Cam's Unix Notes, 4 — **Dropbox file storage quotas, from a networked file-system perspective**

Dropbox (DB; dropbox.com) is "*a file hosting service operated by Dropbox, Inc., that offers cloud storage, file synchronization, and client software*" [1]. Many folks use DB as a convenient solution for portable and effortlessly synchronized file-storage across a spectrum of personal devices (workstations, laptops, smart phone, *etc*.). From the end-user perspective, a drawback of the DB approach (based on a freemium business model) is the *folder sharing* policy, with respect to a user's free storage quota; the folder-sharing functionality is precisely one of the most useful features of the DB system, which is what motivates this whitepaper.

Consider the following scenario: User Alice creates folder 'apples' in her DB space, and then *shares* (in DB terminology) the folder with user Bob. The folder counts against Alice's DB quota (*expected behavior*), and it also counts against Bob's DB quota (*annoying behavior*). This policy seems to be in-place as a workaround that enables the DB system to prevent a felonious user, say Charlie, from making unlimited free storage for himself by creating free DB accounts Charlie1, Charlie2, …, Charlie*N*; more information on this issue can be found online [2].

For users on a distributed Linux file-system, for instance using the popular *Network File System* (NFS; http://en.wikipedia.org/wiki/Network_File_System), a solution can be devised that avoids the 'apples/' folder from being counted against two quotas, *i.e.* against both the sharer (Alice) and the sharee (Bob). Let's begin with a primary NFS server foo.bar.org that defines a local area network comprised of NFS clients hello.bar.org, world.bar.org, *etc*. Then here's a simple solution to the multiple-quota problem:

1) Create a Unix user account 'dropbox@foo.bar.org' on the NFS system.
2) Create a user account 'dropbox@foo.bar.org' at Dropbox (does not have to match the name from step-1, it's just simpler to organize things that way).
3) Logged into a shell as Unix user 'dropbox', issue the command "mkdir $HOME/Dropbox" to create the desired storage root for DB on your local filesystem. Most important, make sure that permissions on that directory (chmod) are suitable for your purposes (*e.g.*, group-readable/writable or whatever).
4) Spawn a dropbox daemon process on one of your networked workstations. The process can be set to run on foo.bar.org or on any other workstation in the NFS network – which exact workstation is immaterial because they are networked(!) and all NFS clients can therefore see the user 'dropbox' $HOME. To do this, download a Python script available from DB [3]; store the script in, say, $HOME/utils/.; and launch it in the background *via* "python $HOME/utils/dropbox.py &". You may wish to specify a cron job to periodically insure that it's running, in case of inadvertent reboots and such.
5) The tedious part: Now *carefully* migrate any of your existing shared materials to the new, local DB home (step-3), and re-invite sharees (do this from within the dropbox@foo.bar.org DB account). This is easily achieved on Linux filesystems using standard built-in commands such as mv, mmv, scp, rsync, and so on.
6) Finally, as the dropbox@foo.bar.org DB administrator, you can now advise your sharees of two options:
   i. ***Opt-in***: An individual can opt to have the DB folder shared with them in the traditional manner – the DB folder is officially *shared* with the user (through the DB web interface), the user will receive auto-notifications from the DB client running on their desktop, and so on. Advantages of this approach include convenience and familiarity, and the ability to access the shared folder *via* multiple means (DB's web interface, iPhone app, *etc*.). Of course a disadvantage of this approach, and the main motivation for the solution presented here, is that the shared space counts against the individual's DB quota.

      Alternatively…

   ii. ***Opt-out***: The individual can opt to *not* have the folder auto-shared (*via* DB), but rather manually access the files over the NFS network. The advantage of this option is that the 'shared' (*accessible* is a better word for this model) folder will *not* show-up among the individual's shared DB folders (online, in

the DB web interface) and therefore will *not* count against the individual's quota. This property is also the only disadvantage of this approach, in terms of easily accessing DB folders *via* mobile devices, web interfaces, *etc*. Regardless, with this second option one can still get/put all the content in the NFS-accessible DB folder as though it were being shared via DB. Again, this is feasible because of the networked file-system, together with tools such as rsync (Linux) and WinSCP (Windows).

References

[1] from http://en.wikipedia.org/wiki/Dropbox(service)
[2] https://www.dropbox.com/help/59/en
[3] https://www.dropbox.com/download?dl=packages/dropbox.py