systems

Dataflow

The following picture illustrates the flow of data in Lobber:



There are two types of data involved: torrent files that describe data about datasets and the datasets themselves. In order to share a dataset a user creates a torrent file - either directly using a BitTorrent client or by using the Lobber java webstart client launched from the Lobber web application. The torrent-file is a relatively small file which is uploaded to the Lobber web application. The dataset itself is *served* from the users BitTorrent client (eg the java webstart client or the Lobber dropbox application).

As soon as the torrent file is registered with the Lobber web application a notification is sent out over the Lobber notification bus. Currently we use stomp but there are plans to replace this with pubsubhubbub. Information about new torrent is also available as RSS feeds (ATOM will also be available when we change to pubsubhubbub for notifications).

At this point one or more Lobber storage nodes will receive the notification and start to download the dataset from the users BitTorrent client (the choice of storage-node follows the lobber authorization model). As soon as a storage node finishes the download the client can stop serving up the dataset.

Storage nodes are special in the sense that they are trusted to keep and *retain* datasets as opposed to regular BitTorrent clients that are not expected to be available. In the larger BitTorrent community the problem of dataset availability is typically solved by so called ratio limits which ensure that users keep sharing data they download. This model does not work for Lobber since we need to guarantee the existence of *every* dataset and not just the average availability of datasets in general.