Name: Milap Bhojak

University: Sankalchand Patel College Of Engineering/Gujarat Technological University

Major: Information Technology

Expected graduation date: June 2015

Degree: Bachelor of Engineering

Home Page/Blog: http://milapbhojak.com/ , http://milapbhojak.wordpress.com/ , http://milapbhojak.quora.com/

Email: milapbhojak.exe@gmail.com , milap@milapbhojak.com

Telephone: 00-989-885-5075

Freenode IRC handle: milap

Title : FTP Mirror Syncing (Re-architect OSL FTP Mirror Infrastructure)

Abstract :

The OSL hosts a three-node FTP mirroring cluster to host various FOSS projects' ISO images and package databases. Currently it comprises of some custom-made bash scripts that utilize ssh, rsync and cron to coordinate the syncing process. The architecture is a single master machine that syncs from upstream and then triggers slave hosts to sync from it. This architecture has served us well over the years but has limitations and needs updating to use more modern features.

There are a variety of methods for syncing the repositories but generally most use rsync. Some have an ssh trigger system (i.e. Ubuntu/Debian) and others want a post script to run to provide results upstream. Finally sometimes we are the master mirror, meaning we are the starting point, so some projects either manually upload and trigger a sync or we pull from a central source using cron.

In addition, we have some simple monitoring that should be updated. Ideally it would be re-implemented as some type of dashboard showing any repo that is out of sync, disk usage per repo over time, and some other simple statistics.

Let Me Explain you something in easy way.

Who is the application for?

Application for Re-architect OSUOSL FTP Mirror Infrastructure.

What Problems will it solve?

Proposed system will optimize FTP mirror infrastructure using Push Message system. It will have significant impact of sync process and sync traffic. It will also provide higher control over ftp infrastructure.

Where will it be used?

System can be used in organization where users required to updates file stores very frequently, which forces mirror server to be synced so frequently

When is it needed?

Proposed system will be efficiently used when number of file/project updates are high, which created high amount sync traffic.

Why is it needed?

Proposed system is needed to optimize sync traffic between master and slaves, and to also insure all mirrors are having latest copy of files from master server.

How it will Work?

So, Here we will use Google PubSubHubbub Protocol.

What is Google PubSubHubbub service?

Google PubSubHubbub is an open protocol for distributed publish/subscribe communication on the Internet. Initially designed to extend the Atom (and RSS) protocols for data feeds, the protocol can be applied to any data type(i.e text, pictures, audio, video) as long as it's accessible via HTTP. Its main purpose is to provide real-time notifications of changes, which improves upon the typical situation where a client periodically polls the feed server at some arbitrary interval. In this way, PubSubHubbub provides pushed HTTP notifications without requiring clients to spend resources on polling for changes.

Under PubSubHubbub, there is an ecosystem of publishers, subscribers, and hubs.

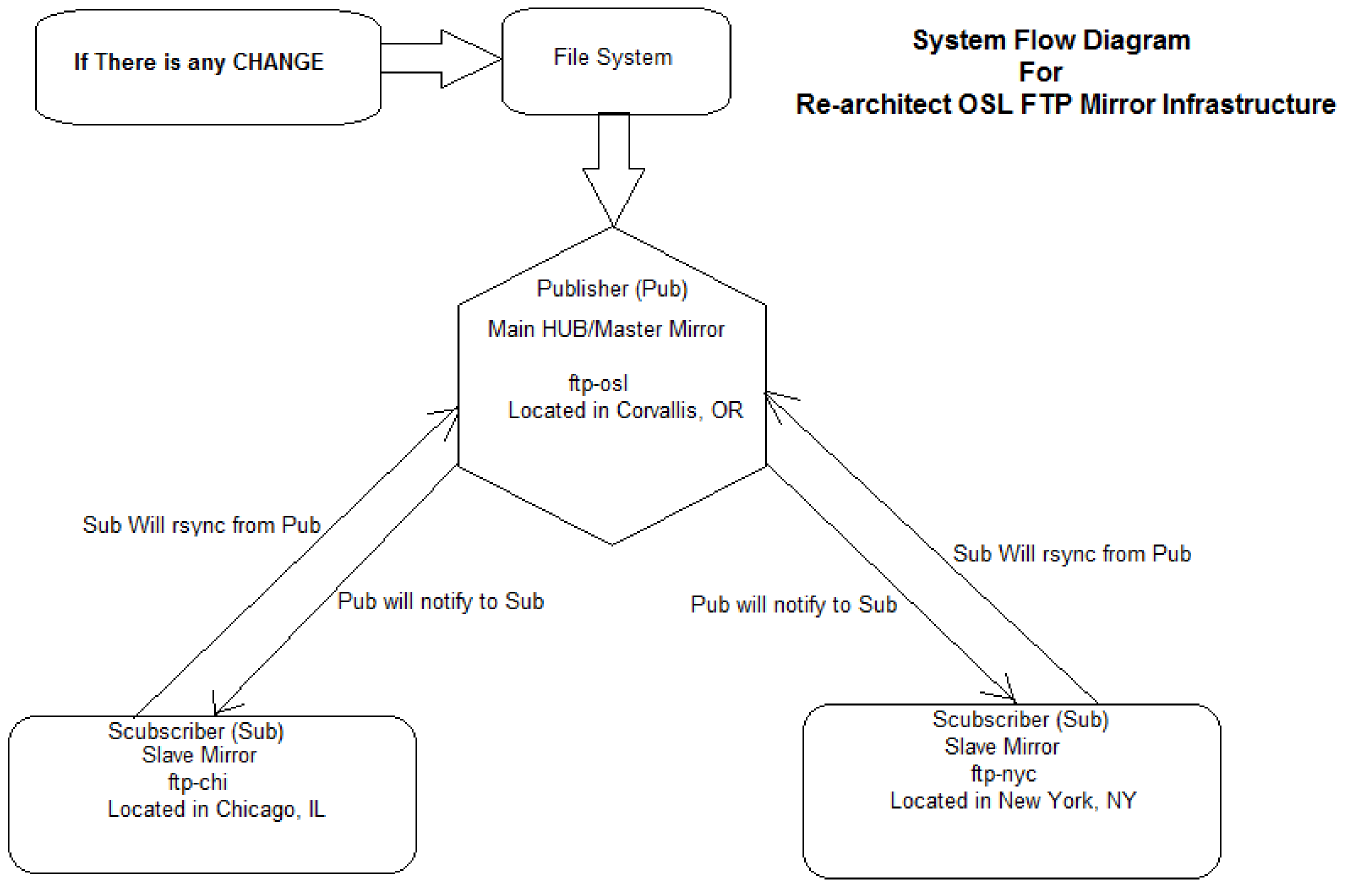
A subscriber first retrieves content from a HTTP resource (URL) by requesting it from the webserver. The subscriber then inspects the contents of the response, and if it references a hub, the subscriber can subscribe to that resource's URL topic[clarify] on that hub. The subscriber needs to run a web accessible server so that hubs can directly notify it when any of its subscribed topics have updated, using a webhook mechanism.

Publishers expose their content with the inclusion of hub references in the HTTP headers. They post notifications to those referenced hubs whenever they publish something. Thus, when a publication event occurs, the publisher calls its hubs and the hubs call their subscribers.

Pubsubhubbub includes a simple verification of intent mechanism in order to prevent abusive subscriptions, and a validation mechanism allows for subscriptions to private or protected web resources. When the subscriber sends the subscription request to the hub, the subscriber address and a code needs to be included. The hub immediately sends a verification message to the subscriber with the URL of the topic and the above code. The subscription request will only be accepted, if the subscriber sends a positive response to the verification request of the hub.

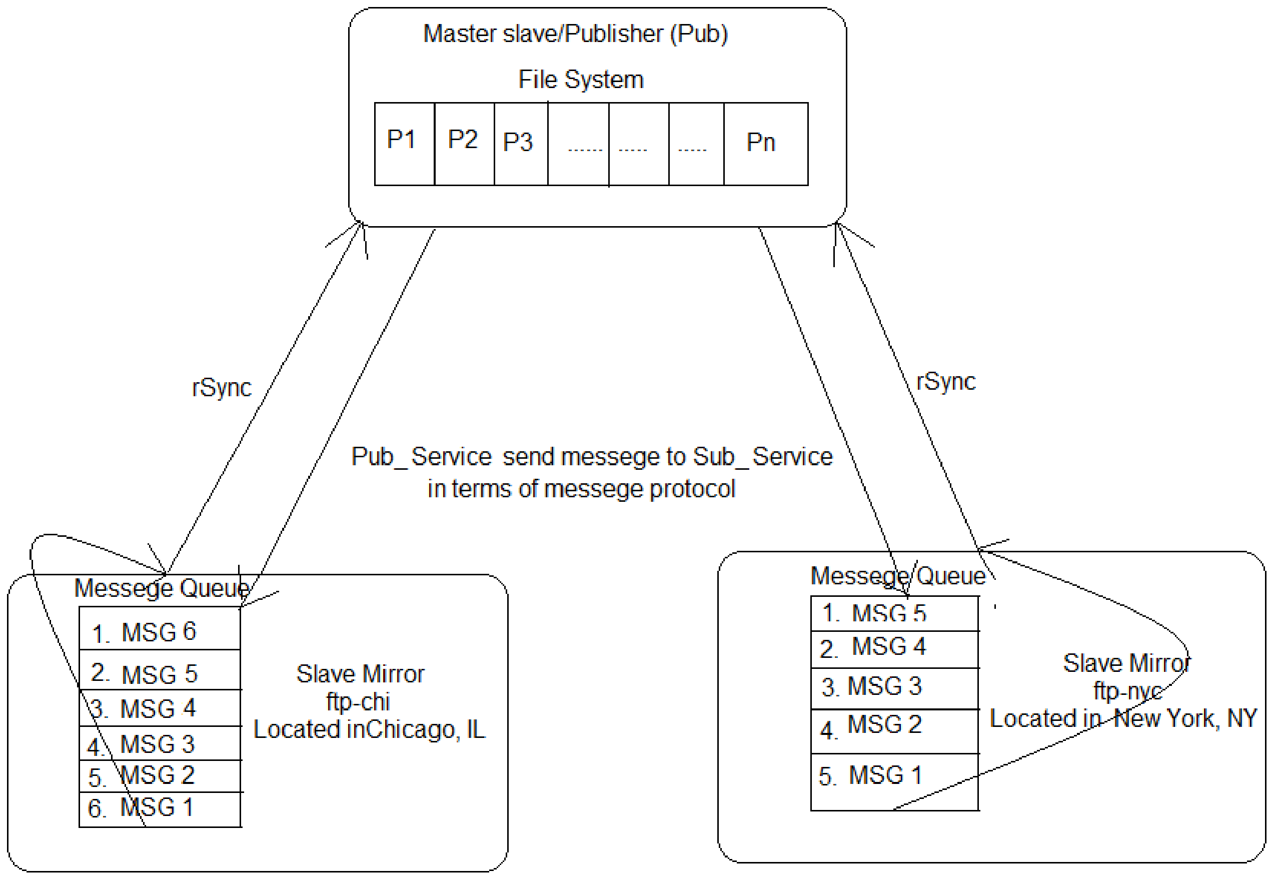
In order to provide a secure chain, subscribers should share a secret with the hub, which will be used by the hub to compute an HMAC key that will be sent to the subscriber. The latter can then easily verify the origin by comparing the supplied signature with a similarly computed signature on their end.

Now **How it will work in OSL FTP mirror Syncing ?**



Explanation of Diagram :

1. Here we can see in below diagram if there is any change will occurs then it will notify to file system.
2. File system service will notify or send message to the Hub or we can say that Master mirror/Publisher.
    a. File system stores data like projects, single file, bulk files etc.
3. Publisher (Pub) will notify/send message to the slave mirror/subscriber (Sub).
4. Now Slave mirror will store that message in queue.
5. rSync will start processing Queue in order to start Sync Process



Note :
1. P1, P2, , , , Pn indicates Project1, Project2, ProjectN.
2. MSG indicates Messege.

Detailed Description with Tentative Timeline:

Week 1 : Design Message Protocol for Publisher, Subscriber, Sync.

Week 2 : Implementing Protocol for Publisher.

Week 3 : Implementing Protocol for Subscriber.

Week 4 : Implementing Protocol for Sync.

Week 5 : Integration of file system notifies.

Week 6 : Integration of file system notifies with Publisher service to generate message for slaves

Week 7 : Patching Publisher Protocol into master slave.

Week 8 : Patching Subscriber Protocol into sync slave.

Week 9 : Implementing Log mechanism.

Week 10 : Dashboard fixing

Week 11 : implementing Dashboard.

1. IP/Region Specific Current Downloads
2. Manage black list
3. Live Bandwidth Bar
4. Live Sync Status
5. Other required information from Log
6. Alerts

Week 12 : Testing and polishing the existing things. Fix minor bugs and problems.

Links to additional information :

About me :

I am Milap, currently pursuing my Bachelors in Information Technology from Sankalchand Patel College of Engineering, Visnagar, India. My interests are researching, music, travelling and of course a lot of coding. I started contributing to VLC media Player after I got inspired by my self. I always think that contributing to open source is a great benefit as we get a lot to learn working in the community and it also helps the people using open source softwares.

Useful Links :

Portfolio : http://milapbhojak.com/

Blog : http://milapbhojak.wordpress.com/ , http://milapbhojak.quora.com/

Github : https://github.com/milapbhojak

Google Code : https://code.google.com/u/milapbhojak.exe/

Google PubSubHubbub : https://code.google.com/p/pubsubhubbub/